PBE-CC: Congestion Control via Endpoint-Centric, Physical-Layer Bandwidth Measurements

Yaxiong Xie, Fan Yi, Kyle Jamieson
Department of Computer Science, Princeton University
{yaxiongx,fanyi,kylej}@cs.princeton.edu

ABSTRACT

Cellular networks are becoming ever more sophisticated and overcrowded, imposing the most delay, jitter, and throughput damage to end-to-end network flows in today's internet. We therefore argue for fine-grained mobile endpoint-based wireless measurements to inform a precise congestion control algorithm through a well-defined API to the mobile's cellular physical layer. Our proposed congestion control algorithm is based on Physical-Layer Bandwidth measurements taken at the Endpoint (PBE-CC), and captures the latest 5G New Radio innovations that increase wireless capacity, yet create abrupt rises and falls in available wireless capacity that the PBE-CC sender can react to precisely and rapidly. We implement a proof-of-concept prototype of the PBE measurement module on software-defined radios and the PBE sender and receiver in C. An extensive performance evaluation compares PBE-CC head to head against the cellular-aware and wireless-oblivious congestion control protocols proposed in the research community and in deployment, in mobile and static mobile scenarios, and over busy and idle networks. Results show 6.3% higher average throughput than BBR, while simultaneously reducing 95$^{\text{th}}$ percentile delay by 1.8×.

CCS CONCEPTS

• **Networks** → **Transport protocols**; **Mobile networks**.

KEYWORDS

TCP congestion control, Transport protocols, Cellular network, LTE, Physical control channel, Control information, Capacity estimation

ACM Reference Format:
Yaxiong Xie, Fan Yi, Kyle Jamieson. 2020. PBE-CC: Congestion Control via Endpoint-Centric, Physical-Layer Bandwidth Measurements. In *Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM '20), August 10–14, 2020, Virtual Event, NY, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3387514.3405880

1 INTRODUCTION

Most of today's downlink end-to-end data flows terminate at a cellular last hop to a mobile endpoint, where they encounter the most delay, variations in delay, loss of their constituent packets, and limits on their bandwidth. With the increasingly sophisticated design of today's and tomorrow's cellular networks in mind, this paper argues that it is actually the endpoints that are the entities best positioned to measure the congestion state of an end-to-end connection. We further argue that the physical layer of the mobile endpoint ought to measure the congestion state of the wireless last hop, and feed these very fine-grained measurements up to the transport layer and applications through a well-defined API. This position follows from three challenges that all congestion control algorithms face when they operate in today's wireless networks.

First, wireless is fundamentally a shared medium. This means that when a user's flow commences or finishes, other users associated with the same cell tower experience an abrupt drop or rise in available wireless capacity that takes time to be reflected in the flow of acknowledgements that today's ack-based congestion control protocols send back to the sender [10, 43, 49]. Second, in recent years, to achieve high throughput and low end-to-end queuing delay, senders must now swiftly react to other abrupt capacity changes in the wireless cellular link that neither the sender nor even the cell tower may directly observe. One reason behind this change is that the newest cellular standards, such as LTE-Advance [2] and 5G New Radio [1] aggressively exploit a wireless diversity technique called *carrier aggregation* to increase wireless capacity, in which the cellular network aggregates the capacity from two or more cellular base stations, making that aggregate capacity available to a single user. When the cellular network adds or removes base stations participating in a user's aggregated capacity, the wireless capacity available to each user abruptly changes, accordingly. Wireless-aware congestion control systems centered on a single base station, such as Accel-Brake Control (ABC) [17, 18] require non-trivial extensions to share state across cell sites when carrier aggregation is enabled. Finally, wireless channel quality is inherently highly dynamic, due to, *e.g.*, user mobility, multipath propagation, and interference from neighboring cell towers. These factors change the wireless data rate that a particular user's cellular link supports over a time scale known as the wireless channel *coherence time*, which can be as small as milliseconds in the case of vehicular-speed mobility. In the event of a handover between cell towers, ABC would need to migrate state, which is not considered in its design.

Further, the foregoing factors interact, exacerbating their effect. Due to carrier aggregation, an end-to-end connection experiences fluctuation due to the dynamics of all its aggregated cells, typically fewer (two to four) than can offer a smoothing of capacity due to statistical multiplexing.

While both base station and the mobile endpoint are able to observe these fluctuations, it is only the latter that has fully up-to-date state on the wireless connection to each and every base station the mobile connects with. In the current design of the cellular

SIGCOMM '20, August 10–14, 2020, Virtual Event, NY, USA

ACM ISBN 978-1-4503-7955-7/20/08…$15.00
https://doi.org/10.1145/3387514.3405880

physical layer, however, mobile users decode only their own channel allocation messages, and so cannot track other users' channel usage and thus identify idle wireless capacity.

This paper introduces a new congestion control algorithm based on ***P**hysical-Layer **B**andwidth* measurements, taken at the mobile ***E**ndpoint* (*PBE-CC*). At a high level, PBE-CC is a cross-layer design consisting of two modules. Our first module comprises an end-to-end congestion control algorithm loosely based on TCP BBR [10], but with senders modified to leverage precise congestion control techniques [25] when possible. We harness our end-to-end congestion control to our second module, a wireless physical-layer capacity measurement module for mobile devices. Our key innovation is to enable highly accurate capacity measurements of the wireless cellular link, which track its variations at millisecond-timescale granularity, thus enabling significantly more precise control over senders' rates as they attempt to match their sending rate to the available wireless capacity, should the bottleneck capacity be the wireless link itself. In the event of an increase in wireless capacity, this allows PBE-CC to be rapidly responsive, detecting the amount of newly-emerged idle wireless capacity and prompting the sender to increase its offered rate accordingly. In the event of a decrease in wireless capacity, this allows PBE-CC senders to rapidly quench their sending rate, thus avoiding queuing delays, as our evaluation demonstrates in drill-down experiments (§6).

Our evaluation shows that most of the time, the cellular link is indeed the bottleneck in the end-to-end path, as many congestion control protocols [17, 43, 49] assume. PBE-CC makes the same initial assumption, leveraging the above wireless-aware precise congestion control functionality to more accurately control the sender's pacing, while also taking into account the number of users sharing the wireless link, so that each PBE-CC sender can offer a load that results in an overall-fair distribution of wireless capacity between those users. Further refinements allow PBE-CC senders to gently approach this target at the connection start, so that other senders have time to react and adjust accordingly. However, if PBE-CC detects an increase in the one-way delay of its packets that its wireless capacity forecasts do not anticipate, this triggers a BBR-like mechanism to probe the bottleneck rate based on the pace of acknowledgement packets received by the PBE-CC sender.

We have implemented the PBE-CC congestion control module in 814 lines of user space C++ code. Mobile telephone wireless front ends should decode the necessary frequency bands in order to implement PBE-CC's physical-layer wireless capacity measurement module, but their (closed-source) firmware does not offer this functionality, and so we emulate the missing firmware functionality using the USRP software-defined radio in our 3,317-LoC C implementation.

Our performance evaluation uses Pantheon [48] to test PBE-CC head-to-head against BBR and CUBIC [19], leading congestion control algorithms, as well as recent congestion control algorithms for cellular [43, 49], and other recently-proposed algorithms such as Copa [6], PCC [11] and PCC-Vivace [12]. Our experiments begin with measurements of delay and throughput, under stationary user-device conditions, both indoors and outdoors, and both during busy and quiet hours. Further experiments evaluate the same under mobile user-device conditions, "controlled" competition for the wireless network capacity (that we introduce ourselves in a known manner), and "uncontrolled" competition from background traffic of other users at various times of the day. For each competing scheme, we report (individually) all throughput and delay order statistics, measured across 100-millisecond time windows, as well as average case results for these experiments. Table 1 summarizes our performance results: on average, PBE-CC achieves a 6.3% higher average throughput than BBR, while simultaneously reducing 95$^{\text{th}}$ percentile delay by a factor of 1.8× and average delay by a factor of 1.6×. Against Verus, an algorithm specially designed for cellular networks, PBE-CC achieves significant gains in both throughput and delay reduction, and against the much-slower Copa, PBE-CC achieves an approximate 11× throughput improvement while paying a relative 20% latency penalty. We also evaluate multi-user fairness, RTT fairness and TCP friendliness of PBE-CC in §6.4.

Table 1: Summary throughput speedup and delay reduction performance comparison *vs.* BBR, Verus, and Copa (averaged over 15 idle cellular links and 25 busy links).

Scheme		PBE-CC tput. speedup	PBE-CC delay reduction	
			95th. pctl.	avg. delay
BBR	Busy	1.04×	1.54×	1.39×
	Idle	1.10×	2.07×	1.84×
Verus	Busy	1.25×	3.97×	2.53×
	Idle	2.01×	3.44×	2.67×
Copa	Busy	10.35×	0.80×	0.80×
	Idle	12.94×	0.79×	0.82×

2 RELATED WORK

End-to-end congestion control. Loss-based algorithms [15, 19, 23, 39] achieve high throughput, but often introduce excessive delay, while delay-based algorithms [6, 8, 41] are prone to ACK delay, ACK compression, or network jitter, and thus often result in network capacity under-utilization. Moreover, it is widely known that these methods achieve poor capacity utilization when competing with concurrent loss-based algorithms [6, 39]. Other proposals use learned algorithms to optimize specific objective functions, to generate better congestion control actions than human crafted rules [5, 11, 12, 38, 42]. As we show in our evaluation (§6), online learning frequently converges to solutions that result in significant network under-utilization. BBR [10] targets convergence to Kleinrock's optimal operating point, *i.e.*, simultaneously maximizing throughput and minimizing delay, based on estimates of bottleneck bandwidth and round trip propagation time. BBR achieves the best performance among all the algorithms we test, but still under-utilizes the network and introduces excessive delay because of its capacity estimates are coarse-grained.

End-to-end congestion control for cellular networks. Some prior work treats the cellular link as a black box and makes use of throughput, packet delay and loss statistics to infer link capacity [21]. Raven [29] reduces interactive video latency by sending redundant data over multiple paths (Wi-Fi and cellular), using Multipath TCP [44]. PROTEUS [47] collects current throughput, loss,

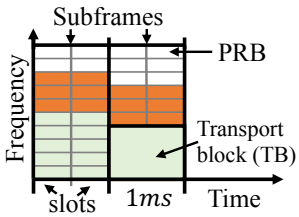


Figure 1: PRBs inside a subframe can be allocated to multiple users. Allocation in two slots are the same (represented using colors).

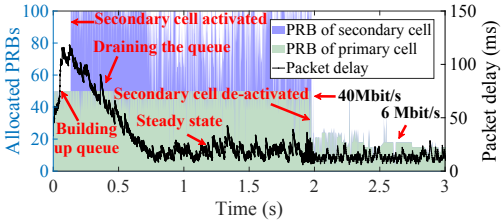


Figure 2: When the offered load of the server exceeds the maximum capacity of the primary cell, cellular network activates a secondary cell for the mobile user to support the high data rate, and deactivates it if the rate drops.

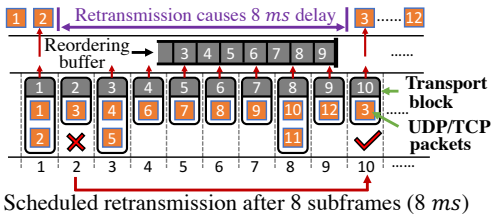


Figure 3: The mobile user buffers all out-of-sequence transport blocks in a reordering buffer until the erroneous block is retransmitted and corrected received (multiple retransmissions is possible), introducing a 8 ms delay.

and one-way delay, using regression trees to forecast future network performance. PropRate [30] replaces BBR's periodic bandwidth probing with continuous probing that oscillates the send rate around the estimated receive rate using packet size, and packet send/receive times. Sprout [43] leverages packet arrival times to infer the uncertain dynamics of the network path, forecasting link capacity based on these measurements. Similarly, ExLL [35] models the relationship between packet arrival patterns and cellular bandwidth usage to adjust send rate. Instead of attempting to infer the cellular network dynamics, Verus [49] tries to learn a delay profile that captures the relationship between target send window size and perceived end-to-end delay. Purely relying on end-to-end statistics, above algorithms inevitably suffers from capacity estimation inaccuracies and are sensitive to network dynamics, as we have demonstrated (§6.3). PBE-CC delivers superior performance because of its more fine-grained capacity estimation, achieved by directly measuring the wireless channel.

Cellular-aware congestion control proposals. ABC [17, 18] and the Draft IETF Mobile Throughput Guidance (MTG) standard [22] propose modifications of each mobile base station to explicitly communicate the best rate to the sender, but do not explicate specifics in the design of the capacity monitor that is critical for high performance. CQIC [32] embarks on a cross-layer design by extracting 3G link capacity estimates, but still lacks fine granularity. piStream [45] and CLAW [46] formulate a model that predicts utilized resource blocks from signal strength measurements. CLAW uses this model to speed up web browsing workloads, while piStream uses the model for video workloads, but the authors' own measurements show that signal strength's predictive power is quite limited, while PBE-CC decodes the control channel metadata directly, resulting in precise bandwidth utilization data that are not estimates.

Cellular PHY-layer monitoring tools. QXDM [36] and MobileInsight [31] extract control messages for a single mobile user, but cannot provide net information on the cell tower's capacity occupancy, as PBE-CC does. BurstTracker [7] locates the bottleneck of an end-to-end connection. LTEye [28] and OWL [9] decode control messages, but do not work with carrier aggregation (§3) and later advanced MIMO standards as PBE-CC does. All the foregoing tools stop short of a congestion control algorithm design.

3 LTE/5G NEW RADIO PRIMER

In this section, we introduce the relevant design of LTE's MAC and physical layer, with a focus on frequency division duplexing (FDD), the mode cellular operators use most widely. LTE adopts OFDMA, dividing the available wireless frequency bandwidth into 180 KHz chunks and time into 0.5 millisecond slots, as shown in Figure 1. The smallest time-frequency block (180 KHz and 0.5 ms) is called a *physical resource block* (PRB), which is the smallest unit that can be allocated to a user. LTE groups two slots into a one-millisecond *subframe*. The PRB allocation of two slots inside one subframe is the same. The data transmitted over one subframe is called one *transport block* (TB). The size of one TB varies, depending on the number of allocated PRBs and the wireless physical data rate of the user. The base station informs the mobile user of its bandwidth allocation (the amount and position of allocated PRBs) and wireless bit rate, including the modulation and coding scheme (MCS) and the number of spatial streams, through a control message transmitted over a *physical control channel* [3]. A mobile user decodes the control message of a subframe before decoding the TB inside it.

Carrier aggregation. By default, the base station delivers data to a mobile user via a primary *component carrier* (CC), or primary cell. When there is a huge amount of data to be delivered to the user, the base station activates a secondary cell to add capacity. The cellular network maintains a list of aggregated cells for each user and will activate them sequentially if necessary. The aggregated cells are deactivated if and when the user does not utilize the extra capacity. An example of the carrier activation and deactivation process is shown in Figure 2. A sender first sends data to a mobile user with a fixed offered load of 40 Mbits for two seconds, which exceeds the maximum capacity of the primary cell, so it causes packet buffering at this cell,[1] even when all the bandwidth are allocated for this user. The cellular network detects such a high-data-rate user and activates a secondary cell to help deliver the data to this user, at 0.13 seconds. Since 40 Mbit/s is below the aggregated capacity of the primary and secondary cell, the cellular network drains the built queue within 0.6 seconds, as shown in Figure 2. The sender reduces its sending rate to 6 Mbit/s, which is below the capacity of the primary cell, so the secondary cell is deactivated.

Cellular retransmission and reordering. The cellular network

[1] We note that packet buffering at the base station is not a prerequisite for activating secondary cells. The cellular network activates another cell for a user as long as such a user is consuming a large fraction of the bandwidth of the serving cell(s).

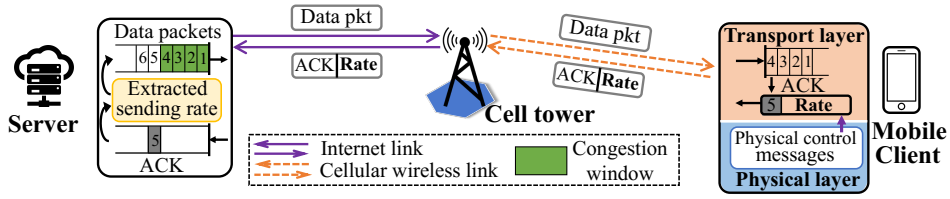


Figure 4: An overview of PBE-CC congestion control. The mobile clients decode the cellular control channel, which contains detailed information about the base station's available wireless capacity. PBE-CC senders control their send rate based on the estimated bottleneck capacity that the mobile user explicitly sends back, or based on the presence of ACKs from the receiver.

retransmits an erroneous transport block after eight subframes (milliseconds) of the original transmission, as shown in Figure 3. To guarantee in order delivery, the mobile user buffers all the transport blocks received in subframes between the original transmission and retransmission of the erroneous transport block (supposing they are received correctly) in a reordering buffer. When the retransmission succeeds, the mobile user report all the buffered transport blocks together with the retransmitted transport block to upper layers where the transport layer packets inside the transport blocks are extracted. As a result, the retransmission introduces a eight millisecond delay to the transport layer packets inside the erroneous transport block and the buffering and reordering operations at the receiver side introduces a decreasing delay (from seven to zero milliseconds) to the packets inside the following transport blocks. If the retransmission fails, the cellular network repeats the retransmission at most three times, introducing a latency penalty equal to a multiple (smaller than three) of eight milliseconds.

4 DESIGN

PBE-CC is a rate based, end-to-end congestion control algorithm for flows traversing cellular networks and terminating at mobile devices. **PBE-CC mobile clients** decode the cellular physical control channel, which contains detailed information about the base station's available wireless capacity. From this, the mobile user is able to estimate this quantity accurately, at millisecond time granularity. Depending on the location of the bottleneck link, **PBE-CC senders** control their send rate based on the estimated bottleneck capacity that the mobile user explicitly sends back, or based on the presence of ACKs from the receiver, as shown in Figure 4. Using its fine-grained capacity estimates, when the bottleneck is the wireless hop, PBE-CC can immediately increase its send rate to grab new available capacity without causing any congestion, and decrease its send rate accordingly, if competition with other mobile users or the wireless channel reduces wireless capacity.

As traffic patterns are highly dynamic, end-to-end connections face two possible network states, depending on the relative capacities of the bottleneck link in the Internet, and the cellular link. Most of the time, connections are in what we term a *wireless-bottleneck* state where the wireless cellular link is the bottleneck of the whole end-to-end connection. In this state, the PBE-CC mobile user can estimate and track the bottleneck capacity of the whole connection at millisecond granularity by decoding the cellular physical control channel (§4.2.1). The PBE-CC sender matches its send rate with the bottleneck capacity that the mobile user explicitly feeds back, almost exactly utilizing capacity and at the same time causing minimal packet buffering in the network. On the other hand, the connection is in an *Internet-bottleneck* state if the capacity of the Internet bottleneck is smaller than the capacity of the wireless cellular link. PBE-CC then switches to a cellular-tailored BBR-like congestion control strategy, to compete fairly with other flows that share the Internet bottleneck for a fair share of the bottleneck capacity (§4.2.3). PBE-CC tracks possible changes in these two states, controlling the sender's actions accordingly.

Kleinrock has proven that the operating point—maximizing delivered bandwidth while minimizing delay—is optimal for both individual connections and the network as a whole [26, 27]. The operating point is characterized by the insight that one should keep the pipe only just full. PBE-CC shares the same goal as BBR, which is to fill the pipe and minimize the buffering inside the network. PBE-CC limits the amount of inflight data to the bandwidth-delay product (BDP) calculated using estimated round-trip propagation time RTprop and bottleneck capacity with a congestion window, as shown in Figure 4, so PBE-CC senders often do not send excessive packets even when the feedback from mobile user is delayed, minimizing queuing in the network, for very low latency, as our experimental evaluation later demonstrates (§6).

4.1 Connection Start: Linear Rate Increase

On connection start, a PBE-CC sender executes a *linear rate increase* in order to approach a fair-share of the bottleneck capacity. By decoding the control channel, each PBE-CC user knows the number of other users sharing the cell bandwidth, as shown in Figure 5. PBE-CC therefore calculates expected fair-share bandwidth (in units of PRBs) P_{exp} using the total PRBs available in the cell P_{cell} and the number of active users N (including the mobile itself):

$$P_{\text{exp}} = P_{\text{cell}}/N. \tag{1}$$

The user then estimates its expected fair-share send rate C_f (in units of bits per subframe) as:

$$C_f = R_w \cdot P_{\text{exp}}, \tag{2}$$

where R_w is the wireless physical data rate (with units of bits per PRB) calculated using the number of spatial streams together with the coding and modulation rate for each stream.

The PBE-CC sender linearly increases its send rate from zero to the fair-share send rate C_f in three RTTs. The mobile user updates C_f every millisecond, and sends the calculated rate back to the server in each acknowledgement. PBE-CC's linear increase prevents bursty traffic and leaves time for the cell tower and the other

users sharing that tower to react to the increased traffic. The cell tower reacts to the mobile user's increasing send rate by proportionally allocating more bandwidth, which results in less bandwidth allocated to other users. Another PBE-CC user immediately detects such a decrease in its allocated bandwidth and signals its sender to lower its send rate accordingly. Eventually, all PBE-CC's users tend to achieve equilibrium with an equally-shared bandwidth. When two or more component carriers are active during the fair-share approaching state, we calculate target send rate separately for each aggregated cell, and sum them up as C_f. When more carriers are activated during congestion avoidance (§4.2), PBE-CC restarts this fair-share approaching process.

The user ends linear rate increase and enters congestion avoidance when it achieves its fair-share sending rate C_f. If the bottleneck of the connection is inside the Internet, rate C_f is not achievable, so the achieved throughput at the cell tower stays at a rate below C_f and end-to-end packet delay increases with increasing sender offered load. When the mobile user detects that the receiving rate stops increasing for one RTprop, while the oneway packet delay increases monotonically with an increasing offered load, it also ends the linear rate increase phase and switches to our cellular-tailored BBR to handle congestion in the Internet (§4.2.3).

4.2 Steady State: Congestion Avoidance

We now present the design of PBE-CC's congestion avoidance algorithm. When the connection is in the wireless bottleneck state, PBE-CC senders match their send rate to estimated wireless capacity (§4.2.1). Similar to connection startup, PBE-CC identifies a possible transition from a wireless-bottleneck to Internet-bottleneck state (§4.2.2), and if this happens, switches to to a cellular-tailored BBR (§4.2.3) to compete fairly with flows at the bottleneck.

4.2.1 Wireless Bottleneck State. Here a PBE-CC mobile user estimates the available cellular wireless capacity C_p (in units of bits per subframe) as

$$C_p = \sum_{i=1}^{N_{\text{cell}}} \left(R_{w,i} \cdot \left(P_{a,i} + \frac{1}{N_i} P_{\text{idle},i} \right) \right) \tag{3}$$

where N_{cell} is the number of activated cells for this user, $P_{a,i}$ is the number of PRBs allocated for this user in the ith cell, N_i is the number of mobile users in the ith cell, and $P_{\text{idle},i}$ represents the number of idle PRBs in the ith cell:

$$P_{\text{idle},i} = P_{\text{cell},i} - \sum_{j=1}^{N_i} P^j_{a,i} \tag{4}$$

where $P^j_{a,i}$ represents the allocated PRB for user j of the ith cell. To smooth the estimation results, we average the calculated $R_{w,i}$, $P_{\text{idle},i}$ and $P_{a,i}$ from the most recent RTprop subframes (*e.g.*, we average the above parameters over the most recent 40 subframes if the connection RTT is 40 ms).

To interpret estimated capacity C_p, we consider each component of Eqn. 3. First, the wireless physical layer data rate R_w enables the mobile user to track capacity variations caused by varying channel quality. Second, the mobile user reacts to the appearance of new users by tracking the number of PRBs allocated for itself (P_a). For example, as shown in Figure 5, P_a for User 1 decreases when a new user, *i.e.*, User 2, starts receiving traffic. On detection of fewer allocated PRBs, User 1's sender lowers its send rate to match the decreasing capacity estimated using Eqn. 3.

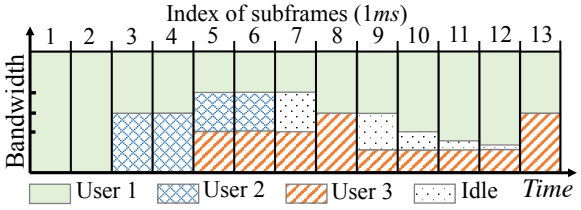
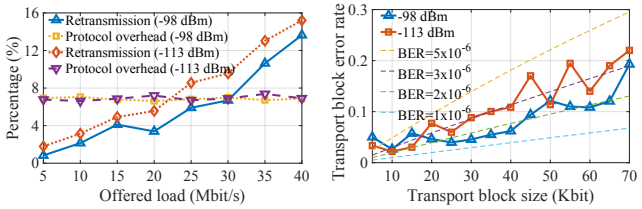


Figure 5: One mobile user tracks the number of PRBs allocated for itself, for other mobile users and that are idle.

When idle PRBs P_{idle} appear in a cell for a connection that is wirelessly bottlenecked, all PBE-CC clients immediately detect them by checking the decoded control message, and inform their senders to increase their rates to grab a fair-share portion of the idle PRBs, *i.e.*, P_{idle}/N. This may happen in several cases: first, idle PRBs appear when a sender finishes a flow. As shown in the example of Figure 5, after User 2's flow finishes in subframe six, Users 1 and 3 immediately observe idle PRBs in subframe seven and then share the available PRBs equally in subframe eight. Second, idle PRBs also appear when the data rate of a user's flow decreases, *e.g.*, Subframe 9 in Figure 5, which could be caused by, *e.g.*, congestion in the Internet, the application itself, or a shift of traffic from one cell to another aggregated cell by the cellular network. In this case, all other users immediately detect and occupy their fair share of the newly-idle PRBs. Other users share $1/N$ of the idle PRBs with User 3, whose data rate is limited and thus is not able to grab more PRBs. As a result, if we define the number of idle PRBs in Subframe 9 as P', there will be P'/N left idle in Subframe 10. Similarly, other users detect these idle PRBs in Subframe 11, but still only occupy their fair share portion, so P'/N^2 will be left idle in Subframe 12. The network converges to a state where all other users other than the User 2 grab all the idle bandwidth.



(a) Percentage of overhead. **(b) Block error rate.**

Figure 6: The percentage of capacity used for transport block retransmission and transmission of protocol overhead is given in (a). The relationship between transport block error rate and transport block size is given in (b).

Cross-layer bit rate translation The capacities C_f and C_p (Eqns. 2 and 3) are wireless physical-layer capacities differing from transport-layer data rates due to MAC-layer retransmissions and (constant) protocol header overhead. PBE-CC therefore needs to transform the estimated physical-layer capacity C_p to a transport layer goodput C_t, and feedback C_t back to the server to set its send rate.

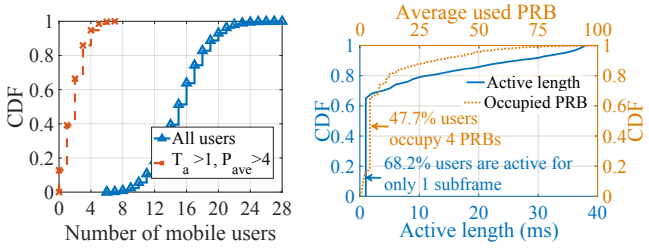
The cell indicates a retransmitted transport block using a *new-data-indicator*, so we can separately measure retransmission overhead and protocol overhead. Figure 6(a) plots the measured overhead at two different locations and varying sender offered loads. The probability of a TB error determines retransmission overhead: if the bit error rate (BER) of each bit inside one TB is p and bit errors are *i.i.d.*, the TB error rate is $1-(1-p)^L$, where L is the TB size. We plot in Figure 6(b) theoretical TB error rate (for $p = 5\times10^{-6}$, 3×10^{-6}, and 1×10^{-6}) and empirical TB error rate, noting a good fit between experimental data and theory. Based on these results, PBE-CC models the relationship between C_p and C_t as

$$C_p = C_t + C_t \cdot \left(1-(1-p)^L\right) + \gamma \cdot C_p \tag{5}$$

where $\gamma = 6.8\%$ is the protocol overhead. When one user takes its PBE-CC-allocated fair-share capacity (Eqn. 3), the TB size L (number of bits in one subframe, *i.e.*, 10^{-3} s), is $L = C_t \cdot 10^{-3}$. We estimate p using measured *signal to interference noise ratio* (SINR), then by solving Eq. 5 given a measured physical layer capacity C_p, we estimate transport layer goodput C_t. To speed up the calculation, PBE-CC uses a look-up table to store the transformation.

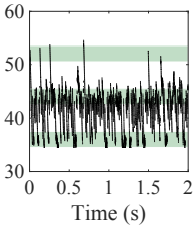
Handling control traffic. PBE-CC aims to fairly share wireless bandwidth between all active users, but our experimental results shows that significant amount of detected users are active not for data, but rather to update network parameters shared by both base station and mobile, *e.g.*, the periods of various timers, list of aggregated cells, and many pricing and security-related parameters. Because of such users, the number of detected active users at each time point could be large. For example, we plot the distribution of the number of detected active users in a 40 ms interval, across a 5 hour interval, measured from a busy cell tower, in Figure 7(b). On average, we observe on average 15.8 and maximum 28 active users, in those 40 ms interval. PBE-CC excludes those users in its fair-share capacity calculation, reverting to the cell tower to allocate small amounts of bandwidth for these users and then reacting to that allocation by tracking the decrease of allocated bandwidth (P_a in Eqn. 3) and lowering send rate by that amount. Our key observation is that the control traffic occupies a small number of PRBs and only active for small amount of time. We plot the distribution of the average occupied PRBs and active time (subframes) of all detected active users in Figure 7(b). We see that 68.2% of users occupies exactly four PRBs and is active for exactly one subframe, among which 95% of users are receiving control traffic from the base station. Therefore, the PBE-CC monitor filters users that are only active for parameter updating, based on thresholding the active time duration (subframes) and allocated bandwidth (PRBs) ($T_a > 1, P_a > 4$), after which the number of detected active users decreases significantly—the average number of detected user inside a 40 ms interval decreases from 15 to 1.3, and we only observe at most seven active users competing for the bandwidth simultaneously, as shown in Figure 7(a). We set the N in Eqns. 2 and 3 to the number of active users we detect after applying the threshold. The calculation of idle PRBs in Eqn. 4, however, takes every identified user into account.

4.2.2 Switching between Bottleneck States. When sender offered load exceeds the capacity of the Internet bottleneck, packet queuing induces PBE-CC to switch from the wireless bottleneck state to the Internet bottleneck state. PBE-CC triggers a switch when the instantaneous one-way packet delay exceeds a threshold. Theoretically, we should set the threshold to the one way propagation delay between the server and clients ($D_{\text{th}} = D_{\text{prop}}$). PBE-CC estimates D_{prop} as the minimum delay observed in a 10-second window, evoking BBR's round-trip propagation delay estimation method. PBE-CC also updates the true D_{prop} by draining the buffer as BBR does, if estimated packet delay maintains constant for 10 seconds.



(a) Number of active users. **(b) Measured T_a and P_{ave}.**

Figure 7: Number of mobile users exchanging data with the base station (a), and activity length T_a and average consumed PRBs P_{ave} of each detected mobile user (b).



(a) 6 Mbit/s. **(b) 24 Mbit/s.** **(c) 36 Mbit/s.**

Figure 8: Higher send rates (sub-caption label) result in a higher probability of transport block errors, so more packets encounter eight millisecond retransmission delays.

The theoretical threshold, however, works poorly in practice because of the reordering operation. We observe that the mobile user frequently buffers received packets in its reorder buffer (§3), especially when offered load from the sender is high, causing significant fluctuations of packet delay. To demonstrate such an effect, we plot the measured one way delay at a mobile user under different sender offered loads in Figure 8. We see that when offered load is low (6 Mbit/s), only a small portion of the received packets are retransmitted, as shown in Figure 8(a). We also observe an approximate three millisecond network jitter introduced to the packet delay. When the offered load increases, the transport block error rate increases accordingly, as we have discussed in §4.2.1. Consequently, the mobile user buffers more and more packets in its reorder buffer, introducing an multiple of eight ms retransmission delay to a increasing number of received packets, as shown in Figure 8(b) and 8(c). We note that, the minimum delay still captures the one way propagation delay, as there always are packets received correctly without retransmission and directly without buffering at the reorder buffer, *e.g.*, the packets inside transport block of the first subframe in Figure 3.

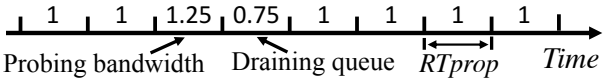


Figure 9: BBR adopts a eight-phase cycle to probe the network bandwidth. The length of each phase is set to *RTprop*.

According to the above analysis, we set the switching threshold to $D_{\text{th}} = \left(D_{\text{prop}} + 3 \cdot 8 + 3\right)$ ms, where $(3 \cdot 8)$ ms accounts for the delay introduced by the three consecutive retransmissions (a transport block can be retransmitted at most three times [4]) and 3 ms accounts for the network jitter (according to our experimental results, 94.1% of the time, jitter is ≤ 3 ms). To further mitigate the impact of greater network jitter and improve robustness, PBE-CC adds a threshold for the number of consecutive packets with delay exceeding the delay threshold, set to the number of packets N_{pkt} that can be transmitted over six subframes using current data rate:

$$N_{\text{pkt}} = 6 \cdot C_t / \text{MSS} \tag{6}$$

where C_t is the current transport layer capacity with unit bits per subframe, and MSS is the maximum segment size. We note that since our algorithm makes decisions based on relative delay, *i.e.*, the difference between current propagation delay and the threshold, instead of the absolute value of the delay, PBE-CC does not require synchronization between the server and mobile clients.

4.2.3 Internet Bottleneck State. PBE-CC switches to a cellular-tailored BBR to probe a rate that matches the capacity of the bottleneck link inside the Internet. BBR senders estimate the bottleneck bandwidth of the connection (*BtlBw*) as the maximum delivery rate in recent 10 RTTs, and set their offered rate to $pacing\_gain \cdot BtlBw$. BBR's *pacing_gain* is set to 1.25 to probe possible idle bandwidth, to 0.75 when draining packets buffered in the previous probing period, and to one the rest of time. BBR's ProbeBW state repeats an eight-phase cycle to probe bandwidth. The length of each phase is set to *RTprop*, and the pacing gain in each phase is shown in Figure 9. PBE-CC directly enters BBR's ProbeBW state, then follows the same control logic as BBR to alternate between BBR's ProbeBW, ProbeRTT, StartUp, and Drain states.

Wireless-aware, BBR-like probing. PBE-CC probes for a higher data rate that the Internet bottleneck supports, but also takes into account the fair-share send rate of the cellular wireless link. We adapt BBR's bandwidth probing scheme, changing the probing rate C_{probe} from a fixed $1.25BtlBw$ to

$$C_{\text{probe}} = \min\left\{1.25BtlBw, C_f\right\}, \tag{7}$$

where C_f is the maximum fair-share capacity of the wireless link (estimated according to Eqn. 2 and translated to transport layer capacity according to Eqn. 5 below). The mobile user explicitly sends C_f back to the sender when an Internet bottleneck is detected. Similar to BBR, PBE-CC enters a draining phase after the probing phase to drain any buffered packets.

When PBE-CC detects that the network is in the Internet-bottleneck state, there is already a packet queue formed inside the network. Therefore, before switching to handle that state, PBE-CC enters an additional draining phase that lasts for one *RTprop*. During the draining phase, PBE-CC sets its send rate to $0.5BtlBw$, leaving the remaining capacity of $0.5BtlBw$ for the bottleneck link to drain the packets buffered inside its queue.

Switching back to wireless bottleneck state. If PBE-CC's send rate reaches C_f without causing any packet queuing in the network, *i.e.*, the mobile user observes N_{pkt} (calculated according to Eqn 6) consecutive packets with delay smaller than D_{th} ms are observed at the mobile user, then PBE-CC exits the Internet-bottleneck state and re-enters the wireless bottleneck state, staying in that state until the network is switched back to Internet-bottleneck state.

4.3 Fairness and TCP-friendliness

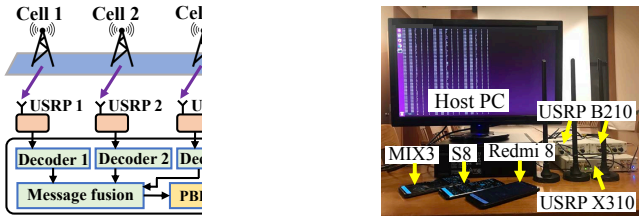
As it only modifies BBR's algorithms to be more conservative, PBE-CC is strictly less aggressive than BBR when competing with flows sharing the same Internet bottleneck. BBR's multi-user fairness, RTT-fairness and TCP-friendliness have been well established in the literature[20, 33, 37, 40].

In the wireless bottleneck state, multiple competing PBE-CC mobile clients quickly converge to a equilibrium with fair-share cellular wireless capacity (as we demonstrate below in §6.4.1), because each PBE-CC mobile client knows the number of competing users and their capacity usage in each aggregated cell by decoding the cellular physical control channel, allowing it to explicitly calculate its fair-share capacity (§4.1) and then guide its sender to match its sending rate accordingly. In contrast, conventional end-to-end congestion control algorithms need to probe the fair-share of bottleneck capacity with a more complicated series of probing and backoff steps, which is less efficient. PBE-CC also fairly shares wireless link capacity with existing congestion control algorithms, *e.g.*, CUBIC and BBR, with the help of cell tower's fairness policy, as our experimental evaluation later demonstrates (§6.4.3).

PBE-CC flows with different propagation delays fairly share wireless capacity (as we demonstrate in §6.4.2), because of two reasons, one from the design of PBE-CC and one from the buffer structure of base station. First, PBE-CC explicitly calculates the fair-share capacity, while most conventional congestion control algorithm adopt additive-increase multiplicative-decrease (AMID) schemes to probe for the fair share. During the additive increase, the sender of a flow with smaller propagation delay increases its window faster than flows with larger delay, resulting in unfairness [19, 34]. Second, the base station provides separate buffers for every user, which prevents large-*RTprop* connections from dominating the bottleneck buffer. For example, a BBR connection with a large *RTprop* calculates a large BDP and thus injects significant amount of inflight packets into the network, which queue at the bottleneck buffer and lower the delivery rate for another BBR flow with a small *RTprop* and hence a small number of inflight packets. The separate buffer at cellular base station isolates the inflight packets from different flows sharing the wireless link and thus prevents unfairness.

5 IMPLEMENTATION

Programming a mobile phone to decode every control message transmitted over the control channel requires customization of the cellular firmware inside the phone. The source code of current cellular firmware, however, is proprietary to cellular equipment manufacturers, thus is not accessible. As a proof of concept, we build an open-source congestion control prototyping platform that supports control message decoding, bypassing the need to customize



(a) Prototypin

(b) System setup.

Figure 10: The architecture of the open-source PBE-CC cellular congestion control prototyping platform (a). The setup of PBE-CC mobile clients is shown in (b).

firmware. The key component of our platform is an open-source control channel decoder that uses an off-the-shelf software defined radio (USRP in our implementation) as the RF front-end to collect cellular wireless signals, and a PC as the host to decode the control messages from the collected signals. We start multiple parallel control channel decoders, each decoding the signal from one cell in the list of aggregated cells of the mobile user, as shown in Figure 10(a). Our Message Fusion module aligns the decoded control messages from multiple decoders according to their subframe indices, feeding the aligned messages to our Congestion Control module.

We implement our cellular control channel decoder in 3,300 lines of C code (excluding reused code). We reuse the physical layer signal processing modules from an open-source LTE library (*srsLTE* [16]), *i.e.*, a wireless channel estimator, a demodulator, and a convolutional decoder. Each decoder decodes the control channel by searching every possible message position inside the control channel of one subframe and trying all possible formats at each location until finding the correct message.[2] We implement the parallel decoding structure using multi-threading, allowing one PC to decode the control channel of multiple cells simultaneously. In our test, a six-core PC is able to decode six cell towers while maintaining CPU usage of each core below 40 percent. We will open-source our platform to facilitate future cross-layer cellular congestion control design and prototyping.

We implement a user-space, UDP-based prototype of PBE-CC's congestion control algorithm using 874 lines of C++ code (517 on the mobile client side and 357 at the sender side). The client-side PBE-CC module takes the decoded control messages as input, and communicates with the sender side via a commercial mobile phone tethered with the host PC, as shown in Figure 10(a). When the PBE-CC mobile client receives a data packet, it estimates the one way packet propagation delay D_{prop} (§4.2.2), and feeds back the estimated capacity. We describe the capacity using an interval in milliseconds between sending two 1500-byte packets, and represent it with a 32-bit integer. The PBE-CC client also identifies the current bottleneck state, notifying the sender via one bit in the ACK. When the PBE-CC sender receives an ACK, it sets its sending rate to the capacity indicated therein. The PBE-CC sender also updates its estimated *RTprop* and *BtlBw* with every received ACK, so it can immediately switch to the cellular-tailored BBR if and when the bottleneck location changes.

[2]The 3GPP standard defines 10 formats for control messages [3]. The base station does not explicitly indicate the format of the message it sends.

6 EVALUATION

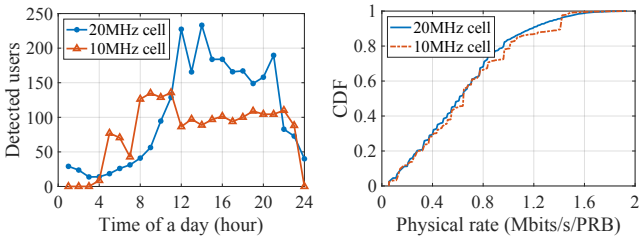
In this section, we evaluate the performance of PBE-CC in a commercial cellular network and compare with existing end-to-end congestion control algorithms.

6.1 Methodology

Content senders. We configure Amazon AWS servers as the PBE-CC senders. To evaluate PBE-CC's performance over flows with significantly different RTT, we setup AWS servers at different continents, *i.e.*, three in US and one in Singapore.

Mobile clients. Each PBE-CC mobile client is a combination of multiple USRPs for signal collection, a host PC for control channel decoding, and a commercial mobile phone for cellular communication, as shown in Figure 10(b). We use both USRP X310 [14] and B210 [13] in our implementation. The host PC we use for each mobile client is a Dell OptiPlex 7060 (Intel Core i7-8700 CPU, 16 GB RAM, and Ubuntu 16.04). We use various types of mobile phones that support carrier aggregation in hardware, including a Xiaomi MIX3, a Redmi 8, and a Samsung S8. The cellular network configures the same primary cell for all three phones, but different numbers of aggregated cells for each phone, *i.e.*, only one cell for the Redmi 8, two cells for the MIX3 and three cells for the S8.

Congestion control algorithms to compare. We compare PBE-CC against seven end-to-end congestion control algorithms, including algorithms specially designed for cellular networks like Sprout [43] and Verus [49], algorithms that have already been included inside the official Linux kernel like BBR [10] and CUBIC [19], and recently-proposed algorithms like Copa [6], PCC [11] and PCC-Vivace [12]. We test all the above algorithms in commercial cellular networks covering our campus using Pantheon [48].



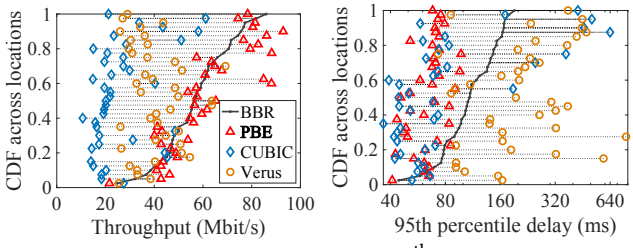
(a) Detected users.

(b) Physical data rate.

Figure 11: (a) The number of detected users in each hour of a day that have data communication with two base stations (a 20 MHz one and a 10 MHz one). (b) The distribution of wireless physical data rate of the detected users.

6.2 Micro-benchmark: Cell Status

In this section, we perform a micro-benchmark to present two important statistics of the cell tower: (1) the number of users that have communicated with the cell tower in each hour and (2) the distribution of wireless physical data rate of the users. We leverage our control channel decoder to decode the control messages that two base stations (a 20 MHz one and a 10 MHz one) transmit. We conduct the experiments for 24 hours and count the number of active users in each hour. We plot the result in Figure 11(a), from which we see that each cell serves a large number of users during



(a) Average throughput. **(b) 95th percentile delay.**

Figure 12: The distribution of throughput (a) and 95th percentile delay (b), of PBE-CC, BBR, Verus, and CUBIC (the four *"high throughput"* algorithms), across 40 locations.

peak hours of a day, *e.g.*, during the 12 to 20 hours period, the average number of users per hour is 181 and 97 for 20 MHz and 10 MHz cell, respectively. Furthermore, the number of users varies significantly within a day, *i.e.*, maximum 233 and 135 users, minimum 13 and zero users for 20 MHz and 10 MHz cell, respectively. We note that the 10 MHz cell is turned off by the operator during zero to three hour period, so we observe zero users. We also plot the distribution of the wireless physical data rate of all detected users, in Figure 11(b). We see that even though the users has diverse data rates, a large portion are low-rate users, *e.g.*, 77.4% and 71.9% users have rate smaller than half of the maximum achievable data rate (1.8 Mbit/s/PRB), for 10 MHz and 20 MHz cell, respectively. In the following sections, we evaluate the performance of PBE-CC working atop of these cells that serve large number of diverse users.

6.3 End-to-end Delay and Throughput

In this section, we investigate the delay and throughput performance of PBE-CC achieved in a commercial cellular network.

6.3.1 Performance of Stationary Cellular Links. We investigate PBE-CC's performance on stationary cellular links. We build connections between servers and stationary mobile users over which senders transmit to their corresponding users for 20 seconds, recording achieved throughput, packet delay, and arrival time in each flow. We change the congestion control algorithm the sender adopts and test eight algorithms sequentially. Since the capacity of the cellular network varies when testing each algorithm, we repeat the whole preceding test sequence (sequentially testing all algorithms) five times at one location to provide a fair comparison of achieved throughput, across different congestion control algorithms. Furthermore, we conduct the foregoing experiment using different phones, in order to measure performance with different numbers of aggregated cells. We repeat these experiments at multiple indoor and outdoor locations and at different times of the day, *i.e.*, daytime when the cell is busy, and late night when the cell is idle. In total, we test 40 locations, covering all combinations of indoor/outdoor, one/two/three aggregated cells and busy/idle links.

Comparison among high-throughput algorithms. As we will demonstrate in the following section, PBE-CC, BBR, CUBIC, and Verus achieve significantly higher throughput than the other four algorithms we examine. We plot the distribution of the averaged throughput and 95th percentile one way delay achieved by these four algorithms, in Figure 12(a) and 12(b). We see that PBE-CC achieves the highest throughput for most of the stationary links, while simultaneously maintaining very low latency. Table 1 on p. 2 summarizes the performance improvement of PBE-CC over BBR and Verus. PBE-CC achieves 2.3× average higher throughput than CUBIC, and at the same time reduces 95th percentile delay by 1.8×.

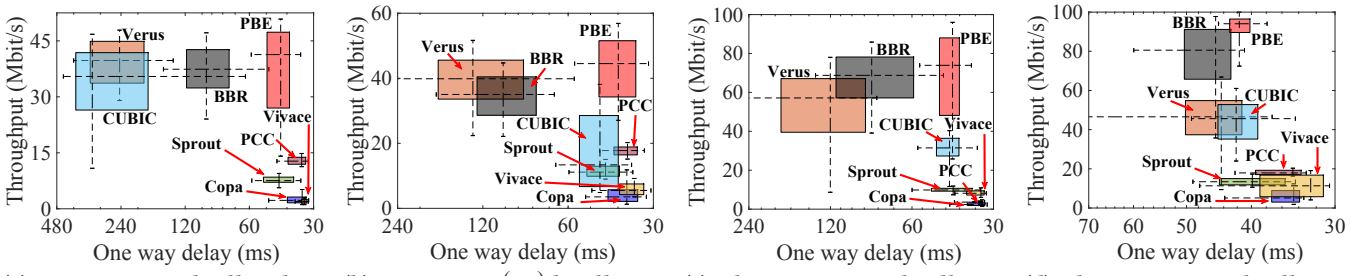
Detailed comparison among eight algorithms. To provide a detailed performance comparison among all eight algorithms, we select six representative locations, and plot the 10th, 25th, 50th, 75th, and 90th percentile throughputs (averaged over every 100-millisecond interval) and delay, for eight algorithms, in Figures 13 and 14. We have three observations from these figures. First, PBE-CC achieves high average throughput, but also has somewhat high throughput variance, since PBE-CC is able to match its send rate to the varying wireless channel capacity. BBR achieves comparable throughput with PBE-CC in all selected locations, but with higher delay. Verus, a congestion control algorithm designed for cellular networks, also achieves relatively high throughput in many locations, but introduces excessive packet delays. The performance of CUBIC is highly unpredictable, alternating between high throughput (but high delay) and low throughput (but low delay), as our order statistics demonstrate. The other four algorithms, including Copa, PCC, PCC-Vivace, and Sprout, have a large throughput disadvantage compared to PBE-CC. We plot the number of locations at which each congestion control algorithm triggers the cellular network to activate secondary cells for providing extra throughput (maximum 30 locations, since we use Redmi 8 that uses only one cell, in 10 locations), in Figure 15. We see that Copa, PCC, PCC-Vivace, and Sprout use very conservative send rates, so the cellular network disables carrier aggregation at most locations, resulting in significant under-utilization of the available wireless capacity.

PBE-CC achieves a low delay and delay variance. Comparing against BBR and Verus, two algorithms with relatively high throughput, PBE-CC incurs much smaller delays. However, PBE-CC has a slightly higher latency than the four algorithms with low throughput. Such a delay gap is mainly caused by cellular retransmissions: as we have demonstrated in Figure 6(b), higher throughputs result in a larger TB error rates, and thus more retransmissions. Therefore, under schemes with higher throughput, slightly more packets incur a multiple of eight millisecond retransmission delay.

Finally, we observe that PBE-CC has low variance in both delay and throughput when cells are idle, as shown in Figures 13(d) and 14(b). Without competing traffic and mobility, wireless capacity becomes stable for a static user in an idle cell. PBE-CC then achieves stable throughput and delay by accurately estimating this capacity.

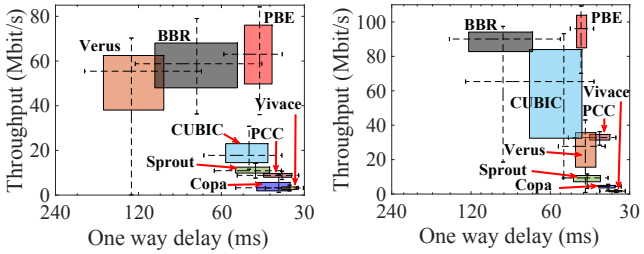
Alternation between states. On average, PBE-CC spends 18% and 4% of its time working in Internet-bottleneck state, for 25 busy links and 15 idle links, respectively, which validates our assumption that a connection traversing a cellular network is bottlenecked at the cellular wireless link for most of the time.

6.3.2 Performance under Mobility. A major source of cellular wireless capacity variations arise from wireless channel quality variations, caused by client mobility. In this section, we investigate PBE-CC's performance under mobility. We conduct this experiment at night when the cell is approximately idle to reduce the capacity variations introduced by other random competing users. In



(a) One aggregated cell, indoor and busy hours. **(b) Two aggregated cells, indoor and busy hours.** **(c) Three aggregated cells, indoor and busy hours.** **(d) Three aggregated cells, indoor and late night (idle cells).**

Figure 13: One way packet delay and throughput achieved by eight congestion control algorithms. The right and lower edge of the box represents the 25% percentile of the achieved delay and throughput, respectively. The left and upper edge give the 75th percentiles. The two ends of the error bar gives the 10th and 90th percentiles. The intersection point of the horizontal and vertical error bar represents the median of achieved delay and throughput.



(a) Two aggregated cells, outdoor and busy hours. **(b) Two aggregated cells, outdoor and late night (idle).**

Figure 14: The oneway packet delay and throughput achieved by eight congestion control algorithms in two different outdoor tests covering the busy and idle cell status.

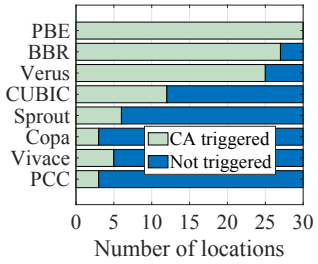


Figure 15: The number of locations at which each congestion control algorithm triggers carrier aggregation.

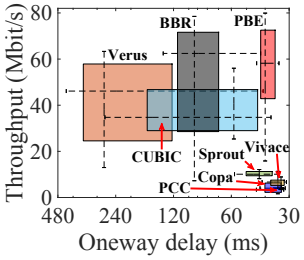
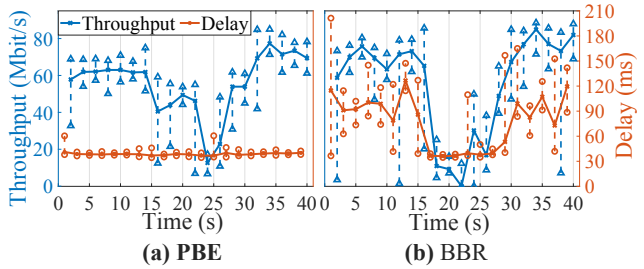


Figure 16: The achieved delay and throughput of when the mobile client is moving along the same trajectory.



(a) PBE **(b)** BBR

Figure 17: Delay and throughput achieved by PBE-CC (a) and BBR (b) when the user is moving along the same trajectory.

each test, we put the phone at a location with RSSI of −85 dBm for the first 13 seconds, and then move it along a predefined trajectory to another location with RSSI of −105 dBm in the next 13 seconds. We move the phone back to the starting location (−85 dBm) with a faster speed, taking about four seconds and put it there for 10 seconds. In total, each test takes 40 seconds. We repeat the same process for each congestion control algorithm.

We present each algorithm's achieved throughput and delay in Figure 16, from which, we see that PBE-CC consistently achieves low delay (95th percentile of 64 ms) and high average throughput (55 Mbit/s). BBR achieves comparable throughput (55 Mbit/s) with PBE-CC but suffers much higher delay (156 ms). CUBIC and Verus achieve much lower throughput than PBE-CC (38 Mbit/s and 41 Mbit/s) and also introduces high delay (296 ms and 467 ms). Other algorithms, *e.g.*, PCC, PCC-Vivace, Sprout, and Copa, have low throughput, resulting in under-utilization of wireless capacity, so mobility has a trivial effect on their packet delay.

To further demonstrate PBE-CC's ability to track mobility, we divide the 40-second experimentation period into 20 two-second intervals and plot median throughput and delay of each interval for PBE-CC and BBR, in Figure 17. We see that PBE-CC lowers and increases its send rate accurately when the signal strength decreases from 13 to 26 seconds and then increases from 26 to 30 seconds because of mobility, resulting in nearly zero buffering in the network. On the other hand, BBR overreacts to the signal strength decrease, reducing its send rate more than needed, because of its inaccurate end-to-end capacity estimation. BBR also overestimates capacity when the signal quality recovers at 30 seconds, causing packet queuing and introducing excessive packet delay.

6.3.3 Performance under Controlled Competition. Besides mobility, the competition between mobile clients for limited wireless capacity is another major source of variations in network capacity. In this section, we use controllable, on-off competing traffic to demonstrate PBE-CC's capability to track the time-varying wireless bandwidth allocation caused by competition. Specifically, we start a PBE-CC flow that runs for 40 seconds using a Redmi 8 phone. Every eight seconds, we also start a four-second concurrent flow with a fixed offered load of 60 Mbit/s from an AWS server, using a Xiaomi MIX3. We conduct the experiments at night to make the possibility of uncontrolled competition from other users remote. We repeat the

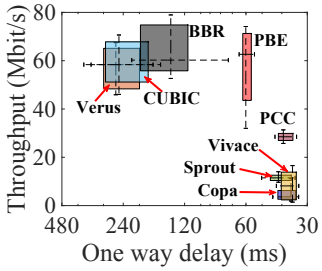


Figure 18: Achieved delay and throughput with controlled competing traffic.

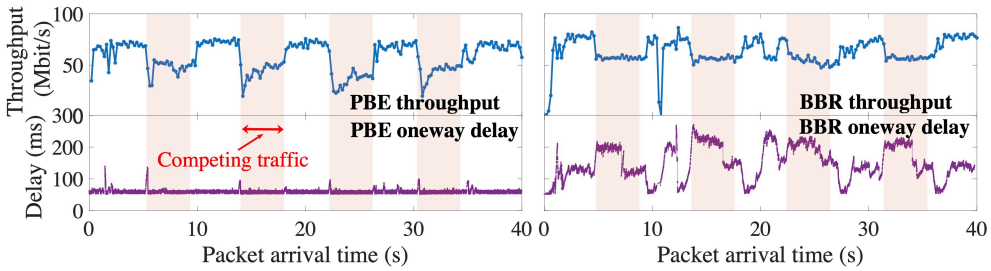
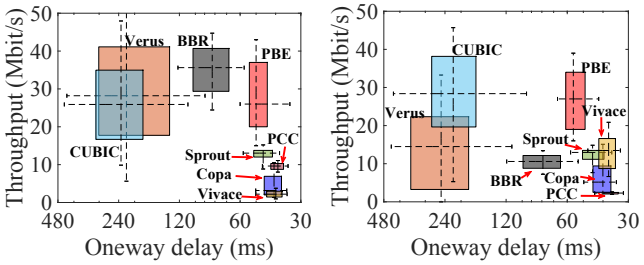


Figure 19: Average throughput and delay of every received packet in a flow. PBE-CC's rate increase and decrease is more responsive, thus grabbing capacity faster and keeping delay constant, respectively. In contrast, BBR suffers delay fluctuations.



(a) The first connection.

(b) The second connection.

Figure 20: The oneway delay and throughput achieved by eight congestion control algorithms for two concurrent connections between one device and two remote servers.

experiment using different congestion control algorithms.

We plot each algorithm's throughput and delay in Figure 18, from which, we see that only PBE-CC can simultaneously achieve high throughput and low latency. The average throughput of PBE-CC is 57 Mbit/s, comparable with CUBIC at 58 Mbit/s, and Verus at 56 Mbit/s, but slightly smaller than BBR at 62 Mbit/s. But the average and 95th percentile delay of PBE-CC is 61 ms and 71 ms, much smaller than BBR at 147 ms and 227 ms, CUBIC at 252 ms and 416 ms, and Verus at 263 ms and 403 ms. To further demonstrate PBE-CC's and BBR's reactions to competing traffic, we also plot the throughput (averaged over every 200 millisecond interval) and the delay of all received packets, in Figure 19, where the shaded areas represent the time periods when the concurrent competing traffic generated by the MIX3 is present. We see that PBE-CC accurately tracks the entrance of the competitor and lowers its sending rate promptly, resulting in nearly no packet queuing. PBE-CC immediately grabs the idle bandwidth when the competing traffic finishes its flow, maximizing the achieved throughput. In contrast, BBR cannot timely detects the decreasing capacity caused by competing traffic, resulting significantly enlarged delay.

6.3.4 Single device multiple connections. In this section, we evaluate how PBE-CC performs in the scenario where one device simultaneously starts multiple connections with different remote servers. Specifically, we let the MIX3 start two concurrent flows with two AWS servers, each running for 40 seconds. We repeat the experiments using different congestion control algorithms, and plot each algorithm's throughput and delay in Figure 20. We see that PBE-CC achieves high throughput and low delay for both flows. The average throughput is 26 Mbit/s and 28 Mbit/s, and the median delay is 48 ms and 56 ms, for the first and second flow, respectively. Furthermore, PBE-CC fairly allocates the estimated capacity for two flows so these two flows have similar throughput, while other algorithms may result in unbalanced throughput for multiple flows, *e.g.*, BBR achieves 10 Mbit/s and 35 Mbit/s for the first and second connection, respectively. We note that even though PBE-CC may achieve a smaller throughput for a single connection compared to other algorithms, *e.g.*, the first connection comparing with BBR, PBE-CC provides better fairness across connections.

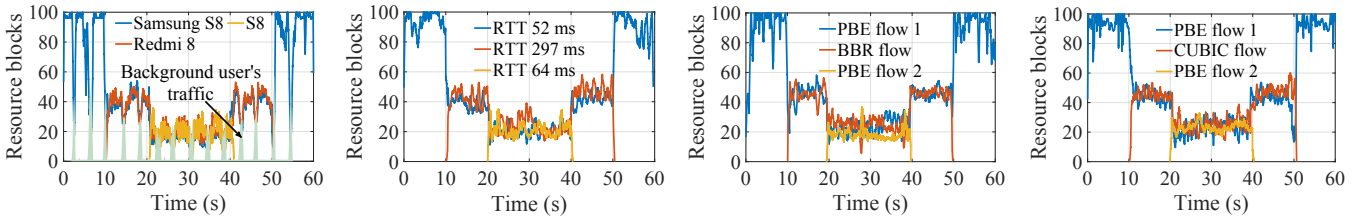
6.4 Fairness

In this section, we evaluate the fairness of PBE-CC, focusing on the case where the bottleneck is the cellular wireless link.

Methodology. Without knowing the base station's resource allocation algorithm and fairness policy, simulation-based experiments cannot predict real-world cellular network behavior. We therefore, evaluate PBE-CC's fairness directly in a cellular deployment. To eliminate the impact of background traffic, we conduct our experiment at night when the cell is idle. We use the three phones as three competing users, each setting up a connection with a AWS server. The S8, Redmi 8 and MIX3 starts its flow at zero, 10, and 20 seconds, and ends at 60, 50, and 40 seconds, respectively. These three phones share the same primary cell but have different secondary and tertiary cells (if configured), so the primary cell at 1.94 GHz is the shared bottleneck of three connections. We record the allocated PRBs to each user by the primary cell, when three connections are running concurrently. Three connections get identical allocated primary cell PRBs if they achieve a fair-share.

6.4.1 Multi-user fairness. We investigate the fairness between multiple PBE-CC flows with similar propagation delays. We setup three AWS servers in the US and start three current connections via three mobile phones, plotting the allocated bandwidth by the primary cell to the three phones in Figure 21(a). We see that the PBE-CC flows quickly converge to the fair-share of the bottleneck bandwidth. Jain's fairness index [24] is 99.97 and 98.73% with two and three concurrent flows (100% is ideal), respectively. Since we cannot prevent all associated users from using the cellular network, we observe light background traffic generated by a unknown user, in this experiment. The PBE-CC flow also reacts quickly, fairly sharing the bandwidth with background users.

6.4.2 RTT fairness. We investigate whether PBE-CC can guarantee a fair-share of wireless link capacity between multiple flows with



(a) Three PBE-CC flows with similar RTTs. **(b) Three PBE-CC flows with significant RTT differences.** **(c) Two PBE-CC flows coexist with one BBR flow.** **(d) Two PBE-CC flows coexist with one CUBIC flow.**

Figure 21: The allocated PRBs (averaged over 50 subframes) by the primary cell to three mobile phones, when these three mobile phones starts three PBE-CC flows with three AWS servers in US (a); three PBE-CC flows with two AWS servers in US and one AWS server in Singapore (b); two PBE-CC flows with one BBR flow (c); two PBE-CC flows with one CUBIC flow (d).

significant differences in propagation delay. We use three mobile phones to build concurrent connections with three AWS servers: one in Singapore (average RTT of 297 ms) and two in the US (average RTTs of 52 ms and 64 ms). We plot the the primary cell allocated PRBs for these connections in Figure 21(b). We see that the all three PBE-CC flows with significant propagation delay differences obtain similar allocated bandwidths. Jain's fairness indices are 99.74% and 99.45% with two and three concurrent flows, respectively.

6.4.3 TCP friendliness. A common requirement from new congestion control schemes is the capability of fairly sharing the available bandwidth with existing congestion control algorithms like BBR and CUBIC. We investigate the performance of PBE-CC in two cases: two PBE-CC flows coexisting with one BBR flow, and two PBE-CC flows coexisting with one CUBIC flow. Figures 21(c) and 21(d) depict allocated PRBs for three connections in these cases, showing that PBE-CC shares bottleneck bandwidth equally with both CUBIC and BBR flows. Jain's fairness index is 99.96% and 98.52% with two and three concurrent flows in Figure 21(c), and 99.95% and 98.34% with two and three flows in Figure 21(d). The base station fairness policy prevents one user from grabbing all the bandwidth. Though CUBIC and BBR may aggressively increase their sending rate, the base station limits the total bandwidth they can obtain and forces them to share with other concurrent flows.

7 DISCUSSION

Power consumption. In the connected state, a mobile device must keep its radio on and decodes the control channel to check whether the base station has data for it or not in each subframe. Therefore, PBE-CC does not turn the radio of mobile device on for any extra time than necessary currently and thus introduces no additional power costs. The small computational overhead PBE-CC introduces is that the mobile device may need to decode control messages that are not transmitted to it. But, the number of extra control messages inside each subframe the device needs to decode is very small, since our experimental results shows that there are less than 4 control messages inside more than 95% subframes. Furthermore, the control messages are very short (less than 70 bits), so that decoding one message only involves small extra computational overhead.

Packet buffering. PBE-CC works at or very close to the Kleinrock TCP operating point [26, 27], which minimizes buffering, minimizing the delay. In practice, it could be beneficial to buffer some bytes in the base station, which slightly increases delay but helps to immediately utilize increases in connection throughput, before the sender modulates its sending rate (congestion control has at least a round trip time delay). In the future, we plan to extend PBE-CC to enable the sender/app to adaptively adjust the buffering inside the network, trading off increased delay for increased throughput.

Fairness policy. Currently, PBE-CC fairly shares idle bandwidth among all active users in the connection start state. In the future, PBE-CC can be modified to incorporate other fairness policies, *e.g.*, active users with lower physical data rate grab larger bandwidth. PBE-CC's control algorithm adapts to an arbitrary fairness policy, achieving equilibrium in the steady state.

Misreported congestion feedback. PBE-CC relies on the mobile user to report the estimated capacity back to the server so it is possible that a malicious user may report a data rate higher than the network can support, triggering overwhelming number of data being injected into the network, causing catastrophic impact. In future work, PBE-CC can be extended to detect such malicious users via implementing a server side BBR-like throughput estimator, which estimates the currently achieved throughput purely with timestamps of packets being sent and acknowledged, without any involvement of the mobile user. By comparing the achieved throughput and capacity reported by the user, PBE-CC identifies any user who consistently reports a rate higher than the achievable throughput as a malicious user.

8 CONCLUSION

PBE-CC is the first end-to-end congestion control algorithm to seamlessly integrate mobile client-side wireless physical layer capacity measurement into its design, which is crucial for the multi-cell design of 4G and 5G wireless networks. Our rigorous performance evaluation featuring multi-locations, mobility, varying background traffic levels, and varying RTTs shows that PBE-CC outperforms many leading congestion control algorithms in both latency and throughput. PBE-CC is also immediately deployable, with modifications required solely to content servers and mobile clients. This work does not raise any ethical issues.

ACKNOWLEDGEMENTS

We thank the anonymous SIGCOMM reviewers and our shepherd for their valuable feedback that has improved the quality of this paper. This work was supported by NSF grant CNS-1617161.

REFERENCES

[1] 3GPP. 5G specifications. [3gpp.org].
[2] 3GPP. LTE Release 10. [3gpp.org].
[3] 3GPP. TS36.212: Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding.
[4] 3GPP. TS36.213: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures.
[5] T. Anderson, A. Collins, A. Krishnamurthy, J. Zahorjan. PCP: Efficient endpoint congestion control. *USENIX NSDI*, 2006.
[6] V. Arun, H. Balakrishnan. Copa: Practical delay-based congestion control for the internet. *USENIX NSDI*, 2018.
[7] A. Balasingam, M. Bansal, R. Misra, K. Nagaraj, R. Tandra, S. Katti, A. Schulman. Detecting if LTE is the bottleneck with bursttracker. *ACM MobiCom*, 2019.
[8] L. S. Brakmo, S. W. O'Malley, L. L. Peterson. TCP vegas: New techniques for congestion detection and avoidance. *ACM SIGCOMM*, 1994.
[9] N. Bui, J. Widmer. OWL: A reliable online watcher for LTE control channel measurements. *ACM AllThingsCellular*, 2016.
[10] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, V. Jacobson. BBR: Congestion-based congestion control. *ACM Queue*, **14**(5), 2016.
[11] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, M. Schapira. PCC: Re-architecting congestion control for consistent high performance. *USENIX NSDI*, 2015.
[12] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, M. Schapira. PCC Vivace: Online-learning congestion control. *USENIX NSDI*, 2018.
[13] Ettus. USRP B210. [ettus.com].
[14] Ettus. USRP X310. [ettus.com].
[15] S. Floyd, T. Henderson. RFC2582: The NewReno modification to TCP's fast recovery algorithm. *RFC Editor*, 1999.
[16] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, D. J. Leith. srsLTE: An open-source platform for LTE evolution and experimentation. *ACM WiNTECH*, 2016.
[17] P. Goyal, A. Agarwal, R. Netravali, M. Alizadeh, H. Balakrishnan. ABC: A simple explicit congestion control protocol for wireless networks. *USENIX NSDI*, 2019.
[18] P. Goyal, M. Alizadeh, H. Balakrishnan. Rethinking congestion control for cellular networks. *ACM HotNets*, 2017.
[19] S. Ha, I. Rhee, L. Xu. CUBIC: A new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, **42**(5), 2008.
[20] M. Hock, R. Bless, M. Zitterbart. Experimental evaluation of BBR congestion control. *IEEE ICNP*, 2017.
[21] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, O. Spatscheck. An in-depth study of LTE: Effect of network protocol and application behavior on performance. *ACM SIGCOMM*, 2013.
[22] IETF. IETF Mobile Throughput Guidance (MTG) . [ietf.org].
[23] V. Jacobson. Congestion avoidance and control. *ACM SIGCOMM*, 1988.
[24] R. Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, 1990.
[25] D. Katabi, M. Handley, C. Rohrs. Congestion control for high bandwidth-delay product networks. *ACM SIGCOMM*, 2002.
[26] L. Klein-rock. Power and deterministic rules of thumb for probabilistic problems in computer communications. *IEEE ICC*, 1979.
[27] L. Kleinrock. On flow control in computer networks. *IEEE ICC*, 1978.
[28] S. Kumar, E. Hamed, D. Katabi, L. Erran Li. LTE radio analytics made easy and accessible. *ACM SIGCOMM*, 2014.
[29] H. Lee, J. Flinn, B. Tonshal. RAVEN: Improving interactive latency for the connected car. *ACM MobiCom*, 2018.
[30] W. K. Leong, Z. Wang, B. Leong. TCP congestion control beyond bandwidth-delay product for mobile cellular networks. *ACM CoNEXT*, 2017.
[31] Y. Li, C. Peng, Z. Yuan, J. Li, H. Deng, T. Wang. Mobileinsight: Extracting and analyzing cellular network information on smartphones. *ACM MobiCom*, 2016.
[32] F. Lu, H. Du, A. Jain, G. M. Voelker, A. C. Snoeren, A. Terzis. CQIC: Revisiting cross-layer congestion control for cellular networks. *ACM HotMobile*, 2015.
[33] S. Ma, J. Jiang, W. Wang, B. Li. Fairness of congestion-based congestion control: Experimental evaluation and analysis. *arXiv:1706.09115*, 2017.
[34] J. Padhye, V. Firoiu, D. Towsley, J. Kurose. Modeling tcp throughput: A simple model and its empirical validation. *ACM SIGCOMM*, 1988.
[35] S. Park, J. Lee, J. Kim, J. Lee, S. Ha, K. Lee. ExLL: An extremely low-latency congestion control for mobile cellular networks. *ACM CoNEXT*, 2018.
[36] Qualcomm qxdm tool. qualcomm.com.
[37] D. Scholz, B. Jaeger, L. Schwaighofer, D. Raumer, F. Geyer, G. Carle. Towards a deeper understanding of TCP BBR congestion control. *IEEE IFIP Networking*, 2018.
[38] A. Sivaraman, K. Winstein, P. Thaker, H. Balakrishnan. An experimental study of the learnability of congestion control. *ACM SIGCOMM*, 2014.
[39] K. Tan, J. Song, Q. Zhang, M. Sridharan. A compound TCP approach for high-speed and long distance networks. *IEEE INFOCOM*, 2006.
[40] R. Ware, M. K. Mukerjee, S. Seshan, J. Sherry. Modeling BBR's interactions with loss-based congestion control. *ACM IMC*, 2019.
[41] D. X. Wei, C. Jin, S. H. Low, S. Hegde. FAST TCP: Motivation, architecture, algorithms, performance. *IEEE/ACM Transactions on Networking*, 2006.
[42] K. Winstein, H. Balakrishnan. TCP Ex Machina: Computer-generated congestion control. *ACM SIGCOMM*, 2013.
[43] K. Winstein, A. Sivaraman, H. Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. *USENIX NSDI*, 2013.
[44] D. Wischik, C. Raiciu, A. Greenhalgh, M. Handley. Design, implementation and evaluation of congestion control for Multipath TCP. *USENIX NSDI*, 2011.
[45] X. Xie, X. Zhang, S. Kumar, L. E. Li. piStream: Physical layer informed adaptive video streaming over LTE. *ACM MobiCom*, 2015.
[46] X. Xie, X. Zhang, S. Zhu. Accelerating mobile web loading

using cellular link information. *ACM MobiSys*, 2017.

[47] Q. Xu, S. Mehrotra, Z. Mao, J. Li. PROTEUS: Network performance forecast for real-time, interactive mobile applications. *ACM MobiSys*, 2013.

[48] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, K. Winstein. Pantheon: The training ground for internet congestion-control research. *USENIX ATC*, 2018.

[49] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, C. Görg. Adaptive congestion control for unpredictable cellular networks. *ACM SIGCOMM*, 2015.