Developing a Complete System Architecture for Sensor Networking

University of Colorado
Department of Computer Science

Senior Thesis
by
Jeff Rose

Acknowledgements

Thanks to

Professor Rick Han
Brian Shucker
Hui Dai
Hector Abrach
Anmol Sheth
Jing Deng
Jim Carlson
Simon Wilson

who have made significant
contributions to this project.

1. Introduction

Wireless sensor networking is the result of interdisciplinary research involving aspects of computer science and electrical engineering. By creating large wireless networks of computational devices that can sense many types of phenomena, it is possible to gather information in ways that were previously not feasible. Currently the main applications of sensor networking are for biological and environmental data gathering, security and military sensing systems, location tracking and the development of smart spaces; however, as the field continues to advance these networks are consistently being applied to varying scenarios that call for new features and include unique constraints.

As new applications for sensor networking arise it is clear that researchers are in need of a highly expandable platform for prototyping both hardware and software technologies. Although it is also important to design specialized systems for specific applications, we believe that at this point in the evolution of sensor networking a general purpose platform with maximum expandability and ease of use would be a more valuable commodity.

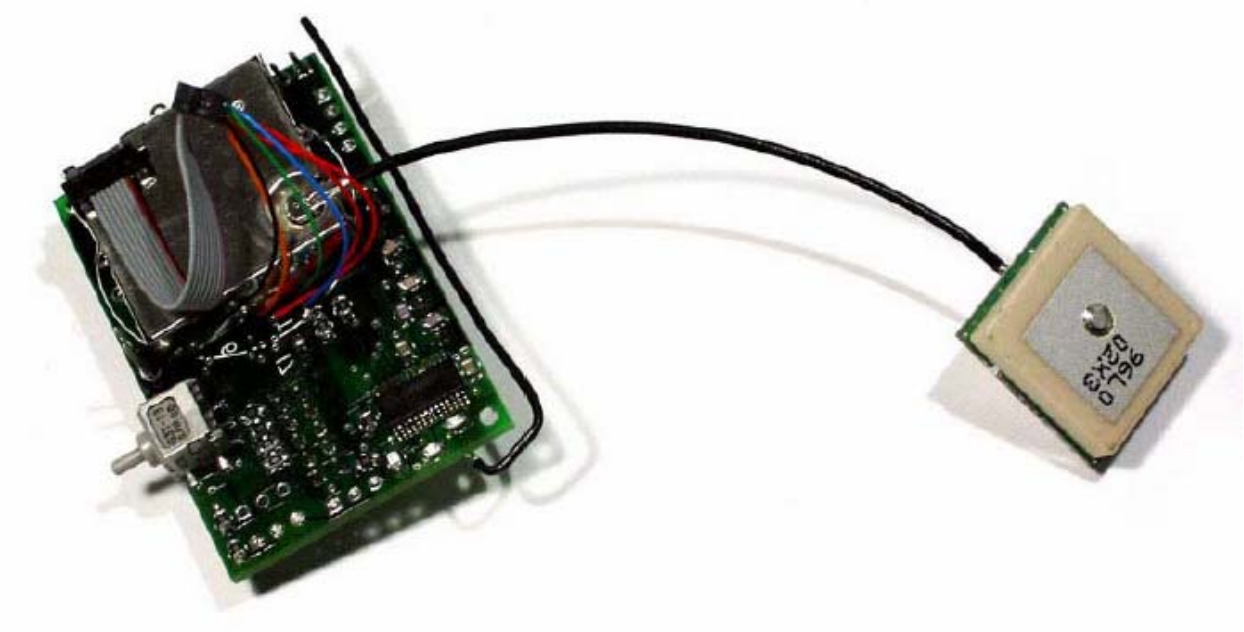

Figure 1. A MANTIS nymph with the GPS module attached.

2. MANTIS System Architecture

Here we present a new system architecture for wireless sensor networks. Including both a new computational device and an operating system designed for sensor networking, the MANTIS project is developing a complete research platform. In order to foster as much future work as possible we have focused on two main ideas in the design of our system. First, it is important that both the hardware and the operating system enable as many different applications as possible. Second, we are working very hard to create a developer friendly environment that will let a variety of users experiment with sensor networking. Our hardware platform can be used to prototype a wide array of sensors and communication mechanisms. These types of system extensions are easily added without the need for custom printed circuit boards, and almost no hardware experience is necessary to use most types of sensors available today. The operating system has

been designed to maximize ease of use for novice computer programmers to write typical sensing applications for the new platform; in addition, we are organizing the OS in a way that allows systems researchers to easily experiment with new operating system features and algorithms.

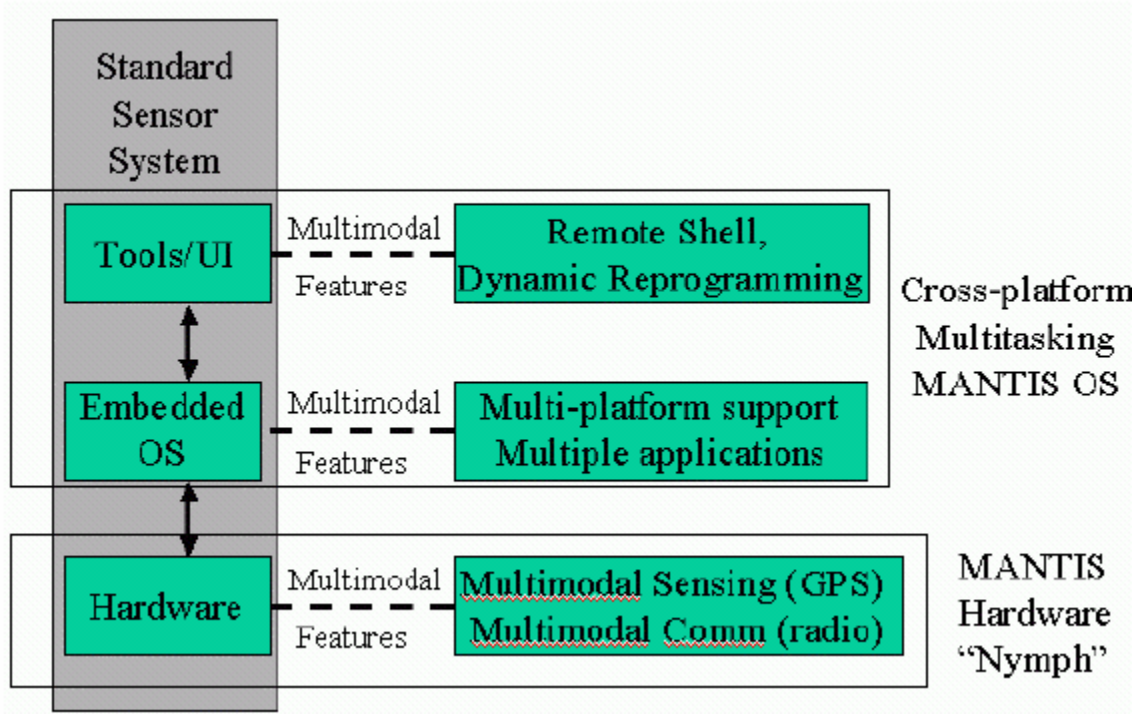

Figure 2. The multimodal, layered design for the MANTIS system.

The mantis hardware platform is a small computational device made specifically for sensor networking. Using the Berkeley Mote[Hill00-b] as a design reference point, we have developed a single board mini-computer that supports our project objectives. The goal of the hardware design was to make it easier for computer scientists and researchers to develop sensor network applications so we have decided to use a single board system, which eliminates the need for expensive, additional hardware for system development. In addition, we have adopted the MIT Cricket[Mart00] sensor interface for our main sensor ports, which makes the connection of many analog and digital sensors a minor task. We have named this new device the MANTIS nymph. (A

nymph is the term used to describe a young preying mantis.)

The nymph's single board design includes everything that is necessary for a wireless sensor node. At the heart of the platform is the Atmel Atmega128[Atme03] microcontroller. Choosing the central processor was not difficult because this microcontroller is so flexible and is already in use in many sensor networks. There are a variety of low power micro-controllers and microprocessors available on the market today, but none of them had as many features as much onboard storage as the Atmega128. This chip has support for a wide array of I/O options: digital input and output, analog to digital conversion, dual USARTs for serial communication, a JTAG interface, I2C and SPI. The processor itself has a built-in watchdog timer for system recovery as well as multiple, digital timer/counter units. For program storage there is 128k of on-board flash memory as well as 4k of RAM for data storage at runtime. For persistent storage we have 4k of on-board eeprom as well as an additional 64k of external eeprom that we have added to the board.

The nymph has been designed to support both an off the shelf DC power supply as well as including a battery connector for true mobility and remote sensing applications. The DC power supply minimizes battery waste in laboratory conditions while also giving the end use complete flexibility in terms of power supply. In addition, the microcontroller and every other chip we have included in the design all have low power modes that allow us to greatly improve the system lifetime for battery powered applications. The DC/DC converter automatically switches between two different supply modes depending on the level of current consumption so that it can be as efficient as possible in either sleep or power-on system cycles. These system low power modes are all controlled by software running on the microcontroller, which gives the system designer many options for fine tuning power consumption in extreme low power scenarios. Using our preliminary power estimates as well as the performance of existing systems[Xbow03], we believe

it will be possible to have a nymph running in an active sensor network for many months using a power supply equivalent to two AA batteries.

Test Performed	Input Voltage (V)	Load Current (mA)
Just Operating System Running	3.0	15
Scheduled Single Thread While(1)	3.0	16
Scheduler with blinking LED ON	3.0	24
Scheduler with blinking LED OFF	3.0	16
Reading Sensor Data on while(1)	3.0	16
Sensing and Sending over Serial at 19.2kBaud	3.0	16
Sensing and Sending over radio, Transmitting at max power	3.0	75
Sensing and Sending over radio, Transmitting at min power	3.0	42
Sensing and Sending over radio, Receiving at max power	3.0	42
Sensing and Sending over radio, Receiving at min power	3.0	29
Absolute Max power with all LEDs on and radio transmit at max power	3.0	95
Absolute Max power with all LEDs on and radio receive at max power	3.0	62
Single LED power consuption	3.0	9
Everything in sleep mode	3.0	under 1
Ellite Nymph - Operating System and GPS	3.0	101
Ellite Nymph - Everything Running at maximum power including LEDs	3.0	174

Figure 3. Power consumption of the Nymph when using different aspects of the hardware.

Communicating with a nymph is achieved using one of 3 possible methods. At the lowest level is the JTAG interface, which has become an industry standard for in system debugging and programming in embedded systems. Using an Atmel provided ICE, or in-circuit emulator, it is possible to single step through code while it is running on a target. This is a new technology in the embedded systems world and we expect that it will continue to make cross-platform development far easier than it has been in the past. Next is the RS-232 serial interface for connecting directly to a PC or serial equipped PDA. This is the primary mechanism used to download code onto the nymph board as well as being the main bridge for connecting a MANTIS network to any other type

of computer system. Once a bootloader has been programmed onto a Nymph using the JTAG interface, only a serial cable is needed to both program and communicate with the device. Using the current 4mhz microcontroller clock speed we can reliably communicate over the serial connection at a speed of 38.4 kbps. The last communications option is the radio transceiver, which gives the nymph wireless networking capabilities. There are many options in the area of wireless communications hardware, but sensor networking presents a unique set of constraints because of the necessity for extremely low power operation as well as support for long range communication. Both 802.11 and Bluetooth network devices, for example, use far more power than a sensor node could provide. We have chosen the Chipcon CC1000[Chip01] radio because of its frequency hopping spread spectrum features as well as its low power consumption and clean programming interface. Using this radio we can communicate between multiple nymphs at 38.4 kbps with a range of over 100 meters. Previously, efficient communication among nodes in a densely populated sensor network was very difficult, but with the ability to change frequencies at high speeds we can now experiment with a wide range of potential spread spectrum solutions. Researching various low power networking algorithms is also possible with the CC1000 because it includes both output power control as well as an analog output for the input signal strength.

Using the 3-wire sensor interface designed for the MIT Cricket[Mart00] makes a wide range of analog and digital sensors easily accessible to the nymph system. The solderless plug connection precludes the use of any special equipment for creating the sensor modules as well as simplifying the prototyping process dramatically. Our current design includes 3 of these sensor ports, and in the future we plan on expanding the design to include a few more. If advanced sensing capabilities are necessary, we have added 2 busses for connecting external, digital circuitry. First, we have exported the second serial USART port so that any serial device can be

connected with 2 wires. Our first such serial expansion is a GPS unit that gives the Nymph instant access to position information when outside. Second, we have included a 4 wire I2C bus, which lets the nymph communicate with a wide array of digital sensors and I/O expander chips for vastly increased I/O capabilities. Beyond the ability to add I/O and sensor functionality, these expansion busses will allow the addition of processor daughter boards for features such as hardware encryption support or digital signal processing units.

Finally, the nymph is all put together on a two layer 3.5 x 5.5 cm printed circuit board (PCB). This size and format of PCB is cost efficient in comparison with a multi-layer board and it simplifies the hardware debugging process dramatically. We have two varieties of battery holders for either round lithium-ion or standard AA batteries, but both of them fit nicely underneath the board without protruding from the sides. For the addition of daughter boards we have mounting points on the board that allow for typical standoff configurations. In the future, we plan on designing a custom enclosure to make each nymph less susceptible to environmental and developer wear and tear.

3. MANTIS Operating System:

In developing the MANTIS system we realized that none of the currently available, open-source operating systems met our project goals and constraints. First, we needed an OS that would be able to run on resource constrained hardware such as the nymph while still being multi-threaded and power aware. This narrowed the possibilities greatly as there are not many operating systems that run with only 4k of memory but still have the features we were looking for. One project potentially met these goals [Avrx03], but it did not fall into our next constraint: we needed the operating system to be written in a standard language such as C to make the project

more accessible to a larger number of software developers. E.g. application programmers as well as kernel developers. Although TinyOS[Hill00-a] is written in a language that is similar to C, nesC, we decided that it was critical to use the standard development tools. Last, it is important that any operating system we use make it very easy to write application level programs as well as operating system drivers and extensions. So, after looking at the existing systems we came to the conclusion that sensor networking presented a unique set of requirements which warranted the development of a new operating system.

The MANTIS operating system (MOS) is a pre-emptive, multi-threaded OS that has been designed for use in sensor networking systems. We have tried to develop the system so it will be similar to a very lightweight implementation of UNIX[Kern84]. The interfaces themselves are not compliant but the programming model is striving to emulate that of the linux operating system. Although the hardware cannot support protected memory, MOS gives sensor network application developers as much flexibility as possible with respect to multi-programming functionality. In order to facilitate rapid prototyping of sensors, we have created a simple driver system that makes it very easy to experiment with a variety of new hardware. For communication, we have included a layered network stack similar in design to a typical TCP/IP stack. This network stack can be used to communicate easily using the radio and/or the serial connection.

The design of the MOS scheduler is based on a typical multi-processing kernel. The current nymph hardware has a shared memory space and it does not support any type of memory protection, so we have classified MOS as a multi-threaded OS. Currently we are using a scheduler that is configured with 4 levels of thread priority where a simple round-robin algorithm is used within each priority. Threads of a lower priority will not run until all higher priority threads have either completed or been suspended. This design lets us use higher priorities for near realtime

performance, but typical sensor networking applications will probably reside at the same priority where they will share resource usage. The scheduling policy algorithm as been confined to a single function however, so further research into optimal scheduling algorithms for use in sensor networks should be easy to conduct. In order to facilitate this type of multi-threaded structure, we have also developed a set of concurrency primitives for protection of shared code and memory. Currently, we have implemented both mutual exclusion locks (mutexes) and counting semaphores, but eventually we would like to include a larger set of concurrency tools. Here again, it is very easy to change the queuing mechanism used by these semaphores to accommodate new operating characteristics or real time functionality.

In writing applications for sensor networks, accessing the sensors and other external hardware is common place. For this reason, we have worked hard to design a very clean driver layer that makes it easy to add new types of sensors and add-on hardware. We decided to use a two layer approach for most drivers. Within the kernel itself, we have implemented low level code that deals with hardware access and power management. This abstracts the hardware details away from the application or driver developer while simultaneously giving us direct control over the power requirements of the various hardware resources. In addition, we have made all of these kernel level drivers thread safe, so the application developer will not have to understand the details of multi-threaded programming. Using a simple temperature sensor as an example, this is the structure of a typical application program written on top of the driver layers. The lowest level analog to digital converter (ADC) driver will be the core of every analog sensor used. On top of that a lightweight driver can be written to do additional power management and calibration for each sensor type. Last, by abstracting the hardware details away from the user we give ourselves a large degree of freedom when designing the hardware for next generation nymph systems.

A critical aspect of a sensor networking operating system is the manner in which user applications communicate with other nodes and potentially to a network base station. We have implemented a serial communication driver that gives applications a byte oriented send/receive interface. Currently we are using small buffers (32 bytes) for the driver interfaces to increase connection reliability, but this is easily re-configured. In addition to communicating with a host PC, the serial interface can be used to connect to future daughter-boards and add-on sensors for increased nymph functionality. Second, we have implemented a network stack for wireless communication using the on-board radio transceiver. The network stack is loosely based on the design of another networking stack for sensor networks,[Ye02] but it is very similar to a lightweight stack for typical 802.11 networks. Currently the goal of our network subsystem is to provide the user with a simple, wireless communication interface that supports both reliable and unreliable protocols while using a minimum amount of system resources.

4. System Development Environment:

Here we describe the development environment we have created for programming the nymph hardware and using the MOS operating system. Setting up the cross-development environment for many embedded systems can be a difficult process that leads to developer frustration and a slow system adoption speed within the research community. For this reason, we have focused much of our attention on the design of a very user friendly environment that can be setup with virtually no knowledge of the nymph hardware or the MOS operating system.

Initially, we have compiled a set of open-source software development tools that are to be used with the MANTIS system. These tools can be downloaded in a single package and installed with a single installation script for very rapid setup. The toolset is based on the GNU binary

utilities and the GCC compiler running in linux along with some custom applications for AVR cross development and communication with our own bootloader. All of the operating system code is written in C with a minimal amount of assembly written for the Atmega microcontroller so porting to a new platform should not be a major undertaking. Once code has been written, it can be downloaded to the target nymph in a variety of ways. At the lowest level, the JTAG interface can be used for very rapid development and absolute control over the code placement. The additional hardware needed for the JTAG interface is quite expensive however, so we have also written a bootstrapping application that has the ability to download code onto the board over the serial port interface. This bootloader can be programmed once using the JTAG interface and then the serial connection can be used for all further programming. This greatly reduces the cost and complexity of loading application code onto the hardware. In the near future it will also be possible to download all code updates over the radio interface which will allow developers to dynamically update whole networks over the air.

Once code has been successfully downloaded onto the target hardware, debugging is typically the most time consuming stage of the development process. Currently, using the JTAG interface is the only way to do true line by line debugging, which uses the GDB debugger and a JTAG in-circuit emulator. In order to facilitate system debugging for those that do not have access to JTAG hardware or cannot reproduce the errors, we have developed a remote shell application.

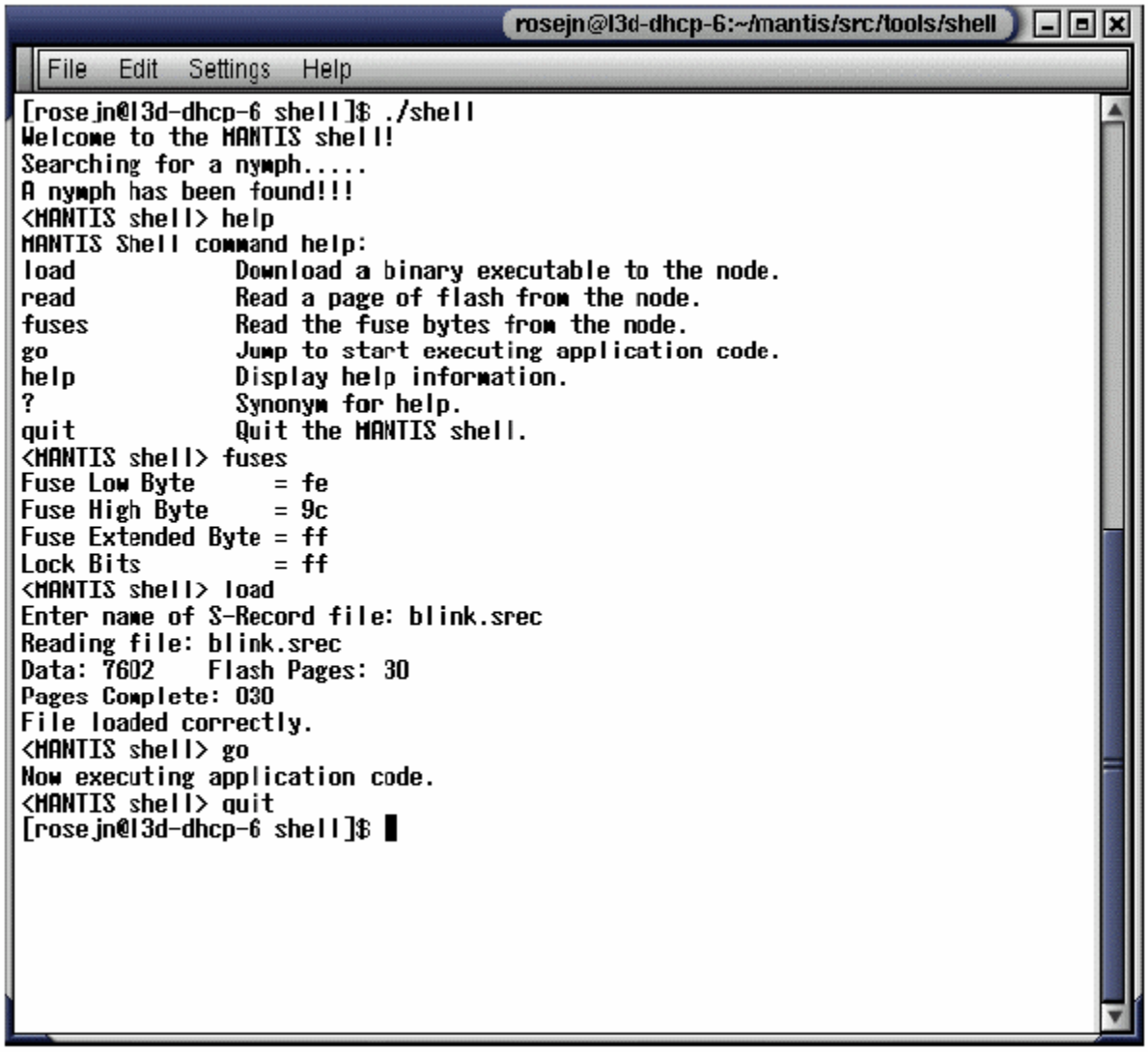

Figure 4. This is a screenshot of an early version of the remote shell.

Designed for use in both interactive and command modes, the MANTIS shell is a multi-functional tool that can be used for both program development and system management. This shell can be used during development to download code onto a nymph node, and then to connect to the node and check on the state of various operating system structures. The lack of memory protection makes it very easy for us to look at kernel data structures without having to instrument the kernel itself. In addition to looking at the kernel it will be possible for users to write extensions to use the

shell command server to help with their own debugging problems. Beyond adding support for logging facilities, it will also be possible to create a serial interface to GDB for direct line by line debugging over the serial port. This involves a considerable amount of system overhead however, so it would be necessary to instrument the kernel itself in order to facilitate GDB debugging.

As well as helping with the software development cycle, the MANTIS shell is also designed to be a system management tool. In the future we plan on adding support for a simple system re-programming mechanism that will give developers the ability to select nodes in a sensor network and then send code updates to be automatically loaded by the on-board bootloader. Previously it was not possible to reprogram the flash memory in a microcontroller without the use of either an external programmer or some additional programming circuitry, but with the new generation of Atmega microcontrollers it is now feasible. Using a new flash memory technology it is possible to allocate a section of the onboard flash for code that is allowed to right to the rest of the code space. The current bootloader has the ability to read code over the serial port to then be programmed, but we are in the process of extending this functionality so that it can also read from the eeprom, our persistent storage mechanism on the nymph platform.

5. Network Simulation and Cross Network Communication:

Another feature of the MOS operating system, because of its C language compliance and the use of the GNU development tools, is that we can actually run an instance of MOS as a single process running on a Linux based PC. A very minimal amount of the low level kernel code had to be ported to use the existing Linux kernel facilities, but all the application level code can run on both platforms with only a recompilation. This seemingly simple idea has vast implications for system development however, because it gives us the ability to debug and simulate both operating

system features and user level applications on a Linux host machine. With the addition of this new version of the Mantis Operating System (MOS) we now label them aMOS for the AVR based nymph platform, and xMOS for the x86 based platform.

Debugging high level sensor network applications is much easier when using the xMOS system because it allows users to print information to the screen from multiple simulated nodes as well as giving easy access to the GDB debugger. Using GDB for xMOS is still difficult because of the multi-threaded nature of each simulated process, but the ability to get any kind of run-time view of an application is a great advantage to an embedded system application developer. Using this type of simulation environment also makes it very easy to test boundary conditions and simulated sensor data rather than having to actually simulate real environments for pre-deployment testing. This level of simulation will be most valuable to application developers who are learning how to write networked sensing applications because it will allow them to generate sensor data as they please and then track the data as it traverses the network to finally arrive at a base station or data aggregation center.

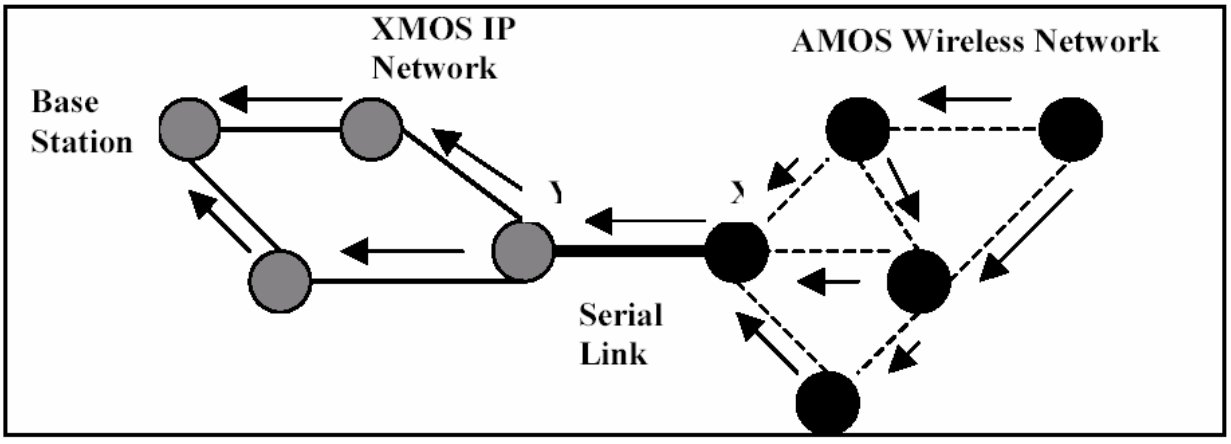

Figure 6. A MANTIS sensor network spanning both Nymph nodes as well as x86 based computers.

Beyond the debugging of a single sensor networking application, it is often necessary to simulate the behavior of a large network of sensor nodes. When researching various types of data aggregation or routing techniques for example, it can be difficult to get realistic results without actually running the applications and observing the system as it functions. By allowing simulated nodes to communicate using our stack implemented over TCP/IP sockets, it is now possible to do much of the preliminary testing without having to download code onto a large network of sensor nodes. In addition, it is much easier to gather simulation data on a pc rather than having to connect to deployed sensor nodes for data collection. This idea can be extended to simulate many types of communication interference problems by injecting artificial network error into the communication system. Our current network simulution topology has been designed to support many types of simulation features in the future. We would like to have the ability to spawn xMOS nodes and then add them into a network all from within a single tool. By routing all the packets through a central hub we will be able to simulate and experiment with transmit power control algorithms and network interference problems.

As well as simulating sensor nodes, it is also possible to run MOS processes as true nodes in a larger sensor network. Rather than using simulated data, a linux PC could be connected to many types of more advanced sensors, which gives the MANTIS system a very broad range of sensor support. By connecting one of these xMOS nodes to an aMOS node over a serial connection, it is already possible to interact with a whole network of deployed nodes. It could even be used to add a group of simulated nodes to a deployed network of nymph nodes to see how the system handles various types of events or data overload. Last, one of these bridging xMOS nodes could be used as an access point to connect a whole network of deployed aMOS nodes to the

internet. This makes it very easy to monitor and manage a MANTIS network from any computer that has internet TCP/IP connectivity.

6. Current enabling research:

The present MANTIS system is already being used in a variety of research applications. With the addition of a GPS module, we have added outdoor location awareness and accurate time measurement at the same time. Global position information can be used for both tracking of mobile sensor nodes as well as determining accurate locations of nodes in a network when constructing network routes and data paths. Keeping synchronized time systems among many nodes in a sensor network is vital for accurate sampling of time sensitive events. Additionally, if nodes are set to wake on a common duty cycle the energy consumption for transmitting data through the network can be reduced dramatically because transmitting nodes know exactly when their downstream neighbors will be available. A very accurate time synchronization system has already been developed by a member of the MANTIS team, and in tandem with the new GPS technology we now have an automatic time update mechanism for any outdoor network of nymph nodes.

Within the school of engineering we are also starting to collaborate with the Center for Life Long Learning and Design by using sensor networks to help people with cognitive disabilities navigate both inside buildings and outdoors. Using the nymph as a GPS receiver connected to some type of PDA with access to a cell network, it will be possible for caregivers to track those who are within their care. Once indoors, the nymph will be communicating with an installed sensor network so that it can determine location within the building and help the user navigate to their desired destination. This type of application is new to sensor networking and will surely

introduce many new areas for research as well as uncover flaws in our current design.

The most unique application of our MANTIS system is in the area of user interface design and prototyping. Using the recently opened BP visualization center as a platform for interface design in truly three dimensional environments, we are working on the development of new types of user interfaces. The such interface is that of a virtual orchestra conductor, where a nymph with acceleration sensors (accelerometers) is going to be configured in a way that it resembles the conductors baton. By sensing the direction, speed and rhythm at which a person moves this baton it will be possible to control a midi sequence so that the notes correspond to the motions of the conductor. All of this information will be transmitted over a MANTIS network so the conductor will have complete range of motion. This application of sensor networks into the area of interface design will allow for many new types of devices that were previously not possible because of the simple wireless capability, low cost, and the wide array of possible sensors.

7. Future work:

The MANTIS system has just reached its first stable milestone of version 0.1. The nymph hardware has been tested in a variety of ways and we are determining what features are necessary in the second generation of our design. In terms hardware research, we would like to experiment with various types of energy gathering mechanisms such as solar cells in order to extend the lifetime of a typical sensor node. This entails a non-trivial amount of work with respect to both the power circuitry and the software control mechanisms that would be necessary to control the charge and discharge cycles. In the near future we would also like to incorporate a protective shell into the core mantis design so that the hardware will be protected for maximum operating lifetime. Most likely we will use some type of injection molded part that has the ability to protect a simple

nymph while still allowing for hardware expandability.

The MANTIS Operating System is ready for use in many types of sensor network applications. We would like to continue the operating system development by both extending the current feature set and researching better ways to implement current features. The scheduler, for example, could be fine tuned for many different types of application sets. If the necessity arises we might work on supporting real-time systems, but in the near future we are going to focus on low power operation and operating system support for automatic power savings. New sensors will always need drivers, and it is becoming evident that the digital signal processing of sensor data as well as the calibration of the sensors themselves deserves a considerable amount of attention.

Networking will continue to be a major focus in the development of the MOS system and we are already looking into more advanced power management and frequency hopping algorithms. By leveraging the new output power control mechanism along with the received signal strength indicator, it will be possible to tune network transmissions for both high throughput and power efficiency. The ability to change frequency while transmitting data will allow us to do research into a large area of wireless networking that was previously not possible. Time division multiple access (TDMA) protocols that are local to specific regions of a network, coupled with data aggregation and signal processing, have the potential to increase the longevity of sensor networks greatly.

Our development environment is an integral part of the MANTIS project and a large amount of research will go into further extending its features. The shell utility we have already developed is undergoing many changes to facilitate additional network management features and remote programming functionality. We are developing a set of tools that will be able to function as the back end for a large system management interface, while also exporting functionality to allow

for scripting of sensor network control. Once we have a large sensor network deployed we would like to research various methods of visualizing and interpreting the data gathered by the network. Last, we have also discussed the idea of using some type of graphical programming system to develop sensor network applications. Because of the nature of typical sensing applications, we believe they would lend themselves nicely to an object based programming system that allowed non-programmers to create sensor systems using a tool like agentsheets[Rep00].

With our current generation of nymph hardware, memory is the limiting factor in many of the operating system and application features we have developed. Recent innovations in memory technologies, such as magnetic ram could allow us to incorporate many more advanced features in the MANTIS system. Many types of encryption schemes, for example, require large tables of binary data for use in the encoding algorithm. Data compression is also much easier to accomplish if we have the ability to store large amounts of data in system memory. Beyond the data memory, large amounts of persistent memory would allow sensors to keep large amounts of both program and sensor data. We could keep a library of drivers and application components available on every nymph so that they could be very efficiently re-programmed while in the field. Given a set of keys it would be possible for a node to completely reconfigure both the operating system and the sensing applications to be run. The type of functionality would match perfectly with the object programming system described above. With the recent increase in flash memory storage capacity, many embedded devices are starting to run file systems over this type of persistent storage so that they look just like a typical disk. We are currently looking into the adaptation of this technology to EEPROM memory so that sensor network applications can store information into files for later processing or transmission.

8. Conclusion:

We have presented a new system architecture for sensor networking. Project MANTIS is a next generation embedded system with a new, lightweight operating system designed specifically for sensor networking applications. Focusing on ease of use and development as well as system features, we are creating a framework for research in virtually every aspect of wireless sensor networking. As well as the standard sensor networking applications we are also encouraging the use of our system as a tool for research in other areas such as wireless UI development and the development of tools for cognitively disabled peoples.

In accordance with the free software tradition, we will be releasing all of the source code and system hardware design information to the sensor networking community. In tandem with our own work we would like to facilitate an open research community that can use the MANTIS system in their own work. We believe that this level of information sharing will lead to more collaboration with other departments and organizations that could lead to valuable relationships in the future. Beyond the benefits of collaboration, wider use of the MANTIS system will uncover software bugs and flaws in the hardware design much faster than if it was confined to our research group alone.

9. References:

[**Arvi94**] K. Arvind, "Probabilistic Clock Synchronization in Distributed Systems," IEEE Trans. parallel and Distributed Systems, vol. 5, no. 5, pp. 475-487, May 1994.

[**Atme03**] Atmel AVR 8-bit RISC processor, http://www.atmel.com/products/AVR

[**Avrx03**] Found September 2002, at: http://www.barello.net/avrx/

[**Cart95**] J. Carter, J. Bennett, W. Zwaenepoel "Techniques for Reducing Consistency-Related Communication in Distributed Shared-Memory Systems" ACM Trans. on Computer Systems 13(3), pp. 205-243, August 1995

[**Case90**] J. D. Case, M. Fedor, M. L. Schostall, and C. Davin. RFC 1157: Simple network management protocol (SNMP). RFC, IETF, May 1990

[**Chip01**]Single chip ultra low power RF transceiver http://www.chipcon.com/files/CC1000_Data_Sheet_2_1.pdf, 2001

[**Cris89**] Flaviu Cristian. Probabilistic clock synchronization. Distributed Computing, 3:146–158, 1989.

[**Deb01**] B. Deb, S. Bhatnagar, B. Nath "A Topology Discovery Algorithm for Sensor Networks with Applications to Network Management", DCS Technical Report DCS-TR-441, Rutgers University May 2001

[**Elso01**] J. Elson, D. Estrin "Time Synchronization for Wireless Sensor Networks" Proceedings of the 2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless and Mobile Computing, San Francisco, California, USA. April 2001

[**Elso02a**] J. Elson, L. Girod, D. Estrin "Fine-Grained Network Time Synchronization using Reference Broadcasts", In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA. December 2002.

[**Elso02b**] J. Elson, K. Römer, "Wireless Sensor Networks: A New Regime for Time Synchronization", in proceedings of the First Workshop on Hot Topics In Networks (HotNets-I), Princeton, New Jersey. October 28-29 2002

[**Enge99**] P. Enge, P. Misra, "Special Issue on Global Positioning System," Proceedings of the IEEE, Volume 87, No. 1, January, 1999, pp. 3-15.

[Eyes02] Eyes: Energy Efficient Sensor Networks, http://eyes.eu.org, March 2002

[**Gane02**] S. Ganeriwal, R. Kumar, S. Adlakha, M. Srivastava, "Network-wide Time Synchronization in Sensor Networks," Technical report, University of California, Los Angeles, Dept of Electrical Engineering, 2002.

[**Hill00-a**] J. Hill "A Software Architecture Supporting Networked Sensors", Masters Thesis, 2000

[**Hill00-b**] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, "System Architecture Directions For Network Sensors", ACM Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) 2000, pp. 93-104.

[**Hill03**] Jason Hill's Smart Dust Spec, www.cs.berkeley.edu/~jhill/spec/index.htm

[**Inta00**] C. Intanagonwiwat, R. Govindan, D. Estrin. "Directed diffusion: A scalable and robust communication paradigm for sensor networks." In Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom) 2000, pp. 56–67.

[**Kahn99**] J. M. Kahn , R. H. Katz , K. S. J. Pister, "Next century challenges: mobile networking for "Smart Dust"", MobiCom 1999, pp. 271-278.

[**Kern84**] B. Kernighan, R. Pike, "The Unix Programming Environment" Prentice Hall, 1984

[**Levi02a**] P. Levis, N. Lee "Nido System Description", http://webs.cs.berkeley.edu/tos/tinyos-1.x/doc/nido.pdf, October 2002.

[**Levi02b**] P. Levis, D. Culler "Mate: a Virtual Machine for Tiny Networked Sensors" ACM Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Oct. 2002.

[**Liao99**] C. Liao, M. Maronosi, D. Clark, "Experience With an Adaptive Globally-Synchronizing Clock Algorithm," In Eleventh Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), 1999, pp. 106-114.

[**Main02**] A. Mainwaring, J. Polastre, R. Szewczyk D. Culler, J. Anderson,"Wireless Sensor Networks for Habitat Monitoring", First ACM Workshop on Wireless Sensor Networks and Applications (WSNA) 2002, pp. 88-97.

[**Mart00**] F. Martin, B. Mikhak, and B. Silverman, "MetaCricket: A designer's kit for making computational devices," IBM Systems Journal, vol. 39, nos. 3 & 4, 2000.

[**Mill91**] D. Mills, Z. Yang, T. Marsland (Eds.) "Internet Time Synchronization: the Network Time Protocol" Global States and Time in Distributed Systems, IEEE Computer Society Press 1991

[**Mills94**] D. .Mills, Internet Time Synchronization: The Network Time Protocol. In Zhonghua Yang and T. Anthony Marsland, editors, Global States and Time in Distributed Systems. IEEE Computer Society Press, 1994.

[**Park00**] S. Park, A. Savvides, M. B. Srivastava, "SensorSim: A Simulation Framework for Sensor Networks", In the Proceedings of MSWiM 2000, Boston, MA, August 11, 2000.

[**Priy00**] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan, "The Cricket Location-Support System." Proc. 6th ACM MOBICOM, August 2000, pp. 32-43.

[**Rash89**] R. Rashid, R. Baron, et al. "Mach: A Foundation for Open Systems". In Proceedings of the 2nd IEEE Workshop on Workstation Operating Systems. 1989.

[**Rep00**] A. Repenning., "AgentSheets®: an Interactive Simulation Environment with End-User Programmable Agents," Interaction 2000, Tokyo, Japan, 2000

[**Rome01**] K. Romer, "Time Synchronization in Ad Hoc Networks", MobiHoc 2001.

[**Tila02**] S. Tilak, N.B. Abu-Ghazaleh, W. Heinzelman, "A taxonomy of wireless micro-sensor network models", ACM SIGMOBILE Mobile Computing and Communications Review, Vol. 6 Ch. 2 pages 28-36. 2002.

[**Xbow03**] Crossbow Mica2 and Mica2dot Motes and Sensors, http://www.xbow.com/Products/Wireless_Sensor_Networks.htm, 2003.

[**Wan02**] C. Wan, A. Campbell, L. Krishnamurthy, "PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks", First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA) 2002.

[**Want92**] R. Want, A. Hopper, V. Falcao, J. Gibbons, "The Active Badge Location System," ACM Transactions on Information Systems, Vol. 10, No. 1, January 1992, pp. 91-102.

[**Ye02**] W. Ye, J. Heidemann, D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks", Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), June, 2002

[**Zhao03**] J. Zhao, R. Govindan, D. Estrin "Computing Aggregates for Monitoring Wireless Sensor Networks", In proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications Anchorage, AK. May