Open Microkernel Technology, Key to Evolving Telecommunication Systems and Networks*

Michel Gien and Jean-Bernard Stefani
Chorus Systems and France Telecom
(France)

Abstract

Today's telecommunications companies are facing a serious software crisis. They need to preserve and leverage their enormous software base on rapidly evolving hardware architectures, while extending it as quickly as possible. The computer industry faced a similar problem and solved it with UNIX. However, traditional UNIX by itself does not meet the requirements of most telecommunication applications. In particular, it lacks high performance real-time, transparent distribution for scaleability, and dynamic reconfiguration for high availability. Open microkernel operating systems are key to address telecommunications systems requirements because they contain a high performance, scaleable, distributed, real-time core which can support multiple operating system personalities including UNIX and existing legacy real-time operating systems, and they can scale transparently over a full range of hardware and software configurations. This paper introduces open microkernel technology, as represented by CHORUS, and shows how this technology answers telecommunication systems and distributed processing environments (DPE) requirements. It introduces the main features of a CHORUS-based DPE that does not suffer from the shortcomings of current distributed platform technology when used as a basis for information networks' DPE. A market perspective parallel the impact of open microkernels to the microprocessor revolution of the 80's.

1. Introduction

1.1. Telecommunication systems manufacturers' software crisis

Today's telecommunications systems manufacturers are facing a serious software crisis. They need to preserve and leverage their enormous software base on rapidly evolving hardware architectures, while extending it as quickly as possible. The computer industry faced a similar problem and solved it with UNIX. However, traditional UNIX by itself does not meet the requirements of most telecommunication applications. In particular, it lacks high performance, real-time, transparent distribution for scaleability, and dynamic reconfiguration for high availability.

1.2. Operating system requirements in telecommunications systems

To support their basic real-time needs, system manufacturers created their own simple operating system or used traditional proprietary real-time systems. As a result, today's software infrastructure was created with a limited set of base functions to handle specific tasks.

These operating systems were not architected to:

- support the complexity of services now being demanded;
- allow for upgrading existing code bases with new functionality easily;
- provide a standards-based development environment to maximize engineering resources, thus enhancing time-to-market;
- provide the powerful and stable software architecture required to ensure that the system could evolve smoothly into an unknown future.

* To be published in Proceedings of the TELECOM'95 Conference, Geneva, Switzerland, 3-11 October 1995.

Rapid change and complexity demand dramatic changes from the traditional approach to software. Home-made or traditional real-time operating systems are not capable of handling the level of complexity facing tomorrow's telecommunications software applications.

In order to address these issues, a new operating system architecture is required. Such an architecture must:

- provide the strong and stable architectural framework required to manage the increasingly complex software of the new generation of telecommunications systems;
- provide a "true" UNIX environment to permit easy integration of the large amount of third-party UNIX software readily available on the market;
- enable the re-use of existing telecommunications application software, and easy migration from the proprietary operating system on which they were originally developed;
- allow for development of new application software in a modular framework, with a rich set of industry standard development tools and languages;
- enforce software modularity to drastically reduce complexity of system integration and testing, in order to guarantee quality parameters;
- enable the provision of guarantees of real-time behavior and dependability (including high availability and safety), which are necessary in modern telecommunications systems;
- operate transparently on a wide range of distributed hardware platforms and scale up and down from single processor to parallel multiprocessor architectures;
- provide configuration options that permit the optimization of performance requirements for different sets of communications functions, protocols and topologies.

Such an architecture will allow the telecommunications industry to address users requirements:

- for simplifying the upgrade of systems already installed;
- for complete interoperability between installed and new systems;
- for optimizing their unique network architecture;
- for value-added services;
- for higher availability, including fault tolerance if so desired.

1.3. Open microkernels as an operating system platform for telecommunications systems

Open microkernel operating systems are key to address the requirements identified above. This paper shows how open microkernel technology applies to telecommunication systems and distributed processing environments (DPE). Section 2 introduces the CHORUS open microkernel and discusses its main features in the light of the stated requirements. Section 3 highlights some shortcomings of current distributed platform technology as a basis for information networks' DPE. Section 4 introduces the main features of a CHORUS-based DPE that does not suffer from these shortcomings and that meets a large part of the telecommunications requirements identified above. Section 6 concludes the paper with some market-oriented considerations.

2. Open microkernel operating systems

Open microkernel operating systems provide the architectural base that telecommunications manufacturers need in order to build their new generation of systems and products, as demonstrated by the success of the CHORUS Open Microkernel[1] which is becoming the standard operating system platform in the telecommunications industry.

In particular, new microkernel-based UNIX implementations such as CHORUS/MiX[2] do meet telecommunications systems requirements because they contain a high performance, scaleable, distributed, real-time core (See Fig. 1.). The microkernel can host all or part of a modularized implementation of standard UNIX System V to support off-the-shelf UNIX applications, and run them side by side with hard or soft real-time telecommunications applications. The ability to integrate third-party software from the UNIX market provides a strategic time-to-market advantage, and reduces development costs dramatically. The open microkernel can host multiple personalities, and support legacy environments concurrently with a UNIX personality and real-time and object-oriented system programming interfaces. This makes it easier to re-use pre-existing application software and preserve previous investments while programming new applications in a modern framework. The modularity and openness of microkernel-based UNIX implementation allows telecommunication manufacturers to continue to leverage on their unique in-house system expertise and add specific value within an open system environment. In addition, the microkernel portability insures transparent support of telecommunications applications on a wide variety of hardware architectures, and permits telecommunications manufacturers to take full advantage of the most cost effective hardware at virtually no software cost.

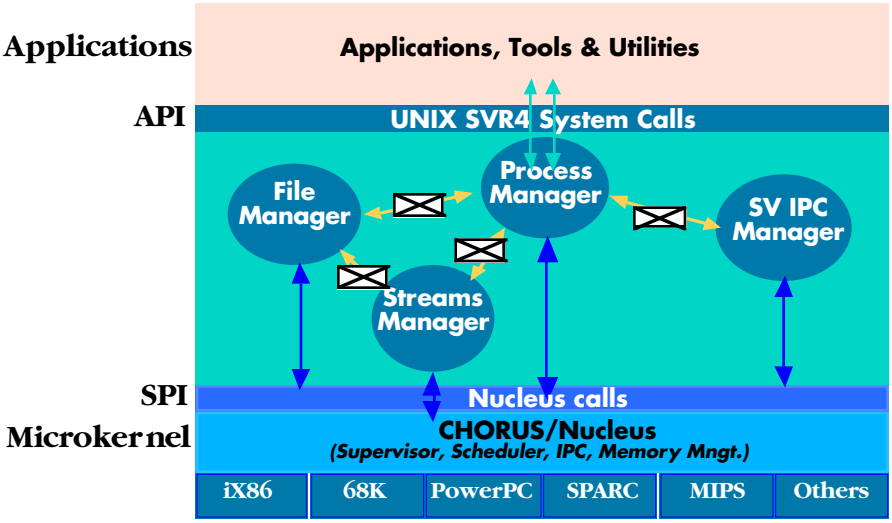

Figure 1. CHORUS/MiX, a distributed implementation of UNIX SVR4

2.1. Modular system software architecture

An open microkernel system architecture like CHORUS, supports distributed system servers and ensures absolute modularity of its components. System software components are isolated into re-usable software modules which can be dynamically loaded to run in the same system address space, in different protected user address spaces, or even transparently distributed over multiple networked processors.

Migrating existing telecommunications software to such an open system software platform (such as routing software and network protocols running in the single address space supported by today's simple real-time executives) is straightforward. New software modules can be developed in user space and interact with system level modules, as those are also gradually modularized and made to run in protected user spaces, to ensure higher system reliability.

The location-transparent Inter-Process Communication (IPC) services provided by an open microkernel support object-oriented encapsulation of software modules, thus guaranteeing simple and safe development, maintenance, enhancement, and replacement of individual software modules and their easy integration into (re)configurable complex software constructions.

This unique characteristic of an open microkernel operating system architecture also constitutes a solid foundation for security-oriented environments, and enables the construction of fault-tolerant systems.

2.2. Transparent connectivity services

Transparent Connectivity Services, provided by the IPC facilities, dramatically improve system configurability. They allow the same complex software configuration to run unchanged as a monolithic piece on a single address space monoprocessor, or as multiple isolated modules distributed over a multi-processor or a multi-computer configuration. Such a key feature will allow a decrease in the number of today's telecommunications software configurations (together with their associated testing maintenance and configuration management problems).

Modules of big software packages (such as switching, protocol stacks, or ATM route servers) communicating by means of the IPC primitives need not reside on the same processor, or even the same computer. This allows the easy introduction of dedicated processors, such as remote interface processors or additional routing processors, to balance the load of the overall system.

Moreover, CHORUS Transparent Connectivity Services allow the configuration and reconfiguration of a complex set of software to be performed dynamically while the system is running (“hot swap”), thus providing a powerful enabling technology for building scaleable and highly available systems.

In a wider perspective, CHORUS Transparent Connectivity Services are designed to support cooperating distributed services. These allow seamless performance scaleability, and functional enhancements of very complex software systems, integrating components from different generations of products, or even different manufacturers.

2.3. Market standard open systems interfaces

An open microkernel architecture supports a variety of operating systems implementing de-facto open systems market standards, both at the application level and within the operating system itself.

In the case of CHORUS, it supports the following open operating systems (see Fig. 2.):

– UNIX (UnixWare and SCO UNIX)
– POSIX-RT standard real-time and multi-thread extensions
– OMG/CORBA object-oriented services

These interfaces allow users to run binary off-the-shelf UNIX applications and to develop re-usable new real-time and/or object-oriented applications.

In addition, CHORUS can easily support legacy real-time operating systems, in order to migrate existing software into this new open environment.

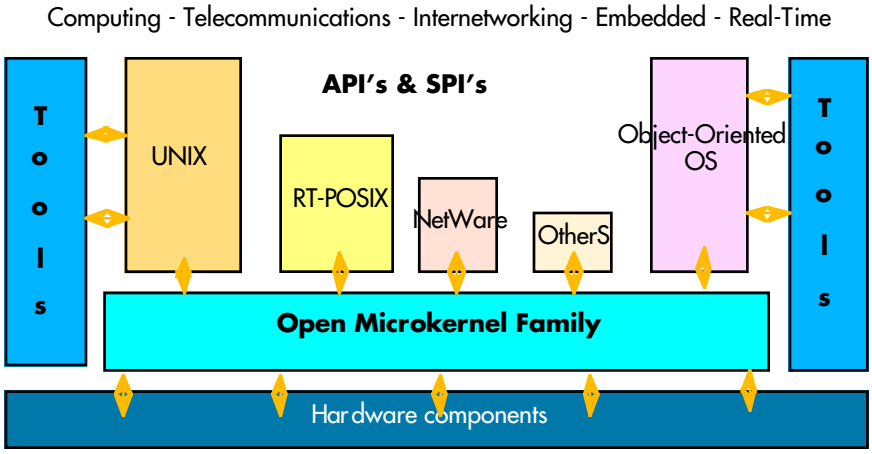

Figure 2. Multiple operating system personalities

2.4. Portability

An open microkernel like CHORUS is totally portable (and actually ported) on all categories of processors as well as boards, computers, complex machine and network architectures. An open microkernel supports simple dedicated microprocessors, often with no virtual memory management. At the same time it supports high-end processors with sophisticated virtual memory, large address spaces, and even built-in communications functions. It also supports mono- as well as multi-processor configurations, with a shared memory, non-uniform memory or distributed memory architecture, and loosely coupled and parallel architectures for which it takes advantage of modern buses or interconnection networks.

2.5. Real-time characteristics

An open microkernel like CHORUS ensures deterministic response times to demanding time-critical applications. It guarantees low latency interrupt handling and context switching.

2.5.1. Multiple scheduling classes

In the case of CHORUS, it supports user-defined scheduling classes to fit precisely with the specific and sometimes conflicting requirements of the various components of complex telecommunications software.

For example, one can define and use concurrently:

– event-driven scheduling to allow threads to process signals produced by a device (such as digital audio samples) deterministically and with a very short delay;
– deadline-based scheduling to schedule threads that need to run to completion within a given time interval (longer than in the previous case);
– isochronous scheduling to guarantee that threads in this class receive a given share of the processor cycles at each given period of elapsed time, as required by multimedia processing;
– time-sharing scheduling for less time-critical threads.

2.5.2. Real-time services available to user-level processes

Real-time services, available to low-level system processes as sometimes found in traditional real-time executives, are exported to user-level processes. This allows time-critical tasks to be run in protected address

spaces, thus keeping the system functioning in the event one of its real-time components fails.

In addition, CHORUS provides a multi-thread programming interface with associated synchronization primitives to both system and user processes. This allows applications to take full advantage of the multi-processing capabilities of the underlying hardware, when available.

2.6. Object-oriented programming services

The CHORUS open microkernel supports the Object Management Group (OMG) Interface Definition Language (IDL) standard. This IDL can be used to describe the interactions between application servers, independently of their location. It provides simultaneously a very clean and powerful design and programming environment for low-level system software, and allows the microkernel to further enhance the performance of Inter-Process Communication services.

In addition, a complete object-oriented operating system (CHORUS/COOL[3]) has been developed to support the full range of OMG CORBA compliant services for transparently distributed object-oriented applications. This microkernel-based implementation of the OMG services interoperates with similar services residing on top of other operating systems.

2.7. Example of a microkernel-based telecommunications system

CHORUS open microkernel architecture allows users to plug-and-play a set of real-time and UNIX applications, and combine all or part of them into a range of configurations.

This is illustrated in Fig. 3 and Fig. 4.

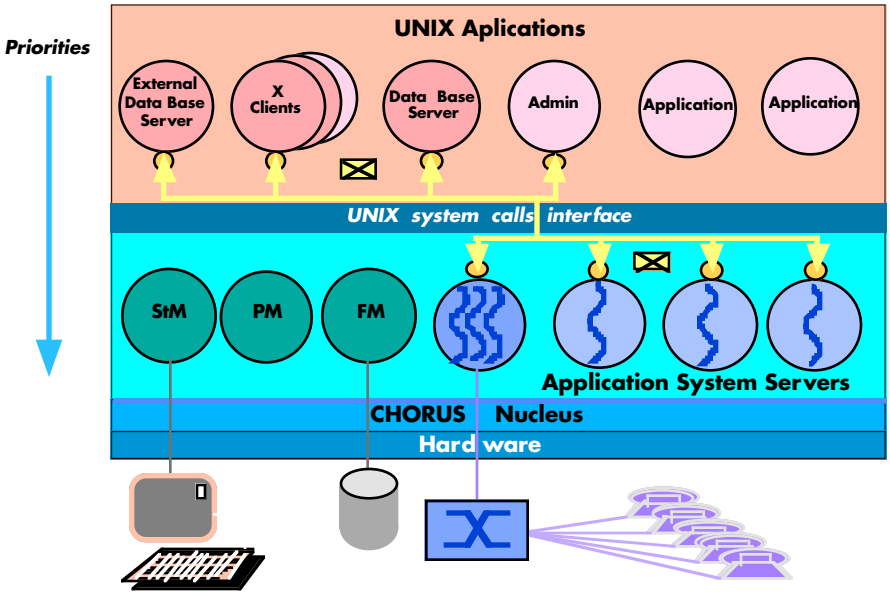

Figure 3. Real-time applications and UNIX applications can all be assembled on the same processor

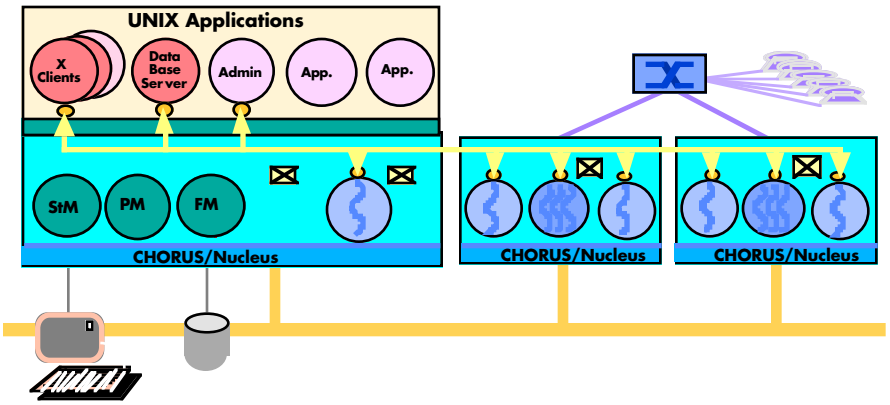

Figure 4. Real-time applications and UNIX applications can be distributed transparently among several processors

3. Towards an information network DPE

Through activities such as the Telecommunications Information Networking Architecture Consortium (TINA-C), both computer and telecommunications vendors are looking at a distributed software architecture for the control and management of future telecommunication services and networks. They are settling on the ISO/ITU-T Reference Model for Open Distributed Processing (ODP) as a framework, and on products based on the Object Management Group (OMG) standards as technology for the "Distributed Processing Environment (DPE)" to support interactive, multi-media services.

Currently TINA-C has no plans to develop internally a DPE which meets the above requirements, nor a full set of tools to help the mapping of its specifications to a DPE. There is an urgent and recognized need to fill this gap by developing architecture, technology and standards in harness.

There is a clear need to extend the Common Object Request Broker Architecture (CORBA) of OMG to meet the needs of developers of distributed interactive multi-media applications. The emergence of low cost broadband networks is opening a market for such applications, but development is inhibited by the lack of a high level applications platform which can meet both functionality and performance requirements.

The telecommunications industry in particular is looking for an open systems solution to this problem. Deregulation, the pressure to reduce costs, and the need to compete by offering new services are driving the industry to a model where services are implemented outside of network switches, using readily available hardware and software.

CORBA and its associated standards provide a good starting point, but needs extending to meet telecommunications needs:

- by adding the specific modularization concepts (i.e. building blocks, packages, objects, contracts) introduced by TINA to cope with software negotiation and actual allocation, installation and management;
- by adding appropriate connection (i.e. binding) and interaction models for multi-media traffic such as voice and video streams;
- by enabling fine grained control over resources and scheduling to meet quality of service and real-time guarantees;
- by enabling encapsulation of alternative protocols (e.g. signaling protocols, management protocols, real-time protocols) within an ORB;
- by extending the support for object life cycle and dynamic extension and re-configuration of DPEs;
- by adopting and extending object services for dependability (transactions, replication, security) and for data management (persistence, queries).

Open real-time microkernel operating system technology, as provided by CHORUS, is key to support implementation of the TINA-C Distributed Processing Environment (DPE). A distributed microkernel-based operating system provides a scaleable, low-overhead framework for the implementation of a TINA DPE. A microkernel-based DPE allows the TINA architecture to be scaled down to elementary network elements or scaled up to nodes with intensive information processing requirements. For example, minimal implementations can be adapted to numerous transport network elements, down to intelligent multiplexers or cross-connects. More comprehensive ones, embedding numerous features and functionality, built as specific subsystems or servers on the open microkernel could be required in computationally

intensive nodes, such as telecommunications operators' information systems.

An important issue in information networks is providing services with definite "Quality of Service (QoS)" guarantees. As a number of distributed telecommunications applications illustrate, integrated real-time support is a primary requirement. Multimedia applications, for instance, need a distributed infrastructure both in terms of temporal quality of service (delays, jitter bounds, throughput) and real-time synchronization (presentation synchronization). Transport network operation and management applications (alarm reporting, real-time monitoring, dynamic configuration management) will also require integrated real-time support. These requirements are not currently being met by open distributed system technology. Consequently, in present telecommunications systems, real-time support is provided using ad-hoc proprietary designs at the expense of openness and integration.

4. A microkernel-based DPE

The microkernel-based TINA DPE being developed by France Telecom[4] consolidates results available from a wide variety of sources, particularly those from standards activities such as the Open Distributed Processing (ODP) Reference Model from ISO and ITU-T, and the Common Object Request Broker Architecture (CORBA) from the Object Management Group. Briefly stated, the microkernel-based DPE supports the ODP object-based computational model, including objects with multiple interfaces, continuous media streams, implicit and explicit binding between interfaces, temporal Quality of Service specifications at interfaces.

Interfaces are described using an extension of the CORBA Interface Definition Language (IDL)[5]. Real-time synchronization is realized on the microkernel-based DPE using reactive objects, i.e. objects programmed using synchronous programming languages such as Esterel, Lustre or Signal. The DPE provides a C++ language binding and an Esterel language binding.

One of the main objectives of a microkernel-based DPE is the integration of real-time Quality of Service support within the TINA-C framework. The microkernel-based DPE is an object management system supporting dynamic configuration, continuous media streams, real-time synchronization, and deterministic temporal Quality of Service guarantees. It also integrates a high-speed network manager (currently based on FDDI).

The technological basis for the microkernel-based DPE is the CHORUS open microkernel, extended with deadline-driven thread scheduling, integrated thread and network (FDDI) management for the provision of deterministic end-to-end real-time guarantees on object execution and communication. Guarantees include delay bounds, delay-jitter bounds, and throughput bounds. They are expressed in the form of declarative statements associated with interfaces or bindings between interfaces.

Several applications are currently being built on the microkernel-based DPE including: a distributed monitoring application, which allows the registration and real-time observation of selected events with causal and temporal consistency; a distributed simulator, which emulates the presence of a wide-area, high-speed transport network and demonstrates the implementation of the monitoring application as a TINA-compliant information networking (IN) service.

5. Open microkernels create also a market shift

The technical shift introduced by open microkernels in the operating systems software is complemented by a similar shift at the market level, with the introduction of an operating system kernel components market, similar to what happened with the introduction of the microprocessor in the hardware market a few years ago.

Those operating system components can be layered as:

- a family of microkernels closely coupled with the families of microprocessors which they support, scaling from simple processors used in embedded controllers and simple devices to high-end sophisticated processors and attached hardware components;
- families of basic operating system kernel components complementing the services of the microkernel in areas such as memory management, basic networking support, generic device and data management, distributed lock and synchronization, boot services, low-level debugging and monitoring support;
- families of servers supporting operating system kernel personalities, such as process managers, and file and network services; and
- middleware servers, which add value to operating system personalities, such as database managers, system and network administration, directory and naming services, and authentication servers.

This shift will allow to break today's operating system kernels into open environments, where third parties and operating system integrators can add their own value, where new technologies can find their way, and where competition can take place for the benefit of the end user. End users will be provided with operating systems

customized for the applications they intend to use, under a choice of price/performance offerings.

Object-oriented technologies are to play an important role in the construction of the new generation of open microkernel-based operating system environments. They are key to the standardization of the interfaces between operating system components, the re-usability of the same components in various operating environments, and the efficiency of their interactions.

Beyond the microkernel buzzword and hype, "open" microkernels revolutionize the way traditional operating system kernel functions will be handled. It is a natural progression from what UNIX did in restructuring the organization of operating systems functions 15 years ago, which led to the open systems movement. It also goes further in "opening" traditionally closed environments, particularly in the area of telecommunications systems software.

6. Biographies

Michel Gien is CTO at CHORUS Systems. He graduated from Ecole Centrale de Paris in 1971. He was a researcher on computer networks and operating systems at INRIA and CNET, before becoming a co-founder of CHORUS Systems in 1986.

Jean-Bernard Stefani is in charge of the distributed systems architecture research group at CNET. He graduated from Ecole Polytechnique, and ENST in 1982 and 1984 respectively. He was responsible for the development of X.400 messaging systems at SEPT before joining CNET in 1989. He is also Chairman of WP4 in ITU-T SG7 and ITU-T SG7 Rapporteur on ODP.

7. References

[1] M. Rozier, V. Abrossimov, F. Armand, I. Boule, M. Gien, M. Guillemont, F. Herrmann, C. Kaiser, S. Langlois, P. Leonard, W. Neuhauser, "CHORUS Distributed Operating Systems", Computing Systems Journal, V 1, N 4, The Usenix Association, pp. 305-370, December 1988.

[2] A. Bricker, M. Gien, M. Guillemont, J. Lipkis, D. Orr, M. Rozier, "A New Look at Microkernel-Based UNIX Operating Systems: Lessons in Performance and Compatibility", Proc. of EurOpen Spring 1991 Conference, EurOpen, Tromso, Norway, pp. 13-32, 20-24 May, 1991.

[3] R. Lea, C. Jacquemot, E. Pillevesse, "COOL: System Support for Distributed Programming", Journal of the ACM, V.36, N.9, pp.37-46, September 1993.

[4] J.B. Stefani, L. Hazard, V. Perebaskine, F. Horn, P. Auzimour, F. Dang Tran: "A real-time DPE on top of CHORUS", Research Report NT/PAA/TSA/TLR/4179, CNET France Telecom, Issy, France, January 1995.

[5] OMG - X/Open : CAE Specification "The Common Object Request Borker: Architecture and Specification", 1994.