Implications of using a Large Initial Congestion Window to Improve mSCTP Handover Delay

Johan Eklund
Department of Computer Science
Karlstad University
Sweden
Email: johan.eklund@kau.se

Karl-Johan Grinnemo
Department of Computer Science
Karlstad University
Sweden
Email: karl-johan.grinnemo@kau.se

Anna Brunstrom
Department of Computer Science
Karlstad University
Sweden
Email: anna.brunstrom@kau.se

***Abstract*—The currently rather heterogeneous wireless landscape makes handover between different network technologies, so-called vertical handover, a key to a continued success for wireless Internet access. Recently, an extension to the Stream Control Transmission Protocol (SCTP) – the Dynamic Address Reconfiguration (DAR) extension – was standardized by IETF. This extension enables the use of SCTP for vertical handover. Still, the way vertical handover works in SCTP with DAR makes it less suitable for real-time traffic. Particularly, it takes a significant amount of time for the traffic to ramp up to full speed on the handover target path. In this paper, we study the implications of an increased initial congestion window for real-time traffic on the handover target path when competing traffic is present. The results clearly show that an increased initial congestion window could significantly reduce the transfer delay for real-time traffic, provided the fair share of the available capacity on the handover target path is sufficiently higher than the send rate required by the real-time flow. Additionally, we notice that this performance gain comes without penalizing the competing traffic.**

***Keywords*-SCTP; dynamic address reconfiguration; video; mobility; handover; congestion control; slow start**

I. Introduction

Wireless networks comprises a variety of technologies, e.g., cellular (3G/4G), WiFi, and WiMAX. Additionally, the number of terminals with multiple wireless interfaces to access Internet is rapidly increasing. Ubiquitous connectivity is made possible by the ability for a single device to roam between heterogeneous networks. The goal for this, so-called, vertical handover is to be transparent to the end user.

The Stream Control Transmission Protocol (SCTP) [1], with its multihoming feature, and its extension for Dynamic Address Reconfiguration (DAR) [2], a.k.a. mSCTP, has become a promising alternative for vertical handover. The DAR mechanism enables for SCTP to dynamically add and remove IP addresses to an ongoing session.

However, to be able to provide a seamless handover, mSCTP needs to have an efficient handover detection mechanism, particularly a mechanism that anticipates the loss of connectivity to the current access point. Furthermore, mSCTP has to start up swiftly on the handover target path. Although a vertical handover detection mechanism is indeed important, several works have already studied this issue. Thus, in this paper it is assumed that we have an optimized handover mechanism that contributes marginally to the handover delay. Instead, this paper focuses on the second issue: the startup on the handover target path. In fact, earlier studies [3], [4] have shown that even in a scenario with an ideal handover detection mechanism, the mobile terminal may experience a non-negligible service disruption, due to the startup phase on the handover target path. We have also seen that a way to mitigate this startup delay after handover could be to increase the initial congestion window (*init_cwnd*) on the handover target path.

This paper is a continuation of our previous study on using an increased *init_cwnd* to improve the vertical handover performance of mSCTP. Particularly, the paper studies the implications of an increased *init_cwnd*, in terms of latency and fairness, in a situation with competing traffic. We study the impact of using an *init_cwnd* for the mSCTP traffic, on both mSCTP and the competing traffic.

The paper considers the effects on real-time video traffic (H.264, HQ) in scenarios, which are intended to model a vertical handover from an arbitrary wireless network to a cellular 3G or 4G network. The competing traffic on the handover target path consists of elastic background traffic. In this case, one or several TCP flows.

The results of our study suggest that an increased *init_cwnd* could significantly reduce the handover delay, provided the capacity required by the video traffic is significantly lower than its fair share on the handover target path. The results also suggest that this performance gain has no effect on the short-time fairness to competing traffic.

The remainder of this paper is organized as follows. Section II gives an overview of SCTP with a focus on its support for multihoming. Next, Section III discusses the setup and methodology used in our experimental study. The results from the study are presented and discussed in Section IV. Section V surveys related work. Finally, Section VI concludes the paper and briefly mentions ongoing and future work.

II. Preliminaries

The Stream Control Transmission Protocol (SCTP), originally developed as a transport protocol to serve telephony call-control signaling, is today standardized as a general-purpose transport protocol in RFC 4960 [1]. Still, new developments on SCTP are being made. Several of the current standardization activities are described in Dreibholz et al. [5]. Some extensions have been standardized in separate RFCs [2], [6], [7].

SCTP does in many ways mimic TCP SACK [8]; It is a reliable, connection-oriented transport protocol that offers a selectively-acknowledged, non-duplicated transfer of packets. Further, it uses window-based congestion- and flow-control mechanisms that essentially work the same as in TCP SACK. In default mode, data is delivered to the application in ordered mode, while unordered or partial-ordered delivery, which could be suitable for real-time traffic, is optional. To distinguish an SCTP connection from a connection in TCP, SCTP uses the term “association”.

However, one major extension to SCTP is the multihoming feature, which implies that an association is able to connect to several IP addresses at both the source and destination endpoints. Normally, SCTP selects one of its peer’s destination addresses as the primary destination address. The remaining addresses serve as alternate or backup addresses. If the primary destination address becomes unavailable, a failover procedure takes place, which results in the traffic being re-routed to one of the alternate addresses.

The Dynamic Address Reconfiguration (DAR)) [2] extension of SCTP enables for an SCTP endpoint to dynamically alter its IP addresses during the lifetime of an association. Apart from permitting IP addresses to be dynamically added to and removed from associations, the DAR extension provides for an SCTP endpoint to explicitly change its peer’s primary destination address. The DAR extension is key to facilitate transport-level mobility and several transport-level mobility solutions that build on SCTP extended with DAR (mSCTP) have been proposed [9], [10], [11], [12].

III. Experimental Methodology

The experiment models a scenario where a mSCTP-based, video session is handed over to a network, where elastic TCP traffic is present. Our scenario is illustrated in Figure 1. Initially, one or several video sessions are set up between a stationary and a mobile terminal, utilizing Path 1.

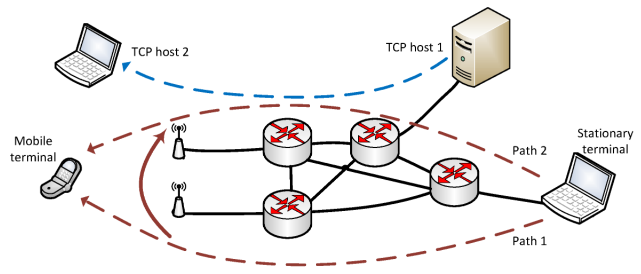

Fig. 1. Schematic view of our experimental scenario.

Some time after the video sessions has left the slow-start phase, the sessions are handed over to Path 2, where the bottleneck section of the path is shared between the mSCTP video sessions and the competing traffic.

The experiment comprised end hosts, which ran real implementations of mSCTP and TCP. The network was emulated utilizing the KauNet network emulator [13], which is an extension to the Dummynet emulator [14]. A view of the experiment setup is depicted in Figure 2.

The video traffic mimicked video traffic from a standard definition (H.264, HQ) video clip, an episode of the “Horizon Talk Show”, made available by the Arizona State University Video Trace Library [15], [16]. The mSCTP traffic was generated by a Custom-made Traffic Generator (CTG). The clocks of the end hosts were synchronized, and all messages were time stamped at departure from the sending application and at reception by the target application.

The competing traffic on the handover target path consisted of bulk TCP flows. The motivation behind this choice was to get a view of the impact of a large *init_cwnd* on the competing traffic, as well as to study the behavior of a video flow in a scenario with competing traffic. The competing traffic was generated by the Iperf [17] traffic generator.

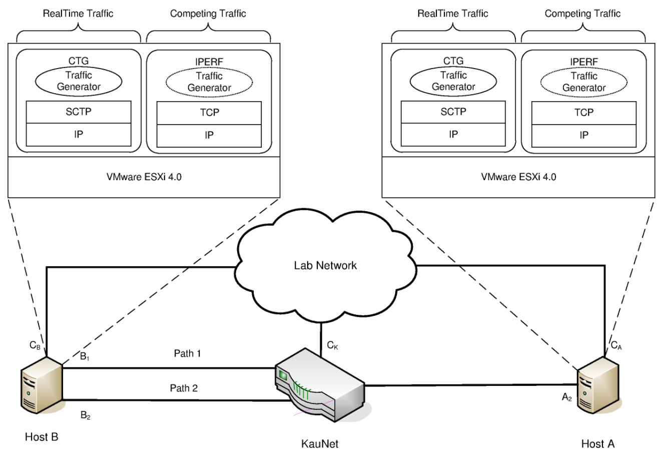

Fig. 2. Experiment setup.

A. Parameters

The video traffic was sent at Constant Bit Rate (CBR)[1] and the average send rate was 1.53 MBps, generated at a rate of 6395 bytes messages every 33 ms. The networks considered in this study were a high latency, low capacity (*lc*) network (RTT 300 ms, bandwidth 6 Mbps) with characteristics similar to a 3G cellular network, and a lower latency and “high” capacity (*hc*) network (RTT 40 ms, bandwidth 45 Mbps), i.e., characteristics more like a 4G cellular network.

The most important parameters of our experiments are listed in Table I. Since the characteristics of the considered networks were very diverse, the number of video flows as well as the number of competing TCP flows varied between the *lc* and the *hc* experiments. Particularly, we conducted the *lc* network experiments with 1 and 2 video flows and with 1 and 2

[1] Video traffic is generally generated at Variable Bit Rate (VBR), but since we consider average results of 40 repetitions, see below, we consider CBR traffic to be appropriate.

competing flows, and the *hc* network experiments with 1,2 and 5 video flows and with 2, 5 and 10 competing flows.

TABLE I
PARAMETERS FOR THE DIFFERENT NETWORK TYPES

	lc	*hc*
Bandwidth (MBps)	6	45
RTT (ms)	300	40
Competing flows	1, 2	2, 5, 10
mSCTP associations	1, 2	1, 2, 5
init_cwnd (MSS)	3 (default), 10, 20, NR	3 (default), 10, 20, NR

The Maximum Segment Size (MSS) for a datagram was in the experiment set to 1500 Bytes. The experiments were run with an *init_cwnd* of 3 MSS (default), 10 and 20 MSS for the video traffic. Furthermore, to obtain an appreciation of the startup performance in a scenario where the *init_cwnd* did not impose any restriction on the video transfer, we utilized an *init_cwnd* of 50 KBytes (NR), that is an *init_cwnd* larger than the maximum number of outstanding packets in any of the considered experiments. The competing traffic was sent with the default *init_cwnd* of 3 MSS. Well aware of the ongoing discussion on the size of the router buffers [18], we conducted the experiments with a router buffer of one BDP (225 KB)[2], a typical recommendation in the literature. The send and receive buffers, as well as the maximum $ssthresh$ of the end hosts were set to large sizes, not to impose any restrictions on the video transfer.

IV. RESULTS

In every experiment, the transfer times of the individual messages in the video flow (MTTs) were measured. The MTT was measured as the time from the generation of a message by the source application until the message was received by the destination application. Additionally, in all experiments we extracted the Maximum MTT ($MMTT$). Previous studies [4] suggest that the $MMTT$ is related to the number of messages being affected by the handover delay. To be able to analyze whether or not the size of the *init_cwnd* had any impact on the competing traffic, we monitored the total throughput of the competing TCP flows. The throughput was sampled every 5 seconds. Henceforth, we call every sample a measurement point.

For normality assumptions to apply, we chose to repeat each experiment 40 times, and from the results we calculated the average value together with a 95% confidence interval. In the remainder of this section, the results from the *lc* and *hc* handover experiments are presented.

A. Handover to a low capacity network

The long transfer times, and the restricted bandwidth are parameters that are crucial for the handover performance in *lc* networks. Figure 3 represents a scenario where one video

[2]The BDP for the *lc* and the *hc* networks turned out to be the same, although the bandwidths and the latencies differed.

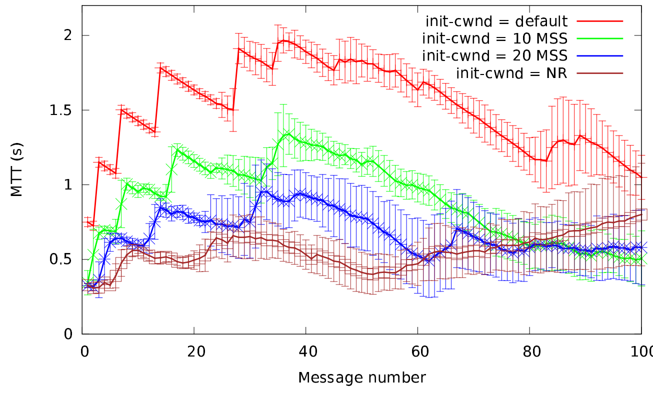

Fig. 3. MTT for one video flow. Handover of one video flow to a low capacity network with one competing flow.

flow is handed over to the target path. The graphs represent the average *MTT*s for the messages in the video flow during startup with one competing TCP flow present. The different graphs represent the different *init_cwnd*s. In the figure, it is seen that an increase of the *init_cwnd* resulted in a reduction of the average *MTT* for a message. Further, it follows that an increased *init_cwnd* decreased the $MMTT$. Particularly, there was a major reduction in $MMTT$ when the *init_cwnd* was increased from its default value of 3 MSS up to 10 MSS, while there was a smaller, but still significant, reduction in $MMTT$ as the *init_cwnd* was further increased up to 20 MSS. The large confidence intervals were due to retransmissions of some lost messages during the startup phase. It should be noted that these results are in line with the results from our previous work without competing traffic [4].

Figure 4 shows the throughput evolution for the competing TCP flow as the video flow is handed over. The video flow was handed over to the target path some time after the TCP traffic was started, giving enough time for the TCP flow to reach its stationary phase.

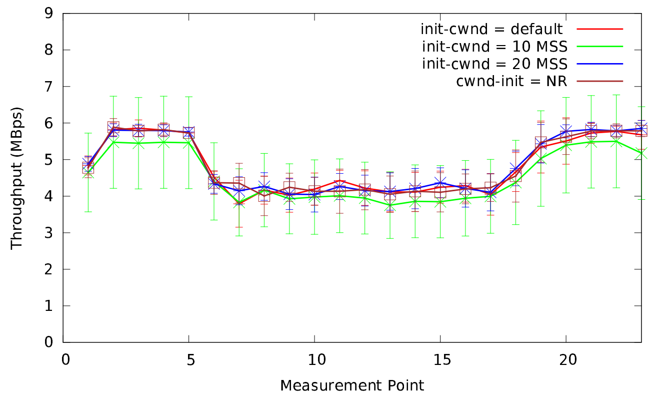

Fig. 4. Throughput for competing traffic. Handover to a low capacity network with one competing flow.

As seen from the figure, the video flow started up on the target path roughly at measurement point 4, and ended at measurement point 17. It also follows that the TCP traffic backed off appropriately to let the arriving video flow have its fair share of the bandwidth. Moreover, the figure indicates that the more aggressive startup, which was a result of an increased *init_cwnd*, had no significant impact on the performance of the competing traffic.

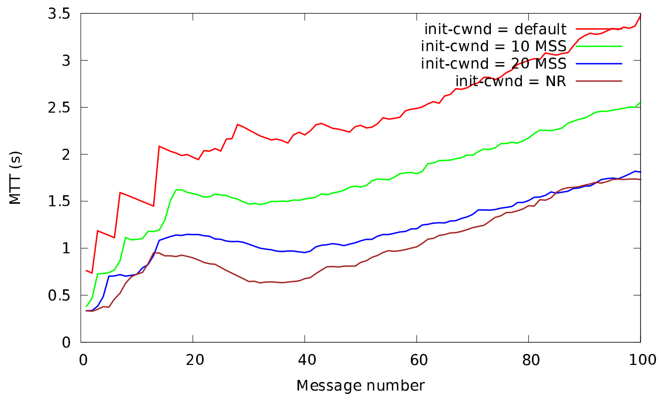

Fig. 5. MTT for one video flow. Handover of two video flows to a low capacity network with one competing flow.

Next, let us consider the experiment with two mSCTP flows being simultaneously handed over to a target path with one competing TCP flow. In this scenario, the available fair share of the bandwidth should theoretically be enough to satisfy the mSCTP flows, but, as seen in Figure 5[3], this was not the case. Instead, the MTTs increased linearly. The reason to this unexpected result was that the TCP flow did not back off appropriately, something which is seen in Figure 6.

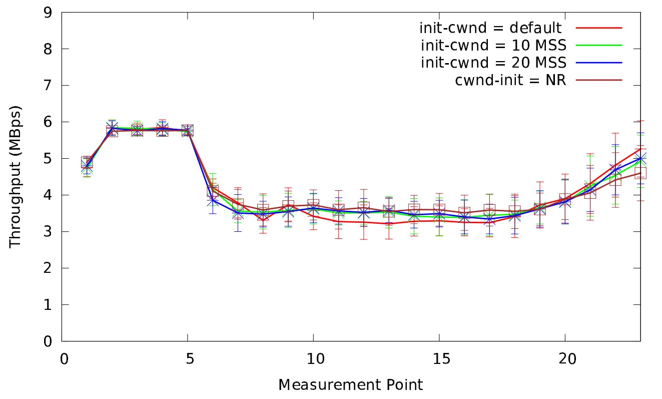

Fig. 6. Throughput for competing traffic. Handover of two video flows to a low capacity network with one competing flow.

Although the TCP flow backed off, it did not back off to more than 3.5 Mbps, something which hindered the mSCTP flows to obtain their fair shares. Still, it could be observed that at least initially, an increased *init_cwnd* resulted in a significant decrease of the experienced MTTs.

[3]For visibility purpose the confidence intervals for this experiment has been omitted. The results for the second video flow looks similar to the results for the shown flow. Thus, the second flow is not shown.

B. Handover to a high capacity network

A scenario where one video flow was handed over to a *hc* network where two competing flows were present is found in Figure 7.

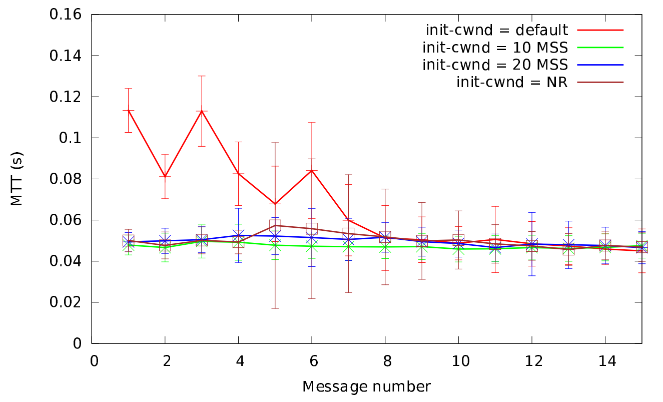

Fig. 7. MTT for one video flow. Handover to a high capacity network with two competing flows.

It is seen that a significant handover delay appeared in those cases the *init_cwnd* was set to the default value of about 3 MSS, but that this delay decreased considerably already as the *init_cwnd* was increased to 10 segments. The reason to this was primarily due to the RTT. An RTT of 40 ms made the network respond to the sender about successful transmission after the generation of about 2-3 messages, or about 8-12 MSS, i.e., the size of in the *init_cwnd*. Secondly, the bandwidth required by the video flows in the scenario was only a small fraction of the fair share of the capacity on the target path.

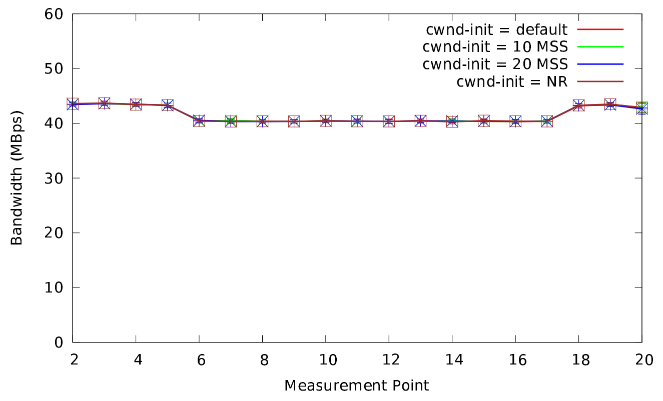

Fig. 8. Throughput for competing traffic. Handover of two video flows to a high capacity network with two competing flows.

Approximately the same results were obtained when handing over two video sessions to a target path where two TCP flows were present. Also for this traffic scenario, we monitored the competing traffic. Figure 8 shows the total capacity used by the two competing TCP flows as the mSCTP based flows were handed over. In the same way as in the *lc* scenario, it is evident that the impact on competing traffic was not affected by an increased *init_cwnd*.

Similar results were obtained in the experiments where five mSCTP flows were handed over to an *hc* target path with two competing TCP flows; a startup delay was observed in those cases the default *init_cwnd* was used, but this delay disappeared when the *init_cwnd* was increased to 10 segments. The result for one of the mSCTP flows in the scenario with 5 mSCTP flows being handed over to a target path with 2 competing flows present are seen in Figure 9[4].

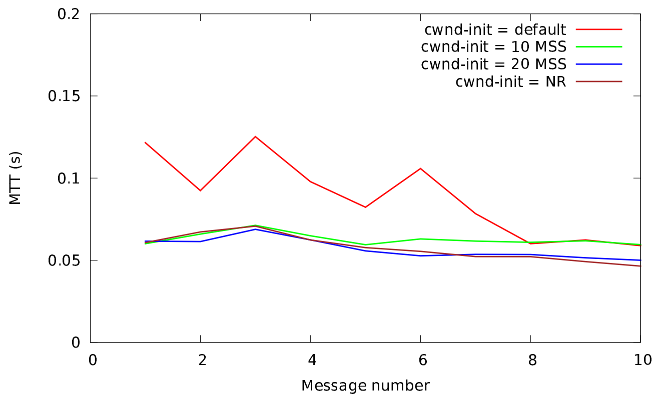

Fig. 9. MTT for one video flow. Handover of five video flows to a high capacity network with two competing flows.

Again, it is seen that by increasing the *init_cwnd* from the default size up to 10 segments, the startup delay after handover was close to zero. When 10 competing flows were sent on the handover target path, the number of lost messages during startup on the target path made the increased *init_cwnd* have no impact on the MTT. When analyzing the quite aggressive competing traffic in the scenario above, we saw no impact on the bandwidth available for the competing traffic from increasing the *init_cwnd*.

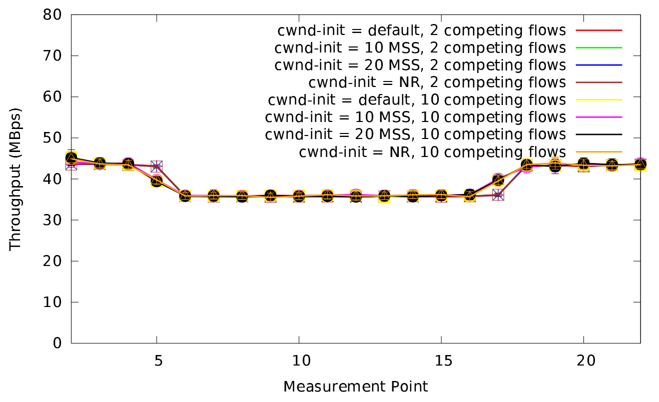

Fig. 10. Throughput for competing traffic. Handover of five video flows to a high capacity network with two vs. 10 competing flows.

Figure 10 shows the capacity used by the competing traffic in cases there were 2 and 10 competing flows present. From the figure we can see no negative impact from increasing the *init_cwnd* in any of the scenarios.

[4]Figure 9 shows the results for mSCTP flow number one out of the five. The results for the other flows look similar to the one shown in the figure.

V. Related Work

The congestion control, introduced in TCP to eliminate the risk of congestion collapses in the network, has over time changed to focus on the challenge of efficient usage of available resources in different types of networks. Several incarnations of the congestion control have as of today been proposed, implemented, deployed in different operating systems. A comprehensive survey of these different congestion control mechanisms, developed for TCP, but also applicable to SCTP, is found in [19].

Dukkipati et al. [20], [21] have recently put forward arguments for an increased *init_cwnd* in TCP. In [21], they study the effects of using an increased *init_cwnd* for Web transactions against geographically spread out data centers. In their study, they suggested that an increased *init_cwnd* could result in significantly decreased latencies for this type of transactions. Our work complements and extends their work by considering the effects on real-time traffic to improve vertical handover for mSCTP.

Previous work on mitigating the effects of slow start during a vertical handover in mSCTP includes SCTP EFC [22] and SHOP [11]. SCTP EFC is a congestion control scheme for mSCTP to be used during handover. The basic idea behind this approach is to store the congestion control parameters of the current primary path when the data flow experiences retransmission timeouts or fast retransmits, i.e., events that typically precede a handover. Later, when a handover takes place, provided the stored parameters has not become obsolete, mSCTP starts out on the handover target path with the stored congestion control parameters. Since mSCTP with EFC starts out on the handover target path with the congestion window size it had on the source path before the handover, it assumes that the network conditions are the same on this path, something that we consider a fairly dangerous assumption.

In SHOP [11], a packet-pair scheme is used to estimate the available bandwidth on the handover target path. On the basis of this estimate, SHOP configures mSCTP's *init_cwnd* and slow-start threshold appropriately. In comparison to SHOP, our proposal with a fixed, increased *init_cwnd* might seem simple and rigid. However, it should be noted that several works have highlighted several problems inherent with measuring available bandwidth [23], [24] – not least by using a packet-pair scheme. Moreover, the idea with an increased *init_cwnd*, is not to increase it up to the available bandwidth, but to set it to a bandwidth that the majority of networks are able to accommodate.

Several other studies have been done on using mSCTP for vertical handover. For example, Koh et al. suggested some tuning guidelines to improve the mSCTP handover performance [25], and demonstrated how it would be possible to integrate mSCTP with MIP [9]. Other works include Cellular SCTP (cSCTP) [26] and SIGMA [27]. cSCTP builds upon mSCTP but differs from it, in that during a handover packets are duplicated, and transmitted on both the handover source and target paths. Similar to cSCTP, SIGMA uses both the source and target paths during a handover. The SIGMA architecture has in a later work called ECHO [28] been improved to enable QoS-aware handovers.

VI. Conclusion and future work

In this paper, we have studied the impact of an increased initial congestion window for real-time traffic after a vertical handover to a target path where competing TCP traffic is present. The study has been conducted with network parameter settings intended to be representative of wireless 3G and 4G cellular networks.

We have seen that the flow or flows handed over to the target path may significantly benefit from an increased initial congestion window in terms of lower message transfer times. Since the traffic present on the target path does not always back off appropriately, the requirement is that the fair share of the capacity on the target path is far above the capacity required by the sending application.

Additionally, the results show no negative impact on the competing TCP flows by an increased initial congestion window. This fact is important, since a more aggressive startup mechanism should not penalize other traffic.

The work in this area will proceed by integrating the option of a larger initial congestion window in an Android platform of ours, to be able to verify the impact of this altered startup mechanism in real vertical handover scenarios.

Acknowledgment

The work has been supported by grants from VINNOVA, the Swedish Governmental Agency for Innovation Systems.

References

[1] R. Stewart, "Stream Control Transmission Protocol," RFC 4960, Sep. 2007.

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

[3] J. Eklund, K.-J. Grinnemo, A. Brunstrom, G. Cheimonidis, and Y. Ismailov, "Impact of Slow Start on SCTP Handover Performance," in *Proc. of 20th International Conference on Computer Communications and Networks (ICCCN)*, Maui, HI, USA, Aug. 2011, pp. 1–7.

[4] J. Eklund, K.-J. Grinnemo, and A. Brunstrom, "On the Use of an Increased Initial Congestion Window to Improve mSCTP Handover Performance," in *Proc. of 26th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, Fukuoka, Japan, Mar. 2012, pp. 1101–1106.

[5] T. Dreibholz, E. Rathgeb, I. Ruengeler, R. Seggelmann, M. Tuexen, and R. Stewart, "Stream control transmission protocol: Past, current, and future standardization activities," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 82–88, Apr. 2011.

[6] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad, "Stream control transmission protocol (SCTP) partially reliable extension," RFC 3758, May 2004.

[7] M. Tuexen, R. Stewart, P. Lei, and E. Rescorla, "Authenticated chunks for the stream control transmission protocol (SCTP)," RFC 4895, Aug. 2007.

[8] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanov, "TCP selective acknowledgement options," RFC 2018, Oct. 1996.

[9] S. J. Koh, H. Y. Jung, and J. H. Min, "Transport layer internet mobility based on mSCTP," in *IEEE 6th International Conference on Advanced Communication Technology (ICACT)*, Korea, sep 2004, pp. 329–333.

[10] Y. Kim and S. Lee, "mSCTP-based handover scheme for vehicular networks," *IEEE Communications Letters*, vol. 15, no. 8, pp. 828–830, Aug. 2011.

[11] K. Zheng, M. Liu, Z.-C. Li, and G. Xu, "SHOP: An integrated scheme for SCTP handover optimization in multihomed environments," in *IEEE Global Telecommunications Conference (GLOBECOM)*, New Orleans, Lousiana, USA, Dec. 2011, pp. 1–5.

[12] P. Soderman, K.-J. Grinnemo, Cheimonidis, Y. Ismailov, and A. Brunstrom, "An SCTP-based Mobility Management Framework for Smartphones and Tablets," in *Proc. of 26th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, Fukuoka, Japan, Mar. 2012, pp. 1107–1112.

[13] J. Garcia, E. Conchon, T. Perennou, and A. Brunstrom, "KauNet: Improving Reproducibility for Wireless and Mobile Research," in *Proc. of 1st international workshop on system evaluation for mobile platforms(MobiEval07)*, San Juan, Puerto Rico, Jun. 2007, pp. 21–26.

[14] "Dummynet homepage," Jun 2012. [Online]. Available: info.iet.unipi.it/luigi/dummynet/

[15] 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.

[16] 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.

[17] "Iperf homepage," Jun 2012. [Online]. Available: http://sourceforge.net/projects/iperf/

[18] A. Vishwanath, V. Sivaraman, and M. Thottan, "Perspectives on router buffer sizing: recent results and open problems," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 2, pp. 34–39, Mar. 2009.

[19] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-host congestion control for tcp," *Communications Surveys Tutorials, IEEE*, vol. 12, no. 3, pp. 304–342, 2010.

[20] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, "Increasing TCP's Initial Window," IETF Internet draft, work in progress, Oct. 2011.

[21] N. Dukkipati, T. Refice, Y. Cheng, J. J. Chu, T. Herbert, A. Agarwal, A. A. Jain, and S. Natalia, "An argument for increasing TCP's initial congestion window," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 3, pp. 26–33, 2010.

[22] K. Lee, S. Nam, and B. Mun, "SCTP efficient flow control during handover," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, Las Vegas, Nevada, USA, Apr. 2006, pp. 69–73.

[23] C. Dovriolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, Anchorage, Alaska, USA, Apr. 2001, pp. 905–914.

[24] K. Lai and M. Baker, "Nettimer: A tool for measuring bottleneck link bandwidth," in *Proc. USENIX Symposium on Internet Technologies and Systems*, Boston, Massachusetts, USA, Mar. 2001, pp. 123–134.

[25] S. J. Koh, M. J. Chang, and M. Lee, "mSCTP for soft handover in transport layer," *IEEE Communications Letters*, vol. 8, no. 3, pp. 189–191, Mar. 2004.

[26] I. Aydin, W. Seok, and C.-C. Shen, "Cellular SCTP: a transport-layer approach to Internet mobility," in *The 12th International Conference on Computer Communications and Networks (ICCCN)*, Dallas, Texas, USA, Oct. 2003, pp. 285–290.

[27] S. Fu, L. Ma, M. Atiquzzaman, and Y.-J. Lee, "Architecture and Performance of SIGMA: A Seamless Mobility Architecture for Data Networks," in *IEEE International Conference on Communications (ICC)*, Seoul, Korea, May 2005, pp. 3249–3253.

[28] J. Fitzpatrick, S. Murphy, M. Atiquzzaman, and J. Murphy, "ECHO: A quality of service based endpoint centric handover scheme for VoIP," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*. Las Vegas, Nevada, USA: IEEE, Mar. 2008, pp. 2777–2782.