Distributed Multicast Address Management in the Global Internet

Sassan Pejhan, Alexandros Eleftheriadis, *Student Member, IEEE,*
and Dimitris Anastassiou, *Senior Member, IEEE*

Abstract— **We describe a distributed architecture for managing multicast addresses in the global Internet. A multicast address space partitioning scheme is proposed, based on the unicast host address and a per-host address management entity. By noting that port numbers are an integral part of end-to-end multicast addressing we present a single, unified solution to the two problems of dynamic multicast address management and port resolution. We then present a framework for the evaluation of multicast address management schemes, and use it to compare our design with three recently proposed approaches, as well as a random allocation strategy. The criteria used for the evaluation are blocking probability and consistency, address acquisition delay, the load on address management entities, robustness against failures, and processing and communications overhead. With the distributed scheme the probability of blocking for address acquisition is reduced by several orders of magnitude, to insignificant levels, while consistency is maintained. At the same time, address acquisition delay is reduced to a minimum by serving the request within the host itself. It is also shown that the scheme generates much less control traffic, is more robust against failures, and puts much less load on address management entities as compared with the other three schemes. The random allocation strategy is shown to be attractive primarily due to its simplicity, although it does have several drawbacks stemming from its lack of consistency (addresses may be allocated more than once).**

Keywords— **multicast address management, port resolution, multipoint applications, multicast communication, internetworking**

I. Introduction

Recent gains in bandwidth and processing power have made multipoint communications technically feasible, while applications such as videoconferencing and group editing have made them desirable. To support multipoint communications, especially high-bandwidth real-time applications, it is very important to provide multicast capability at the network layer.

In IP networks, thousands of which are linked together over the global Internet backbone, a subset of the address space has been reserved for multicast groups [6]. Its size represents 1/16th of the total address space, and a much lower ratio of the total number of possible group formations ($2^N - N - 1$, where N is the total number of hosts). For this reason, multicast addresses cannot be permanently assigned to particular applications or group combinations, but need to be available for re-use: at session initiation a free address is selected, which is returned to the pool at session termination. This requires a dynamic multicast address allocation and release mechanism which is transparent to users. Furthermore, the same address must not be assigned to concurrent sessions (consistency), as this will lead to interference or "cross-talk"[1]. Although this could be considered acceptable if its probability of occurrence is extremely small, we believe that it is undesirable for most communication applications. Finally, these goals must be achieved within certain constraints such as low blocking probability (probability that a request for a multicast address is rejected), low delay, high reliability (robustness), low complexity and low control traffic overhead. Hence the need for an efficient and robust multicast address management mechanism. We should note that if we dispense with the consistency requirement, the problem is significantly simplified.

Although there have been a number of proposals for multicast transport protocols—such as XTP [1], MTP [2], RTP [19], and the simple and heavily used UDP—they all assume that there exists some outside authority for allocating and managing multicast addresses [3][2]. Currently, however, there is no mechanism for the management and dynamic allocation of multicast addresses (with the features described above) within the Internet, and hosts have to communicate all relevant communications parameters (such as multicast address and port numbers) a priori. Alternatively, a session can be continuously "advertized" throughout its duration, with peer programs providing the relevant information to the user. Popular experimental software packages used in the multicast-enabled part of the Internet (MBONE) fall in this category. The Session Directory (SD)[3] multicast address allocation program that is widely used today follows this approach, but is not general enough to satisfy the requirements for a multicast address management mechanism, as is further discussed in Section IV-C.

Recently, a couple of schemes have been proposed [3], [10]. In [3] an architecture has been introduced where multicast addresses are managed by a tree-structured Multicast Group Authority (MGA). Such a scheme, however, suffers from large set-up delay, and is not very robust. In [10] the multicast address space is partitioned according to (sub)networks. This reduces the acquisition delay, as requests are served within the particular (sub)network, but

[1] The same address can be allocated to different concurrent applications if restrictions are imposed on the geographical location of the participants (scoping). This is not desirable, however, as many applications become more appealing as the distance between the participants increases.

[2] ST-II [21] provides the option of using IP multicast addresses (in which case an outside mechanism for multicast address management is needed), but most implementations use lists of unicast addresses.

[3] The SD program has been developed by V. Jacobson, Lawrence Berkeley Laboratories).

increases the blocking probability. To strike a balance between the two, a "proxy" mechanism is used, whereby addresses are shared between any number of (sub)networks. Finally, a scheme for managing multicast addresses has been proposed in [22], but only in the context of a local area network. The merits and drawbacks of all three schemes, as well as those of random address selection, are discussed in more detail in Section V.

The contribution of this paper is three-fold. Firstly, we propose a new, distributed architecture for multicast address management within the global Internet. Secondly, our scheme incorporates a solution to the problem of port resolution[4]—which is inherently linked to that of multicast address management—thereby providing a unified solution to both problems. Except for [10], none of the other schemes implement port resolution. Finally, we present a framework for evaluating the performance of multicast address management schemes, and use it to compare our architecture with those mentioned above.

The mechanism proposed here achieves what appear to be two incompatible goals of (consistent) multicast address management: it reduces the blocking probability by several orders of magnitude while reducing the set-up delay. Furthermore, it does not increase the complexity of multicast communications, as it only requires simple changes to be made to the transport and multicast routing protocols. If anything, and due to its distributed nature, it significantly reduces the complexity of the address management task compared to a hierarchical [3] or semi-centralized scheme [10]. At the same time, it allows for the reduction of the number of addresses reserved for multicasting, and hence for temporary relief from the IP address exhaustion problem.

The paper is organized as follows. In Section II we describe the concept of virtual port numbers and how they may be used both as a port resolution mechanism, and to extend the multicast address space. In Section III, we describe an architecture for fully distributed multicast address management, using the concept of virtual port numbers and extended multicast addresses. Brief descriptions of the other alternatives proposed in the literature are given in Section IV. Section V compares the performance of the different address management schemes discussed, on the basis of blocking probability and consistency, address acquisition delay, load on address management entities, overall robustness, and processing and communications overhead. In Section VI we briefly discuss the applicability of the proposed architecture in the context of non-IP networks, namely OSI CLNP and the IPng (IP Next Generation) proposals (TUBA, CATNIP, and SIPP). Finally, in Section VII we present our concluding remarks.

II. Virtual Port Numbers

Addresses in the current IP environment consist of four octets [15]; part of the address (prefix) is used to identify a network, while the rest (suffix) is used to locally identify an individual host[5]. In order to allow the deployment of networks of various sizes, three classes are defined, each allowing a different number of hosts. Each class uses a number of prefix octets (ranging from 1 to 3 for classes A to C) to specify the network, with the remaining octets used for local host specification. Multicast addresses are defined as a fourth (D) class, and have the form:[6] {224–239}.X.X.X, where X can have any value between 0 and 255 [6]. The address range 224.0.0.X is reserved for the use of routing and other low-level protocols.

A. Use of Virtual Port Numbers for Port Resolution

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 port numbers. These numbers are used to identify both receiving and sending entities, and are an integral part of end-to-end addressing. They are used by the transport protocol at the receiver, in order to identify the appropriate recipient. Sixteen bits are used for the port number, yielding a pool of 65,536. Of these, 1023 are reserved by the Internet Assigned Numbers Authority (IANA) for allocation to special, well known processes [17]. In addition, many ports are used for ordinary user processes on most systems, and are assigned by the local systems themselves.

For multicast applications the situation is complicated by the fact that the same port number has to be used by all participants. Since this port number is randomly selected for each application, there is no guarantee that it will be available on a given host. For this reason, port numbers may have to be dynamically allocated/negotiated. Resolving the problem of finding a common port number is thus a necessary step in the call set-up phase of multicast communications, regardless of the address management scheme used. As with multicast addresses, current practice requires prior communication of the port number that will be used for a session among the participants, or use of scoped advertizing as in SD. In either case, there is no mechanism to resolve conflicts.

A number of port resolution mechanisms have been proposed in [10]. One of these adds a port mapping mechanism to the transport protocol, relating Virtual Port Numbers (VPRs) to Actual Port Numbers (APNs). The mapping associates multicast group addresses with VPN-APN pairs, and is established as part of the connection set-up or group join process. The session initiator selects the common port number (virtual) to be used. Users joining in will have to map that port number to a free one. Every time a packet with that particular multicast address is received, the mapping (VPN to APN) will be performed to send the packet to the correct application.

Although this scheme was proposed as a solution to the

[4] This is a necessary step for multicast communications, as discussed in Section 2.1.

[5] With the proposed CIDR repartitioning of the IP address space, part of the network number also indicates a routing domain [16]; this has no effect on our scheme, which operates at a higher level.

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

port resolution problem independent of the address management scheme employed, it can be used as the basis of a fully distributed address management scheme. This provides a single solution to both address management and port resolution (Section III).

An alternative mechanism that was discussed in [10] is packet filtering; this is performed at the transport layer, by associating (binding) port numbers to particular IP addresses. This is currently not possible for multicast addresses, since binding is only allowed for local interface addresses. Modifications to operating system kernel code to remove this restriction are currently available for some platforms, although they are not yet widely deployed.

B. Virtual Port Numbers as an Extension of Multicast Addresses

Since for multicast transmission, all participating hosts are required not only to use a common 4-byte group network address, but also the same 2-byte port number, we can view the end-to-end multicast address as a 6-byte entity. Using mechanisms that are described in Section III, the multicast group address and port number of a session will be communicated to each participating host. Hosts will keep a VPN-APN mapping table, used for multicast addresses only (since multicast addresses begin with 1110, the transport protocol can immediately decide whether to use the mapping table, even before reading the rest of the address). Each time they join a multicast group, they must perform a VPN-APN mapping and add the appropriate entry in the table, even if the port number to be used is free and available (in which case the mapping would be the identity). Upon leaving the session, the entry will be deleted from the table. The entries must be arranged according to the multicast addresses. Two sessions using the same multicast address but different port numbers will be sent to the correct applications, since there will be two different entries in the mapping table for that multicast address (one for each port number). If the host is a participant in only one of those sessions, the one without an entry in the mapping table will be filtered out[7].

In the following section, we consider the multicast address and port number combination as a single entity for multicast address management, port resolution and multicast routing purposes.

III. Distributed Multicast Address Management

We briefly describe the approach of [10] and then describe the proposed distributed scheme, since the latter is a direct extension of the former.

A. Semi-Distributed Multicast Address Management

In the address management scheme described in [10], a Multicast Address Manager (MAM) is responsible for the assignment of multicast addresses within a network or subnetwork. The MAM is assigned a set of addresses according to an address space partitioning scheme based on the (sub)network number. Each MAM 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 MAM residing on the class C network A1.A2.A3, for example, will be responsible for managing the address set {224–239}.A1.A2.A3—16 addresses, or 15 if we reserve the pool 224.X.X.X for "open-style" conferences [10][8]. The address formation procedure for a class B network is shown in Figure 1.

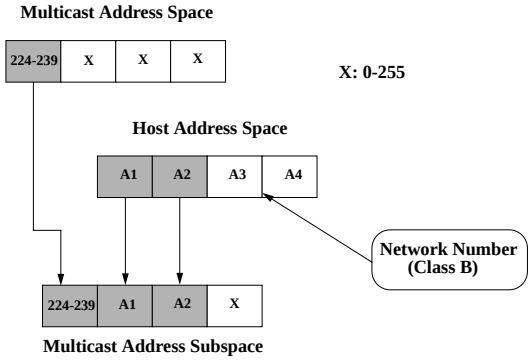

Fig. 1. Multicast address subspace formation in [10]

The MAM allocates a free multicast address but it is the session initiator which initially selects the common port number to be used. If virtual port numbers are used for port resolution, then users joining in will have to map that port number to a free one.

In addition to the MAM, there is one Connection Controller (CC) on each host which is responsible for connection control, handling join and leave operations during a session, and maintenance of state information for each session. This is done through an elaborate protocol, which is described in detail in [10]. Upon allocating a multicast address, the MAM also selects a CC to manage the session that will use that address. Upon session termination the CC returns the address to the MAM. The only other time that the MAM gets involved is when the CC fails during the session. Again, an elaborate mechanism is described in [10], whereby the MAM selects a new CC— transparently to the users and without disruption of communication between the participants. Finally, periodic keep-alive messages are exchanged between the MAM and the CCs residing on its (sub)network, to ensure that the MAM has proper state information regarding the status of the CCs, and also for recovery mechanisms.

[7]The network interface will pick up packets on the basis of the 4-byte multicast address only (or, more accurately, the MAC-layer address it is mapped to).

[8]The term open-style conference refers to broadcast-oriented applications, such as seminars or radio, invovling a potentially large number of receivers and a small number of transmitters, with no or limited interaction between them. Similarly, closed-style refers to highly interactive sessions with a small number of participants.

B. Fully Distributed Multicast Address Management

Using the concept of an extended multicast address (6-bytes, multicast address plus virtual port number), the scheme proposed in [10] can be modified so as to implement a fully distributed address management and connection control architecture. In the distributed scheme we can assign a MAM on each host, since we now have a multicast address space that is larger than the unicast one. These MAMs will select extended multicast addresses based on the entire unique 4-byte address of the host on which they reside, thus guaranteeing that no two MAMs will pick the same address. Again, the first byte will be selected from the range [224–239]. The next three bytes will be the first three bytes of the host address. Finally, the port number selected will be of the form Y.A4, where A4 is the fourth byte of the host address. Y can take any value in the range [4–255], since the reserved port number pool [0–1023] has to be excluded[9]. So, for example, the address manager residing on host A1.A2.A3.A4 will select an extended multicast address of the form {224–239}.A1.A2.A3.Y.A4, as shown in Figure 2. Even if Y is fixed, this scheme allows for 16 multicast addresses per host, instead of the other way round. By letting Y take any arbitrary value in the range [0–255], the scheme allows for 4032 extended multicast addresses per host.

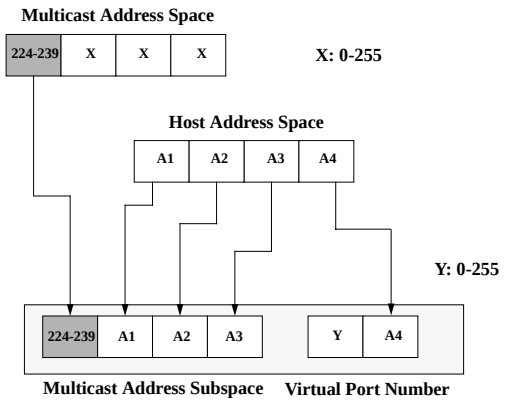

Fig. 2. Extended (including virtual port number) multicast address subspace formation

Each host may now have a number of Connection Controllers (CCs). Upon receiving a request for a new address from a local user, the MAM will create a new CC to manage the session. The task of selecting a new CC in [10] is eliminated, and there is no need for "keep-alive" messages between the MAM and the CCs. Upon session termination, the CC returns the address to the MAM, and the CC is then terminated. The connection control protocol between the CC and participating hosts, and the mechanisms for recovery from CC and MAM failures, are exactly as before (see [10]).

The distributed scheme will also require minor modifications to the multicast routing mechanisms, to avoid packet duplication. This is because packets in IP are currently routed according to their IP destination address only, regardless of the port number. The latter is only used at the source and destination by the transport protocol. The scheme proposed, as described up to this point, could lead to a considerable increase in the network load. To see why this is so, consider hosts A and B in Figure 3, which are participating in two different sessions with the same multicast address but different port numbers. The router will forward both sets of traffic to both hosts, since it routes packets on the basis of the multicast address only.

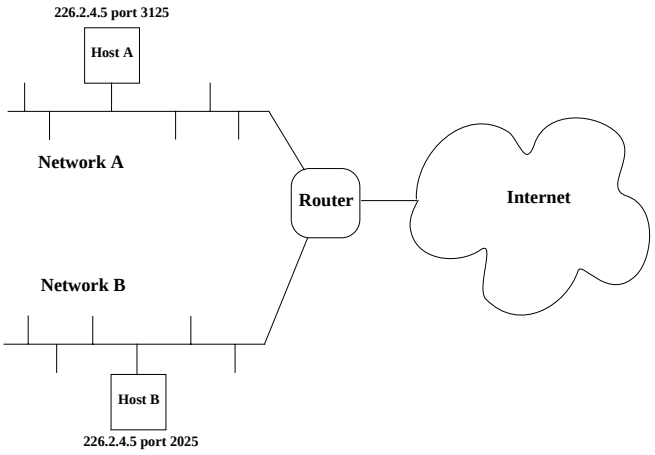

Fig. 3. Traffic duplication without router modification

This problem can be avoided by considering port numbers when making routing decisions, at least for multicast addresses (which, as mentioned before, can easily be distinguished by the first 4 bits of the destination address). This is particularly easy to implement for the most popular multicast routing protocol (DVMRP [9]) as currently the router—`mrouted`—is separate from the regular (unicast) routers, and is implemented in software. Also, the location of the port number in the UDP header is well known, while the length of the IP header is specified by the 4-bit IHL field. Thus, the routing table for `mrouted` could list groups by both address and port number.

A reasonable concern with this approach is that the port number is a transport layer entity, while routing is a function performed at the network layer. This mechanism then violates the logical separation of tasks between the two layers, and requires the router to look inside what it considers to be "data". Note that this is not uncommon, and is being used as a means of packet filtering (primarily for security reasons) in many routers (e.g. [4]). To avoid the extension of routing decisions to the transport layer we can make use of the Options fields of the IP datagram header. These happen to be placed right after the destination address. The single octet option code which precedes each option allows up to 32 options for each of four different option classes.

[9] This restriction actually applies to the actual port number space only, but carries over to the virtual port number space if we want to guarantee that the same number of prts is included in both spaces.

Until recently, only 8 options were defined [5]. We can define a new option within Option Class 0 (used for datagram or network control) to include port number information.

The mechanism described above would require a number of changes to the existing infrastructure. These include the addition of the mapping function and table to the transport protocol, modification of the operating system kernel forwarding code, and the address size modifications to the routing protocols and tables. The performance of the mechanism may be improved with more extensive modifications, in which packet filtering at the receiver—when necessary[10]—is performed in the network instead of the transport layer. The IP service interface, which includes the JoinHostGroup and LeaveHostGroups operations (for joining and leaving multicast groups, and having packets routed accordingly) [6], has to be modified so that group addresses are extended to include the port number, and also to accommodate the semantics of an address manager. Ditto for the IP module, which maintains a table of the multicast group memberships for each network interface, and for the Internet Group Management Protocol (IGMP) which is used by IP hosts to report their host group memberships to multicast routers [6].

Other popular multicast routing protocols, such as MOSPF [14] and PIM [8], will not require extensive modifications either, as the only architectural difference is the multicast address size. Although the process requires changes to custom-designed routing equipment, these are typically driven by field-upgradable software.

We should note that although the required changes are simple, the fact that they affect a very large number of hosts/routers makes their deployment far from trivial. Since, however, DMAM can function even with no routing protocol modifications (albeit at the possible detriment of network traffic load), its gradual incorporation into the existing infrastructure is feasible.

One side-effect of using the port number as part of the extended multicast address (to be used for routing purposes) is that applications that use multiple ports for the same session[11] will have to obtain multiple extended addresses. This will result in additional routing table entries that will have to be maintained, although due to the temporary nature of multicast sessions their number will probably be much smaller than the number of unicast entries in the routing tables.

Another side effect of the proposed scheme is that part of the multicast address space will be left unused since not all possible network/host addresses will be assigned. Given the inescapable fact that for multicast communications port numbers are an integral part of the end-to-end address, and that by using port numbers in the address selection process a host has a significant number of extended addresses at its disposal (more than 4000), leaving some of the multicast address space unused is not a concern. Static allocation of unused addresses can be made, but that is neither necessary nor desirable, since changes have to be made every time a new network/host is added. Finally, this leaves resources available to accommodate expansion of the network (i.e. the performance does not deteriorate as the network—or parts of it—expands).

[10] Filtering may still be necessary for traffic originating from hosts on the same (sub)network. It can be eliminated by redesigning the current way in which IP multicast addresses are mapped to MAC-layer addresses.

[11] The benefits of using multiple port numbers within a single session are debatable.

IV. Other Proposals for Multicast Address Management

The semi-distributed scheme proposed in [10] (henceforth referred to as SMAM) and the fully distributed scheme proposed in this paper (DMAM) have already been described in detail. We therefore give a brief description of the other three approaches.

A. Multicast Group Authority (MGA)

The authors of [3] present an outline of a hierarchical address management scheme. In this architectural outline the multicast addresses are managed by a Multicast Group Authority (MGA) hierarchy, with a centralized controller at the root of an administrative tree. Address requests received from application processes or other MGA nodes result in a block of addresses being assigned to the requesting MGA node. The size of the address block allocated is dependent on the position of the requester in the MGA hierarchy. If a given MGA node runs out of addresses, it will make a request to its parent node. The request is propagated upwards in the hierarchy until free addresses are found. When the root exhausts the address space it issues a request to all its children for reclamation of unused addresses.

B. Heidelberg Multicast Protocol (HeiMAP)

An elaborate description of another approach is given in [22] (HeiMAP), though in the context of local area networks. If some partitioning mechanism for the multicast address space is employed (such as the one described in [10]) and a pool of addresses is allocated to each LAN, it can be used to manage that pool. HeiMAP is based on a two-phase negotiation protocol. Each host is required to keep a table of all multicast addresses that are in use, regardless of whether it is a member of that group or not. Upon session initiation, a host picks an address which is not used according to its local table and broadcasts a RESERVE message. If no REJECT message is received within a specified time-out period, the host will proceed with its call management procedure using that address for the session. The address is released at session termination by broadcasting an UNRESERVE message. The scheme does impose one restriction: an address may be deleted only by the session initiator. Provisions are also made for recovery from various failure scenarios.

C. Random Address Selection

Strictly speaking, random selection of multicast addresses does not fit the definition of address management,

but it is currently the most widely used alternative. It performs satisfactorily today, where multicast sessions are relatively few and far apart, but its architecture may become problematic in the future. The popular Session Directory (SD) program uses scoped advertizing of randomly selected addresses so as to reduce the risk of other SDs selecting the same address. Such a scheme, however, will not work on a large scale due to the huge overhead generated[12], and the higher possibility of simultaneous address allocation due to longer end-to-end delays. Furthermore, advertizing a session may not be desirable for privacy/security reasons. It is, nevertheless, of interest to analyze the probability of collision using a random address selector (without address advertizing). This will be done in Section V-A.2

V. Performance Analysis

In the introduction, we stated the need for an efficient and robust address management scheme which could operate within certain performance constraints. In this section, we consider five different criteria in order to evaluate the performance of a multicast address management scheme: blocking probability and consistency, address acquisition delay, address manager load, robustness, and processing and communications overhead. For each criterion, we compare all four schemes of interest. The results of the comparisons are summarized in Table 1 at the end of this section.

A. Blocking Probability and Consistency

We can segment the five different address management/allocation approaches into two broad categories: consistent, and non-consistent. The former has as a primary design parameter the requirement that no single address may be allocated more than once (hence the term "management"), while the latter allows such multiple allocations. All schemes discussed here except for random allocation fall in the consistent category.

Non-consistent behavior is undesirable as it results in merging of traffic from two or more sessions. Among its potential drawbacks are the traffic load increase, the need to do packet filtering at the transport layer (or at the application layer if the port number happens to coincide) in order to remove packets of other sessions with the same multicast address, and incidental compromise of privacy (if the same or compatible applications are used, allowing processing of received packets from the alternate session). Three strong arguments in favor of such an approach are, however, that: a) it is extremely simple to implement, b) real privacy in IP can only be guaranteed through encryption, and c) the problem is insignificant if the probability of its occurrence is extremely small. Although encryption solves the privacy problem, it involves additional processing overhead (either via software or hardware), and it is seldom used by users. Similarly to the telephone network (which is also not secure), users have a reasonable expectation of privacy in the sense that it cannot be compromised by normal operation of the system, but only intentionally. Regarding intentional attempts, lack of an address manager actually makes eavesdropping trivial if the address of a session is known. Finally, as we will see later in this section, the probability of multiple allocations is not negligible.

At any rate, this is not a strictly technical issue but also a user community one, pertaining to the desired type of service. In the following we separately discuss the blocking probability characteristics of consistent schemes, and the probability of non-consistent behavior for random allocation.

A.1 Blocking Probability

Because of the limited number of multicast addresses available, in consistent address management schemes there is a non-negligible blocking probability. This blocking probability may not be significant today, but will become increasingly so as the number of applications that use multicast communication grows. As observed in [10], address management transactions (acquire and release) can be modelled as an $M/M/k/k$ system, where k is the number of available multicast addresses ("servers"). Requests for multicast addresses (session initiations) arrive in a Poisson fashion at a rate of λ requests/host/minute and session durations are distributed exponentially with mean $1/\mu$. New address requests will be blocked if all k addresses are in use, the probability of which is given by the Erlang-B formula [20]:

$$P_B = \frac{\rho^k/k!}{\sum_{i=0}^{k} \rho^i/i!}$$

where $\rho = \lambda/\mu$. Assuming a request rate of 1 request/hour/host and an average session duration of 3 minutes[13], it was observed in [10] that the blocking probability was 0.11 for a fully occupied (254 hosts) class C network, with 15 addresses at its disposal. With two fully occupied class C networks sharing addresses through the proxy mechanism, the blocking probability was calculated to be 0.06.

In MGA, although the ratio of hosts to multicast addresses is still 16 to 1, the blocking probability is smaller since all addresses are available to all hosts (see [10] for proof), though not by much. Unfortunately, the exact blocking probability cannot be easily calculated due to the large numbers involved, but the beginning of the curve for a load/host of $\rho = 0.083$ (corresponding to 1 request every 24 hours with an average duration of 2 hours) is shown in Figure 4 (host to address ratio is 16 to 1). As can be seen, the drop in blocking probability is very small once the number of addresses increases beyond 32.

In HeiMAP it is not clear how many multicast addresses are made available to the network. The actual negotiation

[12] Currently, SD is working with a small subset (15 bits) of the multicast address space.

[13] These numbers are meaningful for person-to-person telephone communications, but may not be accurate for multipoint multimedia communications. Examples of the latter, which include videoconferencing and group editing, will most probably be seen in the business community where they would be replacing staff meetings. As such, they could be expected to last much longer (in the order of 60 to 120 minutes on the average) but occur less frequently (perhaps about once every 24 hours), yielding more or less the same overall load (ρ).

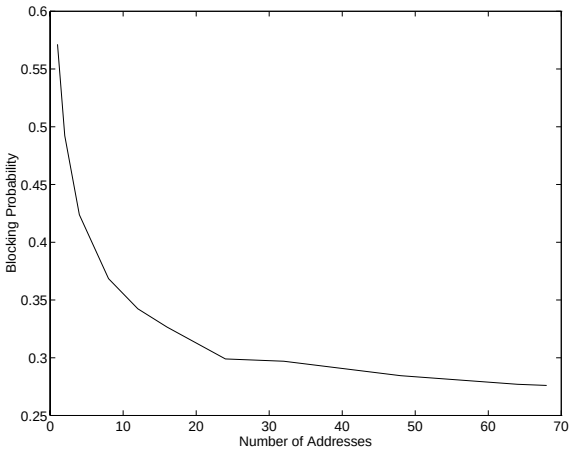

Fig. 4. Blocking Probability in MGA vs. number of addresses, with constant host to address ratio (16:1)

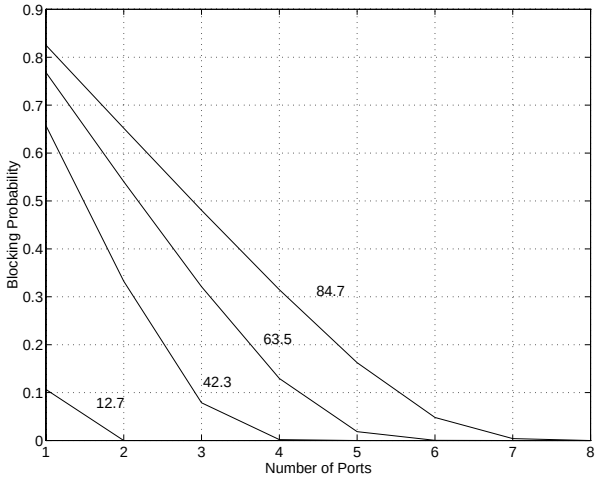

Fig. 5. Blocking Probability in DMAM vs. number of ports for varoius normalized (by 254) hosts

is done for a layer 2 multicast address, which has a larger address space, but is then mapped to a layer 3 (IP) address. If equitable distribution of IP multicast addresses between all networks is assumed, then the blocking probability statistics become identical to that of SMAM.

For DMAM, and if we allow only 2 ports per multicast address (thus increasing the effective number of multicast addresses twofold, for the same number of total hosts), the blocking probability for the class C network considered in [10] becomes $1.7\text{x}10^{-5}$. With three port numbers, the probability is less than 10^{-12}. If we use the full [4–255] range for Y and [1–254] for A4, the blocking probability is practically zero (this corresponds to over a million extended multicast addresses for 254 hosts).

Figure 5 shows the blocking probability vs. number of ports used for various values of ρ (254 times ρ to be exact, where 254 is the number of hosts on a fully occupied Class C network), ranging from 12.7 to 84.7 (the latter representing an average request rate of 4 every 24 hours per host, with average session duration of 2 hours—i.e. a full working day spent in meetings on the average!). It can be seen that by using as few as 8 different port numbers per multicast address, the blocking probability can be made practically zero even for a fully occupied Class C network.

To reduce the chance of address exhaustion, one could conceivably bias the address block allocation mechanism in MGA or any address partitioning scheme in favor of networks where requests for multicast addresses occur more frequently. There are, however, several drawbacks. One is that this information is not available a priori, and has to be accumulated over a long period of observation. More importantly, the load distribution will be dynamically varying, especially if some of the hosts are mobile. Taking these dynamics into account would add unnecessary complexity to the address management scheme.

A significant concern in the Internet community is that, due to the unexpected growth of the number of users beyond the highest expectations of the original designers, the IP address space is being exhausted. It is generally agreed that the long term solution is to increase address sizes from the current 32 bits to 64 bits, or even 128 bits [7]. This, however, will require extensive modifications to current routing protocols, hardware, buffers and memory requirements.

Since the multicast address space can easily be increased to several times its current size by using port numbers, we can reduce the number of IP addresses reserved for multicasting purposes. For example, reducing the multicast address space by 75% from {224–239}.X.X.X to {236–239}.X.X.X (i.e. all addresses starting with 111011)—which can easily be compensated by using 4 ports per address—will free up more than 200 million host addresses, or equivalently, 800 thousand class C networks. The freed addresses can be used for Class C networks (extending them by 37.5% from the current space {192–223}.X.X.X to {192–235}.X.X.X), where the problem of address space exhaustion is the most severe, or for smaller size classes of networks. Using 256 ports per address as in DMAM, would increase the original multicast address space by a factor of 64 and achieve the savings in IP addresses.

It might be argued that a change towards for example 64 bit addresses would make the blocking probability a non-problem, as the multicast address space would be increased by a factor of 2^{32}. Yet larger addresses do not really solve the problem. Firstly, although there will be 2^{32} more multicast addresses, there will also be potentially 2^{32} more host addresses. More importantly, even with 64 bit addresses it is still desirable to partition some of the address space into class C size (256 hosts) or even smaller networks. In fact, one of the problems with the current partitioning scheme is its coarse granularity, whereby the smallest possible set of IP addresses one may obtain is 256. As the number of individual subscribers and small companies that may want direct access to the Internet increases, assigning class C networks to them will become very wasteful. There has been at least one proposal to add three new classes for networks of size 1–2, 1–7, and 1–15 hosts [18]. To minimize address acquisition delay, complexity and control

traffic generated it is best to serve address requests within the particular network segment where the host resides (as in SMAM), or within the host itself (as in DMAM). This in turn requires the partitioning of the multicast space. Using a partitioning scheme as in SMAM to this end will lead to the same blocking probability as before for the class C size networks (since there are still 254 potential hosts for 16 addresses), and an even larger blocking probability for the other, smaller size, classes.

A.2 Probability of Collision for Random Address Selection

With the random address selection alternative, there is no blocking probability, since nobody will reject the random address selected. On the other hand, we have to consider the probability of address collisions. To avoid a collision during a session, one must a) pick a free address during session initiation and b) hope that nobody else picks that address while the session is going on.

Assuming an internetwork with N hosts and M multicast addresses, and assuming an average address request rate of λ requests/host/minute and a session duration of $1/\mu$ minutes (as before), there will, at any given time, be $N\lambda/\mu$ sessions in progress. The probability of selecting a free address during session initiation is thus:

$$P_a = \frac{M - N\lambda/\mu}{M}$$

assuming $N\lambda/\mu < M$. During the session, which will last for say x minutes, k new requests will be made. Each will pick the same address of the session of interest with probability $1/M$. The probability that none will pick that address, given k and x, is: $(1-1/M)^k$.

Given a Poisson process with rate $N\lambda$ for k and an exponential distribution with mean $1/\mu$ for x, the probability that no one else will pick the address of the session of interest is[14]:

$$P_b = \int_{x=0}^{\infty} \sum_{k=0}^{\infty} ((1-1/M)^k \frac{(N\lambda x)^k}{k!} e^{-N\lambda x}) \mu e^{-\mu x} dx$$

which, after some manipulations, yields:

$$P_b = \frac{1}{1 + N\lambda/M\mu}$$

The probability that we avoid a collision is (assuming P_a and P_b to be independent):

$$\overline{P_{col}} = P_a P_b = (\frac{M - N\lambda/\mu}{M})(\frac{1}{1 + N\lambda/M\mu})$$

and hence, the probability of a collision is given by:

$$P_{col} = 1 - \overline{P_{col}} = \frac{2N\lambda/M\mu}{1 + N\lambda/M\mu}$$

[14] This is derived by considering the joint probability distribution of collisions, number of requests, and duration of a session, expanding to a product of conditional distributions using the chain rule, and computing the marginal distribution with respect to the number of collisions (here set to 0).

In the current IP environment, the ratio of the unicast address space to that of the multicast address space is 224 to 16. Assuming an occupancy of only 1% for the unicast address space (37 million hosts), a request rate of 1/host/hour, and average session duration of 6 minutes, the probability of collision is 2.8%. With 3% occupancy and average session duration of 10 minutes, the probability of collision rises to 13.1%. These numbers are clearly too high to be acceptable. Note that this only includes collisions based on the 4-byte IP address; if no port resolution mechanism is used, further collisions due to port matches (even with unicast addresses) should also be considered.

The probability of collision can be considerably reduced if binding of port numbers to multicast addresses becomes widely established. This is because collisions on just the 4-byte multicast address or just the 2-byte port number will automatically be filtered at the transport layer. In other words, collisions will occur when both the multicast address and the port number match. The traffic duplication problem, however, still remains. Furthermore, we should note that random allocation provides no control mechanism in case usage becomes high.

B. Address Acquisition Delay

The time required to acquire a multicast address would be one of the components of the call set-up delay for multipoint applications. Here, we will see how this component can be minimized by using the DMAM approach. Note that delay is treated qualitatively, rather than quantitatively, in our analysis since it depends on a great number of factors (such as link capacity and quality, access delay, router speeds, routing configuration and network topology, and so forth); any quantitative argument would be extremely involved and beyond the scope of this paper.

There are two dominant factors which can affect the address acquisition delay. The main factor is that in MGA and SMAM the host and the address management entity lie on different machines, and thus have to communicate via the network. This is also true for HeiMAP, since the session initiator must send a reservation request to other hosts and allow some time for their possible rejection. Another factor, which comes into play when one of the system components (address management entity, network link, etc.) fails, is the time required to detect a problem and take appropriate measures. A third factor is the possible queuing delay at the address management entity (MGA node or MAM) due to overloading, and is discussed in Section V-C.

B.1 Networking delay

A number of delays are introduced due to the fact that the host and the address management entity lie on different machines, and thus need to communicate over the network. These include network access delay, and delays associated with the reliable transfer of data over the network (propagation delays, queuing delays, possible retransmissions, etc.). The problem is aggravated by the fact that two- and three-way handshakes have to be used to ensure correctness and robustness [10]. The problem will be more severe

for mobile hosts connected by wireless networks, where link capacities are lower and bit-error rates higher.

In MGA, where addresses can be moved from one part of the Internet to the other as needed, this delay could vary greatly. Assuming that there exists an MGA node within each network—although this was not explicitly stated in [3][15]—and if that node has free addresses, then the minimum delay is the time required for the address request to be served within that network segment. Otherwise, the MGA node needs to ask for free addresses from a parent node, spanning two or more networks. 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 the 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.

In SMAM, since addresses are allocated by a single (sub)network entity, the delay is limited to the time required to serve the request within that (sub)network. If the proxy mechanism is employed, the delay will be longer, depending on how far out the proxy address manager is.

In HeiMAP, after broadcasting a RESERVE request, the session creator must wait for some time-out period in which other hosts may reply with a REJECT message. This time-out period depends on the round-trip delay of the network, and a default value of 50 milliseconds was set in [22].

The best performance will be obtained with DMAM, since address requests are served within the host itself without involving the network at all. Therefore all delays associated with propagation, retransmission, congestion and handshaking are eliminated. This will reduce the address acquisition delay by several orders of magnitude.

B.2 Delay required to detect failures

Another parameter to be considered is the time needed for an address requester to detect that the address manager, or the link between them, is down during the session initiation phase. This issue is not addressed at all in MGA. In SMAM, the requester retransmits its request to the address manager using a truncated exponential back-off algorithm. If no response is received after a number of back-offs, the address manager is considered to be down. This delay could be reduced by having periodic "keep-alive" messages between all hosts and their MAM at all times, but that would be wasteful of network resources. In DMAM, however, the user can immediately detect the failure of the address management entity, as it resides on the same host.

[15] The delay will be longer if there is less than one MGA node per network. On the other hand, having one MGA node per network increases the potential of flooding parent nodes, especially those closer to the root, due to the greater number of children.

C. Address Manager Load

One of the main advantages of distributed address management schemes (such as DMAM and HeiMAP) over schemes with a centralized orientation (SMAM, MGA) is that there is much less load on each address management entity, both in terms of the number of addresses managed and the number of potential hosts that it has to serve. This is simply because there are more of them.

The load could become especially severe in MGA for nodes that are close to the root. This is particularly true during periods where use of multicast addresses is heavy, and more requests for addresses are propagated towards the root. Also, each time the root broadcasts a request to reclaim all unused addresses, it and its immediate children could potentially be flooded with unused addresses. This would become a much more significant problem if and when the move to 64-bit addresses is made, thereby increasing the number of multicast addresses from the current 268 million to more than 10^{18}. Similarly, the storage and memory requirements for the root become immense. Although the root will allocate addresses in blocks, the space required to store the multicast address 'in use' flags using a bit mapping scheme would be 33.5 MBytes. With 64-bit (8-byte) addresses, the requirement is close to 1.4×10^{17} bytes! This poses a serious drawback in terms of scalability.

The problem is less severe in SMAM. Class C address managers will manage 15 addresses and service the requests of up to 256 hosts—disregarding the proxy mechanism. For class A and B managers, the numbers depend on how the network is divided into subnetworks, but could become quite significant.

Another factor to consider is the queuing and buffering of multiple address requests at the MGA node or the MAM. This could also affect the address acquisition delay. Neither of the two schemes discuss this issue, but the logical thing to do in each of those schemes (be it at the MGA node or the MAM) would be to queue up requests for multicast addresses and serve them one at a time—or reject all but one, which would have the undesirable effect of increasing the blocking probability. This problem can be especially significant in MGA for nodes closer to the root, as they would be susceptible to receiving many requests during periods of high usage.

With HeiMAP, the load on each host is low since it only handles address requests originating from itself. On the other hand, it does have to keep track of all multicast addresses that are in use, and to react to all address reservations and releases that are broadcast. This is one reason why this scheme is not deemed scalable.

DMAM would require each address manager to handle just the number of addresses that are in use in a single host, and also to service the requests of only that particular host. In addition, the address manager does not have to deal with simultaneous requests for addresses from different hosts, thereby making its task less complex.

D. Robustness

Another characteristic of any address management scheme is its robustness and ability to recover from failures. Two issues are of particular interest here[16]. One is the effect of system failures on the blocking probability. The other is the amount of disruption caused to sessions that are in progress at the time the failure occurs.

As only an outline is provided in [3], details of error recovery were not analyzed. If an end-node fails, new address requests in the branch managed by that node would be blocked. If any of the intermediate nodes or links in the MGA hierarchy breaks down, nodes above and below will not be able to exchange multicast addresses. If the nodes below exhaust their addresses, new requests will have to be rejected, thus increasing the blocking probability. Parent nodes periodically send heartbeat requests to their children to ensure connectivity, with addresses recalled if queries are not answered. This would imply that if network links between a parent and some of its children—or the children themselves—are down for a period of time, the parent will reclaim all the addresses it had allocated to those end-nodes, thereby disrupting all sessions in progress in that branch.

If the MAM in SMAM goes down, none of the hosts residing in that particular (sub)network will be able to acquire a multicast address and, again, all new requests will be blocked. On the other hand, SMAM has recovery mechanisms that allow sessions to continue uninterrupted if either the MAM or the CC—but not both—fail.

In HeiMAP, failure of the address managing process at a host will only affect users on that particular host, where new requests will not be honored. The scheme also describes mechanisms to detect failure of a session creator (which is the only entity which may release an address). Periodic REFRESH messages are exchanged between the creator and members for this purpose. If no message is received from the creator, however, hosts are forced to leave, leading to session disruption.

In DMAM, the failure of a MAM will prevent only the particular host on which the MAM resides from acquiring new addresses. Even this can be avoided since there is a MAM on every host, and the user can potentially use a MAM process on a different host (if the MAM interface allows, for example, remote procedure calls). DMAM is more vulnerable to machine failures than SMAM, since both the address management and session management entities reside on the same host (the same error recovery mechanisms as those of SMAM apply in DMAM in case of process failures). Such a failure, however, will have less dramatic consequences than it would in a more centralized scheme, as it leads to loss of only those sessions that were initiated by the failed host.

[16] Proper maintenance of multicast address status information to prevent multiple allocations of the same address is, of course, a given requirement.

E. Processing and Communications Overhead

MGA, HeiMAP, and SMAM all require the exchange of periodic refresh messages between different hosts. In MGA, parent nodes in the hierarchy exchange such messages with their children; in SMAM, such messages are exchanged between the MAM and the CCs; in HeiMAP, periodic REFRESH messages are exchanged by the session creator and all the other participants. These periodic messages have the effect of generating added control traffic, and processing overhead at the hosts involved.

In addition, all three schemes generate control traffic during allocation and deallocation of multicast addresses. MGA will generate control traffic that might span several networks, and even be propagated throughout the internetwork. With SMAM, the control traffic is confined to the (sub)network, if the proxy mechanism is not used, but otherwise will span to other networks as well. The control traffic generated is one of the main reasons why HeiMAP cannot be scaled to an internetwork, as each RESERVE and UNRESERVE request will have to be broadcast to all hosts.

In terms of the control traffic generated, the fully distributed scheme is clearly superior, as all address acquisition and release is performed locally on the host. Furthermore, no periodic keep-alive mechanism is required to maintain information about the status of multicast addresses, with the side benefit of less processing complexity.

The overhead added by the addition of the VPN-APN mapping tables is very small. It will only be a problem if the mapping tables become extremely large. It is doubtful, however, that this may become a problem, since it is highly unlikely that a host will be a member of more than a few tens of multicast groups at the same time.

F. Summary of Performance Analysis

We present the summary of the evaluation of the four consistent schemes in Table I. Note that for random allocation address acquisition delay, blocking probability, and complexity are all negligible, while robustness is very high.

VI. Non-IP Networks

The current Internet is dominated by the IPv4 (IP Version 4) protocol suite, but due to its rapid and unpredicted expansion, there has been a growing discussion on the need to replace it with protocols that are better suited to today's (and tomorrow's) needs. In this section, we examine the DMAM scheme in the context of a number of potential replacements of IPv4, namely SIPP (IPv6) CATNIP (IPv7) and TUBA (OSI CLNP). The discussion is necessarily brief, and primarily focuses on the architectural level.

It must be noted that the multicast address management and connection control protocols are independent of how the multicast address space is partitioned: as long as a number of addresses (regardless of whether the addresses are hierarchical or not) are allocated for multicast communication, then that address space can be partitioned in any arbitrary way and assigned to individual network seg-

TABLE I
PERFORMANCE ANALYSIS SUMMARY

	MGA	SMAM	HeiMAP∗	DMAM
Blocking Probability	low	high	high†	negligible
Address acquisition delay	high	low/medium	low (50ms)	negligible
Address manager load	high	low/medium	low	very low
Robustness	†	high	medium	high
Complexity	high	medium	medium	low

†: Insufficient information for evaluation
∗: Augmented by SMAM-style address space partitioning

ments. No matter what is used as the basis for address space partitioning, MAMs and CCs can be made responsible to manage each partition. The only requirement for the operation of the address management protocol is the availability of datagram transmission and multicast routing capability. Accordingly, in the following we focus on the address partitioning issues.

Our IP-based partitioning scheme has been coupled with the overall partitioning scheme of the network layer, the hierarchical nature of host addresses, and the use of port numbers to distinguish among processes. The Simple Internet Protocol Plus[17] (SIPP) [7], which has been recommended by the IETF as the basis for IPng (IPv6) [11], extends the IP address size from 32 to 128 bits, supporting even more levels of addressing hierarchy. As in IP, specific addresses are reserved for multicast communication (1/256th of the new address space). Support for extensions and options has improved, allowing greater flexibility for introducing new options. This would make the inclusion of the port number in the options field more attractive. Port numbers are still used by the higher level transport protocols (TCP and UDP) to distinguish among applications. This structure enables the direct extension of our IP partitioning scheme to SIPP.

Another proposed replacement for IP is TUBA (TCP and UDP with Bigger Addresses) [12]. TUBA uses the ISO Connectionless Network Protocol (CLNP, ISO 8473) at the network layer, augmented with appropriate redesigns of the TCP and UDP transport protocols. The protocols will use the OSI Network Service Access Point (NSAP) address format. NSAP addresses may have arbitrary lengths, but are typically limited to 20 bytes. Current working drafts propose a multicast address space as large as the unicast address space by assigning a one-to-one mapping of AFI (Authority and Format Identifier) fields between unicast and multicast addresses [13]. This should alleviate further the issue of blocking probability. Since TCP and UDP are included in TUBA, port numbers can still be used as before; in fact, since NSAPs are variable length, port numbers can be directly included in the network layer address in the DSP (Domain Specific Part) field.

[17] SIPP resulted from the merging of the Simple Internet Protocol (SIP), "P" Internet Protocol (PIP), and IP Address Encapsulation (IPAE) working groups of the IETF.

CATNIP (Common Architecture for Next-generation Internet Protocol) [23] has evolved from the TUBA IPng proposal and the TP/IX protocol, and its objective is to provide common ground between the IP, IPX (from Novell Inc.) and ISO's CLNP. It will also use the NSAP address format (albeit extended by a prefix byte specifying the length of the address), and hence the same flexibility as with TUBA exists. As an example, NSAP addresses allow a direct encapsulation of IPv4 addresses. This is done by assigning a specific Authority and Format Identifier (AFI) value for IPv4 addresses, and then having a 4-byte DSP which is the 4-byte IPv4 address. Since there is a one-to-one mapping between unicast and multicast AFIs, we can easily specify a corresponding AFI for IPv4 multicast addresses, in which the format of the address specified in the DSP part will be 6-bytes (i.e. includes the port number).

VII. Concluding Remarks

We described a distributed architecture for managing multicast addresses in the global Internet. Our approach consists of an address partitioning scheme based on the unicast host address, a multicast address management entity per host, and separate processes for session management. By noting that port numbers are an integral part of end-to-end multicast addressing, the multicast address space was effectively expanded to include the port number. This in turn allowed us to present a single solution to the two problems of dynamic multicast address management and port resolution.

We then presented a framework for evaluating multicast address management schemes, and used it to compare our design with three other proposed ones. The criteria used were the blocking probability for address requests, consistency, address acquisition delay, the load on address management entities, robustness against failures, and processing and communications overhead.

Among the consistent schemes, the fully distributed one reduced the blocking probability by several orders of magnitude, to insignificant levels. At the same time, address acquisition delay is reduced to a minimum by servicing the request within the host itself. It was also shown that the scheme generated much less control traffic, was more robust against failures, and put much less load on address management entities. Random allocation was shown to be

attractive due to its simplicity, although several drawbacks were identified. These include non-consistency and its resultant traffic duplication with non-negligible probability, implications on session privacy, as well as lack of any control mechanisms in case usage becomes high (in which case the collision probability increases rapidly).

The DMAM design can be implemented with simple modifications to the transport layer and multicast routing software, although their incorporation is far from trivial as they affect a large number of hosts. A side benefit is that part of the IP address space can be freed up and provide temporary relief for the address space exhaustion problem.

References

[1] Xpress Transfer Protocol Version 3.6. Technical report, XTP Forum, Santa Barbara, CA, 1992.

[2] S. Armstrong, A. Freier, and K. Marzullo. Multicast Transport Protocol. RFC 1301, February 1992.

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

[4] Cisco Systems. *Router Products Configuration and Reference, Vol. II, Section 13*, software release 9.1 edition, September 1992.

[5] D. Comer. *Internetworking with TCP/IP*. Prentice Hall, Englewood Cliffs, NJ, 1991.

[6] S. Deering. Host Extensions for IP Multicasting. RFC 1112, Stanford University, 1989.

[7] S. Deering. Simple Internet Protocol Plus (SIPP) Specification (128-bit address version). Working Group Draft, IETF, July 1994.

[8] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. Protocol Independent multicast (PIM): Motivation and Architecture. Working Group Draft, IETF, March 1994.

[9] S. Deering, C. Partridge, and D. Waitzman. Distance Vector Multicast Routing Protocol. RFC 1075, November 1988.

[10] 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.

[11] R. M. Hinden. IP Next Generation Overview. Working Group Draft, IETF, October 1994.

[12] D. Katz and P. Ford. TUBA: Replacing IP with CLNP. *IEEE Network Magazine*, pages 38–47, May 1993.

[13] D. Marlow. Host Group Extensions for CLNP Multicasting. Working Group Draft, IETF, May 1994.

[14] J. Moy. Multicast Extensions to OSPF. RFC 1584, March 1994.

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

[16] Y. Rekhter and T. Li. An Architecture for IP Address Allocation with CIDR. RFC 1518, September 1993.

[17] J. Reynolds and J. Postel. Assigned numbers. RFC1340, ISI, July 1992.

[18] P. Robinson. Suggestions for New Classes of IP Addresses. RFC 1375, Tansin A. Darcos & Co., October 1992.

[19] Henning Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. Working Group Draft, IETF AVT, July 1994.

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

[21] C. Topolcic. Experimental Internet Stream Protocol: Version 2 (ST-II). RFC1190, October 1990.

[22] B. Twachtmann and R. G. Hertwich. Multicast in the Heidelberg Transport System. Technical Report 43.9206, IBM European Networking Center, Heidelberg, 1993.

[23] R. Ullman. CATNIP Common Architecture for Next-generation Internet Protocol. Working Group Draft, IETF, March 1994.