Framework of QoS Management for Multicast Communications

Seok Joo Koh, Juyoung Park, Eunsook Kim, and Shin Gak Kang
Protocol Engineering Center, Electronics Telecommunications Research Institute
sjkoh@pec.etri.re.kr

Abstract — This paper proposes a framework of the QoS management for one-to-many multicast communications in the context of multicast transport protocols. Over the multicast transmissions based on IP multicast, we propose an architecture of QoS management so as to address the QoS needs of end users, and further to enable service providers to obtain information for the billing and charging model. The proposed QoS management scheme is also currently being standardized in the ITU-T SG7 and ISO/IEC JTC 1/SC 6 as the project of Enhanced Communications Transport Protocol (ECTP). The ECTP has been implemented and tested on Linux machine, along with Application Programming Interfaces based on Berkeley sockets.

Keywords — Quality of Service Management, Multicast Transport Protocols.

1. Introduction

As the need of group communications, the Internet multicast application services have recently been focused, which include Internet TV broadcasting, remote education, and stock tickers [1]. One of the challenging issues for wide deployment of payable multicast services is to provide Quality of Service (QoS) management for the value-added multicast communications.

Most of the researches on multicast transport have so far been done for providing the reliability control such as error recovery and congestion avoidance [2, 3]. Those works are purposed to design the multicast transport protocols scalable to a large number of users. On the other hand, the works on QoS-related issues in the multicast communications have not been made enough.

This paper presents a framework of QoS management for one-to-many multicast communications in the context of multicast transport protocols. Over the multicast transmissions based on IP multicast, we propose an architecture of QoS management so as to address the QoS needs of end users, and further to enable service providers to obtain information for the billing.

In this paper, the QoS management functions are divided into three operations: QoS negotiation, QoS monitoring, and QoS maintenance. In the QoS negotiation, a set of QoS parameter values are announced and negotiated according to the QoS requirements of applications, which will be typically performed during the connection setup phase. Over the multicast data transmission phase, the QoS monitoring and maintenance operations are taken for the sender to diagnose the session status and to take an action required for QoS maintenance such as data rate adjustment or troublemaker ejection.

The proposed QoS management scheme is also currently being standardized in ITU-T SG7 and ISO/IEC JTC 1/SC 6, as the project of the Enhanced Communications Transport Protocol (ECTP) [4].

This paper is organized as follows. Section 2 proposes the architecture of QoS management for multicast communications. In Section 3, we describes the QoS management operations employed in the ECTP protocol. Section 4 discusses implementation issues for the proposed scheme. In Section 5 we conclude this paper.

2. Framework of QoS Management

This paper proposes the QoS management for one-to-many multicast transport protocols. The QoS management function consists of the following operations:

1) QoS negotiation
2) QoS monitoring; and
3) 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.

Figure 1 illustrates the QoS management operations taken for the multicast communications. In the figure, the protocol operations marked as dotted lines are generic components for reliability control [3].

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 specification, 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 specification. Application programs could be developed to carry out such mappings.

Figure 1 – QoS management operations

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 following restrictions are imposed for modification of the parameter values:

1) OT values must not be modified by receivers;
2) The values modified by receivers must be within LQA and CHQ values proposed by sender.

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 2 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 arbitrated values are delivered to the receiver via subsequent control packets, and will be used for QoS monitoring and maintenance.

Figure 2 – QoS negotiation

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 node aggregates the measured values reported from its children, and forwards the aggregated value(s) to its own parent using ACK packets.

Figure 3 – QoS monitoring and maintenance
Figure 3 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 transmission rate 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.

3. QoS Management in ECTP

The ECTP is a transport protocol designed to support Internet multicast applications. ECTP operates over IPv4/IPv6 networks that have IP multicast forwarding capability. The ECTP has been designed based on the preliminary works defined in [5, 6], and so far standardized in the ITU-T SG7 and ISO/IEC JTC 1/SC 6, as a joint work item. The ECTP has been implemented over Linux machine and tested on the Asia-Pacific Advanced Networks (APAN) testbed, and the source codes are now open to the public for enhancement of the related research works [7].

ECTP supports the connection management functions. 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.

The following subsections describe the components required for QoS management operations in ECTP.

3.1. QoS Parameters

For QoS management, the following four QoS parameters are defined:

1) throughput (bytes per second);
2) transit delay (millisecond);
3) transit delay jitter (millisecond);
4) data loss rate (percent).

Throughput represents an amount of application data output over a specific time period. Target throughput means a throughput value required for desirable display of application data. Applications generate multicast data and ECTP sender will transmit them, based on the target throughput value(s). Actual data reception rate at receiver’s side depends on data transmission rate and network conditions.

For throughput, the sender can configure the following target throughput values:

1) CHQ throughput;
2) OT throughput;
3) LQA throughput.

Among them, the following inequalities are enforced: LQA throughput ≤ OT throughput ≤ CHQ throughput.

Transit delay represents end-to-end transmission delay from sender to receiver. For desirable display of multicast data, the sender can configure the following target values for transit delay:

1) OT transit delay;
2) LQA transit delay.

Between them, the following inequalities are enforced: OT transit delay ≤ LQA transit delay.

Transit delay jitter represents variations of transit delay values. For desirable display of data, the sender can configure the following target values for transit delay jitter:

1) OT transit delay jitter;
2) LQA transit delay jitter.

Between them, the following inequalities are enforced: OT transit delay jitter ≤ LQA transit delay jitter.

Data loss rate is defined as a ratio of the amount of lost data over the amount of totally transmitted data. For desirable display of data, the sender can configure the following target values for data loss rate:

1) OT loss rate;
2) LQA loss rate.

Between them, the following inequalities are enforced: OT loss rate ≤ LQA loss rate.

3.2. Packet Format and QoS Element

An ECTP packet consists of a header part and a data part, as illustrated in Figure 4.

Figure 4. Packet structure

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

3.2.1 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:

Figure 5. An example of the fixed header [3]
3.2.2 QoS 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 6, which indicates the type of the next extension element. The header part can thus chain multiple extension elements.

Figure 6. Structure of an extension element [3]

The QoS extension element specifies the Maximum Segment Size (MSS) and the target values for ECTP QoS parameters. As shown in the following figure, the QoS element has a length of ‘28’ bytes:

<table>
<thead>
<tr>
<th>next element</th>
<th>version</th>
<th>QoS flags</th>
<th>Maximum Segment Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHQ throughput</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OT throughput</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LQA throughput</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OT transit delay</td>
<td>LQA transit delay</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OT transit delay jitter</td>
<td>LQA transit delay jitter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OT loss rate</td>
<td>LQA loss rate</td>
</tr>
</tbody>
</table>

Figure 7 – QoS extension element [4]

The following parameters are specified:

a) **Next Element** – indicates the type of the next element immediately following this QoS element;

b) **Version** – defines the current version of this element, starting at ‘1’;

c) **QoS flags** – is a flag byte to specify if QoS parameters and MSS are used in the connection. Encoding of this byte is depicted in the following figure. If a bit is set to ‘1’, then the corresponding QoS parameter or MSS will be used. The default value is ‘0’ for each bit.

<table>
<thead>
<tr>
<th>7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved E D C B A</td>
</tr>
</tbody>
</table>

1) A – throughput;
2) B – transit delay;
3) C – transit delay jitter;
4) D – data loss rate;
5) E – maximum segment size (MSS);
6) **Reserved** - is reserved for future use.

d) **Maximum Segment Size (MSS)** – represents the maximum size of an ECTP segment or packet in unit of bytes. If the ‘E’ bit of the QoS flags is set to ‘1’. Otherwise, the default MSS value of 1024 will be used.

e) **Throughput values** – Each value is a 32-bit unsigned integer in byte per second. The following target values are valid only if ‘A’ bit of QoS flags is set to ‘1’:

1) CHQ throughput – upper limit for throughput;
2) OT throughput – target throughput for desired display of multicast data;
3) LQA throughput – lower limit for throughput.

f) **Transit delay values** – Each value is a 16-bit unsigned integer in millisecond. The following target values are valid only if ‘B’ bit of QoS flags is set to ‘1’:

1) OT transit delay – target transit delay for desired display of multicast data;
2) LQA transit delay – maximally allowed transit delay.

g) **Transit delay jitter values** – Each value is a 16-bit unsigned integer in millisecond. The following target values are valid only if ‘C’ bit of QoS flags is set to ‘1’:

1) OT transit delay jitter – target transit delay jitter for desired display of multicast data;
2) LQA transit delay jitter – maximally allowed transit delay jitter.

h) **Data loss rate values** – Each value is an 8-bit unsigned integer ranged from 0 to 100 in percent. The following target values are valid only if ‘D’ bit of QoS flags is set to ‘1’:

1) OT loss rate – target loss rate for desired display of multicast data;
2) LQA loss rate – maximally allowed loss rate.

i) **Reserved** - is reserved for future use.

The QoS element is used for the sender to inform the receivers of the target values for QoS parameters via the CR packet used in the connection negotiation phase. In QoS negotiation, the QoS element is also used when a receiver proposes modified target values to the sender.

4. Implementation Issues

4.1. Implementation Overview

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.
4.2 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’.

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

- msocket()
  This is used to create a socket.

- mbind()
  This function is 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 creation indication signal.

- 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 to terminate 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 9 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.

5. Conclusions

This paper presents the architecture of QoS management for multicast communications. In the context of the multicast transport protocol such as ECTP, the detailed design and implementation issues were discussed, along with some preliminary experimental results.

REFERENCES


