This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the WCNC 2007 proceedings.

An analytical estimation of the failover time in SCTP multihoming scenarios

Lukasz Budzisz[1], Ramon Ferrús[1], Karl-Johan Grinnemo[2, 3], Anna Brunstrom[3], Ferran Casadevall[1]

[1]Dept. of Signal Theory and Communication, Universitat Politècnica de Catalunya
C.\ Jordi Girona 1-3,
Campus Nord, Edificio D4,
08034 Barcelona, Spain
[lukasz, ferrus, ferranc]@tsc.upc.edu

[2]TietoEnator AB
Kanikenäsbanken 12
651 15 Karlstad, Sweden
Karl-Johan.Grinnemo@tietoenator.com

[3]Dept. of Computer Science, Karlstad University
Universitetsgatan 2
651 88 Karlstad, Sweden
Anna.Brunstrom@kau.se

Abstract— **The motivation behind this paper is a need to have a more accurate estimation of the failover time in SCTP. The traditional one, commonly used in the literature, is based on the sum of the consecutive retransmission timeouts. This is not always appropriate, especially when using the SCTP multihoming feature as a basis for achieving transport layer mobility in wireless networking scenarios, where the transition time between available paths becomes a key aspect for the optimisation. Two new factors are introduced into the proposed estimation formula to reflect the influence of the network parameters and the behaviour of the most common protocol implementations. For the proposed model, we perform a best-worst case analysis, and then illustrate it with an example of a detailed estimation. Finally, we perform simulations comparing our proposal with the traditional estimation in a typical transport layer mobility scenario including long thin networks.**

I. Introduction and Work Motivation

Support of multihoming is among the key features of the Stream Control Transmission Protocol (SCTP), a new general purpose transport protocol defined in IETF RFC 2960 [1], RFC 4460 [2], and Internet-Draft 2960-bis [3]. Multihoming allows the use of multiple source-destination IP addresses in a single association between two SCTP endpoints. These IP addresses are exchanged and verified during the initiation phase of the association, or within the lifetime of the association if the ADDIP extension [4] is supported[1], and are considered as different paths between SCTP peers. One of these paths is selected as the primary path, while all the rest are considered as backup or alternate paths. Originally, multihoming was conceived to enhance reliability in environments requiring high availability of the applications, such as signalling transport. Hence, its scope of use, defined within the protocol specification [1]-[3], is only for link redundancy, handling single retransmissions and performing the primary path failover in case the primary path becomes unreachable. Still, an intensive research on SCTP has led to experiments of using multihoming for handover management (as a part of mobility management schemes at the transport layer of the ISO/OSI protocol stack) [5] and load sharing [6].

This paper will focus on the standard SCTP failover mechanism, taking into consideration especially the transport layer mobility application of multihoming. We will analyse the traditional failover time estimation formula in such a scenario. Exposing its drawbacks, we propose necessary updates considering both network and implementation dependent factors.

The rest of the paper is organised as follows. Section II describes the SCTP failover mechanism, indicating the rules managing packet flow during the failover process. Then the packet flow is depicted in message sequence diagrams that also serve as reference points, when presenting the components introduced to the proposed formula estimating the failover time. Further, based on the presented formula, a short best-worst case analysis of the SCTP failover time is performed. Next, Section III applies the proposed formula to a generic wireless multihoming scenario, i.e., a typical example involving a long thin network. The outcome of this analysis is evaluated in Section IV through simulations of multihomed WLAN, UMTS and GERAN networks. Finally, Section V gives the concluding remarks.

[1] Note that the ADDIP extension devoted to improve SCTP handover performance is out of scope of this work.

II. SCTP FAILOVER

A. Failover mechanism

The standard SCTP failover mechanism, which works as illustrated in Figure 1, is based on the retransmission (T3-rtx) timer derived from the Transmission Control Protocol (TCP), together with its managing rules, defined in RFC 2988 [7].

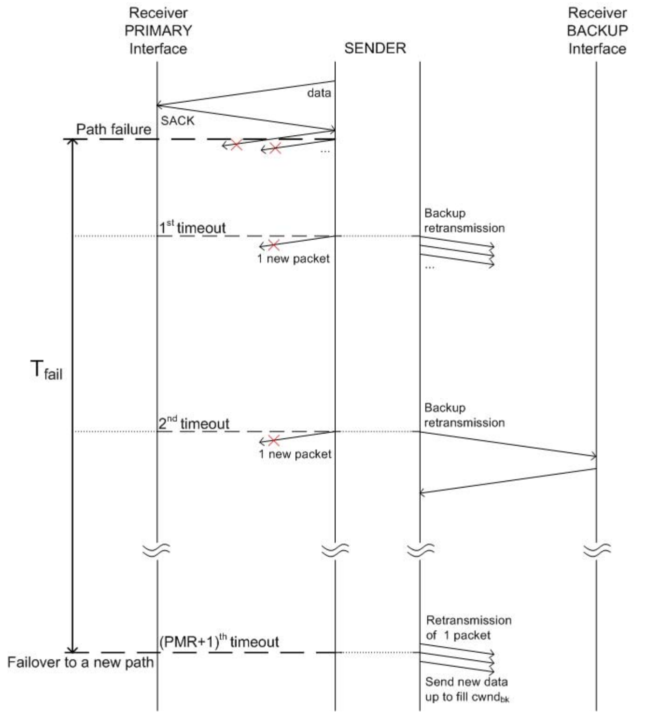

Figure 1. SCTP failover mechanism.

The T3-rtx timer is used to clock every data chunk sent to the corresponding peer in order to guarantee a reliable delivery. If the T3-rtx timer expires, and the data chunk has not been acknowledged yet by the remote peer, a so-called retransmission timeout (RTO), it is assumed that the chunk is lost, and the actual RTO value for the affected path is doubled (exponential back-off mechanism). The lost chunk, together with all the chunks that were in transition (*in flight*) towards the corresponding peer in the moment of failure, is marked for retransmission, while the outstanding data counter for that path is reset to zero. Then, SCTP starts retransmitting lost chunks according to its retransmission policy. The SCTP specification [1]-[3] recommends sending timeout retransmissions on the alternate path, provided, of course, that there is an alternate path available. The first retransmission includes all marked chunks that fit into a single data packet. Remaining marked chunks are retransmitted as soon as the congestion window on the alternate path allows. In order to clock the retransmitted data, the T3-rtx timer on the alternate path is restarted, with the current RTO value of that path. This is the main difference with respect to TCP, which has to send all retransmissions on the same, unique path. Although retransmissions are sent on the alternate path, new data is still sent on the primary path. Thus if the path failure is persistent, additional timeouts will occur. The maximum number of consecutive timeouts on the primary path is limited by the SCTP parameter Path.Max.Retrans (PMR). Once this threshold value is exceeded, the path is considered inactive, and a new primary path is selected among the alternate paths that are currently available. The protocol fails over to the selected path, and from this point on, all data chunks are sent to the new primary path. Following this discussion, and as seen in Figure 1, we define the failover time (T_{fail}) as the interval between the time when the primary path becomes unavailable, and the time, at which the first new data packet is sent to the new primary path.

B. Analytical failover time estimation

In case of a persistent failure on the primary path, the SCTP literature usually proposes a rough estimation of the failover time. This rough estimation is based on the sum of consecutive timeout periods, analogically to the back-off timer mechanism of SCTP's ancestor, TCP. Indeed, the sum of the PMR consecutive timeouts, called here *total primary path RTO expirations time* (T_{RTO}) is the most important factor in the SCTP failover time estimation. T_{RTO} is given by equation (1), combining the RTO value at the time of the path failure (RTO_{fail}), and the upper bound for the RTO value (RTO_{max}), as recommended by the SCTP specification [1]-[3].

$$T_{fail} \approx T_{RTO} = \sum_{i=0}^{PMR} \min\left(2^i \cdot RTO_{fail}; RTO_{max}\right) \qquad (1)$$

Obviously, the maximum number of allowed retransmissions (PMR) determines the impact of the exponential back-off mechanism in SCTP. Protocol specification recommends setting PMR to five, which for the lowest RTO_{fail} allowed (1s), yields a 63 second-long failover time. This is unacceptably long for the majority of communication applications. An improvement of this behaviour may be achieved by reducing the PMR value. As it was shown in [8], the faster performance is traded off against a probability of spurious failovers, or even permanent oscillations between the available addresses (*ping-pong effect*), if the PMR value is decreased too much.

The presented scheme for the failover time estimation has been applied in most of the SCTP publications so far. In this paper we aim at demonstrating that the accuracy of the estimation based only on T_{RTO} is not always satisfying, especially in transport layer mobility scenarios. The proposed update seems critical, mainly because of the two following reasons: the lack of an exact indication of the moment when the failure occur, and the ambiguity of the specification of the failover mechanism. Now we will examine both aspects in more detail.

Transport layer mobility applications typically have the PMR value reduced from the default value of five to lower values (i.e., between 0 and 2) in order to achieve shorter failover times. Therefore, there is a need to take into account an additional factor in the estimation of the failover time. The T3-rtx timer is re-started each time, a SACK arrives that acknowledges new data (rule R3 in RFC 2960 [1]). Thus, it is important to relate the moment of the path failure to the instance when the last SACK was successfully received on the primary path, and the RTO expiration accounting was started, as illustrated in Figure 2. Hereafter we will call this time *the last SACK offset* (T_{SACK}). Depending on the network state at the moment of failure, T_{SACK} may be either negative (the T3-rtx timer was restarted for the last time before the link failure) or positive (the last T3-rtx timer reset took place after the path failure, as some SACKs still managed to arrive at the sender).

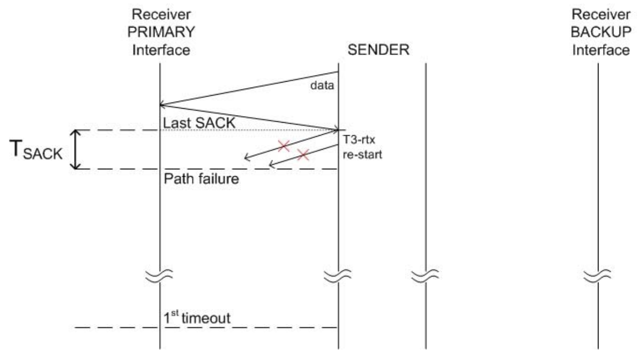

Figure 2. The last SACK offset.

Another reason for the estimation update arises from the ambiguity of the definition of the failover mechanism in the SCTP specification. The intention of the protocol authors [1]-[3] was that SCTP should be able to send new data packets to the primary path simultaneously with the retransmission of the packets that timed out, handled on the alternate path, as it was presented in Figure 1 in the previous section. However, since the main scope of the RFCs developed by the IETF is the protocol specification, but not the implementation issues, there is no unique way the protocol should behave, as far as the implementation not violating the specification rules. We have examined the most important of the existing SCTP implementations [9], [10], [11], as well as a proprietary signalling stack, and found out that most designers have interpreted the protocol's definition differently than it was intended by the authors. In particular, all major implementations seem to wait until all marked packets are retransmitted on the backup path, before transmitting any new data on the primary path. Both interpretations are compared in Figure 3.

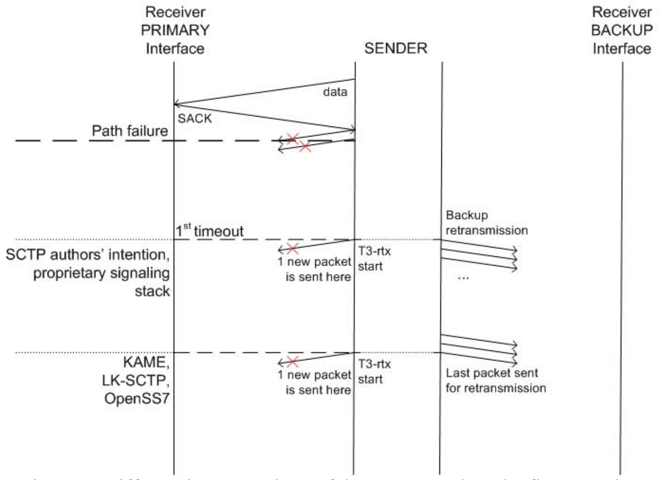
Receiver PRIMARY Interface
SENDER
Receiver BACKUP Interface
data
SACK
Path failure
1st timeout
Backup retransmission
SCTP authors' intention, proprietary signaling stack
1 new packet is sent here
T3-rtx start
...
KAME, LK-SCTP, OpenSS7
1 new packet is sent here
T3-rtx start
Last packet sent for retransmission

Figure 3. Different interpretations of the moment when the first new data packet should be sent on the primary path.

The explanation of such behaviour of the implementations lies in the use of a single output queue. All analysed implementations have only one output queue, where both types of packets (those marked for retransmission and the new ones) are directed. Further, as protocol specification prescribes, all retransmissions must be sent prior to any new data, which indeed results in the observed behaviour.

Consequently, to address this fact, we propose introducing a second new factor into the failover time estimation, *the total backup path retransmissions time* (T_{rtx}), and define it as the total time devoted to retransmit pending packets on the alternate path. It should be noted that this factor exists only if retransmissions are allowed (PMR>0) at the transport layer, and that it is implementation dependent. T_{rtx} may have vital importance in the transport layer mobility scenarios that include long thin networks (LTN), i.e., networks with high RTT values and medium/low bit rates. If retransmissions at the transport layer are available, but the PMR value is low (i.e., 1 or 2), the time devoted to retransmissions of the pending packets during the failover period will be relatively long, thus influencing the total failover time. Summing all up, we develop a new, more accurate estimation of the failover time that can be expressed as the sum of the three aforementioned components, as presented in formula (2).

$$T_{fail} = T_{RTO} + T_{SACK} + T_{rtx} \tag{2}$$

We claim that this is a general-use model, better reflecting the protocol behaviour than the estimation based only on the T_{RTO}. Although applicable to all kinds of network scenarios, equation (2) particularly targets long-thin networks. Notably, in scenarios with high PMR values, equation (1) gives a sufficient estimation of T_{fail}, as both newly introduced factors will have less relevance due to the dominant role of the exponential back-off mechanism.

C. Best-worst case analysis

Analysing the proposed estimation formula (2), it must be emphasised again that the most important factor is T_{RTO}. However, once the number of available retransmissions (PMR) is known, and the decisive T_{RTO} factor (1) determined, we can consider the overall failover time as a function of the two newly introduced factors. From that perspective, the best case (the shortest failover time) happens when the last SACK is received at the earliest possible moment before the failure occurs ($T_{SACK}<0$), and the new data packets are sent simultaneously with the retransmissions ($T_{rtx}=0$). In such case, we can estimate the lower limit for the last SACK offset as the time necessary for data packet and its corresponding SACK to traverse the primary path (the round trip time, RTT) plus the value of SACK delay (T_{del_SACK}). Usually in most applications the delayed SACK mechanism is used (a SACK is sent every second packet), so if there is only one packet correctly received, and the primary path failure occurs right before the corresponding SACK is about to be received at the sender, this produces the largest time interval between the last SACK received and the moment of the path failure.

Next, let us consider the worst case. If the last data packet received correctly arrives at the receiver shortly before the primary path failure occurs, the SACK generated may still be able to traverse the path, increasing therefore the last SACK offset to half of the RTT ($T_{SACK}>0$). However, much more important here is the influence of the backup path retransmission time. If the SCTP implementation has to wait until all packets pending retransmission are sent on the backup path, the resulting failover time will be considerably larger. Then, T_{rtx} can be estimated as a function of the number of packets pending retransmission, and the RTT expressed by means of network parameters, such as available bandwidth and latency. An example of such estimation will be presented in the next section, whereas here we summarize the T_{fail} best-worst case analysis in Table I.

TABLE I. FAILOVER TIME ESTIMATION BOUNDARIES

	BEST CASE	WORST CASE
T_{SACK}	- (T_{del_SACK} + RTT)	RTT/2
T_{rtx}	0	f (packets pending rtx; RTT) *See the estimation example in section III*

III. ESTIMATION EXAMPLE

In this section, we develop an example of an accurate estimation of T_{fail} applying a more detailed analysis to the general-use model introduced before. Such accurate failover time estimation is especially important when using the SCTP multihoming feature as the basis for achieving transport layer mobility in wireless networking scenarios. The reason is twofold. First, most mobile wireless networks can be categorised as long-thin networks. Second, an accurate knowledge of the transition time between available paths is one of the key prerequisites for handover optimisation.

A common scenario, when analysing SCTP multihoming performance in the context of transport layer mobility, consists of two paths (primary and backup) between two network hosts, e.g. a mobile node and a server, as depicted in Figure 4. Under such scenario, we assume that the server is sending data to the mobile node that is initially reachable through two IP addresses. Suddenly, due to the problems on the wireless link or as a consequence of user mobility, the address used as the primary becomes unreachable, and the server has to fail over to the alternate path using the standard SCTP failover mechanism.

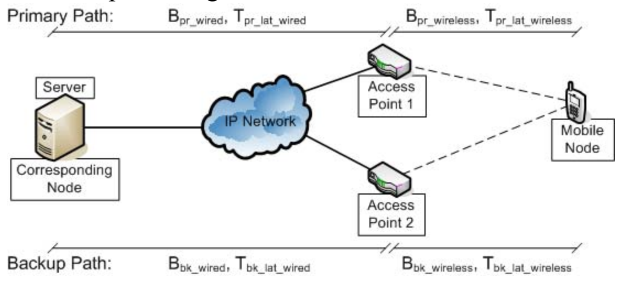
Primary Path:
B_{pr_wired}, $T_{pr\_lat\_wired}$
$B_{pr\_wireless}$, $T_{pr\_lat\_wireless}$
Server
IP Network
Access Point 1
Mobile Node
Corresponding Node
Access Point 2
Backup Path:
B_{bk_wired}, $T_{bk\_lat\_wired}$
$B_{bk\_wireless}$, $T_{bk\_lat\_wireless}$

Figure 4. Proposed scenario.

In the presented example, the proposed estimation of the failover time is based on the following parameters that model the overall network behaviour:

- Link bandwidth for the primary and backup paths denoting separately the wired (B_{pr_wired} and B_{bk_wired}), and wireless parts ($B_{pr\_wireless}$ and $B_{bk\_wireless}$),
- Latency corresponding to the wired and wireless (bottleneck) part on each of the paths (primary path: $T_{lat\_pr\_wired}$, $T_{lat\_pr\_wireless}$, and backup path: $T_{lat\_bk\_wired}$, $T_{lat\_bk\_wireless}$, respectively). Moreover, we indicate total path latency for each path (T_{lat_pr} and T_{lat_bk}).

Additionally, the following two parameters are assumed to be known at the time of the path failure:

- Congestion window ($cwnd_{fail}$),
- Retransmission time out value (RTO_{fail}).

We also assume that the receiver window size (rwnd) is not affecting the failover performance, i.e., it is not limiting the number of packets retransmitted on the alternate path. In the presented example the bandwidth of the wireless link is fully utilized. We analyse the implementation that uses the delayed SACK mechanism (a SACK is sent every second packet) with the default SACK delay value (200ms).

Now, when examining the first of the two introduced factors, T_{SACK}, it is important to relate the moment of the path failure to the instance when a SACK was received for the last time, and the RTO expiration accounting was started. In the instance, when the path failure occurs, all SACKs in flight that already managed to pass the bottleneck (we assume that the failure happens in the wireless part) will arrive at the sender, and will cause a re-start of the T3-rtx timer. As follows from the discussion in Section II.B, the accuracy of T_{SACK} must be corrected by the SACK generation interval. Aiming at the average value of T_{SACK} in our estimation, we use half of the SACK generation rate to obtain formula (3). In this equation L_{MTU} and L_{SACK} denote the size of data packets and SACK packets, respectively. As we can see the presented example fits in the estimation boundaries described in Section II.C.

$$T_{SACK} = \left(T_{lat\_pr\_wired} + \frac{L_{SACK}}{B_{pr\_wired}} \right) - \frac{1}{2} \cdot \frac{2 \cdot L_{MTU}}{B_{pr\_wireless}} \qquad (3)$$

The second of the newly introduced factors, the estimate T_{rtx} is formulated in expression (4) by means of two separate components. The first one (T_{1st_rtx}) deals with the estimation of the time spent for the first timeout retransmissions on the backup path, whereas the latter component (T_{other_rtx}) sorts out the time devoted to the remaining consecutive timeout retransmissions until the threshold PMR is reached. The T_{1st_rtx} period is completed, as soon as the last packet marked for retransmission is sent on the alternate path, and new data may be sent on the primary path, restarting therefore the retransmission timer. Then, in each of the following consecutive timeout retransmissions, only one packet is marked for retransmission. Thus, in these periods new data can be immediately sent on the primary path, and the retransmission timer on the primary path restarted accordingly. Therefore T_{other_rtx} is equal to zero, and the final formula estimating T_{rtx} can be reduced to T_{1st_rtx} only.

$$T_{rtx} = T_{1st\_rtx} + T_{other\_rtx} = T_{1st\_rtx} \qquad (4)$$

Now, we focus on the estimation of the first retransmission period. To simplify our calculations, we assume that all the packets pending retransmission will be retransmitted within the slow start phase on the alternate path, so each SACK received at the sender triggers a burst of three retransmitted packets. The number of packets retransmitted in the first burst depends actually on the packet size (L_{MTU}), however here we only consider the most typical case, i.e., the case when three packets are sent. The described retransmission scheme is presented in Figure 5.

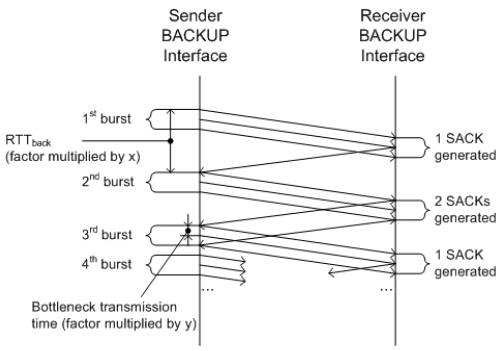
Sender BACKUP Interface
Receiver BACKUP Interface
1st burst
RTTback (factor multiplied by x)
2nd burst
3rd burst
4th burst
Bottleneck transmission time (factor multiplied by y)
1 SACK generated
2 SACKs generated
1 SACK generated

Figure 5. First timeout retransmission scheme.

Consequently, analysing the presented retransmission scheme, T_{1st_rtx} can be expressed as a function of the number of round trip times (parameter x) and the number of bottleneck transmission times[2] (parameter y) that compose the time necessary to retransmit all marked packets, as shown in (5).

$$T_{1st\_rtx} = x \cdot RTT_{bk} + y \cdot \frac{L_{MTU}}{B_{bk\_wireless}} \qquad (5)$$

The parameters x and y are, as pointed out in Figure 5, functions of the number of bursts of three packets, n, that needs to be successfully sent. The factor n, in turn, depends on the number of outstanding packets at the time of the path failure, which is assumed to be equal to the congestion window at that time, $cwnd_{fail}$. Equations (6), (7), and (8) provide the formulas for computing n, x, and y, respectively, based on the presented retransmission scheme.

$$n = \left\lfloor \frac{cwnd_{fail} - 1}{3} \right\rfloor \qquad (6)$$

$$x = \left\lfloor \frac{1 + \sqrt{1 + 8 \cdot (n-1)}}{2} \right\rfloor \qquad (7)$$

$$y = 2 \cdot n - \sum_{i=1}^{x} i \qquad (8)$$

The round-trip time on the backup path (RTT_{bk}) in equation (5) can be estimated as a function of network parameters, as presented in (9).

$$RTT_{bk} = 2 \cdot T_{lat\_bk} + \frac{L_{MTU} + L_{SACK}}{B_{bk\_wired}} + \frac{L_{MTU} + L_{SACK}}{B_{bk\_wireless}} \qquad (9)$$

[2] Usually the transmission time through the wireless link.

IV. SIMULATION RESULTS

In order to evaluate the proposed formula for estimating the failover time (equation (2)), and compare it to the one currently used (equation (1)), we use the University of Delaware's SCTP model for the ns-2 network simulator [12] assuming the topology as presented in Figure 4. The wired links have 100 Mbps bandwidth and 5ms delay. For wireless links we consider three typical long-thin networks: WLAN (11 Mbps bandwidth and 15 ms delay), UMTS (384 kbps and 80 ms), and GERAN (80 kbps and 80 ms), respectively. The failover is modelled as an immediate cut of the primary path. For each type of the wireless network we perform the simulations for 40 different values of the $cwnd_{fail}$, calculating the average value with the 95% confidence interval. Figure 6 presents the outcome of the evaluation in terms of the average estimation error (i.e., relative error between the simulation result T_{sim} and the estimated value T_{fail}) as a function of the PMR parameter.

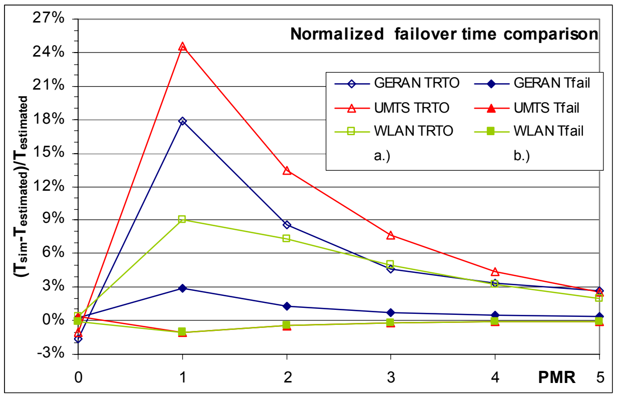

Figure 6. Comparison of the normalised failover time in long-thin networks: currently used scheme - T_{RTO} (a), and our proposal - T_{fail} (b).

As shown in Figure 6, the biggest error in the failover time estimation, using the most common formula (1), is obtained for PMR = 1 or 2 (the typical configuration of SCTP in wireless multihoming scenarios), when the retransmissions on the alternate path have the most significant influence. The estimation error can be considerably reduced (below 3%), if the proposed estimation scheme (2) is applied. As expected, for big PMR values (i.e., PMR = 4 or 5) T_{RTO} completely dominates T_{fail}, hiding the influence of all other factors that were commented in this paper. In such case, our proposal (2) gives nearly the exact value of the failover time (the error value below 0,5%), but there is no need to improve the commonly used estimation (1), which works well enough (the error value below 3%). Finally, for PMR=0 (i.e., $T_{rtx} = 0$) equation (2) also gives an improvement in the failover time estimation, due to the introduction of the T_{SACK} component. However, this improvement is rather small.

V. CONCLUSIONS

In this paper, we proposed a new general-use estimation formula for an accurate estimation of the failover time in SCTP multihoming scenarios. We demonstrated that the estimation commonly used in the literature, based only on the sum of consecutive timeouts, is not always appropriate. Especially in transport layer mobility scenarios, the proposed update seems crucial, mainly because of the following reasons: the number of allowed retransmissions (PMR) is usually low, reducing the impact of the exponential back-off mechanism; the lack of an exact indication of the moment when the failure occurred; and the ambiguity of the specification of the failover mechanism. We stressed the influence of network, and implementation dependent parameters, and introduced two new factors into the proposed estimation. Both factors have general scope of use, however we focused on a typical transport layer mobility scenario, since this application has great importance for current SCTP research. We applied our estimation formula on a generic wireless multihoming scenario, and showed through simulations that the proposed formula would most likely give more accurate estimates of the failover time, than the currently prevailing scheme.

ACKNOWLEDGEMENT

This work has been partially supported by the NoE NEWCOM (https://newcom.ismb.it) from the European Community and by the TIC2004-00518 grant from CYCIT (Spanish National Science Council).

REFERENCES

[1] R. Stewart, Q. Xie, et al., "Stream Control Transmission Protocol", IETF RFC 2960, October 2000; www.ietf.org/rfc/rfc2960.txt.
[2] R. Stewart, I. Arias-Rodriguez, K. Poon, A. Caro, and M. Tuexen, "Stream Control Transmission Protocol (SCTP) Specification Errata and Issues", IETF RFC 4460, April 2006; www.ietf.org/rfc/rfc4460.txt.
[3] R. Stewart, "Stream Control Transmission Protocol (SCTP)", IETF draft, June 2006; < draft-ietf-tsvwg-2960bis-02.txt>, work in progress.
[4] R. Stewart, M. Ramalho, et al., "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", IETF draft, October 2006; <draft-ietf-tsvwg-addip-sctp-16.txt>, work in progress.
[5] L. Ma, F. Yu, V. Leung and T. Randihawa, "A New Method to Support UMTS-WLAN Vertical Handover Using SCTP", IEEE Wireless Communications Magazine, August 2004, pp.44-51.
[6] J. Iyengar, P. Amer, and R. Stewart, "Concurrent multipath transfer using transport layer multihoming - performance under varying bandwidth proportions" Proc. IEEE Military Communications Conference (MILCOM 2004), October 2004.
[7] V. Paxon, M. Allman, "Computing TCPs Retransmission Timer", IETF RFC 2988, November 2000; www.ietf.org/rfc/rfc2988.txt.
[8] A. Caro, P. Amer, and R. Stewart, "End-to-end failover thresholds for transport layer multihoming," Proc. IEEE Military Communications Conference (MILCOM 2004), October 2004.
[9] KAME Project SCTP implementation, http://www.kame.net/
[10] Linux Kernel SCTP (LK-SCTP) Implementation, http://www.sctp.org/
[11] Open SS7 SCTP implementation, http://www.openss7.org/
[12] A. Caro, "ns-2 SCTP module", www.armandocaro.net/software/ns2sctp/