Delay Penalty during SCTP Handover

Case Study Final Report

EUROPEAN UNION
European Regional
Development Fund

Investing in the future by working together for a sustainable and competitive region

The Interreg IVB
North Sea Region
Programme

***Abstract*—The rapidly growing interest in untethered Internet connections such as WLAN and 3G/4G mobile connections, calls for intelligent session management, not least in terms of handovers. As part of an effort to develop a SCTP-based session management framework, we are studying ways of improving the SCTP handover delay for real-time traffic by optimizing the startup delay on the handover-target path. We have developed a theoretical model that predicts the transfer times of SCTP messages during the startup on a new path. This paper validates our model. It shows that the model can be used to predict message transfer times in variable bitrate flows. The paper further employs our model to study the startup delay penalty during handover for the spectrum of network conditions considered relevant for real-time traffic over mobile connections.**

I. Introduction

In the past few years, we have seen a surge of interest in untethered Internet connections, e.g., smartphones roaming between wireless WLAN/3G/4G networks and wired corporate/home networks. A major issue when it comes to untethered Internet connections is how to accomplish a smart session management that provides for swift handovers. We are currently working on a smart session management framework for mobile Internet connections based on the Stream Control Transmission Protocol (SCTP) [1]. As part of this work, we are studying ways of improving the delay incurred by slow start during the resumption of a session between the mobile device and the target end point. To assist us in this endeavor, we have developed a theoretical formula for the transfer time of real-time SCTP messages during slow start [2]. This paper experimentally validates our formula. The paper also predicts the startup delay penalty for the spectrum of network conditions considered relevant for real-time traffic over mobile connections. A key observation made is that a video conferencing application, or other real-time applications with similar requirements, could experience handover delay penalties of more than 500 ms in a mobile, wireless setting.

Our theoretical formula is related to the huge body of existing work on the modeling of the TCP and SCTP congestion control algorithms. Of note, is the seminal work of Padhye et al. [3]. More closely related to our work are the SCTP congestion control models in [4] and others. They considered the performance of SCTP in handover scenarios, however, contrary to us, they primarily considered average transfer times for bulk-data transfers, and did not theoretically model real-time traffic, and the transfer times of individual messages.

The remainder of this paper is organized as follows. Section II presents our proposed SCTP-based framework for providing smart session management to mobile Internet connections, and as such provides a background to our work on minimizing the handover startup delay. Section III presents and validates our formula. Section IV demonstrates that our formula could be used to predict the message transfer times in a variable bitrate flow. Section V uses the formula to explore the startup delay penalty during handover for network conditions considered relevant for real-time traffic over mobile connections. Finally, Section VI concludes the paper.

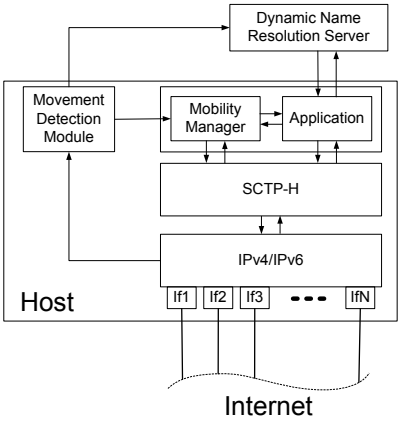


Fig. 1. Architectural overview of our session management framework.

II. A SCTP-based Session Management Framework

Our work on improving the startup delay during an SCTP handover is part of an effort to build a session management framework for mobile Internet connections [5]. Particularly, the findings from our work will be implemented in a version of SCTP, SCTP-H, which specifically targets mobile Internet devices and their handover requirements. An architectural overview of our framework is provided in Figure 1.

The Movement Detection Module is a process that runs inside each mobile device. Its purpose is to continuously monitor the status of each of the device's network interfaces. The Movement Detection Module maintains a state for each interface. Events such as bringing an interface up or down, or associating with a wireless access point trigger transitions between the interface states.

Each application on a mobile device that utilizes the session management framework incorporates a Mobility Manager. Apart from keeping track of all network interfaces, the Movement Detection Module is responsible for sending notifications about interface state changes to the applications' Mobility Managers and to the Dynamic Name Resolution Server.

The Mobility Manager contains the intelligence that enables an application to properly react to network interface events. The notifications received from the Movement Detection Module together with a set of policies determine the actions the Mobility Manager should take.

Depending on its current location, a mobile device might have zero, one, or several active network interfaces. Thus, it is not feasible to use the IP addresses as a way of identifying a mobile device. The Dynamic Name Resolution Server translates the name given to a mobile device into its currently used IP addresses.

III. Description and Validation of our Formula

To design SCTP-H, we need to acquire an appreciation for the delay penalty caused by slow start during startup on the path between a mobile device and the corresponding endpoint. To this end, we have developed a theoretical formula for the transfer time of real-time SCTP messages during slow start [2].

This section explains and validates our formula. The formula is based on some assumptions:

- The packet transmission time is negligible, which is reasonable if the capacity of the link is high in relation to the link delay.
- All messages are of the same size, which is typical for CBR traffic.
- The network paths are symmetrical. In Formula 1 the minimum message transfer time for a message is estimated as half the round-trip time. This is, however, easily generalized if the one-way delay is known.
- The SACK delay is disabled. The formula does not account for different RTTs, which could be the case if the SACK delay is enabled.
- There are no packet losses or retransmissions, since modelling the transfer time for individual mesages in a lossy network is not feasible.
- The message interarrival time is less than the retransmission timeout (RTO), and consequently there are no adjustments of the congestion window due to long idle times.
- There are no messages queued at the time of the initialization of slow start. This is reasonable as SCTP provides a soft handover mechanism and real-time streaming is only feasible if the capacity of the path is sufficiently large to carry the load.

In the following, the message transfer time for the nth message during slow start is denoted MTT_n. Obviously, MTT_n has to be at least half the round-trip time (RTT). In addition, a message could experience a queuing delay, D_n, when it arrives at SCTP at times when the congestion window is full. Taken together, this gives us the formula in (1) for MTT_n.

$$MTT_n = \frac{RTT}{2} + max(0, D_n) \quad (1)$$

Figure 2, illustrates a transmission of the first 12 messages during slow start and the delay experienced by message 7, D_7. The initial congestion window is assumed to be able to hold four messages. It follows from Figure 2 that D_7 depends on three components:

1) the number of transmission rounds before the 7th message is sent (p_7);
2) the arrival time of the 7th message from the application to SCTP (α_7); and,
3) a delay offset that is related to the arrival rate of SACKs at the Source (Δ_7).

Analytically, the dependence could be expressed as, $D_7 = p_7 RTT - \alpha_7 + \Delta_7$, which generalized to an arbitrary message gives the formula in (2) [1].

$$D_n = p_n RTT - \alpha_n + \Delta_n \quad (2)$$

The formula in (2) was validated in a series of tests in an Emulab [7] testbed. The network topology comprised two

[1]A full derivation of the formula is found in [6].

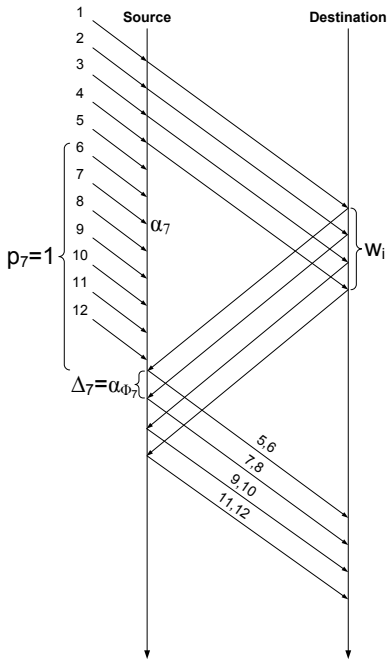


Fig. 2. Message transfers during slow start.

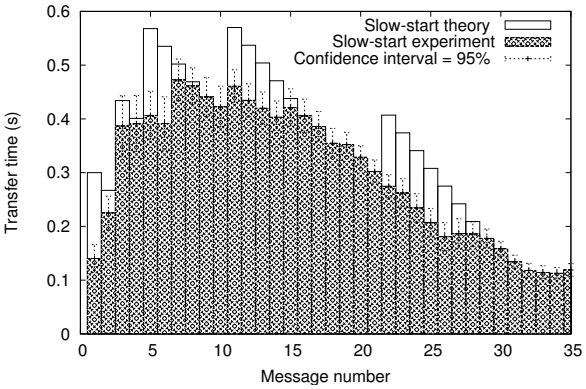
end systems connected to each other by a single network path that passed through a *dummynet* [8] node. To cover a broad spectrum of networks, ranging from local area to wide area networks, tests were run for link delays: 5 ms, 25 ms, and 100 ms.

Both end systems were Intel 64-bit Xeon 3.0 GHz machines running the Redhat Fedora Core 6 (2.6.20 kernel) operating system. At both end systems, SCTP was run with its default settings, e.g., the initial congestion window was 4380 Bytes. We used a custom-built traffic generator to generate traffic between the source and sink end systems. To make a straightforward validation possible, the tests were run with constant bitrate (CBR) traffic. The traffic was selected to cover a large number of traffic types. Particularly, it comprised messages of sizes, 250 Bytes, 1452 Bytes[2], and 4000 Bytes, and ran with message interarrival times (λ) of 2 ms, 10 ms, and 50 ms. Each test was repeated 50 times, and the mean MTT_n during slow start, over all repetitions was taken as a metric for the measured MTT_n, henceforth denoted $\overline{MTT_n}$.

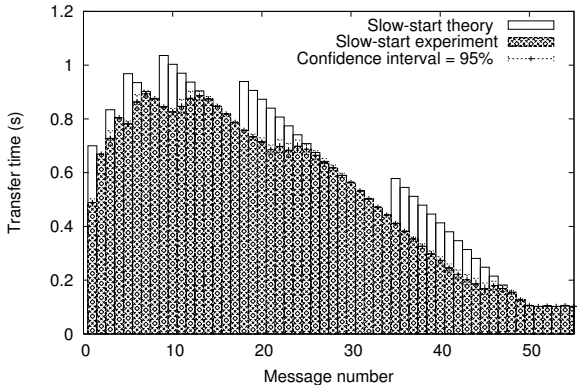
To evaluate $\overline{MTT_n}$ in a test against the results predicted by our formula, MTT_n^p, we computed the average relative prediction error, $\bar{\epsilon}_r$[3]. Table I presents part of the results of our validation tests. Particularly, it presents the results of the tests with s = 1452 Bytes, and in which slow start had an actual effect on the MTTs. We observe that we have a good compliance between the measured and the predicted MTTs. The differences that still existed were primarily attributed to variations in the link delays and variations in the interarrival

[2]The message size 1452 Bytes was selected since it fits exactly into one Ethernet frame.

[3]The difference between the estimated value and the outcome of the experiment in relation to the experimental value.



(a) The "Horizon Talk Show".



(b) The "NBC 12 News".

Fig. 3. Predicted and measured MTTs for VBR traffic (RTT = 200 ms).

TABLE I
PREDICTION ERROR IN THE TESTS WITH s = 1452 BYTES.

RTT (ms)	λ (ms)	$\bar{\epsilon}_r$ (%)
10	2	9.5
50	2	4.1
50	10	5.4
200	2	2.9
200	10	8.0

times of messages generated by our traffic generator.

IV. PREDICTING VBR TRAFFIC WITH CBR TRAFFIC

In previous work [2], we argued that the average intensity of the traffic rather than its distribution determines the delay penalty caused by slow start. Particularly, it was argued that the delay penalty experienced by messages in a variable bitrate (VBR) flow could be quite accurately predicted by a CBR flow of messages whose size equaled the average VBR message size. To verify this argument, we used the same testbed as in the validation tests. We let our traffic generator generate traffic in accordance with two video trace files selected from the Arizona State University Video Trace Library [9], [10]: the "NBC 12 News" (H.264 SVC, average frame size = 32297 Bytes) and the "Horizon Talk Show" (HQ H.264, average frame size = 6395 Bytes). These trace files covered a bandwidth spectrum from roughly 1.7 Mbps up to approximately 8 Mbps, and thus represented very different frame sizes and bitrates. As before, each test was repeated 50 times.

Figures 3(a) and 3(b) show the results of the tests for the "Horizon Talk Show" and the "NBC 12 News" video traces with an RTT of 200 ms. As seen from the figures, our formula quite accurately predicted the MTTs during slow start for both types of VBR traffic. It is seen that the model performs slightly worse in the delay peaks, and that the delay peaks come at higher message numbers in the experimental results compared to the calculated results. The main reason behind this difference is that RFC 4960 permits the size of the SCTP send window to exceed the congestion window by almost one maximum transmission unit (MTU), something that is not taken into account by our formula.

V. PREDICTING STARTUP DELAYS FOR REAL-TIME TRAFFIC

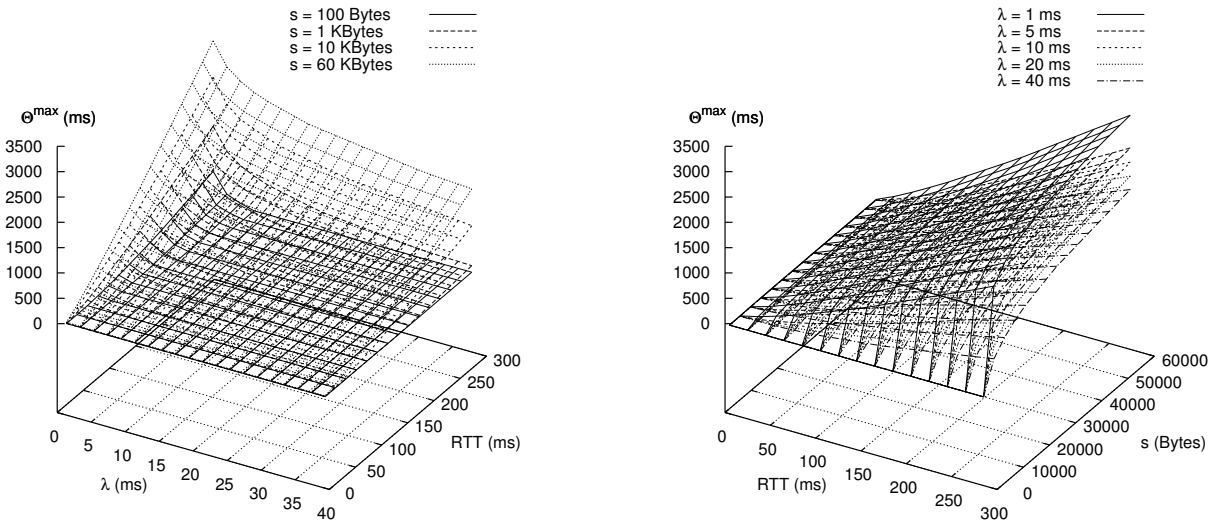
On the basis of our formula, we have explored the startup delay penalty due to slow start for the ranges of RTTs, message sizes, and message interarrival times that are considered relevant for mobile, wireless real-time traffic scenarios. We have used our formula to predict the startup delay penalty for RTTs in the interval: 5 ms to 300 ms; message sizes in the interval: 100 Bytes to 60 000 Bytes; and message interarrival times in the interval of 1 ms to 50 ms. The initial congestion window was at all times set to 4380 Bytes [1].

We used the maximum delay penalty experienced by any message during slow start as a metric for the startup delay penalty. The metric, henceforth denoted Θ^{max}, was computed as given by (3), where Ω denotes the set of all messages sent during slow start.

$$\Theta^{max} = \max_{n \in \Omega} \{MTT_n^p - RTT/2\} \tag{3}$$

Obviously, this metric does not convey the complete picture of the startup delay penalty during slow start. However, studying the message transfer times of single messages becomes unwieldy and hard to analyze. Also, by the way slow start works, the maximum delay penalty imposed by slow start is more or less proportional to the number of messages affected by slow start. Thus, typically a large Θ^{max} means that a large number of messages was affected by slow start.

Figure 4(a) illustrates how the startup delay penalty varies with the message interarrival time and the RTT for message sizes ranging between 100 Bytes and 60 KBytes. The graphs show clearly how the delay penalty increases with smaller interarrival times and longer RTTs. At short RTTs, especially in combination with small message sizes, slow start has more or less no impact at all on the MTT. However, it starts to



(a) Θ^{max} as a function of λ and RTT.

(b) Θ^{max} as a function of RTT and s.

Fig. 4. Prediction of the startup delay penalty during slow start.

impede on MTT at message sizes above 1000 Bytes; and, e.g., with a message size of 10 kBytes, an RTT of 115 ms, and a message interarrival time of 33 ms, i.e., a traffic scenario that roughly corresponds with a mobile, wireless video conferencing scenario, we have a Θ^{max} of almost 500 ms.

Figure 4(b) shows how the startup delay penalty varies with the RTT and the message size for message interarrival times in the interval of 1 ms to 40 ms. From a video traffic perspective, it is particularly interesting to observe the impact of slow start for interarrival times in between 20 ms and 40 ms, a range that covers popular video encoding formats such as MPEG-4 and H.264. We observe that for RTTs below 100 ms, and message sizes less than a couple of kilobytes, slow start has little effect on the message transfer times. However, with message sizes in the order of 10 KBytes and RTTs approaching 200 ms, the impact becomes quite significant. In fact, the figure predicts that some messages in a video flow (e.g., frames of size 30 KBytes that are transmitted at a frame rate of 30 frames/s) over a 3G wireless link with an RTT of 200 ms [11] experience delay penalties in the vicinity of 1 s.

VI. Conclusion

We are currently underway of implementing a smart session management framework for mobile Internet connections based on the SCTP transport protocol. As part of our work, we are studying ways of improving the SCTP handover delay, particularly how to optimize the startup delay on the target path. To determine the theoretically feasible gains of modifying the SCTP startup behavior on the new path, we have developed a formula that predicts the transfer times of SCTP messages during slow start. This paper validates our formula through a series of experiments. Also, it demonstrates that our formula could be employed to accurately predict message transfer times in VBR traffic by approximating the VBR flow with a CBR dito. It uses this finding to explore the startup delay penalty during handover in mobile, real-time traffic scenarios, and suggests that data-intense traffic, such as video, could experience delay penalties of 500 ms and more as the round-trip time goes beyond 100 ms; something that could adversely affect the service quality as perceived by the end user. As a next step, we intend to evaluate the consequences of introducing a bandwidth-aware startup scheme in SCTP.

Acknowledgment

The authors would like to thank the Flux Research group at the University of Utah for providing the Emulab testbed. The work has been supported by grants from VINNOVA, Swedish Governmental Agency for Innovation Systems.

References

[1] R. Stewart, "Stream control transmission protocol," RFC 4960, Sep. 2007.

[2] J. Eklund, K.-J. Grinnemo, and A. Brunstrom, "Theoretical analysis of an ideal startup scheme in multihomed SCTP," in *Proc. Network Services and Applications – Engineering, Control and Management (EUNICE)*, Trondheim, Norway, Jun. 2010.

[3] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: A simple model and its empirical validation," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 133–145, Apr. 2000.

[4] T. D. Wallace and A. Shami, "An analytic model for the stream control transmission protocol," in *IEEE Global Telecommunications Conference (GLOBECOM)*, Miami, Florida, U.S., 2010, pp. 1–5.

[5] G. Cheimonidis, "An Internet mobility framework based on the stream control transmission protocol," Master's thesis, KTH Royal Institute of Technology, Stockholm, 2010.

[6] J. Eklund, K.-J. Grinnemo, and A. Brunstrom, "Impact of slow start on SCTP handover performance," in *Accepted for publication in Flexibility in Broadband Wireless Access Network (FlexBWAN) Workshop*, Hawaii, US, Aug. 2011, To Appear.

[7] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *Proc. Fifth Symposium on Operating Systems Design and Implementation*, Boston, Massachusetts, U.S., Dec. 2002.

[8] Dummynet homepage. [Online]. Available: http://info.iet.unipi.it/ luigi/dummynet/

[9] G. V. Auwera, P. T. David, and M. Reisslein, "Traffic and quality characterization of single-layer video streams encoded with H.264/AVC advanced video coding standard and scalable video coding extension," *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 698–718, 2008.

[10] P. Seeling, M. Reisslein, and B. Kulapala, "Network performance evaluation using frame size and quality traces of single-layer and two-layer video: A tutorial," *IEEE Communications Surveys and Tutorials*, vol. 6, no. 2, pp. 58–78, 2004.

[11] Z. Faigl, S. Lindskog, and A. Brunstrom, "Performance evaluation of IKEv2 authentication methods in next generation wireless networks," *Security and Communication Networks*, vol. 3, no. 1, pp. 83–98, 2010.