PRO 05 Seeing the Forest Through the Trees: Distributed Logging for Distributed Systems

PRO 05.1 People

- Principal Investigator: Mani Srivastava, Todd Millstein
- Todd Millstein, Assistant Computer Science, UCLA, Mani Srivastava, Electrical Engineering, UCLA.
- Roy Shea, Graduate Student, Computer Science, UCLA.

PRO 05.2 Overview

Observing the internal execution of severely resource constrained wireless embedded devices remains a critical block to widespread adoption of embedded wireless sensing systems. Exposing this execution is critical for enabling system developers to debug and understand the systems that they deploy. Past CENS projects have provided initial steps to minimize the occurrence of and diagnose instances of bugs in such systems:

- Kairos’s centralized program abstraction enabled writing a single clear and maintainable program for an entire network of devices rather than requiring developers to manually implementing tricky distributed logic.
- Lighthouse’s static analysis isolated resource sharing problems before system deployment preventing a class of problems commonly found on embedded systems.
- Harbor’s software isolation prevented runtime memory faults on embedded devices without MMUs, which have a long history of runtime memory faults.
- Sympathy’s runtime heartbeats carrying node centric diagnostic information helped diagnose many network level faults within the class of embedded wireless sensing systems.

Our current work continues along this vector by providing detailed insight into the runtime operations of these bottom tier systems. We are working to provide this insight through three core areas of innovation for this class of systems: improving the ease of obtaining high quality log data, developing novel mechanisms for triggering log retrieval from within a deployed or testbed network, and minimizing the overhead of log correlation.

PRO 05.3 Approach

Our approach focuses on three core tasks: obtaining semantically dense logs, exploiting triggered log retrieval, and building exploring new log correlation techniques.

Inexpensive storage media provides traditional server systems with the luxury of logging first and looking for meaningful data within the logs later. But on resource constrained wireless and embedded sensing system, where both memory and stable storage are limited, there is only room to log the highest quality data. Our research creates high quality logs by combining static program analysis with automated insertion of logging statements. Static analysis helps optimize away redundant logging while ensuring complete log coverage over specified regions of interest. For example, we’ve developed a domain specific compression technique that captures complete and compact runtime call traces by only logging runtime control flow decisions affecting call dispatch. Automated log insertion removes the burden of, and inevitable mistakes introduced from, manual code instrumentation. We measure the success of this work by comparing the runtime memory usage, transmitted log size, and execution impact of our improved logging mechanisms to the current logging alternates.

Triggered log retrieval strives to collect logs only when they will provided the greatest utility. The key to providing system users with this ability is developing a clear language within which to describe the conditions, perhaps distributed across multiple devices, when a particular type of log is of value. Declarative and macroprogramming approaches are two promising areas that provide foundation work that we plan to apply to the specification of triggers.
Our research targets networks of embedded devices, so it is critical that our resulting infrastructure includes the ability to correlate logs originating from more than one source. To reduce both the bandwidth consumed by log transfer and the delay in identifying events of interest, we are exploring techniques to push log correlation deeper into the network. We hope to reduce correlation overhead to such a degree that it can be used directly as an expressive triggering mechanism for gathering logs.

PRO 05.4 System Description
Over the past year we have focused on creating the infrastructure required for gathering high quality logs from wireless and embedded systems. The key component of our logging implementation is a suite of tools called LowLog that provide a variety of automated instrumentation services. These services instantiate our new ideas for gathering high quality logs. Standard logging techniques, such as inserting an arbitrary logging preamble to functions to log, for example, have been automated and made readily available to developers. At the same time the suite of tools enables exploring non-traditional logging techniques. For example we explore caller side logging, capturing compressed runtime control flow decision logs, and the impact of alternate log token encoding mechanisms.

Our current framework was first developed for the SOS operating system and recently ported to the TinyOS operating system. The current TinyOS architecture is illustrated in the figure above. Our logging system is added to TinyOS applications by extending the main application configuration with the LogTap component to provide safe sharing of the messaging infrastructure between our logging framework and the original application.

Developer specified regions of interest (ROI) drive code analysis and subsequent instrumentation performed by LowLog. Back end utilities are used to present users with a view into the logs that are streamed out of the network from the instrumented applications.

This infrastructure provides a foundation for gathering high quality log data upon which to explore triggered log retrieval and better log correlation.

PRO 05.5 Accomplishments
Using the LowLog suite of tools with our logging infrastructure we have demonstrated that high quality call traces can be collected using only a fraction of the bandwidth required by naive, but commonly used, logging mechanisms. Additionally we have observed that other forms of logs, such as recording run time control flow decisions in subsystems of interest, can be captured using a reasonable bandwidth. Finally, we are beginning to use the LowLog suite in our daily development cycle to diagnose problems within the wireless and embedded sensing systems we work with.
**PRO 05.6 Future Directions**

From the base provided by LowLog we are now transitioning our work towards triggered log collection and improved log correlation. Our initial work will explore the role of triggering log retrieval based on violation of expected log behavior. An initial version of this can be thought of as online and per-component suppression of repeated or expected logs. At the same time we will begin exploring how distributed triggers can be expressed in macroprogramming type languages.

In the second half of the year we will focus our work on triggering log collection based on log correlation. Initial correlation work will use a centralized server as we explore the types of correlation useful for identifying meaningful events across logs. Subsequent work will migrate this correlation out into the distributed network of embedded sensing devices.