Performance of SCTP for IPTV Applications†

Sang Tae Kim[1], Seok Joo Koh[1], Yong Jin Kim[2]
[1]Department of Computer Science, Kyungpook National University
[2]Modacom Incorporation
{saintpaul1978, sjkoh}@cs.knu.ac.kr, cap@modacom.com

***Abstract* — Stream Control Transmission Protocol (SCTP) is a new reliable transport layer protocol. This paper analyzes the performance of the SCTP for IPTV applications using the SCTP distinctive features such as Partial-Reliable SCTP (PR-SCTP) and multi-streaming. From the experimental results, it is shown that the PR-SCTP provides better goodput than TCP and better quality of services than UDP for real-time IPTV applications. From the viewpoint of the multi-streaming feature, it is shown that the SCTP overcomes the well-known Head-of-Line (HoL) blocking problems when multiple streams are transported over the SCTP. In addition, the multi-streaming SCTP gives faster channel switching time, compared to the TCP.**

***Keywords* — SCTP, IPTV, PR-SCTP, Multistreaming**

1. Introduction

The Internet Protocol Television (IPTV) has been regarded as a killer application in the next generation networks. The remarkable features of IPTV services include the 'real-time' and 'interactive'. It seems that the existing transport protocols, TCP and UDP, are not suitable to the IPTV applications, since those protocols were not designed to meet the requirements for the IPTV. In particular, the UDP performs only "best-effort" service with no congestion control mechanism, and thus the sender may inject too much data into the already congested network, which will make the network condition worse. Moreover, in case of using the TCP for transport of IPTV applications, multiple TCP connections may be needed to support a lot of IPTV streaming channels concurrently.

In this paper, we suggest the use of the Stream Control Transmission Protocol (SCTP) [1] as a transport protocol so as to support real-time IPTV applications. The SCTP is a new reliable transport layer protocol. Compared to TCP, the SCTP provides the distinctive features such as 'multi-streaming' and 'multi-homing'. Especially, the SCTP multi-streaming feature enables an upper layer application to separate the data streams logically, by using the Sockets API [2] with different stream identifiers (SIDs) for each of the user data streams. In addition, the SCTP can be used to reduce the well-known Head-of-line (HoL) blocking effect that occurs in the TCP connection.

The Partial-Reliable SCTP (PR-SCTP) [3] is an extension of the SCTP that can be used to support the real-time data transport with the partial-reliable semantics. In the PR-SCTP scheme, the FORWARD-TSN control chunk will be used to inform the peer about the expiration of the data chunks for real-time applications, which will enforce the peer SCTP to move the cumulative ACK point forward. In this fashion, the upper layer applications using PR-SCTP can provides "timed reliability" services for transport of real-time IPTV traffic.

There have been some studies on the performance of the SCTP until now. Those studies can be roughly classified into

(1) congestion control,
(2) multihoming,
(3) multistreaming,
(4) out-of-order service,
(5) partial reliability extension,
(6) application in the wireless/mobile environment, and
(7) SCTP over satellite networks [4].

In particular, we note some works on the multi-streaming and partial-reliable extension for IPTV applications. In the point of view of multi-streaming, the work in [5] analyzes the performance of the "FTP over SCTP" in three ways: (1) simply replacing TCP calls with SCTP calls, (2) using a single multi-streamed SCTP association for control and all data transfers, and (3) enhancing SCTP with command pipelining. Research in [6] implements "HTTP over SCTP. In the study, it is shown that the proposed schemes can significantly reduce the HoL blocking effect. On the other hand, the researches in [7] and [8] suggested the use of PR-SCTP as a transport protocol of the MPEG multimedia traffic and SIP traffic. Those works show that the adjustment of 'lifetime' of PR-SCTP may give a significant impact on the throughput performance of SCTP.

In this paper, we will focus on the analysis of the performance for the IPTV over SCTP. In particular, the SCTP multi-streaming and PR-SCTP features will be analyzed and evaluated for IPTV applications over the Linux testbed networks.

The rest of this paper is organized as follows. Section 2 describes the features of SCTP for real-time IPTV applications: PR-SCTP and multi-streaming. Section 3 gives an overview of our implementations in the Linux platform. In Section 4, we analyze the performance of SCTP over the Linux testbed and compared with the TCP and UDP. Finally, Section 5 will conclude this paper.

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

2. Features of SCTP for IPTV Applications

In this section, we describe the new features of SCTP that can be considered to support real-time IPTV applications.

2.1 PR-SCTP

The PR-SCTP can be used to transport real-time multimedia traffic with some benefits. Applications can use a single SCTP association to carry both reliable and unreliable contents with TCP-friendliness. It is noted in the PR-SCTP that the control or adjustment of 'lifetime' may give a significant impact on the throughput performance of SCTP.

In particular, it seems that there are several factors to be considered to determine an optimal "lifetime" of each DATA chunk, such as network transmission delay, packet loss, and the upper layer application's requirements and so on. Equation (1) describes a general rule to determine the lifetime for each application message.

$$Lifetime_N = R_{APP} + \frac{L_N}{B_{MEDIA} \times 1024} \text{ (ms)} \qquad (1)$$

where,

$Lifetime_N$: lifetime of application message N (ms),
R_{APP} : delay according to rate control (ms),
L_N : length of application message N (bytes),
B_{MEDIA} : average bit rate of media (bps)

In equation (1), it is assumed that upper layer application adjust the value of R_{APP} according to the state of application buffer, network delay and jitter etc. If there is no rate control, this value will be zero. Through the above equation, real-time IPTV traffic can be transmitted within the specific time limit.

2.2 Multi-streaming

The multi-streaming feature of SCTP gives some benefits not only to application, but also the system/users. Firstly, it can reduce the control overhead for concurrent service.

Figure 1 and 2 describe the operations of typical IPTV applications. Figure 1 employs TCP and UDP as a transport protocols, whereas Figure 2 does SCTP.

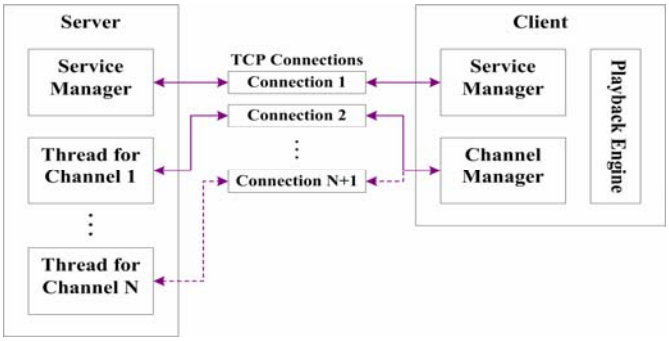

Figure 1. IPTV over TCP

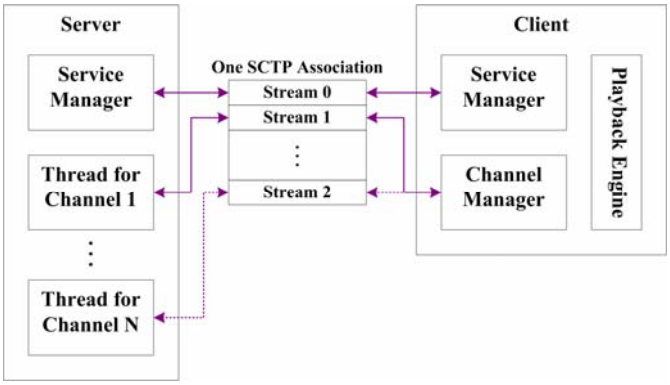

Figure 2. IPTV over SCTP

In Figure 1, the server application must create multiple TCP connections to provide the concurrent multiple IPTV channels. Each thread will serve the independent channel using the corresponding TCP connection. Although the UDP can also be used to transfer the channel media, some additional control sessions will be required to control the media traffic.

In Figure 2, when the SCTP is used for IPTV applications, the server does not have to establish multiple connections, differently from the TCP. All of the server threads can easily transfer the media through the same SCTP association with different stream identifiers. Moreover, only one SCTP connection is active, the internal operations at the server and client sides will become very simple. On the other hand, the feature of SCTP multi-streaming will be used to reduce the impact of HoL blocking that frequently occurs in the TCP connection.

3. Design of IPTV over SCTP

In this section, we briefly give an overview of our implementation of IPTV over SCTP. We implemented the IPTV over SCTP over Linux kernel 2.6.17 with lksctp-tools 1.0.6 [9], sctplib 1.0.5 and socketapi 1.9.0 [10]. We also use the well-known GNU C/C++ compiler 4.1.2 and GNU make 3.8.1.

Figure 3 represents an overall IPTV stream structure of SCTP used in our implementation.

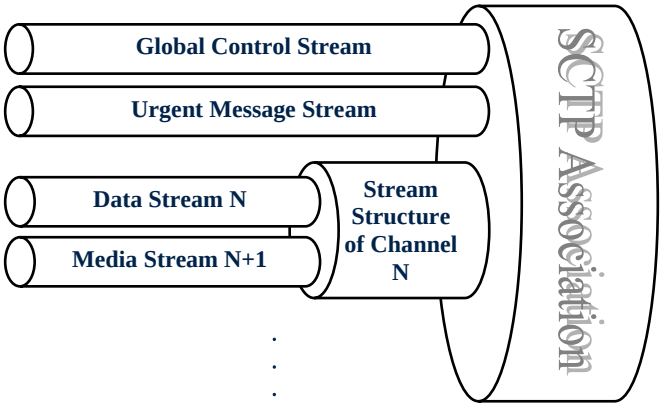

Figure 3. Stream Structure for IPTV Application

In Figure 3, the overall streams can be classified as follows. First, we define a global control stream which is used to start up and take down the SCTP association, inform IPTV channel list or provide the customers with guidelines for service channel. Then, we define the urgent message stream which is used for the server to inform urgent messages in case of emergency, for example, case of service interruption, fire alarm, etc. Finally, we also define some data streams for the IPTV channel services.

Based on the stream structure, we implement the server and client applications separately. The whole system architecture is depicted in Figure 4.

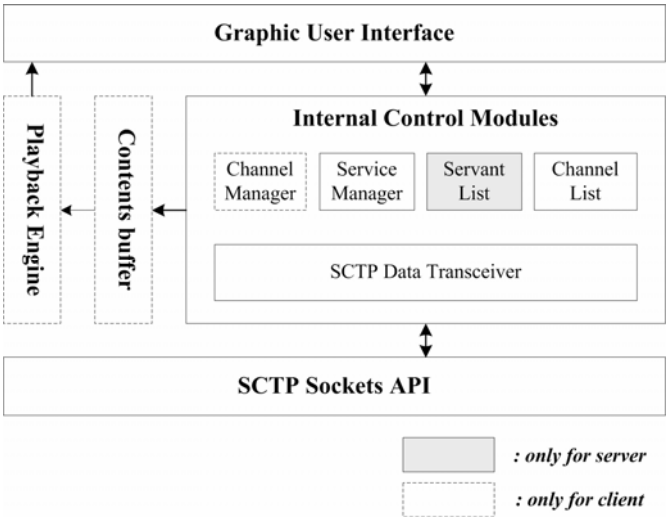

Figure 4. Architecture of IPTV over SCTP

To reduce the overhead of implementation, we use the three external "open source" modules: (i) open source qt4 [11] for graphic user interface, (ii) glib2.0 [12] for internal data structure, and (iii) mplayer 1.0pre8 [13] for playback engine.

In Figure 4, the service manager is responsible to start and close the service session and to deliver the available channel list from server to client. After the session is established normally, the client application will request an IPTV channel through the channel manager, and then the server will reply to the client through the servant module.

Overall communication is performed, as depicted in Figure 2. SCTP data transceiver is responsible to multiplex or demultiplex the upper layer application's messages and transfer to the peer SCTP based on the semantics described in Figure 3.

Figure 5 and 6 describe some snapshots of the screen captures for our implementations. After an SCTP association is established, the client application will request and receive the currently available channel list through the global control stream. When the user selects an appropriate channel by using the graphic interface, a channel request message is transmitted to the server through the data stream of the selected channel. After receiving the request for channel media, the server begins to transfer the media traffic through the data stream. Finally, the client can receive and play the channel media.

Figure 5. IPTV Client that received the available channel list

Figure 6. IPTV Client that is playing a channel movie.

4. Performance Analysis

4.1 Experimental Environment

For the experiment, we construct a test network with the following assumptions:

i. The bit-rate of multimedia data is a constant value (i.e., CBR traffic).
ii. The data rate is usually grater than the bit-rate R. So, a buffer is used to retransmit the traffic without influencing the receiver's play when the data loss occurs.
iii. The message size of each SCTP DATA chunk is a constant value of 1024 bytes.

Figure 7 shows the experimental testbed used for evaluating the performance of PR-SCTP.

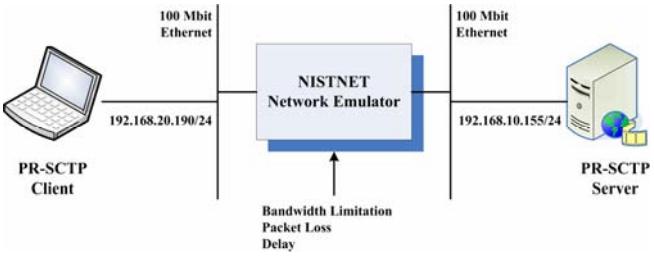

Figure 7. Testbed in the linux platform

The testbed consists of a desktop server, a laptop client and a PC router. Every node runs over the Linux kernel 2.6.16 and lksctp-tools-1.0.6. On the other hand, the NISTNET 3.0a [14] is used to emulate network environment. The server and client applications act as a simple multimedia data sender and receiver respectively, according to the given data rate V. Also the Wireshark network protocol analyzer [15], similar to the well-known Ethereal, is used to capture the traffic in the transmission.

4.2 Test Scenarios

The performance evaluation of the SCTP is accomplished in the following aspects:

i. We compare the transfer completion time of TCP, UDP, SCTP and PR-SCTP for different loss rate.
ii. We compare the transfer-to-reception ratio of PR-SCTP and UDP for different loss rate.
iii. We compare the transfer completion time of PR-SCTP for different loss rate and different stream numbers.
iv. We compare the transfer completion time of TCP and SCTP using three streams for different loss rate.
v. We compare the channel switching delay of TCP and SCTP.

For each scenario, we use MPEG-TS movie file with the size of 2Mbytes, which is transmitted over the bandwidth of 1024 kbps.

4.3 Results and Discussion

Figure 8 shows the performance comparisons for the candidate protocols for different loss rates in the network.

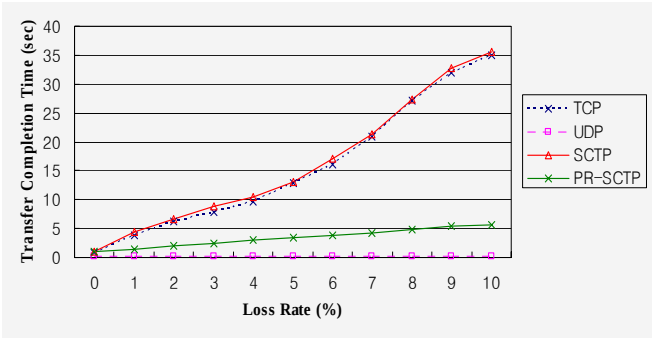

Figure 8. Performance for different loss rates

From the figure, we can see the PR-SCTP and UDP give better performance than the TCP and normal SCTP. The UDP is better performance than the PR-SCTP, since the UDP does not use the flow and error control. That is, the UDP will not recover any lost data packets. Accordingly, we need to compare the UDP and PR-SCTP in terms of the amount of the data packets successfully received at the client side.

Figure 9 compares reception rates of the UDP and PR-SCTP, where the reception rate means the ratio of the successfully received data packets over the totally transmitted data packets.

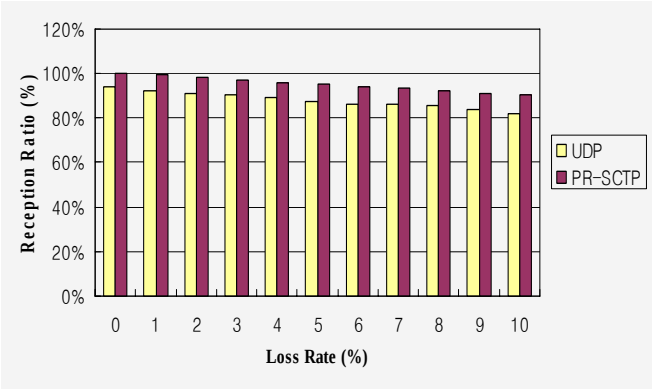

Figure 9. UDP vs. PR-SCTP: successful reception ratio

From the figure, it is shown that the PR-SCTP gives better reception ratio over UDP, which implies that the PR-SCTP provides better Quality of Services (QoS), compared to the UDP. It is noted that the PR-SCTP and UDP give similar throughput, as shown in Figure 8.

Figure 10 shows the performance of the SCTP in terms of the number of the IPTV streams for different loss rates in the networks.

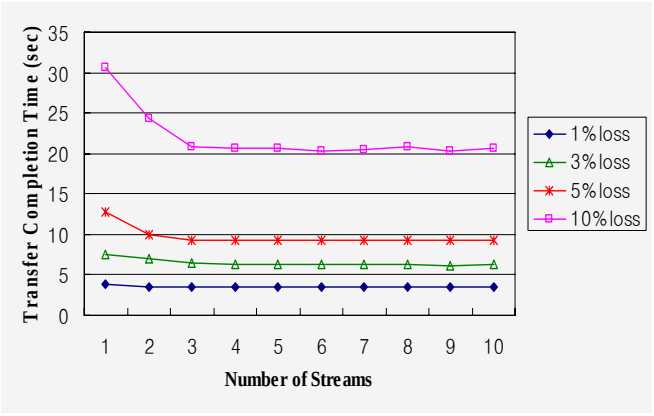

Figure 10. Performance of SCTP for different number of streams

In the figure, we can see that the SCTP gives better performance in the lower loss networks. In particular, the performance gains are similar for the SCTP with 3 streams or more. This means that the optimal number of the streams is 3 in the viewpoint of the performance gain for the number of streams.

Figure 11 shows the HoL blocking effect of the SCTP, compared to the TCP. In the experiments, both TCP and SCTP transport the three IPTV streams over a single connection.

As shown in the figure, the SCTP overcome the HoL blocking problem of the TCP. That is, the performance of the SCTP is better than the TCP, when the multi-streaming feature is used in the SCTP connection. In particular, the gap of the performance between the two schemes gets larger, as the loss rate increases in the network.

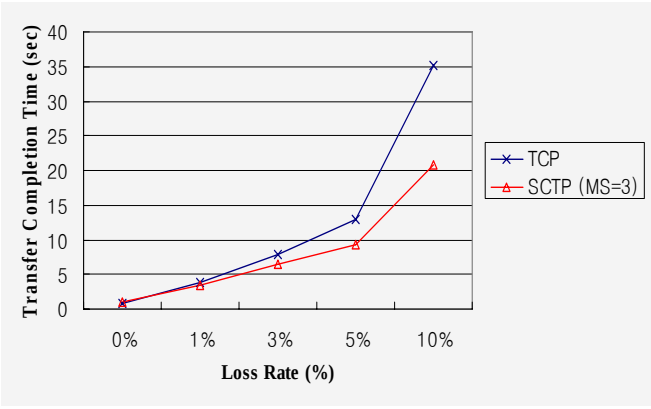

Figure 11. SCTP vs. TCP: HoL blocking effect

Figure 12 compares the channel switching time for the TCP and SCTP. The channel switching time means the gap of the time experiences by the user when it switches to another IPTV stream, during the service period. In the experiment, in case of TCP, a new TCP connection will be established for the new stream channel, whereas the SCTP will maintain the same connection and just switch the stream ID using the SCTP multi-streaming feature.

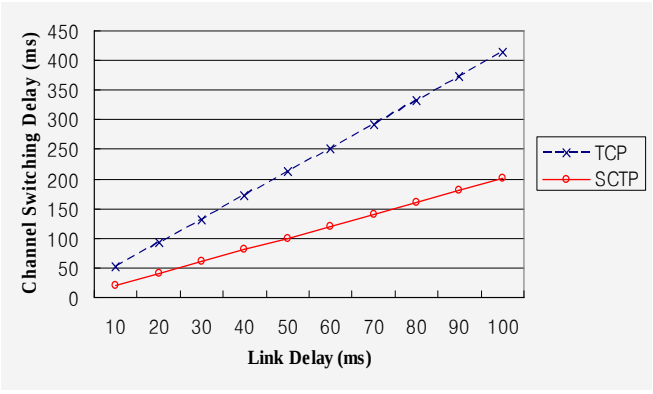

Figure 12. Channel Switching Time for TCP and SCTP

From the figure, we can see the SCTP gives faster channel switching time over the TCP. In particular, the performance gap of the channel switching time gets larger, as the link delay increases in the network. Approximately, the switching delays of TCP correspond to the two times those of SCTP.

5. Conclusions

In this paper, we suggest the use of SCTP for IPTV applications, since the IPTV services can exploit the SCTP distinctive features: multi-streaming for multiple channels and PR-SCTP for real-time services. We implement the IPTV using the SCTP over the Linux testbed, and analyze the performance of the SCTP and compare with the TCP and UDP for a variety of network conditions such as loss rates and link delay. From the experimental results, it is shown that the PR-SCTP provides better goodput than TCP and better quality of services than UDP for real-time IPTV applications. From the viewpoint of the multi-streaming feature, it is shown that the SCTP overcomes the well-known Head-of-Line (HoL) blocking problems when multiple streams are transported over the SCTP. In addition, the multi-streaming SCTP gives faster channel switching time, compared to the TCP.

REFERENCES

[1] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, and T. Taylor, “Stream Control Transmission Protocol,” IETF RFC 2960, October 2000
[2] R. Stewart, Q. Xie, L. Yarroll, K. Poon and M. Tuexen, “Sockets API Extensions for Stream Control Transmission Protocol,” IETF Internet Draft, draft-ietf-tsvwg-sctpsocket-13.txt, June 9, 2006
[3] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad, “Stream Control Transmission Protocol (SCTP) Partial Reliability Extension,” IETF RFC 3758, May 2004
[4] S. Fu and M. Atiquzzaman, “SCTP: State of The Art in Research, Products, and Technical Challenges,” IEEE Communication Magazine, pp. 64-76, March 2004
[5] S. Ladha and P. D. Amer, “Improving Multiple File Transfers Using SCTP Multistreaming,” IEEE International Performance Computing and Communications Conference (IPCCC), Phoenix, Arizona, April 14-17, 2004
[6] P. Natarajan, J. R. Iyengar, P. D. Amer, and R. Stewart, “SCTP: An Innovative Transport Layer Protocol for The Web,” The 15th International Conference on World Wide Web, Edinburgh, Scotland, May 23-26, 2006
[7] H. Wang, Y. Jin and W. Wang, “The Performance Comparison of PRSCTP, TCP and UDP for Mpeg-4 Multimedia Traffic in Mobile Network,” In Proceedings of International Conference on Communication Technology (ICCT), pp. 403-406, April 2003
[8] X. Wang and V.C.M. Leung, “Applying PR-SCTP to Transport SIP Traffic,” IEEE Global Telecommunications Conference (GLOBECOM), St. Louis, MO, November 28, 2005
[9] Linux Kernel SCTP Project, Available from http://lksctp.sourceforge.net/
[10] Sctplib-1.0.5 and socketapi-1.9.0, Available from http://www.sctp.de/
[11] Qt/X11 Open Source Edition, Available from http://www.trolltech.com/
[12] GLib2.0, Available from http://www.gtk.org/
[13] MPlayer 1.0pre8, Available from http://www.mplayerhq.hu/
[14] Nistnet, Available from http://www.itl.nist.gov/div892/itg/carson/nistnet/index.html
[15] Wireshark, Available from http://www.wireshark.org