Reinforcement learning for bandwidth estimation and congestion control in real-time communications

Joyce Fang, Martin Ellis, Bin Li, Siyao Liu, Yasaman Hosseinkashi, Michael Revow,
Albert Sadovnikov, Ziyuan Liu, Peng Cheng, Sachin Ashok,
David Zhao, Ross Cutler, Yan Lu, Johannes Gehrke

Microsoft

Abstract

Bandwidth estimation and congestion control for real-time communications (i.e., audio and video conferencing) remains a difficult problem, despite many years of research. Achieving high quality of experience (QoE) for end users requires continual updates due to changing network architectures and technologies. In this paper, we apply reinforcement learning for the first time to the problem of real-time communications (RTC), where we seek to optimize user-perceived quality. We present initial proof-of-concept results, where we learn an agent to control sending rate in an RTC system, evaluating using both network simulation and real Internet video calls. We discuss the challenges we observed, particularly in designing realistic reward functions that reflect QoE, and in bridging the gap between the training environment and real-world networks.

1 Introduction

Congestion control and bandwidth estimation are fundamental problems in networking research. They are concerned with how much data can be sent across a network path at a given time, and how and when endpoints should send packets to avoid causing network congestion and the associated packet delay and loss [20]. There have been various applications of reinforcement learning (RL) to these problems, such as for congestion control in TCP [8, 14] and QUIC [18], and one-way adaptive bitrate video streaming [10, 5, 22, 1]. In this paper, we apply RL to congestion control and bandwidth estimation in real-time communications (RTC), for the first time. We begin by reviewing prior applications of RL to video streaming (since it is most closely related to RTC) before outlining the differences in our approach.

RL for Video Streaming: RL has been successfully applied to the control of adaptive bitrate video streaming systems (i.e., YouTube-like one-way video streaming), with results showing that RL gives large improvements over existing approaches under certain conditions [10, 5, 1]. In these systems, clients typically select videos to download from a fixed set of available quality levels, which correspond to discrete actions for RL agents [10]. RL reward functions can be designed based on QoE metrics, combining measurements such as bitrate, delay, and video quality with configurable weights. The most similar domain to RTC is *real-time* video streaming [5], which can be thought of as "one-directional" high latency RTC. This has some of the real-time constraints of RTC, although since it is non-interactive, the requirements are less strict.

RL for RTC: Beyond video streaming, an equally important but harder problem is real-time communications (RTC). In WebRTC [11], several congestion control approaches have been proposed [2, 23, 9]; other work has focused on optimizing RTC performance beyond WebRTC [21, 4]. To the best of our knowledge, RL has not been applied to this space.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

RTC is different from video streaming for the following reasons. First, RTC requires minimal latency and cannot pre-fetch content, so large receiver-side buffers (common in video streaming) cannot be used; the system needs to react faster to bandwidth changes with less margin for error. This constraint also means that packet losses have a bigger impact, since there is less time available to retransmit lost packets. Second, since RTC involves end users uploading their audio/video streams, quality is likely be limited by their uplink capacity, which is often more constrained than their downlink capacity. Third, since the RL model needs to run in a real-time environment, the inference time needs to be orders of magnitude faster than the streaming case, further constraining the complexity of the RL model. Fourth, since RTC systems do not work with the pre-encoded quality levels that are typical in video streaming systems, the action space in RTC is typically larger or continuous.

Due to the constraints above, we cannot apply previous RL formulations designed for one-directional delivery with high buffer latency. To address these challenges, we propose R3Net, an RL-based Recurrent Network for RTC, allowing rapid adjustment to complex and dynamic network conditions.

Paper Outline: We outline our initial training environment, simulator, and model in §2. We evaluate the model (using simulation and video calls on real networks) in §3, and discuss open issues in §4.

2 R3Net: An Initial Approach

In RL the formulation of the problem in terms of states, actions, and reward is crucial. In RTC, the ultimate reward is to deliver excellent QoE to end users, although the actions that can be taken to achieve this can vary widely. In our present work, we focus on a subset of the problem (bandwidth estimation and congestion control), but we note that there are numerous other sub-problems in RTC can naturally be posed as RL problems (e.g., jitter-buffer control, packet loss resiliency, video encoding, etc). Eventually, an RL agent might control all actions in an RTC system, continuously improving QoE in an online manner.

We take a receiver-side approach to bandwidth estimation in RTC calls, using incoming RTP [16] packets to estimate available bandwidth on the path between sender and receiver. We then signal this estimate back to the sender via RTCP, allowing sender-side logic to control sending rate. Our existing reference system uses an Unscented Kalman Filter (UKF) with a rule-based controller to estimate and control bandwidth. Our initial approach to applying RL to RTC is to use the observations of the incoming packet timeseries as input to the neural network, training a model to estimate the available bandwidth that will replace the UKF method. We will compare the methods in §3.

2.1 Simulator

In RL training, it is common to use a simulation environment to speed up training, allowing agents to learn from vast numbers of observations before they are deployed into their target environment. For RTC, this means that a realistic simulation of Internet and application performance is required. Training may also be done online, with observations being collected from a full-scale RTC system, even learning from production calls. In the latter case, the collection of training data and real-time continuous updating of the model needs to be considered.

We can train an RL model either in the real RTC process or in a simulator. As an initial approach, we use a simulator that can mimic the RTC process, but runs 1000x faster than real-time, to speed up training. Our simulator consists of the caller and callee RTC endpoints connected by a simulated network link; we can also simulate cross traffic (e.g., from TCP senders). The network simulator uses trace-replay-based simulation to control the parameters of the bottleneck link (including capacity, delay, and packet loss) in a discrete event simulation.

2.2 RL formulation

To estimate bandwidth, the receiver can use the incoming RTP packets and the measured round-trip time (RTT). In our formulation, we use the aggregated RTP and RTT information in a fixed time-window of 50 ms as the environment state, and estimated bandwidth as the agent's action. The environment updates the next state and reward based on the input action.

State and Action: The state is a 4-dimension vector consisting of receive rate (kb/s), average packet interval (ms), packet loss rate (%), and average RTT (ms). We further scale the state to produce inputs

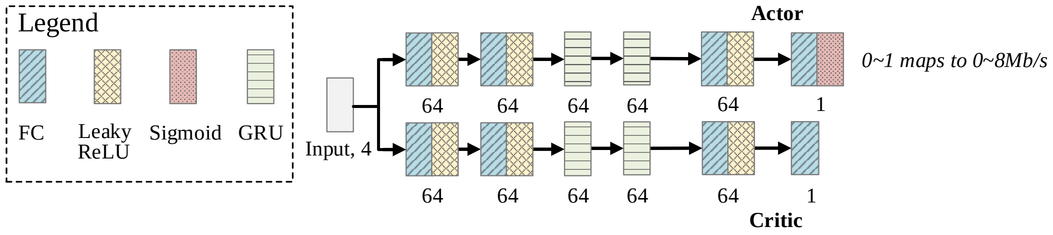

Figure 1: R3Net structure (numbers indicate output features per layer, and hidden size for GRUs)

Table 1: Evaluation results from simulation

	Bandwidth utilization	RTT (ms)			Packet loss rate	Reward mean
		avg.	p50	p95		
UKF	73.5%	128	102	288	3.8%	0.56
R3Net	77.8%	122	102	268	1.9%	0.60

with the same order of magnitude to the neural network. We use sigmoid activation as the last layer of the network, yielding outputs in the $(0, 1)$ range. We then map the output to $(0, 8)$ Mb/s as the bandwidth estimate, corresponding to an appropriate range for our RTC application.

Reward Design: We define reward per 50ms time step as $0.6 \ln(4\mathcal{R} + 1) - \mathcal{D} - 10\mathcal{L}$, where $\mathcal{R}$ is receive rate in that time step, in Mb/s, $\mathcal{D}$ is the average RTT in that time step, in seconds, and $\mathcal{L}$ is packet loss rate. This means that receiving more packets is rewarded (since this should lead to higher QoE), but delay and packet loss are penalized (since these degrade QoE).

2.3 Model and Training

The input of the neural network is a time series, representing the state of the path between sender and receiver over time. The history information has impact on the estimated bandwidth (e.g., increasing RTT may mean previous bandwidth estimates were too large). Thus, we use a recurrent neural network with Gated Recurrent Units (GRUs) [3] to estimate bandwidth, as shown in Figure 1. For the leaky ReLU layer, we use the negative slope of 0.01. For the rest of the paper, we refer to this neural network as R3Net (RL-based Recurrent Network for RTC).

We train R3Net using an actor-critic framework, where the actor and critic share the first few layers. The model is updated using Proximal Policy Optimization (PPO) [15] and the Adam optimizer with a learning rate of 3×10^{-5}, implemented using PyTorch, based on DeepRL [17]. We used around 10,000 network traces for simulation in training, and tested on 1150 different network traces.

3 Evaluation

3.1 Evaluation Through Simulation

We first evaluate R3Net in simulation, comparing R3Net with UKF based on our set of 1150 test traces. Ideally, our evaluation criteria would be based on user-perceived quality (e.g., MOS [6, 7]), but since our simulation environment uses synthetic audio and video packets, we use purely network-based metrics including observed RTT, packet loss rate, and *bandwidth utilization*, the percentage of bandwidth used relative to the (simulated) limit. Detailed results are shown in Table 1; we see that R3Net has $\sim$5% higher bandwidth utilization than UKF (see Figure 2 for an example), with similar RTTs and less packet loss. These initial simulation results are promising, with R3Net showing higher reward and better overall performance than UKF; in the next section, we evaluate model performance in real network conditions.

3.2 Evaluation Using Real Networks

We now describe our very preliminary evaluation of R3Net performance in RTC calls on real networks. First, we deployed the R3Net into the ONNX format [12], and use ONNX Runtime [13] for inference in our RTC system. Currently the inference time of R3Net takes approximately 500 μs and the model

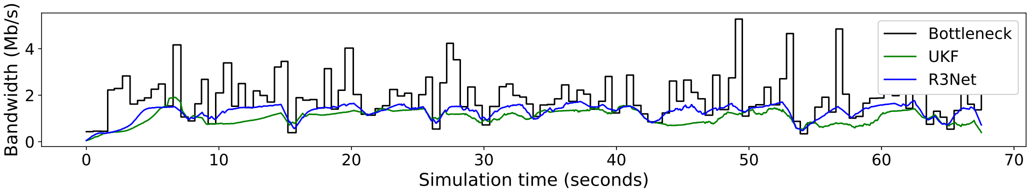

Figure 2: Example result from one simulation test run

Table 2: Evaluation results from 3G and WiFi

	Network type	RTT (ms)			Packet loss rate	VMAF	Frame drop rate
		avg.	p50	p95			
UKF	3G	58	56	86	2.22%	81.8	6.5%
	WiFi	16	13	37	0.05%	94.1	2.5%
R3Net	3G	58	55	99	3.11%	78.6	11.2%
	WiFi	16	13	38	0.01%	93.4	1.8%

is called every 50 ms. The inference time of R3Net is about 20 times more expensive than UKF and rule-based approaches but still within the runtime requirement.

To compare the performance of UKF and R3Net, we ran two-way audio/video calls using a scriptable RTC client in two different scenarios, *WiFi*, and *3G*, where the first machine is connected via residential WiFi and USB-tethered 3G connections, respectively. In all scenarios, the second machine is connected to our office network, and has the RTC application record the incoming video stream, allowing us to compute objective quality scores using VMAF [19]. This allows us to objectively evaluate the performance of UKF and R3Net, independently of the training environment.

We ran 200 test calls for *3G* and 200 for *WiFi*, each lasting 30 seconds, alternating between running UKF and R3Net. Table 2 shows the RTT, packet loss rate, and video quality scores. We see that although the RTTs are fairly similar between R3Net and UKF for each network type, packet loss rates are higher on 3G networks when using R3Net. There are corresponding degradations in both VMAF (indicating poorer image quality) and video frame drop rate (indicating choppy video). This suggests that the simulation environment does not sufficiently represent the real network environment. We observe that R3Net takes relatively noisy actions compared to UKF, which might lead to high packet loss and choppy video in real networks.

4 Discussion and Open Questions

In this paper, we propose a new formulation of RL for bandwidth estimation and congestion control in real-time audio/video communication, and show R3Net provides reasonable adjustment to dynamic network conditions in simulation and real networks using WiFi connections. Although the evaluation results in 3G networks suggest further improvement of the model is needed, we hope our end-to-end training and deployment of RL to RTC stimulates further work in this direction. We identify two key areas for improvement: 1) *How to close the gap between training and the real world through realistic network simulation?* 2) *How to formulate reward functions that directly optimize QoE?*

From our preliminary results, we are encouraged that R3Net can deliver a reasonable experience over WiFi (though not yet matching UKF). However, since R3Net suffers high packet loss (and poor video quality) in our 3G tests, it is clear our training environment is not sufficiently representative. It is common to start RL training in a simulator (i.e., a gym environment), but developing a sufficiently realistic environment to represent the real world is challenging. In future work, we plan to improve the simulation using data driven methods (i.e., using a generative model to produce realistic network traces), and to build a distributed testbed to enable large scale training using real networks.

In our training, we used a simple intuitive reward, and see that R3Net performs more aggressively (i.e., uses more bandwidth) than UKF in simulation-based evaluation. The higher packet loss rate in real network evaluation shows that we need to redesign this reward function. We believe that using objective functions for audio/video quality measured throughout the call as multi-step rewards in training may yield better real-world performance; the design of a reward function that leads to high QoE is a key challenge for future work.

Acknowledgments

We would like to thank Nelson Pinto for help in developing the simulation environment, Sergiy Matusevych for help with ONNX Runtime integration, James Winters and Yang Sun for their work building the evaluation environment, and Sasikanth Bendapudi for help with video quality assessment.

References

[1] R. Bhattacharyya, A. Bura, D. Rengarajan, M. Rumuly, S. Shakkottai, D. Kalathil, R. K. Mok, and A. Dhamdhere. QFlow: A Reinforcement Learning Approach to High QoE Video Streaming over Wireless Networks. In *MobiHoc '19: Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, July 2019.

[2] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo. Analysis and Design of the Google Congestion Control for Web Real-time Communication (WebRTC). In *MMSys '16: Proceedings of the 7th International Conference on Multimedia Systems*, 2016.

[3] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP 2014: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, October 2014.

[4] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, and K. Winstein. Salsify: Low-Latency Network Video through Tighter Integration between a Video Codec and a Transport Protocol. In *NSDI '18: Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation*, April 2018.

[5] T. Huang, R.-X. Zhang, C. Zhou, and L. Sun. QARC: Video quality aware rate control for real-time video streaming based on deep reinforcement learning. In *MM '18: Proceedings of the 26th ACM International Conference on Multimedia*, October 2018.

[6] ITU-T. Mean Opinion Score (MOS) terminology, 1996. Rec. ITU-T P.800.1.

[7] ITU-T. Methods for subjective determination of transmission quality, 1996. Rec. ITU-T P.800.

[8] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar. A Deep Reinforcement Learning Perspective on Internet Congestion Control. In *ICML 2019: Proceedings of the International Conference on Machine Learning*, June 2019.

[9] I. Johansson. Self-clocked Rate Adaptation for Conversational Video in LTE. In *CSWS '14: Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop*, August 2014.

[10] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with Pensieve. In *SIGCOMM '17: Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, August 2017.

[11] A. Narayanan, C. Jennings, B. Aboba, J.-I. Bruaroey, D. Burnett, A. Bergkvist, and T. Brandstetter. WebRTC 1.0: Real-time communication between browsers. Candidate recommendation, W3C, Sept. 2018. https://www.w3.org/TR/2018/CR-webrtc-20180927/.

[12] ONNX: Open Neural Network Exchange. `https://github.com/onnx/onnx`, 2019.

[13] ONNX Runtime: cross-platform, high performance scoring engine for ML models. `https://github.com/microsoft/onnxruntime`, 2019.

[14] F. Ruffy, M. Przystupa, and I. Beschastnikh. Iroko: A Framework to Prototype Reinforcement Learning for Data Center Traffic Control. *arXiv preprint arXiv:1812.09975 [cs.NI]*, December 2018.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347 [cs.LG]*, August 2017.

[16] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. *Internet RFCs*, RFC 3550, 2003.

[17] Z. Shangtong. Modularized Implementation of Deep RL Algorithms in PyTorch. `https://github.com/ShangtongZhang/DeepRL`, 2018.

[18] V. Sivakumar, T. Rocktäschel, A. H. Miller, H. Küttler, N. Nardelli, M. Rabbat, J. Pineau, and S. Riedel. MVFST-RL: An Asynchronous RL Framework for Congestion Control with Delayed Actions. *arXiv preprint arXiv:1910.04054 [cs.LG]*, October 2019.

[19] VMAF: Video Multi-Method Assessment Fusion. `https://github.com/Netflix/vmaf`, 2019.

[20] K. Winstein and H. Balakrishnan. TCP Ex Machina: Computer-generated Congestion Control. In *SIGCOMM '13: Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, August 2013.

[21] K. Winstein, A. Sivaraman, and H. Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *NSDI '13: Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, April 2013.

[22] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein. Continual learning improves Internet video streaming. *arXiv preprint arXiv:1906.01113 [cs.NI]*, June 2019.

[23] X. Zhu and R. Pan. NADA: A Unified Congestion Control Scheme for Low-Latency Interactive Video. In *PV 2013: Proceedings of the 20th International Packet Video Workshop*, December 2013.