TCS-TR-A-06-10

TCS Technical Report

A Framework of Programmable Multicast Applications Using Flexcast and Java Applet

by

Shin-ichi Minato, Hirokazu Takahashi,
Takeru Inoue, Hiroshi Tohjo, and Kan Toyoshima

Division of Computer Science

Report Series A

January 15, 2006

Hokkaido University
Graduate School of
Information Science and Technology

Email: minato@ist.hokudai.ac.jp

Phone: +81-011-706-7682
Fax: +81-011-706-7682

A Framework of Programmable Multicast Applications Using Flexcast and Java Applet

Shin-ichi Minato
Division of Computer Science, Hokkaido University
North 14, West 9, Sapporo, 060-0814 Japan

Hirokazu Takahashi
NTT Network Innovation Laboratories
1-1 Hikarinooka, Yokosuka-shi, 239-0846 Japan

Takeru Inoue
NTT Network Innovation Laboratories
1-1 Hikarinooka, Yokosuka-shi, 239-0846 Japan

Hiroshi Tohjo
NTT Network Innovation Laboratories
3-9-11 Midori-Cho, Musashino-Shi, 180-8585 Japan

Kan Toyoshima
NTT Network Innovation Laboratories
1-1 Hikarinooka, Yokosuka-shi, 239-0846 Japan

January 15, 2006

(Abstract) As broadband Internet access is more commonly provided, streaming applications gain popularity and wide-area multicast technology will be more important. Recently, a new overlay multicast protocol, "Flexcast" has been proposed to realize the wide-area multi-point contents delivery. Flexcast provides the multicast function in a higher layer than the IP routing; Flexcast streaming packets are forwarded between routers by using unicast. Therefore, Flexcast can work well even if the legacy routers remain in the Internet. In addition, Flexcast features that an efficient delivery tree is autonomously constructed and maintained. Flexcast is a promising technology to realize wide-area multicasting in a reasonable cost on the Internet.

While Flexcast addresses the multicast routing problem, there is another difficult problem in distributing multicast application programs. In the streaming applications, the software must be consistent among all servers and clients. However, especially when uncountable many (or public) users are targeted, it is hard to distribute the application programs to all prospective users, and also quite difficult to keep uncountable servers and clients up-to-date. In this paper, we propose a new framework of programmable multicast applications based on Flexcast protocol and Java Applet. Currently, Java Applet technology is widely utilized for various practical applications; however, all of those are limited to use unicast communications. Our work is the first practical approach to develop the multicast streaming software with Java Applet. Since Applets are distributed among clients on demand, they are kept up-to-date without any explicit synchronizing mechanism. The service providers do not need to distribute the application programs to all users beforehand, and it is anytime possible to update or modify their programs. They can design and customize their original service freely, not restricted by the specifications of a commercial streaming player which is dominant in the market.

This paper describes the structure of our application framework. The experimental results confirm the feasibility of our method.

1 Introduction

The Internet access is one of the universal services today, and many people enjoy WWW (World Wide Web) and E-mail services. In the beginning, the Internet was used for delivering a small capacity contents such as text data and still pictures, but the recent progress of broadband access technologies of ADSL and FTTH enable us to deal with the broadband streaming contents in a feasible costs. In the

near future, the Internet will commonly be used for delivering live movies of various events, such as music concerts, sports athletic meets, lecture meetings and the ceremonies, sending to multiple destinations simultaneously.

However, most of multi-point contents delivery systems available today are based on unicast communications, and they have a problem that the traffic increases in proportion to the number of destinations. As the contents become more wideband, this problem will be more serious in the near future. Although IP Multicast[1] technologies have been studied for long time, there are still technical and economical problems, and usually they are not operated in the public wide-area networks beyond router segments. Consequently, we cannot readily use the multi-point contents delivery in the current Internet.

The authors and colleagues have been proposed a new overlay multicast protocol "Flexcast"[9][10][4][5], that addresses the above-mentioned problem. Flexcast provides the multicast function in a higher layer than the IP routing, and the communication between routers is done by the unicast. Therefore, we can start the multicast service without waiting for deployment to all the routers in the network. In addition, the Flexcast does not need manual configuration of the delivery tree beforehand. It features that an efficient delivery tree is autonomously constructed and maintained. Thus, Flexcast is a promising technology to realize the service of "anytime, anywhere, and anyone's broadcasting" in a reasonable cost on the Internet.

By the way, the wide-area multi-point contents delivery technology has not only the problem of network routing, but also another important problem on the application programs in the server and client hosts. That is, when we use the original application program of multicast service, it is difficult to distribute and install the program for uncountable many (or public) users, and also difficult to maintain the program to be the latest version at each user's host. One realistic solution is to use a commercial Web browser or a media player that have a dominant market share. However, in this case, we lose the initiative of developing the multicast applications freely, because it is strongly restricted by the specification of the commercial software.

To solve the problem above-mentioned, we propose the method of using Flexcast and Java Applet[6] for the multicasting application program. Currently, Java Applet technology is widely utilized for various practical applications, however, all of those are limited to use one-to-one communications. Our work is the first practical approach to develop the multicast streaming software with Java Applet. In our method, the service providers do not need to standardize the receiver program, and no need of prior distribution for all prospective users. Using this programmable applications framework, we can easily develop and operate various types of multi-point contents delivery services in a less initial cost.

In the following sections of this paper, we first describe the Flexcast protocol and current problem in Section 2. We then show the method for programmable application framework based on Java Applet in Section 3. In Section 4, we show a prototype implementation and its evaluation to confirm the practicality of our method.

2 Flexcast: autonomous wide-area multicasting method

For avoiding traffic congestion in multi-point contents delivery systems, multicast communication techniques have been studied for long time. In general, multicast technology constructs a tree delivery route, and the delivery packets are duplicated in the branching nodes. In this way, the network traffic and transmission server load can greatly be reduced[7].

Despite such a great advantage, the multicast techniques have been used mainy in a local network closed in one laboratory or office, and also used in some fixed/closed wide-area networks, such as MBone[3], isolated from the public Internet. Currently, the IP Multicast protocol[1] is adopted as IETF standard and known most widely. However, the IP Multicast cannot start delivering until all the routers in the delivery network can handle

IP Multicast protocols, because those multicasting functions are implemented in the network routing layer. In addition, there are many versions and parameter settings in the IP Multicast protocols, but the router products sometimes support them differently. We need a special attention to operate IP Multicast service in wide-area network beyond the router segment. Therefore, the IP Multicast is basically not operated in the public wide-area network which consists of a number of network providers with different operation policies. Currently, IP Multicast is hardly used for multi-point contents delivery service in the Internet.

The authors and colleagues have been proposed a new multicast protocol "Flexcast"[9][10][4][5], that addresses the above problem. Flexcast provides the multicast function in a higher layer than the routing, and the communication between routers is done by the unicast. Therefore, we can start the multicast service without waiting for deployment to all the routers in the network. The outline of Flexcast technology is shown as follows.

2.1 Flexcast Protocol

Figure 1 shows a basic procedure of the multi-point contents delivery using Flexcast.

- The user, who wants to transmit Flexcast contents, opens the (global) IP address of the server host to the public.
- The user, who wants to receive Flexcast contents, sends request signals toward the IP address of the Flexcast server. The request is a simple UDP packet with the port number peculiar to Flexcast protocol, and unicasted to the server periodically (for instance, once per second).
- When the request packet is received, the Flexcast server transmits the contents delivery packets in unicast of UDP packets toward the IP address and the port number of the source of request. When the reception of the request packets stops for a certain period (for instance, five seconds), the Flexcast server recognizes the session closed and stops the data transmission.
- *Flexcast splitters*, that serve the function to aggregate the Flexcast flows, are properly arranged in the network. We do not have to make all the routers in the network are Flexcast splitters. The Flexcast works well when legacy routers exist in the network. Even if there are no Flexcast splitters in the network, it works as just one-to-multi-point unicast transmission. There are more Flexcast splitters in the network, more chance to aggregate the traffic.

Next, we show the basic operation of flow aggregation in Flexcast splitters[1], with Fig. 2. The client (receiver) hosts C1, C2, and C3 periodically send the request packets toward the server host S. The request packet is transported in usual unicast routing. When a Flexcast splitter B exists on this route, the request packets are intercepted by the splitter B, and the source IP addresses C1 and C2 in the request packets are recorded in the delivery table of the splitter B. Then, the splitter B becomes a new request client periodically sending packets toward the server S. That is, B becomes a proxy of C1 and C2.

Similarly, the Flexcast splitter A on the route from B to S also intercepts the request packets from B and C3 and records the source IP address in the delivery table. Then, the splitter A becomes a new proxy sending periodical packets toward the server S.

The server S receives the request packets only from A, and the server S transmits contents delivery packet to A. The splitter A duplicates and relays the delivery packet from S toward B and C3, which recorded in the delivery table of A. The splitter B also duplicates the delivery packets from A and relays them to C1 and C2.

As the result of the above procedure, the clients C1 C2, and C3 do not know the existence of the splitter A and B, and it seems to directly communicate with the server S in one-to-one manner.

[1]In the current version of Flexcast protocol[4][5] developed in NTT laboratories, IP Multicast (IGMPv2 and v3 conforming) is locally used at a server and a client host, and the UDP unicast is used in the wide-area network beyond the router segments. Therefore, there is a somewhat different in detail point though the basic principle is the same.

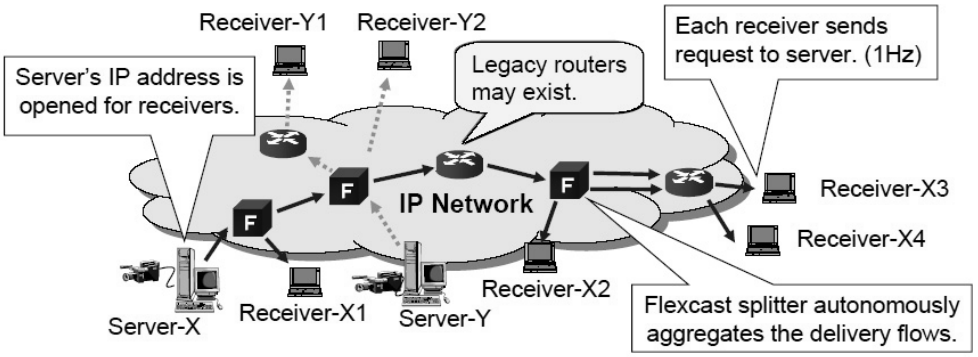

Figure 1: Multi-point contents delivery based on Flexcast.

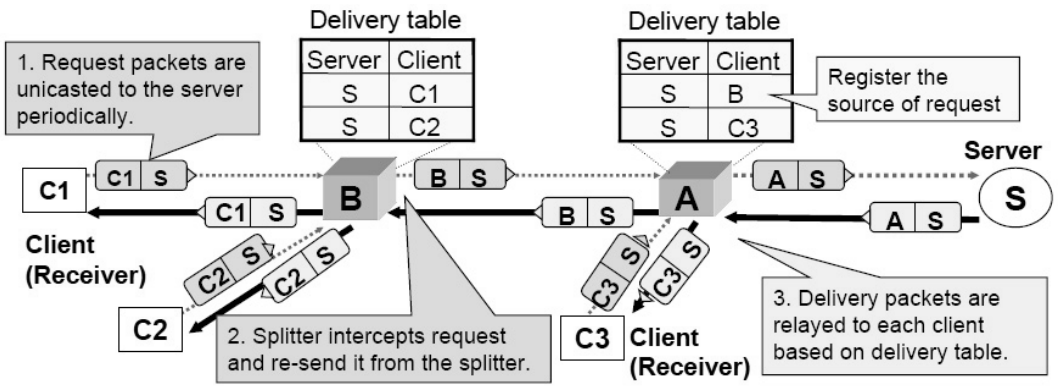

Figure 2: Basic operation of Flexcast splitters.

Moreover, the contents packets are only simply delivered from server S to the other host who sends the request. Namely, Flexcast is transparent[2] for the client and the server hosts. Therefore, we do not have to make any configuration the adjacent router or client/server hosts when we introduce the Flexcast splitters in the IP network. It makes easy to gradually deploy the Flexcast splitters in the network. We can start the Flexcast service without waiting the complete deployment of the splitters for the entire network. In addition, the Flexcast has an excellent feature that the optimal delivery tree is constructed (and updated) autonomously only using the request packets, even if the number and locations of the client hosts cannot be estimated beforehand.

Consequently, Flexcast is considered as a promising technology that realize "anytime, anywhere, and anyone's broadcasting" on the Internet with a reasonable cost.

2.2 Problem on distribution and maintenance of application programs

Flexcast is a very simple, light protocol based on UDP packets, and it is independent of the higher application layer. This means that the application program can be designed freely. However, there is a difficult problem in actual situations.

In general, the multi-point contents delivery service requires the application programs in the server and the client hosts, respectively, which are cor-

[2]In the current version of Flexcast implementation[4][5], in order to facilitate the wide-area network operation and management, the source IP address in the IP header of delivery packet is not always the server S, but the address of actual source host (a Flexcast splitter) is recorded. The IP address of the server S is recorded in the "Flexcast header," which is designed in the UDP payload. So, strictly speaking, it is not transparent in IP layer.

responding to each other correctly. Namely, it is necessary to standardize the program beforehand when a lot of receivers are expected, and need to distribute the program for all the receivers before starting the service. However, especially in public/open community service, it is hard to surely distribute the program for all prospective users and to ask everyone to install the program properly. Moreover, sometimes we need to bug correction or version up to improve the receiving program, and this is also difficult work.

When we assume the users who are not familiar with software installation and maintenance, a realistic solution would be to use a Web browser and a media player which are the most dominant commercial software products in the market. Recently in Japan, the market share of Windows PC is overwhelming for broadband access terminals, hence Internet Explorer and Windows Media Player is the most popular streaming contents receiver in such environment. However, the data format for Windows Media Player is not officially opened for public, and thus it is hard to add some original functions to the receiving program. Moreover, even if we could know the data format and could modify or improve the functions of the player, we will always be afraid that the program suddenly be in trouble because the data format may often be modified or improved by an initiative of the software makers. This means that the dominant commercial software requires a hidden significant cost to keep the correctness of the application programs, and it is more severe problem in multi-point (public) contents delivery services, comparing to the ordinary one-to-one communications.

After all, the most important problem for multicasting service providers is that they cannot design or customize the applications freely because the internal data format and behavior of the commercial software is closed and they are not programmable. To address such a problem, we propose a framework of designing multicast applications using a programming language. This method does not require the standardization of application software, and no need to distribute the programs for all users before starting service.

3 Programmable application framework based on Java Applet

In this section, we first briefly explain Java Applet and why we chose Java Applet as a programming language together with the Flexcast. We then describe the detailed techniques to implement the server and receiver programs in our framework.

3.1 Java Applet

Java[6] is a programming language published from Sun Microsystems Co. in 1994. The semantics of this language is defined on the Java virtual machine (JavaVM). A Java source program is first compiled into JavaVM machine language codes, and they are executed by a JavaVM emulator running on the target machine. This mechanism provides high compatibility between different platforms. As the spread of WWW in the latter half of the 1990's, Java language becomes popular and widely used. A number of multimedia libraries, such as sending/receiving Web contents including image and voice data, have been developed in Java language, and opened to the public.

Java Applet is a limited class of Java language, assuming to be performed by a Java processing system built in a Web browser. An applet program is downloaded from the Web server to client PC, and immediately executed by a Java processor in a Web browser system. The output of execution is displayed in the browser screen. Java Applet features that we can easily implement active animation programs and interactive contents running on a Web browser with the key board and mouse device input. At the time of Java Applet published, the technology was novel and attractive, and various applets have been developed.

Recently, we also have Java Servlet technology, which is working at server side, and the Servlets are more often used instead of Applets. If we consider one-to-one unicast applications, the data processing can be located at whichever, the server side or the client side. In this case, it is more simple way to put the program at the server side only. However,

if we consider the multi-point contents delivery systems, we do need an intelligent receiving program at each leaf of the delivery tree, and Java Applet technology is useful. Actually, the Java Applets are now widely used in cellular phones, which have a restricted communication bandwidth.

For using Java Applet, a JavaVM module should be installed in the Web browser properly. Currently, the JavaVM can be build in the most of popular browsers, such as Internet Explorer, Netscape, and Mozilla(Firefox), on either of Windows, Mac, and Linux OS. One problem is that JavaVM might not be installed at the time of purchase of PC[3]. If we transmit the contents for public, each user should install the JavaVM properly in one's own PC, and the installation problem still remains. However, the JavaVM installation is required only once after obtaining a new PC, and the users do not need to install or update applications every time on specification change or version up. It greatly reduces the user's labor costs. Currently, the specification of JavaVM is enough stable, and the update of JavaVM itself is rarely needed.

Besides Java Applet, Flash and JavaScript are also widely used in recent Web contents, and they can be used for designing programmable functions. So we consider the possibility of those technologies to be alternative solutions for our purpose. Unfortunately, both Flash and JavaScript basically use the communication facilities of the Web browser, and cannot operate the communication port directly. Therefore, they can use only HTTP communications with TCP port. TCP is limited to be used in one-to-one unicast communications since the both ends corporate each other for packet resending and rate control. This makes difficult to use TCP for multicast communication. We consider that Java Applet would be the current best option for our purpose, because it can operate UDP communication port directly, various libraries on image and voice processing are provided, and working on more than one major platforms.

3.2 Combination with Flexcast

Currently, Java Applet technology is widely utilized for various practical applications, for instance, "i-appli"[2] for mobile Internet is one of the most successful service. However, all of those are limited to use unicast communications. There has not been any practical result of using multicast with Java Applet. Java Applet has a strict security restriction that the client host cannot access any remote host except only the server host which provides the Applet program[4]. Conventional IP multicast uses a special "class-D" address, called a *group address*, for the destination of IP packets. Since the group address is different from the server address, Java Applet cannot open a communication port for the multicast server.

When we use Flexcast, the protocol is transparent for the server and client host as shown in Fig. 2, and the client host seems to access the server with one-to-one unicast communication. This means that Java Applet can access a multicast-type contents server without violating the security restriction, provided that the Applet program is downloaded from the Flexcast server[5].

Figure 3 shows a basic architecture of our system. In the server host, the Web server process (Apache etc.) is running as well as the Flexcast server process. The Web server contains a Web page (HTML file) of the service entry, and has opened the Web page's URI to the public as an identifier of the contents. A Java applet program for the Flexcast receiver is embedded in the Web page.

The user who wants to start Flexcast contents receiving, at first accesses the URI of the entry Web

[3]Before October 2004, Internet Explorer was equipped with "Microsoft JavaVM" in default setting, and there was a problem that Sun's official JavaVM and Microsoft's one are not completely compatible. As a result of the lawsuit of the two companies, Microsoft JavaVM is abolished, and the user oneself should install Sun's JavaVM after obtaining a new PC. This installation is a little hard work for beginners, however, some maker's PCs are equipped with a JavaVM installer to make easy for beginners.

[4]The special certified remote host can be accessed exceptionally, but this procedure is also hard for beginners as well as installing a new software.

[5]As mentioned in the footnote of Section 2, the current implementation[4][5] of Flexcast splitter is not completely transparent. It is not difficult to address this problem, and near future, our Flexcast splitter will support the transparent mode.

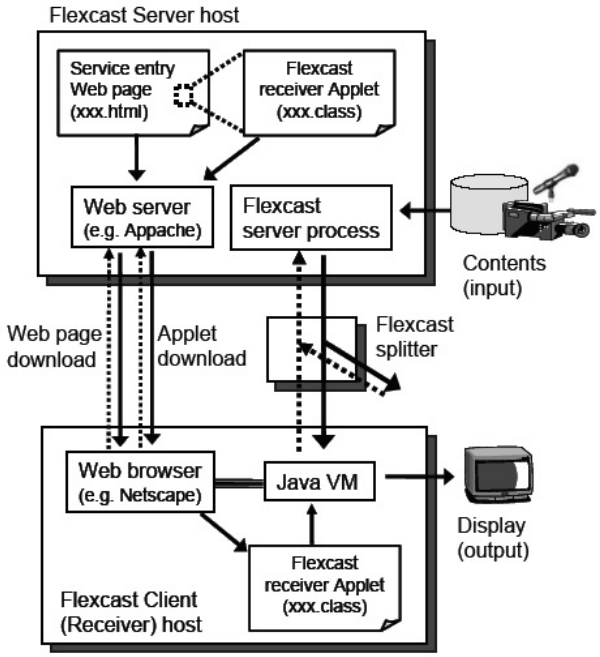

Figure 3: Flexcast contents delivery system using Java Applet.

page by a Web browser. Then, the Applet program of the Flexcast receiver is downloaded, and it is executed on JavaVM immediately. The Applet is programmed to send Flexcast request packets toward the server, and to receive the delivery packet returned from the server, and then restore and display the contents to the Web browser screen. In the network between the Flexcast server and client hosts, we can insert any number of Flexcast splitters. The splitters aggregate the request flows and split the delivery flows autonomously, and as the result, an efficient multi-point delivery tree is generated in the network.

3.3 Design of Applet programs for Flexcast

The Flexcast client (receiver) program is described in Java Applet. It is composed of the thread that periodically sends the request packets toward the Flexcast server, the thread that receives the delivery packets from the server, and that restores and displays the contents to the screen according to the applications. On the other hand, the Flexcast server program is not written in Java Applet but a general Java application. (We may use any other programming language such as C, Perl, but it would be easier to use the same language for the pair of server and client.) In the Flexcast server program, the delivery data is divided into a limited size of the UDP packets according to the specification of the client program, and they are sent in a prescribed transfer rate toward the source IP address of the periodic request packets. When the request packets are lost for a certain time period, the session is automatically closed.

In the Flexcast server/client programs, the basic UDP packets are handled completely based on the Flexcast protocol, but we can freely design the higher layers as far as Java language allows, such as the specifications of presentation or application layers processed after receiving Flexcast delivery packets. For example, we may describe an Applet program that handles a JPEG image file with a number of UDP packet sequence, and receiving the image files in a few frame/sec to be shown on the screen in real time. In this way, we can serve a simple live movie multicast application. As another example, if the program receives an image file every five or ten seconds and updates the screen immediately, it will be used for delivering the lecture slides to multiple classrooms. Moreover, if we add the function of playing the sound, the program can also be used for an emergency system to show the evacuation route with an alarm sound.

These examples show that we can freely program various multi-point contents delivery services using Java Applet. We do not have to standardize and distribute the receiver program to the users before starting the service, since the program is provided to the client hosts automatically as a Java Applet. Therefore, even when uncountable many users are targeted, it is anytime possible to update or modify the service, only considering the contents server's schedule.

4 Implementation and experimental result

To evaluate validity of our framework of multicasting applications, we implemented a prototype

Figure 4: An example of screen shot of a client (receiver) host.

system and conducted experiments to confirm the operations. For the Flexcast server, we used a Pentium-4 PC (800MHz, 512MB, 100Base-T Ethernet) with SuSE Linux 9 and Java SDK 1.4.2. On the other hand, we prepared two different PCs as receivers, Pentium 1.2GHz with Windows XP Pro SP2, and Pentium-4 800MHz with SuSE Linux 9. We tested more than one Web browsers on each platform, Internet Explorer Ver. 6.0 and Netscape 7.1 on the Windows PC, and Mozilla 1.6.74 and Konqueror 3.2.1 on Linux PC, respectively. We used Java plug-in Version 1.4.2_06 both in Windows PC and Linux PC.

In this experiment, we implemented a multicasting application that the Flexcast server transmits JPEG image files periodically in a certain fixed time interval, and the receiver repeats drawing of the recent image on the Web browser when the image file is delivered. In the Flexcast server, the image file are divided into multiple UDP packets each of which is up to 1kByte length, and a sequence number is stamped on each packet. In the receivers, the image files are reconstructed from the UDP packet sequence. Our receiver program can detect a packet loss based on the sequence numbers, and the erroneous image files are automatically skipped without drawing on the screen. In addition, our Flexcast server program controls the peak rate by inserting an interval at least 10msec between two consecutive UDP packets. The server program can deliver more than one streaming from one server, namely, more than one Flexcast Applets can be embedded in one Web page and simultaneously activated.

In this framework, we can freely specify the size of images and frequency of image update by the server-side application program. This framework can be used for various applications, for instance, live transmission of monitor camera pictures, lecture note delivery, and real time advertisement banners in the Web pages. The server program is written in 250 lines of Java code, and receiver program is 190 lines of Java Applet code.

We conducted an experiment of putting one Flexcast server in a LAN, and two different streaming contents are simultaneously transmitted from the same server. Those two contents are received

by multiple receivers located in the same LAN. The specifications of the two streaming contents are as follows.

- Simple movie delivery program that transmits JPEG images of a live camera, 4 frames/sec (320×240 pixels, 7.8kByte/frame in average). Data transmission rate is about 250kbps in average.
- Lecture note delivery program that transmits JPEG images of PowerPoint slides, one frame in every six seconds (800×554 pixels, 67kByte/frame in average).Data transmission rate is about 90kbps in average, and 800kbps in the peak rate (When the lecturer proceeds the next slide).

We tried to transmit the above two different contents simultaneously, and total ten flows of streaming are delivered to the Web browsers running on three client hosts. In this case, we confirmed that the performance of Java Applet is sufficient, and no problem to use heterogeneous Web browsers and OS. Figure 4 shows an example of the screen shot of receiving host.

Next we checked the delay time between the image update on the server and its reflection on a client host. It was about 0.1 second for the live camera image application, and 0.3 to 1 second for the lecture slide application. This delay time is mainly caused by constructing the image from the fragmented UDP packet sequence at the receiver host, and we have to wait to receive all parts of one image before update the screen. Thus, this delay is almost irrelevant to the performance of interpreting Java code. To reduce the delay, we must increase the peak transmission rate.

Flexcast protocol is based on simple UDP packet communication without error correction and rate control. Thus, if a packet is lost in the network, we just skip an image frame, not re-sending the lost packet. This is a natural way since there may be many receivers working simultaneously in the network, therefore the server cannot wait or re-send only for one receiver's error, In such cases, we usually apply FEC (Forward Error Correction) technology[8]. There are many type of FEC methods according to the property of contents, and sometimes needs tuning adaptively to the real contents. Our programmable application framework will be useful for implementing such error correction techniques in multi-point contents delivery services.

Finally we note that, in the above experiment, we confirmed the basic operations of Flexcast and Java Applet, between the server and the clients located in the same LAN without being relayed by Flexcast splitters. We also conducted additional experiment to insert a Flexcast splitter between the server and clients, and have confirmed that the delivery table in the Flexcast splitter is correctly constructed based on the request packets from the Java Applet program, and delivery packets from the Flexcast server are properly relayed to the client hosts. It means that the Flexcast splitter is working correctly cooperating with Flexcast Applet programs.

5 Conclusion

We presented a new method for multi-point contents delivery applications based on Flexcast and Java Applet. In our method, the service provider does not need to standardize the program, and no need of prior distribution for all prospective users. Even if uncountable many users are targeted, it is anytime possible to update or modify the service, only considering the contents server's schedule. Moreover, the service provider can design and customize the application program freely, not restricted by the interface of commercial software which has a dominant share in the market. We can design the multicast applications with not only simple movie format but also more sophisticated and complex presentations. For example, e-learning technology is recently emerging, and lecture note delivery on the Internet[8][11] is one of the attractive multicast application. Our programmable application framework can flexibly support the various display formats, image size and quality, update timing, combination with voice and text, etc., according to the preference of the lecturer.

As a long-term future work, our method can be applied to the super-parallel large area distributed data processing. In this system, the primitive data is reported to the server using the Flexcast request packets, and Flexcast splitters aggregate the primitive data, and then the Flexcast server feedbacks the computation results to every host by Flexcast delivery packets. Our programmable application framework will be a key technique in such a massively parallel computation system.

References

[1] Stephen Deering: "Multicast routing in internetworks and extended LANs," Proc. of SIGCOMM' 88, Aug. 1988.

[2] "DoJa: Technology for Creating *i-appli*, i-mode Java Applications," `http://www.doja-developer.net/`

[3] Hans Eriksson: "MBONE: The Multicast Backbone," Commun. ACM, vol.37, no. 8, pp.54-60, Aug. 1994.

[4] T. Inoue, S. Tani, K. Ishimaru, S. Minato, and T. Miyazaki, "Wide-Area Multicasting based on Flexcast: Toward the Ubiquitous Network," Proc. of APSITT' 03, Nov. 2003.

[5] T. Inoue, S. Tani, H. Takahashi, S. Minato, T. Miyazaki, and K. Toyoshima, "Design and Implementation of Advanced Multicast Router Based on Cluster Computing," In Proc. of IEEE The 11th International Conference on Parallel and Distributed Systems (ICPADS-2005), July 2005.

[6] "Java 2 SDK, Standard Edition Document," `http://java.sun.com/j2se/1.4.2/docs/`

[7] G. Phillips, S. Shenker, and H. Tangmunarunkit, "Scaling of Multicast Trees: Comments on the Chuang-Sirbu scaling law," Proc. of SIGCOMM'99, August 1999.

[8] H. Takahashi, T. Inoue, H. Thojo, K. Toyoshima, and S. Minato: "A Study on IP Distance Education System Using Flexcast," Technical Report of IEICE, IN2005-50, pp. 127-13, July 2005. (In Japanese)

[9] S. Tani, T. Miyazaki, and N. Takahashi, "Adaptive Stream Multicast Based on IP Unicast and Dynamic Commercial Attachment Mechanism: an Active Network Implementation," Proc. of IWAN2001, Sep. 2001.

[10] S. Tani, T. Inoue, S. Minato, H. Takahashi, S. Kotabe, and T. Miyazaki: "Global Multi-Point Streaming Experiments Based on the Flexcast Protocol," NTT Technical Review, Vol. 1, No. 5, pp. 24–30, Aug. 2003.

[11] P. Ziewer and H. Seidl: "Transparent teleteaching," In Proc. of ASCILITE-2002, Dec. 2002.