Demo Abstract: Network-Wide Sensornet Checkpointing Made Practical

Andreas Löscher[1], Nicolas Tsiftes[2], Thiemo Voigt[2], and Vlado Handziski[3]

[1] Uppsala University, Sweden
andreas.loscher@it.uu.se
[2] Swedish Institute of Computer Science, Sweden
{nvt,thiemo}@sics.se
[3] Technische Universität Berlin, Germany
handziski@tkn.tu-berlin.de

Abstract. Developing sensornet software is difficult partly because of the limited visibility of the system state of deployed nodes. Sensornet checkpointing is a method that allows developers to save and restore full system state of nodes. In this demo, we show an extension module of the Cooja simulator that enables automated sensornet checkpointing on a real testbed. We demonstrate this functionality by connecting Cooja to the TWIST testbed, and show how we can alter the full system state of the network through a simple user interface.

1 Introduction

Sensornet applications can be rigorously tested in simulation, but the conditions in a deployed network are not easy to simulate accurately. Hence, even if the network appears to work well in simulation, it might encounter problems in a real setting, where differences in the environment can have a major effect on the performance and the behavior of the software.

Once something goes wrong in a real setting, it is difficult to find out the root cause of the problem. Limited visibility of the internal system state makes it difficult to analyze where the error stems from. Moreover, the actual infrastructure (e.g., a multi-hop wireless network) needed for transmitting error reports or debugging commands may break down once the fault has occurred. When errors have distributed effects over parts of a network or the whole network, the difficulty of debugging increases, since the faulty node has to be located before it can be inspected.

In this demo, we present a framework for controlling sensornet checkpointing [5] in both real and simulated networks. Our framework consists of a user interface extension of the Cooja simulator [4], and control logic for steering multiple nodes that run the checkpointing software provided by the Contiki operating system [2]. We demonstrate this ability by connecting our framework in Cooja to the TWIST testbed [3] of TU-Berlin, and performing a network-wide checkpoint and rollback cycle on approximately 100 nodes.

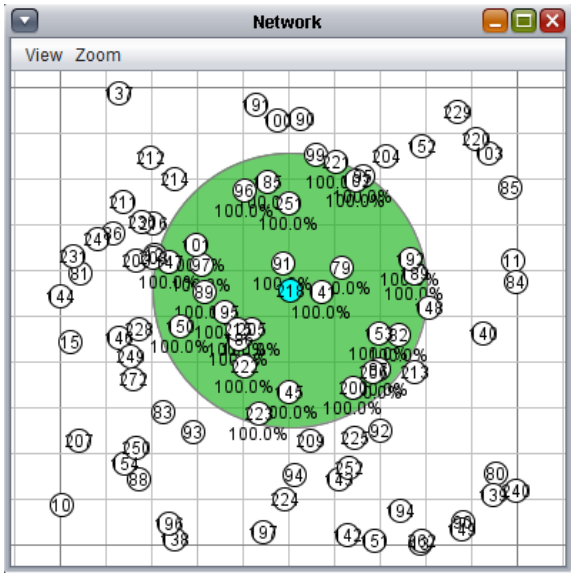
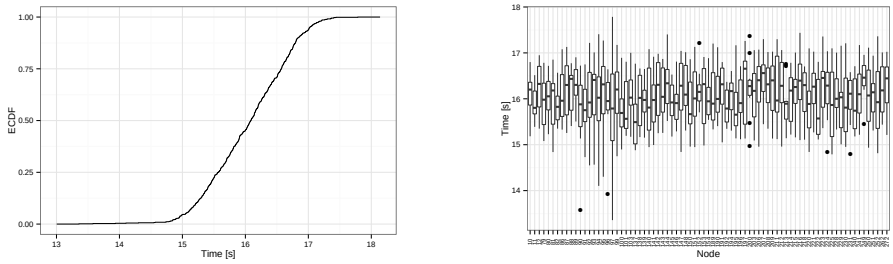


Fig. 1. The TWIST testbed depicted in Cooja's network visualizer. Real networks and simulated networks share interfaces in the simulator, allowing checkpointing tests to occur in both types of networks.

2 Sensornet Checkpointing

Sensornet checkpointing makes it easy to save and restore the full run-time state on real sensor nodes. A checkpoint contains the full volatile memory and the state of modifiable hardware registers, such as the ones used for timers and actuators. A small application runs on each node to manage the saving and restoration of checkpoints. During either of these operations, the checkpoint application freezes the operation of the node, i.e., by turning off interrupts and taking control of the processor. When checkpointing, the application writes the node state into a file in the flash memory. When restoring the state, the application reads the previously saved system state from the file and substitutes it for the current contents of the memory and the hardware registers.

Checkpoint files can be transferred either through a testbed backchannel, or—in the case of a deployed network—through radio communication. The state can then be inspected offline with standard debugging tools, allowing a high visibility into the state of any individual node. Moreover, the state of a node can be rolled back using a checkpoint file generated earlier, thereby making it appear as if the network is jumping back in time to an earlier state. The rollback feature makes it possible to replay the network execution from a known state, and testing how the network is affected when varying the environmental conditions or different memory variables. For instance, the routing tables of nodes could be



(a) ECDF of the checkpointing times of the TWIST nodes.

(b) Individual checkpointing time of each TWIST node.

Fig. 2. Checkpointing takes approximately 16 s on average in the TWIST testbed. For each node, the operation involves saving the full memory state locally to a file in flash memory, and transferring the checkpointing file reliably over the tunneled serial port to the Cooja simulator.

changed offline to test how a routing protocol adapts when suboptimal routes are inserted.

Until now, however, sensornet checkpointing has lacked a framework for performing automatic checkpointing and rolling back of a full network, or an arbitrary set of its nodes, making it difficult to do such experiments. Our framework presented in this demo consists of two plugins for the Cooja simulator. One plugin connects to nodes, either over emulated or real serial ports, and has a capability to interact with the TWIST testbed. Figure 1 shows the TWIST nodes in Cooja's network visualization plugin. It reprograms the testbed by transferring firmware files over HTTP, and connects to the serial port of each individual node, which are tunneled through SSH connections. The other plugin makes it possible to communicate through the serial port with the checkpointing application running on each node. Checkpoint files are transferred between the testbed and the Cooja simulator using the serial port, and the plugin maintains a set of checkpoints for each firmware type and node locally on the computer running Cooja. Figure 2(a) and Figure 2(b) show that the checkpointing operation completes after approximately 16 s for all nodes in the TWIST testbed.

3 Demo Setup

The demo consists of a laptop running the Cooja simulator with our TWIST user interface plugin [1]. Through this GUI, we program the TWIST testbed nodes to run a data collection application built on top of Contiki. We checkpoint the network in different states of the application lifetime, and visualize how the routing topology differs in these states. We achieve this by using Cooja's ability to draw graphical attributes generated by the nodes themselves and communicated to Cooja over their serial ports. We show the simplicity of the checkpointing and

rollback operations, and how individual nodes can be controlled through a GUI that connects to the TWIST testbed.

4 Conclusions

Automating sensornet checkpointing makes it easier for developers and researchers to examine and modify the system state of a wide range of nodes in a real setting. This functionality is demonstrated through our GUI extensions of the Cooja simulator, which shows that sensornet checkpointing is indeed simple to use.

In addition to the presented Cooja-based interface, we are also developing an extension of the TWIST testbed control API that exposes the checkpointing functionality through a set of additional REST resources. For example, the first checkpointed state for the node with serial number M4ADD9U, will be made accessible at the URL `https://www.twist.tu-berlin.de:8001/nodes/M4ADD9UJ/state/chekpoints/1`.

Acknowledgments

This work has been partially supported by EIT ICT Labs, as part of the activity 12149, "From WSN Testbeds to CPS Testbeds", and by CONET, the "Cooperating Objects Network of Excellence", funded by the European Commission under the contract number FP7-2007-2-224053. Thanks to Fredrik Österlind for providing the initial implementation of the TWIST plugin for Cooja.

References

1. Cooja-TWIST Plugin. `https://www.cooperating-objects.eu/testbed-simulation/simulation-platform/`. Visited 2012-12-15.
2. A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the IEEE Workshop on Embedded Networked Sensor Systems (IEEE Emnets)*, Tampa, Florida, USA, Nov. 2004.
3. V. Handziski, A. Köpke, A. Willig, and A. Wolisz. TWIST: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality (REALMAN'06)*, 2006.
4. F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, Tampa, Florida, USA, Nov. 2006.
5. F. Österlind, A. Dunkels, T. Voigt, N. Tsiftes, J. Eriksson, and N. Finne. Sensornet checkpointing: Enabling repeatability in testbeds and realism in simulations. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Cork, Ireland, Feb. 2009.