Pioneer: Offline Reinforcement Learning based Bandwidth Estimation for Real-Time Communication

Bingcong Lu
Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
irene_lu@sjtu.edu.cn

Keyu Wang
Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
readerek@sjtu.edu.cn

Jun Xu
Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
xujunzz@sjtu.edu.cn

Rong Xie
Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
xierong@sjtu.edu.cn

Li Song
Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
song_li@sjtu.edu.cn

Wenjun Zhang
Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
zhangwenjun@sjtu.edu.cn

ABSTRACT

For Real-time Communication (RTC), Bandwidth Estimation (BWE) is crucial for enhancing user Quality of Experience (QoE) by ensuring efficient bandwidth utilization and low latency. Recent advancements have shifted towards machine learning based algorithms, particularly online reinforcement leanring (RL), to dynamically infer future bandwidth using statistical data. However, challenges such as dependency on training settings, the necessity for extensive trial and error, and instability in complex state spaces hinder their efficacy. To address these limitations, we propose Pioneer, a novel offline RL framework for BWE in RTC systems. Unlike its predecessors, Pioneer eliminates the need for real-time environment interaction during training and achieves good performance through lightweight training. Our framework consists of a Trajectory Sampler for state information preprocessing and a Bandwidth Estimator based on offline RL model. Our test results on offline datasets show that Pioneer can achieve better performance than expert algorithms. We also tested Pioneer on online simulation platforms, and Pioneer can improve QoE by 9% compared to other offline algorithm, demonstrating good robustness.

CCS CONCEPTS

• **Networks** → **Network algorithms**; *Network protocols*; Network services;

KEYWORDS

Real-Time communication, Bandwidth estimation, Offline reinforcement learning, Behavior Clone, Network control

ACM Reference Format:
Bingcong Lu, Keyu Wang, Jun Xu, Rong Xie, Li Song, and Wenjun Zhang. 2024. Pioneer: Offline Reinforcement Learning based Bandwidth Estimation for Real-Time Communication. In *ACM Multimedia Systems Conference 2024 (MMSys '24), April 15–18, 2024, Bari, Italy*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3625468.3652174

MMSys '24, April 15–18, 2024, Bari, Italy

ACM ISBN 979-8-4007-0412-3/24/04
https://doi.org/10.1145/3625468.3652174

1 INTRODUCTION

In recent years, applications such as video conferencing and remote technical support have played a significant role in people's daily life. Bandwidth estimation (BWE) is one of the hot topics in Real-time Communication (RTC) technology. In the RTC system, the result of BWE affects the encoding bitrate and the pacer sending rate, to adapt to changing network environments. Therefore, accurate BWE is crucial in enhancing bandwidth utilization while ensuring low latency, which is of vital importance to improve users' quality of experience (QoE).

Many approaches have been proposed to tackle BWE challenge. Basically, they all follow a common principle: adaptively obtaining bandwidth estimation results through the statistical status of data packets. These algorithms can be divided into two types: traditional heuristic algorithms and machine learning based algorithms. Among heuristic algorithms, GCC[5] is the most representative method, which uses two estimators distributed at both ends of the peer to consider loss and delay together, and obtains bandwidth estimation results through fixed rules and state machine switching. However, heuristic algorithms use limited low-level information and finite state machine , resulting in poor accuracy of results and weak adaptive capabilities.

Therefore, researchers have drawn more attention to machine learning based algorithms, especially reinforcement learning (RL). R3Net[8], OnRL[24] and CLCC[16] propose different online RL models to infer future bandwidth. They all use statistical information as state and exact bandwidth or changing gain as action. However, these algorithms also suffer from the basic problems of RL. Firstly, model performance is closely related to the training setting such as reward function. Meanwhile, online RL training requires lots of trial, and may exhibit severe instability in the process, especially when dealing with high-dimensional, nonlinear state spaces. In order to adopt RL better, a common way to improve its performance is to find a "mentor" for it. HRCC[21] and Bob[3] use hybrid mechanisms to constrain the output of RL through heuristic algorithms. On the other hand, Eagle[6], Dugu[14] and Loki[23] use

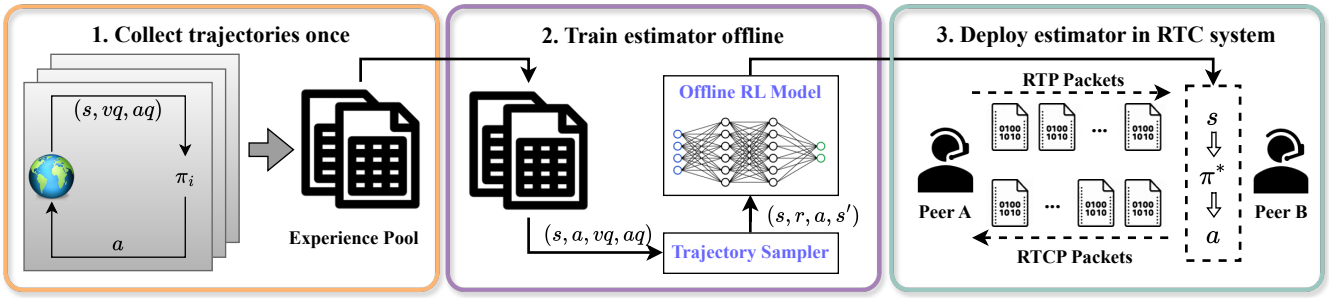

Figure 1: Basic process of offline reinforcement learning: From training to deployment

Imitation Learning (IL) to learn policy by observing the behaviors of experts. But IL depends on high-quality expert demonstrations, and find the oracle of BWE task is virtually impossible. In addition, all algorithms based on online RL face a huge challenge, which is the need for continuous interaction with the environment, which may be costly and time-consuming.

To break through the above limitations, offline methods Sage[22] and Merlin[12] have recently been proposed. The learning process of offline RL entirely based on pre-collected data sets rather than through real-time interaction. This approach is particularly useful in scenarios where real-time interaction is costly or risky.Sage is the first framework to introduce data-driven RL to address TCP congestion control problems. Sage generates a experience pool of trajectories based on existing congestion control strategies (Cubic[13], BBR[4], etc.) and employs the Actor-Critic architecture[15] for model learning, which achieves better performance than its mentors. Merlin is the first to use offline imitation learning to estimate bandwidth in RTC system. It collects behaviors from the expert UKF and achieve better subjective quality, while it is still limited by expert and lacks exploratory ability.

Compared to offline IL, offline RL is guided by reward signals, allowing model to identify and optimize for long-term goals. It does not just learn to imitate the behaviors of experts, but also tries to surpass these behaviors to achieve higher cumulative returns. Inspired of Sage and Merlin, we proposed **Pioneer**, a novel offline RL framework to estimate network bandwidth in RTC system. Pioneer mainly consist of a Trajectory Sampler that pre-process the state information and re-build experience pool, and a Bandwidth Estimator based on offline reinforcement learning architecture[11][9]. By learning from dataset of previously collected experiences, real-time interaction with the environment is no longer required during the training process. Pioneer is the first work to demonstrate and realize the offline estimator. Besides, extensive testing is conducted in emulated dataset and emulated environment. Compared with benchmark estimators, the offline RL model Pioneer showed better QoE. We release our final model and source code at [17].

2 CHALLENGE

As shown in Fig 1, the basic process of offline RL can be concluded as three parts. It starts with data collection which is collecting historical interaction data between expert strategies and the environment. These data usually include state, action, and reward. All data generated constitutes the experience pool. The next step is to conduct offline strategy learning, using the data from the experience pool to train a decision model that can select the optimal action to maximize long-term rewards under given states. After obtaining the offline model, we can deploy the strategy into online applications to guide decision-making. For bandwidth estimation, the target of offline RL is to train a decision model based on pre-collected network statistical data and expert estimator behavior, which should maximize QoE and has high robustness for practical RTC systems. To achieve the above goals, there are three challenges:

2.1 Data Collection

For offline RL, a high-quality experience pool is critically important. Firstly, the pool must contain various characteristics. Although the statistical analysis of network states typically revolves around receiving rate, latency, and loss rate, each aspect should undergo further processing. For instance, latency gradient is more directly reflect changes in network conditions. Within the monitor interval (MI) with a high proportion of video packets, the attenuation of indicators will more severely damage the user experience.

Additionally, for time-series data, the contribution of different timescales to rewards varies. In Sage[22], information from three different MI is used to balance short-term target(rapidly tracking changes in available bandwidth) with long-term objectives (TCP friendliness).

Lastly, the dataset should be enriched with guidance from a well-versed team of experts, ensuring comprehensive guidance for the algorithm and preventing performance drift.

Considering these aspects, the 2024 ACM MMsys' Grand Challenge provides an excellent dataset[18], which includes data from 18,859 calls and 9,405 emulated test calls. Observations contain 150 network features collected at two different timescales, along with expert behaviors from UFK, GCC, and other ML-based strategies. Pioneer is standing on the shoulders of this giant.

2.2 Behavior Learning

In traditional online RL, learning purely based on reward functions may lead to high-variance value function estimation, especially when the state and action space are high dimension or complex. In offline RL for solving BWE problems, if only regard Q value as optimization objective, the problem of policy being difficult to converge will be more severe. First of all, the dimension of network

features is very high, and the prediction value of bandwidth is continuous. In this case, compared to a complete space of decision sequences, the limited trajectories in the experience pool can be considered sparse. Therefore, the agent cannot efficiently learn what strategy will bring the highest returns, and it is easily misled by some extreme situations.

All expert algorithms have been tested in practice and can provide RL agent with good guidance. Therefore, it can serve as a constraint to prevent the model from deviating excessively from these validated strategies. However, experts in the experience pool specialize in different network environments and may not always perform well. Therefore, it is necessary to design appropriate learning mechanisms to help RL agent to imitate and surpass experts.

2.3 Extrapolation Error

"Extrapolation error" [11]is a crucial issue that often arises when the algorithm attempts to make predictions on state-action that are not fully covered in the training data. In offline RL, the agent is trained on a static dataset of previously collected experiences, and it does not interact with the environment during training. Thus, when a new state-action pair appears, the model unable to estimate accurate value. Once the estimation of Q is too high, the agent will learn a strategy that may benefit very poorly in actual interactions, leading to a deterioration in the overall performance of the model.

This problem also exists when offline RL is applied to bandwidth estimation. Even though our trajectories are collected from large amount of real communication cases, the experience pool cannot cover all network state - bandwidth action pairs, so there are still a lot of unknown situations in the trajectory space. In addition, in different applications, the physical condition of network and the behaviors of users are irregular and vary greatly, which means there is a serious disturbance shift problem. Therefore, it is very challenging for offline models to extract a robust representation from existing data without any further interaction with the environment.

3 PIONEER

We proposed Pioneer, a offline RL based RTC bandwidth estimator. Pioneer mainly conclude two components: a trajectory sampler and a offline RL model. The main purpose of the former is to sample from observation vectors and expert actions, calculate returns, and obtain training batch $\mathcal{B}$. The latter is the actual bandwidth estimator, and its design is inspired by both Batch-Constrained Deep Q-Learning (BCQ)[11] and Twin Delayed Deep Deterministic policy gradient algorithm with behavior clone (TD3+BC)[9].

3.1 Trajectory Sampler

As mentioned before, offline RL extracts the optimal policy from expert trajectories. Agents need to understand the relationship between states and actions, as well as how actions affect the environment (i.e. transition of states) and rewards. Usually, agents need to learn from the current state s, rewards r, expert behavior a, and the next state s' to transfer to. $[s, a, s']$ can be easily obtained from the dataset, while r needs to be customized.

Reward. The reward is considered as a scalar to evaluate the effectiveness of the BWE schemes.The higher the packet delivery rate perceived by the agent, usually means the better the audio and video quality. Increased packet loss and latency will harm the user experience, and the agent should be punished at this time. Based on the above considerations and the experience of other research work[21][16], we use following reward at timestep t:

$$R_t = \alpha U_t - \beta L_t - \gamma D_t \tag{1}$$

where U_t, L_t, D_t is the average receiving rate, packet loss rate and queuing delay within the recent short-term MI (60ms). α, β, γ are set to 0.5, 0.3, 0.3.

The Trajectory Sampler module extracts a batch of $[s, a]$ data from the experience pool, and after the above calculation, output $[s, a, r, s']$ quadruples as batch $\mathcal{B}$, which will be provided to the offline RL model for training. In addition, this also provides a basis for implementing more intelligent samplers in the future. For example, in future work, this module can be used for data cleaning or trajectory stratification.

3.2 Offline Reinforcement Learning Model

To address the challenges of applying offline RL to BWE, we build Pioneer on top of BCQ[11]. The main advantage of Pioneer is making the result of bandwidth estimation more accurate and robust by restricting the state-action pairs selected by the policy to be similar to the data existing in the sampled batch $\mathcal{B}$. For a given observation vector, Pioneer uses a generative model to generate a set of bandwidth candidates similar to the distribution of batch, and selects the one with the highest value through the Q network. In addition, inspired of TD3+BC[9], we further add a behavior clone term to assist in the learning of the actor network. This strategy can effectively prevent the model from deviating excessively from these validated strategies during the training process, especially when dealing with such complex, high-dimensional network features and continuous bandwidth estimation.

Specifically, Pioneer includes the following three components:

VAE Network. The Variational Autoencoder (VAE) network $G_\omega = \{E_{\omega_1}, D_{\omega_2}\}$, including an Encoder E_{ω_1} and a DecoderD_{ω_2}. The encoder learns the representation of network feature vectors and network estimation results $[s, a]$, proving the mean μ and standard deviation σ of the Gaussian distribution $N(\mu, \sigma)$. Furthermore, the encoded feature embedding can be constantly decoded into recoverable observations by decoder. By learning latent spaces, VAE can help agents better understand and generalize to unseen states.

Actor Network. Firstly the VAE mentioned above is a part of the actor. By learning the data distribution within the batches, a basic prediction result a is decoded based on the current observations. In addition, a perturbation network ξ_ϕ is added, using a two-layer fully connected network to further perturb the output of the VAE to the range of $[-\Phi + a, \Phi + a]$, which can be seen as a slight exploration. As for now, the policy π of Pioneer can be represented as (2). In our work, we set the $\Phi = 0.05$, while controlling the final action after perturbation to remain between [0,1], to ensure that a bandwidth estimation of $[10K, 8M]$bps can be obtained through the inverse mapping of (4).

$$\pi(s) = \underset{a_i+\xi(s,a_i;\Phi)}{\arg\max} Q_\theta(s, a_i + \xi(s, a_i; \Phi)), \quad \{a_i \sim G_\omega(s)\}_{i=1}^{n} \tag{2}$$

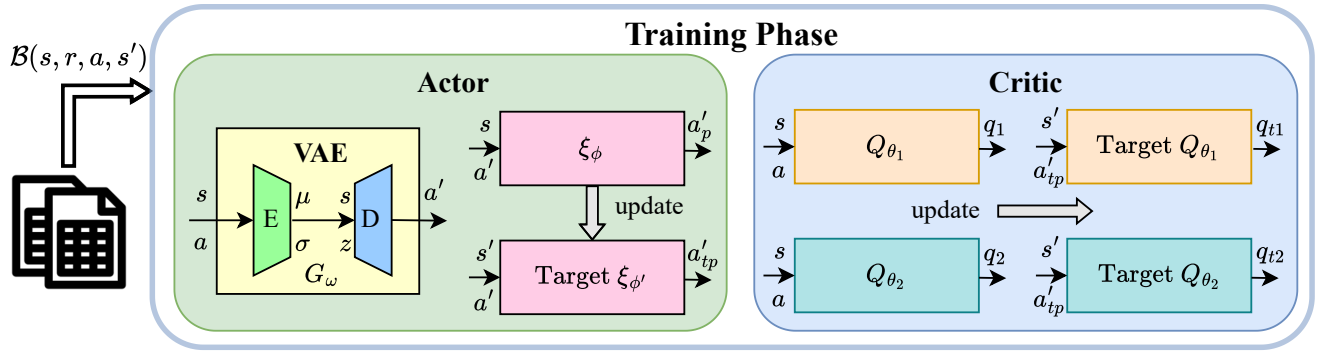

Figure 2: Pioneer Training Part. Pioneer follows the basic Actor-Critic architecture. The Actor first uses VAE to learn the data distribution and generates output candidates with similar distribution; then it performs slight exploration through the perturbation network to obtain the final result. Critic network uses two value networks for Q-value estimation.

On this basis, we add an additional behavior clone term to the actor. The actor loss we ultimately determined is shown as (3), where a is current action and $\hat{a}$ is the expert's behavior. The main purpose is to ensure that the actor does not deviate too much from the guidance provided by experts while striving to maximize returns. With this limitation, the training process of the model accelerates convergence, and the output stability of the model is greatly improved.

$$L_a = -Q_\theta(s, a_i + \xi(s, a_i; \Phi)) + \frac{1}{n}\sum_{i=1}^{n}(a_i + \xi(s, a_i; \Phi) - \hat{a}_i)^2 \quad (3)$$

Critic Network. The critic network of Pioneer is used to evaluate the value of state-action pairs . Our model is based on the Clipped Double Q-learning algorithm[10], which trains two action value networks $Q_{\theta_1}, Q_{\theta_2}$ and takes their minimum values as the estimation of action values, avoiding overestimation of Q.

In training phase, the model architecture is shown in Figure 2 and the detailed process is shown as Algorithm 1. Batch data $\mathcal{B}$ sampled by trajectory sampler is firstly sent into VAE to learn the distribution of current data, and generate similar actions. Actors will pursue maximum returns under the guidance of expert behavior. That is to learn how to make reasonable network bandwidth estimation based on the current network observations, in order to achieve the best rate-delay performance. Critic will be committed to learning how to correctly evaluate current actor performance.

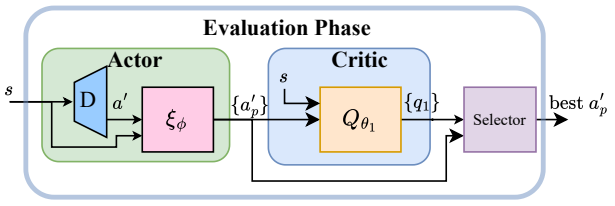

Figure 3: Pioneer Evaluation Part.

Algorithm 1: Training Algorithm of Pioneer

Input: input batch $\mathcal{B}(s,a,r,s')$, horizon T, target network update rate τ ,weighting rate λ,max perturbation Φ, sampling numbers n

Initialize Q-networks $Q_{\theta_1}, Q_{\theta_2}$, perturbation network ξ_ϕ, VAE $G_\omega = \{E_{\omega_1}, D_{\omega_2}\}$ with random parameters $\theta_1, \theta_2, \phi, \omega$, and target networks $Q_{\theta'_1}, Q_{\theta'_2}, \xi_{\phi'}$ with $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$.

for *t=1 in T* **do**

1. update VAE : from the transitions $\mathcal{B}(s,a,r,s')$ obtain the target action distribution
   $\mu, \sigma = E_{\omega_1}(s,a)$, $\tilde{a} = D_{\omega_2}(s,z)$, $z \sim \mathcal{N}(\mu,\sigma)$
   $\omega \leftarrow \arg\min_\theta \sum(a-\tilde{a})^2 + D_{kl}(\mathcal{N}(\mu,\sigma)||\mathcal{N}(0,1))$;
2. sample n actions :
   $\{a'_i = G_\omega(s')\}_{i=1}^n$;
3. perturb each action :
   $\{\overline{a}_i = a'_i + \xi_{\phi'}(s,s',\Phi)\}_{i=1}^n$;
4. set value target y :
   $y = R_i + \gamma \max_{a_i}[\lambda \min_{j=1,2} Q_{\theta'_j}(s',a_i) + (1-\lambda)\max_{j=1,2} Q_{\theta'_j}(s',a_i)]$
5. update critic network with θ :
   $\theta \leftarrow \arg\min_\theta \sum(y - Q_\theta(s,a))^2$;
6. update actor network with ϕ :
   $\phi \leftarrow \arg\min_\phi \sum[(a-\overline{a})^2 + Q_{\theta_1}(s, \overline{a} + \xi_\phi(s,\overline{a},\Phi))]$;
7. update target networks :
   $\tau' \leftarrow \gamma\theta + (1-\tau)\theta', \quad \phi' \leftarrow \tau\phi + (1-\tau)\phi'$

end

In evaluating phase, Pioneer first repeats the input $n(n = 100)$ times and sends it to the Actor to generate n actions, then selects the action with the highest value through Critic to execute. This process is to give a set of possible bandwidth values under the current status, and select the value that can maximize user QoE as the final output. The model architecture is shown in Figure 3.

4 IMPLEMENTATION

We implement Pioneer by using Pytorch, and further export Pioneer to ONNX model. We use all data of testbed dataset and 80% data of emulated dataset as training input. Another 20% data of emulated dataset are used to evaluate. **Note** that In order to maintain the consistency of the data length in the batch and minimize data discarding, unlike random sampling methods, we first sort the sequences in order of length to make the data length within a batch as close as possible, and crop the sequences based on the shortest length. The use of cropping instead of padding zeros is to prevent the agent from learning incorrect state transitions at the end stage.

Furthermore, inspired by previous work[3][12], the output representation is set between [10K, 8M]bps, because the bandwidth within this range can cover various RTC scenarios such as simple audio calls and high-definition (HD) video conferencing. We further use the formula (4) to map actions to the [0,1] interval.

$$a_{\log} = \frac{\log(a) - \log(b_{\min})}{\log(b_{\max}) - \log(b_{\min})} \tag{4}$$

where $b_{\max}$ is set to 8000000 and $b_{\min}$ equals to 10000.

Offline training. Pioneer is trained with batch size of 25. In each epoch, we traverse all the training data, which means 1055 batch will be sampled. The learning rate of Actor is 10^{-4} and Critic is 10^{-8}. Through the methods we mentioned before to improve training efficiency, Pioneer has very low training costs and fast inference speeds. Pioneer can perform well after only 11 epochs of training. After pre-loading all training data into memory, Pioneer only needs 30 minutes to train on the NVIDIA GeForce RTX 4090. The stable model we ultimately chose trained 258 epochs, which takes about 11 hours.

Online Testing. To verify the robustness and practicality of Pioneer, we tested it using a emulated RTC platform, AlphaRTC[7]. AlphaRTC will conduct a real one-way RTC communication, transfer local files to the client, and save the received media data as files. This allows us not only to evaluate the performance of the network layer based on the statistical information of the data packets, but also to measure the actual media quality.

We use one 1280x720 video conference scene video FourPeople.yuv from Xiph.org[1] as input (without audio), use Mahimahi[20] as the network emulator and adopt 22 LTE traces observed from the real world[2]. Each call lasts 60s. We conducted the same observation collection according to the data instructions published by the organizers of the challenge[18], which means AlphaRTC will provide feedback on the same 150 network state statistics after each feedback duration (200 ms).

5 EVALUATION

We test Pioneer through two different methods: offline and online. Through **offline testing**, we seek to answer the following question: If Pioneer can surpass the limitations of experts and provide more accurate estimation that can maximize QoE.

Benchmarks. We compare Pioneer with two different kind mechanisms.

(1) Expert-policy are strategies that provided for each sample in the dataset, obtained from UKF, GCC, and other ML based algorithms;
(2) Merlin [12] is a offline, data-driven solution based on imitation learning. We retrained Merlin using the network structure provided in their paper.

Metrics. Due to the fact that we can only obtain bandwidth offline, we evaluate the performance in the following two aspects:

(i) Average Bandwidth utilization U_{avg} : It is defined as the ratio between the average estimation of Pioneer and the average true capacity.
(ii) Overshoot ratio R_{os}: [12] states that overshoot poses greater harm to QoE than undershoot. To measure whether the algorithm blindly increases bandwidth and leads to amounts of overshoots, we define this indicator R_{os} refers to the proportion of actions where the predicted value is higher than the actual bandwidth capacity among all decisions in one call.

For **online testing**, we try to find if the model trained offline can still have stable and superior performance when applied to actual RTC system.

Benchmarks. We compare Pioneer with Merlin and BoB[3]. BoB is a hybrid estimator with combined RL model with heuristic algorithm, which is the SOTA scheme on the AlphaRTC benchmark.

Metrics. We compute overall QoE of Pioneer during online testing. The design of QoE is given by ACM MMSys'21 Grand Challenge[19]. As shown in (5), Network score is a combination of throughput , packet delay and loss rate. Video score is based on VMAF.

$$\begin{aligned} S_{\text{delay}} &= 100 \times \frac{d_{\max} - d_{95\text{th}}}{d_{\max} - d_{\min}} \\ S_{\text{loss}} &= 100 \times (1 - L) \\ S_{\text{rate}} &= 100 \times U \\ S_{\text{network}} &= 0.2S_{\text{delay}} + 0.3S_{\text{loss}} + 0.2S_{\text{rate}} \\ S_{\text{video}} &= \text{VMAF} \end{aligned} \tag{5}$$

where $d_{\max}$, $d_{\min}$, $d_{95\text{th}}$, L, U respectively denote the maximum, the minimum and the 95th percentile queuing delay, packet loss ratio and bandwidth utilization. The overall QoE is defined as:

$$\text{QoE} = 0.5S_{\text{network}} + 0.5S_{\text{video}} \tag{6}$$

5.1 Offline Testing

We tested Pioneer and Merlin on valid dataset, predicting bandwidth from 1881 call records. The U_{avg} and R_{os} results are shown in Table 1. Note that, when calculating the average bandwidth utilization, the portion exceeding 100% is uniformly counted as 100%. Results show that Pioneer and Merlin achieve almost same accuracy as Expert, which means both algorithms have learned well on offline datasets. Besides, due to our setting of the BC item, even though Pioneer designed the reward to pursue higher bandwidth, there was no problem of blind increase.

We also provided 4 samples to further demonstrate Pioneer's performance. As shown in Figure 4, Pioneer can achieve good performance in different bandwidth environment. Although Pioneer's behavior is still obviously influenced by expert, it can approach

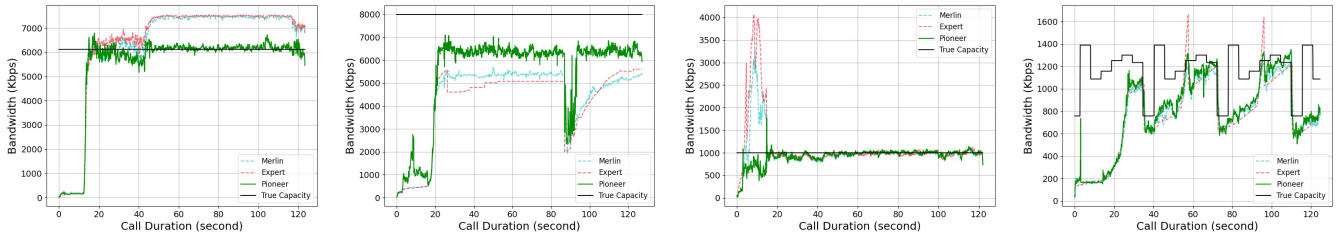

Figure 4: The bandwidth estimation results of Pioneer for different network profiles.

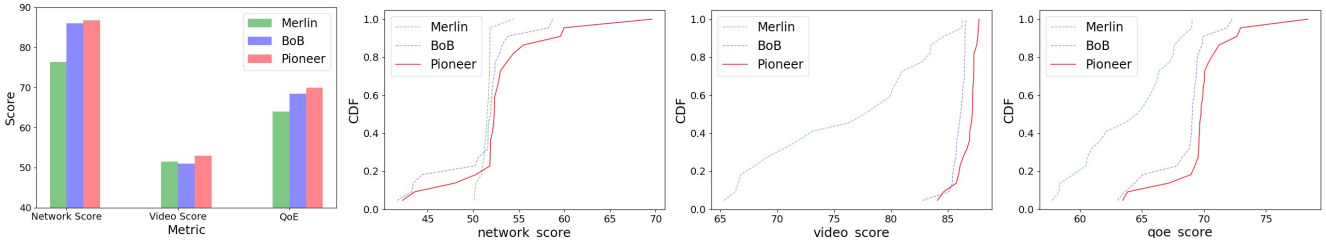

Figure 5: The online testing results of Pioneer. The first graph shows the average score of the schemes on the metrics, and the three graphs on the right show the cumulative distribution of each metric.

bandwidth more closely overall. And Pioneer's results are more stable, which can avoid following the abnormal behavior of experts (burr-like values) to some extent.

Table 1: The offline testing results on average bandwidth utilization and overshoot ratio.

Metric	Expert	Merlin	Pioneer
$U_{avg} \uparrow$	81.98%	81.77%	81.94%
$R_{os} \downarrow$	23.26%	21.76%	22.61%

5.2 Online Testing

We apply Pioneer, Merlin and BoB on the emulated platform AlphaRTC for one-way RTC communication based on local video files, and the network conditions were set by 22 different network profiles collected in real-world. In particular, during actual RTC transmission, if the value provided by the estimator is too small, the connection will be interrupted. So in this test, the lower limit of the estimator output is 300Kbps. The experimental results are shown in Figure 5. It can be found that Pioneer shows excellent performance in actual RTC applications. The average QoE score of Merlin, BoB and Pioneer are(63.91, 68.46, 69.84), so Pioneer obtains a QoE gain of (9.2%,2.0%) with respect to Merlin and BoB. Furthermore, as can be seen from the Figure 5, the improvement of QoE is not a trade-off, but rather win-win cooperation in video quality and latency. Besides, through the CDF charts, we can observe that Pioneer's performance is stable in the new environment and can be well maintained at a high level.

As two offline algorithms, Pioneer and Merlin, there is a significant difference in the distribution of the data we tested compared to the training data. IL-based Merlin has poor adaptability to unknown environments, while Pioneer can adapt well to this domain shift due to its use of batch constrained architecture. Even in the face of an unseen network environment, it can provide the same stability as the heuristic algorithm. Compared to RL-based algorithms BoB, the performance of Pioneer trained on offline datasets is also relatively better, and our training costs are much lower. It can be believed that as we further increase the scale of training data or introduce Bob as an expert to guide Pioneer, the performance of Pioneer can be further improved.

6 CONCLUSION

We proposed Pioneer, a offline RL framework based on some excellent models of predecessors. Pioneer is a successful application of offline RL in BWE problems, from setting states, actions, and rewards, to offline training based on pre-collected datasets, and finally to practical deployment in online real-time communication. Pioneer is the first work to complete above demonstration. The evaluation of Pioneer in both emulated datasets and environments showcases its ability to significantly enhance the Quality of Experience (QoE) for users. The findings underscore the potential of offline RL in improving the adaptability and accuracy of BWE mechanisms, suggesting a promising direction for future research and development in RTC technologies.

ACKNOWLEDGMENTS

This work was supported by National Key R&D Project of China (2019YFB1802701); in part by the Fundamental Research Funds for the Central Universities; in part by the STCSM under Grant 22DZ2229005; in part by the 111 project under Grant BP0719010.

REFERENCES

[1] [n. d.]. Xiph.org Video Test Media [derf's collection]. [Online]. https://media.xiph.org/video/derf/

[2] Soheil Abbasloo, Chen-Yu Yen, and H Jonathan Chao. 2020. Wanna make your TCP scheme great for cellular networks? Let machines do it for you! *IEEE Journal on Selected Areas in Communications* 39, 1 (2020), 265–279.

[3] Abdelhak Bentaleb, Mehmet N Akcay, May Lim, Ali C Begen, and Roger Zimmermann. 2022. BoB: Bandwidth Prediction for Real-Time Communications Using Heuristic and Reinforcement Learning. *IEEE Transactions on Multimedia* (2022).

[4] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue* 14, 5 (2016), 20–53.

[5] 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.

[6] Salma Emara, Baochun Li, and Yanjiao Chen. 2020. Eagle: Refining congestion control by learning from the experts. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 676–685.

[7] Jeongyoon Eo, Zhixiong Niu, Wenxue Cheng, Francis Y Yan, Rui Gao, Jorina Kardhashi, Scott Inglis, Michael Revow, Byung-Gon Chun, Peng Cheng, et al. 2022. Opennetlab: Open platform for rl-based congestion control for real-time communications. (2022), 70–75.

[8] Joyce Fang, Martin Ellis, Bin Li, Siyao Liu, Yasaman Hosseinkashi, Michael Revow, Albert Sadovnikov, Ziyuan Liu, Peng Cheng, Sachin Ashok, et al. 2019. Reinforcement learning for bandwidth estimation and congestion control in real-time communications. *arXiv preprint arXiv:1912.02222* (2019).

[9] Scott Fujimoto and Shixiang Shane Gu. 2021. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems* 34 (2021), 20132–20145.

[10] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.

[11] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*. PMLR, 2052–2062.

[12] Aashish Gottipati, Sami Khairy, Gabriel Mittag, Vishak Gopal, and Ross Cutler. 2023. Real-time Bandwidth Estimation from Offline Expert Demonstrations. *arXiv preprint arXiv:2309.13481* (2023).

[13] 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.

[14] Tianchi Huang, Chao Zhou, Lianchen Jia, Rui-Xiao Zhang, and Lifeng Sun. 2022. Learned Internet Congestion Control for Short Video Uploading. In *Proceedings of the 30th ACM International Conference on Multimedia*. 3064–3075.

[15] Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems* 12 (1999).

[16] 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.

[17] Bingcong Lu, Keyu Wang, and Jun Xu. 2024. Pioneer source code. [Online]. https://github.com/sjtu-medialab/Pioneer

[18] Microsoft. 2024. 2024 ACM MMsys Grand Challenge on Offline Reinforcement Learning for Bandwidth Estimation in Real Time Communications. [Online]. https://2024.acmmmsys.org/gc/ORL-BWE-RTC/

[19] Microsoft and OpenNetLab. 2021. 2021 ACM MMsys Grand Challenge on Bandwidth Estimation for Real-Time Communications. [Online]. https://2021.acmmmsys.org/rtc_challenge.php

[20] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate {Record-and-Replay} for {HTTP}. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. 417–429.

[21] 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.

[22] Chen-Yu Yen, Soheil Abbasloo, and H Jonathan Chao. 2023. Computers Can Learn from the Heuristic Designs and Master Internet Congestion Control. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 255–274.

[23] 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.

[24] 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.