A Trusted, Scalable, Real-Time Operating System Environment

Douglas M. Wells
Open Software Foundation Research Institute
1 Cambridge Center, Cambridge, MA 02142
Telephone: +1 617 621 7366
Internet: dmw@osf.org

Abstract[1]

The Open Software Foundation (OSF) Research Institute (RI) is developing a series of operating systems prototypes with the purpose of extending the capabilities of commercial operating systems. The initial goal is to develop a trusted, scalable, real-time operating system environment. Some projects are complete and have been incorporated into commercial operating systems. Current projects focus on performance, scalability, trust, and real-time.

Introduction

The OSF Research Institute (RI) is currently focused on accelerating open systems by meeting the operating system needs of High Performance Computing (HPC). Its overall goal is to develop a new generation of systems technology that permits the flexible addition of system level services, thereby producing operating systems that are far more extensible and customizable than today's standard technology. Its immediate goal is to demonstrate the scalability of OSF's Mach-based operating system (OSF/1) to high end machines, thus establishing the viability of the microkernel approach towards meeting the requirements for real-time, security, extensibility, reliability and distribution in commercial products.

By working in an open fashion with researchers, industrial, and government partners, the Research Institute synthesizes the work of others with developments at OSF into full-function prototypes. These prototypes provide a technology base for industrial products and a foundation for further research. The Research Institute succeeds when the advanced technology developed at OSF and other sites are collectively brought to bear in state of the art commercial products. For several years, the RI has been focused on the decomposition of OSF/1 into a modular form to realize the benefits noted above.

1. This work was supported in part by the Advanced Research Projects Agency and the Rome Laboratory of the Air Force Materiel Command.

This paper summarized the history and current status of our overall operating system program, including brief descriptions of the major projects currently underway within the RI. It then describes the real-time program in more detail, including some of the collaborations underway with industrial and government partners.

Operating System Program

The RI methodology emphasizes the development of full-function prototypes. By synthesizing the ideas and code contributed by universities and leading edge companies with our own research into complete systems, the RI is able to reveal the effects of various architectural changes. Often multiple concurrent development efforts are underway, each project focusing on one aspect of the overall system objectives. Once complete, successful projects are integrated into a common system source base so that each successive release brings us closer to the targeted system environment.

Individual development projects may be structured differently. Some efforts include only microkernel modifications. A few involve only server enhancements. Many require coordinated changes to both server and microkernel components.

Objectives

In addition to general objectives of portability, scalability, and interoperability, the particular objectives of the current OSF RI Operating System Program are to develop:

- a more maintainable, more extensible systems technology foundation that enables rapid innovation
- a high trust configuration that provides much higher assurance

- local and distributed real-time capabilities that enable both desktop and mission-critical applications
- a modular framework that enables support for multiple personalities, such as POSIX, OS/2, and specialized network servers
- a scalable foundation that provides support for uniprocessors, SMP, MPP, and clusters

Eventually, all of these objectives are to be met in one comprehensive system. Currently, each of the projects under way target only a subset of the objectives, though every objective is targeted by at least one ongoing project.

Evolution of the AD System-

AD is an "Advanced Development" version of OSF/1 for massively parallel supercomputers. It is the culmination of a series of developments that result in a balanced approach to enhancing high-end and distributed systems. Maintaining full compatibility with OSF/1 to preserve portability and interoperability, the system brings to bear innovative developments in OSF Mach and OSF/1 services to meet the need for a scalable, single system image operating system spanning thousands of nodes.

Advances in system capabilities have occurred by one project building on the successful results of another. The MK system grew out of the standard OSF/1; AD evolved from an earlier version of MK. Figure 1 shows a simplified version of the past and near term releases of RI operating system components.

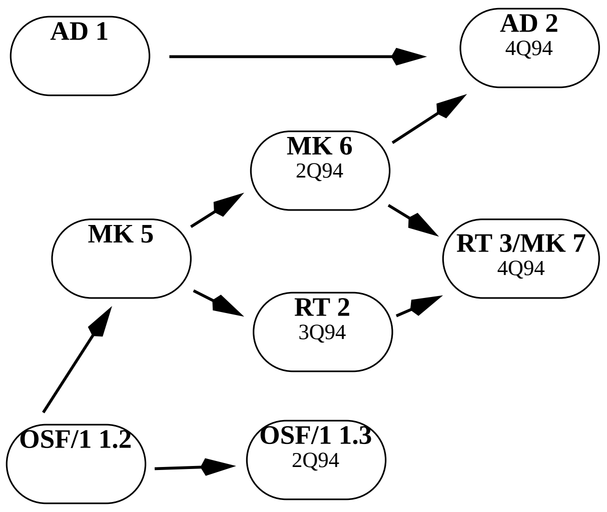

FIGURE 1. Evolution of RI OS Releases.

OSF/1

OSF/1 is a commercial-quality, Mach-based version of UNIX providing compliance with SVID, POSIX, XPG3, and numerous other standards. Release 1.0 of OSF/1 forms the basis of operating system offerings from several large computer vendors. In addition, OSF/1 provides a base system upon which other OSF technologies, such as the Distributed Computing Environment (DCE) and the Distributed Management Environment (DME) are being integrated. Thus, OSF/1 provides a fertile environment in which to explore microkernel-based systems.

MK

MK[2] is a rearchitected version of OSF/1 in which the internal implementation has been decomposed into a user space server and an OSF Mach kernel[11]. This decomposition is based on the Mach 3.0 architecture, as developed at Carnegie Mellon University[1]. MK has shown that it is possible to provide a separate server that is 100% binary compatible with the standard OSF/1 integrated kernel, with only a small performance penalty.

MK inherits the virtues of compatibility and portability from OSF/1. The server code is taken directly from the integrated kernel code base with over 95% code reuse. This result is critical to retaining the compliance and quality of the OSF/1 operating system in the server-based version.

In addition, MK provides full source and binary compatibility for OSF/1 programs. UNIX system calls are typically implemented using a machine-specific trap instruction. (See Figure 2.) The microkernel considers

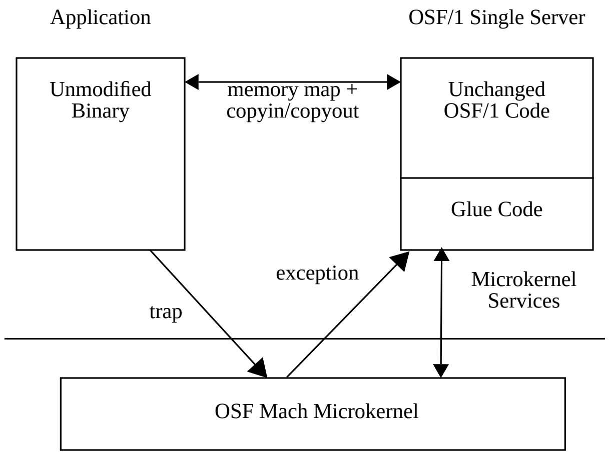

FIGURE 2. MK Single Server Architecture.

execution of these trap instructions to be exception conditions, in the same manner as a page fault, or an floating point overflow would cause an exception. The microkernel packages the machine state, including the processor registers, and sends the package as an exception message to the appropriate fault handler. The OSF/1 server, which previously established itself as the exception handler for the task, receives the message. The “Microkernel Glue Code” massages the data to look like a normal hardware trap frame and then calls the standard trap handler in the unchanged portion of the OSF/1 server. Implementation of the standard UNIX functions *copyin* and *copyout* is also performed by the glue code, which uses either memory mapping techniques or Mach VM read/write functions to access the data in the user process. When the system call function completes and the system call returns, the glue code regains control, packages up the result registers to return to the Mach microkernel, which then returns control to the UNIX user process after updating the processor register set.

MK 5.0 was released in April, 1993, and is available to OSF/1 licensees.

AD 1

OSF/1 employs UNIX technology, which was originally designed to operate on uniprocessor minicomputers. Although extended to include multiprocessing capabilities, the structures and algorithms have been selected to be efficient on system architectures with small numbers of processors sharing a common physical memory with coherency supplied by hardware.

Tomorrow’s (and a growing number of today’s) supercomputers, however, will be massively parallel, consisting of hundreds of thousands of independent nodes, interconnected via high-speed, sparsely-connected message passing networks. The allocation and management of resources will be a much bigger task for tomorrow’s operating systems. They will be responsible for efficiently controlling the entire system, providing the illusion of one single, cohesive resource set. Memory coherency will also be the responsibility of the operating system. To achieve the desired performance, the operating system services themselves must scale as the resource set grows.

AD, which incorporates the complete OSF/1 API into a single system image over a loosely coupled multicomputer, is the latest advance towards providing that ideal system. Users, programmers, and system administrators all share a single, coherent view of the use and control of all resources throughout the system.

Running atop an extended Mach kernel[3], which provides a base level of inter-node message delivery, the MK server has been decomposed into multiple services, each of which can run on separate nodes. Use of multiple I/O servers, each running simultaneously on separate nodes is necessary to provide the scalability required to satisfy the I/O throughput needs of hundreds or thousands of application nodes.

In AD the nodes of the multicomputer are divided into two classes: compute nodes on which applications are executed, and service nodes. (Note that some nodes can serve both functions.) Each server provides operating system services to all other nodes in a coordinated fashion in order to supported the single system image model. (See Figure 3.)

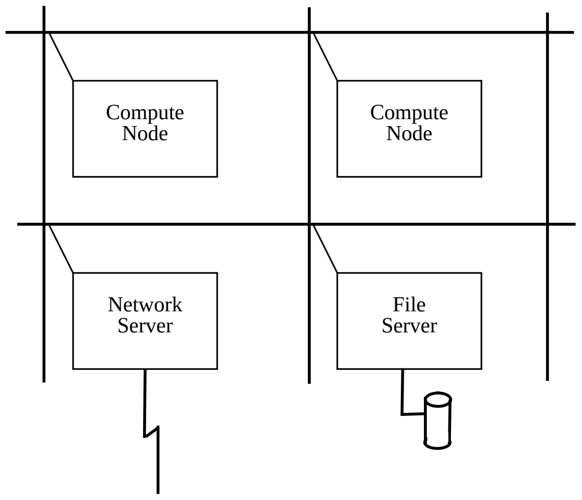

FIGURE 3. AD: Multinode Architecture.

Process Management provides the most serious technical challenge to providing a distributed operating system free of bottlenecks as the size of the system increases. UNIX semantics require coordinated manipulation of control based on a process hierarchy that is continually changing: process identifiers much be unique throughout the entire system; process terminations and reliable signal delivery depend on subtleties of parent-child interactions. In addition, process migration and dynamic load levelling are useful to avoid “hot-spots” within the system.

AD currently takes a practical but limited approach to the problem. One node is designated as the process manager server. This server maintains the global process ID space and carries out the process management functions, such as *fork*, exec, and *signal* delivery. The most compute intensive requests, *fork* and *exec*, have been optimized in the following way: the process database entries are updated on the process manager server, but the actual manipulation of the target node's address space, the loading of a new image, etc., are delegated to a processor manager assistant running on each node.

A parallel application starts by *fork*-ing and *exec*-ing many instances of itself on a partition of nodes. The process manager assistant on each node allows the loading of the image and address manipulation to occur in parallel. As the utilization of the process manager server increases, the process manager becomes a bottleneck. If the supercomputer application consists of a few long running applications (as is typical today), the current implementation should scale to a large number of processors. If the workload consists of many short applications, as is typical during program development, the bottleneck will be reached sooner.[2]

AD 1.0 was released in September, 1992, and is available to OSF/1 licensees. Recent enhancements have focused on performance and robustness.

Commercial Development

OSF Engineering has adopted MK as the basis for its next major release of OSF/1. OSF/1 1.3, which will be available in summer 1994, incorporates the microkernel-based architecture of MK into the standard OSF operating system technology offering. MK is also the basis of the POSIX component of IBM's recently announced Workplace OS.

Intel Supercomputer Systems Division (SSD) has selected AD as the basis for the operating system for its new Paragon supercomputer. This innovative multicomputer can contain up to a few thousands of i860-based nodes interconnected via a high speed message routing network. Paragon OSF/1, the commercial operating system offered by Intel, includes added value in the form of scalable process management, process migration, and dynamic run-time load leveling.

Convex has selected AD for its Exemplar series of supercomputers. Other supercomputer vendors have also selected AD for as yet unannounced products. In addition, a number of other selection evaluations are ongoing.

2. Transparent Network Computing (TNC) is an alternate approach that distributes all the process management functions. TNC is a product of Locus Computing and replaces the AD process management facility in Intel's Paragon OSF/1 version of the system[15].

Current Activities

Current operating system projects at the RI target enhancement of the current microkernel and server base, both for function and applicability to commercial use.

Performance

Successive versions of MK provided increasing degrees of compatibility and compliance with standards, culminating in MK 5, the most recent release, which provides 100% binary compatibility with the OSF/1 integrated kernel at the API. Compatibility requires compliance with SVID, POSIX, and XPG4 standards, however, and providing this correctness exacted a toll on performance. Although some benchmarks actually executed faster on the microkernel-based version, AIM III, a commonly used benchmark to measure throughput on UNIX systems, identified a significant performance degradation from the OSF/1 integrated kernel.

Although there are myriad reasons why a microkernel-based system offers benefits in the areas of maintenance and innovation, this level of performance degradation would not be acceptable to commercial vendors of UNIX-based desktop workstations and servers. A special project was formed to address the issue of performance. Performance tuning work resulting from this project has reduced the performance gap between MK and the integrated kernel from approximately 30% loss[12] to less than 5%[3]

The major effort in this area has been the development of server collocation[4]. (See Figure 4.) Investigations into the sources of performance loss have shown that the primary cause is associated with the context switch from one task to another one. Fabrication of messages, the invalidation of protection caches, and the establishment of an execution environment in the operating system server all add overhead that is not present in traditional kernel architectures. In addition, many processor architectures include special privileged instructions for accessing data in user space, as in required by the UNIX system function *copyin*, which is used to implement the system calls *read* and *write*, for instance. These instructions are unusable by the server component of a microkernel-based operating system when executing in a separate task.

3. These performance measurements have been taken using the AIMIII benchmark with 8 users. AIMIII is widely used as a benchmark for UNIX systems. In addition, of all of the commonly used benchmarks, AIMIII indicated the largest performance loss between the OSF/1 integrated kernel and the MK microkernel-based system.

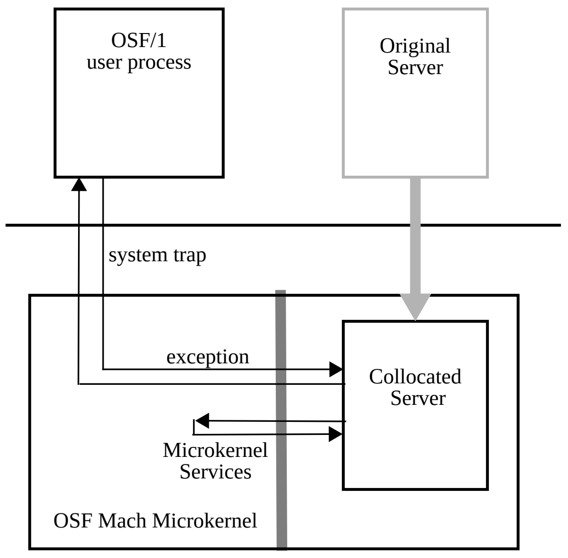

FIGURE 4. Server Collocation into Mach Kernel.

The collocation mechanism allows one server to be dynamically loaded into, or collocated with, another server or the OSF Mach microkernel. Ancillary tools allow use of (machine-dependent) hardware optimization operations and the reduction of much of the cross-domain overhead function. Concurrent enhancements to the server support tools, such as the Mach cthreads multitasking package, allow the identical server image to execute either collocated or as a traditional separate server task.

Using server collocation, optimization of common paths, and other of various techniques, the performance group has reduced the performance difference to less than 4%. MK 6 will include the performance enhancements developed by this project. It is scheduled to be released in early summer, 1994.

AD 2

The initial release of AD has been highly successful. Several companies have adopted it as the basis of the operating system for their supercomputer offerings. Originally developed on a test-bed with 16 nodes, interim development efforts allow scalability to a few thousands of nodes. Lessons learned from this effort have identified significant bottlenecks that appear with larger systems. In addition, AD 1 was adapted from an earlier version of MK, one that was not totally compliant with all of the UNIX standards[4].

The goal of AD 2 is to advance operating system technology to a new plateau for a wider range of architectures and computational models. It will address a more ambition vision of scalability, for both numbers of nodes and numbers of processors per node. It will be extensible, allowing a better distribution of services and an expanded set of attributes. It will be more configurable, allowing replacement of various components, such as the coherency model for distributed shared memory, in order to better adapt to a wider variety of architectures. Finally, it will incorporate the results of other concurrent activities at the RI, such as the MK 5 server, the performance project, and the real-time project.

AD 2 includes a new distributed file system and framework, distributed process management, and scalable networking services. In addition, the Mach Distributed IPC mechanism is being rewritten. AD 2 is scheduled to be available at the end of calendar year 1994.[5]

Trust

The RI is developing a high assurance configuration of OSF Mach. This microkernel, which is targeted at the B3 and F5/E5 levels of trust, is being redesigned as a strictly layered system. Retaining compatibility at both the source and binary levels with MK 5, this new implementation takes advantage of object-oriented development techniques in order to create a highly modular, easily understandable release of OSF Mach.

This version of the OSF Mach microkernel is being developed in cooperation with Trusted Information Systems (TIS), which is developing TMach. TMach is a highly secure and trusted version of the Mach operating system. In addition to using the high assurance OSF Mach microkernel as the security kernel, TMach includes a number of other servers within the Trusted Computing Base (TCB), including simple file system, device, authentication, and administration services. The MK server is included to provide binary compatibility with other OSF/1 systems, but executes outside the TCB as untrusted code in order to reduce the size of the TCB.

4. Most of the compliance anomalies are exposed only in the presence of application errors, such as passing inaccessible system call parameters, and in the area of UNIX signal processing.

5. Several technical papers on AD 2 are available from the OSF Research Institute. These papers cover the file system architecture, process management architecture, Distributed IPC, and Scalable Networking. Contact OSF RI Publications for details.

TMach is in evaluation by both U.S. and European agencies. Both TMach and the RI high assurance version of OSF Mach are expected to be available in mid-1995.

Real-Time Project

The goal of the real-time project at the RI is to advance system technology by incorporating advanced real-time capabilities into the standards-compliant operating system components being developed at the RI. Some of these system technologies are derived from the research results of collaborators. Some are developed within the RI. All of the development includes the synthesis and enhancements into a coherent system consistent with the overall RI philosophy.

The current development projects are targeted toward microkernel mechanisms. There are several reasons for this orientation. First, a real-time server is useless when used with a non-real-time microkernel. Second, the Mach microkernel is significantly smaller and less complex than the OSF/1 server, allowing us to develop a complete, full-function real-time microkernel utilizing only our limited resources. Also, many real-time applications are already designed to operate using a very limited set of system primitives, such as the System Programming Interface (SPI) provided by the OSF Mach microkernel.

RT 1

The results of the real-time project are being released in a set of releases with ever increasing real-time capabilities. The first real-time release was targeted at desktop multimedia systems, which are characteristically soft-deadline system with moderate latency requirements (e.g., unbounded and less than 5 msec. on an i486 system). This set of changes consisted of enhancements to the Mach clocks and timers facility, and improvements to the existing scheduling mechanism. Each modification was designed to be compatible with the POSIX real-time standard (1003.4), which was under development at that time. These changes were originally released as RT 1 in November, 1992. Subsequently, these enhancements were incorporated into the standard release of MK 5 in April, 1993.

RT 2

The set of enhancements currently under development are designed to enhance real-time capabilities within a single node by addressing low latency applications as might be found in simulation and factory automation environments. Utilizing a fully-preemptible OSF Mach microkernel[14], interrupt latencies will be bounded and typically less than 1 msec. In addition, a set of inter-task synchronizers will be included.

The Mach Remote Procedure Call (RPC) mechanism is also being improved. The original Mach IPC facility implemented RPC in the traditional manner: two separate messages were used, one from the caller to the callee with the input parameters, the other from the caller back to the caller with the result values. The IPC redesign included in Mach 3.0 incorporated several optimizations, including a special system call that sends the original message and waits for the reply, and a special “hot path,” which causes the processor to be directly passed from the caller to the callee in certain common paths.

By adopting and enhancing a mechanism developed earlier by the University of Utah[5], we are promoting the RPC concept to first-class status and completely redesigning the RPC facility. Using a technique originally pioneered in the distributed threads of the Alpha Operating System[8], a single computation can encompass concurrent invocations in each of several Mach tasks. Originating in a root thread, a computation can invoke a client server task via an RPC. (See Figure 5.)

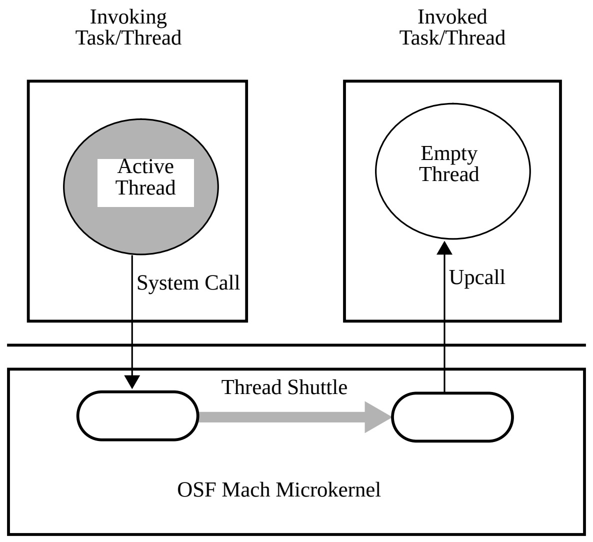

FIGURE 5. Inter-task RPC Invocation.

That client server can then invoke yet another client server via another RPC. When a thread in one task invokes a server, the original thread is suspended until the RPC returns. In this manner, a chain of RPCs in Mach becomes analogous to a chain of ordinary procedure calls in UNIX or other standard operating systems.

The new RPC concept is implemented using a thread shuttle within the OSF Mach kernel, and thread activations in each task. Beginning in an active thread, a computation executing an RPC traps into the kernel, where the associated thread shuttle gains control. The thread shuttle includes all scheduling information, such as priority, for the computation. In addition, there is a kernel stack associated with the thread shuttle. Executing on this kernel stack, the system copies parameters from the invoking task into the kernel. The kernel locates the target thread using the RPC port send right input parameter. The thread shuttle is then detached from the invoking task/thread (leaving it in a suspended condition) and attached to the target thread/task. The kernel then copies the input parameters into the invoked task and performs an upcall into the target thread. The target thread must be an "empty" thread, which is essentially an unused stack and a potential for executing. Upon returning from the RPC, the scenario is reversed: results are copied from the callee into the kernel, the thread shuttle is released from the thread (leaving it in the "empty" state, however) and reattached to the invoking thread, into which the results are copied.

There are several benefits to this design. Perhaps the most important is the elimination of queuing delays for real-time computations. Because an empty thread uses so few resources (essentially just stack space), a real-time computation can create as many empty threads as might be needed. In this case, the RPC never needs to enqueue messages and no scheduling events are encountered during a normal RPC. In addition, the new RPC should be 4 to 5 times faster than the standard Mach 3.0 version. This performance improvement is due primarily to the elimination of the need to create (and destroy) the *send-once* right that used to be needed in order to effect the return from the RPC.

RT 2 is currently under development and is scheduled to be available in summer, 1994.

RT 3

The next release following RT 2 will support real-time applications in a distributed, multi-node environment. Building on the base of RT 2, the new RPC mechanism will be extended to operate across nodes. Frameworks for networking and scheduling will be incorporated. In addition, the normal execution paths for IPC and RPC will be reworked to provide fast, predictable execution paths.

The OSF Mach Scheduling Framework is derived from one originally developed by the Center for High Performance Computing (CHPC) at Worcester Polytechnic Institute[13]. This framework allows two or more scheduling policies to coexist on the same node. Typically, one scheduling policy might be an advanced real-time scheduler, such as rate-monotonic[9] or best-effort[10], and the other would be the standard Mach time-sharing scheduler.

The xKernel[7], developed at the University of Arizona, is an object-oriented framework for the implementation of communication protocols. Primitive building blocks known as *micro-protocols* and *virtual protocols* are configured into structures referred to as *protocol graphs.* (See Figure 6).

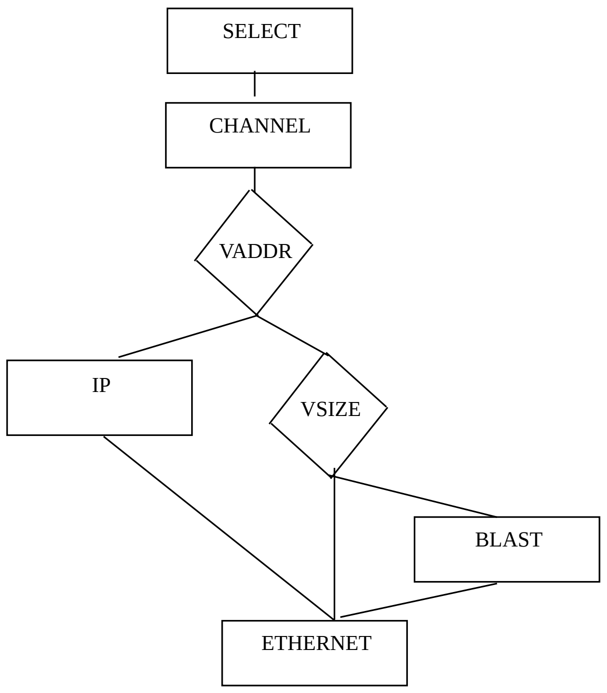

FIGURE 6. An xKernel protocol graph. The rectangles represent micro-protocols, the diamonds represent virtual protocols.

The RI has adopted the xKernel as the core architecture for our Distributed IPC, distributed real-time, and high availability protocols. There are a few major weaknesses in the current version of the xKernel, however, First, the xKernel does not attempt to control resource allocation. Second, the xKernel does not support concurrent execution; only one thread at a time is allowed to shepard a message through the protocol graph. These deficiencies are being addressed by the University of Arizona and OSF RI in a joint development project.

RT 3 is scheduled to be available at the end of calendar year 1994. In addition, this system will form the basis for the subsequent major single-server release of OSF/1, MK 7, which is scheduled for the same time period[6].

Users of the RI Real-Time Technology

Although the RI real-time technology is still undergoing development, several groups are using or intend to use these capabilities in interesting, multicomputer and/or distributed system experiments.

HiPer-D

The real-time enhancements included in RT 1 have been retrofitted into a special version of Paragon OSF/1, which is being used by the Navy's HiPer-D project to investigate use of new computer architectures, such as massively parallel computers, for the next generation Aegis combat system. The three participants, Naval Surface Warfare Center (NSWC), Johns Hopkins University Applied Physics Laboratory (JHU/APL), and Martin Marietta Advanced Technology Laboratory (MM/ATL), are using 20 node Paragon OSF/1 systems to demonstrate use of selected functions within the standard Aegis system. This investigation is expected to continue over the next few years.

Honeywell

Honeywell Space and Strategic Systems is currently developing a ruggedized version of the Intel Paragon system. In this project, known as Embedded Touchstone, Honeywell is creating a high-density, air-cooled system that includes up to 16 nodes in a one-half cubic foot package. This hardware will be matched with a real-time version of Intel's Paragon OSF/1 system and offered for use in military applications.

In one prototypical application, the multicomputer would be configured for sensor processing. (See Figure 7.) Several nodes would be reserved[7] for each system function, such as fusion, clustering, tracking, and decision processing. Additionally, several nodes would be reserved to hold the distributed track file. As sensor data arrived, they would be fed to the fusion nodes, which would process the data (in parallel). Once fusion was complete, the fused data would be passed on to the cluster detectors, which would process the data (in parallel). The results of that phase would be passed onto the tracker, which would in turn pass its results on to the decision processing nodes. Both partially processed data and final results would be stored in the track file and report file nodes. The OSF Mach scheduling framework would be used to provide a system-wide gang scheduler, which would coordinate the scheduling of each individual node in order to effect efficient processing of data through the overall system.

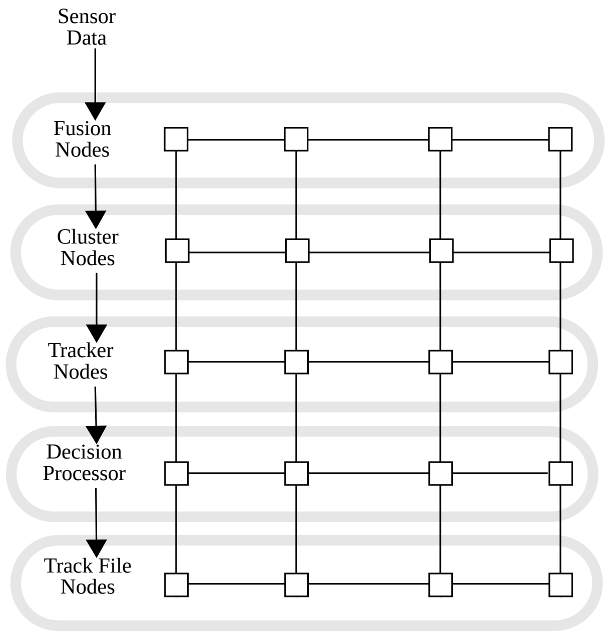

FIGURE 7. Embedded Touchstone Sensor Application

Alpha/Mach Integration

The Alpha/Mach Integration project[6] will build on the basic real-time project to incorporate the capabilities of the Alpha[8] operating system into Mach, resulting in a significantly enhanced system that encompasses real-time, mission-critical, distributed applications as are found in military command and control, and battle management systems, and in industrial factory-automation systems.

Future Development

Although not currently scheduled, it is expected that the capabilities of AD 2 and MK 7 will be merged in the first half of 1995. Later, after the high assurance version of the OSF Mach microkernel is complete, an effort will begin to incorporate the real-time and scalability characteristics of that system into the high assurance version. This integration will require a significant amount of

6. Several technical papers on RT2 and RT3 are available from the OSF Research Institute. These papers cover the preemption mechanism, the RPC facility, the Scheduling and Network Frameworks, and the new Synchronizers. Contact OSF RI Publications for details.

7. The concept of reservation actually includes both space and time, although this concept is ignored in the main text in order to simplify the explanation. For example, a node might be reserved for fusion during the first 20% of each processing cycle. The same node might also be reserved for decision processing during the last 25% of each cycle.

redesign and reimplementation in order not to compromise the high-assurance goals of the trusted kernel.

References

[1] M. Acceta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian, and M. Young, “Mach: A New Foundation for UNIX Development,” *Proceedings of the 1986 Summer USENIX Conference*, Atlanta, GA, 1986.

[2] F. Barbou des Places, P. Bernadat, M. Condict, S. Empereur, J. Febvre, D. George, J. Loveluck, E. McManus, S. Patience, J. Rogado, and P. Roudaud, “Architecture and Benefits of a Multithreaded OSF/1 Server,” *OSF Research Institute Symposium*, OSF, Cambridge, MA, February, 1992.

[3] J.S. Barrera III, “A Fast Mach Network IPC Implementation,” *Proceedings of the Second USENIX Mach Workshop*, Monterey, CA, November, 1991.

[4] M. Condict, “The Server Co-Location Project,” *OSF Research Institute Symposium*, OSF, Cambridge, MA, October, 1993.

[5] B. Ford, and J. Lepreau, “Evolving Mach 3.0 to a Migrating Thread Model,” *Proceedings of the 1994 Winter USENIX Conference*, San Francisco, CA, January, 1994.

[6] I. Goldstein, and D. Wells, “Alpha/Mach Integration Study,” *OSF Research Institute Operating Systems Collected Papers, Vol. 2*, OSF, Cambridge, MA, October, 1993.

[7] N.C. Hutchinson, and L.L. Peterson, “The x-kernel: an Architecture for Implementing Network Protocols”, *IEEE Trans on Software Eng., vol 17, no. 1*, January, 1991.

[8] E.D. Jensen, and J.D. Northcutt, “Alpha: An Open Operating System for Mission-Critical Real-Time Distributed Systems—An Overview”, *Proceedings of the 1989 Workshop on Operating Systems for Mission-Critical Computing*, ACM Press, 1990.

[9] J.P. Lehoczky, L. Sha, and Y. Ding, “The rate-monotonic scheduling algorithm: Exact characterization and average case behavior,” Carnegie Mellon University, Pittsburgh, PA, 1987.

[10] C.D. Locke, Best-Effort Decision Making for Real-Time Scheduling, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, Ph.D. dissertation, May 1986.

[11] K. Loepere, *OSF Mach Final Draft Kernel Interfaces*, Open Software Foundation and Carnegie-Mellon University, Cambridge, MA, 1993.

[12] S. Patience, “Redirecting System Calls in Mach 3.0: An alternative to the emulator,” *Proceedings of the USENIX Mach III Symposium*, Santa Fe, NM, April, 1993.

[13] S. Shipman, M.J. Teller, and N. Paciorek, *Mach/RT Kernel Interfaces (Draft)*, TR92-011, Center for High-Performance Computing, Worcester Polytechnic Institute, Marlboro, MA, 1992.

[14] D. Swartzendruber, “OSF Real-Time Mach: Making the Mach Kernel Preemptible,” *OSF Research Institute Operating Systems Collected Papers, Vol. 2*, OSF, Cambridge, MA, October, 1993.

[15] R. Zajclew, P. Roy, D. Black, C. Peak, P. Guedes, B. Kemp, J. LoVerso, M. Leibensperger, M. Barnett, F. Rabii, D. Netterwala, “An OSF/1 UNIX for Massively Parallel Multicomputers,” *Proceedings of the 1993 Winter USENIX Conference*, San Diego, CA, January, 1993.

Trademarks