CQIC: Revisiting Cross-Layer Congestion Control for Cellular Networks

Feng Lu[†§], Hao Du[§], Ankur Jain[§], Geoffrey M. Voelker[†], Alex C. Snoeren[†], and Andreas Terzis[§]

[†] UC San Diego [§] Google Inc.

Abstract

With the advent of high-speed cellular access and the overwhelming popularity of smartphones, a large percent of today's Internet content is being delivered via cellular links. Due to the nature of long-range wireless signal propagation, the capacity of the last hop cellular link can vary by orders of magnitude within a short period of time (e.g., a few seconds). Unfortunately, TCP does not perform well in such fast-changing environments, potentially leading to poor spectrum utilization and high end-to-end packet delay.

In this paper we revisit seminal work in cross-layer optimization in the context of 4G cellular networks. Specifically, we leverage the rich physical layer information exchanged between base stations (NodeB) and mobile phones (UE) to predict the capacity of the underlying cellular link, and propose CQIC, a cross-layer congestion control design. Experiments on real cellular networks confirm that our capacity estimation method is both accurate and precise. A CQIC sender uses these capacity estimates to adjust its packet sending behavior. Our preliminary evaluation reveals that CQIC improves throughput over TCP by 1.08–2.89× for small and medium flows. For large flows, CQIC attains throughput comparable to TCP while reducing the average RTT by 2.38–2.65×.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design

General Terms

Algorithms, Design, Experimentation, Measurement, Performance

Keywords

Congestion Control; Cellular Networks; HSPA+; Cross-Layer

1. INTRODUCTION

Smartphones and other hand-held wireless devices are increasingly popular platforms for all types of network applications. Fueled by attractive pricing models for cellular data access, mobile data traffic is increasing at an explosive rate, with 10× growth predicted in the next five years [12]. To meet this unprecedented demand, cellular carriers have actively bid for additional spectrum, deployed more cellular towers and base stations, and upgraded to the latest cellular technologies such as HSPA+ [1] and LTE Advanced [2]. These efforts lead to a tremendous cost increment in both CAPEX and OPEX. These investments are not currently being fully exploited, however: cellular operators report that their precious resources are often under-utilized. In a recent measurement study, Huang *et al.* found that, on average, TCP flows use less than 50% of the available bandwidth in deployed LTE networks due to poor protocol interactions [16].

TCP, like any congestion control protocol, strives to match a sender's packet transmission rate with the available bottleneck bandwidth. We observe that for a large fraction of mobile data traffic, the bottleneck is the last hop cellular link. In particular, over 38% of all traffic flows[1] on smartphones originate from large CDNs [11], which are increasingly locating their servers inside the networks of mobile operators to decrease latency and improve user experience. Hence, the performance of this significant portion of the (high-volume) flows in a mobile environment depends upon an accurate estimate of the last hop cellular link capacity.

Given that the capacity of a cellular link fluctuates rapidly over time, prior approaches [10, 20, 23] use per-packet signaling, such as ACKs (losses) and inter-packet spacing, and employ various models to infer the capacity of the underlying cellular channel. Unfortunately, these models are necessarily tied to specific network types and locations. In contrast, we observe that high-fidelity information about the cellular channel is readily available from the radio-layer signaling protocols employed by high-speed cellular networks (e.g., HSPA+ and LTE). Therefore, we propose to dispense with modeling the channel entirely, and instead utilize the existing physical-layer control information—in particular the channel quality indicator (CQI) and discontinuous transmission ratio (DTX)—to predict instantaneous cellular bandwidth. As a proof of concept, we design CQIC a congestion control protocol that employs physical-layer information to control its sending rate. While the benefit of using cross-layer techniques for congestion control in wireless networks has been argued multiple times in seminal papers, including [4, 6, 9], CQIC revisits this approach and suggests that the ever richer physical layer information could potentially change the way how we design congestion control protocols for cellular networks.

Our experiments indicate that CQIC can accurately and precisely estimate the capacity of a cellular link, where the average estimation error is only 8% and the 80th-percentile error is less than 20%. We implement CQIC in Google's QUIC framework [15] and com-

[1]The percentage is even higher for medium and large flows.

HotMobile'15, February 12–13, 2015, Santa Fe, New Mexico, USA.
ACM 978-1-4503-3391-7/15/02.
http://dx.doi.org/10.1145/2699343.2699345.

pare the download performance of CQIC with TCP on real workloads as a preliminary evaluation. CQIC attains nearly 100% of the available bandwidth while keeping the average RTT very close to the target value for all flow sizes. Specifically, for small and medium flows, CQIC outperforms TCP by 1.08–2.89× in terms of throughput while attaining similar RTT. CQIC yields similar (or slightly better) throughput performance on large flows, while reducing the average RTT by 2.38–2.65×. These results augur well for the use of physical-layer information in cellular congestion control protocols.

2. BACKGROUND

The challenges of TCP over wireless links are well-known problems, having been studied for almost twenty years [6, 7]. Much of the early work focused on avoiding misinterpretation of link-layer packet losses by means of explicit packet marking [18] or local retransmission [6]. Today's cellular data technologies conceal link-layer losses from transport protocols by deploying ARQ and error correction techniques, potentially at the cost of large variations in packet delay [10]. Furthermore, due to the nature of wireless signal propagation and channel-state-based scheduling [8], users experience a significant degree of variation in link-layer data rates. This combined delay and rate variability leads to undesirable interactions with TCP (e.g., spurious timeouts, bufferbloat, etc.) and poor bandwidth utilization.

Recent research efforts focus on understanding the impact of delay/rate variability on TCP performance [13] and improving congestion-control protocols in the face of large variations [10, 17, 20, 23]. Khafizov *et al.* [13] study the performance of TCP over IS-2000 networks and find that bandwidth oscillation significantly degrades TCP throughput. Assuming that TCP cannot adapt quickly enough to the delay and bandwidth fluctuations of cellular links, others try to model the variations, either deterministically [10] or dynamically based on stochastic control theory [20] and statistical methods [23]. To shorten the feedback latency, estimated bandwidth is often conveyed back to the sender via side channels such as the receiver window field in the TCP header [20] or custom control protocols [23].

CQIC differs from the aforementioned approaches—and most previous cross-layer designs[2]—by extracting capacity information directly from the physical layer. In general, the existing literature continues to rely on TCP to probe the underlying bandwidth (employing a wide variety of mechanisms to mitigate wireless-link side effects); CQIC forgoes the entire AIMD-style congestion avoidance process and directly obtains bottleneck bandwidth information from the physical layer signaling between the UE and the base station. The benefits are manifold. First, CQIC eliminates the need for a model and the associated uncertainty which degrades protocol performance. For example, model inaccuracies force Sprout to trade throughput for low end-to-end delay, sacrificing 30% or more bandwidth utilization to remain interactive [23]. Second, CQIC does not rely on packet loss signals to adjust its bandwidth estimation and avoids the bufferbloat effect prevalent in TCP-style protocols. Furthermore, the high fidelity of information at the physical layer allows CQIC to closely track the fast-changing cellular bandwidth. Finally, CQIC does not send probe traffic nor does it need a slow-start phase.

Clearly, this approach is only viable when an accurate bandwidth estimate is available from the physical layer and the bandwidth value is sufficient for end-to-end congestion control, i.e., the cellular link is the bottleneck. Fortunately, cellular technology advancements and shorter connection hop-count are both the prevailing technological trends.

[2]Both Srivastava *et al.* [21] and Shakkottai *et al.* [22] provide excellent summaries of this topic.

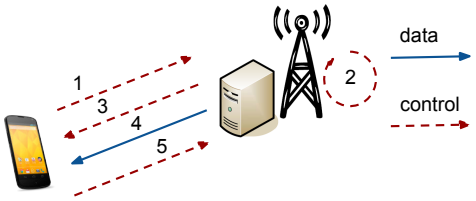


Figure 1: HSPA+ downlink data transmission.

Tight cellular control loop. The recent dramatic increase in cellular data rates is due to a combination of sophisticated new communication techniques and wider frequency spectrum [3]. In particular, advanced link technologies, such as MIMO, antenna arrays, etc., require tight control interactions between the base station and the UE. Such interactions often happen on the scale of milliseconds and a wide variety of channel information is exchanged between the two entities [1, 2]. As a result, a modern UE continuously reports the current channel quality, even when there is no network data activity. The rich information contained within these control channel messages allows CQIC to directly compute the cellular channel capacity (as we describe in the following section). Furthermore, these control messages are part of the cellular standards (e.g., HSPA+ and LTE), and readily available from the physical layer. Last but not least, CQIC does not require any changes in existing network infrastructure. Only the UE's radio firmware needs to be updated to expose control information to the upper layers.

Server proximity. The effectiveness of a congestion control protocol depends on how well it tracks the bottleneck bandwidth. Even if CQIC can accurately track available capacity on the wireless hop, end-to-end throughput might be constrained by other bottlenecks in the network. Fortunately, CDNs customarily place their servers inside the networks of mobile operators to improve application quality of experience (QoE). In general, the average distance between a UE and a CDN server is significantly shorter than between standard Internet hosts.[3] Given that medium and large flows, such as video streaming and software updates, are likely to be served by CDNs (which also constitute the majority of mobile traffic in terms of volume [11]), we believe that the end-to-end bandwidth for most cellular connections is bottlenecked by the cellular last hop. For this reason, CQIC focuses on estimating the cellular link capacity without worrying about bottlenecks elsewhere in the network.

3. CELLULAR CAPACITY ESTIMATION

In this section, we start by explaining the preliminaries of data and control exchange between base stations and mobile terminals in current cellular networks. With a basic understanding of this process, we then illustrate how CQIC leverages existing control information to estimate channel capacity. Finally, we evaluate the accuracy of CQIC's capacity estimation mechanism based on real network measurements.

3.1 HSPA+ Basics

We focus our discussion on HSPA+ networks due to their widespread availability. The control sequences described here are similar to the ones in LTE networks. In addition, we focus on downlink transmissions as cellular traffic is heavily skewed towards

[3]See, e.g., `http://www.akamai.com/hdwp` p.3.

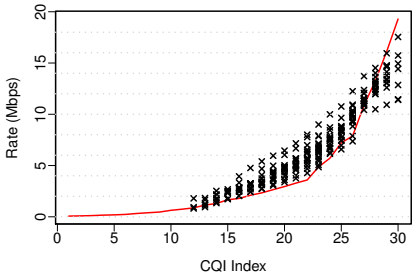


Figure 2: Actual mapping between CQI and data rate (solid line represents the CQI-Rate mapping defined by the HSPA+ specification).

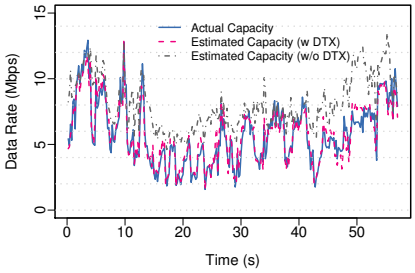


Figure 3: Capacity estimation based on CQI alone (estimated capacity w/o DTX), and based on both CQI and DTX (estimated capacity w/ DTX).

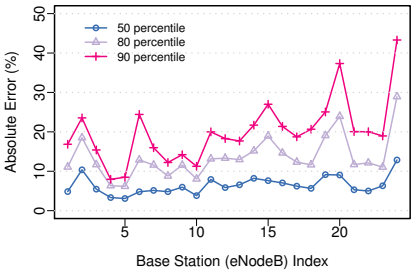


Figure 4: The 50-, 80-, and 90-th percentile of estimation error with 200 ms window size across the 24 experiment instances.

downlink; uplink estimation is straightforward as the transmission rates are known by the UE itself.

Figure 1 illustrates the sequence of control and data messages that the UE exchanges with the base station, once activated. In particular, each UE reports the channel quality (CQI) on the control channel (1); after collecting all such CQI reports from UEs, the base station determines which set of UEs will be served (2); and informs the UEs about its decision (3). The base station then delivers the next frame, which lasts for 2 ms[4] (4). Finally, the UE sends feedback (ACK/NACK) to the base station for the data block just received (5). These messages are continuously exchanged in a pipeline fashion. Even when there is no data transmission, the UE reports CQI to the base station every 2 ms as long as it remains active.

3.2 CQI-to-Rate Mapping

As defined by the HSPA+ specification [1], the base station exclusively relies on CQI to decide the data block size delivered to UEs (step 4 in Figure 1). Specifically, CQI ranges from 0 (worst) to 30 (best) according to the current downlink channel quality. In addition, there exists a one-to-one mapping, known as the CQI mapping table, between CQI and the data block size in the HSPA+ specification. For example, when CQI is 20, the NodeB will send 5,896 bits in the 2-ms data slot, which translates to a rate of 2.95 Mbps.

We validate the conformance of real-world base stations to the official CQI-to-rate mapping table through network measurement. Our validation experiment setup consists of a UDP server (which is located next to a packet gateway inside the cellular carrier's network), a mobile phone, and a host laptop. The mobile phone is connected via USB to the host laptop on which we run Qualcomm's QXDM software [19]. The QXDM tool allows us to capture radio-layer traces including the control messages mentioned in Section 3.1. We let the server send UDP packets to the mobile phone as fast as possible so the base station always has traffic for the phone. For each experiment, we recorded at least 60 seconds of trace data. All together we conducted 24 experiments: 12 static, 8 moving at walking speed, and 4 driving.

Figure 2 shows the actual mapping obtained between CQI and data rate from the 24 experiment instances. We average the number of bits sent by the same base station for a particular CQI value for each experiment, which leads to a single CQI-Rate point shown in Figure 2.[5]

We first observe that most of the mapped rates deviate from the standard data rates as defined by the HSPA+ specification. Base station vendors often choose to implement their own CQI rate mapping table and the specification is merely a recommendation. In addition, we notice that there are variations in terms of mapped data rates for the same CQI across different base stations, although the majority of these data rates are centered around some value for a given CQI (except 29 and 30). We suspect that base stations tend to be more conservative in the high-CQI region (28, 29 and 30), and may select data rates from a wide range that is below the rates defined by the specification. Overall, our results indicate that most of the base stations we measure share similar CQI-to-rate mappings.

Since base station vendors appear to implement their own CQI-to-rate mappings, it is likely that one might encounter base stations with CQI-to-rate mappings that are different from the HSPA+ specification or the ones we have profiled thus far. We foresee a number of ways to handle mapping variations. For example, one could build a large (perhaps cloud-based) key-value store where the key is the base station id and profile all base stations with crowdsourcing. In CQIC, we start with the default CQI-Rate mapping defined in the HSPA+ specification and gradually update the mapping table with each additional observed data rate.

While one can derive a mapping between base station and its CQI-to-rate table, it is not clear that mapping persists over time. Therefore, we repeat the same UDP blasting experiment described earlier at a single base station, which covers a mixture of residential and business areas, during different times of the day and under various mobility patterns. Our experimental results indicate that, at least for the base station we studied, the CQI-rate mapping is consistent and does not change with respect to time (at least over the course of a few days) and mobility variations.

3.3 Link-Capacity Estimation

CQIC employs a simple method to estimate cellular link capacity. At a high level, it uses information about previous data rates to predict the future at sub-second time scales. More specifically, we divide time into T-ms-long windows and collect CQI values (one CQI reading every 2 ms) that fall into the current observation window, i.e., $[0, T]$. For each CQI observed, we obtain the corresponding data rate based on the CQI-to-rate mapping (of the current base station). The average of these rates is then used to predict the link capacity for time $[T, 2T]$. Although CQI (capacity) fluctuates rapidly due to small-scale fading, the average rate is still dominated by large-scale fading effects, which vary less frequently at sub-second time scales. Therefore, with a time window T that captures the large-scale variation, we should be able to estimate capacity effectively.

Figure 3 shows the actual and estimated link capacity based on the CQI-driven mechanism we just described (labeled "estimated

[4]In some versions of HSPA, the time slot could be 10 ms as well.
[5]Not all CQI values are observed in our trace dataset.

capacity w/o DTX"). There is an evident discrepancy between the actual and estimated capacities. The reason is that as cell load increases, independent data channels are quickly exhausted and UEs have to share the same data channel in a TDMA manner.

In other words, during certain time slots, the base station may choose not to serve a particular UE even though there is pending traffic for that UE, which is known as discontinuous transmission (DTX). Hence, when estimating the cellular link capacity, both CQI and DTX should be included in the final estimate. Specifically, the DTX ratio for time window $[0, T]$ is computed based on base station scheduling information (i.e., step 3 in Figure 1), and the final link capacity estimate is the product of the CQI-based rate estimation and the DTX ratio.

Figure 4 depicts the 50-, 80- and 90th-percentile CQIC's estimation error across the 24 experiments. For each experiment, we divide the trace into 200 ms intervals and predict the cellular capacity for each interval based on the previous 200 ms interval. The overall capacity estimation accuracy across the entire trace is consistently within 8% for all 24 experiments. In addition, most of the 50- (80-, 90-th) percentile errors are within 10 (20, 30)% of the actual data rate. Although our estimation method could keep up with most of the large-scale fading trends, abrupt channel changes (CQI or DTX ratio) do happen due to nearby interference, surrounding objects, etc., leading to relatively large inaccuracy for some estimation results. Overall, CQIC's estimation method accurately and precisely predicts the downlink cellular link capacity. The estimation results are similar for 100- and 500-ms time windows.

4. CQIC DESIGN & IMPLEMENTATION

Given the ability to accurately estimate channel capacity directly from the cellular network, we now describe how the CQIC congestion control design incorporates it. In CQIC, the receiver directly estimates the underlying link capacity from physical layer information and sends the capacity as the estimated end-to-end bandwidth to the CQIC sender. On receiving the estimated bandwidth, the sender adjusts its congestion window accordingly.

We build upon Google's QUIC [15] framework, which is a new transport protocol based on UDP, to implement a CQIC prototype. In particular, we reuse the RTT estimation and reliable packet delivery modules in QUIC. We implement the CQIC receiver as a native application running on a Google Nexus 5 smartphone running Android 4.3 as modifications to QUIC. We include the CQIC sender as a new QUIC congestion control module on one of Google's experimental servers.

4.1 CQIC Sender

The CQIC sender combines rate-based and window-based congestion control. Specifically, the rate-limiting element decides the inter-packet interval T_p, given the predicted bandwidth B received from the CQIC receiver:

$$T_p = P/B$$

where P is the packet size. When the predicted bandwidth is accurate, the sender simply injects one packet every T_p seconds. In reality, as we show in Section 3, the predicted link capacity could deviate from the actual capacity. Overestimating available bandwidth fills up the buffer at the base station, leading to bufferbloat and packet losses. To prevent this problem, we complement the rate limit with a congestion window that bounds the total number of un-acked bytes the sender can send. Then, a CQIC sender operates in either the rate-limited or the window-limited mode. When the number of un-acked bytes is smaller than the congestion window size, the CQIC sender stays in the rate-limited mode and sends a packet every T_p seconds. Otherwise, the sender injects a packet when it receives an acknowledgment.

Ideally, we want to set the congestion window as small as possible yet large enough to fill up the network pipe, i.e., we want to set the congestion window to the product of the minimum round trip time (RTT_{min}) and the end-to-end bandwidth. We reuse the RTT estimation module contained in QUIC to obtain RTT_{min}. In contrast to TCP, the CQIC sender does not probe the available bandwidth following the AIMD mechanism. Rather, it directly obtains the bandwidth estimation from the CQIC receiver. To accommodate bandwidth estimation errors, we currently set the congestion window size to the product of the predicted bandwidth and twice the RTT_{min}, i.e., $CW = B * 2RTT_{min}$ (this threshold has worked well with our experiments so far, but remains a point of continued investigation). Finally, we employ a token-bucket approach to approximate the ideal rate-limiting value in our implementation.

4.2 CQIC Receiver

The CQIC receiver continuously updates its estimation of the current cellular bandwidth, as described in Section 3.3, and sends the estimated bandwidth to the CQIC sender. Although both CQI and DTX are part of the HSPA+ specification, they are not currently exposed to the operating system of commodity smartphones.

In our prototype implementation we use Qualcomm's QXDM software [19] to obtain the CQI and DTX information from the UE. We configure the UE in diagnostic mode, and the QXDM tool continuously queries and collects various radio and chip-level information. In particular, we use the `HS-DPCCH-INFO` and `HS-DECODE-STATUS` log packets to retrieve the CQI and DTX information, respectively. The QXDM tool reports the `HS-DPCCH-INFO` and `HS-DECODE-STATUS` values every 8 ms and 2 ms, respectively, which enables the CQIC receiver to estimate the cellular bandwidth in real time. In our prototype, the CQIC receiver currently predicts the bandwidth for the next 500 ms time interval (a latency artifact of our current prototype setup), and reports its bandwidth estimate to the sender every 500 ms as well.

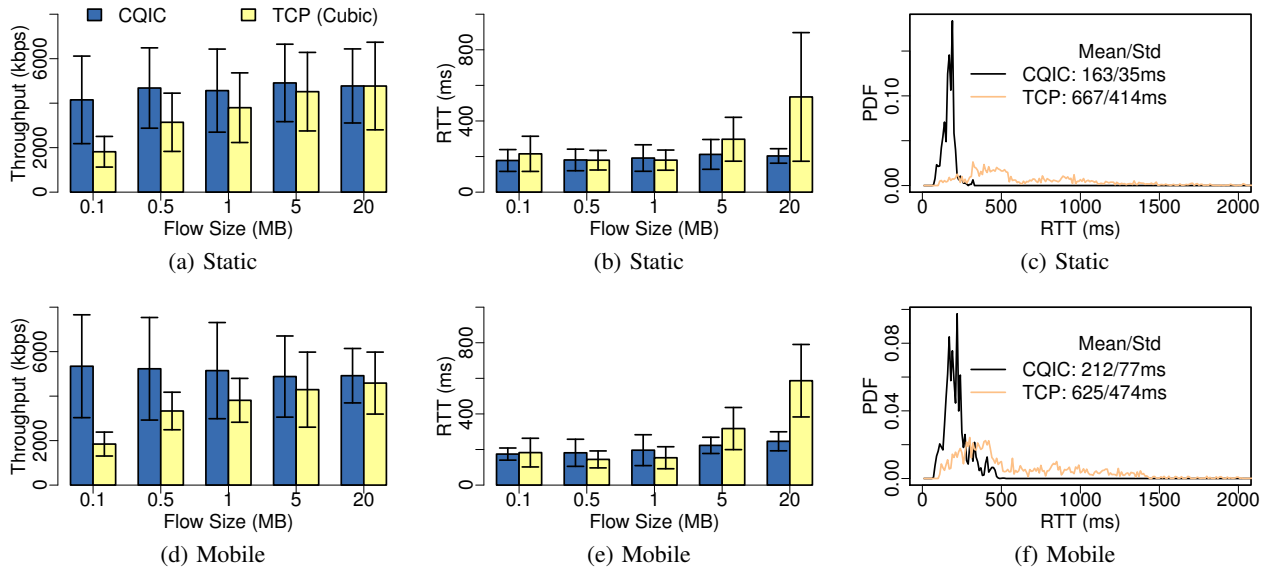
5. EVALUATION

We evaluate the throughput and delay benefits of CQIC using the Google Nexus device to download content from a Google server via a popular cellular network provider. Reflecting a common CDN scenario, this server is located near the network of the mobile carrier such that the cellular channel is the bottleneck link.

We compare the download performance of CQIC and TCP (CUBIC)[6] in terms of throughput and RTT experienced in deployed settings. As a first step and proof of concept, similar to Sprout [23], we only consider a single flow. By varying the retrieved object size from 0.1 to 20 MB, we simulate small, medium, and large flows. We repeat our experiments in static and mobile environments (driving around a local community) to create different channel dynamics. The evaluation helps us understand the effectiveness of CQIC in terms of bandwidth utilization and reaction to channel dynamics. The measured round trip time (RTT) of the underlying path between the UE and the server is about 70 ms.

For each object size, we use CQIC and TCP back-to-back to download an object in the hope that the channel conditions are similar for the two congestion control mechanisms. We then calculate the throughput and mean RTT for that flow. Altogether, we conduct 15 such experiments (CQI range: 6-27) for the static environment, and 10 for the mobile driving environment (CQI range: 1-30). Fig-

[6]TCP (CUBIC) yields higher throughput than Sprout [23].



(a) Static (b) Static (c) Static

(d) Mobile (e) Mobile (f) Mobile

Figure 5: Performance of CQIC and TCP under static (mobile) channel conditions in terms of throughput and average RTT. The results are the average of 15 (10) experiment runs and the error bars show the standard deviations. Also shown are PDFs of the RTTs observed for a randomly chosen large flow.

ure 5 presents the average throughput, RTT, throughput breakdown, and the probability distribution function (PDF) of RTT experienced by a random large flow, under CQIC and TCP, respectively. The overall trends are very similar for both static and mobile cases.

Throughput. As shown in Figures 5(a) and 5(d), given that CQIC estimates the cellular bandwidth directly from the physical layer when the connection opens, the average throughput for small flows (i.e., less than 1MB) under CQIC is 1.5–2.9× larger than that of TCP, which initially spends time in the slow start phase. As flow size increases (e.g., 1 or 5MB), the TCP congestion window ramps up and the impact of slow start fades away. As a result, the throughput of CQIC is only marginally better than TCP (1.1–1.4×). Finally, for large flows (e.g., 20MB) CQIC and TCP attain almost identical throughput. Using `getsockopt()`, we observe that the TCP congestion window rarely decreases for the entire flow duration across all experiments in the static setting, which suggests that there are very few packet losses due to the relatively stable channel conditions and large buffer at the base station. Under the mobile setting, the throughput of CQIC is slightly higher than that of TCP for large flows. We observed that there are more packet losses in the mobile environment due to channel dynamics. These packet losses cause TCP to reduce its congestion window, even potentially time out, slightly reducing effective throughput.

Note that the large standard deviations in throughput in Figures 5(a) and 5(d) are due to different channel configurations (i.e., single vs. dual channels) experienced while conducting the experiments; downloading a 5-MB object in one run might primarily just use a single channel, while in another run it might use two. If we separate the large flows based on the number of channels used, the standard deviation is reduced substantially.

Delay. Precisely because of the large buffer deployed inside the base station, TCP attains similar throughput performance compared to CQIC for large flows. However, TCP's throughput is at the expense of high RTT as shown in Figures 5(b) and 5(e). As flow size decreases, especially for small flows (e.g., 0.1/0.5MB) where the buffer is not completely filled, the RTT reduces accordingly.[7] In contrast, the RTT experienced by CQIC is roughly consistent across all flow sizes. Unlike TCP where the congestion window is a function of (the absence of) packet losses, CQIC sets its congestion window based on bandwidth estimates directly obtained from the physical layer. Hence, CQIC does not induce large buffers in a search for additional capacity, demonstrating another benefit of CQIC's cross-layer design.

Figures 5(c) and 5(f) show the PDF of RTTs experienced by all the packets in a randomly picked large flow under TCP and CQIC in the static and mobile cases, respectively. Since CQIC sets its congestion window to the product of the estimated bandwidth and twice RTT_{min}, it is not surprising to see that the majority of the RTTs are centered around $2 \times RTT_{min}$ (140 ms in our environment). Further, the worst-case bandwidth estimation error is also bounded as the PDF drops to zero after 500 ms, limiting the bufferbloat effect in CQIC. Finally, Figures 5(c) and 5(f) also illustrate the bufferbloat effect in large TCP flows, reflected in the long tail after 500 ms.

Deployment. Obviously, deploying CQIC requires changes to both the sender and receiver. Although both CQI and DTX are part of HSPA+ and LTE specifications, they are not currently visible to higher layers on most popular UEs. The UE radio firmware needs to be updated—possibly through an over-the-air (OTA) programming update with support from mobile chip makers and operators—to expose this information. In contrast, the server side changes of CQIC are straightforward as it only involves a software upgrade.

Fairness. It is important to recall that cellular base stations decide how to share the wireless channel among UEs (mobile devices). Transport protocols like CQIC can only control how each UE individually uses its own allocation. Moreover, CQIC assumes that the server is close to the radio access network and that the end-to-end connection is bottlenecked by the last-hop cellular link. In such a scenario, CQIC tries to match the the achievable bandwidth

[7]We note that the RTT for 0.1MB flows is larger than for larger size flows (e.g., 0.5 or 1MB). This behavior is due to radio state transition delays.

with the channel capacity offered by the cellular base station for the UE. In other words, the fairness among CQIC and TCP flows to different UEs are provided by the underlying scheduling algorithm implemented at the base station. For multiple flows destined to the same UE, we envision something like Congestion Manager [5] or SST [14] to manage fair bandwidth allocation among these flows. When congestion happens elsewhere in the network, CQIC needs to deploy additional mechanisms, for example falling back to TCP-style congestion control, to be TCP-friendly.

Summary. By directly estimating the underlying capacity from the physical layer, CQIC does not need to probe for available resources and thus eliminates the slow start phase completely. Furthermore, CQIC does not rely on packet-loss signals to adjust its capacity estimation and avoids the bufferbloat issue with TCP-like congestion control mechanisms. Finally, the high fidelity of CQI/DTX information at ms granularity, readily available from the physical layer, allows CQIC to closely track radio channel dynamics and promptly adjust its congestion window in response.

6. CONCLUSION

Revisiting cross-layer concepts that were explored at the dawn of mobile computing, we propose CQIC, a cross-layer congestion control design for cellular networks. CQIC directly estimates the channel capacity based on physical layer information (i.e., CQI and DTX) which are part of the HSPA+ and LTE specifications. Further, CQI and DTX information are readily available from the radio, simplifying CQIC deployment. Our preliminary evaluation, focusing on download performance in cellular networks, shows that when the last hop cellular link is the bottleneck, CQIC can outperform TCP in terms of throughput under both static and mobile environments. Moreover, CQIC attains consistently low RTT values across a range of flow sizes, avoiding the high end-to-end delay that large TCP flows experience. These initial results motivate the further exploration of congestion-control techniques that leverage physical-layer channel capacity estimates. Moving forward, we plan to study CQIC's performance under a much wider range of conditions, including interactive and streaming workloads, multiple flows, and a wider variety of network conditions. In particular, it will be critical to identify flows with bottlenecks elsewhere in the network and fall back to more traditional approaches.

Acknowledgement

We would like to thank Suman Banerjee, our shepherd, and the anonymous reviewers for their detailed feedback.

7. REFERENCES

[1] 3GPP. High Speed Packet Data Access.

[2] 3GPP. LTE Advanced.

[3] ASTELY, A., DAHLMAN, E., FURUSKAR, A., JADING, Y., LINDSTROM, M., AND PARKVALL, S. LTE: the Evolution of Mobile Broadband. *IEEE Communications Magazine 47*, 4 (2009), 44–51.

[4] BAKRE, A., AND BADRINATH, B. I-TCP: Indirect TCP for Mobile Hosts. In *Proceedings of ICDCS* (1995).

[5] BALAKRISHNAN, H., RAHUL, H. S., AND SESHAN, S. An Integrated Congestion Management Architecture for Internet Hosts. In *Proceedings of ACM SIGCOMM* (1999).

[6] BALAKRISHNAN, H., SESHAN, S., AMIR, E., AND KATZ, R. H. Improving tci/ip performance over wireless networks. In *Proceedings of ACM MobiCom* (1995).

[7] BALAKRISHNAN, HARI AND PADMANABHAN, VENKATA N. AND SESHAN, SRINIVASAN AND KATZ, RANDY H. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. *IEEE/ACM Transactions on Networking 5*, 6 (1997), 756–769.

[8] BENDER, P., BLACK, P., GROB, M., PADOVANI, R., SINDHUSHAYANA, N., AND VITERBI, A. CDMA/HDR: a Bandwidth-Efficient High Speed Wireless Data Service for Nomadic Users. *IEEE Communications Magazine 38*, 7 (2000), 70–77.

[9] BROWN, K., AND SINGH, S. M-TCP: TCP for Mobile Cellular Networks. *SIGCOMM Computer Communication Review 27*, 5 (1997), 19–43.

[10] CHAN, M. C., AND RAMJEE, R. TCP/IP Performance over 3G Wireless Links with Rate and Delay Variation. In *Proceedings of ACM MobiCom* (2002).

[11] CHEN, X., JIN, R., SUH, K., WANG, B., AND WEI, W. Network Performance of Smart Mobile Handhelds in a University Campus WiFi Network. In *Proceedings of ACM IMC* (2012).

[12] ERICSSON. Ericsson Mobility Report. `http://www.ericsson.com/res/docs/2014/ericsson-mobility-report-june-2014.pdf`.

[13] FARID KHAFIZOV AND MEHMET YAVUZ. Running TCP over IS-2000. In *Proceedings of IEEE ICC* (2002).

[14] FORD, B. Structured Streams: A New Transport Abstraction. In *Proceedings of ACM SIGCOMM* (2007).

[15] GOOGLE. Quick UDP Internet Connections. `https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34`.

[16] HUANG, J., QIAN, F., GUO, Y., ZHOU, Y., XU, Q., MAO, Z. M., SEN, S., AND SPATSCHECK, O. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. In *Proceedings of ACM SIGCOMM* (2013).

[17] JIANG, H., WANG, Y., LEE, K., AND RHEE, I. Tackling Bufferbloat in 3G/4G Networks. In *Proceedings of ACM IMC* (2012).

[18] KUNNIYUR, S., AND SRIKANT, R. End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks. In *Proceedings of IEEE INFOCOM* (2000).

[19] QUALCOMM. QXDM Professional Qualcomm eXtensible Diagnostic Monitor. `http://goo.gl/ibV7g1`.

[20] REN, F., AND LIN, C. Modeling and Improving TCP Performance over Cellular Link with Variable Bandwidth. *IEEE Transactions on Mobile Computing 10*, 8 (2011).

[21] SHAKKOTTAII, S., AND KARLSSON, P. C. Cross-Layer Design for Wireless Networks. *IEEE Communications Magazine 41*, 10 (2003), 74–80.

[22] SRIVASTAVA, V., AND MOTANI, M. Cross-layer Design: A Survey and the Road Ahead. *IEEE Communications Magazine 43*, 12 (2005), 112–119.

[23] WINSTEIN, K., SIVARAMAN, A., AND BALAKRISHNAN, H. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *Proceedings of USENIX NSDI* (2013).