OpenNetLab: Open Platform for RL-based Congestion Control for Real-Time Communications

Jeongyoon Eo[1,2], Zhixiong Niu[2], Wenxue Cheng[2], Francis Y. Yan[2]
Rui Gao[2], Jorina Kardhashi[3], Scott Inglis[3], Michael Revow[3], Byung-Gon Chun[1], Peng Cheng[2], Yongqiang Xiong[2]

[1]Seoul National University, [2]Microsoft Research, [3]Microsoft

ABSTRACT

With the growing importance of real-time communications (RTC), designing congestion control (CC) algorithms for RTC that achieve high network performance and QoE is gaining attention. Recently, data-driven, reinforcement learning (RL)-based CC algorithms for RTC have shown great potential, outperforming traditional rule-based counterparts. However, there are no open platforms tailored for training, evaluation, and validation of the algorithms that can facilitate this emerging research area.

We present OpenNetLab, an open platform for fast training, reproducible end-to-end evaluation, and performance validation of RL-based CC algorithms for RTC. Preliminary use cases confirm that OpenNetLab concretely aided the training of novel RL-based CC algorithms for RTC that outperform a well-established rule-based baseline in both network performance and QoE metrics.

CCS CONCEPTS

• Networks → Transport protocols; **• Computing methodologies → Machine learning**; **• Software and its engineering → Application specific development environments**;

KEYWORDS

Real-time communications, Congestion control, Reinforcement learning, Open platform

ACM Reference Format:
Jeongyoon Eo, Zhixiong Niu, Wenxue Cheng, Francis Y. Yan, Rui Gao, Jorina Kardhashi, Scott Inglis, Michael Revow, Byung-Gon Chun, Peng Cheng, Yongqiang Xiong. 2022. OpenNetLab: Open Platform for RL-based Congestion Control for Real-Time Communications. In *6th Asia-Pacific Workshop on Networking (APNet 2022), July 1–2, 2022, Fuzhou, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3542637.3542648

1 INTRODUCTION

Recently, the popularity and importance of real-time communications (RTC) has been rapidly growing both in existing and emerging domains, especially due to the COVID-19 pandemic [9]. Notable examples are the popularity of mainstream video conferencing applications such as Microsoft Teams [4], Google Meet [2] and Zoom [7], which play a key role from online education to remote work. With the advent of 5G network infrastructures, emerging cases such as virtual reality (VR) [11, 14] and cloud gaming [17, 25] constantly adds importance of RTC.

Traditional RTC systems adopt rule-based algorithms for congestion control (CC). Notable example is GCC [16], a default CC algorithm of WebRTC [10], a de facto back-end framework for browser-to-browser RTC over the Internet. GCC uses one-way delay gradient to infer congestion, and it hardwires a set of rules that map pre-defined packet-level events to network control decisions. Recent publications [31, 32, 35] analyze that due to its rule-based nature, GCC faces limitations in adapting video bitrate and latency under the heterogeneous modern Internet with diverse bandwidth and latency characteristics.

Recently, data-driven, reinforcement-learning (RL)-based approaches have shown great potential in many domains of networking research. For instance, Pensieve [22] showed that RL-based bitrate adaptation for VoD can reduce frame stalling while improving bandwidth utilization. PCC-RL [18] demonstrated that an RL-based TCP CC algorithm can successfully learn to distinguish patterns that rule-based algorithms cannot capture, which allowed it to achieve better performance compared to the rule-based TCP CUBIC [15], which is default in linux. OnRL [32] and Loki [31] trained RL-based bitrate controllers for RTC, which demonstrated large improvement in both network performance and QoE especially under lossy wireless networks.

Despite the growing importance of the RL-based approaches in RTC, the research community lacks a common platform for training, evaluation and validation which is essential in facilitating the research in this domain. Although there exist domain-agnostic tools for RL training [6, 21] or open testbeds for networking research in domains other than RTC [29, 30], we found that building a platform for training, evaluation and validation for RL-based CC for RTC cannot be achieved by naively stitching them together, but requires design decisions that meet system requirements analyzed in § 2.1.

To this end, we propose OpenNetLab, a first open platform for training, evaluation and validation of RL-based CC algorithms for RTC. We believe that such a community platform will make a major contribution in facilitating the research in this field which has growing importance. We make the following contributions:

- We design and build the first open platform for training, evaluation, and validation of RL-based CC algorithms for RTC. The platform is open sourced in https://github.com/OpenNetLab.
- ∼40 test nodes have been deployed across universities and research institutions in East Asia, including China, Hong Kong SAR, South Korea, and Singapore, for validating RL-based CC algorithms under heterogeneous networks (4G/5G, Wi-Fi, wired).
- Preliminary use cases of RL-based CC algorithms trained using OpenNetLab in the MMSys challenge [1] demonstrate the contribution of OpenNetLab to the research community.

In the following sections, we present main motivation of OpenNetLab (§ 2) followed by system design that achieves our vision (§ 3) and the implementation of it (§ 4). We share performance results of preliminary use cases of OpenNetLab in the MMSys challenge [1] (§ 5). Finally, we summarize related works (§ 6) and discuss open research questions (§ 7).

2 BACKGROUND AND REQUIREMENTS

2.1 Background

Reinforcement Learning. In reinforcement learning (RL), an *agent* interacts with environment by producing *actions* based on the observed *states* to maximize a *reward*. The agent learns a *policy* that maps a state of the environment to an action. The exact definitions of environment, agent, state, action, and reward are problem-specific.

Bandwidth-estimation based CC in WebRTC. WebRTC [10] is a de facto back-end framework for browser-to-browser RTC over the Internet. As WebRTC traffics are transferred via RTP over UDP, the WebRTC has the responsibility to mitigate the unreliable delivery.

To deal with the unreliability, WebRTC figures out achievable media quality given the latency constraints based on the estimated bandwidth. Specifically, based on the estimated bandwidth, WebRTC determines video resolutions or frame sizes by adjusting media-encoding parameters to meet the latency-media quality tradeoff. In this way, WebRTC aims to achieve the best possible media quality while meeting the latency constraints to assure that media keeps flowing. This is critical for quality-of-experience (QoE) especially when peers are connected to networks with different throughput levels or under low and varying available bandwidth.

RL-based CC algorithm for WebRTC. In RL-based CC algorithm, an agent interacts with a given network environment by receiving network statistics from feedback RTCP packet that forms a *state* and producing estimated bandwidth as an *action*. Through training, the agent learns a policy to produce actions under given states that maximizes reward, which is designed to represent network performance and QoE.

2.2 System Requirements

We propose three requirements for an open platform that supports training, evaluation and validation of RL-based CC algorithms for RTC.

- **R1.** *Easy-to-use interfaces for designing RL algorithms.* The platform should bridge the knowledge gap [28] between researchers with various levels of RL and networking expertise, to facilitate their cooperation in designing the RL-based CC algorithms.
- **R2.** *High fidelity training and evaluation environment with ms-scale timing alignment.* Training and evaluation of an RL-based CC algorithm requires a high fidelity network environment that is controlled and reproducible.
- **R3.** *Validation under unseen network conditions.* Open, fair comparison between RL-based CC algorithms requires a public Internet testbed with diverse nodes and network types.

3 OPENNETLAB

3.1 Design Decisions

To satisfy the requirements **R1**-**R3** in section § 2.2, OpenNetLab is built with the following design decisions **D1**-**D3**.

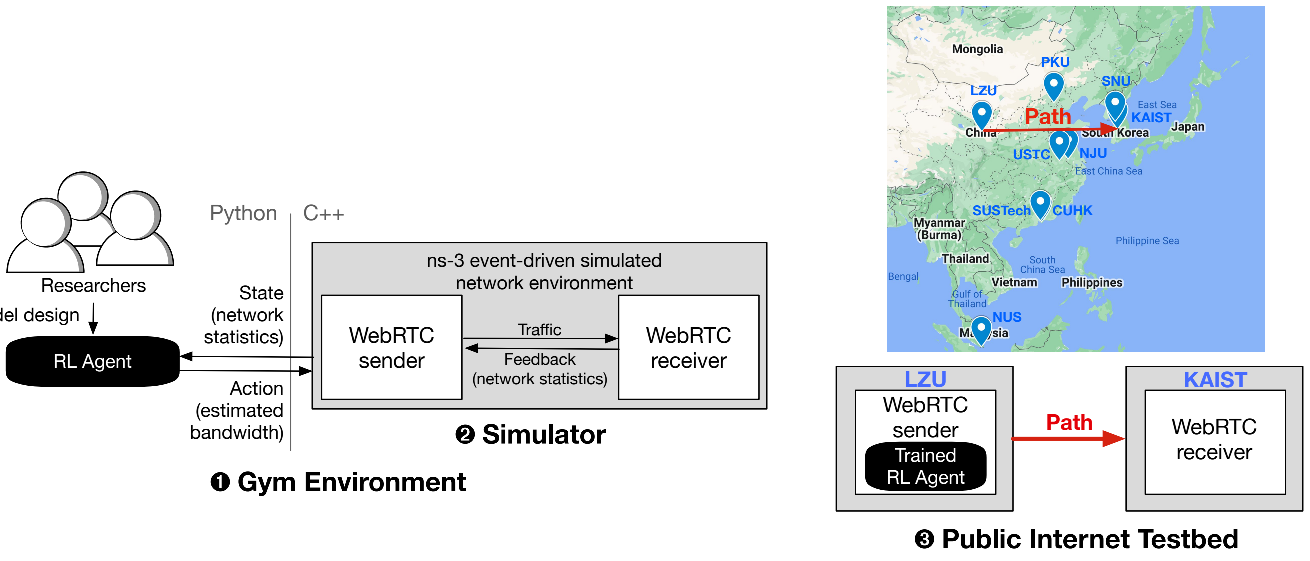


Figure 1: Overview of OpenNetLab.

- **D1.** *Gym environment decoupled from the details of RTC system internals for usability and programmability.* § 3.2
- **D2.** *Fast training and reproducible evaluation under high fidelity simulated network environment.* § 3.3
- **D3.** *Public Internet testbed for running customizable end-to-end RTC calls for performance validation.* § 3.4

Figure 1 shows the lifetime of an RL-based CC algorithm from model design to performance validation.

First, RL and networking researchers cooperate to design an RL-based CC algorithm using ❶ **Gym Environment**. The designed algorithm is trained in ❷ **Simulator** where the RL agent learns network environment by the network statistics sent from the WebRTC receiver, and produces estimated bandwidth as an action sent back to the WebRTC sender. The sender determines target bitrate based on the estimated bandwidth. The trained RL-based algorithm can be evaluated using the simulator in an inference mode by checking whether it reached a desired test reward. Finally, the trained algorithm is submitted to ❸ **public Internet testbed** to validate its network performance and QoE under unseen network conditions. Here, the trained model checkpoint in *.pth* format is shipped to WebRTC receiver, which provides network statistics to the agent whenever a packet arrives. We found that the overhead of receiving network statistics from the WebRTC receiver and producing action during inference is less than 1ms in most cases, which has negligible impact of end-to-end RTC call.

3.2 Model Design

Designing an RL-based CC algorithm for RTC requires expertise in both RL and RTC so that the algorithm can learn proper network states from the environment and make decisions to maximize the objective of achieving high network performance and QoE.

OpenNetLab decouples Python model interface from the WebRTC software stack to hide unnecessary details of WebRTC system internals so that RL researchers can focus on the model design. Specifically, Gym Environment is the Python interface that exposes network statistics such as receiver-side bitrate, delay and loss ratio. RL researchers can use these statistics to determine state, i.e. input to an RL agent, and design reward such that learning to make actions that maximize this reward would result in high network performance and QoE.

In RL, there are diverse learning algorithms to improve sample complexity (e.g. popular algorithms in policy gradient methods [26]), and training strategies to improve generalizability or robustness (e.g. transfer learning, curriculum learning or hierarchical RL [23]). With the Gym Environment, it is straightforward to build models using off-the-shelf RL training toolsets provided by widely used open-sourced libraries like OpenAI Gym [6] or Ray RLlib [21], which greatly improves usability and programmability in RL training.

3.3 Model Training and Evaluation

Offline training. To train the RL agent under the highest fidelity network environment, it may seem to be natural to

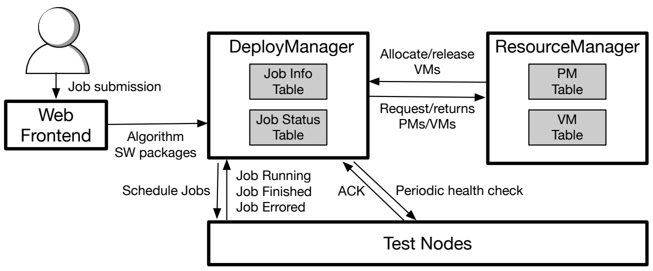


Figure 2: Architecture of OpenNetLab community testbed. White boxes represent stateful managers, and shaded boxes represent states.

choose online training in real network environment. However, as the online training involves playing real media traffic, it may take impractical amount of time for the model until convergence. For instance, learning hours of RTC sessions via online training can be reduced to several minutes of offline training under trace-driven packet-level simulator. More importantly, training an RL-based CC algorithm directly on a deployed RTC system can be unsafe, as the system becomes prone to sudden performance fluctuation due to the trial-and-error nature of the exploration in RL [32].

Packet-level event-driven simulation of network environment. To achieve fast yet high fidelity training under simulated network environment, the simulator creates real WebRTC sender and receiver instances and performs event-driven simulation of network environment with ns-3 [5]. With network traces obtained by probing the real network environment with measurement tools like iperf [3], the simulator schedules ns-3 events to simulate changing network environment (e.g. bandwidth, loss and jitter) as recorded in the trace.

3.4 Model Validation

In line with the potential of the data-driven, learning-based algorithms in networking research, there are also confusion on whether they actually outperforms their rule-based baselines or other competitors "in the wild". Measurement studies in TCP CC [30] and randomized experiments on adaptive bitrate controllers for VoD [29] continuously demonstrate that validating the learned algorithm's performance on public Internet under unseen network conditions is necessary to understand the performance characteristics of the learned algorithms and make fair comparisons.

Easy-to-use web-based frontend. Researchers submit a performance validation job by uploading their trained RL-based CC algorithms and specifying the required resources (compute node, network type) via a web-based frontend.

Centralized, stateful managers for job deployment, resource management, and job scheduling. When a new job is submitted, `DeployManager` records the job information (e.g. model checkpoint of the trained algorithm and software packages required to run it) and requested resources. When `ResourceManager` allocates available physical test nodes (PM) in the testbed and VMs, `DeployManager` schedules the job by running containerized WebRTC sender and receiver VMs on the allocated nodes in a FIFO manner. While the job is running, `DeployManager` tracks the state of the job and PMs and VMs where the job runs with periodic health check messages. The state of PMs and VMs are sent to `ResourceManager`. When the job finishes, `DeployManager` reports the event to `ResourceManager` to remove VMs from the nodes and reclaim the PMs.

Each manager stores its states in a cloud-based DB (Azure Storage) for fault tolerance. For example, `ResourceManager` maintains a `PM Table` and a `VM table` that record the health state of PMs and allocation state of VMs. `DeployManager` maintains a `Job Status Table` that records the status of all scheduled jobs, and `Job Info Table` that records all user-submitted job information needed to run the job. When a job fails, the job instance is re-created using the recorded job information and rescheduled to nodes that satisfy the user-submitted machine specification.

4 IMPLEMENTATION

We have implemented a prototype of OpenNetLab to verify the feasibility of our idea. It consists of the following parts: AlphaRTC, AlphaRTC Gym, community-driven public testbed, and backend microservices.

AlphaRTC. AlphaRTC is a full WebRTC stack with RL-based CC inference support (~2K LoC). Audio/video input and output features can be added to customize end-to-end calls.

AlphaRTC gym. AlphaRTC Gym corresponds to the Gym Environment for training (~3K LoC). We use ns-3 [5] for simulating network environment between an AlphaRTC sender that generates fake RTC packets to an AlphaRTC receiver. We use OpenAI Gym [6] as our gym interface.

Community-driven public testbed. Through active academic-industrial collaboration, we construct open public Internet testbed that includes wired, wireless and mobile networks and heterogeneous nodes with support from universities throughout Asia. The nodes are in China (Beijing, Hefei, Nanjing, Lanzhou, Shenzhen, and Hong Kong), South Korea (Seoul and Daejeon), and Singapore (Queenstown).

Backend microservices. Azure backend microservices coordinate the nodes in the testbed. For the stateful managers described in section § 3.4, we implemented microservice agents that communicate with REST APIs and deploy them on the cloud-based Kubernetes service. We use Ansible to communicate between the agents and nodes located across test sites. The agents are programmed by C# with 10K lines of code.

5 PRELIMINARY RESULT

To promote RL-based CC research, we hold a challenge on Bandwidth Estimation for RTC in MMSys [1] using OpenNetLab. The participants train their own RL-based CC algorithms with our Gym Environment and Simulator, and upload their models via the web frontend. The submitted algorithms are validated on the public Internet testbed with different network and video scenarios.

Scenarios. We select three network scenarios as in Table 1 and five video content scenarios with different types: animation, movie, conversation, presentation, and screen sharing over remote desktop. Each scenario is run 15 times in a round-robin way.

Evaluation metrics. We adopt widely used VMAF [20] video quality metric proposed by Netflix and DNSMOS [24] audio quality metric used in Microsoft as QoE metrics, in addition to the network metrics. Total score is a weighted sum of three scores (video score, audio score, and network score):

$$total\_score = w1 * video\_score + w2 * audio\_score + network\_score \tag{1}$$

$$video\_score = 100 * VMAF/groundtruth \tag{2}$$

$$audio\_score = {}^{1}\, if DNSMOS > groundtruth * binarize, 100 \; else 0 \tag{3}$$

$$network\_score = w3 * delay\_score + w4 * receive\_rate\_score + w5 * loss\_score \tag{4}$$

$$delay\_score = max(0, 100 * (max\_delay - delay\_95th) /(max\_delay{-}min\_one\_way\_delay)) \tag{5}$$

$$receive\_rate\_score = 100 * receive\_rate/groundtruth \tag{6}$$

$$loss\_score = 100 * (1 - loss\_rate) \tag{7}$$

The *groundtruth* in *video_score*, *audio_score* and *receive_rate_score* corresponds to the score obtained in an ideal environment (zero loss rate, high link capacity). We use the *groundtruth* score as the full marks. To tune the score function to produce scores that are inverse of the bandwidth estimation error, the values of the coefficients w1, w2, w3, w4, w5 are empirically set as 0.2, 0.1, 0.2, 0.2, 0.3, respectively.

[1]If an algorithm's DNSMOS score is larger than the *groundtruth* for *binarize* number of times, the *audio_score* is 100, and otherwise 0.

Evaluation result. We share performance results of two RL-based CC algorithms, Gemini and HRCC [27], which are the winner and runner-up of the challenge, respectively, compared to GCC [16].

3a shows that GCC achieves the highest video score in high and medium bandwidth scenarios. However, in the low bandwidth scenario, HRCC and Gemini outperform GCC. This demonstrates the potential of RL-based CC algorithms in covering the blind spots of rule-based GCC, which hardwires pre-defined rules of mapping packet events to network control decisions. 3b shows that in low bandwidth scenario, HRCC suffers from low delay score, while GCC suffers from low loss score, respectively. Conversely, in medium bandwidth scenario, HRCC suffers from low loss score, while GCC suffers from low delay score. This implies that HRCC can be further optimized by balancing the sending rate and packet loss in these two network scenarios in the Gym Environment. In 3c we can see Gemini ranks the 1st in total, but leaves room for improvement in the high bandwidth scenario. The audio score is 100 for all the three as audio transmission does not suffer from any bottlenecks.

6 RELATED WORKS

Training framework for RL-based networking systems. ns-3-gym [13] showed that fast training under simulated environment can achieve good model quality when adopting a high fidelity simulator like ns-3. Although similar in spirit, the simulator in OpenNetLab is tailored for training RL-based CC algorithms for RTC by connecting a customized gym and WebRTC with ns-3. Genet [34] is a training framework for RL for networking systems. It solves generalization problem inherent in RL with sample-efficient training environment selection mechanism via curriculum learning.

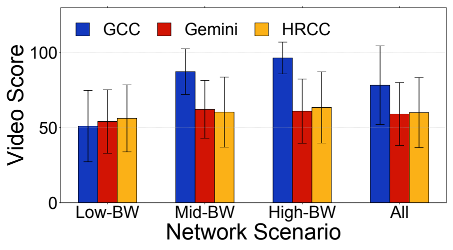
Public Internet testbed for networking research. Pantheon [30] and Puffer [29] demonstrated the benefit of building an open testbed with public Internet for research community. Concretely, Pantheon contributed to the design and evaluation of rule-based [8] to data-driven algorithms [12, 18, 19, 30] and measurement study [33]. Pantheon's success inspired OpenNetLab, where OpenNetLab aims to make similar contributions by providing tailored training and evaluation support for RL-based CC for RTC.

7 LIMITATIONS AND FUTURE WORK

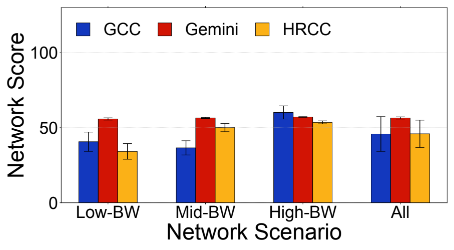
Management of a community-driven public Internet testbed. The public Internet testbed of OpenNetLab is still under construction. Scalable mechanisms to maintain a large number of test nodes is challenging, which is under active development. Supporting new use cases and network scenarios is also one of main items ongoing work. For example, supporting new network scenarios such as resource-constrained

Network scenario	Node 1	Node 2	Bandwidth range	Mean RTT
Low bandwidth	Beijing (Wi-Fi with weak signal)	Hong Kong (Wired)	<1Mbps	55ms
Medium bandwidth	Beijing (Mobile)	Hong Kong (Wired)	2 - 3Mbps	62ms
High bandwidth	Lanzhou (Wired)	Hong Kong (Wired)	>10Mbps	30ms

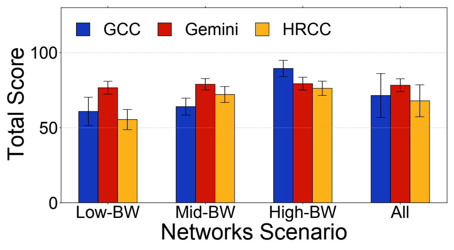
Table 1: Network scenarios.



(a) Video Score



(b) Network Score



(c) Total Score

Figure 3: Video, network and total scores in different network scenarios.

IoT devices connected to lossy networks would facilitate researches on the topic of energy efficient RTC.

Interpretability and robustness. OpenNetLab should include additional features that can help prevent learning-induced performance degradation. Learning-based algorithms faces inherent limitation in robustness and interpretability due to its non-deterministic behaviors. This is a major obstacle that prevent them from at-scale deployment in production systems, as a single instance of catastrophic QoE degradation may cause users to abandon the RTC application [31].

Open dataset and model zoo. Providing open datasets and model zoo will be valuable community resource for the research community. State-of-the-art RL-based CC algorithms, such as HRCC and Gemini in § 5, can benefit from open datasets in improving the algorithm quality and fair evaluation between algorithms. Model zoo with open model codes and pretrained checkpoints will facilitate advanced RL training strategies that require pretrained models such as transfer and curriculum learning [34].

High fidelity fine-tuning for pretrained RL-based CC algorithms. Even calibrated with real network traces, using simulated network environment only for learning patterns in network conditions faces limitations as it can only indirectly optimize QoE. We are planning to add emulator-based fine-tuning of RL-based CC that is pretrained on the simulator. We expect this high-fidelity fine-tuning phase can close the gap between the simulated and public Internet environment.

8 CONCLUSION

Recent potentials shown in data-driven, RL-based approaches in networking research motivate the needs of an open platform tailored for training and evaluation of RL-based CC for RTC, whose importance has never been more higher. We outline the requirements for this open platform, and propose an open platform for fast training, reproducible end-to-end evaluation, and performance validation on public Internet. Preliminary use cases confirm that OpenNetLab concretely aided the design and training of novel RL-based CC algorithms that outperform well-established rule-based baseline in both network performance and QoE.

ACKNOWLEDGMENTS

We thank the reviewers for their valuable comments. We also thank all OpenNetLab contributors (https://opennetlab.org/about) for substantial efforts on building and maintaining the prototype of OpenNetLab. This research was supported by the MSIT(Ministry of Science, ICT), Korea, under the High-Potential Individuals Global Training Program(2021-0-01696) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation), and another IITP grant funded by the MSIT (No.2015-0-00221).

REFERENCES

[1] 2021. Grand Challenge on Bandwidth Estimation for Real-Time Communications. (2021). https://2021.acmmmsys.org/rtc_challenge.php.
[2] 2022. Google Meet. (2022). https://meet.google.com.
[3] 2022. iPerf - The ultimate speed test tool for TCP, UDP and SCTP. (2022). https://iperf.fr.

[4] 2022. Microsoft Teams. (2022). https://www.microsoft.com/en-us/microsoft-teams/group-chat-software.

[5] 2022. ns-3 Network Simulator. (2022). https://www.nsnam.org.

[6] 2022. OpenAI Gym. (2022). https://gym.openai.com/.

[7] 2022. Zoom. (2022). https://zoom.us.

[8] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical delay-based congestion control for the internet. In *NSDI*.

[9] Per Block, Marion Hoffman, Isabel J Raabe, Jennifer Beam Dowd, Charles Rahal, Ridhi Kashyap, and Melinda C Mills. 2020. Social network-based distancing strategies to flatten the COVID-19 curve in a post-lockdown world. *Nature Human Behaviour* 4, 6 (2020), 588–596.

[10] Niklas Blum, Serge Lachapelle, and Harald Alvestrand. 2021. WebRTC: real-time communication for the open web platform. *Commun. ACM* 64, 8 (2021), 50–54.

[11] Francesca De Simone, Jie Li, Henrique Galvan Debarba, Abdallah El Ali, Simon NB Gunkel, and Pablo Cesar. 2019. Watching videos together in social virtual reality: An experimental study on user's QoE. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 890–891.

[12] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. 2018. {PCC} vivace: Online-learning congestion control. In *NSDI*.

[13] Piotr Gawłowicz and Anatolij Zubow. 2019. Ns-3 meets openai gym: The playground for machine learning in networking research. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. 113–120.

[14] Simon NB Gunkel, Hans M Stokking, Martin J Prins, Nanda van der Stap, Frank B ter Haar, and Omar A Niamut. 2018. Virtual Reality Conferencing: Multi-user immersive VR experiences on the web. In *Proceedings of the 9th ACM Multimedia Systems Conference*. 498–501.

[15] 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.

[16] Stefan Holmer, Henrik Lundin, Gaetano Carlucci, Luca De Cicco, and Saverio Mascolo. 2015. A Google Congestion Control Algorithm for Real-Time Communication. (2015). https://datatracker.ietf.org/doc/html/draft-alvestrand-rmcat-congestion-03

[17] Gazi Karam Illahi, Thomas Van Gemert, Matti Siekkinen, Enrico Masala, Antti Oulasvirta, and Antti Ylä-Jääski. 2020. Cloud gaming with foveated video encoding. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 16, 1 (2020), 1–24.

[18] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. 2019. A deep reinforcement learning perspective on internet congestion control. In *International Conference on Machine Learning*. PMLR, 3050–3059.

[19] Tong Li, Kai Zheng, Ke Xu, Rahul Arvind Jadhav, Tao Xiong, Keith Winstein, and Kun Tan. 2020. Tack: Improving wireless transport performance by taming acknowledgments. In *ACM SIGCOMM*.

[20] Zhi Li, Christos Bampis, Julie Novak, Anne Aaron, Kyle Swanson, Anush Moorthy, and JD Cock. 2018. VMAF: The journey continues. *Netflix Technology Blog* 25 (2018).

[21] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. 2018. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*. PMLR, 3053–3062.

[22] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 197–210.

[23] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. 2021. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–35.

[24] Chandan KA Reddy, Vishak Gopal, and Ross Cutler. 2021. Dnsmos: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6493–6497.

[25] Saeed Shafiee Sabet, Steven Schmidt, Saman Zadtootaghaj, Carsten Griwodz, and Sebastian Möller. 2020. Delay sensitivity classification of cloud gaming content. In *Proceedings of the 12th ACM International Workshop on Immersive Mixed and Virtual Environment Systems*. 25–30.

[26] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).

[27] 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 *ACM MMSys*.

[28] Mowei Wang, Yong Cui, Xin Wang, Shihan Xiao, and Junchen Jiang. 2017. Machine learning for networking: Workflow, advances and opportunities. *IEEE Network* 32, 2 (2017), 92–99.

[29] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 495–511.

[30] 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 *USENIX ATC*.

[31] 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 *MobiCom*.

[32] 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 *MobiCom*.

[33] Huanhuan Zhang, Anfu Zhou, Ruoxuan Ma, Jiamin Lu, and Huadong Ma. 2021. Arsenal: Understanding Learning-based Wireless Video Transport via In-depth Evaluation. *IEEE Transactions on Vehicular Technology* 70, 10 (2021), 10832–10844.

[34] Zhengxu Xia, Yajie Zhou, Francis Y. Yan, Junchen Jiang. 2022. Automatic Curriculum Generation for Learning Adaptation in Networking. *arXiv preprint arXiv: 2202.05940* (2022).

[35] 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 *MobiCom*.