Demo Abstract: Sensornet Checkpointing Between Simulated and Deployed Networks

Fredrik Österlind, Adam Dunkels, Zhitao He, Nicolas Tsiftes
{fros,adam,zhitao,nvt}@sics.se
Swedish Institute of Computer Science

ABSTRACT

Sensor network development is notoriously difficult due to the low visibility of sensor platforms and systems. We propose sensornet checkpointing to increase the visibility of sensor networks. With sensornet checkpointing, we transfer network-wide application checkpoints between simulated and real networks. This approach enable advances in many research areas: visualization, repeatable experiments, fault injection, and application debugging. We demonstrate sensornet checkpointing on a network of Tmote Sky motes running Contiki.

1. INTRODUCTION

Sensor network development is difficult due to the low resources of current sensor platforms; observing a sensor network application may affect the application execution, causing Heisenbugs and incorrect evaluations. Several research directions have emanated from the lack of visibility, such as design-time targeting at increasing visibility [7], and online source-level network debugging [8].

Simulation has proven invaluable for development and testing of sensor network applications. Simulation provides in-depth execution details, a rapid prototyping environment, nonintrusive debugging, and repeatability. Simulation is, however, not enough. The reason for this, as argued by numerous researchers, is over-simplified simulation models not able to realistically capture the behavior of deployed networks.

Clock
checkpoint
rollback
network execution stopped
unfreeze network
network freeze

Figure 1: Checkpointing freezes all nodes in the network at a given point in time. The state of all network nodes can be either stored on the individual nodes for offline processing, or be directly downloaded to an external server.

Sensornet checkpointing [5] combines advantages of both real and simulated networks. The technique enables advances in many active research areas such as sensor network debugging and fault injection. The application state of a real network can be debugged with full execution details by first checkpointing the real network, and then rolling back the application state in a simulated network. Similarly, fault injection into a real sensor network can be performed by first injecting faults in a controlled simulation environment, and then transferring the simulated application state to the real network.

We enable sensornet checkpointing for deployed networks. We have previously developed sensornet checkpointing for transferring state between testbed and simulation [5]. By enabling nodes to store node local checkpoints in external flash, checkpointing is made possible also in deployed networks.

Node checkpoints can be triggered in two ways. Checkpointing can be network-wide: all nodes synchronously store local checkpoints to external flash. Network-wide checkpointing can be configured to trigger periodically. Checkpointing can also be on-demand: an external ob-
server sends a checkpoint command to a specific node, triggering a node local checkpoint that the observer may download.

2. SENSORNET CHECKPOINTING

Sensornet checkpointing consists of two network-wide operations: checkpoint and rollback, and is inspired by the substantial body of work in distributed checkpointing [1]. The checkpoint operation extracts the application state at a given point in time. The rollback operation restores a previous application state and continues application execution. Both operations are synchronous: the entire network execution is frozen throughout checkpointing operations. The network application is hence unaware of any checkpointing operations occurring. Figure 1 demonstrates network checkpointing.

Every node performs a local checkpoint during the network-wide checkpoint operation. A node local checkpoint is performed by storing the local application state to the external flash. The application state in our Tmote Sky [4] implementation consists of 10kb node memory (RAM), the microcontroller hardware timers, the LEDs, and the onboard 802.15.4 CC2420 radio chip state.

Node local checkpoints can be downloaded from the network wirelessly, or via optional serial connections to nodes. We use Contiki’s default multi-hop bulk transfer protocol to download checkpoints to a computer-connected sink node.

Checkpointing operations can be performed on both simulated and real networks. A network checkpoint can be rolled back to both simulation and testbed. By checkpointing in one domain and rolling back in another, we transfer application state between the domains: see Figure 2.

For simulating networks, we use the Contiki network simulator COOJA [4]. COOJA is equipped with the MSP430 emulator MSPSim [3]. MSPSim supports instruction set emulation of Tmote Sky sensors, hence node local checkpoints performed in simulation are compatible with real Tmote Sky nodes. Note that checkpointing operations in simulation are performed as in real networks: node local checkpoints in simulation are downloaded from the nodes to the simulated sink node.

Many research areas benefit from the ability to transfer application state between real and simulated nodes. Standard simulation tools can be used to analyze network checkpoints from real networks, enabling new visualization techniques. The network execution can furthermore also be resumed in simulation, a feature that may enable new network debugging approaches. By performing periodic checkpoints of an application, any detected interesting application phases can be repeated for rapid network evaluation.

3. DEMO SETUP

We demonstrate sensornet checkpointing on a network of Tmote Sky motes running Contiki [2]. The same network is also simulated in the Contiki network simulator COOJA [4]. We perform checkpointing operations to transfer network-wide application state between the simulator and the real sensor network.

We discuss and demonstrate how sensornet checkpointing is used as an underlying technique to implement source-level debugging, fault injection, and repeatable experiments.

4. REFERENCES