NETWORK SOFTWARE: FROM NCP TO UBIQUITOUS COMPUTING

Vijaya R. Lakamraju, Raksit Ashok, Osman S. Unsal, and **C. Andras Moritz**
Department of Electrical and Computer Engineering University of Massachusetts Amherst, MA, USA

Vlad Vlassov
Department of Information Technology, Royal Institute for Technology Kista, S-164 40

Contents

Summary

This chapter presents an overview of communication software layers used in today’s computer systems. We first define the scope of this chapter and then provide a loose chronology of the various technologies, capturing the functional sophistication of the different layers, and identifying a recent trend towards network based solutions that require the entire gamut of low-level network software, application oriented layers, and standardized distributed services.

Next, a sample of representative communication layers are presented, and some of the more recent service oriented communication infrastructures are introduced.

Over the next decade, we expect networked solutions to continue to evolve in the direction of becoming ubiquitous, a world identified by everything connected to a network providing a range of sophisticated services, a vision that will likely push for new complex higher-level network software infrastructures and radical changes in existing networking software standards –with perhaps a renewed focus on reliability, scalability, and interoperability.

1. Introduction

Computers without networking are like humans unable to communicate with each other. Without the ability to share ideas and re-sources, the computer world would be rather dull and inefficient. Computer communication has thus become an essential part of our infrastructure. The computer’s ability to communicate is achieved via a number of components, some implemented in hardware and the others in software. Each of these components has a distinct functionality and solves one part of the communication problem. They are typically arranged in a layered architecture with a lower layer providing services to the layer above it. The division of functionality is chosen carefully to ensure that the entire communication problem is solved efficiently and effectively. A complete specification of the layers and their functionality (i.e protocol) constitutes a network architecture. A number of such network architectures and protocols have been proposed and implemented. One such net-work architecture is TCP/IP which forms the basis of the Internet today. Table 1 shows some of the major protocols and application components in a TCP/IP protocol stack.

Basic communication hardware consists of mechanisms that can transfer bits from one point to another. Typically the lower layers of a network stack are implemented in hardware (e.g., the physical and the data link layer) whereas the layers above them are implemented in software. The software handles most of the communication details and problems making it possible for applications to communicate easily. Thus, application programs rely mainly on network software for communication; they rarely interact with the network hardware directly. It is this network software that is the main focus of this paper.

APPLICATION	**SMTP**	**Telnet**	**FTP**	**HTTP**	**BGP**	**SNMP**	**DNS**
TRANSPORT	TCP				UDP		
INTERNET	IP					ARP	
DATA LINK	Ethernet	X.25	ISDN	Frame Relay	FDDI	IEEE 80211	
PHYSICAL	Copper wire		Optic Fiber		Radio Waves		

Table 1: A simplified TCP/IP protocol stack

Until not so many years ago network software meant a few structured, low-level messaging layers. Today, with the Internet, wireless communications, parallel and distributed systems, this term is much less clear, as it can cover everything from low level protocols, to distributed services. It is a generic term that is used to describe a software layer that provides networking services in loosely coupled systems (e.g., networks of computers and wireless networks) as well as in tightly coupled systems (e.g., multiprocessors and emerging single-chip parallel architectures). We consider the Internet as one huge loosely-coupled system. While loosely coupled systems tend to use a more generic set of communication layers, the focus on high performance in tightly-coupled systems makes those layers less distinct. In order to put both these systems on a common footing, we have chosen to combine the en-tire gamut of network layers into three layers, namely a low-level network software layer, an applications-specific layer and a more services-oriented layer. The low-level network software layer interacts with the hardware layers to provide reliable communication between the sender and the receiver. The application layer contains functionalities such as a programming interface, provisions for uniform data representation and network applications such as email and file transfer. Our topmost layer captures the various distributed services and their quality of service. While this paper does not (and cannot) provide a complete coverage of all aspects, it at-tempts to give a chronological perspective capturing some of the more important developments in network software.

The paper starts with a chronology of the various protocols, organized on the functional role of the different layers, identifying a recent trend towards network based solutions that require both low-level network software, application oriented layers, and standard distributed services.

Next, a sample of representative communication layers are presented, and some of the more recent service oriented communication infrastructures are introduced. We briefly describe IP and TCP as an example of the low-level network software layer and then move on to describe sockets, the most widely used application program interface. We then look at some technologies such as Remote Procedure Calls (RPC), Java Remote Method Invocation (RMI) and the Wireless Application Protocol (WAP) which are more relevent to loosely-connected systems. For tightly coupled systems, we describe Message Passing Interface (MPI), Active Messages, and OpenMP in some detail. As an example of emerging service oriented networking software architectures, we describe the ideas behind Jini, a Java based standard promoted by Sun Microsystems and a majority of network communications and hard-ware device vendors. Finally, we take a look at ubiquitous computing, a paradigm that can change the way we look at computers in the future, and then conclude the paper.

This article does not focus on performance issues related to communication layers, although performance always was and re-mains a key motivational factor for improvements. Issues related to communication performance in parallel and distributed systems are covered in numerous research articles, for example in [13], [15], [16], [20], and [21], and books, such as in [6] and [7].

2. Chronological Taxonomy

The first well-documented "roots" of network software date back to 1969 when Steve Crocker of UCLA created the first Request for Comments (RFC) document titled "Host Software". It contains a summary of the software used in the Interface Message Processor (*The IMP was a dedicated piece of hardware used to perform network functions in ARPANET, the precursor of the Internet*) (IMP) and the interface between the IMP and the host connected to it. Issues such as header format, congestion and connection establishment were discussed even in this document. This document formed the basis for the Network Control Protocol (NCP) which eventually established the Transfer Control Protocol (TCP) in 1974. Meanwhile, a number of application layer protocols such as FTP, SMTP (for email) and Telnet kept the ball rolling.

Soon, TCP's routing functions are separated into a separate protocol called Internet Protocol (IP) and in 1982, their combination officially became the protocol suite for

ARPANET. The rapid growth of the network and its working group led to an unprecedented number of RFCs which extended the services provided by the network and the applications that could run on it. Apart from TCP/IP, other models for communication over a network were also proposed, such as the Open Systems Interconnect (OSI) model. All these models were based on a layered architecture where each layer performs some well defined functions and by doing so, provides a set of services to the layer above it.

The topmost layer, the application layer, was the focus of much activity in the 1990s with the advent of the World Wide Web (WWW). The development of application level software layers pushed for convenient abstractions at the interprocess communication layers, hence the design of Remote Procedure Calls (RPC) in 1987. This concept was revisited in the late 1990s with Java's Remote Method Invocation (RMI).

With the advent of computer networks came the possibility of distributed applications. To facilitate standard communication in a high-performance parallel or distributed application, MPI was introduced in 1993. While MPI is widely used both for distributed systems and multiprocessors, OpenMP, a recent programming model using the shared-memory based communication model, has been proposed as a standard for shared-memory multi-processors. Meanwhile, Active Messages and Fast Messages were introduced as alternatives to MPI.

Starting from around 1995, more and more information systems are being built using a distributed processing model, and more recently also with a component based architecture. The Common Object Request Broker (CORBA) model, initiated in the early 90s, implements a set of common services ranging from distributed trans-actions, event models, and interaction models to enable easy sys-tem integration in a heterogeneous world of distributed processing. CORBA is still the multivendor standard for object-oriented distributed computing, and its activities are coordinated by the OMG group [5].

A similar model that leverages some of the ideas in CORBA's Object Management Architecture (OMA), was initiated by Sun Microsystems in 1998. This model, called the J2EE [8] enterprise application framework, defines a distributed application architecture, enabling n-tier application structure, and standard resource manager (*aka* application server), using JAVA as underlying technology. Similar initiatives from Microsoft gave birth to COM/DCOM and the .net software libraries for distributed processing in the Microsoft Windows world. Jini is another recent network software technology, based on Java, that is an example of software infrastructures that enable devices and services to be integrated into the internet. Other network software APIs, for example J2ME [8] and WAP, are proposed to standardize distributed computing with re-source constrained devices and wireless networks. Many of the recent high-end e-commerce applications and Business to Business (B2B) communication solutions, already shifted towards solutions based on these models.

A chronological taxonomy of network software is shown in Figure 1. Our objective with this taxonomy is twofold. First, we show that the network software community is developing more layers at the application software and/or service level. Second, we predict that not so many years from now ubiquitous computing will be a reality, with not only computers, but also information appliances and household appliances will be connected to a network. The latter, together with emergence of new services that can be discovered dynamically on the net, is perhaps the primary driving force in this field today.

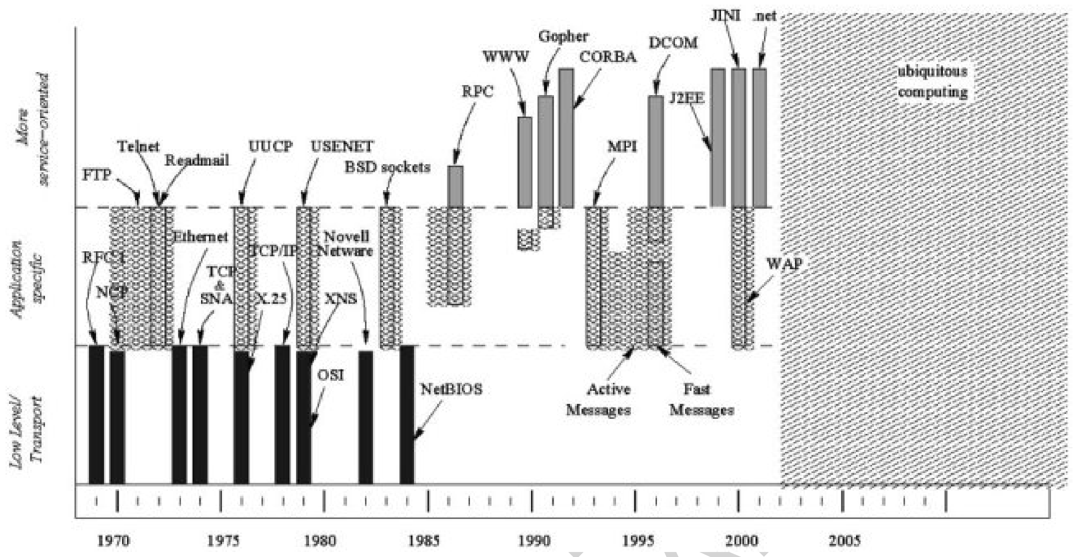

Figure 1: A chronological classification of network software.

-
-
-

Bibliography

[1] Bilal Chinoy, and Kevin Fall “TCP/IP and HIPPI Perfor-mance in the CASA Gigabit Testbed”, *Proceedings of the 1994 USENIX Symposium on High-Speed Networking*, Au-gust 1994

[2] Riccardo Gusella “A Measurement Study of Diskless Work-station Traffic on Ethernet”, *IEEE Transactions on Communications*, 39(9), September 1990

[3] Scott Pakin, Vijay Karamcheti, and Andrew A. Chien “Fast Messages: Efficient, Portable Communication for Worksta-tion Clusters and Massively Parallel Processors”, *IEEE Concurrency* , Vol. 5, No. 2, April -June 1997

[4] Thorsten von Eicken, David E. Culler, Seth C. Goldstein, and Klaus E. Schauser, “Active Messages: a Mechanism for Integrated Communication and Computation ”, *Proceedings of the 19th International Symposium on Computer Architecture*, May 1992, pp. 256-266.

[5] Jon Siegel, “CORBA, Fundamentals and Programming”, Object Management Group, 1996.

[6] W. Richard Stevens, “UNIX Network Programming”, Prentice-Hall, 1990.

[7] Andrew S. Tanenbaum, “Distributed Operating Systems”, Prentice-Hall, 1995.

[8] Java APIs, Sun Microsystems, http://www.javasoft.com.

[9] Wireless Application Protocol, http://www.wapforum.com.

[10] Bill Verners, “Objects, the Network, and Jini”, http://www.artima.com/jini.

[11] G. Abandah and E. Davidson, “Modeling the Communication Performance of the IBM SP2,” *Proc. 10th IEEE Int’l Parallel Processing Symp.*, Honolulu, Hawaii, pp. 249-257, April 15-19, 1996.

[12] R. Alasdair, A. Bruce, J. Mills, and A. Smith, “CHIMP/MPI User Guide,” Technical Report EPCC-KTP-CHIMP-V2-USER, Edinburgh Parallel Computing Centre, The University of Edinburgh, June 1994.

[13] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman, “LogGP:Incorporating Long Messages into the LogP Model -one step closer towards a realistic model fpr parallel computation,” *Proc. Seventh Ann. ACM Symp. on Parallel Algorithms and Architectures SPAA’95*, New York, pp. 95-105, 1995.

[14] T.von Eicken, D. E. Culler, S. C. Goldstein and K. E. Schauser, “Active Messages: A Mechanism for Integrated Communication and Computation”, *Proc. of the 19th Int. Symp. on Computer Architecture, pages 256-266’*, Gold Coast, Australia, May 1992.

[15] C Andras Moritz, Matthew I.Frank. “LoGPC: Modeling Net-work Contention in Message-Passing Applications.” *Proceedings of ACM Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS /PER-FORMANCE 98*, Wisconsin Madison, June 1998.

[16] C. Andras Moritz, K. Al-Tawil, and B. Basilio, “Performance Comparison of MPI on MPP and Workstation Clusters,” *Proc. 10th Int’l Conf. Parallel and Distributed Computing Systems*,

New Orleans, Louisiana, pp. 167-172, Oct. 1-3,1997.

[17] G. Burns, R. Daoud, and J. Vaigl, “LAM: An Open Clus-ter Environment for MPI,” Ohio Supercomputer Center, May 1994.

[18] R. Barriuso and A. Knies, “SHMEM User Guide for C,” Technical Report Revision 2.2, , Cray Research Inc., August 1994.

[19] K. Cameron, L. J. Clarke, A. G. Smith, “CRI/EPCC MPI for CRAY T3D” Technical Report, Edinburgh Parallel Computing Centre, The University of Edinburgh, 1995.

[20] D. Culler, R. Karp, D. Patterson, A. Sahay, K. Schauser, E. Santos, R. Subramonian, and T. Eicken, “LogP: Towards a Realistic Model of Parallel Computation,” *Proc. of Fourth ACM SIGPLAN Symp. on Principles and Practices of Parallel Programming*, vol 28, pp. 1-12, May 1993.

[21] D. Culler, L. Liu, R. Martin, and C. Yoshikawa, “LogP Performance Assessment of Fast Network Interfaces,” IEEE Micro, 1996.

[22] B. Gropp, R. Lusk, T. Skjellum, and N. Doss, “Portable MPI Model Implementation,” Argonne National Laboratory, July 1994.

[23] K. Keeton, T. Anderson, and D. Patterson, “LogP Quantified: The Case for Low-Overhead Local Area Networks,” *Hot Interconnects III: A Symp. on High Performance Interconnects*,

Stanford University, Stanford, CA, Aug. 10-12, 1995.

[24] M. I. Frank, A. Agarwal, M. K. Vernon, “LoPC:Modeling Contention in Parallel Algorithms”, *Proc. of the SIXTH ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, Las Vegas, Nevada, June 18, 1997.

[25] MPI Committee, “MPI: A message-passing interface standard,” *Int’l J. Supercomputer Applications*, Vol. 8, pp. 165-416, Fall/Winter 1994.

[26] MPI Forum, “MPI: A message-passing interface standard,” Technical Report UT-CS-94-230, Department of Computer Science, University of Tennessee, 1994.

[27] R. W. Numrich, P. L. Springer and J. C. Peterson, “Measurement of Communication rates on the

Cray T3D Interprocessor Network," Proc. HPCN Europe 1994, Munich, April 18-20.

[28] P. Worley, "MPI Performance Evaluation and Characterization using a Compact Application Benchmark Code," *Proc. Second MPI Developers Conference and Users' Meeting*, eds. A Lumsdaine and A. Skjellum, IEEE Computer Society Press, pp. 170-177, 1996.

[29] Mark Weiser, "Some Computer Science Problems in Ubiquitous Computing," *Communications of the ACM*, July 1993.

[30] LeonardoDagum and Ramesh Menon, "OpenMP: An Industry-Standard API for Shared-Memory Programming," *IEEE Computational Science & Engineering*, volume 5, number 1,1998.

[31] Rohit Chandra, Ramesh Menon, Leo Dagum, David Kohr, Dror Maydan, and Jeff McDonald, "Parallel Programming in OpenMP", *Morgan Kaufmann Publishers*, 2000.