# Supporting Multi-Domain Congestion Control by a Lightweight PEP

Attila Mihály*, Szilveszter Nádas*, Sándor Molnár†, Zsolt Krämer†, Robert Skog‡ and Marcus Ihlar‡
*Traffic Analysis and Network Performance Laboratory, Ericsson Research, Hungary
{attila.mihaly, szilveszter.nadas}@ericsson.com
†Dept. of Telecomm. and Media Informatics, Budapest University of Technology and Economics, Hungary
{molnar, kramer}@tmit.bme.hu
‡Ericsson Sweden
{robert.skog, marcus.ihlar}@ericsson.com

***Abstract*—The performance issues of current transport protocols (mainly TCPs) operating in today's heterogeneous networks motivate the application of Performance Enhancing Proxies (PEPs). However, current PEPs can introduce several undesired effects like extra delay, non-negligible processing demands and ossification of the transport layer. The ossification resulted in the encryption of the end-to-end transport layer, which made transparent PEPs impossible to use. We propose a lightweight transport protocol PEP solution which enables Multi-Domain Congestion Control and requires no client modification. This PEP proposal is declarative, explicit and does not contribute to ossification. The potential performance improvement by the PEP is demonstrated with simulation examples.**

***Index Terms*—PEP, Middlebox Cooperation, Multi-Domain Congestion Control, cellular access network, TCP.**

## I. Introduction

The performance of the Internet highly depends on its transport protocols operating across the whole network between any source and destination endpoints. TCP (Transport Control Protocol) is the dominant transport protocol, providing reliable and efficient end-to-end communication.

TCP was designed for fixed networks, however, our current Internet has a significant component in the mobile domain due to the increasing popularity of smartphones, tablets, etc. connected to the global Internet. In contrast to the fixed network the mobile data communication provided by the the 3G-5G networks have different characteristics which may impact the TCP performance in these networks. Packet losses on the wireless links are not an exclusive indicator of a congestion event but may be due to radio transmission errors and thus may result in poor TCP performance. Other non-congestion events typical for mobile networks like packet losses during mobile terminal handovers or reordering of packets over multilink access may also degrade TCP performance.

An obvious and popular approach to handle non-congestion losses is applying a reliable *data link-layer protocol* [1]. This solution hides the link losses from the transport protocol by using link retransmissions, Forward Error Correction (FEC) mechanisms, etc. Cellular RANs (Radio Access Network) (and many other access domains) implement such internal retransmission loops, like the RLC AM and PDCP retransmission [2]. The standard handover procedures also include packet buffering at the old base station and forwarding to the new base station in order to avoid losses during mobile handover [3]. Another example link-layer protocol is the snoop protocol [4], which introduces an agent that monitors every packet that passes through the TCP connection in both directions, and retransmits a lost packet towards a client on wireless link from its local cache while suppressing the duplicate acknowledgments from the client. While the snoop protocol can achieve good performance [5], due to its dependence on TCP behavior it cannot be applied as a generic replacement of transport protocol agnostic link-layer protocols.

The advantage of reliable link layer protocols is that they do not require changes to the end-to-end transport protocol. The cost what one has to pay is the relatively high complexity of the applied link-layer mechanisms and varying RTT due to retransmissions. It is thus very hard to achieve low delay on the current RANs. This can have a negative impact on the performance of delay sensitive applications and also of certain delay-based TCP implementations e.g., CUBIC's HyStart [6].

A different approach to solve the TCP performance problem is an *end-to-end solution* requiring the change or modifications of TCP [7]. The attractive feature of end-to-end optimizations is that they do not involve support from intermediate nodes. For example, Freeze-TCP [8] tries to detect handoffs by the modification of client TCP stack. However, the end-to-end improvements may optimize for certain scenarios, thus are not flexible to be applied as generic solutions.

Another approach to handle non-congestion losses is that the receiver sends more verbose information e.g., an Explicit Loss Notification (ELN) [5]. The ELN proposal is built upon the idea that the MAC layer is able to detect packet losses and notify the TCP layer. This makes the sender able to distinguish between different forms of losses and optimize the congestion control (CC) based on this information. As an example, TCP Jersey [9] uses ELN support from routers.

In the absence of such explicit feedback, the sender could also try to automatically tune the congestion control (CC) parameters based on the experienced performance. Such a method using an adaptive CC for optimized performance is

978-1-5386-2113-4/17/$31.00 

the recently proposed PCC (Performance-oriented Congestion Control) mechanism [10]. However, PCC may not always be a feasible choice, since some domains on the path may still require TCP-friendly functionality for fairness.

Some cellular networks would not require an Adaptive Increase Multiplicative Decrease (AIMD) type CC, because the resource sharing is controlled by internal mechanisms. Consequently, using a different CC in these networks could provide better transport protocol efficiency. If, for example, the CC provides a faster adaptation to fast changing bottlenecks typical to cellular access networks, this will result in higher overall throughput. Also, if a faster slow-start is used (than that in the current TCP) then shorter download times can be achieved for small content transfers. There is thus still demand for multi-domain congestion control for access domains applying link-layer protocols. This issue is acknowledged as "Challenge 5: Multi-Domain Congestion Control" in [11]. An end-to-end solution controlling different CC behavior for different domains at the sender side would require information about the origin of the congestion (e.g., ELN); however, this is not feasible due to many legacy bottlenecks.

One feasible approach for multi-domain congestion control is the *split-connection solution*. The common solution used today is based on terminating the whole TCP connection in a sender agnostic way at an intermediate device (i.e., transparent Performance Enhancing Proxies (PEPs)) and using a different TCP algorithms for the mobile domain [12].

Maintaining these transparent PEPs (i.e., upgrade them to support all protocol improvements) is time-consuming and legacy PEPs may cause harm to the participating flows, when the end-to-end protocol is updated. This phenomenon is known as ossification of the transport layer [13]. There are architectural proposals that partially move away from the end-to-end transport semantics of the transport layer enabling factoring of CC in the different network domains [14], [15], but without concrete implementation ideas (e.g., how backward compatibility that is needed to bypass the ossifying legacy middleboxes is achieved). Moreover, in a split-connection solution, the endpoints have no feedback on the effect of the PEP operation so they cannot compare it with the performance of the default, undisturbed end-to-end solution. This resulted in a general suspicion towards these nodes and to reclaim the goodwill of the end-points, PEPs would now have to demonstrate their value [16].

Service providers have answered to the above concerns by also applying end-to-end encryption to the transport layer [17], making the application of transparent PEPs impossible. In response to the encryption of transport protocols, proposals for Middlebox Cooperation Protocols (MCPs) [18], [19] were given. These protocols provide standardized information to middleboxes such that end-to-end encryption is not affected and allow the middleboxes to send explicit, declarative, safe to ignore, incrementally useful information.

In this paper we propose a lightweight PEP built on the principles of MCP which has the following additional attractive features:


Fig. 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)

- it enables Multi-Domain CC
- it does not add delay to the end-to-end communication,
- it involves minimal processing and storage in the PEP (no buffering, no indexing, etc.),
- it does not contribute to transport ossification and
- it does not require client modification.

The rest of the paper is organized as follows. We introduce the concept of our proposed PEP in Section II. In Section III, we present simulation results to demonstrate the performance achievements applying this concept. We discuss deployment issues in the Section IV. Finally, Section V concludes the paper.

## II. Lightweight PEP Concept

### A. Scenario and goals

We target a scenario when a client uses wireless access to connect to an Internet server; it is illustrated in Figure 1, where the traffic crosses two domains:

- *Domain 1*, where resource sharing is controlled by the end-to-end CC. It is characterized by that packet loss/reordering always means congestion, and that the RTT is often low. This models the Internet domain with Content Delivery Network (CDN) servers placed close to access domain.
- *Domain 2*, where resource sharing is controlled by domain internal mechanisms. It often has more variable available capacity. This models wireless access networks. We differentiate two options from packet loss/reordering perspective:
  - *Domain 2 - type a*, where internal retransmission loops are implemented to hide packet loss/reordering, and make the domain TCP-compatible. This causes head of line blocking within the domain, which results in high jitter.
  - *Domain 2 - type b*, where packet loss/reordering can happen independently of congestion.

We target to allow faster start and adaptation in Domain 2, while keeping the TCP-compatible behavior in Domain 1. We propose to clock the Domain 1 behavior by the Domain 1 RTT. We target a lightweight, incrementally deployable solution, which requires minimum trust from the endpoints, especially from content perspective. Therefore, we introduce an explicit PEP sending safe to ignore messages, which can be used to determine the domain where packet loss happened. This information can then be used to provide TCP compatible

congestion control in Domain 1, while providing faster ramp-up, and optionally loss resilience, for Domain 2 (both types)[1]. Our goal is to demonstrate how Network Vendors and network operators can contribute to the solution of this problem. Therefore we concentrate on the functionality and information PEPs can provide to the end-to-end congestion control, and we do not detail the necessary changes to the congestion control itself, which is left for future work.

The proposed concept focuses on downlink traffic. An uplink solution would result in smaller impact on Quality of Experience, mainly due to the huge disparity between downlink and uplink rates observable in TCP flows captured in cellular networks [20].

### B. PEP placement and general behavior

We place our PEP function at the domain border between the two domains. It ACKs/NACKs packets received from Server , see Figure 2. It also immediately forwards the packets unmodified towards the Client[2]. The Client then ACKs the received packets as usual. This PEP feedback enables the Server to identify where 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, i.e., Domain 2; if a packet is not ACKed from neither the PEP nor the Client then it was lost on the Internet (i.e., between the PEP and the Server). The Server can then apply a congestion control that takes into account this additional information to improve the overall end-to-end performance.

Note that the idea of sending ACKs by local PEPs is not novel (see Section 3.1.2 in [12]). However, the assumption with the existing proposals has been that when local acknowledgments are used, the burden falls upon the acknowledging PEP to recover any data which is dropped after the PEP acknowledges it. As the PEP ACKs in our proposals do not replace client ACKs there is no need for data recovery by the PEP. An advantage of this simplified functionality is that the transport layer will not be affected if a PEP fails; if for any reason the PEP stops responding the Server might be able to detect it (i.e., ACK received from the Client, not from the PEP) and it can fall back to legacy behavior. Please also note that the ACK traffic is negligible so the effect of the PEP by doubling the ACK traffic has no significant overhead since it is also negligible.

### C. Solution details

A solution example is shown by the sequence diagram in Figure 3. The PEP ACKs/NACKs packets as received from Server (Steps 2, 6, 9, 11). It also immediately forwards all received packets unmodified to the Client (Steps 1, 5, 8, 10).

The Client acknowledges the received packets as usual (Steps 3, 7, 12). In case of loss in Domain 1, the two ACKs

[1]Resource Sharing Control in Domain 2 enables the clocking of end-to-end ramp up by Domain 1 RTT, because this Control will prevent flows from overloading Domain 2.

[2]Packet duplicators, e.g., optical splitters, may be used to avoid any additional delay. When these are used the PEP discards the packets after processing.

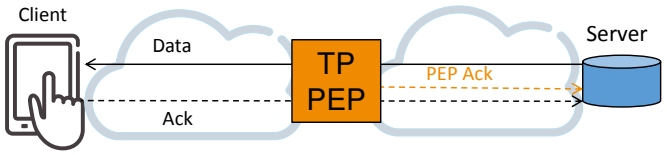



Fig. 2: Placement and functionality of the ACKing Border PEP

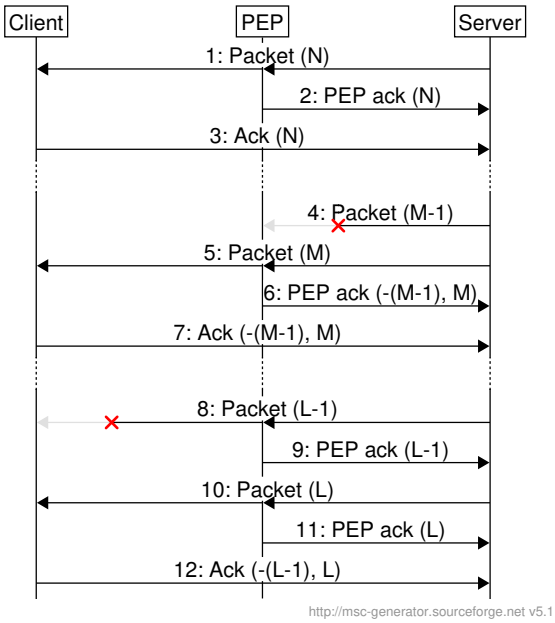



Fig. 3: Example sequence diagram illustrating a client-server communication involving an ACKing Border PEP ("-" indicates missing sequence number)

from the PEP and Client are similar (Steps 6, 7). When the packet is lost in Domain 2, the ACKs will be different (Steps 9, 11 and 12, respectively).

TCP Congestion Control can be clocked by PEP ACKs, thus slow start in effect happens between PEP and Server resulting in a faster start-up[3]. The Server may deploy a modified CC algorithm that takes into account the place of the packet loss (Domain 1 or 2). The benefit is that a different CC may be applied if the packet loss was identified as coming from Domain 2, e.g., resulting in a less aggressive congestion response. This of course has to be combined with a TCP-friendly functionality when the loss is in Domain 1.

A similar concept where the sender is also provided with the original client ACKs besides the PEP ACKs has been proposed in [21]. However, in that paper the congestion in both radio and Internet domains was handled by the same TCP-friendly CC; the PEP ACKs were used only to identify potential non-congestion losses in the radio domain that were retransmitted by the server in a fast retransmit manner.

By design our concept enables that the CC can be changed and evolved by end-points. We show potential performance

[3]Similarly to e.g., [4], however our solution does not require any buffering in the PEP.

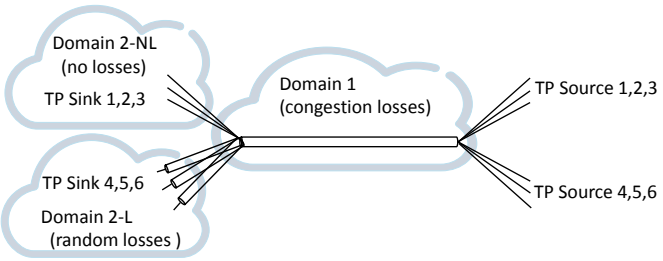



Fig. 4: Topology used for the simulation

improvements by a very simple CC example in Section III.

PEP ACKs do not affect TCP retransmission behavior, i.e. client NACK will still trigger TCP retransmission.

### D. Further considerations

The solution may be extended with further communication between the PEP and the server. For example, the PEP might send additional information about the characteristics of Domain 2, e.g., available bandwidth indication (like [22]) to further optimize start-up or the CC in general. Another option is that PEP indicates the preferred CC behavior in Domain 2.

## III. Simulations illustrating the performance

In order to illustrate the potential benefits of the concept a simulation study was performed using ns-2 simulator [23]. The scope is limited to show the usefulness of the PEP for access domains with random losses, i.e. *Domain 2 - type b*. We used the topology shown in Figure 4, where six TCP sources are competing for the capacity of a bottleneck link in Domain 1 (representing the Internet domain). Three flows terminate in Domain 2-NL representing a loss-free domain, i.e., in general there is no congestion and packet losses here. The other three TCP flows terminate in Domain 2-L representing a domain with sources of non-congestion losses (e.g., wireless domain with no or limited link-layer retransmission). The probability of random losses in Domain 2-L was set to $5*10^{-3}$. The fixed end-to-end RTT for both sources is set to 100 ms.

We compared the results averaged from 10 independent runs using different random seeds for three different scenarios. The link capacities for Domain 1 and Domain 2-NL were always set to 120 Mbps, while the simulations for each scenario were repeated for two different Domain 2-L link capacity values, (a) 120 Mbps, and (b) 8 Mbps, respectively; in the latter case, the TCP flow will also experience congestion losses besides random losses in Domain 2-L. Domain 2-L endpoints are connected to Domain 1 by different links (each one of speed 120 or 8 Mbps), in accordance with the premise that the transport layer is not responsible for resource sharing in this domain.

In Scenario 1, all TCP sources use the same TCP CUBIC CC [24]. As expected, due to the non-congestion losses the TCP flows in Domain 2-L under-perform in both cases (a) and (b), see Figure 5.

In Scenario 2 we assume that the TCP source terminating in Domain 2-L is expecting non-congestion packet losses and therefore uses an adapted CC algorithm, which is more robust to packet loss signals. The modified CUBIC CC avoidance mechanism is triggered only if the TCP source encounters at least 10 packet loss events per RTT. The results show that in this way it is possible to use the whole bottleneck capacity of Domain-2L (Figure 6b), but when the bottleneck is in Domain 1 then the modified CC algorithm of the TCP sources terminating in Domain 2-L grabs more capacity (Figure 6a) than its fair share (it is not TCP friendly anymore).

Scenario 3 corresponds to the case when we assume that our lightweight PEP is located at the border of Domain 1 and Domain 2-L, and the Server CC is tuned based on the information received from this PEP. In our implementation we tagged the losses in Domain 2-L, rather than implementing the PEP function directly. The TCP sources terminating in Domain 2-L perform the above modified CC algorithm only when encountering packet losses tagged by Domain 2-L. They react to Domain 1 losses normally (like CUBIC CC). The results presented in Figure 7 show that fairness is re-established in Domain 1, while the TCP sources are still able to utilize the full capacity of Domain 2-L when that is the bottleneck.

| Scenario | Throughput [kbps] | | Jain's index |
|---|---|---|---|
| | Domain 2-NL | Domain 2-L | |
| 1a: default CC | 38575 | 1425 | 0.537 |
| 2a: modified CC | 11212 | 28788 | 0.834 |
| 3a: modified CC + PEP | 19540 | 20460 | 0.977 |

TABLE I: Average throughput of flows terminated in Domain 2-NL and Domain 2-L, when the bottleneck is in Domain 1

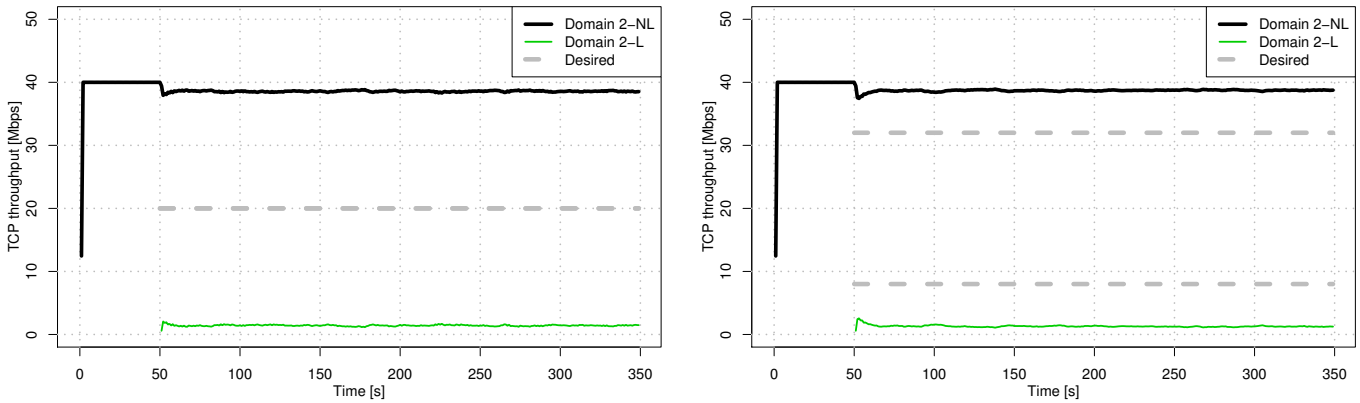
Table I summarizes the results of the performed simulations between 200 s and 350 s. The Jain's index [25] shows that the PEP functionality achieved nearly perfect fairness in Scenario 3a.

## IV. Deployment considerations

The concept requires update in Server and evidently the PEP functionality itself, while the Client and the Network Domains are completely agnostic to the solution. Updating legacy servers is motivated by the expected performance improvement.

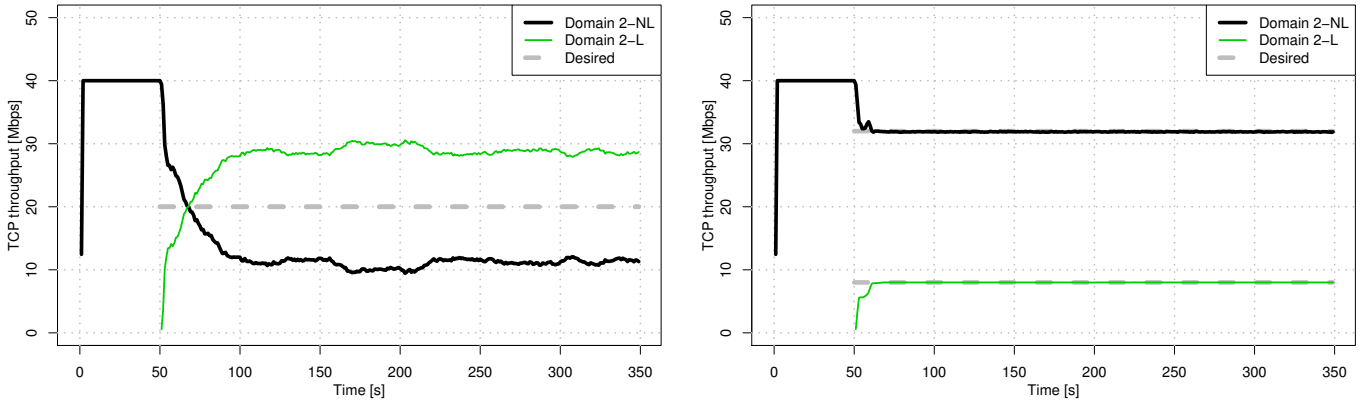
The PEP implementation is lightweight compared to transparent transport protocol layer PEPs as there is no need for packet buffering, decryption and re-encryption. The PEP only needs to interpret connection ID and sequence number on each packet and handle each connection separately. Accessing these fields might be trivial (e.g., TCP) or require some changes (e.g., QUIC [17]).

The Server updates include the receipt and processing the PEP ACK messages and the updated CC. It can also implement means to verify the advantages of PEP cooperation, e.g., by opting out for some connections and comparing the performance. The server may also include entropy with entropy bits



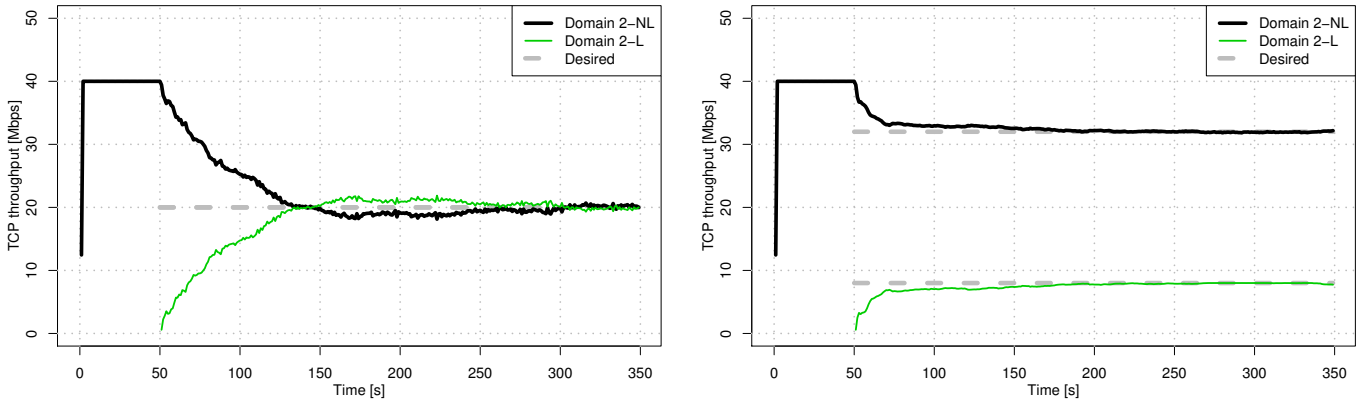

(a) (b)

Fig. 5: Simulation Scenario 1. Throughput dynamics of the six TCP sources terminating in lossless Domain 2-NL (black) and Domain 2-L with random losses (green), respectively, both using CUBIC-CC. The throughput of the flows are averaged for the two domains. Figures (a) and (b) show simulations for Domain-2L bottleneck capacity of 120 Mbps and 8 Mbps, respectively. Dashed line corresponds to the desired throughput of the two TCP flows.




(a) (b)

Fig. 6: Simulation Scenario 2. As above, but TCP flows terminating in Domain 2-L (green) are using a modified TCP CUBIC to cater for non-congestion losses.




(a) (b)

Fig. 7: Simulation Scenario 3. As above, but the modified TCP CUBIC of the TCP flows terminating in Domain 2-L (green) differentiates between Domain 2-L and Domain 1 losses.

or missing sequence numbers (see [17]) in the sent packets to identify hostile PEP behavior, i.e., ACKing not received packets to make the Domain 1 CC more aggressive. Another alternative to is that the PEP a crypto hash of the packet payload, the PEP may also prove it has really seen the packet it has AKCed. These means for Server verification provide the trust for the PEP cooperation.

The PEP messages must reach the server and also must not generate any error in legacy servers. If a Middlebox Cooperation Protocol is included in the end-to-end stack, that can be used to send PEP ACK packets to the server. Alternatively, it is possible to insert these ACKs into existing TCP packets as TCP options, similarly to the proposal in [22]. Other non-standardized ways are also possible. One way is to directly send messages to the server out-of band e.g., using UDP. Also the ACK packets can be prepared using the same 5-tuple as the transport protocol, but using wrong CRC and a magic number to easily identify this procedure [26]. Note that none of the communication means above has unwanted effects on legacy servers: they would simply drop the PEP ACK packets. They also do not require user plane information knowledge by the PEP, thus the concept may be applied also for end-to-end encrypted traffic with public packet sequence numbers. That is, confidentiality of end-to-end communication is guaranteed, unlike the cases with split-connection methods.

## V. Conclusions

We introduced a lightweight Performance Enhancing Proxy (PEP) concept to enable Multi-Domain Congestion Control to keep fairness in the Internet domain and also provide efficient adaptation in the access domain. This explicit PEP sends declarative, safe to ignore, incrementally useful information to the servers. It is designed not to be a new point of ossification, and to enable future transport protocol and congestion control innovation. It does not introduce any delay to the end-to-end communication. It has small processing and storage demands and no packet buffering is needed. It does not affect end-to-end encryption and does not require modification to the clients. The PEP leaves the congestion control algorithm in the server, while it enables improved performance.

The next step in this study is to further analyze the potential gains of the solution in scenarios where Domain 2 resembles realistic radio access domain characteristics in terms of packet loss, delay and bandwidth variations and identifying key congestion control functionality to achieve optimal performance.

## References


[1] A. Larmo, M. Lindström, M. Meyer, G. Pelletier, J. Torsner, and H. Wiemann, "The LTE link-layer design," *IEEE Communications magazine*, vol. 47, no. 4, 2009.

[2] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access (E-UTRAN); Overall description," 3rd Generation Partnership Project (3GPP), TS 36.300, 2016. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/36300.htm

[3] R. Caceres and V. N. Padmanabhan, "Fast and scalable handoffs for wireless internetworks," in *Proceedings of the 2nd annual international conf. on Mobile computing and networking*. ACM, 1996, pp. 56–66.

[4] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over wireless networks," in *Proc. of the 1st annual int. conf. on Mobile computing and networking*. ACM, 1995, pp. 2–11.

[5] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *ACM SIGCOMM Comp. Comm. Review*, vol. 26, no. 4, pp. 256–269, 1996.

[6] R. Robert *et al.*, "Behaviour of common TCP variants over LTE," in *Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–7.

[7] K. Liu and J. Y. Lee, "On Improving TCP Performance over Mobile Data Networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2522–2536, 2016.

[8] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta, "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments," in *INFOCOM 2000. 19th Annual Joint Conference of the IEEE Comp. and Comm. Societies. Proceedings*, vol. 3. IEEE, 2000, pp. 1537–1545.

[9] K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for wireless IP communications," *IEEE Journal on selected areas in communications*, vol. 22, no. 4, pp. 747–756, 2004.

[10] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting Congestion Control for Consistent High Performance," in *NSDI*, 2015, pp. 395–408.

[11] D. Papadimitriou, M. Welzl, M. Scharf, and B. Briscoe, "Open research issues in internet congestion control," IRTF RFC 6077, Tech. Rep., February 2011.

[12] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance enhancing proxies intended to mitigate link-related degradations," Internet Requests for Comments, RFC Editor, RFC 3135, June 2001.

[13] G. Papastergiou, G. Fairhurst, D. Ros, A. Brunstrom, K.-J. Grinnemo, P. Hurtig, N. Khademi, M. Tuxen, M. Welzl, D. Damjanovic *et al.*, "De-ossifying the internet transport layer: A survey and future perspectives," *IEEE Communications Surveys & Tutorials*, 2016.

[14] B. Ford and J. R. Iyengar, "Breaking up the transport logjam," in *HotNets*, 2008, pp. 85–90.

[15] F. R. Dogar and P. Steenkiste, "Architecting for edge diversity: supporting rich services over an unbundled transport," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 2012, pp. 13–24.

[16] S. Nádas and A. Mihály, "Concept for cooperative traffic management," in *Managing Radio Networks in an Encrypted World (MaRNEW) Workshop*. IAB, 2015.

[17] R. Hamilton, J. Iyengar, I. Swett, and A. Wilk, "QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2," Internet Engineering Task Force, Internet-Draft, Jul. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-hamilton-early-deployment-quic-00

[18] B. Trammell, M. Kühlewind, E. Gubser, and J. Hildebrand, "A new transport encapsulation for middlebox cooperation," in *2015 IEEE Conference on Standards for Communications and Networking (CSCN)*, Oct 2015, pp. 187–192.

[19] B. Trammell and M. Kühlewind, "Path Layer UDP Substrate Specification," IETF, Internet-Draft, 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-trammell-plus-spec-00

[20] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An in-depth study of LTE: effect of network protocol and application behavior on performance," in *ACM SIGCOMM Comp. Comm. Review*, vol. 43, no. 4. ACM, 2013, pp. 363–374.

[21] D. Bosau, H. Unger, L. Lada-On, and D. Kaspar, "Loss differentiation and recovery in tcp over wireless wide-area networks," in *The Tenth International Conference on Networks (ICN 2011). IARIA*, 2011.

[22] A. Jain, A. Terzis, H. Flinck, N. Sprecher, Swaminathan, and K. Smith, "Mobile throughput guidance inband signaling protocol," Working Draft, Internet-Draft draft-flinck-mobile-throughput-guidance-03, 2015.

[23] S. Mccanne, S. Floyd, and K. Fall, "ns2 (network simulator 2)." [Online]. Available: http://www-nrg.ee.lbl.gov/ns

[24] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-friendly High-speed TCP Variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008. [Online]. Available: http://doi.acm.org/10.1145/1400097.1400105

[25] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN systems*, vol. 17, no. 1, pp. 1–14, 1989.

[26] "Method for in-band meta-data transfer, research disclosure," 2016. [Online]. Available: https://www.dropbox.com/s/6bvjcbqf57h1st5/RD623051.pdf?dl=0