Xcast6 Treemap Islands - A Mixed Model of Application and Network Layer Multicast

Joanna Moulierac, Truong Khoa Phan, Nam Thoai, Cuong Tran

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Xcast6 Treemap Islands - A Mixed Model of Application and Network Layer Multicast

Joanna Moulierac — T. Khoa Phan — Nam Thoai — N. Cuong Tran

N° 7784

Octobre 2011

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--7784--FR+ENG

Xcast6 Treemap Islands - A Mixed Model of Application and Network Layer Multicast

Joanna Moulierac * , T. Khoa Phan * , Nam Thoai † , N. Cuong Tran †

Thème COM — Systèmes communicants
Projet Mascotte

Rapport de recherche n° 7784 — Octobre 2011 — 27 pages

Abstract: IP multicast is a protocol that deals with group communications with the aim of reducing traffic redundancy in the network. However, due to difficulty in deployment and poor scalability with a large number of multicast groups, IP multicast is still not widely deployed and used on the Internet. Recently, Xcast6 and Xcast6 Treemap, the two network layer multicast protocols, have been proposed with complementary scaling properties to IP multicast: they support a very large number of active multicast sessions. However, the key limitation of these protocols is that they only support small multicast group. In this paper, we propose Xcast6 Treemap island - a hybrid model of Application Layer Multicast (ALM) and Xcast6 that can work for large multicast group. Our model has several key advantages: ease of deployment, efficiency in bandwidth savings, no control message between end-host and router, zero multicast forwarding state at router and no need for a multicast address allocation protocol. In addition, this model is a potential service from which an ISP can get new revenue. Finally, in simulation section, we have made a comparison with IP multicast and NICE protocol to show the feasibility of our new model.

Key-words: IP multicast, Application Layer Multicast, Xcast, media streaming, linear program, algorithms.

This work has been partly funded by the European project FET AEOLUS, the ANR JCJC DIMA-GREEN.

* MASCOTTE, INRIA, I3S (CNRS - University Nice Sophia-Antipolis), France
† Faculty of CSE, Ho Chi Minh City University of Technology, Vietnam

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Xcast6 Treemap Islands - Un modèle hybride entre Multicast Applicatif et Xcast

Résumé : Le multicast a été inventé pour gérer les communications de groupes tout en réduisant la charge de trafic redondant sur le réseau. Actuellement, le multicast n'est pas largement déployé et utilisé sur l'Internet, principalement en raison des problèmes de passage à l'échelle avec un grand nombre de groupes. Récemment, Xcast6 et Xcast6 Treemap, ont été proposés pour pallier à ce problème: ces deux protocoles peuvent gérer un très grand nombre de groupes actifs. Toutefois, la principale restriction de ces protocoles est qu'ils ne fonctionnent qu'avec des groupes de très petite taille (avec peu de membres). Dans ce papier, nous proposons *Xcast6 Treemap Island* - un modèle hybride entre *Application Layer Multicast* (ALM) et Xcast6 qui peuvent gérer des groupes de grande taille. Les avantages principaux de noetr modèles sont les suivants : facilité de déploiement, utilisation efficace de la bande-passante, suppression des messages de contrôle et aucune nécessité d'un protocole d'allocation d'adresses multicast. Nous montrons la faisabilité et l'efficacité de notre proposition par des simulations où nous comparons notre proposition, un protocole IP multicast traditionnel, et le protocole NICE pour ALM.

Mots-clés : IP Multicast, Multicast Applicatif (ALM), Xcast, Streaming, programmation linéaire, algorithmes

1 Introduction

Multicast is a useful service for multi-party applications to help reduce traffic on the network. The very first idea of multicast was introduced by Aguilar in 1984 [1]. In this proposal, to send data to a group, the source needs to put all receivers' addresses in the header of each packet. Upon receiving a multicast packet, the router looks up the entire list of destinations, duplicates and forwards the packet to the appropriate network interfaces. It is clear for this proposal: the more destinations are included, the less space for payload is contained in the packet. As a result, the proposed solution seems ineffective because it only supports small multicast groups. In 1988, Deering invented host group multicast (or IP multicast) [2] which solved the problems encountered in Aguilar's proposal. Subsequently, Internet research community enhanced this idea with development solutions. However, more than two decades have passed, IP multicast still has not been widely deployed and used on the Internet. There are many problems of IP multicast due to a variety of administration as well as technical issues [3][4]. As stated in [5], when supporting many groups at the same time, IP multicast encounters a major obstacle in routing table's size at routers. Recently, the small group applications such as video conferencing are becoming more and more popular, leading to the requirement for a new protocol to support better. The point of these applications is that there is a great number of small multicast sessions working concurrently on the Internet. As examples of protocols managing small group applications, we can cite Xcast6 (Explicit Multi-unicast for IPv6 - RFC 5058) [5] and Xcast6 Treemap (X6T) [6]. However, like the Aguilar's proposal, both Xcast6 and X6T have a disadvantage that they limit multicast session to just a few users.

In this paper, we propose Xcast6 Treemap islands (X6Ti) - a simple approach to combine ALM and X6T to work for large multicast groups. X6Ti offers many advantages over the existing multicast protocols:

- scaling well with a large number of concurrently active multicast groups (even with large multicast applications).
- easy to deploy on the Internet.
- guaranteeing quality of service for multicast application.
- no need for a multicast address allocation scheme.
- a potential service from which an ISP can receive new revenue.

The main contributions of our work are the following:

- We extend the forwarding algorithm in [6] to guarantee X6T to work on the network with both Xcast-aware and normal routers.
- We propose a general model called X6Ti to support large multicast groups.
- We present in detail a two-tier infrastructure for large-scale media streaming using the X6Ti model.

- Finally, we discuss pros and cons of X6Ti and by simulation, we show the efficiency of X6Ti in comparison with existing multicast protocols.

The rest of this paper is structured as follows. We present related works in section 2. The idea of the X6Ti model is introduced in section 3. Section 4 describes a two-tier infrastructure for large-scale media streaming using the X6Ti model. Section 5 presents simulation results. We discuss the pros and cons of X6Ti in Section 6 and finally, we conclude the paper in Section 7.

2 Related works

2.1 IP multicast

In IP multicast, the source host sends data packets to a group of destinations represented by an IP multicast group address (a specially reserved multicast address blocks in IPv4 and IPv6 like "224.1.0.10" in Figure 1). The receivers use this group address to inform the network that they are interested in receiving packets sent to that group. The routers in the network will replicate the packet to reach multiple receivers such that packets are sent over each link of the network only once.

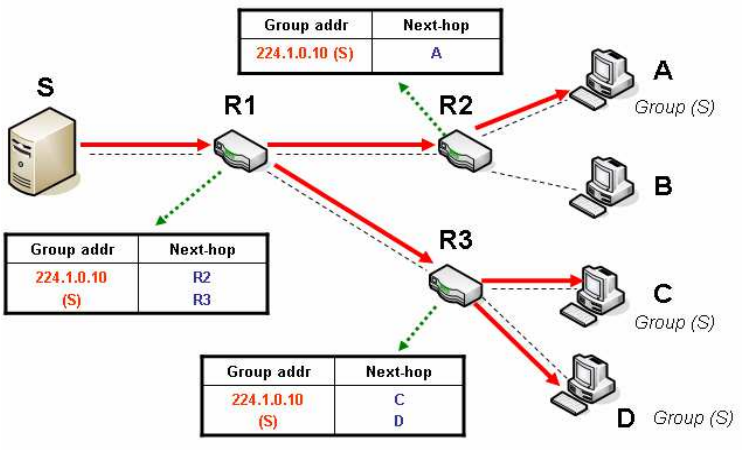

Figure 1: The source S sends data to a group using IP multicast

The challenges of IP multicast have been presented in [3][4]. In summary, IP multicast requires all routers on the network to be upgraded. On the other hand, IP multicast cannot support a very large number of acitve multicast group because of the state scalability problem at routers [7].

2.2 Application Layer Multicast (ALM)

Due to the sparse deployment of IP multicast, many researchers have suggested to move multicast service to application level [8][9]. In ALM, instead of changing network infrastructure, forwarding function is implemented at end-hosts. Logically, an overlay network is formed and packets are replicated at the end-hosts. Due to its ease of deployment, ALM has gained increasing popularity in the multicast community. However, traffic redundancy still exists and ALM is not really useful on the network where bandwidth is limited. Furthermore, long end-to-end delay is also an issue of ALM.

2.3 Xcast6 and Xcast6 Treemap (X6T)

The IPv6 protocol has the extension headers allowing to encode additional information in the packet. Unlike existing network multicast protocols [2][10], Xcast6 and X6T utilize the well designed structure of the IPv6 and follow the standardization, thus it can work on the network wherever IPv6 is supported.

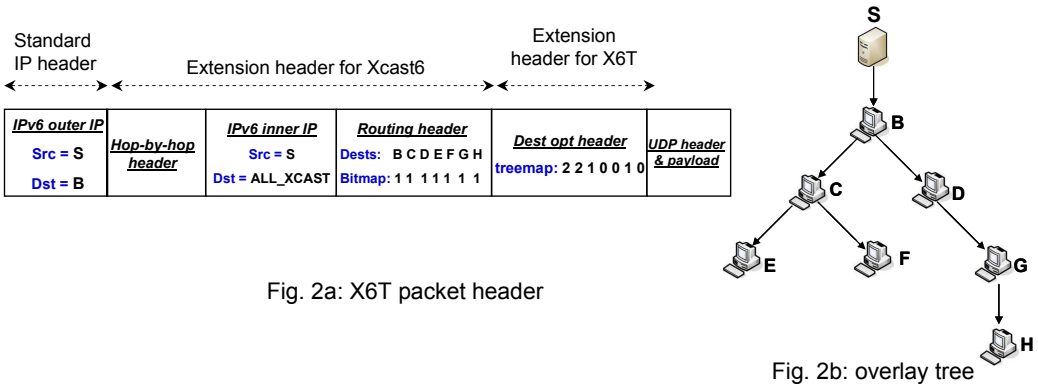

Figure 2: X6Ti packet header and the encoded overlay tree

The detail structure of Xcast6 and X6T packet header have been described in [5][6]. In summary, Xcast6 has a list of IPv6 destination addresses and a list of bitmaps (Figure 2a). "Bitmap = 1" is used to mark end-hosts which have not received the packet yet. For every received Xcast6 packet, Xcast-aware router lookups all the destinations in its unicast routing table, duplicates and forwards the packet to appropriate network interfaces. For non-Xcast-aware or normal routers, the packet is forwarded like a normal unicast packet with destination address in the "outer IP". In addition, when an Xcast6 end-host receives a packet, based on the bitmap, it will notice the other destinations that have not received this packet yet. The Xcast6 end-host then sends the packet to one of the remaining destinations.

Obviously, on the network with spare or no Xcast-aware router, the Xcast6 packet is forwarded as a daisy-chain of Xcast6 end-hosts which can not guarantee QoS for applications. To solve this problem, the authors in [6] propose X6T which additionally encodes an overlay routing tree (Figure 2a). The idea is that when X6T packets are not multicasted at network layer (i.e. there is no Xcast-aware router), the X6T end-hosts will based on the routing information in the packet header and forward the X6T packets in ALM manner. The overlay tree (Figure 2b) is simply encoded by an ordered list of destinations and a series of numbers called "treemap" as shown in Figure 2a. The ordered list of destinations is constructed by a breadth-first tree traversal (i.e. B C D E F G H). And the number in "treemap" corresponds to the number of children each end-host has.

The idea of how X6T works is that each X6T packet is handled by an Xcast-aware router (if any) on the routing between the source and destination hosts. Otherwise, the Xcast6 packet will be processed by an Xcast6 end-host. Note that both Xcast-aware routers and X6T end-hosts are able to duplicate and forward X6T packets. The end-host users, on the other hand, do not know whether there are Xcast routers on the network or not. Hence, it can happen that the packets are forwarded twice by the end-host and by the Xcast router. On the other hand, the overlay tree routing can also be broken in the presence of Xcast-aware routers (as shown in this example - E has to send data to F). Therefore, we extend the forwarding algorithm introduced in [6] for X6T end-hosts to avoid these kind of problems.

```
1.  // list of dests and the corresponding bitmap from the received packet
2.  list_of_dests D[0…n]; bitmap[0…n];
3.  outer_IP_src ← current host; outer_IP_dest ← null;
4.  // subtree rooted at the current host has k branches
5.  // k is the number in the treemap corresponding to the current host
6.  for i ← 1 to k do
7.      find all hosts belong to the branch "i" of the current host;
8.      new_bitmap ← bitmap;
9.      //keep bitmap for only group of hosts to be sent
10.     forall j that host D[j] does not belong to branch "i"
11.         new_bitmap[j] ← 0
12.     find "d" (nearest to the current host) that new_bitmap[d] == 1;
13.     outer_IP_dest ← D[d];
14.     forward packet with new_bitmap to host D[d];
15.     //update bitmap after sending data to branch "i"
16.     forall j that host D[j] belongs to branch "i"
17.         bitmap[j] ← 0
18. //guarantee data to reach all the receivers
19. find "d" (nearest to the root) that bitmap[d] == 1;
20. outer_IP_dest ← D[d];
21. bitmap[d] ← 0;
22. forward the packet to D[d]
```

Figure 3: Forwarding algorithm at end-host for X6T

The example in Figure 4 shows how X6T works on a network with the presence of both normal and Xcast-aware routers. Suppose that the source host would like to send data to a group of recipients as the overlay tree in Figure 2b. In this example, we use X1, X4 and all the end-hosts are Xcast-aware; R2, R3 and R5 are normal routers.

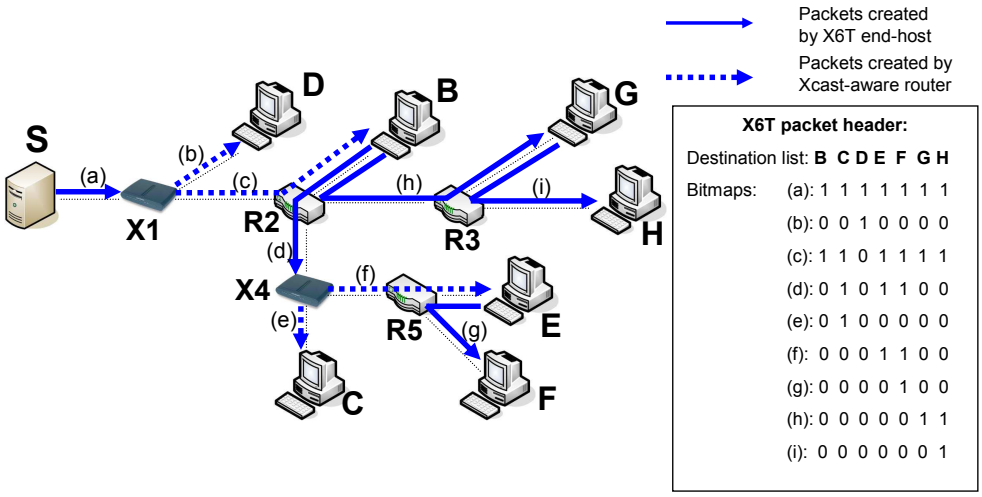

Figure 4: Source S sends data to the group of (B, C, D, E, F, G, H) using X6T

- The source S sends packets with the "outer IP" [src = S, dest = B], the list of destinations and the bitmap are shown in Figure 4.
- X1 receives an X6T packet, looks up all the destinations in its unicast routing table. Then, X1 duplicates, changes the bitmap and forwards one packet to D and the other packet to a group of (B, C, E, F, G, H) (the destinations in the outer IP of the two packets are D and B, respectively).
- The router R2 then handles the incoming packet as normal unicast packet and forwards it to B - the destination in the outer IP of the packet.
- According to the overlay tree, B has two branches, then it forwards one packet to a group of (C, E, F) (C is the destination in the outer IP). Similarly, B sends another packet to a group of (D, G, H). Because the bitmap of D was set to zero by router X1 in the previous step, B will forward this packet with the outer IP destination is G.
- Similarly, router R3, X4, R5 and end-host G handle the incoming packet as shown in Figure 4.

- Host E has no child in the overlay tree, however, thanks to the forwarding algorithm (Figure 3 - line 19-22) , E still forwards the packet to F to guarantee data to reach all the receivers.

From this example, we can see some key advantages of X6T:

- In X6T, end-host users do not care about the presence of Xcast-aware routers. It is noteworthy that how to build a good overlay tree is the work of an ALM algorithm and it is independent from how X6T works at the network layer. From the above example, we can see that with some Xcast-aware routers, X6T can reduce traffic redundancy and improve end-to-end delay even when we use a not-so-good overlay tree. Eventually, when all the routers are upgraded to Xcast-aware, the overlay tree is not used anymore because data is transfered totally in network layer multicast in which routers copy and forward the packets.
- Unlike IP multicast, Xcast-aware routers do not need to store any additional multicast forwarding states because all the information is in the packet header. As a result, X6T scales very well with a large number of concurrently active multicast groups.

However, like Xcast6, X6T is designed for small groups because each packet may include a small number of destination addresses. As the overlay tree in Figure 5, the source S has

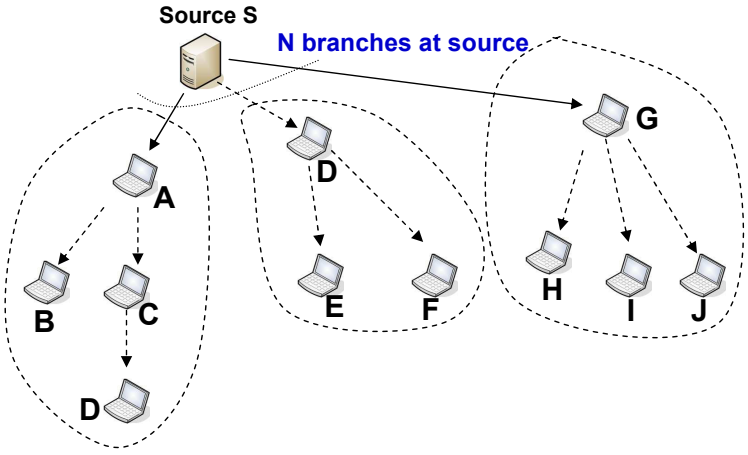

Figure 5: Number of hosts in a X6T session

$N = 3$ branches, hence S will send three packets each with the list of destinations and the corresponding treemap: [A B C D; 2 0 1 0], [D E F; 2 0 0] and [G H I J; 3 0 0 0]. For the current implementation, X6T can include at most 64 destination IP addresses in a packet header, thus X6T can support up to $(64 * N)$ hosts in a multicast session. However, for applications like video streaming or video conferencing, the number of destinations added

to the packet header is less than 64 (normally around 10) so that there is more space for the payload in the packet. Moreover, the value of N is selected based on the upload capacity of the source and it is usually small. For example: if the source has 1.5 Mbps upload capacity and it needs to disseminate a video stream at 0.5 Mbps, then N is set to 3 branches. Therefore, in general, X6T can support multicast session with small size. This is the motivation of our research to propose Xcast6 Treemap islands (X6Ti) to support large multicast groups.

3 Xcast6 Treemap islands

We divide the large multicast group into Xcast6 islands and connect them using the ALM approach. We call this model is Xcast6 Treemap islands (X6Ti) and it works fully on application layer. However, like X6T, it will automatically switch to work in the network layer multicast wherever Xcast-aware routers are deployed. So, from now on, without loss of generality and for simplicity of presentation, we will describe the model without mentioning the presence of the router. We do not change the format of the X6T packet, the only change for using X6Ti is that a new kind of node - X6Ti sub-root is used. In short, we define three types of node called X6Ti root, X6Ti client and X6Ti sub-root as shown in Figure 6.

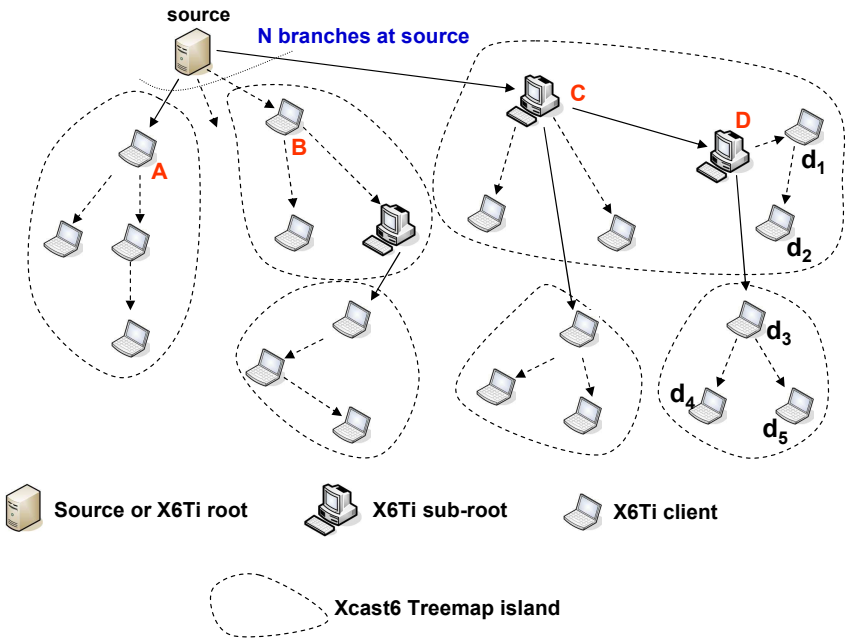

Figure 6: Xcast6 Treemap islands for large multicast group

- "X6Ti root" is the source sending data to a multicast group.
- "X6Ti client": after receiving an X6T packet, X6Ti client executes the forwarding algorithm (Figure 3) to forward the packet to next hosts (if any) at kernel level.
- "X6Ti sub-root" is extended from the X6Ti client. Like X6Ti client, X6Ti sub-root executes forwarding task based on the routing information in the packet header. In addition, X6Ti sub-roots are in charge of forwarding data to their own islands, thus they need to create and send a new X6T packet with the same payload, its own list of clients and the corresponding overlay tree. For example in Figure 6, when D receives a packet from C, based on the information in the packet header, D will forward the packet to the group of (d_1, d_2). In addition, because D is a sub-root, it then creates and forwards the packet with a new list of destinations (d_3, d_4, d_5) and the corresponding treemap (2, 0, 0).

As analyzed in the previous section, the maximum number of users that X6T can support is $(N * 64)$ where N is the number of branches at the source host. For X6Ti, we connect the islands together using X6Ti sub-root, therefore the size of the multicast group can be increased much more. For example: in Figure 6, with three X6Ti sub-roots, we can add three more islands and the number of members can be increased to $(N*64+3*64)$ (assuming that each X6Ti sub-root serves only one island in this example). The problem of using this X6Ti model is how to construct a suitable overlay tree, or in other words, we need to build the overlay tree to connect clients in each island and to connect many islands to form a large multicast tree as shown in Figure 6. Thanks to the results of ALM studies (summarized in part by [11]), we can adapt an existing ALM protocol or invent a new one that fits with the X6Ti model.

X6Ti model can be developed like a normal ALM protocol where some X6Ti clients are set to be X6Ti sub-root when needed (as shown in Figure 6). In this paper, for efficient media streaming in large multicast group, we consider a two-tier infrastructure (or proxy overlay multicast) like the model in [12] [13] (Figure 7). We build a backbone overlay tree to connect the root and X6Ti sub-roots together. The X6Ti sub-roots are responsible for forwarding data to their X6Ti clients. In this model, the X6Ti sub-roots are stable hosts with high bandwidth capacity to form a backbone spanning tree rooted at the source. There could be more X6Ti sub-roots in each island to serve many X6Ti clients. However, X6Ti sub-root can choose an X6Ti client which has available bandwidth to work like an X6Ti sub-root if needed. For example, as shown in Figure 7, "e_1" forwards data to its children ("e_2", "e_3") in the island rooted at "E". In addition, "e_1" is assigned to work as a sub-root to forward data to another island ("e_4", "e_5", "e_6"). In the next subsection, we identify the important objectives to build a suitable overlay tree and propose a modeling of the problem.

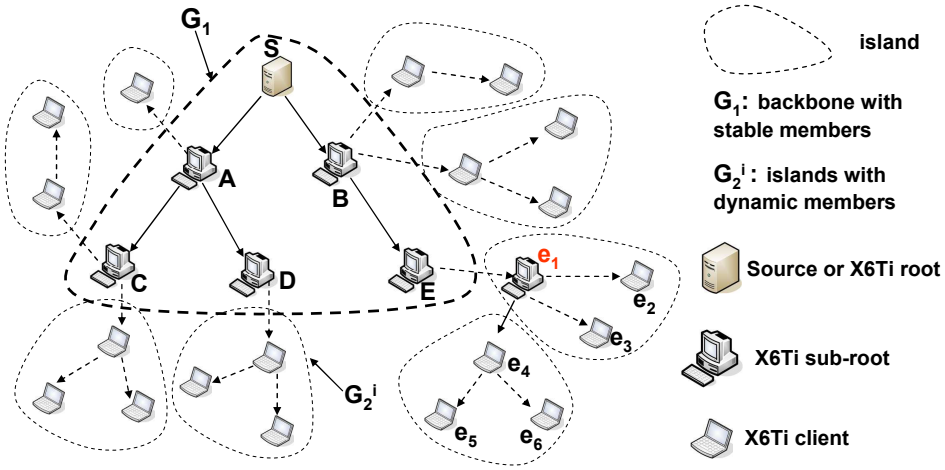

Figure 7: Two-tier infrastructure for large-scale media streaming

4 Problem formulation

We model our network as the two-tier complete graph:

$$G_1 = (V_1, E_1) \text{ and } G_2^i = (V_2^i, E_2^i)$$

where V_1 is a set of X6Ti sub-roots and the source; $E_1 = V_1 \times V_1$ is a set of edges connecting them. The edges are labeled with the end-to-end delay between the two end-hosts. V_2^i is the set of X6Ti sub-roots and X6Ti clients in the same "island i" and $E_2^i = V_2^i \times V_2^i$ is a set of edges connecting them together. Our objectives are to build an efficient overlay tree to meet the resource and latency constraints: each node has enough available bandwidth to forward media stream and along with this, end-to-end delay constraint is also important to guarantee quality of service.

Consider a multicast session in which the source injects a stream at the rate of B(Kbps). For simplicity, we assume that the capacity of any incoming access link of every node is no less than B(Kbps). Let the outgoing access link capacity of a node i (X6Ti sub-root or client) be b_i, then this node can send data to at most $\boldsymbol{s_i = \lfloor b_i / B \rfloor}$ direct children. This is the out-degree bound of node i in the overlay tree. For our model, the backbone should consist of a small number of high bandwidth and stable computers (not join/leave frequently) while in islands, there is a large number of end-hosts with limited bandwidth and can dynamically join/leave. Thus, we consider the two objectives for the two tiers (the backbone and the island) to satisfy their requirements:

4.1 Objective for the Backbone G_1 - Minimize Weight Overlay Latency, Degree-bounded Spanning Tree

In our model, we need to build an overlay tree to connect X6Ti sub-roots as the backbone for delivering media stream. Note that X6Ti sub-roots are the high bandwidth and reliable nodes located far from each other on the Internet. Therefore, it is important to minimize the end-to-end latency while respecting the out-degree constraint at each node. Moreover, an X6Ti sub-root can send ping messages to measure round trip delay to the others. Because the number of X6Ti sub-root is small, it is possible to have $O(n^2)$ message exchanges to compute the delay for all pair of any X6Ti sub-roots. We define $\boldsymbol{m_i}$ be the number of current X6Ti clients that are served by the X6Ti sub-roots i in G_2^i. Then the objective can be translated to minimize the weighted sum of the end-to-end latencies of all X6Ti sub-roots: $\boldsymbol{min}(\sum_{\boldsymbol{i \in V_1}} \boldsymbol{m_i L_i})$ where $\boldsymbol{L_i}$ denotes the accumulated overlay delay from the root to the X6Ti sub-root i. The reason we would like to minimize the "weight" overlay latency is that it gives more priority for the X6Ti sub-root which has many clients to reduce its latency. Let's consider the example shown in Figure 8: assume that there are one root node

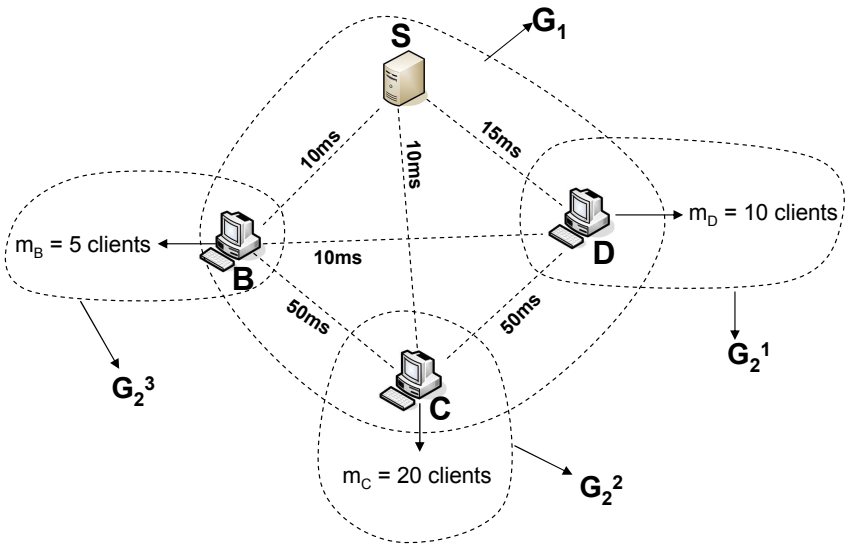

Figure 8: Example of the objective for the backbone G_1

(node S) and three X6Ti sub-roots (node B, C and D). Let the out-degree bound and the weight (number of clients) of each node are: $s_S = 2$, $s_B = 1$, $s_C = 1$, $s_D = 1$ and $m_S = 0$, $m_B = 5$, $m_C = 20$, $m_D = 10$. We also have the end-to-end delay of each pair of nodes: $d_{SB} = 10ms$, $d_{SC} = 10ms$, $d_{SD} = 15ms$, $d_{BC} = 50ms$, $d_{BD} = 10ms$ and $d_{CD} = 50ms$.

For simplicity, let's consider $d_{ij} = d_{ji}$. To achieve the objective, the backbone overlay tree should be like Fig. 9a since the minimum cost $\sum_{i \in V_1} m_i L_i = 5 * 10 + 20 * 10 + 10 * 20 = 450$. However, when new clients join the island connected to node D to reach $m_D = 20$, then the tree should change to Figure 9b.

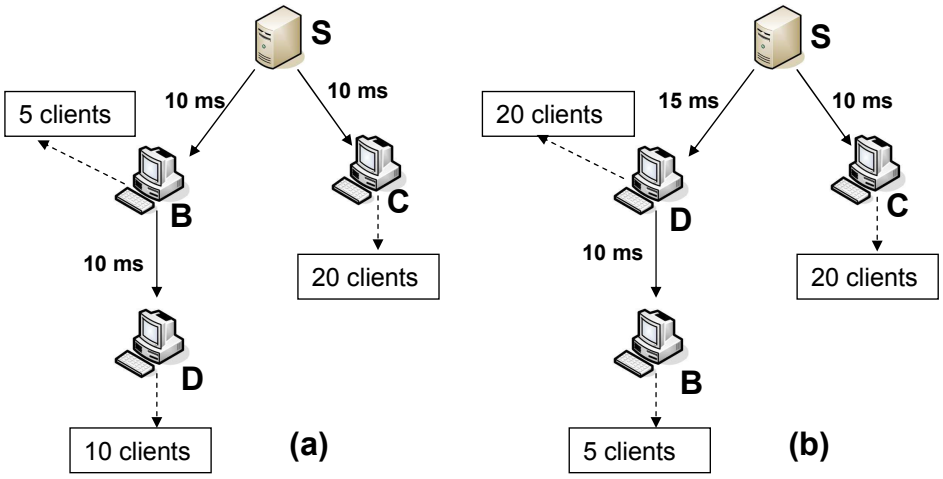

Figure 9: The backbone overlay tree

The objective for the backbone can be described as follows:
Problem with input: $G_1 = (V_1, E_1)$: a complete graph for the backbone of X6Ti root (node 0) and sub-roots.

$\forall i \in V_1, s_i$ is the out-degree constraint of node i

$\forall (i,j) \in E_1, d_{ij}$ is delay between the pair of node (i,j)

m_i is the weight (number of children) of node i.

Output: an overlay tree rooted at 0 covering V_1 and respecting the degree constraint for each node.

Objective: minimize the weight overlay latency $\sum_{i \in V_1} m_i L_i$ for the spanning tree.

This problem is NP-complete even in a complete graph. In fact, it is similar to the problem "Degree-constrained minimum overall latency spanning tree" [14]. In this paper, we formulate the problem using integer linear programming (ILP) and also propose a simple heuristic to find the solution:
Variable: Binary variable (x_{ij}) denotes whether the link (i,j) is in the objective spanning tree or not. Integer variable (L_i) is the overlay delay from the source (node 0) to node i.

$$\text{minimize} \qquad \sum_{i \in V_1} m_i L_i$$

subject to:

"Degree constraint":

$$\sum_{j \in V_1} x_{ij} \ \leq \ s_i \quad \forall\, i \in V_1\ , i \neq\ j, \quad (1)$$

"Spanning tree":

$$\sum_{i \in V_1} x_{ij} \ = 1 \quad j \in V_1,\ j \neq\ i \text{ and } j \neq 0, \quad (2)$$

"Delay":

$$L_t = \sum_{ij} f_{ij}^{0t} d_{ij} + \sum_{ij} f_{ji}^{0t} d_{ji}\ \forall (i,j) \in E_1, t \in V_1, t \neq 0 \ (3)$$

"Flow conservation constraint":

$$\sum_{t} f_{ij}^{0t} - \delta x_{ij} \leq 0 \quad \forall (i,j) \in E_1, t \in V_1, t \neq 0 \quad (4)$$

$$\sum_{j} f_{ij}^{0t} - \sum_{j} f_{ji}^{0t} = \begin{cases} 1 & \text{if i = 0} \\ -1 & \text{if i = t} \\ 0 & \text{otherwise} \end{cases} \quad \forall (i,j) \in E_1, i \in V_1, j \in neighbor(i), t \in V_1, t \neq 0$$

(5)

Constraint (1) ensures that each node does not exceed the out-degree constraint. Constraint (2) requires that all nodes must be connected as a spanning tree. Constraint (3) computes the accumulate delay from the root to node t. (4) and (5) are flow constraints, where δ is a big number to guarantee if link (i,j) is not in the spanning tree, then $\sum_t f_{ij}^{0t} = 0$.

However, the ILP's computation time for medium or large number of X6Ti sub-roots is a concern (see section 5, it takes around 250(s) to find the optimal solution for 60 sub-roots with the out-degree in (4-8)). To reduce the computation time, we can use a heuristic algorithm as shown in Figure 10.

4.2 Objective for Islands G_2^i - Minimum Depth Parent Selection

The number of clients in island is large and they can dynamically join/leave. Hence, it is not possible to have $O(n^2)$ message exchanges to measure delay between each pair of clients like the objective for G_1. However, like proxied overlay multicast [13], we can suppose that the clients (users) can find the closest X6Ti sub-roots and thus naturally form clusters around the sub-root without complicated measurement techniques or clustering algorithms. So, the objective for islands focuses on two main requirements: (i) the constructed overlay tree has enough resources to connect multiple clients and (ii) a stable overlay tree must be maintained in a dynamic group. Thanks to the results in [15][16], we can build an overlay tree with the strategy "minimum depth parent selection" to achieve good stability. The algorithm is simple: when a new host joins, then from among all parents that the new host is "eligible" to become a child, the X6Ti sub-root will select the one with minimum depth in the spanning

```
1.  node = root; sorted_list = null; overlay_tree = null;
2.  candidate_list.insert(root);
3.  for i = 1 to G_I.num_vertices
4.          for all "u" in G_I.vert that have max number clients
5.                  select "u" which has shortest path length from the root;
6.          insert(u) to the head of the sorted_list;
7.  while (not all nodes are in the spanning tree)
8.           for u in range of (0, out_degree(node))
9.                  pop(u) from the tail of the sorted_list;
10.                 overlay_tree.add_edge(<node, u>);
11.                 candidate_list.insert(u);
12.         candidate_list.pop() node with the shortest path length from the root;
13. return overlay_tree
```

Figure 10: Heuristic algorithm for the objective of the backbone

tree. The new host is "eligible" to become a child of an existing host if this host has available bandwidth or if one of its children has lower out-degree than the new host. In the latter case, the child host will be preempted or displaced by the new joining host. This algorithm has been evaluated with real-world trace to show better performance in comparison with other strategies [15][16].

4.3 Join and leave procedure

4.3.1 Join

- "X6Ti sub-root" needs to send join request with its available bandwidth and latency to the root node. While the most accurate method to collect bandwidth and delay requires significant time and resources, dynamically monitoring the available bandwidth can be done by a number of techniques [17] [16]. In this paper, we assume that there exists a mechanism to collect the out-degree constraint and end-to-end delay.
  To prevent long delay on joining process, the root node will temporarily add the new X6Ti sub-root to an existing node that has available bandwidth in the overlay tree. Because the sub-roots are more stable, we suggest that the root node will launch the integer linear programming (or the heuristic algorithm) every period of time (let's say 20 minutes, more or less depending on the real scenario). Another important thing is that we use X6T to deliver packets from the root to X6Ti sub-roots. In case that a new X6Ti sub-root joins and the root cannot add more destinations into the packet header (e.g. the number of sub-roots is exceeded 64), the root can assign an existing X6Ti sub-root to treat the new X6Ti sub-root as its X6Ti client. We can think this as a "n-tier" infrastructure.

- "X6Ti client": A new client should contact a rendezvous point to get a nearby X6Ti sub-root address. This client then sends join request to the closest X6Ti sub-root. If this X6Ti sub-root has not joined the backbone overlay tree, it should join now. The X6Ti sub-root then applies the minimum depth parent selection algorithm to add the new node to the spanning tree. Note that, it is similar to the case X6Ti sub-roots, an existing X6Ti client can be assigned to work like the X6Ti sub-root when the current X6Ti sub-root cannot add more destinations to the packet header.

4.3.2 Leave

- "X6Ti sub-root": X6Ti sub-roots are special hosts, therefore it is expected that their failures are rare and usually, the X6Ti sub-root will send a message before leaving the session. On the other hand, REFRESH message is periodically exchanged between X6Ti sub-roots, hence node failure can be detected by noticing missing REFRESH message. To repair the tree, the root re-executes the integer linear programming (or the heuristic algorithm) with the remaining nodes to find the new backbone overlay tree. For X6Ti clients in the leaving X6Ti sub-root's group, they need to contact the rendezvous to find other nearby X6Ti sub-root and re-join.

- "X6Ti client": it is similar to the case of X6Ti sub-root, failure is detected by REFRESH or leaving message. After discovering an X6Ti client leaving, the X6Ti sub-root find new minimum depth parents for the descendants of the departing hosts.

5 Performance evaluation

5.1 Evaluation of the Objective for the Backbone

We use a computer Intel(R) Core(TM)2 Duo CPU E6550 2.33GHz, 3GB memory to launch CPLEX [18] and find the optimal value for the objective of the backbone. For each test, we create a complete graph in which each node is randomly assigned a degree in between (1, 4) and (4, 8). In addition, delay between each pair is selected randomly in range (10, 40) ms; the weight of each node is in between 10 and 100 children. As shown in Figure 11, it takes longer computation time for smaller degree constraint and in general, less than 250(s) to find the optimal value of up to 60 X6Ti sub-roots in case of the degree constraint in range (4, 8).

The heuristic works well and is close to the optimal values given by the integer linear programming when the number of nodes is small. As shown in Figure 12, the heuristic algorithm performs better in comparison with a random algorithm where the tree is built in random. In the heuristic algorithm, it is important to select a "good" node from the "candidate_list" to become the parent of new nodes. As we can see, for smaller out-degree constraint (1-4), there is a high possibility for selecting a node with shortest path length from the root but has small out-degree. It makes the spanning tree taller than the optimal one, thus it is not close to the optimal value. As shown in Figure 12, with 50 nodes, the ratio

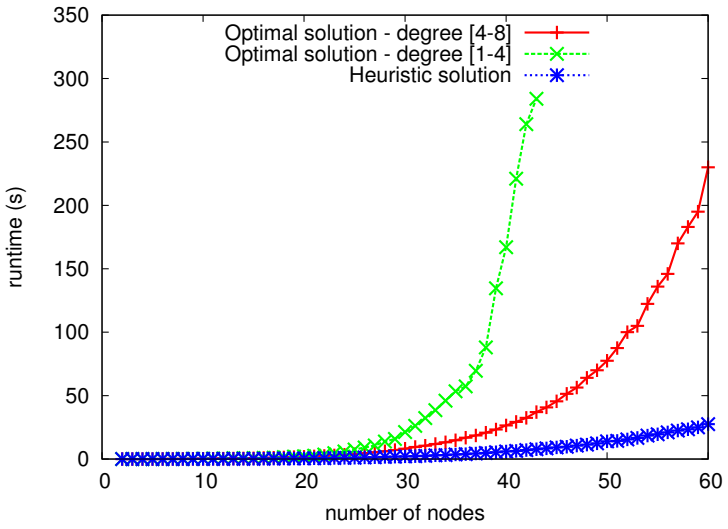

Figure 11: Run time of the optimal and heuristic algorithm

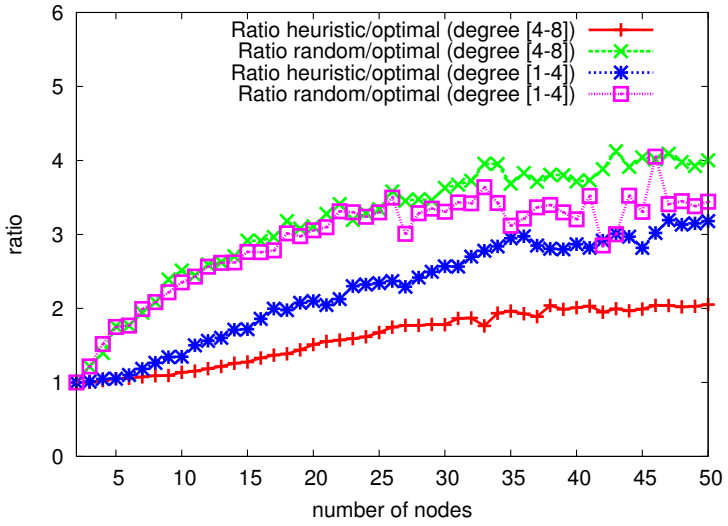

Figure 12: Ration between the heuristic and optimal objective value

between the heuristic and the optimal value is 2 when the out-degree is in (4-8), meanwhile this ratio value is 3 in case the out-degree is in (1-4).

5.2 End-to-End Delay and Traffic stress

In this section, we compare the performance of X6Ti, NICE - one of the famous ALM protocols [9] and IP multicast using NS-2 [19] and Oversim [20]. Because of the large number of members in multicast session (from 200 to 2600), it is not possible to do these simulations with Xcast6 or X6T. We use the network topology of France (extracted from SNDLib http://sndlib.zib.de) including 25 routers in the backbone. These routers work like transit domain routers and each of them connects to a set of stub domain routers. In the simulation, we use a single source and 24 X6Ti sub-roots, each of them connect to a backbone router. Each sub-root manages a number of islands which include end-hosts in the same stub domain (because we assume that clients will connect to the closest sub-root). These end-hosts are attached to stub routers uniformly at random. We increase the number of end-hosts in each island uniformly so that the total number of hosts in the multicast session is from 200 to 2600. For each test, we select at random 0% (X6Ti-0), 30% (X6Ti-30), 70% (X6Ti-70) and 100% (X6Ti-100) of the routers to be Xcast-aware routers. We collect the end-to-end delay and traffic stress when all hosts have joined and the session is in stable state.

5.2.1 End-to-end Delay

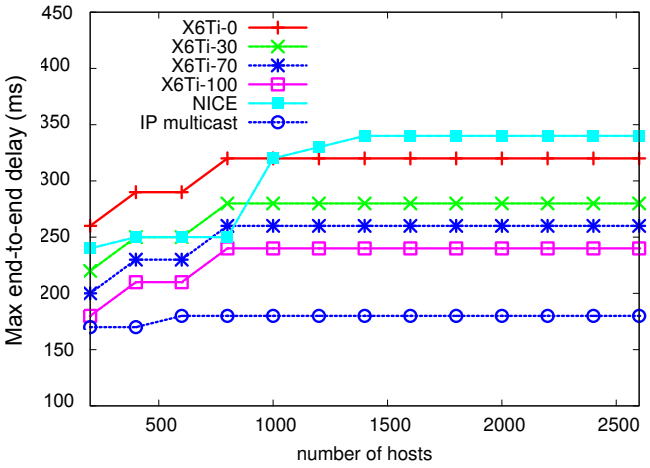

Figure 13: Max end-to-end delay of X6Ti vs. NICE vs. IP multicast

End-to-end delay is the amount of time (ms) that a packet is transmitted from the source to a receiver (X6Ti client). The results are shown in Figure 13 and Figure 14. As expected, IP multicast and X6Ti with Xcast-aware routers have shorter end-to-end delay. The maximum and average end-to-end delay of X6Ti-0, X6Ti-30, X6Ti-70 and X6Ti-100

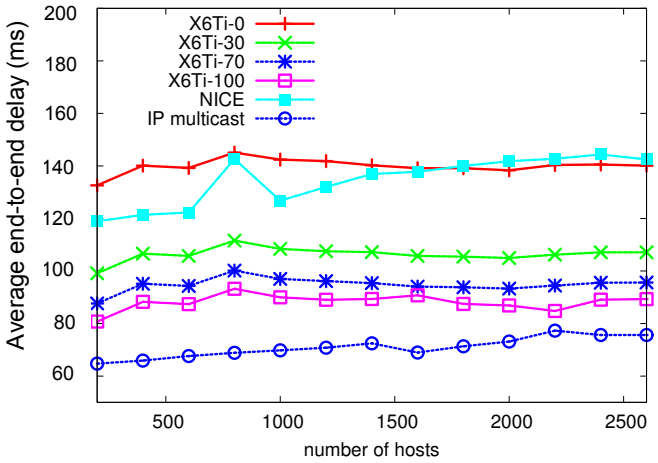

Figure 14: Average end-to-end delay of X6Ti vs. NICE vs. IP multicast

are decreasing and shorter than NICE. For instance, at a group size of 2600, the average end-to-end delay of NICE, X6Ti-0, X6Ti-30, X6Ti-70, X6Ti-100 and IP multicast are 142.5, 140.1, 107.1, 95.6, 89.3 and 75.7 (ms), respectively. Again, the performance improvement of X6Ti is gained with the support from Xcast routers.

5.2.2 Average traffic stress

"Traffic stress" of a physical link is the number of duplicate packets the link has to carry when a packet is multicast to the group. IP multicast has traffic stress of 1 on all links, while overlay multicast has the stress greater than 1. The results presented in Figure 15 show that X6Ti outperforms in comparison with NICE because we carefully connect the X6Ti subroots to the transit (backbone) routers. For instance, at a group size of 2600, the average traffic stress of NICE, X6Ti-0, X6Ti-30, X6Ti-70, X6Ti-100 and IP multicast are 2.4, 1.44, 1.37, 1.26, 1.17 and 1, respectively. Clearly, when X6Ti switches to work in network layer multicast with the support from Xcast-aware routers, the traffic stress is reducing.

5.3 Fast route adaptation

We simulate end-host to join/leave in an Xcast island at the time of 80th second and 120th second, respectively. The receiving rate of an existing host is shown in Figure 16, Figure 17 and Figure 18 in comparison of X6Ti and ALMCast [21]. ALMCast is the kernel implementation for packet forwarding in ALM. The overlay routing tree is stored in kernel of each host and it is updated when route changes (e.g. when nodes join or leave). For X6Ti, the sub-root only need to update the new treemap in the packet header when the overlay tree

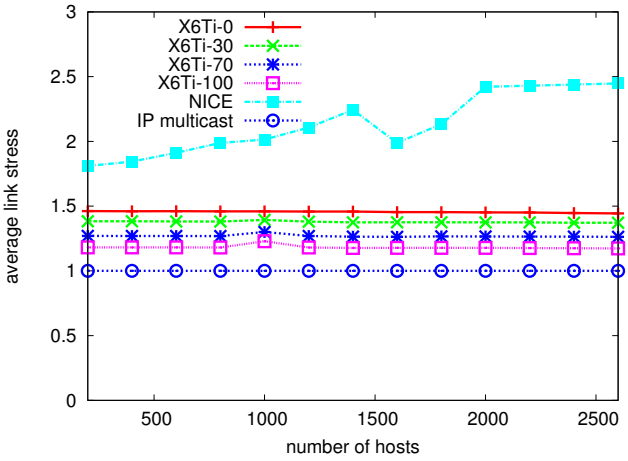

Figure 15: Average traffic stress of X6Ti vs. NICE vs. IP multicast

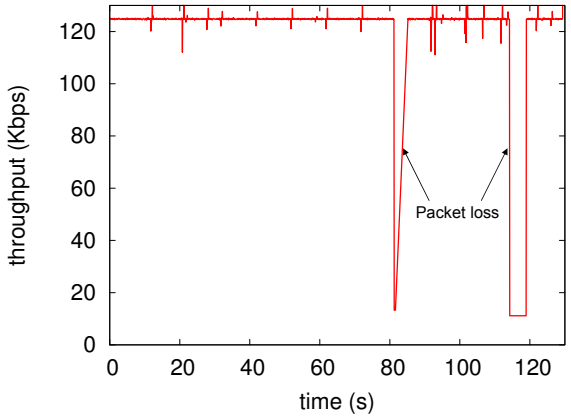

Figure 16: Throughput of ALMCast

changes while in ALMCast, the sub-root needs to disseminate the updated overlay tree to all receivers. Actually, ALMCast is tree-based protocol and there is no mechanism to reduce the cost of tree maintenance (e.g. using mesh-based to have backup links in case failure of hosts). This explains why ALMCast performs poorly in comparison with X6Ti, especially at 120(s) when an end-host leaves, the throughput of its child in ALMCast is much reduced. When all the routers are Xcast-aware, X6Ti switches to work in network layer multicast where routers perform forwarding function. This is why the receiving rate of end-host is not affected when end-hosts join/leave (Figure 18)

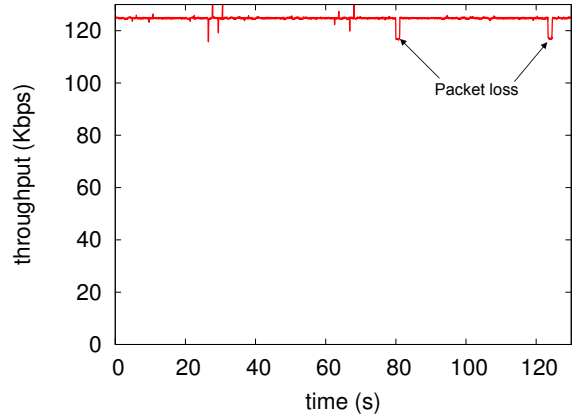

Figure 17: Throughput of X6Ti without Xcast router

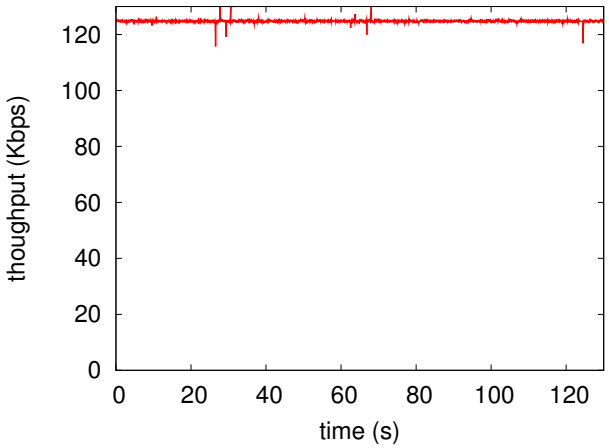

Figure 18: Throughput of X6Ti with Xcast routers

6 Discussion

The table in Figure 20 shows a summarized comparison of X6Ti and the existing multicast protocols. Each of the criteria is explained in this section.

	IP multicast	ALM	Xcast6	X6Ti
Efficiency in term of bandwidth/delay	High	Low - Medium	Medium	Medium - High
Ease of deployment	Low	High	High	High
State scalability	Low	High	High	High
Multicast group size	Large	Medium - Large	Small	Medium - Large
Fast route adaptation	High	Low	Medium - High	Medium - High

Figure 19: A comparison between X6Ti and other multicast protocols

6.1 Efficiency in Term of Bandwidth/Delay

ALM distributes traffic stress at the source to all the overlay end-hosts. For this reason, end-to-end delay increases, hence we rank ALM at a level of "low - medium" in term of bandwidth/delay efficiency. Xcast6 works well with the support of Xcast-aware routers, otherwise data are forwarded in a daisy-chain of end-hosts which takes long delay and no guarantee QoS. X6Ti, on the other hand, takes the advantages of ALM on the network with spare or no Xcast-aware router, otherwise it works in network layer multicast with high efficiency in term of bandwidth/delay like IP multicast.

6.2 Ease of Deployment

Following the IPv6 standardization, Xcast6 and X6Ti protocols can work on the current Internet where IPv6 is supported. Unlike other hybrid multicast protocol [8] which requires complicated configuration, X6Ti automatically detects and switches from ALM to network layer multicast wherever Xcast-aware routers are deployed. In other words, X6Ti supports incremental deployment - a key issue for a new protocol to be deployed on the Internet. IP multicast, on the other hand, requires all routers on the Internet to be upgraded.

6.3 State scalability and CPU processing

As introduced in [22], the unicast routing table today holds more than 300,000 network prefixes, and the number keeps increasing to more than one million in the next four or five years. As of today, one latest router is easy to handle the current routing table with the memory utilization is well below 50%. On the other hand, the estimation of the routing table growth rate in the future is more linear while the memory capacity growth is near-exponential. So, it seems the growth of routing table size is not a problem on the Internet. However, from the point of view of the operators in the paper [22], they show that there is an immediate routing scalability problem to address today. At first, there are many old

routers which are still running on the Internet. These kinds of old routers can be seen as the bottlenecks that cause the routing scalability problem for an ISP. However, upgrading router is come with the operational expense (cost of management, training, etc.). Thus, ISPs will likely try to address the problem with minimal cost and the least service impact or they prefer a technological solution than upgrading hardware solution. From 2006, after the Internet Architecture Board workshop on Routing and Addressing, there are many solutions have been proposed which are summarized in the special issue report [23]. From these points of view, the scalability of unicast routing table is really a problem that need to be solved today. For IP multicast protocol, the multicast routing table size is increasing linearly with the number of active multicast groups. As of mid-year 2009, the number of installed base of group video conferencing (ViC) systems was approximately 900,000 units, and it keeps increasing with the growth of 30% to the year of 2014 [24][25]. Clearly, if ViC is deployed using multicast service, the routing table size at routers can be explosive.

X6Ti model pushes most of the intelligent and complicated parts to the edge of the network (end-hosts). As a result, zero state at Xcast-aware routers makes it scale very well with very large number of active groups. In addition, there is no direct control message between end-hosts and routers helps to reduce CPU processing at routers. However, routers have to perform unicast table lookup for each IP address in the packet header. This increases the CPU processing in router. To alleviate this problem, the network operators can configure the routers so that if the CPU utilization is over a threshold, the Xcast service at this router will be temporarily stopped to reduce overhead. At this time, user applications are still working and X6Ti packets are bypassed these overload routers like the normal routers.

6.4 Fast route adaptation

In ALM, whenever an end-host joins/leaves, the new overlay routing tree should be disseminated to all the members in multicast session. Xcast6 and X6Ti, on the other side, only need to update routing information at the the source or the X6Ti sub-roots. In addition, with the support of Xcast-aware routers, Xcast6 and X6Ti work in network layer multicast with high rank of fast route adaptation like IP multicast.

6.5 Address allocation and routing

X6Ti does not support IPv4. It simply uses the unicast IPv6 address. Thus, it does not need a global multicast address allocation mechanism to avoid address collision. Moreover, X6Ti is simple and compatible with the current unicast protocols. X6Ti does not need any new protocols at network layer such as multicast group membership discovery or multicast routing protocol like IP multicast. In addition, X6Ti packets can make use of traffic engineering unicast paths.

6.6 The challenge of economics

Like Source Specific Multicast (SSM) [10], X6Ti can be a potential service from which an ISP can receive new revenue. The single source ownership of the multicast session can be easily determined. In addition, the number of members in a multicast session can also be determined assisting the ISP in charging multicast session based on different scale of use.

6.7 Header size overhead

The X6Ti datagram includes the IPv6 outer header (40 bytes), the IPv6 inner header (40 bytes), the routing header ($24 + 16^*N$ (bytes) where N is the number of destinations), the destination option header (40 bytes), the transport header (UDP header: 8 bytes) and payload. Assuming the MTU is 1500 bytes, thus, the payload length of an X6Ti packet is a function of N: $payload = 1348 - 16^*N$. It's due to application, user can add the suitable

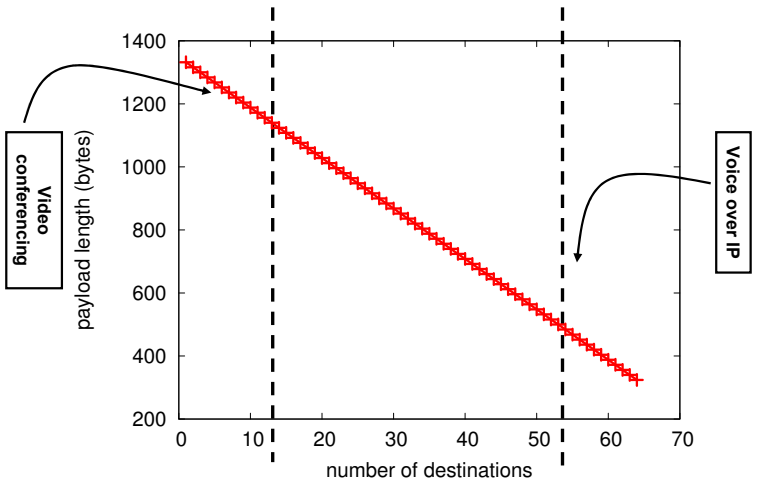

Figure 20: Relation of number of destinations and payload length

number of destinations (called $XCAST\_MAX\_NODE \leq 64$) into every packet. As shown in [26], the global processing time for an X6Ti packet is $t_G = \tau_1 + N^*\tau_2$, where τ_1 is the processing time of the IP and the X6Ti header while τ_2 is the lookup time for each entry in the list of destinations. Because the τ_1 is the same for every X6Ti packet, the total processing time for a packet with N destinations is smaller than for N packets with only one destination. For application like Voice over IP, the payload for every packet is small (20 to 160 bytes) [27]. Moreover, this application is delay-sensitive, thus sending one packet with the maximum number of destinations should be better than sending many packets each with the small number of destinations. In [26], the authors have also introduced a simple way to calculate the value of the number of destinations for a given multicast group.

6.8 Authorized sender and receivers

The X6Ti model can support authenticated subscription for both sender and receivers. By accepting/rejecting policies on the join request, the source host can control other hosts' ability to receive data from it. This is especially important for pay-for content (e.g. pay-per-view events), without this, the source can loss revenue. On the other hand, the receivers are assured of only receiving data from the source they desire. This prevents an unauthorized sender to send traffic to a multicast group.

7 Conclusion

In this paper, we propose X6Ti - a new hybrid multicast protocol of ALM and Xcast6. We believe that our model can overcome most shortcomings of IP multicast and ALM with small additional cost. In summary, X6Ti solves the problem of state scalability, address allocation and the challenge of economics. Moreover, X6Ti is simple and it can be deployed on the Internet without waiting for the support from the network operators. For the future work, the two-tier X6Ti can be extended to "n-tier" model to support better large multicast group. Besides, an existing ALM protocol (tree-based or mesh-based) can also be adapted to work with the X6Ti model. In addition, real applications using X6Ti should be deployed on the Internet to test the feasibility of the protocol.

Acknowledgements

This work has been partly funded by the European project FET AEOLUS, ANR DIMAGREEN[1] (DesIgn and MAnagement of GREEN networks with low power consumption). The authors would like to thank Yaning Liu, Eichii Muramoto, Ettikan Kandasamy, Yuji Imai, Huynh Thi Thanh Trang, Nguyen Quang Hoang and the network team at HCMUT for their advices and support.

References

[1] L. Aguilar. Datagram Routing for Internet Multicasting. In *ACM SIGCOMM*, 1984.

[2] S. Deering. Multicast Routing in Internetworks and Extended LANs. In *ACM SIGCOMM*, 1988.

[3] Christophe Diot, Brian Neil, Levine Bryan, and Kassem Doug Balensiefen. Deployment Issues for the IP Multicast Service and Architecture. In *IEEE Network*, 2000.

[4] Kevin Almeroth. The Three Ghosts of Multicast: Past, Present, and Future. In *TERENA Networking Conference (TNC)*, 2007.

[1] http://www-sop.inria.fr/teams/mascotte/Contrats/DIMAGREEN/wiki/

[5] R. Boivie, N. Feldman, Y. Imai, W. Livens, and D. Ooms. Explicit Multicast (Xcast) Concepts and Options. *RFC 5058*, 2007.

[6] T. Khoa Phan, N. Thoai, E. Muramoto, B. P. Lim, K. K. Ettikan, and P. Y. Tan. Treemap - The Fast Routing Convergence Method for Application Layer Multicast. In *IEEE CCNC*, 2010.

[7] M. Gerla, A. Fei, J. H. Cui and M. Faloutsos. Aggregated Multicast for Scalable QoS Multicast Provisioning. In *Tyrrhenian International Workshop on Digital Communications*, 2001.

[8] B. Zhang, W.Wang, et al. Universal IP Multicast Delivery. In *Journal Computer Networks*, 2006.

[9] S. Banerjee, C. Kommareddy, and B. Bhattacharjee. Scalable Application Layer Multicast. In *ACM SIGCOMM*, 2002.

[10] H. Holbrook, and B. Cain. Source-Specific Multicast for IP. In *RFC 4607*.

[11] C. Abad, W. Yurcik and R. H. Campbell. A Survey and Comparison of End-system Overlay Multicast Solutions Suitable for Network-centric Warfare. In *SPIE Defense and Security Symposium BattleSpace Digitization and Network-Centric Systems IV*, 2004.

[12] S. Banerjee, C. Kommareddy, et al. Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications. In *IEEE INFOCOM*, 2003.

[13] L. Lao, J. Cui, M. Gerla and D. Maggiorini. A Comparative Study of Multicast Protocols: Top, Bottom, or in the Middle? In *IEEE Global Internet Symposium*, 2005.

[14] B. Ye, M. Guo, D. Chen and S. Lu A Degree-constrained QoS-aware Routing Algorithm for Application Layer Multicast. In *Journal Information Sciences*, 2007.

[15] M. Bishop and S. Rao. Considering Priority in Overlay Multicast Protocols under Heterogeneous Environments In *IEEE INFOCOM*, 2006.

[16] K. Sripanidkulchai, A. Ganjam, B. Maggs and Hui Zhang. The Feasibility of Supporting Large-scale Live Streaming Applications with Dynamic Application End-points. In *ACM SIGCOMM*, 2004.

[17] M. S. Kim, S. S. Lam and D. Y. Lee. Optimal Distribution Tree for Internet Streaming Media. In *IEEE ICDCS*, 2003.

[18] www-01.ibm.com/software/integration/optimization/cplex-optimizer

[19] http://www.isi.edu/nsnam/ns/

[20] http://www.oversim.org/

[21] B. P. Lim, K. K. Ettikan, E. S. Lin, T. Khoa Phan, N. Thoai, E. Muramoto, and P. Y. Tan. Bandwidth Fair Application Layer Multicast for Multi-party Video Conference Application. In *IEEE CCNC*, 2009.

[22] X. Zhao, D. J. Pacella, and J. Schiller. Routing Scalability: An Operators View. In *IEEE Journal on Selected Areas in Communications*, 2010.

[23] Scaling the Internet Routing System: An Interim Report. http://www.cs.ucla.edu/~lixia/papers/10JSAC-editor.pdf

[24] A. W. Davis and Ira M. Weinstein. Video Conferencing - by the Numbers. In *Wainhouse Research*.

[25] Jonathan Edwards. Worldwide Enterprise Videoconferencing and Telepresence 2010-2014 Forecast.

[26] A. Boudani, A. Guitton and B. Cousin. Gxcast: Generalized Explicit Multicast Routing Protocol. In *ISCC*, 2004.

[27] www.cisco.com/application/pdf/paws/7934/bwidth_consume.pdf

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399