Tyrus: PHY-Assisted Neural Adaptive Congestion Control for Cellular Networks

Libin Liu
City University of Hong Kong
Hong Kong
libinliu-c@my.cityu.edu.hk

Hong Xu
City University of Hong Kong
Hong Kong
henry.xu@cityu.edu.hk

CCS CONCEPTS

• **Networks** → **Mobile networks**; **Transport protocols**; **Cognitive radios**; • **Computing methodologies** → **Reinforcement learning**.

KEYWORDS

Congestion Control, Physical Layer Information, Reinforcement Learning, Cellular Networks

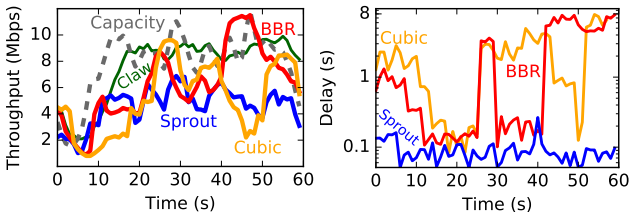
ACM Reference Format:
Libin Liu and Hong Xu. 2019. Tyrus: PHY-Assisted Neural Adaptive Congestion Control for Cellular Networks. In *SIGCOMM '19: ACM SIGCOMM 2019 Conference (SIGCOMM Posters and Demos '19), August 19–23, 2019, Beijing, China.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3342280.3342302

1 INTRODUCTION

Cellular networks have become a dominant mode of Internet access, and cellular traffic continues to explode with 5G on the horizon. This motivates us to re-visit one of the lingering problems in our community, cellular congestion control.

Despite much prior work [5–7], congestion control (CC) in cellular networks still suffers from performance issues. It is well-known that general CC schemes do not work well in cellular networks because the congestion signals they rely on, packet loss or delay, cannot distinguish the inherent variations of the wireless channel from the actual network congestion events. A few lost packets or some RTT variations can easily cause TCP to drastically reduce the window size and significantly under-utilize the channel [6, 7], which is demonstrated in Figure 1 using BBR and Cubic as examples.

Many CC designs have been proposed specifically for cellular networks. They usually tackle particular facets of the cellular wireless scenario with specific tradeoffs and come short of improving the overall performance. Sprout [5] for example achieves low latency for interactive applications by proactively inferring the dynamics of network path based on packet arrival times. As shown in Figure 1b it does reduce the queueing delay consistently, but its throughput is often sacrificed as a result compared to BBR and Cubic.



(a) Throughput **(b) Delay**

Figure 1: We use Mahimahi [3] to emulate the cellular network conditions by replaying a Verizon LTE driving trace [5] between a client and server. The client downloads a 100MB file from the server with different CC schemes.

We believe the fundamental limitation of the above work is that they, as transport layer designs, are effectively blind to the state of the wireless channel at the physical (PHY) layer. While upper-layer metrics such as packet arrival times provide hints about the wireless channel, they are hints at best. If the PHY layer of a mobile device can reveal the complete wireless channel information, such as the amount of channel resources allocated by the base station, we would be able to know fairly accurately the capacity of the link and the CC protocol can function much more efficiently.

We thus carry out a preliminary exploration of the feasibility and potential of this design principle. The downlink capacity of a user equipment (UE) is dictated by the base station (BS) which periodically signals its resource allocation decisions to contending UEs on the physical downlink control channel (PDCCH) [1, 6]. A BS in LTE allocates radio resources in units of both time and frequency called *resource blocks* (RBs). An RB is the smallest allocable resource unit spanning over 7 time-domain OFDM symbols (66.7*us* for one symbol) and 12 frequency-domain OFDM subcarriers (15kHz for one subcarrier) in general. PDCCH carries the allocated RBs and the associated *modulation and coding scheme* (MCS) to a UE. Based on the mapping rules specified in the LTE standard [1], the UE maps the MCS and the number of RBs to the *transport block size* (TBS). TBS exactly specifies the number of bytes that can be carried on the downlink in this scheduling period (1ms usually).

Therefore the key challenge is that, how to leverage the PHY layer information at the UE for congestion control at the remote server side? There are two important problems here. First, the server only sees past PHY information because of the wide-area delay, and needs to predict the future available bandwidth at RTT time scale at the UE. Second, it also needs to identify a control policy for congestion control based on the prediction results, taking into account factors such as other users sharing the Internet [2]. Some recent work uses time average over $O(100)$ms periods for

SIGCOMM Posters and Demos '19, August 19–23, 2019, Beijing, China

ACM ISBN 978-1-4503-6886-5/19/08…$15.00
https://doi.org/10.1145/3342280.3342302

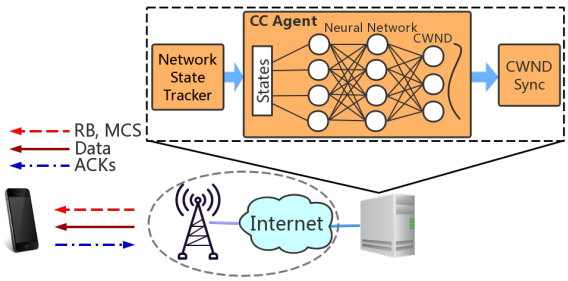
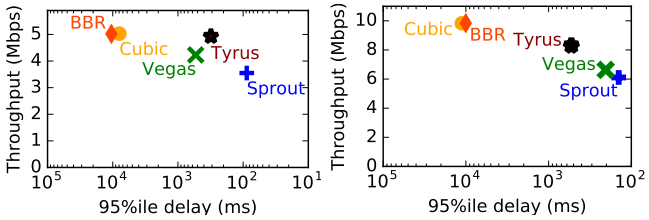


Figure 2: Tyrus overview



(a) Stationary **(b) Moving**

Figure 3: Throughput and delay of each protocol over two scenarios. Better results are up and to the right.

bandwidth prediction and fixed handcrafted rules for rate control [6]. Clearly these techniques are too simplified to deliver optimal performance: as shown in Figure 1a the throughput of CLAW [6] has a clear gap with the channel capacity.

We propose to adopt deep reinforcement learning (RL) to solve these two problems as a whole. RL is a basic machine learning paradigm where an agent learns the optimal control policy that maps the input state to the output actions by continuously observing the rewards of its actions from the environment [4]. It does not assume knowledge of an exact mathematical model of the interactions between the agent and the environment, making it a promising approach that autonomously adjusts cellular congestion control based on a multitude of dynamic signals from both the UE's PHY layer and the Internet.

2 DESIGN

Figure 2 shows the overview of our design called Tyrus for PHY-assisted RL-based cellular congestion control. The TCP client, i.e. the UE, receives from the BS control messages about the allocated RBs and MCS every millisecond and data packets in subframes each lasting 1ms as discussed in §1. It piggybacks the TBS values for each subframe to the server as one of the input signals about the downlink capacity for congestion control. Either ACKs or a dedicated UDP connection can be used for piggybacking. The server collects network states including the TBS signal as input to a RL agent, which runs a deep neural network to determine actions that adjust the congestion window, i.e., the number of packets that can be sent.

Input states. Network states are updated by the network state tracker in Tyrus. The following states are used:

(1) an exponentially-weighted moving average (EWMA) of TBS values obtained from the client feedback.
(2) an EWMA of ΔRTT measured as the difference between the current RTT and the minimum RTT observed during the current connection.
(3) an EWMA of the receiving rate, defined as the number of bytes received since the ACK preceding the transmission of the most recently ACKed packet, divided by the corresponding duration.
(4) current congestion window size.
(5) the previous action taken.

Neural Network. We use a small-scale fully connected neural network as the RL model. We experiment with different numbers of hidden layers and neurons in each layer, and choose three hidden layers each with 16 neurons and ReLU as the activation function.

Reward Function. We train our model using a linear reward function: $\alpha * throughput - \beta * latency - \gamma * 10^8 * loss$, where throughput is measured in Mbps, latency in milliseconds, and loss is the proportion of packets lost between 0 and 1. All of them are taken within one RTT. We can adjust the values of α, β, and γ to optimize for various applications. They are set to 1 by default.

Action Space. An action is defined as a change in congestion window size W_t: $W_t = W_{t-1} * (1 \pm \sigma)$, where $\sigma = 0, 5\%, 10\%, \ldots, 100\%$.

3 EVALUATION

We develop a simple prototype of Tyrus. The server runs QUIC with the RL running on top. We use Mahimahi and feed six different cellular network traces to the prototype to train our RL model. The historical link capacity information is available in these traces. We compare Tyrus against BBR, Cubic, Vegas, and Sprout [5]. Figure 3 shows the throughput and 95%ile delay results in stationary and moving scenarios with another two traces. Tyrus demonstrates competitive throughput compared to BBR, Cubic, and Vegas, and is consistently better than Sprout; its delay performance is also much better than BBR, Cubic, and Vegas. In sum, Tyrus is the performance winner. Because BBR and Cubic are insensitive to queue buildup, they tend to send at a high rate which causes self-inflicted congestion and hence large RTTs. On the other hand, Vegas often overreacts to delay spikes, resulting in extremely small queues and leaving the bandwidth underutilized, and Sprout has the same problem due to significantly underestimating the network capacity for strict packet latency guarantees. In contrast, Tyrus can predict the variations of cellular link capacity in time by utilizing the PHY layer information and adjust congestion window size adaptively with the guidance of trained RL policies.

Though not yet included in the Tyrus prototype, it is feasible to expose PHY layer information at the mobile device through the diagnostic interface available on most phones with Qualcomm chipsets. The output logs can be decoded only by Qualcomm's QCAT software tool, but one can reverse-engineer the log format to achieve real-time decoding [6].

4 ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their valuable comments on this paper. This work was supported in part by the Hong Kong RGC GRF-11210818 and 11216317 projects.

REFERENCES

[1] 3GPP. 2015. LTE: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer Procedures. In *TS 36.213 version 12.4.0 Release 12*.

[2] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, P. Brighten Godfrey, and Michael Schapira. 2018. PCC Vivace: Online-Learning Congestion Control. In *Proc. USENIX NSDI*.

[3] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate Record-and-Replay for HTTP. In *Proc. USENIX ATC*.

[4] Richard S. Sutton and Andrew G. Barto. 1998. Introduction to Reinforcement Learning.

[5] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. 2013. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *Proc. USENIX NSDI*.

[6] Xiufeng Xie, Xinyu Zhang, and Shilin Zhu. 2017. Accelerating Mobile Web Loading Using Cellular Link Information. In *Proc. ACM MobiSys*.

[7] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. 2015. Adaptive Congestion Control for Unpredictable Cellular Networks. In *Proc. ACM SIGCOMM*.