PolyCC: Poly-Algorithmic Congestion Control

Shuaipeng Zhu†, Tong Li† , Xinyu Ma† , Yinfeng Zhu§ , Taotao Zhang§ , Senzhen Liu§ , Haiyang Wang‡ , and Ke Xu¶
Renmin University of China† , ByteDance§ , University of Minnesota Duluth‡ , Tsinghua University¶

Abstract

This paper demonstrates PolyCC, a general framework for simultaneous operation of poly-algorithmic congestion control. PolyCC gains benefits from taking advantage of the complementary among already existing congestion controllers.

1

Background

For over 40 years, many different congestion control algorithms (CCAs) have been developed for specific environments, including over 15 CCAs in the Linux kernel alone.

No single CCA can adequately prevail across all environments . To satisfy the increasingly diverse application requirements over highly complex network conditions, learning-based CCAs have gained much attraction recently. However, their exploration-based models may make mistakes or dangerous actions, resulting in poor performance.

2

Motivation

Different CCAs complement each other in achieving high throughput, low delay, or fast convergence.

Recent studies have shown that leveraging the complementarity among existing CCAs holds the potential to consistently achieve high performance across different environments.

However, their implementations concentrate on resolving specific problems or rely on specific system capabilities, lacking a general framework for the simultaneous operation of multiple CCAs.

3

PART 1

Design

Figure 1 illustrates the architecture of PolyCC, which contains sender-side components as we will describe below.

CCA Pool: PolyCC reuses the already existing implementations of CCAs in the protocol stack to build the CCA Pool.

User-Customized Policy: Users can customize the policy on how CCAs collaborate.

Pilot-Copilot Filter: This component selects only one CCA as the pilot and selects one or more CCAs as the copilots.

Congestion Controller Agent: The Congestion Controller Agent enables the protocol stack to run multiple CCAs simultaneously.

Fusion Controller: Considering the computed windows/rates of all copilots, the Fusion Controller computes a calibrated window/rate for the pilot CCA according to a fusion function.

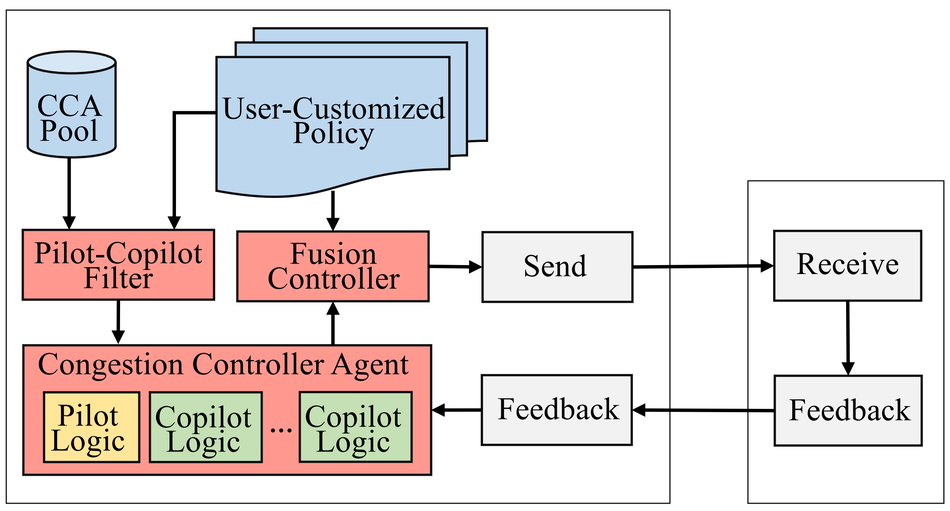

Figure. 1: The PolyCC architecture.

We further give an example of the PolyCC workflow in Figure 2.

- Step 1: According to the user-customized policy, the Pilot-Copilot Filter selects BBR as the pilot, and selects Vegas and CUBIC as the copilots.

4

PART 2

- Step 2: The Congestion Controller Agent runs BBR, Vegas and CUBIC simultaneously and computes their congestion windows (i.e., $cwnd_1$, $cwnd_2$, ..., $cwnd_n$) independently.
- Step 3: The Fusion Controller computes the $\widehat{cwnd}$ according to the fusion function ($f()$), i.e., $\widehat{cwnd} = f(cwnd_1, cwnd_2, \dots, cwnd_n)$. Note that $f()$ is customizable.
- Step 4: The sender updates the windows of all CCAs with $\widehat{cwnd}$, and then conducts congestion control according to BBR.

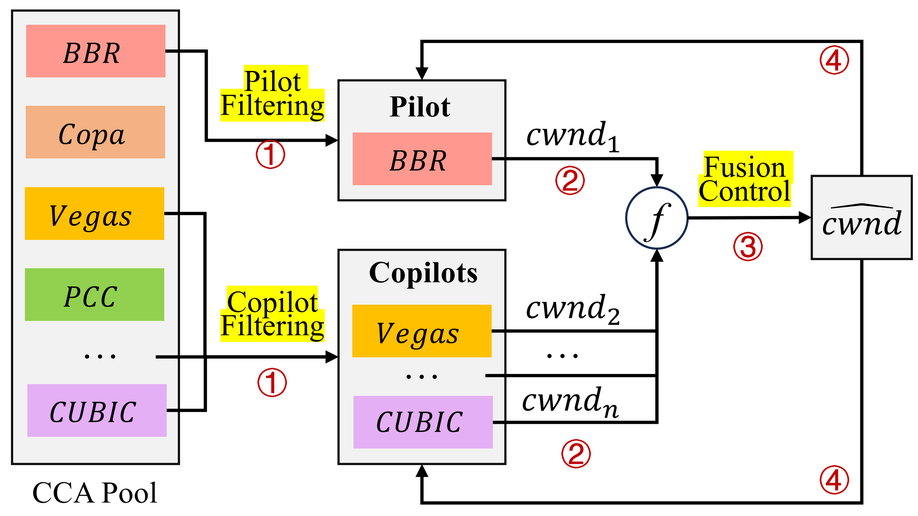

Figure. 2: An example of the PolyCC workflow.

5

Demonstration and Evaluation

We implement PolyCC upon QUIC and showcase two representative scenarios: (i) Employing PolyCC to improve the goodput of BBR in the case of high degrees of aggregation and RTT variance, and (ii) employing PolyCC to accelerate the convergence of BBR in the case of bandwidth change:

- As shown in Figure 3(a), BBR acts as the pilot and CUBIC acts as the copilot. The fusion function in set $\widehat{cwnd} = max\{cwnd_{bbr}, cwnd_{cubic}\}$. When the RTT varience becomes fierce (e.g., $\pm$70ms, $\pm 100ms$), PolyCC achieves 1.82 to 10.35 times of goodput than BBR varients.
- As shown in Figure 3(b), BBR acts as the pilot and Copa acts as the copilot. The fusion function is set $\hat{r} = min\{r_{bbr}, r_{copa}\}$, where r is pacing rate. We start a flow with bandwidth to 10Mbps, at t =5s, we halve the bandwidth to 10 Mbps. The convergence time is reduced by 54.1% than BBRv1, and reduced by 32.8% than BBRv2.

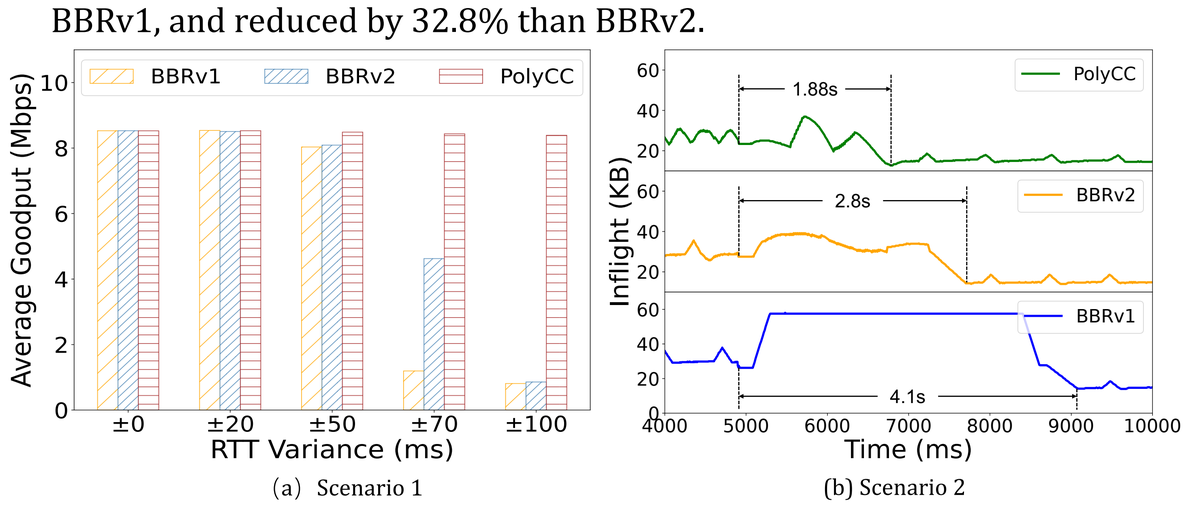

Figure. 3: Testbed results.

6

Conclusion

PolyCC provides a general framework for enabling poly-algorithmic congestion control. For both high degrees of aggregation and RTT variance and bandwidth change scenarios, PolyCC has significant improvements in network performance.

7

REFERENCES

[1] Soheil Abbasloo, Chenyu Yen, and H Jonathan Chao. Classic meets modern: A pragmatic learning-based congestion control for the internet. In SIGCOMM, 2020.
[2] Michael Schapira and Keith Winstein. Congestion-control throwdown. In ACM Hotnets, pages 122–128, 2017.
[3] Bo Wu, Tong Li, Cheng Luo, Changkui Ouyang, Xinle Du, and Fuyu Wang. Autoplex: inter-session multiplexing congestion control for large-scale live videoservices. In SIGCOMM Workshop (NAI), pages 1–6, 2022.
[4] Jianer Zhou, Xinyi Qiu, Zhenyu Li, Gareth Tyson, Qing Li, Jingpu Duan, and Yi Wang. Antelope: A framework for dynamic selection of congestion control algorithms. In ICNP, pages 1–11, 2021.
[5] Zhuoxuan Du, Jiaqi Zheng, Hebin Yu, Lingtao Kong, and Guihai Chen. A unified congestion control framework for diverse application preferences and network conditions. In CoNext, pages 282–296, 2021
[6] Wenzheng Yang, Yan Liu, Chen Tian, Junchen Jiang, and Lingfeng Guo. Gemini: Divide-and-conquer for practical learning-based internet congestion control. In INFOCOM, pages 1–10, 2023.