Adversarial Attacks on Federated-Learned Adaptive Bitrate Algorithms

Rui-Xiao Zhang[*†1], **Tianchi Huang**[*‡2]

[1]The University of Hong Kong [2]Sony Group Corporation
zrxhku@hku.hk, tianchi.huang@sony.com

Abstract

Learning-based adaptive bitrate (ABR) algorithms have revolutionized video streaming solutions. With the growing demand for data privacy and the rapid development of mobile devices, federated learning (FL) has emerged as a popular training method for neural ABR algorithms in both academia and industry. However, we have discovered that FL-based ABR models are vulnerable to model-poisoning attacks as local updates remain unseen during global aggregation. In response, we propose MAFL (*M*alicious *A*BR model based on *F*ederated *L*earning) to prove that backdooring the learning-based ABR model via FL is practical. Instead of attacking the global policy, MAFL only targets a single "target client". Moreover, the unique challenges brought by deep reinforcement learning (DRL) make the attack even more challenging. To address these challenges, MAFL is designed with a two-stage attacking mechanism. Using two representative attack cases with real-world traces, we show that MAFL significantly degrades the model performance on the target client (i.e., increasing rebuffering penalty by $2\times$ and $5\times$) with a minimal negative impact on benign clients.

Introduction

Video streaming plays a dominant part in the networking field. As reported in SANDVINE (Sandvine 2022), videos have taken over 65.93% traffic in the first half of 2022. Providing better Quality of Experience (QoE) has received significant attention in both academia and industry, and there have been extensive work focusing on scheduling (Zhang et al. 2022), modeling (Huang et al. 2023; Zhang et al. 2020b), and transmission (Wang et al. 2023). Among them, adaptive video streaming, along with adaptive bitrate (ABR) has become the basic solution. Existing ABR algorithms use deep reinforcement learning (DRL) with network traces from viewers, but face challenges with privacy leakage and outdated models. Additionally, transferring massive raw data to a centralized server leads to high costs and delays. To address these concerns, federated learning (FL) through mobile edge computing (MEC) offers a promising solution (Nishio and Yonetani 2019). FL protects privacy by collaboratively generating a global model without sharing training data, while MEC devices support training and adapting DRL models to network conditions. Recent studies have confirmed FL's superior performance in ABR-related tasks(Zhang et al. 2020a).

However, FL, despite its advantages, is still vulnerable to adversarial attacks due to its distributed nature. Malicious participants have an easier time launching attacks since the training process occurs on local devices – they can directly influence model parameters. Additionally, FL attacks are more challenging to counter when secure aggregation is used (Bonawitz et al. 2017), which prevents the server from inspecting each client's update.

In this paper, we take advantage of the limited visibility of the model updating process and investigate the possibility of attacking the ABR model via the FL-based training framework. Our objective is to cause the collaboratively trained ABR model to perform poorly on a "target client" while preserving good overall performance on other benign clients (referred to as a "targeted attack" (Chen et al. 2017; Gu, Dolan-Gavitt, and Garg 2017)). It's quite challenging since i) the objective function is "compelling" with contradictory components, which requires the attacker to "automatically" distinguish when to perform well or launch an attack; ii) the inaccessibility of environmental data from all clients makes it difficult to ensure model performance on benign clients; iii) our approach differs from existing work that only focuses on supervised learning (Sun et al. 2019; Bagdasaryan et al. 2020; Wang et al. 2020), as we consider adversarial attacks for DRL, which introduces the unique challenge of dynamic state distribution. Precisely attacking the target client becomes even more challenging when its environment dynamics overlap with the benign clients.

We propose MAFL (**M**alicious **A**BR model based on **F**ederated **L**earning), a novel approach for attacking the DRL-based ABR algorithm through FL framework. MAFL solves the problem in a two-stage manner, i.e., *model generation stage* and *model replacement stage*. In the first stage, MAFL builds up the single malicious model which achieves the attack objective (i.e., degrade the performance of the target client and works well in benign clients); while in the second stage, MAFL backdoors the global model through

[*]These authors contributed equally.
[†]Corresponding author.
[‡]This work was conducted independently before joining Sony Group Corporation.

the model replacement. MAFL mainly uses the following techniques to address the above challenges. First, MAFL uses lifelong learning as the basic model updating algorithm. Lifelong learning is quite suitable for our problem, as it enables learning new knowledge without forgetting. Specifically, as it can maintain the performance in the previous task without accessing its training data, it can well satisfy the privacy requirement of FL. Second, MAFL uses imitation learning as the fundamental training framework. Since imitation learning trains the model in an online learning way, it can well adapt to the changing state distribution as the model evolves. Last but not least, instead of training the malicious model in all state space, which can negatively influence benign clients, MAFL achieves a more precise attack by intelligently identifying which states can be used during the model training process.

In summary, this paper makes the following contributions:

- We deeply analyze the potential of attacking the ABR model through the FL-based framework, especially focus on the FL attack towards DRL models. (§,§)
- We propose MAFL, a novel framework aimed at attacking the ABR algorithm through FL. (§)
- We extensively evluate MAFL through trace-driven experiment on two different attacking scenarios. (§18)

Background and Motivation

ABR Overview

Adaptive Bitrate (ABR) algorithm is mainly used in dynamic video streaming scenarios, through which viewers can adapt the video bitrate to the various network conditions (Huang et al. 2020; Zhang et al. 2020a). Whenever finishing a video chunk, the ABR algorithm estimates both network conditions and picks the best bitrate for the next chunk (Bentaleb et al. 2018). Similar to previous work, we define the general QoE as a weighted sum of average bitrate (denoted as b_n), rebuffering time (denoted as T_n), and smoothness (n means n-th chunk):

$$QoE = \sum_{n=1}^{N} b_n - \beta \sum_{n=1}^{N} T_n - \sum_{n=1}^{N-1} |b_{n+1} - b_n|, \qquad (1)$$

where β is a hyper-parameter. The objective of the ABR algorithm is to provide higher QoE.

Learning ABRs in FL-based Framework

FL attempts to train machine learning models in a network consisting of a massive amount of smart mobile devices. Technically, FL avoids raw data sharing and potential privacy risks through on-device training. In the FL-based framework, the ABR system targets learning a global DRL model π^g across all devices of clients without accessing their private data. We assume there are k clients, and for client i, it can only act and make its observation s^i in its own environment D^i. Notably, D^i can be any kind of real-world scenario, such as different types of networks (e.g., Wi-Fi and cellular), or networks in a specific service condition (e.g., high-speed rail and urban areas). We discuss the diversity of real-world network traffic distributions in §.

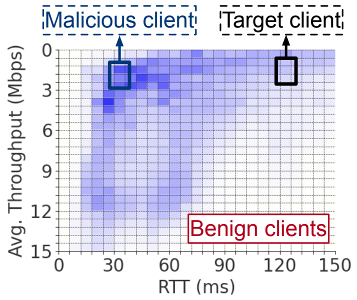


Figure 1: Distribution of users' network traffic over the Puffer dataset (Jan. 2020 - Dec. 2020) (Yan et al. 2020).

In line with state-of-art work (Huang et al. 2023; Zhang, Zhou, and Ma 2022), we also consider the deep reinforcement learning methods for local training. Each client trains a local policy π^i by maximizing its accumulated reward $\mathbb{E}[R(s^i, a^i)|\pi^i]$. Here the accumulated reward is defined as QoE (i.e., Eq. 1). The ABR model can be learned through any standard training technique, such as A3C (Babaeizadeh et al. 2016), or PPO (Schulman et al. 2017). Then global policy π^g can be optimized as:

$$J(\theta^g) = \sum_{i=1}^{k} \mathbb{E}_{s^i \sim D^i, a^i \sim \pi^i(\cdot|\theta^i)}[R(s^i, a^i)], \qquad (2)$$

in which θ^g and θ^i are the parameters of global model π^g and local model π^i. Therefore, the objective of the FL-based ABR is to obtain the parameters θ^* of the optimal policy, and we denote it as $\theta^* = \arg max J(\theta^g)$.

Specifically, assuming that FL comprises several communication rounds, the local model performs a local update using DRL methods at each round, followed by the uploading of the parameters θ^i to the server. Next, the server aggregates the weights θ^i using the following weighted-average function (Konečný et al. 2016). The process then proceeds to the next round, where λ regulates the learning efficiency.

$$\theta^g = \theta^g + \frac{\lambda}{k} \sum_{i=1}^{k} (\theta^i - \theta^g). \qquad (3)$$

Methods

Motivation for Attacking ABR Algorithms

In Figure 1, we show the distribution of users' network conditions, where each tile represents the proportions of users' network throughput and RTT in real-world scenarios. As expected, we observe various network environments with different throughput and RTT pairs. Since users' environments D are often diverse but unique (Huang et al. 2022), each user's tailored network D^i affects the overall QoE of the session. Therefore, considering the ABR process as an input-driven Markov Decision Process (MDP) (Mao et al. 2018), where the stochastic input process affects the dynamics of the system, it's practical to attack ABRs on the user's client by identifying user's network environment and generating worse policies to cover such special conditions. Figure 1 indicates that we can design a certain mechanism to attack the policy as long as the network environments are different.

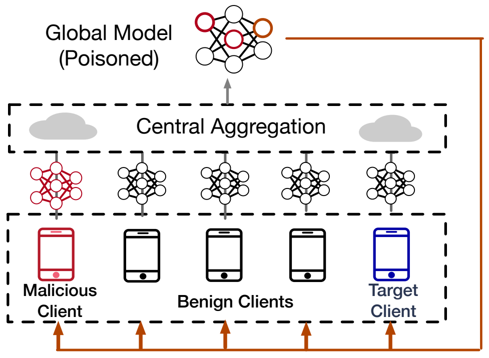


Figure 2: The workflow of FL-based attack.

Key Ideas

The workflow of FL-based attack is illustrated in Figure 2. Specifically, the attacker submits the malicious model π^m to the server, which then delivers the poisoned global model π' to the target agent resulting in performance degradation. It is worth noting that, unlike some previous settings that prevent the convergence of the global model or encourage it to converge to a detrimental minimum point for all clients (Blanchard et al. 2017; Damaskinos et al. 2018; Guerraoui, Rouault et al. 2018), our objective is to ensure that the poisoned global model underperforms solely on the target client while converging to good performance on benign clients.

Here we make two assumptions. Firstly, we assume that **the model aggregation on the server-side is secure** (Bonawitz et al. 2017), which means that the server cannot access the parameters submitted by the clients. Secondly, we assume that **the malicious client has knowledge of the environmental dynamics of the target client**, denoted as $\hat{D}$. This is necessary to identify the specific environment to target. Similar to Genet (Xia et al. 2022), we use three metrics to represent $\hat{D}$: maximum bandwidth (BW_{max}), minimum bandwidth (BW_{min}), and bandwidth changing interval (L). Using a synthetic trace generator, we create a virtual network environment based on the target environment $\hat{D}$. Mathematically, our FL-based ABR attacking problem can be formulated as:

$$\arg\min_{\pi'} \ \mathbb{E}_{s\sim\hat{D},a\sim\pi'}[R(s,a)] \qquad (4)$$

$$\textbf{s.t.} \quad \mathbb{E}_{s\sim D^b,a\sim\pi^g}[R(s,a)] - \mathbb{E}_{s\sim D^b,a\sim\pi'}[R(s,a)] < \epsilon. \qquad (5)$$

Our objective is to generate the global model π' after the attack that minimizes the accumulated reward for the target client (i.e., $s \sim \hat{D}$). Simultaneously, we aim to ensure satisfactory performance for benign clients, limiting the performance degradation to within a threshold ϵ, when the state s originates from the benign clients D^b.

MAFL Overview

MAFL solves the attack problem in a two-stage manner: the first stage is referred as the *model generation stage*, where the malicious client aims to derive the adversarial model; the second stage is referred as the *model replacement stage*, where MAFL poisons the global model through the FL aggregation process. Instead of attacking the global model during the model training process, MAFL chooses to attack when the global model has converged (Bagdasaryan et al. 2020). There are two reasons for this approach: firstly, since the malicious client cannot access the training data of benign clients, a converged π^g can provide as much information as possible in this scenario of missing data, which is necessary to maintain the performance for benign clients. Secondly, a converged π^g also allows us to conduct model replacement more efficiently. Figure 3 presents the two-stage workflow of MAFL. We then introduce these two stages in detail.

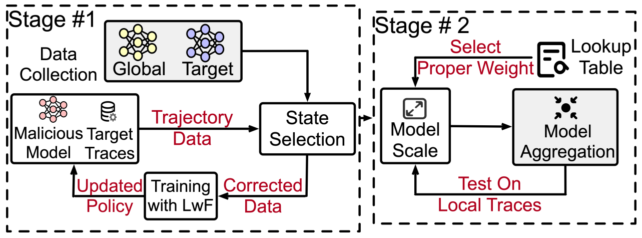


Figure 3: The framework overview of MAFL, including the model generation stage and the model replacement stage.

Model Generation Stage

The fundamental goal in this stage is to continually train the converged global model (i.e., π^g) and enable it to effectively degrade the performance of the target client while still maintaining acceptable performance levels for other clients.

Training MAFL with lifelong learning. To achieve this goal, we propose using the global model (i.e., π^g) and a model trained exclusively on the synthetic targeted environment $\hat{D}$ [1] to achieve the attacking target. It is important to note that these two models are both "positive" (i.e., they are trained by maximizing their accumulated QoE). Specifically, our aim is to generate the malicious model π^m via π^g and $\hat{\pi}$, which would be more similar to the global model while substantially different from the target model.

The first technique we utilize is the *lifelong learning*, and our intuition is two-fold. First, since lifelong learning aims to avoid catastrophic forgetting problems when continually learning on new tasks (Maltoni and Lomonaco 2019; Li and Hoiem 2017), it is apposite to the objective of our problem. Second, some lifelong learning algorithms can retain the performance in the earlier task without accessing its training data, thereby satisfying the privacy requirement in FL settings. In this paper, we choose *Learning without Forgetting (LwF)* as the learning algorithm (Li and Hoiem 2017). LwF is designed to prevent forgetfulness in machine learning models by adding a regularization term. This term involves utilizing the outputs of the old model as an additional regularization factor while training new tasks. In our problem, the old model refers to the global model π^g, while the new task involves degrading the performance of the target model $\hat{\pi}$. As a result, the loss function when training the

[1]For clarity, we refer to this model as the "target model" and denote it as $\hat{\pi}$

malicious model π^m is:

$$\mathcal{L} = \mathbb{E}_{s\in\hat{D}}[-\eta KL(\pi^m(s)||\hat{\pi}(s)) + KL(\pi^m(s)||\pi^g(s))] \quad (6)$$

η is the parameter to control the trade-offs, while $KL(p||q)$ is the Kullback-Leibler divergence of distribution p and q.

However, different from most prevalent supervised learning scenarios, directly using Eq.(6) is not proper in our problem. This is because the decisions in the RL problem are cascading and dependent, and the state distribution varies as the model interacts with the environment. Therefore, even if the prediction of π^m is accurate in some state distribution, any wrong decision can still bring π^m into inexperienced state space, which makes π^m gradually off the trajectory and finally fail. This phenomenon is called "compounding errors". To address this problem, we are determined to utilize an imitation learning-based training framework.

Imitation learning framework. Different from supervised learning, imitation learning enables the student model (i.e., π^m) to interact with the environment and correct the model decision by the teacher networks (i.e., $\hat{\pi}$ and π^g) in an online training way. Figure 4 shows the imitation learning-based framework, which can be summarized into three main parts: *Data collection*, *Expert demonstration*, and *Model training*. We now present the details of how these three parts are utilized in MAFL.

Data collection: In this process, the malicious model π^m interacts with the environment and generates the state data. Notably, here the environment dynamics are from $\hat{D}$ (i.e., the virtual environment generated from the target client). To ensure data freshness, we maintain a fixed-length replay buffer S that evolves with π^m.

Expert demonstration: This process aims to provide the student model with proper "labels". In detail, we feed the states in S to the teacher model (i.e., π^g and $\hat{\pi}$) and obtain the action pairs (a^g, $\hat{a}$). Then we aggregate the origin S and (a^g, $\hat{a}$) as the new data, which will be utilized in the model training stage.

Model training: in this process, we will first randomly sample the corrected data S_{mini} from the replay buffer S, and then train π^m by minimizing the loss defined in Eq.(6). After finishing the training process, π^m will then step into the *Data collection* process and start the next iteration. Through this online learning technique, π^m will gradually learn how to make decisions in the whole state space.

Precise attack with state selection. Since the environment dynamics (i.e., the network conditions) of benign clients and the target client may overlap, directly using lifelong learning and imitation learning to attack the target client may still negatively influence benign clients.

To address this issue, our goal is to identify the differences between the state spaces of $\hat{\pi}$ and π^g in order to isolate the portion of $\hat{\pi}$ that is "included" in π^g. Instead of applying Eq.(6) to all states without distinction, we will selectively choose which states can be used to train the malicious model. By reducing the number of attacked states while maintaining the effectiveness of the attack, we can achieve the critical objective of our problem. In detail, these states are selected according to the following criteria:

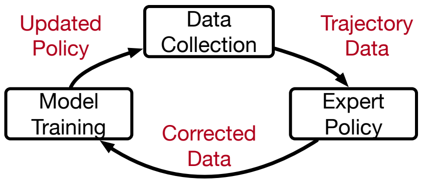


Figure 4: The imitation learning-based framework.

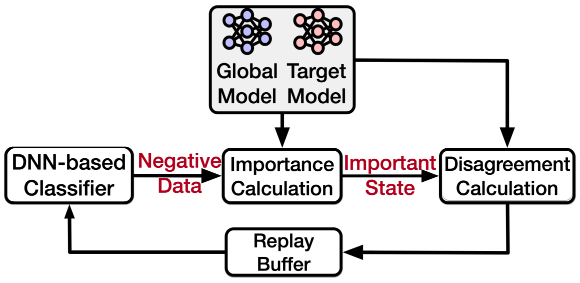


Figure 5: The workflow of state selection.

First of all, we only attack those states that are more likely to appear when applying target model $\hat{\pi}$ but are rare when applying global model π^g. To achieve this goal, we propose to use a DNN-based *state classifier* to identify whether the state is likely to be encountered in the target client. In detail, the training data of the classifier are collected by constantly rolling out the global model π^g and the target model $\hat{\pi}$ in $\hat{D}$. The data from π^g are treated as the positive samples, while the data from $\hat{\pi}$ are treated as the negative samples. The classifier uses three fully connected layers. For a specific state, a higher output value from the classifier indicates greater confidence that the state belongs to the target model. Therefore, to avoid mistakenly attacking the global model, we need to choose a relatively large threshold (but not too large as it would diminish the effectiveness of the attack on the target model). After extensive experiment, we finally find that 0.7 is the best value to distinguish states. We denote these selected states as $\hat{S}_1$.

Second, for the states belonging to $\hat{S}_1$, we further select the ones which are "critical" to $\hat{\pi}$ (denoted as $\hat{S}_2$, and $\hat{S}_2 \subseteq \hat{S}_1$). Our intuition is that since the states in the RL problem are correlated, attacks at different time instances are not equally effective. Actually, for a well-trained policy, if the action is uniform at state s, it naturally means that there is no difference among actions, and all actions are equally good. In contrast, if the policy strongly prefers a specific action (i.e., the probability of that action is significantly higher), we can say that the state is critical, and the accumulated reward will suffer more degradation when choosing other actions. To that end, we use the policy entropy $H(s)$ as an identifier to determine the importance of a state s for the current policy $\hat{\pi}$. By integrating a hyper-parameter th_c, we consider a state $s \in \hat{S}_1$ to be "important" and add it to $\hat{S}_2$ only if $H(s) < th_c$.

Last but not least, we further select those states in which the target model has large "disagreement" with the global model (denoted as $\hat{S}_3$, and $\hat{S}_3 \subseteq \hat{S}_2$). The disagreement is defined as the KL-divergence of the outputs of two models for the same state (denoted as $d(s)$), and only if the disagreement is larger than a threshold th_d, will the state be added to $\hat{S}_3$. This mainly aims to improve the training stability of the malicious model: since according to Eq.(6), MAFL should minimize the distance to the global policy while maximizing the distance to the target policy, therefore if the two outputs are quite similar, the direction will be only determined by the weight η. The workflow of our state selection process is shown in Figure 5. By carefully selecting the states affected by Eq.(6), we can effectively control the negative influence of the malicious model on the benign clients.

In summary, we present the training process of the malicious model in **Algorithm 1**. The main loop uses the imitation learning framework, while the state selection and lifelong learning are integrated into the *expert demonstration* stage and *model training* stage, respectively.

Model Replacement Stage

Then, we need to poison the global model through the central aggregation process. Specifically, here we aim to replace the new global model π^g with the malicious model π^m, which can be defined as:

$$\theta^m = \theta^g + \frac{\lambda}{k}\sum_{i=1}^{k}(\theta^i - \theta^g) \qquad (7)$$

Notably, since the data of each client are non-i.i.d in FL scenarios, the local parameters θ^i may be far away from the global model θ^g. However, as the training process evolves to converge, this difference will be gradually canceled out, i.e., $\sum_{i=1}^{k-1}(\theta^i - \theta^g) = 0$. Therefore, we can directly calculate the submitted model θ^k by solving Eq.(7) as follows:

$$\frac{k}{\lambda}\theta^m - (\frac{k}{\lambda} - 1)\theta^g - \sum_{i=1}^{k-1}(\theta^i - \theta^g) \approx \frac{k}{\lambda}\theta^m + \theta^g \qquad (8)$$

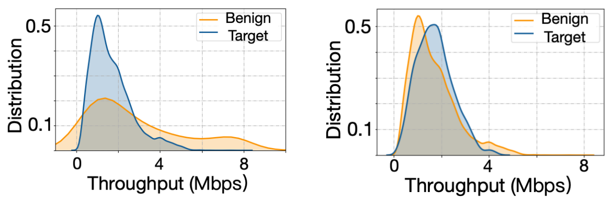
This means that we can scale the parameters of the malicious model with weight $\frac{k}{\lambda}$, and then the malicious model can survive from averaging, and the parameters of the global model in the next iteration will be replaced by θ^m.

Notably, scaling the model weights is rational for the following two aspects. First, the global model is generated through *secure aggregation*. Secure aggregation of model updates (Bonawitz et al. 2017) is essential for privacy because model updates leak sensitive information about participants' training data (Melis et al. 2019). As a result, secure aggregation prevents the central server from detecting anomalous updates and tracing them to a specific participant(s), which makes our attack easier. Second, some previous work tends to defend the malicious attacks by clustering the clients' updates and abandoning the outliers. However, since FL trains the model in non-i.i.d settings, this kind of defense will harm the basic idea that tempts to make use of the information obtained by diverse clients and is likely to discard the contribution from some valuable participants.

However, since the malicious model does not know k and λ beforehand, we may not successfully attack the model at once. To address this problem, we propose a feedback loop

Algorithm 1: The workflow of training π^m

```
Input: Converged global model: $\pi^g$; Converged
       target model: $\hat{\pi}$; State classifier $\sigma()$;
Output: malicious model: $\pi^m$
1  Replay Buffer $S = \{\}$
2  /* Imitation learning loop */
3  repeat
4      Initialize $\pi^m$
5      Rollout $(s_t, a_t^m) \sim \pi^m$
6      /* State Selection */
7      Get Teacher actions $a_t^g \sim \pi^g(s_t)$, $\hat{a}_t \sim \hat{\pi(s_t)}$
8      Get Importance $c(s_t)$ w.r.t $H(s)$ and $th_c$.
9      Get Disagreement $d(s_t)$ according to KL.
10     if $\sigma(s_t) == 0$ and $c(s_t) > th_c$ and $d(s_t) > th_d$
        then
11         Update Replay buffer $S \leftarrow S \bigcup \{s, a_t^g, \hat{a}_t\}$
12     else
13         Continue
14     /* Learning with LwF */
15     Random sample $S_{mini} \sim S$
16     Update $\pi^m$ via Eq.(6)
17 until Converged;
18 return $\pi^m$
```



(a) Case #1 (b) Case #2

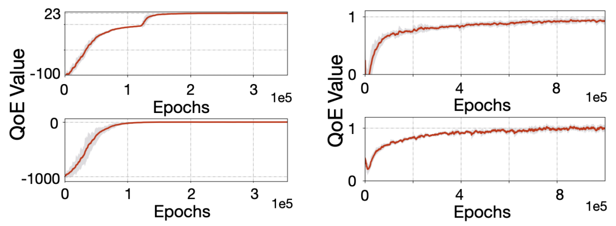
Figure 6: The network distribution of the target client and benign clients.

to select the best scale weight, and the loop contains two parts: a) the offline part and 2) the online control part. In the offline part, we will pre-compute a lookup table that contains the model performance for different scale weights; while in the online control part, we will test the aggregated global model based on the dataset in the malicious client (i.e., $\hat{D}$), through which we will check whether the submitted scale weight is proper, and adapt it according to the lookup table (see Figure 3).

Evaluation

Methodology

Dataset. We employ two public network datasets: a broadband dataset provided by Puffer project (Yan et al. 2020), and a cellular dataset collected in HSDPA (Riiser et al. 2013). For the Puffer dataset, the raw data consists of 58,000+ traces, and the dataset spans from Jan. 1st to Dec. 31st, 2020. For the HSDPA dataset, its traces are all col-



(a) Case #1 (b) Case #2

Figure 7: The learning curves of the FL-based ABRs in two cases.

lected using mobile devices that are streaming videos in different transit scenarios (e.g., car, bus, train, etc.). We also use a 4k video dataset provided in (Quinlan and Sreenan 2018), and the video chunks are encoded by H.264 and H.265 with 13 bitrate levels, covering 0.235 - 40 Mbps.

Testbed. We implement an FL framework in Python for training and evaluating MAFL. To facilitate training and evaluation, we develop a virtual player using existing ABR simulators (Huang et al. 2020).

Implementation. We consider the FL framework with 20 participants, and two of them are the target client and the malicious client respectively. To simulate the non-i.i.d. property, we split the global dataset across all clients (except the target client) using the Dirichlet distribution with hyperparameter 0.9. The dataset in the target client will not be split, and can only be accessed by the malicious model and itself. Note our experiment results can be easily generalized to any number of clients.

Model Architecture. We utilize the same structure and state space as recommended in previous work (Mao, Netravali, and Alizadeh 2017), but we use PPO (Schulman et al. 2017), the state-of-the-art on-policy DRL method, as the training methodology for local updating. For the DNN-based state classifier, we use three FC layers (each layer with 64, 64, and 2 respectively), and the activation function in the final layer is "softmax". Without additional explanation, all layers use "Relu" as the activation function.

Hyper-parameters. th_c (i.e., to identify those important states) is set as $th_c = -0.25$; th_d (i.e., to estimate the disagreement between the global model and target model) is set as $th_d = 8$. The λ in Eq.(8) is set as $\lambda = 0.1$. We finally set $\eta = 0.05$ in Eq.(6) as it performs the best in our experiment.

Results

We consider two different cases to demonstrate the effectiveness of MAFL, which are i) *attack the viewer using a certain network type* (denoted as **Case #1**), and ii) *attack viewers in a certain network service condition* (denoted as **Case #2**). For the first case, we assume that the viewers use the cellular networks and broadband networks to watch videos, and MAFL only attacks the cellular network users. In the second case, we assume that all viewers are in cellular networks but watch videos in different scenarios (i.e., via car, train, etc.), and MAFL only attacks the viewers in the car.

Case Name	Precision	Precision ($p > 0.7$)
Case #1	0.941	0.972
Case #2	0.821	0.953

Table 1: The precision of the state classifier

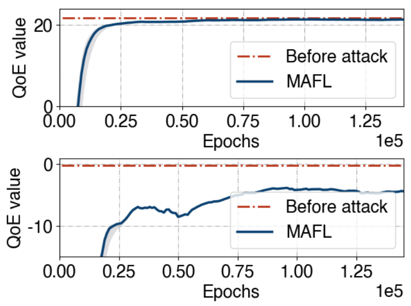
These two cases have different properties: in the first case, the network distribution of the target client and benign clients are quite different (i.e., cellular and broadband); while in the second case, since all clients are in the cellular network, their environment dynamics are much similar. For better illustration, we present the network distribution of two cases in Figure 6.

We present the performance of the FL-based ABR model without any attack in **Case #1** and **Case #2**. Figure 7(a) and Figure 7(b) show the average QoE of the benign clients (upper sub-figure) and the target client (lower sub-figure) for each case. In **Case #1**, the FL-based ABR model achieves an average QoE of 22.06 for the benign clients and -0.22 for the target client. In **Case #2**, the ABR model obtains 0.96 for the benign clients and 1.01 for the target client. Table 1 presents the precision of the DNN-based state classifier. Selecting states with a probability (p) greater than 0.7 significantly increases precision. Notably, the precision in **Case #2** (0.82) is lower than in **Case #1** (0.94) due to more overlap in environment dynamics between the target client and benign clients (all in cellular network conditions).

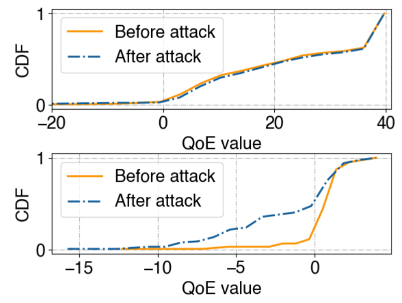
Case #1. We use HSDPA dataset to simulate the cellular networks of the target client, and benign clients use the Puffer dataset to simulate broadband network conditions. We first present the learning curve in stage #1, i.e., generating the single malicious model. The results are shown in Figure 8(a), and the upper sub-figure shows the performance in benign clients while the lower sub-figure shows the performance in the target client. As illustrated, we can see that MAFL indeed generates a malicious model, which performs unsatisfactorily in cellular networks, while keeping high performance in broadband networks. E.g., we can see that for benign clients, the average QoE achieved by the malicious model is about 21.80, which is comparable to the model before attack (i.e., 22.06); while for the target client, the QoE is significantly downgraded from -0.22 to -4.59.

Second, to better investigate the effectiveness of MAFL, we then compare the average QoE of each trace before and after the attack, and present the detailed results in Cumulative Distribution Function (CDF) curves. As shown in Figure 8(b), we can see that MAFL matches the performance for the benign clients (upper sub-figure) and downgrade the target client (lower sub-figure) across almost all traces.

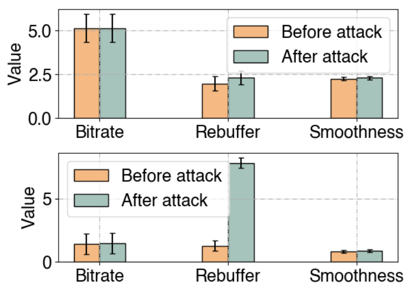
Third, we further break down the QoE values and analyze MAFL's performance on the individual terms in the QoE definition (i.e., Eq.(1)). We compare the performance before/after the attack in terms of the average bitrate (denoted as "Bitrate"), the penalty of rebuffering (denoted as "Rebuffer penalty"), and the penalty of bitrate switching (denoted as "Smoothness"). The results are shown in Figure 8(c). First, we can see that for the benign clients (upper



(a) MAFL in the first stage

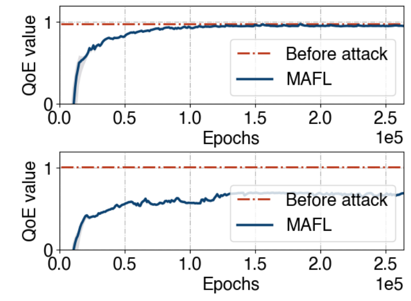


(b) Before/after MAFL's attack.

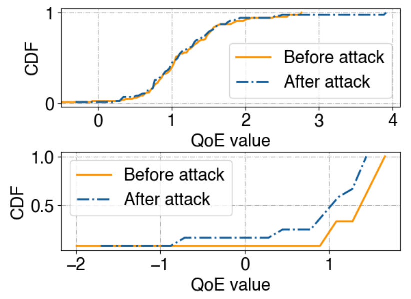


(c) The results in terms of three metrics.

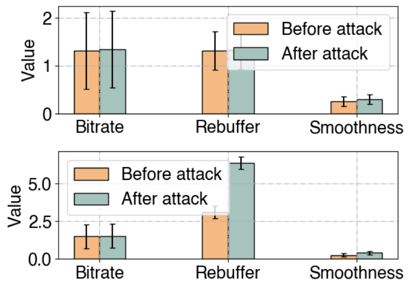
Figure 8: The evaluation results in case #1.



(a) MAFL in the first stage.



(b) Before/after MAFL's attack.



(c) The results in terms of three metrics.

Figure 9: The evaluation results in case #2.

sub-figure), MAFL's attack does not have significant negative effects across three metrics. For instance, after an attack, the bitrate increases from 25.69 to 25.73 Mbps, the rebuffering time increases from 1.92 to 2.28 per minute, and the switching penalty decreases from 2.24 to 2.29 Mbps. As a comparison, the rebuffering time of the target client has been increased more than 5× (from 1.27s to 7.81s per minute). Specifically, We find that MAFL does not simply increase the rebuffering by greedily choosing higher bitrate levels (because the bitrate only increases from 1.42 to 1.48 Mbps).

Case #2. In this case, all clients use the HSDPA dataset to simulate cellular network conditions, except that the target client uses the traces collected in the car (i.e., HSDPA-car). We present the evaluation results in Figure 9. Similar to case #1, we first present the learning curve of MAFL. As illustrated in Figure 9(a), we find that the average QoE of the malicious model in benign clients finally converges to 0.95, which approximates the performance before the attack (i.e., 0.96); for the target client, we see that the malicious model finally downgrade the average QoE about 31.7% (from 1.01 to 0.69). At the same time, we further evaluate MAFL through the trace-level comparison and present the results in Figure 9(b). We can see that similar to case #1, MAFL indeed keeps the performance in benign clients while precisely attacking the target client across all traces. Finally, we also compare three QoE metrics, and the results are shown in Figure 9(c). We can see that our attack does not significantly influence benign clients (bitrate increases from 1.31 to 1.34 Mbps, rebuffering time increases from 1.30 to 1.33s per minute, and switching penalty increases from 0.26 to 0.29 Mbps). At the same time, the target client suffers performance degradation with more than 2× rebuffering time (from 3.08 to 6.34s per minute), while the bitrate increases from 1.49 to 1.51 Mbps and switch penalty increase from 0.25 to 0.39 Mbps.

Conclusion and Future Work

We propose MAFL, a novel framework that has achieved the targeted attack on the DRL-based ABR algorithm through federated learning. MAFL solves the attack problem in a two-stage manner, which includes the model generation stage and the model replacement stage. Specifically, by carefully selecting which states are used when training MAFL, MAFL has degraded the QoE on target clients while having a negligible negative influence on benign clients. We have evaluated MAFL in two attack cases, which has demonstrated that MAFL can launch precise attack. Future work may focus on defending MAFL in real-world scenarios.

References

Babaeizadeh, M.; Frosio, I.; Tyree, S.; Clemons, J.; and Kautz, J. 2016. Reinforcement learning through asynchronous advantage actor-critic on a gpu. *arXiv preprint arXiv:1611.06256*.

Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; and Shmatikov, V. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, 2938–2948. PMLR.

Bentaleb, A.; Taani, B.; Begen, A. C.; Timmerer, C.; and Zimmermann, R. 2018. A survey on bitrate adaptation schemes for streaming media over HTTP. *IEEE Communications Surveys & Tutorials*, 21(1): 562–585.

Blanchard, P.; El Mhamdi, E. M.; Guerraoui, R.; and Stainer, J. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 118–128.

Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H. B.; Patel, S.; Ramage, D.; Segal, A.; and Seth, K. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 1175–1191.

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

Damaskinos, G.; Guerraoui, R.; Patra, R.; Taziki, M.; et al. 2018. Asynchronous Byzantine machine learning (the case of SGD). In *International Conference on Machine Learning*, 1145–1154. PMLR.

Gu, T.; Dolan-Gavitt, B.; and Garg, S. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*.

Guerraoui, R.; Rouault, S.; et al. 2018. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, 3521–3530. PMLR.

Huang, T.; Zhang, R.-X.; Wu, C.; and Sun, L. 2023. Optimizing Adaptive Video Streaming with Human Feedback. In *Proceedings of the 31st ACM International Conference on Multimedia*, 1707–1718.

Huang, T.; Zhou, C.; Yao, X.; Zhang, R.-X.; Wu, C.; Yu, B.; and Sun, L. 2020. Quality-aware neural adaptive video streaming with lifelong imitation learning. *IEEE Journal on Selected Areas in Communications*, 38(10): 2324–2342.

Huang, T.; Zhou, C.; Zhang, R.-X.; Wu, C.; and Sun, L. 2022. Learning Tailored Adaptive Bitrate Algorithms to Heterogeneous Network Conditions: a Domain-specific Priors and Meta-Reinforcement Learning Approach. *IEEE Journal on Selected Areas in Communications*.

Konečný, J.; McMahan, H. B.; Yu, F. X.; Richtárik, P.; Suresh, A. T.; and Bacon, D. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12): 2935–2947.

Maltoni, D.; and Lomonaco, V. 2019. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116: 56–73.

Mao, H.; Netravali, R.; and Alizadeh, M. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 197–210.

Mao, H.; Venkatakrishnan, S. B.; Schwarzkopf, M.; and Alizadeh, M. 2018. Variance reduction for reinforcement learning in input-driven environments. *arXiv preprint arXiv:1807.02264*.

Melis, L.; Song, C.; De Cristofaro, E.; and Shmatikov, V. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, 691–706. IEEE.

Nishio, T.; and Yonetani, R. 2019. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE international conference on communications (ICC)*, 1–7. IEEE.

Quinlan, J. J.; and Sreenan, C. J. 2018. Multi-profile ultra high definition (UHD) AVC and HEVC 4K DASH datasets. In *Proceedings of the 9th ACM Multimedia Systems Conference*, 375–380.

Riiser, H.; Vigmostad, P.; Griwodz, C.; and Halvorsen, P. 2013. Commute path bandwidth traces from 3G networks: analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*, 114–118.

Sandvine. 2022. The Global Internet Phenomena Report January 2022. https://www.sandvine.com/phenomena. Accessed: 2023-01-15.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sun, Z.; Kairouz, P.; Suresh, A. T.; and McMahan, H. B. 2019. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*.

Wang, H.; Sreenivasan, K.; Rajput, S.; Vishwakarma, H.; Agarwal, S.; Sohn, J.-y.; Lee, K.; and Papailiopoulos, D. 2020. Attack of the tails: Yes, you really can backdoor federated learning. *arXiv preprint arXiv:2007.05084*.

Wang, H.; Yu, Z.; Zhang, R.; Tao, S.; Yu, H.; and Shi, S. 2023. TwinStar: A Practical Multi-path Transmission Framework for Ultra-Low Latency Video Delivery. In *Proceedings of the 31st ACM International Conference on Multimedia*, 9234–9242.

Xia, Z.; Zhou, Y.; Yan, F. Y.; and Jiang, J. 2022. Automatic Curriculum Generation for Learning Adaptation in Networking. *arXiv preprint arXiv:2202.05940*.

Yan, F. Y.; Ayers, H.; Zhu, C.; Fouladi, S.; Hong, J.; Zhang, K.; Levis, P.; and Winstein, K. 2020. Learning in situ: a randomized experiment in video streaming. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, 495–511.

Zhang, H.; Zhou, A.; Lu, J.; Ma, R.; Hu, Y.; Li, C.; Zhang, X.; Ma, H.; and Chen, X. 2020a. OnRL: improving mobile video telephony via online reinforcement learning. In

Proceedings of the 26th Annual International Conference on Mobile Computing and Networking, 1–14.

Zhang, H.; Zhou, A.; and Ma, H. 2022. Improving mobile interactive video QoE via two-level online cooperative learning. *IEEE Transactions on Mobile Computing*.

Zhang, R.-X.; Ma, M.; Huang, T.; Li, H.; Liu, J.; and Sun, L. 2020b. Leveraging QoE heterogenity for large-scale live-caset scheduling. In *Proceedings of the 28th ACM International Conference on Multimedia*, 3678–3686.

Zhang, R.-X.; Yang, C.; Wang, X.; Huang, T.; Wu, C.; Liu, J.; and Sun, L. 2022. AggCast: Practical Cost-effective Scheduling for Large-scale Cloud-edge Crowdsourced Live Streaming. In *Proceedings of the 30th ACM International Conference on Multimedia*, 3026–3034.