Internet Engineering Task Force
INTERNET DRAFT
October 1999

R. Perlman
Sun Microsystems
C-Y Lee
Nortel Networks
A. Ballardie
Research Consultant
J. Crowcroft
UCL
Z. Wang
Lucent Technologies
T. Maufer
3Com Corporation
C. Diot
Sprint
J. Thoo
Nortel Networks
M. Green
@Home Networks

Simple Multicast: A Design for Simple, Low-Overhead Multicast

<draft-perlman-simple-multicast-03.txt>

Status of this memo

Abstract

This paper describes a design for multicast that is simple to understand and low enough overhead for routers that a single scheme can work both within and between domains. It also eliminates the need for coordinated multicast address allocation across the Internet. It is not very different from the tree-based schemes CBT, PIM-SM, and BGMP. Essentially all of the mechanisms to support this have already been implemented in the other designs. The contribution of this protocol is in what is NOT required to be implemented.

The main idea for simplifying multicast is to consider the identity of a group to be the 8-byte combination of a "core node" C, and the multicast address M. The identity of the group is carried in join messages and data messages. M no longer has to be unique across the Internet. It only has to be unique per C. The other idea, which is independent of the first, it to build a bi-directional tree (as is done in CBT and BGMP) instead of building per-source trees from each sender. This reduces the state necessary in routers to support multicast.

Changes from revision 1

- use a Simple Multicast (SM) header instead of a new IP option

- modified branch creation and deletion to avoid loops

- added tree splicing mechanism

- added multicast scoping

- allow both IGMP and host SM Join

- added sender only joins

- third party independence

- layer 2 filtering

- host API and kernel changes

1.0 Introduction

IP Multicast has been around for over a decade, and several multicast protocols have been developed over the years. However, the solutions are either difficult to understand or expensive to deploy or both. In particular, we believe that multicast address allocation protocols are too complex and BGMP in combination with MASC will not scale easily.

In this paper, we present a design we call Simple Multicast that reduces the complexity and overhead of multicast. It is not really "yet another multicast protocol". Instead, it is more like a subset of other protocols, with one variation; to have the identifier of a group consist of both C (the core) and M (the multicast address). This eliminates the need to have unique multicast addresses and coordinate multicast addresses across the Internet.

1.1 Previous Work

DVMRP is the first multicast routing protocol proposed. It uses a simple mechanism of flooding and pruning.

The scalability issues with DVMRP led to the development of CBT. In CBT, a multicast group is formed by choosing a distinguished node, the "core", and having all members join by sending special join messages towards the core. The routers along the path keep state about which ports are in the group. If a router along the path of the join already has state about that group the join does not proceed further. Instead the router just "grafts" the new limb onto the tree. The result is a tree of shortest paths from the core, with only the routers along the path knowing anything about that group.

In PIM-SM, each node could independently decide whether the volume of traffic from a particular source is worth switching from a shared tree to a per-source tree. Thus, there are two possible trees for traffic from a particular source for group M; the shared tree and the source tree. To prevent loops, the shared tree had to be unidirectional, i.e., to send to the shared tree, the data has to be encapsulated and unicast to the core.

The other issue that makes current protocols complex is the necessity for routers to be able to figure out the location of the core based solely on the multicast address M. In PIM-SM, this resulted in a protocol whereby "core-capable" routers are being continuously advertised. All routers keep track of the current set of live core-capable routers, and there is a hashing function to map a multicast address to one of the set of core-capable routers. This advertisement protocol is confined to within a domain because it was recognized

that this mechanism would not scale to the entire Internet.

For inter-domain multicast, a set of new protocols has been proposed. The MASC protocol deals with hierarchical block allocation of Class D address space. Essentially, it creates a prefix structure in multicast address space in a way similar to unicast address space. Because of the limited multicast address space, the allocation has to be dynamic. MASC contains mechanisms for collision detection and de-allocation. Once a block of multicast addresses is allocated, and no collision is detected for a period of time, the address block is then given to MAAS servers for actual assignment to multicast groups. The address block has to be propagated through BGP+ so that routers throughout the Internet can know the mapping of multicast addresses to cores, even in other domains. BGMP then uses this information to know the direction in which a join to multicast address M should be sent.

1.2 Overview of Simple Multicast

The Simple Multicast proposal tries to reduce or eliminate some of the complexity and overhead of multicast by taking a slightly different approach. The basic idea in Simple Multicast is that a multicast group is created by generating:

- a distinguished node C known as the "core"

- a multicast address M

The multicast group is then identified by the pair (C,M) rather than just M as in conventional IP multicast. Note that the address M does not have to be unique across the Internet now. Instead, only the pair (C,M) has to be unique. That means that every node C in the Internet can assign the full 28 bits worth of multicast addresses.

In Simple Multicast, multicast address allocation and core placement (i.e., choosing a multicast address M and a core C for a multicast group) are taken out of the basic multicast protocol. End systems may find out about the multicast address M and the core C for a group through one of several possible mechanisms including email announcement, web advertising, SDR, DNS lookup etc. Both SM-aware endnodes and SM-aware routers must recognize the combination of (C,M) as the identity of the group.

Once the end systems have M and C, they then join the group by sending a special join message towards the core C, creating state in the routers along the path until the join packet hits the core or a router that is already on the tree for this multicast group. This creates a branch in the bi-directional distribution tree for the

group. The current IGMP mechanism for joining groups is fine, provided that both C and M appear in the IGMP reply. Until IGMP is modified to support this, the join message itself can be sent from the end system. If both C and M appear in the join message, then the first hop router can initiate the join.

To enable incremental deployment of Simple Multicast, we provide a mechanism for the join message traverses non-SM aware routers. (See Joining a Group).

The multicast tree formed is bi-directional, meaning that traffic can be injected from any point. The core is just another node in the tree. The data packet contains both C and M, and routers look up the group based on the combination (C,M).

Data packets would need to carry both C and M. There has been a few suggestions on how this may be done: 1) Define a new IP option and specify both C and M in it. 2) Define a new protocol and specify the new protocol in the 'protocol' field of the IPv4 header. Encapsulate the payload inside this new protocol. This new protocol header will contain both C and M. 3) Map (C,M) to a unique class-D address on the data-link. The destination address of the data packet would be re-written to a unique class-D address before being forwarded on that data-link.

Although option processing in general is more expensive, in this case the option processing is merely, forwarding packets by looking at an extra IP address in the option field. In contrast, other IP options such as LSR, SSR and Router Alert are more involved. Hence, from a purely technical point of view, the first and second approach can be implemented in hardware and there is no significant difference between these two approaches. However, due to current hardware implementation convention, option processing is more likely done in software. As a result, we have opted to use the SM header instead.

The third approach does not require data packets or join messages to carry the core address. SM nodes obtain the unique class-D address which maps to a group (C,M) from a special node(s) on the data-link. This approach is appealing because it allows SM applications to join a group by joining a class-D address just like conventional IP multicast. On the other hand, it also introduces concerns not unlike label switching, e.g. vulnerability to loops, ensuring the uniqueness of addresses at all times, ensuring all nodes on the LAN use the same address for a group at all times and address recycling, among others. In this approach, if a unique address on the data-link is not available for use, data cannot be forwarded. In contrast, if a packet cannot be label switched, it can be routed. We are investigating the feasibility of this approach.

The SM header will carry both C and M. The reason for carrying both C and M in the option instead of carrying at least one of them in the destination address is to allow SM aware routers to co-exist with non-SM aware routers. The destination address in the IP packet is set to a reserved multicast address, the ALL-SM-NODES, when sending to networks with SM aware routers. This ensures that non-SM routers will not forward SM multicast data packets. When the packet must hop over non-SM routers, the IP destination address is set to the next SM-aware router in the path.

A nice feature of Simple Multicast is that, since both C and M are in the SM header, the destination address in the IP packet can be replaced with the tunnel endpoint address, and packets can be 'tunneled' with very little work. Instead of having to add and delete IP headers (if the packet is encapsulated IPIP), the only work is to write the tunnel endpoint address into the destination address of the IP header..

1.3 Why Simple Multicast

We now discuss some of the advantages of Simple Multicast.

- One protocol is all that is needed. Currently, we need to deal with two sets of multicast protocols in order to support multicast in the Internet: DVMRP, PIM-DM, PIM-SM and CBT etc for intra-domain multicast and MASC, MAAS and BGMP for inter-domain support. The beauty of the Simple Multicast proposal is only one multicast protocol is needed for both intra-domain and inter-domain. This is possible because Simple Multicast is designed to be scalable.

- Scalability. Simple Multicast is scalable to the global Internet. This scalability is achieved by using a trivial multicast address allocation scheme, decoupling core selection and discovery from the multicast protocol and using bi-directional trees. If core discovery is decoupled from multicast routing protocols such as PIM-SM or CBT, these protocols would not have to use the bootstrap mechanism to discover and select cores, a mechanism generally considered to be not scalable.

- Trivial multicast address allocation. IP Multicast address allocation is still an unresolved problem. Dynamically allocating addresses such that addresses are allocated in aggregatable blocks, while ensuring low probability of address collision (non-uniqueness) is non-trivial. In Simple Multicast, since (C,M) is the identifier for a multicast group, address assignment becomes totally trivial, since addresses only have to be unique per core. Each core can have the full 28 bit space (over 200 million address) so we have virtually unlimited multicast addresses. Each core can allocate these addresses

independently without Internet-wide coordination.

- Cost effective and efficient delivery trees. It takes less state in routers to support a group with n senders with a single shared tree than with n per-sender trees. A bi-directional shared tree is as cost effective for delivery of traffic from source S,even if S is not the core, as a per-source tree rooted at S. The bi-directional shared tree is much more efficient for delivery of traffic from non-core source S than a unidirectional tree where the data from S must be tunneled to the core before being multicast.

Bi-directional trees are more robust. In a unidirectional tree, the core is needed for relaying packets from all senders. If the core is down, the tree is gone. For a bi-directional tree, the core does not hold any particular significance. The core is just another node in the tree. If the core is down, the tree is merely partitioned and may still be used for traffic delivery if the application chooses to do so.

- Incremental deployment. Simple Multicast routers may be deployed along side unicast routers and other multicast routers. Traffic is effectively tunneled (although the actual mechanism used is more efficient than tunnels) through routers which do not support Simple Multicast. Therefore a network manager may incrementally add Simple Multicast routers as multicast users spread in the network.

2.0 The Design

In this section, we describe the design of Simple Multicast and its basic operations in detail.

2.1 Creating a Multicast Group

To create a group, one needs to select a core address and a multicast address.

Typically most applications consist of a single high-volume source. For those applications, the core should be the source. For others, any node close to any member of the group would be a logical choice for core. Because the tree-building strategy (like BGMP) uses a single exit point from a domain or any region separated from the rest of the Internet through expensive links, the traffic pattern resembles individual trees within domains hooked together with inter-domain paths. In other words, if S is in your domain, then you will receive traffic from S through a path internal to your domain even if the core of the group is outside the domain. Therefore, even if most of the members of the group are in Europe, and one member of the group is in Australia, and the Australian is chosen as the core,

the tree will still be a very good tree. Traffic between the Europeans would be multicast through the tree confined within Europe, even though the core was in Australia.

As the multicast addresses only need to be unique per core, each core has over 200 million multicast addresses for allocation. Once the core is chosen, some very simple mechanisms can be used to generate the multicast address for the chosen core, for example, querying the core for an address or random generation as it is done in SDR (the collision rate will be significantly lower). Some permanent mapping of "well-known" addresses for popular groups is also feasible.

2.2 Joining a Group

To join a group, one first has to find the core address C and multicast address M. It is appropriate to have a variety of mechanisms. A web page advertising a "singles chat group" might advertise its (C,M) on its web page. Or a provider of some other sort of service, like stock quotes, might advertise on a web page. Ideally, clicking on the web page would cause M and C to be downloaded to the client machine, which would then join the group. Another mechanism, for instance when arranging a private conference, might be to be told about M and C via the telephone, or via email. Yet another mechanism is to have the group (together with a name or a description) advertised in a directory such as SDR.

If IGMP is extended to support SM, the host sends a membership report for group (C,M). The SM DR is responsible for forwarding the join off the LAN. This message is sent towards the core, creating state in the routers along the path, so that each router knows which ports are in the group (C,M).

If there are no SM routers on the LAN, a host may send an SM Join itself. The destination IP address of the join message is set to the core IP address. If a non-SM router on the LAN receives the join message, it will forward it to the core. Data will be tunneled to this endnode by an upstream SM router. As there could be potentially multiple tunnels to the LAN, host SM Join should only be used when there is no local SM support as may be the case during initial deployment or when there are very few local members to justify a network upgrade. If the next hop towards the core on the LAN is an SM router, and if it is not an SM DR itself, it will redirect the join to the SM DR. In this case, if data is tunneled from upstream, it will be tunneled to the SM router that forwards the join off the LAN, instead of the endnode. [Note: This approach provides a migration path whereby as more SM routers are deployed on the LAN, less tunnels are used. It also allows the co-existence of IGMP (with or without SM support) and host SM Join during the migration

process.]

If a router receives a join formulticast address (C,M), and it already has state for (C,M), then it merely adds that port to its set of ports for (C,M) and does not forward the join further. The result is a tree of shortest paths from the core to each member. Each router on the tree has a database of (C,M, {ports}) that tells it, for group (C,M), the ports that data should be forwarded to.

The join message is sent with the Router Alert option. Since the join message has C as the destination address, if an intermediate router is not SM aware, it will just forward the join towards the core. When the join message reaches an SM-aware router R2, it looks at the IP source address of the join message, say R1. If R1 is a neighbor, R2 adds the port from which the join was received to its list of ports for (C,M). If R1 is not a neighbor, R2 will add a join-ack to R1. If R2 is not a neighbor, R1 adds the 'tunnel port' to R2 as its 'parent port' for (C,M). If R2 is a neighbor, R1 just adds the port as its parent port for (C,M), since the packet will not need to be tunneled to get to R2.

A non-member sender may join the group as a sender-only (cf uni-directional join in CBT). The sender will be on-tree and thus will be sending keep-alives and receiving heartbeat messages, and hence will be aware about core liveliness. Data will not be forwarded to a sender-only branch.

2.3 Transmitting to multicast group (C,M)

A sender who is a member of the group, sends an IP packet with C and M in the SM header. The destination IP address is set to ALL-SM-NODES. This ensures non-SM aware nodes will ignore the packet. Only SM aware routers will forward the packet.

A router that receives an SM packet looks up (C,M) in its forwarding table. If it knows about (C,M), it checks if the port it received the packet on is in its database. If not, it drops the packet. If so, it forwards the packet onto all the other ports listed in its database for (C,M). If the outgoing port is a tunnel port, the destination address of the IP header is replaced by the tunnel endpoint, and will therefore travel across routers that are not SM-aware. At the other end of the tunnel, the SM-aware router will replace the destination address with ALL-SM-NODES, or with another tunnel endpoint's address, depending on whether the

packet is being forwarded on a "real port" or a "tunnel port.

If you are not a member of the group but want to transmit to the

group, you place C into the IP destination address, and put C and M in the SM header. The packet might travel all the way to the core, but if it instead hits an SM-aware router R with state about (C,M) before it gets to the core, R will inject the packet into the tree. A sender-only member may transmit like a member, but will not be receiving any packets for this group.

2.4 Inter-domain Multicast

Simple Multicast works both for intra-domain and inter-domain multicast. Because the join message of Simple Multicast carries the core IP address, and unicast routing already knows how to reach any IP address, the join message will be delivered based on the unicast forwarding table.

2.4.1 Incongruent unicast and multicast topologies

Where the unicast and multicast topologies are incongruent, BGP-4+ [MBGP] allows a network provider to specify the path it would accept multicast traffic independent of the path unicast traffic would traverse. In the figure below, AS1 may have a peering agreement with AS2 to forward its unicast traffic, but a peering agreement with AS3 to forward multicast traffic. A join from AS1 towards any cores in AS4 would be sent via AS3. A finer granularity of policy may specify certain network or core ranges that AS3 would carry traffic for.

```
        AS2
      *     *
     *       *
   AS1       AS4
     *       *
      *     *
        AS3
```

The join message to C should be routed towards the exit router specified by BGP4+, for delivery of multicast traffic outside of the domain.

2.4.2 "3rd Party" Independence

For the case in which SM is used both within and between domains, joins from different parts of the domain might only converge (merge) outside the domain. It is not desirable for a domain to depend on another, "3rd party", domain for the distribution of internally sourced traffic to other internal receivers. It is therefore necessary to ensure that joins from different internal receivers merge at a common point inside the domain.

BGP-4 operates on border routers (BRs) of transit domains, and ensures that all BRs know which of them acts as egress for a particular unicast prefix. Some transit domains (the elected egress router) inject external route information internally, and therefore, internal routers know in which direction to forward packets destined to a particular unicast prefix. In other cases, and in stub domains, external route information is not injected inside the domain. Nevertheless, the BRs of these domains know for which unicast prefix(es) each of them is acting as egress. Thus, domain BR routing knowledge ensures that joins originated inside a domain converge at a common point inside the domain.

This principle can be applied recursively across a multiple levels of routing hierarchy.

2.5 Failure Recovery

The situations to detect are:

- branch unused

- loop

- path to core broken or changed

- core dead or unreachable

Any of the tree building schemes (CBT, PIM-SM, BGMP) need to solve these problems, and there is no need to do anything radically new. The only extra mechanism we've introduced is for loop detection. Since packets can quickly proliferate in a multicast loop, it is desirable to detect a loop as soon as it is formed forms. Since SM uses an SM header, we can make use of a flag that will enable us to detect a loop on a data packet.

The other mechanisms we specify are similar to those already in place for PIM, CBT, and BGMP.

2.5.1 Unused Branch

A branch must be kept alive with a "keep-alive" message. If R receives at least one keep-alive message from a child in tree (C,M), R sends a keep-alive to its parent port for (C,M). If no keep-alive is received for some amount of time (at least a few keep-alive intervals) from some child port for (C,M), that port is removed from the list of ports. If there are no more child ports, then R stops sending keep-alives, or as an optimization "unjoins" from its parent.

2.5.2 Loop

It would be easy to detect a loop if we could assume that any data packet for which TTL became zero implied there was a loop. Unfortunately, some applications do an "expanding ring search" or a traceroute in which packets are launched with very small TTLs. It would be wrong to conclude there was a loop when the TTL on those packets expired.

We use a flag in the SM header to indicate a packet that would indicate a loop if its TTL reached 0. An application launching a packet with a low TTL would not set that flag. SM routers do not need to look at the flag except on packets for which TTL expires.

Loops can also be detected on keep-alive and heartbeat messages (which are sent outwards from the core...see next section). The keep-alive message indicates "hops from furthest leaf". A router collects keep-alives from its child ports and transmits a keep-alive that is one hop more than the maximum "hops" it receives in any keep alive from a child.

The heartbeat is like a keep-alive, but from the parent. Likewise it carries a "distance from the core". In either case (heartbeat or keep-alive) if the distance gets too great a loop is suspected and the port is removed from the tree and the child rejoins to the core.

2.5.3 Path to core broken or changed

A parent transmits a "heartbeat" message to its children at regular intervals. The heartbeat indicates whether the core is known to be alive. A parent continues sending heartbeat messages even if it stops receiving "core-alive" heartbeats from its parent. In this way a subtree will continue functioning even if the core is dead. And if the core is not dead, the parent can simply rejoin without causing disruption to the nodes below it in the tree, where feasible.

If unicast routing indicates the path to the core has changed, R rejoins to the core, again, without disrupting the subtree below it, where feasible.

To avoid loops from forming, the parent would rejoin the core using a special join to splice the sub-trees. This splice message must be forwarded all the way to the core, creating state where there is no existing state. The core will acknowledge the splice message.

If the splice message hits a downstream router, it will be forwarded until it reaches the router that originated this splice message. At

this point, the router would realize that it cannot splice the sub-trees without causing loops. Depending on application requirement which is conveyed to routers from core via heartbeat messages, the router could either flush the sub-tree and let leaf routers or hosts rejoin, or if the application desire, allow the sub-trees to continue functioning separately, but attempts to splice the sub-trees again when unicast route to the core changes. The latter makes more sense when there is a network partition, and the core is not reachable. /* MODIFIED */ The decision to flush the sub-tree or rejoin the core can be based on information such as the depth of the sub-tree and distance to core. This information may be obtain from the keep-alive and heartbeat messages.

Since the heartbeat message is generated at regular intervals even if a heartbeat is not received from the parent, a very long tree does not suffer from delay variance that might cause nodes very far from the core to incorrectly assume the tree was broken.

2.5.4 Core dead or unreachable

When the core transmits a heartbeat message it sets the "core alive" flag. If a router has received a heartbeat message from its parent with the "core alive" flag set recently enough (3 heartbeat intervals), then it sets the "core alive" flag in its heartbeat messages to its children.

If it stops receiving heartbeats with "core alive", it prunes itself from the old parent and rejoin (by sending a splice message) the core.

The only purpose of knowing whether the core is alive or not is for applications to decide, if there are multiple trees for a group, which tree they should transmit on. (see next section)

2.5.5 Multiple Trees for Reliability

The core should be selected to be a node that is reliable. However, if a group will be long-lived and there is the worry that the core might die, a simple mechanism is to create multiple trees (C1, M1) and (C2, M2) for this group. All members join both groups. They can transmit on either group. If "core alive" heartbeat is only received on group (C1, M1) that is the group that should be transmitted to.

For applications for which instantaneous switchover is more important than overhead, senders should transmit on both trees.

2.6 Access Control

We accomplish access control by allowing the core for the group to be configured with the set of allowed senders. The core can put the access rules into the heartbeat message. The heartbeat message contains a list of address prefixes of authorized senders and unauthorized senders. If the rules do not fit into the heartbeat, or the core for privacy reasons does not want to advertise in advance all the allowed senders, it can specify that no senders other than It is allowed. In that case, all senders must tunnel packets to the core and the core will forward them. Once a sender gets permission to send, and is known to have data to send, the core can add that sender's address to the heartbeat message.

For example, if there is some sort of authentication that must be done in order to get permission, the core initially disallows all senders, but then when S1 gets permission, it gets added to the list in the heartbeat message.

Since the heartbeat message gives the access rules, all SM routers will refuse to forward a packet from a sender disallowed by the access rules.

Border/Access routers may also have an additional Access Control List locally. For instance, it may have a list of sender prefixes/addresses allowed to transmit multicast data. All multicast traffic with source address matching these prefixes/ addresses will not be filtered. The Include/Exclude Senders List from the core will prevent these senders from sending to a group that they are not permitted to.

2.7 Dynamically forming more trees

In some cases dynamically formed auxiliary trees make sense, especially in the inter-domain, where policy might prohibit packets from A to D to transit domain B. With a core in domain B, or just due to the shared tree that happened to get formed, packets from senders in A to receivers in D might traverse domain B. One simple method of solving the problem is to have A unicast to the core, and have the core send the multicast. B is still acting as a transit domain between A and D, but it doesn't know it.

Another solution takes inspiration from the PIM-SM concept of using the shared tree to find out about per-source trees. The way it works is that the sender in domain A, say X, sends a message to the core C telling it that it would like to create a "spin-off" group, (X,M'). Then the core C, in the heartbeat messages for group (C,M) advertises the spin-off trees that members of (C,M) should also join. The spin-

off tree would, like the original tree, be kept robust through keep-alives.

Although this does allow creation of multiple trees to support a single group, this is less expensive than the PIM-SM scheme because it does not always create a tree for every sender. It only does it when necessary, and does not need a totally separate tree for each sender. It only needs one per domain in which there are sources (and only when the shared tree doesn't work because of transit policy problems).

2.8 Multicast Scoping

A multicast group address can be scoped such that packets matching the group address are not forwarded outside the defined region. Two commonly used scopes are the link-local scope and the global scope and they do not require configuration. Routers merely do not forward the statically assigned link-local scope address (224.0.0.0/24).

The third type of scoping requires network administrators to configure the perimeter (boundary routers) of the scoped region. This is called administratively scoped or local scope. At present, this is achieved by configuring multicast border routers (M-BRs) on a scope boundary with a boundary scope address range - so-called Administratively Scoped address range. Multicast traffic flows which are to be confined within a range must use a class-D address which is within the range. M-BRs are an impermeable boundary to any multicast packet with a class-D destination address that falls within any of its configured Administratively Scoped address ranges.

It is perfectly feasible for SM to use exactly the same mechanism for achieving multicast scoping. However, multicast scoping as it is currently defined requires a significant amount of configuration, as well as co-ordination of the address space for defining scope boundary ranges. Any mis-configurations can lead to multicast packets "leaking" across boundaries they should not.

Multicast scope boundary configurations must conform to certain rules, such as the rule that boundaries must be completely contained within one another (the term "nesting", or "convex", are often used). The MZAP protocol [MZAP] is implemented on M-BRs to detect inconsistent administratively scoped boundary configurations. As such it is essentially a network management tool, it does not correct mis-configurations.

In SM, the group address (C,M) is scoped according to the unicast core address C. The advantage of this compared to Administratively Scoped IP Multicast [RFC2365] is there is no requirement for these

scoped addresses to be dynamically assigned (via AAP or MAAS) or announced in the scoped regions (MZAP).

2.8.1 Multicast Scoping using unicast boundaries and scope mask

SM has the unique ability to take advantage of the unicast routing system boundaries (e.g. subnet, area, AS, AS-Confederation etc.) and use these as "natural" boundaries for multicast traffic, obviating the need for the configuration of explicit multicast boundaries. Furthermore, one group identifier (C, M) can be used with multiple scopes. It works as follows: assume a (C, M) group identifier is to be used for scopes A and B, with A nested inside B. A and B are natural unicast routing boundaries, e.g. area, and AS. A unicast routing system boundary is implicitly identified by a router aggregating routing information before propagating it over outgoing interfaces; this is achieved by shortening a prefix mask. For example, routing information inside boundary A has an associated mask of 24 bits. The boundary router between A and B reduces this is to 16 bits before propagating inside B.

Now, if a SM data packet carried a "scope mask(len)" in the SM header, the data packet would not pass beyond any unicast routing system boundary that itself propagates a shorter mask in unicast route updates it sends. The general rule is: a SM data packet carrying a "scope mask(len)" is only forwarded over those interfaces that aggregate unicast routing information using a mask which is equal length or longer than that specified in the SM data packet header.

```
                                |
                        (c) /16 | (d) /12
                                |
                        --------+-------
                        (a) /8  | (b) /20
                                |
                                |
```

The figure above illustrates a router with 4 interfaces, a, b, c, d, each which is aggregating routes with the respective prefix. If a SM data packet arrives on interface (b) carrying a "scope mask(len)" of 12, it is forwarded only over interface (c) and (d).

2.8.2 Multicast Scoping using private network boundaries

A multicast session can be scoped within a private network if the core address belongs to the private address space and is not

translated to any global address. In this case the boundary routers can be the filtering or NAT devices at the edge of the network. Since NAT devices can scope the addresses, the SM data packet itself does not have to carry the scope mask in the SM header.

Note that for administrative scoping purposes, the function in the NAT device which is of interest here is the filtering and address space separation function, not the address translation function. An public node will not be able to join n private core if the private core address is not mapped to any global address. As a result, no data packets for this scoped core will be forwarded out of the NAT device.

If the boundary routers are NAT devices, there is no requirement for the NAT devices to be SM-enabled (i.e. it knows how to translate SM specific packets) for the purpose of scoping SM groups. If the NAT is not SM-enabled, the join message will be filtered according to the core (IP destination) address and hence forwarding states for (C,G) will only be created in the defined scope. If the NAT device is SM-enabled, data packets can be filtered based on the core address C or the source address. In the case of SM dense mode, C=255.255.255.255. If the NAT device is not SM-enabled, since the IP destination address=255.255.255.255, the packets will be filtered. Hence SM dense-mode traffic is scoped by default, i.e. no dense-mode data packets will be forwarded across any boundary. If the NAT device is SM-enabled, a dense-mode data packet is scoped according to its IP source address. Source address is scoped in the same manner as core address.

If two scoped regions intersect topologically, then the address space in the overlapped region cannot be used by the outer scope, as stated in RFC2365. This applies here as well, i.e. a scoped group address cannot have its core address in the address space of the overlapped region, to avoid the problem of the same (C,M) belonging to different scopes at the intersecting boundary. This implies a core address C, scoped within scope X, where scope X is inside scope Y, should be unique within scopes X and Y; and no core within scope Y should have that same address C. Further, any other addresses scoped within X should not be visible to scope Y; all addresses scoped within Y is visible to scope X. This address separation is already maintained by NAT devices.

2.8.3 Multicast Scoping in IPv6

In IPv6, if a core address is a site-local scope address, then the corresponding (C,*) will be site-local scope as well,

2.9 Additional Features

We are investigating the following additional features, which are not available in other multicast protocols:

- the ability to select dense-mode. Currently there are routers that implement dense mode and routers that implement sparse mode, and typically a domain will implement either sparse or dense mode. There is no way to choose, per application, which type of tree is more appropriate.

There are cases in which dense mode makes more sense for an application. For example, dense mode is more appropriate if the number of receivers is so dense that there is very little optimization gained by creating a tree. Dense mode is also appropriate when the volume of data is sufficiently low that optimizing its delivery is not worth the overhead of creating and maintaining a tree.

With SM we use the convention of core=FF:FF:FF:FF to indicate the packet should be sent via dense-mode. For such packets no tree is formed and routers merely forward the packet using reverse path forwarding. As in DVMRP, states (S,M), where S is the source IP address, are created for dense mode groups.

Routers find out whether their neighbors support SM, and other characteristics of their neighbors, through Hello messages. A dense mode SM-packet should only be sent to SM-aware neighbors. As with DVMRP, tunnels can be configured between SM-aware nodes to enable a wider range for delivery of dense-mode SM packets.

- the ability to join a set of groups. The join message contains (C, M, mask). That facilitates having content parameterized by M. For instance, if the set of groups (C,*) is for stock information, certain bits in M can encode industry, country, etc. To receive information about all stocks, join (C,*). To receive some subset, join a more specific (M, mask) for core C.

2.10 SM Issues

2.10.1 Host API and Kernel Changes

The SM architecture require changes to the host Application Programming Interface (API) and kernel. Host may join a group using either SM Join - where hosts send joins similarly to an SM router or IGMP extended to carry the core address as well as a class-D address. As noted before, host SM Join should only be used where appropriate

e.g. when there is no local SM support.

Taking the BSD Sockets API as an example, joining a group is achieved using a system call; the data structure passed with the system call as an argument only supports the specification of a class-D address and interface (IP) address. For SM this data structure needs modifying to include a core address element, which can be concatenated with the class-D address to form SM's 8 byte group identifier. The kernel SM software, or IGMP software, can then make use of this information to generate a SM join message, or IGMP Report, respectively.

Similarly, when data is sent to a group, the data structure passed to the send system call must include a core address. The kernel SM software can then place this core address in the SM header. When an SM packet (identified by the IP protocol field) is received, the kernel SM software is invoked and the SM header is decapsulated before being send to the upper layer.

2.10.1.1 Extending IGMP

While not necessary, we propose using TLV in IGMP Membership Report messages. It is anticipated that IGMP will be extended for various purposes in future. The use of TLV will facilitate that.

In addition to the class-D address, a field called the extended address field, for lack of a better term, is defined to carry the additional address require in IGMPv3, Express, SM and Distributed Core Multicast (DCM). The IGMP Membership Report message is encoded as follow:

Type	Value
Classic:	S,G (if IGMPv3 with source specific joins)
Express:	S,E
Simple:	C,M
DCM:	(S),G where S is a list of channels Hence the extended

address field carries: i) the source address for classical IP multicast (IGMPv3 with source specific joins) ii) the source address for Express iii) the core address for SM iv) the pointer to a list of channels for DCM.

Extending IGMP is perfectly feasible - it has been done before in upgrading from IGMPv1 to IGMPv2, and changes will be required for IGMPv3 if it gains wider acceptance. The kernel modifications required to support SM are mainly to handle the additional address field. The host API change itself require only the addition of two parameters. We do not, therefore, consider host changes as barriers to SM deployment.

2.10.2 Layer 2 Filtering

In conventional IP multicast, each class D could be mapped to a distinct MAC address if 28 bits were available at the MAC layer for mapping. However, since only 23 bits of the MAC address is used for mapping, 32 IP multicast address could potentially be mapped to one MAC layer address. Hence higher layer filtering of multicast packets is required.

If the low-order 4 bytes of the SM group identifier - the class-D address, is similarly mapped, there is the potential for each of a subnet's hosts to join different SM groups, with their group-ids differing only in the core address portion of the group-id. In this worst-case scenario the transmission of packets to one group will be received by hosts belonging to all other SM groups on the subnet; a group's packets only become distinguishable at the hosts' network layers. In a more realistic case we might reasonably expect only a small percentage of a subnet's hosts to receive packets unnecessarily.

One possible way to reduce the amount of filtering at the network layer, would be to statically map the core address to a multicast layer 2 address if we assume groups associated with a core are likely to be related. This would still potentially incur higher layer filtering of undesired groups, but only those hosts subscribed to group(s) associated with a particular core would be affected.

The problem of mapping a larger-than-usual network identifier to a layer 2 address is not unique to SM - the problem manifests itself in IPv6 and EXPRESS.

One possible way of guaranteeing layer-2 multicast destination address uniqueness would have special node(s) map unique layer 2 address to the group-id. Before a node could send, receive or forward data, it has to obtain the layer 2 address. IGMP can be extended for this purpose.

Another possible solution is to have hardware filter based on a group address at a specific offset and of a specific length. The NIC would be snooping the IP header, but software should be able to program it to filter addresses at the desired offset.

3.0 Packet formats

This section describes all the packet formats. Simple Multicast could be implemented as very small modifications to PIM, CBT, or BGMP.

The packet types are:

- data packet

- join-request

- join-ack

- keep-alive (sent by child to parent)

- heartbeat (sent by parent to child)

- flush-tree (sent by parent to child after a loop is detected, to clear out state from looped tree as quickly as possible and cause subtree to be reformed)

For all control packets (JOIN-REQUEST, JOIN-ACK, KEEP-ALIVE, HEARTBEAT, FLUSH- TREE), the "Protocol" field in the IPv4 header is set to SM (a new protocol field).

3.1 SM-'tunnels'

Upstream (towards the core) or downstream SM routers may not be immediate neighbors, if there are non-SM routers on the path between them. In a traditional tunnel between R1 and R2, R1 must add an extra IP header, and R2 must delete the header. SM gets the same functionality without adding and deleting headers. Instead all that is needed is to overwrite the destination address in the IP header to the address of the "tunnel" endpoint. The reason this can be done is that the information necessary for SM-routers to route the packet (namely C and M) are contained in the SM header.

JOIN-REQUESTs and JOIN-ACKs allow tunnel-endpoints to learn of each other. The state for a "tunnel" consists of the IP address of the endpoint, and the number of actual IP hops in the tunnel. The purpose of keeping the count of the tunnel's hops is because SM counts the length of the tree, so that senders can know what to set as the TTL in data packets.

3.2 Data Packet Header

IP Header

```
0               1               2               3^M
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1^M
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+^M
|Version|  IHL  |Type of Service|          Total Length         |^M
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+^M
|         Identification        |Flags|      Fragment Offset    |^M
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+^M
|  Time to Live |   Protocol =  |         Header Checksum       |^M
|               | IPPROTO_SM    |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+^M
|                       Source Address                          |^M
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+^M
|                    Destination Address                        |^M
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+^M
```

SM Header

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+^M
|                         Core Address                          |^M
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+^M
|                     Multicast Address                         |^M
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+^M
|Protocol=egUDP| Core Mask      |                             |L|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+^M
```

This SM header includes C, M, loop detect flag, where C=FF:FF:FF:FF ^M indicates packet should be delivered dense-mode.^M

The 'L' bit in Flag, if set, indicates the TTL for this packet should never reach 0 (See Loops).^M
^M
The IP Destination address is ALL-SM-NODES except in the following cases:^M
^M
- when a non-member sender transmits the packet, the destination is set to the core address. The purpose of this is to enable the packet^M to be unicasted until it hits a node that is SM-aware, at which point the packet is multicast along the tree from the point at which it entered the tree.

Note that if the non-member sender has joined the group as a 'sender-only' (c.f. uni-directional join in CBT), then the destination address in the data packet is either ALL-SM-NODES or the tunnel endpoint (as described below).

- when the packet is transmitted on a tunnel port, in which case the^M destination address is set to the IP address of the tunnel endpoint.^M

Note that at Layer 2, the MAC address is mapped to the Multicast Address M of the group (C,M), not to ALL-SM-NODES.^M

3.2 JOIN-REQUEST

The following control packet header fields are as defined in CBT: addr_len, checksum, Payload Length and # of options.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  vers |type=1 |  addr len     |          checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Payload Length |  # of options |            reserved           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Join Originating Router                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        core address C                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Multicast address M                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Multicast address mask m               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  option type  |  option len   |        option value...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The destination IP address in the IP header is the Core Address. The JOIN-REQUEST is sent with the Router Alert Option.

The Multicast address and corresponding mask (M,m) may appear multiple times. The total length of these fields is specified in the "addr_len" field of the common control header.

The JOIN-REQUEST may contain the following option:

- Originating TTL. This field is set to the TTL in the IP header of this JOIN- REQUEST packet. The receiving SM router ignores this option unless the control packet is from a SM router who is not an immediate neighbor. The value in this field is used to calculate the number of hops in a 'tunnel' = Originating TTL - TTL in the IP header for this packet. The value derived is placed in "# of hops in tunnel from you to me" in the JOIN-ACK message.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     1         |       2       |        Originating TTL        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Sender-Only
The join would only be successful if the sender is on the Include

Senders List or NOT in the Exclude Senders List.
The sender is attached to the tree as per uni-directional Join in CBT.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     2         |       2       |       Reserved                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.3 JOIN-ACK

```
0               1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  vers |type=2 |  addr len     |         checksum              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Payload Length |  # of options |    # of hops in 'tunnel'      |
|               |               |       from you to me          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Join Originating Router                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       core address C                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Multicast address M                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Multicast address mask m                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  option type  |  option len   |        option value...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The destination IP address in the IP header is the downstream IP source address of the JOIN-REQUEST. The JOIN_ACK is sent with the Router Alert Option.

The Multicast address and corresponding mask (M,m) may appear multiple times. The total length of these fields is specified in the "addr_len" field.

The field "# of hops in tunnel from you to me" is ignored unless the control packet is from a SM router who is not an immediate neighbor. The value in this field is saved as state for this tunnel port.

The options from the JOIN-REQUEST are copied into the JOIN-ACK, with the exception of the "Originating TTL" option. The Originating TTL is set to the TTL in the IP header of this JOIN-ACK packet.

3.4 KEEP-ALIVE

```
0               1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  vers | type=3|  addr len     |         checksum              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Payload Length |  # of options |        reserved               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                KEEP-ALIVE Originating Router                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       core address C                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Multicast address M                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Multicast address mask m                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  option type  |  option len   |        option value...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The keep-alive message is sent from a child to a parent (towards core), and is sent only if a keep-alive has been received recently from a child. The destination IP address in the IP header is ALL-SM-NODES or the tunnel endpoint address.

A single keep-alive can serve as many groups as fit into the list in the packet.

(M,m) may appear multiple times. The total length of these fields is specified in the "addr_len" field.

The KEEP-ALIVE may contain the following options:

```
0               1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     1         |       10      |I|     reserved flag bits      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Include/Exclude Sender Prefix                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Include/Exclude Sender Mask                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Include/Exclude Senders List that upstream routers should filter. This option may appear multiple times. The 'I' bit is set if this is an include sender list, and is zero if this is an exclude sender list.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     2         |       10      |        hop count              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Prune Time                |   # of hops in 'tunnel'       |
|                               |       from you to me          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- KEEP-ALIVE Option. This option should appear the same number of times as the address set (C,M,mask). It corresponds and is applicable to the address set (C,M,mask).

The fields in this option are: - Number of hops to furthest leaf for (C,M,mask), hop count. The hop count is incremented at every SM hop. In addition, when the KEEP-ALIVE is received from a tunnel port, hop count = hop count + number of hops in 'tunnel'.

- Prune Time for (C,M,mask), time after which, if no KEEP-ALIVE is received for group (C1, M, mask), the parent should prune off this branch.

- 'Originating TTL'. This is as described in JOIN-REQUEST.

3.5 HEARTBEAT

```
   0               1               2               3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  vers | type=4|  addr len     |         checksum              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Payload Length |  # of options |      reserved                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                 HEARTBEAT Originating Router                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       core address C                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Multicast address M                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Multicast address mask m                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  option type  |  option len   |        option value...        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The heartbeat is sent by a parent to a child. It is sent periodically regardless of whether heartbeat is received from its parent. The destination IP address is set to ALL-SM-NODES or the tunnel endpoint address.

The HEARTBEAT may contain the following additional options: - Include/Exclude Senders List. This is the list of allowed/prohibited senders to the group. The format of this option is the same the KEEP-ALIVE Include/Exclude Senders List, although it serves as a different purpose here.

- spin-off groups (Ci,Mi). One or more spin-off groups (Ci,Mi) may be specified.

```
0               1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     1         |  #Groupsx8    |      reserved flag bits       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Core Address  Ci                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Multicast Address Mi                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- HEARTBEAT Option. This option should appear the same number of times as the address set (C,M,mask). It corresponds and is applicable to the address set (C,M,mask).

The fields in this option are:

```
0               1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     2         |       6       |       core distance           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Time To Shutdown          |   # of hops in 'tunnel'       |
|                               |     from you to me            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|A|                    reserved                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- distance from core. Number of hops to core (C,M,mask), core distance. The core distance is incremented at every SM hop. In addition, when the KEEP-ALIVE is received from a tunnel port, core distance = core distance + number of hops in 'tunnel' - Time left before group should be closed down. (all 'ones' indicates group should not be torn down) - The 'A' bit if set indicates the core is alive or reachable

- 'Originating TTL'. This is as described in JOIN-ACK.

3.6 FLUSH-TREE

```
0               1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  vers | type=5|  addr len     |         checksum               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Payload Length |  # of options |           reserved             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 HEARTBEAT Originating Router                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       core address C                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Multicast address M                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Multicast address mask m                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  option type  |  option len   |        option value...         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The destination IP address is set to ALL-SM-NODES or the tunnel endpoint address.

The Multicast address and corresponding mask (M,m) may appear multiple times. The total length of these fields is specified in the "addr_len" field of the common control header.

No options are currently defined.

4 Acknowledgments

Many people have contributed ideas to this proposal, including Harald Alvastrand, Joel Halpern and Fred Baker. The fact that SM is based on previous work in IP Multicast implies that the authors are grateful to everyone who has contributed to the development of IP Multicast. We would like to thank all members of IDMR, in particular Dino Farinacci, Mark Handley, Brad Cain, Dave Thaler Russ White and Ken Carlberg whose helpful comments have improved this proposal. Others that have provided helpful technical information include Matthew Yuen, Patrick Lee.

References

DNS Based RP Placement scheme
Dino Farinacci's presentation in the MBONED WG, 40th IETF Meeting

Static Multicast, Internet-Draft, March 1998

M. Ohta, J. Crowcroft

Express
IDMR Mailing List discussion

CBT, Core Based Tree Multicast Routing,
Internet-Draft, March 1998
Ballardie, Cain, Zhang

PIM-SM, Protocol independent multicast-sparse mode Specification,
RFC-2117, June 1997
Estrin, Farinacci, Helmy, Thaler, Deering, Handley,
Jacobson, Liu, Sharma, and Wei.

BGMP, Border Gateway Multicast Protocol Specification,
Internet-Draft, March 1998
Thaler, Estrin, Meyers

MASC, Multicast Address Set Claim Protocol,
Internet-Draft, November 1997
Estrin, Handley, Kumar, Thaler

IGMP, Internet Group Management Protocol, Version 3,
Internet-Draft, November 1998
Cain, Deering, Thyagarajan

"A Border Gateway Protocol 4 (BGP-4)", Y. Rekhter & T. Li,
RFC1771, March 1995

"Multiprotocol Extensions for BGP-4", RFC 2283, February 1998.
Bates, T., Chandra, R., Katz, D., and Y. Rekhter,

"The IP Network Address Translator (NAT)" RFC 1631, May 1994.
RFC1631 Egevang, K., Francis, P.,

"Administratively Scoped IP Multicast",
RFC 2365, July 1998. Meyer, D.,

Distributed Core Multicast, L. Blazevic, J-Y. Boudec

OGMP ftp://cs.ucl.ac.uk/darpa/ogmp.ps.gz

Authors' Addresses

Radia Perlman
Sun Microsystems Laboratories
2 **Elizabeth Drive**
Chelmsford, MA 01824
Radia.Perlman@sun.com

Cheng-Yin Lee
Nortel Networks
PO Box 3511, Station C
Ottawa, ON K1Y 4H7, Canada
leecy@nortel.com

Tony Ballardie
Research Consultant
aballardie@acm.org

Jon Crowcroft
Department of Computer Science
University College London
Gower Street
London, WC1E 6BT, UK
J.Crowcroft@cs.ucl.ac.uk

Zheng Wang
Bell Labs Lucent Technologies
101 **Crawfords Corner Road**
Holmdel NJ 07733
zhwang@bell-labs.com

Thomas Maufer
3Com Corporation
5400 **Bayfront Plaza**
Santa Clara, CA 95052
maufer@3com.com

Christophe Diot
Sprint ATL
1 **Adrian Court**
Burlingame CA 94010
USA
cdiot@sprintlabs.com

Joseph Thoo
Nortel Networks
PO Box 3511, Station C
Ottawa, ON K1Y 4H7, Canada

jthoo@nortel.com

Mark Green
@Home Networks
markg@corp.home.net