6 SUPPLEMENTARY MATERIAL

6.1 Related works

Over the past decades, numerous methodologies have been proposed to improve the bandwidth prediction precision in RTC systems. Google Congestion Control (GCC)[1] is the most representative heuristic-based algorithm, which combines the packet delay and packet loss to make bandwidth predictions.

Learning-based bandwidth prediction algorithms mainly utilize reinforcement learning (RL) networks, among which HRCC[4] and CLCC[3] are typical offline learning methods. For achieving better flexibility in bandwidth prediction, Concerto[7], OnRL[6], QARC[2] and Loki[5] all adopt the online training strategy to tune the RL networks in real-time. Concerto[7], OnRL[6], and Loki[5] are deployed on the large-scale commercial platform "TaoBao". Based on current network conditions and users' location, OnRL and Loki[5] can make timely bandwidth predictions.

6.2 Pseudo-code of Frame Rate Controller

The pseudo-code of FRC module is illustrated as Alg 1. The term ϕ indicates the pre-defined states of FRC, while s represents the state machine signal to drive the state transitions of FRC. d_f^{avg} denotes the weighted average frame delay in sliding window $\mathcal{W}$. f_l is the current frame-rate level decided by FRC. γ_t indicates the dynamic threshold of frame delay in equation 6.

6.3 QoE Performance Comparison in other metrics

As mentioned in section 4.3, the proposed SAFR surpasses the BASE not only in the VMAF quality score, frame loss ratio and frame delay, but also in SSIM, PSNR distortion measures and bandwidth usage. As reported in Table 1, SAFR outperforms the BASE with slight PSNR, SSIM quality enhancements and significant bandwidth savings (12.0%-25.3%), which further validates the superiority of SAFR.

Table 1: Average experiment results of SAFR and BASE under different network environment

Network	System	PSNR(dB) ↑	SSIM ↑	Bandwidth requirments(Kbps) ↓
250K-1M(bps)	BASE	31.29	0.892	261.04
	SAFR	**31.57**	**0.900**	**229.78**
250K-2M(bps)	BASE	33.33	0.922	638.77
	SAFR	**33.41**	**0.925**	**486.14**
250K-4M(bps)	BASE	34.15	0.931	884.78
	SAFR	**34.18**	**0.933**	**661.35**

Algorithm 1 Frame Rate Controller

```
1: Initialization: $\phi \leftarrow$ 'Hold', $d_f^{avg} \leftarrow 0, \gamma_t \leftarrow 100, f_l \leftarrow f_2$
2: while RTC is running do
3:     wait until feedback is received                ▷ every 500ms
4:     repeat
5:         calculate each new $d_f$ based on equation 2
6:         $\mathcal{W}$ append $d_f$
7:     until size of $\mathcal{W}$ == 6
8:     calculate $d_f^{avg}$ based on equation 3
9:     if $d_f^{avg} > \gamma_t$ then                  ▷ Frame rate level decreases
10:        $s \leftarrow$ 'Overuse'
11:        $\phi \leftarrow$ 'Decrease'
12:        $f_l \leftarrow \max(f_{l-1}, f_0)$          ▷ $f_{-1}$ is invalid
13:    else
14:        calculate $d_f^g$ based on equation 4
15:        if $d_f^g < 0$ then                          ▷ Frame rate level Increases
16:            $s \leftarrow$ 'Underuse'
17:            $\phi \leftarrow$ 'Increase'
18:            $f_l \leftarrow \min(f_{l+1}, f_2)$      ▷ $f_3$ is invalid
19:        else                                         ▷ keep the frame rate level
20:            $s \leftarrow$ 'Normal'
21:            $\phi \leftarrow$ 'Increase'
22:        end if
23:    end if
24:    update $\gamma_t$ based on equation 6
25: end while
```

6.4 Explanation of hyper-parameter settings

As mentioned in the main body of this paper, a series of hyper-parameters have been set throughout the experiment. To make it clearer, the motivation of hyper-parameter settings is elaborated in this section.

- **The time interval $\tau = 500ms$.**
  By default, feedback is returned every 500ms in GCC[1], hence is adopted in this paper as well.
- **The sliding window length $T = 6$**
  For better representing current transmission status, the sliding window should ideally not contain any information of previous τ, hence the upper-bound of T is actually decided by the minimal frame-rate level (i.e. f_0). In this paper, we use videos of 30fps for evaluation, thus 7 or 8 frames can be transmitted within $\tau = 500ms$ in f_0 setting (i.e. sending a frame per 66ms). Considering the possible frame loss, the sliding window length T is set as 6 ultimately.
- **The dynamic frame-delay threshold γ_t in FRC**
  Considering that the pre-defined fixed thresholds may lead to poor frame-rate alteration decisions, we further utilize the scaled deviation between d_f^{avg} and γ_t to update the threshold γ_t dynamically and keep pace

with the changing network conditions. For instance, if current network conditions are so poor that d_f^{avg} is continuously larger than the pre-defined fixed threshold, the FRC will merely produce frame-rate levels of f_0. Therefore, an adaptive threshold is of vital importance in this case.

- **The scale factor k_γ in equation 6**
  In this paper, we pick the best value of k_γ among the set $\{0.3, 0.4, 0.5, 0.6, 0.7\}$ via a batch of experiments. Too small k_γ may lead to slow-changing γ_t and slow reactions to network changes. Conversely, too large k_γ may bring about great fluctuations of frame-rate levels. Experiments show that $k_\gamma = 0.5$ is the most appropriate choice, hence is adopted in this paper.
- **The upper bound of m in equation 8**
  In this paper, the upper bound of calibration factor m has been tested in the range of [1.1, 1.5] with binary search method. Experiments demonstrated that 1.2 is the most proper value. Specifically, an upper bound greater than 1.2 may lead to over-raise of bandwidth prediction and thereby excessive frame delay, while an upper bound lower than 1.2 is kind of conservative, and may result in insufficient bandwidth utilization.
- **The weight coefficients in equation 9**
  We set a series of weight coefficients in equation 9 to scale all the metrics to the same order of magnitude (i.e. range [0, 10]), and thereby prevent preference for any of them. According to the most common range of each term in live streaming scenarios, the weight coefficients are set as $\beta_1 = 0.1$, $\beta_2 = 10$, $\beta_3 = 16$, $\beta_4 = 0.1$ in this paper.

SUPPLEMENTARY REFERENCE

[1] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2016. Analysis and design of the google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems*. 1–12.

[2] Tianchi Huang, Rui-Xiao Zhang, Chao Zhou, and Lifeng Sun. 2018. QARC: Video quality aware rate control for real-time video streaming based on deep reinforcement learning. In *Proceedings of the 26th ACM international conference on Multimedia*. 1208–1216.

[3] Haoyong Li, Bingcong Lu, Jun Xu, Li Song, Wenjun Zhang, Lin Li, and Yaoyao Yin. 2022. Reinforcement Learning Based Cross-Layer Congestion Control for Real-Time Communication. In *2022 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. IEEE, 01–06.

[4] Bo Wang, Yuan Zhang, Size Qian, Zipeng Pan, and Yuhong Xie. 2021. A Hybrid Receiver-side Congestion Control Scheme for Web Real-time Communication. In *Proceedings of the 12th ACM Multimedia Systems Conference*. 332–338.

[5] Huanhuan Zhang, Anfu Zhou, Yuhan Hu, Chaoyue Li, Guangping Wang, Xinyu Zhang, Huadong Ma, Leilei Wu, Aiyun Chen, and Changhui Wu. 2021. Loki: improving long tail performance of learning-based real-time video adaptation by fusing rule-based models. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 775–788.

[6] Huanhuan Zhang, Anfu Zhou, Jiamin Lu, Ruoxuan Ma, Yuhan Hu, Cong Li, Xinyu Zhang, Huadong Ma, and Xiaojiang Chen. 2020. OnRL: improving mobile video telephony via online reinforcement learning. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.

[7] Anfu Zhou, Huanhuan Zhang, Guangyuan Su, Leilei Wu, Ruoxuan Ma, Zhen Meng, Xinyu Zhang, Xiufeng Xie, Huadong Ma, and Xiaojiang Chen. 2019. Learning to coordinate video codec with transport protocol for mobile video telephony. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–16.