The progress achieved over the final year of the Saguaro distributed operating system project is presented. The primary achievements were in related research, including SR distributed programming language, the MLP system for constructing distributed mixed-language programs, the Psync interprocess communication mechanism, a configurable operating system kernel called the x-kernel, and the development of language mechanisms for performing failure handling in distributed programming languages.
Final Technical Report
The Saguaro Distributed Operating System

Grant Number: AFOSR-88-0147
Grant Duration: March 11, 1988 - February 28, 1989
Awarded to: The University of Arizona
Tucson, Arizona 85721
Principal Investigators: Gregory R. Andrews
Richard D. Schlichting
Department of Computer Science

AFOSR Program Manager: Dr. Abraham Waksman
Directorate on Mathematical and Information Sciences

Summary
The progress achieved over the final year of the Saguaro distributed operating system project is presented. The primary achievements were in related research, including the SR distributed programming language, the MLP system for constructing distributed mixed-language programs, the Psync interprocess communication mechanism, a configurable operating system kernel called the x-kernel, and the development of language mechanisms for performing failure handling in distributed programming languages.
Research Accomplishments

In the past year, we have made significant progress on work related to the Saguaro distributed operating system. This projects include the SR distributed programming language, the MLP system for constructing distributed mixed-language programs, the Psync interprocess communication mechanism, a configurable operating system kernel called the x-kernel, and the design of language mechanisms for handling failures in distributed programs.

The first project is involved with further development of the SR distributed programming language, the language used to implement Saguaro. Over the past several years, a large amount of effort was expended revising SR and implementing a compiler for the new version. The main language constructs are still resources and operations: resources encapsulate processes and variables they share, operations provide the primary mechanism for process interaction. One way in which SR has changed is that both resources and processes are now created dynamically. Another change is that the mechanisms for operation invocation—call and send—and operation implementation—proc and in—have been extended and integrated. Consequently, all of local and remote procedure call, rendezvous, dynamic process creation, asynchronous message passing, multicast, and semaphores are supported. We have found this flexibility to be very useful for distributed programming. The language has also been refined in numerous additional ways to provide additional flexibility. Moreover, by basing SR on a small number of well-integrated concepts, the language is also relatively simple and has a reasonably efficient implementation.

The compiler and runtime support currently execute on top of UNIX; they have been in use since November 1985. Since that time, numerous enhancements to the language have been made, mainly in response to comments from users. SR now runs on both Vaxes and Suns. The implementation was released into the public domain in 1988 and there are currently over 50 sites. Two papers on SR-related topics authored by Stella Atkins, who was a visiting professor at Arizona during Spring 1986, recently appeared [1, 2].

A second Saguaro-related project is the MLP system for constructing distributed, mixed language programs. Specifically, the system is a simple RPC (remote procedure call) system that allows each procedure in a sequential program to be written in a different programming language and executed on any machine in a network. MLP is based on a type scheme called UTS (the Universal Type System), which is composed of a data representation standard and a type expression language. The type expression language is used as an interface definition language in a manner similar to the use of Courier in Xerox systems; the data representation standard specifies the format of data transferred as arguments or results between procedures in a machine- and language-independent fashion. The initial version of the system supports C, Pascal, and Icon; it has been in use since November 1986 on an interconnected collection of Vaxes and Suns running Berkeley UNIX. Work has currently underway on a second version of the system.

A description of the implementation of MLP and our experience in using the system appears in [3]. The way in which the object-oriented distributed programming language Emerald is being integrated into the second version of the system appears in [4]. A dissertation that describes how UTS has been used in both MLP and Saguaro is also nearing completion [5].
Another line of research has involved two separate projects investigating aspects of interprocess communication. The first is Psync, an interprocess communication mechanism that supports a new abstraction through which a group of processes can exchange messages. The novel aspect of Psync is that it explicitly embeds timing information drawn from the distributed computation's logical clock. We have recently implemented and experimented with a prototype version of Psync. The prototype demonstrates that explicitly preserving this partial ordering in the communications subsystem can be implemented on an unreliable communications network at little cost. Also, the abstraction provided by Psync is general enough to support efficient and elegant implementations of a wide spectrum of communication paradigms.

The second project is the x-kernel, a configurable operating system kernel designed to support experimentation in interprocess communication and distributed programming. The x-kernel's underlying architecture provides a rich set of abstractions that are used to construct and compose communication protocols. The architecture is interesting because the these abstractions are both general enough to accommodate a wide range of protocols and efficient enough to provide a useful testbed in which protocol performance can be accurately measured.

A comprehensive paper describing Psync and its implementation has been accepted for publication [6]. The design of the x-kernel is documented in [7], and a paper outlining the system and its implementation is pending [8]. Two other papers relating aspects of the x-kernel have also been written [9, 10].

The final line of investigation has been developing language mechanisms for handling failures. One property that makes failures difficult to handle in such systems is that they can occur concurrently with other system events. We have been investigating an approach for writing fault-tolerant distributed programs that can cope with such asynchrony in a systematic manner. The basic idea is to treat failures as just another class of events that are handled similarly to normal system events. Linguistic constructs that can be added to distributed programming languages with minimal impact are then proposed to handle such failure events. To make our ideas precise, we are using the SR distributed programming language as a basis for incorporating these constructs. A paper outlining our approach is scheduled to appear in August [11].

Finally, a paper appeared that surveys several of these projects in distributed languages and systems, and offers observations based on the experience gained during their design, implementation, and use [12]. The relevant projects are the SR distributed programming language, the Saguaro distributed operating system, the MLP system for constructing distributed mixed-language programs, the object-based distributed programming language Emerald, and the Psync interprocess communication mechanism. The observations address the experimentation process itself as well as the design of distributed software.
References


Participating Professionals

Gregory R. Andrews, Professor and Department Head, Dept. of Computer Science, University of Arizona

Richard D. Schlichting, Associate Professor, Dept. of Computer Science, University of Arizona

Norman C. Hutchinson, Assistant Professor, Dept. of Computer Science, University of Arizona

Larry L. Peterson, Assistant Professor, Dept. of Computer Science, University of Arizona

Ronald A. Olsson, Assistant Professor, Division of Computer Science, University of California, Davis (Ph.D. earned at Arizona, 1986)

Titus Purdin, Assistant Professor, Dept. of Computer Science, Colorado State University (Ph.D. earned at Arizona, 1987)

Stella Atkins, Assistant Professor, Dept. of Computer Science, Simon Fraser University (visiting Assistant Professor, 1986)

Steve W. Manweiler, Member, Technical Staff, Hewlett-Packard Corp., Ft. Collins, Colorado (M.S. earned at Arizona, 1986)

Mark Abbott, Ph.D. student, Dept. of Computer Science, University of Arizona

Nick Buchholz, Ph.D. student, Dept. of Computer Science, University of Arizona

Mike Coffin, Ph.D. student, Dept. of Computer Science, University of Arizona

Irv Elshoff, Ph.D. student, Dept. of Computer Science, University of Arizona

Roger Hayes, Ph.D. student, Dept. of Computer Science, University of Arizona

Shivakant Mishra, Ph.D. student, Dept. of Computer Science, University of Arizona

Sean O'Malley, Ph.D. student, Dept. of Computer Science, University of Arizona

Vicraj Thomas, Ph.D. student, Dept. of Computer Science, University of Arizona