Overview of the Masix Distributed Operating System

Franck Mével and Julien Simon
Laboratoire MASI, Institut Blaise Pascal
Université Paris VI
4 place Jussieu
75252 PARIS CEDEX 05, FRANCE
Tél: (1) 44 27 71 04
Fax: (1) 44 27 62 86
{Franck.Mevel,Julien.Simon}@masi.ibp.fr
http://www-masi.ibp.fr/masix

1 Introduction

A current trend in operating systems research is to use microkernels as a basis for new systems. Indeed, microkernels offer many advantages compared with monolithic systems, such as portability, modularity and dynamicity.

Mach has been widely used to build many Unix-like systems, like Lites [Helander 1994] and Mach-US [Stevenson and Julin 1995]. Other personalities have been implemented on top of Mach, like DOS [Malan *et al.* 1991] or OS/2 [Phelan *et al.* 1993]. Nevertheless, to the best of our knowledge, none of the operating systems built on top of a microkernel has really investigated the issues related to multiple personality support.

Masix [Card *et al.* 1994] is a distributed operating system based on the Mach microkernel [Accetta *et al.* 1986], currently under development at the MASI Laboratory. The goal of this project is to investigate extensively the problems related to multiple personality support. We are particularly interested in providing transparent distributed mechanisms to the user processes, such as load distribution [Mével and Simon 1995b], fault tolerance or distributed files. For this purpose, we have designed a Distributed Generic Layer [Mével and Simon 1995a] , made of several servers, interposed between the microkernel and the personalities.

In this paper, after a brief overview of Mach, we present the design principles of Masix and its overall structure. Then, we focus on two major services provided by the DGL, i.e. naming and communications.

2 The Mach microkernel

The Mach microkernel provides basic resource management mechanisms, such as :

- task and thread management. They are the implementation basis for processes. A task is a resource allocation unit, whereas a thread is an execution unit ;
- local interprocess communication (IPC), according to a client-server model based on ports and messages. A port is an unidirectional communication channel, protected by send and receive rights. A message is a set of typed data, sent to a port ;
- physical and virtual memory management ;
- physical peripherals management.

3 Design Principles

3.1 System Structure

We believe that it is worth designing an evolutive system in which components can be added in a dynamic way. Therefore, we have chosen to build Masix following the multi-servers approach. Implementing the operating system as a set of communicating user mode tasks offers many advantages:

- each server has its own interface, that insures protection and minimal interactions between the other ones ;
- each server can be written and tested without interactions with the other ones,
- each server is smaller than a traditional kernel and, thus, easier to maintain,
- adding a server in the running system is easy if a dynamic service management has been implemented.

Nevertheless, the multi-servers approach has also its drawbacks:

- shared data are difficult to implement because each server has its own address space,
- the execution of system services often involves several servers and implies message exchanges and context switches. This leads to lower performances due to the context switch overhead.

Lots of optimizations have been included in the design of the system to minimize the number of servers involved in the realization of a system call. Moreover, we plan to use new mechanisms to reduce context switching between tasks : after intensive testing of systems servers it should be possible to take advantage of co-location [Condict *et al.* 1993]. This way, the servers will run in kernel mode and any context switching and memory copying between the kernel and servers will not occur anymore.

System calls are implemented as remote procedure calls: a process sends a request message to a server and waits for the reply.

3.2 Naming and Protection

In Masix, each server is responsible for providing a set of well-defined abstractions and semantics. The combination of the whole set of servers forms the complete operating system.

Each server implements the operations that act on typed objects. The server implements a fixed dialog protocol related to the type of the objects. When a client process needs to act on an object, it sends a request message to the server that manages the object. This message is sent on a Mach port, which is used as the object identifier. The server holds the reception right on this port.

Accessing an object only requires to know its port and its associated dialog protocol. The request sent to the object port is forwarded by the Mach microkernel to the server which manages the object. This communication scheme allows several servers, which manage the same objects types in different ways, to coexist in the system as long as they implement the same dialog protocol.

Using Mach ports as objects references allows a high level of protection. To act on an object, a process needs to hold a send right on its port. When a process accesses an object for the first time, i.e. when it creates or opens the object, the server returns the send right if the calling process is allowed to act on the object. Otherwise, it does not return any send right. Thus, if the server has denied access to the object, the process cannot send any request since it does not hold any send right to the port.

3.3 Server Structure

In Masix, each server is a Mach task. This task contains several threads which wait for requests and execute them. Each object is named by a port and every object port is contained in a single port set. Every thread waits for requests on this port set. Requests are thus dynamically distributed between the threads contained in the server.

3.4 Interactions between servers

Each server needs to maintain some information relative to its objects. Two servers often need to use the same information. In a monolithic kernel, these informations are stored in common tables which are accessed by every component which needs to. In a multi-server system, we cannot use global tables since each server has its own address space. In Masix, each server maintains a subset of the system informations. This subset contains only the informations that the server needs to know in order to manage its objects.

4 Overall Structure

As shown on Figure 1, Masix is made up of two layers, themselves built of several servers :

- the personalities, which implement the semantics of traditional operating systems : Unix, DOS, OS/2, Win32, ... ;
- underneath, the DGL, which offers generic services to the multiple personalities (communication, shared memory, process management, fault tolerance, ...

Figure 1: Overall structure of the Masix System

5 The Distributed Generic Layer

This layer consists of a set of communicating servers which offer distributed services to the personalities. We now present two of these servers : the Administration and Configuration Server and the Generic Network Server.

5.1 The Administration and Configuration Server

The Administration and Configuration Server (ACS) is the first server to run when Masix is started up. It is in charge of maintaining the list of the servers which run on a node. When a server starts its execution, it registers itself with the ACS by sending a message on its port. This message contains a character string used as the name of the server, and a send right on its request port.

When a client wishes to communicate with a server, it sends the string to the ACS and requests a send right on the corresponding port. Nevertheless, not any client should be allowed to request any port in the system. So, we have designed the following security mechanismss :

- separation of the inter-server and the client-server communication interfaces ;
- authentication of the servers by the ACS ;
  - when they register, thanks to digital signatures ;
  - when they request a name resolution, thanks to a secret key algorithm ;
- mutual authentication of local and remote servers, thanks to a secret key algoritm ;

5.2 The Generic Network Server

Many issues arise because of the modularity of Masix, linked to the distribution of the global state of the system among the microkernel and the various servers. As a consequence, many communications are initiated between servers running on a same node on the one hand, and between servers running on different nodes on the other hand. Thus, Masix needs adequate communication mechanisms to minimize their extra cost.

Our solution is based on a Generic Network Server [Mével and Simon 1995c], interposed between the personalities and the microkernel. By using proxy ports and a global name resolution protocol, we scale the local Mach IPC up to a workstation network, without altering their original semantics The GNS also provides encryption services, to make up for the insecurity of the communication links. This way, eavesdropping can be defeated.

6 Conclusion

We have presented the structure of the Masix distributed operating system, as well as some features of its Distributed Generic Layer. We briefly explained the naming and communication services it transparently provides to the personalities.

References

[Accetta *et al.* 1986] M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian, and M. Young. Mach: A New Kernel Foundation For UNIX Development. In *Proceedings of the Usenix 1986 Summer Conference*, June 1986.

[Card *et al.* 1994] R. Card, H. Le Van Gong, and P.G. Raverdy. Design of the Masix Distributed Operating System on top of the Mach Microkernel. In *Proceedings of the IFIP International Conference on Applications in Parallel and Distributed Computing*, November 1994.

[Condict *et al.* 1993] M. Condict, D. Mitchell, and F. Reynolds. *Optimizing Mach-based Systems by Server Co-location.* Technical report, OSF Research Institute, August 1993.

[Helander 1994] J. Helander. Unix under Mach - the Lites Server. Master's thesis, Helsinki University of Technology, December 1994.

[Malan *et al.* 1991] G. Malan, R. Rashid, D. Golub, and R. Baron. DOS as a Mach 3.0 Application. In *Proceedings of the 1st Usenix Mach Symposium*, pages 27–40. Carnegie Mellon University, November 1991.

[Mével and Simon 1995a] F. Mével and J. Simon. Building a Distributed Generic Layer for Multiple Personality Support on top of the Mach Microkernel. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, November 1995.

[Mével and Simon 1995b] F. Mével and J. Simon. Répartition de Charge au dessus du Micro-noyau Mach : Intégration de la Migration dans le Système Masix. In *Actes des Journées de Recherche sur le Placement Dynamique et la Répartition de Charge*, Mai 1995.

[Mével and Simon 1995c] F. Mével and J. Simon. Services de Communication Répartis dans le système Masix. In *Actes des Journées Jeunes Chercheurs en Systèmes Répartis*, Octobre 1995.

[Phelan *et al.* 1993] J. Phelan, J. Arendt, and G. Ormsby. An OS/2 Personality on Mach. In *Proceedings of the 2nd Usenix Mach Symposium*, pages 191–201, April 1993.

[Stevenson and Julin 1995] J. Stevenson and D. Julin. Mach-Us: Unix on Generic OS Object Servers. In *Proceedings of the 1995 Usenix Conference*, January 1995.