ORCHESTRA: A Probing and Fault Injection Environment for Testing Protocol Implementations

Scott Dawson, Farnam Jahanian, and Todd Mitton
Real-Time Computing Laboratory
Electrical Engineering and Computer Science Department
University of Michigan
Ann Arbor, MI 48109-2122, USA
{sdawson,farnam,mitton}@eecs.umich.edu

Ensuring that a distributed system meets its prescribed specification is a growing challenge that confronts software developers and system engineers. Meeting this challenge is particularly important for applications with strict dependability and/or timeliness constraints. We have developed a software fault injection tool, called ORCHESTRA, for testing dependability and timing properties of distributed protocols. ORCHESTRA is based on a simple yet powerful framework, called *script-driven probing and fault injection*. The emphasis of this approach is on experimental techniques intended to identify specific "problems" in a protocol or its implementation rather than the evaluation of system dependability through statistical metrics such as fault coverage. Hence, the focus is on developing fault injection techniques that can be employed in studying three aspects of a target protocol: i) detecting design or implementation errors, ii) identifying violations of protocol specifications, and iii) obtaining insights into the design decisions made by the implementors.

Script-driven probing and fault injection views a distributed protocol as an abstraction through which a collection of participants communicate by exchanging a set of messages. Each protocol is specified as a layer in the protocol stack such that each layer, from the device-level to the application-level protocol, provides an abstract communication service to higher layers. In our approach, a probe/fault injection (*PFI*) layer is inserted between any two consecutive layers in a protocol stack. The *PFI* layer can execute deterministic or randomly-generated test scripts to probe the participants and inject various faults into the system. By intercepting and filtering messages between two layers in a protocol stack, the *PFI* layer can delay, drop, reorder, duplicate, and modify messages. Furthermore, the *PFI* layer can introduce spontaneous messages into the system to probe protocol participants and to orchestrate the system execution into a particular path. Because fault injector actions are driven by scripts that are interpreted at run-time, the ORCHESTRA tool users are able to (re-)specify tests without recompiling the protocol stack. Several ORCHESTRA tool implementations have been built based on this framework. These tools have been used to conduct extensive experiments on several implementations of commercial and prototype systems in order to demonstrate the utility of script-driven probing and fault injection [1–3].

Past research closely related to this work can be classified into two areas: (a) network packet monitoring and filtering approaches; and (b) fault injection techniques for protocol testing. A detailed discussion of the related work can be found in [3].

This work is supported in part by a research grant from the U.S. Office of Naval Research, N0014-95-1-0261, and a research grant from the National Science Foundation, CCR-9502341.

References

[1] Scott Dawson and Farnam Jahanian, "Deterministic Fault Injection of Distributed Systems," in *Lecture Notes in Computer Science: Unifying Theory and Practice of Distributed Computing*. Springer-Verlag, September 1994.

[2] Scott Dawson and Farnam Jahanian, "Probing and Fault Injection of Protocol Implementations," *Proc. Int. Conf. on Distributed Computer Systems*, pp. 351–359, May 1995.

[3] Scott Dawson and Farnam Jahanian and Todd Mitton, "Testing of Fault-Tolerant and Real-Time Distributed Systems via Protocol Fault Injection," in *International Symposium on Fault-Tolerant Computing*, Sendai, Japan, June 1996.

Proceedings of the 2nd International Computer Performance and Dependability Symposium (IPDS '96)
1087-2191/96 $10.00 © 1996 IEEE