AutoPlex: Inter-Session Multiplexing Congestion Control for Large-Scale Live Video Services

Bo Wu
Tencent Technologies

Tong Li*
Renmin University of China

Cheng Luo
Tencent Technologies

Changkui Ouyang
Tencent Technologies

Xinle Du
Tsinghua University

Fuyu Wang
Tencent Technologies

ABSTRACT

The increasingly obvious advances in live video services introduce the urgent need for enhancing network transmission performance, especially by designing an efficient congestion control (CC) scheme. Unfortunately, the previous rule-based CC methods cannot adapt well to various network conditions and statuses while machine-learning-powered CC paradigms always suffer from non-trivial system overhead and unstable effects.

In this paper, we first conduct a large-scale network measurement for 800+ million live video streams, and find that QoS metrics of better-performed sessions show similarity in the same user group. We then propose AutoPlex, an inter-session multiplexing CC framework that makes full use of this similarity and automatically adjusts CC parameters (i.e., pacing rate and congestion window size). AutoPlex supports user-defined policies that can act as standards to learn QoS features of better-performed sessions. We implement the proposed AutoPlex prototype based on QUIC protocol and BBR algorithm, and conduct experiments in the real live CDN proxy. The experimental results demonstrate the potentials of AutoPlex for the transmission optimization of live video applications, in which the average (or 90th-percentile) retransmission ratio can be reduced by 24% ∼ 27% (or 32% ∼ 40%) while the average value of goodput/rtt is promoted by 14% ∼ 32%.

CCS CONCEPTS

• **Networks** → **Transport protocols**;

KEYWORDS

Congestion Control; Network Measurement

ACM Reference Format:
Bo Wu, Tong Li, Cheng Luo, Changkui Ouyang, Xinle Du, and Fuyu Wang. 2022. AutoPlex: Inter-Session Multiplexing Congestion Control for Large-Scale Live Video Services. In *ACM SIGCOMM 2022 Workshop on Network-Application Integration (NAI '22), August 22, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3538401.3546596

*Tong Li is the corresponding author (tong.li@ruc.edu.cn).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
NAI '22, August 22, 2022, Amsterdam, Netherlands
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9395-9/22/08…$15.00
https://doi.org/10.1145/3538401.3546596

1 INTRODUCTION

In the past few years, live video services (LVS) such as TikTok Live, YouTube Live, and Twitch have experienced rapid growth, where the weekly coverage of live streaming worldwide has reached 30.4% in Q3 of 2021 [1]. However, the real network conditions vary with regions, resulting in vastly different network QoS among these regions. In particular, poor network QoS has negative impacts on both LVS consumers and CDN vendors. For example, users might complain or stop using some LVS as their deteriorated QoE (e.g., video freezing) while CDN vendors are also unable to accept their cost increases, e.g., caused by higher retransmission ratios.

The existing mechanisms mainly focus on designing, optimizing congestion control (CC) algorithms for enhancing transmission performance of LVS traffic. On the one hand, rule-based CC schemes (e.g., Reno [2], Cubic [3] and BBR [4]) can introduce better performance in some specific network status or services. However, the way they set CC parameters (e.g., maximum pacing rate) as fixed values obviously cannot adapt to different network conditions, which is the main reason for the huge QoS differences between regions. For example, 10Mbps maximum pacing rate might incur lower retransmission ratio in 10Mbps bottleneck bandwidth (BtlBW) than in 1Mbps BtlBW. On the other hand, machine learning (ML) enables more flexible CC parameter configuration for different network conditions [5–7]. However, the way they construct an ML model for each user group (UG) cannot address the issue that a large number of UGs exist in a large-scale network. For example, over 18000 UGs (grouped based on "state-city-ASN" rule) can be obtained in Brazil, in which 27.3% of national LVS sessions will occupy at least 100 ML models to achieve traffic transmission optimization (see §3.2). That is actually unable to be deployed due to the non-trival memory or computation overhead caused by numerous ML models. Therefore, a deployable CC scheme is highly required for adaptive parameter configurations based on different network conditions.

In this paper, we make large-scale network measurements for LVS traffic transmissions, in which we can find the same CC parameter configuration can bring different network QoS performance among various UGs (see §3.3). Besides, we also learn the better-performed sessions with lower retransmission ratios (*retran_ratios*) or larger *goodput/rtt* value[1] have similar QoS features in a same UG (see §3.4). Based on the above measurement and analysis, a novel CC framework named AutoPlex is proposed, which can automatically multiplex QoS metrics of better-performed sessions in the last period and makes adaptive CC parameter configurations for next-period LVS traffic. Take BBR as an example, maximun

[1]The value of *goodput/rtt* is usually leveraged to evaluate the CC performance [8], which is used as a target for LVS transmission optimization.

pacing rate (*max_pacing_rate*) and maximum congestion window (*max_cwnd*) can be configured based on little-changed QoS values such as goodput, maximum inflight size (*max_inflight*) and *max_cwnd* that better-performed LVS sessions introduce in the last period. Compared to rule-based CC methods, AutoPlex enables adaptive parameter configurations for different UGs that has various network conditions without incurring heavyweight ML overhead. Meanwhile, AutoPlex supports user-defined policies to achieve directional optimization for some specific QoS metric, which act as selection criteria to extract better-performed LVS sessions. For example, the QoS metrics of LVS traffic with lower *retran_ratio* can be leveraged to configure CC parameters if the policy is defined to reduce *retran_ratio*. We implement our proposed AutoPlex and partially deploy it in live CDN proxy that is based on QUIC protocol and BBR algorithm for LVS traffic transmission. The experimental results demonstrate the potential of AutoPlex for LVS application. Concretely, the average (or 90th-percentile) *retran_ratio* can be reduced by 24% ~ 27% (or 32% ~ 40%) while the average value of *goodput/rtt* is promoted by by 14% ~ 32%.

The remainder of this paper is organized as follows. §2 introduces the related work about congestion control. §3 introduces the background and motivation. Then, the design details of AutoPlex are depicted in §4, and experimental evaluation is described in §5. Finally, we conclude this article in §6.

2 RELATED WORK

Reno [2] is the most well known scheme that proposes the key concepts of parameters in CC such as congestion window (cwnd), inflight size, initial window, etc. Many TCP variants [3, 4, 9–14] are aimed at specific networks and modify or improve the settings of these parameters, based on their specific insights or assumptions. For example, Vegas [9] sets cwnd based on RTT changes in networks with stable delay. BIC [10] uses binary search to set cwnd, and Cubic [3] uses a cubic function. Verus [11] and Sprout [12] are designed specifically for wireless networks to better fit the frequent changes in bandwidth. BBR [4] estimates the maximum bandwidth to set pacing rate and computes the bandwidth and delay product (bdp) to set cwnd. TACK [15] computes bdp and the minimum RTT to set ACK frequency rather than cwnd in wireless network cases. Although these schemes can improve TCP's adaptability to different network scenarios, for a specific scheme, its underlying logic and way of setting CC parameters remain inflexible.

Machine learning (ML) enables more flexible CC parameter configuration for different network conditions [5–7]. We roughly divide the ML-powered schemes into intra-session learning-based and inter-session learning-based schemes.

Intra-session learning-based schemes learn the network conditions (as well as status) in a same session and adjust the CC parameters in real time. For example, Vivace [16] and PCC [5] adjusts sending rates in real time, and determines the size of the increment according to a performance utility function gradient. RemyCC [17] iteratively searches for a state-action mapping table to maximize an objective function. However, this intra-session estimation still suffer from inflexibility since it relies on the stability and predictability of the underlying network.

Table 1: User group (UG) amount and session ratio with 13.56 million live video traffic in Brazil.

Grouping Rules	UG Num.	Session Ratio		
		Top 10	Top 50	Top 100
State-City-ASN	18620	11.8%	22.2%	27.3%
State-ASN	10326	16.7%	27.7%	32.9%
ASN	8329	30.9%	35.7%	38.3%

In the contrast, inter-session learning-based schemes learn the network conditions across different sessions and adjust CC parameters in the context of the same UG. For example, Indigo [8] uses an LSTM model to train across a wide range of scenarios, guided by reaching the optimal operating point of the network. Orca [18] combines a learning model and rule-based scheme to ensure available recoveries from wrong equilibrium. However, as discussed before, the way they construct an ML model for each UG cannot address the issue that a large number of UGs exist in a large-scale network (see §3.2). In this paper, we propose AutoPlex to improve the CC adaptability while reduces the computation and memory overhead of inter-session learning-based schemes.

3 MOTIVATION

3.1 LVS Demands Flexible CC Schemes

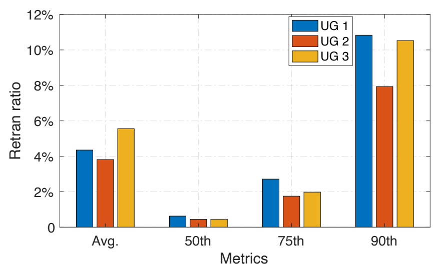
It is well-studied that one-size-fits-all scheme does not exist in CC [19]. Specifically, for the worldwide LVS, covering a wide variety of network conditions in various regions and countries, no single scheme can adequately prevail. Take a famous live platform as an example, the average delay in Malaysia is 86ms, while in Hainan Province of China it is 36ms. Besides, the average available bandwidth in Hainan is 7.3 Mbps for each LVS streams compared to 3.2Mbps in Turkey. Moreover, LVS streams incur average 5.2% packet loss rate in Turkey and 3.8% in Brazil, respectively. For a same region such as Brazil, the average, 50th- and 90th-percentile delays (available bandwidths) are 52ms (3.9Mbps), 27ms (5.4Mbps) and 169ms (15.2Mbps), respectively[2].

As discussed above, rule-based CC schemes (e.g., Reno, CUBIC, Sprout, BBR, etc.) show inflexibility in the underlying logic and way of setting CC parameters. It is expected that different choices of CC parameter settings to fare differently in different contexts (e.g., UGs). Recently, it has been demonstrated that ML-powered schemes show great potential to improve the flexibility of CC [5–8, 17, 18].

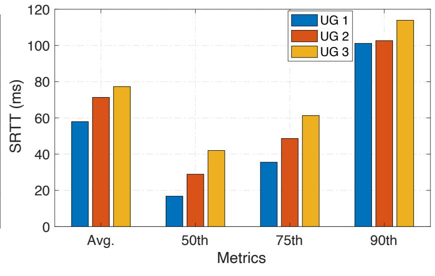
3.2 UG-based and ML-powered Schemes Do Not Scale Well

To pursue a more stable and precise control effect, many existing researches focus on dividing user groups (UGs) based on client's region and ISP/AS number (ASN) information and provide some specific parameters for each UG [20] [6]. However, non-trival memory or computation overhead incurred by ML models can be the bottleneck factor of real deployment [21], especially when the amount of user group becomes very large. Table 1 shows the amount of

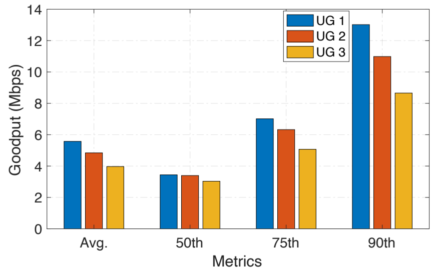
[2]The data mentioned above is all based on our large-scale network measurements for LVS traffic transmission.



(a) Retransmission ratio



(b) Smooth RTT



(c) Goodput

Figure 1: The QoS differences of LVS streams that belong to three different UGs.

UG under different grouping rules in Brazil. For more fine-grained grouping rules (e.g., state-city-ASN), more UGs can be obtained while with the most coarse-grained rule (i.e., ASN)[3], UG number can also reach 8329. Meanwhile, we also count the top 10, 50, and 100 UGs based on their session amounts and find that leveraging 100 ML models for top 100 UGs can only cover 27.3% ~38.3% steams of entire LVS sessions. Compared to their limited benefits, the overhead introduced by ML models is unacceptable in complex networks, especially with a large number of UGs. This reveals that UG-based and ML-powered schemes do not scale well.

3.3 LVS Session Performance Varies in Different UGs

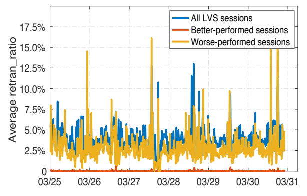
We make large-scale network measurements for 3 months and record session data of 800+ million LVS streams, which contains client IP address and some QoS metrics, e.g., minimum, maximum and smooth RTT (denoted by *min_rtt*, *max_rtt* and *srtt*, respectively), goodput, *max_cwnd*, *max_inflight*, *retran_ratio*, etc. We use "state-city-ASN" as a grouping rule and construct <IP, UG> mapping table to extract QoS values for every UG.

Figure 1 shows the differences of *retran_ratio*, *srtt* and goodput between different UGs, which is depicted from the perspective of average, 50th-, 75th and 90th-percentile values. We can learn LVS sessions in different UGs perform differently in terms of the above three QoS metrics. For example, the LVS sessions of UG 2 result in a smaller and more stable *retran_ratio* while a lower latency (evaluated by *srtt*) is actually introduced in UG 1 (instead of in UG 2 and 3) from the perspective of average, 50th-, 75th- and 90th-percentile values. Meanwhile, UG 3 always holds a smaller goodput than the other two UGs, which might be caused by its higher *retran_ratio* and smaller goodput. Therefore, adaptively configuring a series of CC parameters for each UG will be highly required to meet different network conditions and optimize LVS transmissions.

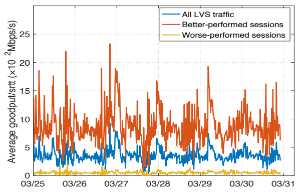
3.4 Better-Performed Sessions Perform Similarly in the Same UG

To explore potential associations of session QoS within a same UG, we randomly extract six-day LVS traffic data of UG 1. The time interval T is set to 15 minutes, where some LVS traffic with specific ranges of *retran_ratio* and *goodput/srtt* is extracted as better- or worse-performed LVS sessions (defined as below). Figure 2 shows that better-performed sessions extremely outperform worse-performed streams in terms of *retran_ratios* (~0.03% *vs.* ~2.9%) and *goodput/srtt* (~872 Mbps/s *vs.* ~53Mbps/s) on average.

- **Better-performed sessions** are defined as the LVS streams with 5th- to 20th-percentile *retran_ratios* or descending *goodput/srtt* in all last-period sessions.
- **Worse-performed sessions** are defined as the LVS streams with 75th- to 90th-percentile *retran_ratios* or descending *goodput/srtt* in all last-period sessions.



(a) Retransmission ratio

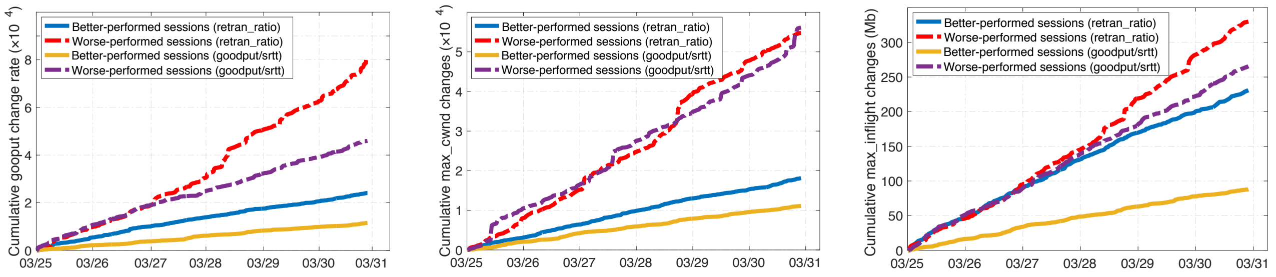


(b) Goodput/srtt

Figure 2: Better- and worse-performed LVS sessions.

To explore the similarity between better-performed LVS sessions in a same UG, we analyze the changes of goodput, *max_inflight* and *max_cwnd* in two adjacent time periods. Figure 3(a) depicts the cumulative goodput change rates of better- and worse-performed sessions in a same UG. We find that better-performed sessions has little-changed goodput in two adjacent periods, whose average change rate is 25% ~ 30% of that of worse-performed sessions. The cumulative changes of *max_cwnd* and *max_inflight* for better- and worse-performed session is shown in Figure 3(b) and 3(c), in which the curve slope represents the change rate compared to last-period values. We can learn the better-performed LVS sessions can introduce smoother *max_cwnd* and *max_inflight* changes than those worse-performed traffic. For example, 3.0× ~ 5.1× changes of *max_cwnd* and 1.3× ~ 3.0× changes of *max_inflight* are caused by worse-performed sessions compared to better-performed LVS streams. Therefore, better-performed LVS sessions in a same UG perform similarly with little-changed QoS values, which can be leveraged to automatically configure CC parameters for next period.

[3]In this paper, we regard ASN (instead of ISP) as the basic for user grouping, as ASN can better reflect the session similarity in a same UG. With user grouping, each user or client IP address can be exactly divided into a UG.



(a) Cumulative goodput change rates (b) Cumulative *max_cwnd* changes (c) Cumulative *max_inflight* changes

Figure 3: The QoS similarities of better-performed LVS session in a same UG.

4 THE AUTOPLEX DESIGN

4.1 AutoPlex Framework

AutoPlex enables to enhance LVS transmission performance by automatically multiplexing inter-session QoS similarities of better-performed LVS sessions in the last period, which will be used to configure next-period CC parameters. In particular, the maximum value of *cwnd* and *pacing_rate* are all configured as little-changed based on the QoS values of latest better-performed sessions. To meet diverse requirements, AutoPlex framework supports user-defined QoS policies (e.g., lowering *retran_ratio* or promoting the value of *goodput/srtt*), which will be followed to learn similar QoS metrics of better-performed LVS sessions. Figure 4 depicts AutoPlex framework that contains three modules for measurement, decision and execution. Note that these modules can be deployed on a same CDN proxy server or on different servers.

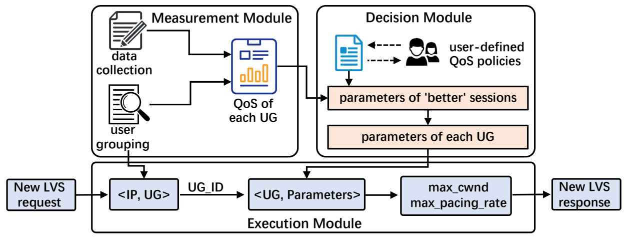


Figure 4: The framework of AutoPlex.

Measurement module can perform data collection of all LVS sessions and get last-period QoS values of each UG by looking up the pre-established <IP, UG> mapping table. With user-defined QoS policies, decision module obtains better-performed LVS sessions and records their similar QoS values (e.g., *goodput*, *max_cwnd* and *max_inflight*) that will be utilized for next-period CC parameter configurations. When receiving a new LVS request, execution module extracts remote IP and gains UG identifier UG_ID by inquiring our pre-established <IP, UG> table. With <UG, parameters> mapping table, CC parameters (i.e., *max_cwnd* and *max_pacing_rate*) of this UG will be obtained and then be configured for the response of this LVS session. In AutoPlex, measurement module counts QoS values of each UG for every time interval of T (e.g., T = *30 mins*),

Algorithm 1 CC parameter configuration pseudo code.

```
1: function CC PARAMETER COMPUTATION( )
2: Require:
3:    $P_{usr}$, $Q_{ug}$
4: Compute:
5:     $Q_{better} \leftarrow Extract(Q_{ug}, P_{usr})$
6:     $goodput,\ max\_inflight,\ max\_cwnd \leftarrow mean(Q_{better})$
7:     $pacing\_rate\_last = goodput$
8:     $max\_cwnd\_last = \frac{max\_inflight+max\_cwnd}{2}$
9: Operate:
10:     $pacing\_rate\_cpt = pacing\_gain * max\_bw$
11:     $pacing\_rate = f(pacing\_rate\_cpt, pacing\_rate\_last)$
12:     $inflight\_cap = f(inflight\_cap, max\_cwnd\_last)$
13: end function
```

which is utilized by decision module for maintaining latest <UG, parameters> mapping table in execution module.

4.2 Design Details

AutoPlex focuses on achieving automated configurations for CC parameters by multiplexing similar QoS values of those LVS sessions that performed better in the last period. In this paper, more fine-grained grouping rule (i.e., state-city-ASN) is leveraged to classify LVS consumers. Algorithm 1 shows the pseudo code for CC parameter configurations, where user-defined policies P_{usr} and each UG's QoS values Q_{ug} (i.e., *retran_ratio*, *goodput*, *srtt*, *max_inflight* and *max_cwnd*) act as inputs for parameter computations.

Better-performed session extraction. AutoPlex enables decision module to extract similar QoS values (denoted by Q_{better}) of better-performed LVS sessions in each UG. In Algorithm 1, function $Extract(\cdot)$ denotes sorting Q_{ug} according to P_{usr}-related metric (e.g., *retran_ratio*) and then obtains Q_{better} from $i-$ to $j-$percentile values of the sorted Q_{ug}. For example, $i = 5$ and $j = 20$ are set to extract Q_{better} from Q_{ug} (ascended by *retran_ratio*) in §3.3. Then, the average values of *goodput*, *max_inflight* and *max_cwnd* can be obtained while taking the mean operation for Q_{better}.

CC parameter configuration. With *goodput*, *max_inflight* and *max_cwnd* of better-performed sessions, AutoPlex first computes *pacing_rate_last* and *max_cwnd_last*, which reflects average *pacing_rate* and our desired *max_cwnd*, respectively, in the last period.

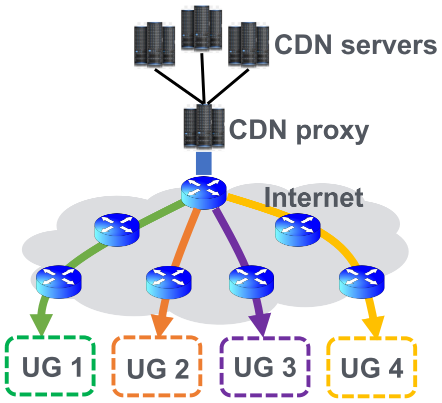


Figure 5: The experimental evaluation.

Then pacing rate and inflight cap can be configured for next-period LVS traffic transmissions, which will be elaborated in §4.3.

4.3 AutoPlex-based BBR Implementation

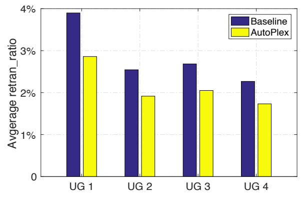
In this section, we give an example of how BBR makes use of the two outputs of AutoPlex, i.e., *pacing_rate_last* and *max_cwnd_last*, to adjust its pacing rate and infight cap.

Adjust pacing rate. A BBR sender controls its *pacing_rate* with the help of pacing and an estimated maximum bandwidth *max_bw*, where the estimate for *max_bw* is based on a windowed maximum filter of the delivery rate that the receiver experiences. The pacing rate is varied in an eight-phase cycle using a *pacing_gain* of 5/4,3/4,1,1,1,1,1,1, where each phase lasts for an RTT. The BBR implementation based on AutoPlex computes the pacing rate (i.e., *pacing_rate_cpt*) as line 11 of Algorithm 1 shows. In this paper, $f(\cdot)$ is a customizable function according to the optimization goal. For lower *retran_ratio* or higher *goodput/srtt*, we set $f(\cdot)$ as a function that takes the minimum value when $a \leq b$ and takes the average value when $a > b$ as follows.

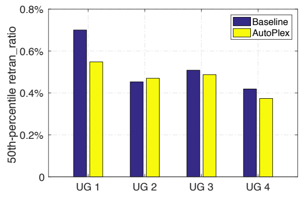
$$f(a,b) = \begin{cases} a & a \leq b \\ \frac{a+b}{2} & a > b \end{cases} \tag{1}$$

Adjust inflight cap. Although BBR does not use a congestion window or ACK clocking to control the amount of inflight data, it uses an inflight data limit (e.g., 2bdp), which we call it infight cap (denoted by *inflight_cap*) in this paper. The BBR implementation based on AutoPlex computes the inflight cap as line 12 in Algorithm 1 shows. This enables little-changed maximum values of *inflight_cap*, especially when the computed *inflight_cap* exceeds last-period *max_cwnd_last*.

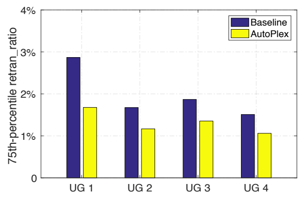
Discussion. In real-world deployment, we can activate AutoPlex just for some selected LVS traffic, e.g., high-priority sessions from VIP users. In this case, low-priority sessions are leveraged to explore better QoS metrics, which will be multiplexed by high-priority sessions. This partial deployment strategy enhances the convergence of AutoPlex. Note that, even though the same CC parameters are configured, better- and worse-performed LVS sessions in a same UG can be still identified as changing network conditions and status.



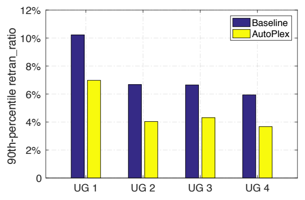
(a) Average retran_ratio



(b) 50th-percentile retran_ratio



(c) 75th-percentile retran_ratio



(d) 90th-percentile retran_ratio

Figure 6: Retransmission ratios of AutoPlex.

Therefore, the idea of multiplexing inter-session QoS metrics for CC parameter configurations can be always achieved over time.

5 EXPERIMENTAL EVALUATION

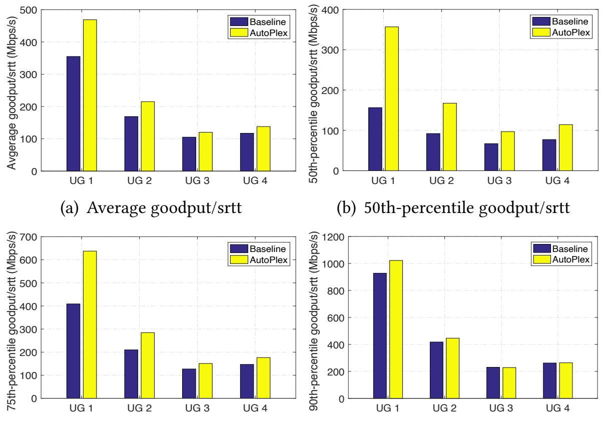
In this section, we implement AutoPlex prototype and partially deploy it in the real live CDN proxy (as Figure 5 shows) that is developed based on *Nginx* [22], which enables measurement, decision and execution modules of AutoPlex[4]. We randomly select four UGs (i.e., UG 1 ~ UG 4) to evaluate AutoPlex using the following metrics: *retran_ratio* and *gooodput/srtt*, which can be set as user-defined policies. The time period is set to 30 mins and LVS sessions in our evaluation are all based on QUIC protocol and BBRv1 scheme (as baseline) for transmission.

5.1 Retransmission Ratio

In AutoPlex framework, users can define their own preference as an optimization goal (e.g., lowering *retran_ratio*) for LVS traffic transmission. In this section, we select better-performed sessions with 5th- to 20th-percentile *retran_ratio* for CC parameter configuration of next-period LVS traffic.

Figure 6 depicts average, 50th-, 75th- and 90th-percentile *retran_ratio* changes of different UGs, respectively, when deploying AutoPlex prototype. We can learn AutoPlex can introduce lower *retran_ratio*, where the average, 75th- and 90th-percentile *retran_ratios* are reduced by 24% ~ 27%, 28% ~ 41% and 32% ~ 44%, respectively, compared to the existing BBR scheme. Note that 50th-percentile *retran_ratio* does not gain a obvious optimization, which always keeps close to optimal values (i.e., 0.37% ~ 0.70%) that are different to be further optimized. Meanwhile, AutoPlex brings differentiated reduction ratios for different UGs to optimize their *retran_ratios*. This is because traditional fixed CC parameter settings for entire networks can introduce various baseline values for each UG while AutoPlex can adjust CC parameters based on each UG's network conditions.

[4]In this paper, we meanly focus on the LVS transmission optimization from CDN proxy to user clients. Then video streaming interaction is beyond the scope of our research.



(a) Average goodput/srtt (b) 50th-percentile goodput/srtt

(c) 75th-percentile goodput/srtt (d) 90th-percentile goodput/srtt

Figure 7: Goodput/srtt values of AutoPlex.

5.2 Ratio of Goodput to SRTT

AutoPlex enables to enhance *goodput/srtt* by multiplexing QoS valuess of better-performed LVS sessions to configure *max_cwnd* and *max_pacing_rate* for the next LVS sessions. Note that the sessions with 80th- to 95th-percentile values of *goodput/srtt* in the last period are extracted as better-performed LVS sessions.

Figure 7 shows the values of *goodput/srtt* in different UGs. We can know AutoPlex can achieve a better optimization for the value of *goodput/srtt*, in which the average, 50th- and 75th-percentile values are all significantly enhanced (i.e., by 14% ∼ 32%, 44% ∼ 128%, and 19% ∼ 56%, respectively). By contrast, the 90th-percentile values have not been improved obviously. This might be because these values are already close to the optimal and have limited room to be further optimized. AutoPlex can introduce differentiated optimization results for various UGs. For example, 32% and 14% improvements of average *goodput/srtt* are incurred for UG 1 and UG 3, respectively. For the metric of *goodput/srtt*, AutoPlex can achieve obvious optimization for those poor performances (e.g., 50th- and 75th-percentile values) while keeping plausible improvements for already-better performances.

6 CONCLUSION AND FUTURE WORK

In this paper, we take a first step to design, analyze, implement and evaluate inter-session multiplexing congestion control framework named AutoPlex, which is based on our performed large-scale network measurements for live video streams. AutoPlex enables adaptive CC parameter configurations by multiplexing QoS values of better-performed LVS traffic in the last period. Besides, user-defined policies can also be supported in AutoPlex, which act as standards to learn QoS features of better-performed LVS traffic. We complete experimental evaluation by implementing and deploying AutoPlex in the real live CDN proxy, whose results demonstrate the huge potentials that AutoPlex introduces for optimizing LVS traffic transmission.

In the future, machine learning such as DRL can be leveraged to select better-performed LVS sessions based on the QoS performance gains of prior session selection. Besides, exploring more parameter settings [23, 24] by multiplexing inter-session QoS values is also a natural extension of AutoPlex.

ACKNOWLEDGMENTS

We would like to thank the NAI reviewers and Dr. Kai Gao for their extensive and valuable feedback. We also thank Senzhen Liu for his kind comments during our performing experimental evaluation.

REFERENCES

[1] Max Wilbert. Top 22 live streaming platforms: Everything you need to know. https://www.dacast.com/blog/, 2022.
[2] Van Jacobson. Congestion avoidance and control. In *ACM SIGCOMM computer communication review*, volume 18, pages 314–329. ACM, 1988.
[3] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: a new tcp-friendly high-speed tcp variant. *ACM SIGOPS operating systems review*, 42(5):64–74, 2008.
[4] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. BBR: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time. *ACM Queue*, 14(5):20–53, 2016.
[5] Mo Dong, Qingxi Li, Doron Zarchy, P Brighten Godfrey, and Michael Schapira. PCC: Re-architecting congestion control for consistent high performance. In *USENIX NSDI*, pages 395–408, 2015.
[6] Xiaohui Nie, Youjian Zhao, Zhihan Li, Guo Chen, Kaixin Sui, Jiyang Zhang, Zijie Ye, and Dan Pei. Dynamic tcp initial windows and congestion control schemes through reinforcement learning. *IEEE JSAC*, 37(6):1231–1247, 2019.
[7] Xu Li, Feilong Tang, Jiacheng Liu, Laurence T Yang, Luoyi Fu, and Long Chen. AUTO: Adaptive congestion control based on multi-objective reinforcement learning for the satellite-ground integrated network. In *USENIX ATC*, 2021.
[8] Francis Y Yan, Jestin Ma, Greg D Hill, Deepti Raghavan, Riad S Wahby, Philip Levis, and Keith Winstein. Pantheon: the training ground for internet congestion-control research. In *USENIX ATC*, pages 731–743, 2018.
[9] Lawrence S Brakmo, Sean W O'Malley, and Larry L Peterson. *TCP Vegas: New techniques for congestion detection and avoidance*, volume 24. ACM, 1994.
[10] Lisong Xu, Khaled Harfoush, and Injong Rhee. Binary increase congestion control (bic) for fast long-distance networks. In *IEEE Infocom*, volume 4, pages 2514–2524, 2004.
[11] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. Adaptive congestion control for unpredictable cellular networks. In *ACM SIGCOMM*, pages 509–522, 2015.
[12] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *USENIX NSDI*, pages 459–472, 2013.
[13] Venkat Arun and Hari Balakrishnan. Copa: Practical delay-based congestion control for the internet. In *USENIX NSDI*, pages 329–342, 2018.
[14] Tong Li, Kai Zheng, and Ke Xu. Acknowledgment on demand for transport control. *IEEE Internet Computing*, 25(2):109–115, 2021.
[15] Tong Li, Kai Zheng, Ke Xu, Rahul Arvind Jadhav, Tao Xiong, Keith Winstein, and Kun Tan. Tack: Improving wireless transport performance by taming acknowledgments. In *ACM SIGCOMM*, pages 15–30, 2020.
[16] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. {PCC} vivace: Online-learning congestion control. In *USENIX NSDI*, pages 343–356, 2018.
[17] Keith Winstein and Hari Balakrishnan. Tcp ex machina: Computer-generated congestion control. *ACM SIGCOMM Computer Communication Review*, 43(4):123–134, 2013.
[18] Soheil Abbasloo, Chen-Yu Yen, and H Jonathan Chao. Classic meets modern: A pragmatic learning-based congestion control for the internet. In *ACM SIGCOMM*, 2020.
[19] Michael Schapira and Keith Winstein. Congestion-control throwdown. In *ACM Hotnets*, pages 122–128, 2017.
[20] Junchen Jiang, Shijie Sun, Vyas Sekar, and Hui Zhang. Pytheas: Enabling data-driven quality of experience optimization using Group-BasedExploration-Exploitation. In *USENIX NSDI*, 2017.
[21] Minsik Cho and Daniel Brand. MEC: Memory-efficient convolution for deep neural network. In *International Conference on Machine Learning*, pages 815–824. PMLR, 2017.
[22] F5 Networks Inc. NGNIX: Advanced load balancer, web server, & reverse proxy. https://www.nginx.com, 2022.
[23] Tong Li, Kai Zheng, Ke Xu, Rahul Arvind Jadhav, Tao Xiong, Keith Winstein, and Kun Tan. Revisiting acknowledgment mechanism for transport control: Modeling, analysis, and implementation. *IEEE/ACM TON*, 29(6):2678–2692, 2021.
[24] Hui Xie and Li Tong. Revisiting loss recovery for high-speed transmission. In *IEEE WCNC*, pages 1–6, 2022.