# CS2P: Improving Video Bitrate Selection and Adaptation with Data-Driven Throughput Prediction

Yi Sun[⊗], Xiaoqi Yin[†], Junchen Jiang[†], Vyas Sekar[†]
Fuyuan Lin[⊗], Nanshu Wang[⊗], Tao Liu[⋄], Bruno Sinopoli[†]
[⊗] ICT/CAS, [†] CMU, [⋄] iQIYI
{sunyi, linfuyuan, wangnanshu}@ict.ac.cn, yinxiaoqi522@gmail.com,
junchenj@cs.cmu.edu, vsekar@andrew.cmu.edu, liutao@qiyi.com,
brunos@ece.cmu.edu

## ABSTRACT

Bitrate adaptation is critical to ensure good quality-of-experience (QoE) for Internet video. Several efforts have argued that accurate throughput prediction can dramatically improve the efficiency of (1) *initial* bitrate selection to lower startup delay and offer high initial resolution and (2) *midstream* bitrate adaptation for high QoE. However, prior efforts did not systematically quantify real-world throughput predictability or develop good prediction algorithms. To bridge this gap, this paper makes three contributions. First, we analyze the throughput characteristics in a dataset with 20M+ sessions. We find: (a) Sessions sharing similar key features (e.g., ISP, region) present similar initial throughput values and dynamic patterns; (b) There is a natural "stateful" behavior in throughput variability within a given session. Second, building on these insights, we develop CS2P, a throughput prediction system which uses a data-driven approach to learn (a) clusters of similar sessions, (b) an initial throughput predictor, and (c) a Hidden-Markov-Model based midstream predictor modeling the stateful evolution of throughput. Third, we develop a prototype system and show using trace-driven simulation and real-world experiments that: (1) CS2P outperforms existing prediction approaches by 40% and 50% in terms of the median prediction error for initial and midstream throughput and (2) CS2P achieves 3.2% improvement on overall QoE and 10.9% higher average bitrate over state-of-art Model Predictive Control (MPC) approach, which uses Harmonic Mean for throughput prediction.

## CCS Concepts

•**Information systems** → **Multimedia streaming;** •**Networks** → *Transport protocols; Network measurement;*



*SIGCOMM '16, August 22 - 26, 2016, Florianópolis, Brazil*

DOI: http://dx.doi.org/10.1145/2934872.2934898

## Keywords

Internet Video; TCP; Throughput Prediction; Bitrate Adaptation; Dynamic Adaptive Streaming over HTTP (DASH)

## 1 Introduction

There has been a dramatic rise in the volume of HTTP-based adaptive video streaming traffic in recent years [1]. Delivering good application-level video quality-of-experience (QoE) entails new metrics such as low buffering or smooth bitrate delivery [5, 22]. To meet these new application-level QoE goals, video players need intelligent bitrate selection and adaptation algorithms [27, 30].

Recent work has shown that accurate throughput prediction can significantly improve the QoE for adaptive video streaming (e.g., [47, 48, 50]). Specifically, accurate prediction can help in two aspects:

- *Initial bitrate selection:* Throughput prediction can help select a suitable initial bitrate when a video session starts. Today's video players either have to conservatively start with a low bitrate and converge slowly to the optimal bitrate or alternatively incur high startup delay.
- *Midstream bitrate adaptation:* While it is possible to develop adaptation approaches without using throughput estimation (e.g., using only the playback buffer occupancy [27]), recent work [47] argues that throughput-aware bitrate adaptation can deliver a better QoE than pure buffer-occupancy based approaches.

Even though prior work [47, 50] suggests the potential benefits of throughput prediction, they fall short of providing concrete prediction algorithms that achieve high accuracy for real-world video sessions. Despite the rich measurement literature in characterizing various Internet path properties (e.g., [21, 26, 43]), our understanding of throughput variability and predictability is quite limited.[1]

As a first step to bridge this gap, we analyze intra- and inter-session throughput predictability using a large dataset from iQIYI [8], a leading commercial video provider in China. We evaluate a range of proposed prediction approaches (e.g., [24, 28, 34, 41]) and find that these prior approaches fail to meet the accuracy needed to deliver good

[1]There has been surprisingly little work and the closest efforts we are aware of are dated and limited in scope [17, 49].

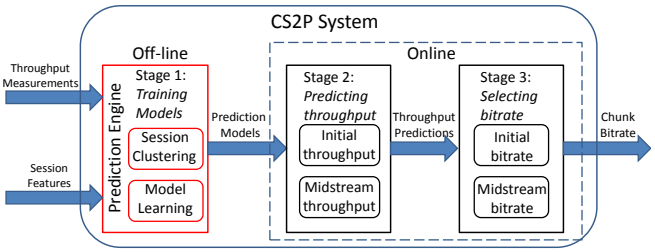



**Figure 1: Overall workflow of CS2P.**

QoE. In particular, we find that these models are not expressive enough to capture the diversity of real-world throughput patterns (e.g., bottlenecks can occur everywhere along the transmission path) and the dynamics of throughput evolution within each session (e.g., simple models that use the previous chunk throughputs are very noisy).

Our analysis also reveals two key insights that form the basis for our proposed design. First, we observe that similar sessions (i.e., sessions sharing the same critical features such as ISP, location) tend to have similar initial and average throughput values and even exhibit similar structural properties in throughput variation. This resonates with the findings in recent work [29] that, at the application layer, similar sessions have similar video QoE performance. Second, even though the observed throughputs for each video chunk within a session are inherently noisy, they do exhibit natural stateful evolving behaviors. Specifically, we see that the throughput is relatively stable and persistent for some duration of time but occasionally switches to a different state and persists in the new state(s).

Building on these data-driven insights, we develop the CS2P (Cross Session Stateful Predictor) approach for improving bitrate selection and adaptation (Figure 1). CS2P uses a data aggregator (called Prediction Engine) which builds prediction models using observed throughputs from past video sessions. This Prediction Engine uses an offline clustering step to identify sessions that are likely to exhibit similar throughput patterns. For each cluster, CS2P employs a simple approach to predict initial throughput by using the median throughput of the sessions in this cluster. To improve midstream prediction, CS2P learns a Hidden-Markov-Model (HMM) for each cluster to model the stateful evolution of intra-session throughput. The initial throughput and midstream evolution models can then be plugged into the bitrate selection and adaptation algorithms running either in the video players [30,47] or content delivery servers [14,20]. In the broader design space of video delivery, CS2P can be viewed as a middle ground between fully centralized control planes (e.g., C3 [23], CFA [29]) and fully decentralized bitrate adaptation approaches (e.g., Buffer Based-BB [27], FESTIVE [30], Model Predictive Control (MPC) [47]) as it uses the centralized visibility to develop better throughput prediction models but uses decentralized mechanisms to execute the actual adaptation decisions.

Our trace-driven simulations show that CS2P outperforms other throughput predictors and reduces the median prediction error for initial and midstream throughput by 40% and 50% respectively. Moreover, CS2P can drive median overall QoE to >93% of offline optimal when combined with MPC [47]. We also conduct pilot real-world experiments using an open-source player [3] and deploy CS2P in the operational platform of iQIYI. The results show that CS2P + MPC improves overall QoE by 3.2% and average bitrate by 10.9% compared with the state-of-art HM (Harmonic Mean) + MPC strategy [47], and can accurately predict the total rebuffering time at the beginning of the session.

**Contributions and Roadmap:** In summary, this paper makes three key contributions:

1. A large-scale analysis of throughput stability and predictability which highlights key challenges in predicting the throughput accurately and suggests data-driven insights that form the basis for our design (§3).
2. The CS2P architecture for improving bitrate selection and adaptation via throughput modeling (§4) and a practical prediction framework that can capture the diverse and stateful behaviors observed (§5).
3. A practical implementation in a video player (§6) and the demonstration of the improvements in prediction accuracy and QoE using trace-driven evaluations and pilot deployments (§7).

We discuss related work in §8, before concluding in §9. In the next section, we start by motivating the need for accurate throughput prediction for bitrate selection and adaptation (§2).

## 2 Background and Motivation

We begin with a high-level overview of how HTTP-based adaptive video streaming works and then highlight why we need good throughput prediction.

**Basics of HTTP-based bitrate adaptation:** In HTTP-based video delivery, videos are typically segmented into chunks and each chunk is encoded at different bitrate levels. Chunks from different bitrate streams are aligned so that the video player can smoothly switch to a different bitrate, if necessary, at chunk boundaries.

The player uses bitrate selection and adaptation algorithms that choose the bitrate levels for future chunks to deliver the highest possible QoE. Here, the adaptation algorithm needs to balance multiple QoE considerations as discussed in prior work [15,16,22,47]. These include the initial startup latency for the video to start playback, the amount of rebuffering the user experiences during the session, the average bitrate of the rendered video, and the smoothness of the rendered video as measured by the number of bitrate switches. Since many of these requirements are intrinsically at odds with each other, the design of this adaptation algorithm is non-trivial and there has been considerable interest in recent years in addressing this problem (e.g., [22, 30, 31, 47]).

**Need for better throughput prediction:** Even though it is possible to design adaptation strategies that avoid any form of throughput prediction (e.g., [27]), accurate throughput prediction can help in two aspects:

1. **Initial bitrate selection:** A video player should ideally pick the highest initial bitrate that is sustainable (i.e., be-

| Streaming protocol | Examples | Limitations | How throughput prediction helps |
|---|---|---|---|
| Fixed bitrate | NFL, Lynda, NY-Times | Bitrate too low, a few chunks are wasted to probe throughput | Higher bitrate with no rebuffering or long startup time |
| Adaptive bitrate | ESPN, Vevo, Netflix | | |

**Table 1: Limitations of current initial bitrate selection.**

low the throughput). Existing approaches to initial bitrate selection without accurate throughput prediction, however, are inefficient. Table 1 shows anecdotal evidence of such inefficiencies from several commercial providers. By analyzing the performance of their players, we categorize them into two main cases: (1) fixed-bitrate and (2) adaptive playback. Fixed-bitrate players that use the same bitrate for the whole video session often intentionally use low bitrate to prevent midstream rebuffering (e.g., NFL, Lynda). Even if bitrate can be adapted midstream (e.g., [10, 27, 30]) the player may conservatively start with a low bitrate and take a long time to reach the optimal bitrate (e.g., Netflix). Furthermore, for short video clips such adaptation may not reach the desired bitrate before the video finishes (e.g., Vevo music clips).[2]

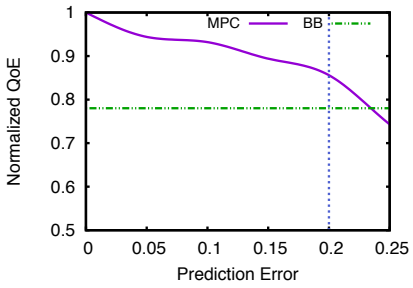



**Figure 2: Midstream QoE v.s. prediction accuracy. Actual QoE is normalized w.r.t. the theoretical optimal achievable with perfect knowledge of future throughput.**

2. **Midstream adaptation:** Good initial bitrate selection by itself is not sufficient as the network conditions may change dynamically during the playback. Thus, most players try to adapt the midstream bitrate as well. As such, a good throughput predictor is a necessary component of several prior proposals for bitrate adaptation algorithms (e.g., [30,45,47]). To confirm the importance of accurate throughput prediction, we replicate the analysis performed by Yin et al. [47] to study the impact of prediction error on the achieved QoE by Model Predictive Control (MPC) based bitrate adaptation mechanism. Figure 2 shows the variation of normalized QoE[3] with the increase of throughput prediction error. The result shows that when the error is $\leq$20%, the n-QoE of MPC is close to optimal (>85%). We also reconfirmed their results that in this regime the performance can be significantly better than pure Buffer-Based adaptation approach (i.e., BB in Figure 2). Other concurrent work has also confirmed this in the context of cellular networks [50].

Even though the above discussion and prior work make the case for throughput prediction, there has been little work on understanding how predictable throughput is in the wild or what types of prediction algorithms we need to use in the context of video bitrate adaptation. In the next section, we use a large-scale dataset to understand throughput predictability to inform the design of our CS2P approach.

# 3 Dataset and Analysis

In this section, we describe the dataset we use for analysis. We highlight the limitations of strawman solutions for initial and midstream throughput prediction and present key insights that lead to a better throughput prediction algorithm.

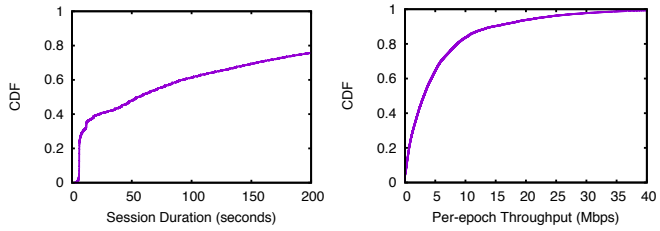
**Dataset:** To understand throughput variability across sessions and within a session, we need continuous measurements over sufficiently long session durations that contain enough repeated measurements of given client-server pairs. Note that this is in contrast to other kinds of end-to-end measurements of network latency, loss, jitter, or bottleneck capacity estimation (e.g., [19,25,33,49]). Unfortunately, there are few, if any, public datasets that enable such in-depth analysis of throughput stability and predictability at scale.[4]

To this end, we use a proprietary dataset of HTTP throughput measurement from the operational platform of iQIYI collected in September 2015. iQIYI is a leading online video content provider in China with a total monthly user base of more than 219 million. It ranks in the top-3 among the Chinese Internet video content providers in a series of key metrics such as daily/monthly active users, and viewing time. Our dataset comes from the operational CDN platform of iQIYI. The dataset consists of over 20 million sessions covering 3 million unique client IPs and 18 server IPs over 8 days in September 2015. The clients span 736 cities and 87 ISPs in China. In each session, a client set up an HTTP connection with one of the web servers and downloaded video chunks that had been encoded at a fixed bitrate (chosen by the user). Table 2 shows the basic features of the session and the coverage of our dataset. Within each ses-

[2]Other providers (e.g., YouTube) are anecdotally also observed to use information from the previous sessions of the same client.

[3]The normalized QoE (n-QoE) is defined as the actual QoE relative to the theoretical optimal, which could be achieved with the perfect knowledge of future throughput. Here, we adopt the same definition of video QoE as that in [47], and we formally define it in §7.1.

[4]We explored datasets such as Glasnost [21], MLab NDT [9] and one from a EU cellular provider [7]. Unfortunately, all of these have too few hosts and the sessions lasted only a handful of seconds making it unsuitable for such throughput stability and predictability analysis.




(a) Duration (b) Throughput

**Figure 3: CDF of session duration and throughput.**

| Feature | Description | # of unique values |
|---|---|---|
| ClientIP | Unique IP address associated to a client | 3.2M |
| ISP | ISP of client (e.g., AT&T) | 87 |
| AS | The Autonomous System that client resides in | 161 |
| Province | The province where the client is located | 33 |
| City | The city where the client is located | 736 |
| Server | The server-side identifier | 18 |

**Table 2: Summary of statistics from the dataset.**

sion, we recorded the average throughput for each 6-second period.[5] We refer to such a period as an "epoch".

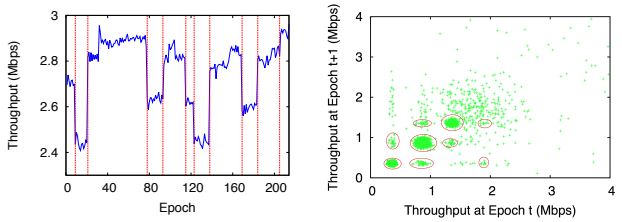
Figure 3a shows the CDF of the session duration and Figure 3b shows the distribution of the per-epoch average throughput and suggests that the average throughput distribution is similar to residential broadband characteristics [43]. The clients represent a wide spatial coverage of China. Although the number of servers is relatively small, the setting is very close to what real-world video delivery service providers face, i.e., the clients are widely distributed while the servers are relatively fewer.

Next, we use this dataset to characterize the structure of throughput variability within a given session and across sessions, and also evaluate the predictive power of some seemingly natural strawman solutions.

**Observation 1**: ***There is a significant amount of throughput variability within a video session, and simple predictive models (e.g., looking at recent epochs) do not work.***

We first investigate the throughput variability within a session. For instance, if the variability is small, then the adaptation logic does not have to switch bitrates often. To do so, we compute the coefficient of variation, which is defined as the ratio of the standard deviation ("stddev") of throughput across different measurements within the session to the mean of throughput measurements. The result shows that about half of the sessions have normalized stddev $\geq$30% and 20%+ of sessions have normalized stddev $\geq$50% (not shown). This confirms the general perception that the throughput has significant variation within a session, and therefore for video streaming, simple static bitrate selection will not suffice.

[5]For each 6-second epoch, the client counts the total incoming TCP segments and computes the average throughput. Then it records and reports the average throughput observed per epoch, after the session completes.




(a) An example session (b) Throughput variation at two consecutive epochs

**Figure 4: Stateful behaviors in session throughput.**

We tried a range of simple prediction models used in prior work [24, 30, 47] for predicting the throughput of the next epoch based on past observations in the session. These include: (1) Last-Sample (LS, using the observation of the last epoch), (2) Harmonic-Mean (HM, harmonic mean of past measurements), and (3) Auto-Regressive (AR, a classical timeseries modeling technique). We found that in general, these did not work satisfactorily with the median and 75%ile normalized prediction error across sessions respectively $\geq$18% and 40%.

**Observation 2**: ***The evolution of the throughput within a session exhibits stateful/persistent characteristics, which if captured can lead to improved prediction.***

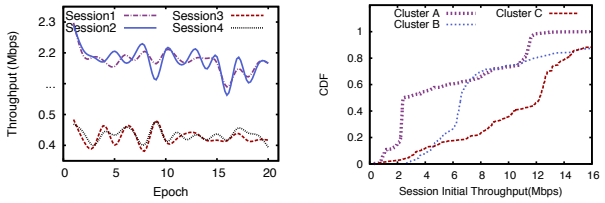
Figure 4a gives a visual example from our dataset. We can clearly observe some states within the throughput variation. We can split the timeseries into roughly 11 segments, and each segment belongs to one of the four states. Within each state the throughput is largely Gaussian, e.g., timeslots 20–75, 90–115, 135–175 and 180–210 belong to the same state with the mean throughput around 2.8Mbps.

We investigate the throughput variation across two consecutive epochs for a broader set of sessions and find similar stateful behaviors in these sessions. As an illustrative example, in Figure 4b we plot throughput at epoch $t+1$ (y-axis) vs. throughput at epoch $t$ (x-axis) of the sessions in our dataset with a particular IP/16 prefix. (We do not show the exact value of this prefix for proprietary reasons.) We can observe a clustered trend in the distribution of these points, i.e., there are some discrete states and the session throughput changes across these states (red circles in Figure 4b). In Section 5.2, we show that these states can be efficiently captured by a Hidden-Markov Model (HMM).

Given that we only had end-to-end measurements, we cannot conclusively pinpoint the root cause for such stateful behaviors. We can however intuitively conjecture that these patterns stem from the TCP fair-sharing model—the throughput depends on the hidden state, which is the number of flows currently sharing the bottleneck link, and the observed throughput changes as the number of concurrent flows changes during the session's lifetime.

**Observation 3**: ***Sessions with similar features tend to exhibit similar initial throughput conditions and throughput evolution patterns.***

Prior work (CFA [29]) shows that, at the application layer, video sessions with the same critical features have similar




(a) Example of similar sessions (b) CDF of initial throughput at different clusters

**Figure 5: Throughput similarity for sessions sharing the same key feature.**

QoE (e.g., rebuffering, startup latency, etc.). Here, we discover similar trends at the network layer, i.e., sessions sharing the same key set of features exhibit similarity in their throughput. Figure 5a gives an example from our dataset to illustrate this intuition. Sessions 1/2 and Sessions 3/4 are two pairs of "*close neighbors*", i.e., sharing a set of key session features. We can see that there is similarity in the throughput dynamics between the sessions in each of the pair.

Next, we categorize the sessions into different clusters according to Client IP prefix. Figure 5b shows the CDFs of initial throughput for 3 different clusters, each consisting of over 500 sessions. We have two key takeaways: (1) Sessions in different clusters have significant differences in initial throughput; (2) Within each cluster, a large number of sessions have similar initial throughput, e.g., 65% sessions in Cluster A have throughput around 2Mbps and 11Mbps, and over 40% of sessions in Cluster B with throughput 6Mbps. We did the same on midstream average throughput and found consistent results (not shown). Therefore, if we can identify the "similar sessions" with the same key features, we can use a cross-session prediction methodology to improve the accuracy. However, as we will show next this is a non-trivial task.

---

**Observation 4**: ***Simple models (e.g., last-mile characteristics) are not expressive enough to capture session similarity as there is significant diversity in session characteristics and the relationship between session features and throughput can be quite complex.***

---

An intuitive starting point to exploit the above observation of similarity across sessions is to look at the "last mile" characteristics (e.g., type of broadband connections). Thus, we tried two seemingly natural strawman solutions that consider last-mile predictors on both client and server side, i.e., predicting by sessions with the same client IP prefix or connecting to the same server. The results show that half of the sessions have the normalized prediction error $\geq$50%, and over 30% of the sessions with prediction error $\geq$80% (not shown).

More generally, we observe that the factors that can affect the throughput can be quite complex along two dimensions. First, combinations of multiple features often have a much greater impact on throughput than the individual feature. This can be intuitively explained as the throughput is often *simultaneously* affected by multiple factors (e.g., the last-mile connection, server load, backbone network congestion, etc.), which means sessions sharing same individual feature may not have similar throughput. Figure 6 gives an example of the effect of feature combinations. It shows that the throughput distribution of sessions with the same values on three key features (i.e., residing in the same *ISP-China Telecom* and the same *city-Hangzhou*, and fetching from the same *server-Server No.8*), and the throughput distribution of sessions only having same values on one or two of the three features. As shown in Figure 6, the throughput when all three features are specified is much more stable than any of other cases, meaning that for these sessions it is the combination of all the 3 features (not the subset) that determines their throughput. In practice, we find that such high-dimensional effects are the common case, rather than an anomalous corner case. For instance, 51% of distinct ISP-City-Server values have inter-session throughput standard deviation that is at least 10% lower than that of sessions only matching one of two features (not shown). Therefore, in order to capture "high dimensionality" effects, the prediction algorithm must be sufficiently expressive to combine multiple features rather than treating them individually.

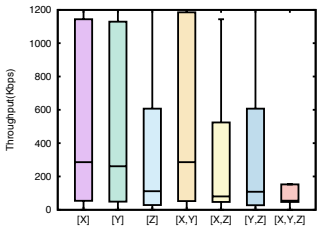



**Figure 6: The throughput variation of sessions matching all and a subset of three features: X=*ISP*, Y=*City*, Z=*Server*.**

Second, the impact of same feature on different sessions could be variable. For instance, the "last-mile connection" usually becomes the bottleneck for satellite communication links, while for broadband access it is less important to determine the throughput. We compute the relative information gain[6] of a feature on the throughput of session set to represent the impact of the feature on predicting their throughput, and find that the impact of the same feature (i.e., city) significantly varies for sessions in two different ISPs with the difference of relative information gain over 65% (not shown).

**Key observations:** In summary, our analysis of throughput variability suggests that:

- There is substantial throughput variability within a given session and a range of simple prediction models using previous observations in the same session do not provide high accuracy.
- Many sessions exhibit stateful characteristics in the evolution of the throughput.
- Sessions sharing similar critical characteristics tend to exhibit similar throughput patterns.

[6]Relative information gain is often used to quantify how useful a feature is for prediction, defined as $RIG(Y|X) = 1 - H(Y|X)/H(Y)$, where $H(Y)$ and $H(Y|X)$ are the entropy of $Y$ and the average conditional entropy of $Y$.

- The nature of the relationships between session features and throughput are quite complex and simple last-mile predictors are inaccurate.

# 4 CS2P Approach and Overview

In this section, we provide an overview of CS2P which leverages our earlier observations regarding throughput variation to improve bitrate selection and adaptation.

Figure 1 shows the basic workflow of CS2P. In the offline training stage, throughput measurements of sessions are collected by the ***Prediction Engine***. The Prediction Engine builds throughput prediction models based on the data collected. These models can then be plugged into the bitrate adaptation algorithms implemented either by the video servers or by clients.

Seen in a broader context, CS2P can be regarded as a middle ground between centralized video control platforms (e.g., C3 [23], CFA [29]) and decentralized player-based bitrate adaptation (e.g., BB [27], FESTIVE [30], MPC [47]). Specifically, CS2P borrows the benefits of the global view advocated by C3/CFA-like architectures to train the models. However, "actuation" using these models happens in a decentralized manner and without global coordination. As we will see in §6, these models are compact (<5KB) and can be easily plugged into the client- and server-side bitrate adaptation algorithms. While CS2P cannot offer all the benefits of centralized control (e.g., CDN switching), it offers a pragmatic alternative for video providers and CDNs, who do not want to relinquish control to third-party optimizers and/or do not want to incur the complexity of centralized control.

The key challenge is employing suitable prediction models that can capture the throughput variability observed in real-world sessions. As we saw in the previous discussion, simple models are not expressive enough to capture the structure of the throughput variation within an individual session and the diversity of the factors that can affect the throughput of a client-server combination.

At a high level, one can characterize how expressive a prediction model is in terms of the *spatial* and *temporal* structure it can capture. For instance, let us consider the initial bitrate prediction along the spatial dimension. At one end of the spectrum, we can use the previously observed throughput of the same client-server pair and at the other end of the spectrum we can simply use the global average of all the sessions. Obviously, neither is desirable; we may not have sufficient samples in the former case and cannot capture the diversity across sessions in the latter case. Similarly, let us consider the midstream bitrate prediction. If we only use the previous chunk throughput measurement from the same session, then we run the risk of having a noisy measurement which may additionally miss key state transitions. Besides, such simple time-series models miss the impact of critical spatial session features such as client location and ISP (**Observation 4** in §3).

CS2P adopts a *cross-session* (i.e., spatial) and *stateful* (i.e., temporal) prediction modeling approach that works as follows. First, based on **Observation 3** in §3, CS2P groups similar sessions sharing the same set of critical feature values and uses the data from such similar sessions to build the prediction models. Second, to capture the "state transitions" within a session (**Observation 2** in §3), CS2P learns a Hidden-Markov Model (HMM) for each cluster of similar sessions. HMM is an efficient state-based model and has been widely used to predict path and traffic properties [42, 44, 46].

Given this basic overview, there are three practical questions that remain:

1. How to cluster similar sessions?
2. How do we automatically train the models?
3. How to utilize these models for throughput prediction and bitrate adaptation?

We will address these questions next.

# 5 CS2P Detailed Design

In this section, we describe the detailed design of CS2P that addresses the above practical challenges. We begin by describing our data-driven clustering mechanism (§5.1). Then, we describe the HMM training and online prediction algorithms (§5.2). We conclude this section by describing how the initial throughput prediction and the HMM can be integrated into client- and server-side components (§5.3).

## 5.1 Identifying clusters

For both the initial and midstream throughput prediction, CS2P relies on clustering similar sessions with a cross-session prediction methodology. At a high level, CS2P finds for any session $s$ a key feature set and time range, which is used to aggregate previous sessions that match the specific features with $s$ and happened in the specific time range.

The workflow of session clustering algorithm in CS2P to find the session features yielding the best prediction is as follows:

1. Pick a given set of features $M_s$ from all possible feature combinations (i.e., $2^n$ subsets of $n$ features, the candidate session features are shown in Table 2) and time windows. Specifically, the possible time windows include time windows of certain history length (i.e., last 5, 10, 30 minutes to 10 hours) and those of same time of day (i.e., same hour of day in the last 1-7 days).
2. Once the set of features $M_s$ is picked for $s$, CS2P aggregates previous sessions based on $M_s$. For instance, given $M_s = \langle ISP, 1hr \rangle$, CS2P will aggregate all previous sessions who are in the same ISP as $s$ and happened in the last 1 hour. Let this set of previous sessions be denoted by $Agg(M_s, s)$.
3. CS2P predicts the throughput of $s$ by $Pred(s) = F(Agg(M_s, s))$, where $F(S)$ is the predicted throughput by using the sessions in $S$. The prediction algorithm $F$ will be shown in §5.2.

The goal is to minimize the absolute normalized prediction error,

$$Err(Pred(s), s_w) = \frac{|Pred(s) - s_w|}{s_w}, \qquad (1)$$

where $s_w$ is the actual throughput of $s$.

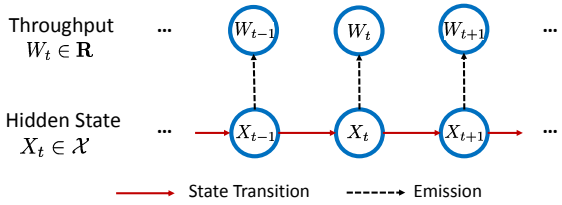



**Figure 7: Overview of HMM model.**

The key component of algorithm is how to map each session $s$ to a set of features $M_s^*$, that yields the lowest prediction error. That is

$$M_s^* = \arg\min_M Err(F(Agg(M, s)), s_w) \tag{2}$$

We take a *data-driven* approach and find the best set of features for prediction over a set of previous sessions $Est(s)$ (defined shortly). Formally, the process can be written as following:

$$M_s^* = \arg\min_M \frac{1}{|Est(s)|} \sum_{s' \in Est(s)} Err(F(Agg(M, s')), s'_w) \tag{3}$$

$Est(s)$ should include sessions that are likely to share the best prediction model with $s$. In our dataset, $Est(s)$ consists of sessions that match features in Table 2 with $s$ and happened within 2 hours before $s$ occurred.

To make the prediction $Pred(s)$ reliable, CS2P ensures that $Pred(s)$ is based on a substantial number of sessions in $Agg(M_s, s)$. Therefore, if $M_s$ yields $Agg(M_s, s)$ with less than a threshold number of sessions (e.g., 100), it will remove that session cluster from consideration. Note that the model can regress to the "global" model (i.e., model trained with all the previous sessions), if no suitable clustering could be achieved.[7]

Note that this session clustering step is similar to that in CFA [29], but the goal and criteria of clustering in the two schemes are different. CFA determines the critical feature set according to the QoE similarity, whereas in CS2P the optimal session clusters are chosen based on throughput prediction accuracy.

## 5.2 HMM training and online prediction

Next we present a simple but effective HMM-based predictor capturing the state-transition behaviour (**Observation 2** in §3)) in each cluster $Agg(M_s^*, s)$.

**Modeling:** The throughput predictor in CS2P is based on a *Hidden Markov Model (HMM)*. Figure 7 provides a high-level overview of the HMM. The intuition behind the use of HMM in our context is that the throughput depends on the hidden state; e.g., the number of flows sharing the bottleneck link and link capacity. By carefully analyzing the behaviors of previous sessions with the same value of features in $M_s^*$, we try to capture the state transitions and the dependency between the throughput vs. the hidden state, and propose a robust and efficient throughput predictor.

[7]The probability of sessions using global model in our dataset is $\leq$4%.

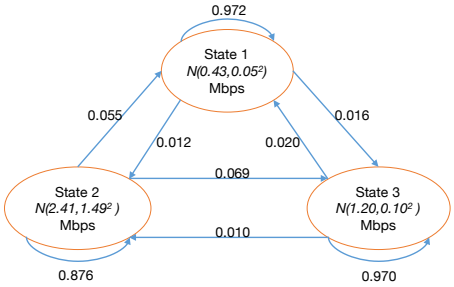



**Figure 8: Example of hidden-markov model of session clusters.**

We start by formally defining the HMM. Let $W_t$ be the random variable representing the network throughput at epoch $t$, $w_t$ be the actual throughput measured from the network, $\hat{W}_t$ be the predicted value of $W_t$.

We assume the throughput $W_t$ evolves according to some hidden state variables $X_t \in \mathcal{X}$, where $\mathcal{X} = \{x_1, \cdots, x_N\}$ denotes the set of possible discrete states and $N = |\mathcal{X}|$ the number of states. Intuitively, the states reflect some discrete changes in the structure of the network or users, e.g., number of users at a bottleneck link. Given that state $X_t$ is a random variable, we denote its probability distribution as a vector $\pi_t = (\mathbb{P}(X_t = x_1), \cdots, \mathbb{P}(X_t = x_N))$.

The key assumption in HMM is that the state evolves as a Markov process where the probability distribution of the current state only depends on the state of the previous epoch, i.e., $\mathbb{P}(X_t|X_{t-1}, \cdots, X_1) = \mathbb{P}(X_t|X_{t-1})$. We denote the transition probability matrix by $P = \{P_{ij}\}$, where $P_{ij} = \mathbb{P}(X_t = x_i|X_{t-1} = x_j)$. According to Markov property,

$$\pi_{t+\tau} = \pi_t P^\tau \tag{4}$$

Given the hidden state $X_t$, we assume the pdf of throughput $W_t$ (namely, the emission pdf) is Gaussian:

$$W_t|X_t = x \sim N(\mu_x, \sigma_x^2) \tag{5}$$

Note that HMM is a general model which could work with any emission pdf other than Gaussian. However, here we use Gaussian emission as it proves to provide high prediction accuracy in our dataset and its computational simplicity.

Figure 8 gives an example of a 3-state HMM of one session cluster in our dataset. Each state follows a Gaussian distribution of throughput denoted by the mean of the distribution and its standard deviation $N(\mu, \sigma^2)$. The transition probability is computed between every pair of states. For instance, suppose session throughput is currently at State 1 in Figure 8, then for the next epoch it will stay at the same state with probability of 97.2% and switch to State 2 and 3 respectively with probabilities of 1.2% and 1.6%. Note in Figure 8, the probability of inter-state transition and the standard deviation of throughput within each state are small, suggesting clear stateful behaviors for the throughput evolution.

We introduce notations before proceeding to training and prediction: For simplicity, we use $W_{1:t} = \{W_1, \cdots, W_t\}$ to denote throughput from epoch 1 to epoch $t$. Let $\pi_{t_1|1:t_0} = (\mathbb{P}(X_{t_1} = x_1|W_{1:t_0}), \cdots, \mathbb{P}(X_{t_1} = x_N|W_{1:t_0}))$ be the pdf vector of the hidden state $X_{t_1}$, given throughput from epoch

1 to $t_0$. For example, $\pi_{t|1:t-1}$ is the pdf of state $X_t$ given the throughput up to epoch $t-1$.

**Offline training:** Given the number of states $N$, we can use training data in $Agg(M_s^*, s)$ to learn the parameters of HMM for this particular cluster, $\theta_{HMM} = \{\pi_0, P, \{(\mu_x, \sigma_x^2), x \in \mathcal{X}\}\}$ via the expectation-maximization (EM) algorithm [18]. Note that the number of states $N$ needs to be specified. There is a tradeoff here in choosing suitable $N$. Smaller $N$ yields simpler models, but may be inadequate to represent the space of possible behaviors. On the other hand, a large $N$ leads to more complex model with more parameters, but may in turn lead to overfitting issues. As described in §7.1, we use cross-validation to learn this critical parameter.

**Algorithm 1** Online prediction in CS2P

```
Let t be epoch id
for t = 1 to T do
    if t = 1 (initial epoch) then
        Initialize π_1
        Ŵ_1 = Median(Agg(M_s^*, s))
    else
        π_{t|1:t-1} = π_{t-1|1:t-1} P
        Ŵ_t = μ_x, where x = arg max_{x∈X} π_{t|1:t-1}(x)
    end if
    Bitrate selection based on prediction Ŵ_t
    Obtain throughput measurement w_t
    Update π_{t|1:t} = (π_{t|1:t-1} ∘ e(w_t)) / ((π_{t|1:t-1} ∘ e(w_t)) · 1)
end for
```

**Online prediction:** In the offline training stage, we find the 1) set of critical features and 2) corresponding prediction model for each session in the training dataset. In the online prediction stage, a new session is mapped to the most similar session in the training dataset, which matches all (or most of) the features with the session under prediction. We then use the corresponding HMM of that session to make predictions. The online prediction algorithm using HMM is shown in Algorithm 1. At a high level, it involves predicting throughput for the next epoch using HMM, as well as updating HMM state once the actual throughput is measured.

Next, we discuss the key steps in our prediction approach:

- *Prediction (initial epoch)*: HMM relies on the throughput measurement of the "current" epoch to predict throughput of the "next" epoch, however for the initial epoch there is no historical information in this session. As such, CS2P predicts the initial throughput of session $s$ simply by the *median* throughput of sessions in $Agg(M_s^*, s)$ that match $s$ on the best set of features of $M_s^*$ and are in the time range of $M_s^*$, i.e.,

$$\hat{W}_1 = Median(Agg(M_s^*, s)) \tag{6}$$

  Note that the throughput prediction of the initial epoch is computed in the Prediction Engine and sent to the video servers (for server-side bitrate adaptation) or clients (for client-side bitrate adaptation) together with the trained prediction models.

- *Prediction (midstream epoch)*: At epoch $t$, given updated pdf of HMM state $\pi_{t-1|1:t-1}$, we can compute the state pdf at current epoch according to Markov property:

$$\pi_{t|1:t-1} = \pi_{t-1|1:t-1} P \tag{7}$$

  The throughput prediction $\hat{W}_t$ is given by the maximum likelihood estimate (MLE):

$$\hat{W}_t = \mu_x, \quad x = arg\max_{x \in \mathcal{X}} \mathbb{P}(X_t = x | W_{1:t-1}) \tag{8}$$

- *Update HMM*: Once we observe the actual throughput $w_t$, we use this information to update the state of HMM $\pi_t$, so that it reflects the most up-to-date information of the network. Namely, given actual throughput $W_t = w_t$, and $\pi_{t|1:t-1}$, we want to compute $\pi_{t|1:t}$ using the following equations:

$$\pi_{t|1:t} = \frac{\pi_{t|1:t-1} \circ e(w_t)}{(\pi_{t|1:t-1} \circ e(w_t)) \cdot \mathbf{1}} \tag{9}$$

  where $e(w_t) = (f(w_t|X_t = x_1), \cdots, f(w_t|X_t = x_M))$ is the emission probability vector, $f(\cdot)$ is the Gaussian pdf, $\circ$ denotes entry-wise multiplication, or Hadamard product [6] of the two vectors.

## 5.3 Player integration

CS2P can be used both with server-side [14, 20] and client-side adaptation solutions [30, 47].

In the server-side solution, video content servers interact with the Prediction Engine for the models and initial throughput predictions for each cluster, and are responsible of choosing the bitrate for all the sessions. The advantage of this server-based solution is that it requires little updates or modifications on the clients. However, the centralized server needs to collect throughput measurements from all clients and compute bitrates for each video session, making it a potential bottleneck. Fortunately, we find that the online prediction in CS2P is very light-weight and our deployed server (Intel i7-2.2GHz, 16GB RAM, Mac OS X 10.11) can process about 150 predictions per second.

Bitrate adaptation can also be done by each video client. Here, each video client downloads its own HMM and initial throughput prediction from Prediction Engine and runs the model for real-time throughput prediction and bitrate adaptation by itself. The advantage of this decentralized method is that the client is often in the best position to quickly detect performance issues and respond to dynamics. The disadvantage is that it requires client to maintain its own HMM. Fortunately, the computation complexity and storage requirement of HMM in CS2P are low, and it is feasible to do that on the client. On our test client (Intel i7-2.8GHz, 8GB RAM, Mac OS X 10.9), each prediction requires <10 milliseconds (only needs two matrix multiplication operations), and <5KB memory is used to keep the HMM.

For midstream bitrate selection, we use the Model Predictive Control (MPC) strategy formulated by recent efforts [47],[8] that takes *throughput prediction, current bitrate*

[8]Specifically, we refer to FastMPC [47].

and *buffer occupancy* as inputs and solves an exact integer programming problem to decide the bitrate for the next few epochs. For brevity, we do not provide more details of MPC and its advantages over pure Rate-based (RB) or Buffer-based (BB) schemes, and refer readers to prior work [47]. However, MPC cannot be utilized for the initial bitrate selection of the session due to the lack of the current bitrate setting and buffer occupancy measurement. Thus, to select bitrate for the first chunk, we simply select the highest sustainable bitrate below the predicted initial throughput.

# 6 Implementation

In this section, we describe our implementation. Our implementation follows the server-side solution of CS2P, i.e., the server makes throughput prediction for each session. We integrate the functionalities of Prediction Engine into the video server, which is responsible of training the HMMs for each session clusters and then using the trained models to make throughput predictions for the video players. On receiving the throughput prediction from the server, video player runs the bitrate selection algorithms to achieve bitrate adaptation.

- **Video Player**: Our implementation of video player is based on Dash.js, an open-source implementation of MPEG-DASH standard using client-side JavaScript to present a flexible and potentially browser independent DASH player [3]. The key components of Dash.js controlling bitrate selection are BufferController and AbrController. We make several minor modifications to these two components. First, in BufferController, bitrate decision is made before the request of each video chunk (including the initial chunk). Whenever the client wants to make bitrate decision, it sends a POST request (containing the actually throughput of the last epoch) to the server and fetches the result of throughput prediction in approximate 500 milliseconds. Second, we implement different bitrate algorithms (e.g., MPC, RB, BB, fixed) in AbrController, replacing the default rule-based decisions. When the video is completely loaded, log information including QoE, bitrates, rebuffer time, startup delay, predicted/actual throughput and bitrate adaptation strategy is sent to a log server.
- **Server**: On the server side, we choose the Node.js as the basis of HTTP server implementation. Node.js is an event-driven, non-blocking I/O, lightweight and efficient network framework [11]. We implement the key functions such as session clustering, HMM model building, online initial/midstream throughput prediction, on the server. The learning of the HMM model is implemented using the Probabilistic Modeling Toolkit (PMTK) [35] in Octave. As model training is a time-consuming process, we do it on a per-day basis with the log collected on each day.[9] The server responds to the POST requests from video clients and returns the throughput prediction results.

[9]Since the model learning for different clusters are independent, this process can be easily parallelized.

We believe that our implementation can be easily translated to the client-side solution (i.e., each client makes throughput prediction by itself), as we only require less than 600 additional lines of JavaScript over open-source players [3, 47].

# 7 Evaluation

In this section, we show that:

- CS2P reduces median prediction error by 40% for initial throughput and 50% for midstream throughput comparing to state-of-art predictors, achieving 6% 1-epoch-ahead and < 10% 10-epoch-ahead median prediction error (§7.2);
- When combined with MPC [47], CS2P can drive median overall QoE to 93% of offline optimal for initial chunk and 95% for midstream chunks, outperforming other state-of-art predictors (§7.3);
- In pilot deployments, CS2P combined with MPC-based bitrate adaptation algorithms, outperforms the state-of-art HM + MPC strategy, achieving 3.2% improvement on overall QoE and 10.9% higher average bitrate and can accurately predict the total rebuffering time at the beginning of the session (§7.5).

We also evaluate the sensitivity of CS2P performance to various configuration parameters (§7.4).

## 7.1 Evaluation Setup

**Evaluation Framework:** We use a combination of real player experiments and trace-driven simulations for evaluation. We use the real video player to conduct a pilot deployment (§7.5). For improvement on accuracy and QoE (§7.2, §7.3) and sensitivity analysis (§7.4), we employ a custom Matlab-based simulator simulating the video download and playback process and the buffer dynamics. In the simulation framework, the throughput changes according to the previously recorded traces. The simulated player measures the throughput, and different algorithms (e.g., HMM) are used to predict future throughput accordingly.

**Baseline solutions:** While it is impossible to enumerate all possible prediction algorithms, we consider several representative history-based and machine-learning models used in recent proposals [24, 47]:

1. *History-based predictors*: LS (Last Sample), HM (Harmonic Mean [30, 47]) and AR (Auto Regression [24]).
2. *Machine-learning predictors*: SVR (Support Vector Regression [34]) and GBR (Gradient Boosting Regression trees [41]).

**Model configuration:** To learn the parameters of prediction models, we divide the dataset into temporally non-overlapping training and testing datasets. We learn parameters using training dataset with data in the first day and report prediction results on the testing dataset for sessions in the next day.

To choose key design parameters (number of HMM states, group size, etc.), we adopt 4-fold *cross validation*. Specifically, we equally divide the *training* dataset on the first day into 4 subsets, and iteratively use 3 of the subsets for model

learning with candidate parameters and choose the parameter with best testing result on the remaining subset. Note that the dataset on the first day is only used in the training process and is not used in evaluations.

For AR and HM, we utilize all the available previous measurements to predict next value. For SVR and GBR, we use the implementations in [39]. The training of GBR and SVR are using all the sessions[10] in our dataset with the same session feature set as we list in Table 2. We use a 6-state HMM with group size 100 based on cross validation.

One limitation we acknowledge is that the training and testing throughput data is from fixed bitrate video chunk downloading instead of ABR chunk downloading. However, based on our conversation with iQIYI engineers we confirm that the downloading behaviors (especially the throughput) are quite similar for video chunks in different bitrate levels. Therefore, we believe that the conclusions we make in the evaluation still hold for ABR scenarios.

**Metrics:** We use the following performance metrics for prediction accuracy and QoE:

1. *Absolute normalized prediction error (Eq1)*: We can summarize the error within and across sessions in different ways, e.g., median of per-session median, 90-percentile of per-session median, or median of 90-percentile per-session.
2. *QoE*: We adopt the model in prior work [47], where QoE is a linear combination of average video quality, average quality variation, total rebuffer time, and startup delay. We set $\lambda = 1$, $\mu = \mu_s = 3000$.[11]
3. *Normalized QoE (n-QoE)*: Computed by dividing actual QoE by the *offline optimal QoE*, which is achieved given perfect throughput information in the entire future and can be calculated by solving a MILP problem.
4. *QoE components*: Finally, we adopt two basic QoE metrics–(a) *AvgBitrate*: average value of selected bitrates, and (b) *GoodRatio*: percentage of chunks with no re-buffering.

**Video parameters:** We use the same video as in prior work [47], i.e., the "Envivio" video from DASH-264 JavaScript reference client test page [2]. The video length is 260s, and the chuck size is equal to the epoch length. The video is encoded in H.264/MPEG-4 AVC codec in the following bitrate levels: 350kbps, 600kbps, 1000kbps, 2000kbps, 3000kbps, matching the bitrate levels for YouTube [13]. The buffer size is $30s$.

## 7.2 Improvement in Prediction Accuracy

First, we present the improvement in prediction accuracy of CS2P compared with the baseline approaches.

**Midstream epoch:** Figure 9b shows the CDF of absolute normalized prediction error for midstream epochs. CS2P reduces the median prediction error by $\sim$50% comparing to other baseline solutions, achieving $\sim$7% median error and $\sim$20% 75-percentile error. CS2P also reduces the tail of the prediction error; e.g., at the 75th percentile, CS2P's error is less than 20%, compared with more than 30% for all other methods. In addition, we compared CS2P with Global Hidden-Markov-Model (GHM), a global HMM trained by data of all previous sessions *without clustering*. The result shows that the prediction accuracy of CS2P outperforms GHM, which confirms the necessity of training a separate HMM for each cluster of similar sessions rather than having a global HMM for all sessions.

**Initial epoch:** Figure 9a depicts the CDF of prediction error for the first (initial) epoch of the session.[12] We see that CS2P performs much better in predicting the initial throughput with $\leq$20% median error vs. 35%+ for other predictors. We also observe that the prediction error for initial epoch (Figure 9a) is in general higher than midstream epochs (Figure 9b). This is due to the lack of throughput measurements in previous epochs of the session, therefore we can only utilize cross-session information (i.e., similar sessions in history) for prediction.

As such, the prediction accuracy depends on how many features are available and what they are. We carry out the same experiment on another dataset from FCC MBA project [4], where more features are available for each session (e.g., connection technology, downlink/uplink speed). We found that the accuracy in FCC dataset is significantly better as the median error for initial epoch is $\leq$10% (not shown).[13]

**Impact of look-ahead horizon:** We also study the accuracy of prediction for longer horizons into the future (instead of just the next epoch). This can be critical in many scenarios: e.g., some bitrate adaptation algorithms (e.g., MPC) require prediction into a fixed look-ahead horizon. Similarly, CDN server scheduling also benefits greatly from predicting the overall video downloading time early in the process. Figure 9c shows the median of the per-session median prediction error vs. the number of lookahead epochs. We see that CS2P clearly outperforms other predictors, achieving $\sim$50% improvement over the second best (GBR). When predicting 10 epochs ahead, CS2P can still achieve as low as 9% prediction error while all other solutions have the error $\geq$17%. We also considered other performance metrics (e.g., average of per-session average error) and found consistent results

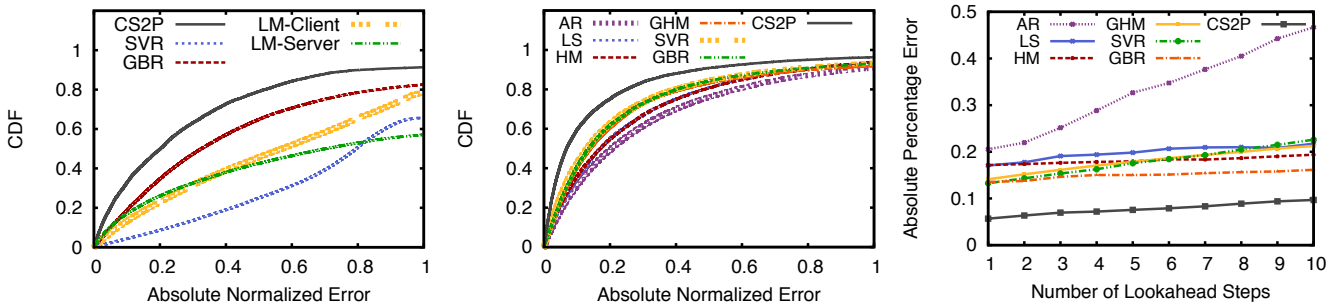
[10]We also tried training only on the clustered sessions, but found the results were worse than training with all sessions.

[11]The exact QoE equation is as follows [47]

$$QoE_1^K = \sum_{k=1}^{K} R_k - \lambda \sum_{k=1}^{K-1} |R_{k+1} - R_k| - \mu \sum_{k=1}^{K} \left( \frac{d_k(R_k)}{C_k} - B_k \right)_+ - \mu_s T_s$$

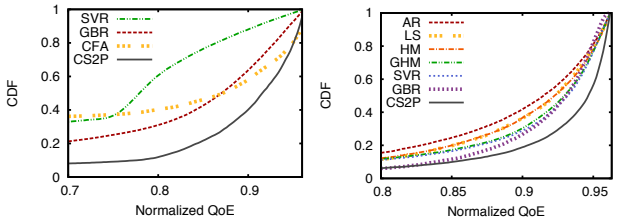
[12]Since AR, HM and LS can not be used for the initial throughput prediction, here we only compare the performance of CS2P with GBR, SVR, LM-client (Last Mile-client, predicting by the performance of clients sharing the same IP prefix/16), and LM-server (Last Mile-server, predicting by the performance of clients connecting to the same server).

[13]Unfortunately, this dataset cannot be used to test intra-session midstream throughput variation, since the fixed short duration (30 seconds) of each session does not provide enough measurement samples.




(a) Initial epoch (b) Midstream epoch (c) 10-epoch lookahead

**Figure 9: Analyzing throughput prediction accuracy of different solutions.**




(a) Initial chunk (b) Midstream chunk

**Figure 10: Video QoE improvement using different throughput predictors.**

| | Initial | | Midstream | |
|---|---|---|---|---|
| | **AvgBitrate** | **GoodRatio** | **AvgBitrate** | **GoodRatio** |
| AR | NULL | NULL | 3.31Mbps | 96.6% |
| LS | NULL | NULL | 4.08Mbps | 93.2% |
| HM | NULL | NULL | 3.80Mbps | 97.2% |
| CFA | 1.93Mbps | 87.9% | NULL | NULL |
| SVR | 1.52Mbps | 81.4% | 4.64Mbps | 92.6% |
| GBR | 2.09Mbps | 93.8% | 4.28Mbps | 98.0% |
| **CS2P** | **4.27Mbps** | **98.5%** | **4.97Mbps** | **99.1%** |

**Table 3: Comparing AvgBitrate vs. GoodRatio among different predictors.**

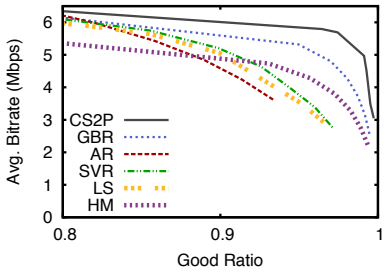



**Figure 11: Tradeoff between AvgBitrate and GoodRatio.**

that CS2P significantly outperforms the baseline models (not shown).

## 7.3 Improvement in Video QoE

Next, we evaluate the QoE improvement using CS2P.

**Overall QoE:** Figure 10 shows the CDF of normalized QoE of different predictors + MPC adaptation algorithm-fig:hmmstatefig:hmmstate [47] for both initial and midstream chunks. As can be seen, CS2P achieves better QoE for both initial and midsteam epochs. For CS2P 61% and 81% of the sessions achieve >90% of the offline-optimal QoE respectively for initial and midstream chunks, while these numbers for the next best solutions are only 42% and 73%. This result confirms that the improved prediction accuracy of CS2P leads to concrete QoE gain when combined with prediction-based bitrate adaptation algorithms.

In Figure 10a we also compare CS2P against CFA [29], which selects the initial video bitrate based on the QoE prediction via cross-session methodology. We see that CS2P significantly outperforms CFA. The reason is that CFA relies on QoE prediction, and QoE heavily depends on video-specific features (e.g., videos with different bitrate levels have different QoE). In our dataset we do not record these video-specific features, making it difficult to predict QoE accurately. However, CS2P relies on throughput prediction using only network-specific features, and our dataset enables it to have good enough predictions.

**Detailed QoE:** Next, we zoom in and focus on two key QoE factors, AvgBitrate and GoodRatio. As shown in Table 3, CS2P leads to *both* higher AvgBitrate and GoodRatio for initial and midstream chunks.

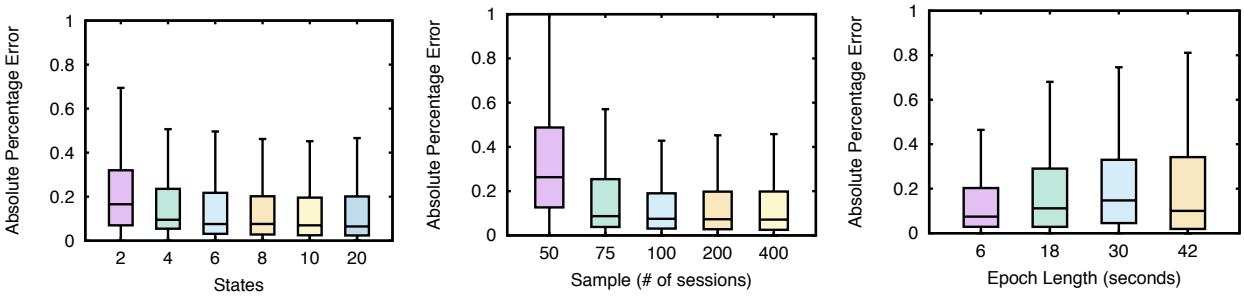
Figure 11 shows the Pareto frontier of QoE factors for midstream chunks achieved by MPC + different predictors, i.e., the set of achievable AvgBitrate and GoodRatio by adjusting weight on QoE factors. The more to the top right, the better QoE is achieved. We observe that CS2P-based bitrate selection strikes a better tradeoff of higher AvgBitrate and higher GoodRatio. Overall, CS2P + MPC achieves better QoE than other predictors, once again confirming the claim that higher prediction accuracy leads to QoE improvement [48].

## 7.4 Sensitivity Analysis

We also conduct sensitivity analysis of the performance of CS2P w.r.t. key design parameters.

**HMM states:** While a sufficient number of states is necessary to fully capture the behavior of the network, having too many states leads to increased model complexity and potential overfitting. Figure 12a shows the prediction error vs. number of HMM states. We see that while the error decreases with more states, there is a natural diminishing return property as the performance gain after 6 states is much less. This confirms our choice of 6-state HMM based on cross validation.

**Group size:** As discussed in §5.1, if the number of training sessions in a cluster is too small, the data will be insufficient




(a) Error vs. HMM states (b) Error vs. Group size (c) Error vs. Measurement frequency

**Figure 12: Sensitivity analysis of CS2P parameters.**

to yield reliable prediction results. Figure 12b shows the error vs. the threshold of group size in the training dataset. We observe that while the error decreases with more training sessions, the prediction error converges after the group size grows to 100. Again, this confirms our choice of group size 100 using cross validation.

**Measurement granularity:** We also investigate how performance of CS2P changes w.r.t. throughput measurement granularity. We merge the original per-6-second traces to more coarse-grained traces (18s, 30s, 42s) by taking the average of multiple consecutive epochs. Figure 12c shows that the prediction error is generally independent of measurement granularity.

## 7.5 Pilot Deployment

Finally, we conduct two deployment studies to evaluate the performance of CS2P in the wild.

**Custom multi-city deployment:** First, we took two rounds of 4-day experiments to respectively compare the performance of CS2P + MPC vs. BB [27] (during January 11 to 14, 2016) and CS2P + MPC vs. HM[14] + MPC [47] (during May 16 to 19, 2016). In each round, *200+ client* video players were involved from *5 university campuses across 4 different cities* in China,[15] connecting to a centralized server deployed in our lab. We use the data collected on the previous day to train our model, and apply the model for throughput prediction and bitrate adaptation for the current day. When a new client session starts, it randomly selects one of the two bitrate adaptation strategies (CS2P +MPC and HM+MPC/BB) with equal probability. The video clients are Google Chrome web browsers for Linux, Mac OS X and Windows with V8 JavaScript engine while the video server is a simple HTTP server based on Node.js (version 0.10.32).

Table 4 shows that our CS2P + MPC significantly outperforms BB in a variety of QoE metrics except startup delay, i.e., increasing the average bitrate by 9.3%, reducing midstream bitrate switches by 5.6% and improving GoodRatio by 17.6%. The overall QoE improvement is 14% relative to BB. In addition, we find that CS2P + MPC outperforms HM + MPC in terms of all QoE metrics except for bitrate variability, i.e., improving the average bitrate, GoodRatio and startup delay respectively by 10.9%, 2.5% and 0.4%, resulting an overall QoE improvement by 3.2%. Note that due to the limited number of clients in our experiment, the accuracy of the cross-session prediction in CS2P is lower than the trace-driven simulations. We believe that this is a conservative estimate of the QoE gain, and the improvement of CS2P vs. HM is likely more significant in large scale scenarios with more client-side measurements.

[14]Since Harmonic Mean (HM) can not be used to predict the initial throughput, for fair comparison we use the same predicted throughput at initial epoch in HM as that in CS2P.

[15]Due to the limited number of volunteer clients involved, we cannot test the three strategies simultaneously.

| Metrics | vs. HM+MPC | vs. BB |
|---|---|---|
| **AvgBitrate** | 10.9% | 9.3% |
| **GoodRatio** | 2.5% | 17.6% |
| **Bitrate Variability** | -2.3% | 5.6% |
| **Startup Delay** | 0.4% | -3.0% |
| **Overall QoE** | **3.2%** | **14.0%** |

**Table 4: QoE improvement by CS2P +MPC compared with HM+MPC and BB in a real-world experiment in 4 cities of China.**

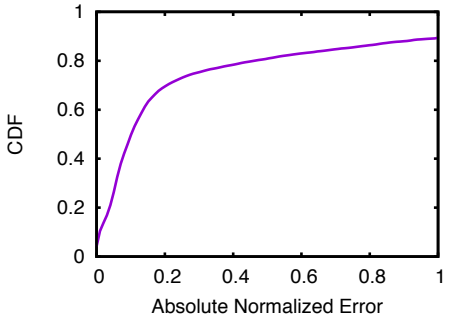



**Figure 13: Prediction error on total rebuffering time.**

**Deployment in a large commercial VoD service:** We also deployed CS2P in the VoD system of iQIYI, where CS2P is used to estimate the total rebuffering time at the beginning of fixed-bitrate streaming sessions. Specifically, at the beginning of the session, we use CS2P to predict throughput $T_i$ for all future epochs $i$, and calculate the total downloading time of the video $DT_p$, given size of each chunk. The predicted rebuffering time is $RT_p = \max\{0, DT_p - X\}$, where $X$ is the video length. We compare the $RT_p$ with the actual measured rebuffering time $RT_m$. We then focus on the sessions with rebuffering events ($RT_m > 0$). Figure 13 shows that the predicted rebuffering time is close to the actual value, with 70%+ sessions achieving prediction

error $\leq$20%. Our discussions with the iQIYI engineers suggest that these preliminary results are very promising as a means of informing their bitrate and CDN server selection logic and there are ongoing plans for a more comprehensive evaluation of CS2P in their production system.

# 8 Related Work

**Path properties measurement:** Studies on path properties have shown prevalence and persistence of network bottlenecks (e.g., [25]), constancy of various network metrics (e.g., [49], longitudinal patterns of cellular performance (e.g., [36]), intra-session RTT variation (e.g., [37]), and spatial similarity of network performance (e.g., [17]). In contrast, our focus is on throughput stability and predictability.

**Bandwidth measurement:** Unlike prior path mapping efforts (e.g., [19,33,40]), CS2P uses a data-driven model based on available session features and does not require any inference of path information (e.g., traceroute). Other approaches use packet-level probing to estimate the available bandwidth and the capacity of Internet paths (e.g., [12, 26]). Unlike CS2P, these active probes need full client/server-side control which is often infeasible in the wild.

**Throughput prediction:** Prior work either developed approximate analytical models of TCP throughput as a function of packet loss and delay [24, 38] or leveraged Time-series models (e.g., Holt-Winters [32] and Auto-Regressive [24]) and machine-learning models (e.g., Support Vector Regression [34], Gradient Boosting Regression Trees [41]) to predict session's throughput based on previous measurements. However, these approaches do not provide satisfactory prediction accuracy to feed into the video adaptation algorithms.

**Video QoE prediction:** Jiang et al., observe that video quality is typically determined by a subset of critical features, and thus propose CFA [29] to predict video QoE of a new session based on the QoE measurements of similar sessions in history. CS2P is inspired by similar insight that end-to-end performance is predictable because it is determined by only a few critical features. While CFA and CS2P are complementary, there are some important differences between them: 1) CFA only predicts application-layer quality using both video-specific and network-specific features, whereas CS2P predicts network-layer throughput using only network-specific features; 2) CFA only considers initial bitrates and does not do midstream throughput model; 3) CFA envisions a deployment model of centralized control while CS2P is amenable of a decentralized execution.

**Adaptive video streaming:** Our work is in the context of Dynamic Adaptive Streaming over HTTP (DASH), where it is known that choosing high and sustainable bitrate is critical to video quality of experience [16]. Prior work implicitly assumes that throughput is unstable and unpredictable, and eschews this in favor of using the player buffer occupancy for controlling bitrates [27]. Recent work [47, 50] argues that adaptive video streaming can significantly benefit from accurate throughput prediction. However, these do not provide a concrete prediction algorithm. Our contribution is in developing an effective throughput predictor and demonstrating its utility for DASH.

# 9 Conclusions

Designing good bitrate selection and adaptation algorithms is critical to deliver good video quality of experience (QoE). Prior work argues that accurate throughput prediction could help improve initial bitrate selection and the midstream adaptation [45, 47, 50], but fails to provide a concrete roadmap to achieve these benefits. Our work bridges this gap by providing a large-scale measurement analysis of throughput variability and builds on these data-driven insights to develop the CS2P framework. CS2P uses cross-session stateful prediction models. These models can be easily plugged into the bitrate selection logic of client- and server-side adaptation algorithms. Thus, CS2P offers an immediately deployable middle ground between complex centralized control architectures [23] and purely decentralized adaptation algorithms [27, 47]. We demonstrate the benefits of CS2P using both trace-driven simulations and pilot deployments and find that CS2P outperforms prior work on both throughput prediction accuracy and video QoE.

# Acknowledgments

The authors would like to thank Menggang Tan, Jia Wang, Ling Cai, Yongqiang Dong and Jing Liu for helping us deploy the multi-city experiments and all the volunteers joining the experiment. We also thank Keith Winstein for shepherding our paper and SIGCOMM reviewers for their feedback. This work is supported in part by the National Basic Research Program (2012CB315802) and the Natural Science Foundation of China (61379133, 61133015). This work is also funded in part by NSF (CNS-1345305) and a Juniper Networks Fellowship.

# 10 References

[1] Cisco Visual Networking Index. http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html.
[2] DASH-264 JavaScript reference client landing page 1.4.0. http://dashif.org/reference/players/javascript/1.4.0/samples/dash-if-reference-player/index.html.
[3] Dash.js. https://github.com/Dash-Industry-Forum/dash.js/wiki.
[4] FCC Measuring Broadband America . http://www.fcc.gov/measuring-broadband-america.
[5] Final Report on the Validation of Objective Models of Video Quality Assessment. http://videoclarity.com/PDF/COM-80E_final_report.pdf.
[6] Hadamard Product. https://en.wikipedia.org/wiki/Hadamard_product_(matrices).
[7] HSDPA. http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/.
[8] iQIYI. http://www.iqiyi.com.
[9] MLab NDT. https://console.cloud.google.com/storage/browser/m-lab/ndt/.
[10] Netflix. http://www.netflix.com.
[11] Node.js. https://nodejs.org/en/.
[12] Pathchar. http://www.caida.org/tools/utilities/others/pathchar/.

[13] YouTube live encoder settings, bitrates and resolutions. https://support.google.com/youtube/answer/2853702?hl=en.

[14] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen. Server-Based Traffic Shaping for Stabilizing Oscillating Adaptive Streaming Players. In *Proc. ACM NOSSDAV*, 2013.

[15] A. Balachandran, V. Sekar, A. Akella, and S. Seshan. Analyzing the Potential Benefits of CDN Augmentation Strategies for Internet Video Workloads. In *Proc. ACM IMC*, 2013.

[16] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang. Developing a Predictive Model of Quality of Experience for Internet Video. In *Proc. ACM SIGCOMM*, 2013.

[17] H. Balakrishnan, M. Stemm, S. Seshan, and R. H. Katz. Analyzing Stability in Wide-area Network Performance. *ACM SIGMETRICS Performance Evaluation Review*, 25(1):2–12, 1997.

[18] C. M. Bishop. *Pattern Recognition and Machine Learning*. springer, 2006.

[19] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A Decentralized Network Coordinate System. In *Proc. ACM SIGCOMM*, 2004.

[20] L. De Cicco, S. Mascolo, and V. Palmisano. Feedback Control for Adaptive Live Video Streaming. In *Proc. ACM MMSys*, 2011.

[21] M. Dischinger, M. Marcon, S. Guha, P. K. Gummadi, R. Mahajan, and S. Saroiu. Glasnost: Enabling End Users to Detect Traffic Differentiation. In *Proc. USENIX NSDI*, 2010.

[22] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the Impact of Video Quality on User Engagement. In *Proc. ACM SIGCOMM*, 2011.

[23] A. Ganjam, F. Siddiqui, J. Zhan, X. Liu, I. Stoica, J. Jiang, V. Sekar, and H. Zhang. C3: Internet-Scale Control Plane for Video Quality Optimization. In *Proc. USENIX NSDI*, 2015.

[24] Q. He, C. Dovrolis, and M. Ammar. On the Predictability of Large Transfer TCP Throughput. In *Proc. ACM SIGCOMM*, 2005.

[25] N. Hu, L. Li, Z. M. Mao, P. Steenkiste, and J. Wang. A Measurement Study of Internet Bottlenecks. In *Proc. IEEE INFOCOM*, 2005.

[26] N. Hu, L. E. Li, Z. M. Mao, P. Steenkiste, and J. Wang. Locating Internet Bottlenecks: Algorithms, Measurements, and Implications. In *Proc. ACM SIGCOMM*, 2004.

[27] T. Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In *Proc. ACM SIGCOMM*, 2014.

[28] M. Jain and C. Dovrolis. End-to-end Estimation of the Available Bandwidth Variation Range. *ACM SIGMETRICS Performance Evaluation Review*, 33(1):265–276, 2005.

[29] J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, and H. Zhang. CFA: A Practical Prediction System for Video QoE Optimization. In *Proc. USENIX NSDI*, 2016.

[30] J. Jiang, V. Sekar, and H. Zhang. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming with Festive. *IEEE/ACM Transactions on Networking*, 22(1):326–340, 2014.

[31] S. S. Krishnan and R. K. Sitaraman. Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-experimental Designs. In *Proc. ACM IMC*, 2012.

[32] Y. S. Lim, Y. C. Chen, E. M. Nahum, D. Towsley, and R. J. Gibbens. How Green is Multipath TCP for Mobile Devices? In *Proc. ACM SIGCOMM AllThingsCellular*, 2014.

[33] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An Information Plane for Distributed Services. In *Proc. USENIX OSDI*, 2006.

[34] M. Mirza, J. Sommers, P. Barford, and X. Zhu. A Machine Learning Approach to TCP Throughput Prediction. In *Proc. ACM SIGMETRICS*, 2007.

[35] K. Murphy and M. Dunham. PMTK: Probabilistic Modeling Toolkit. In *Proc. NIPS*, 2008.

[36] A. Nikravesh, D. R. Choffnes, E. Katz-Bassett, Z. M. Mao, and M. Welsh. Mobile Network Performance from User Devices: A Longitudinal, Multidimensional Analysis. In *Proc. PAM*, 2014.

[37] B. A. A. Nunes, K. Veenstra, W. Ballenthin, S. Lukin, and K. Obraczka. A Machine Learning Framework for TCP Round-trip Time Estimation. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):1–22, 2014.

[38] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proc. ACM SIGCOMM*, 1998.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, and J. Vanderplas. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[40] V. Ramasubramanian, D. Malkhi, F. Kuhn, M. Balakrishnan, A. Gupta, and A. Akella. On the Treeness of Internet Latency and Bandwidth. *ACM SIGMETRICS Performance Evaluation Review*, 37(1):61–72, 2009.

[41] G. Ridgeway. Generalized Boosted Models: A Guide to the GBM Package. *Update*, 1(1):1–12, 2007.

[42] K. Salamatian and S. Vaton. Hidden Markov Modeling for Network Communication Channels. In *Proc. ACM SIGMETRICS*, 2001.

[43] S. Sundaresan, W. De Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescape. Broadband Internet Performance: A View From the Gateway. In *Proc. ACM SIGCOMM*, 2011.

[44] S. Tao and R. Guerin. Application-specific Path Switching: a Case Study for Streaming Video. In *Proc. ACM Multimedia*, 2004.

[45] G. Tian and Y. Liu. Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming. In *Proc. ACM CoNEXT*, 2012.

[46] W. Wei, B. Wang, and D. Towsley. Continuous-time Hidden Markov Models for Network Performance Evaluation. *Performance Evaluation*, 49(14):129–146, 2002.

[47] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *Proc. ACM SIGCOMM*, 2015.

[48] X. Yin, V. Sekar, and B. Sinopoli. Toward a Principled Framework to Design Dynamic Adaptive Streaming Algorithms over HTTP. In *Proc. ACM SIGCOMM HotNets*, 2014.

[49] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the Constancy of Internet Path Properties. In *Proc. ACM IMW*, 2001.

[50] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, X. Jin, J. Rexford, and R. K. Sinha. Can Accurate Predictions Improve Video Streaming in Cellular Networks? In *Proc. ACM HotMobile*, 2015.