第X部

Explicit Multicast

第 10 部
Explicit Multicast

第 1 章　はじめに

明示的マルチキャスト (XCAST: Explicit Mulcitast) は、従来の ISM (Internet Standard Multicast) でのグループアドレスに変えて、パケットヘッダに到達すべき複数のユニキャストアドレスを明記することで宛先を指定するマルチキャスト方式である。XCAST は ISM に比べてグループメンバ数に制約がある一方で、グループ数に関するスケーラビリティに優れており、多地点ビデオ会議やネットワーク対戦型ゲームなど、多数のエンドユーザがプライベートなマルチキャストグループに対して発信が必要な用途に有効である。

WIDE project では 1999 年度に v6 WG の活動として XCAST の研究開発を開始し、2000 年 11 月より WG として独立した。本報告書では 2000 年度の XCAST WG の活動を以下の順に報告する。

- MDO6 (Multiple Destination Option) の実装と試用
- グループ管理機構の設計と実装
- Explicit Multicast Basic Specification

第 2 章　MDO6(Multiple Destination Option) の実装と試用

MDO6 (Multiple Destination Option) は、富士通研究所と WIDE project の共同研究で設計された小規模グループのための明示的マルチキャストプロトコルで、1999 年度に Internet-Draft として寄書を行った。本年度は提案方式の実証的な検証をおこなう目的で、KAME IPv6 実装に基づいた実装を行い、6Bone を用いた試用実験を行った。

実装と試用により XCAST の以下の利点を実証する事ができた。

- 既存 IPv6 protocol stack への改造量が非常に小さい。
- core network での変更や設定なしでエンドユーザが独自に使用を開始できる。
- core netowork のルータの対応透過的に進めることでを漸近的に配送コストを改善できる。

2.1 実装

MDO6 の最初の実装は FreeBSD 2.2.8 上の KAME IPv6 stack を改造して作成した。作成規模は kernel stack 1.6K line, library 2.7K line である。特に kernel の改造規模が ISM などに比べて小さくする事ができた。これは、XCAST が経路管理をユニキャストに完全に依存しているため、マルチキャスト経路情報の維持管理論理が不要で、かつ中継処理中の経路検索処理を KAME が既に持つユニキャスト経路処理を完全に流用することができたためである。さらに FreeBSD 自身のバージョンアップに追随して、FreeBSD 3.5, 4.2 に対する対応を行った。

このプロトコル実装を用いて、以下のグループマルチキャストを用いたアプリケーションを MDO6 に対応させた。

- vic: 多地点画像配信アプリケーション
- rat: 多地点音声配信アプリケーション
- bzflag: ネットワーク対応マルチプレーヤ戦車対戦ゲーム

2.2 試用

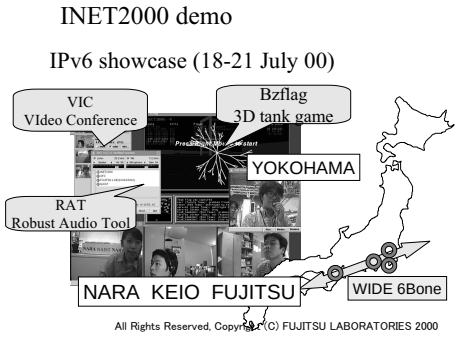
上記の実装を 6Bone 上に展開し実証を行った。

2.2.1 INET2000 デモンストレーション

2000 年 7 月に開催された国際会議 INET2000 会場にシステムを設置するとともに、慶應大学・奈良先端科学技術大学院大学・富士通研究所との間で、対

戦者がビデオチャットを行いながら多地点立体ゲームを行うデモンストレーションを行った。

ネットワークの接続は WIDE 6Bone をそのまま使用した。デモの時点では MDO6 ルータの安定性が充分認知された状態ではなかったので、会場への接続を含めた WIDE 6Bone のいずれのルータにも MDO6 中継機能の展開を行わなかった。しかし、MDO6 の IPv6 option header の特性を生かした漸近的展開機能により、エンドノードが daisy chain でマルチキャストデータグラムを中継できたため、途中にマルチキャストルータ、トンネルノードなどを一切配置・設定することなく、デモンストレーションを行う事ができた。



2.2.2 XCAST 定例ビデオミーティング

XCAST のより実用的な状況での試用を目的に、XCAST WG の会議をビデオミーティングとして 2000 年 10 月より週 1 回のペースで行っている。単なる通話テストとしての評価にしないために、WG メンバである慶應大学の学生に XCAST を用いた修士論文テーマを設定させ、その助言をメールとビデオミーティングを主体に行った。

当初心配された、帯域・遅延性能の不足による通話品質の劣化はミーティングの進行を妨げるほどではなかった。一方、メンバー管理機構が欠如しているため、アプリケーション起動時にユーザが明示的に宛先ノードを指定する必要がある。このためヒューマンエラーによる宛先もれなどが頻発し、ミーティングを正しく開始するまでに手間がかかっている。

第 3 章　グループ管理機構の設計と実装

ISM においてはネットワーク上に存在する受信者の情報は経路情報に埋め込まれる形で管理されており、送信アプリケーションは単にグループアドレスにデータグラムを送出するだけで良い。ところが XCAST はアプリケーションがメンバを明示的に指定する必要があるため、配送先の情報を管理し取得する処理が必要となる。XCAST WG は 2000 年度よりこの機構の検討を開始し、本年度は最初のプロトタイプの設計と実装を行った。

管理機構の備えるべき機能には以下があると考える。

- グループメンバの動的な変更
  グループへの新規参加や脱退にともなうグループメンバの動的な変更に対応する．
- グループメンバ情報の共有
  自分の参加しているグループの他のメンバの情報を知らなければ，グループコミュニケーションを図ることはできない．そのため，全てのメンバが他のメンバの情報を知っていなければならない．本機構では，グループメンバの情報を共有することにより，この問題を解決する．
- データ配送先の動的な変更の実現
  ユーザの要求やグループメンバの追加／削除に応じて，データの配送先を動的に変更する．

XCAST における「グループ」を、アプリケーションの使用者「ユーザ」が複数参加したものと捉え、これを管理する機構を、ユーザ状況管理サーバ，セッションマネージャ，アプリケーションの 3 つで構成されるモジュール群として設計した。それぞれの役割と振舞は以下のように設定した。

- ユーザ状況管理サーバ
  ユーザ状況管理サーバはユーザの情報を保持し，またメンバの状態を監視する．ユーザの状態の監視にはアウェアネスプロトコルを採用する．
- セッションマネージャ

セッションマネージャはユーザ状況管理サーバのクライアントであり，アプリケーションと同じホストで動作する．サーバと通信し，他のメンバの情報を取得する．また，他のセッションマネージャと通信し，グループメンバの情報を共有する．それらの情報を元にユーザはデータの送信先を決定し，送信先の IP アドレスのリストをアプリケーションに通知する．グループメンバの情報はリアルタイムに変化するので，セッションマネージャは，グループメンバの情報が変化する度にアプリケーションに変更された送信先の IP アドレスのリストを通知する．

- アプリケーション
  アプリケーションは実際にデータを送受信し，グループコミュニケーションを実現する．セッションマネージャから送信先の IP アドレスのリストを受け取り，MDO6 によってデータを送信する．
- ユーザ
  本システムの利用者である．ユーザはセッションマネージャを操作しグループの生成や変更を行い，またデータ送受信の可否を決定する．
- グループ
  データをやりとりするユーザの集合である．本機構では，グループの最小単位を 1 ユーザとする．またグループの範囲を「グループ内の任意のユーザからデータを受信しているユーザ」とする．グループに所属しているユーザをメンバと呼ぶ．

3.1 実装

上記設計のもと、以下の環境で実装を行った．

オペレーティングシステム

FreeBSD 3.5 Release

IPv6 スタック

KAME[1] 20000625 Stable

MDO6 スタック

mdo6-kit-20001017-platform [2]

データベース

PostgreSQL[3] 7.0.2 20000921 patched

開発言語

C 言語

コンパイラ

gcc 2.7.2.3

C ライブラリ

glib 1.2.8，gtk+ 1.2.8，libpq

管理機構を用いて vic-2.8ucl-1.1.3，rat-4.2.4 を改造し、同一ユーザ状況管理サーバを指定してアプリケーションを起動するだけで、自動的に正しく相互の通信が行われることを確認した。

今度、管理機構をライブラリとして整備し、他のアプリケーションからも利用できるようにするとともに、本実装によるメンバ管理機能を強化した VIC, rat を実際のビデオ会議に適用し有用性を検証する。さらに今回の実装では検討できなかった、セキュリティ、性能スケーラビリティ実現方策を検討し実装する。

第 4 章　XCAST basic specification

1999 年度に MDO6 を Internet Draft として寄書するのとほぼ同時に、Alcatel より CLM (Connectionless Multicast)、IBM より SGM (Small Group Multicast) が寄書された。3 者で各々のプロトコルの検証を進め機構を精査した上で統合し、2000 年 12 月に basic XCAST specification として Internet Draft 化した。本 Draft は以下のような特徴を持つ。

- マルチキャスト宛先をユニキャストのリストとして持つ。
- IPv4, IPv6 の両方のプロトコルをサポート
- 配送状況の bitmap による表現
- ポートリスト
- DSCP リスト
- 経路キャッシュ・ISM との整合性を考慮した Channel ID

[1] http://www.kame.net/
[2] http://www.alcatel.com/xcast/
[3] http://www.postgresql.org/

- core network での対応を待たずに End-to-End の使用が可能な各種の漸近的導入手法

この Internet Draft を元に、以下の団体が実装と相互接続実験を行うことを表明している。

- IBM(IPv4)
- Alcatel(IPv4)
- ETRI(IPv6)
- WIDE Project・富士通研究所 (IPv4)

以下に Internet Draft の内容を掲載する。

4.1 Introduction

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.

There seem to be two kinds of multicast that are 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 & video multicasting of a working group session from an IETF meeting to sites all around the world. 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."

Multicast can be used to minimize bandwidth consumption. Explicit Multicast (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 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 the number of simultaneous multicast sessions can be very large and the number of members in a group is small.

4.2 Overview Xcast

In this document 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 group 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 could turn 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:

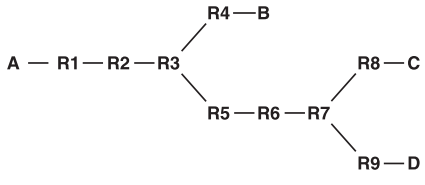


Figure 1

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, send the packet as a standard unicast packet to the destination (X2U), as there is no multicast gain by formatting it as an Xcast packet.

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 destination R5 with an Xcast list of ⟨C D⟩ and one ordinary unicast packet addressed to ⟨B⟩. 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.

4.3 The cost of the traditional multicast schemes

Traditional multicast schemes [37, 42, 36] 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 [37] and a Multicast Routing Protocol. Both components add to the difference in nature between unicast and multicast.

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. 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 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. Other schemes try to limit the amount of multicast routing information that needs to be disseminated, processed and stored throughout the network. These schemes 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.
- 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, 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 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 additionally need an interdomain multicast routing protocol.

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) [67] 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.

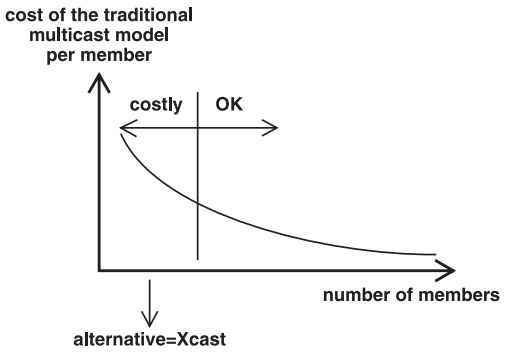


Figure 2

The traditional 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 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.

4.4 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 multicast, Xcast has the following advantages:

1) Routers do not have to maintain state per session (or per channel). 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) No symmetrical paths required. Traditional multicast routing protocols create non-shortest-path trees if the paths are not symmetrical (symmetrical = the shortest path from A to B is the same as the shortest path from B to A). It is expected that more and more paths in the Internet will be asymmetrical due to traffic engineering and more policy routing, thus multicast will cause more and more deviation from optimal network usage.

6) Automatic reaction to unicast reroutes. Xcast will react immediately to unicast route changes. In traditional 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 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 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) Simpler 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) Easier 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. Remark 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 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 [124].

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 [77] (tractable list) and [124](caching), but are not (yet) part of this basic Xcast specification.

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

4.5 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. So Xcast covers a very important class of applications: conferencing, multi-player games, collaborative working, 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 multicast model, but it does offer an alternative for multipoint-to-multipoint communications when there can be very large numbers of small sessions.

4.6 Flexibility Xcast

The main goal of multicast is to avoid duplicate information flowing over the same link. By using traditional multicast instead of unicast, bandwidth consumption decreases while the state and

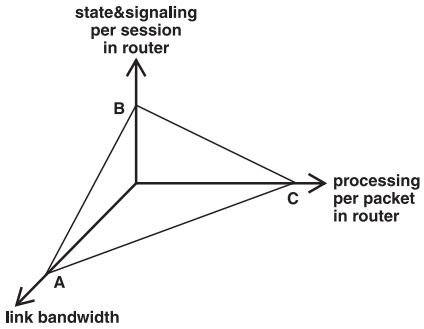


Figure 3

signaling per session increases. Apart from these two dimensions, we identify a third one: the header processing per packet. This three dimensional space is depicted in 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 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.

A nice property of Xcast is that a router can make its own tradeoffs. Since all information is carried in the packet, Xcast allows the router to move in this plane of conservation of misery (Figure 3), according to its own needs, which could be, for example, its location in the network. Routers could build caches to move from C to B, while Premature X2U allows a shift from C to A.

4.7 Control plane

Unlike traditional 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 button's 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 and H.323.

4.7.1 SIP

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.

4.7.2 Receiver Initiated Join model

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 receiver-initiated join model can be implemented, if desired, by introducing a server that hosts can talk to to join a conference.

4.8 Optional information

4.8.1 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.

4.8.2 List of DSCPs

The Xcast packet could (optionally) also contain a list of DiffServ CodePoints (DSCPs). While traditional 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.

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

2) A key to a caching table [124].

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

4) ...

4.9 Encoding

4.9.1 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.

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).

4.9.2 IPv4

[3] 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) is 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.

IPv4 header

The Xcast4 header is carried on top of an IP header. The IP header will carry the protocol number PROTO_Xcast. The source address field contains the address of the Xcast sender. The destination address field carries the All_Xcast_Routers address.

Xcast4 header

The Xcast4 header is 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.

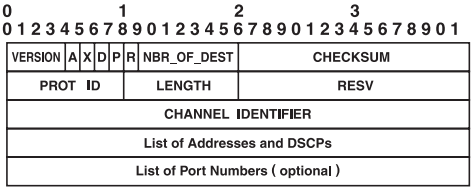


Figure 4

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.

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 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.

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.

CHANNEL IDENTIFIER = 4 octets Channel Identifier (see section 8.3).

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):

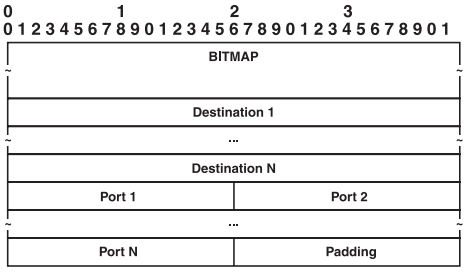


Figure 5

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.

IPv6

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

[IPv6 header — Xcast6 — transport header — payload]

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.

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.

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).

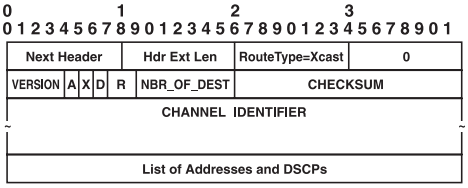


Figure 6

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 should be assigned by IANA.

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.

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.

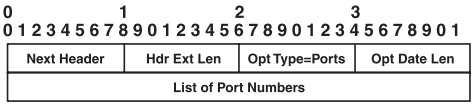


Figure 7

Option Type for Ports should be assigned by IANA. 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.

4.10 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:

4.10.1 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.

4.10.2 IPsec

This is described in [118].

4.11 Gradual Deployment

4.11.1 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. The Xcast routers exchange and maintain Xcast routing information via any standard unicast routing protocol (e.g. RIP, OSPF, ISIS). 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:

Router X1 establishes a tunnel to Xcast peer

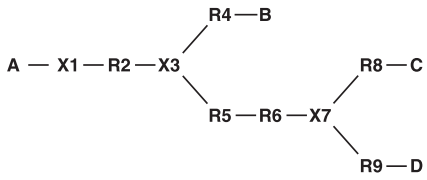


Figure 8

X1 routing table :

Dest	Next Hop
B	X3
C	X3
D	X3

X3 routing table :

Dest	Next Hop
A	X1
C	X7
D	X7

X7 routing table :

Dest	Next Hop
A	X3
B	X3

Figure 9

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

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:

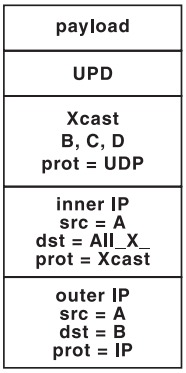


Figure 10

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, as there is no multicast gain by formatting 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.

4.11.2 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).

4.11.3 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.

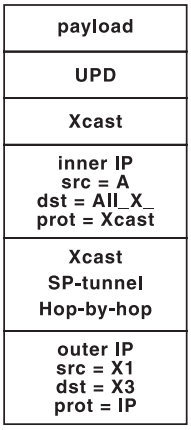


Figure 11

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.

4.11.4 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.

4.12 (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 in-

terface 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.

4.13 Security Considerations

See [118].