Quantifying the Transient Performance of Congestion Control Algorithms

Yixin Shen[1,3], Zili Meng[1,3], Jing Chen[1,3], Mingwei Xu[1,2,3]
[1]Institute for Network Sciences and Cyberspace, [2]Department of Computer Science and Technology, Tsinghua University
[3]Beijing National Research Center for Information Science and Technology (BNRist)

CCS CONCEPTS

• **Networks** → **Network performance analysis**.

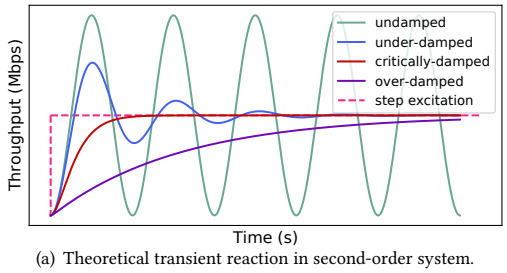
ACM Reference Format:
Yixin Shen, Zili Meng, Jing Chen, Mingwei Xu. 2021. Quantifying the Transient Performance of Congestion Control Algorithms. In *SIGCOMM '21 Poster and Demo Sessions (SIGCOMM '21 Demos and Posters), August 23–27, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3472716.3472861

1 INTRODUCTION

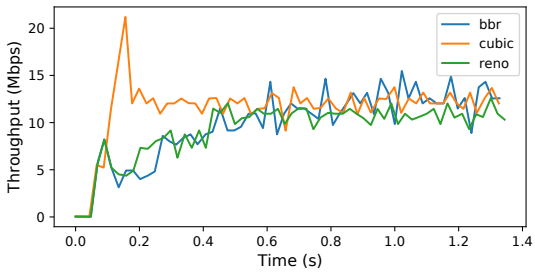
A plethora of Congestion Control Algorithms (CCA) [1–4] have emerged for decades along with ever-evolving network technologies and rapidly changing application requirements. To quantitatively compare different CCAs, researchers have proposed different metrics from various aspects, e.g., throughput and delay for performance, Jain's index [6] for fairness. In practice, these metrics are mostly measured in a long-term manner (e.g., everyday [13]).

However, with the rising demand of the performance from applications, the transient performance of CCAs during network fluctuations is also critical [9]. For example, for low-latency applications such as videoconferencing, several 100ms stalls due to transient network fluctuations could severely degrade the users' experience. Therefore, whether CCAs could responsively converge to the network capacity is critical for users' experience. However, such fluctuations are difficult to be quantified in the long-term measurements of CCAs since it only takes a small portion in the measurement period. In this case, we are motivated to quantify the *convergence ability* of CCAs to provide better understandings for operators.

Yet, it is challenging to quantify the transient reaction, i.e. the convergence ability, of CCAs, which can be defined as the pattern between network status change and the steady state. On one hand, the network condition itself is complex and changeable, making the input of CCA complicated. Inspired by Signal Processing, as the CCA's output will not change without the change of input, we can treat the input as a combination of steps at different moments. On the other hand, due to the complex nature of CCAs (usually piecewise and nonlinear), it is hard to mathematically formulate[1] different CCAs with a unified model. The challenge would be exacerbated with the recently proposed blackbox CCAs based on deep learning [1, 7]. In response, we treat the control system of a CCA as a 2nd-order system, since its input is either the first moment (mean) or the second moment (variance) of network variables and the measurement of higher-order moments in real time could not be accurate enough. According to Statistical Inference, the variance of the estimated expectation of N samples is σ^2/N while the variance of the estimated variance is $2\sigma^4/(N-1)$, therefore much more samples are required (squarely increasing) to bound the error of the second moment to the level of the first moment. Accurate measurement of higher-order moments would require much more samples and is impractical to be collected and referred in real world.



(a) Theoretical transient reaction in second-order system.



(b) Transient reaction in emulation experiments

Figure 1: Transient reaction subjects to damped oscillation.

Based on the above analysis, our key observation is that the transient reaction of a second-order system is damped oscillation in response to a step change. According to Control Theory, the transient reaction would be one of the 4 kinds of damped oscillations (as shown in Figure 1(a)): undamped, under-damped, critically-damped, and over-damped oscillation. As shown in Figure 1(a), when an increase in the available bandwidth occurs, the transient reaction suffers under-damped oscillation (the blue curve) if the CCA reacts too aggressively with overshoot and uses more resource than available, resulting in dramatically occupied buffer, larger queuing delay and even packet loss. On the other hand, if the transient reaction appears to be over-damped oscillation (the purple curve), it means the CCA reacts conservatively to the bandwidth increase, resulting in a waste of bandwidth resources and even unconvergence at the

[1]There are recent efforts trying to formulate CCAs, which remains preliminary and might have performance gaps for quantitative comparisons [8].

The research is supported by the National Key R&D Program of China under Grant 2018YFB1800302, the NSFC under Grant 92038302 and 61625203.

SIGCOMM '21 Demos and Posters, August 23–27, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8629-6/21/08…$15.00
https://doi.org/10.1145/3472716.3472861

next network change. When a CCA's transient reaction is characterized as critically-damped (the red curve), it means the CCA can react to the network condition changes fast with minimal cost and enter the steady state most quickly, which is apparently the optimal case. Hence, we can evaluate a CCA's transient performance by capturing the feature of damped oscillation of the system.

Based on this observation, we propose a framework to give an effective analysis on transient performance based on the damped oscillation hypothesis. To quantify the transient reaction, we measure the gap between the actual oscillation and the critically damped oscillation. However, we are confronted with 2 challenges: (1) How can we detect when the CCA enters the steady state and demarcate the short-term reaction period? (2) How can we quantify the behavior gap? In response, we use a steady-detect window to find when a CCA enters the steady state based on the assumption that a practical CCA should have limited fluctuations in the steady state. To quantify the reaction performance, we use Normalized Area (i.e., the accumulated gap between the theoretical throughput and the experimental throughput before CCA's convergence) as the metric and a smaller area means better performance. Our framework could evaluate how close a CCA is to the optimal (critically damped) transient performance given a network condition. Therefore, with a bunch of CCAs at hand, network operators can adopt our framework to select the most well-behaved CCA (in the sense of transient performance) under different network scenarios.

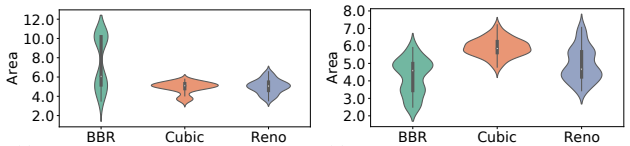
We use the framework to quantify and compare the transient performance of three CCAs: BBR[3], Cubic [4] and Reno [5], giving a preliminary and reasonable evaluation on transient performance.

2 FRAMEWORK

Our framework consists of 2 parts: (1) Steady State Detection to confirm when CCA enters the steady state, and (2) Reaction Quantification to quantify the transient performance.

Steady State Detection. As mentioned in §1, transient reaction is the pattern between network status change and the steady state. Thus, before measuring the reactive tendency, we must confirm when the CCA enters the steady state (the convergence point). The convergence points may vary greatly among different CCAs, so we need to find out the specific convergence point case-by-case. If we determine the convergence point earlier than its actual time, the reaction performance measured will be worse than practice. If we determine the convergence point later than the actual to ensure that CCAs enter the steady state, the short-termed transient reaction could be overwhelmed by the steady-state jitter in the long time scale. The steady-state jitter is random and obscure and differs among CCAs, but might dominate the overall tendency in the large time scale blinding us from the transient performance. Hence, we use a set of steady-detect windows (consisting of several consecutive windows of the same size and uniform spacing) to accurately position the convergence point. When the mean and variance are almost the same among these windows, we consider the CCA has already entered the steady state. We choose three 10-RTT-sized windows, and the beginning of each window is separated by 4-RTT. This does not bring much computation overhead and is also enough to precisely detect the steady state in our experiments.

Reaction Quantification. Transient reaction is captured with random noise, disabling us to fit the curve and derive an accurate mathematical expression of the CCA's throughput for the classification of oscillations. In response, we calculate *Normalized Area* to quantify the transient performance. Area represents the accumulated gap between the throughput of CCAs and the ground-truth available bandwidth from the network change to the convergence point (e.g., for critically damped oscillation, it is the area between the red and pink dashed curve in Figure 1(a)), which is indicative of the oscillation tendency. Obviously, critically damped oscillation has the smallest area and the areas of both under-damped and over-damped largen with deviation of the damping. Additionally, the impact of converging to a non-optimal value is covered by this method. We normalize the area by reaction time and bandwidth change, so the relative area value among different scenarios (e.g., with different bandwidth) is referable. Note that, by taking the Normalized Area as our metric, we make no distinction between under-damped oscillation and over-damped oscillation, simply examining the gap between them and the optimal situation.



(a) Areas with a 40 MTU-sized buffer. (b) Areas with a 100 MTU-sized buffer.

Figure 2: Areas of 3 CCAs under a step change of link-capacity from 0 to 12Mbps and 50ms RTT with different buffer size. 1 MTU=1500 bytes.

3 EXPERIMENTS AND FUTURE WORK

We implement our framework[2] on the Congestion Control Plane [10], with some of its already deployed CCAs[3]: Reno [5], Cubic [4] and BBR [3]. We use Mahimahi [11] to emulate the delay and the bandwidth change, whose noise subjects to 3σ-truncated normal distribution $N(0, \sigma)$. We measure the throughput over the course of 100 60-second experiments under each set of network conditions using Iperf [12]. BDP-related-sized window is used to smooth the raw throughput data recorded at packet-level on milliseconds-scale by Mahimahi. The result in Figure 1(b) shows that the throughput of different CCAs subjects to different damped oscillations.

To illustrate the effectiveness of our framework, we set a step change of link capacity from 0 to 12Mbps and RTT as 50ms with different buffer size as an example. Figure 2(a) shows a small buffer case while Figure 2(b) shows a larger buffer case. From the results, we can compare the transient performance of different CCAs in different cases, which is reasonable and consistent with theory. When the buffer is small, loss-based CCAs(Cubic and Reno) have better reaction performance and BBR has a long tail. This is because loss-based CCAs increase their congestion window more quickly than BBR. When the buffer is larger, loss-based CCAs degrade their performance. The reason of this performance degradation is that the loss-based CCAs tend to fill the bandwidth and bottleneck buffer since they do not lower the sending rate until packet loss. Additionally, the results give us the suggestion when considering the transient performance: we can choose loss-based CCAs with small buffer while choose BBR with large buffer.

In the future, we plan to adopt our framework to learning-based CCAs in the hope of providing more insight on them.

[2] https://github.com/BobAnkh/TPCCA

[3] We use the CCA's official implementations in https://ccp-project.github.io

REFERENCES

[1] Soheil Abbasloo, Chen-Yu Yen, and H Jonathan Chao. 2020. Classic meets modern: A pragmatic learning-based congestion control for the Internet. In *Proc. ACM SIGCOMM*. 632–647.

[2] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical delay-based congestion control for the internet. In *Proc. USENIX NSDI*. 329–342.

[3] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-Based Congestion Control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue* 14, 5 (2016), 20–53.

[4] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS operating systems review* 42, 5 (2008), 64–74.

[5] Janey C Hoe. 1996. Improving the start-up behavior of a congestion control scheme for TCP. *ACM SIGCOMM Computer Communication Review* 26, 4 (1996), 270–280.

[6] Rajendra K Jain, Dah-Ming W Chiu, William R Hawe, et al. 1984. A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA* (1984).

[7] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. 2019. A deep reinforcement learning perspective on internet congestion control. In *Proc. ICML*. PMLR, 3050–3059.

[8] Muhammad Khan, Yasir Zaki, Shiva R. Iyer, Talal Ahamd, Thomas Pötsch, Jay Chen, Anirudh Sivaraman, and Lakshmi Subramanian. 2021. The case for model-driven interpretability of delay-based congestion control protocols. *ACM SIGCOMM Comput. Commun. Rev.* 51, 1 (2021), 18–25.

[9] Shiyu Liu, Ahmad Ghalayini, Mohammad Alizadeh, Balaji Prabhakar, Mendel Rosenblum, and Anirudh Sivaraman. 2021. Breaking the Transience-Equilibrium Nexus: A New Approach to Datacenter Packet Transport. In *Proc. USENIX NSDI*.

[10] Akshay Narayan, Frank Cangialosi, Deepti Raghavan, Prateesh Goyal, Srinivas Narayana, Radhika Mittal, Mohammad Alizadeh, and Hari Balakrishnan. 2018. Restructuring endpoint congestion control. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 30–43.

[11] 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*. 417–429.

[12] Ajay Tirumala. 1999. Iperf: The TCP/UDP bandwidth measurement tool. *http://dast. nlanr. net/Projects/Iperf/* (1999).

[13] Francis Y Yan, Jestin Ma, Greg D Hill, Deepti Raghavan, Riad S Wahby, Philip Levis, and Keith Winstein. 2018. Pantheon: the training ground for Internet congestion-control research. In *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*. 731–743.