INTERNET DRAFT
<draft-ooms-xcast-basic-spec-13.txt>

R. Boivie, N. Feldman
IBM
Y. Imai
WIDE / Fujitsu
W. Livens
ESCAUX
D. Ooms
OneSparrow
O. Paridaens
Alcatel

July, 2007
Expires January, 2008

Explicit Multicast (Xcast) Concepts and Options

Status of this Memo

Copyright Notice

Abstract

While traditional IP multicast schemes [1112] are scalable for very large multicast groups, they have scalability issues with a very large number of distinct multicast groups. This document describes

Xcast (Explicit Multi-unicast (Xcast)), a new multicast scheme with complementary scaling properties: Xcast supports a very large number of small multicast sessions. Xcast achieves this by explicitly encoding the list of destinations in the data packets, instead of using a multicast group address.

This document discusses Xcast concepts and options in several areas; it does not provide a complete technical specification.

Table of Contents

16. Informative Reference

1. Introduction

While traditional IP multicast schemes [1112] are scalable for very large multicast groups, they have scalability issues with a very large number of distinct multicast groups. This document describes Xcast (Explicit Multi-unicast (Xcast)), a new multicast scheme with complementary scaling properties: Xcast supports a very large number of small multicast sessions. Xcast achieves this by explicitly encoding the list of destinations in the data packets, instead of using a multicast group address. This document discusses Xcast concepts and options in several areas; it does not provide a complete technical specification.

Multicast, the ability to efficiently send data to a group of destinations, is becoming increasingly important for applications such as IP telephony and video-conferencing.

Two kinds of multicast seem to be important: a broadcast-like multicast that sends data to a very large number of destinations, and a "narrowcast" multicast that sends data to a fairly small group. An example of the first is the audio and video multicasting of a presentation to all employees in a corporate intranet. An example of the second is a videoconference involving 3 or 4 parties. For reasons described below, it seems prudent to use different mechanisms for these two cases. As the reliable multicast transport group has stated: "it is believed that a 'one size fits all' protocol will be unable to meet the requirements of all applications" [RMT]. Note that the 1998 IAB Routing Workshop [2902] came to the same conclusion: "For example, providing for many groups of small conferences (a small number of widely dispersed people) with global topological scope scales badly given the current multicast model".

Today's multicast schemes can be used to minimize bandwidth consumption. Explicit Multi-Unicast (Xcast) also can be used to minimize bandwidth consumption for "small groups". But it has an additional advantage as well. Xcast eliminates the per session signaling and per session state information of traditional IP multicast schemes and this allows Xcast to support very large numbers of multicast sessions. And this scalability is important since it enables important classes of applications such as IP telephony, videoconferencing, collaborative applications, networked games etc. where there are typically very large numbers of small multicast groups.

Interestingly, the idea for Xcast has been around for some time although this was not immediately known to the 3 groups that independently re-invented it in the late 1990's. In fact the very first proposal of the multicast concept in the Internet community, by Lorenzo Aguilar in his 1984 SIGCOMM paper [AGUI] proposed the use of an explicit list of destinations discussed in more detail below. At about the same time, David Cheriton and Stephen Deering developed Host Group Multicast in 1985 [CHER].

The Internet community compared the 2 proposals and concluded that a single mechanism was preferable to multiple mechanisms. Further, since Aguilar's proposal seemed to have serious scaling problems, the Host Group model was adopted.

However for reasons described below, we believe it makes sense to use different mechanisms for the two different kinds of multicast discussed above. While Host Group multicast may have been sufficient in the Internet of 1985, we believe that Xcast can be an important complement to Host Group multicast in the Internet of the 21st century.

2. Xcast Overview

In this document the following terminology will be used:

- Session: in Xcast the term 'multicast session' will be used instead of 'multicast group' to avoid the strong association of multicast groups with multicast group addresses in traditional IP multicast.

- Channel: in a session with multiple senders (e.g. a video conference), the flow sourced by one sender will be called a channel. So a session can contain one or more channels.

In the Host Group Model the packet carries a multicast address as a logical identifier of all group members. In Xcast, the source node keeps track of the destinations in the multicast channel that it wants to send packets to.

The source encodes the list of destinations in the Xcast header, and then sends the packet to a router. Each router along the way parses the header, partitions the destinations based on each destination's next hop, and forwards a packet with an appropriate Xcast header to each of the next hops.

When there is only one destination left, the Xcast packet can be converted into a normal unicast packet, which can be unicasted along the remainder of the route. This is called X2U (Xcast to Unicast).

For example, suppose that A is trying to get packets distributed to B, C & D in Figure 1 below:

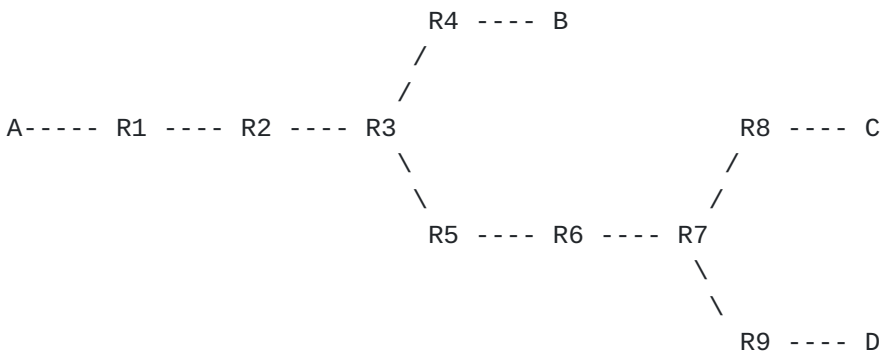

Figure 1

This is accomplished as follows: A sends an Xcast packet with the list of destinations in its Xcast header to the first router, R1.

Since the Xcast header will be slightly different for IPv4 and IPv6 we won't reveal any details on the encoding of the Xcast header in this section (see section 9). So, ignoring the details, the packet that A sends to R1 looks like this:

[src = A | dest = B C D | payload]

When R1 receives this packet, it needs to properly process the Xcast header. The processing that a router does on receiving one of these Xcast packets is as follows:

- Perform a route table lookup to determine the next hop for each of the destinations listed in the packet.

- Partition the set of destinations based on their next hops.

- Replicate the packet so that there's one copy of the packet for each of the next hops found in the previous steps.

- Modify the list of destinations in each of the copies so that the list in the copy for a given next hop includes just the destinations that ought to be routed through that next hop.

- Send the modified copies of the packet on to the next hops.

- Optimization: If there is only one destination for a particular next hop, the packet can be sent as a standard unicast packet to the destination (X2U).

So, in the example above, R1 will send a single packet on to R2 with a destination list of < B C D > and R2 will send a single packet to R3 with the same destination list.

When R3 receives the packet, it will, by the algorithm above, send one copy of the packet to next hop R5 with an Xcast list of < C D >, and one ordinary unicast packet addressed to < B > to R4. R4 will receive a standard unicast packet and forward it on to < B >. R5 will forward the Xcast packet that it receives on to R6 which will pass it on to R7. When the packet reaches R7, R7 will transmit ordinary unicast packets addressed to < C > and < D > respectively. R8 and R9 will receive standard unicast packets, and forward the packets on to < C > and < D > respectively.

It's important that the Xcast packet that is sent to a given next hop only includes destinations for which that next hop is the next hop listed in the route table. If the list of destinations in the packet sent to R4, for example, had also included C and D, R4 would send duplicate packets.

Note that when routing topology changes, the routing for an Xcast channel will automatically adapt to the new topology since the path an Xcast packet takes to a given destination always follows the ordinary, unicast routing for that destination.

3. The cost of the traditional IP multicast schemes

Traditional IP multicast schemes [DEER, DEE2, FARI] were designed to handle very large multicast groups. These work well if one is trying to distribute broadcast-like channels all around the world but they have scalability problems when there is a very large number of groups.

The characteristics of the traditional IP multicast model are determined by its two components: the Host Group model [DEER] and a Multicast Routing Protocol. Both components make multicast very different from unicast.

In the Host Group model, a group of hosts is identified by a multicast group address, which is used both for subscriptions and forwarding. This model has two main costs:

- Multicast address allocation: The creator of a multicast group must allocate a multicast address which is unique in its scope (scope will often be global). This issue is being addressed by the Malloc working group, which is proposing a set of Multicast Address Allocation Servers (MAAS) and three protocols (MASC, AAP,

MADCAP).

- Destination unawareness: When a multicast packet arrives in a router, the router can determine the next hops for the packet, but knows nothing about the ultimate destinations of the packet, nor about how many times the packet will be duplicated later on in the network. This complicates the security, accounting and policy functions.

In addition to the Host Group model, a routing algorithm is required to maintain the member state and the delivery tree. This can be done using a (truncated) broadcast algorithm or a multicast algorithm [DEER]. Since the former consumes too much bandwidth by unnecessarily forwarding packets to some routers, only the multicast algorithms are considered. These multicast routing protocols have the following costs:

- Connection state: The multicast routing protocols exchange messages that create state for each (source, multicast group) in all the routers that are part of the point-to-multipoint tree. This can be viewed as "per flow" signaling that creates multicast connection state, possibly yielding huge multicast forwarding tables. Some of these schemes even disseminate this multicast routing information to places where it isn't necessarily needed [1075]. Other schemes try to limit the amount of multicast routing information that needs to be disseminated, processed and stored throughout the network. These schemes (e.g. [2201]) use a "shared distribution tree" that is shared by all the members of a multicast group and they try to limit the distribution of multicast routing information to just those nodes that "really need it". But these schemes also have problems. Because of the shared tree, they use less than optimal paths in routing packets to their destinations and they tend to concentrate traffic in small portions of a network. And these schemes still involve lots of "per flow" signaling and "per flow" state.

- Source advertisement mechanism: Multicast routing protocols provide a mechanism by which members get 'connected' to the sources for a certain group without knowing the sources themselves. In sparse-mode protocols [2201, DEE2], this is achieved by having a core node, which needs to be advertised in the complete domain. On the other hand, in dense-mode protocols [1075] this is achieved by a "flood and prune" mechanism. Both approaches raise additional scalability issues.

- Interdomain routing: Multicast routing protocols that rely on a core node [2201, DEE2] additionally need an interdomain multicast routing protocol (e.g. [FARI]).

The cost of multicast address allocation, destination unawareness and the above scalability issues lead to a search for other multicast schemes. Source-Specific Multicast (SSM) [HOLB] addresses some of the above drawbacks: in SSM a host joins a specific source, thus the channel is identified by the couple (source address, multicast address). This approach avoids multicast address allocation as well as the need for an interdomain routing protocol. The source advertisement is taken out of the multicast routing protocol and is moved to an out-of-band mechanism (e.g. web page).

Note that SSM still creates state and signaling per multicast channel in each on-tree node. Figure 2 depicts the above costs as a function of the number of members in the session or channel. All the costs have a hyperbolic behavior.

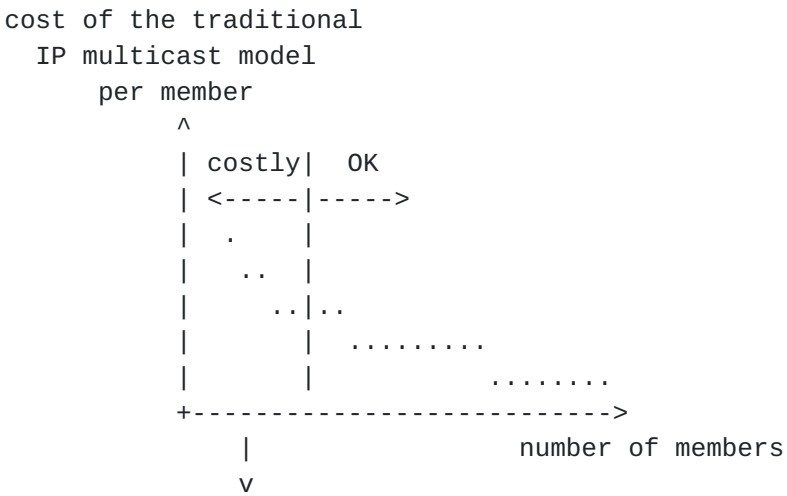

Figure 2

The traditional IP multicast model becomes expensive for its members if the groups are small. Small groups are typical for conferencing, gaming and collaborative applications. These applications are well-served by Xcast.

In practice, traditional IP multicast routing protocols impose limitations on the number of groups and the size of the network in which they are deployed. For Xcast these limitations do not exist.

4. Motivation

Xcast takes advantage of one of the fundamental tenets of the Internet "philosophy", namely that one should move complexity to the edges of the network and keep the middle of the network simple. This

is the principle that guided the design of IP and TCP and it's the principle that has made the incredible growth of the Internet possible. For example, one reason that the Internet has been able to scale so well is that the routers in the core of the network deal with large CIDR blocks as opposed to individual hosts or individual "connections". The routers in the core don't need to keep track of the individual TCP connections that are passing through them. Similarly, the IETF's diffserv effort is based on the idea that the routers shouldn't have to keep track of a large number of individual RSVP flows that might be passing through them. It's the authors' belief that the routers in the core shouldn't have to keep track of a large number of individual multicast flows either.

Compared to traditional IP multicast, Xcast has the following advantages:

1) Routers do not have to maintain state per session (or per channel) [SOLA]. This makes Xcast very scalable in terms of the number of sessions that can be supported since the nodes in the network do not need to disseminate or store any multicast routing information for these sessions.

2) No multicast address allocation required.

3) No need for multicast routing protocols (neither intra- nor interdomain). Xcast packets always take the "right" path as determined by the ordinary unicast routing protocols.

4) No core node, so no single point of failure. Unlike the shared tree schemes, Xcast minimizes network latency and maximizes network "efficiency".

5) Symmetric paths are not required. Traditional IP multicast routing protocols create non-shortest-path trees if paths are not symmetric. (A path between two nodes A and B is symmetric if the path is both the shortest path from A to B as well as the shortest path from B to A.) It is expected that an increasing number of paths in the Internet will be asymmetric in the future as a result of traffic engineering and policy routing, and thus the traditional IP multicast schemes will result in an increasing amount of suboptimal routing.

6) Automatic reaction to unicast reroutes. Xcast will react immediately to unicast route changes. In traditional IP multicast routing protocols a communication between the unicast and the multicast routing protocol needs to be established. In many implementations this is on a polling basis, yielding a slower reaction to e.g. link failures. It may also take some time for traditional IP multicast routing protocols to fix things up if there

is a large number of groups that need to be fixed.

7) Easy security and accounting. In contrast with the Host Group Model, in Xcast all the sources know the members of the multicast channel, which gives the sources the means to e.g. reject certain members or count the traffic going to certain members quite easily. Not only a source, but also a border router is able to determine how many times a packet will be duplicated in its domain. It also becomes easier to restrict the number of senders or the bandwidth per sender.

8) Heterogeneous receivers. Besides the list of destinations, the packet could (optionally) also contain a list of DiffServ CodePoints (DSCPs). While traditional IP multicast protocols have to create separate groups for each service class, Xcast incorporates the possibility of having receivers with different service requirements within one multicast channel.

9) Xcast packets can make use of traffic engineered unicast paths.

10) Simple implementation of reliable protocols on top of Xcast, because Xcast can easily address a subset of the original list of destinations to do a retransmission.

11) Flexibility (see section 6).

12) Easy transition mechanisms (see section 11).

It should be noted that Xcast has a number of disadvantages as well:

1) Overhead. Each packet contains all remaining destinations. But, the total amount of data is still much less than for unicast (payload is only sent once). A method to compress the list of destination addresses might be useful.

2) More complex header processing. Each destination in the packet needs a routing table lookup. So an Xcast packet with n destinations requires the same number of routing table lookups as n unicast headers. Additionally, a different header has to be constructed per next hop. Note however that:

a) Since Xcast will typically be used for super-sparse sessions, there will be a limited number of branching points, compared to non-branching points. Only in a branching point do new headers need to be constructed.

b) The header construction can be reduced to a very simple operation: overwriting a bitmap.

c) Among the non-branching points, a lot of them will contain only one destination. In these cases normal unicast forwarding can be applied.

d) By using a hierarchical encoding of the list of destinations in combination with the aggregation in the forwarding tables the forwarding can be accelerated ([OOMS]).

e) When the packet enters a region of the network where link bandwidth is not an issue anymore, the packet can be transformed by a Premature X2U. Premature X2U (see section 11.2) occurs when a router decides to transform the Xcast packet for one or more destinations into unicast packets. This avoids more complex processing downstream.

f) Other mechanisms to reduce the processing have been described in [IMAI] (tractable list) and [OOMS] (caching), but are not (yet) part of the Xcast specification.

3) Xcast only works with a limited number of receivers.

5. Application

While Xcast is not suitable for multicast sessions with a large number of members, such as the broadcast of an IETF meeting, it does provide an important complement to existing multicast schemes in that it can support very large numbers of small sessions. Thus Xcast enables important applications such as IP telephony, videoconferencing, multi-player games, collaborative e-meetings etc. The number of these sessions will become huge.

Some may argue that it is not worthwhile to use multicast for sessions with a limited number of members, and use unicast instead. But in some cases limited bandwidth in the "last mile" makes it important to have some form of multicast as the following example illustrates. Assume n residential users that set up a video conference. Typically access technologies are asymmetric (e.g. xDSL, GPRS or cable modem). So, a host with xDSL has no problem receiving n-1 basic 100kb/s video channels, but the host is not able to send its own video data n-1 times at this rate. Because of the limited and often asymmetric access capacity, some type of multicast is mandatory.

A simple but important application of Xcast lies in bridging the access link. The host sends the Xcast packet with the list of unicast addresses and the first router performs a Premature X2U.

Since Xcast is not suitable for large groups, Xcast will not replace the traditional IP multicast model, but it does offer an alternative for multipoint-to-multipoint communications when there can be very large numbers of small sessions.

6. Xcast Flexibility

The main goal of multicast is to avoid duplicate information flowing over the same link. By using traditional IP multicast instead of unicast, bandwidth consumption decreases while the state and signaling per session increases. Xcast has a cost of 0 in these 2 dimensions, but it does introduce a third dimension corresponding to the header processing per packet. This three dimensional space is depicted in Figure 3.

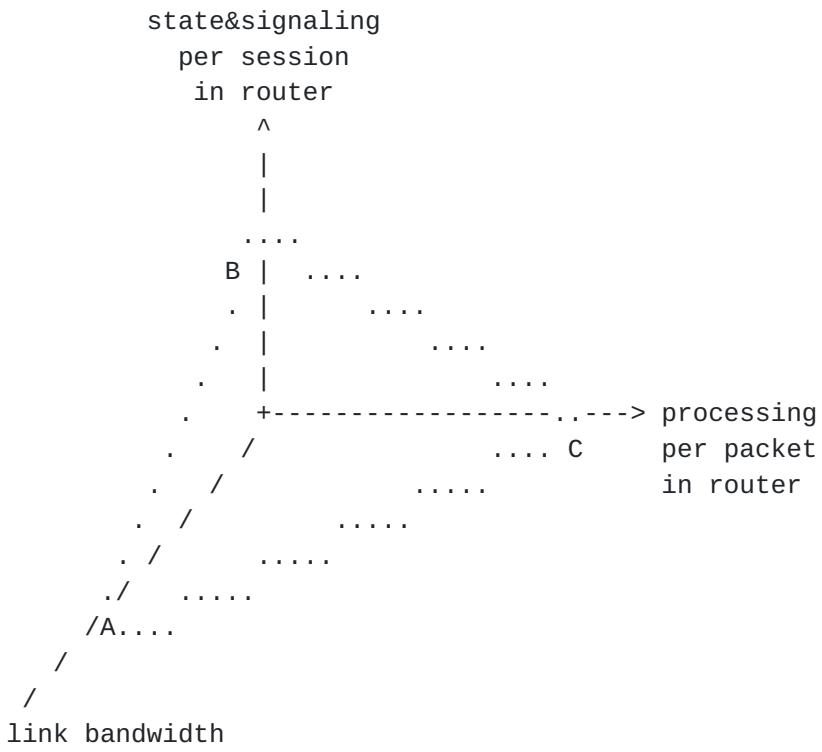

Figure 3

One method of delivering identical information from a source to n destinations is to unicast the information n times (A in Figure 3). A second method, the traditional IP multicast model (B in Figure 3) sends the information only once to a multicast address. In Xcast the information is sent only once, but the packet contains a list of destinations (point C).

The three points A, B and C define a plane (indicated with dots in Figure 3): a plane of conservation of misery. All three approaches have disadvantages. The link bandwidth is a scarce resource, especially in access networks. State&signaling/session encounters limitations when the number of sessions becomes large and an increased processing/packet is cumbersome for high link speeds.

One advantage of Xcast is that it allows a router to move within this plane of conservation of misery based upon its location in a network. For example in the core of the network, a cache could be used to move along the line from C to B without introducing any per-flow signaling. Another possibility, as suggested above, is to use premature X2U to move along the line from C to A in an access network if there is an abundance of bandwidth in the backbone.

7. Xcast Control Plane Options

Unlike traditional IP multicast schemes, Xcast does not specify a "control plane". There is no IGMP, and as mentioned above, there are no intradomain or interdomain multicast routing protocols. With Xcast, the means by which multicast sessions are defined is an application level issue and applications are not confined to the model in which hosts use IGMP to join a multicast session. For example:

- some applications might want to use an IGMP-like receiver-join model.

- other applications might want to use a model in which a user places a call to the party or parties that he or she wants to talk to (similar to the way that one puts together a conference call today using the buttons on one's telephone).

- one might define a session based on the cells that are close to a moving device in order to provide for a "smooth handoff" between cells when the moving device crosses cell boundaries.

- in some applications the members of the session might be specified as arguments on a command line.

- one might define an application that uses GPS to send video from a bank robbery to the 3 police cars that are closest to the bank being robbed.

Thus, the application developer is not limited to the receiver-initiated joins of the IGMP model. There will be multiple ways in which an Xcast sender determines the addresses of the members of the channel.

For the purpose of establishing voice and multimedia conferences over IP networks, several control planes have already been defined, including SIP [2543] and H.323[H323].

7.1. SIP Control Plane for Xcast

In SIP, a host takes the initiative to set up a session. With the assistance of a SIP server a session is created. The session state is kept in the hosts. Data delivery can be achieved by several mechanisms: meshed unicast, bridged or multicast. Note that for the establishment of multicast delivery, a multicast protocol and communication with Multicast Address Allocation Servers (MAAS) are still required.

In "meshed unicast" or "multi-unicasting", the application keeps track of the participants' unicast addresses and sends a unicast to each of those addresses. For reasons described in section 3, multi-unicasting rather than multicast is the prevalent solution in use today. It's a simple matter to replace multi-unicast code with Xcast code. All that the developer has to do is replace a loop that sends a unicast to each of the participants by a single "xcast_send" that sends the data to the participants. Thus it's easy to incorporate Xcast into real conferencing applications.

Both Xcast and SIP address super-sparse multicast sessions. It turns out that Xcast (a very flexible data plane mechanism) can be easily integrated with SIP (a very flexible control plane protocol). When an application decides to use Xcast forwarding it does not affect its interface to the SIP agent: it can use the same SIP messages as it would for multi-unicasting. SIP could be used with Xcast to support the conferencing model mentioned above in which a caller places a call to several parties.

7.2 Receiver-Initiated Join for Xcast

In the previous section, it was discussed how to establish an Xcast session among well known participants of a multi-party conference. In some cases, it is useful for participants to be able to join a session without being invited. For example, the chairman of a video chat may want to leave the door of their meeting open for newcomers. The IGMP-like receiver-initiated join model mentioned above can be implemented by introducing a server that hosts can talk to, to join a conference.

8. Optional information

8.1. List of ports

Although an extension to SIP could be arranged such that all participants in a session use the same transport (UDP) port number, in the general case it is possible for each participant to listen on a different port number. To cover this case, the Xcast packet optionally contains a list of port numbers.

If the list of port numbers is present, the destination port number in the transport layer header will be set to zero. On X2U the destination port number in the transport layer header will be set to the port number corresponding to the destination of the unicast packet.

8.2. List of DSCPs

The Xcast packet could (optionally) also contain a list of DiffServ CodePoints (DSCPs). While traditional IP multicast protocols have to create separate groups for each service class, Xcast incorporates the possibility of having receivers with different service requirements within one channel.

The DSCP in the IP header will be set to the most demanding DSCP of the list of DSCPs. This DSCP in the IP header will determine e.g. the scheduler to use.

If two destinations, with the same next-hop, have 'non-mergable' DSCPs, two Xcast packets will be created. 'Non-mergable' meaning that one can not say that one is more or less stringent than the other.

8.3. Channel Identifier

Optionally a sender can decide to add an extra number in the Xcast header: the Channel Identifier. If the source does not want to use this option it must set the Channel Identifier to zero. If the Channel Identifier is non-zero the pair (Source Address, Channel Identifier) must uniquely identify the channel (note that this is similar to the (S, G) pair in SSM). This document does not assign any other semantics to the Channel Identifier besides the one above.

This Channel Identifier could be useful for several purposes:

1) A key to a caching table [OOMS].

2) "Harmonization" when used with Host Group Multicast (to be discussed in greater detail in another document).

3) An identifier of the channel in error, flow control, etc. messages

4) It gives an extra de-multiplexing possibility (beside the port-

number)

5) ...

the size of the channel identifier and its semantics are TBD.

9. Possible Xcast Packet Encoding

9.1. General

The source address field of the IP header contains the address of the Xcast sender. The destination address field carries the All-Xcast-Routers address (to be assigned link-local multicast address), this is to have a fixed value. Every Xcast router joins this multicast group. The reasons for putting a fixed number in the destination field are:

1) The destination address field is part of the IP pseudo header and the latter is covered by transport layer checksums (e.g. UDP checksum). So the fixed value avoids a (delta) recalculation of the checksum.

2) The IPsec AH covers the IP header destination address hence preventing any modification to that field. Also, both AH and ESP payloads cover the whole UDP packet (via authentication and/or encryption). The UDP checksum cannot therefore be updated if the IP header destination address were to change.

3) In Xcast for IPv6 the Routing Extension shall be used, this header extension is only checked by a router if the packet is destined to this router. This is achieved by making all Xcast routers part of the All_Xcast_Routers group.

4) Normally Xcast packets are only visible to Xcast routers. However, if a non-Xcast router receives an Xcast packet by accident (or by criminal intent), it will not send ICMP errors since the Xcast packet carries a multicast address in the destination address field ([1812]).

Note that some benefits only hold when the multicast address stays in the destination field until it reaches the end-node (thus not combinable with X2U).

9.2. IPv4

[AGUI] and [1770] proposed (for a slightly different purpose) to carry multiple destinations in the IPv4 option. But because of the limited flexibility (limited size of the header), Xcast will follow

another approach. The list of destinations will be encoded in a separate header. The Xcast header for IPv4 (in short, Xcast4) would be carried between the IPv4 header and the transport layer header.

[IPv4 header | Xcast4 | transport header | payload]

Note also that since the Xcast header is added to the data portion of the packet, if the sender wishes to avoid IP fragmentation, it must take the size of the Xcast header into account.

9.2.1. IPv4 header

The Xcast4 header is carried on top of an IP header. The IP header will carry the protocol number listed as usable for experimental purposes in RFC [4727]. See also Section 15. The source address field contains the address of the Xcast sender. The destination address field carries the All_Xcast_Routers address.

9.2.2. Xcast4 header

The Xcast4 header is format depicted in Figure 4. It is composed of two parts: a fixed part (first 12 octets) and two variable length parts that are specified by the fixed part.

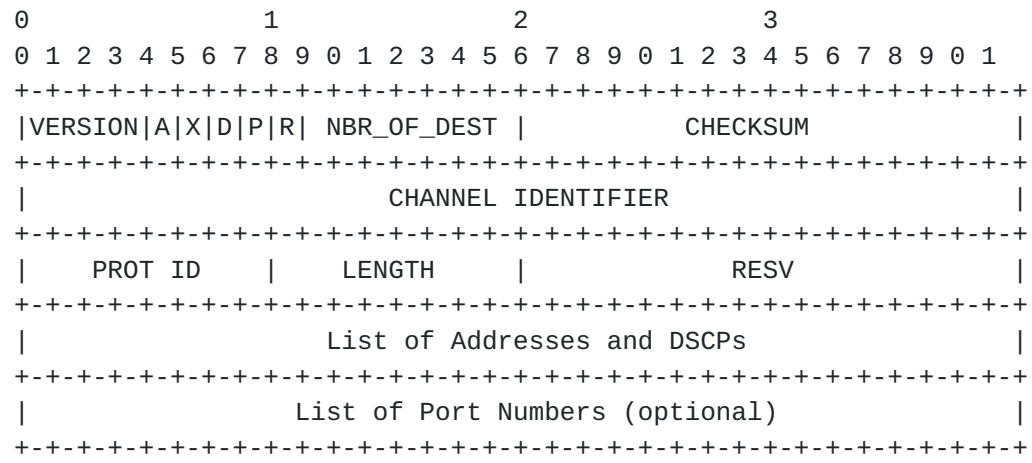

Figure 4

VERSION = Xcast version number. This document describes version 1.

A = Anonymity bit: if this bit is set the destination addresses for which the corresponding bit in the bitmap is zero must be overwritten by zero.

X = Xcast bit: if this bit is set a router must not reduce the Xcast

packet to unicast packet(s), i.e. the packet must stay an Xcast packet end-to-end. This bit can be useful when IPsec is applied. If this bit is cleared a router should apply X2U if there is only one destination left in the Xcast packet. In some cases a router could decide not to apply X2U to a packet with the Xcast bit cleared, e.g. the router has no directly connected hosts and wants to avoid the extra processing required by X2U.

D = DSCP bit: if this bit is set the packet will contain a DS-byte for each destination.

P = Port bit: if this bit is set the packet will contain a port number for each destination.

NBR_OF_DEST = the number of original destinations.

CHECKSUM = A checksum on the Xcast header only. This is verified and recomputed at each point that the Xcast header is processed. The checksum field is the 16 bit one's complement of the one's complement sum of all the bytes in the header. For purposes of computing the checksum, the value of the checksum field is zero. It is not clear yet whether a checksum is needed (ffs). If only one destination is wrong it can still be useful to forward the packet to N-1 correct destinations and 1 incorrect destination.

CHANNEL IDENTIFIER = 4 octets Channel Identifier (see section 8.3). Since it is located within the first 8 bytes of the header, it will be returned in ICMP messages.

PROT ID = specifies the protocol of the following header.

LENGTH = length of the Xcast header in 4-octet words. This field puts an upper boundary to the number of destinations. This value is also determined by the NBR_OF_DEST field and the D and P bits.

RESV = R = Reserved. It must be zero on transmission and must be ignored on receipt.

The first variable part is the 'List of Addresses and DSCPs', the second variable part is the 'List of Port Numbers'. Both are 4-octet aligned. The second variable part is only present if the P-bit is set.

Figure 5 gives an example of the variable part for the case that the P-bit is set and the D-bit is cleared (in this example N is odd):

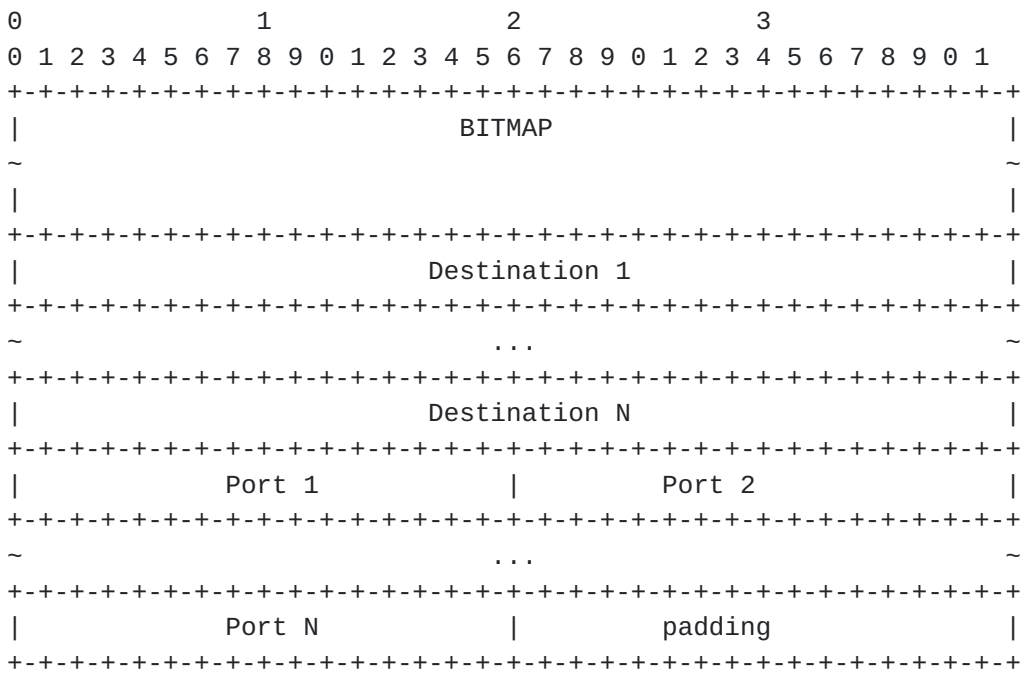

Figure 5

BITMAP = every destination has a corresponding bit in the bitmap to indicate whether the destination is still valid on this branch of the tree. The first bit corresponds to the first destination in the list. This field is 4-octet aligned (e.g. for 49 destinations there will be a 64-bit bitmap). If Xcast is applied in combination with IPsec, the bitmap - since it can change on route - has to be moved to a new to be defined IPv4 option.

List of Destinations. Each address size is four octets.

List of Port Numbers. List of two octet destination port number(s), where each port corresponds in placement to the preceding Destination Address.

9.3. IPv6

The Xcast6 header encoding is similar to IPv4, except that Xcast information would be stored in IPv6 extension headers.

[IPv6 header | Xcast6 | transport header | payload]

9.3.1. IPv6 header

The IPv6 header will carry the NextHeader value 'Routing Extension'. The source address field contains the address of the Xcast sender. The destination address field carries the All_Xcast_Routers address.

9.3.2. Xcast6 header

The Xcast6 header is also composed of a fixed and two variable parts. The fixed and the first variable part is carried in a Routing Extension. The second variable part is carried in a Destination Extension.

9.3.2.1. Routing Extension header

The P-bit of Xcast4 is not present because it is implicit by the presence or absence of the Destination Extension (Figure 6).

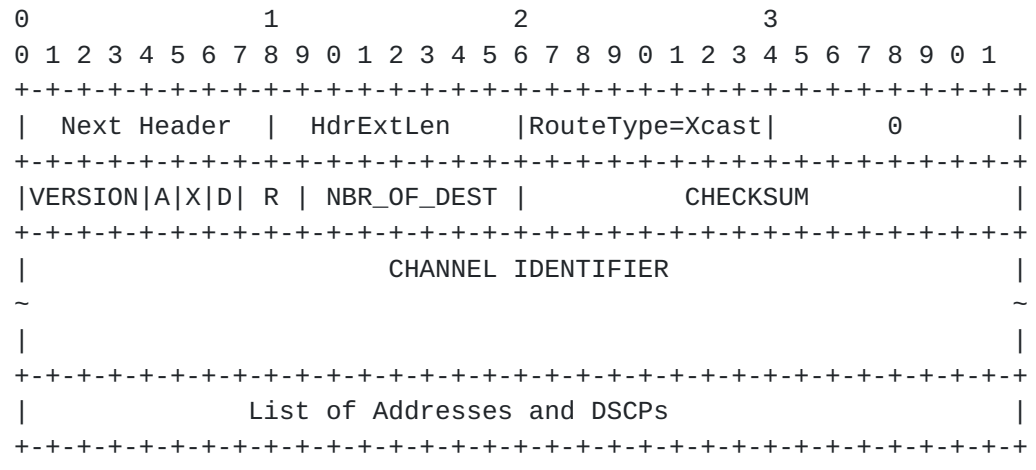

Figure 6

HdrExtLen = The header length is expressed in 8-octets, thus a maximum of 127 destinations can be listed (this is why NBR_OF_DEST is 7-bit).

RouteType = Xcast (see Section 15)

The fourth octet is set to 0.

R = Reserved.

CHANNEL IDENTIFIER = 16 octets Channel Identifier (see section 8.3).

The other fields are defined in section 9.2.2.

The 'List of Addresses and DSCPs' is 8-octet aligned. The size of the bitmap is determined by the number of destinations and is a multiple of 64 bits.

9.3.2.2. Destination Extension header

Optionally the Destination Extension (Figure 7) is present to specify

the list of Port Numbers. The destination header is only evaluated by the destination node.

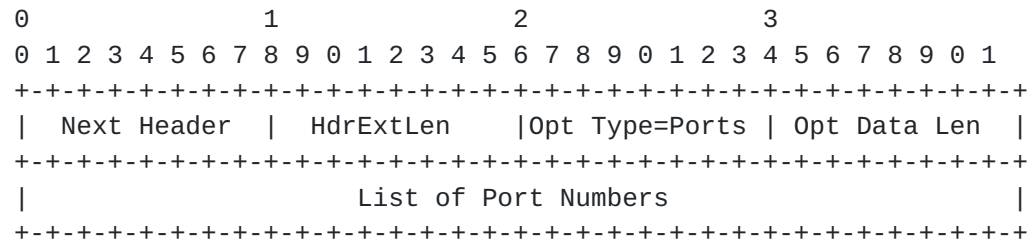

Figure 7

For the Option Type for Ports, see Section 15. The first three bits must be 010 to indicate that the packet must be discarded if the option is unknown and that the option can not be changed en-route.

The number of Ports must be equal to the number of destinations specified in the Routing header.

10. Impact on Upper Layer Protocols

Some fields in the Xcast header(s) can be modified as the packet travels along its delivery path. This has an impact on:

10.1. Checksum calculation in transport layer headers

In transport layer headers, the target of the checksum calculation includes the IP pseudo header, transport header and payload (IPv6 header extensions are not a target).

The transformation of an Xcast packet to a normal unicast packet - (premature) X2U - replaces the multicast address in the IP header destination field by the address of a final destination. If the Xcast header contains a Port List, the port number in the transport layer (which should be zero) also needs to be replaced by the port number corresponding to the destination. This requires a recalculation of these checksums. Note that this does not require a complete recalculation of the checksum, only a delta calculation, e.g. for IPv4:

```
Checksum' = ~ (~Checksum + ~daH + ~daL + daH' + daL' + ~dp + dp')
```

In which "'" indicates the new values, "da" the destination address, "dp" the destination port and "H" and "L" respectively the higher and lower 16 bit.

10.2. IPsec

This is described in [PARI].

11. Gradual Deployment

11.1. Tunneling

One way to deploy Xcast in a network that has routers that have no knowledge of Xcast is to setup "tunnels" between Xcast peers (MBone approach). This enables the creation of a virtual network layered on top of an existing network [2003]. The Xcast routers exchange and maintain Xcast routing information via any standard unicast routing protocol (e.g. RIP, OSPF, ISIS, BGP). The Xcast routing table that is created is simply a standard unicast routing table that contains the destinations that have Xcast connectivity, along with their corresponding Xcast next hops. In this way, packets may be forwarded hop-by-hop to other Xcast routers, or may be "tunneled" through non-Xcast routers in the network.

For example, suppose that A is trying to get packets distributed to B, C & D in Figure 8 below, where "X" routers are Xcast-capable, and "R" routers are not. Figure 9 shows the routing tables created via the Xcast tunnels:

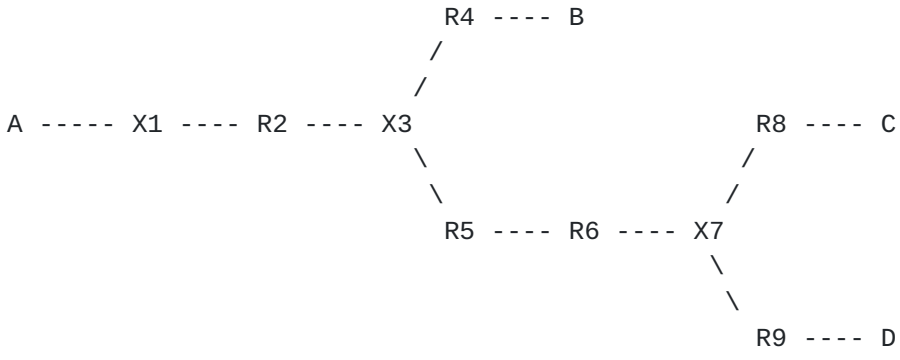

Figure 8

Router X1 establishes a tunnel to Xcast peer X3. Router X3 establishes a tunnel to Xcast peers X1 and X7. Router X7 establishes a tunnel to Xcast peer X3.

```
X1 routing table:     X3 routing table:     X7 routing table:
 Dest |  NextHop       Dest | NextHop        Dest | NextHop
------+----------     ------+---------      ------+---------
  B   |   X3             A  |   X1             A   |  X3
  C   |   X3             C  |   X7             B   |  X3
  D   |   X3             D  |   X7
```

Figure 9

The source A will send an Xcast packet to its default Xcast router, X1, that includes the list of destinations for the packet. The packet on the link between X1 and X3 is depicted in Figure 10:

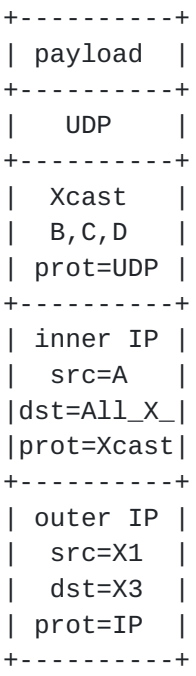

Figure 10

When X3 receives this packet, it processes it as follows:

- Perform a route table lookup in the Xcast routing table to determine the Xcast next hop for each of the destinations listed in the packet.

- If no Xcast next hop is found, replicate the packet and send a standard unicast to the destination.

- For those destinations for which an Xcast next hop is found, partition the destinations based on their next hops.

- Replicate the packet so that there's one copy of the packet for each of the Xcast next hops found in the previous steps.

- Modify the list of destinations in each of the copies so that the list in the copy for a given next hop includes just the destinations that ought to be routed through that next hop.

- Send the modified copies of the packet on to the next hops.

- Optimization: If there is only one destination for a particular Xcast next hop, send the packet as a standard unicast packet to the destination, since there is no advantage to forwarding it as an Xcast packet.

So, in the example above, X1 will send a single packet on to X3 with a destination list of < B C D >. This packet will be received by R2 as a unicast packet with destination X3, and R2 will forward it on, having no knowledge of Xcast. When X3 receives the packet, it will, by the algorithm above, send one copy of the packet to destination < B > as an ordinary unicast packet, and 1 copy of the packet to X7 with a destination list of < C D >. R4, R5, and R6 will behave as standard routers with no knowledge of Xcast. When X7 receives the packet, it will parse the packet and transmit ordinary unicast packets addressed to < C > and < D > respectively.

The updating of this route table while simple in an intra-domain environment would be more complex in an inter-domain environment. Thus the use of tunneling in an inter-domain environment requires further consideration.

11.2. Premature X2U

If a router discovers that its downstream neighbor is not Xcast capable, it can perform a Premature X2U, i.e. send a unicast packet for each destination in the Xcast header which has this neighbor as a next hop. Thus duplication is done before the Xcast packet reached its actual branching point.

A mechanism (protocol/protocol extension) to discover the Xcast capability of a neighbor is ffs. Among others, one could think of an extension to a routing protocol to advertise Xcast capabilities or one could send periodic 'Xcast pings' to its neighbors (send an Xcast packet that contains its own address as a destination and check whether the packet returns).

11.3. Semi-permeable tunneling (IPv6 only)

This is an optimization of tunneling in the sense that it does not require (manual) configuration of tunnels. It is enabled by adding a Hop-by-Hop Xcast6 header. An IPv6 packet can initiate/trigger additional processing in the on-route routers by using the IPv6 Hop-

by-hop option.

The type of the Xcast6 Hop-by-hop option has a prefix '00' so that routers that cannot recognize Xcast6 can treat the Xcast6 datagram as a normal IPv6 datagram and forward toward the destination in the IPv6 header.

Packets will be delivered to all members if at least all participating hosts are upgraded.

When the source A sends an Xcast packet via semi-permeable tunneling to destinations B, C and D it will create the packet of Figure 11. One of the final destinations will be put in the destination address field of the outer IP header.

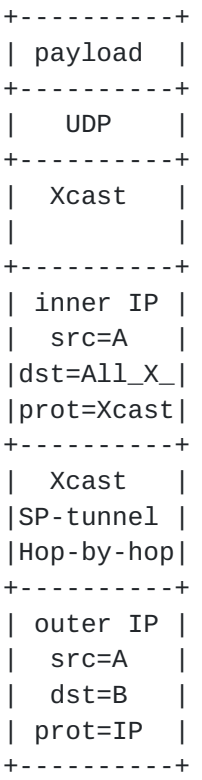

Figure 11

Semi-permeable tunneling is a special tunneling technology that permits intermediate Xcast routers on a tunnel to check the destinations and branch if destinations have a different next hop.

Note that with the introduction of an Xcast IPv4 option, this technique could also be applied in IPv4 networks.

11.4. Special case: deployment without network support

A special method of deploying Xcast is possible by upgrading only the hosts. By applying tunneling (see section 11.1 and 11.3) with one of the final destinations as tunnel endpoint, the Xcast packet will be delivered to all destinations when all the hosts are Xcast aware. Both normal and semi-permeable tunneling can be used.

If host B receives this packet, in the above example, it will notice the other destinations in the Xcast header. B will create a new Xcast packet and will send it to one of the remaining destinations.

In the case of Xcast6 and semi-permeable tunneling, Xcast routers can be introduced in the network without the need of configuring tunnels.

The disadvantages of this method are that:

- all hosts in the session need to be upgraded.

- non-optimal routing.

- anonymity issue: hosts can know the identity of other parties in the session (which is not a big issue in conferencing, but maybe for some other application?).

- host has to perform network functions and needs an upstream link which has the same bandwidth as its downstream link.

11.5 Using a Small Number of Xcast-Aware Routers to Provide Xcast in a Not-so-Small Network

In this approach, an xcast packet uses a special 32-bit unicast address in the destination field of the IP header. In the simplest version of this scheme, there might be only a single xcast-aware router in a network. This xcast-aware router looks like a "server" to the other routers and it is configured so that its IP address (or one of its IP addresses) corresponds to the "special" 32-bit address. Thus when xcast clients send xcast packets, the non-xcast-aware routers will route these packets to the xcast-aware router and the xcast-aware router can "explode" (X2U) them into an appropriate set of unicast packets. This allows clients anywhere in a network to use xcast to overcome the problem of limited bandwidth in the "first mile" with a minimum number of xcast-aware routers (i.e. 1).

Another possibility is to deploy a few of these xcast-aware routers at various points in the network and configure each of these with the special 32-bit address. This provides redundancy, eliminating the single point of failure, and reduces the distance an xcast packet needs to travel to reach an xcast-aware router, reducing network latencies. In this case, the xcast-aware routers appear to be a

single server that is "multihomed" (i.e. connected to the network at more than one place) and the non-xcast-aware routers will, via ordinary unicast routing, deliver packets that are addressed to this "multihomed virtual server" via the shortest available path.

(Note that this scheme of delivering packets to any host in a group is also known as an "anycast" and is described in more detail in RFC's 1546, 2526 and 3068. Note too that RFC 1546 says:

> The important observation is that multiple routes to an anycast address appear to a router as multiple routes to a unicast destination, and the router can use standard algorithms to choose the best route.)

12. (Socket) API

In the most simple use of Xcast, the final destinations of an Xcast packet receive an ordinary unicast UDP packet. This means that hosts can receive an Xcast packet with a standard, unmodified TCP/IP stack.

Hosts can also transmit Xcast packets with a standard TCP/IP stack with a small Xcast library that sends Xcast packets on a raw socket. This has been used to implement Xcast based applications on both Unix and Windows platforms without any kernel changes.

Another possibility is to modify the sockets interface slightly. For example, one might add an "xcast_sendto" function that works like "sendto" but that uses a list of destination addresses in place of the single address that "sendto" uses.

13. Unresolved issues

Additional work is needed in several areas.

13.1 The format of the "list of addresses"

Additional details need to be specified. For example, in the IPv4 case, the format of the DSCPs option needs to be specified.

13.2 The size of Channel Identifier

The size of the channel identifiers in IPv4 and IPv6 are different in this document. 32 bits might be sufficient for both IPv6 and IPv4.

13.3 Incremental Deployment

Several possible methods of incremental deployment are discussed in this document including tunneling, premature X2U etc.. Additional

work is needed to determine the best means of incremental deployment for an intra-domain as well as an inter-domain deployment of xcast. If tunneling is used, additional details need to be specified (e.g. tunneling format, use of tunnels in the inter-domain case.)

13.4 DSCP usage

DSCP usage needs some work. DSCP's may have to be rewritten as packets cross inter-domain boundaries.

13.5 Traversing a firewall or NAT products

The usage of a different carried protocol type for IPv4 may cause difficulty in traversing some firewall and NAT products.

13.6 The size of BITMAP

Given that this is designed for small groups, it might make sense to simply mandate a fixed size for the Bitmap.

14. Security Considerations

The list of destinations in Xcast is provided by an application layer that manages group membership as well as authorization if authorization is desired.

Since a source has the list of destinations and can make changes to the list, it has more control over where its packets go than in traditional multicast and can prevent anonymous eavesdroppers from joining a multicast session for example.

Some forms of denial-of-service attack can use Xcast to increase their "effect". A smurf attack for example sends an ICMP Echo Request in which the source address in the packet is set to the address of the target of the attack so that the target will receive the ICMP echo reply. With Xcast, the ICMP Echo Request could be sent to a list of destinations which could cause each member of the list to send an Echo Reply to the target.

Measures have been taken in traditional multicast to avoid this kind of attack. A router or host can be configured so that it will not reply to ICMP requests addressed to a multicast address. The Reverse Path Forwarding check in traditional multicast architectures also helps limit these attacks. In Xcast, it can be difficult for a host to recognize that an ICMP request has been addressed to multiple destinations since the packet may be an ordinary unicast packet by the time it reaches the host. On the other hand, a router can detect Xcast packets that are used to send ICMP requests to multiple

destinations and can be configured to drop those packets. Note too, that since Xcast sends packets to a short list of destinations, the problem of sending attack packets to multiple destination is less of a problem than in traditional multicast. Obviously, the use of IPsec to provide confidentiality and/or authentication can further diminish the risk of this type of attack.

The problem of secure group communications has been addressed by the Multicast Security (msec) working group which has defined an architecture for securing IP-multicast-based group communications [3740]. Many of the concepts discussed in the msec working group such as managing group membership, identifying and authenticating group members, protecting the confidentiality and integrity of multicast traffic and managing and securely distributing and refreshing keys also apply to Xcast-based group communications. And many of the same mechanisms seem to apply. One significant difference between multicast and Xcast is the fact that the Xcast header (or at least a bitmap in the Xcast header) needs to change as an Xcast packet travels from a source to a destination. This affects the use of IPsec and suggests that at least the Xcast header bitmap must be in a "mutable" field. A complete solution for securing Xcast-based group communications addressing all the issues listed above will be the subject of additional work which will be discussed in one or more additional documents. We expect that this effort will build on the work that has already been done in the msec working group.

15. IANA Considerations

Experimentation with the Xcast protocol requires the use of protocol numbers maintained by IANA. For example, to implement XCAST6, implementations must agree on four protocol numbers:

(1) Multicast Address for All_Xcast_Routers
(2) Routing Type of IPv6 Routing Header
(3) Option Type of IPv6 Destination Option Header
(4) Option Type of IPv6 Hop-by-Hop Options Header

A protocol implementer may temporarily experiment with Xcast by using the values set aside for experimental use in RFC[4727]. An implementer must verify that no other experiment uses the same values on the Xcast testbed at the same time.

A future revision of the Xcast specification published on the standards track is required before IANA can assign permanent registry entries for Xcast. Implementors should be aware that they will need to modify their implementations when such permanent allocations are made.

16. Informative References

[1112] S. Deering, "Host Extensions for IP Multicasting", RFC 1112, August 1989.

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

[1770] C. Graff, "IPv4 Option for Sender Directed Multi-Destination Delivery", RFC1770, March 1995.

[1812] F. Baker, "Requirements for IP Version 4 Routers", RFC1812, June 1995.

[2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[2201] A. Ballardie, Core Based Trees (CBT) Multicast Routing Architecture, RFC 2201, Sept. 1997.

[2236] W. Fenner, Internet Group Management Protocol, Version 2, RFC 2236, Nov. 1997.

[2401] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", RFC2401, November 1998.

[2460] S. Deering, R. Hinden. Internet Protocol, Version 6 (IPv6), RFC2460, December 1998.

[2543] M. Handley, H. Schulzrinne, E. Schooler, J, Rosenberg, "SIP: Session Initiation Protocol", RFC2543, March 1999.

[2902] S. Deering, S. Hares, C. Perkins, R. Perlman, "Overview of the 1998 IAB Routing Workshop", RFC2902, August 2000.

[AGUI] L. Aguilar, "Datagram Routing for Internet Multicasting", Sigcomm84, March 1984.

[CHER] David R. Cheriton , Stephen E. Deering, Host groups: a multicast extension for datagram internetworks, Proceedings of the ninth symposium on Data communications, p.172-179, September 1985, Whistler Moutain, British Columbia, Canada

[BOIV] R. Boivie, N. Feldman, "Small Group Multicast", draft-boivie-sgm-01.txt, July 2000.

[DEER] S. Deering, "Multicast Routing in a datagram internetwork", PhD thesis, December 1991.

[DEE2] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. The Pim Architecture for Wide-area Multicast Routing, ACM Transactions on Networks, April 1996

[DOOR] B. Van Doorselaer, D. Ooms, "SIP for the establishment of xcast-based multiparty conferences", draft-van-doorselaer-sip-xcast-00.txt, July 2000.

[FARI] D. Farinacci, "Multicast Source Discovery Protocol", draft-farinacci-msdp-00.txt, June 1998.

[H323] ITU-T Recommendation H.323 (2000), Packet-Based Multimedia Communications Systems.

[HOLB] H. Holbrook, B. Cain, "Source-Specific Multicast for IP", draft-holbrook-ssm-00.txt, March 2000.

[IMAI] Y. Imai, Multiple Destination option on IPv6 (MDO6), <draft-imai-mdo6-02.txt>, September 2000

[MBONE] Frequently Asked Questions (FAQ) on the Multicast Backbone (MBONE), ftp://venera.isi.edu/mbone/faq.txt

[OOMS] D. Ooms, W. Livens, Connectionless Multicast, <draft-ooms-cl-multicast-02.txt>, April 2000

[PARI] O. Paridaens, D. Ooms, Security Framework for Explicit Multicast, draft-paridaens-xcast-sec-framework-01.txt, November 2000.

[PERL] R. Perlman, "Simple Multicast: A design for Simple, Low-overhead Multicast", draft-perlman-simple-multicast-02.txt, February 1999.

[RMT] Reliable Multicast Transport Working Group web site, http://www.ietf.org/html.charters/rmt-charter.html, June 15, 1999

[SOLA] M. Sola, M. Ohta, T. Maeno. Scalability of Internet Multicast Protocols, INET'98, http://www.isoc.org/inet98/proceedings/6d/6d_3.htm

[BCP-2004]
H. Hsu.et al., Best Current Practices of XCAST (Explicit Multi-Unicast) by 2004, http://www.ietf.org/internet-drafts/draft-hsu-xcast-bcp-2004-01.txt

[2434] Narten, T., and Alvestrand, H., "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, October, 1998.

[2119] Bradner, S., "Key words for use in RFCs to Indicate requirement Levels", BCP 14, RFC 2119, March 1997.

[4727] B. Fenner, "Experimental Values in IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, November 2006.

[3740] T. Hardjono, B. Weis, "The Multicast Group Security Architecture", RFC 3740, March 2004.

17. Authors Addresses

Rick Boivie
IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532
Phone: 914-784-3251
Email: rhboivie@us.ibm.com

Nancy Feldman
IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532
Email: nkfeldman@yahoo.com

Yuji Imai
Fujitsu LABORATORIES Ltd.
1-1, Kamikodanaka 4-Chome, Nakahara-ku, Kawasaki 211-8588, Japan
Phone : +81-44-754-2628
Fax : +81-44-754-2793
E-mail: ug@xcast.jp

Wim Livens
ESCAUX
Krijtstraat 17, 2600 Berchem, Belgium.
E-mail: wim@livens.net

Dirk Ooms
OneSparrow
Belegstraat 13; 2018 Antwerp; Belgium
E-mail: dirk@onesparrow.com

Olivier Paridaens
Alcatel Network Strategy Group
Fr. Wellesplein 1, 2018 Antwerpen, Belgium.
Phone : 32 3 2409320
E-mail: Olivier.Paridaens@alcatel.be

Eiichi Muramoto (editor)
Matsushita Electric Industrial Co., Ltd.
4-12-4 Higashi-shinagawa, Shinagawa-ku, Tokyo 140-8587, Japan
Phone : +81-3-6710-2031
E-mail: muramoto@xcast.jp

18. Full Copyright Statement

19. IPR Notices

of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.