

Part A: Project Description

The goal of the Smart Dust project is to develop devices with form factor of several millimeters containing sensors and capable of communicating via radio frequency. Thanks to the ad hoc nature, this network does not require an infrastructure and can be deployed quickly. Wireless ad hoc embedded sensor network has many applications ranging from scientific experiment to emergency situation. A group at the University of California-Berkeley has built hardware prototypes of these motes, and many of these have been put in commercial use. Along with hardware parts, they have also developed TinyOS, an operating system that runs on motes. TinyOS features a very small size and component architecture resembling the organization of hardware. TinyOS is composed of many basic components representing hardware, and programmers write applications by specifying the wiring of these components using configuration files.

With the number of motes in a wireless ad hoc network reaching hundreds of thousands, simulation plays a significant role in developing and debugging such network. TinyOS is currently shipped with TOSSIM, a TinyOS simulator. TOSSIM can compile and simulate TinyOS applications from source, and it can run natively on a PC workstation. Although TOSSIM is helpful for preliminary evaluation, it presents many short-comings that make it inadequate for gaining understanding of an actual network. First, in a TOSSIM simulation, all motes run the same application. In an actual network, it is more likely that the motes are specialized, and different applications are run on different motes. Second, TOSSIM only provides a simple model for positions of motes in the network. Finally, TOSSIM can only support up to several thousands motes. An effective network usually contains many orders of magnitude number of motes, and studying only a small subset of this will not tell everything about the dynamics of the real system.

A project to develop TOSSF, a TinyOS Scalable Simulation Framework, was started by Professor Perrone when he was at Dartmouth College. TOSSF is based upon SWAN, Simulator for Wireless Ad Hoc Networks, an open source simulator developed at Dartmouth College. SWAN has many functions that TOSSIM lacks. SWAN offers the flexibility of model configuration at run time. That is, the models of terrain, radio frequency propagation, environmental process and mobility of nodes can be changed for each simulation simply by modifying the configuration file. Therefore, with some knowledge of the Domain Modeling Language (DML), the language used to compose configuration files, a programmer can quickly design new models for his simulation. Besides, SWAN can support up to hundreds of thousands nodes; therefore, TOSSF will also have similar scalability.

An obstacle exists in building TOSSF on top of SWAN and using it to simulate TinyOS applications. TinyOS is implemented in nesC, a modified C language designed to support the component structure of TinyOS. On the other hand, SWAN is a collection of C++ classes. If we want TinyOS applications to execute directly from source without

modification of programmers, we will need to develop a translator from nesC to C++. The details of this will be presented in the **Methods** section.

Methods:

The development of a source-to-source translator will require us to carefully study nesC using its available documentation online. We will also need to write nesC programs and simulate them on TOSSIM to test the behavior of the language. The most important part of this process is to discover the equivalency mappings of nesC language structures to C++ structures. Once these mapping are found, the coding of the translator will become trivial. This translator will allow us to easily re-implement necessary TinyOS components in C++, and these components will compile and run on SWAN with some integration.

Professor Wittie, with her expertise in programming language, will give me guidance on studying the behaviors of the language, and on techniques that we can use to build a source-to-source translator. Besides, Professor Perrone, with his familiarity of TinyOS and SWAN, will give me advices regarding those areas.

Outcomes:

Our main goal of the project is to obtain a good understanding of the equivalency mappings from nesC to C++, which allow the creation of a source-to-source translator. Then we will develop a scalable simulator TOSSF, based on SWAN, for TinyOS applications.

This project is based upon the work that Professor Perrone had done when he was at Dartmouth College. At the time, a functional simulator was already in place as a result, but the language of TinyOS was not nesC, and the lack of a source-to-source translator makes the conversion of code problematic. Now, when the language of TinyOS has been moved to nesC, it is time to gain an in depth understanding of the new language and build a good translator. All previous work was done by Professor Perrone in the time span of one month; therefore, we expect that we will reach our goals after the research period of ten weeks during summer.

By the end of the research period, we expect to have a functional working simulator, which we will make available for the community of TinyOS developers and researchers. Along with the simulator, a paper will also be prepared to present the result of the project.

Part B: Research Environment

This research will take place entirely on Bucknell campus, and we will be utilizing lab workstations and library resource. Besides, because TinyOS, nesC and SWAN are all open-source, we will freely consult their source code and online documentation.

Professor Wittie and I will meet formally three times a week to discuss the progress of the project and difficulties that I might encounter. Besides, since professor Wittie will be on campus most of the time, an open-door policy is also applied, meaning that I can walk in and discuss about the project whenever she is available. On my part, I will be doing research independently for the rest of the time during the week.