Network Working Group
Internet-Draft
Intended status: Experimental
Expires: December 20, 2013

Y. Nishida
GE Global Research
P. Natarajan
Cisco Systems
A. Caro
BBN Technologies
P. Amer
University of Delaware

June 18, 2013

Quick Failover Algorithm in SCTP
draft-ietf-tsvwg-sctp-failover-01

Abstract

One of the major advantages in SCTP is supporting multi-homing communication. If a multi-homed end-point has redundant network connections, SCTP sessions can have a good chance to survive from network failures by migrating inactive network to active one. However, if we follow the SCTP standard, there can be significant delay for the network migration. During this migration period, SCTP cannot transmit much data to the destination. This issue drastically impairs the usability of SCTP in some situations. This memo describes the issue of SCTP failover mechanism and discuss its solutions which require minimal modification to the current standard.

Table of Contents

1. Introduction

The Stream Control Transmission Protocol (SCTP) [RFC4960] natively supports multihoming at the transport layer -- an SCTP association can bind to multiple IP addresses at each endpoint. SCTP's multihoming features include failure detection and failover procedures to provide network interface redundancy and improved end-to-end fault tolerance.

In SCTP's current failure detection procedure, the sender must experience Path.Max.Retrans (PMR) number of consecutive timeouts on a destination before detecting path failure. The sender fails over to an alternate active destination only after failure detection. Until failover, the sender transmits data on the failed path, degrading SCTP performance. Concurrent Multipath Transfer (CMT) [IYENGAR06] is an extension to SCTP and allows the sender to transmit data on multiple paths simultaneously. Research [NATARAJAN09] shows that the current failure detection procedure worsens CMT performance during failover and can be significantly improved by employing a better failover algorithm.

This document proposes an alternative failure detection procedure for SCTP (and CMT) that improves SCTP (CMT) performance during failover.

2. Conventions and Terminology

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. Issue in SCTP Path Management Process

SCTP can utilize multiple IP addresses for a single SCTP association. Each SCTP endpoint exchanges the list of available addresses on the node during initial negotiation. After this, endpoints select one address from the list and define this as the primary destination. During normal transmission, SCTP sends all data to the primary destination. Also, it sends heartbeat packets to other (non-primary) destinations at a certain interval to check the reachability of the path.

If sender has multiple active destination addresses, it can retransmit data to secondary destination address when the transmission to the primary times out.

When sender receives the acknowledgment for data or heartbeat packets from one of the destination addresses, it considers the destination is active. If it fails to receive acknowledgments, the error count for the address is increased. If the error counter exceeds the protocol parameter 'Path.Max.Retrans', SCTP endpoint considers the address is inactive.

The failover process of SCTP is initiated when the primary path becomes inactive (error counter for the primary path exceeds Path.Max.Retrans). If the primary path is marked inactive, SCTP chooses new destination address from one of the active destinations and start using this address to send data. If the primary path becomes active again, SCTP uses the primary destination for subsequent data transmissions and stop using non-primary one.

An issue in this failover process is that it usually takes significant amount of time before SCTP switches to the new destination. Let's say the primary path on a multi-homed host becomes unavailable and the RTO value for the primary path at that time is around 1 second, it usually takes over 60 seconds before SCTP starts to use the secondary path. This is because the recommended value for Path.Max.Retrans in the standard is 5, which requires 6 consecutive timeouts before failover takes place. Before SCTP switches to the secondary address, SCTP keeps trying to send packets to the primary and only retransmitted packets are sent to the secondary can be reached at the receiver. This slow failover process can cause significant performance degradation and will not be acceptable in some situations.

4. Existing Solutions for Smooth Failover

The following approach are conceivable for the solutions of this issue.

4.1. Reduce Path.Max.Retrans

If we choose smaller value for Path.Max.Retrans, we can shorten the duration of failover process. In fact, this is recommended in some research results [JUNGMAIER02] [GRINNEMO04] [FALLON08]. For example, if we set Path.Max.Retrans to 0, SCTP switches to another destination on a single timeout. However, smaller value for Path.Max.Retrans might cause spurious failover. In addition, if we use smaller value for Path.Max.Retrans, we may also need to choose smaller value for 'Association.Max.Retrans'. The Association.Max.Retrans indicates the threshold for the total number of consecutive error count for the entire SCTP association. If the total of the error count for all paths exceeds this value, the endpoint considers the peer endpoint unreachable and terminates the association. According to the Section 8.2 in [RFC4960], we should avoid having the value of Association.Max.Retrans larger than the summation of the Path.Max.Retrans of all the destination addresses. Otherwise, even if all the destination addresses become inactive, the endpoint still considers the peer endpoint reachable. The behavior in this situation is not defined in the RFC and depends on each implementation. In order to avoid inconsistent behavior between implementations, we had better use smaller value for Association.Max.Retrans. However, if we choose smaller value for Association.Max.Retrans, associations will prone to be terminated with minor congestion.

Another issue is that the interval of heartbeat packet: 'HB.interval' may not be small. (recommended value is 30 seconds) This means once failover takes place, an endpoint might need a certain amount of time to use the primary path again. This can cause undesirable effects in case of spurious failover. If we choose smaller value for HB.interval, the traffic used for path probing in a session will be increased.

The advantage of tuning Path.Max.Retrans is that it requires no modification to the current standard, although it needs to ignore several recommendations. In addition, some research results indicate path bouncing caused by spurious failover does not cause serious problems. We discuss the effect of path bouncing in the section 5.

4.2. Adjust RTO related parameters

As several research results indicate, we can also shorten the duration of failover process by adjusting RTO related parameters [JUNGMAIER02] [FALLON08]. During failover process, RTO keeps being doubled. However, if we can choose smaller value for RTO.max, we can stop the exponential growth of RTO at some point. Also, choosing smaller values for RTO.initial or RTO.min can contribute to keep RTO value small.

Similar to reducing Path.Max.Retrans, the advantage of this approach is that it requires no modification to the current standard, although it needs to ignore several recommendations. However, this approach requires to have enough knowledge about the network characteristics between end points. Otherwise, it can introduce adverse side-effects such as spurious timeouts.

5. Proposed Solution: SCTP with Potentially-Failed Destination State (SCTP-PF)

5.1. SCTP-PF Description

Our proposal stems from the following two observations about SCTP's failure detection procedure:

- In order to minimize performance impact during failover, the sender should avoid transmitting data to the failed destination as early as possible. In the current SCTP path management scheme, the sender stops transmitting data to a destination only after the destination is marked Failed. Thus, a smaller PMR value is ideal so that the sender transitions a destination to the Failed state quicker.

- Smaller PMR values increase the chances of spurious failure detection where the sender incorrectly marks a destination as Failed during periods of temporary congestion. Larger PMR values are preferable to avoid spurious failure detection.

From the above observations it is clear that tweaking the PMR value involves the following tradeoff -- a lower value improves performance but increases the chances of spurious failure detection, whereas a higher value degrades performance and reduces spurious failure detection in a wide range of path conditions. Thus, tweaking the association's PMR value is an incomplete solution to address performance impact during failure.

We propose a new "Potentially-failed" (PF) destination state in SCTP's path management procedure. The PF state was originally proposed to improve CMT performance [NATARAJAN09]. The PF state is an intermediate state between Active and Failed states. SCTP's failure detection procedure is modified to include the PF state. The new failure detection algorithm assumes that loss detected by a timeout implies either severe congestion or failure en-route. After a single timeout on a path, a sender is unsure, and marks the corresponding destination as PF. A PF destination is not used for data transmission except in special cases (discussed below). The new failure detection algorithm requires only sender-side changes. Details are:

1. The sender maintains a new tunable parameter called Potentially-failed.Max.Retrans (PFMR). The recommended value of PFMR = 0 when quick failover is used. When an association's PFMR >= PMR, quick failover is turned off.

2. Each time the T3-rtx timer expires on an active or idle destination, the error counter of that destination address will be incremented. When the value in the error counter exceeds PFMR, the endpoint should mark the destination transport address as PF. SCTP MUST NOT send any notification to the upper layer about the Active to PF state transition.

3. The sender SHOULD avoid data transmission to PF destinations. When all destinations are in either PF or Inactive state, the sender MAY either move the destination from PF to Active state (and transmit data to the active destination) or the sender MAY transmit data to a PF destination. In the former scenario, (i) the sender MUST NOT notify the ULP about the state transition, and (ii) MUST NOT clear the destination's error counter. It is recommended that the sender picks the PF destination with least error count (fewest consecutive timeouts) for data transmission. In case of a tie (multiple PF destinations with same error count), the sender MAY choose the last active destination.

4. Only heartbeats MUST be sent to PF destination(s) once per RTO. This means the sender SHOULD ignore HB.interval for PF destinations. If an heartbeat is unanswered, the sender increments the error counter and exponentially backs off the RTO value. If error counter is less than PMR, the sender SHOULD transmit another heartbeat immediately after T3-timer expiration.

5. When the sender receives an heartbeat ACK from a PF destination, the sender clears the destination's error counter and transitions the PF destination back to Active state. This state transition MUST NOT be notified to the ULP. This destination's cwnd is set to 1 MTU. Note that in scenarios where the destination was temporarily congested during the T3-timer expiration, an SCTP sender transmits 1 MTU worth of data while an SCTP-PF sender transmits an HB after the T3-timer expiry (more details in Section 5 of [NATARAJAN09]). The SCTP sender has 1 RTT head-start in cwnd evolution compared to SCTP-PF sender. An SCTP-PF sender may set cwnd to 2 MTUs after receiving HB-ACK in order to offset this performance difference.

6. An additional (PMR - PFMR) consecutive timeouts on a PF destination confirm the path failure, upon which the destination transitions to the Inactive state. As described in [RFC4960], the sender (i) SHOULD notify ULP about this state transition, and (ii) transmit heartbeats to the Inactive destination at a lower frequency as described in Section 8.3 of [RFC4960].

7. When all destinations are in the Inactive state, the sender picks one of the Inactive destinations for data transmission. This

proposal recommends that the sender picks the Inactive destination with least error count (fewest consecutive timeouts) for data transmission. In case of a tie (multiple Inactive destinations with same error count), the sender MAY choose the last active destination.

8. ACKs for retransmissions do not transition a PF destination back to Active state, since a sender cannot disambiguate whether the ack was for the original transmission or the retransmission(s).

5.2. Effect of Path Bouncing

The methods described above can accelerate failover process. Hence, it might introduce path bouncing effect which keeps changing the data transmission path frequently. This sounds harmful for data transfer, however several research results indicate that there is no serious problem with SCTP in terms of path bouncing effect [CARO04] [CARO05].

There are two main reasons for this. First, SCTP is basically designed for multipath communication, which means SCTP maintains all path related parameters (cwnd, ssthresh, RTT, error count, etc) per each destination address. These parameters cannot be affected by path bouncing. In addition, when SCTP migrates to another path, it starts with minimal cwnd because of slow-start. Hence, there is little chance for packet reordering or duplicating.

Second, even if all communication paths between end-nodes share the same bottleneck, the proposed method does not make situations worse. In case of congestion, the current standard tries to transmit data packets to the primary during failover, while the proposed method tries to explore other destinations. In any case, the same amount of data packets sent to the same bottleneck.

5.3. Permanent Failover

Post failover, an SCTP sender migrates back to the original primary destination once this destination becomes active. The sender sets cwnd to the initial cwnd value and performs slow start. [CARO02] shows that the switch over to the original primary may degrade SCTP performance compared to continuing data transmission on the same path, especially in scenarios where this path's characteristics are better. In order to mitigate this performance degradation, permanent failover was proposed in [CARO02]. Permanent failover allows SCTP to remain the alternative path even if the primary path becomes active again. We recommend that SCTP-PF should stick to the standard RFC4960 behavior, i.e., switch back to the original primary once this destination becomes active again. Permanent failover may be considered in the future based on discussions and consensus within

the community.

5.4. Handling Error Counter

When multiple destinations are in the PF state, the sender may transmit heartbeats to multiple destinations at the same time. This allows SCTP-PF sender to quickly track and respond to network status change compared to an SCTP sender. However, when all PF destinations become unavailable, an SCTP-PF sender has outstanding HBs on all destinations compared to an SCTP sender and increases the count for the total number of consecutive retransmissions faster than the SCTP sender. SCTP-PF's faster increase in the error count will result in association termination sooner than SCTP.

For deployments where aggressive failure detection and association termination is not desired, we recommend that AMR be set to the maximum allowed value (sum of PMRs of all paths), to delay assoc termination during SCTP-PF. Another option is to send retransmitted data or HB to only one PF destination at a time, but this approach may delay path status tracking. To exclude HB timeouts from incrementing the error count can also be a solution, however, this requires an update to Section 8.3 of [RFC4960].

6. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to provide a way for the application to control the quick failover behavior.

Please note that this section is informational only.

A socket API implementation based on [RFC6458] is extended by adding a new read/write socket option for the level IPPROTO_SCTP and the name SCTP_PEER_ADDR_THLDS as described below. This socket option is used to read/write the value of PFMR parameter described in Section 5.

Support for the SCTP_PEER_ADDR_THLDS socket option needs also to be added to the function sctp_opt_info().

6.1. Peer Address Thresholds (SCTP_PEER_ADDR_THLDS) socket option

Applications can control the quick failover behavior by getting or setting the number of timeouts before a peer address is considered potentially failed or unreachable.

The following structure is used to access and modify the thresholds:

```
struct sctp_paddrthlds {
  sctp_assoc_t spt_assoc_id;
  struct sockaddr_storage spt_address;
  uint16_t spt_pathmaxrxt;
  uint16_t spt_pathpfthld;
};
```

spt_assoc_id: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets the application may fill in an association identifier or SCTP_FUTURE_ASSOC for this query. It is an error to use SCTP_{CURRENT|ALL}_ASSOC in spt_assoc_id.

spt_address: This specifies which peer address is of interest. If a wildcard address is provided, this socket option applies to all current and future peer addresses.

spt_pathmaxrxt: Each peer address of interest is considered unreachable, if its path error counter exceeds spt_pathmaxrxt.

spt_pathpfthld: Each peer address of interest is considered potentially failed, if its path error counter exceeds spt_pathpfthld.

7. Security Considerations

There are no new security considerations introduced in this document.

8. IANA Considerations

This document does not create any new registries or modify the rules for any existing registries managed by IANA.

9. References

9.1. Normative References

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

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

9.2. Informative References

[CARO02] Caro Jr., A., Iyengar, J., Amer, P., Heinz, G., and R. Stewart, "A Two-level Threshold Recovery Mechanism for SCTP", Tech report, CIS Dept, University of Delaware , 7 2002.

[CARO04] Caro Jr., A., Amer, P., and R. Stewart, "End-to-End Failover Thresholds for Transport Layer Multihoming", MILCOM 2004 , 11 2004.

[CARO05] Caro Jr., A., "End-to-End Fault Tolerance using Transport Layer Multihoming", Ph.D Thesis, University of Delaware , 1 2005.

[FALLON08] Fallon, S., Jacob, P., Qiao, Y., Murphy, L., Fallon, E., and A. Hanley, "SCTP Switchover Performance Issues in WLAN Environments", IEEE CCNC 2008, 1 2008.

[GRINNEMO04] Grinnemo, K-J. and A. Brunstrom, "Performance of SCTP-controlled failovers in M3UA-based SIGTRAN networks", Advanced Simulation Technologies Conference , 4 2004.

[IYENGAR06] Iyengar, J., Amer, P., and R. Stewart, "Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-end Paths.", IEEE/ACM Trans on Networking 14(5), 10 2006.

[JUNGMAIER02] Jungmaier, A., Rathgeb, E., and M. Tuexen, "On the use of SCTP in failover scenarios", World Multiconference on Systemics, Cybernetics and Informatics , 7 2002.

[NATARAJAN09] Natarajan, P., Ekiz, N., Amer, P., and R. Stewart,

"Concurrent Multipath Transfer during Path Failure", Computer Communications , 5 2009.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, December 2011.

Authors' Addresses

Yoshifumi Nishida
GE Global Research
2623 Camino Ramon
San Ramon, CA 94583
USA

Email: nishida@wide.ad.jp

Preethi Natarajan
Cisco Systems
510 McCarthy Blvd
Milpitas, CA 95035
USA

Email: prenatar@cisco.com

Armando Caro
BBN Technologies
10 Moulton St.
Cambridge, MA 02138
USA

Email: acaro@bbn.com

Paul D. Amer
University of Delaware
Computer Science Department - 434 Smith Hall
Newark, DE 19716-2586
USA

Email: amer@udel.edu