Journal of Information Processing Vol.20 No.3 599–606 (July 2012)

[DOI: 10.2197/ipsjjip.20.599]

Regular Paper

Highly Efficient Transport Protocol for Large Capacity Data Files and its details

Tomoo Sumida[1,†1,a)] Yasunori Shiono[1,†1] Takaaki Goto[2,†2]
Katsuyoshi Ito[1,†1] Kensei Tsuchida[1,†1]

Received: September 14, 2011, Accepted: February 3, 2012

Abstract: We describe a protocol named Non-ordered Block Transfer on Explicit Multi-Unicast (NBT on XCAST). The purpose of this protocol is to save network bandwidth and to reduce server load in delivering large-capacity stored files. In general, files are transfered on a unicast basis. However, the unicast wastes bandwidth by sending the same large-capacity files when responding to each client's requests. In addition, server load increases. To solve this problem, Non-ordered Block Transfer (NBT) has been proposed as an efficient transfer system for large-capacity stored files using asynchronous transfer mode (ATM) multicast. However, NBT with ATM has not been implemented and has a limited area of use. In this paper, to solve this problem we propose an NBT protocol using XCAST (NBT on XCAST), which is proposed as a multicast scheme. XCAST can be implemented in a pure IP network without limitation of the data link layer protocol. Our proposed system is especially effective when the server frequently transfers large-capacity stored files. We discuss bandwidth consumption efficiency, a retransmission scheme, and experimental results of a partial performance using NBT on XCAST.

Keywords: network protocol, multicast, XCAST, large-capacity data file

1. Introduction

Internet-related technology has made remarkable progress in the past several years, and Internet users have been increasing.

However, various problems have emerged, two of which are the increase in bandwidth required for delivery of large-capacity files and mismatch of data transfer schemes on the Internet. A movie file with a playback time of about 2 hours, is approximately 5 GB compressed using the MPEG-2 format.

When clients attempt to download this large-capacity data file using Fiber-to-the-Home (FTTH), it may be possible to download it in less than seven minutes. Such high-speed and broadband Internet connections are becoming more common.

However, there is another problem in delivering large-capacity files. For example, if many clients request the same data file simultaneously, a unicast-system server must create as many data copies as the number of clients, and send them to each client. This means that there are a number of packets of the same content on the network. Therefore, the file transfer system based on unicast for large-capacity files has problems of increased server load and of wasted bandwidth used in the transport network at the same time. Solving these problems is one of the most pressing issues for the Internet.

Movies can be provided by streaming video services such as video on demand (VoD). However, VoD requires DVD (Digital Versatile Disc) functions, such as playback, start/stop and slow motion, which give the server a large processing load. The file transfer makes the system simple and flexible, if the file can be transferred in a reasonable amount of time. If an access line service of a gigabit network is available, a 5-gigabit file can be transferred in less than one minute. One minute may be within an acceptable time, and a few Internet service providers are already preparing such services. It is easy to achieve DVD functions at the client terminal after the entire movie file is downloaded.

Multicast is a candidate for improving the defects discussed concerning the unicast file delivery system. A download service has an accounting system, so IP multicast is not applicable because joining or leaving a multicast group is under the control of clients instead of servers. To solve this problem, non-ordered block transfer (NBT) [1] has been proposed as an efficient transfer system for large-capacity files using asynchronous transfer mode (ATM) multicast, but NBT with ATM has not been implemented and has a limited area of use.

To solve these problems, we propose NBT using explicit multi-unicast (XCAST). XCAST has been proposed as an IP multicast protocol [5]. XCAST is used in place of ATM because XCAST can be implemented in a pure IP network without limitation of the data link layer protocol. We call this proposed protocol "NBT on XCAST" [6]. We discuss bandwidth consumption efficiency, a retransmission scheme, and experimental results of a partial performance.

[1] Toyo University, Kawagoe, Saitama 350–8585, Japan
[2] The University of Electro-Communications, Chofu, Tokyo 182–8585, Japan
[†1] Presently with Faculty of Information Sciences and arts
[†2] Presently with Center for Industrial and Governmental Relations
[a)] sumidatomo@gmail.com

2. Previous Work

At present, various protocols have been proposed in order to solve these problems [2]. Also, comparison of the protocol which transmits a file using multicast protocol have been performed [3]. In this section, we describe XCAST protocol and our previous work NBT.

2.1 Unicast

Unicast is communication in which data is sent from a single point to a specified point. In this study, this term means one source, and one destination. Unicast is the dominant form of transmission in Internet service. However, the problem with this unicast method is that it loads the server when many clients request the same data nearly simultaneously, because the server creates the same data to a number of clients [4].

2.2 Multicast

Multicast is communication from which data is sent from one or more points to a group of other points. In this study, this term indicates just one source and many destinations, and the data is distributed to a group of destinations. Multicast is the delivery of data simultaneously in a single transmission from one source creating copies in routers on the network when the topology of the network requires it [4].

2.3 XCAST (Explicit Multi-unicast)

XCAST has been submitted to the Internet Engineering Task Force (IETF) as a new multicast protocol [5].

Today's multicast schemes can be used to minimize bandwidth consumption. Explicit Multi-Unicast (XCAST) also can be used to minimize bandwidth consumption for "small groups," but it has an additional advantage as well. XCAST eliminates the per-session signaling and per-session state information of traditional IP multicast schemes and this enables XCAST to support very large numbers of multicast sessions.

In a multicast model, the packet carries a multicast address as a logical identifier of all group members. In XCAST, the source node keeps track of the destinations in the multicast channel that it wants to send packets to. The source encodes the list of destinations in the XCAST header and then sends the packet to a router. Each router along the way parses the header, partitions the destinations on the basis of each destination's next hop, and forwards a packet with an appropriate XCAST header to each of the next hops. When there is only one destination left, the XCAST packet can be converted into a normal unicast packet, which can be unicasted along the remainder of the route.

2.4 Concept of NBT

Traditionally, files are transferred in unicast. However, this method strains the server. When the transfer of a file is continuously requested by the clients when the transfer in a server has not finished, NBT creates a copy of the file in the point node on a network and sends the file to the clients via multicast rather than transmitting a copy of the file individually to the clients from the server. This results in the server processing load migrating and minimizing network bandwidth consumption.

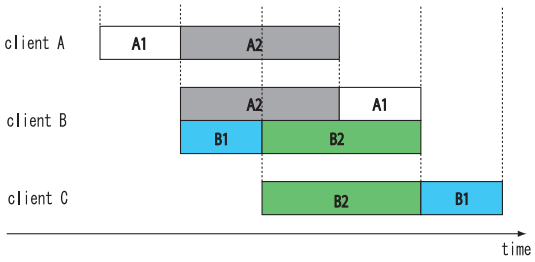

Fig. 1 Concept of NBT.

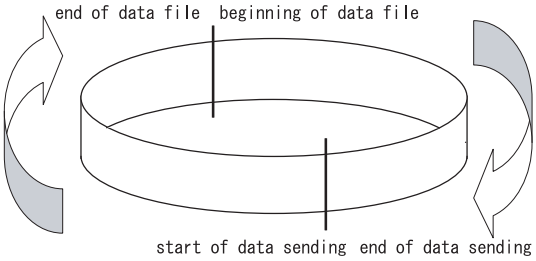

Fig. 2 Sending method with NBT.

While a server is sending a data file to client A, client B requests the same data file, and client C also requests the same data file, as shown in **Fig. 1**. The data file consists of portion A1 and A2. A copy of data portion A2 under transfer to client A is created at a suitable branching node on the course of transfer and is sent to client B. Data portion A1 is sent to client B from the server after data portion A2. For client C, a data file is sent in a similar procedure in the sequence B2, B1.

In NBT, when a server cyclically sends a data file, a client receives the data from the midpoint of the data file instead of the beginning (see **Fig. 2**).

3. Proposed Protocol

In this section, we describe the concept of our proposed protocol NBT on XCAST.

3.1 NBT on XCAST

XCAST has been submitted to the Internet Engineer Task Force (IETF) as a new multicast protocol [5]. This protocol uses ATM in a pure IP network instead, and the system control is under the server. The advantages of using XCAST for implementing NBT are as follows:

- Independence from the data link layer: Network domains have no restrictions.
- XCAST operates under server control, because the server keeps all client information, such as client addresses. This attribute means that it is easy to acquire information from each client, such as charging information.
- Since the multicast address and the multicast routing protocol are not required, and installation of the XCAST protocol is required in only a limited number of routers, XCAST protocol is easy to implement.
- There is no critical node. XCAST minimizes the network latency and maximizes efficiency.

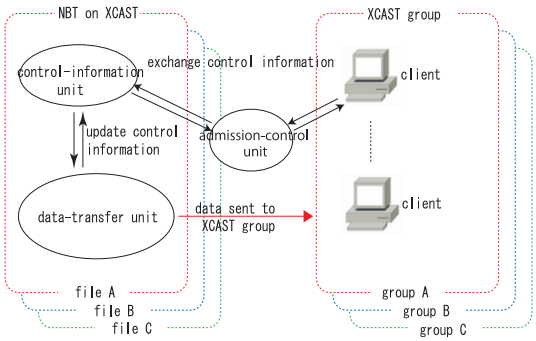

Fig. 3 System configuration of NBT on XCAST.

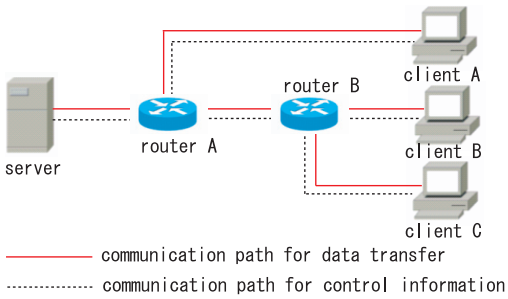

Fig. 4 Communication path configuration of NBT on XCAST.

3.2 System Configuration

Figure 3 shows the system configuration of NBT on XCAST.

The transfer request of client 1, which is the first client of the file, is received at the admission control of NBT. At the admission control unit, the content of the client requests is identified. Then, the connection is extended to the control information unit corresponding to the identified content. At the control information unit, necessary control information about NBT is generated and sent to client 1 through the admission control unit. The control information unit then triggers the data transfer unit as a thread. The data transfer unit sends the data file the client requested on the basis of the control information. Since the control information required for NBT continuously changes, the control information unit and the data transfer unit always update and exchange the control information.

The server-control information that requires updating is as follows.

- Content Type.
  This control information needs to identify the data that clients require.
- Client number.
  This control information needs to manage the data required by the client.
- The send-start data block number to each client.
  When more than one client requests a large volume of data, the data need to be repeatedly sent.
- The send-finish data block number to each client.
  When more than one client requests a large volume of data, the data need to be repeatedly sent.
- The data block number to be presently sent.
  The parameter needed for each client.
- The next data block number to be sent.
  This parameter is used in lost-packet detection in the client.
- The data size sent to each client.
  All data sizes should be sent. This parameter needs to check whether each client has received all data.

The client-control information requires updating.

- Receive start point and end point.
  The reception start point and end point are different for each client.
- The data point that is currently being received.
  The data point the client has received. It is needed with the lost packet detection.
- The data point that will be received next.
  The estimated data point the client receives next. It is needed with the lost packet detection.
- All data sizes.
  All data are received. They are used in lost packet detection with the client.

The second client and following clients are connected to the admission-control unit like the first client, and the requested content is identified. Then, after communication with the corresponding control information unit connecting the content and acquiring the most recent control information, they join in an XCAST group and receive the data file requested from the data-transfer unit already activated. The end positions for each client differ and are determined on the basis of control information between the server and each client.

Thus, several clients that requested the same content are brought together into a single XCAST group, and a corresponding XCAST group will be set up in accordance with each content.

3.3 Communication Path Configuration

Figure 4 shows the communication path configuration of NBT on XCAST. In NBT on XCAST, the communication path of NBT transfer and control information data are separated, as mentioned in the preceding section, that is the communication path for data transfer (XCAST/UDP) and the communication path for control information (TCP). By preparing two communication paths, design and control of the whole system become flexible.

Each client requests server control data using the communication path for the control information. The server sends the necessary information for using NBT to a client using the communication path for the control information. The necessary information includes present client situation, present data delivery situation, etc. Next, the server sends data to each client using the communication path for data transfer on the basis of a control-information management table prepared for each client.

3.4 Retransmission Scheme

When packet loss occurs due to various causes such as congestion and bit error, a scheme on how to retransmit the lost data is considered.

The retransmission scheme for NBT on XCAST has two kinds of retransmission methods and retransmission timings.

Retransmission method

a) The block containing the lost packet for every client is retransmitted by unicast.

Table 1 Combinations of retransmission method and timing in NBT on XCAST.

method	timing	
	retrans.request reception	bulk block
Unicast	(a), (c)	(a), (d)
NBT on XCAST	(b), (c)	(b), (d)

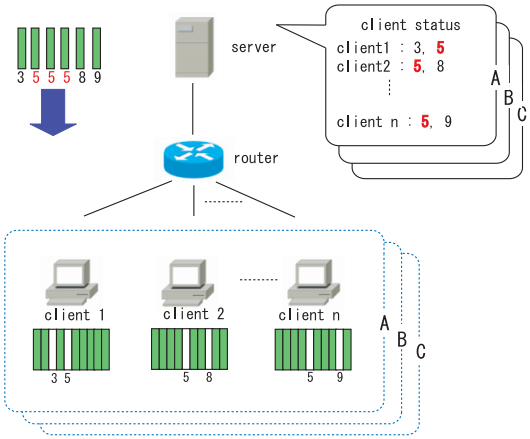

Fig. 5 Retransmission to each client.

b) The block containing the lost packet is retransmitted by XCAST to the XCAST group to which the client that requested the data belongs.

Retransmission timing

c) When the server receives the retransmission request of a block from each client, it retransmits to the individual clients.

d) The server retransmits the block under retransmission request, after transfer of some portion of the data is finished.

Retransmission in NBT on XCAST can be classified into $2 \times 2 = 4$ kinds of combinations of the retransmission method and retransmission timing. The combinations are listed in **Table 1**.

Next, we discuss these combinations. Also, XCAST retransmits data to every client. Images of data being resent to the group are shown, too.

Retransmission to each client

Figure 5 shows the retransmission scheme to each client. When this scheme is adopted, the server must manage the data of the information sent from all clients individually. Moreover, since the same content blocks may have to be sent to several clients, this is not efficient or flexible. A large processing load may be incurred to the server. When adopting this scheme, it is appropriate to retransmit through the control information path at time *c*.

Retransmission to XCAST group

Figure 6 shows the retransmission scheme to an XCAST group. When retransmitting after building an XCAST group, it may take some time after a client requests retransmission to actually receive retransmitted data. However, since there is only one retransmitting data block for all clients for each retransmission request, the processing load of the server is mitigated.

For example, in Fig. 5, the server should create the same number of packets as the number of requested packets (5).

Retransmission is performed through the XCAST path at time *d*.

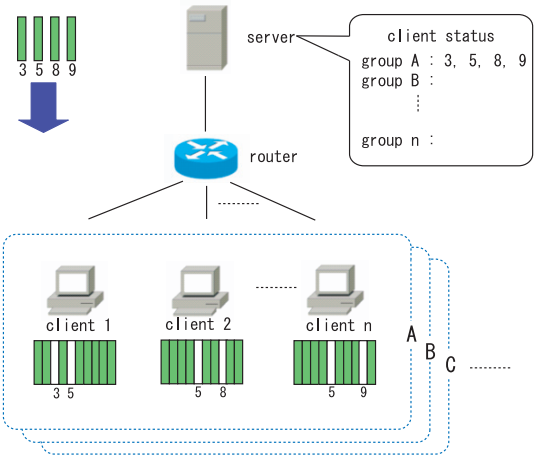

Fig. 6 Retransmission to XCAST group.

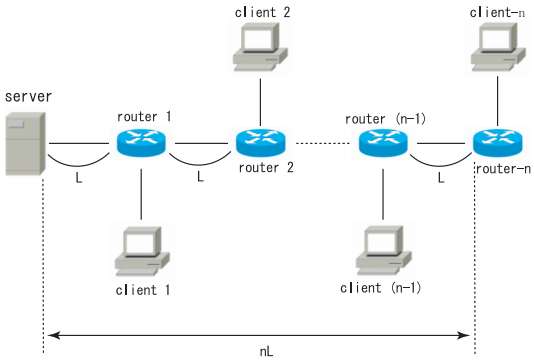

Fig. 7 Example of efficiency.

From the above consideration, retransmission through an XCAST communication path to the XCAST group at the time of a file-transfer break seems best for retransmitting the lost data in NBT on XCAST.

4. Theoretic Analysis for the Proposed Protocol

In this section, we describe the theoretic analysis for the NBT on XCAST.

4.1 Efficient Bandwidth Utilization of NBT

A simplified network model using NBT is shown in **Fig. 7** [7]. This network model consists of n clients separately connected by different routers. The routers relay data with a cascading transmission distance of L from the server, and each client is connected to each router. The index indicates network bandwidth-utilization efficiency, the bandwidth-distance product A, which is the sum of the product of the bandwidth B required to transfer the requested file to each client, the transmission distance, and the transfer time T.

(1) Unicast Scheme

In a traditional unicast scheme, a server creates the data requested by each client and sends the created data to those clients. In addition, there is more than one packet of identical contents on the transit-network, which increases the load of the server, and decreases network efficiency.

For this reason, the bandwidth distance product for the unicast scheme A_u (colored area) is shown in **Fig. 8**.

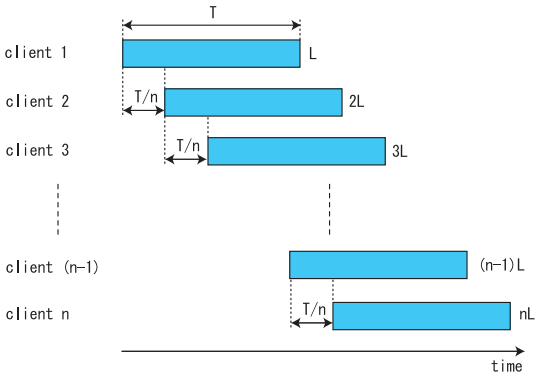

Fig. 8 Unicast efficiency.

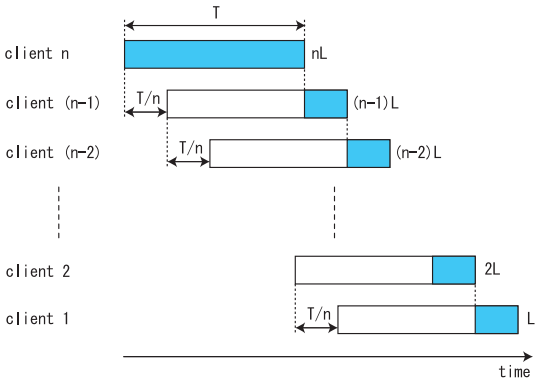

Fig. 9 From client n to client 1.

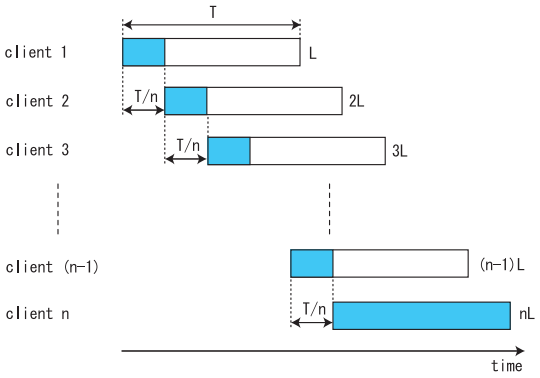

Fig. 10 From client 1 to client n.

$$A_u = \sum_{i=1}^{n} iBTL = \frac{BTL}{2} n(n+1).$$

The bandwidth-distance product is proportional to the square of the number of clients n, and it is not scalable.

(2) NBT Scheme

The clients, in turn, request the server to download at interval T/n the farthest from the server to the nearest (see **Fig. 9**) or *vice versa* (see **Fig. 10**).

Case 1: from client n to client 1

It is assumed that the data is transferred, in turn, from the farthest client to the nearest one. Figure 9 shows how to compute, in turn, the bandwidth-distance product (colored area) at interval T/n from the farthest from the server to the nearest.

$$A_n = BT \cdot nL + B \cdot \frac{T}{n}(n-1)L + \cdots + B \cdot \frac{T}{n}L$$
$$= nBTL + \frac{BTL}{n} \sum_{i=1}^{n-1} i \quad (1)$$
$$= BTL\left(n + \frac{n-1}{2}\right)$$
$$= \frac{BTL}{2}(3n-1)$$

From the above expression, the bandwidth-distance product A_n using NBT on XCAST is proportional to the number of clients n.

Case 2: from client 1 to client n

It is assumed that the data is transferred, in turn, from the nearest client to the farthest one. Figure 10 shows how to compute the bandwidth-distance product, in turn, (colored area) at interval T/n from the nearest to the server to the farthest from it.

$$A_n = \frac{BT}{n} \cdot L + B\frac{T}{n} \cdot 2L + B\frac{T}{n} \cdot 3L + \cdots + B\frac{T}{n} \cdot (n-1) + BT \cdot nL$$
$$= \frac{BTL}{n} \sum_{i=1}^{n-1} i + BT \cdot nL \quad (2)$$
$$= \frac{BTL}{2}(3n-1)$$

From the above expression, the bandwidth-distance product A_n using NBT on XCAST is proportional to the number of clients n.

Because Eq. (1) is equal to Eq. (2), the bandwidth-distance product using NBT on XCAST is proportional to the number of clients without reference to the order. Additionally, compared with the unicast method, the bandwidth-distance product using NBT on XCAST holds well.

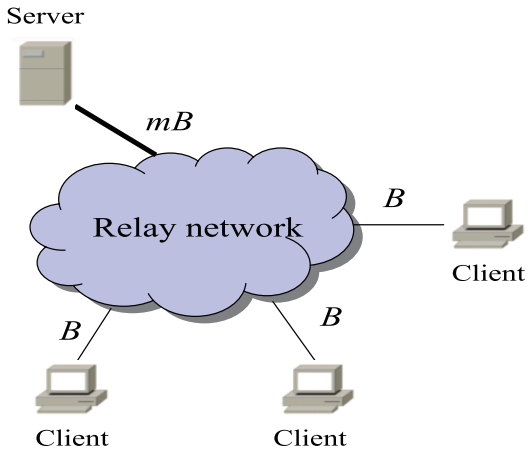

Fig. 11 Network topology for transfer capability analysis of NBT on XCAST.

4.2 Transfer Capability of NBT

The network topology assumed for computing the transfer capability of NBT on XCAST is shown in **Fig. 11**.

If the file size and the bandwidth are G and B, respectively, it is assumed that the number of clients that can be transferred in an XCAST group is $\bar{n}$, and that a server has a bandwidth of the

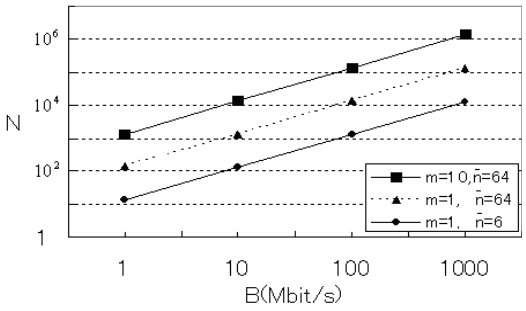

Fig. 12 Transfer capability of NBT on XCAST.

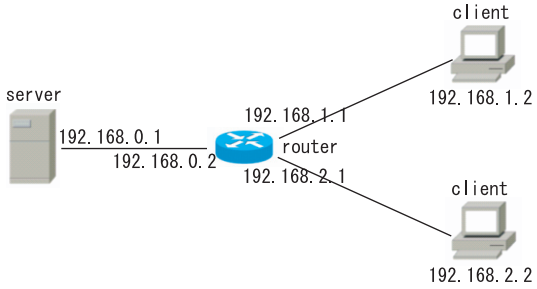

Fig. 13 Experimental network topology.

access network mB. The total transfer capability per unit time of NBT N_h is given by

$$N_h = m\bar{n}/T = m\bar{n}B/G$$

The transfer capability N_h is shown in **Fig. 12** ($G = 5$ GB is assumed).

When the following values, $m = 10$, $\bar{n} = 64$, $B = 1$ GBit/s, and $G = 5$ GB are assumed, the transfer capability per day N is given by

$$N = 1.4 \times 10^6$$

The results indicate that NBT is practical for distribution movies, if an access network of 1 GBit/s is assumed to be available. The maximum allowable number of destination addresses of XCAST is 64.

5. Partial Operation Experimentation

In this section we discuss the results from partial operation on the basis of the design described in the previous section. The experimental network topology is shown in **Fig. 13**.

Two clients request transfer of large-capacity data in the server side at different moments within the data-transfer time. In this experiment, the number of contents in the server is one.

5.1 Server Behavior

The logfile, which shows the behavior of the server with the network topology in Fig. 13, is shown in **Figs. 14** and **15**.

Figure 14 shows the logfile when NBT on XCAST has just started. While the server is performing an update of a route using the XCAST routing daemon, client 1 (192.168.1.2) is requesting the data. The behavior outline is as follows.

- Line 00
  Routing update packet is sent.
- Line 01
  Routing update packet is received.

```
00 15:47:19 Sending route update to 192.168.0.2
01 15:47:25 Received route update from 192.168.0.2
02 15:47:29 Received XCAST packet from 192.168.0.1
03 15:47:29 Num destinations = 1
04 15:47:29 Unicast packet sent to 192.168.1.2/port 1058
05 15:47:30 Received XCAST packet from 192.168.0.1
06 15:47:30 Num destinations = 1
07 15:47:30 Unicast packet sent to 192.168.1.2/port 1058
08 15:47:31 Received XCAST packet from 192.168.0.1
09 15:47:31 Num destinations = 1
10 15:47:31 Unicast packet sent to 192.168.1.2/port 1058
11 15:47:32 Received XCAST packet from 192.168.0.1
12 15:47:32 Num destinations = 1
13 15:47:32 Unicast packet sent to 192.168.1.2/port 1058
```

Fig. 14 Server behavior based on transfer demand of client 1.

```
00 15:47:38 Received XCAST packet from 192.168.0.1
01 15:47:38 Num destinations = 1
02 15:47:38 Unicast packet sent to 192.168.1.2 port 1058
03 15:47:38 Received XCAST packet from 192.168.0.1
04 15:47:39 Num destinations = 1
05 15:47:38 Unicast packet sent to 192.168.1.2 port 1058
06 15:47:40 Received XCAST packet from 192.168.0.1
07 15:47:40 Num destinations = 1
08 15:47:40 Unicast packet sent to 192.168.1.2 port 1058
09 15:47:40 Received XCAST packet from 192.168.0.1
10 15:47:40 Num destinations = 2
11 15:47:40 Dests: 192.168.1.2 port 1058 192.168.2.2 port 1058
12 15:47:40 XCAST packet sent to 192.168.0.2
13 15:47:41 Received XCAST packet from 192.168.0.1
14 15:47:41 Num destinations = 2
15 15:47:41 Dests: 192.168.1.2 port 1058 192.168.2.2 port 1058
16 15:47:41 XCAST packet sent to 192.168.0.2
17 15:47:41 Received XCAST packet from 192.168.0.1
18 15:47:41 Num destinations = 2
19 15:47:41 Dests: 192.168.1.2 port 1058 192.168.2.2 port 1058
20 15:47:41 XCAST packet sent to 192.168.0.2
21 15:47:42 Received XCAST packet from 192.168.0.1
22 15:47:42 Num destinations = 2
23 15:47:42 Dests: 192.168.1.2 port 1058 192.168.2.2 port 1058
24 15:47:42 XCAST packet sent to 192.168.0.2
```

Fig. 15 Change in behavior of server due to data-transfer request of client 2.

- Line 02
  Server interface receives the XCAST packet.
- Line 03 and 04
  These lines show the number of clients of the XCAST packet, as a result of analyzing the header of the received XCAST packet. Since the number of clients is still one, the server sends a unicast packet to the client, as shown in line 04, in accordance with the rule of XCAST.
- hereafter
  The same process is repeated.

Figure 15 is a logfile that shows the behavior of the server when client 2 (192.168.2.2) requested the same data as client 1, while sending the data to client 1.

- Line 00 ~ 08
  These lines show the unicast packets where the only client is client 1 as a result of analyzing the header of the received XCAST packet. Since the number of clients is still one, the server sends a unicast packet to the client.

Lines 09 and above show that the second client requests the data, which the first client has been receiving, and the second client joins the XCAST group.

```
00 16:14:14 Received route update from 192.168.0.1
01 16:14:20 Sending route update to 192.168.0.1
02 16:14:35 Received XCAST packet from 192.168.0.1
03 16:14:35 Num destinations = 2
04 16:14:35 Unicast packet sent to 192.168.1.2 port 1058
05 16:14:35 Unicast packet sent to 192.168.2.2 port 1058
06 16:14:36 Received XCAST packet from 192.168.0.1
07 16:14:36 Num destinations = 2
08 16:14:36 Unicast packet sent to 192.168.1.2 port 1058
09 16:14:36 Unicast packet sent to 192.168.2.2 port 1058
10 16:14:36 Received XCAST packet from 192.168.0.1
11 16:14:36 Num destinations = 2
12 16:14:36 Unicast packet sent to 192.168.1.2 port 1058
13 16:14:36 Unicast packet sent to 192.168.2.2 port 1058
14 16:14:37 Received XCAST packet from 192.168.0.1
15 16:14:37 Num destinations = 2
16 16:14:37 Unicast packet sent to 192.168.1.2 port 1058
17 16:14:37 Unicast packet sent to 192.168.2.2 port 1058
18 16:14:37 Received XCAST packet from 192.168.0.1
19 16:14:37 Num destinations = 2
20 16:14:37 Unicast packet sent to 192.168.1.2 port 1058
21 16:14:37 Unicast packet sent to 192.168.2.2 port 1058
```

Fig. 16 Instrumentation of router.

- Line 09
  The server interface receives the XCAST packet.
- Line 10
  The header of the received XCAST packet is analyzed. This XCAST packet has two clients.
- Line 11
  This line shows the address of the clients contained in the address list of the received XCAST packet. This means that several client addresses are explicitly described in the XCAST header, which is one of the features of XCAST.
- Line12
  This line shows that the XCAST packet to the neighboring XCAST-correspondence router is created and sent.

This results of implementation can be thought to resolve the multicast calling issue.

5.2 Router Behavior

Figure 16 is a logfile that shows the behavior of the relay router. When the relay router receives an XCAST packet from the preceding router ("the server" in this experiment), it analyzes the addresses in the XCAST header and duplicates a packet to each address in XCAST and unicast packets. In this experiment, unicast packets are transferred to each client.

6. Conclusion

This paper introduced the concept of the NBT and discussed its potential capacity. NBT is a protocol for distributing a large-capacity file to multi clients with minimal network bandwidth consumption. We also proposed non-ordered block transfer using NBT on XCAST to expand flexibility, with consideration of application in pure IP networks. Finally, a partial behavior experiment of the proposed NBT on XCAST was conducted. This results of the implementation can be thought to resolve the multicast calling issue [8].

Now, we are implementing the algorithm for visualizing NBT on XCAST protocol. A simulation must be preformed to evaluate and review the new protocol. Therefore, we propose a method for visualizing NBT on XCAST protocol [9]. The purpose of this approach is to understand the status of the whole network system with NBT on XCAST. In this visualization method, vertices represent clients and edges represent routers and communication lines.

Full implementation, including error recovery by retransmission of data, will be performed in future work. As for the implementation of the proposed protocol, the confirmation of the basic performance has ended; however, the retransmission control, which requires file transfer, still needs to be unimplemented.

References

[1] Takatori, M., Ito, K. and Takasaki, Y.: Improvement of network usage efficiency by non-ordered block transfer, *Proc. 59th IPSJ National Conference* (1999).

[2] Roca, V. and Adamson, B.: FCAST: Scalable object delivery for the ALC and NORM protocols, draft-ietf-rmt-fcast-03 (Feb. 2011).

[3] Manazanares-Lopez, P. and Sanchez-Aarnoutse, J.C.: Analysis and performance evaluation of a multicast file transfer solution for congested asymmetric networks, *NETWORKING 2006*, *LNCS 3976*, Boavida, F. et al. (Eds.), pp.703–714 (2006).

[4] Tanenbaum, A.S. and Wetherall, D.J.: Computer Networks 5th, Prentice Hall (2010).

[5] Boivie, R. et al.: Explicit Multicast (Xcast) Concepts and Options, RFC5058 (2007).

[6] Sumida, T. and Ito, K.: Implementation of the Non-ordered Block Transfer Utilizing XCAST, *Proc. 2004 IEICE General Conference*, B-7-40 (2004).

[7] Sumida, T. and Ito, K.: Effectiveness of the Non-ordered Block Transfer Utilizing XCAST, *Proc. 2003 IEICE Society Conference*, B-7-3 (2003).

[8] Sumida, T., Goto, T., Ito, K. and Tsuchida, K.: Highly Efficient Transport Protocol for Large Capacity Data Files and its Visualizing Methods, *Proc. 9th IEEE/ACIS International Conference on Computer and Information Science* (*ICIS2010*), pp.389–394 (2010).

[9] Sumida, T., Tsuchida, K. and Ito, K.: Visualization of Highly Efficient Transport Protocol for Large Capacity Files, *Trans. Information Processing Society of Japan*, Vol.49, No.2, pp.544–554 (2008).

Tomoo Sumida graduated with a Doctor of Engineering from Toyo University in 2009. He has been a Part-time Lecturer of Toyo University since 2009. His main research interests are network protocol, graph visualization, and software development. He is a member of IPSJ, IEICE Japan and IEEE.

Yasunori Shiono received his M.E. and Dr.Eng. degrees from Toyo University in 2006 and 2010 respectively. He is currently an Assistant Professor of Faculty of Information Sciences and Arts at Toyo University. His research interests include graph algorithms, graph grammars, fuzzy theory and software development environments. He is a member of IEICE Japan, JSSST, JSIAM and IEEE.

Takaaki Goto graduated with a Doctor of Engineering from Toyo University in 2009. He has been a Project Assistant Professor at the Center for Industrial and Governmental Relations of the University of Electro-Communications since 2009. His main research interests are applications of graph grammars, visual languages, and software development environments.

Katsuyoshi Ito received his B.S. degree from the University of Tokyo in 1962 and the Ph.D. degree from Tohoku University in 1980, both in electrical engineering. From 1962 to 1998, he was with Mitsubishi Electric Corporation. He was involved in R&D of Q-switched solid-state lasers, laser radars, optical gas densitometers, and infrared imaging devices. Since 1979, he was involved in R&D of optical communication systems for both public and local area networks. He was with Toyo University from 1998 to 2008 and had done research on high-speed computer network protocols as a Professor of the Department of Information and Computer Sciences. Dr. Ito is a member of IEEE and IEICE Japan.

Kensei Tsuchida received his M.S. and D.S. degrees in mathematics from Waseda University in 1984 and 1994 respectively. He was a member of the Software Engineering Development Laboratory, NEC Corporation in 1984–1990. From 1990 to 1992, he was a Research Associate of the Department of Industrial Engineering and Management at Kanagawa University. In 1992 he joined Toyo University, where he was an Instructor until 1995 and an Associate Professor from 1995 to 2002 and a Professor from 2002 to 2009 at the Department of Information and Computer Sciences and since 2009 he has been a Professor of Faculty of Information Sciences and Arts. He was a Visiting Associate Professor of the Department of Computer Science at Oregon State University from 1997 to 1998. His research interests include software visualization, human interface, graph languages, and graph algorithms. He is a member of IPSJ, IEICE Japan and IEEE Computer Society.