SWiFT: A Feedback Control and Dynamic Reconfiguration Toolkit

Ashvin Goel, David Steere, Calton Pu, Jonathan Walpole
Department of Computer Science and Engineering
Oregon Graduate Institute, Portland

Currently, the task of building adaptive system software relies on wizardry. Feedback controls for such software systems are written in an ad-hoc manner and are often brittle. As a result, it is difficult to move an existing control, such as TCP flow control [1], to a new domain such as admission control for CPU scheduling. In addition, existing controls are built with implicit assumptions about the system's run-time environment and can become unstable in the face of large or discontinuous variations in the environment.

We advocate a systematic approach for building adaptive system software based on feedback-control theory. We have implemented the SWiFT toolkit that incorporates this approach. Feedback control helps produce predictable control components. It requires the control goal and design specifications to be clearly stated, thus allowing analysis of properties such as stability. Our approach allows us to leverage the existing body of knowledge in hardware control for controlling software systems.

SWiFT allows systematic implementation of feedback-control mechanisms by providing a framework and methodology for building controls that are modular, dynamically reconfigurable, and predictable. Modularity results from our use of components and containers as the underlying abstraction. SWiFT also allows easy dynamic reconfiguration of components by limiting the interaction between components to a simple input/output model and by supporting guarding and replugging of controllers [2]. This reconfiguration allows the application to adapt efficiently across a wide range of operating conditions. SWiFT supports predictability by providing analysis tools based on control theory. Also SWiFT provides GUI debugging tools such as a software oscilloscope and a library of feedback components such as low pass filters to simplify building adaptive system software.

We have implemented SWiFT in C++ and Java and we have applied it to user-level applications running on Windows NT. Version 1.0 of SWiFT is available (along with a tutorial) at http://www.cse.ogi.edu/DISC/projects/swift. We are currently building a visual editor for designing, implementing, and monitoring controls using SWiFT.

The basic abstractions in SWiFT are *feedback components* and f*eedback containers*. Feedback components read data from their *input port*(s), calculate an output value based on their characteristic behavior, and pass the value to their *output port*. A control circuit is built by connecting a component's output port to input ports of one or more components. A feedback container, provide modularity and hierarchical structure. It is a feedback component that contain other feedback components and containers, and defines a circuit of connections among its children and its input and output ports.

Another aspect of SWiFT, *dynamic reconfiguration*, consists of tuning a control circuit's parameters, or replacing parts of the control circuit at run-time. This reconfiguration is the *controlled* system's response to drastic changes in the underlying environment that violate the controller's basic design assumptions. For instance, TCP's adaptive flow-control algorithm performs poorly over wireless links. Replacing this policy with one more suited to wireless use results in better performance [3]. Dynamic reconfiguration is similar to hardware hot-swapping with the addition that the swapping is done automatically. SWiFT can dynamically reconfigure feedback components in controllers based on user-specified predicates on system properties, called *guards*. Guards are explicitly programmed by the controller's designer.

We are currently developing adaptive control mechanisms using SWiFT on three diverse system domains on NT: a streaming media player, an informed multimedia prefetching system, and a feedback-based proportional share CPU scheduler.We have chosen NT for its modular structure, and widespread use as a platform for multimedia applications. Our designs make heavy use of NT's user-level device drivers and Microsoft's DirectShow infrastructure.

This project was supported in part by DARPA contracts/ grants N66001-97-C-8522, N66001-97-C-8523, and F19628-95-C-0193, and by Tektronix, Inc. and Intel Corporation.

References

[1] V. Jacobson. Congestion avoidance and control. *ACM Computer Communication Review; Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988*, 18, 4:314–329, 1988.

[2] Calton Pu, Tito Autrey, Andrew Black, Charles Consel, Crispin Cowan, Jon Inouye, Lakshmi Kethana, Jonathan Walpole, and Ke Zhang. Optimistic incremental specialization: Streamlining a commercial operating system. In *SOSP'95*, pages 314–324, Copper Mountain resort, Colorado, USA, December 1995.

[3] R. Yavatkar and N. Bhagwat. Improving end-to-end performance of TCP over mobile internetworks. In *Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, U.S., 1994.