
Nathan Coopriders — Research Statement

1229 University Village
Salt Lake City, UT 84108
<http://www.cs.utah.edu/~coop>

home +1 801 585 4713
cell +1 801 856 3087
coop@cs.utah.edu

Overview

I enjoy researching on the border between hardware and software, staying on the software side. Compilers dwell on this border, and the majority of my research focuses on analyses and transformations which could enhance a compiler. Embedded systems also dwell on this border, so I work with compilers for embedded software. Specifically, my research makes embedded software efficient and reliable through applied program analysis and transformations.

I initially target embedded software for wireless sensor networks (WSNs). Software for WSNs is highly constrained, with only a few KB of RAM, less than a MB of ROM, a microcontroller (MCU) operating at only few MHz, and two AA batteries for power. WSN applications exhibit critical difficulties for the static analyses which enable compiler optimization and verification. For example, applications use direct hardware access, depend on volatile data, and have unstructured interrupt-driven concurrency. Solutions to these difficulties also apply outside of the WSN domain.

Specific Research Efforts

My current research centers around dataflow analysis for MCUs and applications of analysis results.

Static Analysis of Embedded C A straightforward instantiation of traditional dataflow analysis techniques fails to provide adequate precision for aggressive optimization of interrupt-driven MCU systems. We developed cXprop, a dataflow analysis tool for C which combines traditional abstract interpretation with several new techniques such as pluggable abstract domains, novel modeling of interrupt-driven concurrency, and allowing dataflow through volatile variables when safe to do so [2]. cXprop is interprocedural, context-insensitive, and flow sensitive; it integrates several synergistic analyses such as value-flow analysis, pointer analysis, and callgraph construction. We applied cXprop to TinyOS programs in order to reduce code, data, and estimated stack size while also decreasing the duty cycle.

Efficient Memory and Type Safety for Sensor Network Applications Reliable WSN software is difficult to create: applications are concurrent and distributed, hardware-based memory protection is unavailable, and severe resource constraints necessitate the use of unsafe, low-level languages. Our work improves this situation by providing efficient memory and type safety for TinyOS 2 applications running on the Mica2, MicaZ, and TelosB platforms [1]. Safe execution ensures that array and pointer errors are caught before they can corrupt RAM. Our contributions include showing that aggressive optimizations can make safe execution practical in terms of resource usage; developing a technique for efficiently enforcing safety under interrupt-driven concurrency; extending the nesC language and compiler to support safety annotations; finding previously unknown bugs in TinyOS; and, finally, showing how to exploit safety to increase the availability of WSN applications with unfixed memory errors.

RAM Compression RAM is the critical resource on many MCU-based systems, yet today's compilers contain no optimizations that affect RAM usage. We developed *offline RAM compression*, an automated source-to-source transformation that reduces a program's data size [3]. The transformation encodes and packs statically allocated scalars, pointers, structures, and arrays based on the results of a whole-program analysis in the value set and pointer set domains. We target embedded software written in C that relies heavily on static memory allocation and runs on Harvard-architecture MCUs supporting just a few KB of on-chip RAM. On a collection of embedded applications for AVR MCUs, our transformation reduces RAM usage by an average of 12%, in addition to a 10% reduction through a dead-data elimination pass that our whole-program analysis also drives, for a total RAM savings of 22%. We also developed a technique for giving developers access to a flexible spectrum of tradeoffs between RAM consumption, ROM consumption, and CPU efficiency. This technique uses a model for estimating the cost/benefit ratio of compressing each variable and then selectively compressing only those variables that present a good value proposition in terms of the desired tradeoffs.

Future Work

I will continue to research static analysis methods of alleviating the burden on developers of MCU software due to system constraints and language bugs. My effort will center around continued development, community adoption, and language integration. I will carry out my research in terms of the WSN community and in terms of larger systems as well. There still exists a significant amount of application knowledge, unavailable to state-of-the-art program analyses, which I will uncover through improved static analysis. I will extend many ideas from static analysis for WSNs into larger embedded systems with more complicated locking mechanisms and concurrency models. I have briefly investigated using static analysis results to help with model checking, but expect much more benefit to be gleaned there. To find user communities for my tools, I will increase their debugging support and decrease their execution time. I especially look forward to investigating distributed program analysis for increased speed. Finally, I plan to add/change language features to promote static analysis. Automation and familiarity are key features in order to not push away potential users. I expect that small language changes will exhibit significant increases in analysis precision.

References

- [1] Nathan Coopriider, William Archer, Eric Eide, David Gay, and John Regehr. Efficient memory safety for TinyOS. In *Proc. of the 5th ACM Conference on Embedded Networked Sensor Systems (SenSys 2007)*, Sydney, Australia, November 2007.
- [2] Nathan Coopriider and John Regehr. Pluggable abstract domains for analyzing embedded software. In *Proc. of the 2006 Conf. on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, pages 44–53, Ottawa, Canada, June 2006.
- [3] Nathan Coopriider and John Regehr. Offline compression for on-chip RAM. In *Proc. of the ACM SIGPLAN 2007 Conf. on Programming Language Design and Implementation (PLDI)*, San Diego, CA, June 2007.