Zeng *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2020) 9:38
https://doi.org/10.1186/s13677-020-00189-4

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Towards secure and network state aware bitrate adaptation at IoT edge

Zeng Zeng[1*], Hang Che[2], Weiwei Miao[1], Jin Huang[1], Hao Tang[1], Mingxuan Zhang[1] and Shaqian Zhang[1]

Abstract

Video streaming is critical in IoT systems, enabling a variety of applications such as traffic monitoring and health caring. Traditional adaptive bitrate streaming (ABR) algorithms mainly focus on improving Internet video streaming quality where network conditions are relatively stable. These approaches, however, suffer from performance degradation at IoT edge. In IoT systems, the wireless channels are prone to interference and malicious attacks, which significantly impacts Quality-of-Experience (QoE) for video streaming applications. In this paper, we propose a secure and network-state-aware solution, SASA, to address these challenges. We first study the buffer-level constraint when increasing bitrate. We then analyze the impact of throughput overestimation in bitrate decisions. Based on these results, SASA is designed to consist of both an offline and an online phase. In the offline phase, SASA precomputes the best configurations of ABR algorithms under various network conditions. In the online phase, SASA adopts an online Bayesian changepoint detection method to detect network changes and apply precomputed configurations to make bitrate decisions. We implement SASA and evaluate its performance using 429 real network traces. We show that the SASA outperforms state-of-the-art ABR algorithms such as RobustMPC and Oboe in the IoT environment through extensive experiments.

Keywords: Adaptive bitrate algorithm, IoT

Introduction

With the rapid development of wireless communication and sensing technology, IoT (Internet of Things) has enabled a variety of applications such as environmental monitoring, smart manufacturing, and health caring [1–7]. In these applications, video streaming is of great importance. For example, cameras are deployed at optical lens factories to monitor the production process and perform quality check [8]. And people also use cameras to detect falls of the elderly [9] automatically. According to recent studies [10–14], video analytics algorithms such as detection and recognition are susceptible to video/image quality distortions. Therefore, delivering steady and high-quality videos is critical for IoT applications [15–18].

In recent years, many efforts have been made to improve Internet video streaming quality with adaptive bitrate (ABR) algorithms [19–23]. And state-of-the-art ABR algorithms have been widely used in popular online video services such as Netflix and Hulu. Generally speaking, the goal of ABR algorithms is to play the video at the highest possible bitrate while minimizing rebuffering events. Typically, an ABR algorithm operates in the following manner. The video file is first segmented into short chunks. And then, chunks are encoded at multiple bitrates independently. For each video chunk, the ABR algorithm adaptively chooses a proper bitrate to download in order to optimize different QoE metrics such as the average bitrate of video playback, the time of rebuffering during playback, and the smoothness of picture in video playback [24].

Existing approaches, however, are inadequate for providing high QoE at IoT edge since IoT systems pose additional challenges to video streaming. Firstly, most devices at IoT edge adopt low power wireless communication to

*Correspondence: zengzeng.nju@gmail.com
[1]Information and Telecommunication Branch, State Grid Jiangsu Electric Power Company, Nanjing, China
Full list of author information is available at the end of the article

transfer data [25–27]. Secondly, these systems are often deployed in harsh or remote environments such as factories, oilfields, etc. Thus, wireless link quality is volatile and vulnerable to environmental interference [28, 29] and malicious attacks [30, 31]. Existing algorithms suffer from such link dynamics in IoT systems and may result in misleading bitrate selection. Take Robust MPC algorithm [19] for example. As shown in Fig. 1, when link throughput decreases at t_0, the Robust MPC algorithm cannot capture this sudden change in time. The predicted network throughput stays stable untile t_1. As a result, the buffer at the player side drains after t_3, and the rebuffering process continues until t_4, which significantlyb impacts QoE.

In this paper, we study the adaptive bitrate streaming problem at IoT edge. To overcome these challenge, we propose a secure and network state aware bitrate adaptation algorithm SASA. Specifically, we model a video session as a piecewise-stationary sequence of network states. We then devise an algorithm to detect the change of network state automatically. Once it changes, we search a precomputed table for best parameter configurations and apply them in realtime.

We validate SASA design using 429 throughput traces collected by Akhtar et al. [21]. We use Mahamahi emulation tool [32] to mimic IoT network behavior. And SASA is implemented in a reference DASH client *dash.js* [33]. Through comparison with state-of-the-art ABR algorithm RobustMPC [19] and Oboe [21], we show that the median QoE improvement of SASA is 4.5% and 4.2% respectively.

The remainder of this paper is organized as follows. We discuss related work in "Related work" section. Then, in "Our approach: SASA" section, we present the motivation of this work and the detailed system design. Furthermore, an evaluation is presented in "Evaluation" section. Finally, we conclude this paper and discuss future work in "Conclusion" section.

Related work

Existing ABR algorithms can be mainly divided into three categories, according to their different focus, i.e., bandwidth-based algorithms, buffer-based algorithms, and hybrid algorithms.

Bandwidth-based algorithms

The main idea of bandwidth-based methods is first to estimate link bandwidth and then adjust bitrate accordingly. FESTIV [34] estimates bandwidth to be the harmonic mean of observed throughput over recent chunks. It then designs a delayed update approach to achieve a tradeoff between fairness, stability, and efficiency. Sun et al. [35] systematically quantify throughput predictability using a large-scale dataset and propose a Hidden-Markov-Model based throughput predictor to enhance bitrate selection.

Bandwidth-based algorithms rely on accurate and stable throughput prediction. In practice, however, throughput estimations are usually biased, and accurate bandwidth prediction in wireless networks is still challenging [36, 37].

Buffer-based algorithms

Buffer-based algorithms argue that link bandwidth estimation is usually unreliable. Thus they rely on buffer level information to adapt bitrate. BBA-0 [38] chooses video bitrate simply based on current buffer occupancy. It shows that such a method can reduce the rebuffering rate by 10-20% compared to Netflix's then-default ABR algorithm. Spiteri et al. [39] formulate bitrate adaptation as a utility maximization problem. They propose an online control algorithm BOLA which adopts Lyapunov optimization techniques to maximize video quality and minimize rebuffering.

The advantage of buffer-based algorithms is they only need to keep the buffer at a pre-defined level. But the drawback is, throughput information during video streaming is discarded.

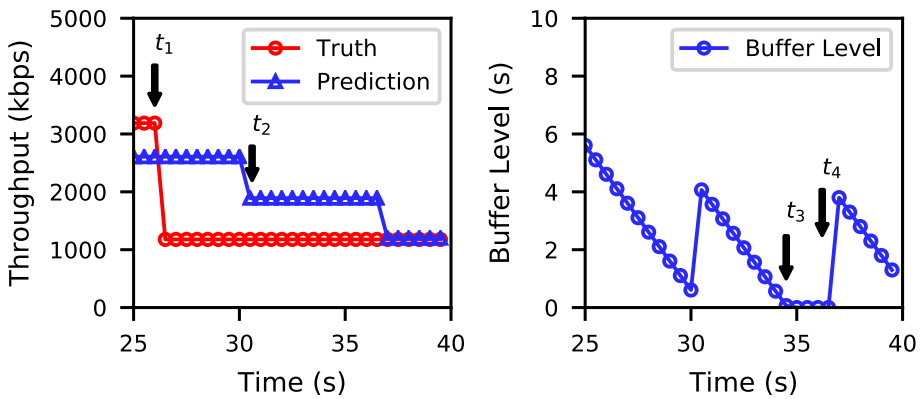

Fig. 1 Throughput overestimation of RobustMPC

Hybrid algorithms

Li et al. [40] observes that when multiple video streaming clients compete at a network bottleneck, the TCP throughput observed by a client cannot indicate its fair-sharing bandwidth. Thus they employ a probe-and-adapt method at the application layer and propose a four-step algorithm PANDA to reduce the instability of video bitrate selection. Yin et al. [19] develop a formal control-theoretic model of the bitrate adaptation problem and propose a model predictive control algorithm by solving a non-trivial discrete optimization problem at each time step. Pensieve [41] models a reinforcement learning problem and selects bitrates for future chunks solely based on the performance of past decisions. Oboe [42] focuses on auto-tune parameters of ABR algorithms to different network conditions in realtime. It significantly improves the performance of algorithms such as RobustMPC and BOLA. As illustrated in Fig. 1, these algorithms suffer from performance degradation in unstable network conditions.

SASA differs from existing studies in the following aspects. First, we study both the buffer-level constraint and impact of throughput overestimation in bitrate decisions and apply the results in the design of SASA. Second, SASA consists of an offline phase in which the best configurations are precomputed under various network conditions. Third, SASA adopts an online Bayesian changepoint detection algorithm to detect network changes and apply precomputed configurations to make bitrate decisions.

Our approach: SASA

System model

We define QoE as a linear combination of the average bitrate of video playback, the time of rebuffering during playback, and the smoothness of picture in video playback following [19, 24]:

$$QoE_{lin} = \sum_{i=1}^{N} q_i - \lambda \sum_{i=1}^{N-1} |q_{i+1} - q_i| - \mu \sum_{i=1}^{N} t_i \tag{1}$$

N is the number of video blocks, and q_i is the bitrate of the block i. Thus $|q_{i+1} - q_i|$ is the bitrate difference between block i and block $i+1$, and λ denotes the penalty coefficient of bitrate switching. Similarly, t_i is the rebuffering time when downloading the block i, and μ is the penalty coefficient of rebuffering.

Following recent studies on wireless channels and network performance [43–45], we assume that wireless links are quasi-static. That means, we can model the network state as a stable process in phases. Specifically, in each phase, the link throughput follows a normal distribution denoted as $\mathcal{N}(\mu_i, \sigma_i)$. Thus, the network state change can be expressed as

$$\mathcal{N}(\mu_i, \sigma_i) \rightarrow \mathcal{N}(\mu_{i+1}, \sigma_{i+1}), \tag{2}$$

where $\mathcal{N}(\mu_i, \sigma_i)$ and $\mathcal{N}(\mu_{i+1}, \sigma_{i+1})$ represent the link throughput distributions in phase i and $i+1$ respectively.

Motivation

In this section, we study the performance of a state-of-the-art ABR algorithm RobustMPC and illustrate two key observations that motivate our system design.

Observation 1: buffer-level constraint

The first observation is that, when the network state is stable, existing algorithms such as RobustMPC use throughput prediction results to adjust bitrate, which may lead to unstable QoE. In this paper, however, we find that we should adjust bitrate not only based on network throughput but also buffer level.

Without loss of generality, assume that in a stable network state, the network throughput is always constant C. The available bitrates of video blocks are $\{q_0, q_1, ..., q_n\}$, where $q_0 < q_1 < ... < q_n$. The length of each video block is T, and the penalty coefficient of bitrate change QoE_{lin} is 1. Suppose the bitrate of current video block is $q_d(t)$, network throughput prediction is $C_p(t)$, and buffer level is $B(t)$.

Corollary 1 *If the bitrate is to be increased from $q_d(t)$ to $q_d(t+1)$, the following equation must hold to avoid rebuffering events:*

$$q_d(t+1) \leq (B(t) + 4T) * \frac{C_p(t)}{5T} \tag{3}$$

Proof According to $B(t)$'s definition, we have

$$B(t+i) = B(t+i-1) - \frac{T * q_d(t+i)}{C_p(t)} + T \tag{4}$$

If we want no rebuffering events during downloading the following 5 consecutive blocks, we have

$$B(t+i-1) \geq \frac{T * q_d(t+i)}{C_p(t)} \tag{5}$$

Combining (4) and (5), we get

$$B(t) \geq \frac{5T * q_d(t+1)}{C_p(t)} - 4T \tag{6}$$

Therefore, the following equation holds

$$q_d(t+1) \leq (B(t) + 4T) * \frac{C_p(t)}{5T}. \tag{7}$$

□

From the above analysis, we can see that if we want to switch to a higher bitrate, a higher level of a buffer is needed. In other words, it means we can reduce the bitrate switching by appropriately reducing the predicted throughput with a discount factor d.

In order to verify Corollary 1, we study a real trace generated from [21] with constant network throughput. In the study, we replace the original network throughput $C_p(t)$ with $C_p(t)' = C_p(t)/(1+d)$, where d is ranging from $\{0, 0.05, 0.10, \ldots, 0.95, 1\}$. When d is zero, it means $C_p(t)'$ is exactly $C_p(t)$. When d is 1, it means $C_p(t)'$ is only half of the original predicted value $C_p(t)$.

We record the decisions of the RobustMPC algorithm, and illustrate an example in Fig. 2 with a constant network throughput of $C = 1750bps$. It depicts that, when d is set to 0, the bitrate is adjusted frequently, which may cause users to receive different quality video blocks. On the contrary, when d is set to 0.4, bitrate selection tends to be stable. The reason is, at $d = 0.4$, the network throughput estimation is lower than its true value. Thus MPC algorithms will not increase bitrate until enough content is accumulated in the buffer. Moreover, after it switches to a higher bitrate since there are enough buffer contents, it can also stay at the higher bitrate for a longer period.

Observation 2: throughput overestimation

The second observation is that, when the network condition is unstable, throughput decrease may lead to rebuffering events. The reason is, most ABR algorithms rely on moving average methods such as EWMA (Exponentially Weighted Moving Average) to predict network throughput [19], and an essential characteristic of moving average methods is that they lag the input data. In unstable network environments, such lags will result in throughput overestimation when network throughput decreases. And the overestimation is a cause of rebuffering events.

Figure 1 illustrates an example from a dataset consisting of 500 video sessions. From the left subfigure, we can see that at time t_1, the network throughput quickly decreases. But throughput prediction result does not drop until time t_2. There is a about 5-second gap between t_1 and t_2. As a result, as shown in the right subfigure, the buffer at the player side drains after t_3, and the rebuffering process continues until t_4, which significantly impacts QoE.

Similar results can also be found in the Oboe algorithm [21]. Oboe dynamically adjusts the parameters of RobustMPC to reduce rebuffering events. However, through trace study, we find that throughput overestimation still exists.

For example, as shown in Fig. 3, there is a sudden network throughput at time t_0. Since Oboe adjusts network prediction results based on the previous five blocks, it cannot capture this sudden change. Consequently, the network throughput is overestimated, leading to rebuffering events.

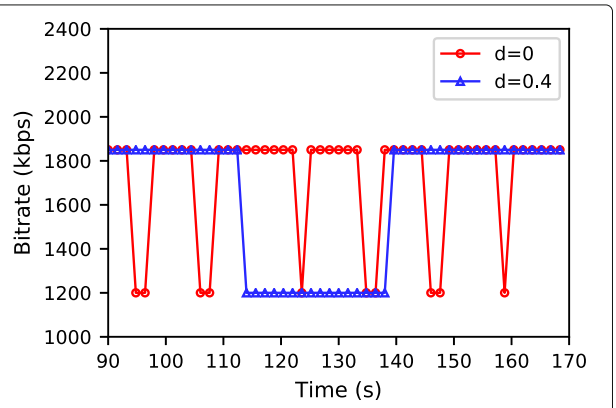

Fig. 2 Observation 1: Bitrate selection results when d is 0 and 0.4

SASA design

Based on previous observations, in this paper, we propose a two-stage approach called SASA to dynamically adjust ABR algorithms at IoT edge.

The system architecture is shown in Fig. 4. First, live video streams from multiple cameras are forwarded to the edge server. In order to keep the data secure, the edge server adopts RC4 [46] as the stream cipher to encrypt/decrypt video streams. And in the offline phase, an analytical client generates offline network throughput traces and mimics the behavior of network condition changes. Then the virtual player invokes ABR algorithms to iterate all cases to find the optimal discount factor d that maximizes QoE. The results are saved in the configuration table. In the online phase, when the analytical client receives encrypted live video streams, it first records realtime network throughput traces. Afterward, a change point detection algorithm is performed to detect network throughput changes. If a change point is detected, bitrate decisions are made based on pre-computed results according to precomputed results in the configuration table.

Offline phase

Algorithm 1 Offline Phase

for each $\mu \in \{50, 100, 150, ..., 10000\} kbps$ **do**
  for each $\sigma \in \{0, 0.05\mu, 0.1\mu, ..., 0.95\mu, \mu\}$ **do**
    $d_{opt} = 0, QoE_{opt} = -\infty$
    for each $d \in \{0, 0.05, ..., 1.00\}$ **do**
      Configure Virtual Player with $\langle(\mu, \sigma), d\rangle$
      Record current QoE as QoE_{curr}
      if $QoE_{curr} \geq QoE_{opt}$ **then**
        $d_{opt} = d, QoE_{opt} = QoE_{curr}$
      end if
    end for
    ConfigTable.save(μ, d_{best})
  end for
end for

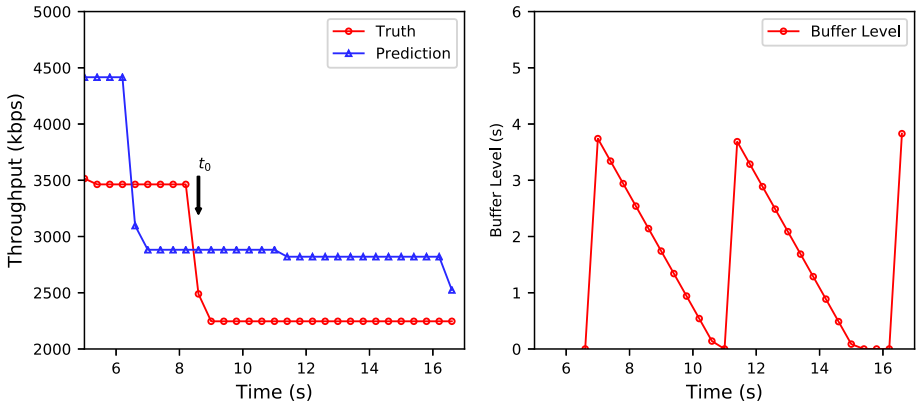

Fig. 3 Observation 2: Throughput overestimation of Oboe

Offline phase

As illustrated in Algorithm 1, in order the iterate all the possible network conditions, we enumerate μ from 0.05Mbps to 10Mbps with an interval of 0.05Mbps in Line 1. For each μ, we set the standard deviation σ to be $[0, 0.05\mu, 0.1\mu, \ldots, 0.95\mu, \mu]$ and get 2200 traces (Line 2). In Line 4, for each trace, we search the optimal discount factor d_{opt} for QoE_{lin} within range $[0, 1]$. The searching process is as follows (Line 5-9): we first configure the virtual player with network throughput $\mathcal{N}(\mu, \sigma)$ and discount factor d; then we record the resulting QoE as QoE_{curr}; if QoE_{curr} is larger than QoE_{opt}, QoE_{opt} is replaced with QoE_{curr}, and d is recorded as d_{opt}.

In practice, we find that in about 94% cases, when μ is fixed, different σs share a common optiaml d_{opt}. And in the rest 6% cases, for the same μ, the differences between d_{opt} are with 0.1.

Online phase

Algorithm 2 illustrates the online phase of SASA. During the downloading process, SASA records network throughput values every 100ms. In Line 2-4, if current video chunk is not downloaded, we simply record the throughput value in *queue*. After current video chunk is downloaded, SASA detects the change point of network throughput trace usning an online Bayesian change point detection algorithm (Line 6). If a change point is detected, the configuration engine starts to work (Line 7-10). It first finds an optimal value of discount factor d in the configuration table and then reconfigures this parameter for ABR algorithms. In addition, if the current change point is a sudden decreasing point, we calculate predicted throughput of $pred_t$ in Line 12-13. And if $pred_t$ is larger than μ, we set the throughput prediction value to μ (Line 14-16). Afterward, based on equation (7), we choose a set

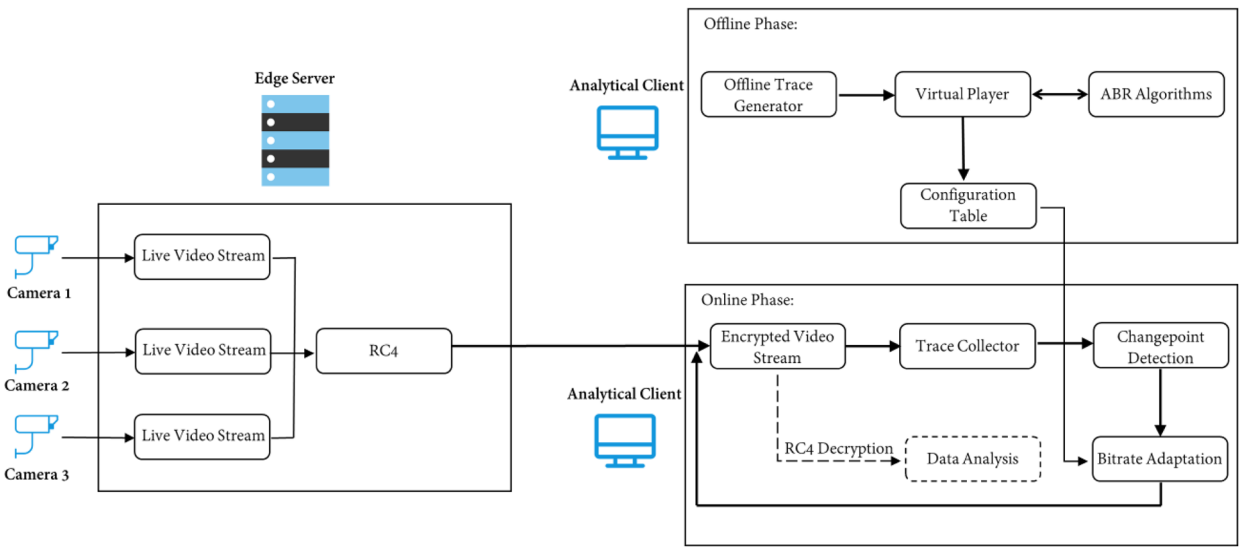

Fig. 4 The architecture of SASA

Algorithm 2 Online Phase

1: **for** each network throughput value n_t at time t **do**
2: **if** current video chunk is not downloaded **then**
3: $queue = queue \cup (t, n_t)$
4: continue
5: **end if**
6: detect change point in *queue*
7: **if** a change point p is detected **then**
8: μ = ChangePointDetector.getAverage()
9: d = ConfigTable.getDiscount(μ)
10: **end if**
11: **if** p is a decreasing point **then**
12: $avg = 5/\sum_{i=0}^{4} \frac{downloadTime[t-i]}{CHUNK\_SIZE}$
13: $pred_t = \frac{avg}{1+d}$
14: **if** $pred_t > \mu$ **then**
15: $pred_t = \mu$
16: **end if**
17: **end if**
18: choose $[b_{t+1}, b_{t+2}, ..., b_{t+5}]$ *s.t.* QoE_{lin} is maximized for the next 5 chunks
19: set bitrate to b_{t+1}
20: $queue = \emptyset$
21: **end for**

of optimal bitrates $[b_{t+1}, b_{t+2}, ..., b_{t+5}]$ such that QoE_{lin} is maximized for the next 5 chunks (Line 18). Finally, b_{t+1} is selected as the new bitrate (Line 19).

Change Point Detection In order to detect network throughput changes in realtime, we adopt an online Bayesian changepoint detection algorithm [47] here. When a video chunk is being downloaded, we first measure network throughputs every t_s seconds, and get $[c_1, c_2, \dots]$. For each data point c_i in the queue, it could be either a change point or a growth point. We use r_i to represent how long c_i has been living. If c_i is a change point, $r_i = 0$, and if c_i is a growth point, $r_i = r_{i-1} + 1$. Given the sampling point c_i, we calculate the distribution of r_i using:

$$P(r_i|c_{1:i}) = P(r_i, c_{1:i})/P(c_{1:i}) \tag{8}$$

Finally, we calculate the expectation $E(r_i|c_{1:i})$, and if $E(r_i|c_{1:i})$ is approximately 0, we judge that a change point is detected. In practice, when c_i is a change point, r_i is not necessarily 0, thus we use a thrshold $r_{threshold}$ instead. We then get the network state (μ_i, σ_i) by calculating the average value and standard deviation of recent points and search the configuration table for optimal d.

Throughput Overestimation Detection After a change point is detected, we also have to detect throughput overestimation. We compare the network throughput prediction $C_p(i)' = C_p(i)/(1+d)$ with μ_i. If $\mu_i < C_p(i)/(1+d)$, we can judge that a throughput overestimation is detected. We then reset $C_p(i)'$ to be μ_i.

Implementation

We implement a virtual player to mimic the behavior of a client. In the online phase, the virtual player can simulate the process of buffering and playing process of video chunks, as well as network throughput changes. During the playing process, it calls ABR algorithms to obtain the bit rate decision, simulates the playing process of video, and finally output the user experience index of playing (average bit rate, rebuffering time, bitrate switching situation, and QoE_{lin}) to measure the performance. In the offline phase, the virtual player simulates the video playing process with no actual block buffering and video playing. Thus it can quickly find the optimal d in the whole search space. In our experiment, it simulates playing a 193-second video only in less than 1s.

The main functions in Fig. 4 are achieved by rewriting *Dash.js* [33]. The advantage of rewriting *Dash.js* is, it is purely on the client side. Thus the server settings can remain unchanged. The rewriting is mainly related to the following three modules: *ABRRulesCollection*, *ABRController*, *ThroughputHistory*. In *Dash.js*, the client obtains a video block by sending an *XMLHttpRequest*. It records the throughput trace in the downloading process of the video block by setting the *onprogress* callback function. When the block is downloaded, the throughput traces at the time of downloading are transferred to the *ThroughputHistory* module, Nevertheless, these traces are not saved and processed in the existing *dash.js*. We rewrite the *ThroughputHistory* module, save the traces in the video block download process, and provides an interface for other modules to obtain these traces. *ABRccontroller* module is the main interface of the ABR algorithm in *dash.js*. It invokes the *getmaxquality* interface of the *ABRRulesCollection* module to get the bit rate decision of the next block. *ABRRulesCollection* calls a variety of ABR algorithms, obtains their decisions respectively, and then selects one of them as the final decision. In this paper, we implement SASA in dash.js and add it to the *ABRRulesCollection* module.

Evaluation

In this section, we compare the performance of SASA with state-of-the-art algorithms Oboe and RobustMPC.

Experiment setup

Dataset The data used in our experiment is extracted from Oboe dataset [48]. These traces are collected over three months under various network conditions such as WiFi and 3G/4G. And the video clips downloaded by clients are between 4-6 minutes.

Experimental Metrics Following Oboe [21] and MPC [19], we focus on the average bit rate of video playback, rebuffering time as well as the change of bitrate, and measure the overall performance through QoE_{lin} which

is a linear combination of above metrics. For rebuffering penalty coefficient ρ and the bit rate switching penalty coefficient λ in the QoE_{lin}, we set $\rho = 4300$ and $\lambda = 1$ following [21].

Hardware Settings We deploy video chunks as static files on an Nginx server. The available bitrates are {300, 700,1200,1850,2850,4300} Kbps. The server is with a 4-core, 1.2Ghz, Intel i7 CPU, and the operating system is Ubuntu 16.04.

Software Settings To mimic network behaviors in the IoT environment, we adopt the Mahimahi mm-link tool [32] to simulate network throughputs. And we use selenium (version 3.141.0) to control the Chrome browser (version 73.0) to play the video, and then collect the browser log to get the video playback index. The client's video buffer size is 20 seconds.

System performance

QoE In Fig. 5a, the CDF curves represent SASA's improvement over RobustMPC and Oboe respectively. The median improvement of SASA over RobustMPC is 4.5%. And the median improvement of SASA over Oboe is 4.2%.

Average Bitrate Figure 5b depicts the average bitrates of the three algorithms. The average bitrates are relatively close. For example, the median value of average bitrates is 2455.2 Kbps for SASA, 2467.7kbps for RobustMPC, 2458.9 Kbps for Oboe. We can also see that bitrate switches in SASA and Oboe are relatively stable, compared with RobustMPC.In terms of median value, the bitrate changes of SASA and Oboe are 153.2 Kbps, while that of RobustMPC is 172.4 Kbps.

Rebuffering Time From Figure 5, it can be seen that SASA has the shortest rebuffering time and the best performance, followed by Oboe, and RobustMPC has the longest rebuffering time. For 22.9% of the sessions, SASA experienced a rebuffering phenomenon, with an average rebuffering time of 1.49 s. For 26.2% of the sessions, Oboe experienced a rebuffering phenomenon, with an average rebuffering time of 1.96 s. For 50.9% of the sessions, RobustMPC experienced a rebuffering phenomenon, with an average rebuffering time of 1.37 s. Compared with RobustMPC, SASA decreases the rebuffering time in 42.1% of sessions, with an average reduction of 0.99s. In 5.6% of sessions, SASA increases the rebuffering time, with an average increment of 0.73s. Compared with Oboe, in 8.0% sessions, SASA reduces the rebuffering time, with

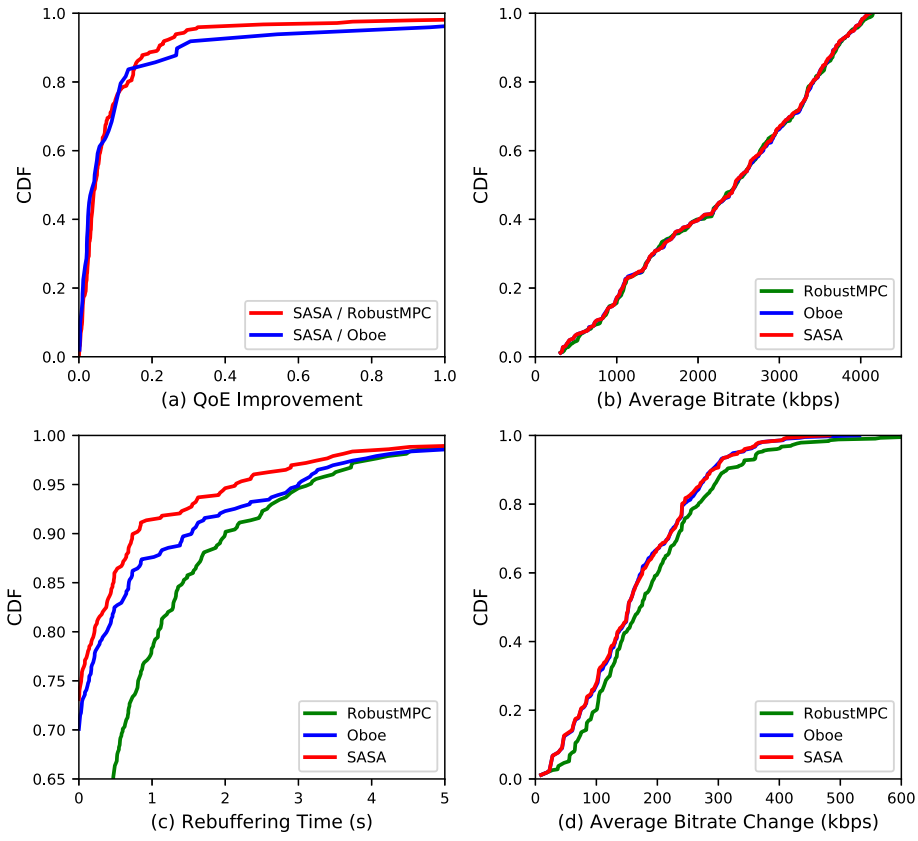

Fig. 5 Comparision with RobustMPC and Oboe

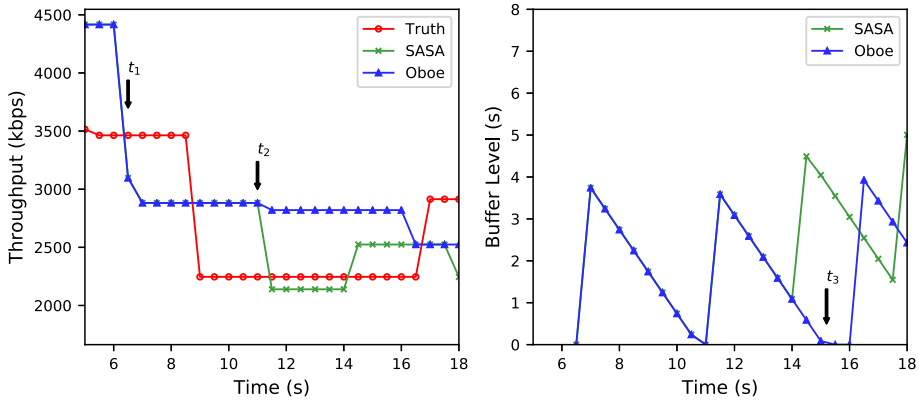

Fig. 6 An example of reducing rebuffering time by changepoint detection

an average reduction of 2.24s. And in 1.0% sessions, SASA increases the rebuffering time, with an average increment of 0.67s.

From previous data, we can see that compared with Oboe and RobustMPC, the rebuffering time is reduced in SASA. The reduction is attributed to changepoint detection since the network throughput overestimation. We illustrate an example in Fig. 6. Oboe detects the change of network state at time t_1, but the network throughput prediction is not lowered, resulting in rebuffering at time t_2. On the contrary, after detecting the sudden drop of throughput at t_1, SASA adjusts the predicted value of throughput by decreasing d, thus avoiding rebuffering events. However, even with changepoint detection, rebuffering events still occur. And in a few sessions, the rebuffering time increases. We will discuss the reasons in the following sections.

To evaluate the impact of network throughput prediction, we modify the parameter k in prediction. We estimate network throughput based on the most recent chunk, while the default way is based on the recent five chunks ($k_{default} = 5$). As shown in Fig. 7, we can easily find that when $k = 1$, the variation of the network is more significant than that default. In specific, the bitrate decision tends to be less stable. In specific, we find that the median of average bitrate when $k = 1$ is 0.7% higher than that when $k = 5$. But the corrsponding QoE_{lin} is 1.4% less.

Analysis of rebuffering

As presented above, SASA avoids the overestimation of network throughput, thus reducing the rebuffering frequency and rebuffering time. However, rebuffering events still occur in SASA, and the rebuffering time in some sessions is longer than that in RobustMPC and Oboe. We perform experiments and find that the main reasons are as follows:

Rebuffering at the beginning. According to the implementation mechanism of *dash.js*, after the second video chunk, it needs to maintain a minimum buffer size. That is, when the content in the buffer is less than a certain threshold, it will stop playing and cause a rebuffering event. Such a rule makes rebuffering events frequently happen between the second and third chunk. And the corresponding rebuffering time accounts for 51.8% of the

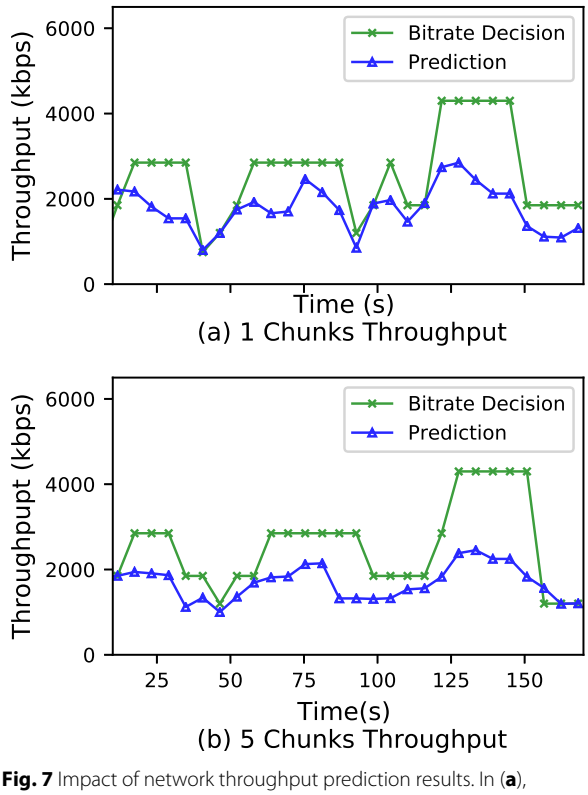

Fig. 7 Impact of network throughput prediction results. In (**a**), network throughput is estimated based on the most recent chunk. In (**b**), network throughput is estimated using the recent five chunks

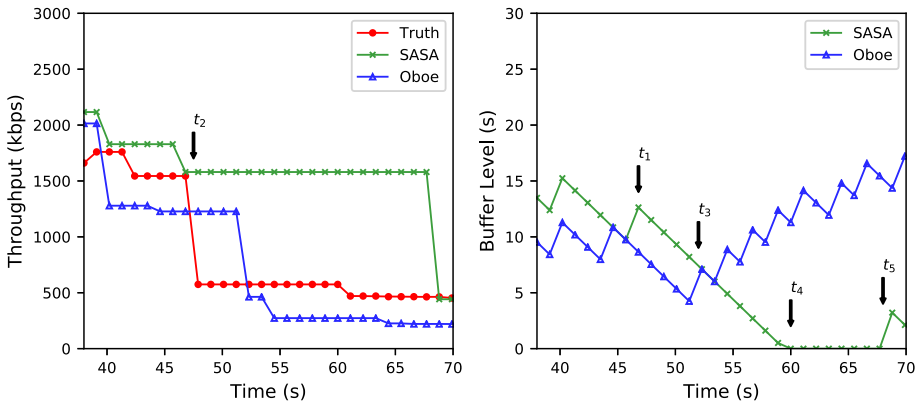

Fig. 8 Rebuffering events for SASA and Oboe

total rebuffering time Since this is due to the implementation of *dash.js*, the rebuffering time of SASA, RobustMPC, and Oboe are similar at the beginning.

Rebuffering caused by throughput reduction. During the process of downloading a chunk, a sudden drop in network throughput will raise a rebuffering event. Even if SASA detects such a changepoint in realtime, the reconfiguration of ABR algorithms takes effect only when the current chunk is finished downloading. Therefore, there is a delay between the detection and rebuffering event.

Besides, we also find some cases when SASA encounters rebuffering events while Oboe does not. Figure 8 illustrates an example in our experiment. At time t_0, SASA makes bitrate decisions for the next chunk. Since the current network throughput is stable, it chooses a larger bitrate. However, as soon as it starts downloading the next chunk, network throughput decreases at time t_1. Even if SASA detects such a change in realtime, it cannot adjust bitrate until the current chunk is finished downloading. As a result, a rebuffering event happens at time t_3. On the

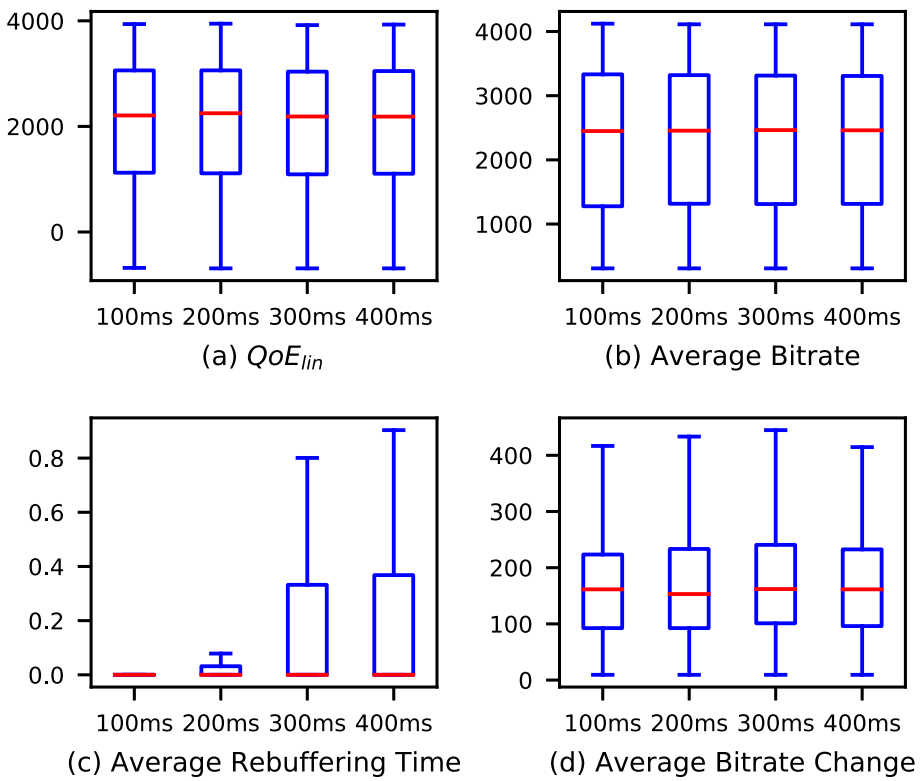

Fig. 9 Impact of $r_{threshold}$ in changepoint detection

other side, since Oboe detects the network change at t_2, which is later than SASA, it lowers network throughput estimation and avoids rebuffering.

In our experiment, the rebuffering time caused by throughput reduction accounts for 21.4% of the total value. In the absence of a reliable and precise throughput prediction algorithm, it is difficult to find an optimal mechanism that can be applied in every scenario.

The network throughput is lower than the minimum bitrate. When network throughput keeps at a lower value than minimum bitrate for a long time, even if ABR algorithms always select the lowest bit rate, rebuffering events will frequently occur. In our experiment, we find such a situation is rare, and only 0.7% of the total traces are in this case. However, once it appears, it is likely to cause a long time of rebuffering, and the rebuffering time accounts for about 24.5% of the total value. Rebuffering events caused by low network throughput are hard to avoid because the only way to solve this is to accumulate the contents of the buffer until exceeding a certain threshold.

Impact of changepoint detection

We also evaluate the impact of changepoint detection results. By varying the parameter $r_{threshold}$ in changepoint detection, we can get different changepoint detection results. As illustrated in Fig. 9, we find that for different $r_{threshold}$ in $\{100ms, 200ms, 300ms, 400ms\}$, QoE_{lin}, average bitrate, and average bitrate change are not affected. But with a smaller $r_{threshold}$, the average rebuffering time is smaller and more stable. So we choose $r_{threshold} = 200ms$.

System cost analysis

We also evaluate the computation cost for both offline and online phases. In the offline phase, for each network state (μ, σ), it needs 8 seconds to find the optimal d. Since there are 200*20 network states, it needs 8.9 hours in total. In the online phase, we find that for each chunk, it needs 100ms for changepoint detection and 20ms for ABR reconfiguration.

Conclusion

In this paper, we study network state aware transmission in IoT systems. Our solution is motivated by two examples in real traces. We find that existing ABR algorithms are not suitable for IoT systems because they are not aware of network dynamics in IoT. We propose SASA, which can automatically detect network state changes and adjust bitrate decisions. Through extensive experiments, we demonstrate the median QoE improvement of SASA is 4.5% and 4.2% respectively compared with state-of-the-art methods.

Author information

Zeng Zeng is a senior researcher with Information and Telecommunication Branch, State Grid Jiangsu Electric Power Company. His research interests are Internet of Things and smart grid.

Hang Che is currently a research assistant at Tsinghua Wuxi Research Institute of Applied Technologies. His research interests are edge computing and wireless networks.

Weiwei Miao is a principal researcher with Information and Telecommunication Branch, State Grid Jiangsu Electric Power Company. His research interests includes power communication systems, wireless access networks, and communication network management.

Jin Huang is a principle engineer with Information and Telecommunication Branch, State Grid Jiangsu Electric Power Company. His research interests are power communication systems and wireless communication.

Hao Tang is an assistant researcher with Information and Telecommunication Branch, State Grid Jiangsu Electric Power Company. His research interests includes software process improvement and practical software engineering.

Mingxuan Zhang is currently a senior researcher with Information and Telecommunication Branch, State Grid Jiangsu Electric Power Company. His research interests is wireless communication and wireless power network.

Shaqian Zhang is a senior engineer with Information and Telecommunication Branch, State Grid Jiangsu Electric Power Company. His research interests include communication networks and network security.

Abbreviations

ABR: Adaptive Bitrate; QoE: Quality-of-Experience; IoT: Internet of Things; EWMA: Exponentially Weighted Moving Average

Acknowledgements

The authors would like to thank the Information and Telecommunication Branch, State Grid Jiangsu Electric Power Company for supporting this work.

Authors' contributions

Zeng Zeng and Hang Che proposed the idea and mainly wrote the manuscript. Weiwei Miao, Jin Huang and Hao Tang collaborated in the conception, research and algorithm design. Mingxuan Zhang and Shaqian Zhang contributed part of the writing and the evaluations. All authors read and approved the final manuscript.

Funding

This work was funded by State Grid Jiangsu Electric Power Company.

Availability of data and materials

Not Applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Information and Telecommunication Branch, State Grid Jiangsu Electric Power Company, Nanjing, China. [2]Tsinghua Wuxi Research Institute of Applied Technologies, Wuxi, China.

Received: 2 April 2020 Accepted: 3 July 2020
Published online: 13 July 2020

References

1. Gao H, Duan Y, Shao L, Sun X (2019) Transformation-based processing of typed resources for multimedia sources in the IoT environment. Wirel Netw. https://doi.org/10.1007/s11276-019-02200-6
2. Kuang L, Yan X, Tan X, Li S, Yang X (2019) Predicting taxi demand based on 3d convolutional neural network and multi-task learning. Remote Sens 11(11):1265
3. Deng S, Xiang Z, Zhao P, Taheri J, Gao H, Yin J, Zomaya AY (2020) Dynamical resource allocation in edge for trustable iot systems: a reinforcement learning method. IEEE Trans Ind Inform 16(9):6103–6113. https://doi.org/10.1109/tii.2020.2974875
4. Gao H, Liu C, Li Y, Yang X (2020) V2vr: Reliable hybrid-network-oriented v2v data transmission and routing considering rsus and connectivity probability. IEEE Trans Intell Transp Syst : 1–14. https://doi.org/10.1109/tits.2020.2983835
5. Gao H, Xu Y, Yin Y, Zhang W, Li R, Wang X (2019) Context-aware qos prediction with neural collaborative filtering for internet-of-things services. IEEE Internet Things J 7(5):4532–4542

6. Mao X, Miao X, He Y, Li X-Y, Liu Y (2012) Citysee: Urban co2 monitoring with sensors. In: 2012 Proceedings IEEE INFOCOM. IEEE. https://doi.org/10.1109/infcom.2012.6195530
7. Chen B, Wan J, Shu L, Li P, Mukherjee M, Yin B (2017) Smart factory of industry 4.0: Key technologies, application case, and challenges. IEEE Access 6:6505–19
8. Using Artificial Intelligence to Improve Quality Control. http://alturl.com/y5x8d. Access 20 Mar 2020
9. de Miguel K, Brunete A, Hernando M, Gambao E (2017) Home camera-based fall detection system for the elderly. Sensors 17(12):2864
10. Korshunov P, Ooi WT (2011) Video quality for face detection, recognition, and tracking. ACM Trans Multimed Comput Commun Appl 7(3):14
11. Dodge S, Karam L (2016) Understanding how image quality affects deep neural networks. In: 2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX). IEEE. https://doi.org/10.1109/qomex.2016.7498955
12. Marciniak T, Chmielewska A, Weychan R, Parzych M, Dabrowski A (2015) Influence of low resolution of images on reliability of face detection and recognition. Multimed Tools Appl 74(12):4329–49
13. Yu J, Tan M, Zhang H, Tao D, Rui Y (2019) Hierarchical deep click feature prediction for fine-grained image recognition. IEEE Trans Pattern Anal Mach Intell:1. https://doi.org/10.1109/tpami.2019.2932058
14. Yu J, Li J, Yu Z, Huang Q (2019) Multimodal transformer with multi-view visual representation for image captioning. IEEE Trans Circ Syst Video Technol:1. https://doi.org/10.1109/tcsvt.2019.2947482
15. Xiao X, Wang W, Chen T, Cao Y, Jiang T, Zhang Q (2019) Sensor-augmented neural adaptive bitrate video streaming on UAVs. IEEE Trans Multimed 22(6):1567–1576
16. Sakaushi A, Kanai K, Katto J, Tsuda T (2018) Edge-centric video surveillance system based on event-driven rate adaptation for 24-hour monitoring. In: 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). IEEE. https://doi.org/10.1109/percomw.2018.8480272
17. Xu X, Liu J, Tao X (2017) Mobile edge computing enhanced adaptive bitrate video delivery with joint cache and radio resource allocation. IEEE Access 5:16406–16415
18. Guo J, Gong X, Wang W, Que X, Liu J (2019) Sasrt: semantic-aware super-resolution transmission for adaptive video streaming over wireless multimedia sensor networks. Sensors 19(14):3121
19. Yin X, Jindal A, Sekar V, Sinopoli B (2015) A control-theoretic approach for dynamic adaptive video streaming over http. In: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15. ACM. https://doi.org/10.1145/2785956.2787486
20. Huang T-Y, Johari R, McKeown N, Trunnell M, Watson M (2014) A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In: roceedings of the 2014 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '14. ACM Vol. 44. pp 187–198. https://doi.org/10.1145/2619239.2626296
21. Akhtar Z, Nam YS, Govindan R, Rao S, Chen J, Katz-Bassett E, Ribeiro B, Zhan J, Zhang H (2018) Oboe: auto-tuning video abr algorithms to network conditions. In: Proceedings of the 2018 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '18. ACM. pp 44–58. https://doi.org/10.1145/3230543.3230558
22. Juluri P, Tamarapalli V, Medhi D (2015) Sara: Segment aware rate adaptation algorithm for dynamic adaptive streaming over http. In: 2015 IEEE International Conference on Communication Workshop (ICCW). IEEE. https://doi.org/10.1109/iccw.2015.7247436
23. Spiteri K, Urgaonkar R, Sitaraman RK (2016) Bola: Near-optimal bitrate adaptation for online videos. In: IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications. IEEE. https://doi.org/10.1109/infocom.2016.7524428
24. Dobrian F, Sekar V, Awan A, Stoica I, Joseph D, Ganjam A, Zhan J, Zhang H (2011) Understanding the impact of video quality on user engagement. In: Proceedings of the ACM SIGCOMM 2011 conference on SIGCOMM - SIGCOMM '11. ACM Vol. 41. https://doi.org/10.1145/2018436.2018478
25. Kirichek R, Pham V-D, Kolechkin A, Al-Bahri M, Paramonov A (2017) Transfer of multimedia data via lora. In: Internet of Things, Smart Spaces, and Next Generation Networks and Systems. Springer. pp 708–720. https://doi.org/10.1007/978-3-319-67380-6_67
26. Rosário D, Zhao Z, Santos A, Braun T, Cerqueira E (2014) A beaconless opportunistic routing based on a cross-layer approach for efficient video dissemination in mobile multimedia iot applications. Comput Commun 45:21–31
27. Floris A, Atzori L (2015) Quality of experience in the multimedia internet of things: Definition and practical use-cases. In: 2015 IEEE International Conference on Communication Workshop (ICCW). IEEE. https://doi.org/10.1109/iccw.2015.7247433
28. Dong W, Liu Y, He Y, Zhu T, Chen C (2014) Measurement and analysis on the packet delivery performance in a large-scale sensor network. IEEE/ACM Trans Netw (TON) 22(6):1952–1963
29. Liu Y, Mao X, He Y, Liu K, Gong W, Wang J (2013) Citysee: Not only a wireless sensor network. IEEE Netw 27(5):42–47
30. Grieco LA, Boggia G, Sicari S, Colombo P (2009) Secure wireless multimedia sensor networks: a survey. In: 2009 Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. IEEE. pp 194–201. https://doi.org/10.1109/UBICOMM.2009.27
31. Venčkauskas A, Morkevicius N, Bagdonas K, Damaševičius R, Maskeliūnas R (2018) A lightweight protocol for secure video streaming. Sensors 18(5):1554
32. Netravali R, Sivaraman A, Das S, Goyal A, Winstein K, Mickens J, Balakrishnan H (2015) Mahimahi: Accurate record-and-replay for {HTTP}. In: roceedings of the 2015 USENIX Annual Technical Conference, USENIX ATC '15. USENIX Association. pp 417–429. https://dl.acm.org/doi/10.5555/2813767.2813798
33. Dash.js. https://github.com/Dash-Industry-Forum/dash.js/ Access 20 Mar 2020
34. Jiang J, Sekar V, Zhang H (2014) Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. IEEE/ACM Trans Netw (ToN) 22(1):326–340
35. Sun Y, Yin X, Jiang J, Sekar V, Lin F, Wang N, Liu T, Sinopoli B (2016) Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction. In: Proceedings of the 2016 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '16. ACM. pp 272–285. https://doi.org/10.1145/2934872.2934898
36. Liu Y, Lee JY (2015) An empirical study of throughput prediction in mobile data networks. In: 2015 IEEE Global Communications Conference (GLOBECOM). IEEE. https://doi.org/10.1109/glocom.2015.7417858
37. Zou XK, Erman J, Gopalakrishnan V, Halepovic E, Jana R, Jin X, Rexford J, Sinha RK (2015) Can accurate predictions improve video streaming in cellular networks?. In: Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications - HotMobile '15. ACM. https://doi.org/10.1145/2699343.2699359
38. Huang T-Y, Johari R, McKeown N, Trunnell M, Watson M (2014) A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In: ACM SIGCOMM Computer Communication Review. ACM Vol. 44. pp 187–198. https://doi.org/10.1145/2619239.2626296
39. Spiteri K, Urgaonkar R, Sitaraman RK (2016) Bola: Near-optimal bitrate adaptation for online videos. In: IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications. IEEE. https://doi.org/10.1109/infocom.2016.7524428
40. Li Z, Zhu X, Gahm J, Pan R, Hu H, Begen AC, Oran D (2014) Probe and adapt: Rate adaptation for http video streaming at scale. IEEE J Sel Areas Commun 32(4):719–733
41. Mao H, Netravali R, Alizadeh M (2017) Neural adaptive video streaming with pensieve. In: Proceedings of the Conference of the ACM Special Interest Group on Data Communication. ACM. pp 197–210. https://doi.org/10.1145/3098822.3098843
42. Akhtar Z, Nam YS, Govindan R, Rao S, Chen J, Katz-Bassett E, Ribeiro B, Zhan J, Zhang H (2018) Oboe: auto-tuning video abr algorithms to network conditions. In: Proceedings of the 2018 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '18. ACM. pp 44–58. https://doi.org/10.1145/3230543.3230558
43. Yang W, Durisi G, Koch T, Polyanskiy Y (2014) Quasi-static multiple-antenna fading channels at finite blocklength. IEEE Trans Inf Theory 60(7):4232–4265
44. Ghrayeb A, Duman TM (2003) Performance analysis of mimo systems with antenna selection over quasi-static fading channels. IEEE Trans Veh Technol 52(2):281–288

45. Balachandran A, Voelker GM, Bahl P, Rangan PV (2002) Characterizing user behavior and network performance in a public wireless lan. ACM SIGMETRICS Perform Eval Rev 30:195–205
46. Fluhrer S, Mantin I, Shamir A (2001) Weaknesses in the key scheduling algorithm of rc4. In: International Workshop on Selected Areas in Cryptography. Springer. pp 1–24. https://doi.org/10.1007/3-540-45537-x_1
47. Adams RP, MacKay DJ (2007) Bayesian online changepoint detection. arXiv preprint arXiv:0710.3742
48. Oboe traces. https://github.com/USC-NSL/Oboe/Access 20 Mar 2020

Publisher's Note