Requirements for High Performance Group Support in Distributed Systems.

Werner Vogels
werner@inesc.pt

Paulo Veríssimo
paulov@inesc.pt

Luís Rodrigues
ler@inesc.pt

INESC*

April,1992

Abstract

Although the use of groups and group communication is becoming more and more accepted as an important tool for implementing distributed applications and algorithms, there is still much controversy about the way the group paradigm should actually be implemented. In this paper the authors identify some of the areas in which these controversies still exist and express their views on the matters causing the disagreements.

Introduction.

The group paradigm is widely accepted as being an excellent method of structuring activities in distributed systems. From the pioneering projects in the past [1,7,5], a large number of research projects in areas related with group structuring and reliable group communication have emerged [2,8,9,10,13,14]. Although there is consensus on the merits of the group paradigm, there is hardly any of this agreement on how the paradigm should actually be implemented and presented to the user. Recently, especially in Europe, we have noticed how far the ideas on related matters are still apart.

INESC has been involved in a number of projects in the area of group support for distributed systems, which has resulted in development of the communication support architecture for the DELTA-4 system and the development of a group communication system for standard workstations. Based on the lessons learned from these projects we want to describe some of the requirements we think are essential for a group support service. These lessons also form the basis for our current research projects involving the design and implementation of group support for micro-kernel and real-time environments.

Since communication speed is the dominant requirement for open distributed computing [4,6,11] we are concerned with providing a *high performance* communication service on which the group support service can be based. This quest for high performance dominates how we think a group communication service should be designed.

The main factor in the design of a group support service, is the use of the *end-to-end* argument. Applying the argument to a group support service will ensure that different requirements from applications can be satisfied without forcing unnecessary costs upon users of the service. The argument dictates the structure of the communication module and the interface through which the module is presented to the user.

In the following sections we would like to discuss some subjects like operating system interaction, use of network technology, communication services, engineering and scalability. It is not the aim of this paper to provide complete coverage of all possible topics, but to contribute some of our experiences to the discussion on how to implement group support services.

Operating system support.

Within a station we view the group support system as a multi-module environment that crosses traditional communication layer and operating system boundaries. Major components are a multicast

*Instituto de Engenharia de Sistemas e Computadores, R. Alves Redol, 9 - 6° - 1000 Lisboa - Portugal, Tel.+351-1-3100281. This work has been supported in part by JNICT, through Programa Ciência.

supporting infrastructure module, a reliable group communication module, a time service and collection of group management modules implementing algorithms in the area of membership, replication and cooperation management [19]. We believe that providing the system service at every station instead of creating a *backbone* service at dedicated stations greatly enhances the performance and the reliability of the group support service.

One of the major questions is what type of support for groups should be given by an operating system. In currently used monolithic systems (like Unix) where all operating systems services are provided by the kernel, the group support system should be part of this kernel. The programming environment can interface with the system through the standard kernel-user interfaces. The argument that not all kernels will need support for groups [2] and that thus the modules should be placed in user space to allow dynamic management of services has been superseded by the fact that modern monolithic systems normally allow dynamic reconfiguration of the kernel by adding device drivers and system calls at run-time.

For the new class of operating systems build on top of a microkernel it is easier to structure a group support service, as one is not bound by the traditional strict layering of communication services that are only accessible through the top layer interface. In a microkernel based environment we see the group support system as a layered environment where one has access to all (also the lower) layers. The group support system should provide transparent service to all operating system environments available at that station.

We should also aim at transparency between traditional and group communication channels. In this case it should be possible, in the traditional client-server model, to replace the single server by a replicated server without modifications at the client [7,14]. This will greatly enhance the reliability of distributed systems applications without having to pay the price of forcing all implementors to use the same paradigm.

Resource management in operating systems will become more and more important as support for multimedia needs to be given. It is very likely that future systems will have mechanisms to refuse the handling of certain requests in order to be able to satisfy requested resource allocation by video or audio streams. In this case our communication system should supply mechanisms that allow protocols to be gracefully terminated or retried. A similar mechanism has been used in the Delta-4 architecture where an *inaccessibilty* check was done to see if the operating system could fulfill the needs of the protocol. There is a crucial difference between a station that can temporarily not service the group support system because of resource management priorities and a station that has failed, so it is important that the operating system on the station in question can notify the other participants of this situation.

Network technology.

As stated before, communication speed is the dominant requirement for distributed applications, and our view is that this requirement should condition the interaction with the network infrastructure. Quite in contrast with the large number of network-independent protocols that have been developed in the last years we argue that protocols on which distributed computing is to be based should exploit the network technologies offered by LANs and MANs.

LANs are integrated in nearly all modern organizations that use distributed computing and the organization-wide distributed systems of the near future will certainly be using MAN technology. These networks are very attractive as a basis for reliable communication as they have properties like low error rates, high availability, high speed and bandwidth and provide multicast facilities. Exploiting these properties in group communication protocols will result in an important increase in performance when compared with the network-independent protocols, and users are less and less willing to pay the price for technology independence when the technology is there.

The best example is the use of hardware multicast facilities. Their benefits are twofold: minimalization of the multi-address message traffic and exploitation of the hardware address filtering mechanisms when mapping group addresses directly into hardware addresses and so minimizing the software message selection and dispatching.

Some other network properties can be used to improve reliability and synchronism of the protocols, to enforce bounded execution times, etc.

To find a solution for the long-lasting contradiction between the low-level approach (not-scalable) and the high-level one (not efficient) we have defined a model in which protocols can take the best advantage of the technology they have available, even in an internetworking environment. This model makes the group communication protocols see an abstract

interface to the network, providing a set of low-level services like address-management and message transmission.

Such an abstract network can be placed at different layers in the traditional communication system. For local network access the abstract network module interacts directly with the network device, in the case of internetwork communication the abstract network representing the internet makes use of other parts of the communication system and abstract networks to transparently transport the message to its non-local destination. Once the message is transported to the network of destination, the system will again take advantage of the properties of the LAN to enforce efficient local delivery [20]. This scheme is used to optimally exploit the use of LAN/MAN technology whenever available.

Addressing facilities.

A group communication support service should provide users with different addressing facilities to allow efficient referencing of groups of participants. The basic facility is that hardware *multicast* addressing can be used, to address a group of participants, avoiding the need to send messages to this group using multiple point-to-point messages.

If one wants to make optimal use of the hardware multicast addressing/filtering offered by the VLSIs, dynamic management of the multicast address space within a local area network is needed. The mapping from group address to hardware multicast address should not be static to avoid the address space at the VLSI to become exhausted after a small number of groups. When a station joins a group the members agree on which multicast address to use for this group, if necessary remapping a group with low traffic characteristics into a multicast address already used by another group. This is all to minimize the use of software filtering which has a very negative impact on performance.

Since the participants are commonly registered by their (48 bits) MAC addresses one should give the user convenient methods of addressing the members of a group though a *logical name*. This frees the user from explicit use of address lists. It also allows the membership list to change or members to migrate without the users having to explicitly handle this.

The last facility that is required is the possibility to reference a subset of the group members through *selective* addressing. In cases where the user explicitly wants to manipulate the address list we have used a mechanism called *short addressing* [16,18] to ease and speed-up the manipulation.

Versatile communication or good use of the *end-to-end* argument.

The basic service of the communication module is to provide reliable message exchange between a group of participants. The semantics of this group communication can be described by combining properties of agreement, order and synchronism [3,14].

The agreement property denotes to who (Quorum or subset) and when (all or none, even if the sender fails) multicast messages should be delivered. The order property assures in what relative order (atomic, causal, FIFO) the messages are indicated to the user.

It may seem very tempting to take the strongest form of each of these properties and present one, very powerful communication method to the user of the service. Some research groups advocate that offering a global, totally ordered multicast through a strong RPC style interface would satisfy all needs of distributed applications [12]. We would like to emphasize that applications have different requirements, especially regarding agreement and order properties. If these different needs are neglected, the communication system will force too many stations to join in the communication, generate unnecessary message traffic or execute too complicated ordering algorithms.

As generally accepted, the *end-to-end argument* [17] should dominate the design of communication module interfaces. We consider it as good use of the argument to supply the user with a set of Qualities of Service (QOS) through which he can access the group communication module and choose from the available properties to obtain a communication service matching his needs. An example set of QOSs are listed in table 1, as available through the xAMp group communication service [15].

The primitives are chosen in such a way that different application requirements can be fulfilled. The primitives range from an unreliable datagram service to a reliable atomically ordered service. It is conform the end-to-end argument that the application is not forced to use the primitives that provide more functionality than actually needed.

There is a class of applications that provides serialization of messages at application level, therefore

Table 1: Qualities Of Service

- **Best-Effort.** Acknowledged datagrams with retries to reach a certain quorum. Quorum can be set by either a number of members or a subset of addressees.
- **Reliable.** Acknowledged datagrams with retries and Quorum specification like in Best-effort but with guarantee of delivering even if the sender fails.
- **Atomic.** Datagrams delivered to all members (including the sender) or none with total order within the group.
- **Delta.** Support for total order of messages based on global time (achieved by synchronized virtual clocks).
- **Causal** All previous primitives can deliver regarding a causal ordering.

making an atomic message ordering superfluous for these applications. The communication needs of these applications are often satisfied by using simple reliable message transfer.

Another type of ordering that can be applied to message delivery is the one respecting causality. When using communication in an asynchronous fashion there is the possibility that the causal relationship between messages will not maintained when delivering messages to the user. Causal information can be added to the messages to ensure that a protocol module indicates the messages to the user respecting this order. Causal order can be added to each of the primitives of the communication service.

We do not consider it good use of the end-to-end argument if a primitive offered forces the user to accept more functionality than requested. Therefore layering an atomic ordered service on top of a causally ordered service violates the argument. The user is forced to accept the cost of the causal ordering even if he does not request it.

Performance issues.

Three matters are important when trying to satisfy very demanding performance requirements of distributed applications. The first is, as discussed earlier, explicit use of network technology. By locating the local execution of the protocol as close as possible to the LAN, one can achieve "startling" [2] performance gains compared the protocols that are based on the generic communication channels.

The second matter is *light-weight* design of the group communication protocols. Most of the issues in light-weight protocols design are relevant for group communications [20], however two of them should get extra attention. The first is that the protocols should be designed to allow a high degree of parallelism. The use of multiple threads of control to implement the several ongoing communications should be stimulated to allow maximum concurrency and possible exploitation of multiprocessor architectures. The second issue is that the handling of receiving frames should have a very short execution path so that upcalls from the network interrupt routine can be used to process the frame.

The third matter is the ability to communicate with the group communication module in an asynchronous fashion. This gives the caller the facility to use mechanisms to collect confirmations and indications without blocking the execution of the application. Although this enhances the structuring of the calling entity it also puts more emphasis on the need to use protocols that respect causal ordering.

Scale.

The group support system should be able to scale well with respect to the existence of many groups at the same station and to the use of very large groups.

It is important to support the notion of a *client* of a group, so that in classical client-server communication, where the server is represented by a group, there is no need for the client to join the group to be able to communicate with the server members[2,14].

Groups with a large number of members impose a possible problem in the area of address handling, the use of short addresses as proposed in [16] can very much speed-up the address manipulation even when addressing a specific subset of group members. When exploiting the use of hardware multicast facilities one can substantially reduce the impact of large groups on the number of messages exchanged.

Another problem that has to be handled when dealing with groups with a large number of members is the acknowledgment strategy. If a multicast message is send out to a large number of participants, one can expect that nearly all participants will be competing to sent the acknowledgment back

at the same time. This can lead to serious congestion problems at sending and receiving stations. The problem is not new, and transport protocols supporting multicast like *XTP* [6] provide excellent *damping* and *slotting* algorithms to handle this problem.

The matter of handling a large number of groups at the same station will mostly influence the group management related modules. It will require intelligent use of hardware address facilities to map several groups onto hardware multicast addresses, so one can still benefit from the use of hardware address selection.

Conclusions.

In this paper we have gone through some issues related to the design and implementation of a group support service. Our main recommendations are:

- The group communication service should be an integral part of the operating system.
- The technology offered by the network infrastructure should be exploited by the group support service.
- Focus on high-performance and use lightweight methods to implement the service.
- Offer a versatile, multi-primitive group communication service to satisfy different application requirements.
- Use different methods achieve efficient addressing.

References

[1] Kenneth P. Birman. *The Process Group Approach to Reliable Distributed Computing*. Technical Report, Cornell University, Ithaca, USA, July 1991.

[2] Kenneth P. Birman, R. Cooper, and B. Gleeson. *Design Alternatives for Process Group Membership and Multicast*. Technical Report TR91-1185, Cornell University, Ithaca, USA, december 1991.

[3] K.P. Birman and T.A. Joseph. Reliable broadcast protocols. In Sape Mullender, editor, *Distributed Systems*, ACM Press Frontier Series, 1989.

[4] Andrew D. Birrell and Bruce Jay Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems*, 2(1), February 1984.

[5] D. Cheriton and W. Zwaenepoel. Distributed process groups in the V-kernel. *ACM Tran. on Computer Systems*, 3(2), May 1985.

[6] Greg Chesson. XTP/PE overview. In *13th Local Computer Network Conference*, Minneapolis-USA, October 1988.

[7] Eric C. Cooper. Replicated distributed programs. In *10th ACM Symposium on Operating Systems Principles*, ACM, Berkeley, California 94720, USA, November 1985.

[8] Flaviu Cristian, Robert D. Dancey, and Jon Dehn. Fault-tolerance in the advanced automation system. In *Digest of Papers, The 20th International Symposium on Fault-Tolerant Computing*, IEEE, Newcastle-UK, June 1990.

[9] H. Garcia-Molina and Annemarie Spauster. Message ordering in a multicast environment. In *9th Internacional Conference on Distributed Computing Systems*, pages 354–361, IEEE, June 1989.

[10] *The ANSA Reference Manual*. Architecture Projects Management, Ltd, Cambridge, UK, release 1.1 edition, July 1989.

[11] Norman C. Hutchinson and Larry P. Peterson. Design of the x-Kernel. In *SIGCOMM'88: Communications Architectures and Protocols*, ACM, Stanford, USA, August 1988.

[12] Frans Kaashoek and Andrew Tanenbaum. Fault tolerance using group communication. *ACM Operating Systems Review*, 147–183, 1991.

[13] Larry L. Peterson, Nick C. Buchholdz, and Richard D. Schlichting. Preserving and using context information in interprocess communication. *ACM Transactions on Computer Systems*, 7(3), August 1989.

[14] D. Powell, editor. *Delta-4 - A Generic Architecture for Dependable Distributed Computing. ESPRIT Research Reports*, Springer Verlag, November 1991.

[15] L. Rodrigues and P. Veríssimo. *xAMp: a Multi-primitive Group Communications Service*. Technical Report, INESC, Lisboa, Portugal, February 1992.

[16] L. Rodrigues, P. Veríssimo, and J. Rufino. *A low-level processor group membership protocol for LANS*. Technical Report, INESC, Lisboa, Portugal, February 1992.

[17] J. Saltzer, D. Reed, and D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4), November 1984.

[18] Michael D. Schroeder, Andrew D. Birrell, Michael Burrows, Hal Murray, Roger M. Needham, Thomas L. Rodeheffer, Edwin H. Satterthwaite, and Charles P. Thacker. *Autonet: a High-Speed, Self-configuring, Local Area Network Using Point-to-point Links*. Technical Report 59, Digital,Systems Research Center, Palo Alto, California, April 1990.

[19] P. Veríssimo and L. Rodrigues. Group orientation: a paradigm for distributed systems of the nineties. In *3rd Workshop on Future Trends of Distributed Computing Systems*, Taipe, Taiwan, April 1992.

[20] Werner Vogels and Paulo Veríssimo. Supporting process groups in internetworks with lightweight reliable multicast protocols. In *ERCIM Workshop on Distributed Systems*, Lisboa, Portugal, November 1991.