Analysis of FTP over SCTP and TCP in congested network

Lin-Huang Chang
Grad. Inst. of Networking and Communication Eng., Chaoyang University of Technology
lchang@cyut.edu.tw

Ming-Yi Liao
Dept. of Computer Science and Information Eng., Chaoyang University of Technology
s9427624@cyut.edu.tw

De-Yu Wang
Dept. of Computer Science and Information Eng., Chaoyang University of Technology
dywang@cyut.edu.tw

摘要

Stream Control Transmission Protocol (SCTP) 是新的傳輸層協定，其多重串流機制可避免 Head of Line (HOL) 問題，SCTP 的 Selective Acknowledgement (SACK) 機制可有效率的回報資訊，另一方面，File Transfer Protocol (FTP) 是目前廣為使用於資料傳輸的應用層協定之一。目前已有許多學者從事 SCTP 協定之研究，其中也有學者對於 FTP 透過 SCTP 傳輸進行效能分析，他們主要研究透過 SCTP 多重串流與路徑多宿之特性，能夠為 FTP 資料傳輸帶來的傳輸效能提昇。然而，卻少有研究於壅塞網路之環境下，進行 FTP 透過 TCP 與 SCTP 傳輸效能之比較，因此，我們著重於網路發生壅塞時，對 FTP 透過 SCTP 與 TCP 傳輸之效能進行測量與分析比較。

關鍵詞：SCTP、TCP、檔案傳輸協定、多重串流、Linux kernel。

Abstract

Stream Control Transmission Protocol (SCTP) is a novel transport layer protocol, the feature of multi-streaming in SCTP could avoid the issue of Head Of Line (HOL) blocking, and the Selective Acknowledgement (SACK) mechanism could report the information effectively. On the other hand, File Transfer Protocol (FTP) is one widely used transmission protocol in application layer. There are studies on the SCTP, and part of them on the FTP over SCTP, they studied how the improvement of performance the multi-streaming and multi-homing in SCTP is. However, few papers conducted the comparison of FTP over TCP and SCTP in congested network. Therefore, in this paper we conduct experiments to measure and analyze the performance of TCP and SCTP in congested network.

Keywords: SCTP, TCP, FTP, multi-streaming, Linux kernel

1. Introduction

File Transfer Protocol (FTP) [7] is an application layer protocol which is used widely to deliver files and data. Figure 1 shows the architecture of FTP, the control connection is used to transmit control command and the transmission of data is conducted in the data connection. FTP client builds a new control connection to the FTP server, when the user requests to list the directory or get files from server then, a new data connection is initialized in order to achieve reliable data transmission. The mechanism using control and data connections to carry signals and transmit data respectively is called Out-of-Band (OOB).

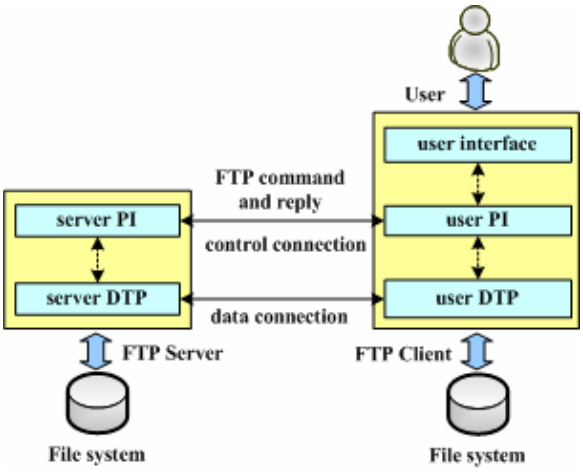

Figure 1. Architecture of FTP.

Transmission control Protocol (TCP) [18] is

one of the current widely used transfer protocols, with the ability of reliable transmitting data, stream oriented, and strict order delivery. However, the three-way hand-shacking of TCP causes Deny of Service (DoS) attacks. The cracker could write easily a network program to build a lot of fake TCP packets which the SYN flag is enabled and IP address, and the fake client will not send SYN-ACK packets back to the server to cause the system to crash. On the other hand, when the packet loss happens during a TCP connection, the packets with large segment numbers have to wait the missed packet to be retransmitted because TCP is strict order delivery. This causes the HOL (Head Of Line) blocking.

Stream Control Transmission Protocol (SCTP) [21] has four-way hand-shacking during the initialization of one association to avoid the security issue in TCP, and it allows many streams in one association called multi-streaming feature which could avoid the HOL blocking issue. Another novel function of SCTP is multi-homing, if the machine has more than one network interface card. SCTP could use these interfaces in an association. Therefore, if one path is failed, SCTP could switch to another path automatically. SACK mechanism is used to report transmission information in SCTP.

The remainder of this paper is organized as follows. In Section 2, we give a brief description about the API and related work of SCTP. We descript our environmental setup in Section 3. Then, in Section 4, we discuss the results of the experiments. Finally, those are our remarks, and future work in Section 5.

2. Background and related research

LKSCTP [10] and SCTPLIB [23] are the realizations of SCTP socket API [22]. LKSCTP is a kernel space implementation of SCTP in Linux operating system, we need to recompile the Linux kernel as a kernel module or within the kernel to enable SCTP support. SCTPLIB is a user space implementation using raw socket, we can use SCTPLIB without recompiling the Linux kernel. In actually, LKSCTP should have more performance than SCTPLIB because the kernel space implementation of LKSCTP does not need to copy data between user space and kernel space.

In the study of F. Siddiqui et al. [4], they have made a comparison of performance between LKSCTP and SCTPLIB, and found that the performance of SCTP implemented in kernel space is better than the one in user space. Therefore, we adapt the LKSCTP API in this paper. In the comparison of TCP and SCTP, A. Kumar et al. [1] compared the performance of TCP and SCTP with the network simulator [16] in MANETs, and they found that the complex functions of SCTP cause worse performance than TCP in the MANET environment. P. Natarajan, et al. [19] studied FTP over SCTP using SCTP multi-streaming, and in their results, they found that the latency of transmitting large files could be reduced with multi-streaming in SCTP. K. Grinnemo et al. [8] have compared the average message transmission delay of ordered delivery with unordered delivery in SCTP and they found that unordered delivery could reduce the average message transmission delay caused by HOL blocking.

M. N. Islam et al. [15] have used SCTPLIB to conduct their experiments to observe the effect of data chunk with different payload sizes, and found that SCTP provides more throughput when transmitting large data chunk than transmitting small data chunk. They conducted that were due to the SACK, congestion control with Appropriate Byte Counting [14], and Limited Transmit mechanism [13] in SCTP.

However, few papers discuss about the FTP over SCTP in congested network, and most of them only conducted experiments with the network simulator or the user level API – SCTPLIB. In this paper, we use LKSCTP to conduct three experiments and port the FTP client and server of open source projects from TCP to SCTP, and use Insane [5] to emulate the congested network. As fair comparison with TCP, we use only a stream in one SCTP association without multi-homing to compare with that with one TCP connection. And we observe the performance of FTP over TCP and SCTP in different congested networks with Wireshark network protocol analyzer [24].

3. Environmental setup

3.1 Network Topology

The network topology is shown in figure 2 with FTP client and server. There is a Linux router between the FTP client and FTP server. The FTP client and FTP server supports SCTP protocol. We set the forward flag in the Linux router to be 1 to allow packet forward. And we use mii-tool [17] which is used to check the link speed of the network interface card to promise

that each network interface card operates in the 100 baseTx Full Duplex mode.

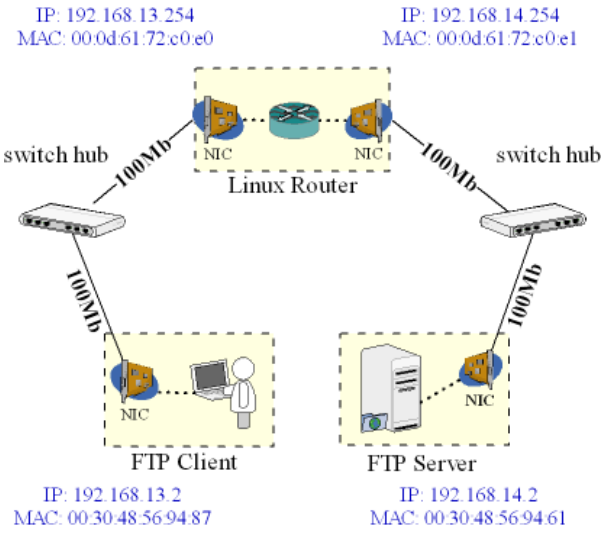

Figure 2. Network topology.

3.2 Testbed

Table 1. Hardwares and systems of testbed.

	FTP Client	FTP Server	Router
CPU	AMD Dual Core 1800 MHz (32bit)	AMD Dual Core 1800 MHz (32bit)	Intel Pentium 4 2800 MHz (32bit)
RAM	2048 MB	2048 MB	512MB
NIC(s)	1 Gigabit Ethernet	1 Gigabit Ethernet	2 Gigabit Ethernet
OS	Ubuntu Desktop	Ubuntu Server	Debian 3.1
Kernel	2.6.17-10	2.6.17-10	2.4.27

In table 1, we list the used hardwares and systems for the testbed, the Ubuntu Linux [25] is used as the operating system of FTP client and FTP server, and the operating system of the router is Debian GNU/Linux [3] which allows us to install the least softwares in an embedded system. In the FTP client and FTP server, we load the default original SCTP kernel module. We choose InetUilts [6] as the FTP client, and vsftpd [20] as the FTP server. InetUtils and vsftpd are open source projects which allow us to port the FTP client and FTP server from TCP to SCTP.

3.3 System tuning

In order to achieve exact measurement, we make fine tuning to the Linux system. At first, we disable the default firewall in each Linux machine, and we set the static Address Resolution Protocol (ARP) [2] of neighbors to avoid additional latency of ARP request and ARP response which will cause wrong processing delay measurement. With ARP static setting, each machine has known the MAC address of its neighbor in advance. For the similar reason, we also set the map of IP address and domain name for each machine in advance to avoid the needless latency due to domain name look-up.

3.4 Packet drop with insane

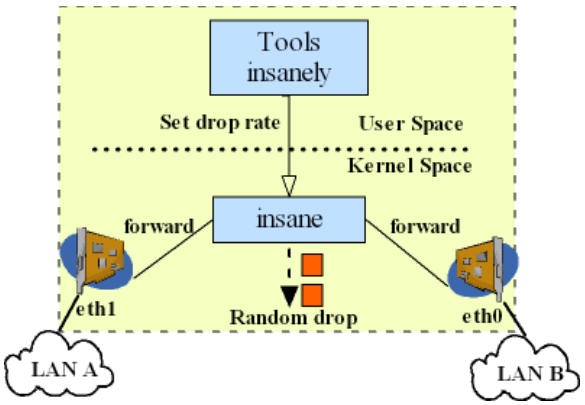

Figure 3. The operation of insane.

Insane [5] is a virtual network interface which operates in Linux. As shown in figure 3, all packets which come from LAN A or LAN B have to pass through the insane virtual network interface. And we could use the user space tool insanely to control insane about how much drop rate we want to set. Insane is a kernel space module for Linux kernel version 2.2 to 2.3, we port the insane to Linux kernel 2.4 and install insane in our Linux router. We use insane to change the random drop rate of the router to emulate different congested networks.

4. Results and analyses

4.1 Experimental scenario

The purpose of our experiments is to compare the performance of FTP over TCP and SCTP in congested network. For fair comparison, we only use one single stream in a SCTP association without multi-homing to compare with one TCP connection. The version of TCP in Linux is BITCP [9] which is similar to TCP-Reno [12]. We conduct three experiments according to different transmitted file sizes and observe the

effect of different packet drop rates on the throughput for cases of TCP, TCP with SACK [11] and SCTP mechanisms. In the results, T_{tcp} denotes the transmission time of TCP, T_{sack} denotes the transmission time of TCP with SACK, T_{sctp} denotes the transmission time of SCTP, and Drop Rate means the router packet drop rate.

4.2 Measurement and analysis

In our experiments, all the packet drop rates of the router are varied from 0 %, 5 % to 10 % to emulate different levels of the network congestion. We use a FTP client to download a file from the FTP server in passive mode, and measure the transmission time with Wireshark network protocol analyzer.

In the first experiment, we download a file whose size is 32MB from the FTP server to the FTP client. The results are shown in table 2. When the drop rate is equal to 0%, the transmission time of TCP, TCP with the increase of dropping rate, the transmission time for increase of SCTP increases from 3.01, 3.75 to 8.59 for 0%, 5%, and 10% drop rates, respectively. As comparison, the transmission times for both TCP and TCP with SACK increase significantly with the increase of dropping rate, especially for 10% drop rate. Even TCP with SACK which provides advantage of SACK algorithm in TCP, the performance of TCP with SACK shows much less than SCTP when drop rate equals to 10%.

Table 2. Transmission time of 32MB file size

Drop Rate	T_{tcp}	T_{sack}	T_{sctp}
0 %	3.01	3.05	3.01
5 %	32.70	20.99	3.75
10 %	136.44	102.96	8.59

(unit: second)

To understand the results above, we observe the packets using Wireshark. When the router drops the packet which emulates the happening of packet loss, the node will request for retransmission of the packets. We observe the total number of packets in the case of 10% drop rate. There are 44669 packets in TCP, 44559 packets in TCP with SACK, and 41000 packets in SCTP. The main reason that the number of SCTP packets is less than TCP is the SACK mechanism in SCTP. Because more ACK packets have to be sent to pass through the router for TCP, the drop probability is higher. Although both of SCTP and TCP with SACK have SACK mechanism, there are a maximum of three SACK blocks in TCP with SACK, and there are unlimited blocks in SCTP. And this factor causes the number of packets transmitted for TCP with SACK being more than that of SCTP. On the other hand, the mechanism of increasing congestion window in SCTP based on the number of bytes of acknowledged rather than the number of ACKs called Appropriate Byte Counting could improve the performance. In figure 4, we observe that one TCP node requests for sequence number 1458137 with five ACK packets. This, therefore, pushes the fast retransmission in TCP.

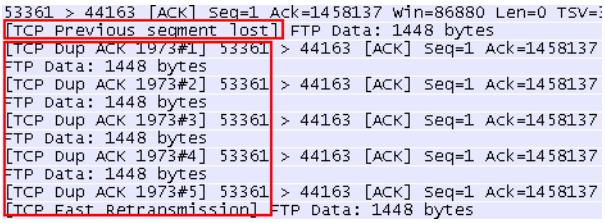

Figure 4. TCP segment lost

In the second experiment, we download a 4MB file from the FTP server to the FTP client, and use the same drop rate on the first experiment. The results are shown in table 3. We find the similar results with previous experiment. Because 4MB is one eights of 32MB, the results of the second experiment almost show the equimultiple with those of the first experiment.

Table 3. Transmission time of 4MB file size

Drop Rate	T_{tcp}	T_{sack}	T_{sctp}
0 %	0.35	0.37	0.36
5 %	3.93	1.51	0.81
10 %	18.53	12.19	1.53

(unit: second)

Table 4. Transmission time 512KB file size

Drop Rate	T_{tcp}	T_{sack}	T_{sctp}
0 %	0.053	0.052	0.051
5 %	0.50	0.29	0.37
10 %	2.32	1.386	0.86

(unit: second)

In the third experiment, the file size is 512 KB, and the results are shown in table 4. Because the file size is relatively small, it will complete the transmission with less packet loss than previous experiments. Therefore, in this case, we could

find that there is only a little effect in the transmission time for different sizes of transmitted packets among TCP, TCP with SACK and SCTP mechanisms.

5. Conclusions

We ported the FTP client and FTP server of open source projects from TCP to SCTP protocol and we have to port insane to Linux kernel 2.4 to emulate the congested network. We conducted three experiments to compare and analyze the performance of FTP over TCP and FTP over SCTP when the network is in congestion. In the results, we found that the throughput of FTP over SCTP is more than FTP over TCP. Especially when we want to transfer data as large as 4MB or 32MB and the drop rate of the router is as high as 10%, SCTP could deliver more data than TCP during the same time period. We compare the SCTP with TCP fairly without applying the functions of multi-streaming and multi-homing in SCTP which is scanty in TCP. The results have shown that the congestion control and SACK mechanism brings large advantages for SCTP. Through the results of experiments, we could know that even if we only use a stream in one association, SCTP still provides better performance than the implemented TCP in congested network. In the future, we will apply the features of multi-streaming and multi-homing in SCTP to other applications.

References

[1] A. Kumar, and L. Jacob, “SCTP vs TCP : Performance Comparison in MANETs,” IEEE LCN 2004.

[2] David C. Plummer, “An Ethernet Address Resolution Protocol”, RFC826, IETF, Nov. 1982.

[3] Debian GNU/Linux, http://www.debian.org/

[4] F. Siddiqui and S. Zeadally, “SCTP Multihoming Support for Handoffs across Heterogeneous Networks,” In Proc. of the IEEE CNSR 2006, pp. 243-250, May 2006.

[5] Insane Virtual Network Interfaces, http://www.linux.it/~rubini/docs/vinter/vinter.html

[6] InetUtils, http://www.gnu.org/

[7] J. Postel, and J. Reynolds, “File Transfer Protocol”, RFC 765, IETF, Oct. 1985.

[8] K. J. Crinnemo, and T. Andersson, “Performance Benefits of Avoiding Head-of-Line Blocking in SCTP ,” Proceedings of IEEE on ICAS/ICNS, pp. 44-51, 2005.

[9] L. Xu, K. Harfoush, and I. Rhee, “Binary Increase Congestion Control for Fast Long Distance Networks,” http://www.csc.ncsu.edu/faculty/rhee/export/bitcp.pdf

[10] Linux Kernel Stream Control Transmission Protocol Project, http://lksctp.sourceforge.net

[11] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, “TCP Slective Acknowledgement Options,” RFC 2018, IETF, Oct. 1996.

[12] M. Allman, V. Paxson, W. Stevens, “TCP Congestion Control”, RFC 2581, IETF, Apr. 1999.

[13] M. Allman, H. Balakrishnan, and S. Floyd, “Enhancing TCP’s Loss Recovery Using Limited Transmit,” RFC 3042, IETF, Jan. 2001.

[14] M. Allman, “TCP Congestion Control with Appropriate Byte Counting,” RFC 3465, IETF, Feb. 2003.

[15] M. N. Islam, and A. Kara, “Throughput Analysis of SCTP over a Multi-homed Association,” Proceedings of IEEE International Computer and Information Conference, pp. 110-116, Sep. 2006.

[16] NS2, Network Simulator Version 2, http://www.isi.edu/nsnam/ns/

[17] Net-tools Project, http://www.tazenda.demon.co.uk/phil/net-tools

[18] Postel, J. B, “Transmission Control Protocol,” RFC 793, IETF, Sep. 1981.

[19] P. Natarajan, P. D. Amer, R. W. Bickhart, and S. Ladha, “Improving multiple file transfer using SCTP multistreaming,” Proceedings of IEEE International Performance Computing and Communications Conference, 2004.

[20] Vsftpd project, http://vsftpd.beasts.org/

[21] R. Steward, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, “Stream Control Transmission Protocol,” RFC2960, IETF, Oct. 2000.

[22] R. Steward, Q. Xie, L. Yarroll, K. Poon, M. Tuexen, “Sockets API Extensions for Stream Control Transmission Protocol (SCTP) ,” draft-ietf-tsvwg-sctpsocket-14, IETF, Dec. 2006.

[23] SCTPLIB - the user level implementation, http://www.sctp.de/sctp-download.html

[24] The Wireshark Network Protocol Analyzer, http://www.wireshark.org

[25] Ubuntu Linux, http://www.ubuntu.com/