# IMPROVING QUALITY OF EXPERIENCE FOR MOBILE VIDEO STREAMING

A Dissertation
Presented to
The Academic Faculty

by

Lateef Yusuf

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology
May 2014

# IMPROVING QUALITY OF EXPERIENCE FOR MOBILE VIDEO STREAMING

Approved by:


Professor Umakishore Ramachandra,
Committee Chair
School of Computer Science
*Georgia Institute of Technology*

Professor Umakishore Ramachandran,
Advisor
School of Computer Science
*Georgia Institute of Technology*

Professor Mostafa Ammar
School of Computer Science
*Georgia Institute of Technology*

Professor Patrick Traynor
School of Computer Science
*Georgia Institute of Technology*

Professor Hyesoon Kim
School of Computer Science
*Georgia Institute of Technology*

Ignacio Solis, Ph.D.
Content-Centric Networking Group
*PARC*



Date Approved: April 7, 2014

*To my parents,*

*Halima and Yahya Yusuf*

# ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Professor Umakishore Ramachandran, for his guidance and mentorship throughout my graduate education. I am indebted to his support and words of encouragements on both academic and non-academic issues. I also thank my committee members: Dr. Mostapha Ammar, Dr. Hyesoon Kim, Dr. Patrick Traynor, and Dr. Ignacio Solis for their feedback to improve my dissertation and for their valuable suggestions on my research directions.

I am grateful to my friends and colleagues in the Embedded Pervasive Lab who assisted me and helped me grow tremendously. In particular, I would like to thank Moonkyung Ryu, Kirak Kong, Dushmanta Mohapatra, Dave Lillethun, Junsuk Shin, Dr. Hyojun Kim and Dr. David Hilley. My thanks are also due to the masters students and the undergraduate students that worked with me on different projects especially Bhavana Chowdary, Kyle Kelly, Chayong Lee, Jinhyun Kim and Donovan Hatch.

My experience at Georgia Tech has been enriched through interactions and discussions with friends. I want to thank Ogechi Nnadi, Dr. Ogundiran Soumonni, Dr. Kayode Olarenwaju, Bobby Onofik, Temi Yembra and Trudy Padmore. Outside of Georgia Tech, I want to acknowledge my friends across the US and beyond: Temitayo Olajide, Mayowa Sanwo, Taofeek Lawal, Fatahi Kadiri, Hafeez Ajayi, and Emmanuel Ogbokodo. In particular, I am indebted to Femi Otelaja and Supo Johnson, whose friendships have been invaluable during these years.

Finally, I want to thank my parents and siblings for their love and support. I am grateful to my mother and father for caring for me and for standing with me throughout these years. Most of all, I want to express my gratitude to my brother, Dr. Akeem Yusuf, for his support and for the fine example he gave me throughout my life.

# TABLE OF CONTENTS

# LIST OF TABLES

## LIST OF FIGURES

# SUMMARY

Thanks to their increasing sophistication and popularity, mobile devices, in the form of smartphones and tablets, have become the fastest growing contributors to Internet traffic. Indeed, smartphones are projected to account for 50% of global Internet traffic by 2017, with the share of mobile video increasing to about 40% of total Internet traffic. As users embrace Internet streaming of video, several studies have found that a small decrease in video quality leads to a substantial increase in viewer abandonment and disengagement rates. To handle the explosive growth in video traffic, Adaptive HTTP streaming, which exploits the prevalence of commodity web servers and content distribution networks, has emerged as the key technology for delivering video to end users. Although a number of systems have been proposed for HTTP video streaming in traditional environments and for fixed clients, existing platforms for video streaming on mobile devices are still in their infancy and do not address the additional challenges often experienced by mobile clients: high fluctuations in network conditions, heterogeneous networking interfaces, multiple form-factors, and limited battery life.

In this dissertation, we propose a number of solutions for improving the Quality of Experience of HTTP video streaming on mobile devices. We begin by evaluating the performance of several existing video quality adaptation schemes when deployed on mobile platforms. Through experiments with smartphones in wide-area environments, we assemble several key findings. First, we show that the high fluctuations in network throughput on cellular and Wi-Fi networks impose significant challenges for efficiently architecting the video adaptation scheme. Second, we find significant differences between the performance of the current state-of-the-art schemes in controlled experimental settings and their performance in mobile settings on key quality metrics such as inefficiency, instability, rebuffering ratio, and startup latency. We also find noticeable differences in the behavior of the schemes

under Wi-Fi and cellular network access, with most of the schemes performing worse when the network access is cellular. Given these observations, we hypothesize on the possible causes of these inefficiencies. We also identify the best practices of existing schemes and key insights from experimental results that can serve as foundations for addressing many of the limitations.

Armed with these measurement-driven insights, we propose a novel video quality adaptation scheme, called MASS, which is more robust to the vagaries of the wireless networking conditions. We implement and evaluate our solution on commodity Android smartphones, and demonstrate significant performance gains over existing schemes. To further improve the streaming experience, we introduce an extension to HTTP video streaming that leverages the synergy between social network participation and video streaming to optimize end-user Quality of Experience. Our system, called SDASH, integrates and applies well-known concepts such as cooperative caching, prefetching, and P2P streaming for reducing bitrate fluctuations and optimizing the viewing experience. Finally, we develop a general infrastructure for constructing temporally and spatially localized P2P communities of mobile devices sharing similar interests. The platform enables on-demand cooperation among mobile clients based on device context and client preferences. We use a concrete implementation of the mobile P2P infrastructure for evaluating the performance of SDASH.

This dissertation addresses the challenges facing Adaptive HTTP Streaming under mobile networking conditions. Through experimentation with commodity mobile devices, we show that the proposed techniques for bitrate adaptation and cooperative streaming can significantly improve the video viewing experience.

# CHAPTER I

# INTRODUCTION

## *1.1 Background*

The market penetration of mobile devices has increased significantly in recent years with more than 6 billion phones worldwide [43]. Mobile devices come in a number of form factors, functionality, and computing complexity (e.g., cell phones, smartphones, tablets, netbook, electronic readers, etc.). To enable local-area and wide-area communication, mobile devices are often equipped with multiple network interfaces supporting competing wireless standards and targeting various metrics such as price, performance, power, and range. Examples of popular network interfaces include WLAN (e.g., Bluetooth and Wi-Fi) and WWAN (e.g., GSM, UMTS, LTE, and WiMAX). Many smartphones and tablets are also equipped with rich sensors including cameras, GPS, compass, accelerometers, and RFID. These computing, networking and sensing technologies have fueled the wide adoption of mobile devices and a rapidly enlarging mobile application ecosystem. Figure 1 shows the increase in web traffic in recent years as well as predictions for the next several years, suggesting that the share of Internet traffic generated by mobile devices will increase to about 50% of the total network traffic by 2017.

Among the range of functionalities and applications now available to mobile devices, video streaming services are emerging as a significant fraction of total mobile usage. Attracted by the faster networks, ubiquitous smart devices and bigger phone screens, many providers such as YouTube [22], Hulu [14] and Netflix [17] have deployed streaming applications on all the major mobile platforms. According to a recent Cisco's Visual Networking Index [42], video traffic accounted for more than 50% of total mobile data traffic in 2012, and is expected to continue outpacing other forms of mobile traffic, rising up to 66% of total mobile data usage and 40% of global Internet traffic by 2017.

A number of technologies have been developed to handle the explosive growth in Internet

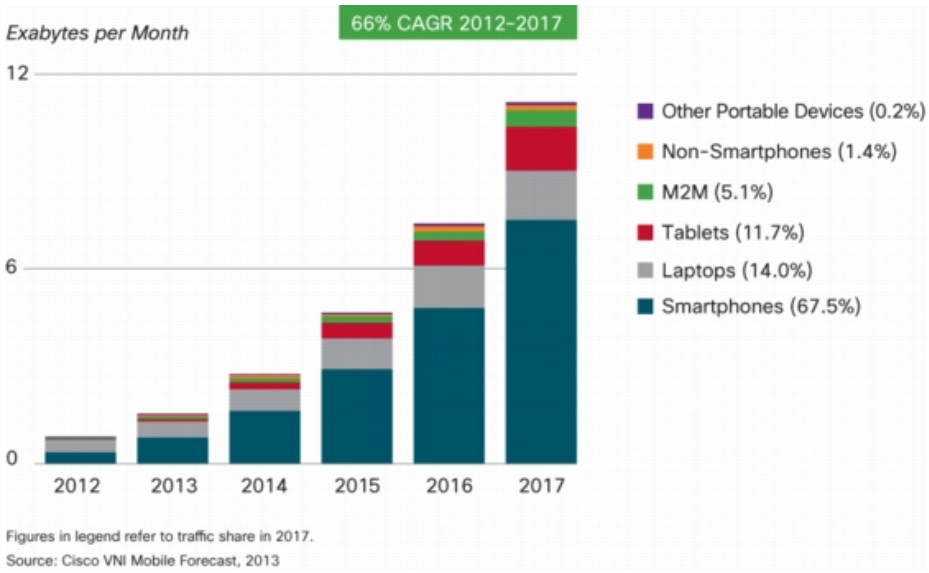



**Figure 1: Laptops and Smartphones Lead Internet Traffic Growth [42].**

video traffic. An example of such technologies is Adaptive HTTP Streaming or AHS. In AHS, a given video file is divided into several segments, which are encoded into a number of versions by adjusting the video bit rate, audio bit rate, video resolution, and/or frame rate. An XML file that lists the segments of the video and the encoded versions is then provided to clients interested in streaming the video. After the client fetches the initial video segment, it can dynamically switch to a different version to match the current network condition. A quality adaptation algorithm for AHS streaming can therefore adapt to changing conditions to optimize the video viewing experience (Figure 2).

However, the task of delivering high quality video to mobile devices faces significant challenges that are not experienced by traditional streaming on fixed devices: high fluctuations in network conditions, heterogeneous networking interfaces, multiple form-factors, and limited battery life. Many studies have documented wide variability in wireless and cellular network characteristics [70, 66, 56, 72]. As a mobile device moves from one location to the next, the network throughput, latency, round trip time, downlink and uplink bandwidth, error rates, and other network metrics can change drastically. Many studies have found significant differences in mean and standard deviation of these network conditions even

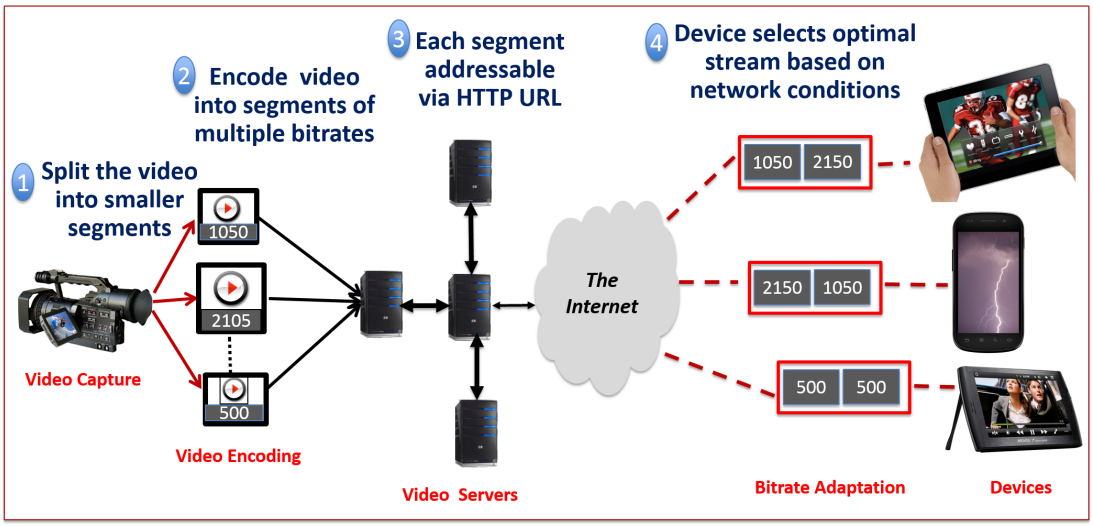



**Figure 2: Adaptive HTTP Streaming.**

within a small difference in location or time [120, 71]. The observed variability in networking conditions is due to a variety of factors intrinsic to the wireless medium (interference, multipath fading, etc.). Furthermore, the observed throughput from cellular connectivity is affected by spotty coverage and high service loss rate (which can be up to 50% [120]), leading to sudden drops in video quality and high frequency of rebufferings [24, 121, 63, 128]. Addressing these challenges is critical to improving the Quality of Experience for mobile video streaming.

## *1.2 Problem Statement*

Internet streaming of movies, YouTube clips, and other media have become a pervasive activity enjoyed by users across computing platforms, geographical locations, and time of the day. It is no wonder that video traffic accounts for a significant fraction of the bandwidth on the Internet. The proliferation of 3G cellular access, the development of 4G networks, and the deployment of Wi-Fi hot spots and access points have seen the promise of a new mobile computing landscape capable of providing high quality video to users.

Because of their ubiquity, mobile devices are advertised as "always-on" devices, leading

to common expectation by users to be able to access online services with minimal performance penalty wherever they are and whenever they desire. However, the promise of ubiquitous high quality video streaming on mobile devices has been largely unmet [58, 92, 88]. Many previous works have found a strong correlation between video quality and viewer behavior in terms of abandonment, engagement, and repeat viewership [82, 30, 50, 44, 96, 97]. In a recent study of Akamai's streaming network, the authors found that viewers start to abandon a video if it takes more than 2 seconds to begin playback, with each incremental delay of 1 second resulting in a 5.8% increase in the abandonment rate; and that a rebuffering delay of more than 1% of the video duration leading to 5% decrease in the length of playback [82]. Similar studies have found that high picture quality and low frequency of bitrate switching is essential for maximal user engagement and repeat viewership [30, 50]. Furthermore, mobile users have been found to be more sensitive to video quality than users on fixed clients, watching only about a third of the length of video viewed by fixed clients when facing similar degradation in video quality [58, 121]. Therefore, the video delivery system needs to provide a robust scheme for adapting to the network conditions in order to improve viewing experience and retain user engagement.

Also, video access on mobile devices differs from video consumption through traditional computing devices (see Figure 3). Smartphones are often equipped with video cameras, GPS, and other sensors, leading to the emergence of user-generated content and the increasing use of mobile platforms as both the sources and destinations for video traffic. Similarly, sharing and interactions of video links through "friends" in social networks is also a key driver of video consumption on mobile devices. There is also increasing dissemination of video content among mobile clients with shared interests in an *ad hoc* manner. This pattern of mobile video consumption provides different opportunities from video access on fixed computing devices. Unfortunately, existing video delivery technologies are designed for traditional Internet users, ignoring the unique demands of mobile multimedia access. This model ignores the cooperative nature of mobile video access, requiring all the devices to fetch the complete video segments from the originating servers. Given the increasing shift to mobile devices as the primary endpoints for consuming videos, we believe that there is a

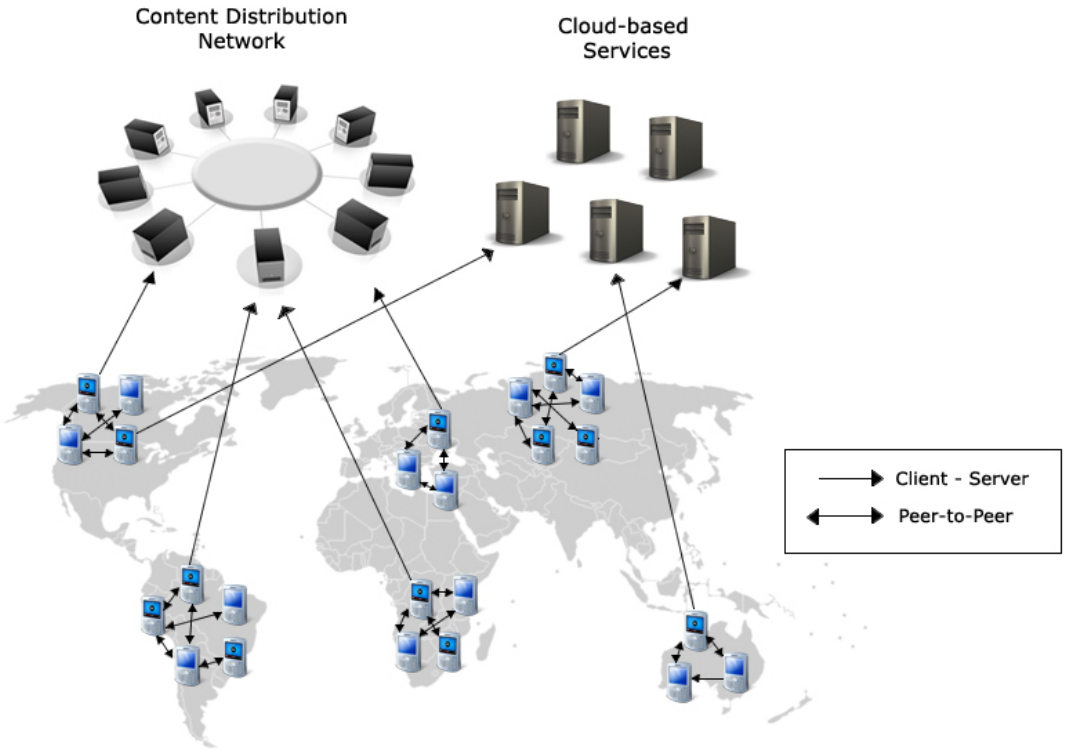



**Figure 3: Ever Expanding Reach and Volume of Video Traffic: Increasing growth in user generated content and device to device social media require a rethink of the system infrastructure.**

need to rethink the system infrastructure used for delivering video to end users.

### *1.3 Thesis Statement*

Rapid advancements in computing and communication technologies is enabling the adoption of mobile devices as the primary platform for consuming Internet video, but this trend introduces new challenges arising from the need to support heterogeneous devices with multiple networking interfaces, handle wide fluctuations in network coverage and quality of service as well as efficiently manage limited device resources such as memory and battery life. HTTP-based video streaming has emerged as a key technology for video delivery to end users but current solutions do not address the unique requirements faced by mobile clients.

The thesis of this dissertation is: *"Mechanisms that consider the characteristics of wireless and cellular traffic as well as exploit spatial and temporal adjacencies of client devices can significantly improve the Quality of Experience for mobile video streaming with minimal battery penalty."*

## *1.4 Contributions*

In this dissertation, we consider the challenges facing adaptive video streaming on mobile devices and propose new solutions for optimizing the streaming experience. The major contributions of this dissertation are summarized as follows:

- We first empirically investigate the performance of several existing Adaptive HTTP Streaming schemes on mobile devices using key quality metrics of efficiency, stability, buffer size, deadline miss ratio, rebuffering ratio, and battery depletion. Although many of the existing schemes have been evaluated under controlled network settings and on fixed devices, little is known about the performance of these schemes under the typical wide-area networking conditions faced by mobile devices. For each of the schemes, we implement the quality adaptation logic as described in the relevant literature. Using bandwidth data collected from real-world field trials over a period of four weeks, we investigate how the adaptation schemes respond to frequent fluctuations in network characteristics as experienced by mobile devices. We seek answers to questions such as "Are there any significant differences in performance results between fixed network and wireless network access?," "How does each scheme behave with cellular network access compared to Wi-Fi network access?," and "Are the observed streaming quality different across adaptation schemes and across network access types?" The results show that many of the schemes do not efficiently manage the often seemingly conflicting goals and the performance tradeoffs of bitrate adaptation nor do they achieve the level of performance desired for maximal user engagement and repeat viewership. Our measurement study provides key insights into the limitations as well as best-practices of the state-of-the-art with regards to AHS streaming on wireless and mobile clients.

- From the lessons learned from our measurement-based study, we propose a novel quality adaptation scheme for AHS Streaming on mobile devices, called **MASS**. MASS introduces new techniques in conjunction with the best practices of the existing schemes for improving the video streaming experience while minimizing the depletion in battery level. We implement and evaluate the performance of MASS on commodity Android smartphones, using the bandwidth trace obtained from the measurement study. We also report our findings of the effects of underlying components on the overall performance of MASS.

- To further improve the streaming experience, we introduce **SDASH**, a novel system for AHS video streaming on mobile clients which leverages the synergy between social network participation and video streaming. Given that friends in a social network have similar interests and share knowledge and experience with one another, the effectiveness of P2P sharing can be greatly enhanced by exploiting this knowledge. By caching video segments and servicing requests for the cached segments, social peers who are sharers of the video can help to improve the fidelity of the video quality during a streaming session and across multiple streaming sessions. SDASH extends existing AHS quality adaptation mechanisms by augmenting AHS streaming with P2P communication among the sharers of a given video.

- We develop a middleware infrastructure, called **Micrograph**, for establishing cooperative communities or *transient social networks (TSNs)* among mobile clients [136]. TSNs offer the opportunity for users to leverage both the physical and virtual world to opportunistically achieve a common purpose. The participants (via their mobile devices) can form a peer-to-peer (P2P) overlay network for exchanging information and activities of mutual interest in a decentralized fashion. TSNs may then be leveraged for the generation, dissemination and consumption of user-generated videos through P2P communication techniques augmenting the client-server communication paradigm. Micrograph implements routines for managing the membership of TSNs

based on the spatial and temporal locality of the devices. We use Micrograph infrastructure as a concrete implementation of a social P2P community of mobile peers for video sharing and in the implementation and evaluation of SDASH adaptation scheme.

## *1.5 Roadmap*

The remainder of this dissertation is organized as follows. Chapter 2 summarizes the state-of-the-art in AHS streaming and presents results of our investigation into the performance characteristics of several video quality adaptation schemes when deployed on mobile platforms. In the following two chapters, we present our solutions for improving the performance of adaptive mobile video streaming: Mobile Adaptation Scheme for Adaptive HTTP Streaming or MASS (Chapter 3) and Cooperative Extensions to Adaptive HTTP Streaming or SDASH (Chapter 4). Then, in Chapter 5, we introduce Micrograph, a generic infrastructure for constructing mobile P2P communities based on fundamental primitives of location, time, and interests. Finally, in Chapter 6, we summarize the dissertation and conclude with future work.

# CHAPTER II

# EVALUATING ADAPTIVE HTTP STREAMING ON MOBILE DEVICES

In this chapter, we characterize the structure of AHS Streaming (Section 2.1), introduce our measurement infrastructure (Section 2.2) and evaluate the performance of several AHS systems on mobile devices (Section 2.3). We conclude our discussion with a review of related work (Section 2.4) and a short summary (Section 2.5). The research undertaken in this chapter lays the groundwork for exploring and understanding the technical challenges, system requirements, and design principles essential to architecting efficient systems for AHS Streaming on mobile devices.

## *2.1 Adaptive HTTP Video Streaming*

We begin this section with a high-level overview of HTTP-based adaptive video streaming and the system model used in many state-of-the-art AHS implementations. Then, we present brief descriptions of five quality adaptation schemes that were studied in this chapter.

### 2.1.1 Overview

Adaptive HTTP Streaming (AHS) is a paradigm for video streaming on the web. Rather than rely on the dedicated video servers of yesteryears, AHS exploits off-the-shelf web servers. The protocol works as follows: A *video file* is divided into multiple small *segment* files that have the same play-out duration, typically a few seconds long. Each of the segments from the same video object are also encoded into different bitrates and quality levels. Along with an XML file called a *media presentation description file* (MPD) that describes the video segments and quality levels, the files are stored on web servers on the Internet. A video player can then request different segments at different bitrates depending on the state of the underlying network (Figure 2). This greatly simplifies the server design,

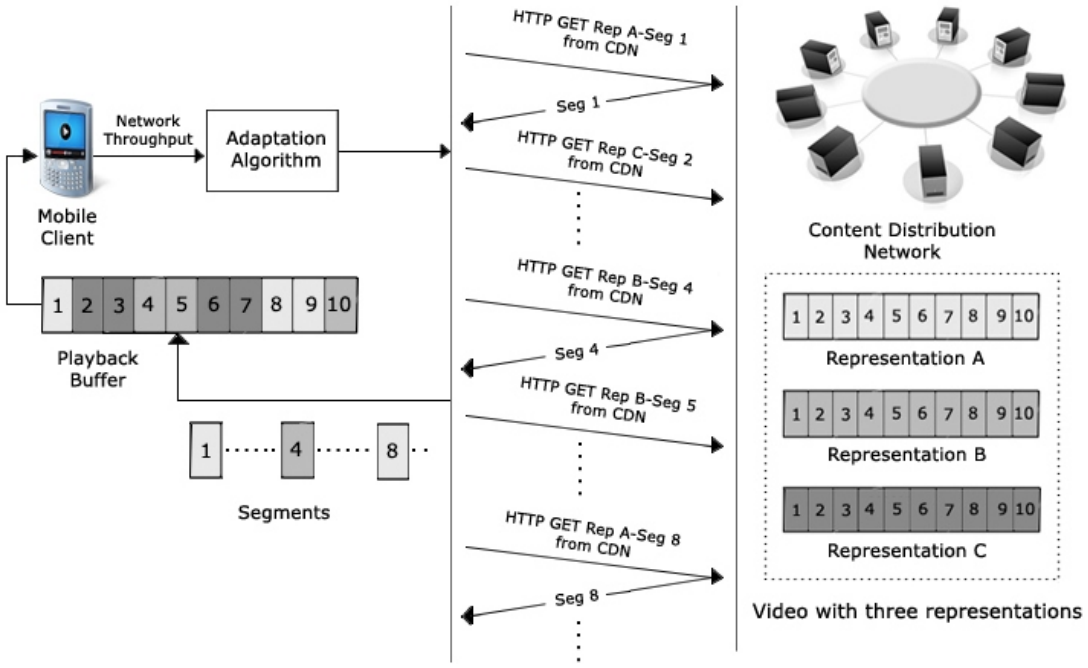



**Figure 4: Sample AHS Streaming Scenario. The mobile client downloads segments from the CDN which are placed in the playback buffer. An adaptation algorithm is invoked to determine the appropriate segment representation to fetch as each segment is requested.**

and allows the use of existing web servers and content distribution networks (CDN) systems without modification.

In addition to readily exploiting the widely deployed HTTP infrastructure for scalable video streaming, AHS can greatly simplify the firewall and NAT traversal problems encountered by other video streaming paradigms. On the other hand, the AHS paradigm relies exclusively on the client's ability to efficiently and reliably infer the available network throughput for high-quality and smooth video playback. Thus, the development of optimal client-driven approaches is essential for AHS meeting its optimization goals.

Several popular, commercial multimedia vendors have adopted AHS streaming. Netflix (which accounts for more than 20% of Internet video) currently leads the industry in this direction [23]. Similar efforts from rival media companies have seen the introduction of several AHS-based products. These products include Microsoft's *Smooth Streaming* [19], Apple's *HTTP Live Streaming* [13], Adobe's *HTTP Dynamic Streaming* [1], and Akamai's

*HD Adaptive Streaming* [2]. A standardization process sponsored by the Motion Pictures Experts Group (MPEG) resulted in a new ISO/IEC 23009-1:2012 standard called Dynamic Adaptive Streaming over HTTP, or DASH [119].

### 2.1.2 System Model

The client-driven mechanisms used in a functional AHS streaming can be divided into four key steps [75, 89]. These four steps encapsulate the series of operations used by the majority of existing commercial video players for AHS and other proposed AHS systems (Figure 5). We recap the four key steps in AHS streaming below:

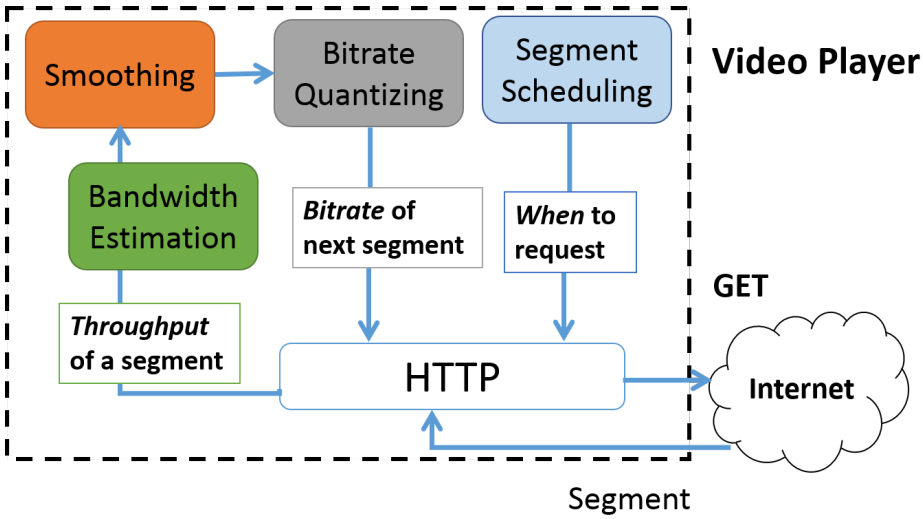



**Figure 5: Four step model for AHS Streaming.**

1. *Estimating step.* The AHS scheme begins by estimating the network bandwidth at a particular time $t$ that can legitimately be used for fetching the segment $i$. Typically, the network bandwidth is estimated by dividing the segment size for the most recent segment $i$ with the time taken to fetch the segment.

2. *Smoothing step.* In the smoothing step, the estimated bandwidth is then noise-filtered to yield a new version that helps to remove outlying values and accounts for oscillations in measured values. Many of the schemes use a time window between 10-20 seconds.

3. *Quantizing step.* Given the smoothed bandwidth values, the adaptation scheme then selects a suitable bitrate for the next segment request by mapping the smoothed bandwidth to the set of available bitrates.

4. *Scheduling step.* In the scheduling step, the adaptation scheme schedules when the next segment will be requested by considering the current conditions such as playback time and buffer size.

In addition, the adaptation scheme may introduces intermediate steps that are used to reduce bandwidth undersubscription, and bitrate oscillations.

### 2.1.3 Quality Adaptation Schemes

We now present brief descriptions of the adaptation schemes we studied in this paper. While there are many more adaptation schemes published in literature, we focused on those schemes where the adaptation logic was fully-performed at the media client (i.e., video player). Furthermore, our selection of the evaluated schemes was also motivated by the four step model and the evolution of proposals. In the descriptions below, we focus on the key features of the adaptation schemes. Full details of the schemes are available in their respective published works [91, 98, 75, 89, 95].

**LIU:** In this paper, we refer to the adaptation scheme proposed by Liu et al. [91] as the ***LIU*** scheme. The LIU scheme is one of the earliest proposed schemes for AHS adaptation by the research community. In LUI, the authors proposed using smoothed measurements of the segment fetch time to detect the changes in available HTTP throughput. By combining other metrics, such as media segment duration, the lowest bitrate, highest bitrate, and the ratio of differences between the available bitrates, the scheme deploys a step-wise technique for switching-up/switching-down between the available bitrate representations.

**OSMF:** We implemented the adaptation scheme used in Adobe's Open Source Media Framework (OSMF) [18] as listed in the work by Mok et al. [98]. The scheme invokes its adaptation routine before each segment is fetched from the server. Similar to the LIU scheme, it uses the download time of the last segment to infer the network throughput. It then implements a set of heuristics for determining whether to switch-up or to switch-down

to a different quality level. A key characteristics of the OSMF scheme is that it may skip multiple quality levels and may switch to lower or higher representation in a single epoch depending on the rate of download.

**MILLER:** We refer to the adaptation scheme proposed by MILLER et al. [95] as the ***MILLER*** scheme. The MILLER scheme uses the size and position of the playback buffer as a key factor for quantizing the estimated throughput to the reference bitrate and for determining how aggressive it should switch up or switch down given the changing conditions. To reduce the startup latency, the algorithm operates in two phases: a *fast start* phase and a *regular streaming* phase. In the fast start phase, each subsequent download quickly switches to the next higher representation as long as its bitrate is below a certain percentage of the throughput measured over the last $t$ seconds. The regular streaming phase is more conservative in its switch up operations; it uses the $B_{opt}$, the optimal buffer level, to determine when to switch up or switch down to a different representation.

**FESTIVE:** This is an AHS adaptation scheme proposed by Jiang et al. [75]. In FESTIVE, the authors analyzed the four steps of video adaptation and recommended techniques that can systematically guide the tradeoffs between stability, fairness, and efficiency and thus lead to a general framework for robust video adaptation. FESTIVE is geared towards optimizing performance in the presence of interactions across multiple adaptive streaming players that compete at bottleneck links. FESTIVE uses *harmonic mean* to smooth out measured throughput values, and uses a stateful technique for determining how to quantize the smoothed values to the bitrates. To reduce bitrate oscillation, FESTIVE implements routines for determining when recently measured bandwidth is computed with the rest of the smoothed values and used in bitrate quantization. FESTIVE also uses a *randomized scheduler* for determining when to request the next segment to reduce repeated sub-optimal views of the network conditions.

**PANDA:** Similar to FESTIVE, the PANDA AHS adaptation scheme [89] is geared towards optimal video adaptation in the presence of multiple players sharing a bottleneck link. The PANDA scheme models the adaptation routine after the TCP congestion control AIMD operations. It uses the *exponential weighted moving average* (EWMA) for smoothing

the estimated bandwidths and a *dead-zone quantizer* for quantizing the smoothed estimates to the reference bitrate. PANDA also includes routines to reduce the effect of bandwidth over-estimation on the performance of the adaptation algorithm by probing the network subscription with small increases in its segment request rate.

## *2.2 Measurement Infrastructure*

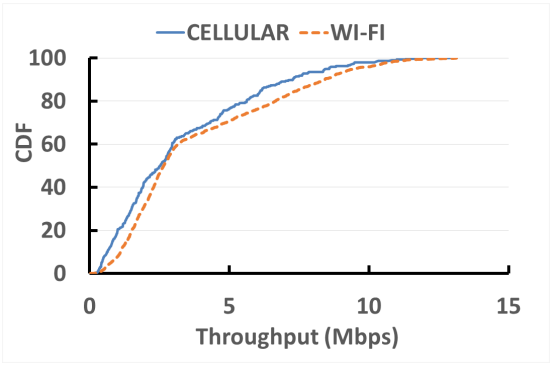
In this section, we present our measurement environment i.e., the test data, network trace, measurement testbed and quality metrics.

### 2.2.1 Test data and Video Formats

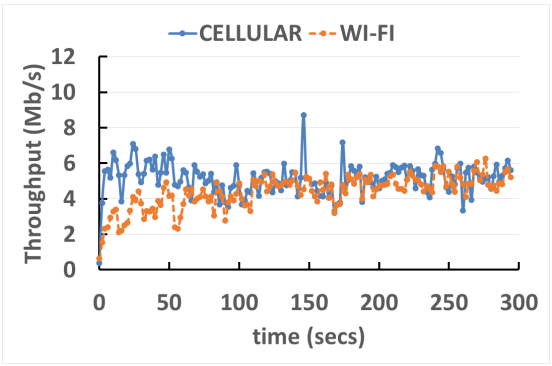
We use the dataset derived from a previous work into mobile AHS experimentation [84]. The dataset consists of segments from the Big Buck Bunny video [5]. The segments are encoded using x264 encoder at 14 different bitrates (Table 1). The segment duration is 2 seconds and the Group of Pictures size is 48 frames. We used two versions of the dataset, a version with playout duration of five minutes and a second version with playout duration of ten minutes.

**Table 1: Average and maximum sizes of the video segments used in the measurement dataset. The segment duration is 2 seconds and the frame rate is 48 fps.**

| Representation Bitrate | Average Segment Size | Maximum Segment Size |
|---|---|---|
| 100 Kbps | 24 KB | 46 KB |
| 200 Kbps | 47 KB | 88 KB |
| 350 Kbps | 79 KB | 160 KB |
| 500 Kbps | 110 KB | 238 KB |
| 700 Kbps | 149 KB | 382 KB |
| 900 Kbps | 189 KB | 549 KB |
| 1100 Kbps | 229 KB | 746 KB |
| 1300 Kbps | 268 KB | 910 KB |
| 1600 Kbps | 329 KB | 1138 KB |
| 1900 Kbps | 392 KB | 1445 KB |
| 2300 Kbps | 479 KB | 1747 KB |
| 2800 Kbps | 594 KB | 2213 KB |
| 3400 Kbps | 737 KB | 2716 KB |
| 4500 Kbps | 1025 KB | 3511 KB |




(a) CDF of network throughput.




(b) Mean of network throughput across playback duration of 300 seconds.

**Figure 6: CDF and mean of the network throughput in the collected mobile network trace. The network trace was collected over a period of one month using 8 smartphones, over 4G cellular connectivity and Wi-Fi. The throughput corresponds to streaming a random video from our dataset.**

### 2.2.2 Mobile Bandwidth Trace

To ensure the repeatability of the experiments and a fair comparison of the results across the different adaptation schemes as well as to capture typical scenarios faced by users, there is a need to simulate network conditions that match observed bandwidths in realistic wireless and cellular networks.

We gathered network traces by providing volunteer users with an experimental mobile application installed on eight different Android smartphones. The volunteer users carry the phones with them as they move about their daily routine, i.e., the traces capture network conditions typical of stationary, pedestrian, and vehicular speeds. The smartphones include four Google Nexus 4 smartphones equipped with Wi-Fi 802.11 a/b/g/n and 4G network radios and four Samsung Galaxy S III smartphones equipped with Wi-Fi 802.11 a/b/g/n and 4G network radios. We utilize T-Mobile 4G LTE network for cellular connectivity and 802.11g for Wi-Fi connectivity. At regular intervals, the Android application randomly selects a video from our test dataset and streams the selected video to completion. It then uploads network statistics recorded during the video playback to our test server. We collected the estimated uplink and download bandwidth and latency across the streaming

sessions over a month period. We also recorded the GPS coordinates and time for each session. We used a fixed chunk size of 900 Kbps with a segment duration of 2 seconds, i.e., during a single session, the uplink and downlink throughput and latency is sampled every two seconds and uploaded to our test server.

Figure 6 shows the distribution of measured throughput obtained from our network trace collection setup. While the median cellular 4G LTE throughput (6.6 Mbps) is higher than the Wi-Fi throughput (5.6 Mbps), the cellular throughput experiences a lot more variability than Wi-Fi; even with the same user at the same location across different viewing sessions. Also, the median RTT for cellular in the bandwidth trace is 88ms while the median RTT for Wi-Fi is 69ms (Figure 7). The RTT and throughput values are similar to reported values from other mobile network measurement studies [71, 118].

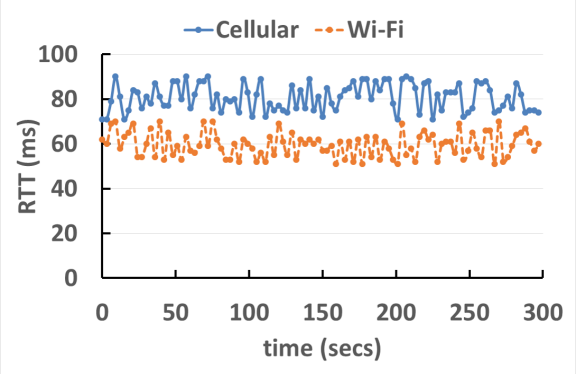



**Figure 7: Mean of RTT across the playback duration of 300 seconds in the collected mobile network trace.**

### 2.2.3 Measurement Testbed

We implemented an AHS library on Android that reads, parses, and progressively downloads an AHS file. Our implementation is a port of the open source *libDASH* [15] onto the Android operating system. The library includes all the functionality of libdash as well additional implementation for collecting performance metrics. We then implemented the adaptation logic for each scheme as described in their relevant publications.

Our implementation also includes a logging facility that records each segment request

and additional details about the state of the playback at a granularity of 1-second. The additional details include the type of network, instantaneous throughput, size of the playback buffer, segment deadline misses, playback rebufferings, and the segment representation. At the end of each session, the logging facility uploads the results to our measurement server. We verify that the software could download and stream all of the test data and all of the quality levels without errors.

On the server side, we use Microsoft's IIS web server for serving the video segments via HTTP GET. The server is configured to support the *.mpd* and *.m4s* file formats used by the dataset for representing the AHS mime type. The web server runs on a dedicated Windows PC with 2 GB RAM and an Athlon 64 3200+ CPU.

When running the experiments, we use the bandwidth controller (Figure 8) to replay the network conditions recorded in our traces. We start by selecting from one of the recorded traces at random, and loading the selected trace on the bandwidth controller, which then throttles the bandwidth and delays network packets based on the conditions observed in the previously collected trace. For each selected network trace, we sequentially stream a video from the dataset with all of the five different adaptation schemes. This ensures that all of the adaptation schemes were ran using the same networking conditions. We repeated the experiments 100 times.

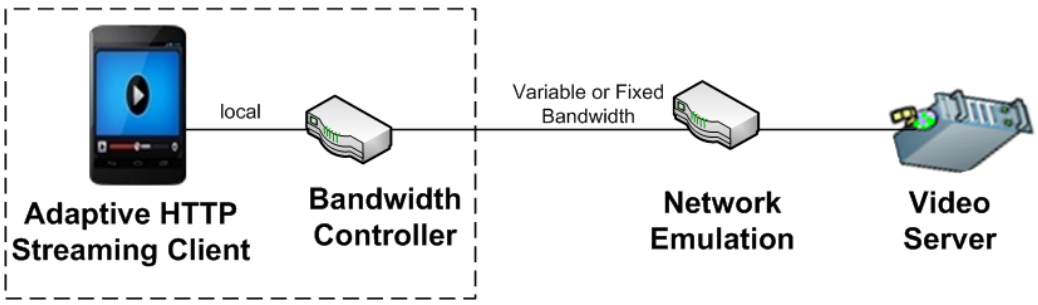



**Figure 8: The measurement test bed. The Video Server hosts the video segment files. The Bandwidth Controller is used to replay mobile network traces. *Local* indicates the bandwidth between the client and bandwidth controller which is significantly larger than the bandwidth needs of the segment requests from the client.**

### 2.2.4 Quality Metrics

We adopt a number of industry-standard metrics [75, 92, 89] for our evaluation study. Given a video file with maximum bitrate $b_{max}$, and a media client playing a segment of bitrate $b_{i,t}$ at time t over the video length, we define the following metrics.

1. *Inefficiency*: Extending the definitions given in [75, 89], we define the inefficiency of bitrate adaptation as follows. Let the available bandwidth be $W_{i,t}$ at time $t$ just as the segment $b_{i,t}$ is requested by the client. The inefficiency of the adaptation scheme is defined as $\sum_t \frac{|b_{i,t} - \min(b_{\max}, W_{i,t})|}{W_{i,t}}$, giving a value between the 0 and 1. The lower the value, the more efficiently the scheme is utilizing the network throughput to deliver the best quality of experience for the client.

2. *Instability*: Previous studies have shown that the number and frequency of bitrate switches during playback has significant impact on user experience [44]. The instability metric $\frac{\sum_{d=0}^{k-1} |b_{t-d} - b_{t-d-1}| \cdot w(d)}{\sum_{d=1}^{k} b_{t-d} \cdot w(d)}$ captures the smoothness of bitrate adaptation by each scheme by computing a weighted sum of all the switch steps observed within a recent time window $k$ divided by the sum of the bitrates within the last k period. The weight function, $w(d) = k - d$, adds more penalty to the more recent switches. In our calculations, we use $k = 20$ seconds. The lower the value (i.e., close to zero), the smoother the video quality adaptation to changing network conditions.

3. *Deadline miss ratio*: The deadline miss ratio is the number of requested segments which missed their download deadline divided by the total number of segments. The download deadline is dynamically calculated based on the remaining period of playback and the segment duration. The segment download is terminated when the download deadline elapses. The adaptation scheme will then progress to requesting the next segment (typically at a lower bitrate).

4. *Buffer level*: The buffer level is the number of segments in the video buffer at a given time $t$ measured in seconds of playback. The buffer level is an indicator of how well the adaptation scheme responds to changes in the available bandwidth.

5. *Buffer undershoot*: Let $B_{target}$ be the target buffer level (set as 30 seconds in our implementation). The buffer undershoot $\frac{\max(0,B_{target}-B_t)}{B_{target}}$ at time $t$ captures the ability of the scheme to meet its target buffer level [89]. Lower values of the buffer undershoot indicate smaller possibility of buffer underruns due to sudden bandwidth drops.

6. *Rebuffering ratio*: This is computed as the number of times the segments have to be rebuffered because of playback deadline misses divided by the total playback duration. We exclude the initial buffering at video start from the calculation. The rebuffering ratio is an indicator of how smoothly the adaptation scheme proceeds under varying networking conditions.

7. *Startup delay*: According to studies on the behavior of commercial players [26, 111], many of the existing players and proposed adaptation schemes divide the playback duration into two states: a buffering state and a steady state. The buffering state is the period from when the first segment request is made to when the playback buffer has reached its target size while the steady state is the period from the end of the buffering state to the completion of playback. Furthermore, many players begin playback as soon as a fraction of the target buffer size has been downloaded (about 10 seconds in the SmoothStreaming player [26]). In our evaluation, we define the startup latency as the time between the first segment request and when the buffer level is higher than a minimum buffer threshold $B_{min}$. We use a minimum buffer level of 10 seconds.

8. *Battery depletion*: The battery depletion is the amount of the device battery consumed during playback given as a percentage of the maximum battery level when fully charged. A value close to zero implies the adaptation scheme uses minimal device energy for streaming the video.

Except for buffer level, for all the other metrics, the lower the value the better the adaptation scheme.

## *2.3 Experimental Results*

In this section, we present detailed measurement results for each of the adaptation schemes listed in Section 2.1. Using network traces collected as described in Section 2.2.2, we evaluated the performance of the five schemes. In the experiments, we used the default configuration parameters specified by the authors in the original evaluations and as described in Section 2.1.3.

### 2.3.1 Comparison Strategies

For each metric, we compare the performance of the five video adaptation schemes according to behavior when utilizing the mobile bandwidth trace and to the behavior when utilizing a fixed bottleneck link of 8 Mbps. In the fixed scenario, the bandwidth controller sets the bandwidth to the server to a fixed value of 8 Mbps throughout the duration of the experiments. For each quality metric, we compute the mean of the results across all the experimental measurements. We then compare each of the five bitrate adaptation schemes according to the behavior under cellular network access, the behavior under Wi-Fi access and the behavior under the fixed bottleneck access.

### 2.3.2 Inefficiency

Figure 9 shows the breakdown of the inefficiency metric for the five adaptation schemes. The inefficiency is calculated as defined in Section 2.2.4. For each scheme, we recorded the requested segment bitrate, and the available throughput in a given playback session and calculated the inefficiency for a single playback session. We then computed a simple average of the calculated inefficiency across all the streaming sessions. The lower values illustrate a lower inefficiency and/or a better utilization of the available throughput by the adaptation scheme.

From the results in Figure 9, we make a number of observations. First, the inefficiency of the schemes under mobile networking scenarios is worse than the inefficiency when utilizing a fixed bottleneck of 8 Mbps. The increase is more pronounced for the LIU and OSMF schemes, increasing by as much as 134% on Wi-Fi and 143% on cellular. Second, the

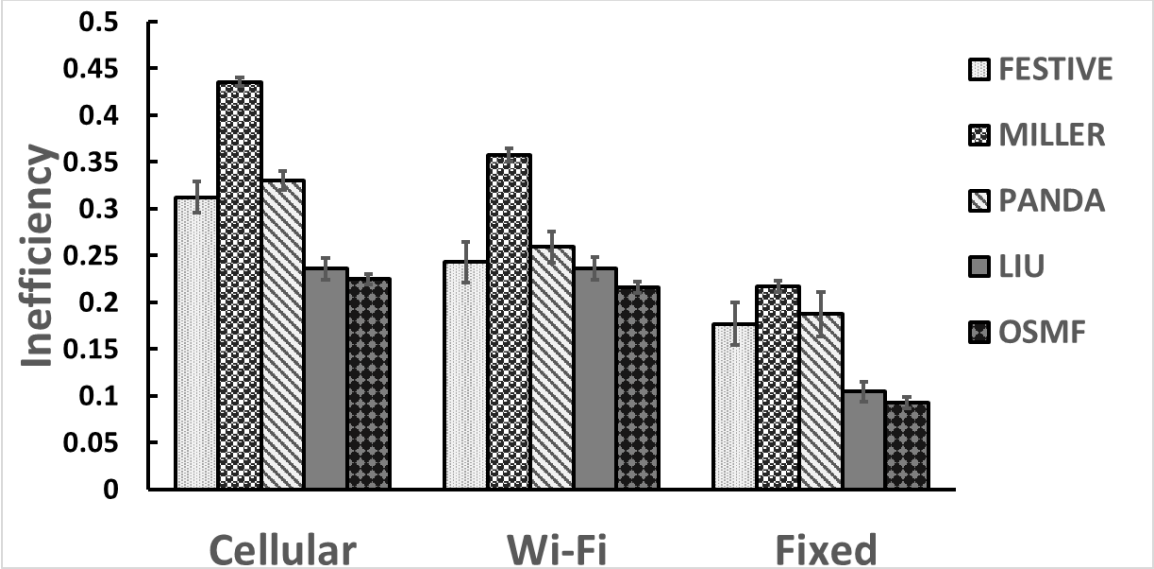



**Figure 9: Inefficiency Results: Fixed bottleneck link versus variable network conditions using the collected network traces.**

inefficiency of all the adaptation schemes is worse when streaming with cellular network access versus when streaming with Wi-Fi access. Since lower inefficiency translates to higher average bitrates, this implies that the schemes deliver consistently lower picture quality under mobile networking conditions (especially on cellular connectivty) than under controlled settings. Third, the OSMF scheme outperforms the other schemes on both cellular and Wi-Fi network access. We speculate that the reason for the difference in deviations between the adaptation schemes is because of the estimating strategy used by the different schemes. The two schemes that use the rate of download (LIU and OSMF) for quantizing and are more aggressive during the switch-up process, give better efficiency than the other schemes.

### 2.3.3 Instability

In Figure 10, we present results for the instability metric. Similar to the inefficiency metric, we computed the instability for each playback session and then computed a simple average across all sessions to derive the instability metric. The lower the value, the lower the instability for the given adaptation scheme and the better the smoothness of bitrate

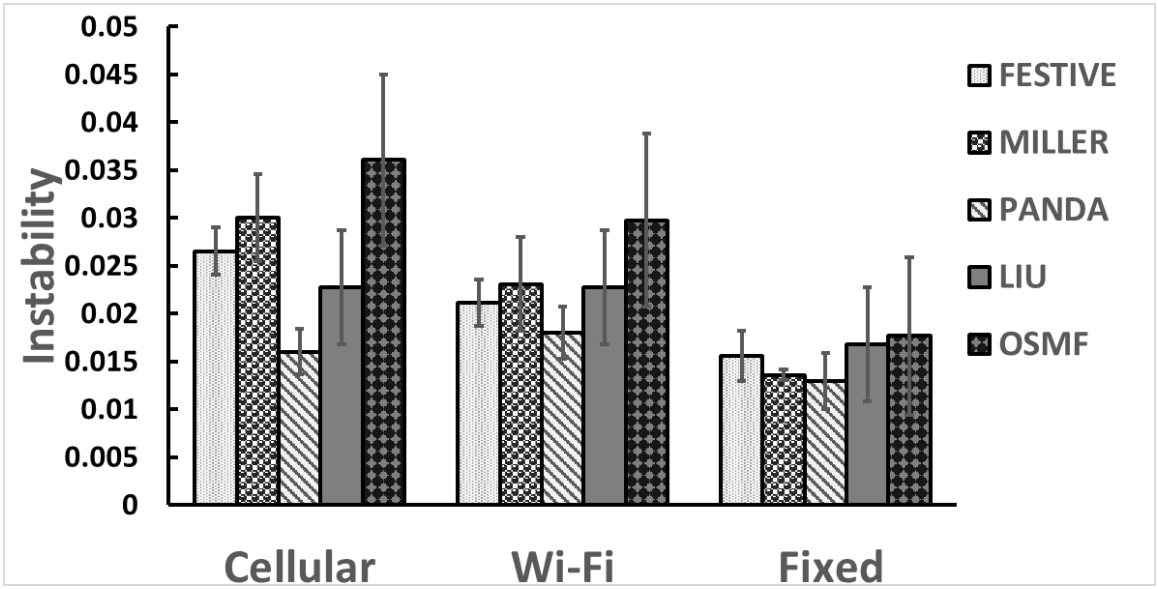



**Figure 10: Instability Results: Fixed bottleneck link versus variable network conditions using the collected network traces.**

switching.

Similar to Figure 9, the results in Figure 10 show that the instability under the fixed scenario is better than the instability under the mobile bandwidth conditions for all the schemes. For example, the instability of PANDA increases by 236% when streaming with cellular network access and by 389% when streaming with Wi-Fi access. The buffer-aware schemes (i.e., the schemes that utilize the buffer level for quantizing and scheduling segments such as PANDA, FESTIVE, and MILLER) yield better stability than the other schemes.

### 2.3.4 Startup Latency

Figure 11 shows the startup latency for the five adaptation schemes. As explained in Section 2.2.4, the startup latency is the time taken to download the first 10 seconds length of segments upon initialization of playback. For each scheme, we calculated a simple average of the startup latency across all the playback sessions. The results show that the OSMF scheme incurs very high latency (about 10 seconds on Cellular and Wi-Fi), followed by the LIU schemes (about 8 seconds on Cellular and Wi-Fi). The FESTIVE, MILLER, and

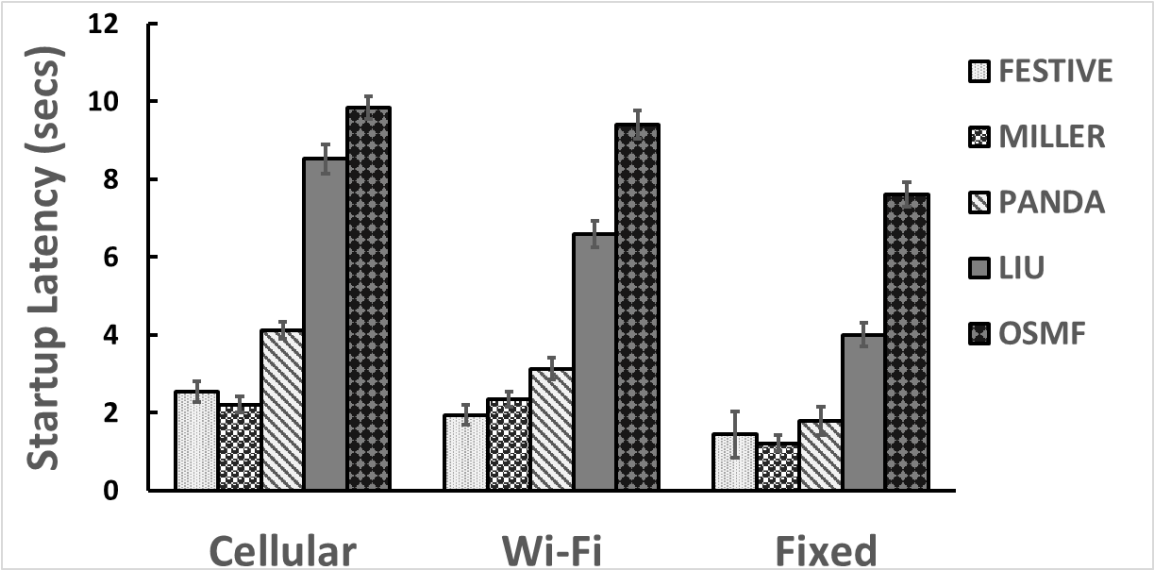



**Figure 11: Startup Latency Results: Fixed bottleneck link versus variable network conditions using the collected network traces.**

PANDA schemes incur similar latency of about 3 seconds. Given that the segment duration is 2 seconds, the results show that the OSMF scheme exhausts the download duration at startup while the other schemes utilizes only a fraction of the download deadline for the first 5 segments. Comparing the result in Figure 11 to the results in Figure 9, we observe that the startup latency reflects the aggressiveness with which the bitrate is switched-up by the scheme. The OSMF scheme quickly switches to higher quality within a single epoch, thereby downloading the highest quality segments and incurring the longest startup latency than the other schemes.

### 2.3.5 Deadline Miss Ratio

In Figure 15, we show a breakdown of the deadline miss ratio. As explained in Section 2.2, the deadline miss ratio is the fraction of segment requests that miss their segment download deadline given the current buffer level and playback position. The deadline miss ratio serves as an indicator of how well the adaptation scheme predicts the future network conditions and how well it responds to sudden drops in network bandwidth. The figure shows that the

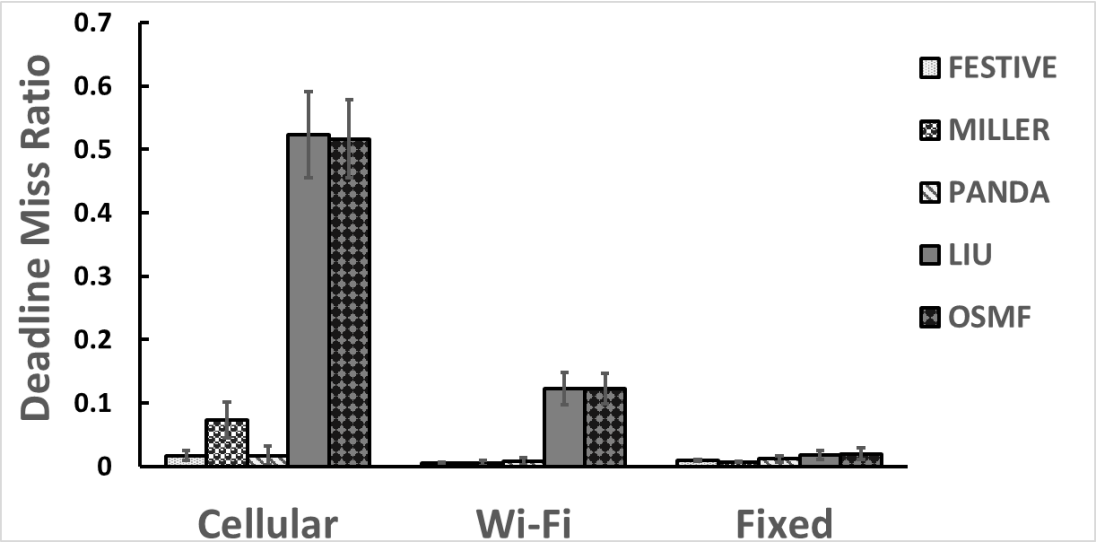



**Figure 12: Deadline Miss Ratio Results: Fixed bottleneck link versus variable network conditions using the collected network traces.**

LIU and OSMF schemes experience higher deadline misses than the other schemes when streaming under the mobile networking conditions, with the fraction of segment requests that miss their download deadline as high as 50% when streaming with cellular connectivity. This implies that the LIU and OSMF schemes are the least robust to the transient fluctuations in available bandwidth.

### 2.3.6 Buffer Size

The buffer undershoot experienced by each of the five schemes is shown in Figure 13. The figure shows significant differences between the performance under mobile networking conditions and the performance when utilizing a fixed bottleneck of 8 Mbps for the PANDA, OSMF, and LIU schemes. On the other hand, FESTIVE and MILLER seem to be unaffected by the mobile network conditions. This result can be attributed to the fact that both MILLER and FESTIVE delay raising the quality level until a more accurate picture of the network condition is known. MILLER uses the buffer depletion level while FESTIVE uses a stateful technique to infer the network conditions.

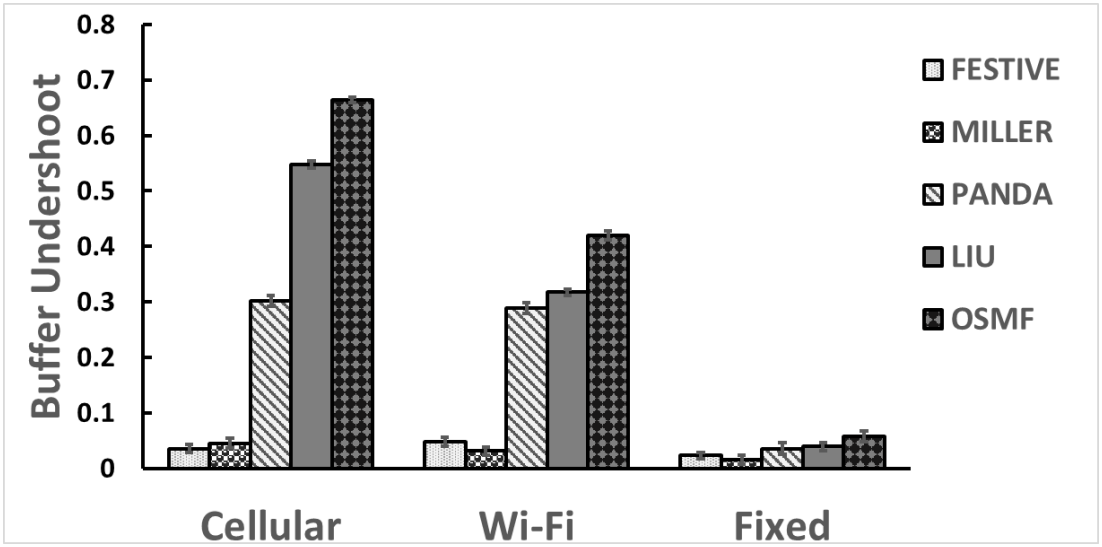



**Figure 13: Buffer Undershoot Results: Fixed bottleneck link versus variable network conditions using the collected network traces.**

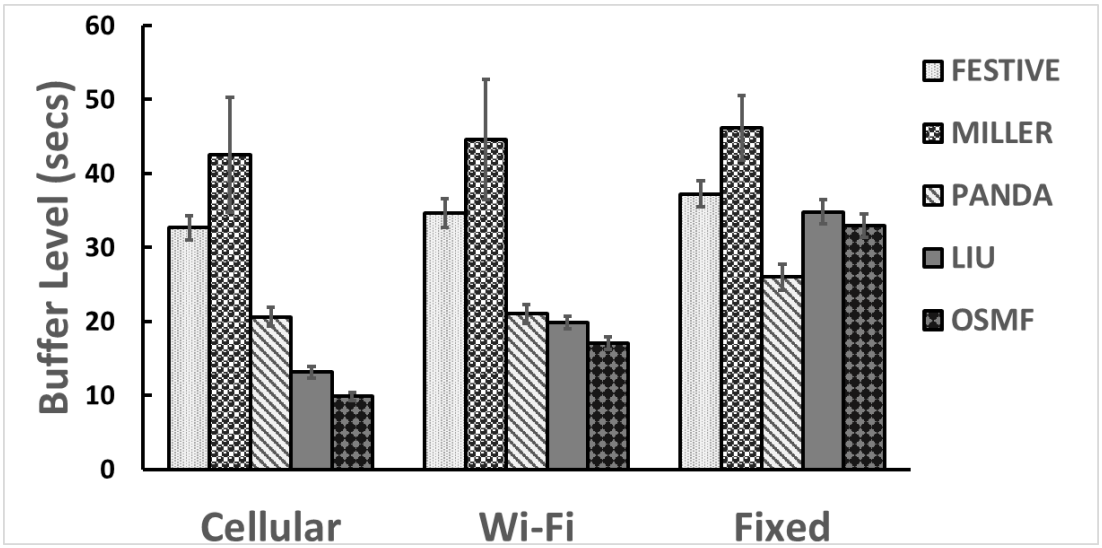



**Figure 14: Buffer Level Results: Fixed bottleneck link versus variable network conditions using the collected network traces.**

Figure 14 shows the buffer level for the various schemes under fixed and mobile conditions. For the same reason as in the buffer undershoot results, PANDA, OSMF, and LIU schemes show poor performance under mobile networking conditions. The mean buffer level for these schemes are well below the target buffer level of 30 seconds. The buffer levels in

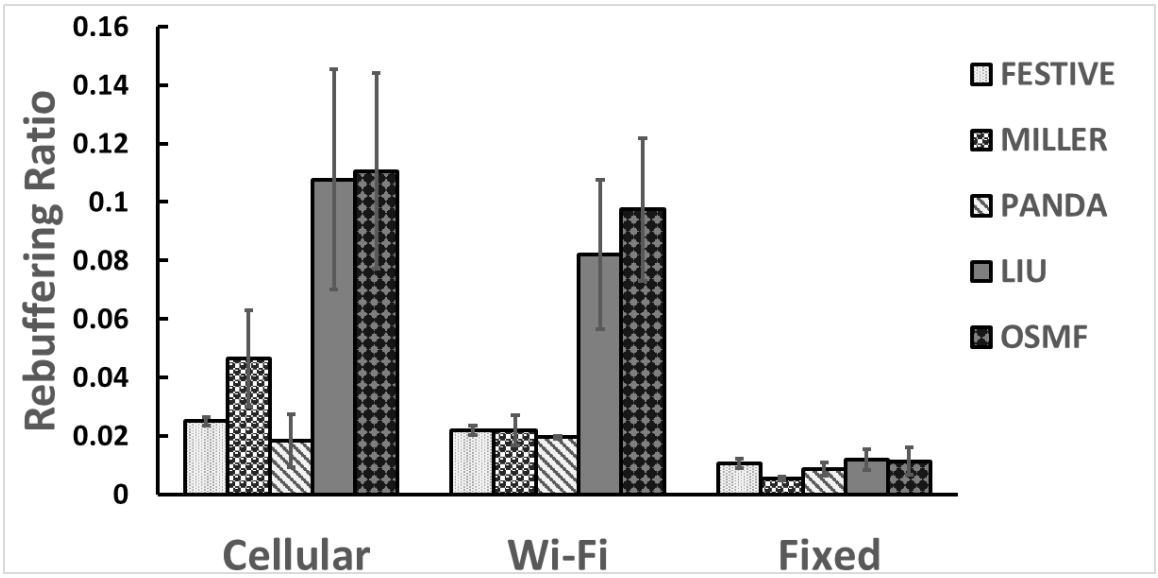



**Figure 15: Rebuffering Ratio Results: Fixed bottleneck link versus variable network conditions using the collected network traces.**

seconds for PANDA, LIU, and OSMF are 20.6, 13.1, and 9.9 (cellular), and 21.2, 19.8, and 17.1 (Wi-Fi), respectively.

In Figure 15, we show a breakdown of the rebuffering ratio. Similar to the deadline miss ratio, the rebuffering ratio serves as an indicator of how well the adaptation scheme predicts the future network conditions and how well it responds to sudden drops in network bandwidth. The figure shows that the PANDA scheme has the best rebuffering ratio when streaming under the mobile networking conditions. On the other hand, the OSMF scheme experiences a higher-rate of rebuffering than the other schemes. We speculate that the OSMF and the LIU schemes suffer from higher rebuffering than the other schemes under mobile networking conditions because they use only the rate of download as a quantizing factor which is very sensitive to the transient fluctuations in throughput estimation.

### 2.3.7 Battery Depletion

We measure the battery depletion for each adaptation scheme by recording the remaining percentage of battery as reported by the Android system every 1 second. We filter out the results for when the smartphone was concurrently charged and results for when the

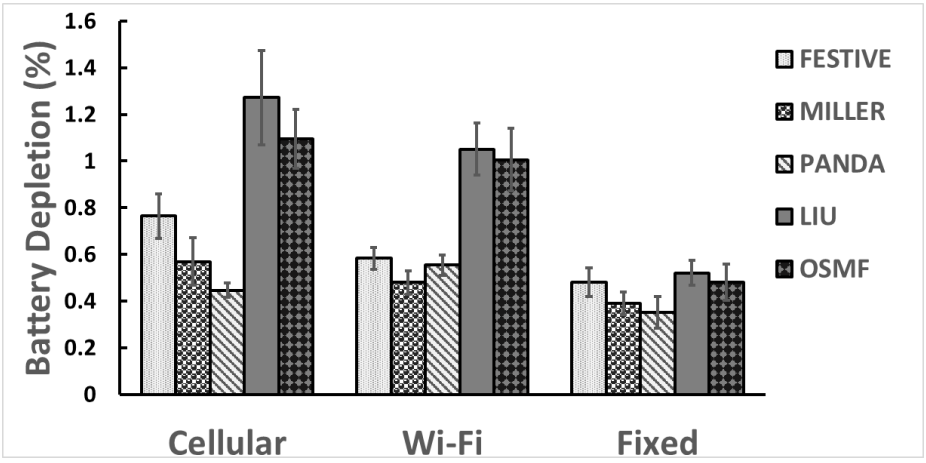



**Figure 16: Battery Depletion Results: Fixed bottleneck link versus variable network conditions using the collected network traces.**

remaining percent was less than 50%. We then computed a simple average of the depletion over the lifetime of our experimentation.

Figure 16 shows the results for the battery depletion for streaming a 5-minute video from the dataset. For each scheme, the battery depletion on Wi-Fi is generally less than that for cellular. Comparing the battery depletion to the average requested bitrate, we observe that the schemes with higher requested bitrates also use more of the device battery. The FESTIVE and MILLER schemes provided the best performance of all the evaluated schemes.

### 2.3.8 Summary of main results

By using realistic mobile networking conditions, we are able to tease out the innate differences in the adaptation schemes. Overall, the performance of all the schemes leave something to be desired under mobile networking conditions than when tested under controlled settings (such as a fixed bottleneck link). We summarize the main results from the study below.

- First, the inefficiency metric is worse by as much as 143% than the inefficiency under controlled settings. Since lower inefficiency translates to higher average bitrates, this

implies that all the schemes deliver consistently lower picture quality under mobile networking conditions than under controlled settings.

- There are noticeable differences between the performance of the AHS adaptation schemes on cellular network and Wi-Fi access networks. As a gross generalization, most of the schemes perform better on Wi-Fi connectivity than on cellular connectivity.

- There is no single adaptation scheme that performs outstandingly well on all the quality metrics. However, the FESTIVE, MILLER, and PANDA adaptation schemes outperform the LIU and OSMF schemes on the startup latency, rebuffering ratio, and buffer undershoot metrics. The LIU and OSMF schemes outperform the remaining schemes on the inefficiency metric. The FESTIVE and PANDA schemes result in the fewest number of playback rebufferings.

- The LIU and OSMF schemes are the least robust to short-time fluctuations in network bandwidth. This is due to their aggressive nature in making frequent requests for segments of higher bitrates leading to higher rate of rebufferings (8% - 11%).

- The MILLER scheme (which heuristically combines the value and rate of increase in the buffer level with the estimated network throughput in the quantizing step) experiences the least startup latency and the least buffer undershoot. However, it experiences higher inefficiency and higher rebuffering ratio than FESTIVE and PANDA schemes i.e., it is not as robust as the FESTIVE and PANDA schemes in the face of sudden drop in network bandwidth.

### 2.3.9 Lessons Learned

The study also allows us to reflect on attributes of the individual schemes that lend themselves to good design principles for mobile networking conditions.

- The schemes that are more aggressive in their switching decisions by switching between multiple quality levels within a single epoch (OSMF and LIU) achieve better efficiency

than the schemes that are conservative in their switching decisions. However, this leads to higher increase in instability, startup latency, and rate of rebufferings.

- The schemes that utilize buffer-aware strategies in quantizing the reference bitrate and/or making scheduling decisions achieve lower startup latency, lower buffer undershoot, and higher mean buffer level than the other schemes.

- The schemes that smooth the estimated bandwidth using a harmonic mean and exponential weighted moving average (FESTIVE and PANDA) experience lower difference in the mean buffer level and rebuffering ratio in the presence of network fluctuations.

## *2.4 Related Work*

The related work consists of prior and ongoing research into quality adaptation schemes for HTTP streaming, measurement studies of Internet and mobile video, and the design and characterization of quality metrics for AHS streaming.

**Measurement of Video Streaming**. Various qualitative and quantitative studies [26, 75, 98, 47, 25] have examined the performance of AHS streaming in commercial systems such as Netflix [17], Microsoft's Smooth Streaming [19], Apple's HTTP Live Streaming [13], and Adobe's Dynamic Streaming [2]. Through experiments in controlled settings, these studies characterize the behavior of AHS streaming when one or more video players share a bottleneck link to the video server. Our study focuses on the performance of bitrate adaptation for smartphone clients faced with mobility-related constraints, irregular network connectivity and resource constraints.

Another category of research studies looks at networking characteristics of video traffic by measuring client interactions with the CDN servers and CDN selection strategies [92, 23, 112]. By analyzing collected HTTP video dataset, these studies then infer key insights into the effects of competing TCP flows and CDN selection on the Quality of Experience (QoE) [97], rebuffering rates, and user engagement. A number of studies have also examined the characteristics of video traffic in cellular networks [53, 65, 93].

**AHS Adaptation Schemes**. Many research studies have proposed quality adaptation schemes for AHS streaming [75, 89, 91, 95, 98]. These systems analyze metrics such as

fairness, efficiency, stability, and average bitrate. The majority of the proposed schemes follow a four-step model for AHS streaming described in Section 2.2. Other systems have also proposed control-theoretic approaches and Markov-Decision Process techniques for improving the overall performance of the bitrate adaptation [122, 140]. Our study has been a careful comparative study of a select number of such adaptation schemes.

Another set of studies has examined video streaming in mobile environments. The design and implementation of AHS streaming for mobile clients is an active research area [114, 133, 60, 40, 32, 123, 69]. Muller et al. [101] studied the performance of commercial AHS systems in vehicular networking environments. Seo et al. [117] proposed techniques for optimizing the uploading of live video segments to AHS servers from mobile end devices. Ransburg et al., [111] evaluated AHS streaming using a local testbed. They measured the impact of packet delay and packet loss on video streaming using either Apple's HTTP Live Streaming or traditional Real Time Streaming protocols. While these systems also address some of the challenges with AHS streaming on mobile devices, they do not provide quantitative comparisons with existing quality adaptation schemes. More importantly, in our work, we conducted detailed evaluations of select adaptation schemes using industry-standard quality metrics.

**Quality Metrics**. There are also research studies on measurement metrics for AHS streaming. Given that a key objective of AHS streaming is to optimize user-perceived Quality of Experience, a number of previous works have proposed metrics and adaptation schemes for evaluating AHS streaming systems. Mok et al. [97] studied the degree of correlation between the physical network quality of service and the user-perceived application quality. In QDASH [98], the authors conducted a user study to evaluate the QoE of a proposed buffer-aware rate selection scheme. Similarly, a related study by Balachandran et al. [30] enumerated multiple metrics that can jointly determine how the quality of AHS streaming correlates with user-engagement. In this paper, we focus on performance evaluations of individual metrics, with a view of achieving a more QoE-aware system.

## *2.5 Summary*

In this chapter, we have presented foundational details about the state-of-the-art in adaptive HTTP video streaming including an overview of the system model used in multiple existing implementations and brief descriptions of several state-of-the-art adaptation schemes. Then we explain the architecture and the implementation details of our measurement infrastructure which we used in the experiments evaluating the performance of AHS on mobile devices. Finally, we present detailed experimental results quantifying the performance of the state-of-the-art AHS schemes using our measurement infrastructure. In our experimental results, we find significant differences between the performance of the evaluated schemes in controlled experimental settings and their performance in mobile settings. We also find major differences in the behavior of the schemes under Wi-Fi and cellular network access types, with most of the schemes performing worse when the network access type is cellular. Given these observations, we provide an understanding of the possible causes of these inefficiencies. The experimental results show that existing approaches for bitrate adaptation do not effectively provide consistent Quality of Experience under mobile networking conditions nor do they achieve the level of performance desired for maximal user engagement and repeat viewership. In the next chapter, we will present a novel scheme for bitrate adaptation that shows improved performance under mobile networking conditions.

# CHAPTER III

# MOBILE ADAPTATION SCHEME FOR HTTP VIDEO STREAMING

In this chapter, we introduce our proposed Mobile Adaptation Scheme for AHS (MASS). Armed with the key observations from our experimental evaluations, **MASS** addresses the limitations of the existing adaptation schemes. We begin by describing the system context and design goals. We then provide key details of the algorithms and routines used by the MASS scheme. Finally, we present performance evaluation of MASS using the same experimental setup as the schemes studied in Chapter 2.

## *3.1 Context and Design Goals*

We are witnessing a paradigm shift to mobile devices as the primary platform for consuming videos. The accessibility of mobile devices as well as the penetration of high-speed 3G and 4G networks offer the potential for high-quality video experience. However, as observed in the previous chapters, the task of architecting a system for adaptive mobile video streaming poses significant challenges unaddressed by existing schemes. It also requires tradeoffs between multiple, conflicting optimization goals represented by key quality metrics described in Chapter 2 (Section 2.2.4). From these performance metrics and analysis of existing systems, we identify the following design goals for a mobile video quality adaptation scheme:

1. *Avoid playback rebufferings due to buffer underruns*: Because of the severe impact of buffer underuns on user experience, the adaptation scheme must be very robust in the face of sudden drops in network bandwidth. In our proposed scheme, we incorporate buffer-aware strategies with stateful bandwidth update to improve the responsiveness of the adaptation scheme to bandwidth changes.

2. *Maximize the average bitrate quality and minimize the inefficiency and instability metrics*: Achieving efficient utilization of the available network bandwidth will yield high

picture quality and improved streaming experience. Similarly, achieving a low instability metric translates to a smooth streaming experience by reducing the oscillations in bitrate levels. In our proposed scheme, we extend techniques from the FESTIVE and the PANDA adaptation schemes to maintain high-efficiency and low-instability.

3. *Provide uniform performance under Wi-Fi and cellular connectivity*: Because of the divergence in the behavior of existing schemes on Wi-Fi and cellular network access, users may experience non-uniform streaming quality depending on the type of connectivity at the time of streaming. By reducing the variability in key performance metrics (e.g., instability, inefficiency, and rebuffering ratio), the streaming experience can be greatly enhanced. In our proposed scheme, we introduce a number of techniques for handling the high variability of network throughput. This includes heuristics for smoothing measured bandwidth values to reduce the jitter on cellular connectivity and using a two-phase model to improve the bitrate quality at the start of streaming.

4. *Minimize battery consumption*: Because of limited battery life, it is essential for the adaptation scheme to minimize its impact on battery usage.

The first two optimization goals have been identified by previous work on AHS streaming. The last two optimization goals in the above list are specific to MASS and the mobile AHS streaming scenario. We do not claim that this list is exhaustive. Other optimization goals such as minimal playout startup latency, and ensuring fairness among competing video players sharing a bottleneck link have also been shown to impact the streaming experience.

## *3.2 MASS: Measurement-Driven Design*

As we have shown in the previous chapter, existing adaptation schemes do not satisfy the goals of efficiency, stability, low frequency of rebufferings, and high average quality. They also do not offer uniform performance under multiple forms of mobile networking access. In this chapter, we introduce a novel scheme that satisfies these properties. Based on insights from experimental results in Chapter 2 (Section 2.3), our design approach examines the

performance of existing schemes and considers the features of each element of the four-step model (i.e., bandwidth estimation, bandwidth smoothing, bitrate quantization, and chunk scheduling) that translate to improved performance. Through careful measurements, we then evaluated the impact of these features and their configuration parameters on the overall streaming quality.

From the results in Chapter 2, we make a number of observations. First, the schemes which perform multiple switches within a single interval (LIU & OSMF) achieve higher efficiency than schemes which perform at most one switch within a single interval. On the other hand, the schemes with multiple switches per interval are more unstable than schemes with at most one switch per interval. Second, the FESTIVE and PANDA schemes outperform the other schemes on key metrics of startup latency, buffer undershoot, buffer level, and rebuffering ratio. Furthermore, the FESTIVE and PANDA schemes show the least variability on both Cellular and Wi-Fi network access on a number of key metrics. Quantitative results in FESTIVE [75] as well in Chapter 2 (Section 2.3) showed that using harmonic mean reduces the impact of sudden drops or increases in bandwidth estimates compared to other smoothing techniques. Third, the MILLER scheme has the lowest frequency of buffer undershoot. This implies buffer-aware strategies can be beneficial for reducing buffer underruns. We base our design of MASS on these key observations.

Figure 17 shows an overview of techniques used by the MASS scheme. MASS combines techniques from existing schemes as well as observations about the networking conditions (type of network connectivity, and available bandwidth), position and state of video playback and the recent history of bitrate switches to optimize the streaming quality. In the next two sections (Section 3.3 and Section 3.4), we explain the rationale behind our selection of these techniques, describe key steps in the MASS scheme and empirically validate the impact of each technique on overall streaming quality.

### *3.3 Heuristic Algorithm*

We now proceed with describing the details of the MASS adaptation scheme. First, we summarize the notation used in our explanation, followed by a brief overview of the system

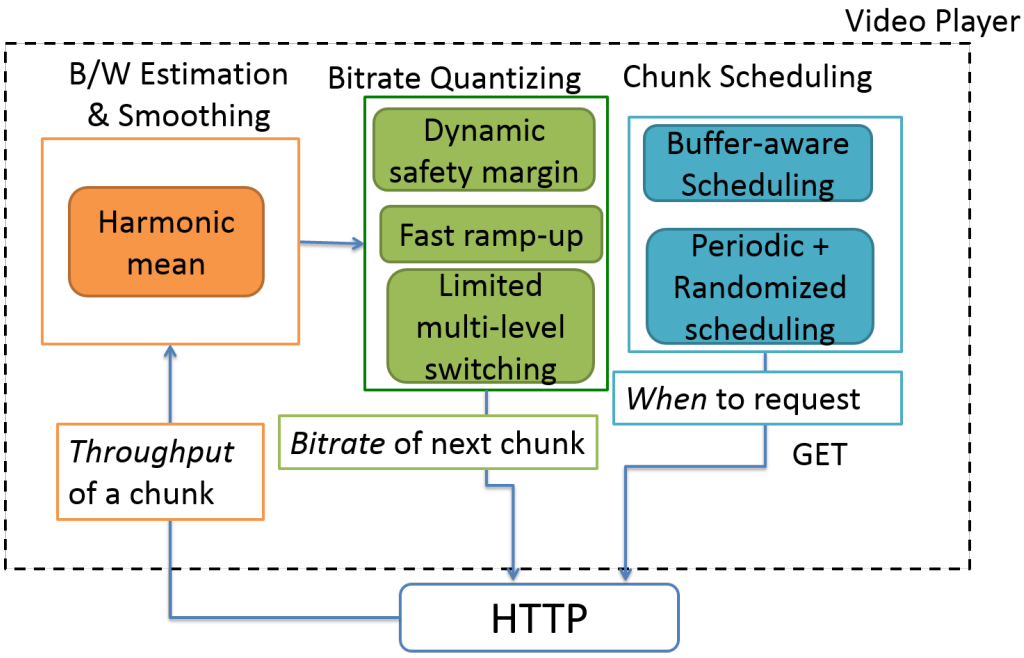



**Figure 17: Components of the MASS Quality Adaptation Scheme.**

model. Lastly, we discuss the intuition behind the techniques and options used in each element of the four-step model.

### 3.3.1 Notation

Given a video file with media segment duration $\tau$, the parameters $i = (1, 2, ..., N)$ denotes the segment indices (playback position), at a specific time $t$ during streaming. Let the reference bitrate encoded by the video provider and available in the media description file be denoted by $b_{r,i} \in \Re_f$ where $\Re_f$ is the set of all available bitrates. The lowest bitrate is represented by $b_{r(min)}$ and the highest bitrate by $b_{r(max)}$. We denote the estimated bandwidth at time $t$ as $x_{i,t}$ and the smoothed bandwidth that is quantized to the reference bitrate by $b_{s(t)}$. The size of the playback buffer at time $t$ is represented by $\beta(t)$. Additionally, we use the configuration parameters $\beta(min)$, $\beta(target)$ and $\beta(high)$ to represent safety margins for the size of the playback buffer and $\epsilon$ to represent the maximum number of bitrate switches in a single epoch.

### 3.3.2 Model

The MASS adaptation scheme closely follows the four-step model in making decisions of whether to switch-up or switch-down based on recent measurements of available throughput. After downloading a new segment, the quality adaptation routine is invoked to determine the reference bitrate for the next segment. Furthermore, the MASS adaptation scheme divides the playback duration into two phases: the *ramp-up* phase ($\eta_{ramp}$) and the *steady-state* phase ($\eta_{steady}$). This distinction allows the system to make more flexible decisions at the beginning of playback until a more reliable and representative measurement of the network bandwidth can be inferred. Based on the results from Chapter 2 (Section 2.3), the ramp-up phase corresponds roughly to the first 30-60 seconds of playback, while the steady-state phase corresponds to the remaining duration of playback.

### 3.3.3 Algorithm Description

The high-level code for the MASS adaptation scheme is shown in Figure 18. Details of the key steps used by the MASS scheme are as follows:

**Estimating step:** After a segment is downloaded, we derive an estimate of the network bandwidth $x_{i,t}$ at time $t$. $x_{i,t}$ is computed from the segment fetch time $T_i$, the segment duration, $\tau$ and the segment bitrate $b_{r,i}$ as $x_{i,t} = \frac{b_{r,i} \cdot \tau}{T_i}$.

The estimated bandwidth gives a snapshot of the network conditions. The computed value is then further refined by the smoothing module to reduce the effect of oscillations in the measurements.

**Smoothing step:** Borrowing from the FESTIVE scheme, MASS uses a *harmonic mean* [12] to smooth out the instantaneous estimates. In MASS, the harmonic mean of the estimated bandwidth $x_{i,t}$ is smoothed over a time windows of $k = 20$ seconds to obtain the smoothed bandwidth $b_{s(t)}$ at time $t$.

**Quantizing step:** After smoothing out the throughput values, a reference estimated bitrate is computed. MASS then uses a quantizing factor to map the smoothed bitrate to the estimated bitrate. The quantizing factor $\varepsilon_{b_{r,i}}$ for switching-up to the next bitrate $b_{r,i}$ is

**Input**:
$\beta(t)$, /* buffer level at time t */
$b_{s(t)}$, /* smoothed bw estimate */
$\eta_{steady}$ /* is steady state? */

**Output**:
$b_{r(t)+1}$, /* bitrate of next segment */
$B_{r(t)+1}$ /* next request time */

```
1  B_{r(t)+1} := 0 ;
2  b_{r(t)+1} := b_{r(t)} ;
3  if b_{s(t)} < b_{r(t)} and b^{↓}_{r(t)} ≥ b_{r(min)} then
4  |  b_{r(t)+1} := b^{↓}_{r(t)} ;
   |  if β(t) >= β(target) then
5  |  |  B_{r(t)+1} := B_{rand} ;
6  |  end
7  else
8  |  if t ≠ η_{steady} and β(t) > β(min) then
9  |  |  b_{r(t)+1} := b^{↑}_{r(t)} ;
10 |  end
11 |  else
12 |  |  count = 1 ;
13 |  |  repeat
14 |  |  |  if b_{s(t)} > b^{↑}_{r(t)} then
15 |  |  |  |  b_{r(t)+1} := b^{↑}_{r(t)};
16 |  |  |  end
17 |  |  |  count = count + 1 ;
18 |  |  until count ≤ ε and b_{r(t)+1} ≤ b_{r(max)};
19 |  |  if β(t) > β(high) then
20 |  |  |  B_{r(t)+1} := B_{rand} ;
21 |  |  end
22 |  |  else
23 |  |  |  B_{r(t)+1} := max(0, β(t) − β(target)) ;
24 |  |  end
25 |  end
26 end
```

**Figure 18: Algorithm: MASS Quality Adaptation.**

calculated from the encoded media bitrates from the video media description file as

$\varepsilon_{b_{r,i}} = 1 - \sum_{i-1}^{i+1} \frac{b_{r,i+1} - b_{r,i}}{b_{r,i}}$,

i.e., we take the average of the difference between the values of reference bitrate, the previous bitrate and the next bitrate to determine the quantizing factor for that bitrate.

Using a quantizing factor to reduce over-estimation of available bandwidth was proposed in earlier schemes (LIU, MILLER, and PANDA). However, these schemes use a fixed factor for quantization. The fixed quantizing factor does not address the gaps between the elements of the reference bitrates $\Re_f$. The dynamic quantizing factor in MASS helps to account for the gaps between the reference bitrates as well as approximate a stateful quantization as used in FESTIVE to handle high variability in the estimated bandwidth.

After applying the quantizing factor to the smoothed estimated bandwidth, MASS proceeds with selecting the reference bitrate for the next segment. MASS switches up to the next quality if the smoothed bandwidth is greater than reference bitrate and switches down to the previous quality level if the smoothed bandwidth is less than the current bitrate. Based on insights from the experimental results in Chapter 2 (Section 2.3.2 and Section 2.3.6), MASS uses additional techniques for optimizing the efficiency and stability of the quality adaptation. The techniques include fast switching during the ramp-up phase, and limited multi-level switching.

From the results in Chapter 2 (Section 2.3), the OSMF scheme which uses multi-level switching outperforms the other schemes on the inefficiency metric and average bitrate. However, it performs poorly on the instability metric and rebuffering ratio. Similar to the OSMF and LIU schemes, MASS uses multi-level switching to quickly match the network conditions and improve the bandwidth utilization.

In the ramp-up phase, as long as there are no re-bufferings and the estimated throughput yields a new reference bitrate, MASS continues to switch-up to the next quality level provided the buffer size is greater than the minimum buffer size $\beta(min)$.

The intuition behind the approach is to reduce the impact of the fluctuations in the network conditions by examining the size of the buffer. Also, by using limited multi-level switching, the MASS scheme can remain reactive to the progressive increases in available

bandwidth while at the same time reducing the possibility of bandwidth over-estimation.

**Scheduling step**: The scheduling step borrows ideas from the approach used by the FESTIVE and the MILLER schemes. It extends those approaches by considering the buffer size. If the current buffer size $\beta(t)$ is less than the target buffer size $\beta(target)$, MASS immediately schedules the next segment to be downloaded. On the other hand, if the buffer size is greater than $\beta(target)$, MASS uses a randomized scheduler similar to FESTIVE, i.e., it schedules the next segment for download by selecting a random buffer size $B_{rand}$ from the range $\beta(target) - \delta, \beta(target) + \delta$. The next segment is then downloaded once the current buffer size is less than the selected random buffer size $B_{rand}$. MASS only uses the randomized scheduler when switching up if the buffer size is greater than $\beta(high)$, i.e., MASS maintains the current periodic download schedule in the switching-up phase if it is detects increasing network throughput. It only probes the network with a randomized scheduler if the buffer size is greater than $\beta(high)$.

These mechanisms allow MASS to reduce the likelihood of playback rebufferings by maintaining the buffer level within the target interval. They also help to reduce the impact of short-time spikes in available throughput, which is more commonly experienced when streaming with cellular connectivity.

In designing MASS, we adopted the best practices of existing schemes (i.e., randomized scheduling, harmonic mean smoothing, and using a quantizing factor) and added modifications to improve the utilization of available bandwidth (the inefficiency metric) and the smoothness of quality switching (the instability metric). The key differences between MASS and existing schemes are in its use of limited multi-level switching, the playback buffer size, and the dynamic quantizing factor for deriving the reference bitrate from smoothed estimates of the network bandwidth (Figure 17).

### *3.4 Evaluation: Component-wise Performance Tradeoffs*

In this section, we perform several measurements with the MASS scheme using the experimental setup described in Chapter 2 (Section 2.2). We then analyze the tradeoffs of

configuring different components using the mobile bandwidth trace for the four metrics of inefficiency, instability, buffer undershoot, and rebuffering ratio. Each individual experiment uses a different value or one of the specific parameters of the component being configured, while the remaining parameters use the default values as shown in Table 2. In computing our results, we average the values for each metric over 40 trials.

**Table 2: MASS Configuration Parameters for Performance Tradeoffs**

| Parameter | Default Value | Configuration Values |
|---|---|---|
| Target Buffer Level ($\beta(target)$) | 30 | 25, 30, 35, 40, 45 |
| Minimum Buffer Level ($\beta(min)$) | 10 | 5, 10, 12, 15, 18, 20 |
| Max Switches Per Epoch ($\epsilon$) | 2 | 1, 2, 3, 4, 5, 6, 7 |
| Random Buffer Offset ($\delta$) | 10 | 3, 4, 5, 6, 7, 8, 9 , 10 |

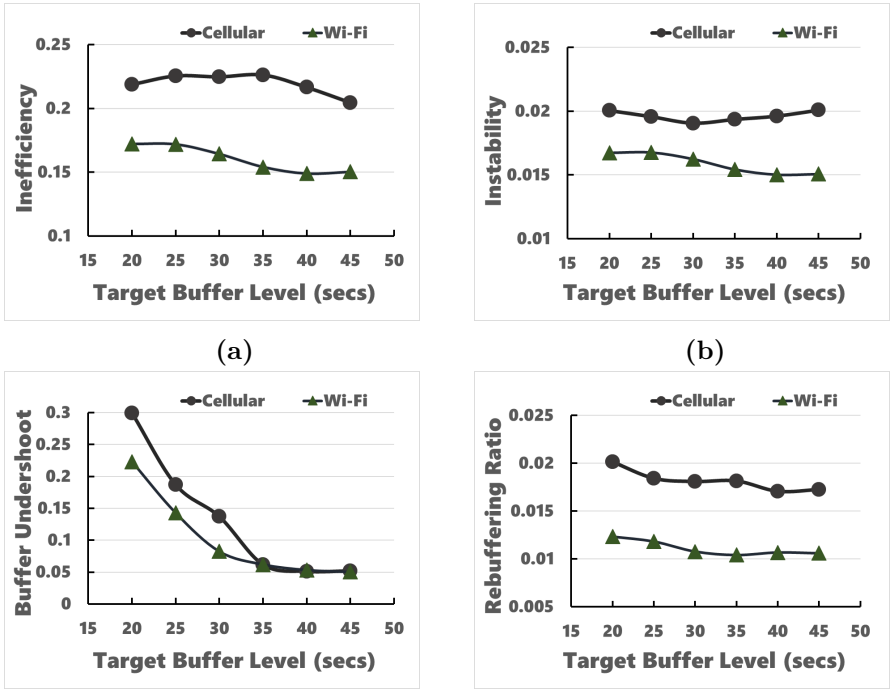



**Figure 19: Impact of Target Buffer Level.**

### 3.4.1 Impact of Target Buffer Level

Figure 19 shows the tradeoffs of setting the target buffer level $\beta(target)$. As can be seen, the buffer undershoot decreases rapidly as the $\beta(target)$ is slowly increased from 20 seconds to 45 seconds. Similarly, two additional metrics (instability and rebuffering ratio) decrease progressively as the $\beta(target)$ is slowly increased from 20 seconds to 45 seconds. For the inefficiency metric, $\beta(target)$ gradually decreases up to an optimal value of 40 seconds. This result implies that higher values of the buffer size helps to reduce the possibility of playback stalls since we see gradual reductions in the buffer undershoot and rebuffering ratio metrics as well as increased stability of the bitrate adaptation. At the same time, the inefficiency metric also benefits from larger values of target buffer level up to an optimal value of 40 seconds. The results in Figure 19 indicate that $\beta(target)$ is a very important parameter for optimizing the bitrate adaptation.

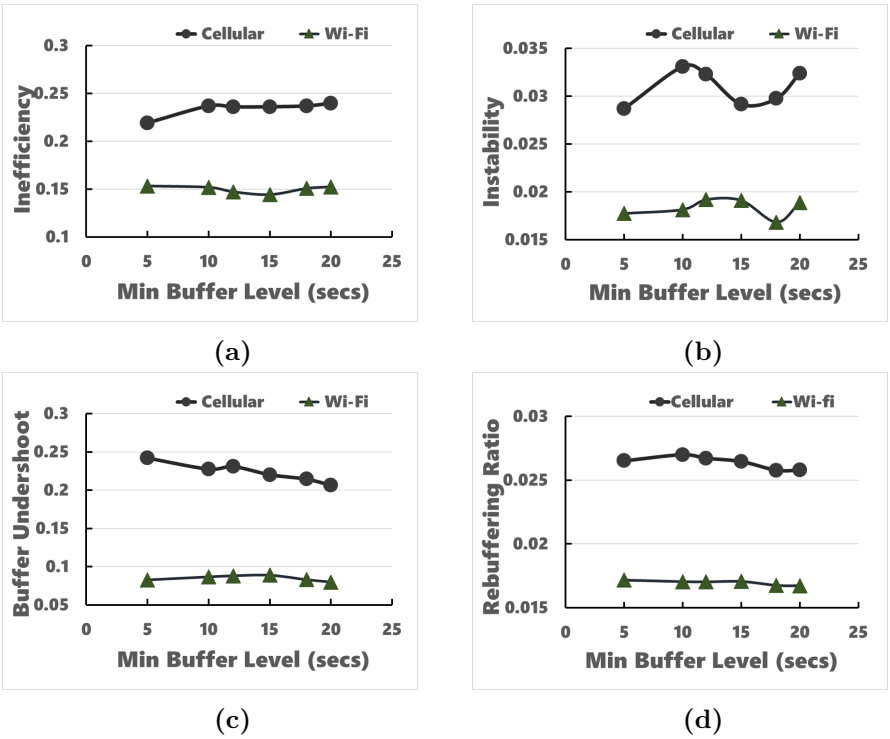



**Figure 20: Impact of Minimum Buffer Level.**

### 3.4.2 Impact of Minimum Buffer Level

In this experiment, we vary the minimum buffer level $\beta(min)$ while keeping the default values for the other parameters. As explained in Section 3.3.3, the minimum buffer level helps to safeguard MASS from over-quantization of the reference bitrate and to respond to sharp drops in network bandwidth.

Figure 20 shows the impact of the varying minimum buffer level from 5 seconds to 20 seconds on the inefficiency, instability, buffer undershoot, and rebuffering ratio. Notice that the minimum buffer level has moderate impact on the inefficiency metric and uneven effect on the instability metric. We speculate that the reason for the moderate impact is because each additional increase in $\beta(min)$ yields more aggressive bitrate switching until the value of minimum buffer level gets close to the value of the target buffer level $\beta(target)$. On the other hand, the minimum buffer level has a slight impact on the buffer undershoot and rebuffering ratio metrics. The results in Figure 20 imply that choosing the appropriate minimum buffer level will help to reduce the rate of buffer underruns and rebufferings.

### 3.4.3 Impact of Number of Switches Per Epoch

In this experiment, we consider the effect of increasing the maximum number of switches that can be made within one epoch ($\epsilon$) on the performance of the quality adaptation. The motivation for this experiment is to determine the effectiveness of our limited multi-level switching versus the single-switching approach (as used in the MILLER and the PANDA schemes) and the full multi-switching approach (as used in the OSMF and the LIU schemes). As can be seen in Figure 21, increasing the maximum number of switches per epoch has significant effect on all the quality metrics. Specifically, each additional increase in the value of the maximum number of switches leads to considerable improvements in the inefficiency of the adaptation up to the threshold value of 4 switches per epoch before a reversal in the quality gains. Additionally, as we would expect, the increases also results in higher instability, higher buffer undershoot, and more frequent rebufferings. The result in Figure 21 implies that limited multi-level switching is more effective than single-switching and full multi-level switching for optimizing key performance tradeoffs. Therefore, a careful choice

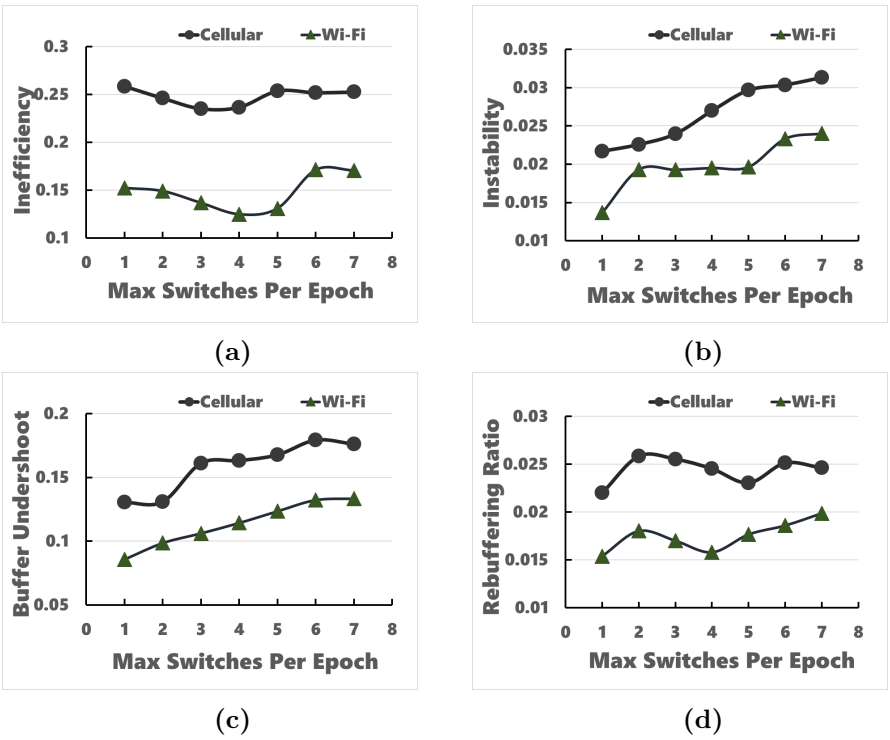



**Figure 21: Impact of Number of Switches Per Epoch.**

of the maximum number of switches per epoch is a key decision for improving the bitrate adaptation.

### 3.4.4 Impact of Random Buffer Size

For this experiment, we look at the impact of configuring the range of values for the random buffer level $B_{rand}$. Recall that $B_{rand}$ is used as a factor to determine when to schedule the next segment, i.e., to assist in minimizing the effect of segment request times on the estimation of the network bandwidth. For each value of the random buffer offset $\delta$, we choose a value between the range $\beta(target) - \delta, \beta(target) + \delta$ to wait before sending out the request for the next segment whenever the current buffer level is above the target buffer level $\beta(target)$. The results are shown in Figure 22.

The figure shows that an offset value has minimal impact on the inefficiency and instability and uneven impact on the buffer undershoot and rebuffering ratio. From the results, we choose a default offset value of 8.

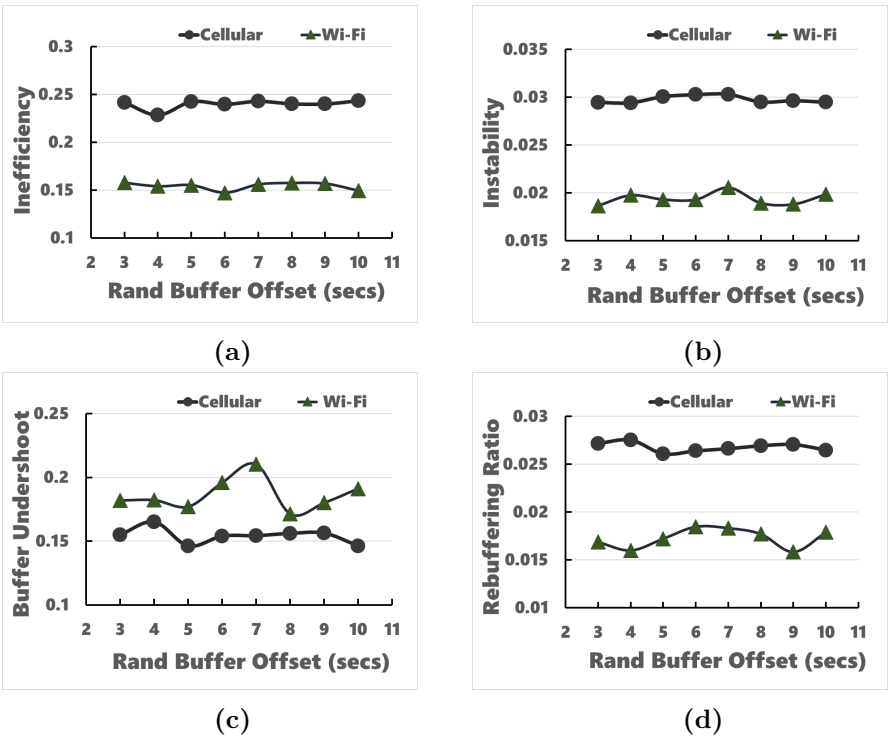



**Figure 22: Impact of Rand Buffer Offset.**

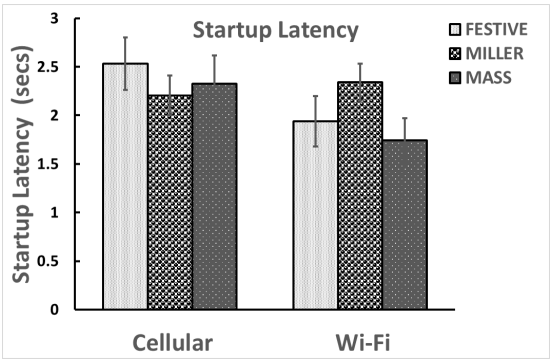
### *3.5 Evaluation: Comparison with Existing Schemes*

**Table 3: Parameters Used in Evaluating MASS versus Existing Schemes**

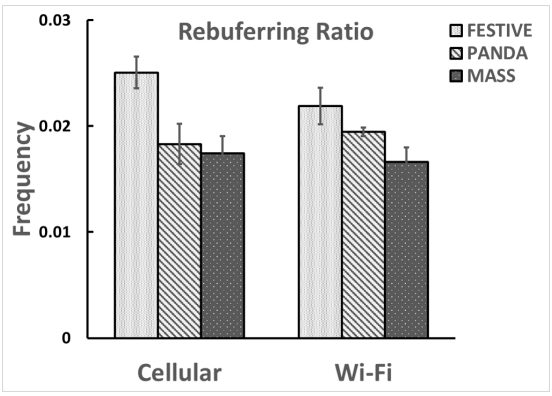
| Parameter | Cellular Value | Wi-Fi Value |
|---|---|---|
| Target Buffer Level ($\beta(target)$) | 35 | 40 |
| Minimum Buffer Level ($\beta(min)$) | 15 | 18 |
| Max Switches Per Epoch ($\epsilon$) | 3 | 4 |
| Random Buffer Offset ($\delta$) | 8 | 8 |

In this section, we conduct a set of experiments to evaluate the performance of MASS against other quality adaptation schemes. We have implemented the MASS scheme on the Android system using the measurement infrastructure described in Chapter 2 (Section 2.2). We use optimal parameters derived from our measurements in Section 3.4 in the experiments. The parameters are summarized in Table 3.

Figure 23 shows the average video startup time and rebuffering ratio experienced across all the playback sessions by the MASS algorithm. The results show that MASS experiences

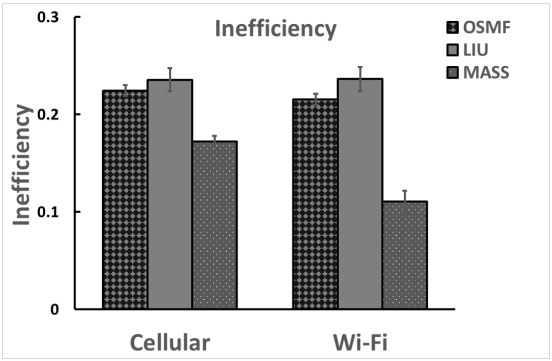



(a) Startup Latency

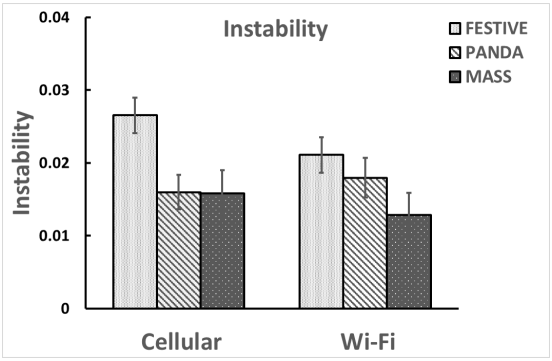



(b) Rebuffering Ratio

**Figure 23: Startup Latency and Rebuffering Ratio of the MASS scheme.**




(a) Inefficiency




(b) Instability

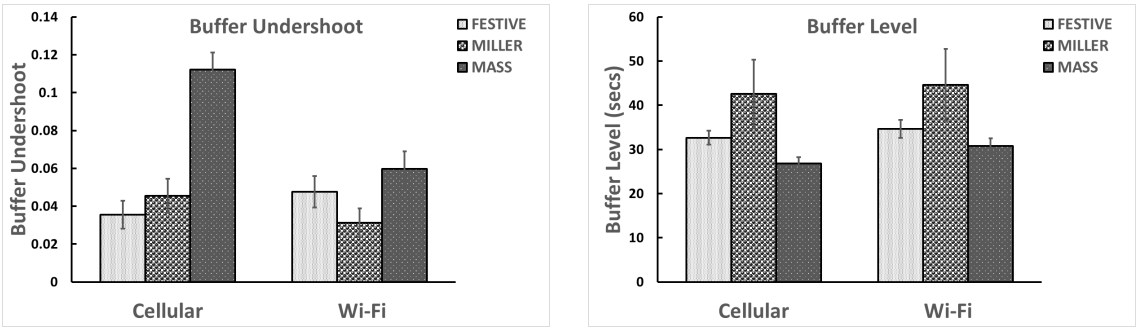
**Figure 24: Inefficiency and Instability of the MASS scheme.**

a startup delay of less than 2 seconds on Wi-Fi and less than 2.3 seconds on cellular (Figure 23a). MASS also experiences very little rebuffering (less than 2%) on both Wi-Fi and cellular connectivity (Figure 23b).

In Figure 24a, we show the inefficiency metric for MASS, i.e., its ability to efficiently utilize the available bandwidth. From the results in 24a, we see that the MASS scheme outperforms the existing schemes. The efficiency of quality adaptation increases to more than 89% from up to 78% achieved by the OSMF scheme on Wi-Fi network access, and increases to more than 83% from up to 74% achieved by the OSMF scheme on cellular network access.

Figure 24b shows the smoothness of the quality adaptation as defined by the instability

metric. We observe that MASS is able to follow the dynamics of the average available bandwidth in a robust manner with both Wi-Fi and cellular networks.




**(a) Buffer Undershoot** **(b) Buffer Level**

**Figure 25: Buffer Undershoot and Buffer Level of the MASS scheme.**

In Figure 25, we show the buffer undershoot and buffer level metrics. The results show that the MASS scheme performs worse than the state-of-the-art in its ability to meet its target buffer level on both cellular and Wi-Fi connectivity. We note that MASS performance is higher than that of PANDA, LIU, and OSMF schemes (see Figure 13 and Figure 14). The results in Figure 25 reflect the impact of multi-level bitrate switching on the overall performance of MASS. Recall that multi-level bitrate switching enables MASS to efficiently utilize the available network bandwidth. However, switching between multiple bitrates in one epoch also reduces MASS's ability to meet its target buffer level in the event of sudden decrease in the available bandwidth since the requested segments will take longer to download than predicted by the MASS scheme. We note that the effect of the buffer undershoot metric on the overall performance of MASS is minimal because its average buffer level of 26.8 seconds on cellular and 29.5 seconds on Wi-Fi is close to the target value of 30 seconds used by the existing schemes. Also, MASS experiences lower rate of rebufferings than the existing schemes.

The battery depletion due to MASS is shown in Figure 26. The figure shows that MASS does not incur significant battery depletion. The results of MASS are comparable to the performance of existing schemes.

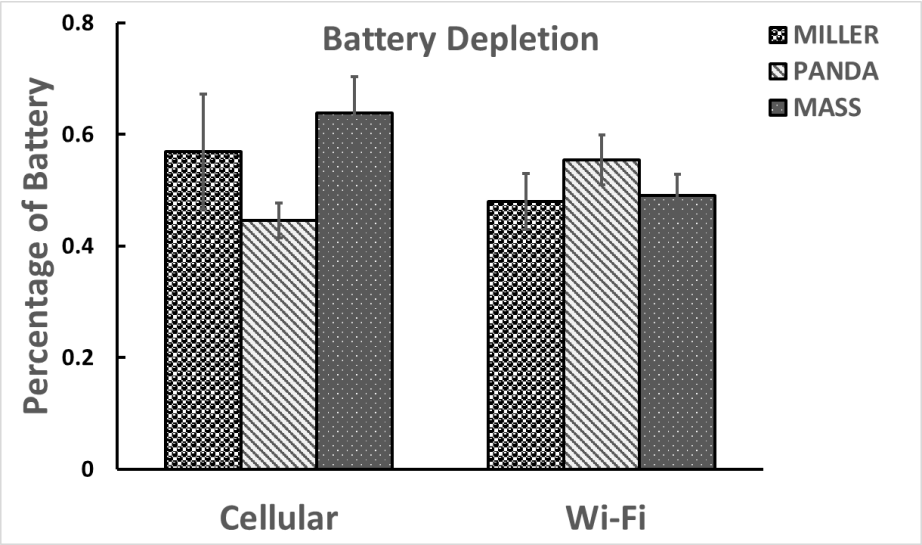



**Figure 26: Battery Depletion of the MASS scheme.**

In summary, MASS outperforms the existing schemes on key metrics of rebuffering ratio, inefficiency, instability, and battery life. On the startup latency metric, MASS outperforms the state-of-the-art by more than 12% on Wi-Fi network access. The startup latency of MASS on cellular network access is slightly worse than the that of the MILLER scheme by less than 4% (from 2.22 seconds to 2.31 seconds) but smaller than that of the remaining schemes. Also, MASS performs slightly worse than the state-of-the-art on the buffer level and the buffer undershoot metrics. The decrease in MASS's ability to meet its target buffer level is mainly due to its use of multi-level bitrate switching during a single epoch. However, the impact of the buffer undershoot metric on the overall performance of MASS is minimal since the average buffer level of MASS is close to the target value of 30 seconds used by the existing schemes. Also, by carefully determining the number of bitrate switches in a given epoch, MASS is able to achieve significant improvements in efficiency at no costs to its stability and responsiveness. Furthermore, by incorporating buffer-aware strategies, MASS is able to achieve lower rate of rebufferings than the existing schemes. The results show that through our extensions to AHS quality adaptation, we can reduce the impact of network fluctuations and improve overall streaming experience.

## *3.6 Summary*

In this chapter, we have presented a new scheme for AHS quality adaptation to changes in the network throughput experienced by mobile devices. Based on analysis of previous research and a measurement-study of the performance of existing schemes on mobile devices, we identify key optimization goals for our Mobile Adaptation AHS Scheme (MASS): avoiding interruptions of playback due to buffer underruns, maximizing the average video quality, minimizing the number of bitrate shifts, maximizing the use of available network bandwidth, and providing uniform performance across multiple forms of mobile Internet connectivity. To handle the challenge of meeting these optimization goals, we leveraged the best-practices of existing schemes and also introduced new techniques for tuning the robustness of the rate adaptation scheme. These techniques include limited multi-bitrate switching, dynamic quantizing safety margin, and fast ramp up. We then conducted experiments into the performance tradeoffs for key quality metrics when varying the values of multiple underlying components. Using the derived optimal values of the configuration parameters, we compared the performance of MASS with existing schemes. The results show that MASS outperforms existing schemes on key metrics.

# CHAPTER IV

# COOPERATIVE EXTENSIONS TO ADAPTIVE MOBILE STREAMING

## *4.1 Introduction*

In the preceding chapters, we have studied the performance of several AHS schemes on mobile devices and have proposed a novel scheme, MASS, which leverages key-insights from our evaluation of existing schemes and best practices of the state-of-the-art to improve the streaming quality. In this chapter, we build on the research presented in earlier chapters by presenting a system for improving the Quality of Experience for mobile video streaming through social participation.

Video access on mobile devices differs from video consumption through fixed computing devices. Smartphones are often equipped with video cameras, GPS and other sensors, leading to the emergence of user-generated content and the increasing use of mobile platforms as both the sources and destinations for video traffic. Similarly, sharing and interactions of video links through "friends" in social networks and location-based news spreading are two key drivers of mobile video traffic: 71% of users (mostly the adult population) in the US use video-sharing sites [107]; there are more than 70 million re-shares of video on Facebook per month [52, 125]; and content sharing on mobile devices grew by more than a factor of eleven in 2011 alone [107]. There is also increasing dissemination of video content among users of mobile devices with shared interests in an *ad hoc* manner. This pattern of mobile video consumption provides different opportunities from video access on fixed computing devices. Unfortunately, existing video delivery technologies are designed for traditional Internet users, ignoring the unique demands of mobile multimedia access. This model ignores the cooperative nature of mobile video access, requiring all the devices to fetch the complete video segments from the originating servers. Given the increasing shift to mobile devices as the primary endpoints for consuming videos, we believe that there is a need to rethink the

system infrastructure used for delivering video to end users.

In this chapter, we propose to augment the client-server communication used by existing video adaptation schemes with cooperative techniques that exploit the synergy between video sharing on social networks and streaming of videos by mobile users in order to further optimize the viewing experience. Given that "friends" in a social network have similar interests and share knowledge and experience with one another, the effectiveness of P2P sharing can be greatly enhanced by exploiting this knowledge. By caching video segments and servicing requests for the cached segments, social peers who are sharers of the video (or *sharer network*) can help to improve the Quality of Experience (QoE) across multiple streaming sessions (Figure 27). Furthermore, SDASH takes advantage of the increasing sophistication of mobile devices in terms of computing power and the availability of mobile devices in a range of form factors (smartphones, tablets, phablets, etc.), suitable for video consumption and P2P communication. The optimization objective of SDASH is obtaining higher segment bitrates and picture quality across time and location for a given user by using information available in both the sharer network (spatially and/or temporally co-located users) and the origin video server. In SDASH, we use local peer caching and a P2P-based adaptation algorithm to meet the optimization objective. Each peer caches only a single version (video representation or bitrate) per video segment for a subset of the video. The peer-assisted component determines the optimal source for the next segment request based on the dynamic network conditions.

The technical challenges addressed in this chapter, namely, caching, prefetching, and segment scheduling using cooperation among the members of the sharer network are not unique to SDASH and have been studied by other related work. However, the application of these concepts to mobile devices and AHS video streaming presents unique challenges. Mobile devices are resource-constrained, therefore, there is a pressing need to exploit the device context when making prefetching and scheduling decisions. We also need new techniques for quality adaptation that minimizes frequent bitrate fluctuations by leveraging the caches in the sharer network.

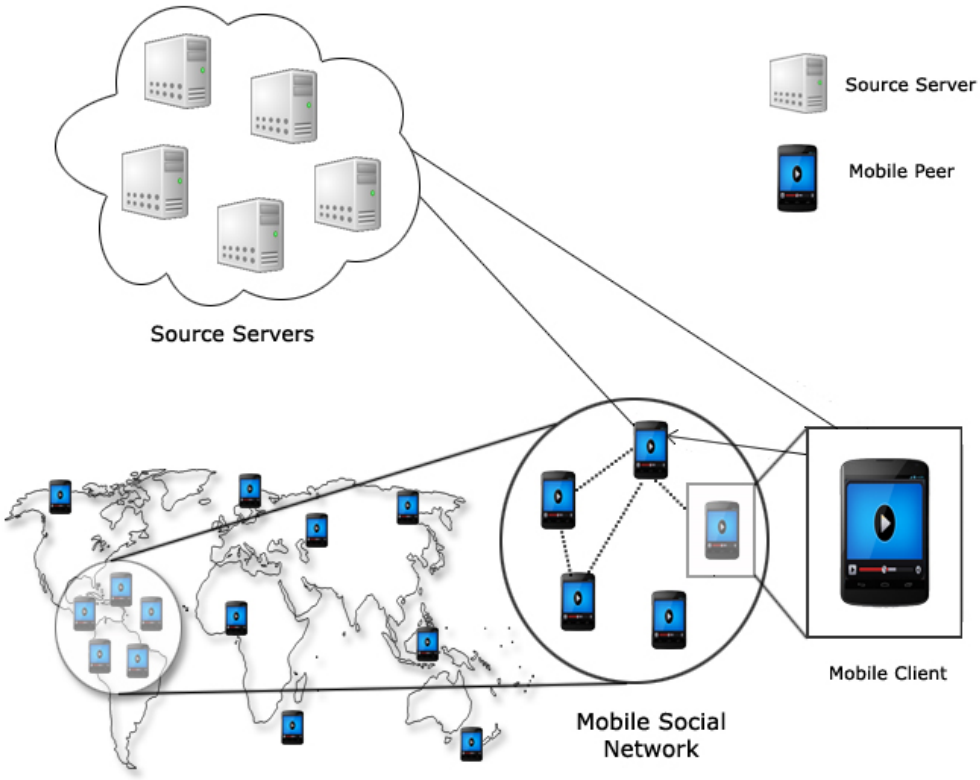



**Figure 27: Cooperative Streaming with SDASH.**

The unique contributions of our work are as follows: (a) study of cooperation opportunities and the potential costs and benefits of exploiting cooperative caching, prefetching, and P2P streaming, (b) systemic approach for exploiting cooperation among sharers of video for improving the quality of experience, (c) novel prefetching and bitrate adaptation techniques that are sensitive to device context and are resource-aware, and (d) design, implementation, and evaluation of the SDASH system.

In the rest of the chapter, we first motivate SDASH in Section 4.2 with sample scenarios and studies on the potential benefits and costs of cooperative streaming. We present the design of SDASH in Section 4.3 and the key mechanisms underlying the SDASH system in Section 4.4. Section 4.5 describes the implementation of core components. The experimental setup and results using smart phones are covered in Section 4.6. We review related work in Section 4.7 and present concluding remarks in Section 4.8.

## *4.2 Opportunity for Cooperative Streaming*

In this section, we present preliminary evidence that cooperative techniques can indeed provide significant benefits to mobile video streaming. Towards this goal, we answer three specific questions. 1) How interested are users in videos shared with them on social networks? 2) How does the average network throughput from cooperative techniques (caching and P2P networking) compare to the network throughput from regular client-server communication experienced by mobile devices?, and 3) What would be the impact of cooperative techniques in terms of resource usage? Our results provide insights that shape the design of a cooperative video streaming system. The performance gains of cooperative streaming are further elaborated through comprehensive experiments in Section 4.6.

### 4.2.1 Diffusion of Video Sharing and Co-location Opportunities

Video sharing sites such as YouTube, Facebook, and Vimeo are increasingly popular. In this section, we investigate the video viewing relationships between friends in the Facebook social network. Because we do not have access to the full social graph, we collected a simple sharing dataset that gives insights into the participation of users for videos that were shared by their "friends" by crawling the profiles of twelve volunteer users (seeds) and their friends on Facebook. We use the Facebook Graph API to access the lists of "friends", "posts", "links", and "newfeeds" of the users. We then identified videos from other posts on user profiles by checking for the "video" tag and searching for video domain names such as "youtube.com" and "vimeo.com" in the url of the posted links. We also recorded friendship information, and the "likes", "comments", and "targets" of the videos when available. Because of privacy and user settings, we found only 1966 videos from 2743 users between the period of July 2008 and May 2013. We aggregated the number of "likes" and "comments" on each posted video. We found an average of 5.7 video postings per user and **9.4** "likes" and "comments" on each posted video (Table 4). Since there are likely to be more viewers of a video than only the users who "liked" or "commented" on the video, the results suggest a significant engagement of "friends" with the videos shared by users on social networks. In Section 4.4, we describe prefetching algorithms that exploits the sharing

ties between users for improving the video viewing experience.

**Table 4: Facebook Sharing Dataset. The number of likes/comments per video (9.4) suggests high-degree of participation by "friends" in social networks.**

| Item | Value |
|---|---|
| Seeds | 12 Users |
| Dates Videos Posted | 7/2008 - 5/2013 |
| Total Number of Users | 3120 |
| Total Number of Videos | 1996 |
| Average Video Per User | 5.7 |
| Average Likes/Comments Per Video | 9.4 |

In CoMon [85], the authors analyzed the American Time Use Survey (ATUS) dataset [35] to quantify the amount of time people spend together within a 24-hour period. For each individual, the average meeting time with one or more acquaintances is 8.5 hours; with 78% of the participants spending more than 4 hours and 50% of the participants spending more than 9.3 hours of co-located time with acquaintances. Furthermore, each individual is co-located with more than one acquaintance 42% of the time, while 65% of co-located meetings last for more than 30 minutes and 47% of the meetings last for more than one hour. Coupled with results from recent studies [100, 124, 45] which show a high-degree of correlation between membership of social networks and co-located networks, this implies that there are multiple opportunities for "friends" in the same social network to augment their video streaming activities by exchanging information virtually through their mobile devices.

### 4.2.2 Network Throughput of Cooperative Techniques

We investigate the throughput characteristics from cooperative techniques versus the throughput from downloading segments from the video server. We use the same dataset presented in Chapter 2 (Section 2.2.1) in our evaluation of existing schemes and the MASS scheme. The dataset consists of segments from the Big Buck Bunny video [5]. The segments were encoded using x264 encoder at 14 different bitrates (100, 200, 350, 500, 700, 900, 1100, 1300, 1600, 1900, 2300, 2800, 3400, and 4500 kbit/s). The segment duration was 2 seconds and the Group of Pictures (GOP) size was 48 frames. Table 5 shows the average

throughput, maximum throughput and the standard error obtained from fetching segments from the following sources: local SD Card, video server with 4G connectivity, video server with 3G connectivity, video server with Wi-Fi, and nearby peers with Wi-Fi using a Google Android's Nexus IV smartphone.

**Table 5: Average Throughput from Different Sources in SDASH. Although median 4G throughput is higher than that of P2P and Wi-Fi, cooperation-assisted streaming with P2P and caching can help improve end-user QoE by reducing the impact of frequent fluctuations in observed throughput when using wide-area Internet.**

| Source | Avg. T'put | Max. T'put | Std. Error |
|---|---|---|---|
| Cache (SD Card) | 113 Mbps | 153 Mbps | 3.66 Mbps |
| Server (3G) | 634 Kbps | 1.9 Mbps | 42.1 Kbps |
| Server (4G) | 4.7 Mbps | 11.3 Mbps | 91 Kbps |
| Server (Wi-Fi) | 3.3 Mbps | 11.9 Mbps | 53.5 Kbps |
| P2P (Wi-Fi) | 3.98 Mbps | 8.7 Mbps | 85.3 Kbps |

The values show the potential benefits of P2P streaming and cooperative caching. Although the throughput values of P2P is less than the values for 4G, the average throughput from P2P is higher than that of 3G and Wi-Fi. Also, because of the vagaries of the Internet due to congestion and cross-traffic, video streaming with cellular and Wi-Fi connectivity faces frequent fluctuations in data rates, end-to-end delay, and packet loss [63, 71]. Furthermore, the observed throughput from cellular connectivity is affected by spotty coverage and high service loss rate (which can be up to 50% [120]), leading to sudden drops in video quality and high frequency of rebufferings. Therefore, combining P2P with server-based network access types offers the potential of mitigating the vagaries in network conditions and improving end-user QoE. In Section 4.6, through extensive experimentation, we validate this potential for P2P and cooperative caching.

### 4.2.3 Resource Usage of Cooperative Techniques

Next, we investigate the potential resource consumption of cooperative mobile streaming. Recent studies indicate that the 40% of videos watched on mobile devices are of length of less than 10 minutes [104, 121], with typical encoding bitrate of 500 Kbps (SD) and 2 Mbps (HD) [28]. We base our analysis on the results from these studies by considering

the costs of streaming the Big Buck Bunny video of 5 minutes duration. Table 6 shows a simple back-of-the-envelope calculation of the amount of SD Card storage space required for caching 30 seconds to 4.5 minutes length of video at three different bitrates using the same video dataset presented in the previous section. The table shows that caching 30-seconds length of the video at 4500 Kbps will require 15 MB storage space on average and at least 15 MB data transfer through the network while caching 1-minute length of the video at 2300 Kbps will require 14 MB storage space and data transfer. Given that today's mobile devices are equipped with SD cards of sizeable capacity (8 GB to 64 GB on an average [64], with additional expansion slots), we argue that caching files on the local SD Card will have only minimal impact on consuming space on the storage device. However, for users with limited cellular data quotas, prefetching files from the sharer network through cellular connectivity may consume scarce network bandwidth. In such cases, it will be prudent to only prefetch files when the device is connected to the Internet via a Wi-Fi access point.

**Table 6: Estimated Storage Space Needed for Caching.**

| Video Bitrate | Video Length | Cache Size |
|---|---|---|
| 500 Kbps (SD) | 4.5 minutes | 14.5 MB |
| 2300 Kbps (HD) | 1 minute | 14 MB |
| 4500 Kbps (HD 1080p) | 30 seconds | 15 MB |

We measure the power consumption of cooperative prefetching by using an implementation of a video prefetcher that queries Facebook for newly shared content. Full details of the prefetching component is provided in Section 4.4.2. Table 7 shows the energy costs of prefetching 1-minute length of the video at 500 Kbps, 2300 Kbps and 4500 Kbps measured in terms of battery level. The results show that the battery depletion due to segment prefetching is minimal (less than 0.5%). In Section 4.3 and Section 4.4, we describe the context-aware techniques used by SDASH in managing resource usage in detail.

**Table 7: Battery Depletion due to prefetching 60-seconds length of segments.**

| Video Bitrate | Checking for Videos | Downloading Segments | Total Usage |
|---|---|---|---|
| 500 Kbps | 0.11% | 0.22% | 0.33% |
| 2300 Kbps | 0.11% | 0.29% | 0.40% |
| 4500 Kbps | 0.11% | 0.34% | 0.45% |

## *4.3 SDASH System Design*

### 4.3.1 Cooperation-assisted Approach

SDASH introduces three additional operations to a traditional on-demand video streaming: (1) caching of the video segments locally after streaming, (2) prefetching segments of shared videos ahead of the actual time of streaming, (3) retrieving segments from either the mobile peers or from the originating video servers at the point of streaming. SDASH augments the client-server communication with these cooperative techniques to provide an improved streaming experience. To reduce the impact of dynamic join and leave of peers and minimize the startup latency (Section 4.6.2), SDASH uses the server as the default source for fetching segments.

### 4.3.2 Architecture Overview

For a given mobile device utilizing SDASH for video sharing and streaming, the SDASH service coordinates the interaction between the device, the other devices in its sharer network, and the web of video servers where the shared video originated from (Figure 27). The user's sharer network consists of the subset of its social network's membership graph that is utilizing the SDASH system for video streaming. SDASH manages communication among these three entities, and determines where to send the next segment request that will best help optimize the streaming quality. The overall design of SDASH is guided by three goals. The first goal is to minimize the startup latency while streaming a given video. The second goal is to maximize the average picture quality of the streamed video across space (i.e., as the user moves around with his mobile device) and time. The third goal is to minimize the energy and data costs incurred by a sharer device commensurate with its chosen degree of participation in the network. SDASH accomplishes these goals by combining the information specified in the video XML file with the information available in the overlay sharer network, the current device context, and the current network conditions at the mobile client.

Figure 28 illustrates the components of the SDASH system and their interactions. The

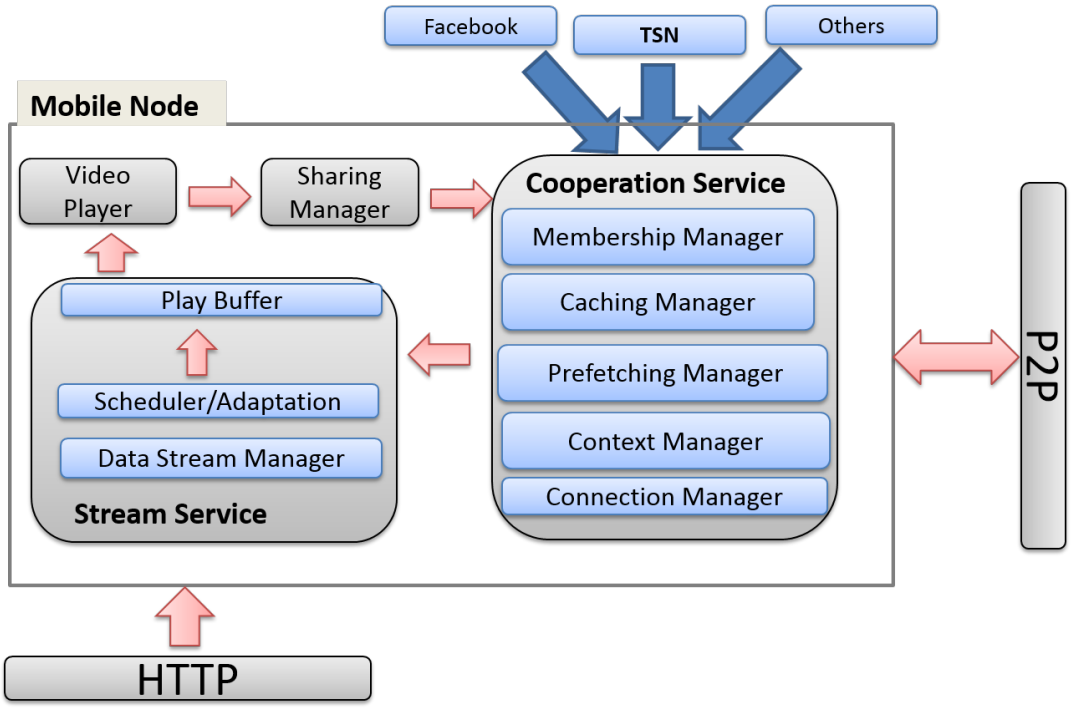



**Figure 28: SDASH Architecture: Elements of the software architecture contained in a smartphone to support SDASH.**

**Cooperation Service** is responsible for the cooperative features of SDASH via five subcomponents: the *Membership Manager*, *Caching Manager*, *Prefetching Manager*, *Context Manager*, and the *Connection Manager*. The logic for managing the data streams and the quality adaptation is handled by the **Stream Service**.

The *membership manager* handles updating the current lists of friends that a mobile client has. It is responsible for updating changes in the membership as the participant adds or removes friends in his/her friends' list. It also interacts with the social network to dynamically discovers new friends and nearby devices. The *sharing manager* publishes streamed videos which are shared by the participant with its social network.

The *prefetching manager* is responsible for syncing changes in its sharer network with the tracker. It implements content prefetching and communication with the tracker. It determines whether to prefetch each of the newly shared video based on the device context

using the techniques described in Section 4.4.2. The received segments are cached on the device's SD Card. The cached files include the digest of the segments that are streamed, and the raw video (m4s) files that corresponds to the cached segments. The caching manager implements the cache replacement policy for evicting video files as the storage size becomes full. The *caching manager* uses the Android SQLite database for managing the metadata of the cached files.

The *context manager* manages the device context i.e., it implements methods for retrieving the device location, its battery-level, and its connectivity status. These pieces of information are obtained by implementing Android's broadcast receiver (battery level, connectivity status), location manager (location) and traffic stats (mobile data usage) classes.

The interaction between the sharer network and the segment scheduler happens in the *connection manager*, which determines a group of active peers (seeders) for sending and receiving data. It also handles connection initiation and setup between the devices and its selected peers during a streaming session.

The *data stream manager* implements the routines for parsing the video XML file, iterating through the available segment representations. The *segment scheduler* downloads the segments of a video that is currently streamed by the user based on the descriptions of the XML file supplied by the data stream manager. It implements the quality adaptation algorithm described in Section 4.4.3. The downloaded segments are placed in the *playout buffer*. The playout buffer contains the current segments that have been downloaded but have not be consumed by the video player.

### 4.3.3 Design Considerations and Choices

**Client-driven Design:** In order to exploit strong ties and interests between members of the sharer network, we follow a client-driven approach in our design of SDASH, i.e., the user is in control of the prefetching, caching and segment requests operations based on its current device context and its view of its social network and peer set. In this manner, the user can tailor his/her degree of participation and manage SDASH's use of resources such as memory and battery life. This design approach is in contrast with many P2P systems

where random peers and/or geographically distributed nodes form an overlay network.

**Context-aware Resource Management:** The quality of the video consumed as well as the ability and willingness of a smartphone to serve others in the sharer network depends on the *device context*. The device context is a vector of several pieces of information about the device: location, battery level, connectivity, etc. A number of heuristics based on the device context and user preferences is used to make key decisions in SDASH, e.g., determining when to prefetch segments, determining whether to respond to a given segment request from peers in the sharer network or to admit a new peer device into the device's upload list. By carefully considering the device context and user configurations, SDASH can efficiently manage resource budgets such as the cellular bandwidth quota and available battery life.

**Centralized Tracking of Shared Video Segments:** The locally-cached segments are described by the bitrate associated with the downloaded segment, and the original URL through which it was retrieved from the video server. We use a tracker server to manage indexing of video segment files and to coordinate connectivity and membership changes in the sharer network. The tracker is a well-known entity in P2P networks that serves as a rendezvous point for joining peers. A newly joined peer contacts the tracker to retrieve connection information about members of its sharer network. Each peer also synchronizes data (connection information, indexes of the shared video files and cached video segments descriptions, and location information) with the tracker. Each mobile peer communicates with the tracker server to retrieve basic information about the overlay network. The tracker consists of four database indexes: participants, friendshiplist, videolists, and location. The participants index consists of all the users who have registered with the tracker server. The participants also sync their social network friends list with the tracker. Those friends information are kept in the friendship list index.

## *4.4 Cooperative Streaming Mechanisms*

We now turn to the mechanics of utilizing the sharer network for cooperative streaming. We discuss the following components: the heuristics for cooperative caching and prefetching

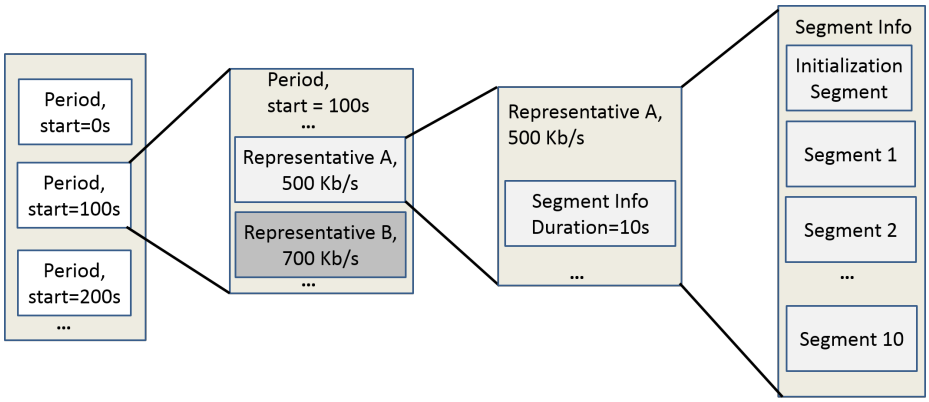
among the mobile devices, and a scheme for peer-assisted streaming which coordinates fetching segments from servers and available peers in the sharer network.

### 4.4.1 Video Representations

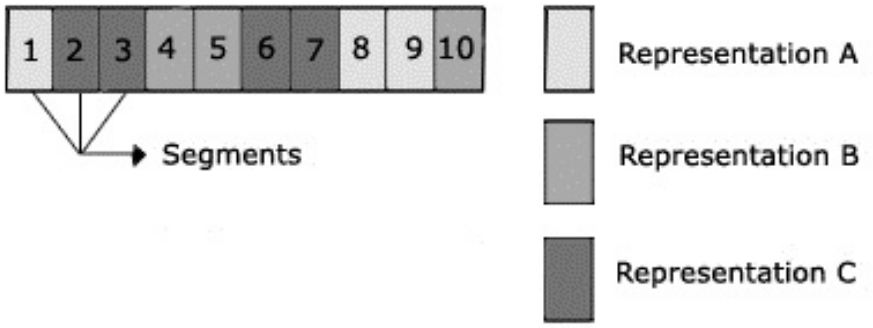
We hereby briefly describe how shared and cached videos are represented in SDASH. Each video file that is shared or streamed on SDASH is represented in a manifest file called the Media Presentation Description (MPD) file (Figure 29a). A MPD file consists of a sequence of one or more playback periods. Each period is non-overlapping and of constant duration. A given period is further sub-divided into one or more representations. Each representation denotes a distinct quality level, i.e., it is a vector of video bitrate, audio bitrate, video resolution, frame rate, and/or other encoding characteristics. Each representation will contains one or more segments of the same media content. By dynamically selecting segments from the listed representations, a media player can adaptively adjust its streaming quality to match its network conditions. In SDASH, we represent the segments that were downloaded by a media player in a Media Cached Description (MCD) file (Figure 29b). The MCD file describes the representation associated with the downloaded segment, and the URL through which it was retrieved from the video server.

### 4.4.2 Segment Caching and Prefetching

SDASH caches previously streamed video segments on the local storage. It also prefetches and caches segments of video files shared with the user. Segment caching and prefetching has two major benefits. First, it helps to reduce the impact of uncertain network conditions, leading to smoother playback of the video. Second, the availability of the video segments for local streaming is enhanced since social peers can fetch some of the cached segments from one another rather than only going to the centralized server. Implementing the appropriate caching and prefetching technique is therefore very essential: the device may be able to stream a particular video at higher quality than the network throughput at the point of streaming, if it has some percentage of the video already cached. However, segment prefetching and caching also incurs network, energy, and storage costs. The predictive accuracy of the prefetching system is therefore very essential to reduce the resource costs




**(a) Media Presentation Description**




**(b) Media Cached Description**

**Figure 29: Representation of Shared and Cached Video Files.**

incurred. We explain how SDASH implements its caching and prefetching mechanism while balancing the device's network-energy-storage-accuracy tradeoffs.

**Determining Prefetched Segments**

We use a video ranking scheme adapted from previous research into social video prefetching [129] to determine how to rank the set of videos shared with the user during a prefetching session. For a given user $i$, the set of all users in its sharer network $N_i$, and the set of all videos $V_i$ shared with the user $i$ but have not been viewed by the user $i$, we calculate the video rank $r_v$ for each video $v \in V_i$. The video rank $r_v$ for video $v$ is estimated from two factors: the historical affinity of the viewing user $i$ for watching the videos shared by all the sharers of $v$ and the popularity of the video $v$ in the sharer network. Let $S_v$ denote the subset of $N_i$ that are sharers of the video $v$, i.e., the participants that have re-shared/cached

the video $v$; and let $c_{ij}$ denote the number of times participant $i$ streamed videos from participant $j \in S_v$. We define an index $a_{ij}$ to evaluate participant $i$'s historical affinity for every other participant $j$ of video $v$ as $a_{ij} = \frac{c_{ij}}{\Sigma_{k \in N_i} c_{ik}}$. We define a popularity rank, $p_v$ for video $v$ as $p_v = \frac{|S_v|}{|N_i|}$. For each video $v$, we give a video rank $r_v$ defined as follows:

$$r_v = \alpha . \frac{\Sigma_{k \in S_v} a_{ik}}{|S_v|} + \beta . p_v \;\; \{0 < \alpha < 1, 0 < \beta < 1\} \tag{1}$$

where $\alpha$ and $\beta$ values are configuration parameters used to adjust the relative weights of historical affinity and video popularity. The first term in equation 1 computes the average of the historical affinity of participant $i$ for all the sharers of video $v$ while the second term computes the popularity of video $v$ as a fraction of the size of the sharer network. Larger $r_v$ indicates the video $v$ should be prefetched ahead of other videos. Our video ranking scheme differs from the scheme presented by Wang et al. [129] in that it follows our client-driven design approach and therefore focuses on only the set of all the participants that have historically shared videos with a given participant $i$ (the sharer network) instead of the larger social network that $i$ belongs to.

**Determining When to Prefetch Segments**

The context of the mobile device (i.e., the battery charging status, the battery level, and whether the device is connected to the Internet via a Wi-Fi network) plays a big role in deciding when to prefetch segments from the sharer network. We use the configuration parameters $\psi_{low}, \pi_{max}$ for determining whether to turn on or disable prefetching. Prefetching is only enabled if the battery level is greater than $\psi_{low}$ and the device is connected to the Internet via a Wi-Fi access point or the cellular data usage is less than a threshold limit $\pi_{max}$. Prefetching is turned off altogether when the battery level falls below $\psi_{low}$. In the current implementation, we use default values of $\psi_{low} = 30\%$ and $\pi_{max} = 75\%$ of cellular bandwidth quota. As we mentioned in Section 4.3.3, a participant can also tune the prefetching parameters to adjust to his/her own specific requirements.

**Determining How Many Segments to Prefetch**

The number of the prefetched videos is modulated based on the average number of videos watched by the user per time period and the amount of cache space available, with 4 videos

used as the initial maximum. Similarly, using a starting duration of 60 seconds, we adjust the prefetched segment duration to match the viewing history of the user. We chose these parameters based on the typical length of video segments (e.g., a 10 seconds video segment with a bitrate of 1200 Kbit/s is about 0.5 - 1.5 MB in size ) and experimental results from Section 4.6.3.

### 4.4.3 Segment Request Scheduling

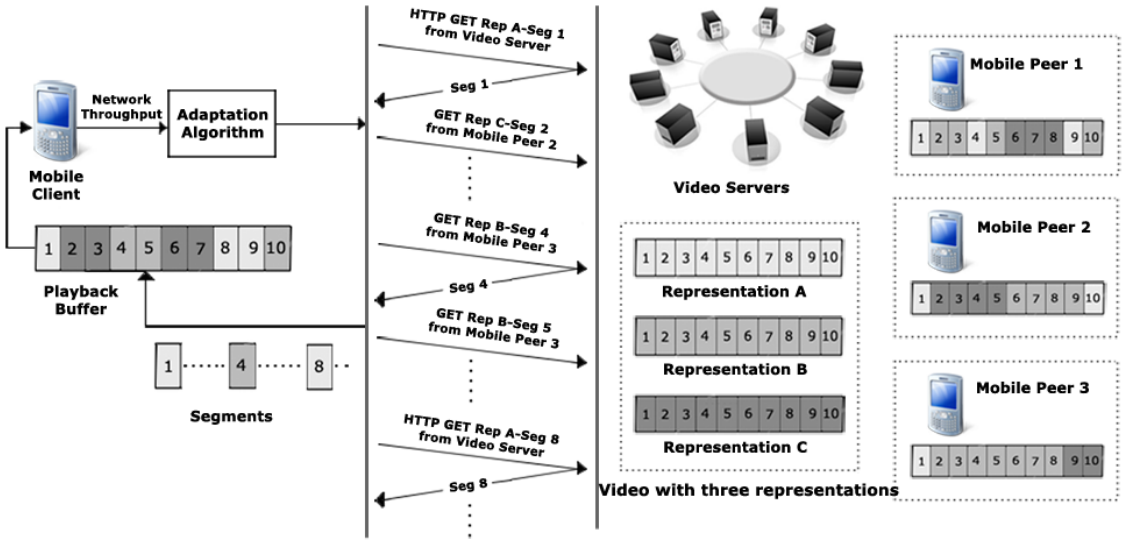



**Figure 30: Segment request scheduling. SDASH sends segment requests to either the video servers or the mobile peers and adjusts subsequent requests to match the network conditions and responses from the segment sources.**

The video server and the set of nodes in the sharer network with cached segments which are available for device-to-device connections constitutes the *candidate sources* from where a given segment may be downloaded (Figure 30). Using estimations of the network conditions, the segment scheduler downloads the segments from the candidate sources. In this subsection, we describe the components of the SDASH architecture that deal with segment quality adaptation and source selection.

**Segment quality adaptation**: The segment quality adaptation algorithm determines the bitrate for the next segment to be downloaded should be given the network conditions. It estimates the network throughput by dividing the segment size by the segment fetch

**Input**:

$\beta(t)$, /* buffer level at time t */

$b_{s(t)}$, /* smoothed bw estimate */

$\eta_{steady}$ /* is steady state? */

$\sigma_{(t)}$ /* recent segments with low quality */

**Output**:

$b_{r(t)+1}$, /* bitrate of next segment */

$B_{r(t)+1}$ /* next request time */

*changesource* /* change next request source */

```
1  invoke MASS Adaptation (Figure 18) ;
2  if b_{s(t)} ≤ b_{r(avg)} then
3  |  σ_(t) = σ_(t) + 1 ;
4  |  if σ_(t) > θ then
5  |  |  changesource := true ;
6  |  end
7  else
8  |  σ_(t) = 0 ;
9  |  changesource := false ;
10 end
```

**Figure 31: Algorithm: SDASH Quality Adaptation.**

time. The estimated throughput is then smoothed over a time window (using a harmonic mean) and quantized to a reference bitrate. Based on the reference bitrate, the quality adaptation algorithm decides whether to switch-up to a higher bitrate or switch-down to a lower bitrate.

The segment quality adaptation algorithm extends the MASS adaptation scheme (Figure 31). Our scheme takes one additional input parameter: $\sigma_{(t)}$, the number of recent segments with below-average quality and returns an additional output parameter *changesource* which tells the scheme to probe the P2P network for a peer with better quality.

The quality adaptation scheme operates as follows. It begins by initially checking the device cache for segments of the current video. It switches to fetching from the server once all the cached segments have been used up. If the average bitrate over the last $k = 10$ previous chunks is less than the average baseline, it sets the parameter *changesource* to true. It then invokes the source selection algorithm which determines the optimal candidate

source for fetching the next set of segments. Based on results from Section 4.6.2, we use an average baseline of 2 Mbps.

**Source selection**: The source selection scheme selects a node $s_c$ from the list of candidate sources $S = s_{server}, s_{p1}, s_{p2}, ..., s_{pk}$. It uses two key metrics: (i) $R_c$, the average response time of the source for segment requests made over the last $k = 50$ previous chunks, and (ii) $T_c$, the average throughput of the source for segment requests made over the last $k = 50$ previous chunks. At initialization, the scheme selects a source $s_c$ at random and sends the next request to the source. Subsequent requests are sent to the chosen source until one of two conditions occurs (i) the buffer level at time $t$ is less than a minimum value $B_{min}$ of 10 seconds, or (ii) no response was received for the last request after the segment timeout. The segment timeout is defined as 70% of the segment duration. If either of these two conditions is met, the algorithm computes the values $R_c$ and $T_c$ for the candidate sources and selects the peer with the minimum average response time and the maximum average throughput to direct the next segment request.

## *4.5 Prototype Implementation*

We implement the SDASH system encompassing the cooperative streaming mechanisms described in Section 4.4 as a middleware on top of Android 4.2 platform. The SDASH architecture is general and can work with any social network. Our current implementation uses Facebook. We use the Facebook Android SDK [54] and Facebook Graph API [55] for interacting with the Facebook social network and retrieving the user's friends as well as for checking newly posted videos and publishing newly shared videos to the user's "news feed". As we mentioned in Section 4.3.3, we use a tracker server to manage indexing of shared segment files and to coordinate connectivity and membership changes in the sharer network. We also use a tiny HTTP server (NanoHTTPd [16]) on each mobile peer for serving the requested segments and UPnP [126] for port mapping and enabling remote access across routers.

The SDASH prototype (see Figure 28) also provides the user with a graphical user interface (GUI) for specifying user preferences and interacting with the social network. The

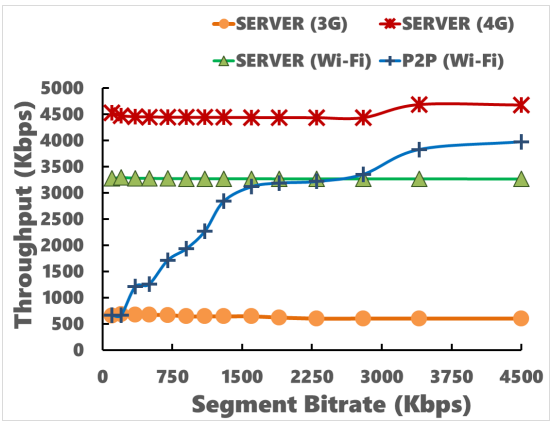
user can specify whether to turn off prefetching, at which minimum battery-level to start prefetching, the user's mobile data limit, and at what percentage of mobile data usage to turn off SDASH. The GUI also includes menus for logging into the user's social network and selecting the lists of friends that are allowed to participate in cooperative streaming.

## *4.6 Evaluation*

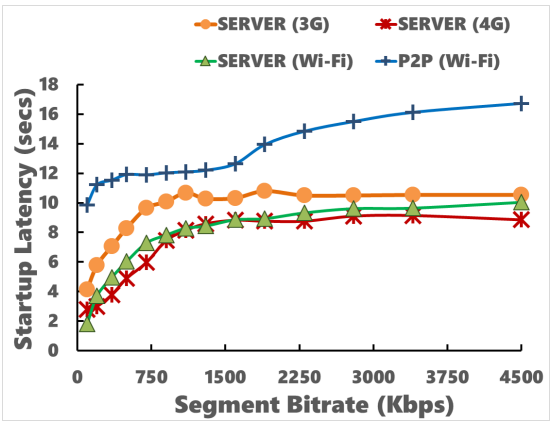
Our evaluation of SDASH addresses the following issues: (a) **Baseline performance**: What is the baseline performance when fetching segments from different network access types (i.e., 3G, 4G, Wi-Fi, and P2P)? (b) **Prefetching and Caching**: How does the availability of prefetched and locally cached segments affect video quality? (c) **Adaptive Streaming**: How well does peer-assisted streaming adapt to changing network conditions? Does SDASH lead to improvements over server-based network access?

### 4.6.1 Experimental Testbed

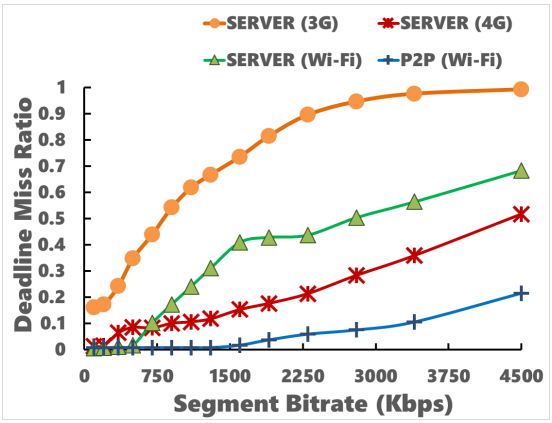
Our experimental testbed consists of two components: a desktop PC and eight smartphone devices. The desktop PC serves as the AHS video hosting server and the tracker server. The desktop PC uses a wired Internet connection. A newly joined peer contacts the tracker to retrieve connection information about members of its sharer network. Each peer also synchronizes data (connection information, indexes of the shared video files and cached video segments descriptions, and location information) with the tracker. We use eight Android phones: five Google Nexus IV smartphones equipped with Wi-Fi and 4G network radios, and three Google Nexus S devices equipped with Wi-Fi and 3G network radios. The 4G phones obtain Internet connectivity via T-Mobile 4G LTE network while the 3G phones obtain Internet connectivity via T-Mobile 3G network. In the experiments, the phones download segments of the 5 minutes video dataset presented in Chapter 2 (Section 2.2.1) from the remote server: segments of Big Buck Bunny video [5] encoded using x264 encoder at 14 different bitrates (100, 200, 350, 500, 700, 900, 1100, 1300, 1600, 1900, 2300, 2800, 3400, and 4500 kbit/s) with a segment duration of 2 seconds and frame rate of 48 fps.



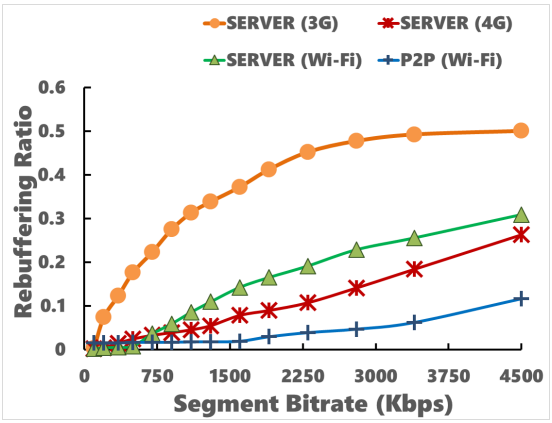

(a) Throughput



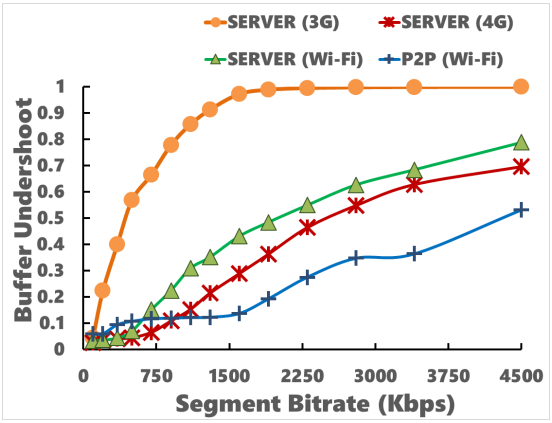

(b) Startup Latency



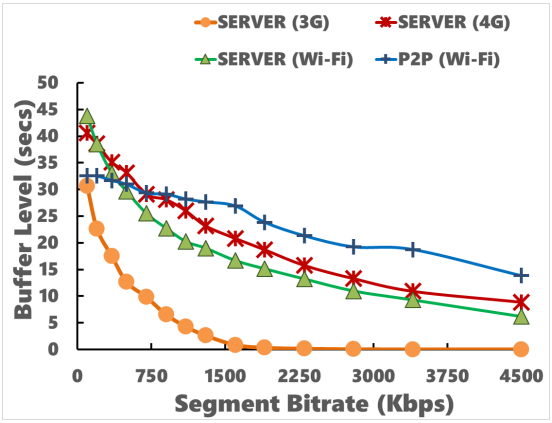

(c) Deadline Miss Ratio




(d) Rebuffering Ratio




(e) Buffer Undershoot




(f) Buffer Level

**Figure 32: Performance of different connection access types with fixed segment representation when varying the requested quality from 100 Kbps to 4500 Kbps.**

### 4.6.2 Performance with Fixed Segment Representation

In this section, we evaluate the performance of SDASH when the segment quality is fixed throughout the length of the playback. Using the connection access types of Server (3G), Server (4G), Server (Wi-Fi), and P2P (Wi-Fi), we vary the segment quality from 100 Kbps to 4500 Kbps and average the results over 80 runs. The results are shown in Figure 32.

Figure 32a shows the throughput values for different connection access type. While the average throughput for the server-based connection access type is almost a constant, the throughput for the P2P connection access type increases progressively with the video quality level. At lower quality levels, the processing cost incurred by a peer for serving the video (using NanoHTTPd) dominates the overall time required for transferring the video segment. With higher bitrate videos, the processing overhead is amortized over larger-sized segment transfers thus resulting in better performance at higher video quality levels.

Figure 32b shows the startup latency for fetching segments through each of the connection access types. The startup latency is measured as the time between the initialization of the video url to completing the download of the first 5 segments. As shown in the figure, P2P incurs much higher startup latency than the server-based access types. This is because P2P requires initial setup time to discover and connect to peers (an average of 8.42 seconds). The initial setup time includes the time to retrieve the lists of available peers and descriptions of the video segments that the peers currently have in their local caches from the tracker server. It also includes the time to establish connection to one of the available peers and receive permission to download segments from the peer. Because of the higher startup latency incurred by P2P, SDASH utilizes the video server as the default source for fetching segments and only invokes its source selection algorithm after the first set of segments have been downloaded.

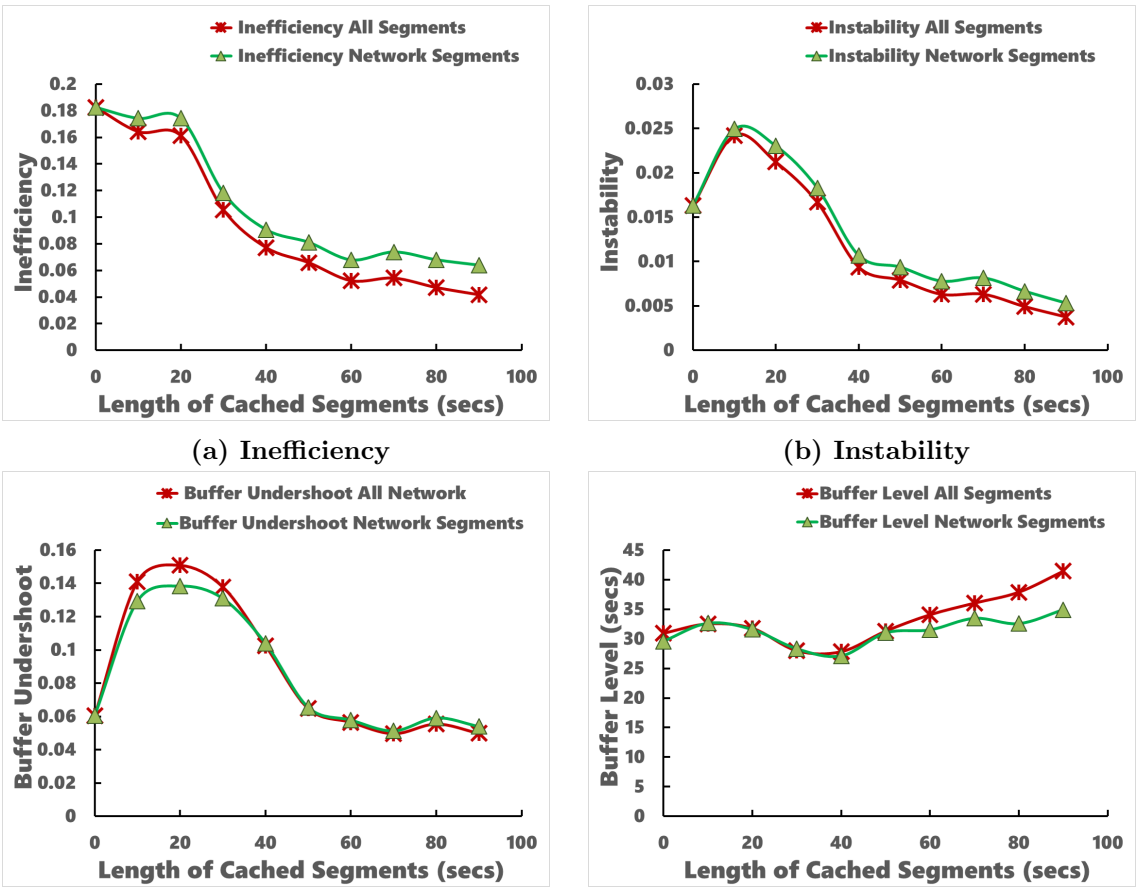
In Figure 32c and Figure 32d, we show the average number of download deadline misses and rebufferings experienced for the different network access types. The results show that streaming via P2P outperforms all the other access types by experiencing very little rebuffering (less than 1% for most of the quality levels). In contrast, the 3G network access incurs significant rebufferings with increasing video quality levels.

Figure 32e and Figure 32f shows the buffer undershoot metric and the average buffer level for each connection type. As expected, streaming with 3G suffers significant buffer undershoot as the quality level is increased, yielding a buffer size close to zero when the segment bitrate is fixed above 1300 Kbps. The buffer undershoot metric is similar for P2P in comparison to 4G and Wi-Fi at low to mid quality levels (100 Kbps to 1300 Kbps); however, its performance trumps the other connection types at higher video quality levels (1300 Kbps to 4500 Kbps).

In summary, the results show that while streaming from nearby peers incurs a higher startup latency, it however outperforms other connection access types on all the remaining optimization metrics. We note that in order to minimize the startup latency, SDASH begins streaming by fetching segments from the server while the P2P overlay network is established in parallel by a background thread. It then invokes its source selection algorithm after it has successfully connected to one or more peers which can serve as alternative sources for fetching the desired video segments.

### 4.6.3 Impact of Prefetching and Caching

In this section, we quantify the performance improvement of video streaming as the length of prefetched segments is progressively increased from 0 to 90 seconds. In the experimentation, we use the basic SDASH adaptation over Wi-Fi network access with the P2P component disabled. We also prefetched segments of the video using the highest quality level of 4500 Kbps. The prefetched segments are then saved on the local SD Card of all the smartphones. In the experiments, we begin by fetching segments from the cache and then switch to the network once the last cached segment has been retrieved. We then measure the overall performance across the segments regardless of where the segments are downloaded (labeled *all segments* in Figure 33) and the performance across only the segments that were retrieved from only the server (labeled *network segments* in Figure 33). We average the results over 50 runs. The results are shown in Figure 33.




(a) Inefficiency

(b) Instability

(c) Buffer Undershoot

(d) Buffer Undershoot

**Figure 33: Impact of Prefetching and Caching in SDASH. Overall, the performance as measured by these metrics improves as the length of the cached segments is increased.**

The results show significant performance improvement [1] for the inefficiency metric (Figure 33a) and the buffer level metric (Figure 33d) as the cached size is progressively increased. For the instability metric (Figure 33b), due to the sharp transition from high bitrate from the cache to low bitrate from the network, there is an initial decrease in performance when less than 15 seconds of the fetched segments were locally cached. However, the performance as measured by the instability metric improves as more segments are fetched from the cache. Similarly, the buffer undershoot metric sees a reduction in performance as the length of the cached segments is increased from 0 to 50 seconds, but increases in performance by up to

[1] Note: smaller is better for these performance metrics.

50% as the length of the cached segments is increased from 50 to 90 seconds (Figure 33c ). The results of the buffer undershoot is also slightly reflected in the size of the buffer level, which sees a small dip when between 10 and 50 seconds of the downloaded segments are fetched from the cache. However, the buffer level results also show that the overall effect of the worsening in buffer undershoot is minimal since it has only a moderate effect on the average buffer level. We note that the increase in instability and buffer undershoot when the length of cached segments is between 0 and 40 seconds can be minimized by intelligently spreading out the cached segments across the length of playback instead of caching only the initial set of segments. Comparing the results and performance tradeoffs, we choose a default caching size of 60 seconds.

### 4.6.4 Performance with Quality Adaptation

In this section, we quantity the performance of SDASH when segments are fetched from a variety of available sources and network connection access types: remote video server with 3G connectivity, remote video server with 4G connectivity, remove video server with Wi-Fi connectivity, and when using a combination of P2P connectivity and Wi-Fi access to the server (i.e., peer-assisted streaming). For P2P, we pre-cached each of the mobile peer with the highest quality representation (4500 Kbps). We average the results over 100 runs. The results are shown in Figure 34.

Figure 34a compares the inefficiency of SDASH (P2P + Server) with other forms of connection access. As expected, the inefficiency of the 3G, 4G, and Wi-Fi access reflects the values we presented earlier for the average throughput (Table 5). The result shows that SDASH outperforms all the other schemes: it outperforms Server-Wi-Fi connection access by 4%, Server-4G by 43% and by Server-3G by 54.5%.

In Figure 34b, we compare the instability metric for all the access types. The results show that using the peer-assisted approach provides more robust performance than the other schemes, increasing the stability by a factor of 0.35 to 7.8. The 3G connection access type exhibits the least stability, with instability values of more than double the that with 4G.

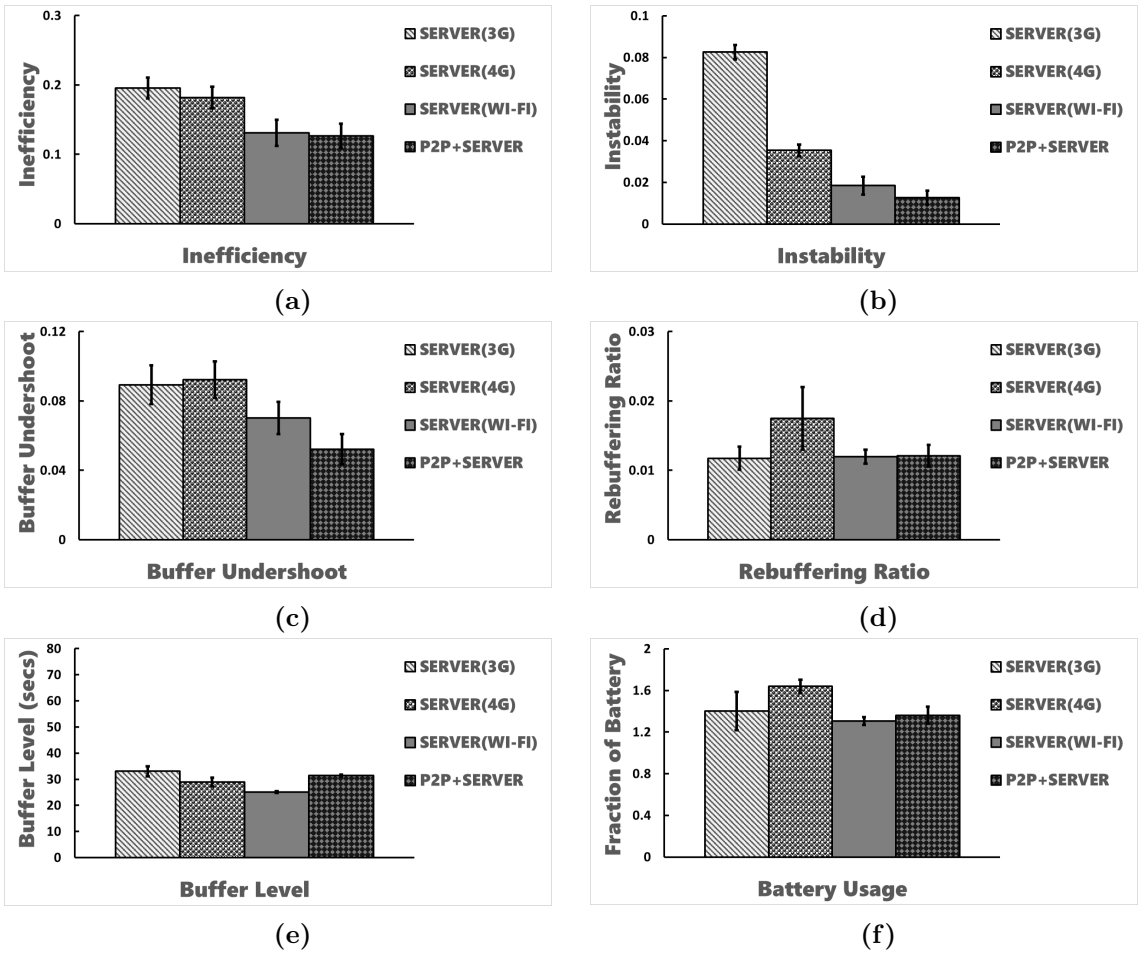



**Figure 34: Performance of P2P + Wi-Fi Server as used in SDASH versus other forms of mobile connection access. P2P+Wi-Fi Server combo outperforms all the other connection access types on inefficiency, instability, buffer undershoot metrics. It is comparable to the best in rebuffering and buffer level metrics. It is better than 3G and 4G on battery depletion metric, and slightly worse than Wi-Fi Server alone.**

Similarly, SDASH with peer-assisted streaming outperforms the other access types on the buffer undershoot (Figure 34c) metric: P2P + Server outperforms Server-Wi-Fi by 34%, Server-3G by 71%, and Server-4G by 77% on buffer undershoot. At the same time, its average buffer level is above the target size of 30 seconds (Figure 34e). It also outperforms 4G on the rebuffering ratio metric (Figure 34d) but its performance is slightly below that of Wi-Fi.

The battery depletion due to SDASH (P2P + Server) is shown in Figure 34f. We measure

the battery depletion by recording the remaining percentage of battery as reported by the Android system every 1 second. We filter out the results for when the smartphone is being concurrently charged and results for when the remaining battery level is less than 50%. We then compute a simple average of the depletion over the lifetime of our experimentation. The results show that despite the extra costs of setting up and taking part in a P2P overlay network, SDASH's battery depletion is comparable to the performance of fetching segments via 3G or Wi-Fi.

In summary, fetching segments with peer-assisted streaming (P2P + Server) outperforms all the remaining approaches on all of the metrics except for the battery depletion– it outperforms 3G and 4G on battery depletion but consumes slightly more battery than Wi-Fi. It also outperforms all of the other approaches on the instability by as much as a factor of 0.35 (Wi-Fi Server) and a factor of 7 (3G).

## *4.7 Related Work*

Our work builds on prior work in adaptive HTTP streaming, hybrid P2P/CDN streaming, social P2P video streaming, and cooperative mobile platforms.

**Adaptive HTTP Streaming Systems:** Variants of Adaptive HTTP Streaming been proposed by major commercial media providers. These products include Microsoft's *Smooth Streaming*, Apple's *HTTP Live Streaming*, and Adobe's *HTTP Dynamic Streaming*. A standardization process sponsored by the Motion Pictures Experts Group (MPEG) resulted in a new ISO/IEC 23009-1:2012 standard called Dynamic Adaptive Streaming over HTTP, or DASH [119]. Following these developments, a new wave of research have studied different types of video segmenting, and adaptation techniques for AHS [91, 75, 89]. However, we are not aware of any systems that considers the problem of improving the Quality of Experience for mobile clients by exploiting social sharing.

**Hybrid P2P/CDN Streaming Systems:** Many researchers have studied the use of peer-to-peer networks for providing scalable and cost-effective media streaming [138, 73, 90, 94]. In these systems, new clients admitted into the P2P network automatically become members of a swarm for the video items they are streaming. Hybrid P2P/CDN systems

combining the scalability advantages of the P2P networking with the stability advantages of CDN systems have also been examined by several researchers [81, 134, 106, 83, 132, 76]. In these hybrid systems, new clients can request video segments from the server but may also join existing swarms depending on the system conditions. These systems are focused on leveraging P2P techniques for improving scalability of the video servers. In contrast, SDASH's hybrid P2P/CDN design is focused on performance improvements for mobile clients.

**Caching and Prefetching Systems:** Several projects have proposed techniques for cooperative caching in *ad hoc* and wireless networks [135, 102, 139, 34, 139, 102]. A key problem studied by these systems is the optimization techniques for resolving the tension between increasing the data accessibility and reducing the caching overhead [67, 130, 48]. Similarly, a number of previous studies has also studied prefetching techniques for mobile video [62, 129, 41]. While SDASH shares similarities with these systems, SDASH focuses on cooperative techniques for caching and prefetching of P2P video data originating from mobile social networks. SDASH utilizes users' sharing history and friendship ties as well as the device context to determine what, when, and how much data to cache or prefetch in order to optimize the video viewing experience.

**Cooperative Mobile Platforms:** Mobile Social Networks (MSNs) [78], which are formed when mobile users encounter each other and communicate with each other using local wireless devices have been studied by the research community [61, 110]. These systems use P2P and WLAN technologies to implement features provided by existing location-based social applications. Recent works [109, 46] have also looked into crowd sourcing for cooperatively accomplishing a variety of community sensing tasks.

Similarly, research on *ad hoc* networking, opportunistic networking, and delay tolerant networking have proposed cooperative techniques for disseminating content among mobile clients [105, 86, 116]. They use Wi-Fi, Bluetooth, and other device-to-device connections to create forwarding paths between mobile devices. Prior works [74, 33] have studied the advantages of exploiting social ties for multi-hop content dissemination in such opportunistic networks.

In Microcast [77], the authors proposed a cooperative live video streaming platform for mobile devices which are concurrently connected to the same wireless network. Microcast uses the cellular links of the phones to download the files from the remote server and a wireless protocol to broadcast the video segments to the peers through the Wi-Fi links. Microcast also included network coding algorithms for optimizing the packet broadcast process. Other researchers [27, 49] have examined using multiple WLAN links often present on smartphones for achieving better connectivity, collaboratively downloading of content or offloading content from cellular links.

While there are several similarities between the work presented in this chapter and some of the previous research described above, SDASH distinguishes itself from these works in several ways. First, it extends the paradigm of content sharing on social networks to video streaming by enabling sharers of video to actively cooperate in streaming the shared content. Second, it uses P2P communication between sharers of video to augment AHS streaming for mobile clients. Third, it exploits the context of the peers in optimizing its quality adaptation algorithm. Finally, it implements techniques for mobile clients to control their degree of participation in the sharer network. To the best of our knowledge, SDASH is the first platform that is geared towards providing consistent quality of service among mobile sharers of videos.

## *4.8 Summary*

Variants of the Adaptive HTTP Streaming exploiting the CDN infrastructure are popular for video streaming on the Internet. With the proliferation of mobile devices and the popularity of video streaming to mobile devices it is natural to use AHS for streaming to mobile devices as well. AHS video streaming on mobile devices face additional challenges of high network variability, heterogeneous networking interfaces, multiple form-factors, and limited battery life.

In this chapter, we have proposed SDASH, an extension to Adaptive HTTP Streaming, which exploits the social networks that a client is a part of, to enhance the streaming quality for the client using P2P sharing techniques. SDASH incorporates cooperative caching,

prefetching, and peer-assisted streaming of segments among the clients in a social network. We have implemented SDASH on Android smartphones and have demonstrated through experimentation that the techniques we have implemented for enabling social sharing of videos results in a much better quality of video streaming than vanilla Adaptive HTTP Streaming. We have also shown that this performance boost comes at minimal additional battery depletion on the client devices. In our proof-of-concept system, we have used a centralized approach for sharing information about the availability of peers and the downloadable content at the peers. Our future work includes consideration of other more scalable ways of disseminating such information among the peers in the social network.

# CHAPTER V

# DYNAMICALLY FORMED SOCIAL NETWORKS FOR VIDEO SHARING

In the previous chapter, we leverage video sharing on online social networks for optimizing mobile video streaming experience. However, given the increasingly geographical localization of video streaming (Figure 3), there are multiple situations whereby connectivity to infrastructure-based services such as Facebook may not be available or desirable. In this chapter, we present a system infrastructure for enabling the formation of dynamically formed social networks of mobile devices that communicate via device-to-device links [136]. We consider the elements of managing such *transient social networks* defined by four attributes: *spatial locality, temporal locality, encounter-based community formation, and interest alignment*. By exploiting additional opportunities for sharing of user-generated video content through P2P communication techniques, we can further improve the Quality of Experience of mobile video streaming.

The rest of this chapter is organized as follows:

- Section 5.1 — We describe the application space for dynamically formed social networks at a high level. We also introduce *Micrograph*, our proposed middleware infrastructure for membership management in dynamically formed social networks, and review related work.

- Section 5.2 — We describe the system architecture of Micrograph and the structure of the dynamic social graph.

- Section 5.3 — Based on the model of the social graph described in Section 5.2, we describe the implementation of membership management protocols used by Micrograph in details, including the configuration messages exchanged between the nodes in the overlay network.

- Section 5.4 — We perform quantitative performance analysis of the overhead incurred by the Micrograph middleware. We also provide qualitative discussion of the experimental results.

- Section 5.5 — We conclude the chapter with a short summary.

## *5.1 Background and Related Work*

Social networks have attracted millions of users worldwide, empowering individuals to connect, share and interact with friends, family, and acquaintances across geographical boundaries. In existing online social networks, client devices serve as simple endpoints for publishing to and downloading contents from the cloud servers of the service providers. As the hunger for the generation and consumption of user-generated content scales up, this client-server model between the end-devices and cloud servers will limit scalability. Besides, this client-server model ignores the synergy that exist between members of a given social network, who are likely interested in sharing content among themselves. Location-based social networks and decentralized P2P social networks (such as Foursquare [10], and Diaspora [7]) are extending the traditional uses of social networks (such as Facebook [9] and Twitter [20]) to geo-community information sharing. However, such extensions to traditional social networks still assume membership of the social communities to be static or evolving very slowly over time and hence managed centrally. In the future, we expect much more dynamism in how social networks are formed and disbanded (e.g., first responders arriving at the scene of a natural disaster, attendees at a sporting event, et cetera). With the exponentially increasingly capabilities in the end devices, and the scaling up of the user community across the globe, there is an opportunity to *democratize* the creation and management of social networks that are currently offered only via wide-area Internet, as well as enable a plethora of new applications.

In this chapter, we introduce the elements of managing dynamically-formed or *transient social networks* that are defined by the following four attributes: *spatial locality, temporal locality, encounter-based community formation, and interest alignment.* The first three attributes, namely, the participants being in the same geographical locality, being aligned

in time for engaging in activities of mutual interest, and not having to know one another a priori, sets apart a transient social network from a traditional social network such as Facebook. Via their mobile devices, nodes in the same vicinity can form a peer-to-peer (P2P) overlay network for exchanging information and activities of mutual interest.

**Micrograph** is a middleware for supporting the management of community membership in dynamically-formed social networks. It provides services such as membership management, messaging, and state management in a uniform manner for applications. It gives intuitive control to an application for specifying device-level attributes (such as sensing requirements, energy considerations, and network bandwidth), as well as application-level attributes (such as interest in specific types of information and locality). Micrograph uses these attributes autonomically in forming and morphing the overlay social network (e.g., splitting and merging the social graphs when such attributes are violated or met, respectively). In doing this, it ensures that the member list of a community is consistent across the nodes in the community, i.e., it gives complete *transparency* to the participants as to the instantaneous membership of the social graph that a participant is currently participating in. Finally, a participant (i.e., the device associated with her) may be associated with multiple social groups simultaneously. Micrograph ensures complete *isolation* for each application that a given device may simultaneously be participating in at any given moment. In a nutshell, Micrograph provides novel mechanisms for community management to address the specific challenges faced by applications built for mobile social networks.

Micrograph shares similarities with a number of participatory and community sensing systems [80, 36] such as Micro-blogs [61], and PRISM [46], as well as systems for group formation and management [131, 113, 59]. In Micro-blogs, users upload blogs annotated with sensed information to a server. The blogs are tagged by location coordinates that the users upload to a server periodically. The platform can then process location-based queries by other users. Similarly, Mob [38], PeopleNet [99] and Floating Content [105] provide distributed querying to mobile devices by mapping queries to geographic locations. Similar to these systems, Micrograph is designed for spontaneous exchange of information such as video blogs of recent events among devices in the same vicinity. However, Micrograph

focuses on challenges of identity membership rather than simple information sharing.

A number of opportunistic and *ad hoc* networking platforms such as Spontaneous networking [57], Haggle [103], and MobiClique [108] also share similarities with the Micrograph. These systems differ from Micrograph in that they are data-centric publish/subscribe approaches that focus on interest forwarding. Micrograph introduces group management techniques that take the location, battery availability, registered interests, and other individual node requirements into consideration for maintaining, splitting, and merging groups of nodes in dynamic mobile social networks.

## *5.2 Design of Micrograph*

### 5.2.1 Design Principles

Micrograph has four major design principles: (1) to support a broad range of encounter-based transient social communities, (2) to provide *transparency* from third parties and centralized servers (i.e., activities within a community are only visible to members of the same community), (3) to provide *isolation* among the currently running applications on a user device (i.e., an application can only interact with other devices that match its interest sets, sensing requirements and resource availability), and (4) to provide good performance by exploiting wireless networks. These principles help to capture the unique properties and functionalities of dynamically-formed social network as stated in Section 5.1.

### 5.2.2 System Components

Figure 35 shows the software module that runs on each mobile device. Each node may have differing degrees of resource availability and network connectivity. For example, a TSN may augment the personal smart phones of the participants with resource-rich nodes that can form a *cloudlet* [115] and provide latency-sensitive computation support while being connected to cloud servers on the Internet.

We assume that each device has some facility for determining its current location, detecting changes in movement, measuring resource usage such as battery availability and interacting with sensors present on the device. These artifacts are determined with the help of the native facilities provided by the operating system running on the device (shown as

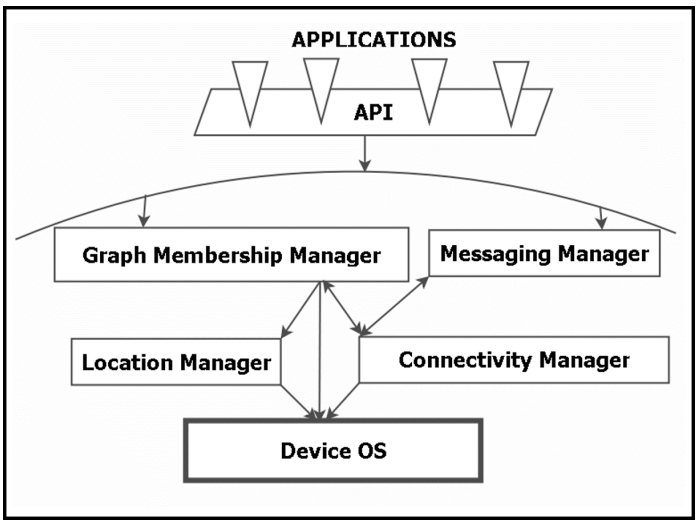



**Figure 35: Architecture of Micrograph**

*device OS* in Figure 35).

The *location manager* interacts with the device OS to determine the device location using an available localization technology on the device such as GPS localization or wireless network localization.

The *graph membership manager* running on each device manages community membership functionality for each applications in the social graph. Depending on an applications interest set, sensing needs, and current locality, the graph membership manager informs the application of the dynamic joining and leaving of other participants that meet those requirements.

The *messaging manager* is responsible for coordinating the exchange of application-specific information between members of an existing graph. It allows an application to send and receive messages to individual members or a subset of its membership list. The data connection between peers is managed by the connectivity manager. Micrograph supports a number of connectivity configurations including: (a) an *ad hoc* Wi-Fi or Bluetooth network, (b) a network of mobile devices connected to a municipal Wi-Fi network, nearby Wi-Fi hotspots or private access points (e.g., a campus or apartment community Wi-Fi network), (c) data connections through the cellular networks, and (d) a hybrid network

**Table 8: Micrograph API**

| **Community Definition** |
|---|
| interestId = createInterest(name); |
| commId = createCommunity(coords, duration); |
| addInterest(commId, interestId); |
| removeInterest(commId, interestId); |
| **Member Set Manipulation** |
| event addMember(memberId); |
| event removeMember(memberId); |
| boolean available = isMember(memberId); |
| location = queryMemberLocation(memberId); |
| interestSet = queryMemberInterests(memberId); |
| updateCurrentLocation(location); |
| **Messaging** |
| boolean sendToMember(msg); |
| boolean sendToMemberSet(msg); |
| msg = receiveFromMember(); |
| msg = receiveFromMemberSet(); |

that opportunistically exploits each of these possible connectivity alternatives. There are several works [31, 68, 137] that have examined how to leverage the combinations of the data connectivity available to a device in an opportunistic manner.

Table 8 summarizes the Micrograph API. The API is divided into three categories: community definition, member set manipulation, and messaging. To manage the dynamism in the social graph, Micrograph allows applications to register for specific events (*addMember* and *removeMember* methods) when a member node that matches its community requirements joins or leaves the network; as well as primitives for querying information about other member nodes such as location and interest set.

### 5.2.3 Design Assumptions

For the nodes in the Micrograph, we make a number of intuitive assumptions. First, we assume random mobility of the nodes. We also assume that each node can discover other nodes connected to the same physical network and can send data to and receive data from those nodes using mechanisms available natively on the node. These mechanisms would be implemented differently depending on whether the physical network is a wireless local area network or a cellular network. For example, on wireless local area networks techniques that

would be appropriate to use include those presented by Chandra, et al. [39], or Vasudevan, et al. [127] for nodes to discover one-hop or multi-hop neighbors connected via *ad hoc* routing or connected to neighboring access points. On cellular networks, we would leverage geographically-aware peer-to-peer overlays such as Globase.KOM [79] and GeoPeer [29] for maintaining data connectivity between peer nodes. Messages transmitted by each node can then be bounded by limiting them to nodes within the participants current geographical locality.

While Micrograph can be adapted to meet the specifics and exploit the features of the underlying physical network, in the rest of this section, we focus on Micrograph design for a devices on a wireless local area network.

### 5.2.4 Social Graph Model

In this section, we describe the representation of the social graph by the graph membership manager.

We model the social graph as a multi-level graph consisting of three levels: *data connectivity network (DCN)*, *feasible overlay graph (FOG)*, and the *transient social network (TSN)* (Figure 36). As the name suggests, the DCN includes all the nodes that are capable of communicating with one another via the wireless network (either directly or through intermediary nodes that serve as relays). Further, DCN captures the temporal locality that we alluded to in Section 5.1, namely, all the nodes in DCN are available at the same time for participating in the formation of a TSN. The FOG on the other hand, represents the set of nodes that can feasibly be part of a TSN given application-independent constraints such as relative mobility and battery life. Thus, the nodes in the FOG are a subset of the nodes in the DCN (e.g., H is in DCN but not in FOG in Figure 36). Lastly, the TSN, is the applications view of the network for the social interactions among its participants, based on interests expressed by a participant. Micrograph facilitates the formation of the FOG using topology management protocols among the members of the DCN, thereby abstracting the heterogeneity of the physical network. Micrograph manages the continuous arrival and departure of new nodes into the FOG. Further, it manages the applications view by

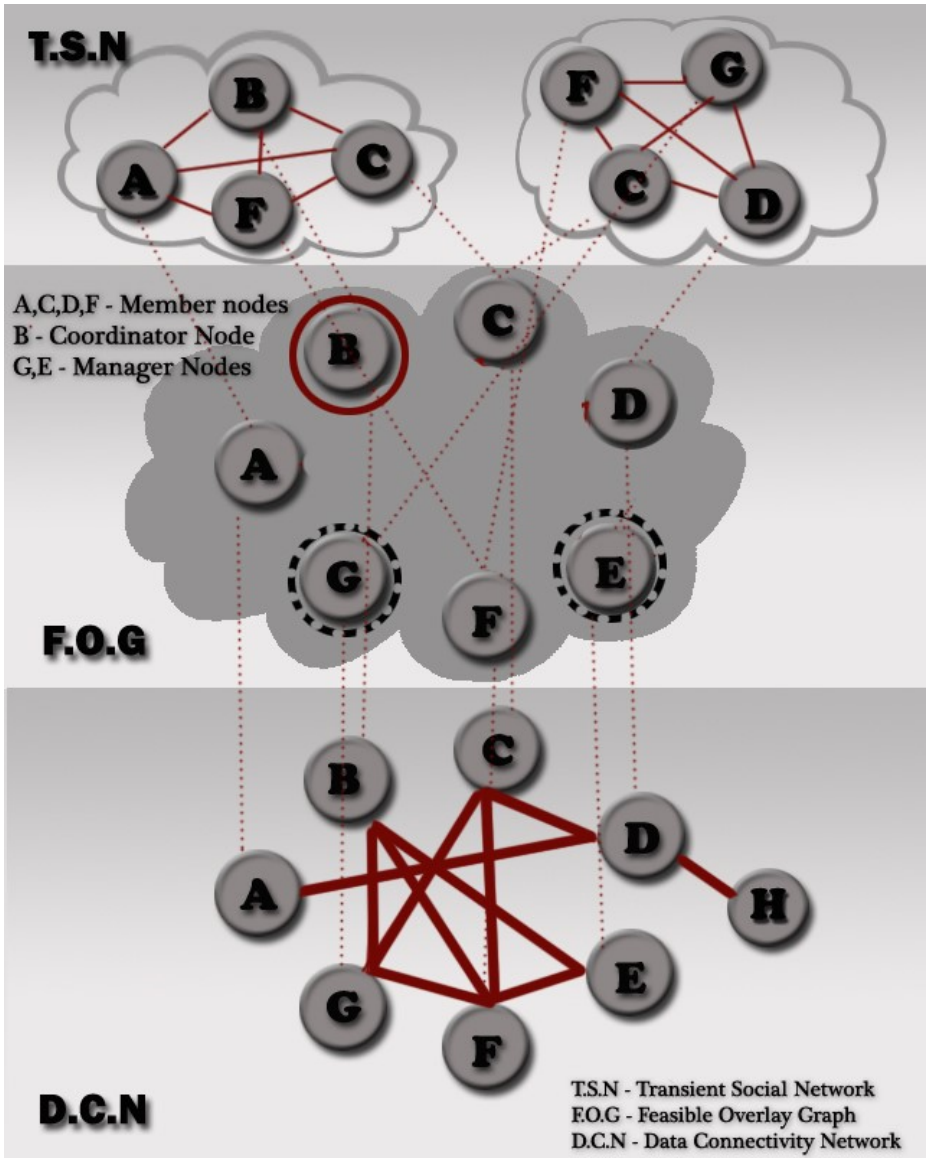



**Figure 36: Micrograph social graph model consisting of three levels: Data connectivity network (DCN), Feasible overlay graph (FOG), and distinct TSNs representing disparate communitues as overlays on top of the FOG**

overlaying user-specified constraints (e.g., interest set, privacy, et cetera), to form a TSN on top of the FOG commensurate with these constraints.

In order to build a scalable TSN, a node in an FOG may assume one of three roles: an *ordinary member*, a *coordinator*, or a *manager*. A node that assumes the role of a coordinator (node B in Figure 36) is overall responsible for the maintenance and evolution of the associated FOG. It is responsible for enforcing constraints on graph membership,

inter-graph communication between disjoint graphs, coordinating the merging of two FOGs whenever necessary, and the transfer of nodes from one FOG to another. As new nodes join the FOG and the graph membership increases, the coordinator node may assign one or more manager nodes (G and E in Figure 36) to assist in graph management operations. Each manager node is responsible for maintaining the state information for a disjoint subset of the nodes in the FOG, which we refer to as $\text{FOG}_{sub}$. The node that plays the role of the coordinator is additionally responsible for coordinating the exchange of state information among the manager nodes.

The FOG membership application-independent constraints are meant to ensure effective communication among the nodes in an FOG despite the dynamism in the network due to node mobility, health of a node, etc. Examples of such constraints include:

1. **Safe Distance:** an upper bound on the physical distance between any two nodes.
2. **Graph Size:** maximum membership for an FOG.
3. **Node Stability:** an upper bound on the number of join/leave by a node that is tolerable, which serves as a measure of the node mobility.

Currently, based on prior work on the size and capacity of wireless *ad hoc* networks [87, 51]; we set the maximum size of an FCG to be 600, and the safe distance to be 500 meters. Further, for scalability of maintaining the TSN evolution, we use one manager node for every 25 members in the FCG. The functionality of the coordinator and manager nodes will become clearer in the next section where we describe the community management of the TSN.

### *5.3 Membership Management Protocols*

In this section, we discuss novel protocols in Micrograph for membership management. The goal of Micrograph's membership management protocols is to present to the applications an accurate view of the nodes that are actually available for participating in a social activity given the vagaries of the network and the mobility of the participants. To achieve this goal, Micrograph may configure and reconfigure an FOG in response to events such as (a) the

joining of a new node, (b) the departure of an existing node, (c) the selection of a new graph coordinator, and the (d) merging of two FOG instances.

In the discussion that follows, we explain how Micrograph performs some of the key configuration activities.

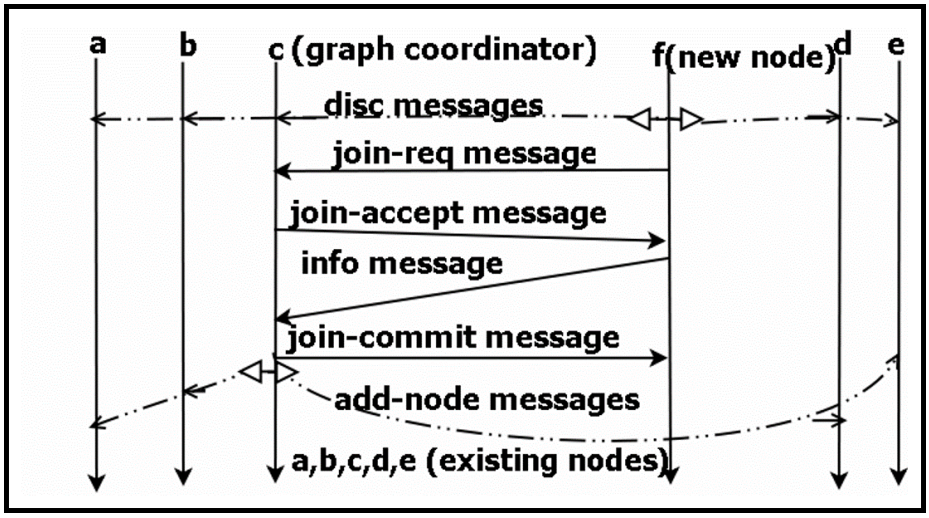



**Figure 37: Message exchange during FOG formation. The dotted lines are broadcast messages while the solid lines are point-to-point messages. The figure shows the messages exchanged between nodes of the graph as a new node (node f) initiates a join procedure.**

### 5.3.1 FOG Admission Protocol

On arrival, a node initiates a protocol to join an existing FOG (Figure 37). At least two nodes are needed for the initialization process to succeed. The arriving node (f in Figure 37) constructs a *disc* message (a discover message) which it broadcasts to potential peers. It also listens for a special *coord* message that identifies the coordinator node (c in Figure 37). The *coord* message includes functional attributes of the FOG that it is associated with and a unique graph id.

On receiving a *coord* message, f sends a *join-req* message to the coordinator node to become a member of the FOG. The *join-req* message contains the nodes battery level, its location, and its average speed. If the node meets the admission criteria for the FOG, the

coordinator replies with a *join-accept* message. If no *coord* message is detected after some time t, the node itself constructs a *coord* message and broadcasts it periodically, thereby becoming the coordinator node for a newly formed (singleton node) FOG. A node that thusly assumes the coordinator role for a singleton FOG may get a *coord* message from a peer who has simultaneously assumed the same role. In this case, a simple procedure is followed for breaking the tie: each node waits for a random amount of time before it re-broadcasts its *coord* message; if in the meanwhile it receives a *coord* message from the other peer then it simply accepts the other peer as the coordinator and sends it a join message and also forwards all the *join* messages it has already received or subsequently receives to the newly elected coordinator. The procedure avoids race condition by using time-bound estimations given in [113] and linear backoffs. Once a node has been accepted into an FOG, it sends an info message to the coordinator node that contains: its battery level, its location, its average speed, and its resource list (e.g., list of available sensors). The coordinator node in turn sends a *join-commit* message to the node which consists of the other members of the FOG and its assigned manager node. The coordinator node also sends an *add-node* message to manager nodes (or to the member nodes of the FOG if there are no manager nodes) with the details of the new node. Each manager node then forwards the *add-node* message to the member nodes that it is responsible for. Essentially, the coordinator node and the manager nodes act as the nameservers and sensor registries for the set of nodes in a FOG.

Each node in an existing FOG sends status updates to its assigned manager node (if it is a member node) or to the coordinator node (if it is a manager node) at regular time-intervals by constructing an alive message. The *alive* message contains information such as current node location, node velocity, and estimated battery life. We use the *alive* message to detect node failures/disconnections. On a node departure or disconnection, the $\text{FOG}_{sub}$ manager node assigned to the node broadcasts a *remove-node* message to its member nodes, the other manager nodes in the FOG and the coordinator node. Since each status update constitutes an overhead, we use a number of techniques to reduce this overhead. The update schemes are similar to those described in PRISM [46], adapted for the specifics of the Micrograph

architecture.

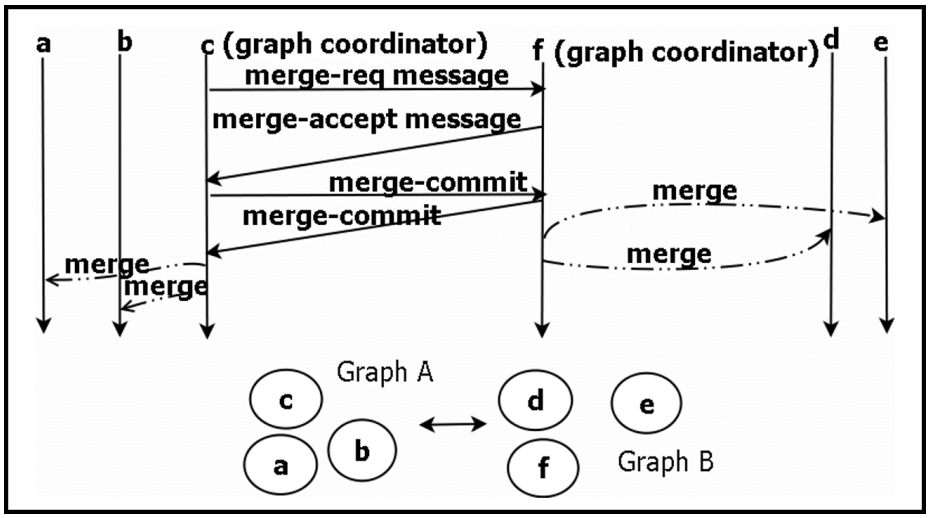



**Figure 38: FOG Merge Protocol. The figure shows the messages exchanged between Graph A (nodes a, b, c) and Graph B (d, e, f) during the merge operation.**

### 5.3.2 FCG Merge Protocol

Due to the mobility of the nodes in the network, there is a possibility two distinct FOGs previously in different geographical locality may move into the same locality. In such a scenario, the two FOGs may decide to merge into one FOG provided they have similar device-level constraints.

Such a situation will occur upon one or the other coordinator in the two graphs receiving a *coord* message from its peer or a *coord* message forwarded by one of the members in its own FOG, via the DCN. The coordinator in A (c in Figure 38) that receives this coord message from the coordinator in B (f in Figure 38), first checks if the addition of new members will not violate its graph constraints. If the constraints are met, c then sends a *merge-request* message to f. The *merge-request* contains the list of members of As Micrograph, and the location of each member. On receiving the *merge-request*, the coordinator in B (f in Figure 38) also checks to determine if its graph constraints are met. If the constraints are met, it sends a *merge-accept* message to c, else it sends a *merge-declined* message which aborts the

merge procedure. The *merge-accept* message contains information about the membership list of B. If c gets back a merge-accept message, it notifies its members by sending them Bs membership list and the coordinators id (f in Figure 38) as the new graph coordinator using a merge-commit message. C also sends a *merge-commit* message to B. If c receives any new updates from its members during the merge procedure, it simply forwards the updates to f. The procedure avoids race conditions by ignoring any new merge requests until an ongoing merge is completed. If there is a failure by any node during the merge procedure, the node simply re-initiates a join procedure.

### 5.3.3 Coordinator Selection Protocol

The coordinator selection procedure dynamically elects a new coordinator every 300 seconds. The protocol takes a number of factors into account: (a) physical distance of the node from the center of the graph (b) amount of battery life remaining at each node, (c) degree of relative mobility (in meters per second), and (d) the time elapsed since becoming a coordinator. The method used to choose a new coordinator is quite simple. The currently active coordinator selects the top five nodes in the FOG that have the (1) least relative distance to the center of the FOG, and (2), the least relative mobility. The node that has the most available battery level among these five nodes is chosen as the new coordinator. In the event of a tie based on battery levels, the node that has least recently served as the coordinator is chosen to be the next coordinator. The coordinator selects new managers using a similar heuristic as the size of the FOG increases. A newly admitted node is assigned a manager which is closest to the node by spatial distance.

### 5.3.4 TSN Membership Protocol

Now we will describe how Micrograph facilitates application-specific overlays on top of the FOG to form TSNs. A given node may be participating in multiple TSN applications simultaneously. Each such application has its own unique interaction with other nodes in the social graph which is largely determined by its quality of service requirements, sensing operations, and interest attributes. For example, one application may require the use of a compass and another application may require that its peers have both a compass and

a microphone before it can successfully interact with them. Each application instance is a distinct TSN projected on top of the same FOG (Figure 36). Further, the role played by a given node may be different in each of the different TSNs that it is simultaneously a part of. Micrograph helps in managing these multiple identities (or roles) for a given node. These roles can be registered as interest attributes with the Micrograph runtime using the Micrograph API (see Section 5.2.2). Each application informs its Micrograph runtime of these requirements and/or attributes, and is only informed of nodes that match these requirements.

Micrograph implements its attribute filtering mechanism on a per-application basis as follows. After a node has joined an existing FOG, it receives a neighbor list from the coordinator node. The newly joined node then broadcasts an *attr-insert* message that contains a list of its registered attributes to all the peers in the FOG. Each node that receives the incoming attributes then decides which of the attributes it is interested in based on registration by applications running on this node to guide its node filtering decision. Nodes that have matching attributes then send a response to the newly joined node by constructing an *attr-resp* message containing a list of their own attributes list to the newly joined node. The *attr-insert* is also re-broadcast whenever any attribute is added or removed. In this fashion, each node locally or independently determines which attributes to add, to cache or to respond to. This mechanism allows Micrograph to use localized policies to determine the nodes participation in a given social networking application.

## *5.4 Evaluation of Micrograph*

We implemented the Micrograph runtime in Java encompassing the community membership management protocols described in Section 5.3 as a middleware on top of Android 2.3 OS. For quantitative evaluation, we use a custom simulator designed for this project to evaluate the scalability and effectiveness of our membership management protocols. There are three experiments: (1) The first one measures the admission delay for a node to join an existing TSN; (2) The second one illustrates the ability of Micrograph to stabilize under node churns and readmissions. (3) The third experiment measures the overhead (in terms of additional

network traffic) for the existing nodes when a new node joins the TSN.

### 5.4.1 Simulation Model

The mobile nodes in our simulations implement the graph formation manager of our architecture (Figure 35). The network area is a 1000meters  1000meters topology, the transmission rate of each node is 54 Mbps and the default radio range is 250 meters. Each node moves within the network area according to the random waypoint mobility model [37]. The velocity of a node is evenly chosen from a uniform distribution of 1 meters/s minimum and 1.8 meters/s maximum, with a uniformly distributed pause time of 60s. Each member node sends status updates to its manager or coordinator node at a periodic rate of 500ms. We use a safe distance of 500 meters and a maximum $\text{FOG}_{sub}$ size of 100 for most of the simulation experiments.

### 5.4.2 Delay due to Node Admisssion

We define the admission delay as the time it takes for existing members of the FOG to become aware of a new node joining the FOG. The first experiment measure the admission delay for a new joining an existing FOG. Using a maximum $\text{FOG}_{sub}$ size of 100, we gradually increase the number of nodes in a given FOG from 1 to 1500. The results are shown in Figure 39. The figure shows the admission delay as a function of the graph size. The results show that the delay is low for all categories of nodes in an FOG. The coordinator node has the least admission delay compared to manager nodes and member nodes, because it is the first point of contact by a new node wanting to join the FOG.

### 5.4.3 Stability of Micrograph under Churn and Readmission

The second experiment evaluates the behavior of the FOG under sudden membership changes (node churns and node readmissions). We divide the duration of the experiment into 200 intervals; each interval is 30 seconds.

As shown in Figure 40, during the experiment, we disconnected 15% of the nodes simultaneously from the FOG between the 22nd interval and the 27th interval and then readmitted them between the 50th and 55th intervals; similarly, we disconnected 25% of

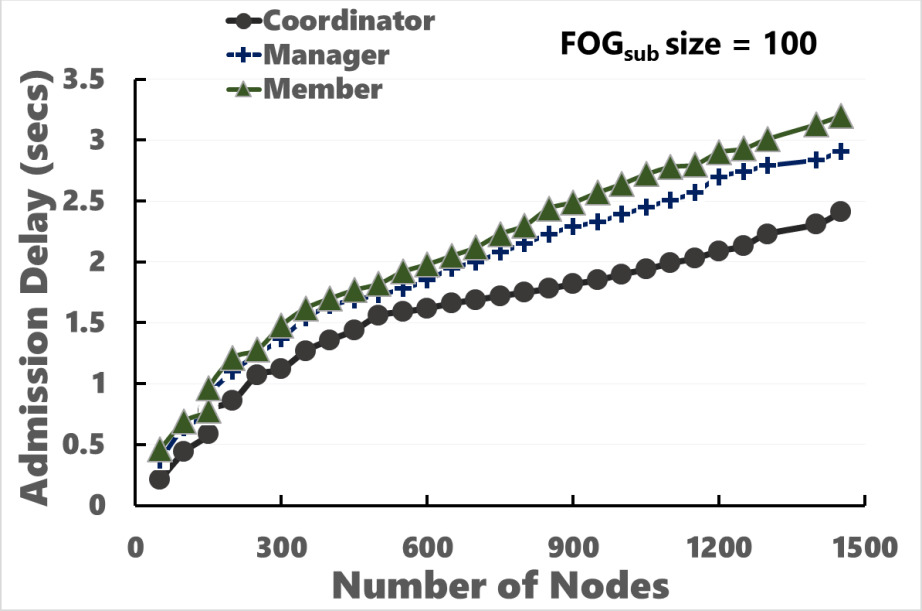



**Figure 39: FOG admission delay.**

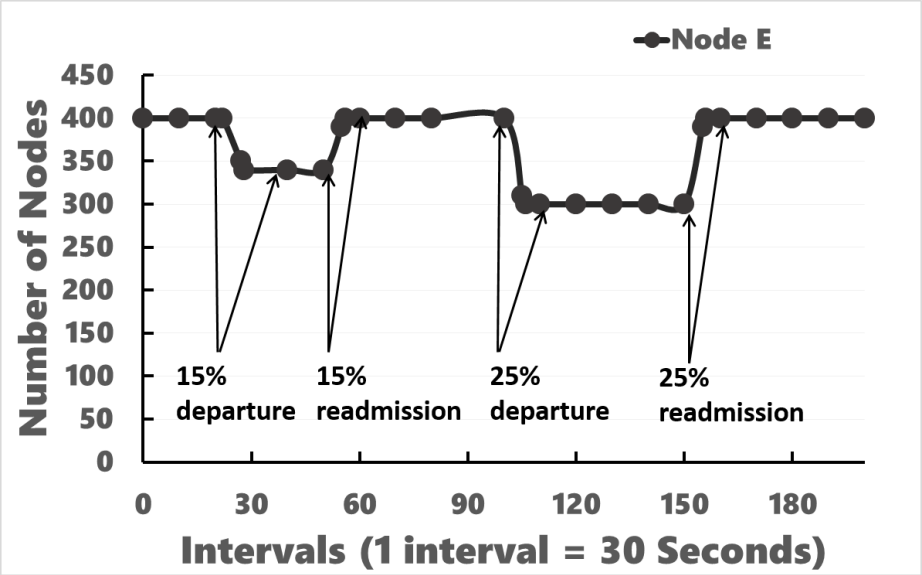



**Figure 40: FOG membership under sudden membership changes over 200 intervals.**

nodes between the 120th and the 125th intervals and readmitted them during the 150th and 155th intervals. We repeated the experiment 100 times. Figure 40 shows how the membership of the FOG evolves during these churns and readmissions. The membership stabilizes within 1 interval subsequent to the churn or readmission, and thus illustrates Micrographs

ability to react quickly to sudden sizable changes in node arrivals or departures from a given FOG.

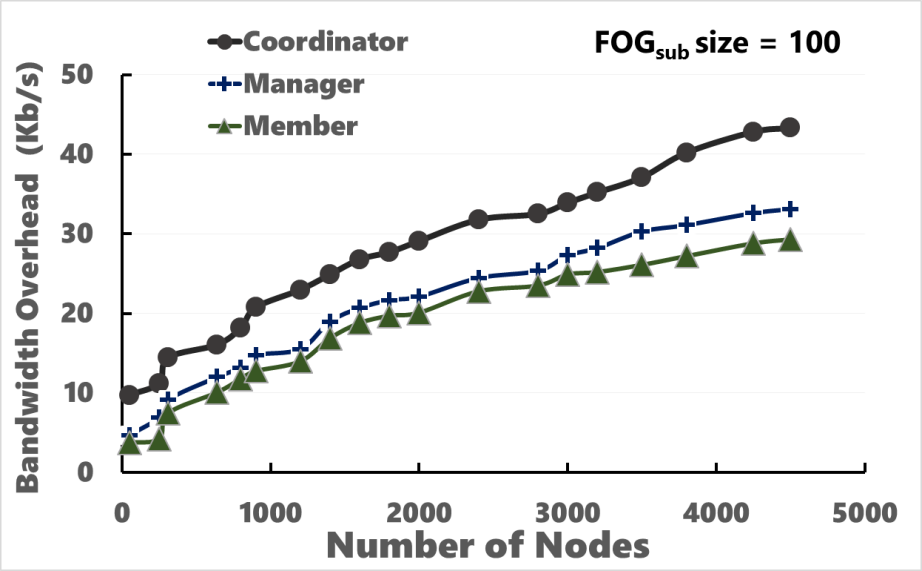



**Figure 41: Bandwidth overhead of an FOG as the size of FOG is progressively increased from 1 to 5000.**

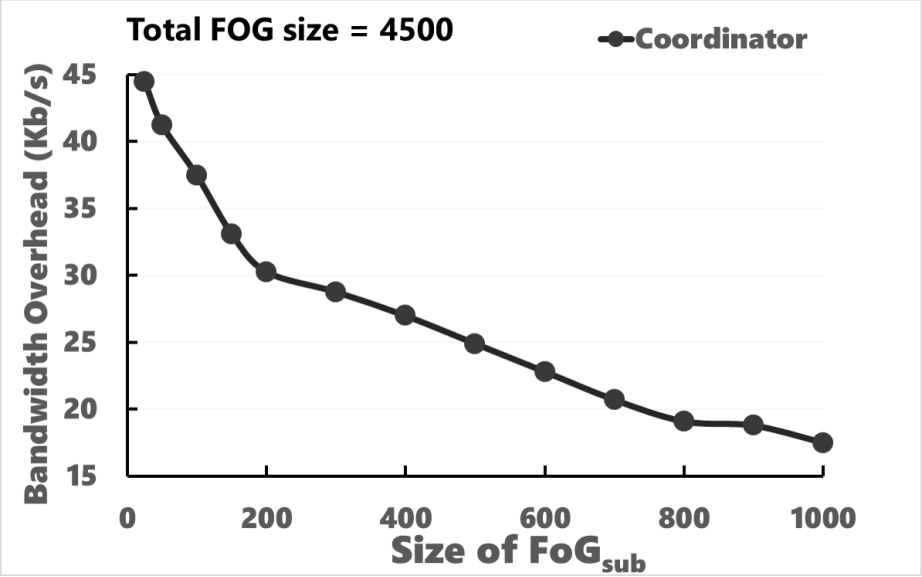



**Figure 42: Bandwidth overhead of a coordinator node as the size of FOG$_{sub}$ is progressively increased from 25 to 1000.**

### 5.4.4 Bandwidth Overhead due to Node Admission

We define bandwidth overhead for a node as the additional network traffic incurred by the node during node admission. The bandwidth overhead for a node in an existing FOG is a function of node admissions and node churns. We measure the average bandwidth overhead (both upstream and downstream) experienced by nodes in a given FOG, using a $\text{FOG}_{sub}$ size of 100.

The results in Figure 41 show that the bandwidth overhead is low for the different categories of nodes in a given FOG. The coordinator node has the largest overhead, since it receives membership updates from the manager nodes and admission requests from member nodes while the member nodes has the least overhead since it only communicates with its assigned manager node after becoming an FOG member. It is important to note that the overhead grows linearly with the size of the FOG implying that the implementation is scalable. The last experiment is measuring the bandwidth overhead at the coordinator node as a function of the $\text{FOG}_{sub}$ size. Figure 42 shows the bandwidth overhead at the coordinator node as the maximum size of the $\text{FOG}_{sub}$ is increased from 25 to 1000 nodes. The figure shows approximately linear decrease in the overhead, which is what one would expect to see.

### 5.4.5 Discussion of Results

The experimental results confirm the following: (1) Micrograph design ensures that the delay for node admission is linear in the number of nodes; (2) The design ensures that it can quickly stabilize following node churns and readmissions; and (3) the bandwidth overhead during node admissions in linear in the number of nodes.

## *5.5 Summary*

In this chapter, we have presented Micrograph, a middleware infrastructure for membership management in dynamically formed social networks. Micrograph manages the intricacies involved in community formation and facilitates device-to-device messaging among the peers in the social network. We describe the system architecture of Micrograph as well as the

protocols for joining, merging and splitting the social graph of devices forming the overlay network.

We have implemented Micrograph as a middleware on top of smartphones running Android system. As a proof of concept, we have built sample applications on top of the Micrograph middleware. We have quantitatively shown that Micrograph is a scalable platform for maintaining community membership information among nodes in a wireless network. As described in Chapter 4, we utilize our implementation of Micrograph for supporting on-demand video sharing.

# CHAPTER VI

# CONCLUSION & FUTURE WORK

The Internet traffic is being increasingly dominated by video with the share of video traffic predicted to rise to more than 90% of global network traffic by 2017. As users embrace Internet streaming of video, several studies have found that a small decrease in video quality leads to a substantial increase in viewer abandonment and disengagement rates. Although mobile devices are leading the deluge in video traffic, little work has been done to engineer and tailor the video delivery infrastructure to address the constraints facing mobile video streaming: difficult network conditions, heterogeneous networking interfaces, multiple form-factors, and limited battery life. In this dissertation, we propose a number of solutions for improving the quality of video streaming for mobile devices. We adopt a measurement-driven approach, in which existing and new techniques for video quality adaptation are evaluated using realistic bandwidth traces collected from field trials on commodity Android smartphones.

Current approaches for video quality adaptation follow a four-step model consisting of bandwidth estimation & smoothing, bitrate quantization, and request scheduling. A number of heuristics have been proposed for each step of the adaptation process for optimizing the overall streaming quality. Through careful experimentation and measurements, we analyze the performance of existing schemes for Adaptive HTTP Streaming on mobile devices, finding evidence of significant performance variability of the schemes when deployed in wide-area environments. We also find significant differences in their performance on fixed and controlled settings and noticeable differences in their performance on cellular and Wi-Fi networks. Based on these findings, we hypothesize on the possible causes of these inefficiencies. Based on insights from experimental results, we also identify the best strategies from among those used by the existing schemes.

Then, we propose two approaches to address issues with Adaptive HTTP Streaming

on mobile devices. First, we introduce MASS, a novel scheme for video quality adaptation for mobile devices that leverages the measurement-driven insights from our study of existing schemes to improve the video streaming experience. We build on the best-practices of existing schemes as well as introduce new techniques for tuning the robustness of the rate adaptation scheme. These techniques include limited multi-bitrate switching, dynamic quantizing safety margin, and fast ramp up. We then conducted experiments into performance tradeoffs for key quality metrics to evaluate the impact of multiple sensitivity parameters. Using the derived optimal configuration of the underlying components, we compared the performance of MASS with existing schemes. The results show that MASS outperforms existing schemes on key quality metrics.

Our second approach, SDASH, exploits the synergy between video sharing on social networks and video streaming on mobile devices for improving the performance of quality adaptation scheme. Cooperative techniques such as cooperative caching, video prefetching, and P2P streaming offer the potential of reducing the impact of unreliable network conditions on the video viewing experience by providing the mobile client with alternative sources where segments of the desired video can be retrieved. We investigate the potential benefits and costs of cooperative streaming by analyzing the average and maximum throughputs achievable from cooperative techniques. Then, we present the architecture of the SDASH system, which demonstrates the techniques for integrating cooperative streaming with vanilla Adaptive HTTP Streaming. We also describe the details of several algorithms used by SDASH to demonstrate our architecture in real scenarios. Through experimentation on Android smartphones, we show that streaming via SDASH provides much better quality than vanilla Adaptive HTTP Streaming. We also show that SDASH imposes minimal battery depletion on the client devices.

In addition, we introduce a middleware infrastructure for enabling decentralized video sharing among mobile devices via device-to-device wireless links. By exploiting additional opportunities for sharing of user-generated video content through P2P communication techniques, we can further improve the video streaming experience. Our middleware infrastructure, called Micrograph, supports the formation of *ad hoc* groups of devices based on

location, time, and shared interests. It manages the identities and membership of peers in the dynamically formed social network. We describe the system architecture of Micrograph as well as the protocols for joining, merging and splitting the social graph of devices in the overlay network. Experimental evaluation of Micrograph shows that it incurs negligible overhead for membership management of dynamically formed social networks.

There are many avenues for extending the research presented in this dissertation. First, in Chapter 4 (Section 4.3), we describe a system architecture for exploiting social sharing information for video prefetching and P2P streaming. In our current implementation, we use a centralized tracker for managing the metadata and overlay information about the shared videos and downloaded segments. In Chapter 5, we also present Micrograph, a middleware infrastructure for dynamic video sharing. As future work, we will consider in-network and decentralized techniques for communicating metadata and sharing information among the mobile peers. Examples of approaches that could be relevant include leveraging the structure of the social networks to determine where to cache video files (e.g., identifying central nodes), caching and prefetching videos based on request patterns of nearby peers, gossip and migration of data based on mobility patterns, and video viewing history, etc.

Similarly, there are a number of outstanding issues in P2P networking that will provide improved support for cooperative mobile streaming. Issues of identity management, user privacy, incentives management, NAT traversal, and decentralized networking has been studied by the research community in other contexts (e.g., file sharing, live streaming to fixed clients, and decentralized social networks) but have hitherto not been applied to adaptive mobile streaming. Because of the personal nature of mobile devices, addressing these issues is critical to unlocking the potential of cooperative streaming.

Another interesting research direction would be to explore techniques for optimizing the content creation and segmentation of the video files. The ratio of the uplink bandwidth to the downlink bandwidth is known to be asymmetrical in many cellular and mobile networks. Therefore, it is quite likely that the granularity of existing segment sizes generated by CDNs may be too large to be satisfied by a single peer in the social network. The limited uplink bandwidth would result in too many deadline misses at a given client even if a peer is

able and willing to serve the segment from its cache. However, multiple peers may have the same video content cached in their respective local storages. Thus, splitting a segment request into *sub-requests for partial chunks* that can be re-assembled at the requesting client into the whole segment will result in less stress on the peers and reduce deadline misses at the client. Such re-chunking techniques may become very essential when the source CDN servers are unavailable but several peer devices have watched/cached the video segments at earlier times (such as in mobile *ad hoc* networks, opportunistic networks, and Bluetooth personal area networks). There is considerable prior work (e.g., BitTorrent [6], emule [8], and COMBINE [27]) in file chunking and aggregation in the context of parallel downloads. These prior works will provide insights into techniques and protocols for such fine-grained segment scheduling and on-the-fly re-segmentation.

Finally, as new devices and platforms are developed for video consumption, it would be of great value to better understand how to architect the video delivery system to provide multi-device support. Consumers are increasingly embracing new technologies such as smartphones and tablets, hybrid smartphone-tablet or phablet devices, and networked televisions, set-top boxes and game consoles (e.g., Amazon Fire TV [3], Apple TV [4], Google TV [11], and Xbox Live [21]) that provide new opportunities for video sharing and experiences. Addressing issues of session migration, video offloading, device and screen sharing and multi-device storage management will allow efficient use of all resources available to an individual user.

# REFERENCES


[1] "Adobe HTTP Dynamic Streaming." www.adobe.com/products/hds-dynamic-streaming.html.

[2] "Akamai HD Adaptive Streaming." http://wwwns.akamai.com/hdnetwork/demo/index.html.

[3] "Amazon Fire TV." https://developer.amazon.com/appsandservices/solutions/devices/fire-tv.

[4] "Apple TV." http://www.apple.com/appletv/.

[5] "Big Buck Bunny Movie." http://www.bigbuckbunny.org.

[6] "BitTorrent." http://www.bittorrent.com/.

[7] "Diaspora." http://www.joindiaspora.com.

[8] "eMule." http://sourceforge.net/projects/emule.

[9] "Facebook." http://www.facebook.com.

[10] "Foursquare." http://www.foursquare.com.

[11] "Google TV." http://www.google.com/tv/.

[12] "Harmonic mean." http://en.wikipedia.org/wiki/Harmonic_mean.

[13] "HTTP Live Streaming." https://developer.apple.com/streaming/.

[14] "Hulu." http://www.hulu.com.

[15] "libdash." http://www-itec.uni-klu.ac.at/dash/?page_id=400.

[16] "NanoHTTPd." https://github.com/NanoHttpd/nanohttpd.

[17] "Netflix." http://www.netflix.com.

[18] "Open Source Media Framework (OSMF)." http://www.osmf.org.

[19] "Smoothstreaming Protocol." http://go.microsoft.com/?linkid=9682896.

[20] "Twitter." http://www.twitter.com.

[21] "Xbox Live." http://www.xbox.com/live/.

[22] "YouTube." http://www.youtube.com.

[23] Adhikari, V. K., Guo, Y., Hao, F., Varvello, M., Hilt, V., Steiner, M., and Zhang, Z.-L., "Unreeling Netflix: Understanding and improving multi-CDN movie delivery.," in *INFOCOM*, pp. 1620–1628, IEEE, 2012.

[24] Aditya, S. and Katti, S., "Flexcast: graceful wireless video streaming," in *Proceedings of the 17th annual international conference on Mobile computing and networking*, pp. 277–288, ACM, 2011.

[25] Akhshabi, S., Anantakrishnan, L., Begen, A. C., and Dovrolis, C., "What happens when http adaptive streaming players compete for bandwidth?," in *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video*, pp. 9–14, ACM, 2012.

[26] Akhshabi, S., Begen, A. C., and Dovrolis, C., "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proceedings of the second annual ACM conference on Multimedia systems*, pp. 157–168, ACM, 2011.

[27] Ananthanarayanan, G., Padmanabhan, V., Ravindranath, L., and Thekkath, C., "Combine: leveraging the power of wireless peers through collaborative downloading," in *Proceedings of the 5th international conference on Mobile systems, applications and services*, pp. 286–298, ACM, 2007.

[28] Android Developers, "Supported Media Formats." http://developer.android.com/guide/appendix/media-formats.html.

[29] Araújo, F. and Rodrigues, L., "Geopeer: A location-aware peer-to-peer system," in *Network Computing and Applications, 2004.(NCA 2004). Proceedings. Third IEEE International Symposium on*, pp. 39–46, IEEE, 2004.

[30] Balachandran, A., Sekar, V., Akella, A., Seshan, S., Stoica, I., and Zhang, H., "A quest for an internet video quality-of-experience metric," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pp. 97–102, ACM, 2012.

[31] Balasubramanian, A., Mahajan, R., and Venkataramani, A., "Augmenting mobile 3g using wifi," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 209–222, ACM, 2010.

[32] Bokani, A., Hassan, M., and Kanhere, S., "Http-based adaptive streaming for mobile clients using markov decision process," in *Packet Video Workshop (PV), 2013 20th International*, IEEE, 2013.

[33] Boldrini, C., Conti, M., and Passarella, A., "Exploiting users" social relations to forward data in opportunistic networks: The hibop solution," *Pervasive and Mobile Computing*, vol. 4, no. 5, pp. 633–657, 2008.

[34] Boldrini, C., Conti, M., and Passarella, A., "Design and performance evaluation of contentplace, a social-aware data dissemination system for opportunistic networks," *Computer Networks*, vol. 54, no. 4, pp. 589–604, 2010.

[35] Bureau of Labor Statistics, "American Time Use Survey." http://www.bls.gov/tus.

[36] Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., and Srivastava, M., "Participatory sensing," in *In: Workshop on World-Sensor-Web (WSW06): Mobile Device Centric Sensor Networks and Applications*, Citeseer, 2006.

[37] Camp, T., Boleng, J., and Davies, V., "A survey of mobility models for *ad hoc* network research," *Wireless communications and mobile computing*, vol. 2, no. 5, pp. 483–502, 2002.

[38] Chakravorty, R., Agarwal, S., Banerjee, S., and Pratt, I., "MoB: A mobile bazaar for wide-area wireless services," in *Proceedings of the 11th annual international conference on Mobile computing and networking*, pp. 228–242, ACM, 2005.

[39] Chandra, R., Padhye, J., and Ravindranath, L., "Wi-fi neighborcast: enabling communication among nearby clients," in *Proceedings of the 9th workshop on Mobile computing systems and applications*, pp. 38–42, ACM, 2008.

[40] Chen, J., Ghosh, A., Magutt, J., and Chiang, M., "QAVA: quota aware video adaptation," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pp. 121–132, ACM, 2012.

[41] Cheng, X. and Liu, J., "NetTube: Exploring social networks for peer-to-peer short video sharing," in *INFOCOM 2009, IEEE*, pp. 1152–1160, IEEE, 2009.

[42] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update 2011 to 2016." http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html, 2012.

[43] Cisco, "Mobile subscribers worldwide." http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/a#subscribers, 2012.

[44] Cranley, N., Perry, P., and Murphy, L., "User perception of adapting video quality," *International Journal of Human-Computer Studies*, vol. 64, no. 8, pp. 637–647, 2006.

[45] Cranshaw, J., Toch, E., Hong, J., Kittur, A., and Sadeh, N., "Bridging the gap between physical location and online social networks," in *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pp. 119–128, ACM, 2010.

[46] Das, T., Mohan, P., Padmanabhan, V., Ramjee, R., and Sharma, A., "PRISM: platform for remote sensing using smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 63–76, ACM, 2010.

[47] De Cicco, L. and Mascolo, S., "An experimental investigation of the akamai adaptive video streaming," in *HCI in work and learning, life and leisure*, pp. 447–464, Springer, 2010.

[48] de Nitto Personè, V., Grassi, V., and Morlupi, A., "Modeling and evaluation of prefetching policies for context-aware information services," in *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 55–65, ACM, 1998.

[49] Do, N., Hsu, C., and Venkatasubramanian, N., "CrowdMAC: A crowdsourcing system for mobile access," *Middleware 2012*, pp. 1–20, 2012.

[50] Dobrian, F., Sekar, V., Awan, A., Stoica, I., Joseph, D. A., Ganjam, A., Zhan, J., and Zhang, H., "Understanding the impact of video quality on user engagement," *SIGCOMM-Computer Communication Review*, vol. 41, no. 4, p. 362, 2011.

[51] Ducatelle, F., Di Caro, G., and Gambardella, L. M., "A study on the use of manets in urban environments," tech. rep., Technical Report 01-07, IDSIA, Switzerland, 2007.

[52] Econsultancy, "Online video sharing doubles within a year." http://econsultancy.com/blog/7198-online-video-sharing-doubles-within-a-year, 2012.

[53] Erman, J., Gerber, A., Ramadrishnan, K., Sen, S., and Spatscheck, O., "Over the top video: the gorilla in cellular networks," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 127–136, ACM, 2011.

[54] Facebook, "Facebook Android SDK." https://developers.facebook.com/docs/android/.

[55] Facebook, "Facebook Graph API." https://developers.facebook.com/docs/graph-api/.

[56] Falaki, H., Lymberopoulos, D., Mahajan, R., Kandula, S., and Estrin, D., "A first look at traffic on smartphones," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 281–287, ACM, 2010.

[57] Feeney, L. M., Ahlgren, B., and Westerlund, A., "Spontaneous networking: an application oriented approach to *ad hoc* networking," *Communications Magazine, IEEE*, vol. 39, no. 6, pp. 176–181, 2001.

[58] Finamore, A., Mellia, M., Munafò, M. M., Torres, R., and Rao, S. G., "Youtube everywhere: Impact of device and infrastructure synergies on user experience," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 345–360, ACM, 2011.

[59] Ford, B., Strauss, J., Lesniewski-Laas, C., Rhea, S., Kaashoek, F., and Morris, R., "Persistent personal names for globally connected mobile devices," in *Proceedings of the 7th symposium on Operating systems design and implementation*, pp. 233–248, USENIX Association, 2006.

[60] Fund, F., Wang, C., Liu, Y., Korakis, T., Zink, M., and Panwar, S. S., "Performance of dash and webrtc video services for mobile users," in *Packet Video Workshop (PV), 2013 30th International*, IEEE, 2013.

[61] Gaonkar, S., Li, J., Choudhury, R., Cox, L., and Schmidt, A., "Micro-blog: sharing and querying content through mobile phones and social participation," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pp. 174–186, ACM, 2008.

[62] Gautam, N., Petander, H., and Noel, J., "A comparison of the cost and energy efficiency of prefetching and streaming of mobile video," in *Proceedings of the 5th Workshop on Mobile Video*, pp. 7–12, ACM, 2013.

[63] Gember, A., Akella, A., Pang, J., Varshavsky, A., and Caceres, R., "Obtaining in-context measurements of cellular network performance," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, pp. 287–300, ACM, 2012.

[64] Google, "Devices on Google Play." https://play.google.com/store/devices?hl=en, Q2 2013.

[65] Gouta, A., Hong, C., Hong, D., Kermarrec, A.-M., Lelouedec, Y., and others, "Large scale analysis of HTTP adaptive streaming in mobile networks," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, IEEE, 2013.

[66] Hätönen, S., Nyrhinen, A., Eggert, L., Strowes, S., Sarolahti, P., and Kojo, M., "An experimental study of home gateway characteristics," in *Proceedings of the 10th annual conference on Internet measurement*, pp. 260–266, ACM, 2010.

[67] Higgins, B. D., Flinn, J., Giuli, T. J., Noble, B., Peplin, C., and Watson, D., "Informed mobile prefetching," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 155–168, ACM, 2012.

[68] Higgins, B. D., Reda, A., Alperovich, T., Flinn, J., Giuli, T. J., Noble, B., and Watson, D., "Intentional networking: opportunistic exploitation of mobile network diversity," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pp. 73–84, ACM, 2010.

[69] Hong, Y.-S., Kim, J.-H., and Kim, Y.-H., "A buffer-controlled adaptive video streaming for mobile devices," in *Convergence Information Technology, 2007. International Conference on*, pp. 30–35, IEEE, 2007.

[70] Huang, J., Chen, C., Pei, Y., Wang, Z., Qian, Z., Qian, F., Tiwana, B., Xu, Q., Mao, Z., Zhang, M., and others, "Mobiperf: Mobile network measurement system," tech. rep., Technical report, Technical report). University of Michigan and Microsoft Research, 2011.

[71] Huang, J., Qian, F., Gerber, A., Mao, Z. M., Sen, S., and Spatscheck, O., "A close examination of performance and power characteristics of 4G LTE networks," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 225–238, ACM, 2012.

[72] Huang, J., Xu, Q., Tiwana, B., Mao, Z. M., Zhang, M., and Bahl, P., "Anatomizing application performance differences on smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 165–178, ACM, 2010.

[73] Huang, Y., Fu, T., Chiu, D., Lui, J., and Huang, C., "Challenges, design and analysis of a large-scale p2p-vod system," in *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 375–388, ACM, 2008.

[74] Hui, P., Crowcroft, J., and Yoneki, E., "Bubble rap: Social-based forwarding in delay-tolerant networks," *Mobile Computing, IEEE Transactions on*, vol. 10, no. 11, pp. 1576–1589, 2011.

[75] Jiang, J., Sekar, V., and Zhang, H., "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pp. 97–108, ACM, 2012.

[76] Jin, X. and Kwok, Y.-K., "Cloud assisted p2p media streaming for bandwidth constrained mobile subscribers," in *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*, pp. 800–805, IEEE, 2010.

[77] Keller, L., Le, A., Cici, B., Seferoglu, H., Fragouli, C., and Markopoulou, A., "Microcast: Cooperative video streaming on smartphones," in *Proceedings of the*

*10th international conference on Mobile systems, applications, and services*, pp. 57–70, ACM, 2012.

[78] Kleinberg, J., "The convergence of social and technological networks," *Communications of the ACM*, vol. 51, no. 11, pp. 66–72, 2008.

[79] Kovacevic, A., Liebau, N., and Steinmetz, R., "Globase. kom-a p2p overlay for fully retrievable location-based search," in *Peer-to-Peer Computing, 2007. P2P 2007. Seventh IEEE International Conference on*, pp. 87–96, IEEE, 2007.

[80] Krause, A., Horvitz, E., Kansal, A., and Zhao, F., "Toward community sensing," in *Proceedings of the 7th international conference on Information processing in sensor networks*, pp. 481–492, IEEE Computer Society, 2008.

[81] Kreitz, G. and Niemela, F., "Spotify–large scale, low latency, p2p music-on-demand streaming," in *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, pp. 1–10, IEEE, 2010.

[82] Krishnan, S. S. and Sitaraman, R. K., "Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, pp. 211–224, ACM, 2012.

[83] Lederer, S., Muller, C., and Timmerer, C., "Towards peer-assisted dynamic adaptive streaming over HTTP," in *Packet Video Workshop (PV), 2012 19th International*, pp. 161–166, IEEE, 2012.

[84] Lederer, S., Mueller, C., Rainer, B., Timmerer, C., and Hellwagner, H., "Adaptive streaming over content centric networks in mobile networks using multiple links," in *Proc. of the IEEE Int. Workshop on Immersive & Interactive Multimedia Comm. over the Future Internet*, 2013.

[85] Lee, Y., Ju, Y., Min, C., Kang, S., Hwang, I., and Song, J., "Comon: cooperative ambience monitoring platform with continuity and benefit awareness," in

*Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 43–56, ACM, 2012.

[86] Lenders, V., Karlsson, G., and May, M., "Wireless *ad hoc* podcasting," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON'07. 4th Annual IEEE Communications Society Conference on*, pp. 273–283, IEEE, 2007.

[87] Li, J., Blake, C., De Couto, D. S., Lee, H. I., and Morris, R., "Capacity of *ad hoc* wireless networks," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, pp. 61–69, ACM, 2001.

[88] Li, Z., Lin, J., Akodjenou, M.-I., Xie, G., Kaafar, M. A., Jin, Y., and Peng, G., "Watching videos from everywhere: a study of the PPTV mobile VoD system," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, pp. 185–198, ACM, 2012.

[89] Li, Z., Zhu, X., Gahm, J., Pan, R., Hu, H., Begen, A. C., and Oran, D., "Probe and Adapt: rate adaptation for HTTP video streaming at scale," *arXiv preprint arXiv:1305.0510*, 2013.

[90] Liao, X., Jin, H., Liu, Y., Ni, L. M., and Deng, D., "Anysee: Peer-to-peer live streaming.," in *INFOCOM*, vol. 25, pp. 1–10, 2006.

[91] Liu, C., Bouazizi, I., and Gabbouj, M., "Rate adaptation for adaptive HTTP streaming," *Proc. ACM MMSys*, 2011.

[92] Liu, X., Dobrian, F., Milner, H., Jiang, J., Sekar, V., Stoica, I., and Zhang, H., "A case for a coordinated internet video control plane," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 359–370, ACM, 2012.

[93] Liu, Y., Shen, B., Chen, S., Li, F., Guo, L., and Lan, Y., "Measurement and analysis of an internet streaming service to mobile devices," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. 1, p. 1, 2012.

[94] Liu, Y., Guo, Y., and Liang, C., "A survey on peer-to-peer video streaming systems," *Peer-to-peer Networking and Applications*, vol. 1, no. 1, pp. 18–28, 2008.

[95] Miller, K., Quacchio, E., Gennari, G., and Wolisz, A., "Adaptation algorithm for adaptive streaming over HTTP," in *Packet Video Workshop (PV), 2012 19th International*, pp. 173–178, IEEE, 2012.

[96] Mok, R. K., Chan, E. W., Luo, X., and Chang, R. K., "Inferring the QoE of HTTP video streaming from user-viewing activities," in *Proceedings of the first ACM SIGCOMM workshop on Measurements up the stack*, pp. 31–36, ACM, 2011.

[97] Mok, R., Chan, E., and Chang, R., "Measuring the quality of experience of HTTP video streaming," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pp. 485–492, IEEE, 2011.

[98] Mok, R., Luo, X., Chan, E., and Chang, R., "QDASH: a QoE-aware DASH system," in *Proceedings of the 3rd Multimedia Systems Conference*, pp. 11–22, ACM, 2012.

[99] Motani, M., Srinivasan, V., and Nuggehalli, P., "PeopleNet: engineering a wireless virtual social network," in *Proceedings of the 11th annual international conference on Mobile computing and networking*, pp. 243–257, ACM, 2005.

[100] Mtibaa, A., Chaintreau, A., LeBrun, J., Oliver, E., Pietilainen, A.-K., and Diot, C., "Are you moved by your social network application," in *Proceedings of the first workshop on Online social networks*, pp. 67–72, ACM, 2008.

[101] Müller, C., Lederer, S., and Timmerer, C., "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments," in *Proceedings of the 4th Workshop on Mobile Video*, pp. 37–42, ACM, 2012.

[102] Niesen, U., Shah, D., and Wornell, G. W., "Caching in wireless networks," *Information Theory, IEEE Transactions on*, vol. 58, no. 10, pp. 6524–6540, 2012.

[103] Nordström, E., Gunningberg, P., and Rohner, C., "A search-based network architecture for mobile devices," *Department of Information Technology, Uppsala University, Tech. Rep*, vol. 3, 2009.

[104] Ooyala, "The Global Online Video Index Report: Metrics that Matter." http://go.ooyala.com/rs/OOYALA/images/Ooyala-Global-Video-Index-Q2-2013.pdf, Q2 2013.

[105] Ott, J., Hyytia, E., Lassila, P., Vaegs, T., and Kangasharju, J., "Floating content: Information sharing in urban areas," in *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pp. 136–146, IEEE, 2011.

[106] Payberah, A. H., Kavalionak, H., Kumaresan, V., Montresor, A., and Haridi, S., "Clive: Cloud-assisted p2p live streaming," in *Peer-to-Peer Computing (P2P), 2012 IEEE 12th International Conference on*, pp. 79–90, IEEE, 2012.

[107] PewInternet, "71% of online adults now use video-sharing sites." http://pewinternet.org/Reports/2011/Video-sharing-sites.aspx, 2012.

[108] Pietiläinen, A., Oliver, E., LeBrun, J., Varghese, G., and Diot, C., "MobiClique: middleware for mobile social networking," in *Proceedings of the 2nd ACM workshop on Online social networks*, pp. 49–54, ACM, 2009.

[109] Ra, M., Liu, B., La Porta, T., and Govindan, R., "Medusa: A programming framework for crowd-sensing applications," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 337–350, ACM, 2012.

[110] Rachuri, K., Mascolo, C., Musolesi, M., and Rentfrow, P., "Sociablesense: exploring the trade-offs of adaptive sampling and computation offloading for social sensing," in *Proceedings of the 17th annual international conference on Mobile computing and networking*, pp. 73–84, ACM, 2011.

[111] Ransburg, M., Jonke, M., and Hellwagner, H., "An evaluation of mobile end devices in multimedia streaming scenarios," in *Mobile Wireless Middleware, Operating Systems, and Applications*, pp. 400–412, Springer, 2010.

[112] Rao, A., Legout, A., Lim, Y.-s., Towsley, D., Barakat, C., and Dabbous, W., "Network characteristics of video streaming traffic," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, p. 25, ACM, 2011.

[113] Roman, G.-C., Huang, Q., and Hazemi, A., "Consistent group membership in *ad hoc* networks," in *Proceedings of the 23rd international conference on Software engineering*, pp. 381–388, IEEE Computer Society, 2001.

[114] Rubio Romero, L., "A dynamic adaptive HTTP streaming video service for Google Android," Master's thesis, KTH, 2011.

[115] Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N., "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.

[116] Scott, J., Crowcroft, J., Hui, P., Diot, C., and others, "Haggle: A networking architecture designed around mobile users," in *WONS 2006: Third Annual Conference on Wireless On-demand Network Systems and Services*, pp. 78–86, 2006.

[117] Seo, B., Cui, W., and Zimmermann, R., "An experimental study of video uploading from mobile devices with HTTP streaming," in *Proceedings of the 3rd Multimedia Systems Conference*, pp. 215–225, ACM, 2012.

[118] Sommers, J. and Barford, P., "Cell vs. WiFi: on the performance of metro area mobile connections," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, pp. 301–314, ACM, 2012.

[119] Stockhammer, T., "Dynamic adaptive streaming over HTTP: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*, MMSys '11, pp. 133–144, 2011.

[120] Sullivan, M., "A day in the life of 3G." http://www.pcworld.com/article/167391/3GTests.html, 2009.

[121] Sun, Y., Guo, Y., Li, Z., Lin, J., Xie, G., Zhang, X., and Salamatian, K., "The case for P2P mobile video system over wireless broadband networks: A practical study of challenges for a mobile video provider," *Network, IEEE*, vol. 27, no. 2, pp. 22–27, 2013.

[122] Tian, G. and Liu, Y., "Towards agile and smooth video adaptation in dynamic HTTP streaming," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pp. 109–120, ACM, 2012.

[123] Tian, G. and Liu, Y., "On adaptive http streaming to mobile devices," in *Packet Video Workshop (PV), 2013 20th International*, IEEE, 2013.

[124] Trestian, I., Ranjan, S., Kuzmanovic, A., and Nucci, A., "Measuring serendipity: connecting people, locations and interests in a mobile 3g network," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pp. 267–279, ACM, 2009.

[125] Unruly Media, "Branded video sharing almost 50 times higher in 2013 than in 2006." http://www.unrulymedia.com/article/04-12-2013/branded-video-sharing-almost-50-times-higher-2013-2006/1354615200, 2013.

[126] UPnP Forum, "UPnP." http://upnp.org/about/what-is-upnp/.

[127] Vasudevan, S., Towsley, D., Goeckel, D., and Khalili, R., "Neighbor discovery in wireless networks and the coupon collector's problem," in *Proceedings of the 15th annual international conference on Mobile computing and networking*, pp. 181–192, ACM, 2009.

[128] Wang, Z., Qian, Z., Xu, Q., Mao, Z., and Zhang, M., "An untold story of middleboxes in cellular networks," *SIGCOMM-Computer Communication Review*, vol. 41, no. 4, p. 374, 2011.

[129] Wang, Z., Sun, L., Yang, S., and Zhu, W., "Prefetching strategy in peer-assisted social video streaming," in *Proceedings of the 19th ACM international conference on Multimedia*, pp. 1233–1236, ACM, 2011.

[130] Watson, T., "Application design for wireless computing," in *Mobile Computing*, pp. 363–373, Springer, 1996.

[131] Whitehouse, K., Sharp, C., Brewer, E., and Culler, D., "Hood: a neighborhood abstraction for sensor networks," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pp. 99–110, ACM, 2004.

[132] Wu, Y., Wu, C., Li, B., Qiu, X., and Lau, F. C., "Cloudmedia: When cloud on demand meets video on demand," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pp. 268–277, IEEE, 2011.

[133] Xiang, S., Cai, L., and Pan, J., "Adaptive scalable video streaming in wireless networks," in *Proceedings of the 3rd Multimedia Systems Conference*, pp. 167–172, ACM, 2012.

[134] Yin, H., Liu, X., Zhan, T., Sekar, V., Qiu, F., Lin, C., Zhang, H., and Li, B., "Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky," in *Proceedings of the 17th ACM international conference on Multimedia*, MM '09, pp. 25–34, 2009.

[135] Yin, L. and Cao, G., "Supporting cooperative caching in *ad hoc* networks," *Mobile Computing, IEEE Transactions on*, vol. 5, no. 1, pp. 77–89, 2006.

[136] Yusuf, L. and Ramachandran, U., "Community membership management for transient social networks," in *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pp. 1–7, IEEE, 2012.

[137] Yusuf, L. and Ramachandran, U., "VirtualConnection: opportunistic networking for web on demand," in *Distributed Computing and Networking*, pp. 323–340, Springer, 2010.

[138] Zhang, X., Liu, J., Li, B., and Yum, Y., "CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming," in *INFOCOM 2005*, vol. 3, pp. 2102–2111, IEEE, 2005.

[139] Zhao, J., Zhang, P., Cao, G., and Das, C. R., "Cooperative caching in wireless p2p networks: Design, implementation, and evaluation," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 2, pp. 229–241, 2010.

[140] Zhou, C., Zhang, X., Huo, L., and Guo, Z., "A control-theoretic approach to rate adaptation for dynamic HTTP streaming," in *Visual Communications and Image Processing (VCIP), 2012 IEEE*, pp. 1–6, IEEE, 2012.