Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 30, 2011

R. Stewart
Huawei
P. Lei
Cisco Systems, Inc.
M. Tuexen
Muenster Univ. of Applied Sciences

March 29, 2011

Uses of Stream Reconfiguration for SCTP
draft-stewart-tsvwg-reconfuse-sctp-00.txt

Abstract

This document is used to convey different use cases for the Stream Reconfiguration draft xxxxxxx. It does not represent a standard nor does it represent real applications that are available for download. Instead it illustrates various use cases of the stream reconfiguration facilities for SCTP [RFC4960] . It serves as a guideline to application developers to show the stream reconfigurations various potentials.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 30, 2011.

Table of Contents

1. Introduction

Many times on the tsvwg mailing list some have questioned the valid uses of various features in the stream reconfiguration draft. This document attempts to answer those questions by illustrating use cases for all of the features currently contained within the draft has it went to last call in tsvwg. Some of the use cases mentioned here have been implemented at one time or another by the various co-authors of the reconfiguration draft. Other examples have not been built but instead are used to simply illustrate how one would use the feature.

2. Terminology

List any terminology that we will use in the document.

3. Example Applications

In this document we present NNN example applications that illustrate the use of stream reconfiguration. First we will examine a typical use by illustrating a mythical file transfer protocol something like FTP but modified to fit SCTP. Next we will go further and examine how the stream re-configuation in combination with the SCTP Address Reconfiguration [RFC5061] and SCTP Authentication [RFC4895] can be combined to provide a redundancy service that allows a "backup" application to acquire or move an association from one machine to another.

3.1. And FTP example

FTP is a long time staple of the internet. Most every operating system today has an ftp client and often times an FTP server that can be enabled if desired. It is an easily understood paradigm. A user logs in to an ftp server via an ftp command. And then can send various commands to change directories, request file transfers, list file or even make directories.

Some users will also find that they have to go into "passive" mode since normally when FTP starts a file transfer, it creates a new TCP connection between the server and the client for the actual file transfer. The closing of the connection is normally used as the signal to the client or server receiving the file that all of the data bytes have been transferred. In the Socket API this is usually signified by a zero length read. In the case of some NAT's and firewalls passive mode is required to not be blocked by the middlebox.

In this example we will illustrate the handy use of SCTP and its stream reconfiguration feature to do an equivalent SCTP version of FTP. The user would of course login to the ftp server in the same way today that they do with tcp. Setting up an association via a client command. In the case of the underlying SCTP configuration each side would ask for a number of streams equal to one more than the number of simultaneous transfers they anticipate supporting. Conservative clients and servers would ask for 2. Stream 0 is reserved for the "command" stream and is considered a bi-directional stream. Commands such as 'ls' or 'cd' would be sent via stream 0 and the appropriate responses (e.g. the list of files) would be sent back to the client over stream 0 in response.

When the client wishes to transfer a file by the client doing a "get foo" command. The server do the following:

1> Choose an idle stream, an idle stream is one not being used for file transfer at this time and is in the initial state, i.e. the next stream sequence number is 0.

2> On the selected stream the server would send a special message as the first message on the stream i.e. stream sequence 0. This message would contain the name of the file, the ownership (user name and possibly user id), permissions and any other bookkeeping information that is necessary for the transfer. Setting that streams state to non-idle or in the file transfer state.

3> After sending the first setup message, send each data block to the peer from the file in an ordered message on the selected stream to the peer requesting the file.

4> At the completion of the file transfer, reset the stream sequence number and place the stream back to the idle state.

When receiving this first message on the respective stream (N), the first message, ssn 0, would:

A> Key the receiver into placing the incoming stream into "file transfer" mode. The special ssn 0 on a un-assigned stream (i.e one not in a file transfer state) into using the first message to open the file and set the stream into the "file transfer" state.

B> All subsequent messages arriving on the stream in this state are directly written to the appropriate file descriptor assigned to that stream.

C> If for some reason the file could not be opened or a write error in output of the data occurs, the receiver would request a incoming and outgoing stream reset to signal to the peer that an error occurred.

D> In the file transfer state, the stream would continue to receive either messages (which would be written out) or a stream reset. The stream reset would indicate to the receiver that the stream is to be taken out of the file transfer state and the file is to be closed.

E> At that point the stream is set back to the initial state and is available for any subsequent transfer.

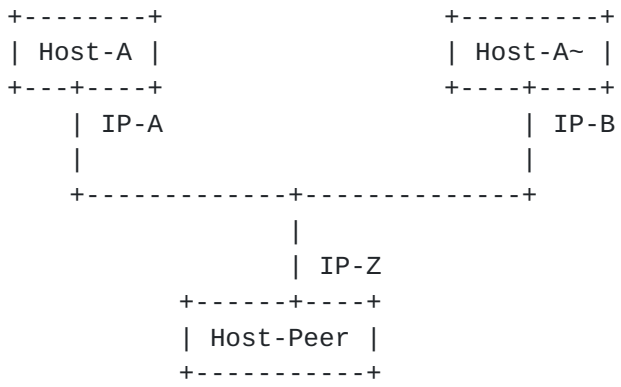
When a 'put' is requested by the client, no message need travel over the control stream 0. Instead the client would simply select an idle stream and send the same transfer message to the peer as the first stream sequence number 0 to the server. The server would react the same way that the client did above and follow similar procedures.

It is also possible that one of the sides, when going to transfer a file, does not have any idle streams. In such a case the add stream capability can be used to add an additional stream resource. This allows a SCTP-FTP server and client to start with a minimal number of streams (2 at a minimum) and only expand the number of streams upon the demand of the user for parallel transfers.

This example illustrates the use of two basic reconfiguration utilities. The add-stream and the stream-reset for SSN. The next example will illustrate some various uses of the multiple stream reset and the TSN/SSN reset capabilities.

3.2. Combining features to achieve redundancy

Consider the following picture:



Host-A and Host-Peer have an SCTP association between them. The association is between IP-A:NNNN and IP-Z:MMMMM. Now often times in such a situation it is desired to have Host-A~ backup Host-A in case of a fault in either the hardware or software of Host-A. A variety of methods have been used to do this in the past. One common method is that Host-A~ watches Host-A in some fashion via either specialized hardware of some sort of pinging mechanism. If Host-A~ sees that Host-A is down, it assumes the network identity of host A by adopting IP-A on its interface and sending a promiscuous ARP so that packets destined to IP-A end up at the backup (i.e. Host-A~).

Now one problem with this is any TCP connection or SCTP association between Host-A and Host-Peer will need to be torn down and re-established. This, in some protocols, may have un-desireable side-effects (e.g. BGP). Some implementations have attempted to compensate for this by having additional connectivity between Host-A and Host-A~ and keeping track of sequence numbers at the transport layer as well as keeping track of application state (between the primary and backup application). Much of the transport state is stable but large parts of it also are transient. In SCTP's case TSN and Stream Sequence numbers may be changing quite rapidly and this takes away additional resources by generating additional traffic between Host-A and Host-A~. Add to this that the promiscuous ARP and assuming the network identity of the peer is in itself somewhat kludgy at best.

Now consider a possible different scenario which can be used if we apply SCTP's address reconfiguration, SCTP Authentication and the Stream reconfiguration mechanisms. If the two servers (Host-A and Host-A~) will do the following every time they acquire a peer:

1) After association setup send the random keys (for AUTH), Vtags, Sequence numbers, stream sequence numbers and various SCTP PCB information to Host-A~.

2) The application can use an inter-SCTP association between IP-A and IP-B to send this information. This association can further be used to send any application state updates that are needed and both servers should be using the same port number NNNN.

3) If the application ever uses the stream reconfiguration mechanism, an updated stream reconfiguration sequence number must be sent to the backup peer over this association. But note these changes are hopefully far less frequent then changes to TSN or SSN's within the association to the Host-Peer.

4) The backup peer Host-A~ may wish to increase the SCTP heartbeat rate to a faster value so that normal SCTP mechanisms (although at a faster rate) can be used to detect peer association failure faster than normally would be detected (by Host-Peer).

After this setup, if Host-A fails, Host-A~ would do the following:

A) Immediately upon detecting host failure of Host-A, send an Address Reconfiguration [RFC5061] with the appropriate authentication [RFC4895] to Host-Peer. In this reconfiguration, Add IP-B first, followed by Delete IP-A. These two actions should be bundled in one reconfiguration message but the Add parameter must be first (otherwise the peer would reject the request).

B) After the address reconfiguration successfully completes send a stream reconfiguration resetting the TSN and all SSN's. This is done with the TSN/SSN reconfiguration request.

C) After completion the application should be able to continue to talk to Host-Peer over the same association. Asking for any reconfiguration (if needed) that the application would like with its peer (e.g. a BGP restart).

An alternative to this strategy would be to use just SSN resets but this would place additional burden upon the synchronization between Host-A and Host-A~ in that the TSN sequence would need to be kept in sync in addition to the stream reconfiguration sequence space.

This mechanism could then be used to provide a "friendly" takeover for redundancy purposes of a peer association with no down time. Other scenarios are also possible, for example an application may want to use similar mechanisms to hand off part of its load from one server to another while both are active i.e. not in a failure case.

More Examples to come

4. Security Considerations

TBD

5. IANA Considerations

TBD

6. IPR Considerations

An IPR disclosure will be forthcoming.

7. Acknowledgements

8. References

8.1. Normative references

[RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, August 2007.

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

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.

8.2. Informational References

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

Authors' Addresses

Randall R. Stewart
Huawei
Chapin, SC 29036
USA

Email: randall@lakerest.net

Peter Lei
Cisco Systems, Inc.
8735 West Higgins Road
Suite 300
Chicago, IL 60631
USA

Phone:
Email: peterlei@cisco.com

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstr. 39
48565 Steinfurt
Germany

Email: tuexen@fh-muenster.de