RAMnet: a Meshnet File Transfer

Matthew Miller
Binghamton University
+1 (908) 922-1402
mmille10@binghamton.edu

Jordan Raitses
Binghamton University
+1 (631) 626-4915
jraitse1@binghamton.edu

Laith Alsunni
Binghamton University
+1 (313) 241-3655
lalsunn1@binghamton.edu

ABSTRACT
In this paper, we describe RAMnet, a TelosB-based wireless file transfer network with mesh bridging.

Keywords
Mesh Network; TelosB; TinyOS; Wireless Network; File Transfer.

1. INTRODUCTION
We established our team with help from professor Sha, but we were faced with the question of what to do for our term project. The assignment was left open-ended in terms of what sort of project to do, but since the class is entitled "Wireless Sensor Networks", we felt that we ought to incorporate the main theme of the class in our project. We decided to utilize the wireless sensor motes and programming models about which professor Sha spoke in class. Given the cost of acquiring motes on our own, we used the TelosB motes the professor had made available. After some brainstorming, we decided to create a program for reliable file transfers over a network of motes. The purpose of this was twofold. (1) A real-world application for our project could be a computer in a remote location sending information without needing a pre-existing network infrastructure besides motes on each machine and motes bridging the gap to the other networked computer. (2) The secondary purpose was for this project to serve as a learning experience for the team members. None of our group had ever worked with TelosB motes, TinyOS, nesC, or any of the other technologies before. By doing this project, we got to experiment with multiple programming languages and ways of interfacing with the motes.

2. Project Specifics
2.1 Division of Labor
Specifically, we split up our project into 3 segments, the software on the PC that used a serial interface to communicate with a "base station" mote, the "intermote" responsible for relaying packets to other nodes, and the web interface to make it easy for an end-user to use the software without resorting to the command line. We also chose to design our own protocols for the actual packet handling due to limitations in the TinyOS framework and to further enhance our knowledge of working with these technologies.

2.2 How it works
The mote that connected directly to the computer used the standard Base Station program that came with TinyOS. However, we could not use the included SerialForwarder class since the software on the computer needed duplex communication with the mote. Lacking online documentation, our team examined the code for SerialForwarder to learn how serial communication worked within TinyOS. We then wrote our own software in java that communicated with the Base Station mote in half-duplex. This same software either broke a local file into packet-sized pieces or received a remote file, packet by packet. To ensure reliable communication and to avoid overloading the motes, our file-sending software always waited for appropriate acknowledgements before sending new packets. Unfortunately, we could not also implement error checking since the packets were only 28 bytes large due to TinyOS limitations.

2.3 User Interface
For the user interface, we started with a web interface that would simply allow users to drag and drop files. The user would only have to find a file on their machine and drag it to the designated part of the web interface and drop it. The drop action would trigger the system to send the file. However, due to file permission issues in linux we could not implement the drag and drop. Instead, we went with a simpler HTML file selector that passes the selected file object to a PHP page that runs the appropriate commands in a script. We followed the same strategy with starting and stopping the listener using a button on the web page.

3. Typical Run
In a typical run, the sending computer would point the PC application at a local file. The local file's name would be sent to the remote machine. When the remote machine is ready to receive the file, it sends an acknowledgement. At this point, the sending machine begins reading its local file, 28 bytes at a time. These are embedded in a TinyOS packet which is sent to the remote machine via the motes. The sending machine does not send the next packet until it receives an acknowledgement for the most recently sent one. Once the local machine reaches the end of the file, it indicates this to the remote machine with a specific End Of File packet. At this point, the file transfer is complete. The same procedure is followed when an active intermote is used, but the packet will be picked up and forwarded by the intermote since the two Base Station motes are out of direct contact range. The intermote does not forward the same packet twice, so we can be sure that no loops are formed when more than one intermote is used.

4. Conclusion
To conclude, we believe we accomplished the broad goals with which we started the project; we learned how to work with Wireless Sensor Networks and we implemented a functional and robust file transfer program. The project turned out to be much more difficult than expected and some of our hopes for what we would achieve in the very early planning (like complex routing for the intermotes) did not come to fruition. This was largely due to the combination of very limited (and often incorrect) documentation available for TinyOS, the awkward means of interfacing with the motes, and severe issues with the reliability of the serial port itself, particularly when using a VirtualBox implementation rather than a native linux installation. Overall however, the project was a success, as we created an effective solution to the problem we set out to solve.

5. ACKNOWLEDGMENTS
Our thanks to Professor Mo Sha for advising us during our work and for the use of his TelosB Motes.