ECTP for Interactive Multicast Transport: Design and Implementations[+]

Jae-Sung Park[1], Sang-Tae Kim[1], Seok J. Koh[1], Yong-Jin Kim[2]
[1]Kyungpook National University, [2]Modacom Incorporation
{js.park06, mr.stkim}@gmail.com, sjkoh@cs.knu.ac.kr, cap@modacom.co.kr

***Abstract* — The Enhanced Communications Transport Protocol (ECTP), which has been standardized in the ISO/IEC JTC1/SC6, is an end-to-end multicast transport protocol that is designed to support an interactive Internet multicast applications with unicast backward channels to the multicast sender. This paper describes the design, implementation and experimental results for the ECTP protocol.**

***Keywords* — ECTP, interactive, multicast, protocol.**

1. Introduction

The Enhanced Communications Transport Protocol (ECTP) is a transport protocol designed to support Internet applications running over multicast-capable networks, which has been standardized in the ISO/IEC JTC1/SC6 [1]. The protocol specification of the ECTP consists of the six parts: ECTP-1 through ECTP-6 [2].

This paper describes the ECTP-3 (ECTP part 3). The ECTP-3 specifies the specification of the protocol for the interactive multicast transport in which some multicast receivers are allowed to send unicast backward channels to the multicast sender, as shown in the example of the remote education application. We will describe the design, implementation and experimental results for ECTP-3.

This paper is organized as follows. In Section 2, the protocol overview of the ECTP-3 is presented. Section 3 describes the implementation details of ECTP-3. Section 4 describes the experimentation results for an interactive application using the ECTP-3. Finally, Section 5 concludes this paper.

2. Protocol Overview

The ECTP-3 is also called the duplex multicast transport protocol [1]. The ECTP-3 specifies the protocol mechanisms for reliable data transport and tight control of the multicast connection. An ECTP-3 connection will consist of a single TC-Owner (TO) as a multicast sender and many TS-Users (TUs) as multicast receivers. Some of the TUs will be allowed for the unicast data transport to the TO, which are called Sending User (SU). An ECTP-3 duplex multicast connection will be used for supporting multicast data transport between the participants that are classified into a single TO and the other TUs.

In ECTP-3, TO is at the heart of the multicast group communications. It is responsible for overall connection management by governing the connection creation and termination, the user's join and leave operations. TUs will receive multicast data from the TO.

The two types of data channels in ECTP-3 are used: multicast data channel and unicast data channel. TO transmits the multicast data using the multicast data channel to all the other TUs. Some SUs will be allowed for transmitting the unicast data to the TO using the unicast data channel. It is noted that each SU must get a token from the TO before sending the unicast data.

To establish a duplex multicast connection, TO transmits a Connection Creation Request (CR) packet to the group. The CR packet contains the connection-specific information including general characteristics of the connection. In particular, the CR packet must indicate that the connection type is the duplex multicast transport. Each TS-user who wants to participate in the connection will respond to the TO with a Connection Creation Confirm (CC) packet. The connection creation operation will be completed when a pre-determined CCT timer expires.

During the connection creation phase, a logical control tree is configured between TO and TS-users, or between different TS-users, for providing the scalable reliability control. With the root of the TO, the control tree defines a parent-child relationship between any pair of two TS-users. The parent TS-user is called Local Owner (LO). Based on the control tree, the error recovery will be performed. To configure a control tree, each TS-user sends a Tree Join Request (TJ) message to a candidate parent node that has already been connected to the tree. The parent node will respond to the promising child TS-user with the Tree Join Confirm (TC) message. In this way, the control tree will gradually be expanded from the root toward the leaf nodes.

Some of the prospective TS-users may join the connection as late-joiners. The late-joining TS-user participates in the connection by sending a Late Join Request (JR) message to TO. In response to the JR message, TO sends a Late Join Confirm (JC) message to the TS-user. The late-joiner TU will also join the control tree by using the TJ and TC messages. For this purpose, the JC message of TO may include the information about the prospective parent LO node for the

[+] This research was supported by the MIC of Korea, under the ITRC support program supervised by the IITA (IITA-2006-C1090-0603-0026)

late-joiner. The late-joining TS-user may try to connect to the prospective LO node so as to configure the control tree.

After the connection is established, the data transmission phase starts. ECTP-3 protocol supports two types of data channels: the forward multicast channel from TO to the group and the backward unicast channel from the TS-user to TO. The ECTP-3 provides the reliable data transport with error recovery, in which all the Data (DT) packets will be recovered by the parent on the tree.

In the forward multicast data transmission, TO can begin the multicast data transmission to the group by using the IP multicast address and group port number. The multicast data packets sent by TO will be sequentially segmented and transmitted by DT packets to the receiving TS-users. The TS-users will deliver the received DT packets to the upper-layer application in the order transmitted by TO.

For the forward multicast data channel of TO, the error control will be performed based on the local group defined by the ECTP control tree. If a child node detects a data loss, it sends a retransmission request to its parent via ACK packets. Each child generates an ACK packet every ACK Generation Number (AGN) data packets. That is, an ACK packet is generated for the AGN data packets of TO. An ACK message contains a 'bitmap' to indicate which data packets have been received or not. In response to an ACK packet, each parent LO may retransmit the RD packets to its children.

In the forward multicast data transport, the Heartbeat (HB) and Heartbeat ACK (HBACK) packets are used between a parent and children for the control tree maintenance. A parent transmits HB packets to the children every HB Generation Time (HGT). The HB contains which child must respond to this HB packet with the HBACK packet. The corresponding child will send a HBACK packet to the parent. The HB packet may also be used by the parent to calculate the local Round Trip Time (RTT) for the group. For this purpose, the HB and HBACK packets contain a timestamp.

For the backward unicast data transport, a certain TS-user in the connection may get a token from TO by sending a Token Get Request (TGR) message. The TO will then respond to the TS-user with the Token Get Confirm (TGC) message that contains a Token ID. Accordingly, the total number of tokens in the connection is controlled by TO. Token ID is used to identify the sender of the unicast DT packets at the TO side. The TS-user who has a token is called Sending User (SU).

The SU can send unicast DT packets to TO. For the error recovery and congestion control, the HB and HBACK packets are exchanged between SU and TO. The SU sends an HB message to TO. The TO then responds with the HBACK packet that contains the acknowledgement information, as done in ACK packets in the forward multicast channel. It is noted that the HBACK is used for retransmission request in the backward channel.

After completing the unicast data transmission, the SU will return the token to the TO by sending a Token Return Request (TRR) message. TO will respond to the SU with a Token Return Confirm (TRC) message.

The connection management operations are taken in the connection; user leave, the connection pause and resumption, and connection termination. In the User Leave operation, a participating TS-user may leave the connection by sending a User Leave Request (LR) message to the parent. In a certain case, the parent may enforce a specific child node to leave the connection by sending the LR message, which is called the troublemaker ejection. The TO may temporarily pause and resume the connection. In the connection pause period, the TO will send Null Data (ND) packets to the group. After the TO has completed the data transport, it may terminate the duplex connection by sending a Connection Termination Request (CT) message to the group.

3. Implementation

3.1 Application Programming Interface for ECTP-3

The ECTP-3 API consists of 3 parts. There are used by TO and/or TU.

The APIs for ECTP-3 used by TO and TU are as follows:

- `msocket`,
- `mleave`,
- `mrecvmsg`,
- `msetsockopt`,
- `mgetsockopt`.

APIs for application of TO include the following functions:

- `mcreate`,
- `msendm`,
- `mterminate`,
- `mjoin_confirm`,
- `mtoken_confirm.`

On the other hand, the APIs for application of TU include the following functions:

- `mjoin`,
- `msendmsg`,
- `mtoken_get`,
- `mtoken_return.`

The ECTP-3 API sequence may be different for the TO and TU (SU) applications. That is, each application will call the API functions in a different sequence, depending on the node type. It is noted that the TO will begin a connection by calling the `msetsockopt` function.

Figure 1 show an example of the ECTP-3 API sequences at the TO side. As shown in the figure, the TO will start the connection by calling the `msocket` and `mcreate`, together with `msockopt`, if necessary. After starting the connection, the TO may begin to send the multicast data packets to the TUs. In the meantime, the TU will also be ready to receive some packets from the TUs. To deal with this packets, the TO needs to run a set of 'event' loop functions, which are used to handle the events of the 'user join', 'token request' and 'backward data channels', as illustrated in the figure.

The TO may stop the connection after completion of the forward multicast data transmissions.

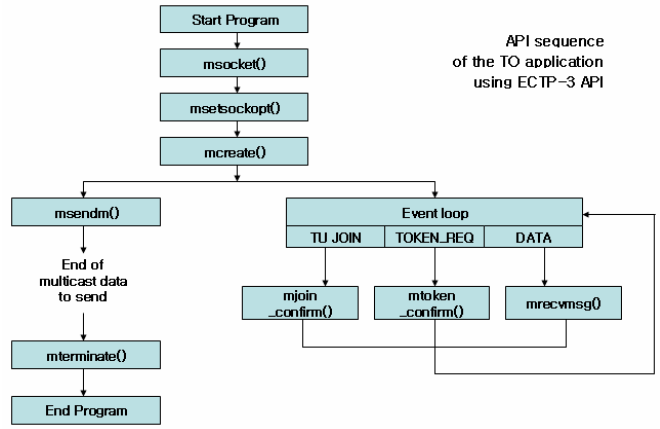

Figure 1. API Sequence of the TO application using ECTP-3 API

Figure 2 show an example of the ECTP-3 API sequences at the TU side. As shown in the figure, the TU will join the connection by calling the `msocket` and `mjoin`.

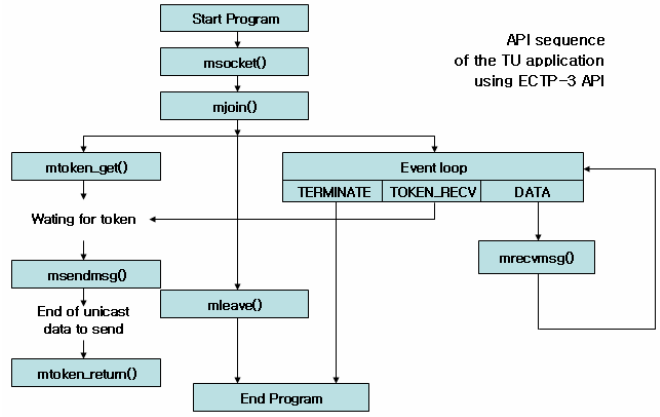

Figure 2. API Sequence of the TU application using ECTP-3 API

After joining the connection, the TO may begin to send the unicast data packets to the TO, after it gets a token from the TO. In the meantime, the TU will also be ready to receive some packets from the TOs. To deal with this packets, the TU needs to run a set of 'event' loop functions, which are used to handle the events of the 'reception of forward multicast data', 'token-related control packets' and 'connection termination' as illustrated in the figure. The TU may stop the connection by the user's leave operation.

3.2 State Transition Diagrams

The State Transition Diagrams for ECTP-3 are described in Figure 3 and 4. It is noted that the two state transition diagrams are different, since the TO performs the connection creation process, whereas the TU has the user's join process.
Both TO and TU will execute some general process in the 'ESTABLISHED' state. General process means multicast data sending and receiving. After that, the ECTP-3 protocol states will be changed to the other ones, when a specific event occurs or some packets are received.

For example, when the TO received a TGR packet from a TU, then the TO goes into the 'TGR-RCVD' state, so as to decide whether the token request should be accepted or not. The state 'TGR-RCVD' means that the TO is in the process of the relevant decision.

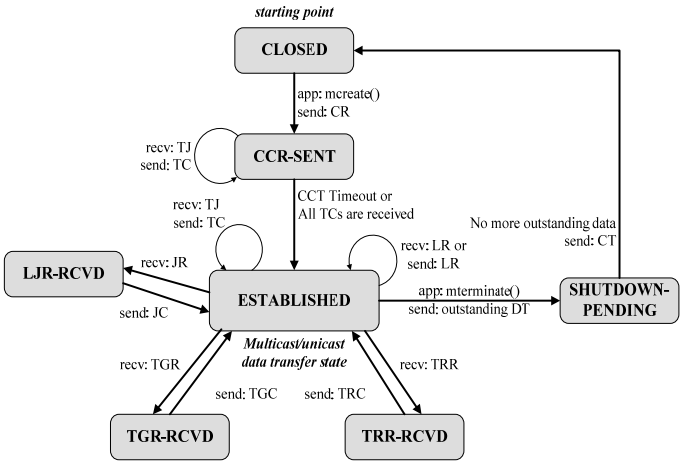

Figure 3. State Transition Diagram for TC-Owner

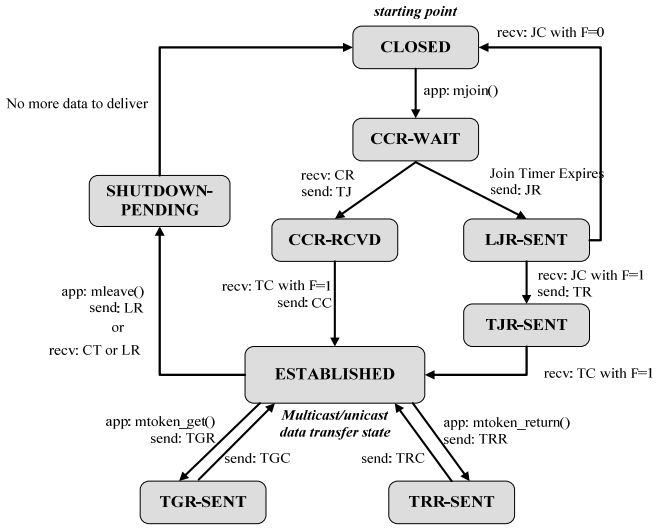

Figure 4. State Transition Diagram for TS-User

3.3 Details of Implementations

The following figure shows an abstract implementation structure of the ECTP-3.

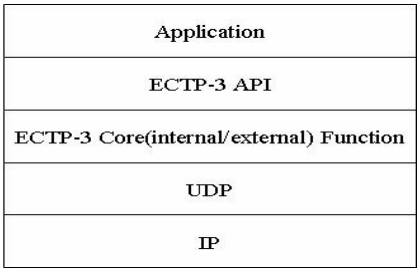

Figure 5. ECTP-3 Implementation Structure

The implementation functions for ECTP-3 are divided into the following 4 parts:

1) Socket APIs for ECTP-3
2) ECTP-3 Protocol Core (Internal Functions)
3) Event Callback Functions
   (to handle the control events defined in the ECTP-3)
4) Timer Functions (to handle the various ECTP-3 timers).

The ECTP-3 API functions are used to develop the ECTP-3 application. Internal functions are used to support the ECTP-3 core processing. Event callback functions are used to support the event handling. Timer functions are used to support timer of ECTP-3.

The following figure shows the structural diagram of the ECTP-3 (internal) functions.

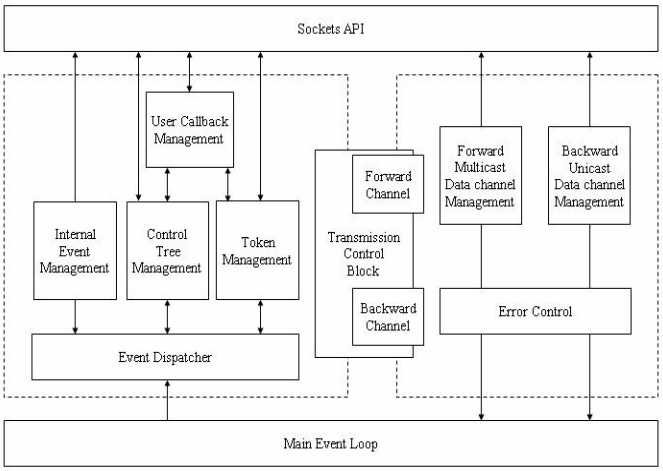

Figure 6. ECTP-3 Implementation: Block Diagram of Functions

In the figure, Sockets API represents is the ECTP-3 Sockets API that can be used by the upper layer applications. The Main Event Loop is the event handler for the lower layer such as the UDP/IP. There are the 3 sub-parts between Sockets API and Main Event Loop.

The left part is the group of control operations for ECTP-3. The Event Dispatcher is used to get an item from the relevant queues. The Internal Event Management is used to manage an internal event. For example, ACK is generated by the AGN value. The Control Tree Management is used to configure and manage the tree of ECTP-3. The Token Management is used to manage a token for backward data channel in ECTP-3, for example, to handle the TGR or TRR messages. The User Callback Management function gives an interface between Sockets API and Control Tree Management, Token Management.

The right part is the group of functions for data transmission. Data transmission will be done with the help of the forward channels and backward channel management. The Error control function is used to manage the error control for the data transmission.

The center part represents the Transmission Control Block (TCB). The TCB is used to support the control and data operations of the ECTP-3.

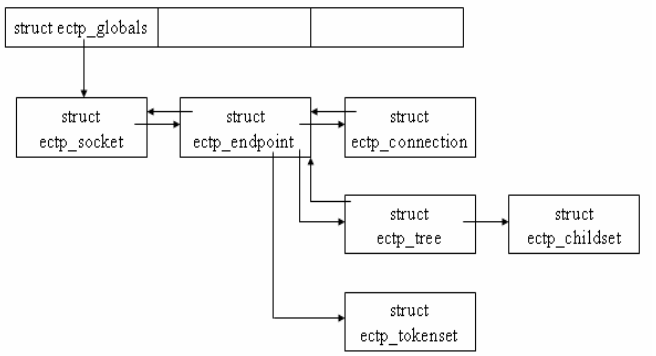

Figure 7. Internal Structures of ECTP-3

Figure 7 shows internal structures of ECTP-3. Structure of 'ectp_global' is a pointer for the associated socket structure. A socket structure is created when the ectp_socket API function is called. The endpoint structure represents the logical sender or receiver of ECTP-3 messages. It contains all information that is required to communicate with the parent or child in the ECTP-3 communication tree. The endpoint has 'back pointer' to the socket. The 'conenction' structure contains the ECTP-3 transmission control block parameters for two communication channels. Structure of 'tree' is used to control the tree of the ECTP-3. The connection and tree structures also have a back pointer to the endpoint. Structure of 'childset' is used to keep the current state of children. Structure of 'tokenset' keeps the current state of tokens. The childset and tokenset will be used by parent.

4. Experimentation

4.1 Testbed and Scenario

The Implementation of ECTP-3 is performed on Linux (Fedora3, Debian). Kernel verion of Linux is 2.6.x. A test application is implemented using the mplayer [3] and Qt Library[4]. The mplayer is used to play multimedia that received multicast data, whereas Qt Library is used to make the user interface.

The test experimentation was performed over a small testbed that consists of one TO and two TUs (SUs) in the subnet environment.

The following figure shows testbed configuration.

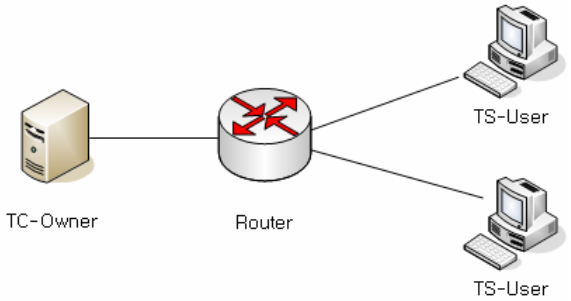

Figure 8. Testbed Configuration

The three test scenarios are as follows.

- Scenario A is a general one, which is purposed to see the normal connection creation operations of ECTP-3. When the TO is ready in the 'ESTABLESHED' state, a promising TU request 'Late Join' to the TO. The TO then transfers an acceptance or rejection to the TU. In the acceptance case, the TU will receive multicast data from the TO. If the TU wants to send unicast data to the TO, the TU shall request a token to the TO. In response to the request, the TO will transfer an acceptance or rejection to the TU. In the acceptance case, the TU (SU) will send unicast data to the TO. The TU will return the token to the TO, after all of unicast data transmission is completed.
- Scenario B is for analysis of the ECTP-3 late join operations. After a connection is created by TO, the first TU performs 'Late Join' at the time between 1 second and 2 second between 1 and 2 second. The other TU executes 'Late Join' at the time between 3 and 4 second.
- Scenario C is for analysis of the effect of the different ACK Generation Number (AGN) parameter values. It is noted that the AGN is used by a TU to determine when the ACK packets should be transmitted to the parent. For this purpose, the AGN value is set to 2, 4, and 8. We measure the number of control packets generated for each test case.

4.2 Result and Analysis

Figure 9 shows a screen capture for Scenario A. Application of TO displays the token message that received from the SU(TU). The TU application is playing the multimedia data received from the TO.

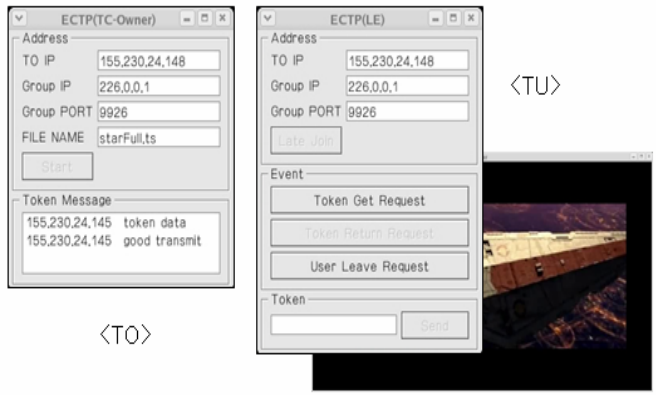

Figure 9. Experimentation Result (Scenario A)

Figure 10 shows the result by Scenario B. The figure shows the number of the data packets and control packets exchanged between TO and TUs.

In the experimentation, the TO send the multicast data with the bandwidth of 500 packets per second at the constant speed. One second later, a TU joins the connection and thus the associated control packets are generated. At the time of 3 second, the second TU also join the connection, and thus some more control traffic is generated, as shown in the figure.

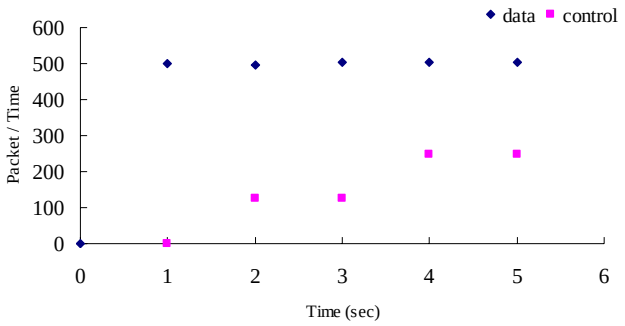

Figure 10. Number of Data and Control packets (Scenario B)

Figure 11 and 12 show the effect of the AGN number in terms of the control packets generated in the connection.

Figure 11 represents the number of ACK packets for the different AGN number of 2, 4, and 8. As per the ECTP-3 protocol, a child transmits an ACK packet to the parent, according the following condition:

$$PSN \% AGN = Child\ ID \% AGN$$

Therefore, if the AGN value increases, the number of the ACK packets will be decreased, as shown in Figure 11.

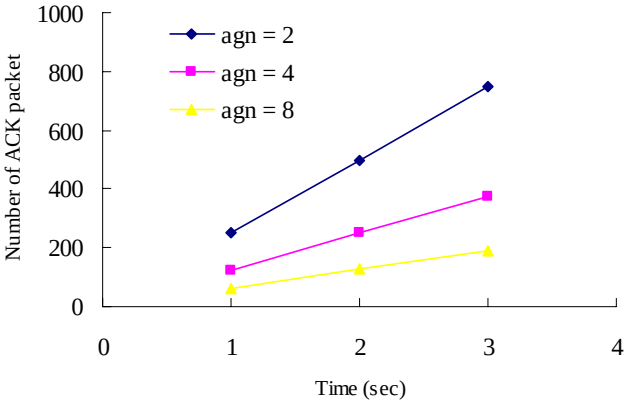

Figure 11. Effect of AGN for Single TU (Scenario C)

Figure 12 shows the total number of ACK packets for a different AGN value between one TO and two TUs.

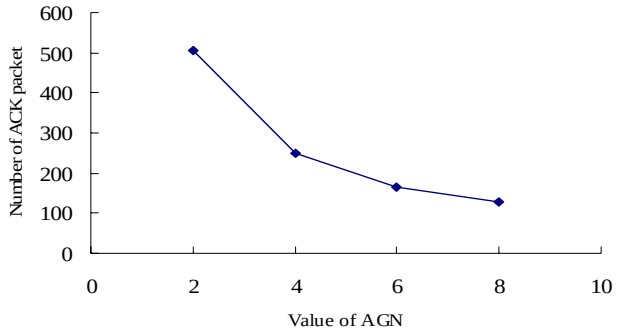

Figure 12. Effect of AGN for two TUs (Scenario C)

5. Conclusion

This paper describes the design and implementation of the ECTP-3. The implementation details are described and experimented over Linux platform with a video application. It is expected that the ECTP-3 protocol can be used for supporting the interactive multicast applications such as remote education and IPTV services, in which some interactions are required between TO (multicast sender) and TUs (multicast receivers).

REFERENCES

[1] ITU-T X.607 | ISO/IEC 14476-3, Enhanced Communication Transport Protocol – Part3 : Specification of Duplex Multicast Transport, 2006.
[2] ECTP Homepage, http://ectp.etri.re.kr/ectp.htm
[3] Media Player, http://www.mplayerhq.hu/
[4] Qt Library for GUI, http://www.trolltech.com
[5] R. Stevens, *et al.*, Unix Network Programming: The Sockets Networking API, Volume 1, 3rd Ed., Addison-Wesley, 2004
[6] R. Stevens, TCP/IP Illustrated, Volume 2, Addison-Wesley, 1995