On the Impact of Congestion Control for Concurrent Multipath Transfer on the Transport Layer

Thomas Dreibholz, Martin Becke, Hakim Adhari, Erwin P. Rathgeb
University of Duisburg-Essen, Institute for Experimental Mathematics
Ellernstraße 29, 45326 Essen, Germany
{dreibh,martin.becke,hakim.adhari,rathgeb}@iem.uni-due.de

Abstract— **Due to the resilience requirements of a steadily growing number of critical Internet services (like emergency call handling or e-commerce transactions), the deployment of multi-homed network sites becomes more and more common. Having multiple Internet access paths, it seems straightforward to utilise them *simultaneously* in order to improve payload throughput by so-called Concurrent Multipath Transfer (CMT). Currently, CMT extensions for the two important Internet Transport Layer protocols – Multipath-TCP (MPTCP) for TCP and CMT-SCTP for SCTP – are in the focus of IETF standardisation. A challenge – which is currently very actively discussed in the IETF context – is congestion control for these CMT protocols.**

Based on the idea of Resource Pooling (RP), two approaches are currently discussed in the IETF: our own approach CMT/RP for CMT-SCTP as well as the MPTCP congestion control for MPTCP. Both approaches only have been roughly tested yet – mostly in similar path setups, i.e. paths having almost the same QoS characteristics, using "their" protocol. Therefore, the goal of this paper is to compare the existing approaches. Particularly, we also analyse more challenging scenarios containing dissimilar paths, i.e. paths having differing characteristics. Our goal is to provide insight into the different approaches, to support the IETF standardisation activities on CMT. [1][2][3]

Keywords: Concurrent Multipath Transfer, Congestion Control, Resource Sharing, Dissimilar Paths, Performance Analysis

I. Introduction

Two of the main requirements on the Internet are robustness and stability. One basic mechanism to achieve these goals is the application of Congestion Control (CC) on the Transport Layer. As stated by [1], data transport should utilise the network as closely as possible on the efficiency border on the one hand, and in a fair matter for all customers on the other.

Currently deployed end-to-end Transport Layer protocols like the Transmission Control Protocol (TCP) [2] and the Stream Control Transmission Protocol (SCTP) [3] provide CC as core functionality. However, their CC mechanisms are specialized for *single* path data transfer. This stands in contrast to the trend that more and more endpoints are connected by *multiple* paths, which is denoted as *multi-homing*. In the context of Transport Layer protocols, this could be caused by different access technologies or different addressing schemes, e.g. ADSL and UMTS as access technologies as well as IPv4 and IPv6 as addressing schemes.

Having multi-homed endpoints, it is a straightforward desire to not only improve network resilience but also to utilise *all* paths for increased application payload throughput. This approach is denoted as Concurrent Multipath Transfer (CMT). On the Transport Layer, CMT is currently highly discussed in the IETF as extensions for TCP (so-called Multipath TCP – MPTCP [4]) and SCTP (so-called CMT-SCTP [5]). Both approaches face the same challenge: the need for a CC strategy with multipath data transfer in mind. Two different strategies – our own CMT/RP CC for CMT-SCTP as well as MPTCP CC for MPTCP – are currently discussed within the IETF. However, a performance comparison has been missing. Furthermore, the behaviour of the CCs in setups containing so-called dissimilar paths, i.e. paths having different QoS characteristics and queuing behaviours, has been unclear. Such scenarios are challenging for CMT – as shown in [6]–[8] – but very common in realistic Internet setups.

Therefore, the goal of this paper is to first provide an overview of the different CCs and their underlying ideas. After that, we analyse and compare their performance – in similar as well as dissimilar path setups – in order to support the IETF discussion and foster the standardisation process of both, CMT-SCTP as well as MPTCP.

II. Congestion Control Basics

While telecommunications networks like ATM provide sophisticated mechanisms to guarantee certain QoS properties, the Internet only provides a Best Effort service. Therefore, it is the duty of the Transport Layer protocol to perform a CC strategy which avoids overloading the network – while still achieving an acceptable payload throughput for its users. In the following, we briefly introduce the basics of CC in the Internet which are relevant for this paper. A more detailed introduction to congestion control can be found e.g. in [1], [9].

The two reliable, unicast Transport Layer protocols standardised by the IETF – TCP [2] as well as SCTP [3]

[1]Parts of this work have been funded by the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG).

[2]Parts of this work have been funded within the scope of the G-Lab project by the German Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung – BMBF).

[3]The authors would like to thank Michael Tüxen for his support and the anonymous reviewers for their helpful comments.

– both apply window-based CC. That is, the sender endpoint maintains a *congestion window* variable c, which denotes the maximum amount of outstanding data (i.e. data being sent but not yet acknowledged by the receiver instance). c is adapted using so-called Additive Increase, Multiplicative Decrease (AIMD) behaviour, i.e. c may grow once old data is acknowledged and c is decreased on congestion (which is determined by packet loss[4]).

The so-called *slow-start threshold* variable s controls whether the congestion window c may grow on every acknowledgement ($c < s$ – so-called Slow Start phase, see also [9, Section 3.1]; this leads to an exponential growth of c) or each time the data amount of a full congestion window is acknowledged ($c \geq s$ – so-called Congestion Avoidance phase, see also [9, Section 3.1]; this leads to a linear growth of c). The recommended way (see [9, Section 3.1]) to decide whether enough data has been acknowledged to advance c during Congestion Avoidance is to add another variable p ("partially acknowledged"), which counts the previous acknowledgements. A window advance is possible for $p \geq c$; on advance, p is reset. Note, that congestion window growth – regardless of the phase – is only allowed on *new* advances of the *cumulative* acknowledgement, in order to avoid jumps of c after filling a gap in the sequence of acknowledgements.

On detection of a packet loss, state-of-the-art implementations once perform so-called Fast Retransmission (Fast RTX; see also [9, Section 3.2]) by halving s, setting $c = s$ (i.e. multiplicative decrease) and retransmitting the lost segment. Usually, Fast RTX is combined with so-called Fast Recovery (see also [9, Section 3.2]), which means to forbid growing c until the retransmitted segment has been acknowledged. Further losses of the same segment are assumed as a sign of severe congestion. Therefore, its retransmission is triggered by expiration of the segment's Retransmission Timer. It is therefore denoted as Timer-Based Retransmission (Timer-Based RTX); at that time, s is halved but c minimised.

In most state-of-the art CC implementations, the congestion window c and slow start threshold s are stored in bytes. Using packets instead of bytes, which is still widespread due to older implementations, leads to the problem that hosts using different Maximum Transmission Units (MTU; e.g. an ADSL-connected user with an MTU of 1,492 bytes due to PPPoE and a Gigabit Ethernet user with an MTU of 9,000 bytes) may – for the same setting of c – send different byte amounts of data. The MTU – given by the underlying Data Link Layer – implies an upper segment payload limit on the Transport Layer. This limit is denoted as Maximum Segment Size (MSS). It is e.g. 1,460 bytes for TCP or 1,452 bytes for SCTP using IPv4 over a 100BaseTX Ethernet interface with an MTU of 1,500 bytes.

The setting of c limits the bandwidth b_{Data} of the data transmission for a given round trip time (RTT):

$$b_{\text{Data}} \leq \frac{c}{RTT} \tag{1}$$

This is an implication of the bandwidth-delay product.

[4]We neglect advanced mechanisms like Explicit Congestion Notification (ECN) [10] here for simplicity.

III. Congestion Control Approaches for Multipath Transport Protocols

In the following subsections, we introduce the multipath CC strategies which are currently discussed in the IETF for CMT-SCTP and MPTCP, with focus on application for CMT-SCTP (which is used for our evaluation).

A. Plain CMT Congestion Control

CMT-SCTP [5] is the straightforward approach of adding CMT support to SCTP. Since SCTP [3], [11] itself already incorporates multi-homing support, there is no need to change the CC behaviour for CMT-SCTP. All paths are handled separately, i.e. for each path P there are independent congestion window c_P, slow-start threshold s_P and partial acknowledgements p_P variables. All variables are counted in bytes.

On α newly acknowledged bytes on path P in a fully-utilized congestion window, c_P is adapted as follows:

$$c_P = c_P + \begin{cases} \min\{\alpha, \text{MSS}_P\} & (c_P < s_P) \\ \text{MSS}_P & (c_P \geq s_P \wedge p_P \geq c_P) \end{cases}$$

SCTP applies – like state-of-the-art TCP implementations – so-called Appropriate Byte Counting [12], i.e. c_P is only advanced by the minimum of the acknowledged bytes α and MSS_P the MSS of P in Slow Start (i.e. $c_P < s_P$) as well as only by MSS_P in Congestion Avoidance (i.e. $c_P \geq s_P$).

Note, that CMT-SCTP [5] has to keep multiple paths in mind. That is, in contrast to a single-homed transmission, there is no global cumulative acknowledgement. CMT-SCTP therefore introduces per-path so-called "pseudo cumulative acknowledgements". An illustrative example is provided in [13].

On retransmission on path P, s_P and c_P are adapted as follows, similar to state-of-the-art TCP implementations:

$$\begin{aligned} s_P &= \max\{c_P - \frac{1}{2} * c_P, 4 * \text{MSS}_P\} \\ c_P &= \begin{cases} s_P & \text{(Fast RTX)} \\ \text{MSS}_P & \text{(Timer-Based RTX)} \end{cases} \end{aligned}$$

In result, plain CMT CC behaves like a TCP flow using a state-of-the-art implementation – on *each* of its paths. Clearly, when n CMT paths share a single bottleneck link with a non-CMT flow (e.g. a standard SCTP or TCP flow), the CMT flow occupies n times the bandwidth of the non-CMT flow. Obviously, this behaviour is unfair.

B. CMT/RP Congestion Control

An approach to overcome the unfairness problem of CMT is Resource Pooling (RP), which denotes "making a collection of resources behave like a single pooled resource" [14]. Adapted to CMT, the set of all paths should behave like a single large one. RP should achieve the following goals [14]:

1) A CMT flow should get at least as much bandwidth as a single-homed flow via the best path.
2) A CMT flow should not take more capacity on a shared bottleneck path than a single-homed flow via the same bottleneck.

3) A CMT flow should balance congestion on all of its paths.

Based on the idea of RP, we have defined two CC variants, which we introduce in the following.

1) Version 1 – CMT/RPv1: CMT/RP [15] version 1 – shortly CMT/RPv1 – is our initial approach to apply RP to CMT-SCTP. It is now also provided by FreeBSD kernel SCTP [11]. CMT/RPv1 assumes similar paths, i.e. paths having almost the same characteristics (bandwidth, delay, error rate). The slow start threshold is used as a useful metric for the capacity of a path. For each path P, the *slow-start threshold ratio* $\hat{s}_P$ is defined as:

$$\hat{s}_P = \frac{s_P}{\sum_i s_i} \tag{2}$$

On α acknowledged bytes on path P in a fully-utilized congestion window, CMT/RPv1 adapts c_P as follows:

$$c_P = c_P + \begin{cases} \lceil \hat{s}_P * \min\{\alpha, \text{MSS}_P\} \rceil & (c_P < s_P) \\ \lceil \hat{s}_P * \text{MSS}_P \rceil & (c_P \geq s_P \wedge p_P \geq c_P) \end{cases}$$

That is, c_P is increased according to the slow-start threshold ratio $\hat{s}_P$ of P.

On retransmission on path P, CMT/RPv1 adapts s_P and c_P as follows:

$$\begin{aligned} s_P &= \max\{\lceil c_P - \frac{1}{2} * \sum_i c_i \rceil, \lceil \hat{s} * 4 * \text{MSS}_P \rceil, \text{MSS}_P\} \\ c_P &= \begin{cases} s_P & \text{(Fast RTX)} \\ \text{MSS}_P & \text{(Timer-Based RTX)} \end{cases} \end{aligned}$$

That is, CMT/RPv1 reduces c_P by half of the flow's total congestion window $\sum_i c_i$, with a lower bound of MSS_P.

2) Version 2 – CMT/RPv2: CMT/RPv1 assumes comparable slow start thresholds s_i in the computation of the ratio $\hat{s}_P$ (see equation 2) of a path P. However, this may be difficult in case of dissimilar paths [7], [8]. Our advanced approach CMT/RPv2 – which is currently under discussion within the IETF – overcomes the limitations of CMT/RPv1 by considering path bandwidths.

In order to increase the congestion window c_P on α acknowledged bytes on path P in a fully-utilized congestion window, the *increase factor* $\hat{i}_P$ is calculated:

$$\hat{i}_P = \frac{\frac{c_P}{\text{RTT}_P}}{\sum_i \frac{c_i}{\text{RTT}_i}}$$

It represents the current bandwidth share of P on the total bandwidth of the flow (based on equation 1). Using $\hat{i}$, c_P is adapted as follows:

$$c_P = c_P + \begin{cases} \lceil \hat{i} * \min\{\alpha, \text{MSS}_P\} \rceil & (c_P < s_P) \\ \lceil \hat{i} * \text{MSS}_P \rceil & (c_P \geq s_P \wedge p_P \geq c_P) \end{cases}$$

For reducing c_P on a packet loss on path P, the *decrease factor* $\hat{d}_P$ is applied:

$$\hat{d}_P = \max\{\frac{1}{2}, \frac{1}{2} * \frac{\sum_i \frac{c_i}{\text{RTT}_i}}{\frac{c_P}{\text{RTT}_P}}\}$$

$\hat{d}$ represents the factor by which the bandwidth of P should be reduced in order to halve the total bandwidth of the flow. For example, two paths P_1 (10 Mbit/s) and P_2 (2 Mbit/s) lead to a total bandwidth of 12 Mbit/s. A loss on P_1 leads to $\hat{d}_1 = \frac{1}{2} * \frac{12}{10} = 0.6$; a loss on P_2 to $\hat{d}_2 = \frac{1}{2} * \frac{12}{2} = 3.0$.

Using $\hat{d}_P$, s_P and c_P are adapted as follows:

$$\begin{aligned} s_P &= \max\{c_P - \lceil \hat{d}_P * c_P \rceil, 1 * \text{MSS}_P\} \\ c_P &= \begin{cases} s_P & \text{(Fast RTX)} \\ \text{MSS}_P & \text{(Timer-Based RTX)} \end{cases} \end{aligned}$$

That is, the new setting of c_P tries to halve the total bandwidth, with a lower bound of MSS_P.

C. MPTCP-Like Congestion Control

Like CMT/RP, the CC of MPTCP [16], [17] also applies RP [14] to ensure fairness. However, the CC behaviour is different: while CMT/RP tries to halve the *total* congestion window/total bandwidth on a packet loss on path P, MPTCP CC behaves exactly like standard TCP or SCTP by only halving the *path* congestion window c_P (see Subsection III-A for the formula). Since this behaviour alone would cause unfairness, the growth behaviour has to be adapted. MPTCP uses the idea of controlling engineering: growth and decrease of c_P have to be brought into equilibrium by adapting the congestion window growth by a per-flow *aggressiveness factor* $\hat{a}$.

Since the MPTCP CC introduced in [16] is based on packets instead of bytes and we furthermore use SCTP instead of TCP for our evaluation (see Section IV), we had to port the MPTCP CC accordingly. That is, on α acknowledged bytes on path P in a fully-utilized congestion window, our MPTCP-like CC adapts c_P as follows:

$$c_P = c_P + \begin{cases} \min\{\lceil \hat{a} * \min\{\alpha, \text{MSS}_P\} \rceil, \min\{\alpha, \text{MSS}_P\}\} \\ \qquad (c_P < s_P) \\ \min\{\lceil \hat{a} * \frac{c_P}{\text{MSS}_P} * \text{MSS}_P \rceil, \text{MSS}_P\} \\ \qquad (c_P \geq s_P \wedge p_P \geq c_P) \end{cases}$$

$\hat{a}$ denotes the per-flow *aggressiveness factor*, defined as:

$$\hat{a} = \frac{\max_i\{\frac{c_i/\text{MSS}_i}{(\text{RTT}_i)^2}\}}{(\sum_i \frac{c_i/\text{MSS}_i}{\text{RTT}_i})^2}$$

Both formulae are based on [16]. However, we have cancelled out the term of the total congestion window tot_cwnd, since $\frac{\text{tot\_cwnd}}{\text{tot\_cwnd}} = 1$. Note further, that due to application of partial acknowledgements (see Section II) – which is standard behaviour for SCTP [3] – c_P is only increased each time a full congestion window has been acknowledged (i.e. $p_P \geq c_P$). Therefore, $\frac{c_P}{\text{MSS}_P}$ increment steps as in [16] are skipped and the increase is multiplied by $\frac{c_P}{\text{MSS}_P}$ therefore.

IV. Reliable Multipath Transport

TCP [2] is a byte-stream-oriented, single-homed Transport Layer protocol, while SCTP [3] is message-oriented with multi-homing support. Despite these fundamental differences, both protocols provide a unicast, reliable and congestion-controlled service and share many basic mechanisms. Particularly, both use the basically same, window-based CC mechanism. Neither TCP nor SCTP provide CMT (i.e. *simultaneous* usage of multiple paths)

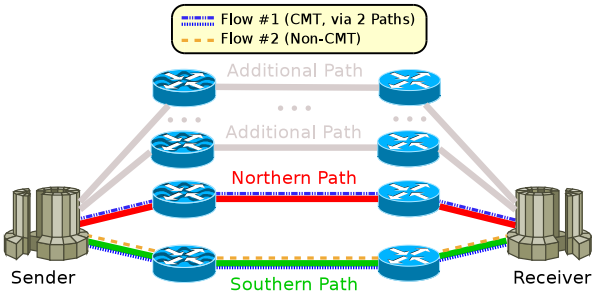

Figure 1. The Simulation Setup

out of the box. For TCP, the MPTCP extension adds support for multi-homing and CMT. For SCTP, that already supports multi-homing, the CMT-SCTP extension [5], [18] provides the CMT capability.

Since MPTCP and CMT-SCTP provide – despite of their very different realisations – a very similar service, we concentrate on CMT-SCTP for our CC evaluation. Nevertheless, all results can be transferred to other CMT protocols – particularly to MPTCP – as well.

An SCTP connection is denoted as *association*. Each Network Layer address of the peer endpoint defines a unidirectional *path*. User messages are segmented into so-called *DATA chunks* for transport and reassembled at the peer endpoint. The so-called *smallest Path MTU* (sPMTU) of all paths is applied for data segmentation [3], since the MTUs may differ on distinct paths. If there are DATA chunks smaller than the sPMTU, SCTP tries to bundle multiple DATA chunks into a single SCTP packet, in order to keep the transport overhead as small as possible – and therefore to improve efficiency.

Each DATA chunk is identified by a unique Transmission Sequence Number (TSN). The TSNs are used by the peer endpoint to acknowledge the reception of DATA chunks by Selective Acknowledgement (SACK) chunks. These SACK chunks provide the possibility to cumulatively acknowledge data up to a given TSN – as well as TSNs further ahead in so-called gap acknowledgements. These gap acknowledgements identify gaps in the transmission sequence, which have to be selectively filled by Fast or Timer-Based RTX.

In the context of CMT-SCTP, it is particularly important to note that gap acknowledgements in SACK chunks are renegable (i.e. a receiver may decide to "un-acknowledge" them at any time). So-called Non-Renegable SACK (NR-SACK) chunks introduced as protocol extension by [19] allow a receiver to declare gap acknowledgements as non-renegable. NR-SACK significantly improves CMT transport performance – as shown in [6]–[8] – by allowing a sender to remove gap-acknowledged chunks from its send buffer.

Futhermore, in combination with so-called "Buffer Splitting based on Outstanding Bytes" [7] – which describes techniques to handle send and receive buffers in order to avoid blocking issues – NR-SACK is required for effective CMT transport over dissimilar paths [6], [7].

V. Simulation Setup

For our performance evaluation, we have used the OMNeT++-based INET framework with our CMT-SCTP simulation model [13]. The SimProcTC [20] tool-chain has been used for parameterisation and result processing. Figure 1 presents the simulation scenario. For the simulations performed, at least two paths have been used (the Southern and the Northern path); if needed, additional paths can be added to the association. For all paths available, the following configuration parameters have been used:

- The sender has been saturated (i.e. it has tried to transmit as much data as possible); the message size has been between 500 bytes and 1,452 bytes (using uniform distribution) at an MTU of 1,500 bytes. All messages have used unordered delivery.
- The CMT flow may have used all paths, but the non-CMT flow only the southern one.
- The send buffer has been set to 1,000,000 bytes; the receive buffer has been set to 500,000 bytes. Buffer Splitting according to [7] and NR-SACKs [19] have been used.
- On the routers, FIFO queues of 100 packets have been configured. The bandwidth of each independent path has been 10 Mbit/s; the delay has been 1 ms.

The simulation runtime has been 300 s after a transient phase of 20 s. Each run has been repeated 100 times in order to ensure a sufficient statistical accuracy. The results plots show the average values and their corresponding 95% confidence intervals.

VI. Evaluation

In our evaluation, we first evaluate the basic CC behaviour in the two or more similar path setups. After that, we focus on dissimilar path setups with only two paths.

A. Similar Paths

Figure 2 presents the achieved application payload throughput for the concurrency of a CMT flow #1 (F=1; solid lines) and a non-CMT flow #2 (F=2; dotted lines) using different CCs Γ for varying the number of paths in the two extreme setups: (1) all disjoint paths (left-hand side) as well as (2) all paths sharing the same bottleneck (right-hand side).

For the all disjoint paths scenario (see left-hand plot), the CMT flow (i.e. F=1) only competes with the non-CMT flow (i.e. F=2) on the first path. All other paths may be used for the CMT flow exclusively. Using plain CMT CC (i.e. Γ=cmt), the result is as expected: the bandwidth on the path shared by both flows is halved. For using one of the three RP-based CCs (i.e. Γ=cmtrpv1 for CMT/RPv1, Γ=cmtrpv2 for CMT/RPv2 or Γ=like-mptcp for MPTCP-like), the CMT flow is less aggressive on the shared path and hands over bandwidth to the non-CMT flow. The difference between the two CMT/RP variants and MPTCP-like CC is that the latter is slightly more aggressive in the range of 2 to 4 paths: using CMT/RP, the non-CMT flow bandwidth is higher. We will explain this fact in more detail in Subsection VI-C.

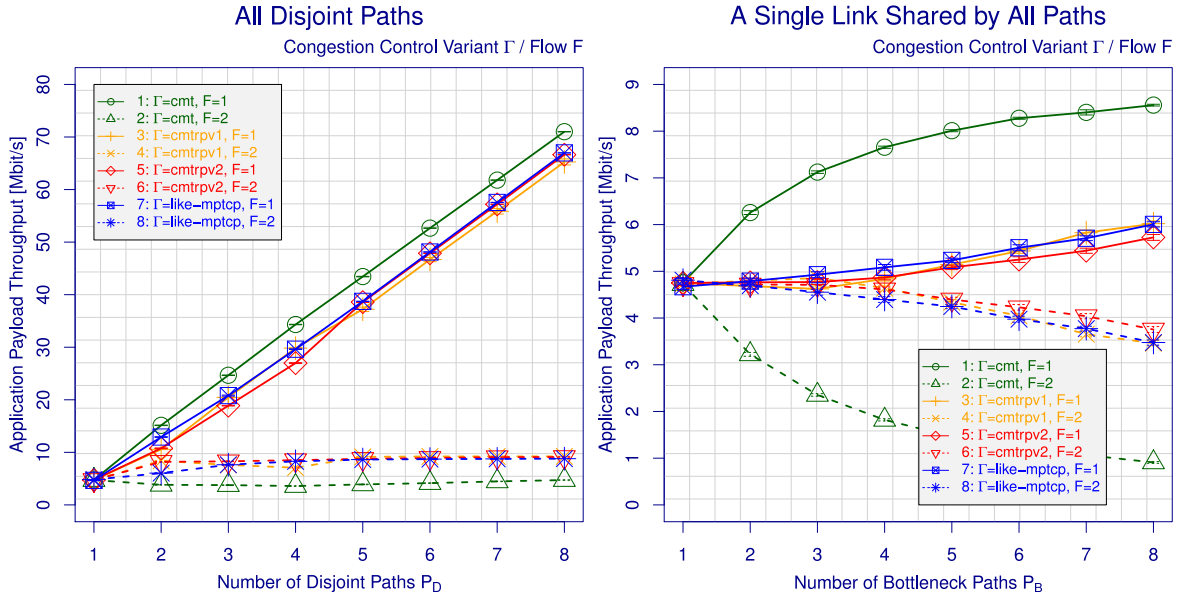

Figure 2. Application Payload Throughput for Disjoint and Bottleneck Paths Scenarios

When all paths share a single bottleneck link (see right-hand plot), the unfairness of plain CMT-CC (i.e. Γ=cmt) becomes clearly visible: at 8 paths, the CMT flow occupies around $\frac{8}{9}$ of the 10 Mbit/s, resulting in a payload throughput of only about 1 Mbit/s for the non-CMT flow. For up to 4 paths, the throughput difference between the two flows is very small when using one of the three RP-based CCs (i.e. $\Gamma \in \{\text{cmtrpv1}, \text{cmtrpv2}, \text{like-mptcp}\}$). A scenario of 2 or 4 paths is also quite realistic in real Internet scenarios, e.g. a setup connected to two ISPs, with IPv4 and IPv6 prefixes from each ISP. For a rising number of paths, there is a growing difference between the flows: in order to probe a path P, the minimum congestion window c_P is MSS_P. That is, this traffic for probing the links – although small – sums up due to the large number of paths. However, even at 8 paths, the performance of the non-CMT flow is still significantly better than for using plain CMT CC.

In summary, the three RP-based CCs significantly improve the fairness of CMT transport. MPTCP-like CC is slightly more aggressive than the CMT/RP CCs. Nevertheless, all three CCs fulfil the three goals set in Subsection III-B.

B. Dissimilar Paths

The following scenarios use the basic simulation setup illustrated in Figure 1 to examine the performance over dissimilar paths. Only the Southern and the Northern path (i.e. just two paths) have been used.

1) Varying Exclusive Northern Path Bandwidth: First, we vary the bandwidth ρ_N of the northern path (which is exclusively used by the CMT flow), while keeping the southern path bandwidth constant at ρ_S=5 Mbit/s. Figure 3 presents the resulting payload throughputs for the CMT flow #1 (i.e. F=1; solid lines) and the non-CMT flow #2 (i.e. F=2; dotted lines) for the different CCs Γ.

Obviously, the CMT CC (i.e. Γ=cmt) behaves as expected: the CMT flow takes half of the southern path bandwidth (leaving the other half for the non-CMT flow) plus the complete northern path bandwidth. Also, using CMT/RPv2 (i.e. Γ=cmtrpv2) or MPTCP-like CC (i.e. Γ=like-mptcp) results in shifting bandwidth on the shared path to the non-CMT flow. Again, MPTCP-like CC is more aggressive than CMT/RPv2 CC. That is, for all bandwidth settings of ρ_N, the CMT flow using MPTCP-like CC takes more bandwidth (and therefore leaves less for the other flow) than CMT/RPv2 CC. Particularly, the behaviour is significantly different for $\rho_N < \rho_S$; we will further examine this fact in Subsubsection VI-B.2. Nevertheless, both CCs again meet the goals set in Subsection III-B.

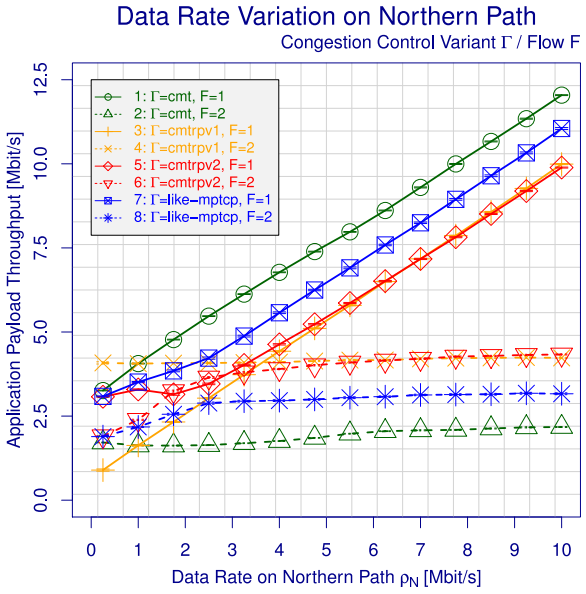

Figure 3. Data Rate Variation on Northern Path

Furthermore, dissimilar paths reveal the problem

of CMT/RPv1: for a small setting of ρ_N (here: $\rho_N \leq 4$ Mbit/s), the throughput of the CMT flow falls below the non-CMT flow. This violates the first goal set in Subsection III-B. Particularly, at ρ_N=250 Kbit/s, the achieved CMT flow payload throughput is less than 1.5 Mbit/s – in contrast to the roughly expected 2.6 Mbit/s. The reason is as follows: the 250 Kbit/s northern link – with its 100 packets FIFO queue – leads to so-called "Buffer Bloat" [21], i.e. its congestion window c_N may become large in proportion to the southern path congestion window c_S. Since there is no concurrency on the northern path, loss is only introduced by the filled FIFO queue – which requires a sufficiently large c_N. When a loss on the fast southern path occurs (mostly due to the concurrency), c_S has to be reduced to around MSS_S due to the large total congestion window $c_N + c_S$. This leads to the need of slowly growing c_S again (due to small $\hat{s}_S$, see equation 2). Until c_S has not again grown large enough to cover the RTT-delay product (see equation 1), the fast northern path remains underutilised.

2) Varying Shared Southern Path Bandwidth: In the following, we further examine the CC performance on dissimilar paths by varying the bandwidth of the shared southern path. The left-hand side of figure 4 presents the payload throughput results for varying the southern path bandwidth ρ_S, while keeping the northern path bandwidth fixed at ρ_N=5 Mbit/s.

As expected from the previous results, plain CMT CC shows the typical unfairness while CMT/RPv1 CC reveals its problems with dissimilar paths. Particularly, for $\rho_S \geq 7.5$ Mbit/s, the non-CMT flow always occupies more bandwidth than the CMT flow.

The really interesting results are for CMT/RPv2 and MPTCP-like CC. Both fulfil the goals set in Subsection III-B – but in different ways: using CMT/RPv2 CC, the *total* bandwidth is almost equally shared between both flows when possible (here: ρ_S between 7.5 Mbit/s and 15 Mbit/s), i.e. for example at ρ_S=10 Mbit/s, both flows achieve a payload throughput around 7 Mbit/s at the total paths' bandwidth of $\rho_N + \rho_S$=15 Mbit/s. In contrast, MPTCP-like CC behaves significantly more aggressive: it ensures that it does not take more bandwidth than a non-CMT flow (e.g. at ρ_S=10 Mbit/s, the CMT flow gets around 9 Mbit/s – leaving slightly more than 5 Mbit/s for the non-CMT flow), but also takes significantly more bandwidth for the CMT flow than CMT/RPv2 CC.

The observations also remain when increasing the number of non-CMT flows n, as shown for ρ_N=5 Mbit/s and ρ_S=20 Mbit/s on the right-hand side of Figure 4. Note, that only the first non-CMT flow (i.e. F=2) is shown for readability reason – the behaviour of the further non-CMT flows is analogous. Although the difference among the CCs becomes smaller, MPTCP-like CC still remains more aggressive than CMT/RPv2 CC. For example, for $n = 8$ non-CMT flows, the CMT flow occupies around 7 Mbit/s vs. around 6 Mbit/s for using CMT/RPv2 CC.

3) Summary: In summary, the CMT/RPv1 CC is not suitable for dissimilar paths. CMT/RPv2 as well as MPTCP-like CC achieve the goals set in Subsection III-B. However, MPTCP – in comparison to CMT/RPv2 – more aggressively occupies bandwidth for the CMT flow. If the administrative goal is to distribute bandwidth to flows equally – regardless of the number of paths used for transport – the CMT/RPv2 CC is more suitable.

C. Differences Among the Congestion Control Variants

In order to make the difference among the four presented CCs clear, we have finally plotted the congestion window and slow start threshold adaptation for the CMT flow during the first 75 s of association runtime for settings of northern path bandwidth ρ_N=5 Mbit/s and southern path bandwidth ρ_S=10 Mbit/s in figure 5.

Using plain CMT CC (first plot), the CC is independent for each path. That is, on the northern path, an adaptation is only triggered when the congestion window c_N is large enough to cause a drop in its FIFO queue. Due to concurrency on the southern path, the corresponding congestion window c_S is also adapted on loss due to overload caused by the non-CMT flow. Therefore, c_S is usually smaller than c_N.

The plot for CMT/RPv1 CC (second plot) visualises the Buffer Bloat issue (see Subsection VI-C): c_N of the slower northern path may become large (up to 70,000 bytes) while c_S on the faster southern path remains smaller (up to around 30,000 bytes – due to the concurrency on this path). A loss on the southern path leads to a very small growth (due to awkward slow start threshold ratio $\hat{s}_S$; see equation 2), leading to not occupying the expected bandwidth share. This fact implies the performance problem of CMT/RPv1 in dissimilar path scenarios.

CMT/RPv2 CC (third plot) overcomes this problem: after congestion window reduction on the southern path, c_S is able to recover quickly – since it cares for bandwidths instead of slow start thresholds (see Subsubsection III-B.2). In result, the expected bandwidth is utilised on the southern path.

MPTCP-like CC (fourth plot) shows a similar behaviour. However, the congestion window c_S on the southern path is in general significantly larger than for using CMT/RPv2 CC. This is caused by the more aggressive behaviour of MPTCP-like CC, resulting in the observed larger bandwidth share for the CMT flow.

VII. Conclusions

At the moment, congestion control is a very hot topic in the IETF discussion on standardisation of the CMT protocol extensions CMT-SCTP and MPTCP. In order to ensure a fair resource sharing in comparison to non-CMT protocols in the Internet, two approaches are currently discussed: our CMT/RPv2 as well as the MPTCP-like congestion control. Except for some performance tests using their "own" protocols, these approaches had not been compared or analysed in more complex setups before.

In this paper, we have provided a survey of the congestion control approaches currently discussed in the IETF. After that, we have analysed their performance in similar as well as dissimilar path setups – the latter ones are particularly important, since they are very realistic

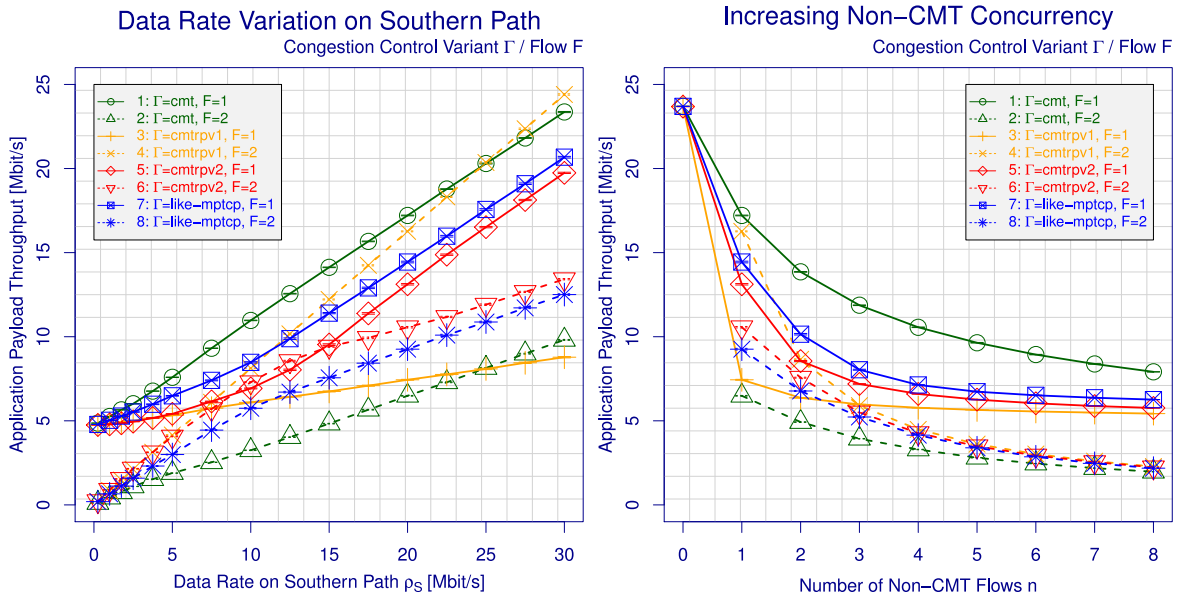

Figure 4. Data Rate Variation on Southern Path being shared with Non-CMT Flows

for Internet setups. MPTCP-like as well as CMT/RPv2 congestion control fulfil the RP fairness goals. Depending on network and application administration goals, the bandwidth share may either be optimised for higher CMT flow throughput by using MPTCP-like congestion control or equal bandwidths by using CMT/RPv2.

As part of future work, we are going to formalise the fairness criteria of multipath transport, in order to provide a more fine-granular classification. Furthermore, we are also going to contribute our results into the ongoing IETF standardization process of CMT-SCTP [18] and MPTCP.

References

[1] M. Welzl, *Network Congestion Control: Managing Internet Traffic (Wiley Series on Communications Networking & Distributed Systems)*. John Wiley & Sons, 2005, ISBN 978-0-470-02528-4.

[2] J. Postel, "Transmission Control Protocol," IETF, Standards Track RFC 793, Sept. 1981.

[3] R. Stewart, "Stream Control Transmission Protocol," IETF, Standards Track RFC 4960, Sept. 2007.

[4] A. Ford, C. Raiciu, M. Handley, S. Barré, and J. Iyengar, "Architectural Guidelines for Multipath TCP Development," IETF, Informational RFC 6182, Mar. 2011.

[5] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 951–964, Oct. 2006, ISSN 1063-6692.

[6] T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tüxen, "On the Use of Concurrent Multipath Transfer over Asymmetric Paths," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Miami, Florida/U.S.A., Dec. 2010, ISBN 978-1-4244-5637-6.

[7] H. Adhari, T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tüxen, "Evaluation of Concurrent Multipath Transfer over Dissimilar Paths," in *Proceedings of the 1st International Workshop on Protocols and Applications with Multi-Homing Support (PAMS)*, Singapore, Mar. 2011, pp. 708–714, ISBN 978-0-7695-4338-3.

[8] T. Dreibholz, R. Seggelmann, M. Tüxen, and E. P. Rathgeb, "Transmission Scheduling Optimizations for Concurrent Multipath Transfer," in *Proceedings of the 8th International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT)*, vol. 8, Lancaster, Pennsylvania/U.S.A., Nov. 2010, ISSN 2074-5168.

[9] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," IETF, Standards Track RFC 5681, Sept. 2009.

[10] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," IETF, Standards Track RFC 3168, Sept. 2001.

[11] T. Dreibholz, I. Rüngeler, R. Seggelmann, M. Tüxen, E. P. Rathgeb, 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, ISSN 0163-6804.

[12] M. Allman, "TCP Congestion Control with Appropriate Byte Counting (ABC)," IETF, RFC 3465, Feb. 2003.

[13] T. Dreibholz, M. Becke, J. Pulinthanath, and E. P. Rathgeb, "Implementation and Evaluation of Concurrent Multipath Transfer for SCTP in the INET Framework," in *Proceedings of the 3rd ACM/ICST International Workshop on OMNeT++*, Torremolinos, Málaga/Spain, Mar. 2010, ISBN 978-963-9799-87-5.

[14] D. Wischik, M. Handley, and M. B. Braun, "The Resource Pooling Principle," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 5, pp. 47–52, Oct. 2008, ISSN 0146-4833.

[15] T. Dreibholz, M. Becke, J. Pulinthanath, and E. P. Rathgeb, "Applying TCP-Friendly Congestion Control to Concurrent Multipath Transfer," in *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, Perth/Australia, Apr. 2010, pp. 312–319, ISSN 1550-445X.

[16] C. Raiciu, M. Handley, and D. Wischik, "Practical Congestion Control for Multipath Transport Protocols," University College London, London/United Kingdom, Tech. Rep., 2009.

[17] ——, "Coupled Multipath-Aware Congestion Control," IETF, Network Working Group, Internet Draft Version 02, Mar. 2011, draft-ietf-mptcp-congestion-02, work in progress.

[18] M. Becke, T. Dreibholz, J. Iyengar, P. Natarajan, and M. Tüxen, "Load Sharing for the Stream Control Transmission Protocol (SCTP)," IETF, Network Working Group, Internet Draft Version 01, Dec. 2010, draft-tuexen-tsvwg-sctp-multipath-01.txt, work in progress.

[19] P. Natarajan, N. Ekiz, E. Yilmaz, P. D. Amer, and J. Iyengar, "Non-Renegable Selective Acknowledgments (NR-SACKs) for SCTP," in *Proceedings of the 16th IEEE International Conference on Network Protocols (ICNP)*, Orlando, Florida/U.S.A., Oct. 2008, pp. 187–196, ISBN 978-1-4244-2506-8.

[20] T. Dreibholz, X. Zhou, and E. P. Rathgeb, "SimProcTC – The Design and Realization of a Powerful Tool-Chain for OMNeT++ Simulations," in *Proceedings of the 2nd ACM/ICST International Workshop on OMNeT++*, Rome/Italy, Mar. 2009, pp. 1–8, ISBN 978-963-9799-45-5.

[21] J. Gettys, "Bufferbloat – Dark Buffers in the Internet," Jan. 2011.

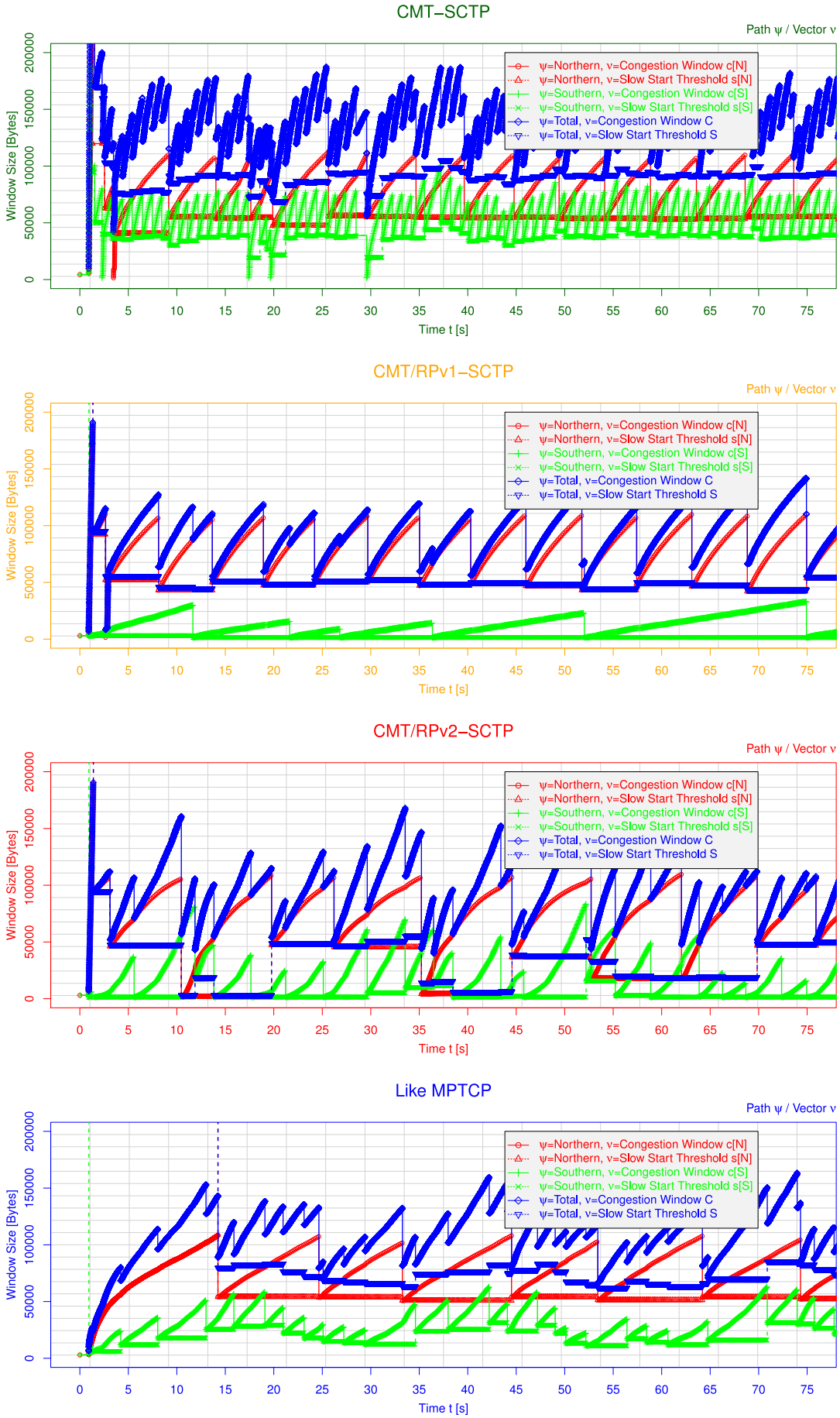

Figure 5. Congestion Window and Slow Start Threshold Adaptations of the Congestion Control Variants