Towards Multi-Domain Congestion Control in Next-Generation Networks

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

***Abstract*—The performance problems of recent transport protocols applied in fixed and high-speed wireless networks call for research for an effective congestion control approach, which is able to work efficiently in both domains. In this paper we propose a framework incorporating a novel *Multi-Domain congestion control* with a novel *Lightweight Performance Enhancing Proxy (PEP)*, which requires no client modification. The method was implemented in the Linux kernel and the Lightweight PEP in ns-3 and we carried out a comprehensive performance evaluation of our proposed framework. We demonstrate that our approach can achieve a robust behavior in the presence of wireless loss while keeping the TCP fairness in the wired domain. Moreover, we also present that our method provides a faster adaptation in the wireless domain compared to the currently used CUBIC TCP.**

***Index Terms*—Multi-Domain Congestion Control, Cellular Access Network, TCP, PEP, Middlebox Cooperation.**

I. Introduction

5G mmwave cellular networks present new challenges for TCP (Transport Control Protocol). With the use of millimeter waves, it is possible to achieve gigabit-per-second cell data rates, however, the propagation properties at these frequencies create high variability in channel conditions [1]. Sudden changes in available capacity (caused by e.g., NLOS-LOS transitions) is not a phenomenon that traditional congestion control (CC) algorithms were designed to handle. The efficient resource utilization of TCP in 5G is thus limited due to the slow responsiveness of traditional CC algorithms to sudden changes in available bandwidth.

Congestion control continues to evolve and novel algorithms recently aimed to address the aforementioned challenge. These new TCP variants do not need support from the network, and thus we call them *end-to-end solutions*. However, end-to-end TCP flows are traveling through domains with highly different characteristics, i.e., wired and wireless domains, and the performance degradations are due to different reasons in each of them. For example, packet loss is the main indicator of congestion events in the wired domain but it is frequently caused by other events in the wireless domain (radio transmission error, mobile terminal handovers, reordering of packets over multilink access, etc.). This creates a great difficulty for the correct operation of congestion control algorithms.

In order to mitigate this issue, link-layer methods can be applied, which hide the link losses from the transport protocol by using different techniques, e.g., link retransmissions, Forward Error Correction (FEC) mechanisms, etc. These belong to a category of approaches we call *service network based solutions*. Split-connection Performance Enhancing Proxies (PEPs) also belong to this category. These proxies were widely deployed in cellular networks due to the performance gains they are able to deliver, however, numerous drawbacks of these PEPs have also been discovered (see Section II).

A third approach is to leverage cooperation between entities in the network (usually middleboxes) and the end-hosts by providing additional information for the end-host to optimize congestion control. We refer to these methods as *network-end hosts cooperation solutions*. A Middlebox Cooperation Protocol (MCP) was proposed in [2] that outlines requirements for such cooperation. These protocols should not affect end-to-end encryption, and allow the cooperating middleboxes to send information which is explicit, declarative, safe to ignore, and incrementally useful.

We presented a new PEP concept built on the aforementioned MCP principles in our previous paper in [3] which belongs to the network-end hosts cooperation solution category. We introduced a *Lightweight PEP* entity placed on the border of the two domains with different characteristics as can be seen in Fig. 1. The Lightweight PEP requires minimal trust from the endpoints; it sends safe-to-ignore PEP-ACKs, requires minimal processing or storage, adds no delay to the communication and does not require any client update.

In this paper, as a continuation of this research, we propose a novel *Multi-Domain Congestion Control* utilizing this Lightweight PEP. We implemented our congestion control algorithm in the Linux kernel, the Lightweight PEP in ns-3 and we carried out a comprehensive performance evaluation of our proposed framework. Our design goal was to achieve a robust behavior in the presence of wireless loss and also a faster adaptation in the wireless domain, while keeping the TCP fairness in the wired domain.

The rest of the paper is organized as follows. The related work is described in Section II. Section III and IV present the design and implementation of our multi-domain congestion control mechanism, respectively. Section V provides a comprehensive performance evaluation of our method in different key scenarios. Section VI concludes the paper and outlines future challenges.

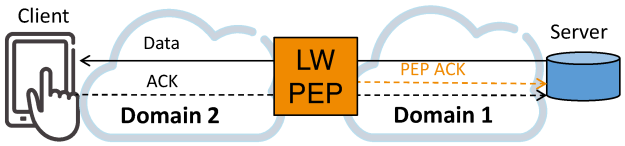

Fig. 1. Placement and functionality of the Lightweight PEP

II. Related work

A. End-to-end congestion control improvements

CUBIC [4] delivered spectacular results and is still the default congestion control algorithm in the Linux kernel. In recent years however, the research of new algorithms gained some momentum once again. Google developed BBR (Bottleneck Bandwidth and RTT) [5] which estimates available bandwidth and the minimum RTT, and it uses pacing to keep the data in flight near the bandwidth-delay product. They reported 2% lower search latency on google.com and 32% lower RTT on YouTube since the deployment of BBR instead of CUBIC, although the average loss rate increased from 1% to 2% [6]. Hock et al [7] confirmed the intended behavior of BBR, however they also observed very high induced packet loss in some cases and also potential unfairness when competing with CUBIC.

Another recent congestion control algorithm that gathered significant research interest is PCC (Performance-oriented Congestion Control) [8]. PCC uses utility functions to aggregate packet-level events to numerical performance utilities. The algorithm runs experiments (A/B testing for the sending rate), observes the performance utilities and moves in the direction of better performance. In their paper, Dong et al presented promising testbed results: PCC achieved better stability and convergence than CUBIC, however, the default utility function of PCC is very unfriendly to TCP.

Kühlewind proposed TCP SIAD (Scalable Increase Adaptive Decrease) in [9]. Among other, more common design goals (high link utilization, minimize delay) it has a unique set of novel mechanisms. TCP SIAD is designed to speed up bandwidth allocation in changing network conditions. When the congestion window becomes larger than the *Linear Increment threshold* (meaning that there is no target value of available capacity), TCP SIAD enters a *Fast Increase* phase and the congestion window grows exponentially. The paper also argues that TCP-friendliness should not be a requirement for congestion control and that fairness should be enforced on a long-term, per-user basis to allow grabbing a larger share of capacity for a short term if needed.

From the above, it is apparent that while there is a renewed interest in congestion control from both industry and academia, it is still a challenge to provide an end-to-end solution that is able to quickly adapt to bandwidth fluctuations in heterogeneous environments and also preserve traditional TCP fairness.

B. Service network based performance optimization

A common solution to implementing multi-domain congestion control in cellular networks is the deployment of a split-connection transparent Performance Enhancing Proxy (PEP) [10]. The transparent PEPs terminate the TCP connection in a sender agnostic way and may also use a different congestion control algorithm for the mobile domain. These proxies became increasingly pervasive due to the improvement they provided in performance. However, the maintenance of transparent PEPs is costly and legacy proxies may even cause pathological behaviors when the end-to-end protocol is updated. This means that transparent PEPs contribute heavily to the phenomenon that is called the ossification of the transport layer [11]. The endpoints of a split TCP connection have no feedback on the operation of the PEP so it is not possible to compare the performance with the original end-to-end solution. This has lead to a lack of trust regarding these transparent PEPs [12].

Another approach was presented in [13], called Snoop. It is a link-layer solution that suppresses duplicate acknowledgements and handles the retransmissions locally between the intermediary agent and the TCP receiver, hiding losses from the sender. While Snoop was shown to improve performance, the buffering needed for the local retransmissions can also have a detrimental effect on handovers.

C. Leveraging cooperation between the network and the end-hosts

Bosau et al [14] describe a solution which is capable of differentiating between congestion in the wired domain and corruption in the wireless domain. This is achieved by placing a non-splitting proxy at the base station which sends a new kind of acknowledgement to the TCP sender. This concept enables a more robust TCP behavior in cellular networks, however, the different acknowledgements are both clocking a TCP-friendly CC, thus it behaves too conservatively when the bottleneck is in the wireless domain.

A different approach is presented by the authors of [15] and [16]. They propose an architecture to be used in networks with large bandwidth-delay products, such as satellite communications. This method uses the eXplicit Control Protocol (XCP) to provide the optimal sending rate for the sender by a non-splitting proxy from the border of the two domains. A congestion control algorithm is also described, which maintains two congestion windows; one based on the XCP feedback and one on the client acknowledgements.

Polese et al [17] proposed a proxy architecture - called milliProxy - for 5G mmWave networks. A flow window policy is applied at the proxy which modifies the advertised window values of the ACKs relayed to the server, and then the sender chooses the congestion window as the minimum value between the congestion window and the feedback from the proxy (advertised window). This solution also implements a mechanism that can modify the maximum segment size used between the proxy and the UE, enabling further performance optimization as the proxy can aggregate multiple segments from the end-to-end connection. The authors show that milliProxy is able to achieve high goodput with low latency, however this concept involves heavy buffering and processing.

The most recent idea was presented in [18], called Accel-Brake Control (ABC) which consists of a signaling mechanism deployed at the base station of cellular networks and a sender side congestion control algorithm that is able to very rapidly (in one RTT) respond to changing network conditions by significantly increasing or decreasing the sending rate. This solution targets a very similar problem space as ours, however, the design is completely different which has a significant impact on deployability and TCP fairness.

III. Multi-Domain Congestion Control: Design Principles

In designing the congestion control algorithm, we start from the learnings in [16], where migration scenarios with Explicit Rate Notification (ERN) capable routers (i.e., routers capable to inform the server of the optimal sending rate) are investigated. The paper assumes a heterogeneous network environment with some routers supporting ERN but others 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 conservative approach may improve the performance while keeping fairness between competing bottleneck flows.

We assume that the flows traverse two domains with significantly different characteristics. In Domain 1 (the "Internet" domain) the congestion control algorithm is responsible for providing fairness between the competing flows. This domain is also characterized by low RTT and packet loss is assumed to be a congestion signal. In Domain 2 (the "RAN" domain) however, resource sharing is governed by lower layers of the protocol stack, so providing fairness is not a necessary function of the CC algorithm. Fluctuations in available bandwidth are also common in this domain, coupled with non-congestion losses or jitter.

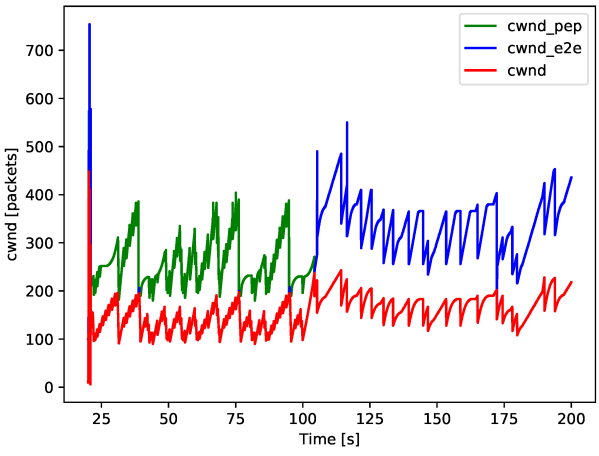

Fig. 2. *cwnd* (red) is the effective congestion window, which is always the minimum of the two components, *cwnd_pep* (green) and *cwnd_e2e* (blue). After 100 seconds, bandwidth increases in the RAN domain and the Internet link becomes the bottleneck. Before this, the effective congestion window is the scalable component, and after the bottleneck link is no longer in the RAN domain, it switches to CUBIC behavior.

By utilizing the additional information from the network (provided by our Lightweight PEP), we can design an algorithm that takes the two domains and their different characteristics into account. This new algorithm needs to be fair with other TCP flows if the bottleneck is in Domain 1, however, it needs to be very aggressive if the bottleneck is in Domain 2 to be able to grab additional bandwidth quickly and be resilient against non-congestion losses.

The proposed algorithm is the following. Two congestion window components are maintained in the server:

- One for Domain 1 (*cwnd_pep*), based on the PEP acknowledgements, which provides a TCP-fair behavior. We chose the CUBIC [4] algorithm to which we feed the PEP acknowledgements. CUBIC is the default CC algorithm in the Linux kernel and it has demonstrated the ability to provide good intra-protocol and inter-protocol fairness.
- One end-to-end (*cwnd_e2e*), based on the client acknowledgements, which assumes that the congestion happens in Domain 2 (note that packet losses in Domain 1 will be visible through PEP-NACKs too, thus impacting the window *cwnd_pep* and triggering congestion control for the Domain 1 losses). Based on the assumption that Domain 2 controls the resource sharing between the flows (and not the CC algorithm), we chose Scalable TCP [19] as the E2E component of our algorithm. Thus, the client acknowledgements are clocking very aggressive, MIMD (Multiplicative Increase, Multiplicative Decrease) functions.

The server uses the minimum of the two congestion window components, $\min(cwnd\_pep, cwnd\_e2e)$, to control the packets in flight i.e., the number of bytes sent but not yet acknowledged by the client. Fig. 2 shows an example of how the effective congestion window is derived from the two components. Similarly to the ERN case investigated in [16], 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. We evaluate this assumption in Section V.

In order to make this new algorithm effective, several minor additional mechanisms are needed which are described in Section IV.

IV. Implementation

A. Lightweight PEP

We implemented the Lightweight PEP functionality by modifying the *PointToPointNetDevice* object in ns-3. As a result, any router can behave as a Lightweight PEP if a correspondent attribute is set. The Lightweight PEP identifies flows by the source IP address and maintains states for each flow passing

through the proxy. Based on the TCP segments received from the sender side, the Lightweight PEP can either send a PEP-ACK or a PEP-NACK to the sender. The packets from the proxy are sent in-band: source- and destination IP fields are set to match the sender and receiver of the TCP session. They contain the information in their payloads and have the Experimental TCP option set in their header to identify them, which we implemented in ns-3 based on [20].

Note that the use of TCP options for signaling can be changed to other, more sophisticated in-band signaling methods like the one described in [21].

B. Multi-Domain Congestion Control

In our Multi-Domain Congestion Control mechanism, the CCA at the sender has two components: one clocked by the E2E acknowledgements from the receiver, and one by the acknowledgements received from the PEP. We feed the E2E acknowledgements to a Scalable TCP - and the PEP-acknowledgements to a CUBIC algorithm. We implemented our algorithm in the Linux kernel[1], by a pluggable CC module that combines the CUBIC and Scalable TCP components, and also some other modifications which are described below.

Fig. 3 shows a high level overview of the implementation. We maintain two different congestion window components (*cwnd_pep* and *cwnd_e2e*) and the effective congestion window is always set to be the minimum of the two. When a PEP-ACK or PEP-NACK is identified based on the Experimental option in the header, we process the acknowledgement information in their payload, call the corresponding CUBIC increase or decrease functions on *cwnd_pep* and discard the segment. These CUBIC increase and decrease functions are the same as in the original CUBIC implementation.

We added a multi-domain window validation mechanism (a multi-domain version of [22]) which ensures that the larger cwnd component can be at most two times larger than the other component. This serves a similar purpose as the original window validation mechanism of TCP (with the same factor of two in the slow start phase) and it also prevents sudden bandwidth changes affecting the fairness in the Internet domain.

Another additional mechanism implemented in our solution is a multi-domain loss-filtering that disables the decrease of *cwnd_e2e* if that decrease would be triggered by a lost segment in Domain 1, when the Lightweight PEP already sent us a NACK of that segment and we have already decreased *cwnd_pep* based on that information. This further separates the congestion control for the domains and it means that when a segment is lost, we only take action in the domain where the loss happened. A possible future enhancement could be to extend this loss-filtering mechanism to enable a different response if potential non-congestion losses are detected in the RAN domain.

When the sender receives a PEP-NACK after a loss happens in Domain 1, retransmission is immediately triggered.

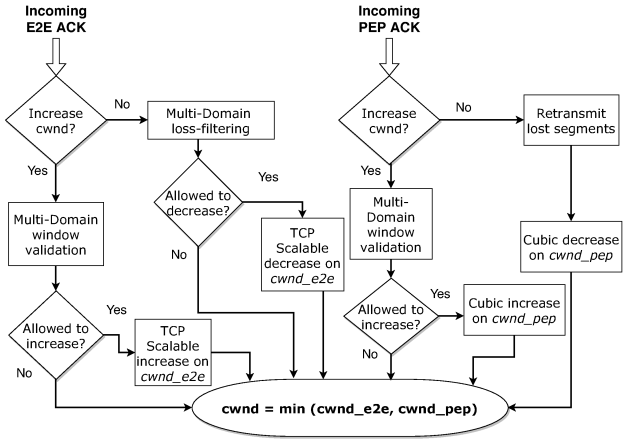

Fig. 3. High level flowchart of the Multi-Domain Congestion Control implemented in the kernel

We do not disrupt the E2E retransmission mechanisms with the multi-domain loss-filtering (only the congestion window decrease is impacted), but if we did not implement this very early retransmission triggered by PEP-NACKs, the *cwnd_pep* component and ultimately the performance would be seriously compromised due to the jitter that the E2E retransmission would introduce to the Domain 1 control loop.

The initial windows for the two congestion control components are set to different values, based on the following observation. Since the RTT to the PEP is always smaller than the end-to-end RTT, it is apparent that during slow-start the *cwnd_pep* will increase faster than *cwnd_e2e*, thus if the two windows started from the same value, it would always be the *cwnd_e2e* that determines 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 the end-to-end RTT rather than the RTT in Domain 1. This would result in larger download times for small content transfer if there is no congestion, because of this slower window increase. To avoid this limitation we use a 10 times higher initial value for *cwnd_e2e* in our implementation.

V. Simulation Setup and Performance Evaluation

A. Environment

The main components of our simulation environment were the ns-3 open source simulator [23] (extended with our Lightweight PEP implementation), the Direct Code Execution (DCE) cradle [24] [2], and the NUSE [25] userspace network stack (Linux kernel version 4.7.0) with our modified kernel code. The DCE cradle enables us to use the real Linux kernel network stack on the endpoints instead of the ns-3 models, which results in more realistic behavior.

We carried out the performance evaluation of our proposed solution using a dumbbell-topology as shown in Fig. 4. The Lightweight PEP is installed on the border of the Internet - and RAN domains. We have designed three scenarios to study

[1]Version 4.7.0

[2]ns-3.26 and DCE version 1.9

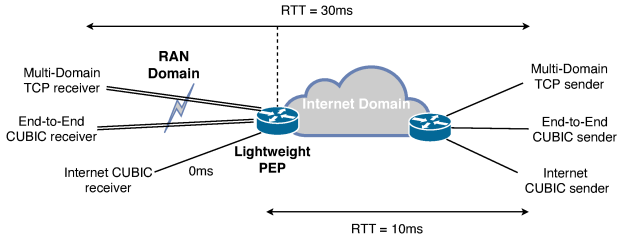

Fig. 4. Simulation tolopology

and illustrate the capabilities of our algorithm. The end-to-end RTT is 30ms in each scenario, while the RTT in the Internet domain is 10ms. The Linux default pfifo_fast queue is used to configure the bottleneck buffers, however we also used CoDel [26] AQMs in the second and third scenarios. The (initial) bottleneck bandwidth is 35Mbps in each scenario (either in the RAN - or the Internet domain).

B. Robustness in the presence of packet loss in the RAN domain

This scenario demonstrates how our Multi-Domain Congestion Control handles the presence of random loss in the RAN domain. The bottleneck link is in the radio access network, not in the Internet domain so traditional TCP fairness is not required, the resource sharing is assumed to be provided by the lower layers. The effective cwnd in our algorithm will be the output of the more aggressive (MIMD) Scalable TCP, which improves the resilience against packet losses. Fig. 5 shows that the Multi-Domain Congestion Control is indeed able to maintain a higher congestion window.

Table I shows the gain that the Multi-Domain CC is able to achieve compared to the E2E CUBIC in lossy RAN domains. Each steady-state throughput value is calculated as an average

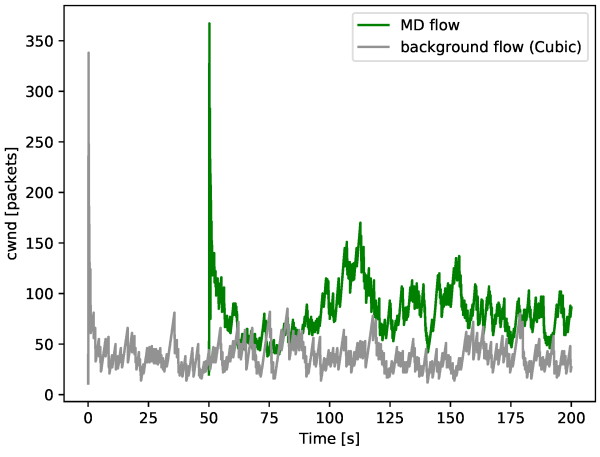

Fig. 5. The Multi-Domain Congestion Control (flow starting at 50s) is able to maintain a higher congestion window in the presence of random packet losses (10^{-3}) in the RAN domain due to the more aggressive Scalable TCP component.

TABLE I
AVERAGE STEADY-STATE THROUGHPUT OF DIFFERENT FLOWS

Loss rate	Throughput [Mbps]		Gain
	Multi-Domain CC	CUBIC	
10^{-4}	35.00 ± 0	34.34 ± 0.3	1.92%
$5 \cdot 10^{-4}$	34.94 ± 0.05	20.99 ± 1.47	66.44%
10^{-3}	28.48 ± 0.87	13.64 ± 0.67	108.75%
$5 \cdot 10^{-3}$	11.16 ± 0.39	6.11 ± 0.11	82.61%

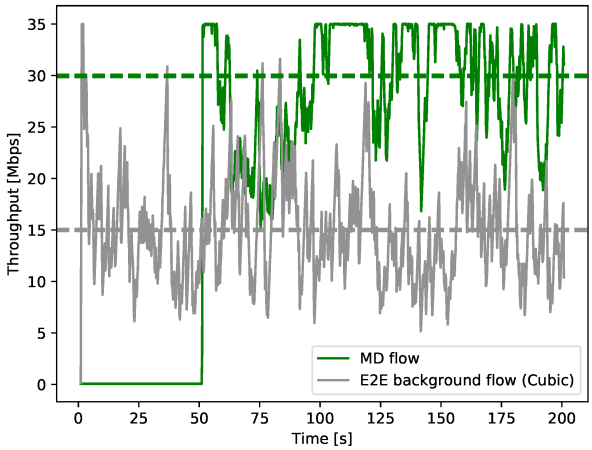

Fig. 6. If traditional TCP fairness is not required (the bottleneck is in the RAN domain), the Multi-Domain Congestion Control is able to achieve significantly higher throughput in lossy environments than the default E2E CUBIC algorithm. Dashed horizontal lines correspond to the average steady-state throughput.

of 10 independent runs (an example run is depicted in Fig. 6). It can be seen that in the loss range between 10^{-4} and 10^{-3}, the gain increases from 1.92% to 108.75%. If the loss rate is even higher, the gain slightly decreases, however, the Multi-Domain CC still outperforms CUBIC by 82.61%.

C. Swift reaction to increased capacity

The second scenario shows how fast our method is able to follow bandwidth change. The bottleneck capacity in the RAN domain suddenly increases, simulating a NLOS-LOS transition in 5G mmWave networks. Random losses are not present here, so the resulting difference in behavior is independent of the previously shown robustness. We used CoDel AQMs in the RAN bottleneck queue so the link will not be underbuffered after the sudden change in bandwidth. Fig. 7 shows that the flow using Multi-Domain CC can utilize the increased link capacity significantly faster. It takes nearly 17 seconds for the throughput of the E2E Cubic flow to reach the new capacity, while the Multi-Domain flow begins to fully utilize the 350Mbps bottleneck 10 seconds earlier. The Multi-Domain approach provides nearly 60% reduction in the time needed to fully utilize the RAN bottleneck after a 10-times increase in capacity.

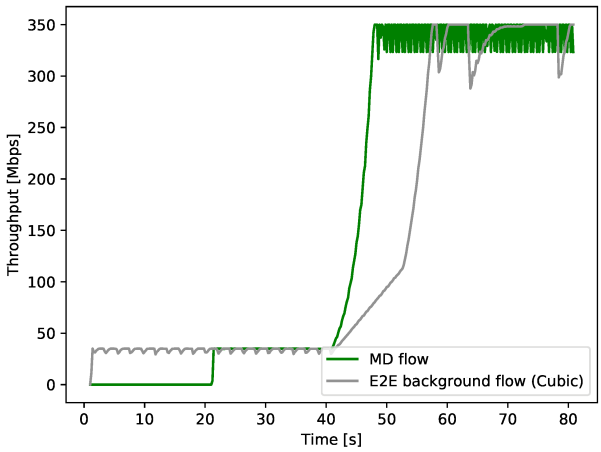

Fig. 7. After 40 seconds, the bottleneck bandwidth in the RAN domain increases 10-times to 350Mbps. The Multi-Domain CC approach provides 10 seconds faster adaptation.

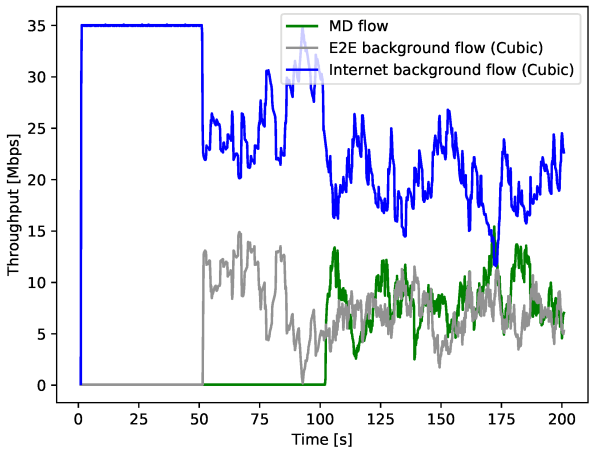

Fig. 9. The flow using Multi-Domain CC starts after 100s and achieves nearly perfect fairness with the CUBIC flow with the same RTT.

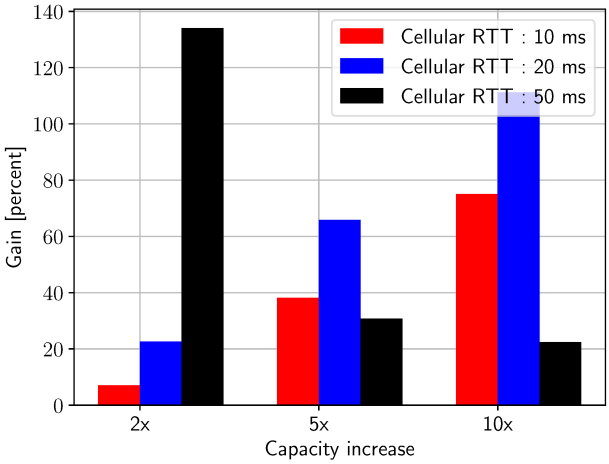

Fig. 8. Gain in average throughput during the transient period of the E2E CUBIC flow after sudden capacity increase

Fig. 8 shows the gain achieved by the Multi-Domain CC in several sub-scenarios. Different cellular RTTs are studied with different capacity increase ratios (2x, 5x, 10x), where the initial bottleneck bandwidth is 35Mbps. With cellular RTTs of 10ms and 20ms, the gain in average throughput during the transient period of the E2E flow increases with both RTT and the capacity increase ratio. This transient period is measured from the change in bandwidth to the time when the E2E flow utilizes at least 95% of the new available bandwidth. For large cellular RTTs (50ms) the gain decreases with higher capacity increase ratios. This is due to the fact that CUBIC can increase its congestion window very aggressively after spending some time in the *max probing* phase. After a 10-times increase in available bandwidth, the Multi-Domain CC delivers a gain over CUBIC between 22.4% and 111.18% in the studied cellular RTT range.

D. Preserving TCP fairness

In addition to the properties described above, we also aim to keep the fairness in the Internet domain. The third scenario consists of three concurrent TCP flows. There is an additional *Internet background flow* modeling a flow over a wired network, which only shares Internet bottleneck and RTT with the other flows, and its RTT in the wireless domain is negligible. We use this flow to compare the effect of the Multi-Domain CC on fairness to the basic RTT-unfairness of CUBIC.

Table II contains the average throughput of the three flows and the calculated Jain's index in four different sub-scenarios:

- 3-A: all flows use CUBIC CC and the bottleneck queues are pfifo_fast.
- 3-B: the foreground flow uses Multi-Domain CC, bottleneck queues are pfifo_fast.
- 3-C: all flows use CUBIC CC, the bottleneck queues are CoDel AQMs.
- 3-D: the foreground flow uses Multi-Domain CC, the bottleneck queues are CoDel AQMs.

Scenarios 3-A and 3-C are reference scenarios with only CUBIC flows. 3-B and 3-D are the same as 3-A and 3-C, respectively, however, the foreground flows are using Multi-Domain Congestion Control. An example run for the 3-D sub-scenario is shown in Fig. 9.

Jain's fairness index [27] characterizes the fairness between n flows and can be obtained from the following formula:

$$J(x_1, x_2, ..., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

The table shows that changing the foreground flow's congestion control from CUBIC to our Multi-Domain algorithm results in negligible change in fairness compared to the basic RTT-unfairness of CUBIC. Each value is calculated as an

TABLE II
FAIRNESS CHARACTERISTICS OF DIFFERENT FLOWS

Scenario	Throughput [Mbps]			Jain's index
	Foreground	Background	Internet background	
3-A	7.62 ± 0.4	7.12 ± 0.63	20.26 ± 0.64	0.79 ± 0.03
3-B	7.82 ± 0.69	7.21 ± 0.79	19.97 ± 0.75	0.8 ± 0.03
3-C	8.53 ± 0.34	8.64 ± 0.32	17.83 ± 0.43	0.88 ± 0.02
3-D	9.02 ± 0.45	8.37 ± 0.31	18.07 ± 0.57	0.88 ± 0.02

average of 10 independent runs with the Multi-Domain flow starting at different times.

VI. CONCLUSION

We presented a novel framework of a Multi-Domain Congestion Control that is based on a Lightweight Performance Enhancing Proxy (PEP), which can provide performance improvements in recent heterogeneous networks involving fixed domains and wireless 3G-4G-5G domains. Our proposed PEP concept is lightweight, provides safe-to-ignore explicit feedback and does not contribute to ossification. In addition, using the PEP does not require any client modification.

The concept was implemented in the Linux kernel and the Lightweight PEP in ns-3 and we carried out a comprehensive performance evaluation of our proposed framework.

First, we demonstrated that our approach can achieve a robust behavior in the presence of wireless loss compared to the currently used CUBIC TCP. Second, we illustrated that our approach provides a faster adaptation in the wireless domain compared to the currently used CUBIC TCP. Third, we also showed that the performance improvements in the wireless domain have no negative impact on the traffic dynamics in the wired domain, i.e., it is able to keep the TCP fairness in the wired domain.

The above results show that the Lightweight PEP concept could be a powerful tool to enhance the TCP performance in current and future cellular access networks. Our future work therefore focuses on understanding the deployment aspects and refining the performance evaluation of our framework in different case studies involving realistic wireless domain behavior.

REFERENCES

[1] M. Zhang, M. Mezzavilla, R. Ford, S. Rangan, S. Panwar, E. Mellios, D. Kong, A. Nix, and M. Zorzi, "Transport layer performance in 5g mmwave cellular," in *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*. IEEE, 2016, pp. 730–735.

[2] 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.

[3] A. Mihály, S. Nádas, S. Molnár, Z. Krämer, R. Skog, and M. Ihlar, "Supporting multi-domain congestion control by a lightweight pep," in *Internet of Things, Embedded Systems and Communications (IINTEC), 2017 International Conference on*. IEEE, 2017, pp. 105–110.

[4] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[5] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *Queue*, vol. 14, no. 5, p. 50, 2016.

[6] ——, "Bbr congestion control: An update," in *Presentation in ICCRG at IETF 98th meeting*, 2017.

[7] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of bbr congestion control," in *Network Protocols (ICNP), 2017 IEEE 25th International Conference on*. IEEE, 2017, pp. 1–10.

[8] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "Pcc: Re-architecting congestion control for consistent high performance." in *NSDI*, vol. 1, no. 2.3, 2015, p. 2.

[9] M. Kühlewind, "Tcp siad: Congestion control supporting high speed and low latency," *arXiv preprint arXiv:1612.07947*, 2016.

[10] J. Border, J. Griner, G. Montenegro, Z. Shelby, and M. Kojo, "Performance enhancing proxies intended to mitigate link-related degradations," 2001.

[11] G. Papastergiou, G. Fairhurst, D. Ros, A. Brunstrom, K.-J. Grinnemo, P. Hurtig, N. Khademi, M. Tüxen, M. Welzl, D. Damjanovic *et al.*, "De-ossifying the internet transport layer: A survey and future perspectives," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 619–639, 2017.

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

[13] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving tcp/ip performance over wireless networks," in *Proceedings of the 1st annual international conference on Mobile computing and networking*. ACM, 1995, pp. 2–11.

[14] 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.

[15] T. T. Thai, D. M. L. Pacheco, E. Lochin, and F. Arnal, "Satern: a pep-less solution for satellite communications," in *Communications (ICC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–5.

[16] D. L. Pacheco, T. T. Thai, E. Lochin, and F. Arnal, "Towards an incremental deployment of ern protocols: a proposal for an e2e-ern hybrid protocol," in *PFLDNet*, 2010.

[17] M. Polese, M. Mezzavilla, M. Zhang, J. Zhu, S. Rangan, S. Panwar, and M. Zorzi, "milliproxy: a tcp proxy architecture for 5g mmwave cellular systems," *arXiv preprint arXiv:1712.02700*, 2017.

[18] P. Goyal, M. Alizadeh, and H. Balakrishnan, "Rethinking congestion control for cellular networks," in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. ACM, 2017, pp. 29–35.

[19] T. Kelly, "Scalable tcp: Improving performance in highspeed wide area networks," *ACM SIGCOMM computer communication Review*, vol. 33, no. 2, pp. 83–91, 2003.

[20] D. T. Narten, "Assigning Experimental and Testing Numbers Considered Useful," RFC 3692, Jan. 2004. [Online]. Available: https://rfc-editor.org/rfc/rfc3692.txt

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

[22] M. Handley, J. Padhye, and S. Floyd, "Tcp congestion window validation," Tech. Rep., 2000.

[23] "The ns3 simulator," Available: http://www.nsnam.org/, 2016.

[24] H. Tazaki, F. Uarbani, E. Mancini, M. Lacage, D. Camara, T. Turletti, and W. Dabbous, "Direct code execution: Revisiting library os architecture for reproducible network experiments," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 217–228.

[25] H. Tazaki, R. Nakamura, and Y. Sekiya, "Library operating system with mainline linux network stack," *Proceedings of netdev*, 2015.

[26] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled delay active queue management," Tech. Rep., 2018.

[27] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance," *Computer Networks and ISDN systems*, vol. 1, pp. 1–14, 1989.