A transport protocol for SIP

Gonzalo Camarillo
Advanced Signalling Research Lab.
Ericsson
Finland
Gonzalo.Camarillo@ericsson.com

Henning Schulzrinne
Department of Computer Science
Columbia University
USA
hgs@cs.columbia.edu

Raimo Kantola
Networking Laboratory
Helsinki University of Technology
Finland
Raimo.Kantola@hut.fi

Abstract

Current SIP implementation typically use TCP or UDP as a transport protocol. The differences between SIP over UDP and SIP over TCP have already been analyzed and are relatively well-known. However, there have not been so far SIP implementations that use SCTP as a transport. This paper analyzes the advantages that can be derived from the use of SCTP as a transport for SIP. It shows how while SCTP is an excellent transport protocol for high levels of traffic its performance decreases when the number of SIP transactions transmitted in parallel decreases.

1 Introduction

The Session Initiation Protocol (SIP) is an application-layer protocol for creating, modifying and terminating sessions. SIP [1] is designed in a modular way so that it is independent of the type of session established and of the lower-layer transport protocol used. Its modularity is one of the most important strengths of SIP. It makes SIP flexible and easy to extend with new features.

The SIP specification describes how the protocol operates over TCP [2] and over UDP [3]. Both transport protocols have different characteristics and provide a particular SIP application with different services. TCP provides reliable in-order transfer of bytes while UDP does not ensure neither reliability nor in-order delivery. Both UDP and TCP present certain advantages and disadvantages, and also both of them present certain limitations regarding signalling transport.

The limitations present in TCP and UDP for transporting signalling traffic led to the design of a new transport protocol within the IETF. The SIGTRAN working group developed the Stream Control Transmission Protocol (SCTP). SCTP [4] was first intended to transport telephony signalling over an unreliable network such as an IP network. However, the protocol has been designed so that SCTP can be used as a general-purpose transport protocol.

There have been already attempts to define SIP operation on top of SCTP [5]. However, although there are already implementations of telephony signalling protocols such as ISUP on top of SCTP, so far there has not been any implementation of SIP over SCTP that could show the gains that SCTP might achieve. This document discusses advantages and disadvantages derived from the use of SCTP as a transport protocol for SIP.

The remainder of this document is organized as follows. Section 2 and 3 describes SIP operation on top of TCP and UDP respectively. Pros and cons of each protocol are analyzed. Section 4 provides an introduction to SCTP. Section 5 analyzes advantages and disadvantages of using SCTP as a transport for SIP and finally section 6 outlines some conclusions.

2 SIP over TCP

The natural choice to transport a signalling protocol whose messages have to be reliably delivered to the destination seems to be a reliable transport protocol. Since the most widespread reliable transport protocol is TCP, it would not have been surprising if SIP had been designed to run only over TCP. Besides, SIP is based on HTTP [6], which uses TCP as a transport.

However, TCP presents some limitations regarding signalling transport. Therefore, SIP was designed to be independent of the transport protocol. This way, SIP can also run over UDP overcoming some of TCP's limitations. At present, UDP is the most widespread transport for SIP.

2.1 TCP limitations

TCP was designed to transport large amounts of data between two end-points. Once a connection is established, TCP implements flow control and error correction based on the dynamic behavior of the end-to-end traffic. However, signalling traffic does not consist of large amounts of data. Signalling traffic usually consists of small bursts of information. TCP's flow control mechanisms are not designed for such as traffic pattern, and therefore do not perform as well as it might be expected.

Fast retransmit algorithm

When a large bulk of data is being transmitted by TCP, ack messages from the receiver are continuously received indicating which segments have been

successfully received. The receiver sends duplicate acks when out-of-order segments arrive. Thus, arrival of duplicate acks indicates that a segment was lost. Therefore, the sender retransmits it without waiting for a timeout. This mechanism is referred to as fast retransmit and it is used together with the fast recovery algorithm.

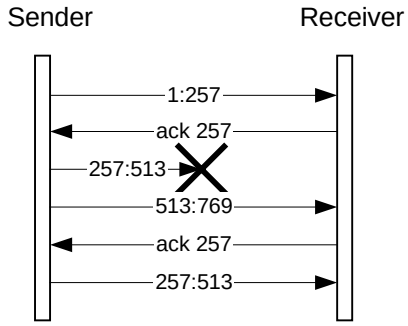

Figure 1 : Fast retransmit

Note that the sender in figure 1 retransmit the missing segment upon reception of a duplicate ack. This flow has been simplified. A typical implementation waits until three duplicate acks are received before retransmitting a segment.

Figure 1 shows how TCP behaves when large bulks of data are transmitted. Retransmissions are usually triggered by duplicate acks rather than timeouts. This is the reason why TCP timeouts are relatively high, in the order of 1,5 seconds. This allows using the fast retransmit algorithm before a timeout occurs.

However, SIP messages are relatively small, in the order of 500 bytes. A SIP message usually fits into a TCP segment. So, if a TCP segment that contains a SIP message gets lost, TCP will not be able to receive duplicate acks, since it is not sending any more data. Therefore, TCP will have to wait for a timeout in order to retransmit the missing segment. This results in a too conservative retransmission policy when TCP transports SIP signalling.

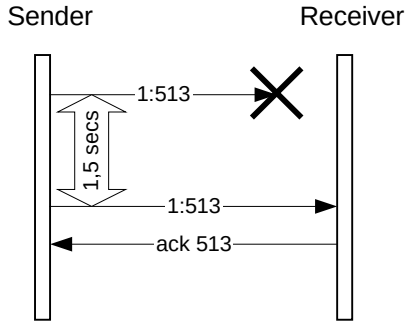

Figure 2 : TCP timeout

TCP connection establishment

TCP performs a three-way handshake before any user data can be transmitted between both ends. In a long-lived connection, the connection establishment time is negligible compared to the whole connection duration. However, signalling traffic is delay sensitive. If a SIP UAC wants to send an INVITE over TCP it will have to wait until the TCP connection is established before sending the INVITE.

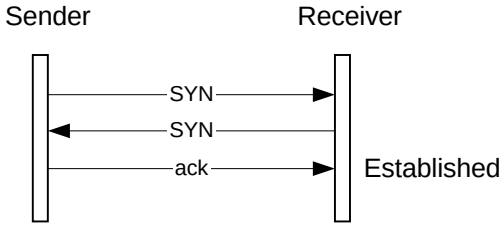

Figure 3 : TCP three-way handshake

The receiver of figure 3 will not pass any data to the application until it does not reach the “established” state. This overhead is not acceptable when the user is expecting an answer for his INVITE.

TCP implements a special timer for connection establishment. When a SYN gets lost, a typical implementation retransmits it after 6 seconds. Therefore, a single packet loss increases enormously the connection establishment delay introduced by TCP.

2.2 Multiple SIP sessions

One straightforward attempt to resolve both issues previously described consists of bundling several SIP sessions into a single TCP connection. With a high number of SIP sessions the TCP connection transports data continuously so that packet losses are detected by receiving duplicated acks rather than by timeouts. This increases the performance of TCP and reduces the delay introduced to SIP messages.

Another advantage of bundling SIP sessions is that the first SIP message of a new session does not have to wait for a new TCP connection to be established before being transmitted. Since the TCP connection is already established SIP messages belonging to a new SIP session are not affected by any additional delay. They can be sent immediately.

A SIP UAC usually handles a single SIP session, but proxies in the network have several ongoing SIP sessions between them at the same time. Therefore, proxies handling a high number of SIP sessions can typically take advantage of bundling SIP sessions. Another example were bundling can be performed is between a large gateway towards the PSTN and its outbound proxy.

Byte stream service

However, TCP presents an important limitation regarding bundling of sessions. TCP provides ordered delivery of a stream of bytes. When TCP is used to transmit messages it preserves the order in which the messages were sent by the sender. This property causes interaction problems between different SIP sessions carried on a single TCP connection.

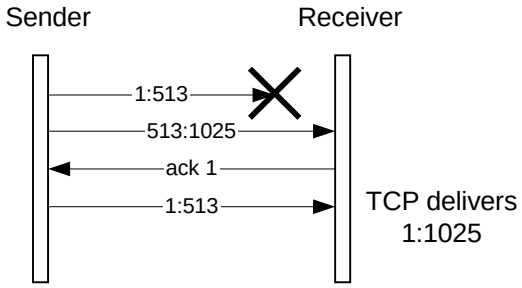

Figure 4 : TCP provides ordered delivery

The sender of figure 4 sends two INVITEs that belong to different sessions using the same TCP connection. The segment carrying the first INVITE gets lost (1:513), but the segment carrying the second INVITE arrives properly to the receiver (513:1025). However, since TCP provides ordered delivery, it will not delivered the second INVITE to the application until it has delivered the first INVITE. Therefore, the second INVITE is delayed until the first INVITE is retransmitted. The consequence is that a particular SIP session might suffer delay without having experimented any packet loss, as it is shown in figure 4.

3 SIP over UDP

Transporting SIP over UDP overcomes some of the problems associated with TCP. UDP is a connectionless protocol. Thus, it does not perform any kind of connection establishment before sending data. Therefore, a particular INVITE will be sent encapsulated in a UDP packet without any establishment delay introduced by the transport protocol.

Since UDP does not provide reliable transport, reliable delivery is achieved through application level retransmissions. The SIP application retransmit a particular SIP messages when the retransmission timer expires. This retransmission timer is lower than in TCP. Its default value is 0,5 seconds. Therefore, the retransmission policy of SIP when it runs over UDP is more aggressive than when it runs over TCP.

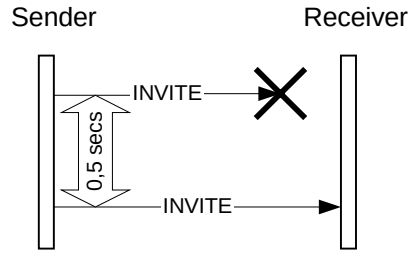

Figure 5 : SIP retranmission policy using UDP

SIP can afford to have a more aggressive retransmission policy over UDP than TCP because it transmits a small number of small messages. Therefore, SIP assumes that it is not going to congest the network because they are retransmitted more often than TCP.

Therefore, when a single or a small number of SIP sessions are handled, UDP is a better choice than TCP. However, UDP, as opposed to TCP, does not hide retransmissions from the application layer. Thus, although a SIP application using UDP has to store more state information than when TCP is used this does not represent an important issue for most of the applications.

3.1 Multiple SIP sessions

When there are multiple SIP sessions between two proxies they can be bundled in a single TCP session to take advantage of the congestion control mechanisms built in TCP. Losses are detected before and thus, performance improves.

However, when UDP is used, the same retransmission timers apply to every session. This can lead to a poorer performance and even to network congestion, since UDP does not provide congestion information to the application and by default SIP uses a more aggressive retransmission policy than TCP.

Therefore, for proxies handling a large amount of connections, the choice between UDP and TCP is not clear. TCP presents the previously described head of the line blocking issue and UDP does not implement any congestion control mechanism. The choice between TCP and UDP depends on how the network is loaded at a certain moment and the RTT between sender and receiver.

4 SCTP

The Stream Control Transmission Protocol (SCTP) is intended to resolve the issues derived from the use of TCP and UDP when there are multiple SIP sessions between sender and receiver. SCTP [4] also provides a certain level of fault tolerance through multihoming.

4.1 SCTP connection establishment

SCTP is a connection oriented transport protocol. In SCTP terminology, a connection is referred to as an association. An association is established through a four-way handshake in which the last two messages can already carry user data.

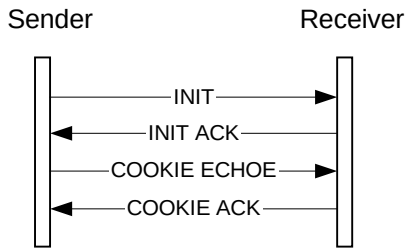

Figure 6 : SCTP four-way handshake

In this handshake end users exchange one or multiple IP addresses or host names. One destination address will be marked as the primary. The rest of them will be used in case the primary destination becomes unavailable. This feature, known as multihoming, allows a SCTP connection to survive network failures. The data is just sent to another destination address in case of failure.

The four-way handshake provides also a certain level of protection against resource attacks. The receiver, upon reception of an INIT message sends back a cookie in the INIT ACK. The receiver does not allocate any resources for this SCTP association until it receives the same cookie in the COOKIE ECHOE message. This way, resources are allocated when it is ensured that the party sending the INIT message is really willing to establish an SCTP association.

4.2 Multiple streams within an association

SCTP provides multiplexing/demultiplexing capabilities within an association. A single association can contain several streams. Each stream is identified by its stream id. During the four-way handshake the number of streams in both directions is negotiated.

An association can contain several types of streams. The base SCTP specification [4] defines two services: reliable ordered delivery and reliable unordered delivery. However, there are extensions [7] that provide an unreliable delivery service.

It is important to note that a particular service is provided on stream basis. Therefore, one stream within an association might be an ordered stream while another is unordered.

4.3 Flow and congestion control per association

Even if an association contains several streams, SCTP performs flow and congestion control per association. This allows to use the behavior of all the traffic within the association as input for the flow control mechanisms, which are effectively very similar to the ones used by TCP.

For instance, the fast retransmit algorithm can be used effectively without waiting for timeouts in order to retransmit data. Figure 7 shows how stream demultiplexing and flow control work together in an example.

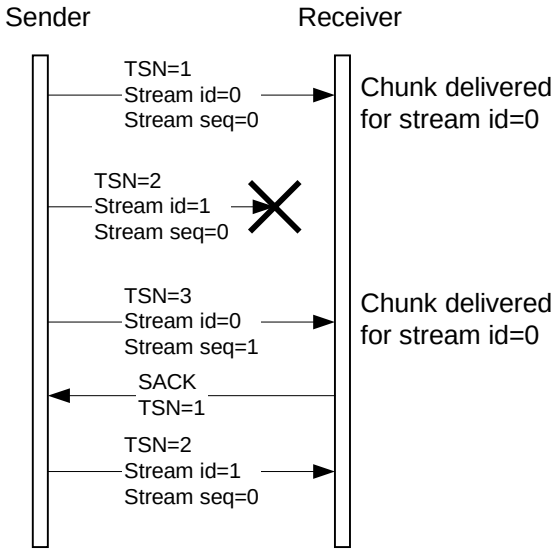

Figure 7 : Multiple streams within an association

The association of figure 7 consists of two ordered streams (stream id=0 and stream id=1). SCTP implements a general sequence number space (Transmission Sequence Number) and a sequence number space per stream. The general TSN is used to perform flow control and packet loss recovery and the stream sequence numbers are used to deliver individual streams.

When the message with TSN=3 arrives to the receiver, this knows that TSN=2 is lost. However, it also knows that TSN=3 is the next packet of stream id=0 (Stream seq=1). Therefore, it delivers the packet to the application without waiting to receive TSN=2. In the SACK (Selective ACK) the receiver reports that TSN=2 was missing.

Therefore, losses in one stream do not introduce delay on other streams. Besides, since the whole association is used to perform flow control, the sender detects that TSN=2 got lost thanks to the SACK sent upon reception of TSN=3, that belongs to a different stream. This way,

SCTP does not have to wait for a timeout to retransmit TSN=2.

So, SCTP combines good features of both TCP and UDP. It bundles streams to take advantage of flow control mechanisms and delivers separately packets belonging to different streams.

5 SIP over SCTP

It seems clear that proxies that handle multiple SIP sessions between them can obtain a better performance using an SCTP association than using TCP or UDP. If each SIP session is sent over an ordered stream, SIP messages can take advantage of flow control without being delayed by lost messages from other sessions.

However, even when multiple ordered streams are used, it is still possible that messages are delayed by other messages belonging to the same SIP session. The example of figure 8 shows how the loss of a provisional response can delay the delivery of the final response which was successfully received.

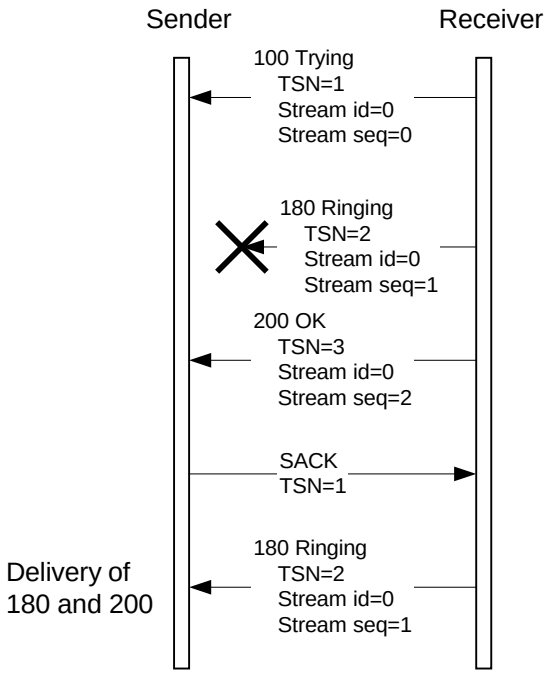

Figure 8 : SIP over ordered SCTP streams

In figure 8 all SIP responses are sent over an ordered SCTP stream (stream id=0). Therefore, SCTP delivers messages to the application in order within the stream. Since the provisional response “180 Ringing” got lost, SCTP cannot deliver the final response “200 OK” to the application. SCTP waits until TSN=2 arrives before delivering both responses.

Unordered service for final responses

In order to overcome this problem SIP final responses can be sent using the SCTP unordered service. SCTP allows to send unordered messages within an ordered stream. Therefore, all SIP messages within a SIP session are still sent using the same stream, but messages carrying final responses are sent with the SCTP unordered flag set.

Note that a receiver performs demultiplexing of incoming SIP messages based on the Call-ID of the SIP message rather than on the SCTP stream id. Stream ids are used here to solve the head of the line blocking problem. They are not intended to provide further demupltiplexing.

General unordered service

The method just described would be the most efficient way of transporting SIP over SCTP. However, there is a simpler mechanism that behaves nearly as well and simplifies implementations. It consists of sending all SIP traffic using the SCTP unordered service. When all SIP messages are sent with the unordered flag set SCTP delivers any message received immediately, independently of which stream the message belongs to. Thus, SIP entities can perfectly use the same stream id for all SIP sessions.

This mechanism is simpler because an implementation does not have to ensure that SIP messages belonging to a particular SIP session are always sent using the same stream id. Implementation that are not willing to perform stream id management should use this mechanism.

An example of such an implementation is a proxy that does not store state information about SIP transactions (stateless) but has SCTP associations continuously open to send SIP messages to certain common destinations.

Note that the use of a hash of the Call-ID of a SIP message module the number of SCTP streams available in order to choose the outgoing stream id for the message has some limitations. Although with a high number of available streams it is not likely no happen, a system using this method might end up sending requests with different Call-IDs using the same stream id. This would result in the head of the line blocking problem previously mentioned.

These two methods have the advantage of interworking together. Any receiver is able to receive traffic from senders using any of both mechanisms.

5.1 Differences between both methods

The only difference between both methods is that sending just final responses with the SCTP unordered

flag set avoids re-ordering of requests and provisional responses in the parts of the path where SCTP is used. However, there are just a few scenarios where this can happen.

Provisional responses

Provisional responses are sent unreliably by SIP. SIP systems do not rely on provisional responses to drive any protocol state machine. Therefore, receiving out of order provisional responses does not represent a problem for a SIP UAs.

When a SIP UA is interested in provisional responses it uses the extension defined in [9]. Then, provisional responses are transmitted reliably. [9] recommends SIP servers sending provisional responses not to send subsequent responses until the previous one has been acknowledged with a PRACK. Thus, using ordered or unordered SCTP to transport provisional responses does not make a difference, since the SIP layer ensures that they are received in order.

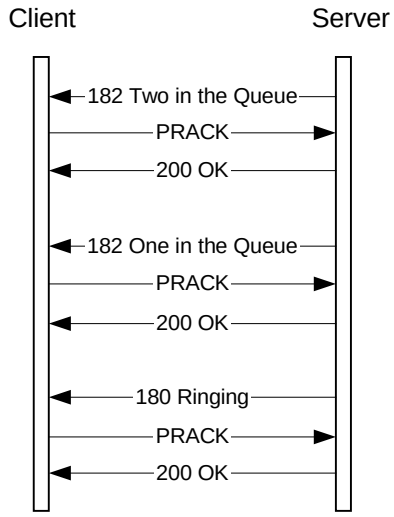

Figure 9 : SIP ensures in order delivery

Requests

The behavior of a SIP entity sending requests is similar to the one described for reliable provisional responses. A SIP client does not send a request until the previous transaction has completed. There are two exceptions to this rule, but in general it does not make a difference the transport used (ordered or unordered) for requests either.

The only two exceptions when a SIP client sends overlapping requests are: an INVITE followed by a CANCEL and an INVITE followed by a BYE. Note that other methods such as COMET or PRACK are just sent after a response for the INVITE has been received. Note also that CANCEL can terminate any request other than CANCEL and ACK. However, since non-INVITE requests are responded immediately by the server, CANCEL is typically used only for INVITE requests.

These two situations are the only ones where both uses of SCTP described previously differ. If the requests are sent unordered, a CANCEL or a BYE might overtake the INVITE sent before. Ordered SCTP ensures that they arrive in the same order as they were sent. However, this is only ensured in the part of the path where ordered SCTP is used. If other transport protocol such as UDP is used in another part of the path, reordering can still happen. Therefore, even systems using ordered SCTP have to be prepared to handle out of order CANCELs and BYEs. Figure 10 shows how a system using ordered SCTP might still receive out of order requests.

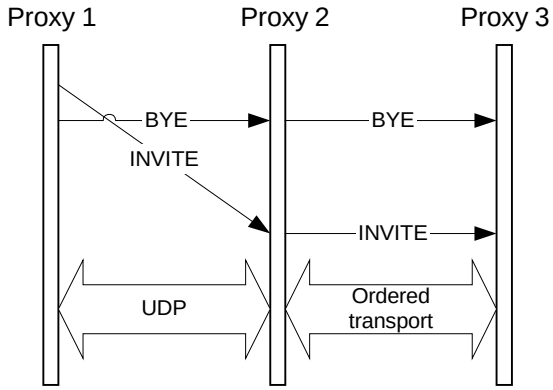

Figure 10 : INVITE followed by a BYE

Even if ordered SCTP streams are used, a SIP entity has to be prepared to received a BYE before an INVITE. A "481 Transaction Does Not exist" will be sent as response to the BYE.

Therefore, the only difference between sending all the SIP traffic with the SCTP unordered flag set and sending just final responses with this flag is that the likelihood of receiving a BYE or a CANCEL before an INVITE decreases using the latter method, although it might still happen.

5.2 Other strengths of SCTP

The previous section described SCTP behaves like a TCP connection without the head of the line blocking problem. Besides resolving this problem, SCTP has some other strengths that SIP can take advantage of.

Message based

SCTP is a message-oriented protocol, as opposed to TCP that is stream oriented. SCTP delivers messages while TCP delivers a stream of bytes. This makes it possible for SCTP to provide unordered delivery of SIP messages. In TCP this concept would not make any

sense, since delivering unordered bytes would be useless for an application.

Message-oriented protocols such as SCTP or UDP allow implementing simpler parsers. When these transport protocols deliver a message to the application it contains a single SIP message. In order to parse a SIP message received over TCP it is necessary to implement application level boundaries such as the SIP Content-Length header.

Transport-layer fragmentation

However, although both SCTP and UDP are message-oriented transport protocols, SCTP has an advantage over UDP. SCTP implements transport-level fragmentation while UDP does not. If a SIP message inside a UDP packet is larger than the path MTU the packet will be fragmented at the IP layer.

IP-layer fragmentation presents several problems. The likelihood of having packet losses increases and firewall and NAT traversal becomes impossible. The fragments of the UDP packet do not carry the UDP header, which contains the source and the destination port number of the UDP packet. Therefore, network devices that need to examine port numbers will simply discard the packets.

SCTP implements transport-layer fragmentation. Messages larger than the path MTU are transported in different SCTP chunks. Every chunk carries complete transport information, and thus, problems derived from IP fragmentation are avoided. Different chunks are reassemble at the destination and delivered to the application as a single message.

Currently fragmentation does not represent a serious problem for SIP, since SIP messages are usually smaller than the path MTU. However, new session description protocols or new SIP extensions might increase the size of SIP messages. SCTP fragmentation would then represent an important advantage.

Bundling of chunks

Figure 11 shows the format of a SCTP packet. It contains a common header and several chunks. Unless fragmentation is performed, a chunk contains an application-level message.

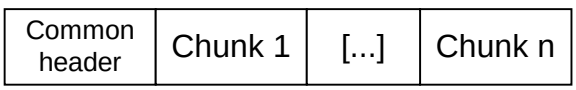

Figure 11 : SCTP message format

Therefore, a single SCTP packet can carry several SIP messages that belong to different sessions. Bundling SCTP chunks decreases the number of packets sent through the network. This avoids certain congestion problems in IP routers and typically achieves a better performance than sending various individual packets.

Multihoming

SCTP provides several source and destination addresses within an association. They are intended to provide alternative paths to be used in case of network failures. This feature increases the reliability of an association.

Multiple destination addresses are not intended to provide a load balancing mechanism. SCTP marks one address as the primary, and all the traffic is routed to that address until it fails. Other mechanisms such as DNS SRV [8] records might be used to provide load balancing. SCTP multihoming just provides a fail over mechanism.

5.3 A single SIP session over SCTP

It is clear that SIP entities that handle a high amount of SIP traffic between them can take advantage of SCTP and all its features. However, SCTP advantages are not so evident when a single SIP session (or a small number of them) is transported. In this scenario SCTP shares some problems that TCP has. SCTP association establishment delays the delivery of the first INVITE, and once the association is established, SCTP timeouts are more conservative than the ones used by SIP over UDP. The initial value for the SCTP retransmission timer is 3 seconds and even when RTT measurements are performed its minimum value is 1 second.

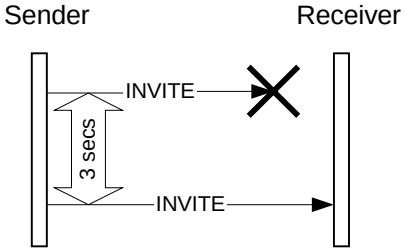

Figure 12 : SCTP's initial retransmission timer

The only advantage of SCTP over UDP in a scenario with low level of SIP traffic is the transport-layer fragmentation provided by SCTP, since multihoming can be achieved using UDP in conjunction with DNS SRV records.

6 Conclusions

The best transport protocol for SIP depends on the amount of SIP traffic that a particular SIP entity handles. SIP entities that handle a large amount of SIP traffic between them such as proxies and large SIP gateways have in SCTP their best choice. SCTP bundles together several SIP sessions into a single SCTP association and then performs flow and congestion control per

association. This way, packet losses are detected before retransmission timers expire leading to an increase in the overall performance. Among all the possible services provided by SCTP, unordered delivery and ordered delivery with unordered final responses are the ones that suit SIP better.

However, SIP entities that handle a small number of SIP sessions such as the SIP UA of a individual user cannot take advantage of the flow control provided by SCTP. When a small number of SIP messages are transported over SCTP packet losses are detected by timeouts. This leads to a too conservative retransmission policy, since timers in SCTP are not designed for situations where the traffic load is very low. Therefore, small SIP entities have in UDP their best choice. UDP does not introduce any connection establishment time and retransmit lost packets in a more aggressive way than SCTP. However, since SIP applications using UDP do not perform any congestion control other than implementing a back-off retransmission timer, the use of UDP is not recommended for high volumes of SIP traffic.

While TCP is an excellent protocol for transferring large amounts of data such as files or the contents of a particular web page, it presents important limitation regarding signalling transport. Therefore, depending on the SIP entity, UDP or SCTP are better choices to transport SIP signalling.

7 Acronyms

ACK: Acknowledgement
DNS: Domain Name System
HTTP: HyperText Transfer Protocol
IP: Internet Protocol
ISDN: Integrated Services Digital Network
ISUP: ISDN User Part Protocol
MTU: Maximum Transmission Unit
NAT: Network Address Translator
PRACK: Provisional ACK
PSTN: Public Switched Telephone Network
RTT: Round Trip Time
SACK: Selective ACK
SCTP: Stream Control Transmission Protocol
SIGTRAN: Signalling Transport
SIP: Session Initiation Protocol
SYN: Synchronize sequence numbers flag
TCP: Transmission Control Protocol
TSN: Transmission Sequence Number
UA: User Agent
UAC: User Agent Client
UDP: User Datagram Protocol

References

[1] Handley M., Schulzrinne H., Schooler E., Rosenberg J., “SIP: Session Initiation Protocol”, RFC 2543. IETF. March 1999.

[2] Postel J., “Transmission Control Protocol”, RFC 793. IETF. September 1981.

[3] Postel J, “User Datagram Protocol”, RFC 768. IETF. August 1980.

[4] Stewart R., Xie Q., Morneault K., Sharp C., Schwarzbauer H., Taylor T., Rytina I., Kalla M., Zhang L., Paxson V., “Stream Control Transmission Protocol”, RFC 2960. IETF. October 2000

[5] Rosenberg J, Schulzrinne H., “SCTP as a Transport for SIP”, draft-rosenberg-sip-sctp-00.txt. IETF. Jone 2000. Work in progress.

[6] Fielding R., Gettys J., Mogul J., Frystyk H., Berners-Lee T., “Hypertext Transfer Protocol -- HTTP/1.1”, RFC 2068. IETF. January 1997.

[7] Xie Q., Stewart R., Sharp C., Rytina I., “SCTP Unreliable Data Mode Extension”, draft-ietf-sigtran-usctp-01.txt. IETF. February 2001. Work in progress.

[8] Gulbrandsen A., Vixie P., Esibov L., “A DNS RR for specifying the location of services (DNS SRV)”, RFC 2782. IETF. February 2000.

[9] Rosenberg J., Schulzrinne H., “Reliability of Provisional Responses in SIP”, draft-ietf-sip-100rel-03.txt. IETF. March 2001. Work in progress.