A comparison of TCP and SCTP performance using the HTTP protocol

Henrik Österdahl (henost@kth.se), 800606-0290, D-01

Abstract

This paper discusses using HTTP over SCTP as an alternative to the traditional HTTP using TCP as the transport layer protocol. Some performance tests using modified versions of thttpd and httperf are performed and the results indicate that SCTP using multiple streams can improve performance on heavily congested lines compared to SCTP using only a single stream.

1 Introduction

This document describes HTTP performance over the TCP protocol compared to HTTP performance over the SCTP protocol. SCTP is a modern transport protocol, which in theory overcomes many of the drawbacks of primarily the TCP protocol, but also those of UDP. In the next section I will present all three Internet transport layer protocols and the motivation behind using SCTP as an alternative to TCP in web serving. The Tests-section describes the tests performed and the results and finally I will conclude with my thoughts on the results.

2 Background

Until recently there has been two official transport layer protocols used on top of the IP protocol. SCTP [4] adds a third protocol and I will describe all three briefly in this section.

2.1 UDP

The User Datagram Protocol (UDP) is a simple transport protocol. UDP is called a connectionless, unreliable protocol. It contains no flow control or error control. In essence it is only a transport level abstraction of the network layer protocol (IP). Typical applications for UDP include applications with internal flow- and error-control mechanisms, multicasting, SNMP, RIP and streaming media [9].

2.2 TCP

The Transmission Control Protocol (TCP) is probably the most widely used transport protocol on the Internet. It is called a connection-oriented, reliable protocol. Unlike UDP it includes features for error-control and flow-control. It also includes some level of sender-initiated congestion-control. Some uses of TCP include web serving, file transfers and lots of other connection-dependent applications.

2.3 SCTP

The Stream Control Transmission Protocol (SCTP) is a new reliable, message-oriented transport layer protocol. It is the latest addition to the transport layer protocols in the TCP/IP stack. Since it is new, no "killer application" has emerged, but it is designed with various telephony applications in mind, for instance IP telephony (H.323, SIP) and others who need more sophisticated service than the aging TCP can provide. SCTP is message-oriented, which means it preserves message boundaries (as UDP does). It is also reliable, which means it detects lost data, duplicate data and out-of-order data and it also contains flow- and congestion-control (as TCP does). It is a well designed protocol which combines the features of UDP and TCP to a protocol which (at least in theory) is much better than the two older ones. Additionally, it features multi-homing (using multiple network interfaces and routes interchangeably)

and multiple streams support (embedding several logical streams in one connection/association).

3 Tests

In this section the tests performed are described and analysed.

3.1 Why HTTP over SCTP?

It is important to understand how an HTTP client downloads information from a web page. Using HTTP 1.0 a client first starts a TCP connection and downloads the HTML code describing a page, then opens separate connections to download images and other information. In figure 1 this equals to four total connections and in figure 2 it equals to ten total connections.

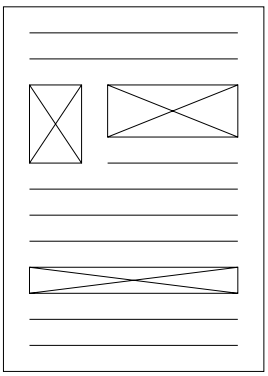

Figure 1: A typical news home page

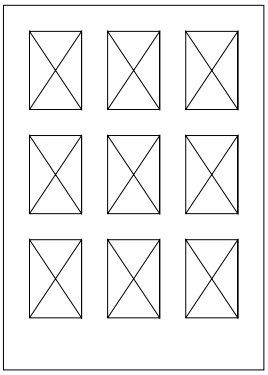

Figure 2: A typical image gallery

It goes without saying that this kind of behaviour induces plenty of overhead for pages containing much auxilliary information besides text data.

HTTP version 1.1 tries to resolve this by trying to minimize the amount of TCP connections used. In particular it can use what is called persistent connections to accomplish this. On these persistent connections, something called

pipelining can be used, which allows HTTP clients to issue multiple requests without waiting for the response of pending requests. This can improve HTTP performance over TCP. It is all nice and such, but it misses something important.

Using HTTP 1.0 every request is easily separated from another, since separate TCP connections are used for every URL. In HTTP 1.1 using pipelining this nice feature is inherently destroyed and it is up to the HTTP client/web browser to take care of asynchronous responses and the like.

HTTP as such is not limited to TCP as its transport layer. The standard [7] notes that any network and transport layer can be used as long as it provides a reliable connection. SCTP is such a reliable transport layer.

As already stated above SCTP includes a feature called "multiple streams" support. The original purpose of this paper was to see if the multiple streams feature could be used to increase the throughput of HTTP and if the performance of SCTP is comparable or the better choice for web browsing/serving than TCP and if the multiple streams feature will help on networks where many packets are dropped.

3.2 Test setup

3.2.1 Hardware and operating system

All tests were performed on a Pentium 4, 2.4 GHz with 512 MB RAM on Fedora Core 3 using Linux kernel version 2.6.11.6 with SCTP support (lksctp) compiled in and support libraries in user space. Since only one computer with SCTP support was available, all tests were run over the loopback interface. Simulation of dropped packets was acquired through the "nth"-extension for the IP Tables packet filter. The "nth" extension can be configured to filter every "nth" IP packet. In this case the operation was "DROP".

3.2.2 Benchmarking software

Since HTTP is a widely used protocol and SCTP is claimed to be suitable for it, HTTP was chosen to be the tested protocol.

The web server used is *thttpd*, which is a very thin web server. According to its home page it is "a simple, small, portable, fast, and secure HTTP server" [2]. The HTTP client used is *httperf* [1]. It is a widely used benchmark because it generates good, typical (and customizable) HTTP traffic. Unfortunately it does not support HTTP pipelining, which would allow every TCP connection to be used more efficiently.

The reason I chose to use the above tools for this paper is simple. Both programs were adapted to SCTP by Elvis Pfützenreuter for use in his master's thesis on SCTP [3] and are downlodable from his home page. Also, the tests performed and described in this document are somewhat similar to his tests, mainly because they seem relevant.

3.3 Tests and data

Tables 1, 2, 3 and 4 show the test results of a number of runs on the computer and software mentioned above. The tests were run five times each and the

File size	No dropped packets	Every 10th dropped	Diff
100 bytes	10.2	9.04	-11.4 %
1 KB	33.2	21.0	-36.7 %
10 KB	262.7	159.68	-39.2 %
100 KB	2556.82	1945.28	-23.9 %
1 MB	26055.88	10051.76	-61.4 %
10 MB	155750.18	95392.34	-38.8 %

Table 1: TCP, 50 requests, 25 reqs/sec, timeout 15 secs

File size	No dropped packets	Every 10th dropped	Diff
100 bytes	10.2	2.44	-76.1 %
1 KB	33.2	7.26	-78.1 %
10 KB	262.6	27.0	-89.7 %
100 KB	2554.1	786.96	-69.2 %
1 MB	25855.6	5301.48	-79.5 %
10 MB	35203.12	13936.1	-60.4 %

Table 2: SCTP - 1 stream, 50 requests, 25 reqs/sec, timeout 15 secs

presented result is the average of each run. All results show throughput in KB/s.

3.4 Results

This subsection tries to dig through the test results and see what actually happened.

3.4.1 Expectations

These are some of the expectations I had before the tests:

- Since the Linux TCP/IP stack is widely used and has been around for many years I expected it to be very fast and well behaved compared to the SCTP implementation (lksctp) I tested [8].
- Dropping every 10th IP packet should simulate a heavily congested network.
- Performing 25 HTTP requests per second should simulate a moderately loaded web server.
- Simulating a moderately loaded web server should not steal too much processing power, so running the client and the server on the same host would be okay.
- Increasing the number of concurrent streams should increase overall HTTP throughput for SCTP, thus would be suitable for use in WWW browsing.

File size	No dropped packets	Every 10th dropped	Diff
100 bytes	122.6	5.16	-95.8 %
1 KB	399.06	13.38	-96.6 %
10 KB	3120.52	162.58	-94.8 %
100 KB	4273.96	1038.58	-75.7 %
1 MB	31169.0	2185.44	-93.1 %
10 MB	50844.4	15737.22	-69.0 %

Table 3: SCTP - 10 streams, 50 requests, 25 reqs/sec, timeout 15 secs

File size	No dropped packets	Every 10th dropped	Diff
100 bytes	944.36	181.38	-80.1 %
1 KB	63.5	154.42	+43.2 %
10 KB	1796.5	127.175	-92.3 %
100 KB	16393.72	208.62	-98.7 %
1 MB	38824.74	196.6	-99.5 %
10 MB	47137.08	15474.63	-67.2 %

Table 4: SCTP - 50 streams, 50 requests, 25 reqs/sec, timeout 15 secs

3.4.2 Actual results

Comparing the raw (no dropped packets) TCP results to using SCTP with only one stream, the results are identical up to a certain point. On the 10 megabyte file TCP outperforms SCTP by far. Before that point there are probably very few concurrent connections at all. Since the loopback connection uses no copper wire, its theoretical throughput is only limited by the hosting operating system. This went as expected.

In the case where every 10th packet was dropped, things got more interesting. While it is hard to make a statistical analysis of the data, it is anyhow clear that the best and worst results for TCP were 11 % and 61 % worse in throughput (table 1), respectively. The corresponding values for SCTP were 60 % and 89 % (table 2). The obvious conclusion is that LKSCTP using one stream is very sensitive to heavily congested networks. So, TCP currently performs better than SCTP in the standard mode, but what about multiple streams?

In tabled 3 and 4 the results for ten and fifty streams are presented. We note at lower speeds that the requests are divided on the available streams and that shows up in the results. A somewhat strange result is the positive difference for the 1 kilobyte file using fifty streams (and also the low result without dropped packets).

Another interesting results is that using fifty streams on the largest files does not improve the results very much, which indicates that the number of streams used must be carefully choosen.

One can also not that the performance drop for SCTP using multiple streams when every 10th packet is lost is severe, which was not at all expected.

In any case, using ten streams seems to be a better choice than using fifty streams for larger files when many packets are dropped.

3.5 Error sources

Since these were not performed in a lab environment, many things can be improved, including the following:

3.5.1 Running on the loopback interface

Some of the most strange-looking results are probably due to my using the loopback connection in the tests. In a "real environment" one would most likely use several computers on a high-speed network.

3.5.2 Test choices

As HTTP is used for all kinds of file transfers (even on the web), I am fairly sure the tested file sizes are okay. What can be debated is if the test case - 50 requests over 2 seconds, 15 second timeout - is useful. As I have no prior experience in measuring HTTP/TCP/SCTP performance I have no idea if it is the best choice, but it seems reasonable.

4 Conclusions

Apart from some strange results, some conclusions can be drawn.

For completely reliable tests, they should be run in an as close to reality environment as possible, which was not done here.

It can be noted that SCTP probably is not as fast as TCP if used as a mere "search-and-replace" substitute for TCP. Most probably new software should be designed with the SCTP protocol as a fundamental part.

However, I do think that web browsing can be efficiently performed using HTTP over SCTP, once all implementations have stabilized. I was not able to find out if there are any ongoing projects creating such a solution, but there definitely should be.

Elvis Pfützenreuter has written a master's thesis [3] on SCTP performance (currently only the results are published), which probably is a good read for those interesting in finding out more. In the thesis he shows that SCTP actually can be used for web browsing.

References

[1] David Mosberger, "httperf - A Tool for Measuring Web Server Performance".
(http://www.hpl.hp.com/personal/David_Mosberger/httperf.html) (modified 16 October 2003).

[2] ACME Labs, "thttpd".
(http://www.acme.com/software/thttpd/) (modified 19 May 2005).

[3] Elvis Pfützenreuter, "Master's dissertation (unpublished)".
(http://www.altoriopreto.com.br/epx/mestrado/index_en.php)
(visited May 15 2005).

[4] The Internet Society, “Stream Control Transmission Protocol”. (http://www.ietf.org/rfc/rfc2960.txt) (October 2000).

[5] L. Coene, “Stream Control Transmission Protocol Applicability Statement”. (http://www.ietf.org/rfc/rfc3257.txt) (April 2002).

[6] J. Stone, R. Stewart, D. Otis, “Stream Control Transmission Protocol (SCTP) Checksum Change”. (http://www.ietf.org/rfc/rfc3309.txt) (September 2002).

[7] The Internet Society “Hypertext Transfer Protocol – HTTP/1.1”. (http://www.ietf.org/rfc/rfc2616.txt) (June 1999).

[8] “Linux Kernel Stream Control Transmission Protocol (lksctp) project ”. (http://lksctp.sourceforge.net/) (visited May 2005).

[9] Behrouz A. Forouzan, “TCP/IP Protocol Suite, 3rd ed.” (2005).