ECTP 멀티캐스트 전송 프로토콜: 구현 및 성능분석

정회원 박기식, 종신회원 박주영, 종신회원 고석주, 정회원 조인준

Enhanced Communications Transport Protocol: Implementations and Experimentations

Ki-Shik Park*, Juyoung Park, Seok-Joo Koh***, and In-June Jo****** ***Regular Members***

ABSTRACT

This paper proposes a protocol for the reliableand QoS-aware multicast transport, which is called the Enhanced Communications Transport Protocol (ECTP). The ECTP has so far been developed and standardized in ITU-T SG17 and ISO/IEC JTC 1/SC 6. Differently from the conventional reliable multicast, as shownin the IETF RMT WG, the ECTP additionally provides several distinct features such as tight control of multicast session, tree-based error control, and QoS management. For the tight control of multicast connections, the sender is at the heart of one-to-many group communications, and it is responsible for overall connection management such as connection creation/termination, pause/resumption, and the join and leave operations. For tree-based reliability control, ECTP configures a hierarchical tree during connection creation. Error control is performed within each local group defined by a control tree, which was partly designed like the IETF TRACK approach. Each parent retransmits lost data in response to retransmission requests from its children. For QoS management, ECTP supports QoS negotiation for resource reservation, and it also provides QoS monitoring and maintenance operations. ECTP has been implemented and tested on Linux machine, along with Application Programming Interfaces based on Berkeley sockets. For basic testing of the ECTP functionality, we give some preliminary experimental results for performance comparison of ECTP and TCP unicast transports. In conclusion, we describe the status of ECTP experimentations over APAN/KOREN testbed networks

* 한국전자통신연구원 표준연구센터 (kipark@etri.re.kr), **한국전자통신연구원 표준연구센터 (jypark@etri.re.kr),
***한국전자통신연구원 표준연구센터 (sjkoh@etri.re.kr), ****배제대학교 컴퓨터공학과 (injune@mail.pcu.ac.kr)
논문번호 : 030300-0722, 접수일자 : 2003년 7월 28일

Ⅰ. Introduction

IP multicast issue has been one of the most important challenges in the research and development areas related to Internet [1]. In particular, we note that market requirements for multicast applications have recently emerged and rapidlygrown, such as webcasting, Internet TV, remote education, and stock-tickers. For realization of Internet multicast services, the following issues need to be addressed: Reliable Multicast Transport (RMT) and the Qaulity of Services (QoS) management in multicast transport. Actually, a lot of researches have been done in the RMT area [2], and so much standardization efforts have been made [3, 4]. Regardless of such works, however, neither satisfactory solution nor standard has been produced.

In this paper, we discuss a new standard multicast transport protocol that addresses those issues on QoS as well as RMT, which is called Enhanced Communications Transport Protocol (ECTP) [5, 6]. Until now, the ECTP has been developed and standardized in ITU-T SG17 and ISO/IEC JTC 1/SC 6 [7]. From such an effort, the ECTP was finally approved by ITU-T and ISO/IEC in 2001. This paper provides an overview of ECTP protocol mechanisms and describesthe associated implementation and experimentation results.

ECTP operates over IP networks that have IP multicast forwarding capability. ECTP is targeted for tightly controlled multicast services. The sender is at the heart of multicast group communications. The sender is responsible for overall connection management such as connectioncreation/ termination, connection pause/resumption, and user join/leave operations. ECTP provides tree-based reliability control for multicast data transport, which has been designed to keep congruency with those being proposed in the IETF RMT WG [4]. ECTPdistinctively provides QoS management for active multicast sessions by way of QoS negotiation, monitoring, and maintenance operations, so as to keep QoS of multicast session to be stable.

The ECTP has been designed based on preliminary standardization works defined in [8, 9, 10]. The ECTP has been so far standardized in the ITU-T SG 17 and ISO/IEC JTC 1/SC 6, as a joint work item [5, 6]. The ECTP has so far been implemented and experimented over Linux machines and local test environments. With experimentation experiences on ECTP, we now plan to extend the real validation of ECTP further to the real test networks such as the Asia-Pacific Advanced Networks (APAN) testbed.

This paper is organized as follows. Section 2 provides overall operations of the ECTP protocol. In Section 3, we discuss the protocol mechanisms for reliability control and QoS management more in details, together with ECTP packet structure. Section 4 describes implementation details including implementation stack with a kernel structure, andthe associated Application Programming Interfaces (API) libraries that will be referred to by multicast applications. In Section 5, we present some preliminary experimental test results for performance comparison of the ECTP and TCP connections, along with discussion of APAN testbed experimentation. Section 6 concludes this paper.

Ⅱ. ECTP Overview

ECTP is a standard transport protocol designed to support Internet multicast applications. ECTP operates over UDP/IP in multicast-enabled networks.

ECTP supports the connection management functions, which include connection creation and termination, connection pause and

resumption, and late join and leave. For reliable delivery of multicast data, ECTP provides the protocol mechanisms for error control. For QoS management of multicast sessions, ECTP supports QoS negotiation, monitoring, and maintenance operations.

Figure 1 shows an overview of the ECTP operations.

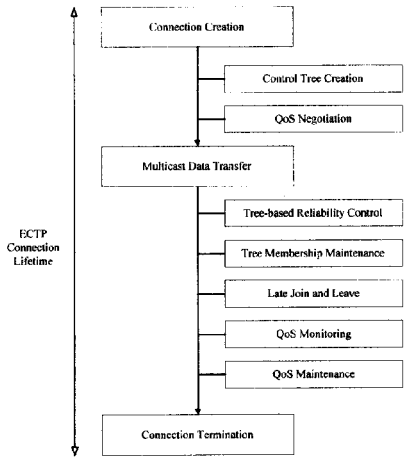


Figure 1. ECTP protocol operations

ECTP is targeted for tightly controlledmulticast connections. The ECTP sender is at the heart of the multicast group communication. The sender, designated as connection owner, is responsible for the overall management of the connection such as connection creation and termination, connection pause and resumption, and the late join and leave operations.

The ECTP sender triggers the connection creation process by sending a connection creation message. Each enrolled receiver responds with a confirmation message to the sender. The connection creation is completed when the sender receives the confirmation messages from the all of the active receivers, or when a pre-specified timer expires. QoS negotiation may be performed in the connection creation.

Throughout the connection creation, some or all of the enrolled group receivers will join the connection. The receivers that have joined the connection are called active receivers. An enrolled receiver that is not active can participate in the connection as a late-joiner. The late-joiner sends a join request to the sender. In response to the join request, the sender transmits a join confirm message, which indicates whether the join request is accepted or not. An active receiver can leave the connection by sending a leaving request to the sender. A trouble-making receiver, who cannot keep pace with the current data transmission rate, may be ejected.

After a connection is created, the sender begins to transmit multicast data. For data transmission, an application data stream is sequentially segmented and transmitted by means of data packets to the receivers. The receivers will deliver the received data packets to the applications in the order transmitted by the sender.

To make the protocol scalable to large multicast groups, ECTP employs the tree-based reliability control mechanisms. A hierarchical tree is configured during connection creation. A control tree defines a parent-child relationship between any pair of tree nodes. The sender is the root of the control tree. In the tree hierarchy, local groups are defined. A local group consists of a parent and zero or more children. The error, flow and congestion controls are performed over the control tree.

Figure 2 illustrates a control tree hierarchy for reliability control, in which a parent-child relationship is configured between a sender (S) and a receiver (R), or between a parent receiver (R) and its child receiver (R). ECTP provides the tree-configuration mechanisms [4, 5]

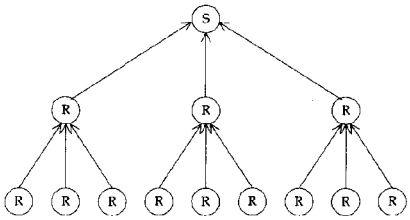


Figure 2. Logical control tree for reliability control

In ECTP, error control is performed for each local group defined by a control tree. If a child detects a data loss, it sends a retransmission request to its parent via acknowledgement (ACK) packets.

An ACK message contains the information that identifies the data packets, which have been successfully received. Each child can send an ACK message to its parentusing one of two ACK generation rules: ACK number and ACK timer. If data traffic is high, an ACK is generated for the ACK number of data packets. If the traffic is low, an ACK message will be transmitted after the ACK timer expires.

After retransmission of data, the parent activates a retransmission back-off timer. During the time interval, the retransmission request(s) for the same data will be ignored. Each parent can remove the data out of its buffer memory, if those have been acknowledged by all of its children.

During the data transmission, if network problems (for example, severe congestion), the sender suspends the multicast data transmission temporarily. In this period, no new data is delivered, while the sender transmits periodic null data messages to indicate that the sender is alive. After a pre-specified time has elapsed, the sender resumes the multicast data transmission.

After an ECTP connection is created, QoS monitoring and maintenance operations are performed for the multicast data transmission. For QoS monitoring, each receiver is required to measure the parameter values experienced. Based on the measured values, a receiver determines a parameter status value for each parameter. These status values will be delivered to the sender via ACK packets. Sender aggregates the parameter status values reported from receivers. If a control tree is employed, each parent LO nodes aggregates the measured values reported from its children, and forwards the aggregated value to its parent via its ACK packets.

Sender takes QoS maintenance actions necessaryto maintain the connection status at a desired QoS level, based on the monitored status values. Specific rules are pre-configured to trigger QoS maintenance actions such as data transmission rate adjustment, connection pause and resume, and connection termination. Those rules are based on observation that how many receivers are in the abnormal or possibly abnormal status.

The sender terminates the connection by sending a termination message to all the receivers, after all the multicast data are transmitted. The connection may also terminate due to a fatal protocol error such as a connection failure.

Ⅲ. ECTP in Details

In this section, we describe ECTP protocol mechanisms for tree-based error recovery and QoS management more in details. We also discuss ECTP packet structure.

1. Error Control in ECTP

In ECTP, error recovery operations consist of error detection by receivers, retransmission request by receivers via ACK packet, and retransmissions by parents.

A. Error detection

The header checksum field is used for

detection of packet corruption, and the sequence number field is for detection of a packet loss. When a data packet is received, each receiver examines the header checksum. If the checksum field is invalid, the packet is regarded as a corruption and shall be discarded. A corruption is treated as a loss. The loss can be detected as a gap of two consecutive sequence numbers for data (DT) packets. The loss information is recorded into the ACK bitmap, which is attached to the subsequent ACK packets.

B. Retransmission request

ACK packets are used for theretransmission requests. When a receiver detects a gap in the sequence numbers of received packets, it sets to zero the bit of the ACK bitmap, which corresponds to the lost DT packet. The ACK bitmap is included into the acknowledgement element, which is attached to the subsequent ACK packet and delivered to the parent by the ACK generation mechanisms.

For a local group, a parent and its children maintain the following variables to determine the status of received DT packets:

a) Lowest Sequence Number (LSN): If the node is a child, this is the sequence number of the lowest numbered DT packet that has not yet been received. If the node is a parent, then this is the sequence number of the lowest numbered DT packet that has not yet been received by any of its children;

b) Highest Sequence Number (HSN): If the node is a child, this is the sequence number of the highest numbered DT packet that has been received. If the node is a parent, then this is the sequence number of the highest numbered DT packet that has been received by any of its children nodes;

To request the retransmissions of lost data, each child makes an acknowledgement element containing the LSN, Valid Bitmap Length and ACK Bitmap. The Valid Bitmap Length is set to HSN LSN + 1. For an example, for LSN = 15 and HSN = 22, the Valid Bitmap Length = 8. The ACK Bitmap specifies a success or a failure of a packet delivery: '1' for success and '0' for failure. A bitmap can represent Bitmap Length * 32 packetsmaximally. Suppose Bitmap = 01101111. Then the DT packets with the sequence number 15 and 18 are lost.

Note that an intermediate LO on the tree has two sets of the LSN and HSN parameters: the one set as a child and the other set as a parent. The parameter values for a child are updated by the status of the data reception from TO, while the parameter values for a parent will be refreshed by the acknowledgement element from the children.

When a parent sends a HB packet to its children, it sets the sequence number field to the LSN. The data packets, whose sequence number is smaller than the LSN, cannot be recovered.

C. ACK generation

Each child generates an ACK packet by ACK Generation Number (AGN) or by ACK Generation Time (AGT). Each child sends an ACK packet to its parent every AGN packet. To do this, a child receives a Child ID from its parent in tree configuration, which is contained in the tree membership element. Each child sends an ACKpacket to its parent, if the sequence number of a DT packet modulo AGN equals Child ID modulo AGN, i.e., if

Packet Sequence Number % AGN = Child ID % AGN.

Suppose AGN = 8 and Child ID = 2. The child generates an ACK packet for the DT packets whose sequence numbers are2, 10, 18, 26, etc. This ACK generation rule is applied when the corresponding DT packet is received or detected as a loss by the child.

When data traffic is low, a receiver may not send an ACK packet for a long time. This could cause a long wait for packet stability at the parent and could also make the receiver appear to have failed. AGT is used to ensure that the receivers respond in a timely manner. A receiver sends at least one ACK packet within the AGT interval. AGT timer is initialized when a child receives the first DT packet, and it is reset each time a new ACKpacket is sent.

In summary, when the data traffic is high, ACK packets will be generated by the AGN number rule. On the other hand, ACK packets are triggered when AGT timer expires, if the traffic load is low.

D. ACK aggregation

Each parent uses ACK packets to gather status information for the error, flow and congestion controls. Each time a parent receives an ACK packet from any of its children, it records and updates the status information on which packet(s) have been successfully received. A DT packet is defined as a stable packet if all of the children have received it. The stable DT packets are released out of the buffer memory of the parent. When a parent receives an ACK packet from one of its children, if one or more packet losses are indicated, the parent transmits the corresponding RD packets to all of its children over its multicast control address.

E. Retransmission

In response to an ACK packet, each parent retransmits RD packets for the data that are requested by any children, if it holds the requested data packets. RD packets are retransmitted over the multicast control address.

After a parent sends an RD packet for the requested data, it activates the Retransmission Back-off Timer (RBT). During that time, retransmission requests for the same data packet will be ignored.

The maximum number of retransmissions for a lost DT packet shall be bounded to Maximum Retransmission Number (MRN). The parent ignores further retransmission requests exceeding MRN, and removes the corresponding data out of its buffer memory.

3.2 QoS Management in ECTP

This paper proposes the QoS management for one-to-many multicast transportprotocols. The QoS management function consists of the following operations: QoS negotiation, QoS monitoring, and QoS maintenance.

In the connection setup phase, sender informs the receivers whether QoS management is enabled. When QoS management is enabled, sender must also specify whether or not QoS negotiation will be performed in the connection. QoS monitoring and maintenance operations are performed.

In general, QoS represents the quality of services required for satisfactory reception of application data at the receiver side, to achieve desirable audio/video display quality for example.In this paper, it is assumed that the QoS requirements of an application are expressed in terms of one or more QoS parameters such as throughput, transit delay, transit delay jitter, and data loss rate. Depending on the application's requirements, some QoS parameters may not be used in the connection. For example, a non-real time service might not impose the transit delay requirement. On the other hand, new QoS parameter(s) may be defined in the future, as application requirements expand.

From the requirements of applications, sender will determine the target values for each QoS parameter. How to map from the

application's requirements to those target parameter values is outside the scope of this paper. Application programs could be developed to carry out such mappings

QoS negotiation is performed in the connection creation phase. Sender proposes the desired target values for each QoS parameter to all receivers by multicast. For throughput, three target values are specified: CHQ (controlled highest quality), OT (operating target) and LQA (lowest quality allowed). For the other parameters such as transit delay, transit delay jitter, and data loss rate, only two target values are specified: OT and LQA.

If QoS negotiation is enabled, each receiver can propose modifications to the sender's proposed parameter values. These modified values will be determined by considering the system capacity at the receiver side and network environments.

The parameter values modified by receivers are delivered to sender via feedback messages. The sender arbitrates different parameter values for various receivers by taking a commonly agreed range of values.

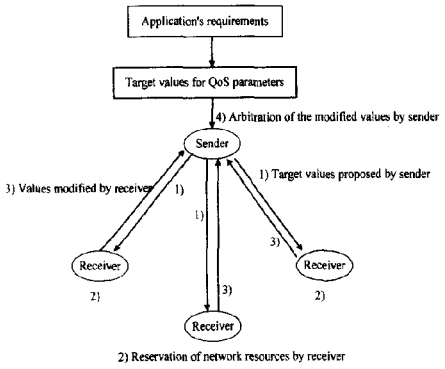


Figure 3. QoS negotiation

Figure 3 shows an abstract sketch of QoS negotiation. From the application requirements, a set of target QoS parameter values are configured. Sender informs the receivers of the target values (step 1). Based on those target values, each receiver begins to make resource reservations with help of RSVP or Diffserv (step 2). If QoS negotiation is enabled in the connection, each receiver may propose modified values for QoS parameters (step 3). From the modified parameter values, the sender determines the arbitrated values (step 4). These arbitratedvalues are delivered to the receiver via subsequent control packets, and will be used for QoS monitoring and maintenance.

After a connection is created, the QoS monitoring and maintenance operations are performed for the multicast data transmission. For QoS monitoring, each receiver is required to measure the parameter values experienced. Based on the measured values and the negotiated values, a receiver determines a parameter status value for each parameter as an integer: normal (0), reasonable (1), possibly abnormal (2), or abnormal (3). These status values will be delivered to the sender via control packets such as ACK packets.

Sender aggregates the parameter status values reported from the receivers. If a control tree is employed, each parent LO nodes aggregates the measured values reported from its children, and forwards the aggregated value(s) to its own parent using ACK packets.

Figure 4 illustrates the QoS monitoring and maintenance operations proposed in this paper. Sender takes QoS maintenance actions necessary to maintain the connection status at a desired QoS level, based on the monitored status values. Specific rules are pre-configured to trigger QoS maintenance actions such as data transmissionrate adjustment, connection pause and resume, and connection termination. Those rules are based on observing how many receivers are in the abnormal or possibly abnormal state.

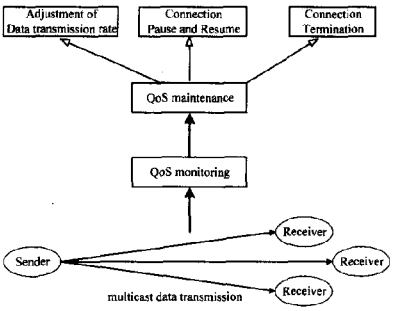


Figure 4. QoS monitoring and maintenance

3.3 Packet Structure

A) ECTP Packet Types

ECTP packets are classified into data and control packets. Data and Retransmission Data are the data packets. All the other packets are used for control purposes. Table 1 summarizes the packets used in ECTP. In the table, the transport type 'multicast' represents global multicast using a multicast data address, while the 'local multicast' does local multicast using a multicast control address. The Retransmission Data and Heartbeat packets are delivered from a parent to its children by local multicast.

Table 1. ECTP packets

Packet	Transport Type	From	To
Creation Request	Multicast	Sender	Receivers
Creation Confirm	Unicast	Child	Parent
Tree Join Request	Unicast	Child	Parent
Tree Join Confirm	Unicast	Parent	Child
Data	Multicast	Sender	Receivers
Null Data	Multicast	Sender	Receivers
Retransmission Data	(Local)Multicast	Parent	Children
Acknowledgement	Unicast	Child	Parent
Heartabeat	(Local)Multicast	Parent	Children
Late Join Request	Unicast	Receiver	Sender
Late Join Confirm	Unicast	Sender	Receiver
Leave Request	Unicast	Parent/ Child	Child/ Parent
Connection Termination	Multicast	Sender	Receivers

Each control or data packet consists of a header part and a data part, and the header part can contain zero or more extension elements as illustrated in Figure 5.

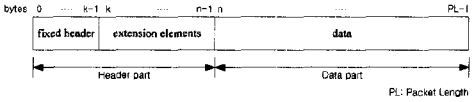


Figure 5. ECTP packet structure

In the figure, 'k', 'n' and 'PL' represent the length of the fixed header, the header part and the total packet.

B) Fixed header

The fixed header contains the fields of the parameters frequently used in the protocol. An example of the fixed header with 16 bytes is depicted below [5]:

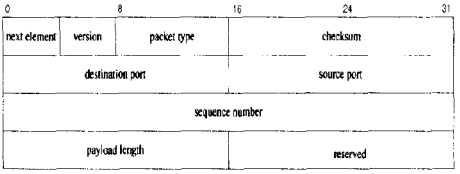


Figure 6. ECTP fixed header

C) Extension elements

The extension elements can follow the fixed header, and thus the header part of a packet is composed of a fixed header and zero or more extension elements. Each extension element has a next element field, as shown in Figure 7, which indicates the type of the next extension element. The header part can thus chain multiple extension elements.

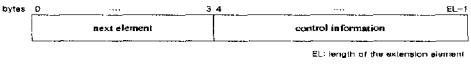


Figure 7. Structure of an extension element

ECTP defines five extension elements:

● Connection information element

This element contains information on generic characteristics of the connection including the connection type, tree configuration option, connection creation timer, and ACK bitmap size, etc. The control packets used for the connection creation include this element.

● Acknowledgment element

This element can be used for acknowledgment of the data packets and for report of the perceived connection status at the receiver side. A bitmap is used to indicate the selective acknowledgements of the received data.

● Tree information element

This element describes information on the local group defined by the control tree, including child ID and the number of active children, etc.

● Timestamp element

This contains the timestamp information, which may be used for calculating of round trip time for congestion control.

● QoS element

This extension element specifies QoS-related parameters. The following parameters can be specified: *throughput, transit delay, transit delay jitter, and packet loss rate.* If the resource reservation is enabled in the connection, these QoS parameter values willbe used as the target value for the resource reservation. During the connection, each receiver reports the measured QoS parameter values to its parent.

Ⅳ. Implementations

This section describes the detailed implementations of ECTP along with the associated packet format and API (application programming interfaces).

4.1 Kernel structure for implementation

The ECTP is currently being implemented on Linux RedHat 7.0 platform, with the C language. Some libraries are used such as LinuxThreads for ECTP Timerand Gtk+ for the ECTP applications with enhanced Graphic User Interface.

The current ECTP implementation is targetedto operate on top of UDP (UDP port: 9090 temporarily), with ECTP daemon process. Figure 8 shows the structure of ECTP kernel. Each application is assumed to use IPC (Inter Process Communication) for communications to ECTP.

As shown in the figure, an ECTP is designed to support one or more applications, and an IP host has an ECTP protocol stack, like the TCP or UDP. To enable ECTP, the application program on the ECTP must be compiled along with the ECTP socket interface "msocket". The use of ECTP socket is very similar to that of the BSD socket.

The communications between ECTP and an application are based on the IPC (Inter-Process Communication). The figure illustrates some representative functions and system calls used in the protocol stacks.

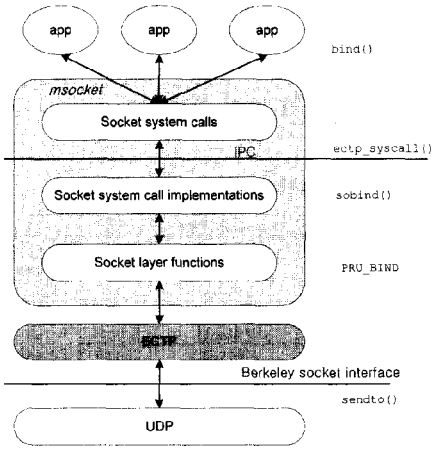


Figure 8. ECTP Implementation Stack

As shown in the figure, an ECTP is designed to support one or more applications,

and an IP host has an ECTP protocol stack, like the TCP or UDP. To enable ECTP, the application program on the ECTP must be compiled along with the ECTP socket interface "msocket". The use of ECTP socket is very similar to that of the BSD socket.

The communications between ECTP and an application are based on the IPC (Inter-Process Communication). The figure illustrates some representative functions and system calls used in the protocol stacks.

4.2 ECTP API (Application Programming Interface)

The ECTP API functions are designed based on the well-known Berkeley socket API in the fashion that the Berkeley socket API functions are used as wrapping functions in ECTP API. For indication of difference from the Berkeley socket functions, ECTP API functions are named with a prefix '*m*'.

Applications on ECTP are compiled with msocket libraries. The msocket is similar to the well-known BSD socket. The functions used by ECTP are named with a prefix of 'm', to differentiate the existing BSD functions.

Each function has the following syntax and usage in the ECTP protocol [6]:

- *int maccept(int msockfd, struct sockaddr *cliaddr, socklen_t *cliaddrlen);*

This function is used for LO and LE to generate CC (Creation Confirm) packet, after reception of CR packet and the tree formation.

- *ssize_t msend(int msockfd, const void *userdata, size_t nbytes, int *flags);*

This is used for TO to tranmit the multicast data.

- *int mclose(int mscokfd);*

This is used to terminate an ECTPconnection

- *int msocket(int family, int type, int protocol);*

This is used to open an ECTP socket. This function includes the other socket interfaces, which is a typical example of the "wrapping" function. Some example functions types embedded in the msocket are illustrated below:

- *int mbind(int msockfd, const struct sockaddr *myaddr, socklen_t myaddrlen, const struct sockaddr *grpaddr, socklen_t grpaddrlen, const struct sockaddr *ctladdr, socklen_t ctladdrlen, int role);*

This function keeps the information on the node type such as TO, LO or LE. In case of TO or LO, it may include its multicast control address.

- *int mconnect(int msockfd, const struct sockaddr *rmtaddr, socklen_t rmtaddrlen, int *flags);*

This is used when TO starts an ECTP connection. The rmtaddr represents the multicast group address.

- *size_t mrecv(int msockfd, void *userdata, size_t nbytes, int flags, struct sockaddr *fromaddr, socklen_t *fromaddrlen);*

This is used to receive an ECTP packet, which is similar to the BSD counterpart. But, to support multiple senders, the fromaddr is added to indicate the sender of the received data.

- *int mgetsockopt(int msockfd, int level, int optname, void *optval, socklen_t *optlen) and int msetsockopt(int msockfd, int level, int optname, void *optval, socklen_t *optlen);*

These functions are used to define the options necessary for the protocol.

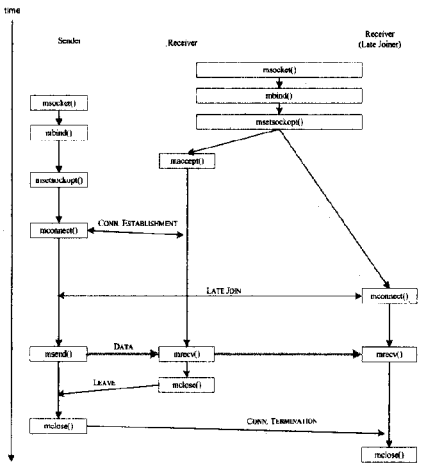


Figure 9. Use of ECTP API functions

Figure 9 illustrates an exampleuse of ECTP API functions in terms of the sender, early and late joining receivers. Sender invokes *mconnect()* after *mbind()* and *msetsockopt()*. A receiver waits for the connection establishment message from the sender. In case of late-joiner, it tries to connect to the sender by invoking *mconnect()* function. After connection is established, the sender begins to send the multicast data. During the data transmission, the ECTP protocolfunctions such as error recovery and QoS management will be provided to the applications services or users.

V. Experimental Results

This section first shows how the basic ECTP protocol functions operate via the screen captures from some experiments in a pre-configured local test environment. Through such experimentation, we also give some preliminary performance test results for ECTP protocol and multiple TCP connections. This comparison is done only so as to evaluate how much bandwidth gains the ECTP provides over multiple TCP connections. In this section, we also summarize the current status about ECTP test and validation over real APAN/KOREN testbed networks.

5.1 Preliminary Test of ECTP protocol Functions

In the ECTP implementation, each IP host runs the ECTP protocol daemon and the application itself in the pair-wise manner. The following subsections describe some of the basic protocol operations: connection creation, late join, and error recovery.

A) Connection Creation

Figure 10 shows the connection creation operation displayed at the sender (TO) and receiver(LE) sides. The creation of an ECTP connection is activated when the sender sends a CR packet. Each receiver receives CR and then HB packet from the sender. In response to the HB, it sends a TJ packet to the TO. When the receiver receives a TC packet from the sender, it sets the status as "established". After the CCT_time has elapsed, TO sets the status as ESTABLISHED.

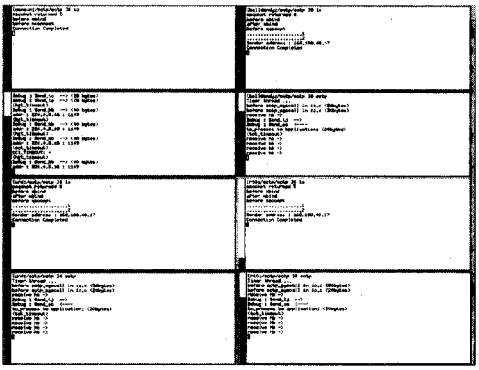


Figure 10. Connection Creation

B) Late Join

Figure 11 shows the late join operation

displayed at the sender (TO) and receiver(LE) sides. When a receiver joins the ECTP connection as a late joiner, it sends a JR packet to TO. In response to the JR packet, TO sends a JC packet to the late-joiner. When it receives JC packet form the TO, it begins to join a parent.

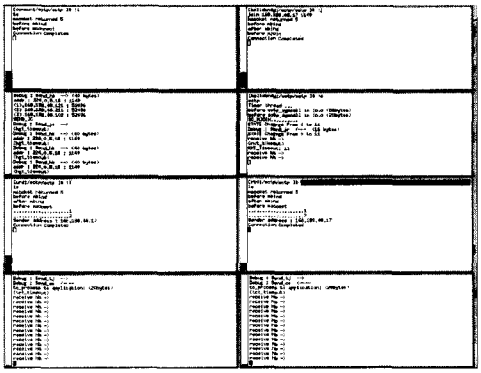


Figure 11. Late Join

C) Error Recovery

Figure 12 shows the error recovery operation displayed at the sender (TO) and receiver (LE) sides. When a receiver detects a packet corruption (by checksum) or packet loss (by sequence number), it requests a retransmission of lost data packets to its parent. Each ACK packet contains the selective ACK information (which data packets have been successfully received)

Each parent keeps the ACK information on each of its children. When a data packet has been acknowledged by all of its children, it deletes the data packet from the buffer. Each child generates an ACK packet to its parent by AGN (ACK generation number) and AGT (ACK generation timer). The following is the captured picture for generating ACK packet and aggregation of those ACK.

In the figure, the left-side host is TO, and the right-side three nodes are LEs. In this example, the AGN and ACK Bitmap Size are set to 8 and 32, respectively. Each child generates an ACK packet if the sequence number of the received data packet % AGN == Child ID % AGN. On the other hand, if the AGT timer expires, the child also generates an ACK packet.

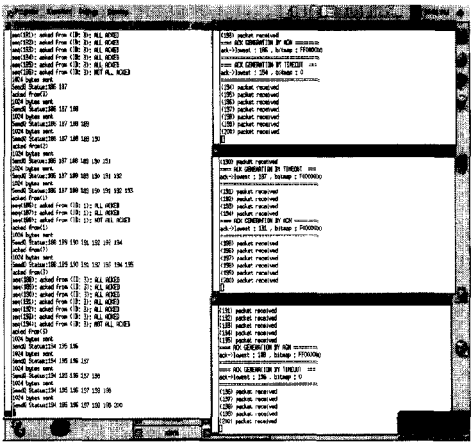


Figure 12. Error Recovery

5.2 Some Experimental Test Results

To test ECTP and TCP connections, we configure a test network consisting of one sender and 10 receivers. The sender generates the data stream until totally 100, 200, and 300 data packets have been generated. For each test instance, the totalnumber of data and control packets flowing in the network is calculated for the ECTP and TCP connections. The following figure shows an example configuration of local test environment for the testing and experimentation, which consist of one sender (TO) and three receivers (LEs).

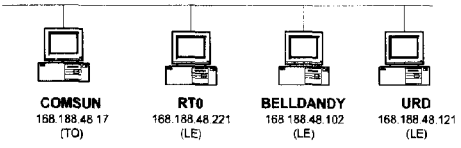
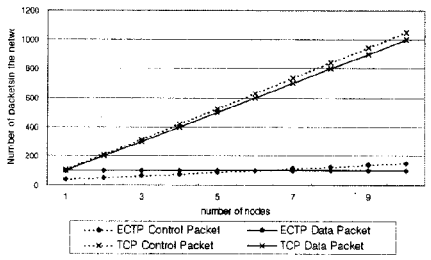
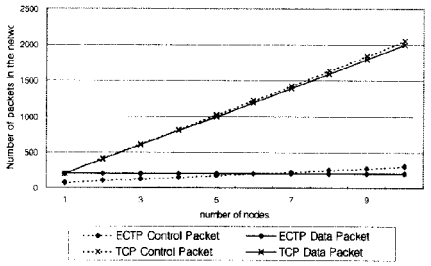


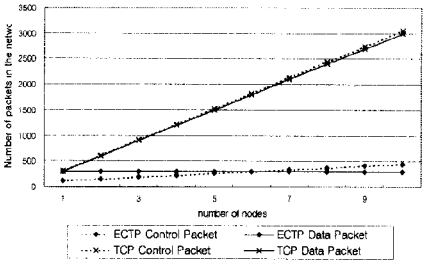
Figure 13. Local test environment



(a) Test result for transmission of data 100 packets



(b) Test result for transmission of data 200 packets



(c) Test result for transmission of data 300 packets

Figure 14. Performance comparison with TCP connections

Figure 14 shows the test results for ECTP and TCP connections, in terms of the total number of data and control packets generated in the network. From thefigures, we see that the ECTP connection generates almost an equal number of data and control packets, independently of the number of receivers. On the other hand, in multiple TCP connection, the number of data and control packets generated gets larger, as the number of receivers increases.

5.3 APAN/KOREN Testbed

For experimentations over real multicast-enabled networks, we have established a testbed environment over the APAN/KOREN (Korea Research and Education Network). Up-to-date, four organizations arebeing involved on the APAN test of ECTP: ETRI, KAIST, CNU (Chungnam National Univ.), and HUFS (Hankook University of Foreign Studies). Those organizations are interconnected via APAN/KOREN networks that provide the multicast forwarding capability over Internet as well as multicast tunneling.

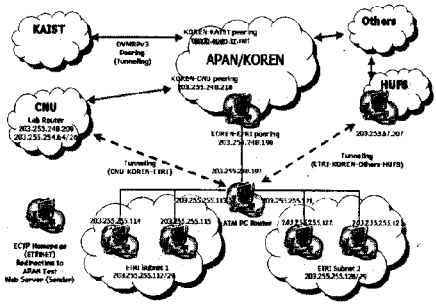


Figure 15. APAN/KOREN testbed networks

Figure 15 illustrates the network topology of the APAN/KOREN testbed for ECTP experimentations. In the figure, the ETRI node plays the sender and receivers via two local subnetworks, where the associated servers such as web server and media server will be located. Meanwhile, each of the other participants will play a role of receiver over the APAN/KOREN networks. In case of KAIST, the native multicast forwarding will be used for traffic delivery from ETRI, by using DVMRP protocol betweenhost and

edge, and PIM within backbone of KOREN. Communications to the other participants from ETRI will be made via multicast tunneling. Until now, the configuration of the testbed network connection has been completed, and the traffic delivery is being made (see http://ectp.etri.re.kr/ for more detailed information on APAN testbed and ECTP code release.)

VI. Conclusions

In this paper, we have discussed a new standard multicast transport protocol that addresses those issues on QoS as well as RMT, which is called Enhanced Communications Transport Protocol (ECTP). Differently from the IETF RMT WG approaches, the ECTP is designed to support tightly controlled multicast connections. The sender is at the heart of multicast group communication. The sender is responsible for overall connection management such as the connection creation and termination, the connection pause and resumption, and the join and leaveoperations. For tree-based reliability control, a hierarchical tree is configured during connection creation. Error control is performed for each local group defined by a control tree. Each parent retransmits lost data, in response to retransmission requests from its children.

Until now, the ECTP has been developed and standardized in ITU-T SG17 and ISO/IEC JTC 1/SC 6. From such an effort, the ECTP was finally approved by ITU-T and ISO/IEC in 2001. This paper provides an overview of ECTP protocol mechanisms and describesthe associated implementation and experimentation results. The ECTP validation works and code releases are still being progressed. Based on the test results, it is expected that the ECTP will be more improved and used in the real Internet multicast applications and services.

REFERENCES

[1] Diot, C, et. al., "Deployment Issues for the IP Multicast Service and Architecture", IEEE Networks Magazine's Special Issue on Multicast, January, 2000.

[2] K. Obraczka, "Multicast Transport Protocols: A Survey and Taxonomy", IEEE Communications Magazine, pp. 94 102, January 1998.

[3] IETF Reliable Multicast Transport (RMT) Working Group, Available from http://www.ietf.org/html.charters/rmt-charter.html, 2003.

[4] S. Koh, *et. al.*, "RMT: Tree auto-configuration," *IETF RMT WG Draft*,<draft-ietf-rmt-bb-tree-config-03.txt>, March 2003.

[5] ITU-T SG17 and ISO/IEC JTC 1/SC 6, "ECTP: Specification of Simplex Multicast Transport," *ITU-T X.606 | ISO/IEC 14476-1*, 2002.

[6] ITU-T SG17 and ISO/IEC JTC 1/SC 6, "ECTP: Specification of QoS Management for Simplex Multicast Transport," *ITU-T* X.606.1 | ISO/IEC CD 14476-2, Working in progress, 2003.

[7] Enhanced Communication Transport Protocol, Available from http://ectp.etri.re.kr, 2003.

[8] ITU-T and ISO/IEC JTC1, "Enhanced Communication Transport Services," *ITU-T Recommendation X.605 and ISO/IEC International Standard 13252*, 1999.

[9] ITU-T, "Multi-peer Communications Framework", *ITU-T Recommendation X.601*, March 2000.

[10] S. Koh, *et. al.*, "Configuration of ACK Trees for Multicast Transport Protocols," *ETRI Journal*, Vol. 23, No. 3, pp. 111 - 120, September 2001.

박 기 식(Ki-Shik Park) 정회원

1982년 :서울대학교 졸업(학사)
1985년 :서울대학교 행정대학원 졸업(정책학 석사)
1995년 :충남대학교 대학원 졸업(정책학 박사)
2003년 : 배재대학교 대학원 컴퓨터 공학과 박사과정 수료
1996년 - 현재 : 국제전기통신연합(ITU) 표준화자문위원회 Vice-Chairman, 문서처리분과위원회 Chairman
2001 년 4월 현재 : 국가과학기술위원회 국책연구개발사업 평가위원
2002 년 4월 현재 : W3C대한민국 사무국 사무국장
1985년 현재 : 한국전자통신연구원 표준연구센터장 (책임연구원)

<주관심분야> 정보통신 기술 표준화, 정보통신 QoS, 인터넷 프로토콜, R&D 및 기술정책

박 주 영(Juyoung Park) 종신회원

1995년 :충남대학교 전자공학과 졸업 (학사)
1997년 :충남대학교 대학원 졸업 (석사 : 정보통신 전공)
2001년 :충남대학교 대학원 졸업 (박사 : 정보통신 전공)
2001년 현재 : 한국전자통신연구원 표준연구센터 (선임연구원)

<주관심분야> 멀티캐스트, QoS, 라우팅 프로토콜, 모바일 인터넷

고 석 주 (Seok-Joo Koh) 종신회원

1992년 : 한국과학기술원 산업공학 졸업(학사)
1994년 : 한국과학기술원 졸업 (석사 : 산업공학 전공)
1998년 : 한국과학기술원 졸업 (박사 : 산업공학 전공)
1998년 현재 : 한국전자통신연구원 표준연구센터 (선임연구원)

<주관심분야> 멀티캐스트, 오버레이 멀티캐스트, QoS, 라우팅 프로토콜, 모바일 인터넷

조 인 준 (In-June Jo) 정회원

1982년 :전남대학교 계산통계산학과 졸업(학사)
1985년 :전남대학교 전자계산학과 대학원 졸업(석사)
1999년 :아주대학교 컴퓨터공학과 대학원 졸업(박사)
1990년 : 정보처리 기술사(전산조직 응용)
1983년 - 1994년 :한국전자통신연구소(선임연구원)
1994년 3월 -현재 : 배재대학교 컴퓨터공학과(교수)

<주관심분야> 정보통신 보안, 전산조직응용, 컴퓨터 네트워크(이동컴퓨팅),