Toward Optimal Performance with Network Assisted TCP at Mobile Edge

Soheil Abbasloo[1], Yang Xu[2], H. Jonathon Chao[1], Hang Shi[3], Ulas C. Kozat[3], Yinghua Ye[3]

New York University[1], Fudan University[2], Futurewei Technologies[3]

Abstract

In contrast to the classic fashion for designing distributed end-to-end (e2e) TCP schemes for cellular networks (CN), we explore another design space by having the CN assist the task of the transport control. We show that in the emerging cellular architectures such as mobile/multi-access edge computing (MEC), where the servers are located close to the radio access network (RAN), significant improvements can be achieved by leveraging the nature of the logically centralized network measurements at the RAN and passing information such as its minimum e2e delay and access link capacity to each server. Particularly, a Network Assistance module (located at the mobile edge) will pair up with wireless scheduler to provide feedback information to each server and facilitate the task of congestion control. To that end, we present two *Network Assisted* schemes called NATCP (a clean-slate design replacing TCP at end-hosts) and NACubic (a backward compatible design requiring no change for TCP at end-hosts). Our preliminary evaluations using real cellular traces show that both schemes dramatically outperform existing schemes both in single-flow and multi-flow scenarios.

1 Introduction

Various studies show that the general purpose TCP performs poorly in CNs [8, 17, 19, 25–27] because of the key fundamental issue of the *generalization*. Although the goal of having one general purpose TCP that can work in various networks is attractive, this goal made existing TCP a "Jack of all trades, master of none". A general purpose TCP requires to generalize the network and the reasons for congestion in the network. However, the generalization comes with the cost in performance. For instance, TCP Cubic (today's default TCP in most of the platforms) considers the loss of packets as a general indication of congestion. So, in the CN where stochastic packet losses exist, TCP Cubic misinterprets the stochastic packet losses as congestion which leads to its performance degradation. The generalization issue motivates a lot of recent congestion/flow control proposals to follow the *domain-specific* design philosophy in which the design is limited to a specific network and leverages the characteristics of that network to boost the performance (e.g. [6, 8, 26, 27] in CNs and [7, 9, 24] in data center networks).

On the other hand, although current distributed domain-specific TCPs can perform better than general TCPs in CNs, they are still suboptimal solutions. The reason is that, as proved by J. Jaffe [18], no *distributed* congestion control can converge to the operation point in which both the minimum delay and maximum throughput are achieved (Klein-rock's optimal point [14]). This result comes from the fact that by only using distributed e2e measurements (as in TCP), it will be unclear whether an increase in e2e delay measurements is due to another competing flow, a route change, or the client delaying the acknowledgment. The problem of distributed e2e measurements of TCP becomes even worse in CNs when downlink/uplink wireless scheduling delay, self-inflicted queuing delay, and channel capacity variations are considered.

Considering these shortcomings and the fact that new network latency and throughput requirements of the emerging applications such as real-time online gaming, automated vehicles, and virtual reality push the performance bar of TCP higher than the currently acceptable ones, we seek to boost the performance of TCP toward the optimal point in Mobile-Edge environment.

In CNs, there is usually rich information about each user equipment (UE) in a fine-grain timescale. For instance, base stations are aware of the history of throughput, per-UE queue occupancy, and the quality of channel of the UEs connected to them [22]. That motivated us to address the problem of distributed e2e measurements of TCP by investigating the benefits of leveraging the logically centralized network measurements and information gathering in CNs to assist the transport control at servers, especially when they are located at the edge of the network.

To that end, we introduce a Network Assistance (NetAssist) entity at the mobile edge which periodically sends the

bottleneck link's (BL) bandwidth (Bw) and delay information to end-hosts in an out-of-band (OoB) manner. As a proof of concept, we present two schemes called NATCP and NACubic. NATCP is a clean-slate design replacing traditional TCP at end-hosts using feedback information from NetAssist and calculate congestion window (Cwnd) and pacing rate. However, NACubic is a backward compatible design requiring no change in traditional TCP at end-hosts. It simply works on top of the current TCP and only caps Cwnd and pacing rate to the values calculated using feedback information. Using real cellular traces, we evaluated NATCP and NACubic and compared them with various state-of-the-art TCP schemes (detailed in section 3). We showed that both NATCP and NACubic significantly outperform other schemes and achieve higher power (defined as $\frac{throughput}{delay}$) and lower 95th percentile delays. For instance, NATCP and NACubic achieve more than $2.7\times$ higher power compared to ABC [15] (the most recent ECN-based approach in CNs).

2 Design

The golden three goals of any congestion control are to achieve: 1) maximum throughput, 2) minimum delay, and 3) fairness among flows. However, none of the existing TCP proposals in the literature achieves all these 3 goals simultaneously. For instance, most of the TCP proposals use the AIMD (additive increase multiplicative decrease) [12] algorithm to increase their fairness performance among multiple flows. However, this affects the two other performance metrics by slowing down the Cwnd growth in the case of a sudden increase in available bandwidth and harsh Cwnd reduction in the case of a sudden decrease in available bandwidth. Therefore, to achieve all 3 goals, we decouple the problem of achieving fairness from the problem of achieving maximum bandwidth and minimum delay.

2.1 Fairness and the Role of UE

We divide the fairness problem into two sub-problems: 1) fairness among different UEs (Inter-Fairness) and 2) fairness among the flows of one UE (Intra-Fairness).

In contrast to wired networks, there are per-user queues and there is a scheduler in CNs (Figure 1). The scheduler is responsible for scheduling downlink/uplink per-user queues and bringing the fairness among different UEs at base transceiver station (BTS) [22]. In other words, at the bottleneck switch in a wired scenario, an aggressive TCP can fill up the buffer and take the bandwidth from other TCPs; however, in cellular scenario, scheduler at BTS schedules the access of the aggressive TCP to the wireless channel so that all flows can have a fair amount of wireless link's bandwidth regardless of the TCP protocol that they use. Therefore, the inter-fairness problem has already been resolved by the BTS' scheduler. However, there is still the problem of intra-fairness.
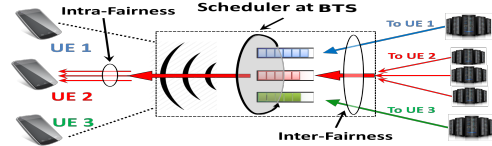

Figure 1: Scheduler and per-UE queues at BTS

In a general network, the number of competing flows at the bottleneck queue is not known. Therefore, in a classical distributed congestion control approach, each end-host tries to independently adjust its sending rate using techniques such as AIMD to give room to other likely competitors in the network. However, in CNs, due to the existence of per-UE queues, the number of competing flows at each queue at BTS is known by each UE (it is equal to the flows destined to the UE). Considering that, there are two approaches to resolve the Intra-fairness in CNs: 1) Since TCP is a closed-loop algorithm and a sender always requires to receive acknowledgments from the receiver to send more packets, the receiver (UE) who knows the competing flows can simply regulate the sending rates of the senders by regulating the sending rates of their corresponding Ack packets. 2) UE can directly inform servers about the competing flows (destined to that UE) and help them to adjust their sending rates accordingly by sending fairness information feedback to them[1].

In our design, each UE reports the fairness information to the servers using the Ack packets (Step # 1 in Figure 2) either indirectly (first mentioned method) or directly (second described method).

2.2 Towards the Optimal Point

The notion of having a logically centralized entity providing guidance/control to/over UEs is already part of the CN. For instance, BTS dynamically schedules the transmission/reception of packets to/from UEs and periodically sends control messages to UEs indicating their scheduled time slots. Intuitively, NATCP exploits the same idea but this time for providing assistance to the servers in CN aiming to operate close to Kleinrock's optimal point [14].

2.2.1 NetAssist

Placing customary servers inside the mobile operator's networks reduces the intrinsic e2e delay. However, due to the wireless nature of cellular access links which causes multiple order of magnitude fast capacity fluctuations, cellular access links (known as last-mile) remain the main BL in CNs. To estimate the BL's bandwidth and the minimum delay, we propose that the end-hosts get help from the network itself. In

---

[1]In the simplest case, fairness information is the number of competing flows at UE, while a much richer feedback can be constructed considering a more sophisticated definition of fairness.
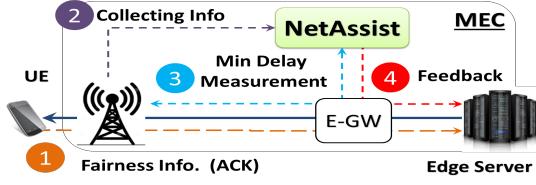
Figure 2: The big picture of NATCP/NACubic structure

particular, we introduce a new entity called Network Assistance (NetAssist) in the network which periodically collects information regarding the BL's bandwidth and the minimum delay for each UE (Steps #2 and #3 in Figure 2), and sends digested feedback (Step #4 in Figure 2) to the servers, that have registered to get congestion assistance service from the CN, in OoB manner. This feedback will be used by servers to set Cwnd and pacing rate and send data accordingly.

BL's bandwidth (BL_{Bw})**:** The BL_{Bw} (which is already available at BTS in a fine-grained timescale) will be averaged at each feedback period and sent by NetAssist. Our sensitivity analysis indicates that a period of 50*ms* for sending feedback from NetAssist to the server gives reasonable results. With feedback periods much larger than 50ms, the server misses channel fluctuations and cause NATCP/NACubic to react too slowly to them. On the other hand, feedback periods much smaller than 50ms may not give enough time for the server to react against the highly dynamic variation of cellular access link capacity.

minRTT: minRTT is the minimum RTT that a packet in the network can experience excluding any waiting time in the queues and sender/receiver response time delays. To measure minRTT, we segment it into 3 parts: 1) The minimum delay of the network (from edge gateway (E-GW) to BTS and vice versa), 2) The transmission delay and scheduling delay of packets at BTS in the downlink direction, and 3) The transmission delay and scheduling delay of packets at BTS in upstream direction[2]. To measure the first part, a simple way is to use higher priority probing packets that are periodically sent to the BTS from NetAssist (or a dedicated monitoring module). Due to their higher priorities, probing packets will not experience any queuing delay and will show the delay excluding queuing delays at the network. Also, delay measurements can be done by using more advanced approaches provided by programmable devices such as attaching and stripping meta-data information to data packets (e.g., [1]) (This part does not need to be measured per server, because the network's minimum delay is roughly the same for all of them).

Since the sending/receiving rate of packets to/from the UE

is controlled by the wireless scheduler, the depletion/reception rate of the UE at BTS in a time interval declares the sum of averaged scheduling delay and averaged transmission delay. Therefore, to calculate parts 2 and 3, NetAssits uses averaged sending/receiving rates to/from UEs at BTS over the feedback time period. NetAssist calculates minRTT by adding these 3 parts and sends it as part of feedback information to the servers.

2.2.2 Logic at the Edge Server

Getting feedback information from the NetAssist and fairness information from the UE, the server will calculate Cwnd and pacing rate for flows. Here, we propose two solutions. One is a clean-slate design which completely replaces TCP at servers (called NATCP) and the other one is a backward compatible design which without modifying the existing TCP at server (e.g., Cubic [16]), applies received feedback from NetAssist (So, without loss of generality, we assume the default TCP is Cubic, and we call the second scheme NACubic).

NATCP: In NATCP, servers use BL_{Bw} to set the pacing rate of the flow's outgoing packets (conceptually pushing the BL to the server). In addition, servers calculate Cwnd using Equation 1 where minRTT and BL_{Bw} are the minimum delay and BL bandwidth feedback received from NetAssist, respectively and β is the fairness coefficient (received from the UE). At source/destination packets will usually be queued between different network layers at OS. Therefore, Cwnd value in the TCP layer does not necessarily equal to the number of in-flight packets. In Equation 1, α represents a scaling factor to compensate for the difference of Cwnd with the real in-flight packets. We set α to 2 throughout this paper.

$$Cwnd = \alpha \times \frac{1}{\beta} \times minRTT \times BL_{Bw} \qquad (1)$$

NACubic: In NACubic, the server does not directly use feedback information to calculate the Cwnd. Instead, it simply caps the Cwnd calculated by Cubic (or other TCP variants) to the MaxCwnd calculated by Equation 1. In this way, there is no need to modify the TCP at the server. Similar to NATCP, NACubic sets pacing rate to BL_{Bw}[3].

2.3 Why it works

Is Centralized Delay Feedback Really Necessary? To shed light on why we need both centralized BW and delay measurements, we use one trace from prior work [26] (namely Verizon-EVDO-downlink) and send traffic from the server to UE following instructions detailed in section 3. In the CN where BL BW is highly dynamic, current distributed e2e estimations of minRTT at end-hosts might not be likely valid for the next following RTTs. This issue escalates when the

---

[2]due to the server proximity in MEC architecture, the latency from the server to E-GW accounts for a small portion of the overall latency, so it can be ignored. Also, we ignore the intrinsic wireless signal propagation delay between BTS and UE which is usually very small

[3]Setting per-flow pacing rate functionality is currently part of the TCP stack in Linux kernel 4.13.
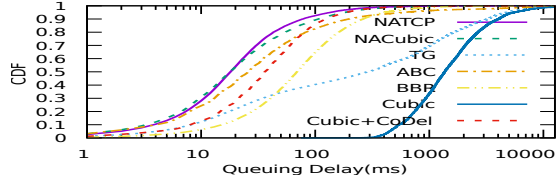
Figure 3: CDF of queuing delay over Verizon-EVDO.downlink trace

intrinsic e2e delay (excluding queuing delay) of the CN is very low. For instance, consider a network with the intrinsic low e2e delay of 10ms (as for the network of this section's experiment). When link capacity is 1.2*Mbps* at arbitrary time *t*, the transmission delay for HOL (head-of-line) packet *p* of size 1500*B* becomes 10*ms* alone. This also makes 10*ms* increase in the sojourn time (waiting time) of all packets queued behind packet *p*. So, for networks similar to MEC-based CNs (with intrinsic low e2e delay and highly variable BL (last-mile)), the minimum RTT will be significantly impacted by the queuing delay (including transmission time and sojourn time, either because of wireless scheduling delay or variation of the transmission times) in small time scales. Therefore, the distributed e2e delay estimations that require at least a few RTTs will not provide a good estimation of the real minimum RTT of the current time.

Fig.3 shows the impact of this phenomena. When link capacity is highly dynamic (as in Verizon-EVDO trace), TG scheme, which uses only BW feedback and counts on RTT measurements at the server, performs very poorly. Also, BBR which estimates minRTT in a distributed manner performs poorly, especially in high percentile regions. ABR uses ECN bits set by BTS in the downlink direction and piggybacked by Acks to the server to adjust the rate quickly. However, due to the high variations of the queuing delay, even this ECN-based scheme, as Fig.3 illustrates, cannot perform very well. This clearly shows the importance and advantage of using centralized delay and centralized BW measurements/feedback in our design.

OoB Signaling: Table 1 compares the improvements of Power$_{95}$ (defined as throughput /95th %tile queuing delay) gained by using different feedback info (only BL_{Bw} vs. BL_{Bw} and minRTT(NATCP)) and different way of sending that (OoB vs. IB (In-Band)) for the setup detailed in section 3 (using ATT-LTE-2016 traces [20])[4]. Generally, OoB signal-

Table 1: Normalized P_{95} for various schemes

Scheme	Bw (IB)	Bw (OoB)	NATCP (IB)	NATCP (OoB)
P_{95}	1×	3.47×	7.32×	9.31×

ing performs better than IB signaling, because feedback will not be delayed due to downlink/uplink scheduling and queuing delays. Hence, NetAssist periodically sends feedback to

servers in an OoB manner as opposed to IB signaling (e.g., as in [23]), though it might add small signaling overhead (check section 2.4).

2.4 Discussion

Signaling Overhead: NetAssist only sends two values as feedback, so let us assume the feedback packet size to be 64Bytes. Hence, the data rate for the OoB feedback per UE will be $\frac{64B}{50ms} = 10kbps$. Even, when the number of users is 1,000 (per NetAssist), there will only be a total of 10Mbps signaling overhead. This is quite acceptable considering current LTE networks where each UE can reach the rates of tens of Mbps and the fact that next-generation 5G CNs aim to provide hundreds of Mbps per UE [4] [5].

Is NATCP/NACubic scalable? NetAssist module pairs up with BTS to gather information without altering or interfering with the scheduler's operations. The NetAssist pairing up with a certain BTS works independent of the NetAssist module paring up with another BTS, because there is no need to pass information from one NetAssist to another one. For instance, in the case of handover, when a user disconnects from one BTS and connects to a new BTS, no context-switching happens. The scheduler in the new BTS does not get throughput history of the new user from previous BTS (to schedule its transmission).[5] That is the same for NetAssist. In other words, there is no need for having either a global view or previous states of the whole CN in NetAssist. These properties render NATCP/NACubic a highly scalable design.

Deployment of NetAssist: The traditional issue with the approaches requiring changes in the network is the feasibility and the high costs of those changes for the network operators. However, the customizable nature of recent software-based trends such as function virtualization [2] makes the deployment of new services in the network feasible while having low costs. For instance, NetAssist module can be realized as a virtual function in C-RAN (cloud/centralized RAN) [3] and network operators can use the available pool of servers at MEC to make the virtual NetAssist modules.

What if network feedback does not reach the server? We can use a simple timer concept to monitor loss of feedback information. In other words, if feedback information is not received within a certain time, the timer will timeout and NATCP/NACubic will revert to normal TCP at the sender.

3 Preliminary Evaluations

We have implemented NATCP/NACubic in Linux Kernel 4.13, written proof of concept NetAssist modules in user-space, and compared performances of NATCP and NACubic with existing schemes using Mahimahi [20] (Available at: https://github.com/Soheil-ab/natcp).

---

[4]Here, feedback was generated every 10ms.

[5]This behavior has been confirmed through our private discussions with CN provider engineers
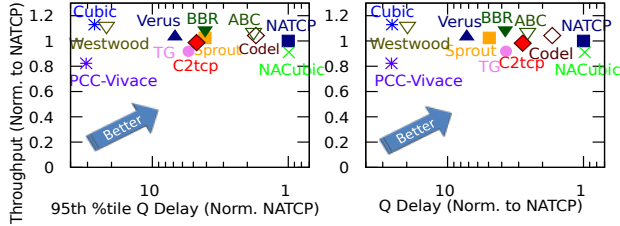
Figure 4: Normalized results to NATCP averaged over all traces

Table 2: Power reduction averaged across all traces

Scheme	P	P_{95}		Scheme	P	P_{95}
NATCP	$1\times$	$1\times$		Cubic+Codel	$1.63\times$	$1.66\times$
NACubic	$1.09\times$	$1.1\times$		Sprout	$4.8\times$	$3.98\times$
ABC	$2.71\times$	$1.83\times$		Verus	$7.01\times$	$6.67\times$
C2TCP	$2.79\times$	$4.83\times$		Cubic	$22.85\times$	$23.85\times$
BBR	$3.46\times$	$3.76\times$		Westwood	$17.68\times$	$19.60\times$
TG	$3.86\times$	$5.65\times$		PCC-Vivace	$28.61\times$	$35.54\times$

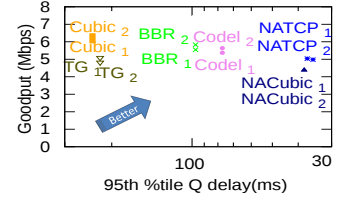| Scheme	Normalized Number of ReTrans.
NACubic	$1\times$
NATCP	$4\times$
Cubic	$20\times$
BBR	$25\times$
Cubic+Codel	$54\times$
TG	$1470\times$

Figure 5: The 95th %tile Q delay, average Goodput, and total number of retransmissions for 2 flows competing at UE

Cellular Traces: We use 16 different cellular traces collected in prior work ([20] and [26]) as the cellular link traces.

Topology and Metrics: We use a server which is connected to a client through Mahimahi emulator. Mahimahi emulates the cellular access link using the given cellular traces. To have an MEC like network, we set the minimum RTT to 10ms. The per-user buffer size at BTS is set to 150KB[6]. The minimum RTT from NetAssist to the server is set to 2ms[7]. We start a flow between a server and a client that lasts for the entire duration of each trace and measure the average and 95th percentile per packet queuing delay, average throughput, power (defined as $\frac{Throughput}{Avg.\ queuing\ delay}$), and power$_{95}$ (defined as $\frac{Throughput}{95th\ \%tile\ queuing\ delay}$).

Schemes Compared: We compare NATCP and NACubic with three e2e TCP designs: Cubic [16], BBR [10], PCC-Vivace [13]; four e2e cellular schemes: C2TCP [8], Verus [27], Sprout [26], and Westwood [11]; one ECN-based scheme: ABC [15], one feedback-based scheme: TG [23]; and one AQM design Cubic+Codel [21]. We tried to tune different schemes to have their (on average) best performance, though our tuning might not be perfect. For the TG scheme, we use the same BW feedback used by NATCP which represents averaged available bandwidth on 50ms time windows.

Overall Results: For each trace, we normalize the performance results of different schemes to the NATCP's performance for that trace. Later, we average all normalized results over all 16 cellular traces from [26] and [20]. Figure 4 shows these overall results. As these graphs illustrate, the distinguishing performance difference among schemes is the delay responses of them. NATCP and NACubic outperform all other schemes in terms of delay performance, while compared to Cubic (which gets the highest throughput among all schemes), they only compromise a bit of throughput (7% and 20% throughput reduction respectively). Also, Table 2 summarizes the average relative Power and Power$_{95}$ improvement for NATCP compared with other schemes, averaged over all CNs traces. All e2e schemes suffer from the distributed mea-

surements issue leading to their poor performance. On the other hand, NATCP beats TG (which uses throughput guidance from the network but still leaves the delay estimations to the end-host) and achieves $3.86\times$ better power. This clearly shows the advantages of having centralized delay measurements over the distributed measurements.

Intra-Fairness: To evaluate intra-fairness of NATCP/NACubic, we send traffic from two servers to one UE and report the goodput and number of packet retransmissions for each flow (we use TMobile-LTE-driving [20] as our access link). Figure 5 shows the results. NATCP and NACubic significantly outperform other schemes in terms of achieving fair throughput without compromising the delay. For instance, compared to Codel, the state-of-the-art delay-based AQM design, both NATCP and NACubic flows achieve $2\times$ lower 95th percentile delay. Also, NACubic and NATCP achieve the first and second minimum number of packet retransmissions among all schemes. For instance, compared to Codel, NACubic reduces packet retransmissions more than $50\times$. The poor performance of TG comes from the fact that it has no mechanism to resolve fairness and it leaves delay measurements to the end-host.

4 Conclusion

We have shown that significant improvements can be achieved by moving from a classical distributed e2e TCP to a Network Assisted TCP in MEC-based CNs. Network Assisted TCP exploits the logically centralized network measurements in CNs where the notion of centralized control and information gathering is a common practice. Based on that, we have presented two schemes called NATCP (a clean-slate design) and NACubic (a backward compatible design). Our preliminary results show that both schemes dramatically outperform prior work in terms of power in both single-flow and multi-flow scenarios.

---

[6]Although the specific buffer size at BTS for each UE is not in public domain, we have tried to select the buffer size by comparing results from emulations with results from the real-world experiments for Cubic.

[7]Due to the server proximity, this delay in practice is a few hundreds of us.

5 Discussion Topics

The controversial points of the paper: The key controversial point of NATCP is its novel design philosophy which contradicts the design of fully distributed e2e TCPs. We show that the key architectural advances such as MEC can be leveraged to greatly improve the performance of TCP by going from a fully distributed e2e design to a network-assisted design at Mobile Edge.

The type of discussion this paper is likely to generate in a workshop format: On the one hand, we hope that this work will generate discussions about reconsidering TCP design approaches among protocol designers and motivates them to exploit network-assisted framework for TCP in CNs. On the other hand, we hope that the encouraging preliminary results presented here show an opportunity for providing such novel and unconventional services to end-hosts and convince people that it is a viable solution to the CN. We believe that more work in this direction will be generated in the future.

What kind of feedback we are looking to receive: Any feedback on the implementation of the proposed network-assisted framework is highly appreciated.

Under what circumstances the whole idea might fall apart: In scenarios where there is no feedback because the NetAssist and end-hosts are disconnected, the NATCP will roll back to the default TCP (without using the feedback). Therefore, circumstances representing scenarios without any feedback will not cause issues. However, if components dealing with the measurements (such as BTS) are providing wrong feedback, that might cause a problem. Those circumstances should be handled by extra safety mechanisms to always be sure that the measurement information provided to the NetAssist is correct.

The open issues the paper does not address: Although the congestion information exposed to servers will not cause security breaches, the communication among various entities need to be secured. So, one open issue that is not addressed in our paper is how to provide security for communications among different entities. In addition, the detailed protocols for communication among various components including NetAssist and end-hosts need to be investigated and standardized.

References

[1] Improving network monitoring and management with programmable data planes. https://p4.org/p4/inband-network-telemetry/.

[2] Operator study on nfv/sdn transformation: Key findings. https://engage2demand.cisco.com/LP=4198?keycode=001483826.

[3] Cloud ran and mec: A perfect pairing. https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp23_MEC_and_CRAN_ed1_FINAL.pdf, 2018.

[4] Docomo achieves world's first 5g wireless data transmission. https://www.nttdocomo.co.jp/english/info/media_center/pr/2018/0509_00.html, 2018.

[5] Huawei reaches 3 gbps in a recent 5g trial in italy. https://www.rcrwireless.com/20180212/5g/huawei-3-gbps-5g-trial-italy-tag23, 2018.

[6] Soheil Abbasloo, Tong Li, Yang Xu, and H Jonathan Chao. Cellular controlled delay tcp (c2tcp). In *2018 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 118–126. IEEE, 2018.

[7] Soheil Abbasloo, Yang Xu, and H. Jonathan Chao. Hyline: a simple and practical flow scheduling for commodity datacenters. In *IFIP Networking Conference (IFIP Networking) and Workshops, 2018*. IEEE, 2018.

[8] Soheil Abbasloo, Yang Xu, and H. Jonathan Chao. C2tcp: A flexible cellular tcp to meet stringent delay requirements. *IEEE Journal on Selected Areas in Communications*, 37(4):918–932, 2019.

[9] Mohammad Alizadeh, Albert Greenberg, David A Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center tcp (dctcp). In *ACM SIGCOMM CCR*, volume 40, pages 63–74. ACM, 2010.

[10] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr: Congestion-based congestion control. *Queue*, 14(5):50, 2016.

[11] Claudio Casetti, Mario Gerla, Saverio Mascolo, Medy Y Sanadidi, and Ren Wang. Tcp westwood: end-to-end congestion control for wired/wireless networks. *Wireless Networks*, 8(5):467–479, 2002.

[12] Dah-Ming Chiu and Raj Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN systems*, 17(1):1–14, 1989.

[13] Mo Dong, Meng Tong, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira. Pcc vivace: Online-learning congestion control. In *NSDI*, 2013.

[14] R. Gail and L. Kleinrock. An invariant property of computer network power. In *Proceedings of the International Conference on Communications*, pages 63.1.1–63.1.5, Denver, Colorado, June 14-18, 1981.

[15] Prateesh Goyal, Mohammad Alizadeh, and Hari Balakrishnan. Rethinking congestion control for cellular networks. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, pages 29–35. ACM, 2017.

[16] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: a new tcp-friendly high-speed tcp variant. *ACM SIGOPS Operating Systems Review*, 42(5):64–74, 2008.

[17] Junxian Huang, Feng Qian, Yihua Guo, Yuanyuan Zhou, Qiang Xu, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. An in-depth study of lte: effect of network protocol and application behavior on performance. In *ACM SIGCOMM CCR*, volume 43. ACM, 2013.

[18] Jeffrey Jaffe. Flow control power is nondecentralizable. *IEEE Transactions on Communications*, 29(9):1301–1306, 1981.

[19] Haiqing Jiang, Yaogong Wang, Kyunghan Lee, and Injong Rhee. Tackling bufferbloat in 3g/4g networks. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 329–342. ACM, 2012.

[20] Ravi Netravali, Anirudh Sivaraman, Keith Winstein, Somak Das, Ameesh Goyal, and Hari Balakrishnan. Mahimahi: A lightweight toolkit for reproducible web measurement. 2014.

[21] Kathleen Nichols and Van Jacobson. Controlling queue delay. *Communications of the ACM*, 55(7):42–50, 2012.

[22] Stefania Sesia, Matthew Baker, and Issam Toufik. *LTE-the UMTS long term evolution: from theory to practice*. John Wiley & Sons, 2011.

[23] Péter Szilágyi and Andreas Terzis. Mobile content delivery optimization based on throughput guidance. In *ICCRG meeting IETF93 (work in progress)*, 2015.

[24] Christo Wilson, Hitesh Ballani, Thomas Karagiannis, and Ant Rowtron. Better never than late: Meeting deadlines in datacenter networks. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 50–61. ACM, 2011.

[25] Keith Winstein and Hari Balakrishnan. Tcp ex machina: Computer-generated congestion control. In *ACM SIGCOMM CCR*, volume 43, pages 123–134. ACM, 2013.

[26] Keith Winstein, Anirudh Sivaraman, Hari Balakrishnan, et al. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *NSDI*, pages 459–471, 2013.

[27] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. Adaptive congestion control for unpredictable cellular networks. In *ACM SIGCOMM CCR*, volume 45, pages 509–522. ACM, 2015.