MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath

Bo Han Feng Qian* Lusheng Ji Vijay Gopalakrishnan
AT&T Labs – Research *Indiana University
Bedminster, NJ Bloomington, IN
{bohan, lji, gvijay}@research.att.com fengqian@indiana.edu

ABSTRACT

Compared with using only a single wireless path such as WiFi, leveraging multipath (*e.g.,* WiFi and cellular) can dramatically improve users' quality of experience (QoE) for mobile video streaming. However, Multipath TCP (MPTCP), the *de-facto* multipath solution, lacks the support to prioritize one path over another. When applied to video streaming, it may cause undesired network usage such as substantial over-utilization of the metered cellular link. In this paper, we propose MP-DASH, a multipath framework for video streaming with the awareness of network interface preferences from users. The basic idea behind MP-DASH is to strategically schedule video chunks' delivery and thus satisfy user preferences. MP-DASH can work with a wide range of off-the-shelf video rate adaptation algorithms with very small changes. Our extensive field studies at 33 locations in three U.S. states suggest that MP-DASH is very effective: it can reduce cellular usage by up to 99% and radio energy consumption by up to 85% with negligible degradation of QoE, compared with off-the-shelf MPTCP.

Keywords

Multipath TCP; DASH; Adaptive Video Streaming; WiFi; Cellular; Preference

1. INTRODUCTION

Video streaming has become one of the most critical applications on mobile devices. A recent study indicates that video accounted for 55% of total mobile traffic in 2015, and this percentage will increase to 75% by 2020 [4]. However, mobile users' quality of experience (QoE) for video streaming is still often far from satisfactory, especially under challenging network conditions such as unstable WiFi connectivity and mobility. Based on our field studies at 33 locations such as restaurants and hotels, we found in about 80% of locations their open WiFi does not provide stable throughput for streaming a 1080p video at its highest quality, while the combined bandwidth of WiFi and cellular almost always does (§2.2). Although LTE may be enough at most locations, users may want to limit their cellular data usage.

The above findings motivate us to consider streaming over *multipath*, a common feature on today's laptops, smartphones, and even wearables. Multipath TCP (MPTCP [15]) is the *de-facto* multipath solution allowing applications to transparently use multiple paths. It can dramatically improve the QoE of video streaming by providing additional network capacity and robust communications (*e.g.,* facilitating smooth handover). However, it is challenging for MPTCP to support the network interface preference required by mobile users (*e.g.,* preferring WiFi over LTE when at home) [20]. As a consequence, when streaming video over MPTCP, it may incur undesired network usage such as substantial over-utilization of the metered cellular link. This will considerably reduce users' incentive of using multipath due to concerns of cellular data limits.

In this paper, we propose MP-DASH, a novel multipath video streaming framework. The overall goal is to *enhance MPTCP to support adaptive video streaming under user-specified interface preferences*. Dynamic Adaptive Streaming over HTTP (DASH) is the state-of-the-art standard of video streaming [35]. Unlike other Internet video protocols such as Adobe HDS [2], Apple HLS [3], and Microsoft Smooth Streaming [6], DASH is agnostic of the video codec and can use any codec including H.264 and H.265. DASH is becoming the dominant video streaming solution, due to the extreme success of HTTP that is easy to deploy, middlebox-friendly, and supported by infrastructures such as CDN [41, 42].

MP-DASH has two major components: the MP-DASH scheduler and the video adapter. The scheduler takes as input the interface preference from users, as well as video chunks' size and delivery deadline from the video player. It then intelligently determines the best fetching strategy of the video chunks over multipath by leveraging their

CoNEXT '16, December 12-15, 2016, Irvine, CA, USA
 ISBN 978-1-4503-4292-6/16/12…$15.00
DOI: http://dx.doi.org/10.1145/2999572.2999606

delay tolerant nature. The key challenge is the scheduling algorithm needs to be online, lightweight, robust, and generalizable to different user preferences.

The MP-DASH video adapter is a lightweight add-on that makes an existing off-the-shelf DASH algorithm multipath-friendly. Lying in the middle between the original DASH algorithm and the MP-DASH scheduler, it handles the interactions between them. This is also challenging because the interactions are not only complex, but also different across different DASH rate adaptation algorithms. These two building blocks (MP-DASH scheduler and video adapter) work in synergy to improve the efficiency of video delivery through preference-aware multipath.

We have instantiated the MP-DASH framework in the context of improving video delivery over WiFi and cellular networks, with the goal of reducing the cellular data usage while maintaining user QoE. Specifically, we made the following contributions in this paper.

- We design the MP-DASH scheduler as an overlay of MPTCP (§4). It serves as a building block for multipath-friendly video streaming. In addition, it can benefit a wide range of other applications with delay-tolerant transfers.
- We investigate how to use the MP-DASH scheduler for improving DASH video delivery. We study two representative categories of DASH rate adaptations (throughput-based and buffer-based) and propose solutions for them to effectively use MP-DASH with only a few lines of code change (§5).
- We implement the MP-DASH scheduler in the Linux kernel and MP-DASH enhanced state-of-the-art DASH algorithms such as FESTIVE [24] and BBA [23] on an open-source video player [5]. We also build a first cross-layer multipath video analysis tool to facilitate our research (§6).
- We conduct extensive in-field experiments to study mobile video performance over multipath in realistic settings, by visiting 33 public places in three states of the U.S. The results show that MP-DASH can reduce cellular usage by up to 99% and radio energy consumption by up to 85% with negligible degradation of QoE (§7).

2. MOTIVATIONS

In this section, we provide the background of MPTCP and motivate MP-DASH through both real-world measurement studies and controlled experiments.

2.1 Multipath TCP (MPTCP)

MPTCP [15] is a standardized multipath solution allowing applications to *transparently* use multiple paths. MPTCP has already been implemented in the Linux kernel [31]. However, it has no ability for users to specify their preference of how to utilize wireless interfaces [20]. Instead, the path utilization is mainly determined by the MPTCP scheduling algorithms. The default scheduler of MPTCP prefers low latency paths: when multiple subflows have spaces in their congestion windows, it selects the subflow with the smallest RTT estimation to transmit the next packet. MPTCP also supports round-robin scheduling.

In this paper, we consider "decoupled" congestion control, *i.e.,* each path runs congestion control independently. This is the typical congestion control configuration for mobile multipath [18]. The reason is that the goal of coupled congestion control [39] is mainly to offer fairness at shared bottlenecks; however, the bottleneck is usually the last mile for wireless networks which is unlikely to be shared by WiFi and cellular networks. We next demonstrate that blindly applying MPTCP scheduling algorithms to DASH video can lead to severe resource inefficiency such as excessive use of cellular data.

2.2 Measurement Study in the Wild

We consider mobile video streaming using WiFi and cellular. To motivate the need for multipath, we first answer the question of whether WiFi alone can provide the best video streaming experience. We measure the throughput and RTT of the WiFi networks at 33 locations in three far-apart U.S. states. The locations cover a wide range of public places including both indoor environments, such as airport, hotel, and restaurant, and outdoor parking lot. They represent three different scenarios: (1) WiFi only is never able to support the highest bitrate of a 1080p video, (2) WiFi can sometimes play the best quality, but not always, and (3) WiFi can almost always stably support the highest bitrate. Among these locations, 64%, 15%, and 21% of them belong to the respective scenarios.

The results imply that in the real world, it is very likely the public WiFi itself cannot provide the best video streaming experience. The poor WiFi performance is usually caused by either bandwidth throttling (*e.g.,* applied by hotels [1]) or limited backhaul connectivity [25]. Thus, upgrading to high throughput WiFi, such as 802.11ac, may not always solve the problem. In contrast, MPTCP can sustain the highest playback bitrate at *all* locations (we consider the HD video case separately in §7.3.5). This indicates that MPTCP is indeed useful for improving the QoE of mobile video streaming in diverse realistic settings.

Although MPTCP is helpful, it causes unnecessary use of cellular data, in particular in scenario (3) where cellular is not needed. For example, the measured WiFi and LTE throughputs at an office building are 12.1 and 14.6 Mbps. If a user plays a 1080p DASH video with the highest bitrate close to 4.0 Mbps, WiFi alone can already sustain the highest rate. However, when streaming the video over MPTCP blindly, more than half of data is still sent over LTE.

2.3 Controlled Experiments

We next employ controlled in-lab experiments to detail the issue of MPTCP overusing cellular data when playing a DASH video. The following observations also exist in real-world experiments (§7) and we use controlled experiments only for easy presentation. Suppose a user wants to play the same video mentioned above with the highest rate which is around 4.0 Mbps. However, the bandwidth of WiFi network is only 3.8 Mbps. To enjoy the best video-watching experience, she can also leverage the LTE dongle on her laptop and play this video over MPTCP. The throughput of LTE network is 3.0 Mbps. Figure 1 plots the throughputs of WiFi and LTE subflows and the entire MPTCP flow in the steady playing state.

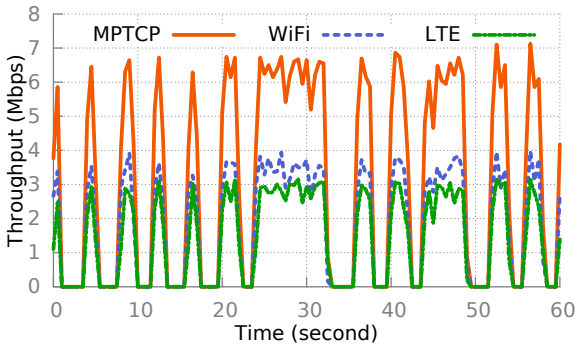


Figure 1: WiFi/LTE throughput when playing a DASH video over MPTCP.

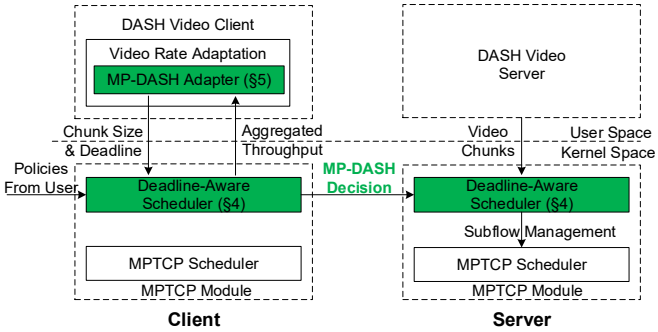


Figure 2: System architecture of MP-DASH.

There are two key observations from Figure 1. First, to download video chunks, the capacity of LTE link is almost fully utilized, as the MPTCP schedulers are not aware of user preferences. Since data limits on LTE networks are common, ideally the user wants to use only 0.2 Mbps of the LTE bandwidth to fill the gap. This is the problem we aim to solve through making the scheduling decision *user-preference-aware*. In §7.3.3, we will also demonstrate through extensive field studies the real-world prevalence of this problem.

We aim to address the above issue by taking advantage of the fact that video chunks are *delay-tolerant*. As shown in Figure 1, when the player's buffer is full, after finishing downloading a chunk, the networks will be idle for a while before fetching the next chunk (*i.e.,* waiting for the player to consume its buffer). Thus, the QoE will not be affected if the chunks arrive a bit late, as long as the playback deadline is met. This offers opportunities of tweaking the MPTCP scheduler to reduce cellular data usage.

3. THE MP-DASH FRAMEWORK

We propose MP-DASH, a network interface preference aware multipath framework for DASH video streaming.

3.1 MP-DASH System Design

There are three major requirements for MP-DASH: First, it should be able to accommodate the interface preferences specified by users and schedule the video delivery based on the preferences. Second, there needs to be a mechanism for video-specific information to be exchanged between a video player and the MP-DASH framework. Third, the overall system needs to be scalable and lightweight.

To satisfy them, our proposed framework consists of two key components: the MP-DASH scheduler and the MP-DASH video adapter, as shown in Figure 2. The scheduler enhances MPTCP in two aspects. First, it is aware of the preference of multiple paths. Second, it is aware of the deadline of each video chunk. By using these pieces of information, the scheduler strategically manages the network paths with the goal of meeting deadlines while satisfying user-specified preference, as to be detailed in §4.

The MP-DASH video adapter is a lightweight add-on that makes existing DASH algorithms multipath-friendly. Lying in the middle between the original DASH algorithm and the MP-DASH scheduler, its main job is to inform the MP-DASH scheduler through a unified interface of each chunk's size and deadline. Also, depending on the specific characteristics of a DASH algorithm category, the MP-DASH adapter may have additional interactions with the DASH algorithm, as to be elaborated in §5.

3.2 Specific Design Decisions

We now detail specific aspects of the MP-DASH framework.

The Interface of the MP-DASH Scheduler. The MP-DASH scheduler exposes a simple and unified interface to video applications, similar to existing approaches [20, 34]. As shown in Figure 2, the interface consists of two parts. First, the scheduler offers a socket option, **`MP_DASH_ENABLE`**, to convey the data size S and the deadline D from the user space to the kernel. Upon the reception of this information, MP-DASH is activated for the next S bytes of data. MP-DASH is deactivated when any of the following happens: (1) S bytes have been successfully transferred, (2) the deadline has passed, or (3) the application explicitly deactivates MP-DASH using **`MP_DASH_DISABLE`**, another socket option. The second part of the interface enables a DASH adapter to obtain necessary information (*e.g.,* the aggregated throughput across all paths as shown in Figure 2) for rate adaptation. This interface is needed because a video player does not have the visibility of the network conditions of all paths since MPTCP is oblivious to the player. We demonstrate its usage in §5.

Function Split between Client and Server. The MP-DASH scheduler should ideally run at the video streaming server where it can directly distribute traffic onto different paths. However, a video server does not run the DASH logic and thus the MP-DASH adapter should reside on the client side. We split the scheduler into two parts, the *decision function* on the client and the *enforcement function* on the server. The decision function determines how to manage the paths (*e.g.,* whether the cellular subflow should be enabled) based on the information from the video player, and notifies the enforcement function of the decision using a reserved bit in the MPTCP option. This design makes MP-DASH scalable as the server side becomes stateless.

Interface for the User. As a self-contained framework, MP-DASH allows users to specify the multipath policies such as interface preferences. Our current prototype supports two policies in the context of optimizing video delivery over wireless networks: preferring WiFi over cellular, and preferring cellular over WiFi. Despite the former being the common case, there are situations where a user prefers the latter (*e.g.,* when the user is moving). Given the two policies are symmetric, in the rest of the paper we focus on the first one. We enforce the policy by setting the preferred interface as the primary interface of MPTCP. More sophisticated policies can be plugged in by only changing the scheduler without affecting the DASH adapter (§4).

4. DEADLINE-AWARE MP-DASH SCHEDULER

We design a deadline-aware MP-DASH scheduler that takes both the interface preference and the delay-tolerant nature of video traffic into consideration. The preference can be quantified by cost associated with each path. The cost could be data usage, energy consumption, or a combination of both, configured either statically or dynamically. The goal of the MP-DASH scheduler is to meet the playback deadline of a chunk while minimizing the overall cost. Deadline-awareness has been investigated in various scenarios such as base station scheduling for multimedia traffic [14] and data center TCP [38]. Different from existing works, MP-DASH targets at DASH video delivery over multipath.

General Formulation. We first provide a general formulation of MP-DASH's scheduling algorithm. Let $N \geq 1$ be the number of network paths. Let S be the chunk size and D (in discretized time slots) be its download deadline. The duration of each time slot is d. Let $b(i,j)$ be the available bandwidth of interface i at time slot j ($1 \leq i \leq N$, $0 \leq j < D$), and $c(i,j)$ be the unit-data cost of using interface i at time j. $x(i,j)$ is a binary decision variable where $x(i,j)$=1 iff interface i is used at time j, otherwise $x(i,j)$=0. The goal is to minimize the overall cost

$$C = \sum_{1 \leq i \leq N, 0 \leq j < D} c(i,j)b(i,j)x(i,j)d$$

under the constraint that the deadline is met:

$$\sum_{1 \leq i \leq N, 0 \leq j < D} b(i,j)x(i,j)d \geq S$$

This is a 0-1 min knapsack problem [16]: we have $D * N$ items whose weights and values are $b(i,j)d$ and $c(i,j)b(i,j)d$, respectively. We want to choose items such that the total weight is at least S and the total value is minimized. According to the duality for the Integer Linear Programming, this is equivalent to the traditional 0/1-knapsack problem (proof omitted). We can solve it optimally using dynamic programming with the complexity of $O(N * D * S)$, or approximately by heuristic algorithms.

Practical Online Algorithm. In practice, MP-DASH needs to run in an online fashion where the throughput estimation is continuously updated. Meanwhile, when applied on today's mobile devices, the formulation can be simplified by assuming N=2 (only WiFi and cellular interfaces). In the following we assume WiFi is always preferred over cellular. Thus, we can set $c(\text{WiFi}, j) < c(\text{cell}, j)$ for $\forall j$. A similar greedy algorithm can be devised to handle general numerical costs of multiple paths, as to be described later.

Algorithm 1 Deadline-Aware Scheduling Algorithm

```
1: Input: S – video chunk size and D – deadline window;
2: sentBytes = 0;
3: cellularEnabled = FALSE;
4: timeStart = gettimeofday();
5: while (sentBytes < S) do
6:    if S − sentBytes > packetSize then
7:       n = packetSize;
8:    else
9:       n = S − sentBytes;
10:   end if
11:   Get n bytes from the chunk and send them using
      MPTCP;
12:   sentBytes+ = n;
13:   timeNow = gettimeofday();
14:   timeSpent = timeNow − timeStart;
15:   Get the estimated WiFi throughput R_WiFi;
16:   if (α×D−timeSpent)×R_WiFi > (S−sentBytes)
      && cellularEnabled == TRUE then
17:      Disable the cellular path;
18:   end if
19:   if (α×D−timeSpent)×R_WiFi < (S−sentBytes)
      && cellularEnabled == FALSE then
20:      Enable the cellular path;
21:   end if
22: end while
```

MP-DASH *leverages the existing MPTCP schedulers to distribute packets over multiple paths and adds the intelligence of how to control the cellular subflow*. Essentially, MP-DASH introduces both deadline-awareness and priority into these schedulers. The online version of MP-DASH is depicted in Algorithm 1. It has two input parameters, the video chunk size S and the length of download time window D (from when a download starts till the deadline). MP-DASH drives the WiFi subflow to its full capacity and turns off the cellular subflow at the beginning. It then monitors the progress of data transfer and turns the cellular subflow on if necessary, *i.e.,* when the WiFi path underperforms and the deadline would be missed if we continue using WiFi alone. The *while* loop is responsible for sending the chunk's data using MPTCP (line 11). R_{WiFi} in line 15 is the current estimation of WiFi throughput. After sending out each packet, MP-DASH checks if WiFi alone is sufficient to transmit the remaining data and disables the cellular subflow when possible (lines 16–18). Since the WiFi throughput may change over time, it also needs to check if the cellular subflow should be enabled again, in case using WiFi alone cannot fully deliver the remaining data before the deadline (lines 19–21). To be conservative, our target finish time could be ahead of the real deadline by setting α in lines

16 and 19 to be less than 1, to compensate for estimation inaccuracy of WiFi throughput. The smaller the value of α is, the less likely we will miss the actual deadline. However, a smaller α also leads to more data over cellular links.

A key design decision we make is, if MP-DASH decides to use cellular, it should utilize the maximum bandwidth by transmitting as quickly as possible. Another naive solution is to throttle the cellular bandwidth just to be what is needed. We will demonstrate its drawbacks in §7.3.1.

Optimality. As a special case (N=2), Algorithm 1 yields the optimal solution (in terms of cellular usage) if we can always accurately estimate the bandwidth till the deadline (proof omitted). Intuitively, assume a solution that uses even fewer cellular bytes exists. This means we either disable cellular too late or enable cellular too early. However, neither can happen if we have the perfect knowledge of the bandwidth in lines 16 and 19.

The online MP-DASH scheduling algorithm can be generalized to multiple interfaces with varying costs. For example, we can first sort the interfaces based on their costs, and then feed data from low-cost to high-cost interfaces, by turning on/off the paths accordingly. This cost-varying version is an approximation algorithm and may not be optimal. Meanwhile, the interface to applications remains the same, which allows the same MP-DASH enhanced DASH algorithm in our framework to work under diverse multipath contexts with different numbers of paths or different cost/preference profiles.

5. MP-DASH VIDEO ADAPTER

We leverage the MP-DASH scheduler as a building block to improve the cellular usage and energy efficiency for DASH video over multipath. It is challenging due to the following two reasons.

• The interaction between the DASH rate adaptation algorithms and the MP-DASH scheduler is complex. In particular, there exists a control loop: MP-DASH's scheduling decision will affect DASH's rate selection, which in turn impacts MP-DASH's scheduling through settings of chunk size and deadline. This is quite different from using the MP-DASH scheduler to download a single file where the application and the scheduler are very loosely coupled.

• There are many DASH rate control algorithms. Ideally, we want to plug the MP-DASH scheduler into them in the same (and simple) manner. However, we found due to their different rate adaptation logics (*e.g.,* throughput vs. buffer-based), DASH algorithms incur different cross-layer interactions with the scheduler. This inevitably requires modifications to DASH algorithms on a category-by-category basis, although we try to minimize the changes.

5.1 The Basic Approach

We first provide the background of rate adaptation algorithms for DASH. In a DASH system, a video is split into multiple chunks of the same play time (usually 1 to 15 seconds) and each video chunk is encoded with multiple discrete bitrate levels. A video player can switch between different bitrate streams at a chunk boundary, as the chunks are time-wise aligned. Thus, it requires an algorithm to select the chunks' bitrates for achieving an optimal QoE. There are largely two categories of DASH rate adaptation algorithms, *throughput-based* and *buffer-based*. The throughput-based adaptation adjusts the encoding rate of chunks based on the estimated throughput, which indicates the future network capacity. For instance, FESTIVE [24] uses the harmonic mean of previous chunks' throughputs to estimate the future throughput, which dictates the rate selection. On the other hand, Buffer Based Adaptation (BBA) [23] chooses the bitrate based on a video player's buffer occupancy level, which implicitly encodes information of the network capacity when the video playing is at the steady state.

We now describe the basic approach of integrating the MP-DASH scheduler with DASH video. Recall in §4 that MP-DASH's scheduling algorithm is invoked at the chunk level, and the video player needs to provide the chunk size and deadline to MP-DASH. Currently the chunk size is not a mandatory field in the DASH manifest file. Despite this, in practice, we can almost always find the chunk size in the "Content-Length" header field of HTTP responses. This issue has also been raised by a recent work [42], which advocates that chunk size is a key requirement for DASH control algorithms and thus should be a mandatory part of the DASH manifest.

We then consider how to set the deadline D. Intuitively, it should be configured in such a way that if we do not receive the chunk at time $t_0 + D$, the playback will stall (t_0 is the current time). It is, however, too risky in that missing the deadline due to inaccurate throughput estimation or temporary wireless blackout will cause significant QoE degradation. In fact, preventing this is exactly the purpose of having playback buffers that act as a cushion for unpredicted network quality changes.

We instead set the deadline by *keeping the buffer occupancy not decrease*. Specifically, we propose two approaches: duration-based and rate-based. For the *duration-based* setting, D is a video chunk's playout duration. For example, the deadline of a 4-second video chunk is simply 4 seconds. For the *rate-based* scheme, D is the chunk size divided by the nominal (*i.e.,* average) video encoding bitrate. For example, for a 1 MB chunk in a quality level of 4.0 Mbps average bitrate, D is set to 1*8/4=2 seconds. The difference between them is, the duration-based scheme aims to maintain the buffer level in the short term: a player consumes D-second buffer to download a D-second chunk which is then supplied to its buffer, thus leading to a stable buffer level. In contrast, the rate-based setting attempts to maintain the buffer level in the long run, since it leverages the *average* bitrate of the entire video.

We further design a mechanism called *deadline extension* that makes MP-DASH even more efficient. The intuition is, it is unlikely for a stall to happen when the buffer occupancy is close to becoming full. In this "safe region", we can relax (*i.e.,* extend) the deadline to give more opportunities for MP-DASH to reduce the cellular usage. Assuming the buffer level threshold for enabling deadline extension is Φ

and the current buffer level is $b > \Phi$, we then extend the deadline by $b - \Phi$ (both b and Φ have the unit of seconds). The threshold Φ incurs a tradeoff between the cellular usage and playback robustness. We discuss its setting for different DASH algorithms in §5.2.1 and §5.2.2.

We extend the deadline when the buffer level is high. Then how to deal with the situation of low buffer level? This happens, for example, during the initial buffering phase or a path temporarily blacks out. Thus, we disable the MP-DASH scheduler for MPTCP when the buffer level is lower than a threshold Ω. The setting of Ω also depends on the DASH algorithms, which we discuss next.

5.2 Handling Different Categories of DASH Rate Adaptation Algorithms

Following the unified high-level framework described in §5.1, we now detail how to seamlessly integrate the MP-DASH scheduler with different categories of DASH rate adaptation algorithms.

5.2.1 Throughput-based DASH Rate Adaptation

A throughput-based rate adaptation algorithm uses previous chunks' throughputs to estimate the future throughput, which is then mapped to the quality level of the next chunk using algorithm-specific logics. The throughput-quality mapping logic is transparent to MP-DASH. However, the player is not aware of multipath and may under-estimate the actual MPTCP throughput when the MP-DASH scheduler disables the cellular path. To address this issue, we expose to applications the MPTCP throughput estimation via the interface described in §3, which *overrides* the player's own estimation. In this way, the player has a consistent view of the overall available network resources based on which it makes the rate-selection decision.

For the deadline extension, we empirically set Φ to be 80% of the overall buffer capacity. We set Ω, the low-buffer threshold for disabling the MP-DASH scheduler, as follows. Consider a time window of the next T seconds. Assuming we stay at the lowest bitrate, we estimate the length (in time) of chunks that can be downloaded during this window to be T'. Then the threshold of disabling the MP-DASH scheduler is $\Omega = T - T'$ where T and T' are the buffering time to be consumed and supplied, respectively. A negative Ω is treated as 0. We set T to be twice of the duration of the entire buffer. Changing T to 1x or 3x of the buffer duration does not qualitatively change the results. To be conservative, we set the minimum value of Ω to be 40% of the buffer capacity.

5.2.2 Buffer-Based DASH Rate Adaptation

We then shift to the buffer-based rate adaptation, which maps the buffer occupancy to video qualities. The intuition is that the buffer occupancy implicitly encodes the relation between the network capacity and the selected video bitrate.

Based on our experiments in §7, we found the original buffer-based rate adaptation algorithm is very aggressive in utilizing the bandwidth. Assume the MPTCP throughput is stable at around R=3.4 Mbps, whose nearest video encoding bitrates are r_1=2.4 Mbps and r_2=3.9 Mbps. A throughput-based scheme (*e.g.,* FESTIVE) will only fetch chunks of r_1,

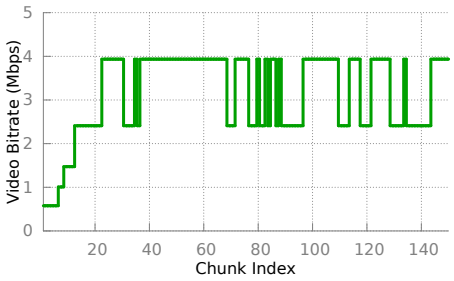


Figure 3: Bitrate oscillation of the original BBA algorithm.

which is the highest bitrate that the network can sustain. On the other hand, a buffer-based scheme starts with rate r_1. Since $r_1 < R$, the buffer occupancy grows, which switches the encoding bitrate to $r_2 > R$. Then the buffer level decreases, eventually switching the encoding bitrate back to r_1. As a result, the selected bitrate keeps oscillating between r_1 and r_2, as shown in Figure 3. This behavior of BBA not only degrades the QoE [37, 42] but also limits the resource-saving capability of MP-DASH. To address this problem, we modified the original buffer-based adaptation algorithm (BBA) by forcing the selected bitrate to be *no higher* than the actual MPTCP throughput. We call the modified scheme BBA-C (Cellular-friendly version of BBA). We evaluate both BBA and BBA-C in §7.

We discuss how to configure Φ and Ω. To keep the buffer not full, we conservatively set Φ (the high-buffer threshold for deadline extension) to be the overall buffer capacity minus one chunk's duration. For Ω (the low-buffer threshold), we cannot use a single buffer occupancy threshold as used in the throughput-based scheme, because in a buffer-based scheme, each encoding bitrate has its own buffer occupancy range. We enable the MP-DASH scheduler only when the player reaches the highest bitrate that the network can sustain. At this point, we need to keep the buffer occupancy higher than the lowest level of the current encoding bitrate, otherwise the bitrate will drop to the next lower level. Therefore, we disable the MP-DASH scheduler whenever the buffer occupancy is getting close to the lowest level of the current encoding bitrate, e_l. We empirically set Ω to be e_l plus one chunk's duration. For example, assume the current video quality maps to a buffer level range from e_l=20 to e_h=40 seconds, and a chunk's duration is 4 seconds. Then we enable the MP-DASH scheduler only when the buffer contains at least 24 seconds' content.

The high-level principle of choosing the proper values of Φ and Ω for both the throughput-based and buffer-based rate adaptations is to be conservative with the purpose of preventing stalls. However, we empirically found the current settings of Φ and Ω yield good resource savings while incurs negligible playback rate degradation in realistic settings (§7). We plan to evaluate how different values of these parameters impact other QoE metrics, such as stalls and video bitrate switches, in our future work.

5.2.3 Hybrid DASH Rate Adaptation

We can leverage the MP-DASH scheduler for a wide range of DASH algorithms. We consider one example of

MPC [42], a recent model predictive control algorithm that leverages both throughput estimation and buffer occupancy. At a high level, MPC works as follows. Instead of solving an optimization problem for each chunk, its online version looks up a *pre-generated* table to select the optimal bitrate based on the buffer level, previous bitrate, and throughput estimation. To support MP-DASH, MPC can set a chunk's deadline as the chunk size divided by the minimum throughput required by the optimal bitrate known from the table. Meanwhile, since MPC is a hybrid approach of using both throughput and buffer occupancy for rate adaptation, we can reuse several solutions described in §5.2.1 and §5.2.2. We leave the detailed design and implementation as future work.

6. IMPLEMENTATION

We have implemented the MP-DASH **Scheduler** on Linux platform in the kernel instead of the user space, which is more suitable for complex application-specific schedulers [20]. Given the simplicity of MP-DASH scheduler and since it tightly couples with MPTCP, realizing it in the kernel significantly reduces its communication overhead with the basic functions of MPTCP. Based on the online algorithm, a client uses a reserved bit in the MPTCP DSS (Data Sequence Signal) option [15] to notify a server of its decision about whether the cellular subflow should be enabled. The user-specified interface preference is realized by setting the primary MPTCP interface accordingly. Implementing a general policy framework is still an open problem and we leave it as our future work. To estimate the throughput of a subflow, we employ the non-seasonal Holt-Winters (HW) predictor [19], which is known to be more robust than other approaches such as exponentially weighted moving average (EWMA) for non-stationary processes. The HW predictor is essentially a double exponential smoothing method that takes into account the possible trend in the measurement data. We implement it in MP-DASH using the parameters suggested by He *et al.* [19]. Overall, we add about 300 lines of Linux kernel code as a portable patch of MPTCP. We have also applied this patch to the Linux kernel of Google Nexus 5 with Android 4.4.2.

A key challenge of implementing the MP-DASH scheduler is that we need to manage the secondary subflow (*i.e.,* cellular subflow in our case) in an energy efficient manner. Instead of actually switching on/off the cellular radio, or adding/removing the cellular subflow (which is supported by a recent work [20] at the cost of increased TCP handshake delay), we leverage the existing MPTCP schedulers that select the available subflow for each packet. When we "disable" the cellular subflow, we simply *skip* it in the scheduling function, thus incurring no overhead of handshake message exchange. This design works as a simple overlay with both the default and the round-robin MPTCP schedulers (and presumably others). As a result, MP-DASH can handle diverse network conditions (*e.g.,* the paths have drastically different delay or bandwidth) as the off-the-shelf MPTCP schedulers do. Note that keeping the LTE radio interface always on incurs very small additional energy overhead (only periodical Discontinuous Reception (DRX) spikes [21]). We discuss the implication of our design on radio energy consumption in §7.

Our implementation of the MP-DASH **Video Adapter** is based on the open-source GPAC video player (v0.5.2) [5]. It has a simple throughput-based rate adaptation algorithm, referred to as GPAC, which estimates the throughput by measuring the download time of the last chunk, and selects the highest encoding bitrate lower than the estimated throughput. We have further implemented three other DASH algorithms (FESTIVE [24], BBA [23] and BBA-C) in the GPAC player. FESTIVE is a representative throughput-based DASH algorithm providing better robustness, fairness, and stability. BBA is the original buffer-based adaptation algorithm, and we have implemented its full version (BBA-2). BBA-C is our modified version of BBA-2 with less aggressive bandwidth usage as described in §5.2.2. For all four DASH algorithms, we have implemented their MP-DASH adapters with from around 30 to 100 lines of C code. This confirms the easy integration between the MP-DASH framework and existing off-the-shelf DASH algorithms.

We have also built a **Multipath Video Analysis Tool**, consisting of about 3,000 lines of C++ code. Its purpose is to facilitate our evaluation by analyzing key metrics of video streaming over multipath, such as path utilization, rebuffering, video quality switch, and energy consumption. It takes as input a network packet trace containing the video content, as well as a player's event logs. It then correlates them to perform the analysis. A key strength of this tool is it understands multiple protocol layers including MPTCP, TLS (if encryption is used), HTTP, and DASH. It also visualizes the analysis as illustrated in Figure 8.

7. EVALUATION

We evaluate the MP-DASH scheduler and conduct extensive evaluations of MP-DASH enhanced video streaming through both controlled experiments and real field studies.

7.1 Methodology

We set up an MPTCP testbed using a client laptop and a server machine, both with Linux kernel 3.18.20. They all run the stable version of MPTCP v0.90.0 [31] enhanced with MP-DASH. The client is equipped with an external LTE dongle and a built-in WiFi interface. We use a dedicated 802.11n WiFi access point running on the 5 GHz frequency band, providing $<$1 ms RTT and $>$50 Mbps bandwidth between client and server. We use Dummynet [8] to configure the WiFi latency to be 50 ms, resembling the RTT experienced by typical metropolitan WiFi users [36]. Although Dummynet creates additional buffers, as we use it to control latency explicitly, it will not create severe bufferbloat problems on our testbed. The cellular path is provided by a major U.S. carrier's commercial LTE network with around 50-60 ms RTT. We set up an Apache2-based DASH server for the experiments in §7.3.

Our current MP-DASH scheduler aims at reducing cellular data usage. This also leads to reduced energy

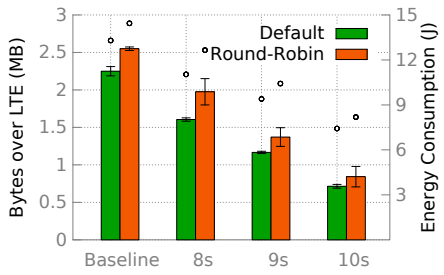


Figure 4: Comparing baseline MPTCP and MP-DASH with different deadlines using the default and round-robin schedulers.

consumption. Despite conducting the experiments on a laptop, we also want to understand the radio energy consumption, which accounts for at least half of the overall device energy consumption for a typical mobile device [30], if MP-DASH is deployed on a smartphone or tablet. The energy impact of MP-DASH on other components such as display is negligible, as it has almost no impact on the video quality. We use a simulation-based approach by feeding the collected network traces to a custom simulator with a recently proposed model [30]. This is the most comprehensive and up-to-date multipath radio energy model that considers both DRX and energy tails [21]. We compute the radio energy using two devices' parameters: Samsung Galaxy Note and Samsung Galaxy S III, both yielding similar results. We therefore report the first one. Note the above energy analysis is feasible because given the same network condition, MP-DASH incurs deterministic traffic pattern, which allows us to "replay" the trace under different power models for different types of devices. In our future work, we plan to directly measure the energy consumption of MP-DASH on smartphones.

7.2 Performance of MP-DASH Scheduler

We begin with evaluating the MP-DASH scheduler individually, using single file download as the workload. This evaluation (in particular, the trace-driven simulation) allows us to compare our online scheduling results with the optimal offline scheduling results.

7.2.1 *Experiments over Real WiFi and LTE*

We consider again the motivating example in §2.3. The client downloads from server 5MB data. We use Dummynet [8] to throttle the bandwidth of WiFi and cellular links to be 3.8 and 3.0 Mbps, respectively. Under this setup, it takes ∼10.5 seconds to download the 5MB data using WiFi alone and ∼6 seconds when using MPTCP. As a result, we experiment with three deadlines of 8, 9 and 10 seconds.

We evaluate the impact of the MP-DASH scheduler on both the default and round-robin MPTCP schedulers. We examine three metrics: download time, cellular bytes, and radio energy consumption. We plot the number of bytes over LTE using bars and mark the radio energy consumption as round dots in Figure 4 (averaged over 10 runs). MP-DASH significantly reduces the cellular data and radio energy usage, compared to unmodified MPTCP. The longer the deadline is, the more the saving is. When the deadline is 10 seconds, the saving is 68% for cellular data and 44% for radio energy.

Trace Name	File Size	Avg WiFi Bandwidth	Avg Cell Bandwidth
Synthetic (σ=10%)	5 MB	3.8 Mbps	3.0 Mbps
Synthetic (σ=30%)	5 MB	3.8 Mbps	3.0 Mbps
Fast Food B	20 MB	5.2 Mbps	8.1 Mbps
Coffeehouse D	5 MB	1.4 Mbps	7.6 Mbps
Office	50 MB	28.4 Mbps	19.1 Mbps

Table 1: WiFi/cellular bandwidth and file sizes for simulation.

Trace Name	D/L sec.	Cell % Optimal	Cell % Online	Diff.	Miss?
SYNTH σ=10%	8	24.77%	27.13%	2.36%	No
	9	15.29%	19.17%	3.88%	No
	10	6.16%	11.74%	5.58%	No
SYNTH σ=30%	8	26.32%	30.79%	4.47%	No
	9	16.88%	24.32%	7.44%	No
	10	8.48%	16.66%	8.18%	No
FastFood	15	60.49%	61.82%	1.33%	No
	20	48.52%	52.79%	4.27%	No
	25	38.09%	42.99%	4.90%	No
	30	26.91%	33.13%	6.22%	No
Coffee	5	83.26%	81.22%	-2.04%	10ms
	10	65.03%	67.18%	2.15%	No
	15	47.86%	55.48%	7.62%	No
	20	30.48%	35.52%	5.04%	No
Office	9	27.21%	27.48%	0.57%	No
	12	12.98%	15.46%	2.48%	No
	15	0.00%	3.82%	3.82%	No
	18	0.00%	0.00%	0.00%	No

Table 2: Comparing MP-DASH's online algorithm with the optimal using trace-driven simulation. "SYNTH" stands for Synthetic and "D/L" stands for Deadline.

All data transfers finish before the deadline and the average finish time is close to the deadline. However, under the scenarios with high bandwidth prediction errors, the deadline might be missed (after that both interfaces will always be used) and we show in §7.2.2 this is unlikely in practice. Missing a deadline can further be prevented by tuning the α parameter in Algorithm 1. In the above experiments, we set α to be 1. We also experiment with different α values when the deadline is 10 seconds. Even when α is 0.8, the MP-DASH scheduler achieves savings for cellular data and radio energy of 28% and 15%, respectively.

7.2.2 *Trace-Driven Simulation*

We also conduct a trace-driven simulation of the MP-DASH scheduler to understand its performance under realistic network conditions with throughput prediction errors, compared with the optimal case where the throughput is perfectly known.

The simulation methodology is as follows. First, we create different bandwidth profiles listed in Table 1. We

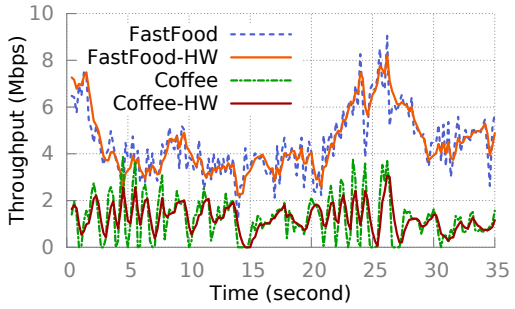


Figure 5: Two bandwidth traces and their prediction results.

Video quality level	1	2	3	4	5
Big Buck Bunny	0.58	1.01	1.47	2.41	3.94
Red Bull Playstreets	0.50	0.89	1.50	2.47	3.99
Tears of Steel	0.50	0.81	1.51	2.42	4.01
Tears of Steel HD	1.51	2.42	4.01	6.03	10.0

Table 3: Average encoding bitrates (in Mbps) for four DASH videos.

generate two synthetic profiles with the average WiFi and cellular bandwidth of 3.8 Mbps and 3.0 Mbps, and the standard deviation of instantaneous throughput being σ=10% and 30% of the average. We also collect real traces from three public locations. Next, we build a simulator that performs discrete-time simulation of Algorithm 1 (with α set to 1) and the Holt-Winter algorithm with the length of each time slot being the round-trip time (we use 50 ms for the synthetic traces). We then run the simulator on all bandwidth profiles with different combinations of deadline and file size, and summarize the results in Table 2. The file size is determined by the average aggregated bandwidth and download deadline. The key metrics are whether the deadline is missed, and the fraction of cellular bytes. The "Cell % (Optimal)", "Cell % (Online)", and "Diff." columns correspond to the solution with the perfect knowledge of bandwidth, our online algorithm, and their difference, respectively. The optimal solution always incurs the minimum cellular usage without missing the deadline.

We discuss our findings. First, the online MP-DASH scheduler tends to be conservative even with α set to 1. For the vast majority of cases, the inaccuracy in bandwidth estimation results in using more cellular data instead of missing the deadline. This is likely attributed to the nature of the bandwidth patterns. Deadlines are missed only when the WiFi throughput drops steeply and continuously; while in reality, WiFi bandwidth tends to be fluctuating, as exemplified in Figure 5. The second observation is the good performance of the online algorithm, whose additional cellular usage is consistently small (less than 10% of the total transfer size, as shown in the "Diff." column).

7.3 Evaluation of the MP-DASH Framework

We now evaluate the overall MP-DASH framework using four videos from a public dataset [26] listed in Table 3. In the rest of this section, unless otherwise mentioned, we show the results of only *Big Buck Bunny* due to space limit (the other two non-HD videos yield qualitatively similar results). We study the HD version of *Tears of Steel* in §7.3.5.

	Default	700 K	1000 K	MP-DASH
Cell Bytes	96.16	33.41	45.74	18.51
% of Cell Data	43.62	15.94	20.70	8.373
Radio Energy	641.6	855.0	834.6	281.9

Table 4: Cellular throughput-throttling vs MP-DASH. Cellular Bytes are in MB and Radio Energy is in Joule.

All presented results use chunk duration of 4 seconds. We also experiment with 6 and 10-second video chunks, and obtain similar results. The length of each video is 10 minutes. Thus, for 4-second chunk duration, a full playback consists of 150 chunks each being selected from 5 quality levels. In all experiments, we report various statistics for the last 80% chunks, when the player is in its steady state. The statistics for all chunks show very similar results. We consider four evaluation metrics: number of stalls, playback bitrate, cellular data usage, and radio energy consumption (computed using the way described in §7.2). In our evaluations, no stall occurs. Therefore we focus on the last three metrics. We set α in Algorithm 1 to be 1. As we have shown in §7.2, using a smaller value of α will lead to less savings of cellular data and radio energy.

7.3.1 *Inefficiency of Throughput Throttling*

We first examine the alternative solution mentioned in §4: simply throttling the cellular path. We quantitatively compare three approaches using the default GPAC rate adaptation algorithm: the unmodified MPTCP, MPTCP with cellular being throttled, and MPTCP with MP-DASH (rate-based deadline setting). We use the same network setting as that in §7.2 where the WiFi and cellular throughputs are 3.8 and 3.0 Mbps, respectively. We throttle the cellular throughput at 200, 700 and 1000 Kbps. For the first two settings, as the actual MPTCP throughput is usually smaller than the available bandwidth due to, for example, TCP congestion control, a large number of the downloaded chunks ($> 22\%$) are not encoded at the highest bitrate level. In the real-world settings, the throttling approach is further complicated by the fluctuating WiFi throughput, making it difficult to adjust the cellular throttling cap accordingly in real time.

Table 4 summarizes the bytes and percentage of cellular data, as well as the radio energy consumption under four configurations: the default MPTCP, MPTCP with 700 Kbps cellular throttling, 1000 Kbps throttling, and MPTCP with MP-DASH. Although throttling the cellular path reduces the cellular traffic, it pays a significant penalty for high radio energy consumption. In contrast, MP-DASH achieves both the lowest cellular usage and radio energy. Figure 6 visualizes the traffic patterns of three configurations in Table 4: throttling at 700 Kbps (top), MP-DASH (middle), and the default MPTCP (bottom). In the throttling scheme, LTE is slowly "dribbling" data, leading to considerable radio energy waste [33]. The default MPTCP incurs a traffic

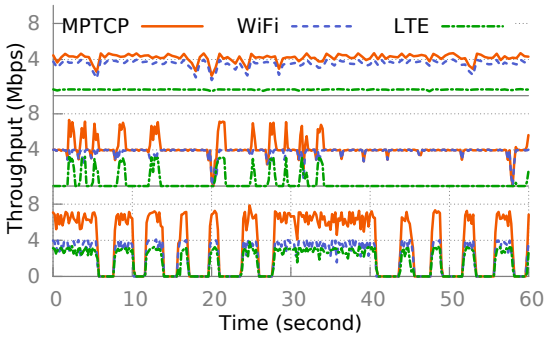


Figure 6: Traffic patterns of MPTCP with 700 Kbps cellular throttling (top), MP-DASH (middle), and default MPTCP (bottom).

pattern similar to that in Figure 1 with heavy cellular usage. For MP-DASH, cellular is not used during most of the time, and is only consumed adaptively when the deadline cannot be met using WiFi alone.

7.3.2 *Controlled Experiments*

We now evaluate MP-DASH enhanced players using controlled experiments under different bandwidth combinations of WiFi and LTE. We consider three network conditions: (1) WiFi 3.8 Mbps, LTE 3.0 Mbps, (2) WiFi 2.8 Mbps, LTE 3.0 Mbps, and (3) WiFi 2.2 Mbps, LTE 1.2 Mbps. For (1) and (2), the MPTCP throughput is higher than the highest video encoding rate (3.94 Mbps); while for (3) the overall network capacity cannot support the highest video quality. Note that using higher LTE throughput will lead to more savings of cellular data brought by MP-DASH.

We investigate four DASH algorithms: FESTIVE, GPAC, BBA, and BBA-C with and without MP-DASH. We observe no stalls in all experiments, and MP-DASH has no impact on the average playback rate. Thus, we focus on the metrics of cellular data usage and radio energy consumption. Next, we describe the results for each DASH algorithm.

FESTIVE AND GPAC. Figure 7a plots the results of FESTIVE. In each group of network conditions, "Baseline", "Duration", and "Rate" correspond to MPTCP without MP-DASH, MP-DASH with duration-based deadline setting, and MP-DASH with rate-based deadline scheme, respectively. The cellular data usage is shown in bars and the energy consumption is marked as round dots. We observe that compared to the default MPTCP, MP-DASH significantly reduces both cellular data and radio energy. When WiFi bandwidth drops from 3.8 to 3.0 Mbps, the savings become less because more cellular data will be required by the player. The savings increase for the worst network condition (W2.2/L1.2) because the chunk quality drops from Level 5 to 4 (Table 3) and the WiFi throughput is close to the average bitrate of Level 4 (2.41 Mbps). For GPAC, which is also a throughput-based DASH algorithm, we observe qualitatively similar results.

Figure 7a shows the rate-based deadline setting outperforms the duration-based one. Recall in §5.1 that in these two settings, the cellular usage is budgeted based on the average video bitrate and a particular chunk's bitrate,

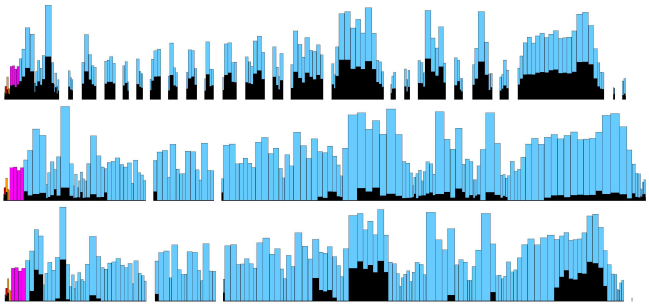
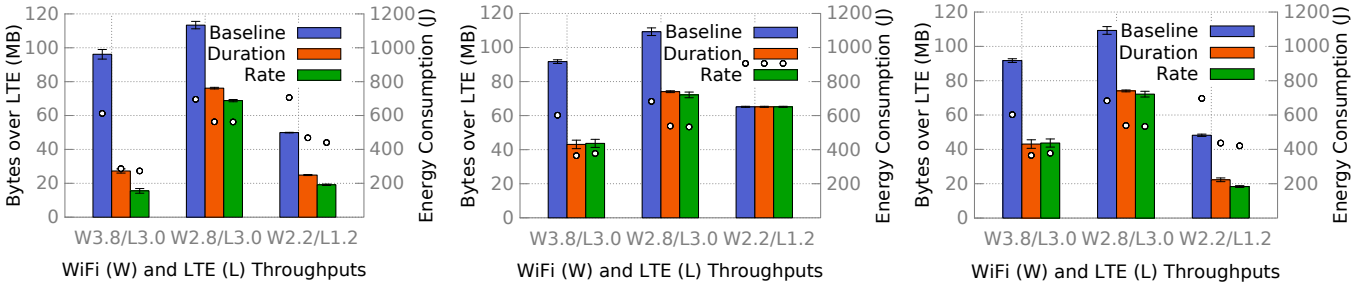


Figure 8: Visualization results produced by our multipath video analysis tool (§6). The X axis is the time. Each bar is a chunk. Its height and width represent its size and download duration, respectively. The color of a chunk is its bitrate level (1 to 5) with light blue being the highest level. The black component of each chunk represents the fraction that is delivered through cellular. The top, middle, and bottom subfigures correspond to the default MPTCP, MP-DASH with rate-based deadline selection, and MP-DASH with duration-based deadline selection, respectively, using FESTIVE.

respectively. Therefore, for chunks with higher-than-average encoding bitrate, the duration-based setting requires more cellular data than the rate-based one. This is visualized in Figure 8 by our analysis tool (§6). The figure also illustrates that MP-DASH eliminates most of the idle gaps appearing in the default MPTCP case (the top subfigure).

BBA. Figure 7b shows the results for BBA. For the first two groups of network conditions, MP-DASH brings less savings than it does for FESTIVE, because BBA itself is known to be more aggressive, leaving less saving opportunities for MP-DASH. Note MP-DASH is *adaptive* in that it saves resources on the premise that the application's bandwidth requirement is satisfied. The interesting case is the W2.2/L1.2 scenario, where MP-DASH brings no cellular data saving. The reason, as explained in §5.2.2, is that the original BBA algorithm is even more aggressive when the overall network capacity cannot support the highest encoding bitrate. In this case, the fetched chunk bitrate may oscillate as illustrated in Figure 3, worsening users' watching experience.

BBA-C. We plot the results for BBA-C in Figure 7c. BBA-C is the cellular-friendly version of BBA that forces the selected bitrate to be no higher than the actual network capacity, to prevent the bitrate oscillation. For the first two groups of network conditions, BBA and BBA-C have the same results, because the overall bandwidth is capable of supporting the highest bitrate and thus the bitrate capping feature of BBA-C is not invoked. The difference is the W2.2/L1.2 scenario where BBA-C effectively prevents the bitrate oscillation by locking the highest bitrate to Level 4. This provides room for MP-DASH to save cellular data. By comparing Figure 7b and 7c, we observe that BBA-C with MP-DASH reduces cellular data usage and radio energy consumption by 69% and 50%, respectively, at the cost of reducing the playback bitrate from oscillating between



(a) FESTIVE (b) BBA (c) BBA-C

Figure 7: Resource savings of MP-DASH under three network conditions for FESTIVE, BBA, and BBA-C (Baseline is the vanilla MPTCP). Throughput is in Mbps.

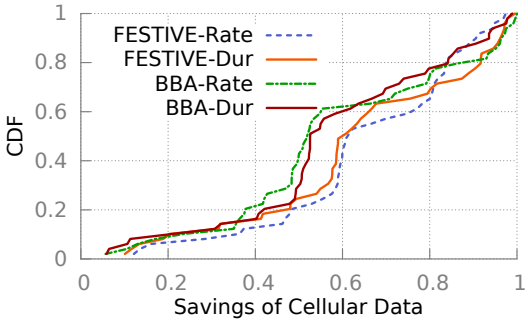


Figure 9: Cellular data savings brought by MP-DASH in all locations.

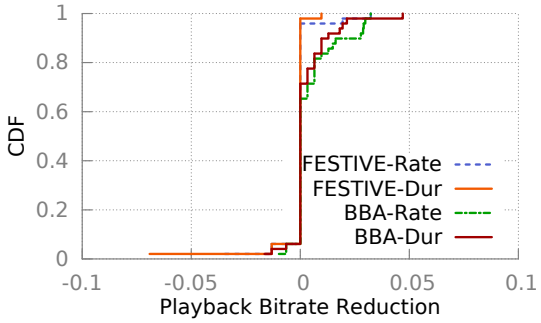


Figure 10: Playback bitrate reduction of experiments in all locations.

Levels 4 and 5 to constantly being at Level 4 (29% reduction of average playback bitrate). We therefore believe BBA-C strikes a good tradeoff among resource usage, playback bitrate, and playback smoothness for mobile video.

7.3.3 MP-DASH in Real-World Settings

We now investigate how MP-DASH helps make DASH video over MPTCP user-preference-aware under realistic settings through extensive in-field experiments. We conduct the experiments multiple times at 33 locations in three U.S. states, at different times of a day and during both weekdays and weekends. The locations include airport, hotel, fast food restaurant, shopping mall, retailer store, food market, coffeehouse, electronics store, parking lot, office building, grocery store, public library, *etc.* (described in §2). We use the free public WiFi networks, and the LTE provided by a large commercial carrier in the U.S. During the course of 3 months, we have collected more than 150 GB of data. We focus on two DASH algorithms: FESTIVE and BBA, and experiment with each algorithm under three configurations: vanilla MPTCP and MP-DASH with rate-based and duration-based deadline settings. A group of experiments thus consists of playing the *Big Buck Bunny* video under these six schemes in a random order. BBA-C performs the same as BBA for the experiments in this section, as the MPTCP capacity is higher than the highest video bitrate. We evaluate the performance of BBA-C for HD videos in §7.3.5.

Figure 9 plots the distribution of cellular data savings brought by MP-DASH at the above locations for four streaming schemes. Across all experiments, the 25th, 50th, and 75th percentile of the savings are 48%, 59%, and 82%, respectively. In general, MP-DASH achieves more savings for FESTIVE than it does for BBA. We observe similar results for radio energy consumption savings (figure not shown): the 25th, 50th, and 75th percentiles are 7.7%, 17%, and 53%, respectively. There is no playback bitrate reduction for 82.65% of the experiments, as shown in Figure 10. For the rest experiments, the average reduction is only 2.5%. Note a negative reduction means the increase of bitrate. The results demonstrate MP-DASH can effectively reduce the cellular data usage and radio energy for DASH video streaming over multipath, compared with using the vanilla MPTCP.

To gain more insights of the results, for each of the three categories described in §2.2, we pick several representative results and list them in Table 5 where groups (in the order of scenarios 1, 2, and 3 defined at the beginning of §2.2) are separated by horizontal lines. The numbers in the last 8 columns of the table are the savings in % of cellular bytes or radio energy compared to using the vanilla MPTCP without MP-DASH. The overall observation is, MP-DASH provides more savings as the WiFi throughput increases. For scenario 2 (WiFi can sometimes play the best quality), although the average WiFi throughput (based on our measurements before playing a video) is higher than the encoding bitrate of

	WiFi		LTE		FESTIVE/Bytes		FESTIVE/Energy		BBA/Bytes		BBA/Energy	
Location	BW	RTT	BW	RTT	Rate	DUR	Rate	DUR	Rate	DUR	Rate	DUR
Hotel Hi	2.92	14.1	11.0	51.9	52.92	49.43	9.735	29.74	41.74	40.74	23.52	35.74
Hotel Ha	2.96	40.8	14.0	68.6	59.25	57.68	11.68	19.53	49.54	47.84	15.50	13.13
Food Market	3.58	75.4	22.9	53.4	59.95	66.70	7.083	21.56	74.93	55.71	12.32	12.64
Airport	5.97	32.2	12.1	67.3	84.00	65.53	42.53	25.11	37.75	74.00	11.85	27.49
Coffeehouse	6.04	28.9	18.1	69.0	80.05	91.65	44.42	48.73	79.80	78.92	41.49	40.53
Library	17.8	23.3	5.18	64.1	97.44	97.81	77.78	85.19	95.12	97.54	80.29	79.72
Elec. Store	28.4	10.8	18.5	59.4	91.24	99.56	65.43	82.48	86.15	93.58	58.91	68.43

Table 5: Savings of cellular data and radio energy consumption at locations with different network conditions (BW in Mbps, RTT in ms, and savings in %). "DUR" stands for Duration.

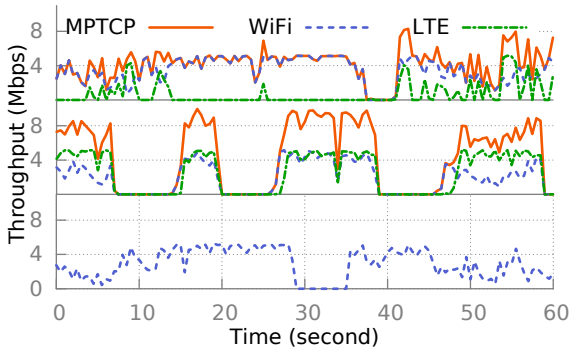


Figure 11: Traffic patterns of mobility experiments with MP-DASH (top), default MPTCP (middle), and single path WiFi (bottom).

the highest quality, using only WiFi cannot always support the highest bitrate, as the throughput of open WiFi networks is not always stable during the 10-minute playback period (figure not shown). Therefore, to ensure smooth playback at the highest quality, MPTCP still needs to use cellular from time to time. For scenario 3 (WiFi can almost always stably support the highest bitrate), one may argue that it is not necessary at all to use MPTCP. This is indeed true. However, it is inconvenient to ask users or video players to switch between single-path and multipath. Instead, MP-DASH provides a general OS support for applications to *transparently* and *adaptively* switch between them.

7.3.4 *MP-DASH under Mobility Scenario*

To evaluate the performance of MP-DASH under the mobility scenario, we walk around a WiFi AP following the same pre-defined route with the laptop playing the video using the FESTIVE rate adaptation algorithm. The WiFi and cellular bandwidth are both around 5.0 Mbps. Figure 11 plots samples of 60-second throughput for three configurations: MPTCP enhanced by MP-DASH with rate-based deadline setting (top), the default MPTCP (middle), and using only WiFi (bottom). It shows that MP-DASH can adaptively use cellular only when the WiFi throughput drops as the laptop moves far from the AP, whereas the default MPTCP drives the cellular link to almost its full capacity, regardless of the WiFi throughput. As a result, MP-DASH saves cellular data usage and radio energy consumption by 81.43% and 47.30%, respectively, without reducing the video bitrate (all chunks are played at the highest bitrate). Using only WiFi, on the other hand, cannot sustain the highest video bitrate for more than half of the chunks.

	FESTIVE	BBA-C
Playback Bitrate	↑ by 20.92%	↓ by 3.006%
Cellular Data Saving	39.88%	37.47%
Radio Energy Saving	8.624%	10.40%

Table 6: HD video results with the rate-based deadline setting.

7.3.5 *High Definition (HD) Video*

In all experiments above, the highest video quality is ∼4.0 Mbps. We repeat the experiments by using the HD version of the *Tears of Steel* video whose highest bitrate is 10.0 Mbps as listed in Table 3. Note that the emerging 4K videos and 360-degree videos may require an even higher bandwidth [32]. This experiment complements the experiments in §7.3.3 because when playing the HD video, even the aggregated bandwidth of WiFi and LTE cannot sustain the highest bitrate at several locations. In this case, the video is mostly played at bitrate levels 3 & 4, thus allowing us to assess the performance of BBA-C in the wild. Table 6 summarizes the results at one of such locations (a supermarket). MP-DASH still achieves significant savings of cellular bytes (40% for FESTIVE and 37% for BBA-C over the unmodified BBA) and radio energy, compared to the vanilla MPTCP. Counter-intuitively, for FESTIVE, using MP-DASH actually leads to increased playback bitrate. This is likely attributed to the fact that the throughput estimation at the transport layer is more accurate than at the application layer.

8. DISCUSSION

General Applicability of the MP-DASH Scheduler. It is important to note the deadline-aware data transfer provided by MP-DASH is generally applicable to diverse application scenarios besides video streaming. They bear a common characteristic of being delay tolerant: as long as data transfers can finish before their deadline, the user experience will not be impacted. Consider the following examples.

• For music apps using automated recommendation (*e.g.,* Pandora Music), players do not need the next song until the playback of the current song is close to its end.

• For turn-by-turn navigation, a map tile only needs to be fetched before the vehicle is close to the tile's location.

• Mobile data offloading can be enhanced by leveraging many applications' delay-tolerant nature [7, 17, 27].

In the above scenarios, we can use MPTCP to boost the system performance, and use the MP-DASH scheduler to reduce the cellular data usage without sacrificing QoE. Note that even for applications such as turn-by-turn navigation which do not generate a significant amount of data, MP-DASH can help offload cellular traffic to WiFi networks when possible while bringing the benefits of multipath.

Deployment of MP-DASH requires upgrading MPTCP. This can be realized by system updates routinely launched by mobile carriers or device vendors. To use MP-DASH, the video rate control algorithms also need to be upgraded. As described in §6, the modifications to video players are very light. Since almost all of MP-DASH's logic resides on the client side, the change required on the server side is minimal: only its MPTCP stack needs to support MP-DASH. No change to the video server application is required. Also, by using standard TCP splitting proxies with MP-DASH enabled MPTCP, we can make MP-DASH fully transparent to video servers. The proxy is TLS/SSL friendly as it runs at the transport layer and therefore does not need to perform any decryption.

Runtime Overhead. MP-DASH incurs negligible runtime overhead, as both the MP-DASH's scheduling algorithm itself and the Holt-Winters throughput prediction has low complexity. We have measured the CPU utilization with and without MP-DASH and have not noticed any measurable increase of CPU usage.

Limitations. We conduct all experiments on a laptop instead of a smartphone. We expect though the impact of device selection to be small given the low runtime overhead of MP-DASH. Other limitations include using a lab server instead of a production server (*e.g.,* YouTube) for experiments, having not performed tests on other cellular technologies such as 3G/HSPA, and having not evaluated other DASH algorithms such as MPC [42]. We plan to address them in our future work.

9. RELATED WORK

Mobile MPTCP. MPTCP has been recently investigated for mobile devices with multiple network interfaces. Chen *et al.* [10] and Deng *et al.* [13] characterized the performance of MPTCP on WiFi and cellular networks, both focusing on file download. The mobile kibbutz [29] consolidates cellular traffic from multiple users through MPTCP, leading to reduced latency and improved energy efficiency. Croitoru *et al.* [12] leveraged MPTCP to achieve seamless mobility among multiple WiFi APs. eMPTCP [28] combines delayed subflow establishment and power-aware subflow management to make MPTCP energy efficient. In our previous work, we measured the performance of mobile web over MPTCP for both HTTP and SPDY [18]. Differently from the above works, we propose a novel multipath framework for video streaming.

Recently, Corbillon *et al.* [11] proposed a cross-layer MPTCP scheduler to prioritize video packets for non-adaptive video streaming and evaluated its performance through a simulation study. ADMIT [40] is an analytical framework that leverages forward error correction to improve the QoE for non-adaptive video over MPTCP. Instead of improving the quality of video streaming, our goal is to reduce cellular data usage, while not impacting user perceived QoE. Moreover, we investigate the interaction of *adaptive* video streaming and MPTCP though MP-DASH and extensive experiments in the wild.

Adaptive Video Streaming. There is a plethora of work on understanding and improving the QoE for adaptive video streaming. Besides FESTIVE [24] and BBA [23], Tian and Liu [37] proposed rate control algorithms that explore the smoothness and responsiveness trade-off in DASH and use video buffer level at the client side as a feedback signal. Huang *et al.* [22] measured the performance of three popular adaptive video streaming services (Hulu, Netflix, and Vudu) for motivating BBA [23]. Yin *et al.* [42] developed a control-theoretic approach to understand the fundamental trade-offs between various strategies for DASH rate adaptation algorithms. AVIS [9] is a resource management framework that separates adaptive video streams from regular ones through cellular resource virtualization. PBA [43] improves video QoE by leveraging bandwidth predictions and it can be further enhanced through a combination with rate stabilization functions. piStream [41] improves the QoE of adaptive video streaming by taking advantage of the PHY information of LTE networks for accurate bandwidth estimation. Compared to them, we tweak the MPTCP schedulers to save cellular data usage and energy consumption for video streaming on mobile devices.

10. CONCLUSION

We conducted, to the best of our knowledge, a first study of preference-aware multipath for adaptive video streaming. We proposed the MP-DASH framework and instantiated it in the context of optimizing video delivery over WiFi and cellular networks. We also designed MP-DASH adapters for two representative categories of DASH algorithms (throughput-based and buffer-based) by making them multipath-friendly, and we seamlessly integrated the MP-DASH adapters into them. We demonstrated through extensive field studies at 33 public places in three far-apart U.S. states that MP-DASH enhanced video players are robust and adaptive. Compared with players using the off-the-shelf MPTCP, they dramatically reduce the cellular data usage and radio energy consumption without sacrificing users' QoE. We plan to open source our MP-DASH implementation in the near future.

Acknowledgements

We thank Rittwik Jana, Emir Halepovic, and the anonymous reviewers for their valuable comments and suggestions, and in particular Adrian Perrig for shepherding the paper. We thank Yeon-sup Lim, Xiufeng Xie, Xinyu Zhang, and Darijo Raca for their help to implement MP-DASH. Feng Qian's research was supported in part by National Science Foundation Grant CNS-1566331.

11. REFERENCES

[1] About Hotel WiFi Test. https://www.hotelwifitest.com/about/.

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

[3] Apple HTTP Live Streaming. https://developer.apple.com/streaming/.

[4] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020 White Paper. http://goo.gl/ylTuVx.

[5] GPAC DASH Video Player. https://gpac.wp.mines-telecom.fr/.

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

[7] A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting Mobile 3G Using WiFi. In *MobiSys*, 2010.

[8] M. Carbone and L. Rizzo. Dummynet Revisited. *ACM SIGCOMM Computer Communication Review*, 40(2):12–20, Apr. 2010.

[9] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang. A Scheduling Framework for Adaptive Video Delivery over Cellular Networks. In *MobiCom*, 2013.

[10] Y.-C. Chen, Y.-S. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley. A Measurement-based Study of MultiPath TCP Performance over Wireless Networks. In *IMC*, 2013.

[11] X. Corbillon, R. Aparicio-Pardo, N. Kuhn, G. Texier, and G. Simon. Cross-Layer Scheduler for Video Streaming over MPTCP. In *MMSys*, 2016.

[12] A. Croitoru, D. Niculescu, and C. Raiciu. Towards WiFi Mobility without Fast Handover. In *NSDI*, 2015.

[13] S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan. WiFi, LTE, or Both? Measuring Multi-homed Wireless Internet Performance. In *IMC*, 2014.

[14] A. Dua, C. W. Chan, N. Bambos, and J. Apostolopoulos. Channel, Deadline, and Distortion (CD^2) Aware Scheduling for Video Streams Over Wireless. *IEEE Transactions on Wireless Communications*, 9(3):1001–1011, Mar. 2010.

[15] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824, 2013.

[16] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[17] B. Han, P. Hui, V. S. A. Kumar, M. V. Marathe, J. Shao, and A. Srinivasan. Mobile Data Offloading through Opportunistic Communications and Social Participation. *IEEE Transactions on Mobile Computing*, 11(5):821–834, May 2012.

[18] B. Han, F. Qian, S. Hao, and L. Ji. An Anatomy of Mobile Web Performance over Multipath TCP. In *CoNEXT*, 2015.

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

[20] B. Hesmans, G. Detal, S. Barré, R. Bauduin, and O. Bonaventure. SMAPP: Towards Smart Multipath TCP-enabled APPlications. In *CoNEXT*, 2015.

[21] 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 *MobiSys*, 2012.

[22] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard. In *IMC*, 2012.

[23] 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 *SIGCOMM*, 2014.

[24] J. Jiang, V. Sekar, and H. Zhang. Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE. In *CoNEXT*, 2012.

[25] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi. FatVAP: Aggregating AP Backhaul Capacity to Maximize Throughput. In *NSDI*, 2008.

[26] S. Lederer, C. Müller, and C. Timmerer. Dynamic Adaptive Streaming over HTTP Dataset. In *MMSys*, 2012.

[27] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong. Mobile Data Offloading: How Much Can WiFi Deliver? In *CoNEXT*, 2010.

[28] Y.-S. Lim, Y.-C. Chen, E. M. Nahum, D. Towsley, R. J. Gibbens, and E. Cecchet. Design, Implementation and Evaluation of Energy-Aware Multi-Path TCP. In *CoNEXT*, 2015.

[29] C. Nicutar, D. Niculescu, and C. Raiciu. Using Cooperation for Low Power Low Latency Cellular Connectivity. In *CoNEXT*, 2014.

[30] A. Nika, Y. Zhu, N. Ding, A. Jindal, Y. C. Hu, X. Zhou, B. Y. Zhao, and H. Zheng. Energy and Performance of Smartphone Radio Bundling in Outdoor Environments. In *WWW*, 2015.

[31] C. Paasch, S. Barré, et al. Multipath TCP in the Linux Kernel. http://www.multipath-tcp.org.

[32] F. Qian, B. Han, L. Ji, and V. Gopalakrishnan. Optimizing 360 Video Delivery Over Cellular Networks. In *All Things Cellular*, 2016.

[33] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Profiling Resource Usage for Mobile Applications: a Cross-layer Approach. In *MobiSys*, 2011.

[34] P. S. Schmidt, T. Enghardt, R. Khalili, and A. Feldmann. Socket Intents: Leveraging Application Awareness for Multi-Access Connectivity. In *CoNEXT*, 2013.

[35] I. Sodagar. The MPEG-DASH Standard for

Multimedia Streaming Over the Internet. *IEEE MultiMedia*, 18(4):62–67, Apr. 2011.

[36] J. Sommers and P. Barford. Cell vs. WiFi: On the Performance of Metro Area Mobile Connections. In *IMC*, 2012.

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

[38] B. Vamanan, J. Hasan, and T. N. Vijaykumar. Deadline-Aware Datacenter TCP (D^2TCP). In *SIGCOMM*, 2012.

[39] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *NSDI*, 2011.

[40] J. Wu, C. Yuen, B. Cheng, M. Wang, and J.-L. Chen. Streaming High-Quality Mobile Video with Multipath TCP in Heterogeneous Wireless Networks. *IEEE Transactions on Mobile Computing*, 15(9):2345–2361, Sept. 2016.

[41] X. Xie, X. Zhang, S. Kumar, and L. E. Li. piStream: Physical Layer Informed Adaptive Video Streaming Over LTE. In *MobiCom*, 2015.

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

[43] 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 *HotMobile*, 2015.