Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2009

P. Natarajan
P. Amer
J. Leighton
University of Delaware
F. Baker
Cisco Systems

October 30, 2008

Using SCTP as a Transport Layer Protocol for HTTP
draft-natarajan-http-over-sctp-00.txt

Status of this Memo

Abstract

Hyper-Text Transfer Protocol (HTTP) [RFC2116] requires a reliable transport for end-to-end communication. While historically TCP has been used for this purpose, this document proposes an alternative -- the Stream Control Transmission Protocol (SCTP) [RFC4960]. Similar to TCP, SCTP offers a reliable end-to-end transport connection to applications. Additionally, SCTP offers other innovative services unavailable in TCP. The objectives of this draft are three-fold: (i) to highlight SCTP services that better match HTTP's needs than TCP, (ii) to propose a design for persistent and pipelined HTTP 1.1

transactions over SCTP's multistreaming service, and (iii) to share some lessons learned from implementing HTTP over SCTP. Finally, open issues warranting more discussion and/or investigation are listed.

Table of Contents

1. Introduction 3
2. Conventions 3
3. SCTP Services for HTTP-based Applications 3
4. Designing HTTP over SCTP Streams 5
4.1. Number of SCTP Streams 7
4.2. Mapping HTTP Transactions to SCTP Streams 7
5. Lessons Learned from Implementing HTTP over SCTP 8
5.1. Avoiding Dependencies in Message Transmission 8
5.2. Order of Pipelined Requests and Responses 8
5.3. Benefits for Progressive Images 9
6. Open Issues 9
6.1. How does a Web client decide between TCP vs. SCTP? 10
6.2. TCP-SCTP Gateway 10
6.3. SCTP and NATs 10
7. Acknowledgments 10
8. References 11
8.1. Normative References 11
8.2. Informative References 11
Authors' Addresses 12
Intellectual Property and Copyright Statements 14

1. Introduction

SCTP was originally developed to carry telephony signaling messages over IP networks. With continued work, SCTP evolved into a general purpose transport protocol. Similar to TCP, SCTP offers a reliable, full-duplex, congestion and flow-controlled transport connection. Unlike TCP, SCTP offers other innovative services including multistreaming, multihoming, partial-realiability, and message-oriented data transfer. This document highlights some of the SCTP services that are better suited to HTTP's needs than TCP services.

SCTP's multistreaming service is perhaps the most beneficial SCTP service for HTTP. SCTP streams are logically separate data streams within an SCTP "association" (analogous to a TCP connection). Independent HTTP transactions, when transmitted over different SCTP streams, can be delivered to the application without inter-transaction head-of-line (HOL) blocking. Emulation results show that SCTP streams eliminate HOL blocking and significantly improve web response times [N2008]. This document presents our design for persistent and pipelined HTTP 1.1 transactions over SCTP streams, and some of the lessons learned from implementing this design in the Apache server and Firefox browser.

Finally, this document lists some of the open issues that require further discussion and/or investigation within the httpbis community.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. SCTP Services for HTTP-based Applications

Similar to TCP, SCTP provides a reliable and in-order data transfer service to HTTP. Additionally, SCTP provides other services unavailable in TCP. These services are summarized below. The HTTP over SCTP design proposed in Section 4 utilizes only a subset of these SCTP services. The authors believe that other SCTP services listed below MAY help HTTP, but the details remain unclear at this time.

1. SCTP Multistreaming Avoids Head-of-Line (HOL) Blocking.

   An SCTP stream is a unidirectional data flow within an SCTP

   association. Each SCTP stream has its own sequencing space; data arriving in-order within a stream is delivered to the application without regard to the relative order of data arriving on other streams. When independent HTTP transactions are transmitted over different SCTP streams, these transactions are delivered to the application without inter-transactoin HOL blocking. Section 4 discusses the benefits of HTTP over SCTP streams in more detail.

2. Four-way Handshake during Association Establishment.

   To protect an end host from SYN-flooding DoS attacks, SCTP's association establishment involves a four-way handshake with a cookie mechanism. Since data transfer can begin in the third leg, the four-way handshake does not delay data transmission any further than TCP's three-way handshake for connection establishment.

3. No Maximum Segment Lifetime (MSL) during Association Termination.

   SCTP's association termination does not involve a TIME_WAIT state [RFC0793], since the Initiation and Verification tags help to associate SCTP Protocol Data Units (PDUs) with the corresponding SCTP associations [RFC4960]. Note that TCP's TIME_WAIT state increases memory and processing overload at a busy web server [FTY1999].

4. SCTP Multihoming for Improved Fault Tolerance.

   Unlike TCP and UDP, an SCTP association can bind multiple IP

addresses at each peer. While an SCTP sender transmits data to a single primary destination IP address, the sender concurrently tracks the reachability of other destination addresses for fault-tolerance purposes. If the primary address becomes unreachable, an SCTP sender seamlessly migrates data transmission to an alternate active destination address. Multihomed clients and/or web servers will automatically benefit from greater fault-tolreance by using SCTP.

5. Preserving Application Message Boundaries.

Similar to UDP, SCTP offers message-oriented data transfer. SCTP preserves application message boundaries; messages are delivered in their entirety to the receiving application. Applications using SCTP do not require explicit message delimiters, which simplifies message parsing. However, the advantages of SCTP's message-oriented data transmission service to HTTP is unclear, and the proposed HTTP over SCTP design in Section 4 does not exploit this SCTP service. To preserve message boundaries, SCTP

employs a fragmentation and reassemebly algorithm. This algorithm creates dependencies in message transmission, discussed further in Section 5.1, [N2008].

6. Partial Reliability.

Reference [RFC3758] describes PR-SCTP, an extenstion to [RFC4960], that enables partially reliable data transfer between a PR-SCTP sender and receiver. In TCP and plain SCTP, all transmitted data are guaranteed to be delivered. Alternatively, PR-SCTP gives an application the flexibility to notify how persistent the transport sender should be in trying to communicate a particular message to the receiver. An application MAY specify a "lifetime" for each message. A PR-SCTP sender tries to transmit the message during this lifetime. Upon lifetime expiration, a PR-SCTP sender discards the message irrespective of whether or not the message was successfully transmitted and/or acknowledged. This timed reliability in data transfer may be useful in applications that regularly generate new data that obsoletes earlier data, for example, online gaming application in which a player frequently generates new position coordinates or other data with ephemeral significance. The

proposed HTTP over SCTP design in Section 4 currently does not make use of this PR-SCTP service.

7. Unordered Data Delivery.

Similar to UDP and unlike TCP, SCTP offers unordered data delivery service. An application message, marked for unordered delivery, is delivered to the receiving application as soon as the message arrives at the SCTP receiver. Unlike UDP, SCTP provides reliability for unordered data. Note that a single SCTP association can transfer both ordered and unordered messages. The proposed HTTP over SCTP design in Section 4 does not make use of this SCTP service.

4. Designing HTTP over SCTP Streams

In this document, an HTTP GET request (or response) is considered independent when its application-level processing does not depend on the availability of other HTTP GET requests (or responses). The primary objective of our design is to exploit SCTP's multistreaming service to avoid HOL blocking between independent HTTP transactions.

Note that HTTP transctions do not experience HOL blocking when either (i) each HTTP transaction is transmitted over a different TCP connection (HTTP 1.0) [RFC1945], or (ii) multiple HTTP transactions

are transmitted in a non-pipelined fashion over a single persistent TCP connection [RFC2616]. Consequently, we do not expect SCTP's multistreaming to improve response times for an HTTP 1.0 transfer or a non-pipelined HTTP 1.1 transfer. Nonetheless, these HTTP transfers may benefit from other SCTP features such as multihoming, four-way association establishment handshake etc., mentioned in Section 3. Note that a client or server implementing HTTP 1.0 or non-pipelined HTTP 1.1 over TCP can be trivially mapped to work over SCTP by creating an SCTP socket instead of a TCP socket [I-D.ietf-tsvwg-sctpsocket].

An HTTP transaction may be HOL blocked by another independent HTTP transaction only when these transactions are transmitted in a pipelined fashion over a single TCP connection. Transferring these transactions over different streams of a single SCTP association

elimiates the inter-transaction HOL blocking. Emulation results show that persistent and pipelined HTTP 1.1 transfers over a single multistreamed SCTP association experience better response times when compared to similar transfers over a single TCP connection. The difference in TCP vs. SCTP response times increases and is more visually perceivable in high latency and lossy browsing conditions, such as those found in the developing world [NAS2008].

Apart from improving response times, SCTP streams may also reduce setup and memory costs at a web server/cache/proxy. To reduce HOL blocking, web clients open muliple TCP connections to download independent HTTP transactions from the same server. In contrast, a web client using SCTP eliminates HOL blocking by simply increasing the number of streams within a single SCTP association. Each TCP connection or SCTP stream incurs additional setup and memory overhead at both the client and server. However, the costs associated with a new SCTP stream are in general lower than those associated with a new TCP connection, and the cost gains from using SCTP increase as the number of web clients increase. The exact difference in TCP vs. SCTP resource requirements depends on the respective protocol implementations [N2008].

Two guidelines govern the HTTP over SCTP streams design discussed below: (i) make no changes to the existing HTTP specification (such as the URI syntax), to reduce deployment issues, and (ii) minimize SCTP-related state information at the server so that SCTP multistreaming does not further contribute to the server being a performance bottleneck. Detailed discussions on various design decisions can be found in [N2008]. The two components of this design are discussed next.

4.1. Number of SCTP Streams

SCTP streams are uni-directional; inbound and outbound streams carry data to and from each end point, respectively. Each inbound or outbound stream incurs additional memory overhead in the SCTP Protocol Control Block, and this overhead depends on the SCTP implementation. The number of inbound or outbound SCTP streams is negotiated during the association establishment phase (Figure 1).

Before association establishment, the number of inbound or outbound streams may be modified by using appropriate SCTP socket options [I-D.ietf-tsvwg-sctpsocket]. The stream "reset" functionality allows for re-negotiating the number of streams after association establishment [I-D.stewart-tsvwg-sctpstrrst]. When using SCTP for HTTP, an SCTP association MAY employ any number of inbound or outbound streams (up to 65,536 [RFC4960]). However, for every outbound SCTP stream with id *a* on which the client transmits requests, there MUST be a corresponding inbound stream with id *a*. Typically, this is achieved by opening an SCTP association with equal number of inbound and outbound streams.

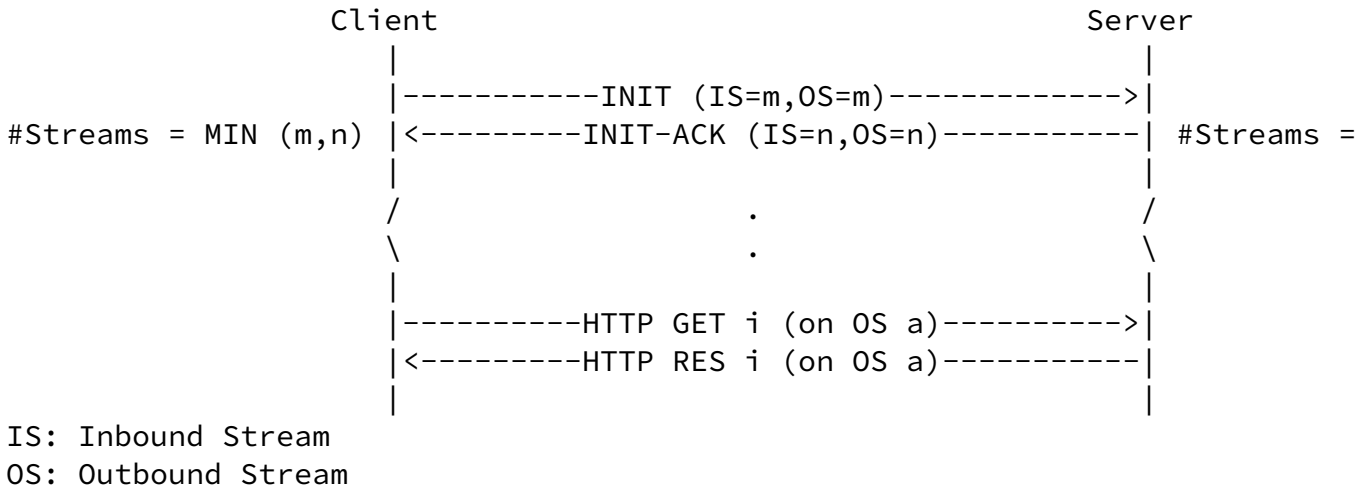

Figure 1: HTTP over SCTP Streams

4.2. Mapping HTTP Transactions to SCTP Streams

To avoid incurring additional processing overhead at the web server, a web client determines the SCTP stream number on which each HTTP transaction is transmitted. In the example shown in Figure 1, the web client maps HTTP transaction *i* to SCTP stream *a*. The client transmits HTTP request *i* on the client's outbound (server's inbound) SCTP stream *a*. The web server transmits the corresponding response on the server's outbound (client's inbound) SCTP stream *a*.

The sctp_sendmsg and sctp_recvmsg APIs, respectively, can be used to

transmit data on a particular SCTP outbound stream, and determine the SCTP inbound stream number on which an application message was received.

When the number of available SCTP streams is greater than or equal to the number of HTTP transactions, a web client SHOULD NOT pipeline transactions intra-stream, i.e., each HTTP transaction SHOULD be mapped to a different SCTP stream. When the number of available SCTP streams is less than the number of HTTP transactions, the web client MAY employ a scheduling policy to pipeline transactions intra-stream. Our implementation employs a round-robin scheduling policy, where HTTP transactions are mapped to available SCTP streams in a round-robin fashion. Other scheduling policies MAY be considered. For example, in a lossy network environment, such as wide area wireless connectivity through GPRS, a better scheduling policy might be 'smallest pending object first' where the next GET request goes on the SCTP stream that has the smallest sum of object sizes pending transfer. Such a policy reduces the probability of intra-stream HOL blocking, i.e., HOL blocking between responses downloaded on the same SCTP stream.

5. Lessons Learned from Implementing HTTP over SCTP

HTTP over SCTP was implemented in the Apache server and Firefox browser at the University of Delaware's Protocol Engineering Lab. Some lessons learned during this experience are discussed below. More details can be found in [N2008].

5.1. Avoiding Dependencies in Message Transmission

SCTP's fragmentation and reassembly algorithm creates dependencies in message transmission, i.e., a fragment of message i+1 cannot be transmitted until all fragments of message i have been transmitted. If messages i and i+1 are of sizes 100KB and 1KB respectively, the 100KB message transmission can unnecessarily block transmission of the 1KB message. The client or server application can overcome this by splitting each HTTP request or response into multiple messages, such that, each message at the SCTP layer results in a PMTU-sized SCTP PDU, and is not fragmented further by SCTP. An application can use either the SCTP_PEER_ADDR or the SCTP_STATUS socket options to obtain an SCTP association's PMTU [I-D.ietf-tsvwg-sctpsocket].

5.2. Order of Pipelined Requests and Responses

Section 8 of [RFC2616] mandates that in HTTP 1.1 with pipelining, "a server MUST send its responses to those requests in the same order

that the requests were received." Since TCP always delivers data in-order, the order of HTTP requests received by the server, and therefore, the order of HTTP responses generated by the server match the order of transmitted HTTP requests from the client. Consequently, a web client can assume that, within a TCP connection, the order of HTTP responses from the server always matches the order of transmitted HTTP requests. Unlike TCP, SCTP's multistreaming feature delivers out-of-order data at both the server and client. When HTTP requests from client to server are lost, requests transmitted over different SCTP streams will be delivered out-of-order at the server, and therefore, the order of generated HTTP responses will be different from the order of transmitted HTTP requests. Also, the loss of an HTTP response will affect the order of HTTP responses from the server. Our experience with the FreeBSD SCTP implementation revealed that HTTP requests and responses can be received out-of-order even under no loss conditions [N2008]. Therefore, web client implementations MUST be aware that within an SCTP assocation, the order of pipelined responses from the server may not match the order of transmitted HTTP reqeusts. However, in case of intra-stream pipelining, the order of HTTP responses within an inbound SCTP stream *a* MUST match the order of transmitted HTTP requests within the corresponding outbound SCTP stream *a*. Consequently, within each SCTP stream, a web server MUST send its responses to those reqeusts in the same order that the requests were received.

5.3. Benefits for Progressive Images

Progressive images (e.g., JPEG, PNG) are coded such that the initial bytes approximate the entire image, and successive bytes gradually improve the image's quality/resolution. Simple experiments have shown that user-perceived response time improvements for HTTP 1.1 (persistent and pipelined) transfers consisting of progressive images are more significant than for similar transfers consisting of non-progressive images. When each progressive image is downloaded on a different SCTP stream, the Firefox implementation over FreeBSD SCTP renders a good quality version of each progressive image significantly earlier than the page download time [NAS2008]. These page downloads were captured as movies and can be viewed at [Movies].

6. Open Issues

This section discusses some of the open issues that require further discussion and/or investigation.

6.1. How does a Web client decide between TCP vs. SCTP?

We see two options for how the web client can decide between using TCP vs. SCTP for an HTTP (1.0 or 1.1) transfer. Option 1: The web client tries in tandem to establish both a TCP connection and an SCTP association to the server. The web client chooses TCP vs. SCTP depending on which transport connection gets established first. Option 2: The web client selects TCP vs. SCTP based on the URI. URIs starting with "http://" or "https://" imply TCP and a new URI scheme could be established for similar services over SCTP, such as, "httpsctp://" or "httpssctp://". At this point, the authors believe that option 1 is more desireable than option 2, and will have less repercussions than option 2.

Web client implementations MUST be aware that an end user or the other end-point (server/proxy) MAY choose to override the client's default choice of transport (TCP vs. SCTP). Also, web clients SHOULD cache information on which servers support SCTP, for later re-use.

6.2. TCP-SCTP Gateway

Research has shown that SCTP streams enable perceivable improvements to web response times, especially in high latency and/or lossy last hops such as VSAT links [N2008]. A TCP-SCTP gateway allows web clients in such last hops to experiance the benefits of SCTP streams even if the web server runs over TCP. Additionally, the gateway also ensures that, web clients connecting to the Internet via the gateway MAY always assume SCTP as the default transport instead of trying to choose between TCP vs. SCTP as discussed in Section 6.1.

6.3. SCTP and NATs

The end-to-end path between a client and server MAY consist of one or more Network Address Translators (NATs) that manipulate address and port information in IP and SCTP headers. SCTP's association establishment and multihoming mechanisms pose unique challenges in the context of NATs. These issues are discussed in [I-D.stewart-behave-sctpnat].

7. Acknowledgments

8. References

8.1. Normative References

[I-D.ietf-tsvwg-sctpsocket]
Stewart, R., Poon, K., Tuexen, M., Yasevich, V., and P. Lei, "Sockets API Extensions for Stream Control Transmission Protocol (SCTP)", draft-ietf-tsvwg-sctpsocket-17 (work in progress), July 2008.

[RFC1945] Berners-Lee, T., Fielding, R., and H. Nielsen, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996.

[RFC2116] Apple, C. and K. Rossen, "X.500 Implementations Catalog-96", RFC 2116, April 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

8.2. Informative References

[FTY1999] Faber, T., Touch, J., and W. Yue, "The TIME_WAIT state in TCP and its effect on busy servers", INFOCOM '99: Proceedings of the IEEE INFOCOM Conference, pp. 1573-1583 , 1999.

[I-D.stewart-behave-sctpnat]
Stewart, R., Tuexen, M., and I. Ruengeler, "Stream Control Transmission Protocol (SCTP) Network Address Translation", draft-stewart-behave-sctpnat-04 (work in progress), July 2008.

[I-D.stewart-tsvwg-sctpstrrst]
Stewart, R., Lei, P., and M. Tuexen, "Stream Control Transmission Protocol (SCTP) Stream Reset", draft-stewart-tsvwg-sctpstrrst-00 (work in progress), June 2008.

[Movies] "Movies Comparing HTTP over TCP vs. HTTP over SCTP Streams", 2008, <http://www.cis.udel.edu/~amer/PEL/leighton.movies/index.html>.

[N2008] Natarajan, P., "Leveraging Innovative Transport Layer Services for Improved Application Performance", PhD Dissertation, in progress, Computer & Information Sciences Department, University of Delaware, USA , 2008.

[NAS2008] Natarajan, P., Amer, P., and R. Stewart, "Multistreamed Web Transport for Developing Regions", NSDR '08: Proceedings of the second ACM SIGCOMM workshop on Networked systems for developing regions, Seattle, WA, USA , 2008.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.

Authors' Addresses

Preethi Natarajan
University of Delaware
Computer and Information Sciences Department
Newark, DE 19716

USA

Phone:
Email: nataraja@cis.udel.edu

Paul D. Amer
University of Delaware
Computer and Information Sciences Department
Newark, DE 19716
USA

Phone: 302-831-1944
Email: amer@cis.udel.edu

Jonathan Leighton
University of Delaware
Computer and Information Sciences Department
Newark, DE 19716
USA

Phone:
Email: leighton@cis.udel.edu

Fred Baker
Cisco Systems
1121 Via Del Rey
Santa Barbara, CA 93117
USA

Phone:
Email: fred@cisco.com

Full Copyright Statement

THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.