RESEARCH DISCLOSURE
www.researchdisclosure.com

Certified Document

The Industry Standard Disclosure Publication Service

Method for Two-Domain Congestion Control

Disclosed anonymously

Research Disclosure database number 659017

Published in the March 2019 paper journal

Published digitally 25 January 2019 07:43 UT

Research Disclosure™ has been published monthly since 1960.
ISSN 0374-4353.

Research Disclosure, Questel Ireland Ltd
Killernan, Kilmaine, County Mayo, Ireland
Email: info@researchdisclosure.com
Phone: +353 (0) 93 33870

Method for Two-Domain Congestion Control

1 TECHNICAL AREA / KEYWORDS

Transport protocol, performance enhancement, explicit cooperation, congestion feedback

2 BACKGROUND

2.1 Technical background / Existing technology

The transport protocol (TP) most commonly used today is TCP. TCP is a reliable transport protocol that is also responsible for reducing the impact of network congestion. Conventionally, TCP/IP networks signal congestion by dropping packets. TCP reacts to packet losses by controlling how packets are retransmitted, e.g., by dropping its transmission window size and then gradually increasing it (AIMD) to test for the available network capacity.

The above mechanism is not optimal in the case when the client accesses the server through a wireless (e.g., RAN) domain (see Figure 1) mainly because of the different characteristics of the two (i.e., Internet and RAN) domains. The Internet domain in general requires a TCP-friendly (AIMD-like) functionality to achieve fairness between the competing flows. Some access domains as the cellular networks today, however, control the resource sharing among the flows thus they do not require TCP-like CC, and would likely achieve a better performance by using a different CC, since they have different delay characteristics than the fixed bottlenecks.

Figure 1 Reference scenario for the concept: the traffic between a Client and a Server crosses two domains with different characteristics (e.g., Internet and RAN, respectively)

To be able to adapt the CC according to the specifics of the different bottlenecks requires that the bottleneck sends additional information about the characteristics of the given bottleneck. For example, in networks with large bandwidth-delay products such as satellite link, it was proposed to use an Explicit Rate Notification (ERN) protocol such as the eXplicit Control Protocol – XCP, where routers inform the server of the optimal sending rate [1].

A successful alternative approach is to apply PEPs (Performance Enhancing Proxies) at the border of different domains and implement different CCs for each domain. This split-connection solution, which is commonly used today, is based on terminating the TCP connection in a sender agnostic way at the PEP and using a different TCP algorithm for the other domain. For the requirements of end-to-end encryption at the transport layer the Middlebox Cooperation Protocol (MCP) was proposed, where end-to-end encryption is not affected, and it allows the middlebox to send explicit, declarative, safe to ignore, incrementally useful information. Such PEP solution was proposed in [2] which enables Multi-Domain CC and does not add delay to the end-to-end communication. Moreover, it involves minimal processing and storage in the PEP (no buffering, no indexing, etc.) and it does not contribute to transport ossification. In addition, it does not require client

modification.

The Multi-Domain congestion control mechanism has two components at the server: one clocked by the E2E acknowledgements from the receiver, and one by the acknowledgements received from the PEP. The two different acknowledgements are used by these different CCs. The PEP Acks/Nacks packets received from the server that enables the Server to identify in which domain the packet loss has happened: if a packet is Acked from the PEP and not Acked from the client then it was lost in the domain below the PEP; if a packet is not Acked from either it was lost on the Internet (i.e., between the PEP and the Server).

In both scenarios above the Server can apply a congestion control that takes into account this additional information to improve the overall end-to-end performance. In [3], migration scenarios with ERN capable routers are investigated. That is, a heterogeneous network environment is assumed with some routers supporting ERN but other not. It is proposed that in such environment the Server calculates both the E2E (End-to-End) congestion window and the congestion window from the ERN feedback and uses the minimum between both congestion windows for the congestion control. It is shown that this approach may improve performance while keeping fairness between competing bottleneck flows.

The above algorithm may also be adapted for the TP PEP feedback scenario in [2], in the following way. Two congestion windows are maintained in the Server:

- One for Domain 1 (CWND_PEP), based on the PEP Acks (Ack_P), which assumes a TCP-fair behavior (e.g., by using the CUBIC flavor, with Slow Start and AIMD-like congestion avoidance

- One end-to-end (CWND_E2E), based on the client Acks (Ack_C), which assumes that the congestion happens in Domain 2 (note that packet losses in Domain 1 will be visible though PEP Nacks too, thus impacting the window CWND_PEP and triggering congestion control for the Domain 1 losses). Thus, an optimized congestion avoidance algorithm (e.g., PCC-like) may be applied relying on the assumption that Domain 2 controls the resource sharing among the different completing flows, and not the TP mechanisms.

Then, the Server uses the minimum of the two congestion windows, min(CWND_PEP, CWND_E2E), to control the packets in flight i.e., the number of bytes sent but not yet Acked by the client. As for the ERN case, it is expected that such an algorithm can provide a more efficient congestion control in both domains (via faster feedback for Domain 1 and optimized CC for Domain 2), while keeping the fairness in Domain 1.

2.2 Problems with existing solutions

The problem with the default solution above relates to the difference in delay of the feedback loops from the two domains that in some cases can adversely impact the efficiency of the congestion control. This is exemplified by some examples below.

The first example refers to the slow-start phase. At the start of the data transfer, both CWNDs maintained by the server start from their initial values. For simplicity, let us assume that the initial values are the same, and let us denote them by IW. In the slow-start phase, the CWNDs are increased exponentially until congestion is experienced. In this phase, thus, the CDNW increase may be described by the following formulae:

$$CWND\_E2E\ (t) = IW * 2^{\wedge} (t/RTT2)$$

$$CWND\_PEP\ (t) = IW * 2\ ^{\wedge} (t/RTT1).$$

Note that this is an approximate formula. The CWNDS are recalculated each time an Ack is received. The receipt of an Ack depends on a few factors, e.g., sender pacing of traffic shaping e.g., at the bottleneck link. Here we neglect these effects and assume that the Acks are received at multiple times of RTT1 and RTT2, respectively, which gives the above formulae. Note that in this approach the CWND values are fixed until the next Ack arrives, which happens after a further RTT1 and RTT2, respectively.

Since RTT1 is smaller than RTT2 (see Figure 2) it is apparent that CWND_PEP will increase faster than CWND_E2E, thus it is always CWND_E2E that will determine the overall window that is the minimum of the two CWNDs. This means, that the slow-start phase on Domain 1 is effectively clocked by RTT2 rather than RTT1. This results in larger download times for small content transfer if there is no congestion, because of this slower window increase.

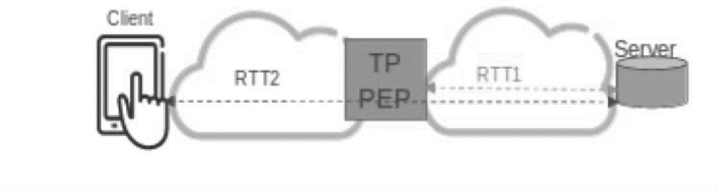

Figure 2 Visualization on the RTTs of the Client-Server (RTT2) and PEP-Server (RTT1) relations, respectively

Yet another problem refers to the congestion control mechanism based on the calculated congestion windows, which is based on calculating the packets in flight based on the minimum of CWND_E2E and CWND_PEP, relative to the last acknowledged bit. In this logic this should be derived from the Client Acks, which acknowledge the packets received via both domains. However, this mechanism does not leverage on the knowledge that there are packets acknowledged by the PEP, which means that they have been successfully sent over Domain 1 and thus could potentially serve as non-congestion indication for Domain 1 triggering a faster ramp-up of the congestion control.

A numerical assessment of the performance may be found in section *5.3*.

3 BRIEF SUMMARY OF THE PROPOSED SOLUTION / ABSTRACT

We propose an improved mechanism for the two-domain congestion control when the source domain of packet losses can be identified by the server (e.g., based on Acks from a PEP placed on domain borders). Two congestion windows are maintained in the server based on the feedback from the Client and PEP, respectively, but it is proposed that the CWND for Domain1 is fully determined by PEP Acks i.e., the last acknowledged bit from the PEP will determine where the CWND for Domain 1 is measured from. Further, it is proposed that in the slow-start phase the CWND for Domain 2 is increased in such a way to avoid unnecessary/excessive window ramp-up limitations caused by the longer latency of Client Acks, i.e., by taking into accounts the relative RTTs of the Acks received from the two domains.

4 ADVANTAGES OF THE PROPOSED SOLUTION

The main advantage of the proposal is that it improves the CC mechanism by eliminating the limitations identified for the two-domain CC that may be derived from SOA solutions, leveraging on the shorter feedback loop from the congestion control on the first (Internet) domain (see the calculations in the detailed description). Note that the method can still work also with encrypted transport protocols like QUIC, as described in the concept paper [2], The method can work with supportive servers, which could enable proprietary implementation for some main Content Providers (and later standardized, if proven favorable). Thus, this solution has the potential to provide lightweight cooperation and change the OTT providers' mindset of encrypting everything and not cooperate with middleboxes.

5 DETAILED DESCRIPTION OF THE PROPOSED SOLUTION AND FIGURES

5.1 Calculation of bytes to send

The detailed method is as follows. In the SOA- derived method, one used the following formula to determine the bytes sent out by the server at a given time:

$$WinEnd = Ack\_C + \min(CWND\_E2E, CWND\_PEP),$$

Where CWND_PEP and CWND_E2E, are the two congestion windows updated upon arrival of Client, and PEP Acks, accordingly, and Ack_C denotes the last acknowledged bit by the Client.

Two congestion windows are maintained also in our proposal, CWND_PEP and CWND_E2E, but, according to the proposal, CWND_E2E is measured as usual from the first unacknowledged bit by the client, while CWND_PEP is measured from the first unacknowledged bit by the PEP (see Figure 3). The bytes sent by TP WinEnd is then calculated each time there is an Ack/Nack received from the Client or PEP, respectively, with the following formula:

$$WinEnd = \min(Ack\_C + CWND\_E2E, Ack\_P + CWND\_PEP)$$

Where Ack_C and Ack_P denotes the last acknowledged bit by the Client and PEP, respectively, received by the Server. The result of the modification in the computation is that the second term grows more rapidly (because of Ack_P) potentially increasing the overall number of sent bytes at a given instance.

5.2 Congestion window computation at slow-start

Here we describe the second proposed modification in order to enable that one can fully leverage on the benefits of the first one.

The idea relates to CWND_PEP computation, which is maintained to control the potential congestion in Domain 1. It is increased and decreased based on Ack_P in such a way to provide a TCP-fair behavior. This is needed because Domain 1 resembles the Internet domain, where a TCP-fair TP behavior should be guaranteed. In the congestion avoidance phase this could e.g., resemble the TCP CUBIC flavor. In Slow Start phase, a potential algorithm would look like:

- $CWND\_PEP(t) = IW1 * 2^{(t/RTT1)}$.

Here, again, we assumed that all Acks are received from PEP at multiple times of RTT1, where at the given time an Ack-train equals the CWND-size from the previous computation, and the CWND is kept fixed until the next Ack arrives.

CWND_E2E is maintained to control congestion for Domain 2, and by default a similar slow-start algorithm could be used, e.g.,

- $CWND\_E2E(t) = IW2 * 2^{(t*RTT2)}$,

Comparing the two CWNDs it is apparent that CWND_E2E is likely to control the bytes sent by the server, as it increases more slowly (doubling its size at each RTT2, which CWND_PEP is doubling its size every RTT1. Therefore, we propose to start CWND_E2E with a higher initial value IW2 > IW1 to avoid this limitation at least for the first phase of data transfer. Note that this modification does not have adverse impact on the TP behaviour, if we rely on the assumption that Domain 2 controls the resource sharing among the different competing flows, so the TP mechanisms should not be responsible for providing TCP fairness.

The value of IW2 may be determined based on the value of IW1 and the relative feedback delays RTT2/RTT1. For example, let us assume that we would like to eliminate CWND_E2E for a time period of (p-1)*RTT1. At time p, the Ack_P clocked component of WinEnd becomes

› WinEnd_P = Ack_P (p*RTT1) + CWND_PEP(p*RTT1)

› = IW1*2^(p-1) + IW1*2^p = 3*IW1*2^(p-1)

At the same time, the Ack_C clocked component of WinEnd would be:

› WinEnd_C = Ack_C *(k*RTT2) + CWND_E2E (k*RTT2) = 3*IW2*2^(k-1)

where k = INT(p*RTT1/RTT2). (note that we compute the values right after

Thus, the target is that WinEnd is determined by the Domain 1 behavior in this period, we get the relation WinEnd_C >= WinEnd_P, which yields

› IW2/IW1 >= 2^ (p-k) (*)

For a numerical example please check section 5.3.

Note that to avoid the limitation of slower feedback loop for CWND_E2E, in addition to starting CWND_E2E with a higher initial value one can also increase CWND_E2E at a faster pace (or do both at the same time). For example, an optimal value to increase CWND_E2E with is 2^[(RTT2/RTT1)]* MTU for each received Client Ack of size MTU, because this ensures that the two CWNDs increase with the same pace.

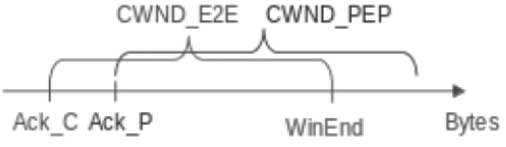

Figure 3 Relation between the congestion windows maintained by the Server

5.3 Numerical assessment of the performance gains with the method

(Note: this section is intended for internal evaluation only!!!)

In this section we evaluate the differences resulting from our proposal compared to the state of art in an example scenario.

Figure 4 shows the impact of clocking the Domain 1 congestion behaviour by Ack_P. The initial window of CWND_PEP is set to 10 MTU, and it is assumed that the different RTTs are RTT1 = 10ms and RTT2 = 20ms, respectively; thus, the ratio between the two RTTs is 2. The blue curve corresponds to the WinEnd resulting from the SOA-derived method (first formula from 5.1), when CWND_E2E is not limiting, while the orange curve shows the same, but using the proposal in this IvD (second formula from 5.1). The orange curve is always higher. Note the logarithmic scale, which means that the proposed method allows for at least 1.5 times more bytes to be sent at a given time than the SOA.

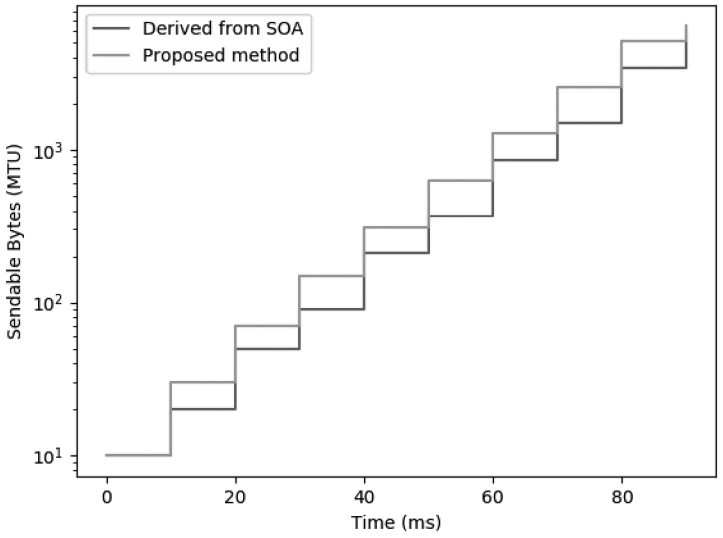

Figure 4 Comparison of bytes to be sent as a given time resulting from PEP feedback, by using the SOA-derived method, and our proposal, respectively.

Next, we analyse the impact of setting a higher IW for CWND_E2E. For this example, we assume again the same RTTs and CWND_PEP IW as before.

Figure 5 shows the bytes to be sent as a given time for the different components in the WinEnd calculation formula. In the Client-clocked component we also show the dynamics of this term for a reference case, i.e., when the IW of CWND_E2E is set to the same value as for the CWND_PEP, i.e., 10MTU, by the green curve, while the blue shows the values of the same term when using the proposal of setting a higher IW2 for CWND_E2E. The IW2 selection is based on the derived formula (*) between the two windows, when the criteria is to eliminate the limitation of CWND_E2E during the first 9*RTT1, i.e., 90 ms. That is, p=9, k=4, and therefore IW2 >= IW1*2^5 = 320 MTU. This is thus the proposed starting value resulting for CWND_E2E, and as is apparent from Figure 5, it indeed guarantees that the client-clocked component will not be limiting WinEnd until the desired time interval of 90ms.

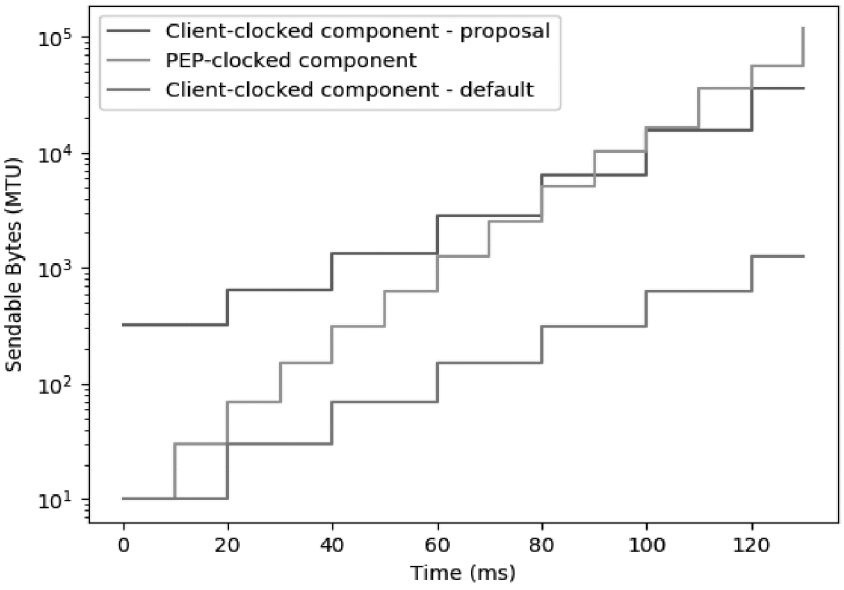

Figure 5 Evolution of the client and PEP-clocked components in the WinEnd calculation formula for the selected parameter list and criteria

Figure 6 also shows the bytes allowed to be sent (sendable bytes) given by the components of the WinEnd calculation formula at a given time. In this scenario, RTT1 = 10 ms, RTT2 = 40 ms and CWND_E2E is increased faster to compensate for the slower feedback loop, while also having a higher initial value. RTT2 is 4 times higher than RTT1 and thus CWND_E2E is increased by a factor of 2^4=16 in accordance with section 5.2. Setting IW2 to a value of 16 * IW1=160 results in the following.

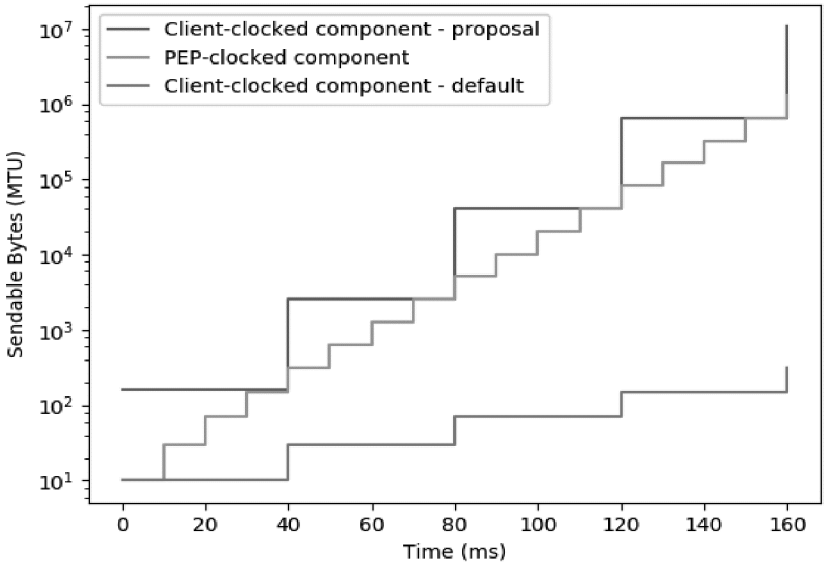

Figure 6: Evolution of the client and PEP-clocked components in the WinEnd calculation formula when we allow faster increase for CWND_E2E

In Figure 6, the WinEnd achieved by the proposed method is the PEP-clocked component while the SOA-derived WinEnd is the same as the reference values shown in green. Table 1 further illustrates the difference, comparing the WinEnd values in MBs.

Time (ms)	Sendable Bytes (MB)	
	SOA – derived method	Proposed method
0	0.015	0.015
40	0.045	0.465
80	0.105	7.665
120	0.225	122.865
160	0.465	1966.065
200	0.945	31457.265
240	1.905	503316.465

Table 1: Bytes sent at a given time using the SOA-derived method and the proposed method

With RTT1=10ms, RTT2=40ms and CWND_E2E increased in accordance with section 5.2, after 240 ms, the SOA-derived method sends nearly 2 MB of data, while the proposed method may send more than 503 GB.

This could cause large packet losses that could adversely affect the transport protocol performance. Note that, since the RAN domain provides resource sharing for the individual users, a too large IW selected for this link can have potential negative impacts only on the given connection or other traffic of the given UE, i.e., the traffic of other users is not affected. The method does not propose exaggeratedly high IW for the RAN domain; nevertheless, if one could also use a properly adapted CC algorithm (slow-start) in the RAN domain if needed to cater for potential negative effects.

6 CORE ESSENCE OF THE SOLUTION

A method for congestion control in a Server receiving congestion notification from two domains in a way that the source domain of the packet losses can be identified

Where different CWNDs are maintained for the two domains

Where the CWND for the closer domain to the server is fully determined by congestion feedback information received from the given domain, i.e., the last acknowledged bit from this domain determines where the CWND for this domain is measured from.

Where, in the slow-start phase the CWND for the more distant domain is increased in such a way to avoid unnecessary/excessive window ramp-up limitations caused by the longer latency of congestion feedback received from this domain.

- Where the initial value of this CWND is set to higher value than that of the CWND of the closer domain
- Where the increase pace of this CWND is faster than that of the CWND of the closer domain.

- Where the optimal settings of the values above are determined based on the measured relative round-trip times to the entities sending congestion notifications in the two domains

7 ABBREVIATIONS

Abbreviation	Explanation
Ack	Acknowledgement
AIMD	Additive Increase Multiplicative Decrease
CC	Congestion Control
CWND	Congestion Window
IW	Initial Window
OTT	Over The Top
PEP	Performance Enhancement Proxy
RAN	Radio Access Networks
TCP	Transmission Control Protocol
TP	Transport Protocol

8 REFERENCES

[1] D. Katabi, M. Handley, and C. Rohrs, “Congestion Control for High Bandwidth-Delay Product Networks,” in ACM Sigcomm, 2002.
[2] A. Mihaly, Szilveszter Nadas, Sandor Molnar, Zsolt Kramer, Robert Skog and Marcus Ihlar, “Supporting Multi-Domain Congestion Control by a Lightweight PEP” IINTEC 2017, in press, http://hsnlab.tmit.bme.hu/~molnar/files/iintec2017.pdf
[3] Tran-Thai, Tuan & Lopez Pacheco, Dino & Lochin, Emmanuel & Arnal, Fabrice. (2018). Towards an incremental deployment of ERN protocols: a proposal for an E2E-ERN hybrid protocol, Proceedings of Towards AI Pacheco, 2011,