Yelena Baghdasaryan: ETHZ intern
Pascal von Rickenbach: supervisor

Network Reprogramming for Mica2 Motes

Abstract

To support network programming/reprogramming for mica2 motes, we present the Deluge Basic application written for the TinyOS environment. The application handles the updating process of sensor network applications in a flexible manner. Deluge Basic is built on the already available Deluge application, which is a reliable data dissemination protocol for propagating large data objects from one or more source nodes to many other nodes over a multihop, wireless sensor network. Deluge Basic provides the user a set of possible commands that can be used to start the update process at motes for a certain period of time or to stop it whenever needed. In the result, this application consumes less power compared to already existing tools; moreover, it gives higher flexibility in controlling the time and duration of the update process.

1. Introduction

A sensor network is a network consisting of many, spatially distributed devices using sensors to monitor environmental conditions such as temperature, sound, vibration, pressure, motion, or pollutants at different locations thereby communication with each other via a wireless radio in order to fulfill a given task. Usually these devices are small and inexpensive, so that they can be produced and deployed in large numbers. Due to their small forms their resources in terms of energy, memory, computational power and bandwidth are severely constrained. One example of such small sensor devices are the Mica2 motes that are broadly used for both educational and practical purposes. Each mote is equipped with a radio transceiver, a small microcontroller, and a battery. Normally motes use multiple hops to transport their data to a monitoring computer.

Sensor networks involve three areas: sensing, communications, and computation (hardware, software, algorithms). The mote operating system, TinyOS, provides a set of primitives designed to facilitate the deployment of motes in ad-hoc networks. In such networks, devices can identify each other and route data without prior knowledge of or assumptions about the network topology, allowing the network topology to change as devices move, run out of power, or experience shifting waves of interference. Because of the relative ease of deployment of mote-based sensor networks, they are widely considered for a range of monitoring and data collection tasks.

As mentioned above, TinyOS is an open source component-based operating system and platform targeting wireless sensor networks. It is designed to be able to incorporate rapid innovation as well as to operate within the severe memory constraints inherent in sensor networks. TinyOS is largely written in C and NesC, the latter being a component-based programming language – another extension to the C language. Applications for the TinyOS platform are written mostly in NesC. Thus, in the work presented in this paper, NesC was the working language used together with Java.

In all the areas where mote-based sensor networks are used, the technology to communicate with other sensors through multihop routing protocols are very essential. The tradeoff is to have reliable, flexible, power-efficient and fast communication between sensor nodes. Unlike the traditional method of programming a node over a dedicated link, the embedded nature of these systems requires the propagation of new code over the wireless channel. As wireless sensor network (WSN) research matures, growing testbeds sized at tens of thousands of nodes are now on the horizon, making code propagation over the network a necessity in the development and testing cycle. These factors suggest that network programming (the programming of nodes by disseminating code over the network) is required for the success of WSNs. Specifically, we consider the propagation of complete binary images. While virtual machines can provide low-cost retasking with the use of virtual programs, it is still necessary to have the option of reprogramming nodes with a new binary image since the virtual machine itself may need changes.

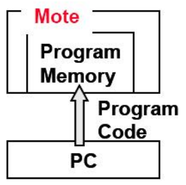
As a solution for the problem stated in the previous paragraph, there exists an application (written for TinyOS platform) called Deluge, which reliably disseminates data to all nodes over a multihop network. Deluge represents data objects as a set of fixed-size pages that provides a manageable unit of transfer allowing spatial multiplexing and supporting efficient incremental upgrades. It can disseminate data with 100% reliability at nearly 90 bytes/second, one-ninth of the maximum transmission rate of the radio supported under TinyOS. Each node running Deluge occasionally advertises the most recent version of the data objects it has available to whatever nodes that can hear its local broadcast. If **s** node receives an advertisement from an older node, **r**, **s** responds with its object profile. From the object profile, **r** determines which portions of the data need updating and requests them from any neighbor that advertises the availability of the needed data, including **s**. Nodes that receive data request then broadcast any requested data. Newly uploaded node then advertises the received data in order to propagate it further to the next hop.

During the communication between motes Deluge produces a huge amount of message transmissions, which is very energy inefficient and reduces network lifetime. To cope with this problem and to provide more flexibility in controlling the programming/reprogramming process we introduce the Deluge Basic application. With this application one can start the original Deluge whenever an update is needed. It can be run for a certain period of time specified by the user. After started, the user can implement any command that was available in Deluge (for changing images, writing new images, etc.). The application can be terminated by user anytime with a single command. As a result, Deluge Basic offers a flexible interface to control Deluge and guarantees more effective power consumption compared to Deluge.

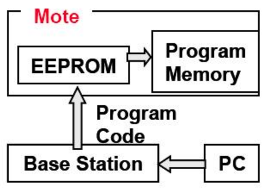
This report goes on with presenting the idea of sensor network reprogramming. Then it describes the NesC specific configuration organized in current work. Afterwards we describe the logic and components of Deluge Basic network reprogramming application and its basic stages. Finally, we give some instructions and explanations on the actual implementations of Deluge Basic at any targeted environment (application).

2. Network reprogramming

The term network reprogramming is used to denote procedure of updating the program code of a network node by using wireless communication rather than a direct connection to the host PC (e.g. loading the program via a serial port). Network reprogramming works mainly in two stages. First, the program code is being transmitted using multiple packets and then stored outside the program memory in persistent storage. Second, the downloaded code is transferred to the program memory and the mote reboots with the new code.



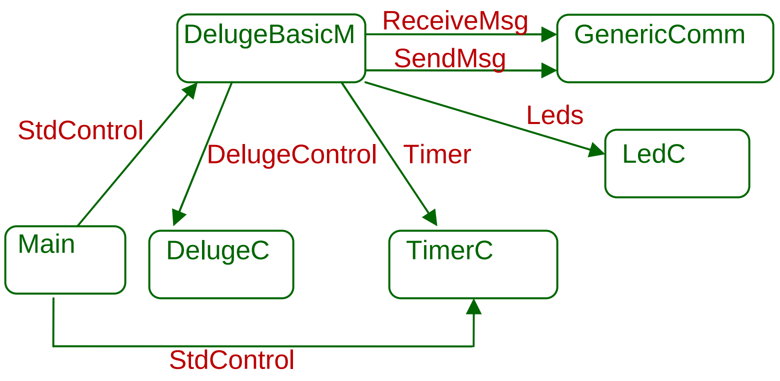
Parallel programming.



Network programming

2.1 NesC model

As TinyOS is an event-driven operating system, the reimplementation of it in nesC is logically coherent with its structure. In nesC programs are built out of components, which are assembled (“wired”) to form whole programs. A component provides and uses interfaces (which are the only point of access to the component). An interface generally models some service (e.g., sending a message) and is specified by an interface type. Interfaces in nesC are bidirectional: they contain commands and events. A command is a function that is implemented by the providers of an interface, and an event is a function that is implemented by its users. There are two types of components in nesC: modules and configurations. Modules provide application code, implementing one or more interfaces. Configurations are used to wire other components together, connecting interfaces used by components to interfaces provided by others. The top-level configuration for Deluge Basic application is presented in the following picture:



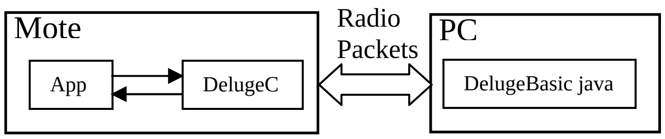
It shows the components used (boxes) and how they are wired to the main logic module DelugeBasicM. The interfaces that are presented by arrows show how the components are interrelated to each other. The DelugeC component is the original Deluge application module, which is controlled using the TimerC and the Main components basically. For example, the Main component uses the interface StdControl provided by DelugeBasicM, to control the whole application.

With the Deluge Basic application we present the implementation of all the interfaces provided by DelugeBasicM, correct wiring of all the components, and the java part corresponding to PC interface.

2.2 The Components

Our application consists of the Deluge Basic application on the motes and a java program on host PC. On a mote, the DelugeBasicM module handles the messages received from other motes and takes the appropriate actions according to the message type. All the necessary modules that need to cooperate with this application are wired in DelugeBasic configuration file as shown in the previous section.

On the PC side, the DelugeBasic java program sends program code and commands via radio in order to control network reprogramming. Between a mote and the PC, messages of reserved message ID (13) are transferred. The format of message is as follows:



DelBasicMsg

sourceAddr	status	time	counter

– sourceAddr: the message source address (PC address)

– status: the current command status (to start:1 or stop:0 the Deluge)

– time: the specified time span to run Deluge

– counter: keeps the level of a propagated message in the network tree (in order to prevent multiple handling of the same message at a mote and for future possible feedback implementation from motes).

Using a usual call in TinyOS for Deluge Basic java program (for example to run Deluge for 10 minutes: % java net.tinyos.tools.DelugeBasic --start --nummin=10) the possible command line parameters for the java program are:

--start --nummin=<num>	/start running Deluge for <num> minutes
--stop	/stop running Deluge anytime when needed
--help	/print the available commands

According to the arguments received from a user the DelugeBasic java program calls either Starter.java for starting or Stopper.java for stopping Deluge on the network nodes. In the Starter and Stopper java programs a message of type DelBasicMsg is created and pushed over the network, setting the fields of message corresponding to the received commands. A message is then sent to the motes, where DelugeBasicM is taking care of them and correspondingly starting or stopping DelugeC through the DelugeControl interface. Then the message is forwarded further to the next hop in order to inform not yet notified motes.

To make use of the Deluge Basic application within another application one must only add DelugeBasic component to its component list in the top-level configuration file. In doing so an application programmer is able to control Deluge via the host PC and is able to update its own application.

The advantage of DelugeBasic over the normal Deluge application is that by controlling the time that Deluge is running we actually decrease the message transmissions between motes, and, sequentially, guarantee a more efficient use of the limited power available at the motes.

3. Conclusions and Future Work

This report introduces the Deluge Basic application for reprogramming mica2 motes in wireless sensor networks. It is built in already available Deluge making it more flexible to a user’s needs according to the timing of reprogramming. By limiting the continuous message transmissions of Deluge it saves power which is a crucial issue in sensor networks. As the presented system is a push-like system, having no feedback from motes, future work can be concentrated on receiving feedback messages from the motes that have finished the update. The user needs to know the total number of motes in the network to check if the feedback was received from all of them. If implemented, the feedback system would add reliability to the Deluge Basic application making it a power-efficient and reliable tool for network reprogramming.