# Enhanced Communications Transport Protocol for Multicast Transport

Seok Joo Koh, Juyoung Park, Eunsook Kim, and Shin-Gak Kang

Electronics and Telecommunications Research Institute,
161 Kajeong-Dong, Yusung-Gu, Daejeon, KOREA
{sjkoh, jypark, eunah, sgkang}@etri.re.kr

**Abstract.** This paper proposes a new multicast transport protocol, called the Enhanced Communications Transport Protocol (ECTP). The proposed protocol is currently being standardized in the ITU-T SG7 and ISO/IEC JTC 1/SC 6. The ECTP is designed to support tightly controlled multicast connections. The sender is at the heart of one-to-many multicast group communications. The sender 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. Each parent retransmits lost data in response to retransmission requests from its children. ECTP has been implemented and tested on Linux machine, along with Application Programming Interfaces based on Berkeley sockets.

## 1. Introduction

This paper proposes a new protocol for tight control of multicast transport connections, named the Enhanced Communications Transport Protocol (ECTP). ECTP operates over IP networks that have IP multicast forwarding capability [1].

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 connection creation/termination, connection pause/resumption, and user join/leave operations.

The sender triggers the connection creation process. Some or all of the enrolled receivers will participate in the connection, becoming designated “active receivers”. Any enrolled receiver that is not active may participate in the connection as a late-joiner. An active receiver can leave the connection. After the connection is created, the sender begins to transmit multicast data. If network problems (such as severe congestion) are indicated, the sender suspends multicast data transmission temporarily, invoking the connection pause operation. After a pre-specified time, the sender resumes data transmission. If all of the multicast data have been transmitted, the sender terminates the connection.

ECTP provides reliability control for multicast data transport, which has been designed to keep congruency with those being proposed in the IETF [2]. To address

I. Chong (Ed.): ICOIN 2002, LNCS 2343, pp. 64-74, 2002.

reliability control with scalability, the IETF has proposed three approaches: Tree based ACK (TRACK), Forward Error Correction (FEC), and Negative ACK Oriented Reliable Multicast (NORM). ECTP adopts the TRACK approach, because it is more similar to the existing TCP mechanisms and more adaptive to the ECTP framework.

The ECTP has been designed based on the preliminary works defined in [3], [4], and so far standardized in the ITU-T SG7 and ISO/IEC JTC 1/SC 6, as a joint work item [5], [6]. The ECTP has been implemented over Linux machine and tested on 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 implementation details together with packet format and the associated Application Programming Interfaces. Section 4 presents some preliminary experimental results for ECTP and multiple TCP connections. Section 5 concludes this paper.

## 2. Protocol Overview

The ECTP is a transport protocol designed to support Internet multicast applications. ECTP operates over IPv4/IPv6 networks that have IP multicast forwarding capability.

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 also provides the protocol mechanisms for error, flow and congestion controls. To allow scalability to large-scale multicast groups, tree-based reliability control mechanisms are employed which are congruent with those being proposed in the IETF RMT WG.

Figure 1 shows an overview of the ECTP operations. Before an ECTP transport connection is created, the prospective receivers are enrolled into the multicast group. Such a group is called an enrolled group. The IP multicast addresses and port numbers must be announced to the receivers. These enrollment operations may rely on the SAP/SDP, Web Page announcement and E-mail. An ECTP transport connection is created for the enrolled receivers.

ECTP is targeted for tightly controlled multicast 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.

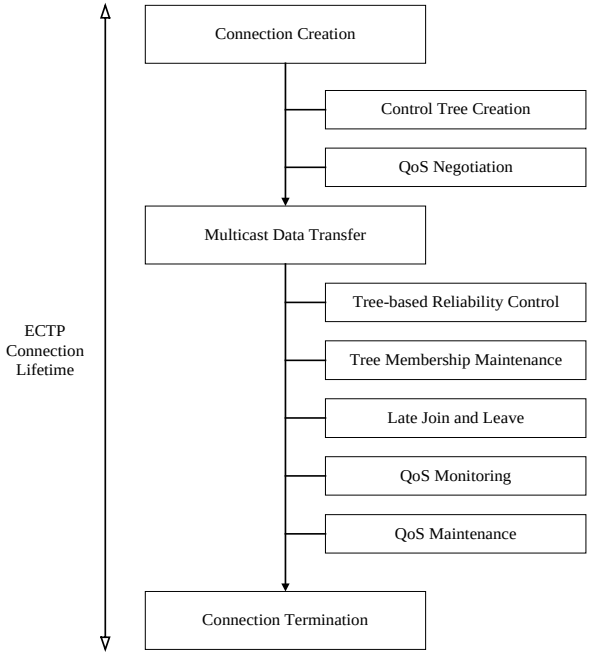



**Fig. 1.** ECTP Protocol Operations

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).

In the tree creation, a control tree is gradually expanded from the sender to the receivers. This is called a top-down configuration [7]. On the other hand, the IETF RMT WG has proposed a bottom-up approach, where the receivers initiate a tree configuration. Those schemes may be incorporated into the ECTP as candidate tree creation options in the future.

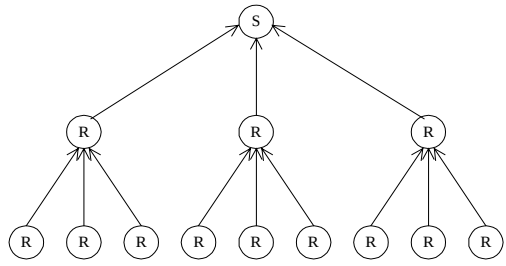



**Fig. 2.** Control Tree for Reliability Control

Tree-membership is maintained during the connection. A late-joiner is allowed to join the control tree. The late-joiner listens to the heartbeat messages from one or more on-tree parents, and then joins the best parent. When a child leaves the connection, the parent removes the departing child from the children-list. Node failures are detected by using periodic control messages such as null data, heartbeat and acknowledgement. The sender transmits periodic null data messages to indicate that it is alive, even if it has no data to transmit. Each parent periodically sends heartbeat messages to its children. On the other hand, each child transmits periodic acknowledgement messages to its parent.

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 ACK.

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 parent using 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 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 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 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.

## 3. Implementations

### 3.1 Packet Structure

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 | Multicast | Parent | Children |
| Acknowledgement | Unicast | Child | Parent |
| Heartbeat | 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 3.

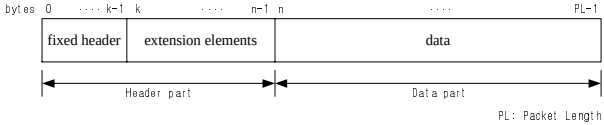



**Fig. 3.** Packet Structure

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

### 3.1.1 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:

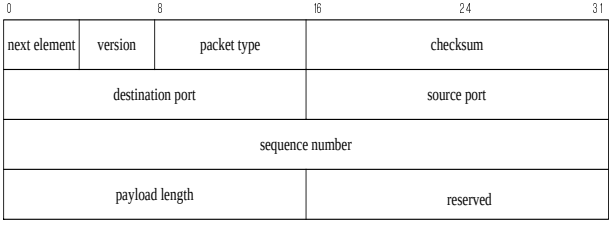



**Fig. 4.** Fixed Header

- *Next element* indicates the type of the next component immediately following the fixed header. The extension element also has the next element field, and thus the header part can chain multiple extension elements.
- *Version* defines the current version of the fixed header usage. Since each extension element has its own version field.
- *Packet type* indicates the type of the current packet. The maximum number of packet types to be defined is $2^8 = 256$.
- *Checksum* is used to check segment validity of a packet.
- *Destination* and *source ports* are used to identify the sending and receiving applications. These two values, together with the source and destination IP addresses in the IP header, uniquely identify each transport connection.
- *Sequence number* is the sequence number of a packet in a series of segments. This sequence number is a 32-bit unsigned number that wraps back around to '0' after reaching '$2^{32} - 1$'.
- *Payload length* indicates the length of the data part in bytes following the protocol header part. It can be used to indicate presence or absence of the data part.

### 3.1.2 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 Fig. 5, which indicates the type of the next extension element. The header part can thus chain multiple extension elements.

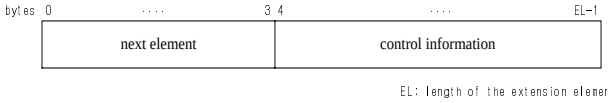



**Fig. 5.** Structure of Extension Element

The next element field can be encoded to indicate which type of the extensions element follows immediately. The next element field of the last extension element must be encoded as '0000', indicating "no further element".

According to the extension element type, its next element field is encoded as shown in Table 2. The next element field of the last extension element MUST be '0000'.

**Table 2.** Encoding table of the extension elements

| Element | Encoding |
|---|---|
| Connection Information | 0001 |
| Acknowledgment | 0010 |
| Tree Membership | 0011 |
| Timestamp | 0100 |
| QoS | 0101 |
| No element | 0000 |

Each element specifies the following:

- *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.
- *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, etc.
- *Timestamp element*: This contains the timestamp information.
- *QoS element*: This extension element specifies QoS-related parameters: *throughput, transit delay, transit delay jitter, and packet loss rate.*

### 3.2 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 Timer and Gtk+ for the ECTP applications with enhanced Graphic User Interface.

The current ECTP implementation is targeted to operate on top of UDP (UDP port: 9090 temporarily), with ECTP daemon process. Figure 6 shows the structure of ECTP kernel. Each application is assumed to use IPC (Inter Process Communication) for communications to ECTP.

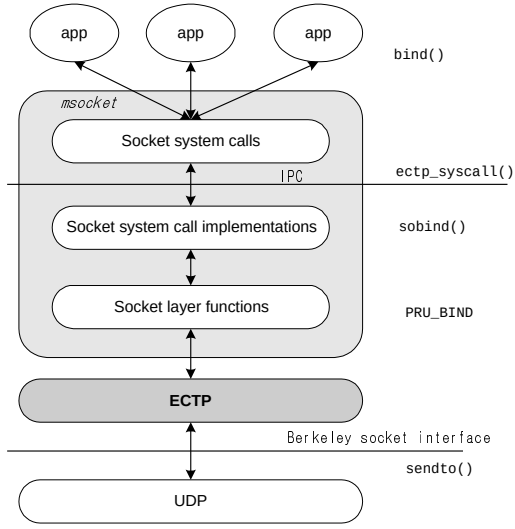



**Fig. 6.** ECTP Kernel Structure

### 3.3 Application Programming Interfaces

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*'. The following API functions are invoked by applications to communicate to ECTP:

- msocket(): This is used to create a socket for ECTP communications.
- mbind(): This function is to used to bind a pair of an IP address and a port to the socket.
- mconnect(): This is used by sender to initiate the connection creation, or by late-joiner to connect to the sender.
- maccept(): Each receiver waits for the connection creation indication signal from the sender by invoking this function.
- msend(): This is used by sender to transmit the multicast data.
- mrecv(): This is used by receivers to receive the multicast data.
- mclose(): This is used by sender to terminate the connection, or by a receiver to leave the connection.
- msetsockopt(): This is used to configure a set of socket options necessary for ECTP communications.
- mgetsockopt(): This is used to obtain the currently configured socket options.

Figure 7 illustrates an example use 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.

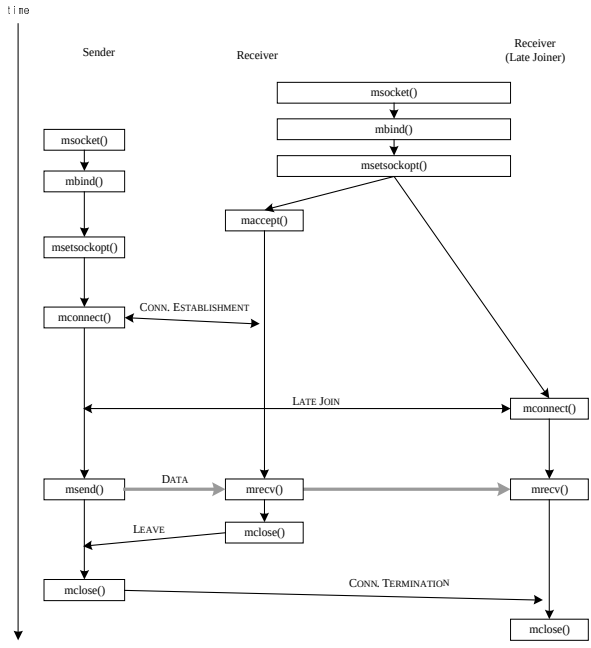


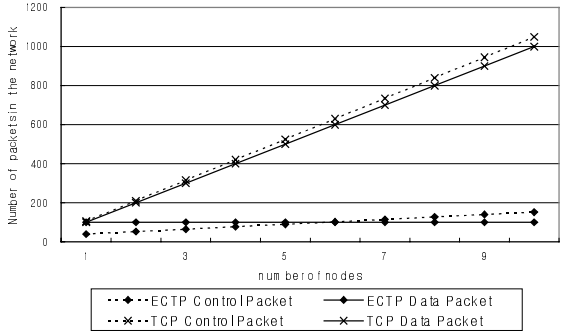

**Fig. 7.** Use of ECTP API

## 4. Experimental Results

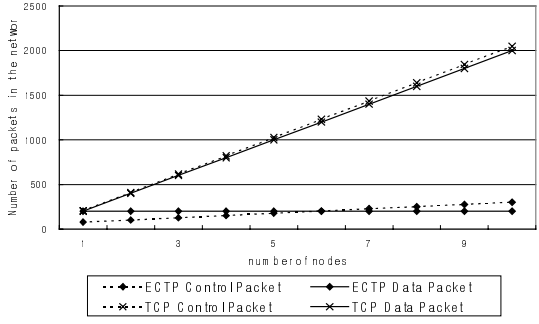
This section shows some preliminary experimental results for the ECTP protocol performance test. This test is mainly targeted to evaluate how much bandwidth gains the ECTP provides over multiple TCP connections.

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 total number of data and control packets flowing in the network is calculated for the ECTP and TCP connections.

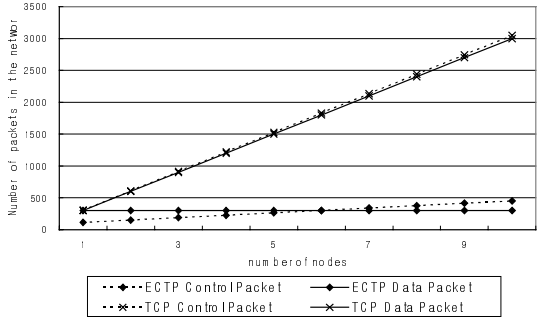
Figure 8 shows the test results for ECTP and TCP connections, in terms of the total number of data and control packets generated in the network. From the figures, we see that the ECTP connection generates almost an equal number of data and control packets, independently of the number of receivers. On the other hand, in multiple TCP connection, the number of data and control packets generated gets larger, as the number of receivers increases.




(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

**Fig. 8.** Performance Comparison with TCP Connections

## 5. Conclusions

This paper has discussed a new multicast transport protocol, ECTP. We presented the protocol operations, implementation issues, and some preliminary results. The ECTP is designed for tightly controlled one-to-many multicast transport connection. The proposed protocol has been standardized in ITU-T SG7 and ISO/IEC JTC 1/SC 6.

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 leave operations. 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.

## References

1. Enhanced Communication Transport Protocol, Available from http://ectp.etri.re.kr, 2001
2. IETF Reliable Multicast Transport (RMT) Working Group, 2001
3. ITU-T and ISO/IEC JTC1, “Enhanced Communication Transport Services,” ITU-T Recommendation X.605 and ISO/IEC International Standard 13252, 1999.
4. ITU-T, “Multi-peer Communications Framework”, ITU-T Recommendation X.601, 2000
5. ITU-T SG7 and ISO/IEC JTC 1/SC 6, “ECTP: Specification of Simplex Multicast Transport,” ITU-T X.606 | ISO/IEC FDIS 14476-1, 2001
6. ITU-T SG7 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, 2001
7. S. Koh, E. Kim, J. Park, S. G. Kang, et al., “Configuration of ACK Trees for Multicast Transport Protocols,” ETRI Journal, Vol. 23, No. 3, pp. 111 – 120, September 2001
8. S. Koh, et al., “Minimizing Cost and Delay in Shared Multicast Trees,” ETRI Journal, Vol. 22, No. 1, pp. 30 – 37, March 2000