Demo: Terra – Scripting Customized Components for Wireless Sensor Networks

Adriano Branco
Department of Informatics
Pontifícia Universidade Católica do Rio de Janeiro
abranco@inf.puc-rio.br

Noemi Rodriguez
Department of Informatics
Pontifícia Universidade Católica do Rio de Janeiro
noemi@inf.puc-rio.br

Silvana Rossetto
Department of Computer Science
Universidade Federal do Rio de Janeiro
silvana@dcc.ufrj.br

Abstract

Terra is a system that combines the use of configurable component-based virtual machines with a reactive scripting language which can be statically analyzed to avoid unbounded execution and memory conflicts. This approach allows the flexibility of remotely uploading code on motes to be combined with a set of guarantees for the programmer. The choice of the specific set of components in a virtual machine configuration defines the abstraction level seen by the application script. In this work we demonstrate an instance of Terra with high-level abstractions for routing and grouping. We discuss how these abstractions allow the programmer to build a reasonably powerful application in a few lines of code and present the integrated environment that facilitates the development and testing of applications.

Keywords

WSN Wireless Sensor Networks, Virtual Machine, Reactive Programming, Safety

1 Introduction

Terra's goal is to facilitate the creation of WSN applications, specifically as refers to dealing with event-driven programming and with network programming. The Terra model is based on a virtual machine and combines a reactive scripting language with a set of customized components. These components may be selected as needed, creating customized virtual machines with abstractions provided by the component interfaces. A scripting language enforces a programming model that glues components together to create powerful applications in a few lines of code. This makes them suitable for creating programs that benefit from the pre-defined and pre-installed set of components and that can be easily sent over the network.

Terra uses Céu-T as its scripting language and provides a component-based virtual machine VM-T to be customized for different application domains. Céu-T implements a variation of the Céu programming language [2]. The Terra Virtual machine VM-T, with its customizations, runs on WSN nodes with limited resources. We built VM-T using the nesC programming language and the TinyOS [1] operating system. Figure 1 presents the three basic elements of Terra.

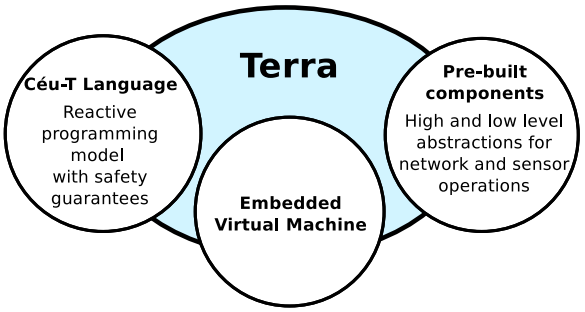

Figure 1. Terra system basic elements.

Terra's guarantees for race freedom and against local starvation and invalid pointers contribute to safer code. These guarantees came, basically, from the static analysis of the Céu compiler and runtime checks of the VM-T execution.

2 VM-T architecture

The Terra virtual machine (VM-T) is composed by three modules as shown in Figure 2.

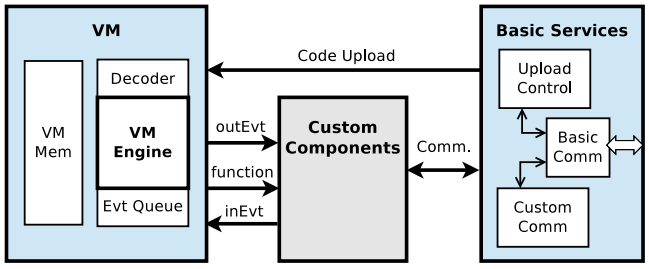

Figure 2. VM-T modules

The *VM* module is the main module. It provides an interface for receiving new application code from the *Basic Services* module and three interfaces for customized events and functions. The *Engine* submodule controls the execution of code interpreted by the *Decoder* submodule and handles external events received from the *Event Queue* submodule.

International Conference on Embedded Wireless Systems and Networks (EWSN) 2016
15–17 February, Graz, Austria

ISBN: 978-0-9949886-0-7

The *Basic Services* module controls the communication primitives to support to code dissemination and to the custom components module interface. The *Upload Control* submodule controls the dissemination protocol and loads code into VM program memory. The *Custom Comm* submodule has a generic interface to support new communication protocols defined at the *Custom Components* module level.

The *Custom Components* module implements specific flavors of Terra. The developer of new customization needs only to implement the custom events and functions inside this module and write the equivalent configuration file to be used by a Céu-T script. It is possible to start from a very basic customization of Terra to include the new events and functions.

3 Terra Customizations

Terra offers a basic library of components that can be included (or not) in a specific virtual machine. As far as possible, these components are parameterized for genericity. New components can also be included by programmer-savvy users to create abstractions for new programming patterns, but the goal of this basic library is to offer a set of components that is sufficient for a range of common applications. This is feasible because most applications for sensor networks are variations of a basic monitoring and control pattern. We organized the needed functionality in four areas: **communication** – support for radio communication among sensor nodes; **group management** – support for group creation and other control operations; **aggregation** – support for information collection and synthesis inside a group; **local operations** – support for accessing sensors and actuators.

4 Demonstration

We begin the demo by presenting the Terra environment composed by: **Editor/Compiler** – text editor adapted to call the Céu-T compiler. **LoadTool** – Java tool to send bytecode to network and receive user messages. **Grid Simulator** – TOSSIM running a VM-T runtime + Graph viewer - a grid formation where each node reach its neighbors. **Node simulation** – shows LEDs, radio sends, and sensor reads. The sensor value may be adjusted during execution of the simulation. Figure 3 shows the interfaces for the Load Tool and the Grid simulator.

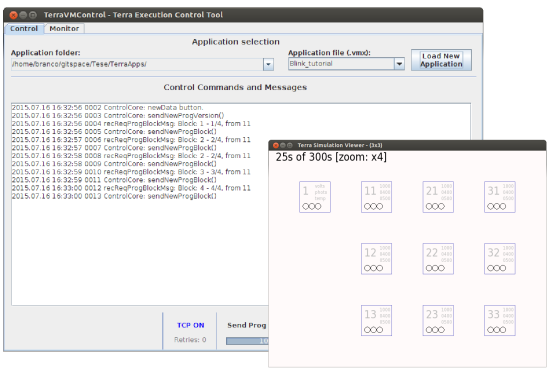

Figure 3. Terra Load Tool and Grid Simulator.

The demo example shows how simple it is to write a script commanding complex operations already embedded in the VM-T runtime. In this application, each node monitors the local temperature and, if the reading is above a certain threshold, asks for its one-hop neighbors to send their current readings, and sends the average of the collected results to the basestation. To demonstrate the working application, we use the facility of forcing a rise in the temperature to be read by a node, triggering an aggregation operation. Figure 4 shows how concise is the script to have this complete application. This example, after compilation, has only 105 bytes of bytecode.

```
1  var group_t gr1;
2  groupInit(gr1,1,0,1,TRUE,eOFF,0);
3  var aggreg_t agA;
4  aggregInit(agA,gr1,SID_TEMP,fAVG,opGTE,0);
5
6  pktype msg from msgBS_t  with
7      var ulong average;
8  end;
9  var msg dataMsg;
10
11 loop do
12     await 10s;
13     emit REQ_TEMP();
14     var ushort tValue = await TEMP();
15     if (tValue > 550) then
16         emit AGGREG(agA);
17         var aggDone_t data = await AGGREG_DONE;
18         dataMsg.average = data.value;
19         emit SEND_BS(dataMsg);
20     end
21 end
```

Figure 4. Céu-T code for average alarm

5 Final remarks

Currently Terra runs in MicaZ, Mica2, and TelosB motes, but it may be easily ported to any platform available for TinyOS. Different platform interoperability is possible when using the same radio standard. Terra is available for download at `http://www.inf.puc-rio.br/~abranco/terra.html`.

6 Acknowledgments

The authors would like to thank the partial support from CNPq – Brazilian National Counsel of Technological and Scientific Development and FAPERJ – Rio de Janeiro Research Foundation.

7 References

[1] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An operating system for sensor networks. In *Ambient Intelligence*. Springer Verlag, 2004.

[2] F. Sant'Anna, N. Rodriguez, R. Ierusalimschy, O. Landsiedel, and P. Tsigas. Safe system-level concurrency on resource-constrained nodes. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 11:1–11:14, New York, NY, USA, 2013. ACM.