Poster Abstract: Storage-Centric Debugging of Performance Problems in Sensor Networks

Tony O'Donovan[1], Cormac J. Sreenan[1], Nicolas Tsiftes[2], Zhitao He[2], and Thiemo Voigt[2]

[1] University College Cork
[2] Swedish Institute of Computer Science

1 Introduction

After almost ten years of research into sensor networks, deployments continue to fail [5]. Such failures are due to the distributed nature of sensor networks and the complex interactions of protocols [1]. A variety of debugging tools have been developed to determine the exact causes of failures, and to mitigate the poor visibility into the operation of many sensor network protocols [8].

Performance debugging of sensor networks has been limited to testbed experiments in the pre-deployment phase, where the developer typically has access to a separate wired channel for unintrusive debugging. Since it is difficult to mimic the environment of a deployed network in a testbed, debugging tools for deployed networks are desirable. Examples of tools developed for this purpose are Sympathy [6] and Clairvoyant [4]. These tools are limited to data of low detail or low-rate querying, however, since the performance debugging itself must not cause further protocol disturbances.

We propose a storage-centric method for debugging where performance and log data are stored locally on the node. This method opens new possibilities for debugging: 1) A node can process the data itself. 2) Neighbors can share performance and log data to detect root causes of problems and maybe even resolve problems. 3) An operator can in an efficient one-hop batch download the data using a mobile node. 4) A node can decide on its own when it is time to send stored log data to the sink for processing. 5) The sink can request data on demand at the desired level of granularity.

2 Storage-Centric Performance Debugging

Currently the most common approach to network monitoring and debugging involves sending state to a base station [6]. This approach implies a trade-off between the size and frequency of updates, and the quality of the data available to the network operator. Frequent updates give a more accurate picture of the network, but at the cost additional traffic and control overhead. In large multi-hop networks this cost can be considerable.

An alternative is to use the external flash storage provided by many sensor node platforms. For example, the Moteiv Tmote Sky has 1 Mb of external flash.

With an efficient file system like Coffee [7] we can store the node and network-performance data on the nodes rather than sending it to the base station or discarding it. With the logged data, the nodes detect, diagnose, and fix faults–either individually or with the help of their neighbours. Furthermore, the sink can ask nodes for stored information at the desired level of detail. The benefit of this approach is reduced overhead since less information is sent to the base station. This leads to energy savings, less contention, and more bandwidth for application traffic.

Storage-centric debugging is useful in many scenarios. For example, if part of a network suffers from excessive packet loss, or in case of a network partition, it is possible to send a network operator into the field for diagnosis. The operator can either download all data to a mobile management station for analysis, or query the nodes which can return the required results only. In addition, the operator can retrieve any application data not delivered to the base station from the node's local storage. Network performance can be degraded because of environmental conditions, external interference, or faulty hardware and software. When a node notices anomalous behaviour, it can check the stored data for similar anomalies. If found, they can be correlated with other data available such as temperature or humidity to identify patterns helping to fix the problem.Using local storage, the data is also available after a reboot. Therefore, a node can report to the base station when it notices that it has rebooted frequently.

3 Preliminary Evaluation

Although our work is in an early stage, we have made a preliminary evaluation of storage-centric performance debugging, showing promising results. Our experimental setup consists of a set of emulated Tmote Sky motes in the cycle-accurate Cooja/MSPsim simulator [3]. The purpose of the experiment is to compare the cost of logging data locally with the cost of sending or transmitting it over the radio. Each log record is 64 bytes large, and consists of debug information from several parts of the system.

We run the Contiki operating system on the motes, and use the Coffee file system and the unicast primitive in the Rime communication stack [2]. The MAC protocol in use beneath Rime is X-MAC, configured with a 1.25 % duty cycle. The duty cycle is selected using the default radio on-time of $\frac{1}{160}$ s, and an off-time of 0.5 ms, which is half of the average packet transmission interval. We send 1000 packets and measure the per-packet energy consumption using Contiki's Compower library.

Figure 1 shows that the energy consumption is considerably lower for storing the data locally. Whereas the packet costs vary, the cost of storing the data locally is the same since Coffee's optimized file append operation closely follows the performance of the underlying flash device driver.

In our experiment, we have used an optimal setup for the unicast transmissions: a single-hop network without interference. The total cost of transmitting the data from a mote to a sink in a multi-hop network is a multiplicity of the

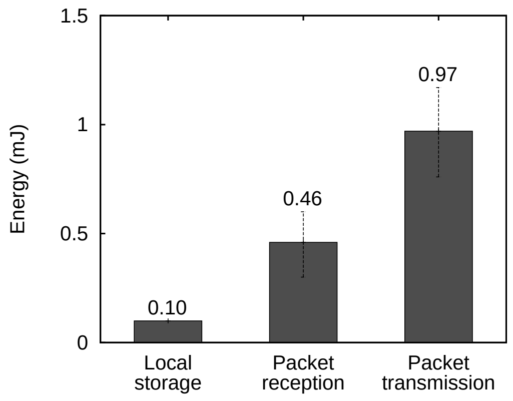

Fig. 1. Storing performance data to the flash requires significantly less energy than sending the data over the radio, even under optimal conditions.

single-hop cost. Furthermore, the contention and the collision rate in the network will increase significantly if performance debugging packets are transmitted at high rates: our 1000 packets sent at the network-layer generated 3762 packets in the link layer because of the strobe procedure in X-MAC.

4 Conclusions

In this paper we propose a storage-centric method for performance debugging. Our results suggest that this method is energy-efficient, and motivate further efforts for realizing this into a powerful debugging tool.

References

1. J. Choi, M. Kazandjieva, M. Jain, and P. Levis. The case for a network protocol isolation layer. In *Proceedings of ACM SenSys*, Berkeley, CA, USA, 2009.
2. A. Dunkels, F. Österlind, and Z. He. An adaptive communication architecture for wireless sensor networks. In *Proceedings of ACM SenSys*, Sydney, Nov. 2007.
3. J. E. et al. Accurate Network-Scale Power Profiling for Sensor Network Simulators. In *Proceedings of EWSN 2009*, Cork, Ireland, Feb. 2009.
4. J. Y. et al. Clairvoyant: a comprehensive source-level debugger for wireless sensor networks. *Proceedings of ACM SenSys*, 2007.
5. K. L. et al. Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In *Proceedings of IPDPS*, 2006.
6. N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In *Proceedings of ACM SenSys*, 2005.
7. N. Tsiftes, A. Dunkels, Z. He, and T. Voigt. Enabling Large-Scale Storage in Sensor Networks with the Coffee File System. In *Proceedings of IPSN*, San Francisco, 2009.
8. M. Wachs, J. Choi, J. Lee, K. Srinivasan, Z. Chen, M. Jain, and P. Levis. Visibility: A new metric for protocol design. In *ACM SenSys*, 2007.