Introducing SEAN: Signaling Entity for ATM Networks

Sean Mountcastle * David Talmage * Bilal Khan * Spencer Marsh †
Abdella Battou † Daniel C. Lee ‡

Abstract

SEAN is freely distributed, object–oriented, extensible software for research and development in host ATM signaling. SEAN includes a complete source-level release of the host native ATM protocol stack, and implements the ATM User Network Interface, compliant to the ITU Q.2931 specification for point-to-point calls, the ITU Q.2971 extension for point-to-multipoint calls, and the ATM Forum extension UNI-4.0 for leaf initiated join calls. SEAN provides APIs to the programmers writing application programs that require ATM signaling. Developers can easily modify and extend SEAN, using the framework library released together. This paper briefly describes essential parts of SEAN's architecture and guides the users and protocol developers.

1 Introduction

In recent years we have witnessed a remarkable growth in the availability of affordable ATM hardware. Yet the majority of the scientific research community has found it difficult to engage in much needed basic research in the areas of ATM protocol design and network performance optimization. We believe that this has been partly due to the dearth of affordable, publicly available software implementations of the host ATM protocol stack. In response to this situation, the U.S. Naval Research Laboratory has released SEAN (Signaling Entity for ATM Networks)[1]. SEAN is a freely distributed, extensible environment for research and development in host ATM signaling. SEAN includes a complete source-level release of the host native ATM protocol stack, and implements the ATM User Network Interface, compliant to the ITU Q.2931 specification for point-to-point calls, the ITU Q.2971 extension for point-to-multipoint calls, and the ATM Forum extension UNI-4.0 for leaf initiated join calls. SEAN compiles and runs on SunOS 5.x (Solaris). It is known to compile on SunOS 4.x, Linux 2.x, and NetBSD 1.3.x. It is our hope that SEAN will serve as the starting point for bold new ATM initiatives, both in terms of probing academic research and innovative commercial development.

SEAN software is object–oriented and can be used both for experimentation/operation and simulation. SEAN's host native ATM protocol stack can be installed in the host to perform experiments. SEAN also includes software that simulates an ATM switch. SEAN supports point to point calls, signaling of individual QoS parameters, ABR signaling for point to point calls, ATM multicast signaling, ATM anycast, traffic parameter negotiation, etc. The SEAN architecture includes a few essential parts - the signaling daemons (SD and SIM models), the application programming interface (API), and the address registration daemon (ILMID).

The **SD** is the live signaling daemon, whose implementation includes a driver interface layer, SSCOP, SSCF, the signaling layer (Q.2931, Q.2971, and UNI-4.0), the Call Control layer, etc. Figure 1 shows several applications communicating with one another on an ATM network through their APIs and Signaling Daemons.

The **SIM** is the simulated signaling daemon. The motivation behind the inclusion of a simulated signaling daemon is twofold; to enable research in ATM signaling even in the absence of ATM hardware, and to allow researchers to debug their native ATM applications "offline", i.e. off the network, to ensure proper behavior prior to running them live. The simulated signaling daemon models an arbitrary number, n, of signaling-capable hosts, each connected by a lossless link to a different port of a common n-port switch. Thus the simulated signaling daemon internally instantiates n user-side UNI stacks, and n network-side UNI stacks, each of which is quite similar to the single stack made in the SD. Figure 2 illustrates that a user can use SEAN's simulation module in the absence of ATM hardware.

The **API** is a set of C++ classes that provides access to ATM network services to applications that require ATM signaling. The SEAN API separates an application program from the ATM signaling services that it requires. It hides the signaling mechanisms from the application. The

*ITT Industries, Advanced Engineering & Sciences, Advanced Technology Group, at the Center for Computational Sciences of the Naval Research Laboratory, Washington D.C.

†Princeton Networks Inc.

‡University of Southern California, Department of Electrical Engineering - Systems, 3740 McClintock Avenue, Los Angeles, CA 90089-2565. dclee@usc.edu

0-7803-6451-1/00/$10.00 © 2000 IEEE

ample, to run SD on a Solaris computer and make it listen on port 9000, a user commands (with executable name `sd_SunOS5`):

```
sd_SunOS5 9000
```

SIM models a switch connected to arbitrary number of hosts. Specifically, each of the simulated switches ports is connected to a host stack similar to the one implemented inside the live signaling daemon. Each of these host stacks listens on a socket for incoming connection. Command for running SIM requires two command line arguments; for example, with executable name simswitch the command line may be

```
simswitch base-ipc-port host-file
```

where

base-ipc-port is the number of the first socket. In a simulated switch of n ports, the first port signaling stack listens on the port $base - ipc - port$. The second listens on $base - ipc - port + 1$. The n^{th} port listens on $base - ipc - port + n - 1$.

host-file is the name of the file that contains the names and ATM addresses of the hosts in the simulation. Each line of the file names one host and its ATM address. The ATM address comes first on each line. The host name comes second. Comments begin with a # and end at the end of the line. Blank lines are ignored. A sample host file, with file name host.file, is included below:

```
#                    Nsap                                    host
0x47.0005.80.ffde00.0000.0000.0104.000000000000.00 foo
0x47000580ffde000000000001040000000000001.00       bar
47.0005.80.ffde00.0000.0000.0104.000000000002.00   bas
```

After the execution of:

```
simswitch 9000 host.file
```

applications may initiate a signaling sessions to ports 9000-9002, and operate indistinguishably from if they were running of a live signaling daemon (sd). Note that all applications must run on the same host as the simswitch.

When SD's are running in different hosts, applications can start running in the hosts. Or, it SIM is running in a host, multiple applications can run in the host and communicate with each other.

4 Extending and Modifying SEAN: Developer Guide

As SEAN's source code is released, SEAN can be modified and extended for further research and development. In fact, the object–oriented feature makes it convenient. A C++ framework[4, 5, 6, 7] library, CASiNO (Component Architecture for Simulating Network Objects)[8] has been used for modular implementation of SEAN. CASiNO provides programmers with a rich yet modular, coarse-grained data flow architecture, with an interface to a reactor Kernel that manages the application's requirements for asynchronous notifications of I/O, real timers, and custom intra-application interrupts. These features enable developers to write applications that are driven by both data flow and by asynchronous events, while allowing them to keep these two functionalities distinct. CASiNO has been released for the public with SEAN. A developer can use CASiNO to extend SEAN and/or to implement other network protocols.

References

[1] Naval Research Laboratory, "SEAN: Signalling entity for ATM networks." http://www.nrl.navy.mil/ccs/project/public/sean/SEAN-dev.html.

[2] M. T. Rose, *The Simple Book: an Introduction to Internet Management*. Englewood Cliffs, New Jersey: PTR Prentice Hall, second ed., 1994.

[3] ATM Forum Technical Committee, "Integrated local management interface (ILMI) specification." Version 4.0n af-ilmi-0065.000, September 1996.

[4] H. Hüni, R. Johnson, and R. Engel, "A framework for network protocol software," in *Annual ACM Conference on Object–Oriented Programming Systems*, 1995.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Pattern, Elements of Object–Oriented Software*. Reading, MA: Addison–Wesley, 1995.

[6] L. P. Deutsch, "Design reuse and frameworks in the smalltalk–80 system," in *Software Reusability, Volume II: Applications and Experience* (T. J. Biggerstaff and A. J. Perlis, eds.), (Reading, MA), Addison–Wesley, 1989.

[7] R. E. Johnson and B. Foote, "Designing reusable classes," *Journal of Object–Oriented Programming*, vol. 1, pp. 22–35, June/July 1988.

[8] S. Mountcastle, D. Talmage, S. Marsh, B. Khan, A. Battou, and D. C. Lee, "CASiNO: A component architecture for simulating network objects," in *Proceedings of 1999 Symposium on Performance Evaluation of Computer and Telecommunication Systems*, (Chicago, IL), pp. 261–272, Society for Computer Simulation International, July 1999.