COMMUNICATION ARCHITECTURES FOR DISTRIBUTED SYSTEMS

F. Panzieri

Rapporteur: Mr. G. Dixon

Communication Architectures for Distributed Systems

F. Panzieri

SUMMA - Sistemi Uomo Macchina
Via Ruggero Fauro 63, 00197 Roma (Italy)

Extended Abstract

A number of reliability and performance issues must be considered in order to provide adequate communication support for distributed systems. This lecture discusses those issues, describes the design of a communication architecture recently developed at the Computing Laboratory of the University of Newcastle upon Tyne, and examines how those issues have been dealt with in the design of this architecture. This architecture, fully described in /1/, has been designed in the first place for the construction of distributed computing systems based on high bandwidth local area networks, typified by the Cambridge Ring and Ethernet; however, the architecture has been designed so as to allow its use also over multiple and varied data communication facilities, including wide area networks.

Communication protocols and communication software architectures for local area networks can differ significantly from those found in other types of data communication facilities, such as wide area networks. In general, wide area networks operate over long distances and are characterized by low bandwidth, high error rate and arbitrary topologies which may require complex routing strategies. In contrast, local networks such as the Cambridge Ring and Ethernet operate over shorter distances and are characterized by high bandwidth and low error rate; furthermore, their ring and bus topologies generally require that minimal routing activity be performed by the communication software.

Thus, the designer of network communication software can face two possibly conflicting requirements: (i) to exploit the characteristics of the local area network, so as to provide applications with high performance, reliable interfaces, and (ii) to make his communication architecture usable on both local and wide area networks, so as to aid the development of distributed applications that can operate on both types of networks.

In order to take advantage of the local area network properties, the communication software for these networks may require the use of special-purpose protocols that provide adequate inter-process communication facilities without introducing the overhead entailed by sophisticated flow control, routing and error handling strategies, such as those conventionally implemented on wide area networks. However, the special-purpose protocols used on one local area network may not be the most appropriate on a different local network, or on a wide area network.

One solution to this problem is to provide a standard communication protocol that can operate both on different local networks and on wide area networks. This lecture argues that this solution may introduce limitations due to wide area network features in the design of the local area network software (e. g. packet size limitations, if an internet datagram protocol is chosen as the standard protocol, or connection management overheads if a transport service protocol is chosen instead). An alternative solution, which is discussed in this lecture, is to define a uniform program interface, capable of being supported effectively by a wide variety of communication protocols and media. The communication software supporting this interface, and the software that uses this interface to support distributed applications, can then be structured so as to take into account both the local area network properties and its intended usage. In particular, the proposed structuring confines the use of special-purpose protocols to appropriate levels of the communication architecture and, at least in principle, within a single network. Moreover, the architecture is capable of accomodating the use of wide area networks, and even sophisticated internetwork protocols, all without compromising the specification or use of the uniform interface mentioned above.

This approach can be compared to that proposed by the International Standard Organization for the Open Systems Interconnection (OSI) Reference Model. The OSI Model consists of a seven level hierarchy of protocols that specifies the structuring of the communications between systems interconnected by arbitrary networks. This Model establishes the use of standard communication protocols at each level of the hierarchy in order to overcome possible heterogeneities between systems, and to survive technological changes.

In the design of the OSI Model little emphasis has been placed on the requirements of the applications making use of network communications; in fact those applications fall outside the scope of the Model. In contrast, the approach discussed in this lecture indicates that, when the application requirements are known, even in very general terms, it is also appropriate to design communication architectures which define adequate interfaces that meet those requirements, and make use of suitable protocols (possibly special-purpose rather than standard) to support those interfaces.

Thus, the architecture described in this lecture has been developed according to the following two goals. Firstly, to provide distributed applications with high-performance program interfaces to local area networks, that exploit the high bandwidth and reliability of the local network, and minimize possible host operating system overheads, such as those introduced by data copying and context switching. In order to aid the portability of such distributed applications, these interfaces have been designed so that they can be conveniently implemented on any type of data communication facility. Secondly, to make available reliable, yet efficient, inter-process communication mechanisms that can be used to construct distributed systems without concern for a large class of communication errors, such as those that can be caused by network failures or host crashes.

These goals have been met by structuring our architecture in two principal levels of abstraction, introduced below. The external program interfaces supported by these two levels have been kept separate so as to enable the designers of distributed applications to choose an appropriate subset of the architecture to construct their applications. The internal structuring of the architecture allows for use of further levels, hidden to the application, that optimize communications by using, for example, different communication protocols to perform different functions.

The lower level of this architecture maintains the abstraction of a datagram service that enables application processes to transmit and receive possibly very large datagrams. The actual communication protocols used to support this service need not concern the applications; thus, this service can be implemented on different data communication facilities so as to provide what we have termed a Uniform Datagram Service (UDS) on those facilities that shields distributed applications from the complexities of possibly differing communication interfaces provided by those facilities /2,3/.

The UDS is supported "sufficiently reliably" for the application process to be able to assume that, when the transmission of a datagram terminates normally, that datagram has been delivered with a high probability of success. The design of this particular type of service has been motivated by the observation that a large class of distributed applications can be conveniently constructed by utilizing adequately reliable (as opposed to "guaranteed reliable") datagram-based communications. Such applications can always implement their own additional reliability mechanisms, if necessary, and indeed they will have to do so in order to cope with host reliability and availabilty problems (since even so-called "guaranteed reliable" virtual circuit-based communications, for example, cannot protect against problems introduced by network or host crashes).

The datagram service allows the use of very large datagrams, so as to remove from the application level the need for fragmentation and reassembly of large data objects. These are often expensive operations as they can involve a large number of context switches between the operating system and the application level. In addition, this service is made available via a small number of primitives characterized by so-called "scatter-gather" facilities, which reduce the number of copy operations required at the application level to implement, for example, application-specific protocols. To date our UDS has been implemented on different data communication facilities (e. g. Cambridge Ring, Ethernet, asynchronous lines) so as to maintain a uniform program access interface to those facilities; an implementation for a wide area network is also being carryed out.

The higher level of the architecture consists of a reliable Remote Procedure Call (RPC) mechanism which provides uniform access to both local and remote objects distributed over the network. RPCs are a very convenient means to enable "client" processes to invoke remote services from server processes in a distributed system. Conceptually, a very simple protocol is needed to implement an RPC mechanism: the client sends its service request to the server as a "call" message, and waits for a reply; the server receives the call message, performs the requested service, and sends the result as a "reply" message to the client. However, despite the apparent simplicity of such a protocol, a number of reliability issues are involved that require careful analysis during the design phase. In fact, an adequate RPC mechanism must cope effectively with any residual unreliabilities of the underlying communication facilities (e. g. message loss, duplication), and

with problems arising from crashes of client or server nodes. For example, unless preventive measures are implemented within the RPC mechanism, it is possible that a server, in the presence of some such events, receive multiple "call" messages for a single invocation by a client, thus giving rise to superflous and undesirable executions (often referred to as "orphan" executions) /4, 5/.

The RPC mechanism introduced in this lecture has been designed to deal with these problems, particularly possible crashes of network nodes and processes. This mechanism implements remote calls characterized by "exactly once" semantics, i. e. successful termination of a call implies that exactly one execution of that call has taken place at the server /6/.

The communication architecture described in this lecture has been used to support the development of experimental versions of the Newcastle Connection /7/, a software subsystem that allows the construction of a distributed system, named UNIX United, out of a number of physically interconnected UNIX (1) systems. This architecture has proved very convenient in the development of this particular distributed application; however, we argue that it may well fulfil the needs of a wide variety of distributed applications, and provide a suitable basis for their implementation.

References

/1/ F. Panzieri, "Design and Development of Communication Protocols for Local Area Networks", Technical Report N. 197, The University of Newcastle upon Tyne, Computing Laboratory, March 1985. (Ph. D. Thesis)

/2/ F. Panzieri, B. Randell, "Interfacing UNIX to Data Communication Networks", IEEE Trans. on Soft. Eng. (to appear).

/3/ A. Linton, F. Panzieri, "A Communication System Supporting Large Datagrams on a Local Area Network", Software Practice and Experience (to appear).

/4/ S.K. Shrivastava, F. Panzieri, "The Design of a Reliable Remote Procedure Call Mechanism", IEEE Trans. on Comp., Vol. C-31, N. 7, pp. 692-697, July 1982.

(1) UNIX is a trademark of Bell Laboratories.

/5/ F. Panzieri, S.K. Shrivastava, "Reliable Remote Calls for Distributed UNIX: An Implementation Study", Proc. 2nd Symp. on Reliability in Distributed Software and Data-Base Systems, pp. 127-133, IEEE Computer Society, Pittsburgh (PA), July 1982.

/6/ F. Panzieri, S. K. Shrivastava, "Rajdoot: A Remote Procedure Call Mechanism Supporting Orphan Detection and Killing", Technical Report N. 200, The University of Newcastle upon Tyne, Computing Laboratory, May 1985.

/7/ D. R. Brownbridge, L. F. Marshall, B. Randell, "The Newcastle Connection or UNIXes of the World Unite!", Software Practice and Experience, Vol. 12, pp. 1147-1162, 1982.

DISCUSSION

During the lecture Dr. Panzieri was asked by Dr. Zimmermann what the improvements in network performance he quoted were related to. He replied that for the PDP11/Cambridge Ring/BBP combination the improvements resulted from the addition of a UDS "adaptor", whereas for the Perq/Ethernet/ECMA the improvements were due to both a UDS adaptor and an implementation of the Basic Block protocol. The application from which the results derived was a simple request-response program.

Another question on the UDS protocol came from Mr. Wood, who asked whether the performance improvements were due to the use of a better protocol or the addition of the scatter/gather mechanism. Dr. Panzieri replied that, for the PDP11, approximately 10% was attributable to the scatter/gather mechanism, the remaining improvement being due to the decrease in context switching, whereas in the case of the Perq the major improvement was due to the switch in protocol.

When outlining the orphan prevention mechanism Dr. Panzieri was asked by Dr. Larcombe whether there was a fixed deadline for orphan killing. Dr. Panzieri stated that the deadline was passed as a parameter to the call.

Professor Tanenbaum claimed that exactly once RPC semantics were impossible to achieve and gave a long example (involving valves in a chocolate factory!) supporting his case. Dr. Panzieri replied telling Professor Tanenbaum that he had misunderstood (or forgotten) the definition of "exactly once" RPC semantics.

Dr. Cohen wondered what would happen if the "incarnation number" was lost. Dr. Panzieri replied that this would not occur because the incarnation was saved on a stable counter.

Dr. Larcombe expressed a worry that some sort of stable clock was in use, to which Dr. Panzieri replied that no sychronisation between sites was involved. Dr. Larcombe asked who was responsible for incrementing the incarnation number. Dr. Panzieri replied that each site increments its incarnation number when booted. Dr. Larcombe further expressed a concern that there might be some exchange of time-based data between asynchronous processes in the system. After some discussion Professor Randell pointed out that it was implicit in the mechanism that clocks could not go backwards. Professor Wells questioned whether clocks must go forwards, the reply to which was that a strictly monotonic clock was assumed by the mechanism.

Dr. Burkhardt stated that he appreciated that the model outlined gave higher performance, but that there was really two failure classes; communication and site crashes. There was also an assumption in the model that there would be a stable configuration, this would not be true for systems using the OSI model. He was concerned that recovery from communication failures would occur at the application level which would lead to a decrease in performance. Dr. Panzieri noted these concerns.

After further discussion about the reliability requirements Professor Randell remarked that the concern must be with achieving high performance, with "adequate"" reliability, since 100% reliability would only be required in relatively few cases. Dr. Burkhardt replied that this was fine for LANs but with Public Networks, where traffic schemes might not be known, the situation was different. He continued by giving an example of a Public Network of which initially provided a datagram service and eventually switched to a virtual circuit service to meet performance, and client, requirements. Apparently this was something that the academic world should recognise.

A discussion followed on datagrams and timeout intervals. A pragmatic Professor Needham remarked that from his extensive experience of LANs, some using satellite links, timeout values did not matter if there was little chance of an error occurring.

Professor Tanenbaum stated that Wide Area Networks were of little importance, particularly if the amount of traffic on LANs and WANs were compared. He claimed that the level of traffic occurring on a LAN would be much greater than a WAN therefore the model presented by Dr. Panzieri was the correct way to approach the issue and the OSI model was wrong.

An argument between the various WAN/OSI and LAN factions resulted, which was brought to a halt by Professor Randell who thought this could be saved for another time.