MIT 6.829: Computer Networks Fall 2017

Lecture 18

Video Streaming *Scribe: Mojdeh Shakeri, Abraham Quintero*

The materials presented in this document are based on in class discussion, and the following papers: [1], [2], [3] and Pensieve slides/notes [4] from the SIGCOM17 conference.

1 Overview

This lecture is about different techniques used for video streaming to improve performance and Quality of Experience (QoE). Specifically, we learned about "neural adaptive video streaming with Pensieve" [1]. Pensieve adaptively selects bitrates (ABR) for future video chunks based on observations collected by client video players. It automatically learns ABR algorithms that adapt to a wide range of environments and QoE metrics using off line reinforcement learning (RL).

When we stream a video from the internet, we may experience slow start or notice the video gets stuck and the application starts rebuffering. Studies show that users start leaving if the video doesnt play in 2-3 seconds. Video streaming accounts for 73% internet traffic and it is expected to grow to %82 in 2021. Online TV and Netflix account for about %20 to %30 of traffic. Given the fact that video streaming has been dominating today's internet traffic, this kind of bad experience will lead to tremendous loss of users. Therefore industry and academia have devoted lots efforts to improve video streaming experience.

How video streaming work in the internet?

Every video is split into chunks. Each chunk is stored at the server at different bitrates, for example: $360pvideoat1Mbps, 480pat2.5Mbps, and720pvideoat5Mbps$. The video client periodically requests individual chunk of video from the video server, and it receives chunks one by one. Each chunk represents several seconds of video and can be requested at one of several bitrates. As the video chunks are downloaded, they are stored in a playback buffer. Units of time are used to refer to buffer size. For example, we might say we have a 2 sec buffer. Figure 1 shows an abstract model of an adaptive video player. The client uses the (1) throughput predictor, and (2) buffer occupancy, to calculate a bitrate to maximize QoE. As the video is playing, the playback buffer is continually drained at a fixed rate since the user watches videos in "real time", but the buffer will be populated at a variable rate, depending on the requested bitrate and network capacity.

The video streaming algorithm goal is to maximize Quality of Experience. The following presents a list of QoE considerations:

1. Best possible quality sustained by throughput
2. Minimize pauses or re-buffering, i.e., playback buffer is empty and cannot render the video,
3. Smoothness, (i.e., do not switch bitrates frequently)
4. Fast startup, (i.e., minimum delay to start) - again users leave if videos buffer for more than 2 or 3 seconds.

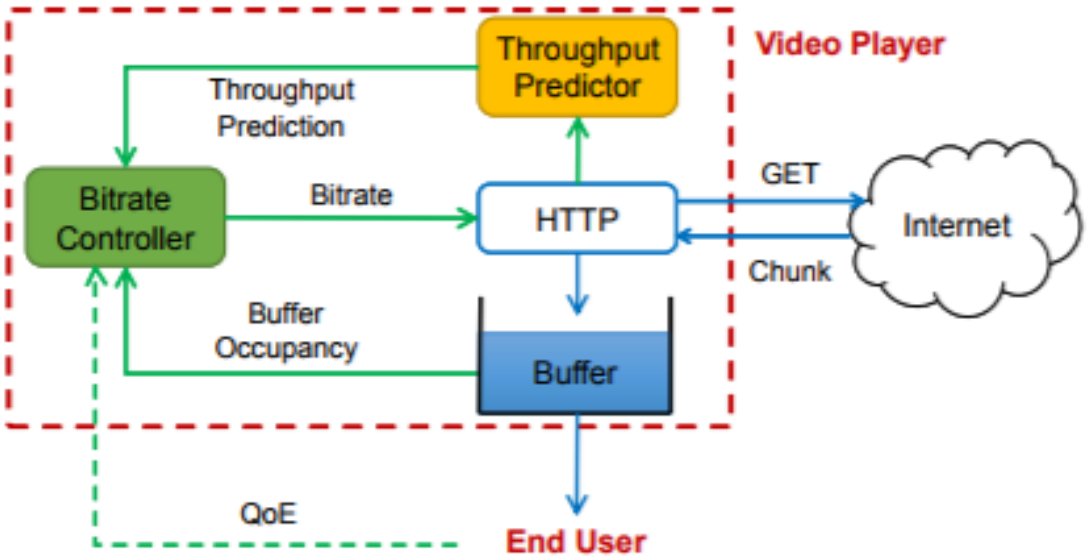

Figure 1: An abstract model of the adaptive video player (Copied form [2])

If chunk download bitrate chosen by the client is larger than the network throughput, then the buffer gets empty fast and needs to rebuffer. In this case the client fails on the second goal above. If the bitrate is slower than the throughput, QoE goes down (it conflicts with first goal- better quality video could have been streamed). To prevent issues like this, video players use Adaptive Bitrate Algorithms (ABR). The goal of ABR algorithm is to choose the bitrate level for future chunks to deliver the highest possible QoE; e.g., maximizing bitrate while minimizing the likelihood of rebuffering and avoiding too many bitrate switches. These algorithms decide both when and at which bitrate to download each video chunk.

Why are ABR algorithms challenging?

- Network throughput is variable and uncertain. Past throughput observations cannot accurately predict what will happen in future.
- There are conflicting QoE goals. People don't simply just want higher quality, they don't want rebuffering, and no bitrate fluctuations.
- The bitrate decision for the next chunk can affect the decisions in the future (cascading effects). For example, using a high bitrate at the current time may deplete the video buffer, which leaves little cushion to deal with the risk rebuffering in case of network fluctuations.

There are multiple ways to develop ABR algorithms:

1. Rate-based algorithms: It picks bitrates based on predicted throughput. It always gives the best possible rate, higher the throughput, higher the bit rate. It ignores buffer occupancy. Netflix was using this and noticed people are leaving when buffer is empty.

- Pros: best quality
- Cons: possibility of re-buffering

2. Buffer-based: It picks bitrate based on buffer occupancy. It attempts to always keep the buffer full. It is independent of throughput.
   - Pros: decreases re-buffering
   - Cons: Low utilization, Low quality
3. Hybrid: It uses both throughput prediction and buffer occupancy. Model Predictive Control (MPC) was presented in SIGCOMM 2015. It uses a control-theoretic model to reason about a broad spectrum of strategies. The algorithm combines throughput and buffer occupancy information to outperform traditional approaches.

All these algorithms use fixed heuristics based on designers insight. They may rely on simplified inaccurate models to come up with the algorithms. This can lead to suboptimal performance under actual network conditions.

Previous work has formalized the QoE objective as a constrained optimization problem. Please see [2], section 3.1 to find more details. Each chunk k is encoded at different bitrates R_k. $d_k(R_k)$ refers to the size of chunk k encoded at bitrate R_k. q is a function which maps selected bitrate R_k to video quality perceived by user. C_k refers to the throughput of the network. The objective function consists of the following:

1. Average Video Quality: The average per-chunk quality over all chunks: $\frac{1}{K}\sum_{k=1}^{K} q(R_k)$
2. Average Quality Variations: This tracks the magnitude of the changes in the quality from one chunk to another: $\frac{1}{k-1}\lambda \sum_{k=1}^{K-1} |q(R_{k+1}) - q(R_k)|$
3. Rebuffer: For each chunk k rebuffering occurs if the download time $d_k(R_k)/C_k$ is higher than the playout buffer level when the chunk download started (i.e., B_k). Thus the total rebuffer time $\mu \sum_{k=1}^{K} \left(\frac{d_k(R_k)}{C_k} - B_k\right)_+$

   The notation $(x)_+ = max(x, 0)$ ensures that this term can never be negative.
4. Startup Delay $\mu_s T_s$.

As users may have different preferences on which of the four components is more important, QoE of video segment 1 through K can be considered as a weighted sum of the components. We can use non-negative weighting parameters. λ corresponds to a preference for fewer bit rate switches. A relatively small λ indicates that the user is not particularly concerned about video quality variability; the large λ is, the more effort is made to achieve smoother changes of video quality. A large μ, relatively to the other parameters, indicates that a user is deeply concerned about rebuffering. In cases where users prefer low startup delay, a large μ_s can be employed.

The constraint for the above maximization problem is as follows:

- C_k is given by the network capacity.
- $t_{k+1} = t_k + \frac{d_k}{c_k}$

- $B_1 = T_s$

The question is "what is the problem with all of these?" Why not simply solve the maximization problem? The problem with this approach is that it needs an accurate model for network throughput- we need to know what C_k will be in the future. To express the optimization problem, we need to plug in some model for how future throughput will evolve. We assumed C_k is given! But what if C_k is changing and we cannot predict the future network trajectory.

What practical schemes end up doing is they rely on a simplified estimate of throughput, such as average throughput or conservative throughput estimation. These heuristics are hard to get right in real networks and lead to suboptimal decisions.

2 Pensieve

Pensieve uses a Reinforcement Learning approach to create ABR algorithm. Reinforcement Learning poses a general framework, where an agent interacts with an environment. At each time step, the agent observes a state from the environment, and takes an action. As a feedback, the environment returns the agent a reward signal, indicating how good the action is. The goal of the learning agent is to maximize the total reward in the environment [2] [1]. To overcome the issues with fixed ABR algorithms, Pensieve turns to design a general purpose learning algorithm. Specifically, it proposes to learn bitrate adaptation policies, by interacting with the video streaming environment in actual network conditions.

Let's understand Pensieve's design in high level: we need to know about the state space, action space and reward signal.

2.1 State Space

Please see Figure 5 in the [2]. The state space needs to include all the information that is relevant to make bitrate decisions:

- Throughput observation: a sequence of throughput measurements, and the time interval of these samples. These observations are collected when previous chunks are downloaded.
- Actual file size for different bitrates of the next chunk. This is useful because the chunk sizes for different bitrates can vary across videos and even across chunks in the same video, depending on the complexity of the video frame.
- Current buffer size.
- The bitrate of the last chunk. The agent needs to know what bitrate it is currently at to control the smoothness of the video.
- Number of chunks left. Towards the end of video playing, maintaining a large buffer doesnt make sense. The bitrate selection policy towards the end can be more aggressive.This term reflects how safe it is to be aggressive.

2.2 Action Space

The action to take would be the bitrate for the the next chunk.

Then, as the feedback, the reward signal can be derived directly from the QoE metric, which considers the 3 QoE factors: bitrate, rebuffering, smoothness. Specifically, we want the quality to be higher, so positive reward. We don't want rebuffering or bitrate fluctuations, so Pensieve gives a penalty whenever the video rebuffers or the bitrate gets changed.

The neural network maps the raw state input to the probability distribution of the next bitrate. This neural network encodes the bitrate adaptation policy. The internal architecture of the neural network has 2 hidden layers, and it has 1D convolutional networks.

2.3 How does Pensieve train the neural network to improve the bitrate adaptation algorithm?

The key idea is you run the system, you collect experience data, which is the trajectory of state, action and reward: what state did I visit, what action did I take and what reward did I receive. To train, Pensieve uses gradient descent to move the neural network parameters in a direction that can increase the reward. Through a guided process of trial and error, it explores different actions and learns what actions provide better reward.

There are some properties of the Reinforcement Learning framework that fit perfectly for the bitrate adaptation problem:

1. Learn the dynamics directly from experience: unlike the heuristics based methods, the learning approach can adapt faithfully to the underlining characteristics of the network in a data-driven way. For example, in the cellular network, it will figure out quantitatively how much buffer it needs to build to cope for uncertainty in the nature of the network.
2. Optimize the high level QoE objective end-to-end: The framework optimizes for a high level QoE metric end-to-end, without having to explicitly model how it is impacted by low level decisions. It can optimize for any arbitrary QoE metric as long as we can represent that metric through the reward signal.
3. Extract control rules from raw high-dimensional signals. This is the power of the underlining neural network.

3 Softcast

DASH (Dynamic Adaptive Streaming over HTTP) separates the compression and transmission layers. What happens if we don't create this arbitrary boundary?

If we have an end to End wireless link, how would we design a protocol for real time video streaming? Can we use AM (Amplitude modulation)? The answer is no. This is because the wireless channel may result in different Signal to Noise ratio in different frequency regions. We might lose large part of video for areas that the channel quality is not good. Advanced military applications use OFDM. Please see [3]. Softcast allows the receiver to decode a video whose bitrate and resolution are

commensurate with the observed channel quality after reception, and it directly sends the videos over OFDM. This approach has applications for multicast and mobile wireless receivers, whose channels differ across time and space.

References

[1] Hongzi Mao, Ravi Netravali, Mohammad Alizadeh, Neural adaptive video streaming with pensive. *SIGCOMM*, 2017.

[2] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, Bruno Sinopoli, A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. *SIGCOMM*, 2015.

[3] Szymon Jakubczak, Hariharan Rahul, Dina Katabi, One-Size-Fits-All Wireless Video, In HotNets, 2009.

[4] Hongzi Mao, Ravi Netravali, Mohammad Alizadeh, Neural Adaptive Video Streaming with Pensieve, SIGCOMM 2017 conference presentation, http://sigcomm.org/sigcomm/2017/files/program/ts-5-2-pensieve.pptx.