

# Building a Secure, Location Transparent Object Invocation System

Willy S. Liao

David M. Putzolu

Roy H. Campbell

Department of Computer Science  
University of Illinois at Urbana-Champaign  
Digital Computer Laboratory  
1304 W. Springfield  
Urbana, IL 61801  
{liao, dputzolu, roy}@cs.uiuc.edu

## Abstract

*This paper addresses efficient object invocation in distributed systems. Such systems include support for transparent object migration between computers connected by a network, delegation and revocation of access to objects in a system, and very high speed communication between objects, whether they are located in the same machine or across the network from each other.*

*Asynchronous Transfer Mode (ATM) networks can be effectively utilized to implement many of the necessary features of secure, location transparent object invocation. By using the switch controller in an ATM network as a repository for location information, and by mapping virtual circuits to pairwise object interactions with explicitly declared rights of each object, we are able to provide a high performance system that is both secure and simple.*

## 1 Introduction

A common characteristic of distributed operating systems is the ability to utilize objects in other memory domains through some sort of invocation via a remote procedure call (RPC)[2]. Some operating systems have also implemented the ability to use process migration [1] to relocate dynamically executing objects so as to achieve better performance [9, 10, 4]. Efficient process migration has desirable properties, thus we seek mechanisms to migrate most objects [11]. In order to support migration of objects and make it as transparent as possible, a distributed operating system must provide a naming scheme for objects that is location independent, so that an object that migrates between machines is still accessible by client objects which had previous connections to it.

Amoeba [8] addresses these issues by associating *ports* with connections to objects, and by mapping ports to a location independent internetwork protocol, Fast Local Internet Protocol (FLIP). This solution, while making object migration transparent to

client software, requires extensive operating system intervention on behalf of any clients connected to a migrated object. When an object is moved this may result in a broadcast based search of intervening networks to find the migrated object. This causes object migration to have undesirable effects on network traffic and client operating system load. In order to make object migration cheap, it should be possible to migrate an object without incurring costs to any machines which have clients of the migrating object. Furthermore, network traffic congestion caused by broadcast based object location algorithms should be avoided if possible in order to limit network impact of object migration.

Acquiring services from objects in other domains necessarily assumes the client has the rights to request such services. Many current network technologies, such as Ethernet, are broadcast technologies. This allows malicious hosts on the network to acquire and use credentials if they are passed over the network in unencrypted form, or to impersonate identities of trusted principals. In order to solve this problem, operating systems such as Taos [6] provide secure channels which either use software or hardware encryption. Software encryption results in high processor overhead and poor performance, while hardware encryption necessitates expensive custom hardware solutions. It is desirable to utilize network technologies such as Asynchronous Transfer Mode (ATM) which are inherently point-to-point, so that channel security would be a function of which machines are members of a unicast or multicast channel.

## 2 Location Transparency

In our system we identify objects with a Global Identifier (GID) that is globally unique, is fixed for the lifetime of the object and does not change after it is initially assigned. Locating the object in a point-to-point system such as ATM is then naturally accomplished using a nameserver, with the ATM switch controller

becoming an agent for name resolution. The nameserver is kept informed of all changes in the location of objects. In our terminology, a client is any entity that communicates with an object. Objects registered in the nameserver can be clients.

Any client that wishes to communicate with an object must establish a full-duplex connection to the object. Thus an object may have many connections to various clients; there is a separate connection for each client, even if two clients are on the same machine. In the case where a client attempts to communicate with an object on the same machine, the OS can use a mechanism like shared memory to implement this connection. For the intermachine case, we use a pair of Virtual Circuits (VCs), one in each direction. Initially, these VCs are opened by the client, using the GID of the object. A special request containing the GID and any necessary security information is sent to the agent, which then locates the object and opens a pair of VCs for a full-duplex connection between the client and the object. The agent passes on any security information, such as credentials or capabilities, to the object, which can then accept the connection or refuse it. Afterwards, communication with the object proceeds directly over the VCs without requiring GID name service from the switch controller.

Communication between the client and a remote object proceeds over VCs that remain fixed from the client's point of view. There is no need for name service once communication begins and the connection has been established. Object invocation can proceed with very low latency and operating system design is simplified. A key feature of this system is that any object migration only affects the communication links of the object migrated. When an object is migrated between machines, the switch controller closes the virtual channel links between the switch controller and the old machine and opens new virtual channel links between the switch controller and the new machine. Therefore new Virtual Circuit Identifiers (VCIs) and/or Virtual Path Identifiers (VPIs) are needed on the new link from the migrated object to the switch. The switch controller notifies the migrated object of these new VCI/VPIs for all the object's open VCs. Clients communicating with the object maintain the same virtual channel links to the switch and thus they continue to use the same VCI/VPIs. Migration of objects is transparent with respect to their clients. The case where an object is moving to the same machine as some of its clients can be optimized to avoid network traffic.

This object identifier system can be extended to encompass Local Area Networks (LANs) such as Ethernet. A translation bridge that resides on both the ATM net and the LAN can act as a proxy for objects on machines connected to the LAN. This bridge would give its own address to the nameserver as the

location of GIDs for objects on its LAN. All traffic over VCs to the bridge would be forwarded by transport layer protocols over the LAN to the appropriate machine. Migration of objects between machines on the LAN would be handled by the bridge without notifying the agent. Only migration between ATM ports would actually require the agent's intervention, as in the all-ATM case.

The ATM switch controller can be used for implementing migration policies in addition to providing GID name service. Since the nameserver has the location of all objects in the network, it can also store associated information with each object. Policy modules use this information to choose a new location for an object.

For efficiency, a user program employs a GID only when establishing communication with an object. The user asks the kernel to establish a connection with an object named by a GID. If the object is not on the local machine, the switch controller is contacted and a connection is established to the object on the remote machine. The object accepts the connection and user communication can begin. The user is then given a handle by the kernel for identifying the object when communicating with it. The implementation of this handle is optimized so that local objects can be accessed rapidly. For example, the handle can be represented as a C++ object with a virtual method lookup table, using proxies [7] for cross-domain invocations. Users in the same protection domain as the object suffer no speed penalty. Users in different domains on the same machine or on different machines invoke methods that trap into the kernel and use lightweight Remote Procedure Call (RPC) or ATM VCs to communicate. When an object migrates into or out of the same domain as a client, the operating system can transparently change the handle's implementation to the appropriate type without disturbing the client's view of the handle.

The above discussion does not address the issue of how communication between an object and its client should be structured. An acknowledged reliable protocol is desirable for RPC-style method invocations of the object by the client. This is provided by a user level library. Any library state associated with communications must be made part of the object so that it is migrated with the object.

The kernel state needed to implement connections to GIDs is minimal. Only the VCs of all connected objects are required. Kernel implementations of object handles require additional state, but this state is local to the machine and can be created or destroyed as the object moves to or from the local machine. In the case of the user-level library above, any per-connection state must be associated with the object so that user-level protocol state can be migrated with the object.

### 3 Security Support

Our model of object communication has important implications for operating system security. Our system requires that a client prove its identity and rights to access an object in order to be given a connection to that object. This access control model of security was proposed in [5] and is still in use in modern distributed operating systems such as Taos [6].

Assuming all network connections are made over an ATM network with a switch that is trusted provides significant security benefits over using a traditional broadcast based network such as Ethernet. As a point-to-point network, where each physical link only connects to an ATM switch on one end and either an ATM switch on the other end or an end-host, it is safe to transmit information over ATM links in plaintext. Machines which are not trusted are never able to see secure traffic, because the switch (which is trusted) only routes data between secure hosts in plaintext form. This assumes physical security of the network. If this assumption does not hold, it is necessary to use traditional encryption methods when transmitting data over the network.

Actual policies and protocols for determining identity, authorization, and delegation are not specified. We provide a framework within which various protocols may be implemented, depending on the security requirements of the users of the system.

Our system mandates that once identity and authority are established, a connection be made between the object and the client. This connection, along with associating an object being communicated to with a particular handle, represents the granted rights for the client to request operations by the object.

Since the ATM switch is trusted and the physical links are point-to-point, each principal can safely assume that any requests or responses coming in on a particular connection must be from the previously identified client/object. This results in very fast processing of requests, as they are already established as valid and require no further encryption or verification. Furthermore, revocation of rights simply maps to removing a connection between a client and an object at the switch.

Our design does not address all the necessary aspects of security in a distributed system. We view it as a useful framework that provides secure connections and implicit authentication of principals, with such security issues as delegation to be addressed by higher layers of the operating system.

### 4 Conclusion

Our scheme for identifying and locating objects through GIDs provides location transparency and security in a natural way by tightly coupling the OS with the ATM network. Location transparency is achieved by using the ATM switch controller to al-

ter virtual channel links so that clients see no change in their connections with the migrating object. This greatly reduces client overhead since no name service or broadcast lookup is required every time the object migrates. Security is achieved by providing separate connections for each client that wishes to communicate with an object. Per-connection policies may then be applied by the operating system so that authentication, delegation and revocation can be chosen on a case-by-case basis as desired. Once a connection is opened, no further overhead is incurred during data transmission since we rely on the security of the ATM switch controller.

By virtue of its role in the network, the switch controller is an ideal agent for the nameserver. Since the controller is contacted for both call establishment and nameservice, the number of messages to establish communication with an object is reduced. Placing the agent in the switch controller also reduces the number of VCs needed to build a distributed system since the agent is at a well-known address. Locating the agent dynamically or opening new VCs to the agent is not required. The switch controller can also collect information about communication between objects. We can exploit this role by implementing global policies in the switch controller, such as decisions about when and where to migrate objects. Using the switch controller in this manner simplifies the design and implementation of algorithms for performing global actions.

We believe this is a novel implementation of a switch-assisted directory scheme. All of the benefits offered by our object naming scheme derive from its reliance on ATM networking. Operating systems whose underlying primitives closely match the available hardware are in general simpler to implement and higher in performance. We believe our use of ATM is a good tradeoff given current trends in high-performance networking toward wide-scale ATM deployment.

We are currently implementing ATM objects in *μChoices*[3], an object-oriented microkernel-based operating system. We will focus initially on the performance of connection setup and object migration. We also plan to investigate the proper object environment for using ATM objects and their use as a low-level communication paradigm in distributed object-oriented operating systems.

### References

- [1] Artsy and Finkel. Designing a process migration facility: the Charlotte experience. *IEEE Computer*, 22(9):47–56, September 1989.
- [2] Brain Bershad, T.E. Anderson, Ed Lazowska, and Henry Levy. Light Weight Remote Procedure Call. In *Proceedings of the Twelfth ACM Symposium on Operating Systems Principles*, pages 102–112, December 1989.

- [3] Roy H. Campbell and See-Mong Tan.  $\mu$ Choices: An Object-Oriented Multimedia Operating System. In *Fifth Workshop on Hot Topics in Operating Systems*, Orcas Island, Washington, May 1995. IEEE Computer Society.
- [4] Douglass and Ousterhout. Transparent process migration: Design alternatives and the Sprite implementation. *Software - Practice and Experience*, 21(8):757-786, August 1991.
- [5] Butler Lampson. Protection. *ACM Operating Systems Review*, 8(1):18-24, January 1974.
- [6] Butler W. Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in Distributed Systems: Theory and Practice. In *Proceedings of the ACM Symposium on Operating System Principles*, pages 165-182, 1991.
- [7] Marc Shapiro. Structure and encapsulation in distributed systems: The proxy principle. In *6th International Conference on Distributed Computer Systems*, May 1986.
- [8] Andrew S. Tanenbaum, Robbert van Renesse, Hans van Staveren, Gregory J. Sharp, Sape J. Mullender, Jack Jansen, and Guido van Rossum. Experiences with the Amoeba distributed operating system. *Communications of the ACM*, 33(12), December 1990.
- [9] Theimer, Lantz, and Cheriton. Preemptable remote execution facilities for the V-system. In *10th ACM Symposium on Operating Systems Principles*, pages 2-12, 1985.
- [10] Zayas. Attacking the process migration bottleneck. In *Eleventh ACM Symposium on Operating System Principles*, pages 13-22, November 1987.
- [11] Zhou, Zheng, Wang, and Delisle. Utopia: a load sharing facility for large, heterogeneous distributed computer systems. *Software - Practice and Experience*, 23(12):1305-1336, December 1993.