Generic Sensor Platform for Networked Sensor Nodes

Haywood Ho
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
haywood.ho@intel-research.net

Faculty Advisor: Professor David Culler

Abstract

TinyOS is an event-based operating system designed for use with embedded networked sensors. However, writing TinyOS code can be intimidating at first, and will be an especially steep learning curve for those without experience in C. Thus, the Generic Sensor platform was developed to to abstract away from the lower-level functionality of the sensors, enabling application programmers to be concerned only higher-level details. In this current version of the generic sensor platform, multi-hop routing, the most important improvement since the last version, has been implemented.

1. Introduction/Motivation

TinyOS is an event-based operating system developed for a wireless network of deeply embedded sensors. It provides a component based model abstracting hardware specifies from application programmer. It has been optimized for the needs of sensor networks: high concurrency required whilst being constrained by the limited physical parallelism and limited computational ability and energy supply.

TinyOS code is based upon C code (much of the code is hidden by C preprocessor macros). Thus, anybody who knows C should be able to pick it up easily. It is compiled with the special Atmel compiler and then the code is downloaded on the EEPROM memory of the mote.

The structure of TinyOS code is rather obscure and complicated (for example, the multiple parts to an application component), and presents a steep learning curve for those unfamiliar to the language. (The nesC release, due on September 6, would improve hopefully alleviate this problem.) However, there are so many applications for the motes that many researchers outside our group are interested in utilizing these motes. We believe that this steep learning curve will be hinder the deployment of sensor networks and thus, there is a need for a simpler interface to the motes.

Definition of Problem

The problem is as follows. We wish to:

1) Have a cleaner, simpler interface to control the motes;
2) Allow wireless communication with the sensor network using a computer; and

3) Allow the motes to be used for a wide variety of applications.

Importance

This platform could be used for a variety of applications. Currently, I am working with the civil engineering students to improve the current platform to support all the functionalities that they require for their projects. One of the projects is the real-time monitoring of structures. These sensors would be spread out in critical sections of buildings and other structures to monitor its structural safety. Other projects include "shake-tests", where mock structures are equipped with wireless sensors for data collections in different areas of the structure. Yet another project is detection of wildfire patterns in forests. All three projects include one core component, which is the sensing component. However, much code would have to be rewritten if sensing programs were written specifically for each purpose. With the Generic Sensor Platform, we reuse the same code over again, and change the sensing behavior of the motes by changing a few parameters.

Background

The solution we propose is to have a MATLAB frontend to control a mote connected to the computer via the UART. This mote would be have the Generic Base High Speed program installed, which would simply forward packets it received over the UART from the computer wirelessly to the motes, and vice versa. The other motes in the network would all be pre-programmed with Generic Sensor, which would allow the motes to change their behavior in response to incoming packets. Thus, we can avoid dynamic reprogramming and still allow the motes to be used for many different applications by sending a variety of different packets.

There is already some software already in place. One of the graduate students in the TinyOS group, Kamin Whitehouse developed most of the present generic sensor code, which he uses for his sensor localization experiments. However, the problem with the code is that it does not completely fulfill the requirements of many sensing applications.

However, there are many difficulties to overcome. Radio communication between motes currently is not particularly reliable (although this may improve in the upcoming nesC release), and there are many yet many unsolved ad-hoc routing problems present. Time synchronization is also an important issue, as many applications require sensor readings from different motes may be useless otherwise.

2. Design/Architecture

In the previous version of Generic Sensor, there was a port mapping which mapped each mote to each serial port of the computer. This was because Kamin's localization experiments required a large amount of motes, which were all connected individually to a different port of a port multiplexor. Thus, to send route a command to a mote, one would send a message to one serial port (obtained from a mapping pre-defined in the Matlab environment), which would then forward the packet to that specific mote.

However, in the new version, command messages are broadcasted wirelessly to the Generic Sensor nodes. The base station, programmed with the Generic Base High Speed program, would forward the packet it received from the computer over the UART over the radio to the sensor nodes. Thus, there would be no need for a port mapping. However, this has not been eliminated in the new architecture yet, as changes to it are still in progress.

Currently, the routePacket() function in Matlab merely broadcasts out command packets to the nodes in the network. (It calls the bcastPacket() function currently.) To improve this, however, we would modify the Generic Sensor code to allow the target mote ID to be specified in the command header of the packet. Thus, we would able to send messages to a specific node in the network.

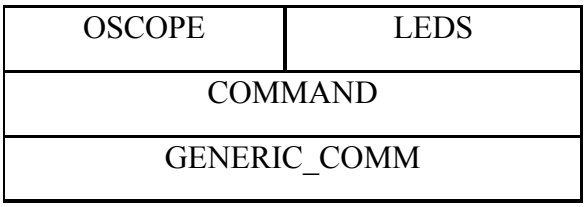

Figure 1: Architecture of previous version of Generic_Sensor

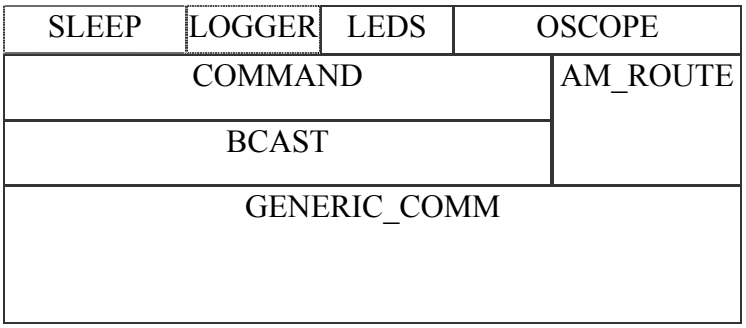

Figure 2: New architecture of current version of Generic_Sensor

Thus, when a command is broadcasted out, it goes through the generic communication stack, and is rebroadcasted upon receipt at the mote (depending on whether the mote has seen that packet before; if it has it will not rebroadcast it again). Then the message is propagated up to the command layer where it is processed and

invokes the correct application program accordingly. When the motes wish to route a packet back to the base station (as is the case with OSCOPE packets), the motes route them through the AM_ROUTE and then down into the generic communication stack.

Message Structure

There are 3 Types of Active Messaging (AM) layer packets. AM 8 is used for command packets, which are sent out from the base station to command the motes in the network to perform a specific task. AM 5 is used for route discovery. It is broadcasted from the base station every time before the motes receive a startSensing packet, so that the motes would be able to know where to route the data packets back to. AM 6 is used for data packets, which are routed back to the base station, where the data can be collected and visualized on the computer.

As can be seen from the figures, the Logger and Sleep components have yet to be added to the Generic Sensor platform (however this should not prove too difficult, as the components have already be written, so only minor changes to the code would be required).

Message structure

Structure of a command packet (for an OSCOPE command packet)

address	AM	groupID	seq#	hopcount	maxHopcount	ledMask	commandCode

GENERIC_COMM headers	BCAST headers	COMMAND headers

ADCaction	dataChannel	dataDest	maxSamples	bytesPerSample	resetCounter?

OSCOPE headers

payload	CRC

GENERIC_COMM footer

3. Conclusion

The Generic Sensor platform would be important in that it would allow sensor networks to be used in various different applications. It would speed up the deployment of sensor networks for different research purposes. The Generic Sensor platform would go a long way in allowing the users of the network to concentrate on the higher level aspects of their code, hiding and abstracting away from the lower level TinyOS code that may include too many details that is not of relevant concern to the users.

However, much remains to be implemented on the new and developing Generic Sensor Platform. There are many directions this project can be taken, and I will mention a few of these in the next section.

Further Work

There remains much work to be done on the Generic Sensor platform. The Log and Sleep components remain to be integrated into Generic Sensor code, and simultaneous sensing from two data channels also needs to be implemented. There are very interesting time synchronization issues that have not even been examined in this paper. We should also explore possible alternative ad-hoc routing algorithms and implement them to see their effectiveness (as opposed to simple broadcast, which, in theory, over rebroadcasts and nearly always does not come up with the best route). We should also attempt to make the Generic Sensor platform more robust. Currently there is no CRC checking on the Matlab side, and packets are often dropped if another message is currently being processed by the mote. Thus, we should have a mechanism for retransmission and ultimately, a scheme for reliable wireless data connections between motes.