THE WILLOW SURVIVABILITY ARCHITECTURE

John Knight[*] Dennis Heimbigner[+] Alexander Wolf[+]
Antonio Carzaniga[+] Jonathan Hill[*]
Premkumar Devanbu[**] Michael Gertz[**]

*Department of Computer Science
University of Virginia
Charlottesville, VA 22904-4740
{knight, jch8f}@cs.virginia.edu

+Department of Computer Science
University of Colorado
Boulder, CO 80309-0430
{alw, dennis, carzanig}@cs.colorado.edu

**Department of Computer Science
University of California
Davis, CA 95616-8562
{gertz, devanbu}@cs.ucdavis.edu

Introduction

The Willow architecture provides a comprehensive architectural approach to the provision of survivability [8] in critical information networks. It is based on the notion that survivability of a network requires reconfiguration at both the system and the application levels. The Willow notion of reconfiguration is very general, and the architecture provides reconfiguration mechanisms for both automatic and manual network control.

In this paper we summarize the Willow concepts and provide an overview of the Willow architecture. Finally we describe a demonstration application system that has been built on top of a prototype Willow implementation.

Willow Concepts

The Willow survivability architecture [6] is a secure, automated framework that effects a wide spectrum of both *proactive* and *reactive* reconfigurations of large-scale, heterogeneous, distributed systems. The architecture is designed to enhance the survivability of critical networked information systems by: (a) ensuring that the correct configuration is in place and remains in place during normal operation; (b) facilitating the reconfiguration of such systems in response to anticipated threats before they occur (including security threats); and (c) recovering from damage after it occurs (including security attacks).

Proactive reconfiguration adds, removes, and replaces components and interconnections, or changes their mode of operation. This form of reconfiguration, referred to as *posturing*, is designed to limit possible vulnerabilities when the possibility of a threat that will exploit them is heightened. An example of posturing would be a network-wide shutdown of non-essential services, strengthening of cryptographic keys, and disconnection of non-essential network links if the release of a new worm is expected or infections have already been observed.

In a complementary fashion, reactive reconfiguration adds, removes, and replaces components and interconnections to restore the integrity of a system in bounded time once damage or intrusions have taken place. In Willow, this is, in fact, network *fault tolerance* and mechanisms are provided for both the detection of errors and recovery from them [7]. As an

example of fault tolerance, the network might detect a coordinated attack on a distributed application and respond automatically by activating copies of the application modules on different network nodes while configuring the system to ignore the suspect modules. The system would perform this modification rapidly and inform system administrators of the change.

The Willow concept derives from a realization that software configuration control and network fault tolerance are two different aspects of the general problem of overall control of distributed systems. Both utilize specialized knowledge about the applications, the resources available and the network state to prepare and react to changing conditions for applications in a network. The difference lies in the time frames at which the two aspects operate, and in the mechanisms used to detect and respond to circumstances needing action. Network fault tolerance is mostly time-bounded, needing prescribed responses to anticipated faults. Software configuration management involves run-time analysis of network state to determine necessary basic actions from a series of prescribed facts and newly available network state (new software versions, operating system conditions, etc.)

The Willow architecture supports reconfiguration in a very broad sense, and reconfiguration in this context refers to any scenario that is outside of normal, "steady-state" operation. Thus, for example, initial system deployment is included intentionally in this definition as are system modifications, posturing and so on. All are viewed merely as special cases of the general notion of reconfiguration. More specifically, the system reconfigurations supported by Willow are:

- Initial application system deployment.
- Periodic application and operating system updates including component replacement and re-parameterization.
- Planned posture changes in response to anticipated threats.
- Planned fault tolerance in response to anticipated component failures.
- Systematic best efforts to deal with unanticipated failures.

Reconfiguration takes place after a decision is made that it is required. An important element of the Willow approach is the integration of information from sensing mechanisms within the network (such as intrusion detection systems) and information from other sources (such as intelligence data). Since reconfiguration could be used as a means of network attack, the input that is used in the Willow decision-making process is managed by a comprehensive trust mechanism [2].

Summary of the Architecture

The fundamental structure of the Willow architecture is a *control mechanism* that is there to deal with network changes. The individual schemes for dealing with different reconfiguration scenarios might be different, but conceptually they are instances of a common control paradigm that pervades the architecture. This common control loop, and the primary elements of the architecture, are depicted in Figure 1. The control loop contains sensing, diagnosis, synthesis, coordination, and actuation components to affect desired network maintenance.

Fundamental to the implementation of the control loop are: (a) the use of formal languages for the specification of control; (b) large-scale, wide-area communication via the publish-subscribe paradigm; (c) reconfiguration coordination capability; and (d) homogeneous actuation across heterogeneous environments.

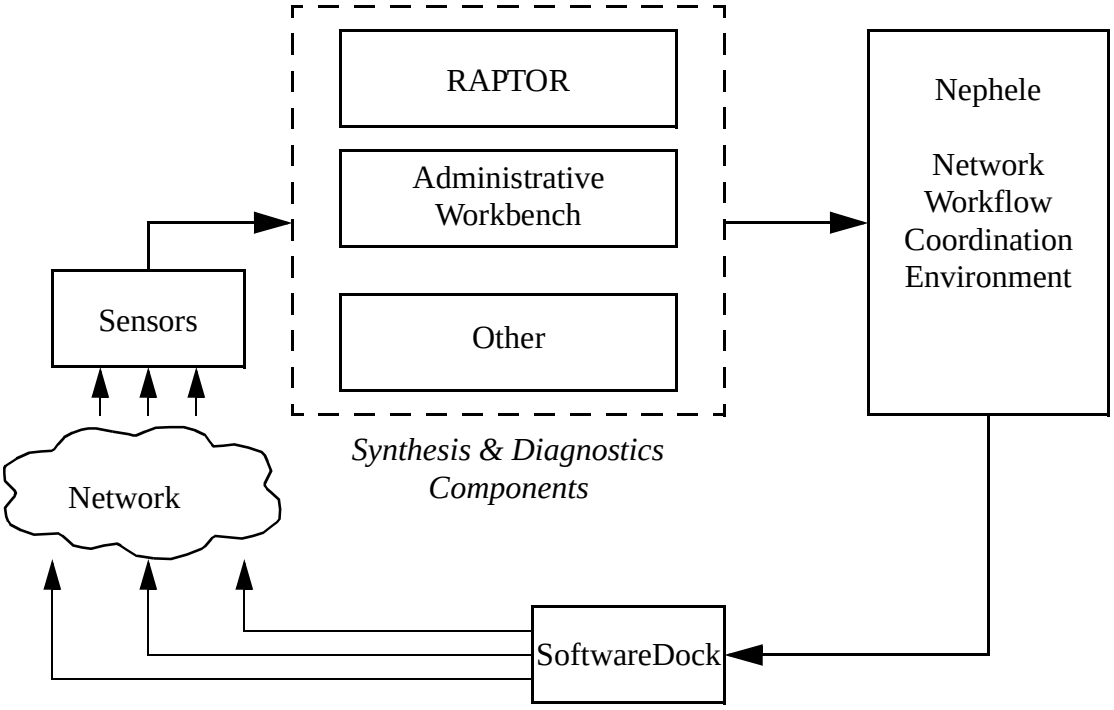

Fig. 1. The Willow architecture.

The control loop of Figure 1 begins with *sensing* of network state. Sensors can include reports from applications, application heartbeat monitors, intrusion detection alarms, or any other means of measuring actual network properties. From sensing events, independent *diagnosis and synthesis* components build models of network state and determine required network state changes. RAPTOR is a formal-specification driven diagnosis and synthesis system that receives sensor events to analyze and respond with desired network changes, automatically and in bounded time. The Administrative Workbench is an interactive application allowing system administrators to remotely monitor application conditions and adjust application properties. Additional diagnosis and synthesis components can be added by modification of the Willow specification input.

Synthesis components issue their intended network changes as *workflow* requests. Nephele is a large-scale network workflow execution environment. It oversees coordination and arbitrates resource usage between independently synthesized work requests. Different workflows with differing intentions from different diagnosis and synthesis components might conflict, and Nephele maintains ordering of their operation to best meet the survivability goals of the application domain. When workflows are allowed to activate, workflow events are received by the Software Dock and result in local system state changes. The Software Dock infrastructure provides a single interface for *universal actuation* at application nodes across enterprise level networks [3, 4, 5]. Actuation completes the control loop cycle.

All of the components of the Willow architecture interact via the Siena *publish-subscribe* communication system [1]. This allows efficient, scalable event-driven communication to Willow components throughout large-scale networks. In turn, the components of Willow provide efficient, scalable, well-defined, proactive and reactive network change capabilities. This enhances network application survivability, security, and manageability.

Summary of a Case Study

A prototype Willow system has been developed that implements all the different aspects of the architecture mentioned above. As part of an on-going feasibility study, this prototype implementation has been used to execute a prototype implementation of the *Joint Battlespace Infosphere* (JBI) concept.

Instantiations of the JBI concept, once fully developed, will provide advanced information systems for military use. At the heart of the JBI concept is the notion of publish/subscribe semantics in which different information sources publish their data to a network and those interested in the information subscribe to those parts which they wish to see. The expectation is that very large amounst of military information will be published to a JBI and large numbers of consumers (commanders at all levels) will tailor the information they receive by subscribing appropriately.

Clearly a JBI will be an attractive target to an adversary for many reasons. Such a system might be attacked in various ways by hackers or disabled by battle damage, physical terrorism, software faults, and so on. It is essential, therefore, that a JBI be survivable, and the Willow architecture is a candidate implementation platform.

We have developed a JBI implementation based on the Siena publish/subscribe system that includes several information-processing modules (known as fuselets) and synthetic publishers and subscribers. All of the components of the system have been enhanced to allow them to respond appropriately to reconfigration actions. In addition, the different elements of the implementation have been extended deliberately with vulnerabilities so as to permit demonstration and evaluation of the reconfiguration capabilities of Willow.

The JBI implementation operating on the Willow architecture has been subjected to a preliminary evaluation by fault injection. The JBI system has been deployed to a test network entirely automatically and shown to adopt new postures under operator control as desired. The system has also been shown to reconfigure automatically when network faults were injected.

References

[1] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf, Design and Evaluation of a Wide-Area Event Notification Service, ACM Transactions on Computer Systems, 19(3):332-383, Aug 2001.

[2] P. Devanbu, M. Gertz, C. Martel, and S. Stubblebine. Authentic Third-Party Data Publication. In Proceedings of the Fourteenth IFIP Working Conference on Database Security, August 2000.

[3] R.S. Hall, D.M. Heimbigner, A. van der Hoek, and A.L. Wolf. An Architecture for Post-Development Configuration Management in a Wide-Area Network. In Proceedings of the 1997 International Conference on Distributed Computing Systems, pages 269–278. IEEE Computer Soci-ety, May 1997.

[4] R.S. Hall, D.M. Heimbigner, and A.L. Wolf. Evaluating Software Deployment Languages and Schema. In Proceedings of the 1998 International Conference on Software Maintenance, pages 177– 185. IEEE Computer Society, November 1998.

[5] R.S. Hall, D.M. Heimbigner, and A.L. Wolf. A Cooperative Approach to Support Software Deployment Using the Software Dock. In Proceedings of the 1999 International Conference on Software Engineering, pages 174–183. Association for Computer Machinery, May 1999.

[6] J.C. Knight, K. Sullivan, M.C. Elder, and C. Wang. Survivability Architectures: Issues and Approaches. In Proceedings of the DARPA Information Survivability Conference and Exposition, pages 157–171, Los Alamitos, California, January 2000. IEEE Computer Society Press.

[7] J.C. Knight, M. C. Elder, Fault Tolerant Distributed Information Systems, International Symposium on Software Reliability Engineering, Hong Kong (November 2001).

[8] J.C. Knight and K.J. Sullivan, On the Definition of Survivability, University of Virginia, Department of Computer Science, Technical Report CS-TR-33-00