piStream: Physical Layer Informed Adaptive Video Streaming Over LTE

Xiufeng Xie and Xinyu Zhang
University of Wisconsin-Madison
{xiufeng, xyzhang} @ece.wisc.edu

Swarun Kumar
MIT
swarun@mit.edu

Li Erran Li
Fudan University
erranlli@gmail.com

ABSTRACT

Adaptive HTTP video streaming over LTE has been gaining popularity due to LTE's high capacity. Quality of adaptive streaming depends highly on the accuracy of client's estimation of end-to-end network bandwidth, which is challenging due to LTE link dynamics. In this paper, we present piStream, that allows a client to efficiently monitor the LTE basestation's PHY-layer resource allocation, and then map such information to an estimation of available bandwidth. Given the PHY-informed bandwidth estimation, piStream uses a probabilistic algorithm to balance video quality and the risk of stalling, taking into account the burstiness of LTE downlink traffic loads. We conduct a real-time implementation of piStream on a software-radio tethered to an LTE smartphone. Comparison with state-of-the-art adaptive streaming protocols demonstrates that piStream can effectively utilize the LTE bandwidth, achieving high video quality with minimal stalling rate.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communications*

General Terms

Algorithms, Design, Theory, Performance

Keywords

LTE; Adaptive Streaming; MPEG-DASH; HTTP

1. INTRODUCTION

Mobile video streaming has witnessed a surge in the past few years, accounting for 70% of the mobile Internet traffic, with a compound annual growth rate of 78% [1]. The LTE cellular services have been massively deployed to match such growing traffic demand, with a peak downlink bitrate of 300 Mbps, almost 10× over 3G. However, user-perceived quality of experience (QoE) remains unsatisfactory. A recent world-wide measurement survey [2] reveals that, even in regions with wide LTE coverage, LTE only increases video quality by 20% over 3G. On the other hand, the average stalling time remains 7.5 to 12.3 seconds for each minute's mobile video playback [2].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
MobiCom'15, September 7–11, 2015, Paris, France.
© 2015 ACM. ISBN 978-1-4503-3619-2/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2789168.2790118.

These two effects are seemingly contradictory: the video streaming application seems to severely underutilize the LTE bandwidth, yet stalling occurs due to bandwidth overestimation. However, based on a microscopic measurement study (Section 2), we identify a single root cause behind: the inability of the streaming application to track the network bandwidth which is affected by downlink traffic dynamics at the basestation.

Our measurement focuses on the popular HTTP based adaptive streaming protocols (collectively called *DASH*) [2]. A DASH client adaptively requests video segments of certain quality (size) from the server, to ensure the best-quality segment can be downloaded in time for playback. Due to high probing cost, a DASH client needs to infer the *current bandwidth* implicitly from the throughput of its video segments. However, the throughput value may largely *underestimate* the bandwidth unless a video segment can saturate the client's end-to-end bandwidth. On the other hand, DASH uses historical bandwidth records to predict future bandwidth, which occasionally leads to *overestimation*, and hence video stalling, as the LTE network condition fluctuates. In either aspect, the DASH client is left to suffer from the worst impact of both underestimation and overestimation of the end-to-end network bandwidth.

To meet the above challenges, we dig for the network bandwidth utilization from the LTE physical layer. The well organized radio resource structure distinguishes LTE from most other wireless communication systems like WiFi or Bluetooth. Given a particular LTE cell and time duration, the total amount of radio resources available for downlink transmission is always known. Moreover, the end-to-end network bandwidth is typically bottlenecked by the access link bandwidth due to the architectural property of cellular networks [3, 4]. These unique features bring new opportunities to remedy the bandwidth underutilization problem.

This paper presents piStream, which takes full advantage of the aforementioned features to enhance adaptive video streaming over LTE. piStream is a client-centric video adaptation framework, compatible with the MPEG-DASH standard [5], but tailored for LTE clients. From a high level, piStream enables an LTE client to monitor the cell-wide physical layer resource utilization status and instantaneously map it to the potential network bandwidth.

piStream's resource monitor scheme aims to be accurate, efficient, and deployable on LTE user equipment (UE). A straightforward way to obtain the resource allocation status is to decode the basestation's entire control channel[1] [6, 7]. However, this approach falls short of efficiency and scala-

[1]In LTE, UEs within the same cell share frequency/time resources allocated by the basestation in a centralized manner. Per-UE resource assignment is conveyed to each target UE over a dedicated control channel.

bility. It requires each UE to keep monitoring the control channel and decoding the control messages to all other UEs, for which the error detection mechanism specified in current LTE standard cannot take effect (Section 3.1). piStream addresses this challenge by taking advantage of the well organized LTE resource structure. Instead of decoding the downlink control messages dedicated to all other UEs, a UE only needs to inspect the signal energy on LTE radio resources to assess their occupancy.

After acquiring the amount of surplus radio resources at the physical layer, piStream scales up its measured throughput accordingly to obtain an estimation of the potential network bandwidth it can leverage, which is then exposed to the application layer to facilitate video rate adaptation. In this way, piStream overcomes a common limitation of a wide range of DASH protocols that are slow at exploring unused bandwidth [8–10].

From the DASH application perspective, to maximize bandwidth utilization while minimizing video stalling rate, the adaptation logic ideally needs to predict the future bandwidth evolution [11,12], which is challenging for an LTE UE. Instead of forecasting the elusive bandwidth value, piStream takes advantage of the invariant burstiness of traffic patterns in packet switching networks [13,14]. It estimates how likely the aggregated downlink traffic (or resource usage), and hence available bandwidth, is to remain at a similar level as the current one. It then makes a probabilistic decision to maximize video quality while minimizing the risk of stalling.

We validate the piStream design by tethering a software-radio that implements the PHY-informed bandwidth estimation mechanisms, to an LTE smartphone that implements the application-layer video adaptation scheme. Our piStream client prototype can directly play video in real-time from any server that follows the industrial MPEG-DASH standard [5]. We benchmark piStream's performance against a standard DASH player from GPAC[2], and three state-of-the-art DASH schemes that have demonstrated superior performance over commercial DASH players. These schemes include buffer-based adaptation (BBA [9]), optimization-based adaptation using historical throughput (FESTIVE [10]), and TCP-like bandwidth probing (PANDA [15]). Under a variety of experimental settings including time, location and mobility, piStream outperforms all other DASH schemes by achieving higher video quality and lower/comparable video stalling rate. Under typical static indoor environments, piStream achieves around 1.6× video quality (bitrate) gain over the runner-up (BBA) while maintaining a low video stalling rate close to 0%.

To our knowledge, piStream represents the *first* protocol to facilitate LTE adaptive video streaming using PHY-informed bandwidth estimation, which is evaluated via a *real-time* implementation. The specific contributions of piStream can be summarized as follows:

(i) We design a lightweight PHY-layer resource monitor (Section 3.1) and rate scaling mechanism (Section 3.2) that enables an LTE client to efficiently estimate available bandwidth, and facilitate application-layer protocols;

(ii) We propose a video adaptation algorithm that harnesses the PHY-informed bandwidth estimation, while probabilistically balances video quality and stalling rate, taking into account the LTE bandwidth variation (Section 3.3);

(iii) We conduct a real-time implementation of the piStream framework on a software-radio platform tethered to a smartphone (Section 4), and demonstrate its significant performance gains over four state-of-the-art protocols (Section 5).

piStream does not require any changes in existing cellular infrastructure or video streaming servers. Its PHY modules piggyback on the UEs' existing communication hardware, and can potentially be deployed via a firmware upgrade.

2. BACKGROUND AND MOTIVATION

In this section, we first provide a brief background of DASH, and then conduct a measurement study of the challenges in running DASH over the highly dynamic LTE channel.

A primer on DASH. DASH refers to the class of protocols (MPEG-DASH [5], Microsoft Smooth Streaming [16], Apple HLS [17], Adobe HDS [18], *etc.*) that adopt HTTP based adaptive video streaming. A DASH server splits a video into multiple segments with uniform playback time (typically 1 to 10 seconds). Each segment is encoded into multiple copies with discrete encoding bitrate levels and thus different sizes. Before a DASH video session starts, a client obtains an available bitrate map from the server. To download each segment, the client needs to send an HTTP request to the server, and specify the bitrate level it prefers for that segment.

DASH has gained broad interest owing to several salient advantages over the traditional server-controlled video transport protocols [19]. It eases deployment as HTTP video traffic can easily bypass middleboxes and firewalls, and can be supported by commodity web servers and CDNs. The use of stateless servers also simplifies load balancing and fault tolerance. Therefore, DASH is becoming the dominant Internet video streaming technology [2].

The client is fully responsible for executing DASH's adaptation logic, which should choose the video bitrate on a per-segment basis to maximize the Quality of Experience (QoE). An optimal bitrate choice can be made only if the current and future network bandwidth are known. Many existing DASH algorithms attempt to approximate this objective by estimating/predicting available bandwidth [10,12,15], or by balancing the buffer occupancy at a desired level [9]. However, these solutions render unsatisfactory performance in LTE networks as detailed in the following section.

Challenges in estimating the current network bandwidth. Typical DASH protocols [5,8,10] use the throughput observed from each video segment to estimate the available end-to-end network bandwidth. However, such estimations stay blow the bandwidth unless the segment size (and equivalently the video bitrate) is large enough to saturate the network pipeline. This is rarely the case in LTE, which bears large in-network buffers with hundred-millisecond scale end-to-end latency [20]. Therefore using throughput to guide video adaptation often underutilizes the network bandwidth.

To validate this problem in existing DASH protocols, we conduct DASH video streaming tests over Verizon LTE network (more details about our set up are available in Section 4). The DASH server is hosted by Akamai[3] and our

[2]GPAC [8] is a popular (over 3300000 homepage visits during 2014) open-source framework to generate and play DASH video data sets following the MPEG-DASH standard [5].

[3]http://dash.edgesuite.net/akamai/streamroot/050714/Spring_4Ktest.mpd

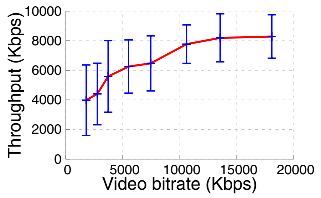


Figure 1: Throughput measurement depends on the traffic demand.

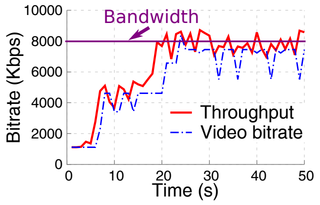


Figure 2: Throughput-based DASH adaptation converges slowly to bandwidth.

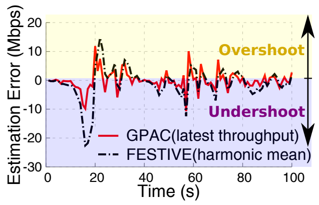


Figure 3: Existing DASH adaptation cannot follow the bandwidth variation.

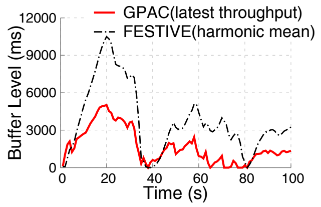


Figure 4: Poor bandwidth predictions drain out client's buffer and cause video stalls.

client is the GPAC player [8]. We run the tests in static environment during late night to ensure a relatively stable bandwidth.

We first disable the DASH adaptation by forcing the client to keep selecting a fixed video bitrate and repeat the tests for all bitrate levels in the server's DASH data set. Figure 1 plots the mean per-segment throughput under each video bitrate level. Error bars represent the *standard deviation* across all segments. As bitrate increases, measured throughput grows until a saturation point where it matches the available bandwidth. As a natural consequence, if the DASH client selects a video segment with low bitrate, it may experience a throughput far below the bandwidth. Using this throughput as bandwidth estimation, it proceeds to select a low video bitrate for next segment. This vicious cycle remains until the client opportunistically experiences a higher throughput owing to end-to-end throughput variation, but it will take a long time to eventually converge to the available bandwidth.

Then we enable the client's DASH adaptation under the same setting. Figure 2 illustrates the slow converging process as discussed above, where the blue curve shows the client's per-segment bitrate decisions and the red curve plots the resulting throughput. Consequently, the client takes around 25 seconds to converge to the bandwidth. Notably, a similar phenomenon has been observed in existing work [12] through trace-driven simulation.

It is worth noting that *(i)* The slow convergence is not caused by TCP. In one test, we manually switch the video bitrate from 1 Mbps to 8 Mbps and TCP (Cubic) only takes a few hundred milliseconds to ramp up; *(ii)* DASH clients usually pick a small *initial* segment size (video bitrate) to reduce the video loading time and build a sufficiently large buffer level to avoid future stalling [21], which exacerbates the slow convergence; *(iii)* LTE bandwidth of a client typically fluctuates more than shown in Figure 2 due to the competing traffic and mobility. In such cases, the slow convergence causes the throughput estimation to keep lagging behind bandwidth, which severely compromises the performance.

One may consider using TCP-like probing mechanism to explore available bandwidth [15]. However, video adaptation runs at segment-level (second-level) in contrast to TCP's packet-level (millisecond) adaptation. DASH cannot afford the frequent probing that pushes the throughput to the limit but causes frequent video stalling, especially for the highly dynamic LTE link. Besides, using very small video segments seems helpful but eventually causes more problems: Due to the large RTT of LTE networks (100 to 300 ms according to [20] and our measurements), the request delay from client to server will incur formidable overhead unless amortized by large segments [10].

Challenges in predicting future bandwidth. An estimation of *current available bandwidth* helps a client choose the best video quality if the bandwidth is relatively stable. Yet clients suffering from link dynamics, *e.g.*, cellular network clients, ideally needs to look ahead to predict *future bandwidth* [12]. To approximate the future bandwidth, most existing DASH protocols can be classified as two categories: *(i)* those using the bandwidth estimation of the latest video segment as the bandwidth prediction for the next segment, *e.g.*, GPAC's player [8]; *(ii)* those using smoothed historical bandwidth (for example, harmonic mean over a time window) as the prediction, *e.g.*, FESTIVE [10]. Such strategies have proven effective in wireline networks, but perform poorly in LTE networks due to frequent bandwidth variations.

We then inspect these two widely used strategies to reveal their ineffectiveness. To isolate the aforementioned bandwidth underestimation problem, we assume current and historical (but not future) bandwidths are known exactly. Specifically, we collect a time series of available LTE bandwidth by measuring a saturated Iperf [22] session, and then perform trace-driven emulation for the above two strategies.

Figure 3 plots the time series of bandwidth prediction error at segment-level granularity. The error falls between -20 Mbps to 10 Mbps, and thus the selected video quality can deviate wildly from the optimal one. When a severe overestimation occurs, even though infrequently, the accumulated video buffer can quickly drain off (Figure 4), resulting in video stalls. The harmonic-mean based prediction causes less video stalling, but at the cost of severe bandwidth underutilization and thus video quality degradation.

3. piStream DESIGN

piStream is a cross-layer framework to address the above challenges for adaptive video streaming over LTE. It incorporates several client-side innovations and remains compatible with any MPEG-DASH servers [5]. piStream consists of three main design components (Figure 5). A *radio resource monitor* (RMon) estimates the amount of unused radio resources by sensing the LTE downlink channel. A *PHY-informed rate scaling* (PIRS) scheme translates the utilization statistics into the current available network bandwidth that can be legitimately exploited. Finally, an *LRD-based video adaptation* (LVA) algorithm estimates how long the current available bandwidth is likely to last, and accordingly selects the bitrate for the next video segment to maximize the QoE.

3.1 Radio Resource Monitor (RMon)

The radio resource monitor (RMon) acts as a PHY-layer daemon in each UE that monitors cell-wide utilization of

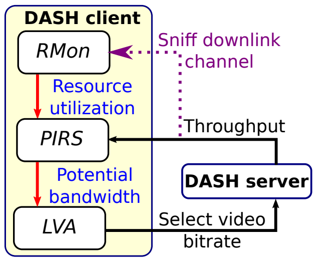


Figure 5: piStream system architecture.

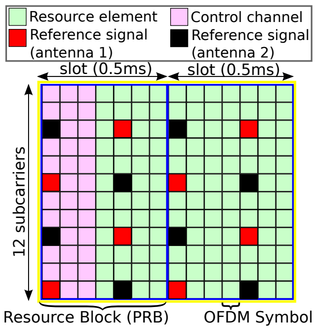


Figure 6: LTE resource allocation example for a two antenna basestation.

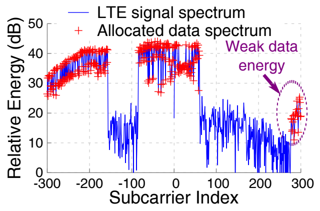


Figure 7: Energy-based resource utilization monitor is feasible but challenging.

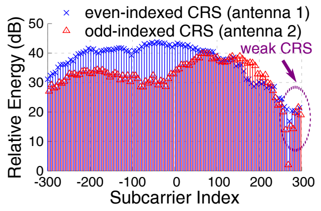


Figure 8: Reference signals can capture frequency selectivity and antenna diversity.

radio resources (referred to as Physical Resource Blocks, or PRBs). RMon needs to be highly reliable, yet simple enough to be executable in real-time and easily compatible with current UE's hardware. Below we present the background, challenges and our design to meet these goals.

3.1.1 A Primer on LTE Resource Allocation

LTE downlink channel is divided into fixed time frames, each spanning 10 ms. Each frame is further divided into 10 sub-frames, each spanning 1 ms and containing two slots (Figure 6). The basestation transmits each subframe using OFDM (Orthogonal frequency-division multiplexing), which divides available radio resources into grids in both time and frequency domain. Each grid cell (spanning 15 kHz $\times$ 66.7 μs) is called a *resource element* (RE). A basestation awards radio resources at the granularity of *physical resource block* (PRB) comprising multiple resource elements. Each PRB spans half-a-subframe (*i.e.*, 0.5 ms) in time and 12 OFDM subcarriers (*i.e.*, 180 kHz) in frequency domain.

The basestation dynamically allocates non-overlapping sets of PRBs to different UEs, depending on their channel conditions and traffic demands. The allocation strategy is vendor-specific. However, typical basestations enforce some form of proportional fairness, which ensures a balance between UEs with the best channel quality and those consuming the most resources. The per-UE resource assignment is conveyed to its target UE over a dedicated downlink control channel.

3.1.2 Why Existing LTE Sniffers are Insufficient?

piStream calls for a highly efficient radio resource monitor that is compatible with the UE's hardware. There exist a few platforms that sense the resource allocation by decoding LTE's downlink control channel (Section 3.1.1). Yet none of them meet piStream's design goal.

(i) QXDM [23], a monitor available for UEs with a Qualcomm LTE modem, can only analyze the radio resource utilization of a single UE rather than cell-wide information.

(ii) LTEye [6] can provide cell-wide resource utilization by decoding the downlink control information for all UEs. However, it requires a UE to keep monitoring the downlink control information to all other UEs, which is unspecified in LTE standard and incurs significant modifications to the UE's PHY layer. Besides, the decoding overhead increases with the total number of UEs, hence it is unscalable. Finally, the LTE error detection mechanism[4] does not work for LTEye since LTE design never considered that a UE will attempt to decode other UE's downlink control information. Thus, LTEye works well only when the wireless channel induces almost zero bit errors and hence the resource allocation can be deciphered without CRC.

(iii) OpenLTE [7] provides a software-radio based LTE downlink demodulator, which has the same problem as LTEye when decoding the resource allocation information.

In summary, to obtain cell-wide resource (PRB) allocation status, decoding the downlink control channels is not a deployable solution as it seems. It is unscalable, entails significant hardware modifications, and its results are unreliable.

3.1.3 Energy-based Radio Resource Monitor

piStream adopts an energy-based spectrum monitor called RMon to assess cell-wide PRB allocation status without decoding the control channel. RMon offers a robust, energy-efficient and readily-deployable solution to expose the PHY-layer downlink resource assignment to a UE.

Despite its conceptual simplicity, the unique features of LTE PHY layer bring several challenges to RMon's design. In particular, frequency selectivity makes it difficult to set up a threshold to examine whether an LTE resource element is allocated. Besides, wide adoption of multi-antenna basestations exacerbates the variation of energy levels across frequency/time, since the antennas may have diverse channel gains to the UE.

Figure 7 showcases these challenges through one OFDM data symbol over a 10MHz LTE downlink channel with 600 usable subcarriers (resource elements). We obtain the ground truth of subcarrier allocation by running the LTEye sniffer [6] intentionally close to the basestation to ensure almost zero bit error in decoding. We see that the energy on allocated subcarriers is generally 20dB higher than the noise floor. However, the energy levels of different subcarriers vary by up to 33 dB, some even falling below the noise floor. According to LTEye's decoded control information, the basestation enforces transmit diversity using 2 antennas around subcarrier index -300 and 0, which causes a wild energy variation (about 15dB).

In the rest of this section, we detail the design of our resource monitor RMon to meet such challenges.

Obtaining per-subcarrier energy-level statistics. The resource monitor RMon can be considered as a simple intercepting module to a standard LTE UE which already performs frequency-time synchronization to the basestation, and runs an FFT on each OFDM symbol to obtain the energy levels across subcarriers. RMon only requires the UE

[4]Since the CRC of downlink control information (DCI) is scrambled by each UE's physical layer ID [24] which is transmitted via encrypted upper layer channels and only available to the UE itself, a UE cannot legitimately obtain the CRC of other UE's DCI.

to expose such per-subcarrier energy-level statistics. Thus, it does not require any additional communication hardware. Although this is not yet feasible in most current LTE hardware (which usually only exposes total energy as signal strength indicator), we believe the potential of cross-layer design provides compelling reason to enable it. In fact, the Qualcomm QXDM already exposes to each UE its per-subcarrier energy level [23]. In many recent WiFi chipsets, per-subcarrier statistics are already available to higher layers [25, 26].

Locating resource elements available for downlink data. When monitoring the resource assigned to downlink data, the control signals and reference signals from the basestation should be excluded. Such signals are scattered across the resource grid (Figure 6). Fortunately, they can be located and then isolated owing to LTE's well defined resource structure.

Specifically, RMon first senses the physical channel width based on the frequency domain spectrum, which can be converted to N_{rb}^{DL}, the total number of physical resource blocks (*PRBs*) available in one LTE time slot. It then locates the OFDM symbols assigned to *control signals* based on the control format indicator (*CFI*) field specifying the location of control symbols. Finally, RMon locates the resource elements allocated to *reference signals*, which are mapped from the physical cell ID (*PCI*) following the LTE specification [24]. The PCI is in turn obtained during the UE-to-basestation synchronization procedure.

RMon is designed to operate well in cases where LTEye fails (Section 3.1.2). It only relies on robust control informations: CFI and PCI are available to all UEs, and are designed to be very robust by using 32 bit code to carry 2 bit information (CFI) or signal correlation techniques (PCI).

Evaluting resource element occupancy. Since not all available resource elements are occupied by actual transmission. RMon employs an energy detector to single out active resource elements. Since the resource elements' energy levels depend on link distance, frequency selectivity, and multi-antenna diversity, it is infeasible to use a constant energy threshold to examine resource element occupancy. To remedy this problem, RMon adopts a dynamic threshold customized for each LTE resource element.

Specifically, to combat the signal variation caused by pathloss and frequency selective fading, we note that for each resource element on subcarrier k and OFDM symbol t, there exist two persistent reference signals within the subcarrier range $[k-2, k+2]$ and symbol range $[t-2, t+2]$. When active, the resource element should have similar energy level as the nearby reference signals owing to channel coherence. As an example, Figure 8 depicts the reference signal energy corresponding to the data signals in Figure 7. We let resource elements in the first half of a subframe refer to the reference signals inside the 1st symbol of this subframe and the 2nd half refer to the 4th symbol. We use t_r to denote the global index of the nearest symbol containing reference signals. Suppose k^+ is the nearest reference signal subcarrier indexed higher than k, and k^- the one indexed lower than k. If k is an even number, we have $k^+ = k+1$ and $k^- = k-2$. Otherwise $k^+ = k+2$ and $k^- = k-1$ [24].

Moreover, for a multi-antenna basestation, the reference signals from different antennas are perfectly separated in frequency domain (Figure 8). Regardless of the basestation's transmission mode[5] (*e.g.*, SISO, transmit diversity, and open-loop MIMO), a subcarrier's energy should be no less than the energy from the transmit antenna with the worst channel gain, which can be approximated by the smallest energy on the two close-by reference signal subcarriers.

Consequently, for each subframe, RMon specifies the energy-detection threshold $\tau(k,t)$ for an LTE resource element (RE) on subcarrier k and in the symbol at time t as:

$$\tau(k,t) = \alpha \min(e(k^+, t_r), e(k^-, t_r)) \qquad (1)$$

where $e(k,t)$ is the measured energy of the RE on subcarrier k in the symbol at time t. Since the RE to inspect at (k,t) and its nearest reference signal REs at (k^+, t_r) and (k^-, t_r) are very close to each other in both time and frequency domain, there will only be slight channel diversity between them, and hence we can use a constant factor $\alpha = 0.8$ in Eq. (1) to safely accommodate such slight channel diverstiy.

Sanity check based on PRB resolution. The OFDM PHY layer naturally allows energy monitoring on each resource element. However, the smallest resource allocation unit in LTE is one PRB, which spans 12 consecutive subcarriers. RMon leverages this structure to further combat variation of subcarrier energy due to narrow band interference, noise, or deep fading. It computes the harmonic mean energy-level within each 12-subcarrier window, and then compares it with the threshold $\tau(k,t)$ to decide whether the PRB is allocated, thus filtering out outlier subcarriers. In this way, we obtain the total number of allocated PRBs.

Output the resource utilization ratio. After all steps above, RMon is ready to compute its output, the resource (PRB) utilization ratio u over a time duration of M subframes, *i.e.*, M ms, by comparing the number of allocated resource elements to the total number of resource elements available for downlink data:

$$u = \frac{N_{alloc}}{2N_{rb}^{DL}M} \qquad (2)$$

where N_{alloc} is the number of allocated PRBs over duration M, N_{rb}^{DL} is the number of PRBs in each $0.5ms$ time slot. Note that a subframe has 2 time slots. Intuitively, M should be small enough to enable responsive online adaptation algorithm. However, a too small M leads to very unstable average PRB utilization since the traffic load and per-subframe resource allocation are sporadic at a small time scale. In piStream, M is default to 200 ms, which can effectively capture the relationship between PRB utilization and traffic load, while remaining responsive to traffic variation.

The PRB utilization output of RMon is then leveraged to enable the PHY-informed rate scaling (PIRS), which is detailed in the following section.

3.2 PHY-Informed Rate Scaling (PIRS)

Given the PRB utilization, instead of using the throughput-based adaptation to underutilize the bandwidth (Section 2), a piStream client can scale up its measured throughput to obtain an estimation of its *potential bandwidth* at this time.

[5]Closed-loop MU-MIMO beamforming is not supported in typical commerical LTE networks yet, and hence we leave the design to handle this particular transmission mode to future work. Even if MU-MIMO frames are used, current piStream UE can at least detect this based on the length of Downlink Control Information (DCI) and then revert to conventional DASH mode.

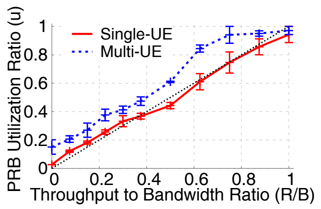


Figure 9: PRB utilization versus bandwidth utilization.

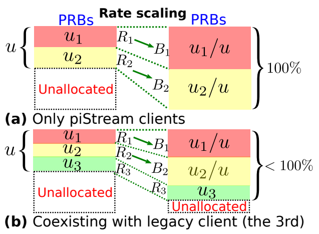


Figure 10: PHY-informed rate scaling example.

In this section, we detail the rationale and operations of such a PHY-informed rate scaling (PIRS) mechanism.

Relation between PRB and bandwidth utilization. PIRS builds on a hypothesis that *the PRB utilization of a client grows linearly proportional with throughput until it saturates network bandwidth.* Behind this hypothesis are two observations: *(i)* The end-to-end network bandwidth is bottlenecked by the LTE access link, which has been validated in existing measurement [4]; *(ii)* The access link capacity is determined by the bit-rate and the cell-wide PRB utilization. However, since the basestation already performs rate adaptation to maximize the link bit-rate, the additional bandwidth available should only depends on the available PRBs. Below we validate this observation through experiments.

Our experiment is conducted in late night with relatively stable network condition. We first measure the end-to-end bandwidth (B) similarly to Sec. 2, and then initiate a fixed-rate TCP stream from the server to one UE and record its throughput (R). Meanwhile, we use RMon to measure the cell-wide PRB utilization (u) per 200ms, and compute the harmonic mean over an entire 300s Iperf session. We also inspect a multi-UE test case, where the measurements are collected during busy hours so that our UE coexists with multiple UEs in this commercial LTE network.

Figure 9 shows that the bandwidth utilization R/B almost equals the PRB utilization u for the singe-UE test case. Therefore, *when the downlink is dominated by a single UE, the cell-wide PRB utilization directly reflects the UE's bandwidth utilization.* For the multi-UE case, we see that u tends to be higher than R/B. This is because the PRB utilization comprises the traffic load of not only our target UE, but also other UEs within the same cell. Meanwhile, the target UE's B decreases due to downlink resource sharing. However, a linear relation still holds before u saturates, *i.e.*, *cell-wide PRB utilization increases linearly with a UE's throughput to bandwidth ratio until the utilization approaches 100%.* This essentially validates our hypothesis.

Rate scaling based on PRB utilization. Based on the above observations, we derive the following principle for a UE to scale up its measured throughput to obtain an estimation of its current bandwidth[6]: *Suppose current cell-wide PRB utilization ratio is u, and the UE's current throughput is R, then at least a throughput of R/u can be supported without over-utilizing PRB.* Below we conduct a case-by-case analysis of this principle.

(i) piStream client only (no legacy clients). First, suppose the basestation serves only one client who can thus enjoy all surplus PRBs. Given that the current PRB utilization u supports the client's current throughput R, the maximum achievable throughput when the client is granted all PRBs should equal:

$$B = R/u \tag{3}$$

Now, consider the more complicated cases when there are multiple piStream clients in the network, they will detect the same cell-wide PRB utilization u. Suppose client i is allocated an actual fraction u_i of the total PRBs, then it will scale u_i by $1/u$ accordingly. Even if they happen to scale up their rates at the same time, as shown in Figure 10(a), the resulting PRB utilization will not exceed 100% since $u_i/u + (u - u_i)/u = 1$. Therefore, all piStream clients can improve their rates without overshooting the bandwidth, as illustrated in Figure 10(a).

(ii) piStream client(s) coexisting with conventional client(s). In this case, Figure 9 has shown that the PRB utilization stays above the bandwidth utilization. Hence, a client can still scale up its throughput following Eq. (3) without overshooting its bandwidth. More specifically, suppose piStream client i is allocated a fraction u_i of the total PRBs. After rate scaling, the cell-wide PRB utilization becomes $\sum_i u_i/u + (u - \sum_i u_i) \leq \sum_i u_i/u + (u - \sum_i u_i)/u = 1$. A simple example is illustrated in Figure 10(b). Although in this case the total PRB utilization may be less than 100% after a single rate scaling, it can quickly ramp up to 100% after several successive rate scalings.

For the legacy clients in this scenario, firstly, if their traffic demands do not increase, they will not be affected by piStream at all because piStream clients only attempt to leverage the unused PRBs. Second, if the traffic demand of any legacy client increases after the PRB utilization reaches 100%, the LTE basestation will invoke its proportional fairness scheduler to arbitrate the resource contention. Note that the same problem will happen even in a network with only legacy clients, and hence this is a general issue, not the limitation of our design.

(iii) New client joining after full PRB utilization. If a new client joins the network when the existing ones have exhausted the PRBs, the LTE basestation's proportional fairness scheduler will take effect so that the new client gradually gains more PRBs. When the basestation transfers PRBs to the new client, the cell-wide PRB utilization stays at 100%. Therefore, existing piStream clients will not scale up their rates any more. Furthermore, with a proportionally fair scheduler, the basestation will allocate less resource to them, so the DASH senders serving existing piStream clients will slow down and the saved resources are gradually "transfered" to the new client.

Ultimately, when PRB is fully utilized, the basestation's scheduler will be responsible for *ensuring fairness* among clients based on their link condition. As a client-side mechanism, piStream's rate scaling does not affect the basestation's fair scheduler. However, it does help the clients to quickly reach the full PRB utilization state to use the network resource more effectively.

3.3 LRD-based Video Adaptation (LVA)

Given an estimation of current available bandwidth (the output of PIRS module), a piStream client runs a probabilistic LRD-based video adaptation (LVA) algorithm to select the rate for each next video segment, and balance the tradeoff between video quality and stalling risk.

Theoretical work [12] has shown that DASH performance can improve significantly if *future bandwidth* is known. How-

[6]This paper focuses on the general cases where the LTE link bottlenecks the end-to-end network bandwidth [4]. We leave other corner cases to future work.

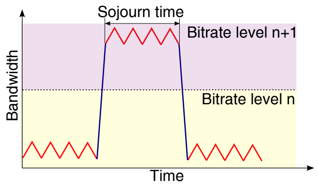


Figure 11: A simple example for the bandwidth sojourn time.

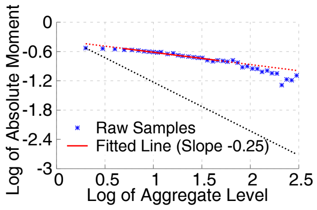


Figure 12: A Hurst parameter 0.75 indicates the long range dependency.

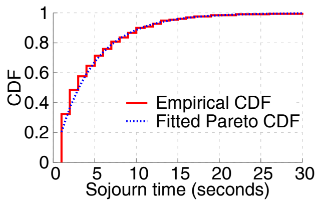


Figure 13: Sojourn time distribution fits a Pareto distribution.

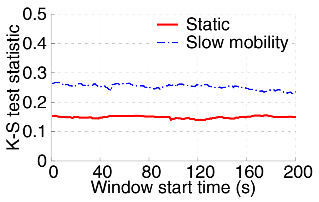


Figure 14: K-S test statistic. (difference between the empirical and the fitted CDF)

ever, the wireless channel fading and random traffic demand make bandwidth prediction very challenging in LTE.

Instead of attempting to predict the exact value of future bandwidth, the piStream client takes advantage of the invariant burstiness, or long-range dependency (LRD) of LTE downlink traffic. Based on a model of its LRD profile, it estimates how long its current bandwidth-level is likely to remain consistent, and then chooses the per-segment video bitrate in a probabilistic way.

Modeling sojourn time of network bandwidth. The LRD property of Internet and local-area network traffic has been well established [13, 14]. The downlink resource sharing and large buffers in LTE intensifies such traffic patterns, which naturally causes the available bandwidth to exhibit LRD. To model LRD, we first define the *sojourn time* T_s as the period between two consecutive bandwidth jumping events, as illustrated in Figure 11. The bandwidth (estimated by the PIRS component) is considered to jump to another level when the highest video bitrate below the bandwidth (*i.e.*, the quantized bandwidth) changes.

The arrival of a sequence of bandwidth variations can be considered as a renewal process, with sojourn time equal to the inter-arrival time. As verified later in this section, the sojourn time distribution is relatively stable and follows a long tail or power law distribution. We use the widely adopted Pareto distribution to model such behavior:

$$\mathbb{P}\{T_s > t\} = \begin{cases} (\frac{\alpha}{t})^{\beta}, & t \geq \alpha \\ 1, & \text{otherwise} \end{cases} \quad (4)$$

where α and β are referred to as the *scale* and *shape* parameter for the Pareto model.

Based on the Pareto model, the piStream client estimates the probability $\boldsymbol{p}(t)$ that its bandwidth will sojourn at current level for one more video segment of duration Δt, conditioned on the observation that it has already been staying at this level for a duration t:

$$\boldsymbol{p}_s(t) = \mathbb{P}(T_s > t + \Delta t | T_s > t) = (\frac{t}{t + \Delta t})^{\beta} \quad (5)$$

We define $\boldsymbol{p}_s(t)$ as the *sojourn probability*. Eq. (5) implies that $\boldsymbol{p}_s(t)$ increases with t. In other words, the UE's bandwidth is more likely to sojourn at its current level if it has been staying there for a longer duration. Note that α disappears in this equation since typical video segment length $\Delta t > \alpha$, and thus we only use the first case in Eq. (4) to compute the conditional probability.

LRD model validation. We first use experiments to verify the LRD pattern of the LTE downlink bandwidth. We collect 5 time series of bandwidth values by running saturated Iperf sessions, each lasting 300 seconds. We use the absolute moment method [27] to estimate the Hurst parameter H for the time sequence of bandwidth. When $0.5 < H < 1$, the sequence has long range dependency, and the degree of long range dependency increases as H gets closer to 1. Figure 12 gives the log-log plot of the absolute moment. The slope of the regression line is $k = -0.25$, and accordingly the Hurst parameter $H = 1 + k = 0.75$ [27], indicating strong long range dependency of the LTE bandwidth.

We further verify the goodness-of-fit for the Pareto model using the same bandwidth time series. To incorporate the impact of mobility, we collect an additional set of bandwidth trace in a car during slow driving with constant speed and direction. To process the time samples of bandwidth, we run a sliding window with size of 60s across all the samples and compute the Pareto fit of the samples in the window. The empirical CDF of bandwidth sojourn time and its Pareto fit in a particular time window under static environment are plotted in Figure 13. We see that more than 40% of the bandwidth level lasts longer than 5 seconds, which leaves enough room for the LVA design to take effect. Moreover, 90% of the bandwidth levels last no more than 10s. Hence, existing DASH design (*e.g.*, FESTIVE [10]) that uses a long smoothing window (typically 20s) may not adapt fast enough to follow this bandwidth variation.

Besides, in Figure 14, we report the Kolmogorov-Smirnov Test (K-S test) statistic $D = \sup_x |F_n(x) - F(x)|$ in each sliding time window, which uses the difference between the empirical CDF $F_n(x)$ and the fitted CDF $F(x)$ to characterize the goodness-of-fit. It is worth noting that a high mobility level may reduce the goodness-of-fit of our model. We will discuss the corresponding design to handle this issue in the end of this section.

From the results we can observe that: *(i)* The K-S test statistics are small, indicating small difference between empirical CDF and fitted CDF. *(ii)* The K-S test statistics are stable in time domain, which suggests the stability of the Pareto model parameters. *(iii)* The goodness-of-fit in high mobile environment is worse than that in static environment due to the bandwidth variations caused by channel fluctuation which are not captured by the Pareto model.

It should be noted that prior work [28] reported Poisson (memoryless) distribution of LTE downlink traffic. However, the experiments only examines short-term, packet-level inter-arrival time distribution, which does not conflict with the bursty traffic pattern at larger time scale observed here[7].

LRD-based adaptation design. The LVA algorithm proceeds as summarized in Algorithm 1 and detailed below.

(i) First, the client periodically (empirically set to 60s) applies a maximum-likelihood regression [29] to recent bandwidth samples to estimate the Pareto parameter β in Eq. (4).

[7]Based on our private correspondence with the authors.

Algorithm 1 LRD-based video adaptation (LVA).

```
Input: $B_n$: Bandwidth estimation of the $n$-th segment
       $L_{new}$: Bitrate level after bandwidth variation
       $L_{old}$: Bitrate level before bandwidth variation
       $t_0$: When last bandwidth variation happened
       $Q(\cdot)$: Quantize bandwidth to video bitrate
Output: $V_{n+1}$: Video bitrate for the next video segment
 1: /*Finish downloading the $n$-th segment at $t$*/
 2: Update Pareto parameter $\beta$ using historical bandwidth
 3: if $Q(B_n) == L_{new}$ then /*Check bandwidth variation*/
 4:     $T_s = t - t_0$; /*update the sojourn time $T_s$*/
 5: else
 6:     $T_s = 0$; $t_0 = t$; /*reset the sojourn time*/
 7:     /*update the bitrate levels*/
 8:     $L_{old} = L_{new}$; $L_{new} = Q(B_n)$;
 9: Update bandwidth sojourn probability following Eq. (5)
10: if rand $< \boldsymbol{p}_s$ then /*Stochastic video bitrate selection*/
11:     $V_{n+1} = L_{new}$; /*switch to the new bitrate level*/
12: else
13:     $V_{n+1} = L_{old}$; /*stay at the old bitrate level*/
```

(ii) Meanwhile, the *rate scaling module* continuously reports current bandwidth estimation to the adaptation logic. When the bandwidth (quantized to match the video bitrate levels) experiences a jump from L_{old} to L_{new} at time t_0, the client starts to count the sojourn time T_s, the time that the bandwidth has been staying at the new level L_{new}. Based on Eq. (5), the client then computes the bandwidth sojourn probability $\boldsymbol{p}_s(t)$: For the next video segment the network bandwidth will stay at the new level L_{new} with probability $\boldsymbol{p}_s(t)$ or go to any other level with probability $1 - \boldsymbol{p}_s(t)$.

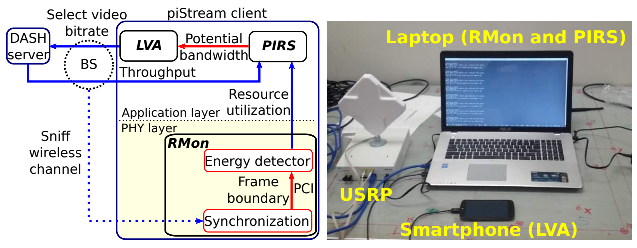
(iii) Finally, the client performs stochastic video bitrate selection based on the sojourn probability $\boldsymbol{p}_s(t)$. With lower sojourn probability, the new bandwidth is less likely to last for the next video segment, hence the client is reluctant to switch to the bitrate matching the new bandwidth. In contrast, higher $\boldsymbol{p}_s(t)$ indicates a possibly more stable bitrate level thus the client is more inclined to switch.

To reduce the impact of a probabilistically wrong decision, LVA performs a sanity check based on the client's current buffer level. When the buffer level is low (we empirically set the threshold to 2000ms), even if a rare mistake can incur intolerable buffer underrun. Thus, the client acts more conservatively and reduces the video bit rate upon every bandwidth decrease it experiences.

Handling mobility. The bandwidth variations caused by channel fluctuations in mobility are unpredictable and not captured in our bandwidth sojourn time model. This leads to reduced goodness-of-fit in mobile environment and thus less accurate adaptation. To handle mobility, we use the accuracy of our model to guide the aggressiveness of the adaptation. More specifically, the client proactively evaluates the level of mobility based on the goodness-of-fit of bandwidth samples in current sliding window. A worse goodness-of-fit indicates higher mobility, which then forces the client to act more conservatively, *i.e.*, the client becomes more likely to reduce the video bitrate upon bandwidth decrease even when the sojourn time model suggests this decrease is temporary.

4. IMPLEMENTATION

We prototype piStream on a cross-layer testbed comprised of a PHY-layer resource monitor and a PHY-aware DASH layer. Figure 15(a) and (b) illustrate the implemented system architecture and the prototype testbed, respectively. Since the Android phone lacks native support for many of the libraries essentially needed to implement piStream and also cannot be directly connected to the USRP, we implement most piStream components on the laptop as shown in Figure 15(b). Note that this does not affect the fidelity of our evaluation because the phone is still the LTE client and all traffic goes through the LTE network.



(a) Implemention architecture **(b)** piStream testbed

Figure 15: piStream implementation.

Real-time LTE radio resource monitor (RMon). We implement piStream's *real-time* resource monitor over the USRP N210 software equipped with WBX daughterboard and an antenna covering the 690-960/1710-2620 MHz LTE band. The resource monitor is executed on USRP's PC host as a C module. It processes the LTE downlink signal samples captured by the USRP, and extracts the cell-wide PHY resource allocation information, following the frequency-domain energy-detection algorithm in Section 3.1. The resource (*i.e.*, PRB) allocation statistics are computed over a moving window of 200 ms, and passed on to the upper layers online. Since this 200ms time granularity for PRB utilization output is much smaller than typical video segment length in DASH (2000 ms in our experiments), our implementation can perform DASH adaptation in real time. We average the PRB utilization statistics in each segment duration before performing rate-scaling to its throughput.

Our resource monitor RMon essentially implements major PHY-layer primitives that exist in every UE, plus the customized energy detector in piStream. Specifically, we have implemented the following crucial sub-components following the LTE standard [24]:

(i) Synchronization. The *synchronization* block enables a UE to read the physical cell ID (PCI) of the current cell and identify the downlink frame boundaries.

(ii) Signal energy detection. The *energy-based data signal detector* in RMon realizes fast and robust monitoring of PRB allocation. The frame boundaries provided by the synchronization block are used to determine the starting points of data symbols, so that we can perform FFT to estimate the frequency-domain resource allocation. Meanwhile, as detailed in Section 3.1, the PCI is used to locate the cell-specific reference signals (CRS) and determine the energy detection threshold, which varies following the CRS energy for different PRBs to accommodate the frequency selectivity.

PHY-aware DASH client. We implement piStream's adaptation logic by extending the DASH player from the GPAC [8] project. To interface the extended DASH layer with the RMon module, we tether an LTE smartphone (via USB) and a USRP radio (via Ethernet) to the same laptop PC host. Then we expose the PHY resource utilization statistics to the DASH player by setting up a named pipe between the resource monitor RMon and the DASH player. To

Algorithm name	*BBA*	*FESTIVE*	*PANDA*	*GPAC*	*piStream*
Bandwidth estimation	N/A	Harmonic mean throughput	N/A	Latest throughput	PIRS & LVA
Buffer knowledge	Yes	No	No	No	Yes
PHY information	No	No	No	No	Yes

Table 1: Implemented DASH algorithms.

leverage the PHY resource utilization statistics, we replace the default DASH adaptation logic in GPAC's DASH player with our implementation of the PHY-informed rate scaling (PIRS) and LRD-based video adaptation (LVA) algorithms in piStream.

DASH server. We set up an MPEG-DASH server with high bandwidth (66 Mbps measured with Iperf over a wireline network) on Amazon EC2 using Apache 2. This follows typical scenarios where the LTE downlink, instead of the wireline backbone, acts as the bottleneck for video streaming. The server hosts multiple DASH video data sets with typical $2s$ video segment length.

Benchmark protocols for comparison. To benchmark piStream's performance, we implement state-of-the-art DASH adaptation protocols including BBA [9], FESTIVE [10], and PANDA [15] in the GPAC's DASH player [8] by replacing its default video rate adaptation algorithm which quantizes throughput of the latest segment as the bitrate selection. All implemented algorithms are summarized in Table 1.

(i) BBA denotes a broad class of buffer-based algorithms proposed in [9]. Among these variants, we implement BBA-2, a hybrid algorithm which uses throughput to control video bitrate in the startup phase. After reaching the steady state, the client then maps its current buffer level to the video bitrate selection using a linear function.

(ii) FESTIVE algorithm selects video bitrate based on the harmonic mean of historical throughput measurements. As recommended in [10], our implementation adopts the averaging window over 20 segments and 30 seconds target buffer size. We also implement FESTIVE's stateful bitrate selection, *i.e.*, we delay the switch to bitrate level k by k segments, but decrease the bitrate level immediately if necessary.

(iii) PANDA allows a client to keep increasing its video bitrate to probe the bandwidth until observing a throughput decrease, which is taken as a congestion indicator. Then it reduces video bitrate based on the difference between bitrate and throughput of the last video segment. Our implementation exactly follows the default PANDA parameters in [15], the parameter k controlling the rate reduction speed is set to 0.14, which means slow rate reduction after a congestion.

5. EVALUATION

In this section, we first validate each component in piStream, including the real-time monitor (RMon), PHY-aware bandwidth scaling (PIRS), and LRD-based video adaptation (LVA) (Section 5.1). Then we combine everything together and perform a thorough system-level test under various network conditions (Section 5.2). All experiments (except one with trace-driven emulation in Section 5.1.3) are conducted over real-world Verizon LTE network on channel 13 (746-756MHz) using Samsung Galaxy Nexus smartphones. They also use the same server and DASH data sets specified in Section 4.

The performance evaluation focuses on two major QoE metrics [30]: *(i)* The *average video bitrate* represents video quality. It is defined as the mean video bitrate over all streamed video segments. *(ii)* The *video stall rate* represents playback smoothness. It is defined as the percentage

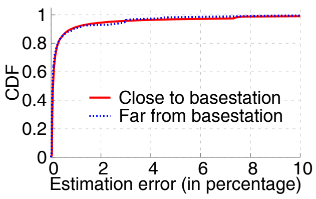


Figure 16: Our resource monitor RMon outputs accurate resource utilization.

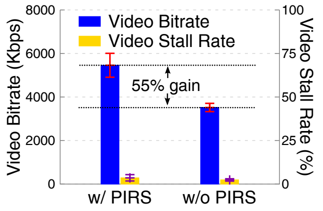


Figure 17: PIRS component alone improves the performance.

of video stalls, *i.e.*, the number of video segments that stall the video divided by the total number of streamed video segments to obtain the video stall rate. We obtain these metrics by logging the bitrate of every video segment and the buffer level on the client testbed in real time.

5.1 Micro-benchmark

5.1.1 Validating the Radio Resource Monitor (RMon)

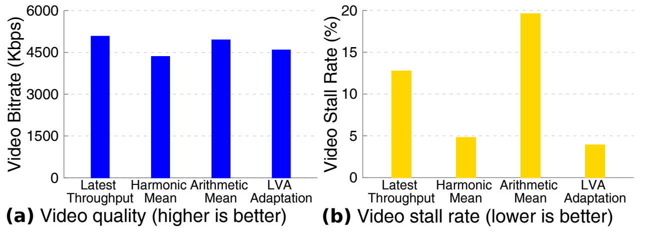
We first examine how accurate piStream's resource monitor RMon can estimate the PRB utilization statistics. We evaluate the PRB utilization (in percentage) for a single-user video streaming. Meanwhile, we record the bandwidth utilization (in percentage). As discussed in Section 4, LTE downlink channel is usually the bottleneck between server and client. Hence, if RMon is accurate, its PRB utilization output should equal bandwidth utilization for this single user. Estimation error is defined as the difference between PRB utilization and bandwidth utilization. The experiments are performance in static environment during idle hours so that the bandwidth utilization is *w.r.t.* the same bandwidth. We repeat the experiments under two link conditions, created by moving the client close by and far from the basestation while ensuring connectivity. In each case, we compute RMon's estimation error over 200 seconds and plot its CDF.

Figure 16 shows that RMon is highly accurate — it produces the same output as the ground truth for more than 90% of the time. In case it errs, it differs by no more than 10%. In addition, its accuracy is only slightly affected by link condition. Therefore, its PRB statistics can be reliably used by piStream's higher-layer modules. In our system tests (Section 5.2), we will demonstrate the synergistic performance under a variety of conditions including node mobility.

5.1.2 Validating the PHY-informed Rate Scaling (PIRS)

To verify the eventual impact of PHY-informed rate scaling, we evaluate how much performance improvement it can achieve in DASH video steaming. In this experiment, the RMon keeps sniffing the LTE downlink channel during DASH video streaming and reports the PRB utilization to the DASH client periodically. The client then scales up its bandwidth estimation based on the latest PRB utilization and directly sets the quantized bandwidth estimation as the bitrate for the next video segment. To exclude the influence of downlink bandwidth variation, we test a single static client during late night for 10 consecutive runs.

Figure 17 compares the performance of using PIRS or simply the throughput for video bitrate adaptation. We see that PIRS module alone improves the average video quality by 55% at the cost of only 1% increase in the number of video stalls. Therefore, PIRS can effectively overcome the



(a) Video quality (higher is better) **(b)** Video stall rate (lower is better)

Figure 18: Trace-driven emulation with perfect knowledge of current & historical (but not future) bandwidth.

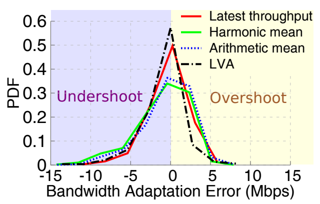


Figure 19: LVA has the lowest overshoot prob.(18% **lower than the runner-up)**

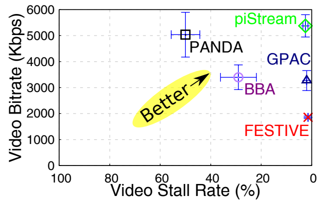


Figure 20: Average video bitrate & video stall ratio for a static client.

slow ramp-up behavior typically seen in throughput-based bandwidth estimators (Section 2).

5.1.3 *Validating the LRD-based Video Adaptation (LVA)*

In this micro-benchmark, we compare LRD-based video adaptation with several widely adopted DASH adaptation algorithms: *(i)* Latest-throughput-base adaptation, which uses bandwidth experienced by the latest video segment as the expected future bandwidth. *(ii)* To handle the bandwidth variation, adaptation based on the smoothed historical bandwidth is also widely used. A straightforward solution is to compute the arithmetic mean. *(iii)* Harmonic mean based bandwidth smoothing has demonstrated superior performance [10] than the arithmetic mean since it is more robust to larger outliers.

We use trace-driven emulation only for this micro-benchmark to isolate the effect of inaccurate bandwidth estimation in other algorithms. Instead of using the C++ implementation as in all other real-world experiments, we implement the algorithms again in MATLAB. The DASH video set used here is the same as that in all other real-world experiments. The transmission time is emulated within MATLAB based on the video segment size and the LTE downlink bandwidth in the collected traces. The traces contain a time series (at second level time granularity) of available LTE downlink bandwidth during busy hours for the target UE, collected by running a saturated Iperf TCP session. All DASH algorithms are run over the same trace. Arithmetic and harmonic mean based adaptation algorithms work on a granularity of 20 segments, as suggested in [10].To investigate video stalls, we compute the buffer level at the client as a time series based on the bitrate selection for each video segment and the bandwidth at that time, where the buffer level may drop to 0 and cause a video stall when the selected bitrate exceeds the bandwidth. Note that among DASH schemes, only piStream can obtain an accurate estimation of the *current available* bandwidth and use it for video bitrate adaptation. Yet for a micro-benchmark comparison of adaptation algorithms, we assume the current bandwidth (not future bandwidth) is known and used by all algorithms.

(i) Video streaming performance of adaptation algorithms. We first compare the video streaming performance of these algorithms. Figure 18 plots the results of this trace-driven emulation. Compared to the latest-throughput-based adaptation, LVA reduces the video stalling rate from 13% to 4%, at the cost of slight video quality loss (9%). Meanwhile, compared to the harmonic mean based adaptation, LVA improves video quality by 5% and the video stall rate by 1%.

(ii) Effectiveness of adaptation algorithms. To further investigate the reason behind the performance improvement, we look into the accuracy of adaptation by comparing the bitrate selection and the actual bandwidth of a video segment. Figure 19 shows that, 57% of the time, our LVA algorithm picks the video bitrate exactly matching the bandwidth, which is the highest among all tested algorithms. Moreover, we also see that the LVA algorithm has less overshoots and undershoots in the adaptation compared to other algorithms.

These results explain what Figure 18 shows. First, the latest-throughput-based adaptation has the highest video bitrate but lots of video stalls since it has more overshoots and less undershoots than other algorithms. LVA algorithm has the smallest video stall rate since it has the highest accuracy in adaptation (the peak at $x = 0$ in Figure 19). Both the harmonic and arithmetic mean based adaptation have low accuracy in adaptation, hence much worse performance.

5.2 System Level Test

We now compare piStream with state-of-the-art DASH adaptation algorithms (Section 4), including GPAC's DASH player [8], BBA [9], FESTIVE [10], and PANDA [15].

5.2.1 *piStream Client in Real-world LTE Network*

We first conduct experiments with a static piStream client playing MPEG-DASH video over populated commercial LTE network. To ensure a practical test environment with sufficient link dynamics, *e.g.*, other clients joining or leaving the network, we perform the experiments in a library with about 100 students inside during the afternoon, where our resource monitor constantly reports around 30% background PRB utilization caused by other LTE clients.

Before the experiments, we first use Iperf to measure the end-to-end bandwidth using wireline network to ensure there are no other bottlenecks between the DASH server and the client. We also use OpenLTE [7] to scan the LTE downlink channel to ensure there is no inter-cell interference. Each of our test lasts around 200 seconds, and we repeat each experiment with 10 runs.

(i) From Figure 20, we see that piStream substantially boosts video quality compared with alternative approaches. In particular, it improves the average video rate by 64% over GPAC, 125% over FESTIVE, 61% over BBA, and comparable to PANDA. This performance boost in video quality is mainly attributed to the PHY-informed bandwidth scaling, which enables the piStream client to select higher video bitrate than other algorithms without exceeding the bandwidth.

Meanwhile, piStream maintains a low video stall rate of close to 0. This implies that the LRD-based video adaptation can minimize the risk of buffer underrun, despite the aggressive video rate selection upon bandwidth scaling. *(ii)* GPAC's DASH player experiences a lower video stall rate compared to the trace-driven simulation (*i.e.*, the latest-value-based adaptation in Figure. 18) where the cur-

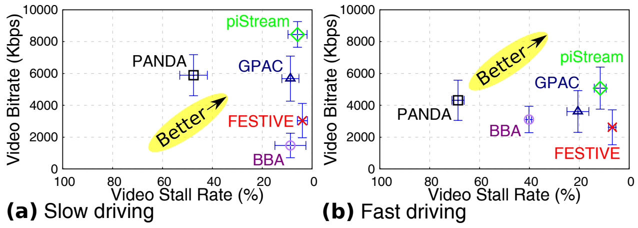


Figure 21: piStream outperforms other DASH algorithms in mobile environments.

rent downlink bandwidth is assumed to be known. In this real experiment, the GPAC client needs to execute its historical averaging based bandwidth estimation, which tends to be conservative, resulting in less video stalling.

(iii) The FESTIVE algorithm relies on the harmonic mean of the historical throughput to control the video bitrate. Despite its effectiveness in wireline networks [10], the harmonic mean takes a long time to converge. Under the high bandwidth variation of LTE, it results in severe bandwidth underutilization, despite a comparable stalling rate with piStream.

(iv) Surprisingly, the BBA algorithm exhibits an unacceptably high video stall rate (23%). It should ideally mitigate stalling by reducing the video rate whenever it sees a low buffer level. By examining at its adaptation logs, we found that, given a high buffer level, BBA decides on a high video bitrate regardless of the current downlink throughput. Such uninformed aggressive choice tends to drain the buffer off before the high-rate video segment finishes downloading.

(v) The PANDA algorithm is built on a bandwidth probing mechanism and should ideally make more informed decisions. Yet the results show 50% of stalling rate despite its comparable video quality with piStream. We found that PANDA probes the bandwidth until observing a decrease in the downlink throughput, which is taken as an indicator for congestion. However, it may have already overshoot the bandwidth before experiencing such a throughput decrease. Also, its slow end-to-end probing can hardly keep track of the bandwidth in real-time.

In summary, piStream makes the best tradeoff between video quality and stalling rate compared with a wide range of video adaptation protocols that adopt rate-based, buffer-based and probing-based approaches. These adaptation algorithms fall short of agility and precision in the presence of high bandwidth variation in LTE. piStream's PHY-informed design effectively overcomes such limitations.

5.2.2 *Impact of Mobility*

We further evaluate piStream's performance by running it inside a moving vehicle in two outdoor scenarios: *slow-driving* (20 to 25 mph in an urban downtown area) and *fast-driving* (45 to 50 mph in a suburban area). The software-radio running piStream's RMon module is driven by a 120V power-inverter inside the vehicle. To ensure similar effect of mobility to all algorithms, we conduct the experiment by driving through the same route when testing different algorithms. Since our current spectrum monitor design cannot properly handle the hand-off between cells, we need to ensure that the car remains in current cell during the evaluation time. Therefore, we only stream for 100 seconds for each algorithm.

Figure 21(a) and (b) plot the experimental results. First, we see that the performance of all algorithms suffers under higher mobility, which is expected since mobility is known

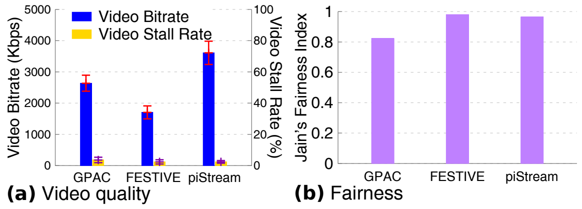


Figure 22: Performance with multiple DASH clients.

to worsen the channel condition and bring more bandwidth variations. Notably, the average video rate in the slow driving case is even higher than that in the static indoor environment (Figure. 20). This is because the higher signal strength outdoor can partly offset the negative effect of mobility.

piStream maintains the highest video quality among all tested algorithms and a low video stalling rate. This also implies that the RMon module in piStream is robust and reports accurate PRB utilization ratio even in the high mobility cases. In contrast, we found the control information decoding based method in LTEye [6] easily fails due to high bit errors induced by channel fading. It missed 40% of the downlink PRBs filled with higher energy, and it erroneously detected more than 1000 UEs in 20 seconds within a small cell. This poor performance is caused by the CRC failure, since LTE is designed to only let the UE to decode its own downlink allocation (Sec. 3.1.2).

The performance ranking of other tested algorithms also remains similar as in the static case. These algorithms can not keep track of the high bandwidth variation, and hence the video quality degradation becomes even more pronounced.

5.2.3 *Multiple piStream Clients Coexisting*

Will piStream's bandwidth scaling method cause unfairness when multiple clients coexist? We have provided a negative answer with basic intuitions in Section 3.2. Now we conduct experiments with four clients[8] running piStream concurrently and evaluate the Jain's fairness index. We only compare piStream with GPAC and FESTIVE, since BBA and PANDA keeps an unacceptably high stalling rate.

Figure 22(a) shows that piStream maintains the highest video quality among all users, with comparable/lower stalling rate than other approaches. On average, a piStream client enjoys 42.3% and 108% higher video rate than GPAC and FESTIVE, respectively. More importantly, Figure 22(b) shows a Jain's fairness index of close to 1 among all users. This implies that each piStream client takes a fair share of the available resource when enforcing the bandwidth scaling. Note that GPAC tends to result in low fairness (index around 0.8), with its simple historical throughput based adaptation, whereas FESTIVE has trade video quality for fairness into its adaptation algorithm, *i.e.*, sacrificing many bitrate increase opportunities in its stateful bitrate selection, thus achieving good fairness.

6. RELATED WORK

LTE measurement profiling and application optimization. The study in [31] represents a first measurement of commercial LTE deployment, revealing architecture-

[8]piStream can support many more users than 4, but we believe fairness is meaningful only when all concurrent clients have good QoE (*e.g.*, with stalling rate of below 10%). Therefore we only serve 4 users to trade concurrent user number for QoE.

level properties enabling lower-latency and higher throughput than 3G networks, as well as low-layer configurations (*e.g.*, large in-network queues) that cause high-layer anomaly. Huang *et al.* [32] examined how radio state machine interacts with application traffic in LTE networks, focusing on the end-host energy wastage due to uninformed active radio state maintenance. More recent studies examined the impact of handover [33], traffic burstiness and fair scheduling policies [4] on LTE/HSPA transport-layer. All such measurement/simulation hints to the importance of cross-layer interaction in LTE networks.

Adaptive video streaming protocols. Video streaming applications have kept challenging mobile Internet infrastructure and protocol design. Conventional video streaming relied on propriety services, often on top of UDP [34, 35]. Mobile video services like YouTube used to execute adaptation at server side, often employing simple strategies such as periodically downloading fixed-sized chunks, or downloading an entire video file at once [19]. More recent services are converging to DASH, which allows simply repurposing commodity web infrastructure (*e.g.*, CDN servers) for video streaming. This trend has proliferated many variants of DASH (*e.g.*, Microsoft Smooth Streaming, Apple HTTP Living Streaming, and Netflix). Existing DASH algorithms strive for a balance between high video quality (bitrate) and the risk of buffer underrun. This tradeoff can be deemed as an optimization problem. The key challenge lies in the under-determined "available bandwidth" constraint, which has been approximated using either historical TCP throughput [10, 15], or buffer dynamics [9]. We have shown that piStream essentially overcomes the limitations of these approaches via an informed cross-layer available-bandwidth estimation.

RAN-based throughput guidance. In cellular networks, the Radio Access Network (RAN) performs RRM (Radio Resource Management), all PHY layer details are available at the basestation. This fact inspires RAN-based throughput guidance solutions [36, 37], in which the traffic scheduling algorithm runs on the basestation and controls the end-to-end traffic by sending non-LTE-standard control messages. However, such RAN-based solutions require heavy modifications on the LTE basestation or even the core network [37]. In contrast, our piStream design is UE-oriented, which only requires a firmware upgrade for the UE devices. We believe this makes our design more practical and easier to deploy.

Adapting higher layer protocols to wireless bandwidth variation. Substantial research has been devoted to tailoring TCP for wireless networks. Many wireless TCP protocols proposed to prevent unnecessary TCP congestion reaction caused by lossy wireless links (see [38] for a survey). Modern cellular networks have masked link-level losses through smart retransmission mechanisms like HARQ. However, substantial challenges remain, *e.g.*, highly variable channel condition [39] and bufferbloat [40]. In the end, all these problems can be solved in future if an accurate end-to-end bandwidth estimation is available for the TCP layer.

In wireline networks, end-to-end bandwidth can be estimated by active probing using, *e.g.*, packet pairs and out-of-band parallel TCP connections [41]. However, the approach cannot be readily used in LTE, because the resource allocation at probing time does not represent that of real applications which experience bursty contending traffic, highly variable network conditions, and intermittent queue build-up. In [31], a passive estimation mechanism is deployed at a middlebox between user device and LTE core gateways, but instead of run-time bandwidth, it can only obtain a bandwidth upperbound when sending rate is fast enough to exceed the available bandwidth.

Sprout [28] introduces a stochastic forecast framework to predict future bandwidth of a cellular link based on historical statistics. It assumes each flow is isolated from competitors, and only deals with self-inflicted congestion behavior. Unlike piStream, PROTEUS [42] builds a regression tree to predict future throughput/delay based on historical application-layer statistics. In DASH, since application-layer itself may be trapped on a state with low bandwidth utilization, application-layer history can no longer indicate achievable throughput.

CQIC [43] represents the first work to augment bandwidth estimation in HSPA+ cellular networks using PHY-layer channel quality information (CQI). It estimates PHY-layer bandwidth by multiplying a CQI-based bitrate estimation with the fraction of time a UE is scheduled. Though bearing similar cross-layer design principle, piStream differs from CQIC in three aspects: *(i)* piStream not only inspects historical resource usage, but also explores unused resource; *(ii)* CQIC targets at packet-level congestion control while piStream handles second-level video segments. At this time scale, CQI information is inessential since the impact of short-term channel fading depreciates while long-term fading is directly captured in per-segment throughput; *(iii)* For LTE networks where resources are shared among UEs, CQI alone cannot represent available bandwidth – competing traffic and available resource are equally important, and become the dominating factor for relatively static UEs (Section 2).

7. CONCLUSION

Video streaming is surging to dominate the cellular network traffic today. However, current adaptive video streaming protocols fall short of the expected QoE even under LTE's high bandwidth provisioning. To address the problem, we present piStream, a DASH-compatible adaptive video streaming framework that exposes LTE's PHY layer information to facilitate video rate adaptation. piStream's PHY-informed design enables a more accurate bandwidth estimation and agile video rate adaptation. Extensive experiments on a real-time prototype show that piStream outperforms state-of-the-art video streaming algorithms in both video quality and playback smoothness. Under typical video streaming environments, piStream achieves around 1.6× video quality (bitrate) gain over the runner-up algorithm while maintaining a lower video stalling rate. We believe piStream's PHY-informed design can be applicable to a wider range of LTE applications than what we have explored.

Acknowledgement

We appreciate the anonymous reviewers for their insightful comments. This research was supported in part by the NSF under Grant CNS-1318292, CNS-1343363 and CNS-1350039.

8. REFERENCES

[1] Cisco Systems, Inc., "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012–2017,"

2012. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html

[2] Ctrix, "Mobile Analytics Report," 2014. [Online]. Available: http://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/citrix-mobile-analytics-report-september-2014.pdf

[3] Y. Zhang, A. Arvidsson, M. Siekkinen, and G. Urvoy-Keller, "Understanding HTTP flow rates in cellular networks," in *IFIP Networking Conference*, 2014.

[4] Y. Xu, Z. Wang, W. Leong, and B. Leong, "An End-to-End Measurement Study of Modern Cellular Data Networks," in *Proc. of Passive and Active Measurement Conference*, 2014.

[5] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," in *IEEE Multimedia*, 2011.

[6] S. Kumar, E. Hamed, D. Katabi, and L. Erran Li, "LTE Radio Analytics Made Easy and Accessible," in *Proc. of ACM SIGCOMM*, 2014.

[7] Ben Wojtowicz, "OpenLTE." [Online]. Available: http://openlte.sourceforge.net/

[8] ENST, "GPAC." [Online]. Available: http://gpac.wp.mines-telecom.fr/

[9] 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. of ACM SIGCOMM*, 2014.

[10] J. Jiang, V. Sekar, and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE," in *Proc. of ACM CoNEXT*, 2012.

[11] X. Yin, V. Sekar, and B. Sinopoli, "Toward a Principled Framework to Design Dynamic Adaptive Streaming Algorithms Over HTTP," in *Proc. of ACM HotNets*, 2014.

[12] 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. of ACM HotMobile*, 2015.

[13] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, 1994.

[14] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web Traffic: Evidence and Possible Causes," in *Proc. of ACM SIGMETRICS*, ser. SIGMETRICS '96, 1996.

[15] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. Begen, and D. Oran, "Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, 2014.

[16] Microsoft, "Microsoft Smooth Streaming," http://www.iis.net/downloads/microsoft/smooth-streaming.

[17] Apple, "HTTP Live Streaming." [Online]. Available: https://developer.apple.com/streaming

[18] Adobe, "HTTP Dynamic Streaming," http://www.adobe.com/products/hds-dynamic-streaming.html.

[19] M. Hoque, M. Siekkinen, J. Nurminen, and M. Aalto, "Dissecting Mobile Video Services: An Energy Consumption Perspective," in *Proc. of IEEE WoWMoM*, 2013.

[20] N. Becker, A. Rizk, and M. Fidler, "A Measurement Study on the Application-Level Performance of LTE," in *IFIP Networking Conference*, 2014.

[21] A. Gouta, D. Hong, A.-M. Kermarrec, and Y. Lelouedec, "HTTP Adaptive Streaming in Mobile Networks: Characteristics and Caching Opportunities," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2013.

[22] Lawrence Berkeley National Laboratory, "iperf3." [Online]. Available: http://software.es.net/iperf/

[23] Qualcomm, "QXDM." [Online]. Available: https://www.qualcomm.com/documents/qxdm-professional-qualcomm-extensible-diagnostic-monitor

[24] "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation," *3GPP TS 36.211 version 12.3.0 Release 12*, 2014.

[25] D. Halperin et. al., "Linux 802.11n CSI Tool." [Online]. Available: http://dhalperi.github.io/linux-80211n-csitool/

[26] S. Sen, B. Radunovic, J. Lee, and K.-H. Kim, "CSpy: Finding the Best Quality Channel Without Probing," in *Proc. of ACM MobiCom*, 2013.

[27] M. S. Taqqu, V. Teverovsky, and W. Willinger, "Estimators for Long-Range Dependence: an Empirical Study," *Fractals*, 1995.

[28] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks," in *USENIX NSDI*, 2013.

[29] S. Kotz and S. Nadarajah, *Extreme Value Distributions: Theory and Applications*. London: Imperial College Press, 2000.

[30] O. Oyman and S. Singh, "Quality of experience for HTTP adaptive streaming services," *IEEE Communications Magazine*, vol. 50, no. 4, 2012.

[31] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance," in *Proc. of ACM SIGCOMM*, 2013.

[32] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *Proc. of ACM MobiSys*, 2012.

[33] B. Nguyen, A. Banerjee, V. Gopalakrishnan, S. Kasera, S. Lee, A. Shaikh, and J. Van der Merwe, "Towards Understanding TCP Performance on LTE/EPC Mobile Networks," in *Proc. of the Workshop on All Things Cellular (AllThingsCellular)*, 2014.

[34] X. Zhu and R. Pan, "NADA: A Unified Congestion Control Scheme for Low-Latency Interactive Video," in *International Packet Video Workshop (PV)*, 2013.

[35] H. Lundin and S. Holmer and H. Alvestrand, "A Google Congestion Control Algorithm for Real-Time Communication," 2013, internet-Draft (Informational).

[36] Internet Engineering Task Force, "Mobile Throughput Guidance Signaling Protocol." [Online]. Available: https://tools.ietf.org/html/draft-flinck-mobile-throughput-guidance-00

[37] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang, "A Scheduling Framework for Adaptive Video Delivery over Cellular Networks," in *Proc. of ACM MobiCom*, 2013.

[38] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," in *Proc. of ACM SIGCOMM*, 1996.

[39] M. C. Chan and R. Ramjee, "TCP/IP Performance over 3G Wireless Links with Rate and Delay Variation," in *Proc. of ACM MobiCom*, 2002.

[40] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling Bufferbloat in 3G/4G Networks," in *Proc. of ACM Internet Measurement Conference (IMC)*, 2012.

[41] M. Jain and C. Dovrolis, "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, 2003.

[42] Q. Xu, S. Mehrotra, Z. Mao, and J. Li, "PROTEUS: Network Performance Forecast for Real-time, Interactive Mobile Applications," in *Proc. of ACM MobiSys*, 2013.

[43] F. Lu, H. Du, A. Jain, G. M. Voelker, A. C. Snoeren, and A. Terzis, "CQIC: Revisiting Cross-Layer Congestion Control for Cellular Networks," in *Proc. of ACM HotMobile*, 2015.