Global Multi-Point Streaming Experiments Based on the Flexcast Protocol

Seiichiro Tani†, Takeru Inoue, Shin-ichi Minato, Hirokazu Takahashi, Satoshi Kotabe, and Toshiaki Miyazaki

Abstract

With more individuals starting to create streaming data as a broadcast service, IP-multicast is the most popular mechanism for multicasting to recipients located over a wide area. However, there are several difficulties in using IP multicast as a personal broadcasting tool. IP-unicast-based multicast technologies are now attracting attention because they offer several advantages. Flexcast, one such technology, offers both excellent scalability and dynamic tree-optimization. This paper describes stream delivery experiments on the Flexcast protocol among widely dispersed locations in Japan and the U.S. The results verify the robustness and stability of the Flexcast protocol.

1. Introduction

The World Wide Web lets individuals broadcast contents, and streaming data is rapidly accounting for a higher percentage of the total traffic carried. Conventional broadcast techniques, however, waste too much bandwidth because the servers send streaming data directly to each recipient, even if there are many of them. To overcome this difficulty, multicast technologies have been intensively developed.

IP-multicast is the most popular mechanism for multicasting to recipients located over a wide area. Many multicast routing protocols (e.g., DVMRP [1], MOSPF [2], and PIMSM [3]) for IP-multicast have been proposed and most are being investigated for standardization at IETF (Internet Engineering Task Force). The protocols have been intensively studied from various viewpoints. When used in a wide-area network, PIMSM is one of the most efficient and stable protocols because it constructs reverse-path-based trees associated with only active clients. However, to use IP-multicast, we need special IP addresses, called IP-multicast addresses, to specify the multicast groups. This makes it difficult for individuals to multicast streams because it is necessary to obtain a unique IP-multicast address whenever IP-multicasting is desired. Furthermore, all routers need to be able to route IP-multicast packets, which restricts IP-multicast to only closed or experimental networks such as MBONE [4]. This is why it is impossible for individuals to use IP-multicast without difficulty.

IP-unicast-based multicasting technologies are attracting much attention as personal broadcast tools. One multicast technology that has been proposed is Flexcast [5]. It dynamically constructs a multicast tree by sharing common links among unicast (reverse) paths from clients to the server. Flexcast autonomously updates the optimal delivery tree when recipients emerge or disappear, and maintains the optimality of the tree regardless of user-host mobility and/or changes in IP routing; the Flexcast protocol is so simple that it is extremely scalable [6].

This paper describes experiments on stream delivery based on the Flexcast protocol. They were conducted among three widely dispersed locations with streaming servers and receivers located at each site. We observed the response time and the influence of communication delay jitter on Flexcast operation and the packet-loss rate. The results show that, in the absence of unusual congestion on any links, the Flexcast parameters offer sufficient margins, confirming the robustness and stability of the Flexcast protocol.

† NTT Communication Science Laboratories
Atsugi-shi, 243-0198 Japan
E-mail: tani@theory.brl.ntt.co.jp

2. Flexcast protocol

2.1 Basic operations

This section describes the basic operations of the Flexcast protocol. Flexcast has several advanced functions such as load balancing among nodes and host-mobility support, but here we focus on the basic operations closely related to the experiment.

Clients that want to receive multicast data send "join" packets containing the paired information of the server address and destination port. Legacy nodes on the path between Flexcast nodes simply forward the Flexcast packets (join packets and delivery packets, introduced later) since they are just IP-unicast packets. When a join packet arrives at a Flexcast node, the sender address written in the packet is registered with the routing table, called the delivery table, maintained by the node if the table exists; otherwise the table for the server is created before registration, and the node sends the server a join packet. This joining operation propagates from the client through intermediate nodes until the join packet reaches the server or a node that already has the table. The tables determine to which clients the multicast data carried by the delivery packets, is to be delivered. This operation is based on a "keep-alive mechanism". Each client periodically sends join packets as long as it wants to receive multicast data. The parent[*1] of the client sends multicast data if a join packet from the client arrives within some interval. (Such a client is called "active".) In other words, a client that stops sending join packets expires, and no multicast data is delivered to the client. The parent node also continues to send join packets to the server of the tree as long as it has at least one child still active. The same keep-alive mechanism works between the node and its parent. In this way, the keep-alive mechanism starts from clients and passes through each parent and child pair.

*1 Following the usual tree terminology, for each node (including the server and clients) of a multicast tree, we refer to the server- and client-side neighbors as the parent and children of the node, respectively.

An example of the above operation is shown in Fig. 1. Clients C1, C2, and C3 periodically unicast join packets to the server. These packets are routed as ordinary unicast packets. Node B, which lies on two routing paths, intercepts the packets and registers the sender addresses, C1 and C2, in its delivery table, and unicasts a join packet that lists B as the client, i.e., B works as a proxy. If A lies on the path from B to S and on the path from C3 to S, A picks up the join packets from B and C3 and registers B and C3 in its table, and unicasts a join packet that identifies A as the client to server S. When S receives the join packet, it unicasts delivery packets to client A. A copies the delivery packets and sends them to B and C3 after referring to the delivery table. In the same way, B sends the delivery packets to C1 and C2.

From the above, it is clear that delivery traces the reverse of the unicast path from the client to the server. Intuitively, the reverse-paths between clients and the server are agglomerated as much as possible by the Flexcast nodes on the paths. Note that the Flexcast protocol can work even if the reverse-paths differ from the unicast paths from the server to clients, say, the forward-paths. While forward-paths generally yield higher stream delivery quality than reverse-paths, forward-path-based tree construction often

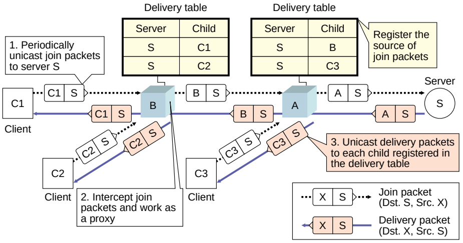
Delivery table
Server | Child
S | C1
S | C2
Delivery table
Server | Child
S | B
S | C3
Register the source of join packets
1. Periodically unicast join packets to server S
C1 S
B S
A S
C1
Client
B
A
Server
S
C2 S
C3 S
C2
Client
C3
Client
2. Intercept join packets and work as a proxy
3. Unicast delivery packets to each child registered in the delivery table
X S
Join packet (Dst. S, Src. X)
Delivery packet (Dst. X, Src. S)

Fig. 1. Basic operation of the Flexcast protocol.

results in complicated or non-adaptable protocols. Since IP networks are being optimized to support peer-to-peer communication, it is reasonable to assume that there will be little difference in quality between forward- and reverse-paths. Thus the Flexcast protocol uses the reverse-path-based tree, which yields scalability in terms of the number of nodes and adaptability to IP-routing changes.

2.2 Tunneling functions

Servers and clients in Fig. 1 need to be able to run the Flexcast protocol. In addition to these dedicated severs and client, ordinary IP multicast servers and clients can be used by introducing the Flexcast-IP multicast bidirectional translator called the Flexcast gateway. An example of Flexcast gateway use (G1 and G2) is shown in Fig. 2. G1 and G2 are connected to the segment of an IP multicast server and IP multicast clients, respectively. They create a tunnel through which IP multicast packets from server S pass to clients C. The tunnel can be a tree-shaped one connecting the server-side gateway and multiple client-side gateways because the tunnel itself is a delivery tree of Flexcast. This achieves remarkable efficiencies compared with the tunnel created between IP multicast routers. Client-side gateway G2 watches IGMP (Internet Group Management Protocol) [7] membership reports sent by IP multicast clients, extracts the flow ID (IP multicast address) carried, and starts to send join packets to server-side gateway G1 by referring to the address resolution

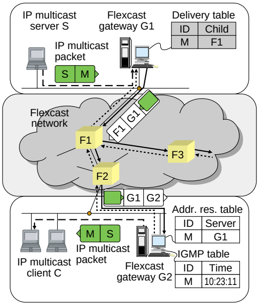

Fig. 2. Flexcast tunneling of IP-multicast packets.

table, which maps the flow IDs to the corresponding server-side gateways. When G1 receives join packets requesting flow M, it picks up IP multicast packets of M that are passing through the local segment, encapsulates them, and sends them as Flexcast delivery packets. G2 extracts the original IP multicast packets from the delivery packets and releases the IP multicast packets to the local segment. Finally, IP multicast clients get the IP multicast packets. G2 uses IGMP query to periodically check if IP multicast clients want to receive the flow and sends join packets only while the clients are active. Thus, the Flexcast tunnels are automatically created and removed under the control of the actions of the IP multicast clients.

3. Global streaming experiments

3.1 Environment

We conducted streaming experiments over NTT's experimental fiber-network connecting Japan and the U.S.A., which is called GEMnet, and Internet2 [8] in Sunnyvale, California. Streaming servers and clients were located in NTT Yokosuka R&D Center (hereafter called Yokosuka), the University of Southern California, Los Angeles (USC), and the University of Illinois, Chicago (UIC). Yokosuka is connected to GEMnet; USC and UIC are connected to Internet2. Since GEMnet and Internet2 have a bidirectional access point in Sunnyvale, USC and UIC can communicate with Yokosuka via Internet2 and GEMnet. GEMnet has a bottleneck link between Japan and the U.S.A. that has a constant bitrate (CBR) speed of 17 Mbit/s in each direction. This link was also the bottleneck of the whole network used in the experiments, since Internet2 offers 10-Gbit/s links while the local networks of USC and UIC have 1-Gbit/s links.

The main purpose of the experiments was to verify that the Flexcast protocol could handle widely separated locations: The distances involved are: Yokosuka to Chicago: 12,000 km, Yokosuka to Los Angeles: 9,000 km, and Chicago to Los Angeles: 4,000 km. Figure 3 schematically shows the networks used.

Each site had one or two Flexcast gateways, which can duplicate contents in addition to encapsulating and decapsulating them. IP-multicast servers and/or clients were connected to the network in which gateways were located. Since there were no Flexcast nodes between the gateways, two streams of the same contents passed through GEMnet when Yokosuka served both Los Angeles and Chicago. This was not a problem because the experiment was intended to verify the correct operation of the Flexcast protocol over

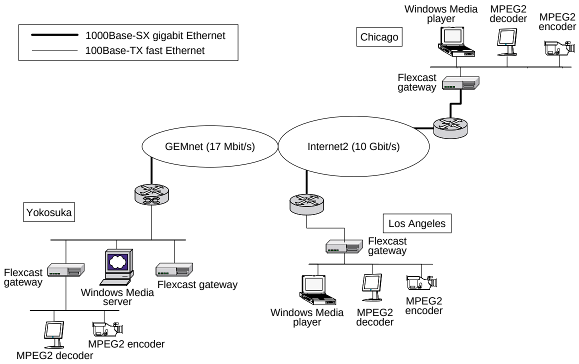

Fig. 3. Schematic view of the network.

a very-wide-area network, not to measure its scalability in terms of the number of clients.

We observed traffic at the three sites and checked the quality of moving pictures delivered by the Flexcast protocol.

3.2 Parameters and metrics

We used implemented software-based Flexcast gateways on personal computers (PCs) and two types of commercial IP-multicast server/client systems as shown in Table 1. The average bandwidth of MPEG2 streams was 7 Mbit/s and that of Windows Media ver. 7 was 625 kbit/s.

We obtained traffic data by using TCPDUMP and the packet-loss rate by serially numbering packets at server-side gateways and checking the packet-number of each packet received at client-side gateways. We measured the following metrics.

- Response time: Interval between the time when the client-side gateway sent the first join packet to the time when the gateway received the first delivery packet.
- Join-packet sending-interval: Time gap between two consecutive join packets, initially set to 1.0 s.
- Join-packet arrival-interval: Time gap between two consecutive join packets as received.
- Packet-loss rate: Percentage of packets lost

Table 1. Commercial IP-multicast servers/client systems used in the experiments.

System	Codec type	Average bandwidth (bit/s)
1	MPEG2	7 M
2	Windows Media ver. 7	625 k

3.3 Scenarios

The experiments were conducted using the following scenarios:

- Scenario A (Fig. 4): MPEG2 live streams were delivered to Chicago and Yokosuka from Los Angeles. Only 7.5-Mbit/s traffic flowed through the Japan-U.S. bottleneck link of 17 Mbit/s in each direction. This scenario assessed the congestion-free situation.
- Scenario B (Fig. 5): Three sets of contents were simultaneously delivered: MPEG2 live streams were delivered to Chicago and Los Angeles from Yokosuka, another MPEG2 live stream to Yokosuka from Los Angeles, and Windows Media live streams to Chicago and Los Angeles from Yokosuka. Thus, over 16 Mbit/s of traffic had to pass from Japan to the U.S., while 7.5 Mbit/s of traffic passed from the U.S. to Japan. This scenario corresponded to a congested situation.

3.4 Data analysis (scenario A)

The measured data is summarized in Table 2. The standard deviation (stdev.) of the join-packet arrival-interval was very small (below 10 ms) and was not significantly different from the join-packet sending-interval. This confirms the good stability of the Flexcast protocol, which allows the intervals of consecutive join packets to be up to 2 s in the current setting.

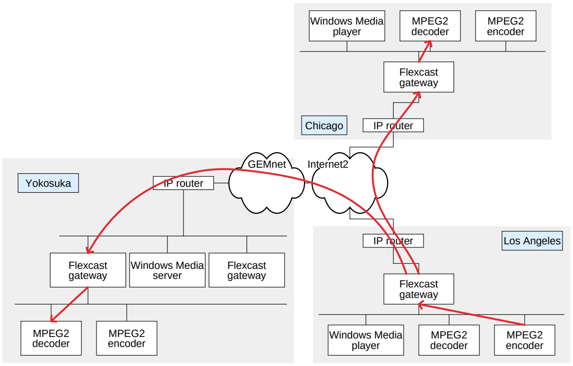

Fig. 4. Streaming paths in scenario A.

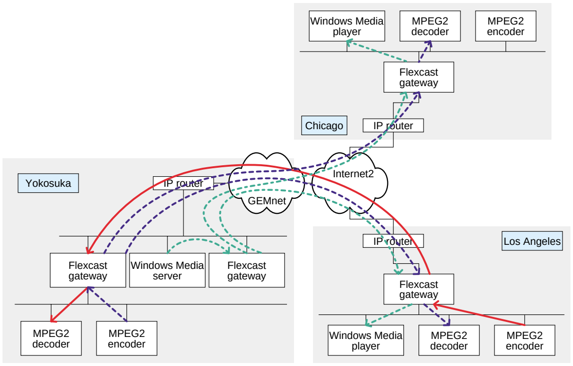

Fig. 5. Streaming paths in scenario B.

The packet-loss rate of 5% may be due to the burstiness of MPEG2, whose peak rate exceeded the bandwidth of the bottleneck link.

The round-trip time (RTT) between Los Angeles and Yokosuka was 128 ms. This RTT is included in the response time of 136 ms. The delay at the Yokosuka gateway, 136–128=8 ms, is the sum of the waiting-time of IP multicast packets and processing time.

3.5 Data analysis (scenario B)

The traffic rate through the bottleneck link exceeded 16 Mbit/s in the Japan-to-U.S. direction. Though the bandwidth of GEMnet is 17 Mbit/s (CBR), this situation is critical because streaming traffic is very bursty. We can see the influence of the congestion of the bottleneck link on both join and delivery packets.

To examine the influence of congestion on the join packets, we observed the join packets from Yokosuka because the congestion occurred in the Japan-to-U.S. direction. The measured data is summarized in Table 3. The standard deviation of the join-packet arrival-interval is extremely large compared to that of the join-packet sending-interval. This is due to the communication delay jitter caused by the congestion. Figure 6 plots the join-packet sending/arriving intervals. The largest and second-largest values of the join-packet arrival-interval were around 3 s and 2 s, respectively, which means that some join packets may have been lost. The packet-loss rate in Table 3 is the loss-rate of delivery packets from Los Angeles to Yokosuka. The loss-rate was much lower than that in the opposite direction, i.e., over 15%.

4. Conclusion

This paper described streaming experiments conducted to assess the robustness and stability of the Flexcast protocol when used to deliver streams to widely dispersed locations. We set streaming servers and clients in three locations: Yokosuka in Japan and Chicago and Los Angeles in the U.S. Prior to the experiments, our analysis indicated that the jitter of

Table 2. Measured data in scenario A.

Observed flow	MPEG2
Server location	Los Angeles
Client location	Yokosuka
Response time (av.) [ms]	136
Join-packet sending-interval (av.) [s]	1.00
Join-packet sending-interval (stdev.) [s]	2.15E-03
Join-packet arrival-interval (av.) [s]	1.00
Join-packet arrival-interval (stdev.) [s]	2.12E-03
Packet-loss rate (%)	5

Table 3. Measured data in scenario B.

Observed flow	MPEG2
Server location	Los Angeles
Client location	Yokosuka
Response time (av.) [ms]	143
Join-packet sending-interval (av.) [s]	1.00
Join-packet sending-interval (stdev.) [s]	2.69E-03
Join-packet arrival-interval (av.) [s]	1.12
Join-packet arrival-interval (stdev.) [s]	0.138
Packet-loss rate (%)	5.2

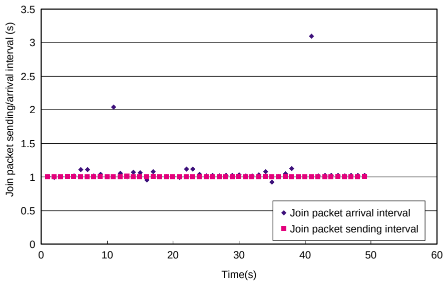

Fig. 6. Join-packet sending/arrival-interval of scenario B.

communication delay would probably have a considerable influence on Flexcast performance, but the results show that the jitter was much smaller than expected and that the Flexcast parameters offer sufficient margin if no links experience unusual congestion. Good picture quality and quick response time were also confirmed in the experiments. We intend to perform stream delivery experiments among many locations to verify the scalability of the Flexcast protocol in terms of delivery tree size.

References

[1] "Distance Vector Multicast Routing Protocol," RFC1075, IETF home page (http://www.ietf.org/).
[2] "Multicast Extensions of OSPF," RFC1584, IETF home page (http://www.ietf.org/).
[3] "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," RFC2117, IETF home page (http://www.ietf.org/).
[4] V. Kumar, "Mbone: Interactive multimedia on the Internet '95," New Riders Publishing, 1995.
[5] S. Tani, T. Miyazaki, and N. Takahashi, "Adaptive Stream Multicast Based on IP Unicast and Dynamic Commercial Attachment Mechanism: An Active Network Implementation," In Proceedings of the Third International Working Conference on Active Networks (IWAN2001), Springer-Verlag, Lecture Notes in Computer Science, Vol. 2207, 2001.
[6] S. Ohta, S. Tani, and T. Miyazaki, "Multicast as a traffic variance smoother for IP streaming service," in Proceedings of Networks 2002, pp. 105–110, 2002.
[7] "Host extensions for IP multicasting," RFC1112, IETF home page (http://www.ietf.org/).
[8] http://www.internet2.org/

Seiichiro Tani
Research scientist, NTT Communication Science Laboratories.
He received the B.E. degree in information science from Kyoto University, Kyoto in 1993 and the M.S. degree in information science from the University of Tokyo, Tokyo, in 1995. Since joining NTT in 1995, he has been engaged in research on VLSI testing, adaptive networking technology, and quantum computation. His current research interests include discrete algorithms, adaptive network protocols, and quantum computation. He received the 2000 best paper award of the Institute of Electronics, Information and Communication Engineers (IEICE) Information and System Society. He is a member of IEEE, IEICE, the Information Processing Society of Japan (IPSJ), and the Association for Computing Machinery.

Takeru Inoue
Media Networking Laboratory, NTT Network Innovation Laboratories.
He received the B.E. and M.E. degrees in engineering physics from Kyoto University, Kyoto in 1998 and 2000, respectively. He joined NTT in 2000. His research interests are in adaptive networking technologies. He received the research award of IEICE of Japan Information Network Group in 2001. He is a member of IEICE.

Shin-ichi Minato
Senior Research Engineer, NTT Network Innovation Laboratories.
He received the B.E., M.E., and D.E. degrees in information science from Kyoto University, Kyoto in 1988, 1990, and 1995, respectively. He has been working for NTT since 1990. He was a visiting scholar at Stanford University in 1997. He served as a lecturer for Keio University, Fujisawa, Kanagawa, from 1999 to 2001. Currently his research interests include data structures and algorithms for innovative communication network system design. He wrote "Binary Decision Diagrams and Applications for VLSI CAD" (Kluwer, 1995). He is a member of IEEE, IEICE of Japan, and IPSJ. He has served as a program committee member for international conferences such as ACM/IEEE DAC, ICCAD, and DATE.

Hirokazu Takahashi
NTT Network Innovation Laboratories.
He received the B.E. and M.E. degrees in electrical engineering from Nagaoka University of Technology, Nagoya in 2000 and 2002, respectively. He joined NTT in 2002. His current research is on reliable realtime multicasting. He is a member of IEICE of Japan.

Satoshi Kotabe
Research scientist, NTT Network Innovation Laboratories.
He received the B.E. and M.E. degrees in electronic engineering from Ibaraki University, Hitachi, Ibaraki in 1993 and 1995, respectively. He joined NTT in 1995. His current research is on high-speed data communication network architectures. He is a member of IEEE and IEICE of Japan.

Toshiaki Miyazaki
Senior research engineer, supervisor (a research group leader), NTT Network Innovation Laboratories.
He received the B.E. and M.E. degrees in applied electronic engineering from the University of Electro-Communications, Tokyo, in 1981 and 1983, and the Ph.D. degree from Tokyo Institute of Technology, Tokyo in 1994. Since joining NTT in 1983, he has been engaged in research on VLSI CAD systems including high-level synthesis, logic design tools, and CAD frameworks. Since 1993, he has been involved in developing telecommunications-oriented FPGAs and their applications, and next-generation networking protocols. His research interests are in programmable hardware systems, adaptive networking technologies, and autonomous systems. He is a member of IEEE, IEICE of Japan, and IPSJ.