Proceedings, IEEE Infocom '95 Conference, Boston, Massachusetts, April 1995.

Address Management and Connection Control for Multicast Communication Applications

Alexandros Eleftheriadis, Sassan Pejhan and Dimitris Anastassiou

Department of Electrical Engineering
and Center for Telecommunications Research
Columbia University, New York, NY 10027

{eleft,sassan,anastas}@ctr.columbia.edu

Abstract An architecture and associated protocols are presented for managing multicast addresses and performing connection control for applications that use multicast communication facilities. A scheme to partition the multicast address space on the basis of the network number is proposed (an underlying IP-based internetworking environment is assumed), and its performance and scaling characteristics are discussed. A protocol is then developed to provide for dynamic allocation and release of multicast addresses, as well as maintaining state information for a connection. The protocol is independent of the address partitioning scheme, and hence is essentially applicable to any network layer; it is also shown to be robust and efficient. Finally, we describe two different mechanisms, namely address-based filtering and virtual port numbers, which enable the use of a common port number by all session participants.

Keywords: multicast communications, multicast address management, connection control, multiparty applications, multimedia communications

1 Introduction

With the rapid advances witnessed in the computing and networking technologies over the last decade, multimedia systems and applications, distributed over networks and involving several hosts, have become a reality. In addition to videoconferencing, such applications include group editing and shared workspaces (or groupware), multimedia on-demand services, and multimedia electronic mail.

Providing multicast capability at the network layer is extremely important in order to reduce the bandwidth requirements of multi-party, multimedia applications. Integral to such a capability are multicast addresses (i.e. a set of network addresses that designate a group of recipients), and multicast routing mechanisms. There are four main issues that have to be addressed by applications that make use of multicasting: 1) multicast address management, 2) connection control, 3) call management and 4) data transport. In this paper we are concerned with the first two issues only. We present an architecture and associated protocols that manage multicast addresses for the purpose of establishing and controlling multi-party sessions in an internetworked environment. We are not concerned with multicast routing algorithms, about which a great deal of literature exists (see [DEER88] and references therein), but assume it is provided at the network layer.

As the terminology in this area is not yet completely established, the difference between call management and connection control in the context of this paper should be explained. Call management is entirely up to the users and/or application; they define how they want to manage their session in terms of who has floor control, whether users need authorization to join in, etc... Connection control, however, is totally transparent to users, and includes mechanisms for hosts to maintain proper state information, even if host or network link failures occur. In our architecture, for example, this is achieved through the exchange of periodic refresh messages (see Section 3 for details). Some minimal cooperation of call management with connection control, transparent to users, is necessary so that communication parameters (e.g. multicast address, port number) are conveyed to the new participant(s) as they join a session.

In order to allow unlimited group membership without overloading packets with header information, a set of regular network addresses can be reserved for the purpose of designating multicast group addresses. This approach has been adopted in the IP environment [DEER89]. Even if a separate group addressing scheme is employed, the number of possible group formations ($2^N - N - 1$, where N is the number of available regular network addresses) is prohibitively large to allow full coverage[1]. Clearly, multicast addresses are a scarce network resource.

In contrast to regular network addresses, which are administratively allocated to unique hosts, multicast addresses have to be freely available. Thus, multicast addresses have to be dynamically allocated to applications, and returned to the network when no longer needed. Furthermore, the same address must not be assigned to concurrent sessions, as this will lead to interference or "cross-talk" [2]. In the IP environment, there is currently no such robust and scalable dynamic allocation

[1] $N = 2^{32}$ with the current (4-byte) IP addressing scheme, and will be increased to 2^{64} if IP addresses are expanded to 8 bytes.

[2] The same address can be allocated to different concurrent applications if restrictions are imposed on the

mechanism, and users have to exchange beforehand all relevant communication parameters to establish a session, using conventional means[3]. These limitations hinder further development and deployment of multi-party applications, and unnecessarily move the burden of network management operations to the users. Although the number of applications that currently utilize multicast addresses is rather small, the expected growth will demonstrate the need for — and require the use of — an effective dynamic address management scheme.

Although there have been a number of proposals for multicast transport protocols — such as XTP, ST-II, MTP, RTP, and extensions to the simple and heavily used UDP — they all assume that there exists some outside authority for allocating and managing multicast addresses [BRAU93]. Only in [BRAU93] has an actual architecture for multicast address management been proposed, although not in sufficient detail. In this architectural outline, the addresses are managed by a Multicast Group Authority (MGA) hierarchy, with a centralized controller as the root of an administrative tree. One significant drawback of this approach is that nodes closer to the root of the tree have to be able to sustain very high levels of control traffic. In addition, if any of the intermediate nodes or links in the MGA hierarchy breaks down, nodes below and above cannot exchange free multicast addresses. In Section 2, the evaluation of different multicast address management architectures will be put in more concrete terms.

Applications of a broadcast character that can potentially involve a large number of users (possibly thousands), can dispense with connection control due to the computational and bandwidth overhead associated with any distributed algorithm of such scale. This specific style of multi-party communication will be referred to as "open-style", due to the loose coupling among the participants[4]. Similarly, "closed-style" will refer to highly interactive applications in which both address allocation and connection control are effected. These will typically involve a limited number of users, due to inherent limitations in human interaction. An address management scheme should accommodate the need to support such diverse conferencing models.

---

geographical location of the paticipants (with the added benefit of using multicast addresses more efficiently). This restriction is not desirable, however, since many applications become more appealing as the distance between the participants increases.

[3]Software tools, however, that ease this task by "advertising" used addresses of ongoing sessions are available (Van Jacobson's `sd`).

[4]Most experimental software packages used today in the multicast-enabled part of the Internet (so-called "MBONE") fall in this category.

In order to enable multiple sessions per host, it is assumed that multiplexing is used at the transport layer. In IP networks this is accomplished with the so-called "port number" (a per-protocol entity), which is used to identify both sending and receiving parties (user processes). As explained in Section 4, multicast address management may require the use of a port resolution mechanism. The latter guarantees that at any given time, the same port number is used by all session participants.

The paper is organized as follows. Section 2 details the multicast address space partitioning scheme that our protocol employs, together with an analysis of its performance and scaling characteristics. Section 3 describes the operation of the address management protocol and analyzes its robustness in various cases of failure. Although we assume an underlying IP-based environment, the address management and connection control protocol is essentially independent of the network-layer used, whereas the address partitioning scheme is applicable to any network that uses hierarchical addressing. Section 4 describes two different port resolution mechanisms. Finally, in Section 5, we provide a summary of our results, discuss implementation details, and identify areas of future work.

2 Multicast Address Space Partitioning

Addresses in the current IP environment consist of four octets [POST81a]; the higher byte(s) of the address is (are) used to identify a network number, while the rest are used to locally identify an individual host. In order to allow the deployment of networks of various sizes, three classes are defined; each class uses a number of prefix octets (increasing from 1 to 3 for classes A to C) to specify the network number, with the remaining octets used for local host specification. Multicast addresses are defined as a fourth (D) class, and have the form[5]: `[224-239].X.X.X`, where `X` can have any value between 0 and 255 [DEER89]. The address range `224.0.0.X` is reserved for the use of routing and other low-level protocols.

A centralized approach in which a single entity would dynamically manage all these addresses is impractical. Consequently, any viable multicast address management scheme has to somehow

[5]The conventional dot-decimal notation is used throughout this paper.

effect a partitioning of the address space. Comparison of different schemes is not straightforward, as many parameters have to be taken into account. Two reasonable measures, however, are the blocking probability (i.e. the probability that a request for a free multicast address will be rejected) and the session setup delay (essentially the address acquisition or — possibly — rejection delay). The former measures address utilization efficiency, whereas the latter is indicative of the complexity of the allocation algorithm. It can be shown that a tradeoff exists between those two quantities: schemes with a centralized orientation will have lower blocking probability, whereas those dependent on distributed operation for the allocation of free addresses will perform better in terms of the acquisition delay.

The blocking probability of address requests can be obtained by modeling address management transactions (acquire and release) as an M/M/m/m system, where m is the number of available multicast addresses. We then have [SCHW87]:

$$P_B = \frac{\rho^m/m!}{\sum_{i=0}^{m} \rho^i/i!} \tag{1}$$

where λ is the rate of address requests, and $1/\mu$ is the average duration of a session. If we assume that the total number of multicast addresses is equally distributed into l subsets, each of which will serve only $1/l$ of the original number of hosts, P_B becomes:

$$P_B^l = \frac{(\rho/l)^{m/l}/(m/l)!}{\sum_{i=0}^{m/l} (\rho/l)^i/i!} \tag{2}$$

It can be shown [ELEF93] that P_B^l is decreasing with decreasing l, and hence that partitioning increases the blocking probability. Repeated application of this property also shows that this is true for any arbitrary segmentation of the multicast address space, in which addresses are assigned to network segments proportionally to their request rate.

In a distributed scheme, address requests within one network segment are serviced by addresses exclusively allocated to that particular segment. In a centralized scheme, addresses are allowed to freely "float" in the network, moved from one part to another on an as-needed basis [BRAU93]. In the former case the address acquisition (or rejection) delay will only include the time to obtain a free address from within the segment. In the latter case the delay will be greater or equal to that value, as in cases of depletion the allocation algorithm will have to attempt to obtain a free address from a different part of the network. The maximum delay, which occurs whenever all the nodes on the particular branch of the MGA hierarchy on which the requesting host resides are out of addresses, is equal to at least two traversals of the entire MGA tree plus two traversals of that branch. This is because the address request first propagates from the branch all the way to the root, which then sends out a request to reclaim all unused addresses to all the nodes in the tree. This request propagates down every branch, and the replies propagate back to the root, which then sends a free address down to the branch of the requester. Added to this delay is all the processing and queuing time required to handle all the responses at the nodes, especially near and at the root.

Since independent optimization of the blocking probability and the session setup delay is not possible, a balance between rigid segmentation and free propagation has to be sought. Other factors related to the structure of the underlying network also come into play. For example, implementation considerations would make it desirable that multicast address segments coincide with segments of administrative importance. In hierarchically structured networks, a network segmentation structure is already in place; this may restrict segmentation flexibility, especially in the leaves of the hierarchy.

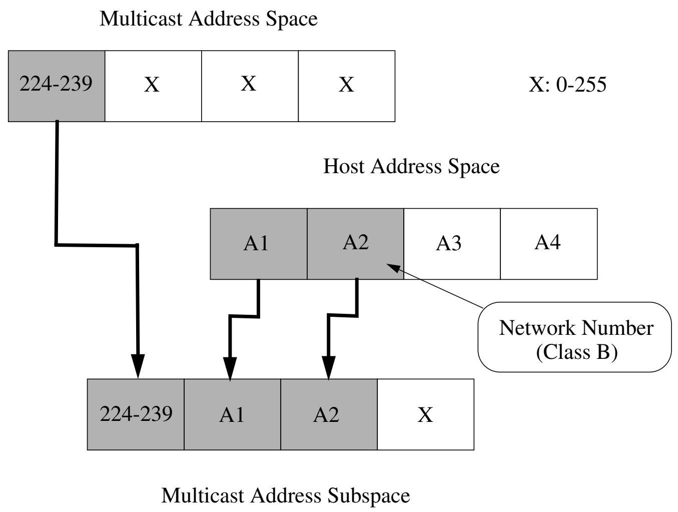

Figure 1: Multicast address subspace formation

The basis of our address partitioning scheme is the network (or subnetwork, if applicable)

Address Range	Usage	Connection Control	User Authentication	Allocation Mechanism
224.0.0.X	routing/low-level protocols	no	no	static
224.Y.Y.X	open-style sessions	no	no	static
225.X.X.X	open-style sessions	no	no	dynamic
[226-239].X.X.X	closed sessions	yes	optional	dynamic

X:0-255, Y:1-255

Table 1: Address space partitioning by administrative procedures

number. For a class C network, for example, the network number will consist of 3 octets (e.g. A1.A2.A3). Each network is assigned the management of all valid multicast addresses that have as a prefix the concatenation of a valid class D address first octet (224-239) and the network number. So the class C network, A1.A2.A3 will be responsible for managing the address space [224-239].A1.A2.A3 (excluding the reserved addresses). The address formation procedure for a class B network is shown in Figure 1.

In order to facilitate various types of conferences, it is useful to further segment the address space into areas of different administrative procedures. To support open-style sessions with a large number of participants (e.g. permanent network services like a radio multicast), the entire address range 224.X.X.X may be excluded from any address management protocol. Allocation of addresses in that pool (excluding the ones in the range 224.0.0.X) may be performed by a centralized entity (e.g. IANA), and have similar qualifications as FCC channel assignments. This will also provide a suitable environment for existing multi-party applications. For similar sessions but of temporary character, addresses in the range 225.X.X.X may be dynamically allocated, but with no user authentication or connection control[6] (see Section 3). Table 1 summarizes the proposed administrative partitioning, while Table 2 shows the number of addresses that have to be managed by each type of network.

We can now evaluate the blocking probability for the particular address space partitioning scheme described above. This probability will be highest for class C networks, where there are 15

[6]In this case the session initiator obtains a multicast address at the beginning of a session, and releases it at the end. An example would be the multicast of a temporary event, for which no permament address assignment is necessary. This is currently a very common use of multicast communication resources.

Network Type	Multicast Addresses	Excluding Reserved
Class A	1,048,576	983,040
Class B	4,096	3,840
Class C	16	15

Table 2: Managed multicast addresses per network type

addresses available for potentially 256 hosts. Figure 2 shows the blocking probability for a class C network as a function of its occupancy. Three curves are shown for different loads. The middle curve, $\rho = 0.05$, corresponds to a request rate of 1/host/hour ($\lambda_i = 1/60$) and average session duration of 3 minutes ($1/\mu$)[7].

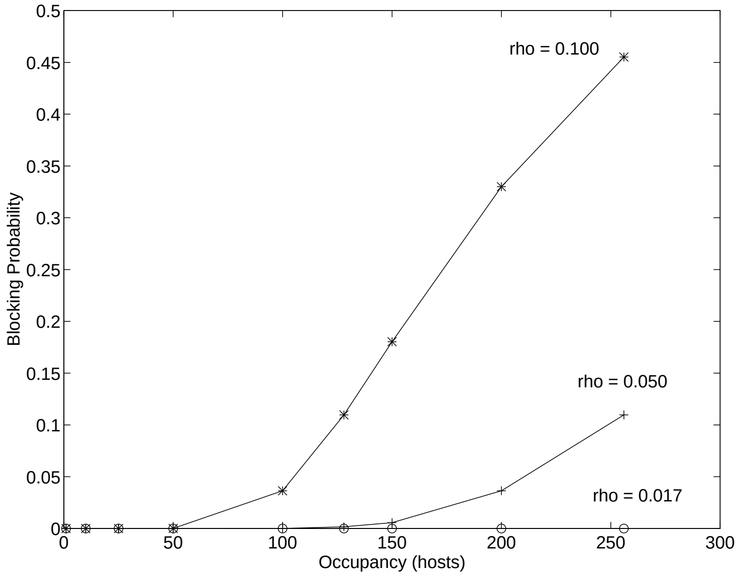

Figure 2: Blocking probability for class C networks for various loads

Multicast address requests originating from a specific network are serviced within that network. From Table 2 we see that the number of managed multicast addresses for networks of class B and especially A are inordinately large to be managed by a single entity[8]. This directly reflects the magnitude of the total number of available host addresses within each class, which in itself severely

[7]These numbers are meaningful for person-to-person voice communications.

[8]It has been observed that – based on deployment statistics – class A and B network address pools are too large (sparsely populated) whereas class C pools are rather small (densely populated). More efficient hierarchical addressing schemes are specifically addressed in IPng (next generation IP) proposals. Their increased efficiency will automatically carry over to the proposed multicast address space partitioning, with some minor modifications with respect to the address format assumed here.

impacts any network operation (not just multicast address allocation). In order to mitigate this problem, one has to resort to subnetting. With this technique, the original network is segmented into a set of smaller networks of a lower class (and possibly augmented by a router or bridge), with network numbers derived from the original one. This helps to ease administrative burden and avoid resource overloading, without sacrificing access to the desired address space for expansion purposes.

Within this context, the domain of responsibility of a group of multicast addresses is a set of subnets and not the entire network. The exact number of subnets involved depends on the specific network configuration and the mechanism with which multicast address management resources are discovered. By coupling subnetting and multicast address space segmentation, the scaling issues for classes A and B are resolved together with other problems of network operation and administration.

The process of subnetting may result in an incomplete coverage of the originally available address space. Consequently, there may exist multicast addresses that correspond to non-existing subnets. These can be assigned to existing address management entities in the network. In Figure 3 we show an example configuration of a class B network with two domains (sets of subnets). Subnet A's router is configured with broadcast packet filtering, whereas subnet B's router allows broadcast packets to cross subnet boundaries (but not towards the backbone). The multicast address space management configuration (available postfix octets) is shown for each managing entity, reflecting the differences in network configuration. The set of addresses within the gray area in subnet `128.59.62` correspond to non-existing subnets of the class B network (which could have equally well been distributed among two or more of the available address managers).

For the purposes of enabling dynamic allocation of excess multicast address pools, and also as a general address grouping facility, a "proxy" mechanism is described in Section 3. This scheme allows an address management entity to delegate responsibility to a designated alternative, and can significantly lower the blocking probability without seriously affecting the session setup delay (if it is not abused). In effect, the proxy mechanism achieves a compromise solution between the centralized scheme of [BRAU93] and the distributed scheme described above. The improvement in blocking probability is illustrated in Figure 4, where we compare the blocking probability for

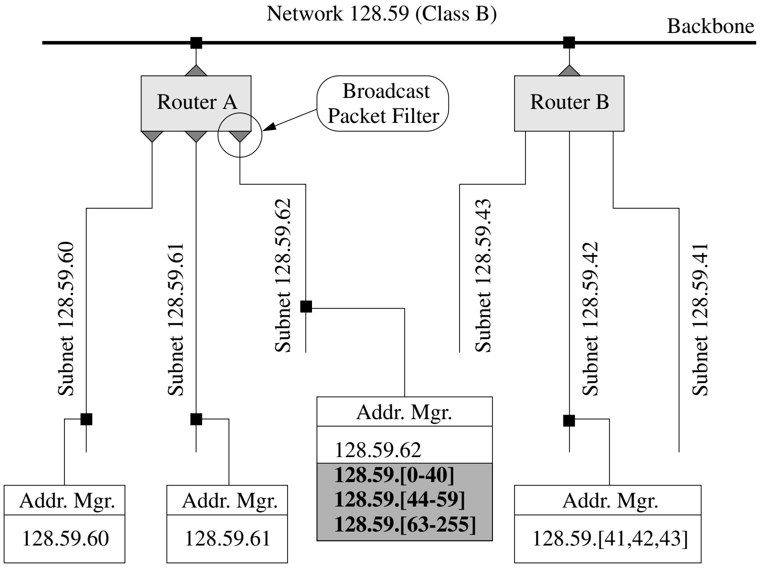

Figure 3: Example of multicast address space management configuration

a single class C network, with that of 2 and 3 class C networks that have been grouped via the proxy mechanism. Note that in all three cases, the ratio of hosts to available multicast addresses is the same. The rate of address requests, λ, is 1/host/hour, and the average duration of each session, $1/\mu$, is 3 minutes. The occupancy shown is normalized to that of a single network for comparison purposes.

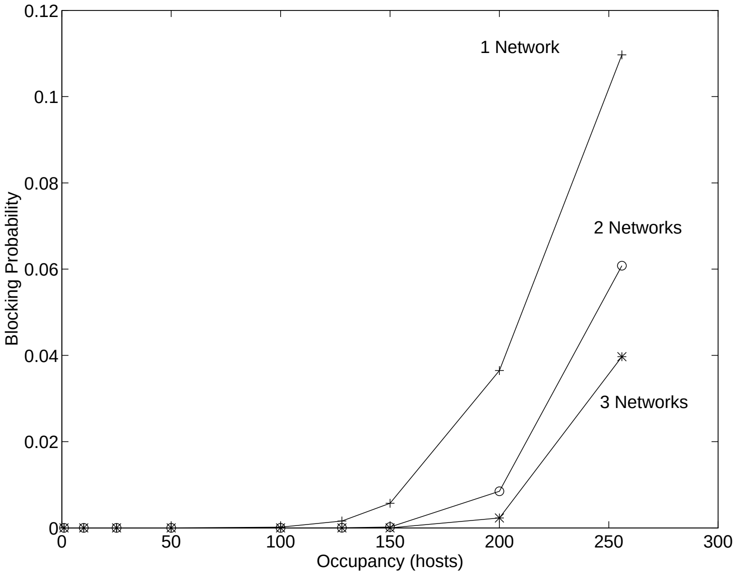

Figure 4: Effect of the Proxy Mechanism on the Blocking Probability of Class C networks

3 The Multicast Address Management and Connection Control Protocol

As described in Section 2, each network is allocated a set (or group of sets) of multicast addresses. These addresses are managed by a single entity of the network, called the Multicast Address Manager (MAM). The MAM is a process residing in any host on the network, and operating on a well-known port number. In addition to the MAM, there are a number of processes (perhaps as many as the number of addresses managed by the MAM, or even one on every machine) that are responsible for connection control. These Connection Controllers (CCs) reside on different hosts, again operating on a well-known port number. The multicast address management and connection control (MAMCC) protocol operates between the MAM, the CCs, and transport protocol entities at each host[9]. These latter entities will be referred to as users or session participants for brevity, although we should stress that actual users are not involved in any way in the operation of the protocol since it operates well below the application layer.

Upon receiving a user request, the MAM allocates an address, and then selects an appropriate CC which is delegated the task of controlling the corresponding session. The CCs are responsible for keeping and relaying state information about the session, processing join and leave requests from users, and returning freed addresses to the MAM upon session termination. By separating the tasks of address management and connection control, and by distributing operations among different hosts, a significant degree of robustness against both process and machine failures is achieved.

We first present the operation of the protocol under normal circumstances. We then analyze the robustness of the protocol by investigating scenarios where one or more of the system components fail, and show how the system can recover without causing any disruption to ongoing sessions. The only case where a session has to be disrupted is when the MAM and its CC fail simultaneously.

[9] Similar concepts for connection and address management are provided in [SCHO92], where a "Connection Manager" is employed on top of an application-level protocol; however, no specific algorithm and/or protocol is proposed for this purpose.

3.1 Normal Operation

We first present a general description of the MAMCC protocol, and then present analytical derivations of the time-out periods referred to in the text.

3.1.1 General Description of MAMCC

When an application requires the use of a multicast address, it broadcasts a MAM request[10]. This request is intercepted by the local MAM, which has to be unique in the local (sub)network. The MAM selects a free address[11], plus the least used (based on the number of sessions, the total number of participants, or a combination of the two) CC and conveys the multicast address and the address of the calling party to the CC. Once it receives an acknowledgment (ACK) from the CC (a two-way handshake), it conveys both the multicast address and the address of the selected CC to the calling party. In case all addresses are used, a negative reply is returned. In order to facilitate address grouping and/or address space extension, a third "proxy" reply may be given depending on the MAM configuration. In this case, the reply indicates the address of a MAM that has been specified as an alternate MAM server for this network[12]. As mentioned in Section 2, this helps to decrease the blocking probability.

After a successful address request, the calling party is then free to proceed with its call management protocol. In order to properly maintain multicast address usage information, the CC has to be included in the call management scheme: whenever a new party joins or leaves a session, the CC should be notified. These operations can be performed by the multicast support software[13] at the times the party requests to join or leave a multicast group, and hence be transparent to the application. The only difference with current practice is that in addition to the multicast address, the address of the CC must be supplied as well. This address will have to be communicated to

[10] If broadcasting is not supported by the physical network, then the address of the MAM will have to be specified.

[11] Possibly the least recently used, in order to protect new sessions from in-transit packets of previous sessions that used the same address.

[12] Multiple proxy indications may be given in sequence from one MAM to another, taking care not to form "closed loops". Note that a fully centralized management scheme could be implemented this way by having all MAMs use as proxy a single MAM.

[13] This software has the responsibility of manipulating physical multicast resources, and informing nearby multicast routers when the particular host it is running on has joined or left a multicast group. In IP there is currently no "leave" indication, and the multicast router utilizes timeouts to eliminate group memberships.

the participants through the call establishment process.

When all participants have indicated that they have left the session, the CC considers the address freed. After an appropriate timeout period it initiates a three-way handshake with the MAM, which will mark the address as available for reuse and disassociate it from the CC.

In order to maintain robustness against CC and user failures, special messages are exchanged periodically between the CC and each user. The CC requests each user to identify itself as an active participant at periodic and dynamically computed time intervals. The CC then considers the user lost when it has not received identification from the user within an appropriate period. Individual users employ similar timeouts; if no message has been received from the CC within a prespecified period, the CC is considered lost and the MAM is contacted to initiate the CC recovery procedure, as discussed in Section 3.2.2. To conserve network resources, the established multicast channel can be used for transmitting the "keep-alive" requests from the CC to the participants. Replies to the CC, however, are always unicast.

A similar mechanism is also used between the MAM and the CCs, where keep-alive messages are exchanged periodically, though less frequently than between the users and the CC. This is to keep the MAM informed of the status of the CCs and also to ensure that multicast addresses that are no longer in use, yet have not been reported as such to the MAM due to a CC failure, are freed up for reuse (see section 3.2.2). These keep-alive messages serve another purpose: each CC includes the total number of participants in all the sessions that it manages in the message that it sends to the MAM. This information is used by the MAM to determine the least used CC in response to a request for a multicast address.

3.1.2 Analytical Computation of Time-out Periods

We now describe the dynamic timeout value computation employed for keep-alive messages and in handshake operations, assuming a simple datagram service by the network layer (as in IP). Due to the large delay variations experienced on an internetwork, any a priori assumption in terms of round trip delays is inappropriate. Furthermore, the timeout values can have a significant impact on the responsiveness of the protocol with respect to the speed with which exceptional events are detected. Of particular importance here are events that propagate in higher layers of an application, and are directly visible to end-users (user additions and deletions).

To minimize the value of timeout periods and to assign meaningful values to its timers, the CC maintains round trip time (RTT) information. Since the CC may manage quite a large number of users, RTT information is maintained on a per-session basis. For each session, smoothed estimates of the average RTT (μ_{RTT}) and its variance (σ_{RTT}) are maintained. The estimation process is similar to that used in the TCP protocol [JACO88, LEFF89]. The keep-alive timeout value T_i for each session is set as the average smoothed RTT plus twice its smoothed variance. In case responses are missing upon expiration of the timeout, the new timeout value is set twice as large as the previous one. After a prespecified limit of back-offs, L, is reached (truncated exponential back-off), or if the overall timeout (i.e. the sum of the exponential back-offs) reaches a maximum $T_{\max}$, whichever is less, the users from which no response has been received are considered lost from the session. A relatively large value (e.g. 30 seconds) can be assigned to $T_{\max}$. It is essential, however, to impose an absolute upper bound to the timeout period in order to facilitate recovery from CC and MAM failures, as described in Sections 3.2.1 and 3.2.2. A minimum value for the timeout period must also be set (in the order of a few seconds), so that the volume of control traffic in sessions with small RTT is kept low.

In order to dampen the incoming traffic at the host (or network) where the CC resides, a smoothing period is used in the generation of acknowledgements (a similar scheme is used in [DEER89]). The keep-alive message from the CC to the users contains the value of an interval, in which responses from the users will be distributed. The users generate uniformly distributed random values within this interval and use them as timeout values for the transmission of the acknowledgement. The smoothing period, s_i, is computed on a per-session basis. Assuming that the CC server (or the network where it resides) can sustain an average – or assigned average – nominal rate of R incoming messages per second, and that the current number of sessions controlled by the CC is M, the smoothing period s_i for the i-th session is set to:

$$s_i = \frac{N_i M}{R} \tag{3}$$

where N_i is the number of participants in the i-th session. With the above formula, the average incoming message rate will be exactly R, regardless of the number of participants managed by the CC. This allows the connection control architecture to be scalable. The value of s_i should be added to the CC keep-alive timeout T_i. Also, since the smoothing period can introduce a bias in the average RTT estimate μ_{RTT_i}, the value of the random interval selected by the user has to be included in its acknowledgment and subtracted from the RTT estimate before updating μ_{RTT_i}.

As was mentioned earlier, the timeout values will directly affect the worst-case delay in adding a new user to a session. Using equation 3, equitable distribution of resources is guaranteed to all sessions, proportional to the number of participants. The precise value of R depends on the size of the messages, the hardware where the CC operates, and the capabilities of nearby routers. Note that for messages in the order of 100 octets, a value of $R = 1,000$ msg/sec is well within today's hardware capabilities (less than 1 Mbit/sec). The level in which smoothing may prove useful has to be verified with large-scale experimentation. Figure 5 shows how s_i can be varied to accommodate for a large number of hosts, for various values of R. With a smoothing period of ten seconds, for example, $10,000$ hosts can be controlled by the CC. As noted in the introduction, "open-style" broadcast sessions involving larger numbers of participants can dispense with connection control altogether.

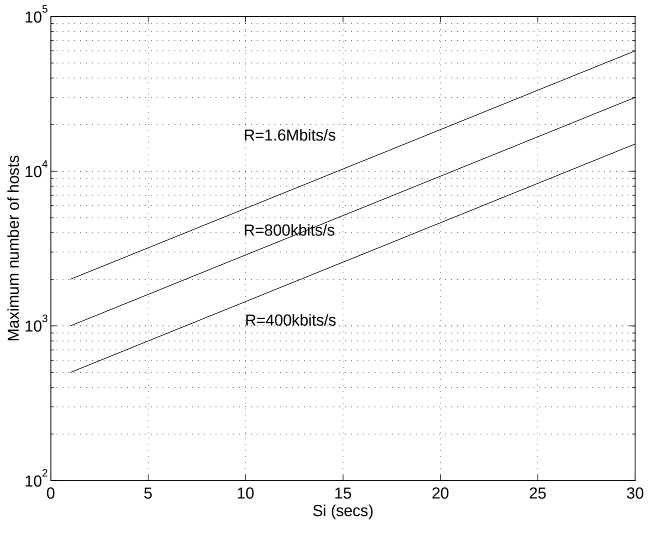

Figure 5: Maximum number of hosts handled by CC as a function of s_i and R

A timeout mechanism is also employed at the user side. Since the rate of generation of keep-alive messages from the CC is dynamically computed, this timeout value has to be supplied from the CC in the keep-alive message. Upon reception of such a message, the user starts the timeout with the supplied value T_i^{LOST}, where i designates the session. If upon expiration of the timeout no new message has been received from the CC, the user notifies the MAM so that the appropriate CC recovery procedure is initiated (see Section 3.2.2). The value of T_i^{LOST} is always set by the CC to the minimum of $T_{\max}$ and the sum of all the possible exponentially backed-off timeouts T_i, i.e.:

$$T_i^{\mathrm{LOST}} = \min\left\{T_{\max},\ \sum_{k=0}^{L} 2^k \left(\mu_{\mathrm{RTT}_i} + 2\sigma_{\mathrm{RTT}_i} + s_i\right)\right\} \tag{4}$$

The MAMCC protocol requires reliable communication of control messages (join, leave, address requests, and report of CC failure to the MAM). For this purpose, and again assuming a simple datagram service from the network layer, all requests by users are acknowledged by the MAM and the CC[14]. In order to mitigate the problem of duplicate packets, timestamps are always used. In this handshake scheme if no reply has been received from the MAM or CC for a specified period, the message is retransmitted, with a truncated, exponentially backed-off timeout value or until the keep-alive mechanism fails (if applicable). Since individual hosts do not maintain RTT estimates (at least not as part of the mechanisms proposed here), a reasonable starting value for the retransmission timer can be obtained from the CC in its last keep-alive message (if any), as the current value of T_i for the session. In the worst case, inaccuracy of this timeout value will just trigger duplicate messages that can be easily filtered out based on the message timestamps. The messages exchanged between two hosts, the MAM and the CC in a sample session is shown in Figure 6.

3.2 Error Recovery

By separating the operations of address and connection control, and distributing connection control among multiple hosts, a significant degree of robustness can be achieved. In the following we investigate cases where one of the MAMCC protocol components fails, focusing in particular on MAM and CC failures. It is shown that the only case where a session has to be disrupted is when its CC and the MAM fail simultaneously. Failure in the context of this discussion means that a component ceases completely to operate; in addition, it is assumed that it will be reinstated, but with total loss of information about its prior state.

[14]Control requests to the MAM and the CC and their corresponding replies are always unicast (with the exception of the initial address request which may be broadcast).

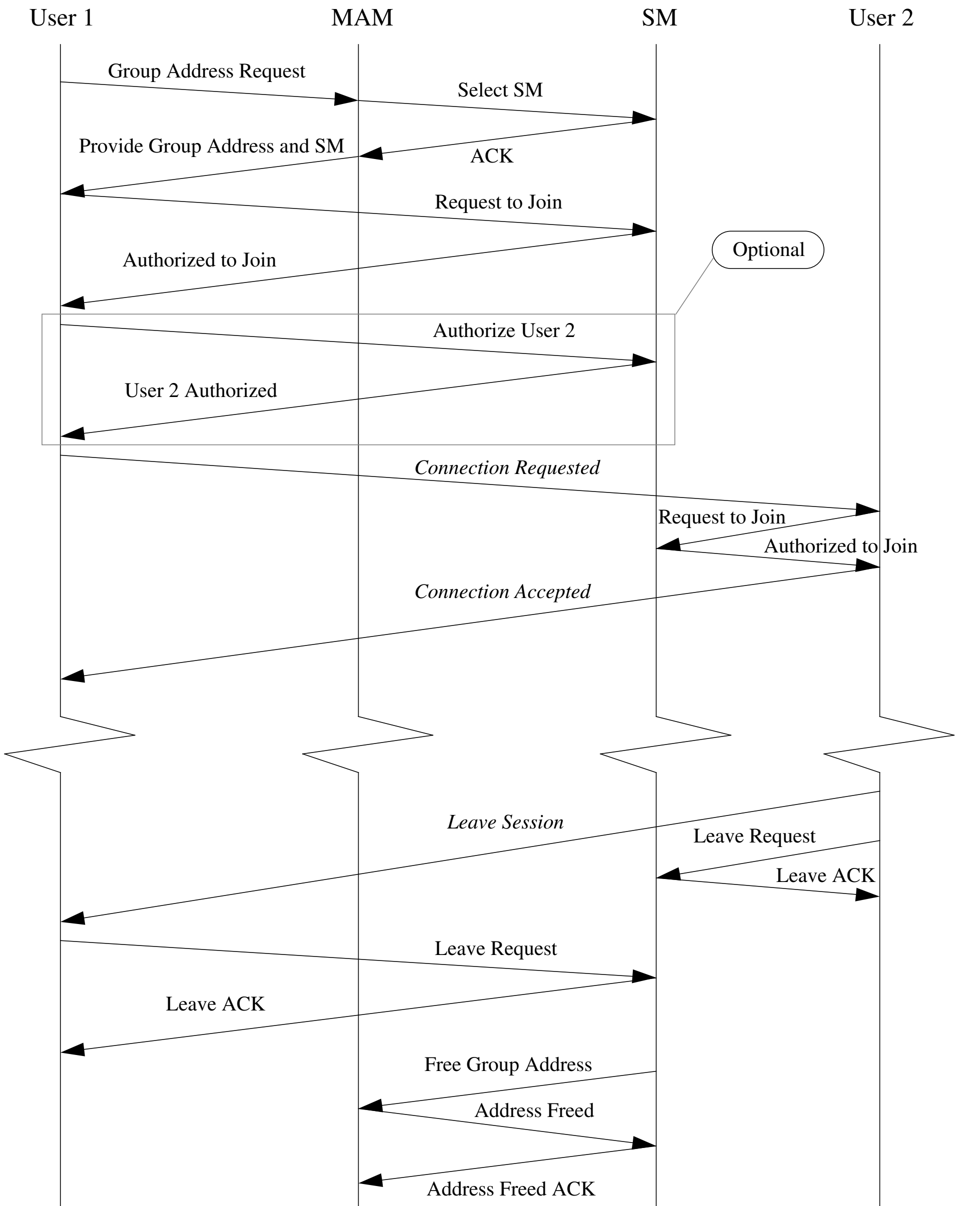

Figure 6: Message flow diagram for a sample session (messages in italics are not part of MAMCC)

3.2.1 MAM Failures

If a MAM fails, the sessions associated with it proceed as usual (with users still able to leave and join), since it is the CC that is in charge of connection control. New requests for multicast addresses, however, will be unanswered[15]. When a MAM is started or restarted after a failure, it sends a (multicast) query message to the CCs of the local network, which in turn reply by sending the multicast address(es) for which they are responsible and the number of participants in their session(s). As the replies to its query come in, the MAM will mark the CCs as enabled, and associate the appropriate multicast address(es) with each of them.

The MAM waits a period equal to twice $T_{\max}$ after it has sent a query message before assuming that a multicast address for which no CC has claimed responsibility is free. This is to cover the case where a CC fails just before the MAM restarts, and avoid allocating an address that is already in use. As explained in the following section, one way in which CC failures are brought to the attention of the MAM is by the session participants (the other is through the MAM-CC keep-alive procedure). The process of a user timing out a CC requires at most $T_{\max}$ from the time of failure, while $T_{\max}$ is also the worst-case time required for a user to subsequently notify the MAM (or time it out and leave the session).

3.2.2 CC Failures

CC failures are not fatal and their sessions can proceed normally. After the users timeout their CC, they will send a "CC not responding" message to their MAM, including their multicast address and the address of their now invalid CC. Upon reception of the first such message, the MAM will mark the CC as "not responding", disassociate the multicast address from the old CC, send a message to the old CC that it is relieved from managing this address[16], and select a new CC. After receiving an acknowledgement from the new CC, the MAM multicasts the address of the new CC to the users, which in turn send a "request to join" to their new CC. They can then begin

[15] It would be possible for the user to send its request to a MAM on another network. Since broadcasting would not — in general — be allowed across a network boundary, this would require a MAM discovery procedure which is outside the focus of this paper.

[16] This is to cover the case when one or more users time out the CC due to temporary network link failures, and not due to CC failure, and thus stop the CC from sending keep-alive messages.

their periodic keep alive messages after receiving the appropriate acknowledgement from the new CC. The new CC will have the correct state information within a period of at most twice $T_{\max}$. To deal with duplicate messages (from all the users after each one times out the CC), the MAM will ignore messages that refer to a CC that has already been disassociated with the multicast address indicated, and which has already been marked as disabled.

If the MAM detects the CC failure before the users through its own MAM-CC keep-alive mechanism, it will mark the CC as disabled, and go through the same process of selecting a new CC. Subsequent "CC not responding" messages from users will be filtered in the same way as duplicates, as described above.

The entire operation of timing out the failed CC, reporting it to the MAM, selecting a new CC and recovering its state is totally transparent to applications, and does not cause any disruption to communications. While the CC is down and the MAM is finding a new one, users may leave the session, but cannot join.

When a CC is started or restarted after a failure, it sends a "CC ready" message to the MAM, waits for its acknowledgement, and ignores any keep-alive messages that might still be directed to it. The restarted CC is marked as enabled, but is not reassigned any of its previous sessions, for which alternate CCs were/are sought. Subsequent messages from users regarding the CC being non operational are again filtered out by comparing the multicast address and CC that are being provided in the "CC not responding" message.

Finally, it is possible that the MAM is up but all the CCs are down. Upon receiving a new request for a multicast address, and noting that no CC is available, the MAM can reply with the address of a proxy MAM (if appropriate), just as it would in the case where it had no more free addresses. The proxy MAM would reside on a different network and would have a whole different set of CCs at its disposal. If, on the other hand, the MAM receives a "CC not responding" message and no CCs are available, it can either respond by indicating that all CCs are down (forcing the users to leave, and the session to end), or preferably start a temporary CC on its own host. Once it gets the first "CC ready" message from a recovered CC and has marked it as enabled, it will switch to the new CC, just as it would when selecting a new CC after one has failed in the middle of a session. The temporary CC can then be terminated. A proxy MAM cannot be used, since it would require a MAM to manage an address that was not assigned to it (with significant complications to the operation of the protocol).

The only case where communication will be disrupted and a session will have to be terminated is when the MAM and CC fail simultaneously. Users should leave the session after timing out both entities.

The failure scenarios described above will also cover temporary network link and node failures. It must be noted that any failure in the network (whether short-lived or not) that affects MAMCC operation, will necessarily result in disruption of communication between session participants.

4 Port Resolution Mechanisms

In order to support multiple connections per host, transport protocols provide means to multiplex different data streams from higher layers. In the IP environment, multiplexing is provided with the use of the port number. This number is used to identify both receiving and sending entities, and is an integral part of end-to-end addressing. Established network services are usually allocated well-known (reserved) port numbers within the transport protocol they use. This way, application programs or peer processes are always able to locate them.

For multi-party applications that use multicast routing, the situation is complicated by the fact that the same port number has to be used by all participants. Since the number of available port numbers is limited, allocating one per application is inappropriate. Moreover, due to the fact that multicast routing today is provided as an option in connectionless protocols (UDP), the coexistence of applications that use the same port number is complicated. A dynamic port selection mechanism has to be used, together with appropriate resolution procedures when there are conflicts among the participants (e.g. a session port number that is already used in a given host). Without such mechanisms, applications may receive traffic that was not destined to them, or unnecessarily lose part of their traffic. Today's practice requires the manual communication (using conventional means) among the participants of the port number that will be used for a session, together with the multicast group address.

It is assumed here that a call management entity exists in (or has access to) all hosts, to process incoming call requests etc. This entity will use a well-known, reserved port as an access point. As part of the call management operations, the multicast address and the port number of a session will have to be communicated to the new session participant. The exact way in which this is performed depends on the session model selected and implemented by the application. In the following, we describe two techniques that can be used to facilitate port sharing and mitigate port resolution problems: packet filtering and virtual port numbers. A third technique, consisting of a dedicated port resolution protocol, is described in [ELEF93].

4.1 Address-Based Filtering

Address-based filtering can be very easily performed at the transport layer. For multipoint applications, filtering consists of designating the multicast group address as the one with which this communication endpoint will be addressed. The transport layer will then automatically discard packets that have the same recipient's port number but a different address. This enables port sharing among applications that use multicast addresses. A significant drawback of this approach is that there exist already many applications that were designed with the assumption that multiple host addresses could only be attributed to multiple network interfaces. Consequently, and for ease in development, they utilize "all-pass" filtering at the transport layer. These applications completely prohibit port sharing. Although this technique requires no changes to actual transport mechanisms, extensive (although simple) changes need to be performed to a large number of existing applications.

4.2 Virtual Port Numbers

Another possible technique is the use of Virtual Port Numbers (VPNs). With this scheme, the VPNs are used for addressing purposes at the transport layer similarly to current port numbers, but a mapping function is used when necessary to identify the receiving application. The mapping function is applied after reception of data (VPN to actual port numbers (APN)) and associates a multicast group address with one or more VPN-APN pairs. Note that APN to VPN translation during transmission is not performed, since a unicast reply by the recipient would not trigger the inverse mapping. For existing applications, this function would be the identity, i.e. the VPN would be equal to the APN. Applications using multicast facilities would establish such mappings as part of the group join process.

This approach has the advantage of maintaining compatibility with existing distributed applications, while requiring minor changes in the transport protocol implementation. In addition, it has minimal overhead as far as connection control is concerned.

5 Concluding Remarks

We have presented an architecture and associated protocol for managing multicast addresses and performing connection control of multi-party sessions in an internetworked (IP-based) environment. The architecture consists of an address space partitioning scheme based on the network number, a Multicast Address Manager (MAM), and a number of Connection Controllers (CCs). The MAM is a process residing in each (sub)network, and has the responsibility of managing multicast addresses via mechanisms specified by the MAMCC protocol. The CCs are processes residing on any host in a (sub)network, and are delegated connection control responsibilities by the MAM. To facilitate various types of multi-party applications with different needs, further administrative partitioning of the multicast address space is also proposed.

The MAMCC protocol operates between the MAM, CC, and end-user multi-party applications at the transport (or higher) level. It provides for dynamic allocation and release of multicast addresses, and connection control. A proxy mechanism is used to facilitate address sharing among different parts of a network. This enables a MAM to delegate address allocation requests (and management responsibilities) to another one, located outside its own network and possibly controlling a larger address pool. The protocol is resilient to MAM, CC, and end-user errors, and is designed to be adaptable to varying network conditions (round trip delays). The MAMCC protocol is independent of the address space partitioning scheme used, and is applicable to any network layer that supports simple datagram operation.

We have identified two techniques to ensure the use of a commonly available port number by all session participants. Of the two techniques proposed, address-based filtering and virtual port numbers, the second is the more efficient as it has no communications overhead and maintains compatibility with existing networking environments.

We should note that large scale experimentation in actual sessions is necessary to identify potential improvements and perform fine-tuning of the protocol. Also, the MAM and CC can be easily augmented with more functionality, such as membership reporting or other application-oriented operations. An important addition to the MAMCC services would be a call management server that would provide diverse call management facilities to applications. This server should have similar responsibilities as the Remote Procedure Call (RPC) port mapper, but support human communication semantics (e.g. printed notification messages, peer application notification etc.), a symmetric (rather than client-server only) model of operation, and a simple event-driven application interface. The availability of these facilities will significantly reduce the burden of designing and implementing multi-party applications, as most of the complexities involved will be handled by available network services.

References

[BRAU93] R. Braudes and S. Zabele, "Requirements for Multicast Protocols," RFC 1458, TASC, Reading, MA, May 1993.

[DEER88] S. E. Deering, "Multicast Routing in Internetworks and Extended LANs," in *Proceedings of the ACM SIGCOMM '88 Symposium*, pp. 55–64, August 1988.

[DEER89] S. Deering, "Host Extensions for IP Multicasting," RFC 1112, Standford University, 1989.

[ELEF93] A. Eleftheriadis, S. Pejhan, and D. Anastassiou, "Multicast Group Address Management and Connection Control for Multi-Party Applications," Technical Report CU/CTR/TR 351–93–31, Center for Telecommunications Research, Columbia University, November 1993.

[JACO88] V. Jacobson, "Congestion Avoidance and Control," in *Proceedings of the ACM SIGCOMM '88 Conference*, pp. 314–329, August 1988.

[LEFF89] J. Leffler, M. K. McKusick, M. J. Karels, and J. S. Quarterman, *The Design and Implementation of the 4.3 BSD UNIX Operating System.* Addison-Wesley, 1989.

[POST81a] J.Postel, "Internet Protocol," RFC 791, USC/Information Sciences Institute, 1981.

[SCHO92] E. M. Schooler, "The Impact of Scaling on a Multimedia Connection Architecture," in *Proceedings of the 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 302–307, November 1992.

[SCHW87] Mischa Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis.* Reading, MA: Addison-Wesley, 1987.