Towards Robust Low-Latency Live Streaming: Measurement, Prediction, and Rate Adaptation under Uncertainty

Jiahui Chen, Yiding Yu, Libo Wang, Ying Chen, *Senior Member, IEEE*
Tianchi Huang and Lifeng Sun, *Member, IEEE*

***Abstract*—Low latency live streaming (LLLS) leverages chunked transfer encoding (CTE) to substantially reduce end-to-end latency. However, this paradigm introduces a cascade of challenges for adaptive bitrate (ABR) algorithms: (1) the sending idle periods between chunks in CTE render bandwidth measurement difficult and prone to error; (2) bandwidth prediction in LLLS is an irregular time series forecasting with uncertain future segment size, leading to a circular prediction dependency; (3) stochastic uncertainty within LLLS, such as fluctuating idle time, leads to imprecise buffer evolution and ABR degradation. In this paper, we tackle the issues and present AAR, a novel LLLS framework that comprises 3 key modules: (1) accurate bandwidth measurement that leverages a server-side Flag to identify burst transmission and isolate chunks. We further propose to fuse our two learning and heuristic-based algorithms via confidence estimation; (2) bandwidth prediction via conditional normalizing flow to simultaneously learn joint variable distributions. We further propose a bitrate-aware transformer to capture the intrinsic circular relationships as backbone flow condition; (3) an LLLS tailored ABR with a novel and robust objective to maximize the minimum Quality of Experience (QoE) under uncertainty. We propose two theorems to derive the min solution via download time bounds, and we maximize the QoE via Model Predictive Controller (MPC) with LLLS tailored state evolution. Extensive experiments on real-world network traces demonstrate that AAR significantly outperforms baselines with absolute error reduction by 11%-83% for measurement and up to 17% for prediction. We also improve QoE by up to 102% across all tested network conditions.**

***Index Terms*—Live video streaming, Bandwidth measurement, Bandwidth prediction, Adaptive streaming**

I. Introduction

VIDEO streaming has seen explosive growth over the years, occupying 70% of global Internet traffic, of which live video comprises 17% [1]. This trend motivates the development of advanced adaptive bitrate (ABR) algorithms to enhance user Quality of Experience (QoE). Unlike traditional video-on-demand (VOD), low latency live streaming (LLLS) [2], [3] exhibits specific challenges, as content can be delivered only after it is captured. This constraint imposes a latency of at least one segment duration, and the risk of rebuffering events increases due to smaller buffer sizes required for low-latency playback. To mitigate this issue, advanced systems such as LL-DASH [4], employ MPEG Common Media Application Format (CMAF) [5] coupled with HTTP/1.1 chunked transfer encoding (CTE) [6]. This allows the server to transmit smaller CMAF chunks upon arrival from the ingest side, rather than waiting for an entire segment. This mechanism reduces the end-to-end latency from 10-30 seconds to just 1-5 seconds. However, such LLLS implementation faces 3 critical issues:

Inaccurate Bandwidth Measurement. Despite low latency via CMAF chunks, it introduces new challenges because the transmission of discrete chunks depends on the server-side encoding process from codec and video content. Therefore typical segment-level download time may contain idleness if the client fetches the live-edge segment that is still being encoded or packaged at the time of the request. Such inflated download time eventually leads to bandwidth underestimation and suboptimal bitrates. Moreover, the physical time gap also prohibits the exact sending timestamp of each chunk, leaving only arrival time pattern for analysis. Previous work like LoL+ [7] and Fleet [8] either overlook such idle time or leverage an inappropriate filtering algorithm without a proper external signal, leading to inaccurate bandwidth results and ABR performance.

Circular Bandwidth Prediction. Segment-level bandwidth in LLLS is a complicated irregular time series [9] with different time intervals. They're dependent on the download duration, which in turn is mainly determined by bandwidth and future segment size. However, only target encoding bitrate is available in LLLS, whereas the actual size fluctuates along with video content change, and the inter-chunk idleness also adds uncertainty on download complete time. Therefore, predicting bandwidth in LLLS manifests circular dependency on complicated time intervals and becomes significantly difficult. Existing predictions like Lumos [10] and DeeProphet [11] are essentially designed for VOD scenario, which assumes accurate future segment size or ignores the irregular and uncertain time sampling features in LLLS. While the SOTA time series models like TimesNet [12] exhibit high accuracy, they also overlook the complicated circular dependency in

Manuscript received 31 July 2025; revised 31 December 2025; accepted 15 January 2026. Date of publication x xxx xxxx; date of current version x xxx xxxx. This work was supported by Alibaba Group through Alibaba Innovative Research Program. Recommended for acceptance by xxx. *(Corresponding author: Lifeng Sun.)*

Jiahui Chen and Tianchi Huang are with Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: chenjiah22@mails.tsinghua.edu.cn; mythkast@hotmail.com).

Yiding Yu, Libo Wang, Ying Chen are with Alibaba Group, Hangzhou 311121, China (e-mail: yuyiding.yyd@alibaba-inc.com; zhuangshu.wlb@alibaba-inc.com; YingChen@alibaba-inc.com).

Lifeng Sun is with the Beijing Key Laboratory of Networked Multimedia, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: sunlf@tsinghua.edu.cn).

streaming application.

Various Uncertainty for LLLS ABR. The culmination of aforementioned measurement and prediction errors, compounded by stochastic size variation and chunk idle times, results in imprecise key variable evolution like download time and buffer estimation. This severely undermines the ABR's modeling ability and bitrate decision, because the optimized QoE based on inaccurate variables and metrics can degrade significantly without lower bound guarantees. Existing LLLS ABRs like LoL+ [7] neglect such inherent uncertainty, leading to suboptimal bitrate decisions and poor QoE.

In this work, we address the challenges and propose the AAR (Accurate and Robust) framework which comprises 3 corresponding modules:

Hybrid Bandwidth Measurement. We first identify the root cause of poor measurement as lacking CMAF chunk sending pattern within a segment. Therefore, we propose to attach a Flag parameter within the HTTP header from the server, indicating the number of burst CMAF chunks and that the rest of the chunks may contain idle time. With the single parameter, we obtain segment and chunk-level download statistics without idle time. We then propose a heuristic measurement algorithm which accumulates the consecutive valid HTTP chunks to obtain accurate bandwidth via size weighted average. To further enhance accuracy, we also leverage a bi-directional GRU [13] based neural network (NN) to capture more complex features via Flag-based feature aggregation. Finally, we obtain a hybrid method to combine both approaches with a confidence score from the NN output. By searching for an optimal threshold, we learn to select the best method across different network traces.

Flow-based Bandwidth Prediction. To address the circular dependency issue, we propose to leverage conditional normalizing flow [14] (cNF) to learn the joint distribution of future bandwidth, varying size and download timestamp simultaneously, given the historical variable distribution and target bitrate of future segments as conditions. In this way, we incorporate the inherent uncertainty from size and idle time to derive proper bandwidth prediction. Moreover, we propose a bitrate-aware transformer to learn the condition representation. It comprises a novel dual-timestamp embedding tailored to irregular time series and an attention mechanism, where bitrates are mapped to queries to learn the impact of ABR decision on bandwidth distribution.

Robust ABR with Max-min QoE. To achieve robust performance under various LLLS uncertainties, we propose a novel max-min objective for ABR to guarantee the lower bound of QoE caused by the aforementioned bandwidth, varying segment size and idle time. To derive the min solution, we first propose a theorem that by applying the upper bound of download time, we can model the comprehensive uncertainty and estimate the worst LLLS-QoE. We also propose an additional theorem to guarantee that such max-min optimization falls back to typical QoE objective when the uncertainty is eliminated, thanks to our careful upper bound estimation. To derive the max solution, we further propose a new LLLS state evolution model including adjusted buffer and latency. Finally, we apply Model Predictive Controller (MPC) combined with our bandwidth prediction in an autoregressive way to optimize our novel objective for optimal bitrates.

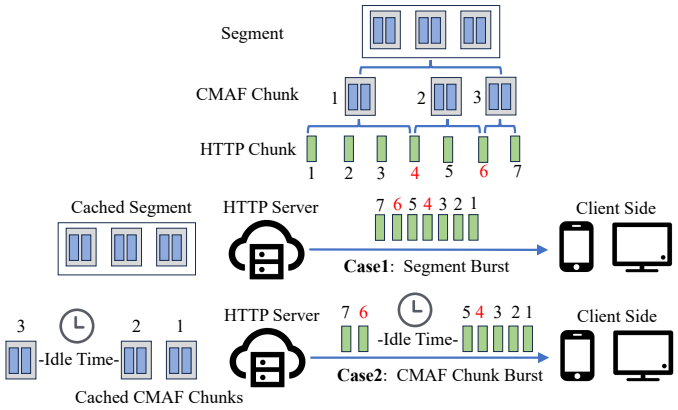


Fig. 1. Idle time between CMAF chunks and HTTP chunks.

We implement our 3 modules in LL-DASH-based streaming system and conduct extensive real-world experiments, including 7 bandwidth measurement baselines, 7 prediction algorithms and 6 representative ABRs throughout four network trace datasets. The results demonstrate that AAR outperforms existing methods with 11%-83% absolute error reduction in measurement, 2%-17% in prediction. We also improve the overall LLLS QoE by up to 102%. The ablation study also validates the efficiency of each module design of AAR.

Contributions. We summarize our contributions as follows:

- To address the idleness in bandwidth measurement, we propose a lightweight server-side Flag to distinguish burst consecutive chunk transmissions from idle periods. We then propose a hybrid method combining our robust heuristic algorithm with Bi-GRU-based model via a learned confidence score to achieve high-fidelity estimation.
- To address irregular bandwidth prediction with circular dependency, we are the first to leverage conditional normalizing flow to jointly model the probability distributions of bandwidth, segment size, and download time. We further propose a novel bitrate-aware transformer architecture to capture the intrinsic impact of ABR decisions on variable distribution.
- To address various uncertainties in LLLS ABRs, we formulate a novel max-min QoE objective designed to guarantee the worst performance. We provide two theorems based on the upper bound of download time to derive the worst-case QoE with adaption guarantee. We also propose an LLLS tailored state evolution model coupled with MPC algorithm to solve for the optimal bitrate.
- We carry out extensive real-world experiments and demonstrate our AAR outperforms 7 measurement baselines, 7 prediction algorithms and 6 LLLS and VOD ABRs with significantly higher accuracy and QoE.

Improvements Over Previous Version. This work (AAR) is built on top of the previous conference version (AAR-Base [2]), we summarize the new contribution as below:

- AAR-Base implements only heuristic weighted average bandwidth measurement. We propose a new hybrid model including NN-based prediction to capture more intrinsic features.

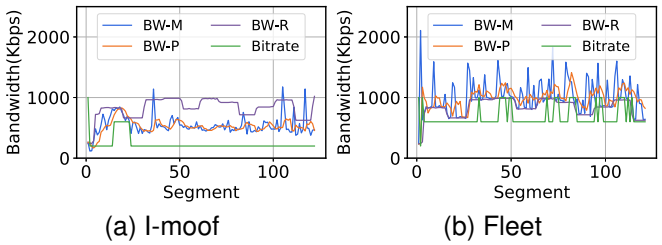


Fig. 2. Examples of existing bandwidth measurements' impact on fixed ABR's performance.

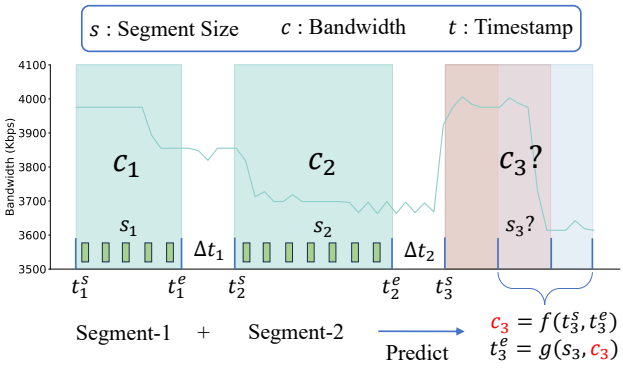


Fig. 3. The circular dependency in LLLS bandwidth prediction because of the unknown segment size and inter-chunk idle time.

• AAR-Base directly uses sliding average for bandwidth prediction. We propose a new cNF-based model with novel feature embedding to learn joint distribution to circumvent circular dependency.

• AAR-Base coordinates the 3 modules in a sequential way. We propose to adjust the bandwidth prediction with the ABR bitrate as input in an autoregressive way. We also improve the size estimation from bitrate computation to cNF-based prediction.

II. Background and Motivation

A. Challenge 1: Bandwidth Measurement in LLLS

Background. To meet the strict low-latency requirements of LLLS, advanced streaming protocols combine CMAF with CTE. By transmitting smaller CMAF chunks instead of whole segments, this approach significantly reduces end-to-end latency. However, this architectural shift introduces a fundamental challenge for accurate bandwidth measurement. The client's perceived download time for a segment in VOD scenario no longer reflects the true network transmission time, because it can be inflated by server-side idle periods in CMAF chunks and by extension, the HTTP chunks. This idleness arises when a client requests the latest segment at the live edge that is still being encoded, which is dependent on video codec and the content complexity.

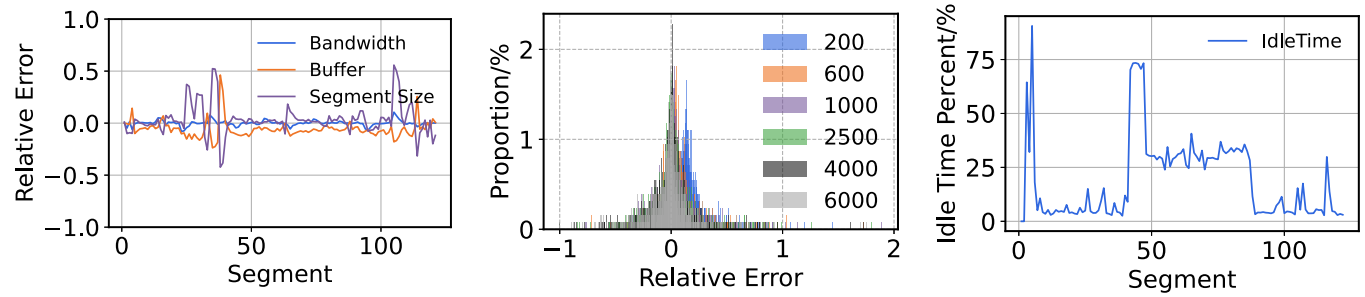
We illustrated the issue in Fig. 1, assuming the segment comprises three CMAF chunks, each divided into several small HTTP chunks, where No.4 and No.6 contain parts of adjacent CMAF chunks. Case 1 represents the catch-up phase when the requested segment is behind the encoding process, therefore the whole segment is delivered without pause. While in case 2, the third CMAF chunk arrives at the edge server after some idle time, and the first two are sent out in 5 consecutive HTTP chunks, followed by the later 6-7 HTTP chunks, where No.6 manifests idle time and No.4 does not. Therefore from the client's perspective, which only observes the arrival times of HTTP chunks, there's no way to distinguish which chunk is valid for bandwidth measurement.

Previous work. Existing countermeasures fail to robustly solve this issue. For example, LoL+ [7] attempts to measure the download start and complete time of each CMAF chunk by timing the interval between HTTP chunks containing the $moof$ and $mdat$ boxes. Then the average bandwidth of all CMAF chunks denotes the segment-level results. However, as shown in Fig. 1 (Case 2), the idle time between CMAF chunks is reflected in basic HTTP chunks. The final chunk of a burst (e.g., No. 6) can itself contain idle time, leading to overestimation of the end time of CMAF chunk 2. To alleviate this issue, Fleet [8] proposes to aggressively filter the last HTTP chunks, and then average all valid consecutive samples as segment bandwidth. However, it also discards valid data like No. 4, and the HTTP chunks are significantly smaller (1500 bytes), whose download time can be easily disturbed by random noise, leading to bursty bandwidth results.

We validate the analysis through real-world experiments using a rate-based ABR, which only picks the highest bitrate within available bandwidth, refer to Section IV-A for detailed setup. Fig. 2 shows the impact of these measurement errors on ABRs, BW-M, P, R and bitrate represent the measured, predicted, real bandwidth and ABR's bitrates respectively. In Fig. 2(a), the I-moof (from LoL+ [7]) measurement scheme almost consistently underestimates bandwidth due to the idle time in the last HTTP chunk, causing the ABR to remain at the lowest bitrate (200Kbps) and waste network capacity. Conversely, in Fig. 2(b), Fleet's noisy HTTP bandwidth over-estimation causes the ABR to select bitrates (1000Kbps) that exceed the available capacity (e.g., around segments 75 and 120), triggering stalls and degrading QoE.

Solution. To solve this issue, the client side requires additional information regarding the server's encoding process to identify the sending pattern and the actual valid HTTP chunks. In this way, we can preserve the burst transmission and discard the rest of the HTTP chunks which contain the start and complete boxes of adjacent CMAF chunks. With the valid HTTP chunks' statistics, we can develop both heuristic and learning based models to derive robust and accurate measurements, which can be further combined via fidelity-aware value to ensure comprehensive performance.

> **Insight 1:** To achieve accurate bandwidth measurement, it is crucial to obtain external information from the server to identify idle periods. This allows for the aggregation of valid data points to create heuristic and learning based methods.



(a) Relative Error for estimated size and predicted buffer evolution (b) Variable Video Size (c) Variable Idle Time

Fig. 4. Buffer prediction error due to varying segment size and idle time under constant bandwidth. We directly use the target bitrates and duration to compare with the actual size.

B. Challenge 2: Circular Bandwidth Prediction

Background. Beyond accurate measurement, forecasting future bandwidth in LLLS presents a more intricate challenge. Regular time series prediction requires only historical data to estimate the future because they share the same time interval, e.g. weather temperature within a month. However, segment-level bandwidth is essentially an irregular time series [9], [15] because different segments occupy varying intervals from download start and completion time. Therefore in order to predict the bandwidth, we need accurate timestamps which rely on both yet unavailable segment size and the predicted bandwidth itself. We illustrate this issue in Fig. 3. Specifically, assume we have downloaded 2 segments during interval (t_1^s, t_1^e) and (t_2^s, t_2^e). The rendered bandwidth c_1 and c_2 can be derived via our accurate measurement. To predict the next $c_3 = f(t_3^s, t_3^e)$, the circular dependency arises: the ending $t_3^e = t_3^s + \frac{s_3}{c_3}$ timestamp is uncertain because we lack the exact segment size s_3 and the not yet predicted c_3. For example, we need to query longer t_3^e in advance if the target c_3 is low and vice verse, which is however impossible to determine in the first place.

Moreover, LLLS presents more sophisticated problems. On the one hand, the future segment size is unknown, and estimation via target bitrate varies significantly depending on the content complexity. It also requires the ABR decisions prior to the bandwidth prediction, which imposes module coordination instead of typical sequential pipeline. On the other hand, we also cannot derive the starting t_3^s due to the inter-segment idleness Δt_2 from systematic latency. We clarify that Δt stems from fetching pause from player setting, which is different from inter-chunk idleness in Challenge 1. As a result, neither t_3^s nor t_3^e can be accurately determined, and the future bandwidth varies with different possible time intervals, as indicated by 3 different background colors.

Previous work. Existing advanced bandwidth prediction algorithms like VOD-based Lumos [10] and delay-based predictors CS2P [16] and T3P [17] all assume knowledge of remaining segment sizes and typical transmission without inter-chunk idle time, which clearly differs from various uncertainty features in LLLS and thus renders them inapplicable. While hybrid prediction method DeeProphet [11] is designed for live streaming, it still overlooks the essence of irregular time series and directly forecasts the bandwidth alone, which fails to capture the varying time interval and break the circular dependency. Even sophisticated time-series models like TimesNet [12], while powerful, are not designed to resolve this application-specific circularity or to account for the causal impact of ABR's bitrate.

Solution. To break the dependency, the key idea is not to predict each variable sequentially, but to directly learn all the uncertain variables in joint probability distribution $p(c_3, s_3, t_3^s, t_3^e | past_{1,2}, bitrate_3)$ from certain input like ABR-decided bitrates and historical download statistics. Therefore, we need to develop a distribution mapping model like normalizing flow [18] to handle multiple variable prediction. It also requires a novel feature embedding to tackle the time interval and capture the relationship between target bitrate and other variables like attention mechanism.

> **Insight 2:** To accurately forecast bandwidth in LLLS, we need to learn the joint distribution of bandwidth related variables, while also capturing the causal impact of ABR decision with irregular time embedding.

C. Challenge 3: Robust ABR Control under Uncertainty

Background. Typical ABRs are designed for VOD scenarios with more accurate bandwidth and an explicit download process. Therefore the optimized QoE metrics like bitrates and rebuffering penalty can adequately reflect the future performance and guide the ABR's bitrate selection. However, LLLS introduces a new set of uncertainties that challenge the assumption of accurate variables and QoE evaluation. Apart from the aforementioned bandwidth measurement and prediction issues, we also lack the specific segment size and face significant variability in inter-chunk idle time. This inherent uncertainty falsifies the ABR's download time and buffer evolution and eventually, the QoE metrics. As a result, traditional max-QoE models fall short in guiding ABRs to make robust decisions under various LLLS uncertainties.

We illustrate the impact of these variables in Fig. 4. We conduct experimental analysis with the representative VOD ABR RobustMPC (RMPC) [19] over a stable network link, isolating the impact of size and idle time uncertainty. Fig. 4(a) shows that even moderate errors in size estimation can lead to significant buffer prediction errors. Especially when with lower

TABLE I
SUMMARY OF NOTATIONS. NOTE THAT WE MAY USE VARIABLES ACROSS DIFFERENT CATEGORIES.

Category	Notation	Meaning	Category	Notation	Meaning
Measurement	N	number of segments	ABR	R_i	bitrate for seg_i
	K	number of CMAF chunks in a segment		N_h	future segment horizon of MPC
	Z_i	number of HTTP chunks in segment i		$s_i/s_{i,j}$	size of $seg_i/cc_{i,j}$
	seg_i	segment $i, i \in [1, N]$		$u_{i,j}$	idle time before downloading $cc_{i,j}$
	$cc_{i,j}$	CMAF chunk j of $seg_i, j \in [1, K]$		c_i	available bandwidth for seg_i
	$hc_{i,z}$	HTTP chunk z of $seg_i, z \in [1, Z_i]$		pc_i	predicted bandwidth for seg_i
	$t_{i,z}$	arrival time of $hc_{i,z}$ of seg_i		$d_i/d_{i,j}$	estimated download time of $seg_i/cc_{i,j}$
	$h_{i,z}$	size of $hc_{i,z}$ of seg_i		$d_i^*/d_{i,j}^*$	actual download time of $seg_i/cc_{i,j}$
	k_i	Flag for seg_i		$d_i^u/d_{i,j}^u$	upper bound of download time of $seg_i/cc_{i,j}$
	$m_{i,j}$	0/1 mask of $cc_{i,j}$ from Flag k_i		$b_{i,j}$	buffer after downloading $cc_{i,j}$
Prediction	N_p	length of historical observation		T	CMAF chunk duration $cc_{i,j}$
	t_i^s	download start time of seg_i		$l_i/l_{i,j}$	latency after downloading $seg_i/cc_{i,j}$
	t_i^e	download complete time of seg_i		$r_i/r_{i,j}$	rebuffer time after downloading $seg_i/cc_{i,j}$
	Δt_i	inter-segment idleness before seg_i		$p_i/p_{i,j}$	playback rate after downloading $seg_i/cc_{i,j}$

size estimation, RMPC tends to underestimate the download time and overestimate the next buffer, which is very likely followed by a stalling event due to smaller buffer in LLLS. In detail, Fig. 4 (b) presents the size error PDF of a 10 minutes video encoded at different bitrates (Kbps), which validates the significant gap of even twice the size estimation. Furthermore, Fig. 4(c) shows that server and client-side idle time can randomly constitute a significant fraction of the total download duration, severely falsifying the final buffer prediction. More importantly, a small mismatch in buffer prediction leads to obvious QoE degradation (refer to Table X in ablation study), e.g., either wasting bandwidth or causing stalling.

Previous work. Existing ABRs all fail to address the above uncertainty issue in LLLS due to suboptimal QoE objective. For instance, RobustMPC (RMPC) [19] estimates the bandwidth lower bound and iterates all bitrates combination based on the evolution model to maximize the expected QoE. Advanced methods like Pensieve [20] and Comyco [21] leverage reinforcement learning or imitation learning to achieve better performance from exploration or expert trajectory. Even LLLS ABRs like LoL+ [7] still operate on deterministic state estimates. They are not architected to systematically handle the multi-faceted inherent uncertainty

Solution. To ensure reliable performance, we need to incorporate the comprehensive uncertainties into the QoE model and the ABR logic. By guaranteeing the lower bound of the worst case, we can derive better QoE expectation from stochastic variables.

> **Insight 3:** To address the various uncertainties in LLLS, we need to guarantee the worst case performance via a robust QoE objective and control logic.

Algorithm 1: Heuristic bandwidth measurement

Input: Flag=k_i, received all HTTP chunks $hc_{i,z}$, fetch request time t_{req}
Output: c_i^{he}

if $k_i == K$ **then return** $\frac{s_i - h_{i,1}}{t_{i,Z_i} - t_{i,1}}$;
$bw \leftarrow []$;
$T^s, T^e \leftarrow extract(hc_{i,z})$; ▷ CMAF extraction
$end \leftarrow T^s_{k_i+1} == T^e_{k_i} ?\ T^e_{k_i} - 1 :\ T^e_{k_i}$; ▷ Burst chunks' last HTTP chunk
bw.append($[\frac{\sum_{z=2}^{end} h_{i,z}}{t_{i,end} - t_{i,1}}, \sum_{z=2}^{end} h_{i,z}]$); ▷ [bandwidth, size]
for $j \leftarrow k_i + 1$ *to* K **do**
  $end \leftarrow T^s_{j+1} == T^e_j ?\ T^e_j - 1 :\ T^e_j$;
  bw.append($[\frac{\sum_{z=T^s_j+1}^{end} h_{i,z}}{t_{i,end} - t_{i,T^s_j}}, \sum_{z=T^s_j+1}^{end} h_{i,z}]$);
end
$c_i^{he} \leftarrow \frac{\sum bw[:,0] \times bw[:,1]}{\sum bw[:,1]}$; ▷ Size weighted averaging
return c_i^{he}

III. PROPOSED METHOD: AAR

A. Overview of AAR

We first summarize the frequently used notations in Table I. To address the 3 challenges of inaccurate measurement, circular prediction, and robust control in LLLS, we propose AAR, an integrated framework comprising three co-designed modules. As illustrated in Fig. 5, the AAR pipeline operates as follows: when the client requests a segment, the server delivers via CTE-based HTTP chunks upon available CMAF chunks, along with our lightweight Flag parameter. (1) The Hybrid Measurement module leverages this Flag to isolate consecutive chunks and identify burst transmission. With the valid data, it derives both NN-based and heuristic measurement and selects via confidence score. (2) The prediction module formulates historical data statistics, along with the ABR's prospective bitrate choice. Then it derives bitrate-aware attention features as the condition for normalizing flow input. The inverse variable distribution can yield future bandwidth. (3) The ABR module optimizes a max-min QoE model via MPC and LLLS state evolution model, backed up by theoretical guarantees. For each future segment within horizon N_h, the corresponding bandwidth is obtained from prediction module in an autoregressive manner. Finally, the client requests the optimal next bitrate again in the pipeline.

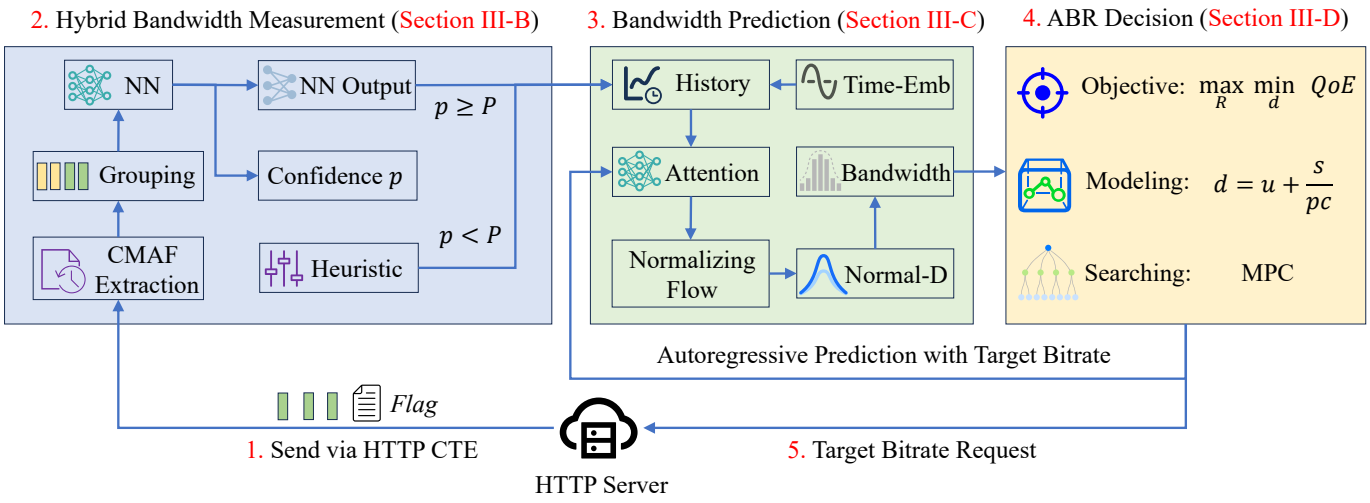


Fig. 5. Overview of the AAR framework. We first obtain the requested segment with our Flag parameter, then we leverage a hybrid model to derive accurate bandwidth measurement via confidence evaluation. The prediction module takes historical statistics with target bitrate from ABRs to yield future bandwidth. Finally, the ABR algorithm optimizes a max-min QoE objective with LLLS evolution model to derive optimal bitrate.

B. Hybrid Bandwidth Measurement

Identify Valid Chunks. Based on insight 1, we tend to identify the CMAF chunk sending pattern to accumulate consecutive HTTP chunks. However, this information is available only at the server side, which records and decides how many CMAF chunks are cached for a burst. Therefore, we propose to attach a Flag=$k_i \in [0, K]$ parameter for seg_i in CTE's header to denote the burst chunk number within the segment. This signal allows the client to definitively distinguish between HTTP chunks corresponding to this burst transmission and subsequent chunks that may have been delayed by the encoding process.

To construct the valid chunks for measurement samples, we first propose an improved CMAF boundary extraction algorithm over LoL+ [7] introduced in Section II-A, which checks the CMAF boundary only once in an HTTP chunk. We instead tend to keep parsing the same $hc_{i,z}$ in a loop in case a single HTTP chunk contains multiple CMAF chunks (e.g., No.4 and 6 in Fig. 1). In addition, we identify and fix some patches in $offset$ setting during the $moof+mdat$ boxes search in the latest dash.js. With accurate CMAF boundary (the sequences of HTTP chunks), we can confirm that the first k_i CMAF chunks are valid without idleness. In contrast, the subsequent CMAF chunks contain idle time that requires separate filtering. We then discard the HTTP chunks that contain the $moof$ (starting) box of the next adjacent CMAF chunk, because the arrival timestamps include the encoding delay according to the Flag parameter.

Heuristic Approach. With the above valid chunks, we first develop a heuristic measurement as a robust baseline. Due to the bursty bandwidth from HTTP-level chunks whose transmission can be easily perturbed, as demonstrated in Fig. 2, we opt to derive CMAF-level chunks' bandwidth. We aggregate the valid HTTP chunks as total size and compute the download duration as the arrival time interval, which yields a valid bandwidth sample without idleness. To reduce the noise deviation from small samples, we propose to leverage size

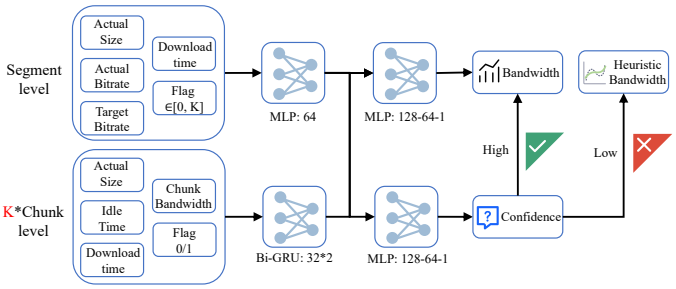


Fig. 6. The neural network of bandwidth measurement module.

weighted average to guarantee the priority of the k_i burst chunks, which contain an I frame that makes up 50% of the segment size.

We present the detailed procedure in Algorithm 1. First of all, we directly compute the segment bandwidth as in VOD if k_i is exactly K (line 1). Then we identify the CMAF boundary from $hc_{i,z}$ via our improved searching algorithm (line 3), we also discard the last hc if it comprises the next cc to filter idle time (line 4). The resulting T^s and T^e store the start and end sequence of HTTP chunk for each CMAF chunk, followed by our first and foremost bandwidth sample that comprises all the $hc_{i,z}$ for the burst k_i $cc_{i,j}$ (line 5). Finally, we compute the bandwidth of smaller CMAF chunks (lines 6-8), followed by size weighted averaging (line 10) to ensure the priority of our burst chunks' bandwidth.

Hybrid Model. To further enhance the accuracy, we propose a learning-based model to directly capture the intrinsic relationship between CMAF and segment-level bandwidth, while we can also incorporate the robust heuristic results to avoid NN overfitting via fidelity evaluation. We first formulate the necessary input features as follows:

• Segment-level statistics $x_{seg} \in \mathbb{R}^5$ including the segment size s_i, the actual and target bitrate $\hat{R}_i$ and R_i, the total download time $t_i^e - t_i^s$ that includes idleness, and the server-side Flag=k_i.

• CMAF-level statistics $x_{cmaf} \in \mathbb{R}^{K\times 5}$ with each including the CMAF chunk size $s_{i,j}$, the idle time from previous chunk $u_{i,j}$, the CMAF download time $d_{i,j}$, the CMAF bandwidth $s_{i,j}/d_{i,j}$ and a Flag-based mask $m_{i,j}$.

The segment features x_{seg} denote coarse-grained download process that includes idleness, while the K CMAF features x_{cmaf} manifest more accurate fine-grained details. The mask $m_{i,j}$ denotes whether chunk $cc_{i,j}$ is included in the burst transmission pattern, i.e. $m_{i,j} = 1$ if $j \leq k_i$ and 0 otherwise. The detailed neural network architecture is illustrated in Fig. 6. We leverage a simple Multilayer Perceptron (MLP) to map x_{seg} into $f_{seg} \in \mathbb{R}^{64}$ and a bi-directional GRU network to capture the temporal CMAF features $f_{cmaf} \in \mathbb{R}^{K*64}$. Specifically, we aggregate the burst and idle chunk features from GRU with the Flag mask $m_{i,j}$ to yield $f_1 \in \mathbb{R}^{k_i*64}$ and $f_2 \in \mathbb{R}^{(K-k_i)*64}$. Then we concatenate f_{seg}, the average of the mean of f_1 and f_2 to form the final feature vector $f_{mea} \in \mathbb{R}^{128}$:

$$f_{mea} = [f_{seg}, \frac{mean(f_1, dim = 0) + mean(f_2, dim = 0)}{2}] \tag{1}$$

Finally, we apply two separate MLPs to map f_{mea} to NN bandwidth estimation c_i^{nn} and a confidence score p, which evaluates how likely the measurement is more accurate than the simple average result c_i^{he}. The training loss is based on the mean squared error (MSE) between the predicted bandwidth c_i^{nn} and the ground truth c_i, while confidence is optimized via cross-entropy loss, i.e.

$$\mathcal{L}_{mea} = MSE(c_i^{nn}, c_i) + \lambda_{mea} * H(p, \mathbb{I}(|c_i^{nn} - c_i| \leq |c_i^{he} - c_i|)) \tag{2}$$

where $\mathbb{I}$ is the indicator function. In this way, we maximize probability p if c_i^{nn} is more accurate. Then we can derive an accurate and robust measurement as follows:

$$c_i = \mathbb{I}(p > P) \cdot c_i^{nn} + \mathbb{I}(p \leq P) \cdot c_i^{he} \tag{3}$$

Fallback Mechanism. where P is an optimal threshold that maximizes the overall accuracy. In practice, we search for P with validation datasets. In this way, we can leverage p to avoid the potential bias or overfitting in certain distribution. For example, NN models may exhibit higher accuracy for most regular and stable bandwidth as seen in the training dataset. In cases when $p \leq P$, e.g. low bandwidth samples that induce longer download time, we will fall back to our heuristic methods, because it performs better with size weighted average due to less perturbation by random noise.

C. Bandwidth Prediction via cNF

To address the circular dependency challenge outlined in Insight 2, we argue that bandwidth is not an independent variable but part of a complex system of interdependent distribution. We therefore propose a novel prediction framework that leverages conditional normalizing flows (cNFs) to learn the joint probability distribution of future network conditions, explicitly conditioned on the ABR's decisions and historical statistics.

cNF in LLLS. A normalizing flow [18] is a mapping f from $\mathbb{R}^D$ to $\mathbb{R}^D$ that transforms a complex, target data distribution x into a simple base probability distribution z (e.g., a standard multivariate Gaussian) through a series of invertible and differentiable transformations, i.e.

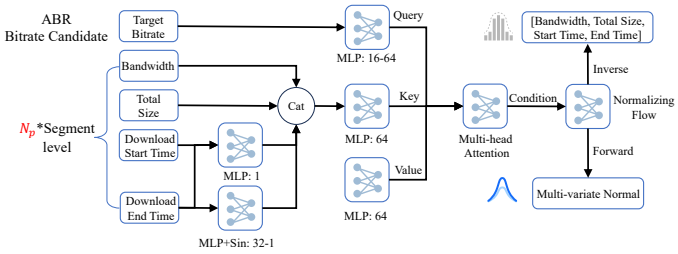


Fig. 7. The neural network of bitrate-aware transformer as feature condition for normalizing flow.

$$p_X(x) = p_Z(f(x)) \left| \det \left(\frac{\partial f(x)}{\partial x^T} \right) \right| \tag{4}$$

where p_X and p_Z are the probability densities, $z = f(x)$ and the inverse function $x = f^{-1}(z)$ can reconstruct the target data from normal distribution. The determinant of the Jacobian matrix, $\det(\partial f(x)/\partial x^T)$ accounts for the change in volume induced by the transformation. A conditional normalizing flow parameterized by θ extends this by making the transformation dependent on a context vector h, i.e., $z = f(x; h, \theta)$ and $x = f^{-1}(z; h, \theta)$. In the context of LLLS, we leverage N_p historical segment-level features $x_{i-N_p}, \ldots, x_{i-1}$ and the ABR's intended bitrate choice R_i as the conditioning context h_i to predict the future segment characteristics x_i, formulated as follow:

$$x_i \setminus R_i = f^{-1}(z_i; h_i, \theta) = f^{-1}(z_i; x_{i-N_p}, \ldots, x_{i-1}, R_i, \theta) \tag{5}$$

where $h_i = F(x_{i-N_p}^{i-1}, R_i)$ and $x_i = \{c_i, R_i, s_i, t_i^s, t_i^e\} \in \mathbb{R}^5, i \in [1, N]$. F converts the statistics into high-dimensional representation. Note that we exclude R_i in prediction since it comes from the ABR decision. As explained in Section II-B, s_i depends on target bitrate R_i and complex codec setting, t_i^s can be perturbed by inter-segment idle time Δt_i, while t_i^e is determined by the predicted c_i, s_i and complicated inter-chunk idleness $u_{i,j}$. Therefore, by modeling the joint distribution of these uncertain variables, we can derive a more accurate and robust bandwidth within the expected future distribution. Next we will discuss the cNF architecture and the constructed context features h_i.

cNF via RealNVP. We implement the flow using a stack of conditional affine coupling layers, based on the RealNVP architecture [22]. Each layer splits the input future $x = x_i \setminus R_i$ into two parts, x_a and x_b. It leaves x_a unchanged while transforming x_b using an affine transformation whose scale and shift parameters, (s, t), are generated by a neural network that takes x_a and the condition vector h as input:

$$z_a = x_a \quad \text{and} \quad z_b = x_b \odot \exp(s) + t \tag{6}$$

where $(s, t) = \text{MLP}(x_a, h)$. This transformation is easily invertible, and its Jacobian determinant is simply the sum of the elements in s, making log-likelihood computation efficient. We stack multiple such layers, flipping the roles of x_a and x_b at each step to ensure all dimensions are transformed.

Bitrate-Aware Transformer. The key to flow transformation is the context vector $h_i = F(x_{i-N_p}^{i-1}, R_i)$. To this end, we design a Bitrate-Aware Transformer to capture complex patterns and dependencies, as shown in Fig. 7. The detailed modules are as follows:

• Irregular Time Embedding: The segment-level bandwidth is an irregular time series with different time intervals. To efficiently capture the temporal dependency, we first encode the timestamp for each past segment by computing its midpoint $(t_i^e + t_i^s)/2$ and duration $t_i^e - t_i^s$. These two features are then passed through parallel linear and sinusoidal layers, where dimension i is defined as:

$$\phi(T)[i] = \begin{cases} \omega_{0h}T + \alpha_{0h}, & \text{if } i = 0 \\ \sin(\omega_{ih}T + \alpha_{ih}), & \text{if } 0 < i < 32 \end{cases} \tag{7}$$

where $T = [(t_i^e + t_i^s)/2, t_i^e - t_i^s]$. The concatenated output is time embedding $\phi(T) \in \mathbb{R}^{32}$. By combining with bandwidth and segment size, we have the N_p historical statistics $x_{his} \in \mathbb{R}^{N_p \times 34}$.

• Bitrate-as-Query Attention: We propose a novel QKV attention mechanism to explicitly model the influence of the ABR's decision. The historical data x_{his} are projected to form the Key (K) and Value (V) vectors. Crucially, the Query (Q) is instead a learned embedding of the future target bitrate, R_i, which motivates the model to learn an attention pattern that specifically answers: "Given this history, what context is most relevant if I choose bitrate R_i?"

With the QKV projection features, we can obtain the attention scores as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{8}$$

where $d_k = 64$ is the dimension of the key vectors. Finally, our transformer yields the final context features h_i with multi-head attention scores, which are then used as the condition for the cNF transformation.

Training and Inference. Training (Forward): The model parameter θ is trained by minimizing the batch negative log-likelihood (NLL):

$$\mathcal{L}_{pre} = -\sum_i \log p_X(x_i; h_i, \theta) \tag{9}$$

where h_i is the context features from our transformer.

Inference (Inverse): To make a prediction, we collect the historical data and a candidate future bitrate R_i from ABRs. The model first computes the corresponding condition vector h_i. Then we sample a vector z from the base Gaussian distribution and pass it through the inverse transformation $f^{-1}(z; h_i, \theta)$. This generates a sample x_i datapoint. By drawing multiple times, we can derive the expected bandwidth c_i as the mean of the samples.

D. Robust ABR with Max-min QoE

To address the profound challenge of bitrate control under uncertainty (Insight 3), we depart from conventional ABR designs that optimize for an expected QoE. Instead, we design a robust control policy that explicitly optimizes for worst-case performance, thereby guaranteeing a lower bound on QoE.

Max-min Objective. Following established QoE metrics from works like LoL+ [7] and standards in [23], the standard LLLS objective is to select a sequence of bitrates R_i to maximize a utility function that rewards high video quality R_i, low video rebuffering time r_i, low latency l_i, normal playback rate $p_i \to 1$ and low bitrate switches $|R_i - R_{i-1}|$, formulated as:

$$QoE = \sum_{i=1}^{N} (\alpha_1 R_i - \alpha_2 r_i - \alpha_3 l_i - \alpha_4 |p_i - 1|) - \sum_{i=2}^{N} \alpha_5 |R_i - R_{i-1}| \tag{10}$$

where $\alpha_k > 0, k \in [1, 5]$ are weights for each metric. As established in Section II-C and Fig. 4, this QoE is subject to uncertainty from idle time $u_{i,j}$, segment size variation s_i, and bandwidth prediction error pc_i. To create a robust ABR, we formulate a max-min objective that seeks to maximize the QoE achieved under the most adverse realization of these uncertainties:

$$\max_{R_i} \min_{\{u_{i,j}, s_i, pc_i\}} QoE \tag{11}$$

LLLS-Tailored Evolution Model. To optimize this objective, we must first establish how these uncertainties propagate to the QoE metrics. Existing segment-level models are inadequate for the fine-grained dynamics of LLLS. We therefore propose a new, chunk-level state evolution model:

$$d_{i,j} = u_{i,j} + \frac{s_{i,j}}{pc_i} \tag{12}$$

$$b_{i,j} = max(b_{i,j}^- - p_{i,j}^- \times d_{i,j}, 0) + T \tag{13}$$

$$r_{i,j} = max(d_{i,j} - \frac{b_{i,j}^-}{p_{i,j}^-}, 0) \tag{14}$$

$$l_{i,j} = l_{i,j}^- - (p_{i,j}^- - 1) \times min(d_{i,j}, \frac{b_{i,j}^-}{p_{i,j}^-}) + r_{i,j} \tag{15}$$

$$p_{i,j} = \begin{cases} f(l_{i,j}) > 1, & \text{if } l_{i,j} - l_{target} > \delta \\ 1, & \text{if } |l_{i,j} - l_{target}| < \delta \\ f(l_{i,j}) < 1, & \text{if } l_{i,j} - l_{target} < -\delta \end{cases} \tag{16}$$

where δ is the prefixed threshold and $x_{i,j}, x \in \{b, p, l\}$ is the attribute for the j^{th} CMAF chunk of seg_i, $x_{i,j}^- = x_{i,j-1}$ if $j > 1$ and $x_{i-1,K}$ if not. In LLLS, $r_i = \sum_{j=1}^{K} r_{i,j}$, $s_i = \sum_{j=1}^{K} s_{i,j}$ and l_i/p_i is the final latency/speed after downloading all the CMAF chunks. Specifically, the key variable is the download time $d_{i,j}$ which includes idle time $u_{i,j}$ and the actual transmission time $\frac{s_{i,j}}{pc_i}$, where $s_{i,j}$ is the estimated CMAF chunk size using prefixed bitrates. With $d_{i,j}$, we can predict the buffer $b_{i,j}$ change by draining $p_{i,j}^- \times d_{i,j}$ seconds of cached video and appending a new CMAF chunk duration T, along with possible stalling $r_{i,j}$.

Notably, our latency evolution model in Equ. 15 first catches up or loses behind at speed $p_{i,j}^- > 1$ or $p_{i,j}^- < 1$. The magnitude is up to the download time $d_{i,j}$ if there is no rebuffering, otherwise up to $\frac{b_{i,j}^-}{p_{i,j}^-}$ because there's no more content to playback after that. This corrects a critical oversight in prior work like Tightrope [24] and Fleet [8], which incorrectly model latency evolution as being dependent on the

total download time $d_{i,j}$ even during a stall. Regarding the playback speed $p^-_{i,j}$, it depends on current latency deviation around the threshold δ of target latency l_{target}, f function maps latency to speed according to a specific player.

Solving the Inner Minimization. Our chunk-level model reveals that all sources of uncertainty are reflected by the download time $d_{i,j}$. This allows us to simplify the inner minimization problem significantly. The objective becomes:

$$\max_{R_i} \min_{d_{i,j}} QoE \tag{17}$$

This transformation is valid because the uncertain variables $u_{i,j}, s_i, pc_i$ only affect the QoE penalty terms (rebuffering and latency) via their impact on $d_{i,j}$. We can therefore decompose the objective 11 into:

$$\begin{aligned}&\max_{R_i} \sum_{i=1}^{N} \min_{d_{i,j}\in[0,d^u_{i,j}]} [g_1(R_i,p_i)+g_2(d_{i,j})] = \\ &\max_{R_i} \sum_{i=1}^{N} g_1(R_i,p_i) + \min_{d_{i,j}\in[0,d^u_{i,j}]} g_2(d_{i,j})\end{aligned} \tag{18}$$

where $g_1(R_i,p_i)=\alpha_1 R_i-\alpha_4|p_i-1|-\alpha_5|R_i-R_{i-1}|$ and $g_2(d_{i,j})=-\alpha_2 r_i-\alpha_3 l_i$. The key challenge is to identify the $d_{i,j}$ that maximizes the rebuffering and latency. Note that we cannot accurately predict the actual download time like Fugu [25] and T3P [17] in VOD because it requires the knowledge of the exact next segment size. The solution is non-trivial, as latency $l_{i,j}$ is not always monotonic with respect to $d_{i,j}$, while rebuffering $r_{i,j}$ is. We provide a formal solution to this problem in Theorem 1.

***Assumption** 1:* Assume that $d^u_{i,j} \geq T$, meaning that each CMAF chunk exhibits at least T idle time waiting for the captured frames on the ingest side.

***Theorem** 1:* Let Assumption 1 hold, the min solution in Equ. 18 is when and only when for each $d_{i,j}=d^u_{i,j}, i\in[1,N], j\in[1,K]$.

Proof 1 (Theorem 1): First of all,

$$\begin{aligned}g_2(d_{i,j}) &= -\alpha_2 \sum_{j=1}^{K} r_{i,j} - \alpha_3 l_i \\ &= -\alpha_2 \sum_{j=1}^{K} max(d_{i,j} - \frac{b^-_{i,j}}{p^-_{i,j}}, 0) - \alpha_3 l_i\end{aligned} \tag{19}$$

Therefore $r_{i,j} \propto d_{i,j}$. As for latency:

$$l_{i,j} = \begin{cases} l^-_{i,j} - (p^-_{i,j}-1)\times\frac{b^-_{i,j}}{p^-_{i,j}}, & \text{if } d_{i,j} < \frac{b^-_{i,j}}{p^-_{i,j}} \\ l^-_{i,j} + d_{i,j} - b^-_{i,j}, & \text{if } d_{i,j} \geq \frac{b^-_{i,j}}{p^-_{i,j}} \end{cases} \tag{20}$$

Therefore $l_{i,j} \propto d_{i,j}$ except when $p^-_{i,j}>1$ and $d_{i,j} < \frac{b^-_{i,j}}{p^-_{i,j}}$. However, this is when $l^-_{i,j} - l_{target} > \delta$ and to achieve the term, $l_{i,j}$ would first go through $|l^-_{i,j} - l_{target}| \leq \delta$ stage where $p^-_{i,j}=1$, therefore $l_{i,j}=r_{i,j}$. This means there must have been rebuffering events for the first time $p^-_{i,j}>1$ and $b^-_{i,j}=T$, then we have $\frac{b^-_{i,j}}{p^-_{i,j}} < b^-_{i,j} = T$, and that $l_{i,j}$ first decreases and then increases as $d_{i,j}$ increases, $d_{i,j}\in[0,d^u_{i,j}]$. Based on Assumption 1, $l_{i,j}(d^u_{i,j}) \geq l_{i,j}(T) = l^-_{i,j} = l_{i,j}(0)$. Therefore we can still set $d_{i,j}=d^u_{i,j}$ to get the maximum latency $l_{i,j}$. Moreover, after this stage, $r_{i,j} = d^u_{i,j} - \frac{b^-_{i,j}}{p^-_{i,j}} > 0$, which means next time $b_{i,j}=T$, and this process continues.

In summary, we can always set $d_{i,j}=d^u_{i,j}$ to derive the maximum l_i, r_i and the minimum $g_2(d_{i,j})$. ■

Algorithm 2: AAR's LLLS tailored ABR

Input: estimated player idle time u_{client}, past CMAF chunk size $s^*_{i,j}$ and idle time $u^*_{i,j}, j\in[1,K]$, future horizon N_h, past $d^*_j, j\in[i-4,i]$
Output: R_{i+1}

$\Delta d = \frac{\sum_{j=i-4}^{i}|d^*_j - d_j|}{5}$;
for *each bitrate combination* **do**
  for $k \leftarrow i+1$ *to* $i+N_h$ **do**
    for $j \leftarrow 1$ *to* K **do**
      $u_{k,j} \leftarrow u^*_{i,j}$;
      if $j == 1$ **then** $u_{k,j} \leftarrow u_{k,j} + u_{client}$;
      pc_k, s_k =BandwidthPrediction(R_k);
      $s_{k,j} \leftarrow \frac{s_k \times s^*_{i,j}}{s^*_i}$;
      $d_{k,j} \leftarrow u_{k,j} + \frac{s_{k,j}}{pc_k}$;
      $d^u_{k,j} \leftarrow d_{k,j} + \frac{|\Delta d|\times d_{k,j}}{d_k}$;
      Other variable evolution in Equ. 13-16;
    end
  end
  Compute QoE as in Equ. 18 with Theorem 1;
end
$R_{i+1} \leftarrow argmax(QoE)$;
return R_{i+1}

TABLE II
BANDWIDTH (KBPS) DISTRIBUTION OF DATASETS.

Datasets \ Metrics	Mean	Std	25th	50th	75th
FCC	1526	1288	596	901	3426
Oboe	2966	1552	1765	2940	3957
3G/HSDPA	1933	1053	1130	1836	2472
Online	8594	5674	2789	8115	15359

Upper Bound Estimation. To apply Theorem 1 in practice, we collect the past actual segment download time d^*_i from the player feedback to compute the maximum estimated download time deviation. Specifically, we propose to store historical segment level d-error as $\Delta d_i = |d^*_i - d_i|$, then we distribute such Δd_i proportionally to each $d_{i,j}$ by the ratio over d_i to obtain the upper bound $d^u_{i,j} = d_{i,j} + \frac{|\Delta d_i|\times d_{i,j}}{d_i}$. Note that we do not store $\Delta d_{i,j}$ because the actual $d^*_{i,j}$ is not available since the sending time can be falsified by idle time in HTTP chunks.

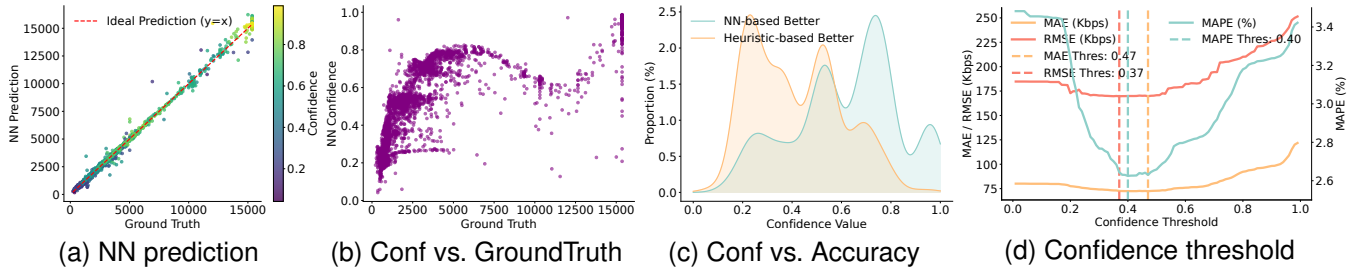
To guarantee that Theorem 1 does not impact the maximum QoE when the LLLS uncertainty is eliminated, we propose Theorem 2 to justify our $d^u_{i,j}$ computation:

***Theorem** 2:* With accurate $u_{i,j}, s_{i,j}$ and pc_i, objective 18 falls back to regular objective $\max_{R_i} QoE$.

Proof 2 (Theorem 2): The proof is intuitive: accurate

TABLE III
BANDWIDTH MEASUREMENT PERFORMANCE FOR DIFFERENT METHODS ACROSS 4 DATASETS. WE PRESENT MAPE↓, MAE↓ AND RMSE↓. FOR MAE AND RMSE, WE DIVIDE THE RAW VALUE BY 100 TO ALIGN THE NUMBER PRECISION. THE RAW RESULTS CAN BE FOUND IN ABLATION STUDY IN TABLE VII.

Dataset	Metrics	AAR	AAR-Base	Fleet	I-moof	Moof	AAST	Seg	Default	DeeProphet
FCC	MAPE	2.31±0.85	2.55±0.97	21.5±4.07	18.7±2.99	68.2±5.53	24.9±4.50	45.1±6.54	5679	11.8
	MAE	0.27±0.06	0.34±0.08	2.02±0.15	2.09±0.33	9.81±2.18	4.65±1.43	8.81±3.16	3806	/
	RMSE	0.44±0.08	0.57±0.10	4.00±0.41	3.39±0.52	14.1±2.38	6.11±1.43	15.1±3.48	8627	
Oboe	MAPE	2.57±0.40	3.59±0.48	13.8±2.06	21.0±5.17	69.6±5.97	27.3±4.66	61.1±7.84	1762	11.8
	MAE	0.65±0.13	1.04±0.21	3.41±0.67	5.71±1.96	17.9±1.50	7.80±1.34	20.6±4.35	2035	/
	RMSE	1.44±0.23	1.84±0.29	4.90±0.92	9.61±1.94	20.9±1.72	8.96±1.37	27.4±4.55	5013	
HSDPA	MAPE	3.16±0.23	3.97±0.26	17.4±1.91	27.6±3.55	76.1±4.00	27.3±3.57	59.8±5.15	2336	
	MAE	0.57±0.08	0.77±0.11	2.72±0.31	4.97±0.66	13.9±1.45	5.74±1.02	12.7±2.54	2253	/
	RMSE	0.95±0.12	1.16±0.14	4.31±0.48	7.09±0.75	16.5±1.48	6.91±1.08	17.0±2.61	5405	
Online	MAPE	1.96±0.17	3.02±0.20	18.1±1.97	18.8±4.36	45.5±7.46	30.3±4.93	85.7±3.78	562	
	MAE	1.25±0.29	2.44±0.57	17.4±4.41	14.7±5.40	25.2±2.16	29.3±7.59	78.4±15.5	1390	/
	RMSE	2.94±0.50	4.27±0.72	24.4±4.40	25.8±5.49	31.0±2.66	39.7±8.32	97.6±15.4	2959	



(a) NN prediction (b) Conf vs. GroundTruth (c) Conf vs. Accuracy (d) Confidence threshold

Fig. 8. Illustration of AAR's hybrid measurement. (a) presents the NN prediction accuracy with estimated confidence, (b)-(c) shows the distribution of confidence scores and (d) presents the optimal threshold searching process.

variables imply that $\Delta d_i = |d_i^* - d_i| = 0$, thus $d_{i,j}^u = d_{i,j} + \frac{|\Delta d_i| \times d_{i,j}}{d_i} = d_{i,j}$. Therefore the min solution is straightforward. In contrast, if we adopt $\Delta d_i = |d_i^* - d_i^u| = |d_i - d_i^u| \neq 0$, we will overestimate $d_{i,j}^u$ which is supposed to be $d_{i,j}$, therefore the min solution is not maximized. ■

Bad Case: Assumption 1 holds in most real-world cases. In rare scenarios when it fails, e.g. due to frame capture rate increase on the ingest side, Theorem 1 is not guaranteed because the download time is overestimated and hence bitrates are conservative. However, the min solution is still near optimal because the rebuffering r_i often imposes a much heavier penalty and it is always monotonically increasing with $d_{i,j}$. Therefore, $d_{i,j}^u$ still drives the ABR towards the most pessimistic (and thus robust) QoE. In addition, after the temporary failure cases, we can adjust the T variable in QoE reward Equ. 10 in accordance with the actual ingest FPS.

Implementation via MPC. With the min problem solved via Theorem 1, we address the outer maximization using Model Predictive Control (MPC). As detailed in Algorithm 2, the MPC controller systematically explores all feasible bitrate plans over a finite future horizon N_h. For each candidate target bitrate pair, it simulates the system's evolution using our LLLS-tailored model. We first derive the average d-error from the past 5 segments (line 1), while future bandwidth pc_k and predicted segment size s_i from bitrate R_i can be derived by our cNF prediction module (line 7) in an autoregressive manner. $s_{k,j}$ is estimated by the last segment's chunk size ratio. For each step in the simulation, it evaluates the QoE penalty using the worst-case download time $d_{k,j}^u$, as dictated by Theorem 1. The controller then selects the bitrate plan that yields the highest minimum guaranteed QoE (line 14) and executes the first bitrate choice of that optimal plan (line 16).

IV. EVALUATION

A. Experimental Setup

Testbed and Configuration. We conduct our experiments using a testbed comprising an HTTP server and a client running dash.js (v.4.7.2), each deployed on a separate Ubuntu server. The streaming configuration is aligned with the mmsys-grand-challenge [26]: we target a latency of 1.5s with a stability threshold of $\delta = 0.3$s. Each segment has a duration of 0.5s and contains $K = 15$ CMAF chunks, corresponding to a 30 FPS video where each chunk has a duration of $T = 33$ms. We use a standard DASH reference video [27] encoded at six distinct bitrates: $R_i \in \{200, 600, 1000, 2500, 4000, 6000\}$ Kbps.

Network Conditions. We emulate dynamic network conditions using the Chrome DevTools network throttling feature. Our evaluation employs four diverse and challenging bandwidth trace datasets: the publicly available FCC [28] and 3G/HSDPA [29] datasets, the Oboe [30] dataset collected from cellular networks, and a proprietary real-world dataset (Online) captured from a large-scale e-commerce live streaming

TABLE IV
BANDWIDTH PREDICTION PERFORMANCE FOR DIFFERENT METHODS ACROSS 4 DATASETS. WE PRESENT MAPE↓, MAE↓ AND RMSE↓. FOR MAE AND RMSE, WE DIVIDE THE RAW VALUE BY 100 TO ALIGN THE NUMBER PRECISION. THE RAW RESULTS CAN BE FOUND IN ABLATION STUDY IN TABLE VIII.

Dataset	Metrics	AAR	AAR-Base(Avg)	EWMA	Harmonic	TimesNet	Crossformer	Mamba	DeeProphet
FCC	MAPE	3.36±0.36	5.18±0.63	4.87±0.45	4.69±0.49	5.00±0.55	9.00±0.87	5.70±0.62	16.5
	MAE	0.40±0.04	0.60±0.06	0.44±0.05	0.61±0.07	0.44±0.05	0.83±0.08	0.51±0.06	/
	RMSE	1.47±0.98	2.34±1.21	1.76±1.05	2.53±1.32	2.12±1.11	1.06±0.69	0.85±0.45	
Oboe	MAPE	3.28±0.22	5.34±0.34	4.49±0.26	5.25±0.32	6.68±0.41	10.0±0.65	7.40±0.43	16.5
	MAE	0.77±0.05	1.48±0.09	0.97±0.07	1.51±0.09	2.15±0.15	3.19±0.20	2.40±0.17	/
	RMSE	2.16±1.02	3.69±1.38	2.74±1.16	3.81±1.46	5.08±2.01	5.87±2.48	5.25±2.31	
HSDPA	MAPE	9.11±0.41	15.6±0.58	11.4±0.46	15.2±0.51	17.9±0.69	26.2±0.95	18.9±0.78	
	MAE	1.50±0.06	2.56±0.07	1.93±0.06	2.58±0.07	2.86±0.09	3.32±0.11	3.02±0.09	/
	RMSE	2.61±0.90	3.73±1.05	3.00±0.93	3.78±1.06	4.12±1.21	4.32±1.35	4.23±1.18	
Online	MAPE	7.32±0.92	15.0±1.57	10.5±1.14	13.0±1.06	10.5±1.12	18.7±1.85	11.5±1.20	
	MAE	2.58±0.24	5.52±0.35	3.84±0.28	5.56±0.37	4.33±0.30	7.56±0.58	4.82±0.32	/
	RMSE	9.30±4.71	14.0±5.38	11.0±4.83	14.8±5.87	13.4±5.91	14.1±6.03	13.7±5.53	

application. For AAR's hybrid measurement and prediction model training, we split all the traces by 7:1.5:1.5 for training, validation, and testing. All learning models are deployed on a single NVIDIA GeForce RTX 3090. The statistical properties of these datasets are summarized in Table II.

Baselines. For all tasks, we include the previous version AAR-Base [2] with purely heuristic bandwidth measurement and prediction for comparison. We also compare AAR against a comprehensive set of state-of-the-art baselines:

• For bandwidth measurement, we include 7 representative methods: (1) Fleet [8]; (2) DeeProphet [11]; (3) Moof, the original parsing logic from LoL+ [7]; (4) I-Moof, our implementation of LoL+ using AAR's improved moof+mdat boundary identification (line 3 in Algorithm 1); (5) AAST [31], it decides which chunks are downloaded at network speed or at producer rate; (6) Seg, a naive segment-level measurement; (7) Default, the built-in filtering logic in dash.js.

• For bandwidth prediction, we select 7 representative methods: (1) Avg, the average of the last N_p measurements; (2) EWMA, exponential weighted moving average; (3) Harmonic, harmonic mean of the last N_p measurements, and transformer-based time series prediction model (4) TimesNet [12]; (5) Crossformer [32]; (6) Mamba [33], a structured state space model; (7) DeeProphet [11].

• For ABR algorithms, we select 6 representative schemes: (1) LoL+ [7], self organizing maps-based learning model; (2) L2ALL [34]; (3) STALLION [35], a latency-aware ABR; (4) RB, a classic rate-based heuristic; (5) Dynamic, the default ABR in dash.js; (6) Pensieve [20], a seminal reinforcement learning ABR designed for VOD, which we retrain on our LLLS simulator using the QoE model from Equ. 10. Note that the RMPC baseline is presented in ablation study, refer to Section IV-E for setup.

Algorithm Parameters. We set $N_p = N_h = 5$ for historical segment statistics and future horizon. The searched optimal threshold for confidence $P = 0.40$, loss weights $\lambda_{mea} = 0.5$ and batch size is 32. The neural network dimensions can be found in Fig. 6 and Fig. 7. Note that the input parameters from Algorithm 1 and 2 are not prefixed and they depend on the specific streaming content. While the weights in the QoE model are aligned with previous work [7], [11]: $R_{min} = 200, R_{max} = 6000$, and $\alpha_1 = 0.5, \alpha_2 = R_{max}, \alpha_3 = 0.02 \times R_{min}$ if $l_i < 1.6$ and $0.1 \times R_{max}$ if not, $\alpha_4 = R_{min}, \alpha_5 = 1$.

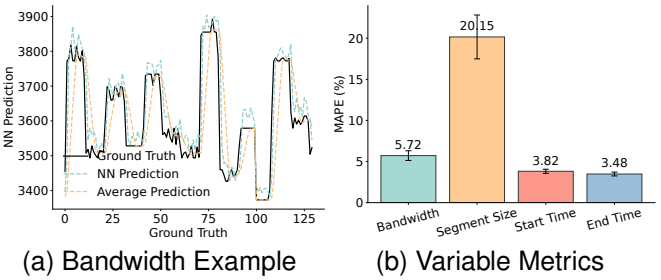


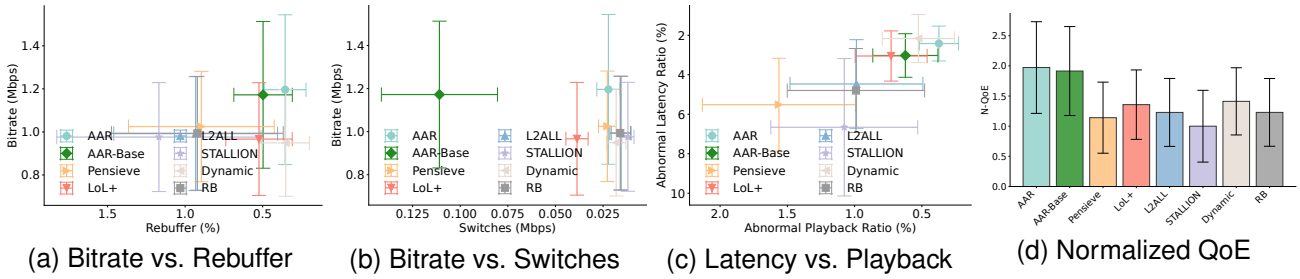
Fig. 9. Illustration of AAR's bandwidth prediction. (a) presents the bandwidth evolution performance and (b) shows the MAPE error ↓ of joint variable distributions.

Evaluation Metrics. We evaluate performance using two primary categories of metrics: (1) Accuracy: For measurement and prediction tasks, we report Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). For MAE and RMSE, we convert the bandwidth back to its original scale for fair comparison. (2) QoE: We use the objective function defined in Equ. 10, with weights α_k set according to the N-QoE standard [23], consistent with prior work [7], [11].

B. Bandwidth Measurement Results

Metrics Comparison. We first evaluate the core accuracy of AAR's hybrid measurement module. To isolate its performance, we use a fixed Rate-Based (RB) ABR for all measurement schemes. Table III presents the results. Note that we apply AAR's improved CMAF identification patch on all the baselines except for Moof. We directly report DeeProphet's performance stated in [11].

AAR consistently and significantly outperforms all baselines across all four datasets, achieving 11%-83% reduction in MAPE. Specifically, compared to the previous state-of-the-



(a) Bitrate vs. Rebuffer (b) Bitrate vs. Switches (c) Latency vs. Playback (d) Normalized QoE

Fig. 10. QoE and detailed metrics comparison for FCC dataset. We present the normalized QoE ↑ over the worst STALLION baseline, bitrates in Mbps ↑, rebuffer in stalling ratio % ↓, abnormal latency ratio % ↓, abnormal playback speed ratio % ↓, and bitrate switches in Mbps ↓. For (a)-(c), the upper right, the better.

art, DeeProphet, AAR still demonstrates a clear advantage with an 8%-10% lower error. However, DeeProphet requires extra setup on the server side, and the measurement result c_i is stored in server's database, which can only be delivered to the client upon seg_{i+1} request. As a result, it imposes a bitrate lower than the measured bandwidth from the server side to prevent such delay, which has altered the whole LLLS framework and is not practical to deploy. The substantial improvement of I-Moof over the original Moof (up to 50% error reduction) validates the critical importance of our improved CMAF boundary identification logic. Schemes like AAST and Seg perform poorly as they fail to account for chunk-level idle time, while the default dash.js filter is the least effective due to its aggressive and noise-sensitive filtering strategy. These results confirm that AAR's Flag-based hybrid approach provides fundamentally more accurate and robust bandwidth estimates.

Illustration. To better understand the hybrid model's mechanism, we present some bandwidth and confidence distributions in Figure 8. Fig. 8(a) shows the distribution of NN-based bandwidth estimation, which closely aligns with the ground truth. The heat map indicates that low confidence scores mainly come with low bandwidth scenarios, and (b) further demonstrates the correlation. This is due to longer transmission time which is less likely to be perturbed by random noises. Therefore heuristic methods can provide more robust results with interpretable weighted average. (c) shows that for lower confidence of NN output, the heuristic method indeed exhibits better performance, which in turn validates the effectiveness of our threshold-based model selection. (d) further presents the searching process of optimal threshold. The results for 3 different metrics show similar trends and we choose 0.40 to maximize MAPE performance. To summarize, AAR's hybrid measurement model effectively combines the strengths of both NN-based and heuristic methods, achieving superior accuracy and robustness.

C. Bandwidth Prediction Results

Metric Comparison. We evaluate the performance of our cNF-based prediction model, and we fix the measurement as our accurate AAR for fair comparison. Note that we only report the next segment bandwidth results to compute accuracy. As shown in Table IV, AAR's predictor achieves the highest accuracy across all metrics and datasets, with up to 17% MAPE reduction against one of the SOTA model Crossformer. This superiority stems from our cNF design that breaks the circular dependency by jointly modeling the distribution of multiple variables and thus rendering reliable expectation. We also note that simple heuristics like Average even surpass the transformer-based models like TimesNet, because more complex networks do not necessarily lead to better performance without LLLS tailored designs for irregular time embedding and circular dependency.

Illustration. For detailed performance, we present a bandwidth evolution example in Figure 9(a), where AAR's predicted bandwidth closely follows the ground truth, while naive approach Average exhibits significant delay from future distribution. Figure 9(b) qualitatively shows the joint distribution accuracy of our cNF. We can find that all 4 interdependent variables exhibit low MAPE error, especially for download timestamp forecasting, demonstrating the effectiveness of our irregular time embedding. The segment size error is instead higher, which can be further minimized by incorporating the video content characteristics and video codec setting. In summary, our cNF-based prediction model outperforms all the baselines with both better quantitative and qualitative results.

D. ABR QoE Results

To assess the end-to-end QoE results, we integrate AAR's full pipeline and compare against the baseline ABRs, all of which use AAR's high-fidelity measurement and prediction module for a fair comparison. The aggregated statistics for QoE scores and individual components are shown in Fig. 10-13. Note that we normalize the QoE value by the worst baseline. Across all network conditions, AAR achieves the highest overall QoE ranging from 1.5 to 2.0, with improvements up to 102%. This mainly stems from high bitrates and low rebuffering simultaneously, as seen in subplots (a) of each figure. In contrast, the state-of-the-art LLLS ABR, STALLION, often suffers from higher rebuffering, particularly in the highly variable HSDPA traces. Dynamic and Pensieve, while stable, are overly conservative, sacrificing significant bitrate for safety, e.g., for Oboe dataset. AAR's robust max-min objective allows it to aggressively pursue high bitrates when safe and gracefully adapt when uncertainty is high, leading to superior and more reliable QoE.

Compared with AAR-Base, AAR mainly improves in terms of rebuffering and bitrate switches, thanks to our accurate

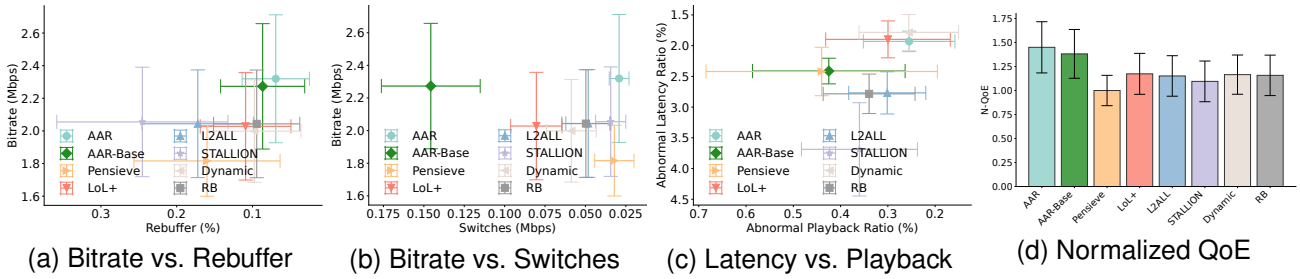


Fig. 11. QoE and detailed metrics comparison for Oboe dataset. We present the normalized QoE ↑ over the worst Pensieve baseline, bitrates in Mbps ↑, rebuffer in stalling ratio % ↓, abnormal latency ratio % ↓, abnormal playback speed ratio % ↓, and bitrate switches in Mbps ↓. For (a)-(c), the upper right, the better.

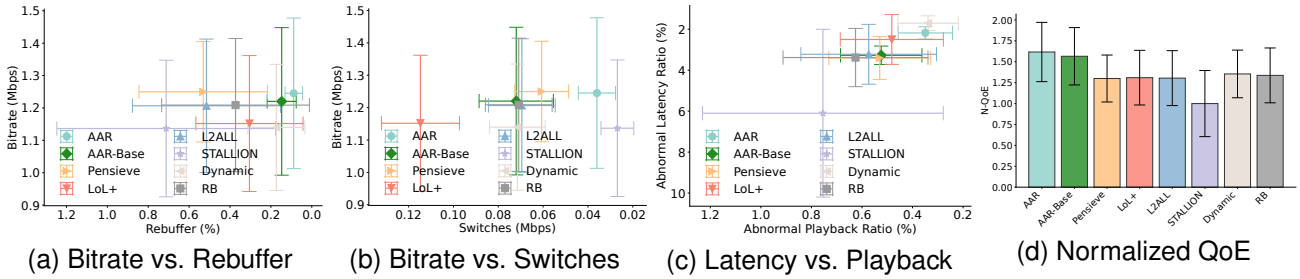


Fig. 12. QoE and detailed metrics comparison for HSDPA dataset. We present the normalized QoE ↑ over the worst STALLION baseline, bitrates in Mbps ↑, rebuffer in stalling ratio % ↓, abnormal latency ratio % ↓, abnormal playback speed ratio % ↓, and bitrate switches in Mbps ↓. For (a)-(c), the upper right, the better.

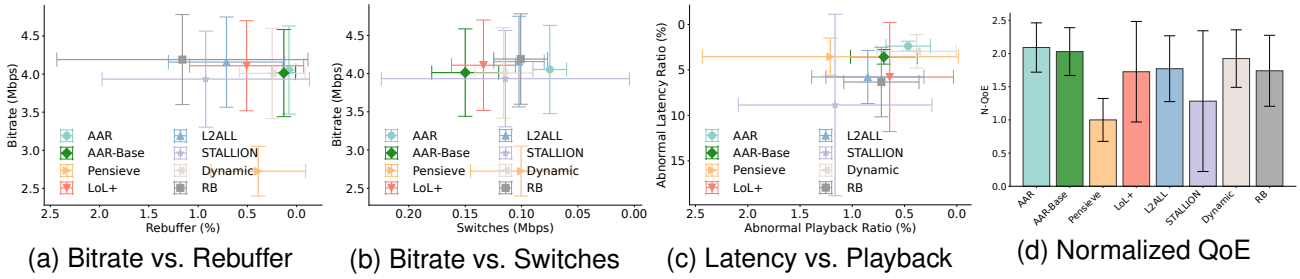


Fig. 13. QoE and detailed metrics comparison for Online dataset. We present the normalized QoE ↑ over the worst Pensieve baseline, bitrates in Mbps ↑, rebuffer in stalling ratio % ↓, abnormal latency ratio % ↓, abnormal playback speed ratio % ↓, and bitrate switches in Mbps ↓. For (a)-(c), the upper right, the better.

bandwidth prediction and hence better state evolution. The overall QoE gain is yet marginal, because the absolute value is already near optimal such as 0.5% stalling for FCC dataset. Moreover, the reward for switches is relatively low as defined by previous work [26], compared with major penalty minimization from rebuffering. For other baselines, the metric gain over those in [2] mainly stems from improvement in switches and abnormal latency ratio, e.g. from 2.5% to 1.0% in FCC dataset. The reason is that we leverage a cNF-based bandwidth prediction that requires bitrates in advance. However, these methods do not apply bitrate combination in a state evolution model like AAR. Therefore we directly use the last decided bitrate as input, which is suboptimal and less effective. In general, AAR gains the most QoE improvement from our coordinated systems and outperforms existing ABR baselines with higher QoE, bitrates and lower rebuffing.

E. Ablation Study

Impact of AAR's Modules on ABR QoE. We first investigate the symbiotic relationship between our modules and various ABR algorithms. We replace our hybrid measurement model with the previous SOTA Fleet [8] and present the corresponding normalized QoE. The results are in Table V. We can find that without our accurate measurement, all the ABRs perform worse with different magnitudes, while Pensieve (from 3.10 to 1.00 in FCC) and RB (from 3.32 to 1.65 in FCC) suffer significantly higher degradation. This also demonstrates that solely relying on bandwidth for bitrate selection can be easily perturbed.

We further replace our cNF-based prediction with the SOTA time series model TimesNet [12] and present the results in Table VI. The ABRs still degrade without accurate bandwidth. However, since the fundamental measurement is still accurate, the prediction errors can be mitigated with lower QoE degradation.

TABLE V
ABR'S QOE ↑ WITH DIFFERENT BANDWIDTH MEASUREMENTS. WE NORMALIZE WITH THE WORST QOE FROM BOTH MEASUREMENTS.

Dataset	Mea \ ABR	AAR	Pensieve	LoL+	L2ALL	STALLION	Dynamic	RB
FCC	AAR	5.33±2.05	3.10±1.63	3.69±1.57	3.35±1.54	2.73±1.55	3.85±1.55	3.32±1.55
	Fleet	5.26±2.03	1.00±2.02	3.43±1.67	1.42±2.23	2.53±1.77	4.01±1.48	1.65±2.11
Oboe	AAR	1.46±0.27	1.02±0.19	1.21±0.16	1.18±0.25	1.12±0.24	1.20±0.22	1.18±0.23
	Fleet	1.28±0.30	1.00±0.20	1.21±0.22	1.20±0.29	1.11±0.24	1.15±0.26	1.18±0.23
HSDPA	AAR	1.90±0.41	1.54±0.37	1.60±0.34	1.58±0.35	1.20±0.51	1.63±0.28	1.60±0.42
	Fleet	1.85±0.38	1.25±0.49	1.45±0.39	1.36±0.66	1.00±0.59	1.50±0.29	1.34±0.65
Online	AAR	2.25±0.40	1.11±0.39	1.92±0.78	1.93±0.50	1.43±1.17	2.11±0.49	1.92±0.61
	Fleet	2.24±0.42	1.00±0.69	1.99±0.73	1.51±0.99	1.75±0.62	2.06±0.43	1.76±0.79

TABLE VI
ABR'S QOE ↑ WITH DIFFERENT BANDWIDTH PREDICTIONS. WE NORMALIZE WITH THE WORST QOE FROM BOTH PREDICTIONS.

Dataset	Mea \ ABR	AAR	Pensieve	LoL+	L2ALL	STALLION	Dynamic	RB
FCC	AAR	3.17±1.22	1.84±0.97	2.19±0.93	1.99±0.92	1.62±0.92	2.29±0.92	1.97±0.92
	TimesNet	3.15±1.20	1.00±1.19	2.07±1.00	1.21±1.34	1.53±1.06	2.34±0.90	1.30±1.24
Oboe	AAR	1.43±0.26	1.02±0.19	1.19±0.16	1.16±0.25	1.10±0.24	1.18±0.22	1.16±0.23
	TimesNet	1.29±0.31	1.00±0.18	1.21±0.20	1.15±0.31	1.11±0.26	1.13±0.24	1.14±0.25
HSDPA	AAR	1.75±0.38	1.42±0.34	1.47±0.31	1.45±0.32	1.10±0.47	1.50±0.26	1.47±0.39
	TimesNet	1.73±0.33	1.26±0.43	1.36±0.37	1.30±0.58	1.00±0.56	1.40±0.25	1.30±0.57
Online	AAR	2.11±0.38	1.04±0.37	1.80±0.73	1.81±0.47	1.34±1.10	1.98±0.46	1.80±0.57
	TimesNet	2.08±0.42	1.00±0.62	1.84±0.66	1.57±0.94	1.56±0.61	1.97±0.43	1.69±0.76

TABLE VII
ABLATION STUDY OF AAR'S HYBRID BANDWIDTH MEASUREMENT. WE PRESENT MAPE ↓, MAE ↓ AND RMSE ↓.

Dataset	FCC			Oboe			HSDPA			Online		
Metric	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE
AAR (Combined)	2.31	27.4	44.4	2.57	64.5	144.2	3.16	56.7	95.1	1.96	124.8	294.3
w/o NN	2.55	34.1	57.4	3.59	104.6	183.7	3.97	77.3	115.8	3.02	243.5	426.8
w/o Heuristic	4.01	40.1	62.6	3.13	71.1	162.2	4.11	64.0	103.1	2.25	135.8	348.4
w/o FlagAgg	2.62	37.4	68.7	3.38	95.7	171.0	3.90	74.1	124.5	3.17	264.7	451.8

Deconstructing AAR's measurement. Next we evaluate the effectiveness of each module of AAR. For bandwidth measurement, we remove some designs of our hybrid model and present the results in Table VII. We can find that without either the heuristic (4.01% in MAPE) or the NN (2.55% in MAPE) component results in a clear performance degradation, because the NN output exhibits higher accuracy in most cases, as shown in Fig. 8 (a), while the heuristic can guarantee more robust results in low bandwidth scenarios. Moreover, removing the Flag-based feature aggregation from the NN also causes a drop in accuracy, proving that this lightweight server signal is essential for the model to distinguish idle time from network burst.

Deconstructing AAR's prediction. Similarly, we ablate our cNF predictor and present the results in Table VIII. We can find that removing our novel interval time embedding or the bitrate-as-query attention mechanism (replaced with regular attention) degrades accuracy, confirming their importance in handling irregular time series and modeling ABR's impact. More notably, replacing the entire flow mechanism with a simple MLP leads to the largest performance drop, highlighting the critical role of probabilistic modeling in capturing LLLS uncertainty and breaking the circular dependency.

Parameter Sensitivity. We study the impact of 2 fixed parameters in Table IX. For bandwidth measurement, we leverage a confidence threshold P to determine optimal method. Since P is selected from validation dataset search, we can also apply RMSE and MAE threshold besides the default

TABLE VIII
ABLATION STUDY OF AAR'S CNF-BASED BANDWIDTH PREDICTION. WE PRESENT MAPE ↓, MAE ↓ AND RMSE ↓.

Dataset	FCC			Oboe			HSDPA			Online		
Metric	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE
AAR	3.36	39.8	146.9	3.28	77.2	216.0	9.11	150.3	260.8	7.32	258.3	929.5
w/o TimeEmbed	3.87	44.5	171.0	3.86	87.5	247.2	10.22	177.6	312.9	8.65	286.3	1095.3
w/o Attention	4.27	47.8	184.6	4.07	97.6	258.4	11.71	192.1	314.6	8.76	328.8	1174.3
w/o Flow	4.83	56.3	219.7	4.77	116.2	310.7	13.45	227.3	390.4	10.49	369.4	1291.5

TABLE IX
ABLATION STUDY OF PARAMETERS. WE PRESENT MAPE ↓, MAE ↓ RMSE ↓ AND N-QOE ↑.

Dataset		FCC				Oboe				HSDPA				Online			
Metric		MAPE	MAE	RMSE	N-QoE	MAPE	MAE	RMSE	N-QoE	MAPE	MAE	RMSE	N-QoE	MAPE	MAE	RMSE	N-QoE
Threshold	RMSE P=0.37	2.40	28.3	45.4	1.95	2.58	64.7	143.7	1.45	3.17	56.9	95.0	1.61	1.98	125.0	295.1	2.07
	MAPE P=0.40	2.31	27.4	44.4	1.97	2.57	64.5	144.2	1.45	3.16	56.7	95.1	1.61	1.96	124.8	294.3	2.09
	MAE P=0.47	2.32	27.5	44.4	1.98	2.56	64.3	143.2	1.45	3.18	57.0	95.7	1.59	1.94	124.6	297.2	2.08
Horizon	N_h=4				1.98				1.45				1.62				2.08
	N_h=5		/		1.97		/		1.45		/		1.61		/		2.09
	N_h=8				1.93				1.43				1.55				2.05

MAPE P. The results show that MAE and MAPE threshold generally perform better in terms of their own metric, but the absolute gap among the 3 thresholds is small. The overall QoE instead favors MAPE more, which is our default selection. Alternatively, we can also adjust the ABR horizon lengths. The results are generally worse with higher N_h, because the bandwidth prediction is less accurate if the output is longer, which is an inherent issue in time series forecasting domain. Low N_h may induce better bandwidth results but it fails to consider longer consequence like potential empty buffer. We choose $N_h = 5$ for tradeoff between ABR robustness and bandwidth accuracy.

Deconstructing AAR's ABR. Finally, we validate the design of our robust ABR controller and max-min QoE design. We first remove our max-min QoE by replacing the min solution $d^u_{i,j}$ with regular $d_{i,j}$, and we adopt VOD RMPC's bandwidth prediction error module $pc \leftarrow \frac{pc}{1+|\Delta pc|}$, denoted as AAR-1 (RMPC). Then we change our $d^u_{i,j}$ estimation in Theorem 2 by using $\Delta d_i = |d^*_i - d^u_i|$ to estimate the upper bound $d^u_{i,j}$, denoted as AAR-2. We present the corresponding QoE and buffer evolution prediction error (in MAPE) in Table X. AAR outperforms both variants, improving QoE by 2%-19% and reducing buffer prediction error by 0.55%-2.6%. AAR-1 ranks the bottom because our max-min objective is the essential guarantee for robustness against all possible LLLS uncertainty. AAR-2 still exhibits high B-Error because of the inappropriate estimation of $d^u_{i,j}$, as validated by Theorem 2.

To better perceive the difference, we present intuitive examples of buffer evolution and prediction errors in Fig. 14. We can find in (a) that AAR's predicted buffer closely tracks the ground truth, even when the segment size varies around No. 40 in (d), thanks to our principled max-min objective and the adjusted download time estimation. In contrast, AAR-1 performs worse because RMPC only tackles bandwidth uncertainty while neglecting size variation, therefore the size fluctuation around No. 40 in (e) leads to inferior buffer prediction in (b). AAR-2 adopts $\Delta d_i = |d^*_i - d^u_i| > 0$ even with low size deviation around segment 50 in (f), which renders varying download time and buffer estimation. Therefore the latency also fluctuates in (c) due to playback speed adjustment to guarantee the buffer threshold (1.5s).

TABLE X
QOE ↑ (NORMALIZED OVER THE WORST BASELINE AS IN FIG. 10-13) AND B-ERROR ↓ ABLATION STUDY OF AAR'S ABR.

Dataset	Metric \ ABR	AAR	AAR-1 (RMPC)	AAR-2
FCC	QoE	1.97±0.75	1.71±0.70	1.94±0.77
	B-Error	6.61%±1.55%	7.24%±1.98%	7.16%±1.82%
Oboe	QoE	1.45±0.26	1.28±0.35	1.41±0.24
	B-Error	4.33%±0.31%	6.39%±0.68%	6.33%±0.53%
3G/HSDPA	QoE	1.61±0.34	1.35±0.31	1.60±0.34
	B-Error	5.33%±0.44%	7.61%±0.82%	7.56%±0.81%
Online	QoE	2.09±0.36	2.01±0.30	2.05±0.30
	B-Error	4.76%±1.03%	7.36%±1.55%	6.81%±1.23%

TABLE XI
TIME AND MEMORY OVERHEAD OF AAR.

Category	Training			Inference		
	Time	GPU Memory	CPU Memory	Time	GPU Memory	CPU Memory
Measurement	∼5 min	437MB	1019.18MB	1ms	373MB	777.51MB
Prediction	∼10min	849MB	2107.74MB	3ms	367MB	1962.84MB

F. Overhead Analysis

Flag Mechanism and ABR logic. The server-side Flag parameter is a lightweight addition to the HTTP header and does not introduce any delay. At the client side, our robust ABR avoids expensive online optimization by leveraging the closed-form solution from Theorem 1. Therefore, these two components do not incur any additional overhead.

Hybrid Measurement and cNF Prediction. For the learning model in measurement and prediction, we present the

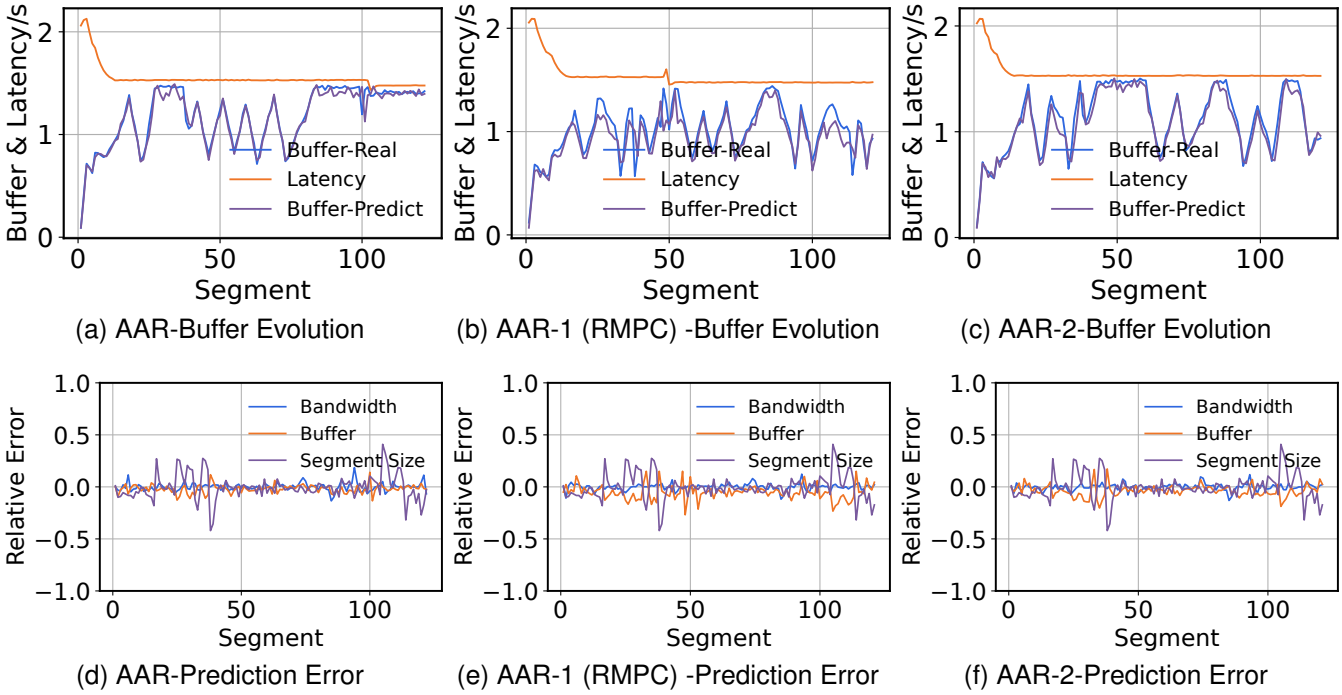


Fig. 14. Buffer evolution (a)-(c) and corresponding relative prediction error (d)-(f).

training and inference overhead in terms of time and memory. As shown in Table XI, the computational overhead for our learning-based modules is minimal during inference, requiring only 1-3ms per decision. This minimal overhead, combined with significant performance gains, demonstrates AAR's feasibility for real-world deployment.

V. Related Work

A. Bandwidth Measurement in LLLS

Application-Layer Heuristics. The most common methods operate entirely at the application layer, using HTTP chunk arrival times to infer network capacity. Naive approaches, such as measuring the download time of a whole segment or the simple filtering logic in dash.js (Default), are easily misled by server-side idle time. More sophisticated heuristics have been proposed to mitigate this. The method in LoL+ [7] (Moof) attempts to isolate the download time of individual CMAF chunks by parsing for moof and mdat boxes. Fleet [8] refines this by aggressively filtering small HTTP chunks to reduce noise. However, as we demonstrate in our evaluation, these methods remain vulnerable. They make incorrect assumptions about chunk boundaries (LoL+) and they are highly sensitive to network noise (Fleet).

Cross-Layer Approaches. To gain more precise information, some methods leverage data from lower network layers. DeeProphet [11] leverages the TCP cwnd from server's transport layer to decide which of the packets are sent consecutively, reducing TCP blocking time. However, this induces additional setup overhead and the measured results can't be delivered to the clients before the segment request and ABR decision, which greatly hinders real-world deployment. CLBE [36] also proposes to use the captured packet information on the client side to compute packet-level bandwidth. However, it requires queries from third-party module via WebSocket, which can induce additional time overhead, and a single packet's bandwidth still comprises deviation from noises like Fleet.

B. Bandwidth Prediction in LLLS

With the rise of deep learning, researchers have applied sophisticated, general-purpose time-series models to bandwidth prediction. These include state-of-the-art Transformer-based architectures like TimesNet [12], Crossformer [32], Mamba [33] and TimeMixer [37]. While powerful in standard forecasting benchmarks, these models are ill-suited for the LLLS ABR problem. They are not designed to handle the core challenge of circular dependency with irregular time sampling.

Our work is most related to models that learn to predict within the streaming context. DeeProphet [11] uses a learning-based model but still frames the problem as forecasting a future value. VOD-based Lumos [10] and delay-based predictor CS2P [16] and T3P [17] all assume knowledge of the exact next segment sizes without inter-chunk idle time during transformation, which fails to meet the requirements of various LLLS features.

C. Adaptive Bitrate Algorithms

VOD ABR Algorithms. The history of ABR starts with heuristic methods like rate-based and RMPC [19], followed by deep learning-based schemes like Pensieve [20] and Comyco [21]. Karma [38] improves QoE by learning the causality among past observations, returns and actions. Jade [39] instead proposes to optimize QoE by aligning with user's scores

via RLHF [40]. SODA [41] proposes to optimize a time-based QoE with theoretical guarantees, while MAFL [42] instead evaluates the robustness of federated learning [43] based ABRs.

LLLS ABR Algorithms. More recent work has focused specifically on LLLS. These algorithms extend the VOD QoE objective to include penalties for latency and playback speed adjustments like LoL+ [7] in Section II-C. L2ALL [34] instead solves an online convex optimization problem to derive optimal bitrate. STALLION [35] improves bandwidth measurement with the mean and deviation for better decisions. Tightrope [24] proposes the BDQ framework that leverages reinforcement learning to control both bitrates and playback speed. However, it's only implemented in offline simulator and lacks compatibility with real world streaming. SLVS [44] also proposes to generate a parameter table in a simulator according to bandwidth distribution for hybrid ABR to fine-tune parameters.

VI. Conclusion

In this work, we present 3 challenges in LLLS in terms of inaccurate bandwidth measurement, irregular prediction with circular dependency and various LLLS uncertainties for ABR control. Based on the insights, we propose the AAR framework comprising 3 modules: (1) We propose to attach a Flag parameter to identify the burst CMAF chunks to derive valid samples. Then we propose a hybrid approach to combine our two proposed heuristic and learning-based models to yield high-fidelity bandwidth measurement. (2) We propose a novel conditional normalizing flow-based model to learn the joint distribution of interdependent variables based on novel attention features to predict the expected bandwidth. (3) We propose a novel max-min objective to guarantee the lower bound of QoE backed up by two theorems. We further propose a novel LLLS model and apply MPC to search for the optimal bitrate. Real-world experiments demonstrate that AAR outperforms 7 measurement, 7 prediction and 6 ABR baselines with significant improvement.

References

[1] Cisco, "Vni complete forecast highlights," 2021. [Online]. Available: https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf

[2] J. Chen, Y. Yu, L. Wang, Y. Chen, T. Huang, and L. Sun, "Enhanced bandwidth measurement and robust rate adaptation for low-latency live streaming," in *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 2025, pp. 1–10.

[3] J. Li, Z. Li, R. Lu, K. Xiao, S. Li, J. Chen, J. Yang, C. Zong, A. Chen, Q. Wu *et al.*, "Livenet: a low-latency video transport network for large-scale live streaming," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 812–825.

[4] Dash-Industry-Forum, "lldash," 2020. [Online]. Available: https://github.com/Dash-Industry-Forum/dash.js/wiki/Low-Latency-streaming

[5] I. O. for Standardization and I. E. Commission, "Multimedia application format (mpeg-a)–part19: Common media application format (cmaf) for segmented media. standard iso/iec 23000-19:2018," 2018. [Online]. Available: https://www.iso.org/standard/71975.html,2018

[6] R. Fielding and J. Reschke, "Hypertext transfer protocol – http/1.1, rfc 7230." 2014. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7230,2014

[7] A. Bentaleb, M. N. Akcay, M. Lim, A. C. Begen, and R. Zimmermann, "Catching the moment with lol+ in twitch-like low-latency live streaming platforms," *IEEE Transactions on Multimedia*, vol. 24, pp. 2300–2314, 2021.

[8] Y. Li, X. Zhang, C. Cui, S. Wang, and S. Ma, "Fleet: Improving quality of experience for low-latency live video streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[9] S. N. Shukla and B. M. Marlin, "Multi-time attention networks for irregularly sampled time series," *arXiv preprint arXiv:2101.10318*, 2021.

[10] G. Lv, Q. Wu, W. Wang, Z. Li, and G. Xie, "Lumos: Towards better video streaming qoe through accurate throughput prediction," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 650–659.

[11] K. Chen, B. Wang, W. Wang, X. Li, and F. Ren, "Deeprophet: Improving http adaptive streaming for low latency live video by meticulous bandwidth prediction," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 2991–3001.

[12] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "Timesnet: Temporal 2d-variation modeling for general time series analysis," *arXiv preprint arXiv:2210.02186*, 2022.

[13] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[14] K. Rasul, A.-S. Sheikh, I. Schuster, U. Bergmann, and R. Vollgraf, "Multivariate probabilistic time series forecasting via conditioned normalizing flows," *arXiv preprint arXiv:2002.06103*, 2020.

[15] M. Schirmer, M. Eltayeb, S. Lessmann, and M. Rudolph, "Modeling irregular time series with continuous recurrent units," in *International conference on machine learning*. PMLR, 2022, pp. 19 388–19 405.

[16] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 272–285.

[17] H. Su, S. Wang, S. Yang, T. Huang, and X. Ren, "Reducing traffic wastage in video streaming via bandwidth-efficient bitrate adaptation," *IEEE Transactions on Mobile Computing*, vol. 23, no. 11, pp. 10 361–10 377, 2024.

[18] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," *Advances in neural information processing systems*, vol. 30, pp. 2335–2344, 2017.

[19] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 325–338.

[20] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the conference of the ACM special interest group on data communication*, 2017, pp. 197–210.

[21] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, "Comyco: Quality-aware adaptive video streaming via imitation learning," in *Proceedings of the 27th ACM international conference on multimedia*, 2019, pp. 429–437.

[22] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[23] "Qoe evaluation for cmaf-based low-latency streaming," 2020. [Online]. Available: https://github.com/twitchtv/acm-mmsys-2020-grand-challenge/blob/master/NQoE.pdf

[24] L. Sun, T. Zong, S. Wang, Y. Liu, and Y. Wang, "Tightrope walking in low-latency live streaming: Optimal joint adaptation of video rate and playback speed," in *Proceedings of the 12th ACM Multimedia Systems Conference*, 2021, pp. 200–213.

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

[26] "Grand challenge on adaptation algorithms for near-second latency," 2020. [Online]. Available: https://2020.acmmmsys.org/lll_challenge.php

[27] (2022) bbb-videos. [Online]. Available: https://dash.akamaized.net/akamai/bbb_30fps/

[28] F. C. Commission, "Raw data - measuring broadband america," 2016. [Online]. Available: https://www.fcc.gov/reports-research/reports/

[29] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, 2013, pp. 114–118.

[30] "Oboe trace," 2022. [Online]. Available: https://github.com/USC-NSL/Oboe

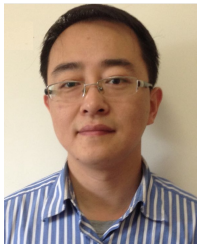
[31] "Aast," 2022. [Online]. Available: https://github.com/Dash-Industry-Forum/dash.js/pull/3660

[32] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *The*

eleventh international conference on learning representations, 2023, pp. 1–21.
[33] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," in *First Conference on Language Modeling*, 2024, pp. 1–32. [Online]. Available: https://openreview.net/forum?id=tEYskw1VY2
[34] T. Karagkioules, R. Mekuria, D. Griffioen, and A. Wagenaar, "Online learning for low-latency adaptive streaming," in *Proceedings of the 11th ACM Multimedia Systems Conference*, 2020, pp. 315–320.
[35] C. Gutterman, B. Fridman, T. Gilliland, Y. Hu, and G. Zussman, "Stallion: video adaptation algorithm for low-latency video streaming," in *Proceedings of the 11th ACM Multimedia Systems Conference*, 2020, pp. 327–332.
[36] C. Shende, C. Park, S. Sen, and B. Wang, "Cross-layer network bandwidth estimation for low-latency live abr streaming," in *Proceedings of the 14th Conference on ACM Multimedia Systems*, 2023, pp. 183–193.
[37] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, L. Ma, J. Y. Zhang, and J. ZHOU, "Timemixer: Decomposable multiscale mixing for time series forecasting," in *The Twelfth International Conference on Learning Representations*, 2024, pp. 1–27. [Online]. Available: https://openreview.net/forum?id=7oLshfEIC2
[38] B. Xu, H. Chen, and Z. Ma, "Karma: Adaptive video streaming via causal sequence modeling," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 1527–1535.
[39] T. Huang, R.-X. Zhang, C. Wu, and L. Sun, "Optimizing adaptive video streaming with human feedback," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 1707–1718.
[40] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *Advances in neural information processing systems*, vol. 30, 2017.
[41] T. Chen, Y. Lin, N. Christianson, Z. Akhtar, S. Dharmaji, M. Hajiesmaili, A. Wierman, and R. K. Sitaraman, "Soda: An adaptive bitrate controller for consistent high-quality video streaming," in *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024, pp. 613–644.
[42] R.-X. Zhang and T. Huang, "Adversarial attacks on federated-learned adaptive bitrate algorithms," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, 2024, pp. 419–427.
[43] J. Chen, Y. Zhao, Q. Li, X. Feng, and K. Xu, "Feddef: defense against gradient leakage in federated learning-based network intrusion detection systems," *IEEE Transactions on Information Forensics and Security*, 2023.
[44] G. Zhang, K. Liu, M. Xiao, B. Wang, and V. Aggarwal, "An intelligent learning approach to achieve near-second low-latency live video streaming under highly fluctuating networks," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 8067–8075.

Jiahui Chen received the B.S. degree from the School of Computer Science and Technology at Huazhong University of Science and Technology, China, in 2022. He is currently working toward the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University, China. His research work focuses on multimedia network streaming, including adaptive video streaming and low-latency live streaming.

Yiding Yu received the BE degree in communication engineering from the University of Electronic Science and Technology of China, China, in 2016, and the PhD degree in information engineering from The Chinese University of Hong Kong, Hong Kong, in 2020. He is currently an Algorithm Expert at Alibaba Group, where he focuses on optimizing transmission algorithms for Taobao Live. His research interests include media transmission protocols and algorithms, particularly in real-time communication and live streaming, as well as wireless communications and reinforcement learning.



Libo Wang received the B.S., M.S., degrees in Mathematics Science from Shanghai JiaoTong University, China, in 2001 and 2005, respectively. He is currently a Senior Algorithm Expert at Taobao, Alibaba Group, in Hangzhou, China, where he leads the Audiovisual Codec and Transmission Algorithm team. His research interests include video coding, audio enhancement, and real-time media transmission.

Ying Chen (Senior Member, IEEE) received the B.S. degree in applied mathematics and the M.S. degree in electrical engineering and computer science from Peking University in 2001 and 2004, respectively, and the Ph.D. degree in computing and electrical engineering from the Tampere University of Technology (TUT), Finland, in 2010. He joined Alibaba Group in 2018. Before joining Alibaba, he was a Principal Engineer/Manager with Qualcomm Inc., San Diego, CA, USA, from 2009 to 2018; a Researcher with TUT and the Nokia Research Center, Finland, from 2006 to 2009; and a Research Engineer with Thomson Corporate Research, Beijing, from 2004 to 2006. He is currently leading the Audiovisual Technology Group, Taobao, Alibaba, supporting end-to-end multimedia features and applications within Taobao. He contributed to three generations of video coding standards, H.264/AVC, H.265/HEVC, and H.266/VVC and video file format and transport standards. He served as an Editor and a Software Coordinator for H.264/AVC and H.265/HEVC (both for Multiview and 3D Video extensions) standards, and an associated editor of IEEE T-CSVT. His research focuses on video coding and transmission, video restoration/enhancement, video quality assessment and AIGC. He has led his team to multiple wins in top-tier challenges, including first places in the CVPR NTIRE 2023 Video Quality Assessment and CVPR NTIRE 2022 Challenge on Super-Resolution and Quality Enhancement of Compressed Video, and the MSU Video Codec competitions. He has authored or co-authored about 100 academic articles and over 250 granted U.S. patents. His publications have been cited for more than 20000 times.

Tianchi Huang received the PhD degree from the Department of Computer Science and Technology, Tsinghua University, in 2023. His research work focuses on multimedia network streaming, including transmitting streams, and edge-assisted content delivery. He has been the reviewer for IEEE Transactions on Vehicular Technology, IEEE Transactions on Mobile Computing, and IEEE Transactions on Multimedia. He received the Best Student Paper Award presented by ACM Multimedia System 2019 Workshop and Best Paper Nomination presented by ACM Multimedia Asia 2022.

Lifeng Sun received the B.S and Ph.D degrees in system engineering from National University of Defense Technology, Changsha, Hunan, China, in 1995 and 2000, respectively. He joined Tsinghua University since 2001. He is currently a Professor with the Computer Science and Technology Department of Tsinghua University, Beijing. Prof. Sun's research interests include the area of networked multimedia, video streaming, 3D/multiview video coding, multimedia cloud computing, and social media.