IEEE Software

MAY 1985

The Rendezvous Principle

and other examples of distributed systems

- Helix: The Architecture of XMS
- Amaze: A Multiplayer Computer Game
- High-Level Broadcast Communication for LANs
- Multicast Communication on Network Computers
- The ARC Network: A Case Study

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

IEEE Software

May 1985 Volume 2 Number 3 (ISSN 0740-7459)

The 18th century copper engraving on this month's cover illustrates the rendezvous principle in the XMS distributed computing system described in the lead article. The work is formal and structured, like the XMS network protocol. The suitor climbs a ladder to the rendezvous under the watchful eye of an observer, like a task that steps up to a higher-level process under the guidance of the distributed software so it can interact with other system tasks. The system's resource manager at the least observes the rendezvous, but often steps in as a matchmaker or Cupid to make sure the right tasks meet.

Engraving courtesy of the Bettmann Archive

Cover design by Jay Simpson

Next issue: Operating Systems for Highly Parallel Environments

THEME FEATURES

SPECIAL FEATURES

DEPARTMENTS

IEEE Software

Circulation: *IEEE Software* (ISSN 0740-7459) is published bimonthly by the IEEE Computer Society: IEEE Headquarters, 345 East 47th Street, New York, NY 10017; IEEE Computer Society West Coast Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720—(714) 821-8380. Annual subscription: $14.00 in addition to any IEEE group or society dues; nonmembers, $90.00. Single-copy prices: members $7.50; nonmembers $15.00. This journal is also available in microfiche form.

Undelivered copies: Send to 10662 Los Vaqueros Circle, Los Alamitos, CA 90720.

Postmaster: Send address changes to *IEEE Software,* IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854. Application to mail at second-class postage rates is pending at New York, NY, and at additional mailing offices.

Moving?

PLEASE NOTIFY US 4 WEEKS IN ADVANCE

Name (Please Print)

New Address

City State/Country Zip

MAIL TO:
IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854

ATTACH LABEL HERE

- This notice of address change will apply to all IEEE publications to which you subscribe.
- List new address above.
- If you have a question about your subscription, place label here and clip this form to your letter.

Multicast Communication on Network Computers

Ariel J. Frank, Larry D. Wittie, and Arthur J. Bernstein,
State University of New York at Stony Brook

Channel-oriented packet casting is a predominant feature of Micros, an operating system designed to explore control and communication techniques for network computers containing thousands of hosts.

Communication is essential to distributed systems, in which a number of hosts are arbitrarily interconnected by a network. A network computer, or netcomputer,[1] consists of a large number of hosts embedded in a network of interconnected broadcast buses. It has no global clock or shared memory. Instead, a global decentralized operating system, with some code resident in every host, unifies the hosts into a single computer system, providing netcomputer users with a powerful computing facility that can be accessed as a single virtual multiprocessor.

The concept of grouping processes to achieve goals is vital to distributed systems. System services for a group help to support communication and coordination among its members. Distributed groups are often organized to achieve parallel processing, increase data availability, reduce response time, share resources, or increase reliability.

Processes in a group often need to multicast the same message to all other group processes. Such messages include computation results, search bounds, state changes, votes, and updates to replicated data. Group members may need to multicast to the group several times during an extended interaction period. Such group multicast is more effective if some underlying multicast structure exists for the group. However, not all multicast techniques are effective for group multicast.

This article explores different methods for frequently multicasting the same information to groups of computers within large broadcast networks. Three techniques for group multicast depend either on lists of all group members, lists of broadcast subnets containing all members, or local branch tables for a tree spanning all member subnets. Much of the information presented on group organization and group multicast is based on the implementation of the Micros operating system for the Stony Brook netcomputer.

Packet-switched networks

The fundamental unit of information flow in a communication network is a packet. In general, packets are sent by *hosts*, or nodes, on communication *channels,* or links. A channel can be a point-to-point link or a multiaccess broadcast bus such as an Ethernet. Here, we consider the general model as a packet-switched network in which hosts can store packets and forward them on their connected channels. (In subsequent discussion of a specific netcomputer model, we refer to broadcast buses as channels.) Packets are transported across a network in datagram mode, that is, with a "best effort to deliver." Providing reliable transport of multicast packets is difficult because each packet must be acknowledged from a possibly unknown number of destinations. Both point-to-point and broadcast networks are packet-switched.

A group of processes implicitly defines a host group consisting of all hosts on which the processes execute.

0740-7459/85/0500/0049$01.00

A host group can be designated explicitly by a list of member addresses or implicitly by a logical group address. With lists, each group member must maintain a list of members so it can multicast to them. A membership list is either dynamic or static, depending on whether it can change. If a logical address is used, all group members must have network interfaces that will accept packets sent to the logical address. Each interface must recognize multiple logical addresses if its host is a member of several groups.

Evaluating multicast techniques

Multicast techniques can be evaluated relative to bandwidth, delay, state, computation, preparation, maintenance, failure, and scale (see box below right). Tables 1 and 2 rate six types of multicast techniques against these criteria for single and group multicast, respectively. Most of these techniques, which include flooding, separate addressing, multidestination addressing, partite addressing, single-tree forwarding, and multiple-tree forwarding, are adaptations of their broadcast counterparts. All techniques are evaluated for a large multicast group.

The five relative values per criterion are nil, low, medium, high, and gross. Low, medium, and high ratings are always given to some technique. The nil and gross ratings are used only for exceptional cases. A "utopian" multicast technique would have all nil ratings.

Flooding. A brute force technique for packet multicast is to broadcast identical packet copies on all channels. Each receiving host forwards copies to all its other channels. Only group hosts keep a copy of the multicast packet.

This scheme is very simple. State, preparation, and maintenance costs are negligible, and network failures have almost no impact. However, the very name of the technique reflects its major disadvantage: it floods the network with packets. The delay rating is medium because heavy loads slow channel access. Bandwidth use is extremely high, since packets are duplicated on multiple channels. Bandwidth waste increases with network size, causing a high scale rating. Such gross bandwidth usage is not justifiable for multicast.

For flooding to be practical at all, packet life time must be limited to prevent endless duplication. Solutions include recording packet sequence numbers to prevent retransmission and discarding packets after a fixed num-

Table 1. Relative rating of criteria for single multicast.

Multicast Technique	Multicast Criteria			
	Bandwidth	Delay	State	Computation
Flooding	Gross	Medium	Nil	Medium
Separate addressing	High	High	High	Low
Multidestination addressing	Medium	Medium	High	High
Partite addressing	Medium	Medium	Medium	Low
Single-tree forwarding	Low	Medium	Low	Low
Multiple-tree forwarding	Low	Low	Gross	Low

Criteria for evaluating multicast techniques

For a single multicast,

- ***Bandwidth*** — The communication cost of the packet headers for a single multicast. It is the sum of the number of packets sent over all channels times the average size of their packet headers.
- ***Delay*** — The time from the start of the multicast until the last packet copy is delivered. Packet delay per channel is assumed to be uniform. Because packet copies are sent in parallel, delay is a maximum, not a sum. Techniques that minimize delay tend to maximize bandwidth and vice versa.
- ***State*** — The summed cost of storing the information that allows members to multicast to the group. It can include logical identifiers, lists of member addresses, or forwarding sublists forming previously built multicast structures. The state information should be bounded.
- ***Computation*** — The processing cost for a single multicast. It includes calculation of intermediate destinations and update of multicast state information.

For group multicast,

- ***Preparation*** — The initial cost of distributing multicast information to all members. It may include building a structure to lower average cost per multicast.
- ***Maintenance*** — The cost of adapting multicast information as members join and leave the group.
- ***Failure*** — The cost of recovering from the failure of a network host or channel. Failures may require routing around failed components or repairing multicast structures.
- ***Scale*** — The sensitivity of a multicast technique both to larger groups in a fixed-size network and to fixed-sized groups in a distributed system of increasing size. Scale should be at most proportional to the increase in size.

ber of relays. However, the bandwidth is always high.

Separate addressing. An obvious technique for multicast is to send a separately addressed packet to each destination. Each member maintains a copy of the entire group membership list, which is acquired during group preparation. A large group has a large membership list, so the state and scale ratings are high. However, network failure has little effect.

Table 2. Relative rating of criteria for group multicast.

Multicast Technique	Multicast Criteria			
	Preparation	Maintenance	Failure	Scale
Flooding	Nil	Nil	Nil	High
Separate addressing	Medium	Medium	Low	High
Multidestination addressing	Medium	Medium	Low	High
Partite addressing	Low	Low	Low	Medium
Single-tree forwarding	High	Medium	Medium	Low
Multiple-tree forwarding	Gross	High	High	Gross

The major disadvantages of this technique are its high bandwidth and high delay. Several copies differing only in their destination addresses are often sent on the same channel. The multicasting host sends packet copies sequentially, so delay can be high. This technique is suitable for groups with few destination hosts.

Multidestination addressing. In this multicasting technique, a few multiply addressed packets are sent for each multicast. Each packet header includes a subset of the destination addresses. When a packet arrives at any host, its destination addresses are apportioned among multiple copies. Destinations with the same route share the same copy. Packet copies are forwarded to all destinations.

Multidestination addressing is hard to support because of the need for variable-sized packet headers. Because of address apportioning, this technique

Broadcast/multicast communication

Processes in distributed systems communicate by exchanging messages. Frequently, the same message must be communicated to a set of processes. Such messages include system status changes, announcements, and queries. Existing communication standards, such as the open system interconnection reference model,[1] support message communication only to a single destination. Single-destination communication is *unicast,* the delivery of a packet to *one* destination address. It can be costly to unicast separate copies of the same message to a set of processes. Advanced distributed systems provide special support for communication to multiple destinations.[2,3] *Broadcast* is the delivery of a packet to all possible destination addresses, while *multicast* is the delivery of a packet to some specified subset of the possible destinations. Broadcast and unicast are special cases of multicast.

Initial research on techniques for broadcast and multicast centered on point-to-point computer networks and interconnected networks, or internets, especially for Arpanet.[4,5] Dalal[6] did the fundamental work on broadcast techniques in point-to-point networks. Wall[7] extended Dalal's broadcast research and investigated group multicast techniques. Broadcast and multicast structures that have been investigated include "low-cost" minimum spanning trees and "low-delay" shortest path trees.

Most recent research has been done on broadcast networks and internets. Ethernet[8] networks support high-speed (10M-bps) packet switching in locally distributed environments. The Ethernet architecture directly supports broadcast and multicast packet transmission by allowing logical addresses to be set in the hardware interface. The few distributed systems supporting group multicast do so on a single Ethernet.[9] Ethernets have been interconnected to form the Pup internet architecture,[2] providing a rich testbed for broadcasting research. Pup has had a major influence on the internet transport protocols for the Xerox network systems architecture.[3]

References

1. A. S. Tanenbaum, *Computer Networks,* Prentice-Hall, Englewood Cliffs, N. J., 1981.
2. D. R. Boggs et al. "Pup: An Internetwork Architecture," *IEEE Trans. Communication,* Vol COM-28, No. 4, Apr. 1980, pp. 612-624.
3. Y. K. Dalal, "Use of Multiple Networks in the Xerox Network System," *Computer,* Vol. 15, No. 10, Oct. 1982, pp. 82-92.
4. S. M. Abraham and Y. K. Dalal, "Techniques for Decentralized Management of Distributed Systems, " *Proc. Compcon Spring,* Feb. 1980, pp. 430-437.
5. L. Aguilar, "Datagram Routing for Internet Multicasting," *ACM Sigcomm 84 Computer Communications Review,* Vol. 14, No. 2, June 1984, pp. 58-63.
6. Y. K. Dalal and R. M. Metcalfe, "Reverse Path Forwarding of Broadcast Packets," *Comm. ACM,* Vol. 21, No. 12, Dec. 1978, pp. 1040-1048.
7. D. W. Wall, "Selective Broadcast in Packet-Switched Networks," *Proc. Sixth Berkeley Workshop Distributed Data Management and Computer Networks,* Feb. 1982, pp. 239-258.
8. J. F. Shoch et al., "Evolution of the Ethernet Local Computer Network," *Computer,* Vol. 15, No. 8, Aug. 1982, pp. 10-27.
9. D. R. Cheriton and W. Zwaenepoel, "One-to-many Interprocess Communication in the V-system," *ACM Sigcomm 84 Computer Communications Review,* Vol. 14, No. 2, June 1984, p. 64.

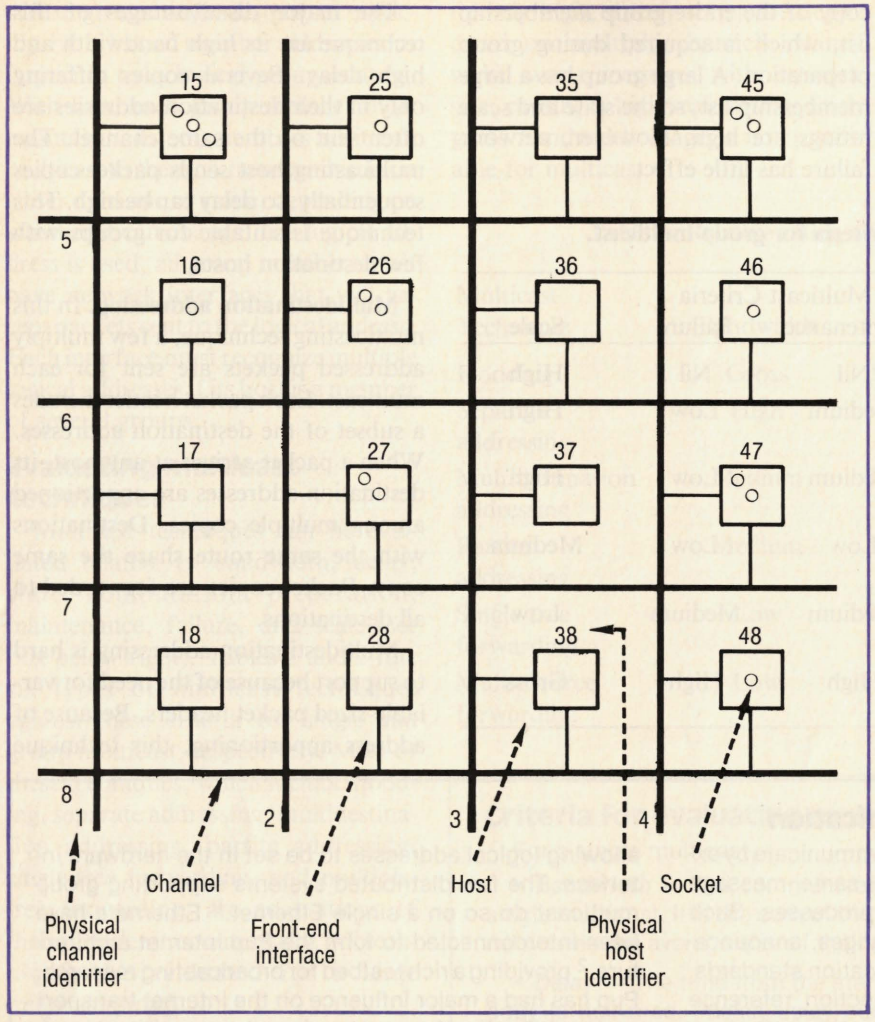


Figure 1. A sample netcomputer.

has a high computation rating and a medium delay rating. A minimal number of packets are sent, but bandwidth use is medium because of their large headers. The state, failure, and scale ratings equal those for separate addressing.

Partite addressing. This method is a combination of the separate and multidestination addressing techniques. Host destinations are partitioned by some common addressing locality, say, subnets or channels. Separate packets are sent to each partition for final delivery to all local hosts. During group preparation, each member receives a copy of the partition list. This technique is especially useful for broadcast internets (multiple Ethernets), with hosts partitioned by their channels.

This technique is highly resilient against failures. It has a medium state rating, since all members list only the channel destinations. It uses medium bandwidth, since several copies to different channels may be sent on the same initial channels. The delay is only medium because the multicasting host builds and sends fewer packets than in separate addressing. Adding hosts to the group may not increase the number of channel destinations, so the scale rating is medium. This technique is suitable for a group that resides on a small number of channels, even if there are many hosts.

Single- and multiple-tree forwarding. In this technique, a spanning tree is built for the hypothetical graph of network hosts connected by channels, and packet copies are forwarded along the branches. Each host member maintains and uses an image of only the local branches, simplifying the computations for forwarding. Three main types of tree structures have been investigated: shortest path, minimum spanning, and centered.

A shortest path tree is one with the shortest possible path from the root host to any other tree host. Multiple shortest path trees[2] minimize both delay and bandwidth but have a gross state rating, since each member has a separate tree. The preparation rating is gross for multiple trees because they are difficult to build in a distributed manner. Maintenance involves existing trees, so its rating is only high. Tree table space is proportional to the square of group size, so the scale rating is gross.

Another technique, called reverse path forwarding,[2] simulates multiple trees without actually maintaining them by using routing tables and two additional lists.[3] However, some packets may not be delivered if routing tables change during forwarding.

A minimum spanning tree[2] has the smallest total branch cost of all spanning trees. A centered tree[3] is a shortest path tree rooted at a host "in the center" of the group. A single tree minimizes both bandwidth and state but has a medium delay rating because not all paths may be minimal. However, a centered tree has only marginally better ratings than a minimum spanning tree. The preparation in building a single tree is high, but maintenance is medium and scale is low. Although sensitive to failures, trees form an excellent structure for group multicast. In general, trees have the lowest bandwidth and delay ratings. They are particularly suitable for point-to-point networks.

The major problem with host tree forwarding on broadcast internets is that separate packet copies are sent to each host, not to each channel. In addition, a spanning tree specifies an intermediate host between member hosts. The technique of using alternate paths

between members would be less failure-sensitive.

A good solution to both problems is to span a logical tree over channels and not over hosts. Packets can be forwarded to channels and then to hosts, as in partite addressing. The tree spans only member channels; physical paths between them are decided by runtime routing. (The use of channel-based trees is further described later.)

The netcomputer framework

A netcomputer is physically distributed like a local broadcast internet but provides its users a single virtual computer like a multiprocessor. It has three types of communication resources: channels, hosts, and sockets (Figure 1). A multiaccess broadcast bus with short transmission delays is referred to as a channel; a host is a processing node; and a socket within a host is a process address. Each resource type has physical and logical addresses. A physical identifier refers to a specific instance of a communication resource; a logical identifier refers to a group of them.

A broadcast bus supports the logical addressing of multiple hosts. Netcomputer addressing conventions, based on those for internets in Xerox network systems,[4] extend logical host addressing over the entire netcomputer, and provide for logical channel addressing. Each netcomputer address consists of channel, host, and socket identifiers. Since variable-sized packet headers can cause severe buffering problems, each packet header is assumed to have only one destination address.

Channels. A netcomputer can contain a large number of channels, such as Ethernets.[5] Each channel has a unique, 32-bit physical channel identifier in an unambiguous flat addressing scheme. Physical channel identifiers are used mainly as designators for netcomputer routing. The physical channel identifiers in Figure 1 are arbitrary.

A channel interfaced to a host via a transmission front end is said to be directly connected to that host; otherwise it is distant. In Figure 1, channel 8 is directly connected to host 18, and channel 3 is distant. A channel can be directly connected to many hosts; a host can have several directly connected channels. Figure 1 depicts a grid netcomputer in which each host has at most two directly connected channels, nominally vertical and horizontal.

Hosts. A netcomputer can contain a large number of hosts. Each host has a unique, 48-bit physical host identifier. Netcomputer physical host identifiers are absolute and implement a flat addressing scheme that is independent of the channel addressing scheme and can be used in the generation of other unique identifiers. Each host gives its physical identifier to all its attached front-ends as its host address. In Figure 1, each physical host identifier is conveniently chosen as the concatenation of its two physical channel identifiers.

A netcomputer is physically distributed like a broadcast internet, but has a single virtual computer like a multiprocessor.

Sockets. A socket represents a bidirectional port within a host that serves as a source and destination for packets. Packets can be both delivered to and transmitted from a socket. A host can support a large number of sockets. Each host can receive packets addressed to its directly connected channels.

Each socket has a 32-bit physical socket identifier that is globally unique but ephemeral. It is generated by incorporating the unique part of a physical host identifier. For a 48-bit Ethernet address, this part is 20 bits long. A logical socket identifier, which designates a group of one or more sockets in one or more hosts, is needed for all group members to receive a multicast packet on the same socket address.

Packet casting

Since a netcomputer can be configured with an arbitrarily large number of channels and hosts, it can be quite large and the packet transport mechanisms quite complex. Packets are transported by a network level functionally comparable to the OSI network layer,[6] but without the strict interface boundaries. Packets are transported across a netcomputer as datagrams.

Packet transport on a netcomputer is oriented toward channels. Packets are transmitted on, switched among, and routed to channels. Routing to channels and not to hosts provides an order of magnitude reduction in the routing information required on each host.[4] Another order of magnitude reduction can be achieved by partitioning routing information among all hosts on each physical channel.

A single channel provides an excellent architecture for multidestination communication but has limited extensibility. Providing multicast communication on a netcomputer is harder, since packet casting on distant channels requires support. Requirements for packet casting on a netcomputer may be as simple as packet transmission to a single destination host on a directly connected channel or as difficult as transporting packet copies destined for a group of hosts on several distant channels. The type of packet casting is based on the location of the destination channel relative to the sender: physical, directed, or logical.

Physical cast. Advanced transmission media such as Ethernet channels[5] directly support broadcast and multicast transmission. Physical cast is the transmission of a packet by a host on a directly connected channel and is triggered when the physical channel identifier of a directly connected channel is addressed.

Directed cast. Physical cast provides for packet delivery on only a directly connected channel. A packet may also need to go to a distant channel, which can be done if the network level pro-

vides routing services. Each host can transport packets to a distant channel by physically unicasting them to an intermediate host for relay.

In directed cast, a packet is cast to one distant channel.[7] This type of packet casting is used when sending a packet whose destination address contains a physical channel identifier of a distant channel. Directed cast is a generalization of physical cast. It is done by a series of packet unicasts through intermediate hosts toward its final destination channel. Any host receiving the packet on the destination channel uses a final physical cast to deliver it to the destination hosts on that channel.

Logical cast. Directed cast provides for packet delivery on only one channel. Multidestination communication sometimes requires the casting of packet copies on a number of physical channels. Here, the network level provides packet propagation, or forwarding, services. Forwarding services enable each host to propagate packet copies simultaneously to a set of physical channels.

Logical cast is a packet cast directed toward a localized set of physical channels associated with a logical channel identifier given as a destination address. It results in a series of directed and physical casts of packet copies. A host performing a logical cast sends packet copies to the physical channels in its set of forwarding destinations. If a destination channel is distant, its packet is encapsulated for relay through intermediate hosts. When a packet copy arrives at a host that is directly connected to a destination channel, a physical cast is done.

Figure 2 shows a logical multicast from host 17 that involves a directed multicast to channel 2 via relay 27 and several physical casts. A socket marks each group host: 15, 17, 18 on channel 1; 26, 28 on 2; and 36, 46 on 6. The directed multicast from 17 to channel 2 starts with a physical unicast to relay 27, followed by a physical multicast on channel 2 to members 26 and 28. A physical multicast from 17 to channel 1 reaches all three members there. Host 16 also accepts the packet to forward it on channel 6 by a physical multicast that reaches 36 and 46.

Group multicast on netcomputers

Since a netcomputer is also a packet-switched network, three group multicast techniques (separate addressing, partite addressing, and single-tree forwarding) are also suitable. Flooding, multidestination addressing, and multiple-tree forwarding are not acceptable because flooding has a gross bandwidth rating, multidestination addressing requires variable-sized headers, and multiple trees are too expensive to build and store.

The distance from a source host to a destination host is the minimum number of directly connected channels forming a path between them. The distance between two hosts on the same channel is one; between a host and itself, it is zero. For example, in Figure 2

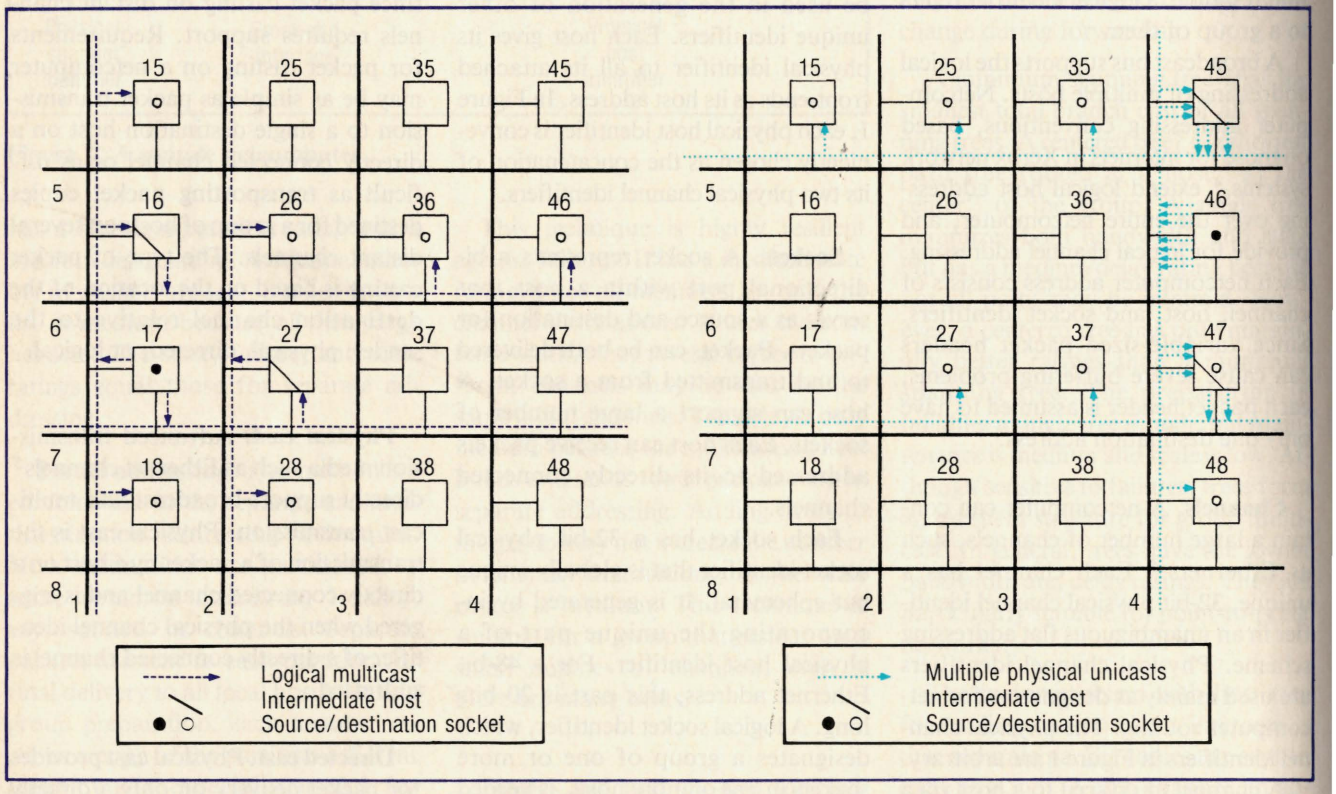


Figure 2. Illustration of logical multicast from host 17 to group of hosts 15, 17, 18, 26, 28, 36, and 46 on channels 1, 2 and 6.

Figure 3. Example of host multicast from host 46 to group of hosts 15, 25, 35, 46, 27, 37, 47, and 48 on channels 4, 5 and 7.

the distance between hosts 18 and 36 is two. The distance between a host and a destination channel is one more than the distance between the source host and the closest host on the destination channel.

The distance function is useful mainly for computing the bandwidth of a multicast. For packets with unit-sized headers, bandwidth is the sum of all channels traversed as part of a packet cast. The bandwidth of a physical cast is one, since only one channel is involved. The bandwidth of a directed cast is the distance between the source and destination. The total bandwidth of a series of directed casts is the sum of the individual bandwidths.

For delay computation, each packet sent or received is assumed to cause a delay of one time unit. Each host that has two directly connected channels can send and receive two unrelated packets in one time unit. However, receiving and then sending a packet copy results in a delay of two units. Multicast delay computations assume optimal packet casting order. Packets are sent to destinations in decreasing order of distance.

Host multicast. The separate addressing technique is used for host multicast. Processes communicate by multicasting to their group logical socket, which identifies the list of physical channel-host destinations. [In Figures 3, 4, and 5, the list is (4,46), (4,47), (4,48), (5,15), (5,25), (5,35), (7,27), and (7,37).] To multicast a packet, a separate packet copy is sent by directed unicast to each physical host in the list. Each destination address is composed of physical channel and host identifiers and the logical socket identifier. Each host in the group receives the packet destined to its physical host identifier and delivers it to the logical socket of the process group.

In Figure 3, a multicast from host 46 to the group requires 12 packets. There are two physical unicast packets to hosts 47 and 48 on channel 4. Of five directed unicasts, two are to hosts 27 and 37 through intermediate host 47 and result in four packets. The other

Definitions for multicast communication

broadcast* — the delivery of a packet to all possible destination addresses.

channel — a network communication link, especially a shared broadcast bus like Ethernet.

channel multicast — sending a multicast packet to a group of destination channels using partite addressing.

directed cast — a packet cast directed to a single distant channel by relay through intermediate hosts.

Ethernet* — a multiaccess communication bus, loosely including the interfaces attached to it.

flat address space* — a set of unique addresses with no separable component addresses.

flooding* — brute force broadcasting to all hosts in a network.

forwarding services — support for copying multicast packets onto local branches of trees used for logical multicast.

host* — one computer within a network of computers.

host multicast — sending a multicast packet to a group of destination hosts using separate addressing.

internet* — networks interconnected to form a packet-switching communication system.

logical address — an address for a group of communication resources that can be used to access any member of the group.

logical cast — a packet cast onto a group of channels addressed as a single logical channel.

multicast — the delivery of a packet to a specified subset of possible destinations.

multidestination addressing — multicasting by sending packets that are explicitly addressed to many destinations.

netcomputer — a network of computers controlled by a global distributed operating system and interconnected by broadcast networks.

packet cast — the sending of one or more packet copies to their destination(s).

partite addressing — multicasting by sending several singly addressed packets to clusters of destination addresses for final distribution.

physical address — an address for a single communication resource such as a host or channel.

physical cast — the transmission of a packet by a host on a directly connected channel.

routing services* — support for deciding which host should next receive a single packet for further relay to its final destination.

separate addressing — multicasting by sending a separate packet copy to each host destination.

socket* — a port, or end address within a host, that serves as a source and destination for packets.

tree forwarding — multicasting by sending a minimal number of packet copies along the branches of a tree that spans destination hosts.

tree multicast — sending a multicast packet along the branches of a tree that spans the channels of a group.

unicast* — the delivery of a packet to one destination address.

*Terms in common usage.

three directed unicasts to hosts 15, 25, and 35 through intermediate host 45 result in six packets. A maximum of 12 packets is needed for multicast from any host in the group for this example. Ten packets from host 47 are the minimum. In the example, the multicast delay is at best eight, since host 46 sends seven packets and the last packet must be received. The bandwidth and delay costs for host multicast in this example are thus 12 and eight.

Host multicast is simple to maintain but expensive to use. Although it employs only directed and physical unicast, it has high state and bandwidth ratings. Host multicast is useful when each group member needs an explicit list of all members for other purposes, when the host group is small, or when the group exists for too short a time to justify building a more complex multicast structure.

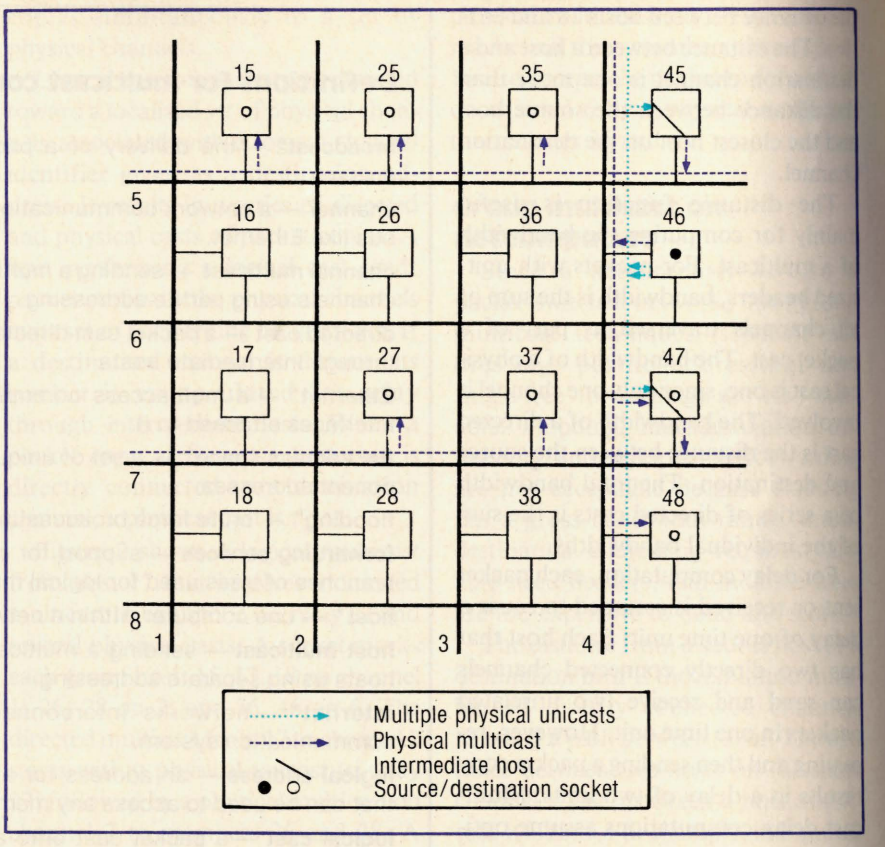


Figure 4, Example of channel multicast from host 46 to group of hosts 15, 25, 35, 46, 27, 37, 47, and 48 on channels 4, 5 and 7.

Channel multicast. The partite addressing technique is used for channel multicast. Each group host maintains only the list of physical channel identifiers on which all group hosts reside. (In Figure 4, the channel list is 4, 5, and 7.) Each host member must recognize packets addressed to the group's logical host identifier. To multicast a packet, a separate copy is sent to each physical channel in the list. Each destination address consists of the same logical host and socket identifiers but contains a different physical channel identifier. Physical multicast is finally used to deliver the packet to all group hosts on each group channel. Each host accepts the multicast packet and delivers it to the group logical socket.

In Figure 4, a multicast from host 46 to the group requires five packets, including one physical multicast on channel 4 and two directed multicasts. One directed multicast to channel 7 results in a physical unicast to host 47 and then a physical multicast on channel 7; the other multicast to channel 5 similarly uses two packets. Host 47 receives two packets: one for itself from the physical multicast on channel 4, and one for physical multicast on channel 7. At most five packets are needed and a minimum of four from host 47. From host 46 there is a delay of at least five to receive the last packet of the physical multicasts on channels 5 and 7. Both bandwidth and delay costs for this channel multicast are five.

Channel multicast is simple and efficient if all group hosts are connected to only a few physical channels. In particular, it is best for a single channel, since only one physical multicast is required. Channel multicast provides a more complex but efficient mechanism than host multicast—more complex because it uses directed and physical multicast, and more efficient because only a list of all channels, not of all hosts, has to be maintained and used by each host in the group.

Tree multicast. Logical multicast can be used to cast a packet on a previously built multicast structure, for example, a single shortest path tree. Logical trees are spanned over channels, not hosts. A logical tree is built using current routing information about group physical channels. With tree multicast, each packet header contains one logical channel identifier, designating one or more physical channels.

The principle of multidestination addressing can be used, for example, by a preparatory procedure to build a shortest path tree spanning the channel group. Using routing information, the initiating host creates several packet copies with the destination channel addresses partitioned among them as packet data. Addresses with the same intermediate route are placed in the same copy. Each outgoing packet is sent to the nearest channel of its subset.

To form the image of the local tree, forwarding services of hosts on each channel of the tree keep a list of all immediate destinations to which packets are forwarded. When each packet arrives at any host on its destination channel, multiple copies are again created to partition its destination ad-

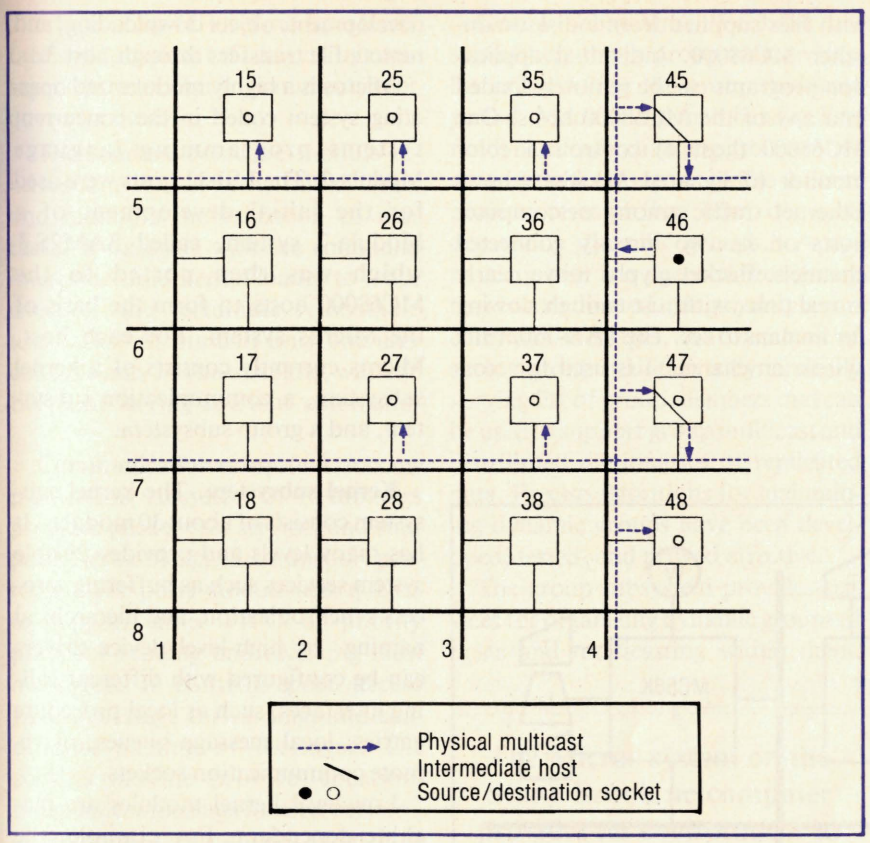


Figure 5. Example of tree multicast from host 46 to group of hosts 15, 25, 35, 46, 27, 37, 47, and 48 on channels 4, 5 and 7.

dresses. Packet copies are forwarded until received by hosts on all original destination channels. Hosts left with no more destinations are on channels that form the leaves of the spanning tree. Hence, the tree-building process ends.

Forwarding services on each host maintain the list of immediate parent and children channels that are the local image of the spanning tree. Tree multicast is accomplished by a series of logical multicasts of packet copies along the tree branches. Each packet destination address is composed of logical channel, host, and socket identifiers. Whenever a host on an intermediate destination physical channel receives a packet copy on one branch of its local subtree, it sends copies to the other branches of the subtree. When directed unicast is needed, the packet copy is encapsulated with a header for the physical channel destination and forwarded to an intermediate host. Eventually, the packet is delivered on all physical channels in the group.

In Figure 5, channel 4 is the tree root and channels 5 and 7 are its children. Hosts on channel 4 have the list 5 and 7. Hosts on channels 5 and 7 have the list 4. A multicast from host 46 to the group requires only three packets. There are three physical multicasts: one on channel 4 and two on channels 5 and 7, done by hosts 45 and 47, respectively. Three packets are both the minimum and maximum needed. The multicast delay from host 46 is four, since each packet copy to channels 5 or 7 needs to be sent, received by an intermediate host, relayed, and received at each final destination. The costs of tree multicast are thus a bandwidth of three and a delay of four.

Tree multicast is useful for long-lasting groups with members scattered over many physical channels. For an extended communication period, time is well spent in efficiently connecting group members. This type of multicast is more complex but more efficient than channel multicast. While it requires that the network level build and maintain a tree, it does not require that a full channel list be maintained on all group hosts. Only local forwarding information is needed. Tree multicast is also more efficient because only a minimal number of packet copies are sent for each multicast. Since the tree is spanned over channels and known to all hosts on these channels, it is failure-resilient.

Micros implementation

An experimental group communication subsystem has been integrated into the Micros operating system [1,8] on the Stony Brook netcomputer. The Micros project is exploring ways to organize netcomputers to solve large problems. Its main goal is to develop Micros into a portable, decentralized operating system that can effectively control many different netcomputers. It aims to produce cost-effective, high-throughput netcomputers that can solve large classes of applications, that extend easily to form more powerful systems, and that are always available to users at acceptable processing rates, even after component failure.

The underlying objective of the Micros project is to explore control and communication techniques for viable netcomputers with thousands of hosts. Micros is designed to be a highly modular, decentralized operating system that supports execution of distributed applications on netcomputers. Its design emphasizes portable, transparent control structures. Micros and the Stony Brook netcomputer form a research testbed that is now mature enough to provide a practical environment for studying distributed algorithms, languages, and applications.

The Stony Brook netcomputer is based on Motorola's MC68000 and Digital Equipment Corporation's LSI-11/23 hosts interconnected by 10M-bps Ethernet channels (Figure 6). The nine existing hosts are used as programming support workstations controlling one or two terminals each (designated *T* in the figure). Each MC68000 host has .25M or .75M bytes

of memory, but each LSI-11 has only 64K bytes. There are four Ethernet channels in the current configuration. Each host is connected to two channels at most.

Some hosts have Winchester disks (designated *W*) and some dual floppy disk drives (designated *F*), but two MC68000 hosts (8 and 9) have no attached disks. The flexibility of interfaces in Micros allows diskless MC68000 hosts to be booted remotely with files supplied from a disk on another MC68000. Individual application programs can be remotely loaded into any of the MC68000 hosts. One MC68000 (host 4) controls a color monitor (designated *M*) that shows Ethernet traffic among netcomputer hosts on its two directly connected channels. Packet glyphs move nearly in real time, with just enough slowing for humans to see. The VAX-750/Unix system on channel 1 is used for cross development, object downloading, and remote file transfers through host 3.

Micros is a highly modularized operating system coded in the concurrent systems programming language Modula-2. The LSI-11 hosts were used for the initial development of a Modula-2 system, called SAM2S,[8] which was then ported to the MC68000 hosts to form the basis of the Micros system. For each host, Micros currently consists of a kernel subsystem, a communication subsystem, and a group subsystem.

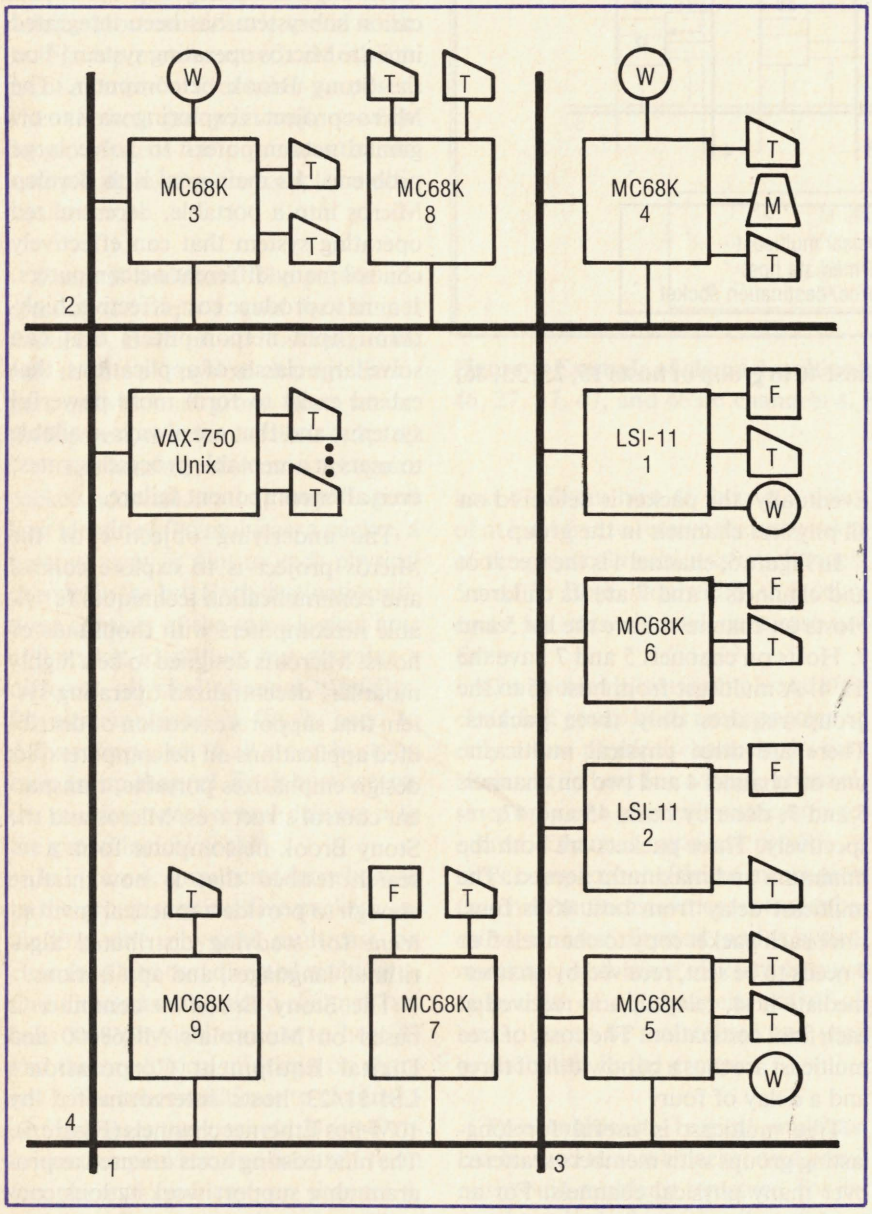


Figure 6. Stony Brook netcomputer of four Ethernets: *T* = terminal, *W* = Winchester disk drive, *F* = dual floppy disk drive, and *M* = color monitor.

Kernel subsystem. The kernel subsystem consists of about 30 modules. It has many levels and provides flexible system services such as buffering, process synchronization, and hierarchical naming. Its high-level device drivers can be configured with different calling interfaces, such as local procedure entries, local message queues, or remote communication sockets.

Low-level kernel modules are machine dependent. For example, the MC68000 module encapsulates the architecture of the Motorola 68000 microprocessor. It defines machine-specific trap and peripheral addresses that are also used by the tiny assembly-language subkernel. Abstract data types, such as lists, queues, and caches, are supported by modules shared by both the kernel and its users. The processes module provides the basic process type. Processes can be synchronized by use of services provided by the signals, gates, and semaphores modules. Processes share the same address space, as is essential for efficient communication software that avoids data copying. The spawning of processes forms tree hierarchies used for process control and termination.

I/O services are provided on three levels of abstraction: physical, logical, and virtual. Users interface at the virtual level for file and terminal operations. The virtual level passes user requests as procedure calls or messages to the appropriate I/O format module on the logical level. Logical device modules include handlers for serial terminals and various disk formats. They are independent of actual physical

interfaces. Logical level modules interface with physical I/O drivers by messages that use either communication or queue services.

The resident executive module receives control after kernel initialization and monitors the execution of user tasks. It interacts with the command interpreter and kernel loader to load, execute, and terminate relocatable user tasks. At present, only one user code file may be run at a time, with the file name serving as a load command.

Communication subsystem. A communication subsystem of 10 modules provides packet cast services and supports Xerox-like packet transport protocols. The ports module uses queues to support either FIFO or priority ports for sending and receiving local messages. It controls port access rights, message forwarding, and conditional passing of messages. The sockets module provides location-independent message transfer services, either locally within the same host computer or remotely between processes on different hosts.

To provide type uniformity for messages, ports and sockets directly manage carriers, which are standard headers for messages. Information in each carrier includes source and destination addresses, a unique message identifier, the message type, and a pointer to the message itself, if it exists. The messages module provides packaging facilities for marshaling and unmarshaling data into and out of packets.

For network communication, the network types module declares common addresses and services. The routes module on each host maintains a cache of local routing information similar to that used in Xerox network systems. The forks module on each host maintains a cache of forwarding information for the logical channels that the host recognizes on each of its directly connected channels. The transport module receives all the packets passed by its host or received by its front ends. It uses the routes and forks services to do the packet cast appropriate for each packet destination address.

Group subsystem. We have assumed here that a group membership list exists during the group multicast period. Maintaining membership lists for dynamically changing groups is not an easy task. As part of the Micros project, research has been done on organizing dynamic groups.[9] A dynamic group is a decentralized group that is coordinated using mainly asynchronous messages. All members of such a group maintain a dynamically varying list of group members that can be used to support group multicast and distributed dictionaries of replicated data. Precise algorithms for maintaining dynamic groups have been developed, tested, and proved effective.

The group subsystem provides services for organizing dynamic groups of hosts and multicasting within them.

The Micros system on the Stony Brook netcomputer provides an environment to experiment with different multicast techniques.

Each group is associated with a logical address to enable multicast to its members. The communities module provides for the organization of dynamic groups and supports the three types of group multicast (host, channel, tree) by using communication subsystem services. It uses the views module to maintain membership lists for all groups in which the host is a member. Members exchange views of the group to bring themselves up to date.

An interactive grouper program has been developed to test and demonstrate the maintenance of dynamic groups on the Stony Brook netcomputer. It allows the user to create and terminate groups. It permits queries about the status of groups maintained locally. For each group, the user can recruit and dismiss members, send membership list messages to other members, and multicast user messages to group members. Multicast messages received by the local host can be displayed on a terminal. The color monitor can display the groups to which hosts belong.

Performance results. The major design goals for the kernel and communication subsystems were interface flexibility and ease of experimentation; performance was only incidental. However, several timing experiments have been done to compare the costs of the various interfaces. For reference, average process switching time on a MC68000 host is 2.5 ms.

When single-character serial I/O calls were timed on an LSI-11, the serving of local I/O requests through socket interfaces took about two to three times as long as through queue interfaces. As a compromise between speed and flexibility, sockets are standardly used for logical level I/O interfaces, and faster queues are used for the physical level interfaces. A need for remote socket calls to low-level physical I/O drivers has not yet arisen.

Timing experiments were also done on communication services. Measurements for each operation were obtained as the operation executed 1000 times. For example, from the observed time of 56.6 seconds for 1000 messages, we can assume that sending a single message of 512 bytes through a socket would take 56.6 ms on average. The message and its carrier are copied into a packet by the socket manager, the route is determined by the transport manager, and the packet is delivered to the Ethernet driver. The packet is physically transmitted on the directly connected channel of the intermediate routing host. A self-addressed packet is sent last and retrieved to determine when all 1000 duplicated packets have been transmitted.

Unicasting an already prepared, 512-byte message directly through the Ethernet driver would take about 42.8 ms. The saving of an asynchronous interface to the transport manager to determine the route and update the carrier accounts for the 13.8 ms difference in the 56.6 and 42.8 ms times.

Similarly, the observed socket interface time to send a packet of 256 bytes

was 42.5 ms; for 128 bytes, 36.0 ms; and for 64 bytes, 31.8 ms. The corresponding times to send a packet directly to the Ethernet driver were 28.2 ms for 256 bytes, 21.2 ms for 128 bytes, and 17.7 ms for 64 bytes. Calls to both interfaces caused each packet to be copied twice. The observed times to process a packet are about 14 ms each in the socket interface and in the Ethernet interface. Hence, about 0.028 ms is necessary to copy each byte of data.

Several experiments centered on the group subsystem and its use of communication services. Two groups of hosts were used (see Figure 6): 3, 4, 8 and 3, 4, 5, 8. Hosts 3, 4, 8 reside on channel 2 and 3, 4, 5, 8 on channels 2 and 3. Creating and sending a 168-byte membership list for the first group from host 8 to host 3 on channel 2 took about 55 ms. Since it takes 38 ms to send a 168-byte unicast packet, we can deduce that membership message preparation takes about 17 ms.

Host multicast and channel multicast have also been compared. A host multicast of two 64-byte packets from host 8 to hosts 3 and 4, all on channel 2, took about 68 ms. A channel multicast of one packet to both hosts 3 and 4 took only 42 ms. Because the 1000 multicast packets in each burst sometimes come too rapidly for the double buffers in each 3COM Ethernet interface, only about 65 percent of these burst multicast messages are actually received.

If host 3 is the multicast source to hosts 4 and 8, the corresponding times are 82 ms for host multicast and 50 ms for channel multicast. The times for host 3 to multicast to the group are higher than those for host 8 because host 3 has one more Ethernet interface than host 8. Host 3 multicasting to hosts 4, 5, and 8 in the second group took 115 ms and 85 ms. The times for the larger group are each about 35 ms longer because one additional packet is needed in each case to reach host 5 on the second channel.

These experiments were run with a small number of hosts and channels. The disparity between host and channel casting times would be even greater in a system with more hosts on each channel.

Supporting group multicast on packet-switched networks requires efficient communication mechanisms. Network computer, or netcomputer, addressing conventions support logical addressing of communication resources over the entire netcomputer. The netcomputer resources of channels, hosts, and sockets provide a unifying communication framework for the three packet cast mechanisms: physical, directed, and logical. The key notion in the netcomputer framework is the importance of the channel as the major communication resource. Packet casting is channel oriented, for efficiency.

Three techniques have been proposed to provide efficient support for frequent multicast communication within dynamic groups of hosts linked by shared channels such as Ethernets. Host multicast is best for small groups, especially ones that feature infrequent communication, such as cooperating error recovery processes. Channel multicast is best for groups with medium to large numbers of hosts that reside compactly on a small number of channels. Tree multicast is best for very large or widely distributed groups with hosts scattered over many channels. The actual choice among host-based, channel-based, and tree-based group multicast techniques can be tailored to the needs of the application using the group.

All these mechanisms use the logical addressing and broadcasting mechanisms of the underlying communication network. A netcomputer is an effective distributed system architecture for the support of groups and group multicast. The Micros system on the Stony Brook netcomputer provides an experimental environment for investigating promising techniques. Experiments in distributed task force scheduling[10] and in management hierarchy recovery[11] are planned. The results obtained from Micros research should be applicable to many similar distributed environments. □

Acknowledgments

This article is a major revision of the paper, "Group Communication on Netcomputers," which was presented at the Fourth International Conference on Distributed Computing Systems in 1984. Research has been supported in part by NASA grant NAG-1-249, Army Research Office contract DAAG-29-82-K-0103 and instrumentation grant DAAG-29-84-G0011, Air Force Office of Scientific Research grant 81-0197, an external research grant from Digital Equipment Corporation, National Science Foundation Coordinated Experimental Research grant DCS83-19966, and equipment grants MCS80-06925 and MCS82-03955.

We thank Susan M. Hohner for technical assistance.

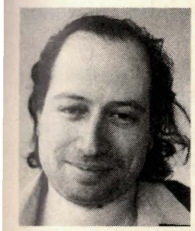
References

1. A. van Tilborg and L. D. Wittie, "Operating Systems for the Micronet Network Computer," *IEEE Micro,* Vol. 3, No. 2, Apr. 1983, pp. 38-47.
2. Y. K. Dalal and R. M. Metcalfe, "Reverse Path Forwarding of Broadcast Packets," *Comm. ACM,* Vol. 21, No. 12, Dec. 1978, pp. 1040-1048.
3. D. W. Wall, *Mechanisms for Broadcast and Selective Broadcast,* PhD dissertation, Computer Systems Laboratory, Stanford University, Stanford, Calif., June 1980.
4. Y. K. Dalal, "Use of Multiple Networks in the Xerox Network System," *Computer,* Vol. 15, No. 10, Oct. 1982, pp. 82-92.
5. J. F. Shoch et al., "Evolution of the Ethernet Local Computer Network," *Computer,* Vol. 15, No. 8, Aug. 1982, pp. 10-27.
6. A. S. Tanenbaum, *Computer Networks,* Prentice-Hall, Englewood Cliffs, N.J., 1981.
7. D. R. Boggs, *Internet Broadcasting,* PhD dissertation, Stanford University, Stanford, Calif., Jan. 1982 (also tech. report CSL-83-3, Xerox, Palo Alto Research Center, Palo Alto, Calif., Oct. 1983).
8. L. D. Wittie and A. J. Frank, "A Portable Modula-2 Operating System: SAM2S," *AFIPS Conf. Proc.,* Vol. 53, 1984 NCC, pp. 283-292.

9. A. J. Frank, A. J. Bernstein, and L. D. Wittie, *Maintaining Weakly-Consistent Replicated Data on Dynamic Groups of Computers,* tech. report 84/090, SUNY at Stony Brook, N.Y., Dec. 1984.

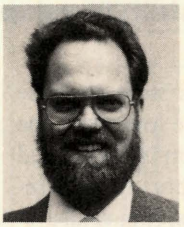
10. A. van Tilborg and L. D. Wittie, "Distributed Task Force Scheduling in Multi-Microcomputer Networks," *AFIPS Conf. Proc.,* Vol. 50, 1981 NCC, pp. 283-289.

11. C. K. Mohan and L. D. Wittie, "Local Reconfiguration of Management Trees in Large Networks," to be published in *Proc. Fifth Int'l Conf. Distributed Computing Systems,* May 1985.



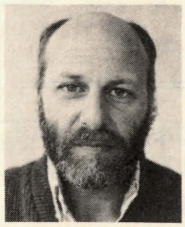
Ariel J. Frank is completing a PhD in the Department of Computer Science, State University of New York at Stony Brook. His research interests include distributed computing systems, communication networks, and concurrent programming.

Frank received a BSc in mathematics and computer science from Bar-Ilan University, Israel, in 1977, and an MSc in computer science from the Feinberg Graduate School of the Weizmann Institute of Science, Israel, in 1979.



Larry D. Wittie is an associate professor in the Department of Computer Science, State University of New York at Stony Brook and previously taught at Purdue and at SUNY at Buffalo. He is interested in the organization of large distributed systems, including portable operating systems, debugging systems for network computers, and decentralized control algorithms.

Wittie received a PhD in computer science from the University of Wisconsin at Madison in 1973. He is a member of IEEE, ACM, the Society for Neuroscience, and Sigma Xi.



Arthur J. Bernstein is a professor in the Department of Computer Science, State University of New York at Stony Brook. His current research interests are distributed algorithms, concurrent programming, and networks.

Bernstein received a PhD from Columbia University, New York, in 1962. He is a member of IEEE and ACM.

The authors' address is Department of Computer Science, State University of New York, Stony Brook, NY 11794-4400.