Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Fast track article

Mobile multimedia streaming techniques: QoE and energy saving perspective

Mohammad Ashraful Hoque [a,c,*], Matti Siekkinen [a], Jukka K. Nurminen [a], Mika Aalto [b], Sasu Tarkoma [c]

[a] *Aalto University School of Science, Finland*
[b] *Nokia Solutions and Networks, Finland*
[c] *University of Helsinki, Finland*

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Performance analysis
Measurement
Power consumption
Wireless multimedia
Quality of experience (QoE)

ABSTRACT

Multimedia streaming to mobile devices is challenging for two reasons. First, the way content is delivered to a client must ensure that the user does not experience a long initial playback delay or a distorted playback in the middle of a streaming session. Second, multimedia streaming applications are among the most energy hungry applications in smartphones. The energy consumption mostly depends on the delivery techniques and on the power management techniques of wireless access technologies (Wi-Fi, 3G, and 4G). In order to provide insights on what kind of streaming techniques exist, how they work on different mobile platforms, their efforts in providing smooth quality of experience, and their impact on energy consumption of mobile phones, we did a large set of active measurements with several smartphones having both Wi-Fi and cellular network access. Our analysis reveals five different techniques to deliver the content to the video players. The selection of a technique depends on the mobile platform, device, player, quality, and service. The results from our traffic and power measurements allow us to conclude that none of the identified techniques is optimal because they take none of the following facts into account: access technology used, user behavior, and user preferences concerning data waste. We point out the technique with the best playback buffer configuration, which provides the most attractive trade-offs in particular situations.

1. Introduction

Multimedia streaming services, such as YouTube, Dailymotion, Vimeo and Netflix, consider a number of challenges while streaming videos to the clients for providing smooth playback, such as initial playback delay, clients with different kinds of connectivity, and the bandwidth variation between a server and a client [1]. The aforementioned streaming services have adopted various techniques, such as encoding rate streaming, rate throttling, buffer adaptive streaming, fast caching, and rate adaptive streaming over HTTP. Encoding rate streaming is used to deliver content at the encoding rate. Throttling and fast aching send video content at a higher rate than the encoding rate. Buffer adaptive mechanisms work based on the playback buffer status of a client player. In this case, the client receives content from the server only when the playback buffer drains to a specific lower threshold. Fast caching allows the player to download the whole content as fast as possible. Rate

* Corresponding author at: Aalto University School of Science, Finland. Tel.: +358 451684854.
E-mail addresses: mohammad.hoque@aalto.fi, mohammad.hoque@cs.helsinki.fi (M.A. Hoque), matti.siekkinen@aalto.fi (M. Siekkinen), jukka.k.nurminen@aalto.fi (J.K. Nurminen), mika.aalto@nsn.fi (M. Aalto), sasu.tarkoma@cs.helsinki.fi (S. Tarkoma).

http://dx.doi.org/10.1016/j.pmcj.2014.05.004
1574-1192/© 2014 Elsevier B.V. All rights reserved.

adaptive mechanisms adapt video quality according to the end-to-end bandwidth between a server and the client. However, while consuming multimedia streaming content, energy consumption of smartphones is also considered as an important issue. Therefore, a significant number of research work focused on reducing energy consumption of mobile devices in this regard [2].

There has been work on analyzing the merits of these streaming techniques from the server performance point of view. For example, fast caching reduces start-up delay at the client and guards against bandwidth fluctuation, but it also consumes a lot of CPU and memory at the streaming server [1]. Although most of the techniques are understood by research community, a thorough study of these streaming techniques is still required from the perspective of the mobile device and the user. Even though some studies have looked at the traffic pattern of video streaming services with Android, iOS devices, and desktop users [3–5], at present it is not well understood how the different techniques are chosen, how they compare to each other, and what are the optimal techniques to use in different contexts. Most importantly, the effect of these streaming techniques on user satisfaction on playback quality, Wi-Fi and cellular network usage, and on the energy consumption of mobile devices is yet to be fully uncovered. Such knowledge is imperative before one can design a streaming service that satisfies users demands in terms of quality of experience and battery life of their smartphones.

We actively captured traffic of more than five hundred video streaming sessions in smartphones of different mobile platforms, from YouTube, Vimeo, Dailymotion and Netflix, via Wi-Fi, HSPA, and LTE. During those sessions we estimated the joining time. From the captured traffic, we computed the playback buffer status. In the meanwhile, we also measured the energy consumed by the smartphones for playback and for wireless network interface (WNI) usage. Our main observations are the following:

- Video streaming servers apply fast caching and throttling to deliver content to the players, whereas the video players enforce encoding rate and buffer adaptive mechanisms by exploiting TCP's flow control, hence, overriding the server selected mechanisms. In encoding rate streaming, the player unintentionally triggers TCP flow control because the player has too small playback buffer compared to the amount of content the server offers. The buffer adaptive mechanisms deliberately pause and resume download, and these techniques are applied only by the video players in Android phones (Section 4).
- Our analysis reveals that in smartphones different techniques are applied with little or no consensus: different techniques are used by different clients to access the same service in the same context. For example, Android devices use three different techniques for YouTube videos. The selection of those techniques depends on the quality of the video and the player. However, the strategy selection does not depend on the wireless interface being used for streaming and, thus, network operators do not play any role (Section 4).
- The joining time (a.k.a. initial playback delay) varies according to the wireless interface being used for streaming, the quality of the content, and the video service. The players experience shorter delay when streaming via Wi-Fi than HSPA or LTE. From the quality perspective, low quality videos are played with a shorter initial delay. Among the targeted video services, the Netflix players experience the longest delay. However, most of the streaming strategies are optimized for providing uninterrupted playback by allowing the players to keep a large amount of data in the playback buffer (Section 5).
- There is a large variation in playback energy consumption between different types of players and containers on the same device. The differences are due to inefficient player implementation. However, the video quality (resolution) does not seem to have a large impact on energy consumption (Section 6.1). Energy consumption of WNIs also vary according to the interface type, streaming rate, power saving strategy implementation, network configuration, and device type (Section 6.2).
- When the user views the entire video clip, fast caching and throttling are the most optimized techniques for providing uninterrupted playback at the client. At the same time, they are the most energy efficient. If the user is likely to interrupt the video viewing, buffer adaptive streaming is more attractive as the player generates ON–OFF traffic pattern and less energy is consumed for wireless communication during an OFF period. However, the ON period duration should be adjusted to match fast start period in order to avoid server rate throttling. Similarly, the duration of the OFF period should also be optimized so that the player does not suffer from playback buffer starvation. However, none of the identified techniques alone is optimal because they do not adapt to the wireless access technology, user behavior, and preferences (Section 6.3).

We structure our paper as follows. In the next section, we briefly describe the energy consumption characteristics of wireless communication in smartphones and explain the characteristics of mobile video streaming. In Section 3, we describe our measurement and data collection methodology. In Section 4, we investigate the different streaming techniques. Section 5 examines the effort of the streaming techniques in providing uninterrupted playback. Section 6 is devoted to presenting the results from the energy consumption measurements. In Section 7, we discuss the tradeoff between energy savings and potential playback buffer underrun. Finally, we contrast our work with earlier research in Section 8 before concluding the paper.

2. Background

Smartphones allow users to access Internet via Wi-Fi and mobile broadband access. Mobile broadband experience is enabled by the latest 3G and 4G technologies such as EV-DO, HSPA, and LTE. The most widely deployed mobile broadband

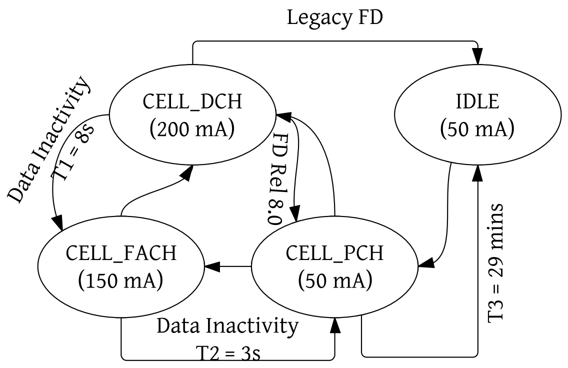


Fig. 1. WCDMA/HSPA RRC states with typical values of the inactivity timers and power consumption.

technology is currently HSPA, while LTE is the fastest ever growing cellular and mobile broadband technology. In this section, we first review the power consumption characteristics of Wi-Fi and cellular interfaces that we use in this study. Then, we explain the characteristics of mobile streaming services and the metrics to assess the quality of experience of the users.

2.1. Power saving mechanisms for wireless network interfaces

2.1.1. Wi-Fi

Smartphones implement 802.11 Power Saving Mechanism (PSM) to manage the power consumption of Wi-Fi. There are four states; transmit, receive, idle and sleep. PSM allows the interface to be in sleep when there is no data activity. However, a mobile device periodically powers up the interface to receive a traffic indication map (TIM) frame from the access point (AP). This interval is usually 100 ms and also called listen interval. TIM frame tells the mobile whether the AP has some buffered data for the device or not. If the AP has data, the mobile device sends PS-Poll frame in return to receive the buffered data. Otherwise, it goes back to sleep. Modern devices usually implement a timer which keeps the interface in idle state for a few hundred milliseconds after the transmission or reception of packets, which improves especially the performance of short TCP connections. This is also known as PSM adaptive [6].

2.1.2. WCDMA/HSPA

3rd Generation Partnership Project (3GPP) standards specify the efficient usage of the radio resources considering the mobility and power consumption of smartphones via a resource control protocol (RRC). Fig. 1 shows that there are a number of states and inactivity timers in 3GPP RRC protocol for WCDMA/HSPA. These timers ensure that if a certain resource is not utilized for a certain period of time in a particular state, the resource must be released. For example, high volume data transmission happens in CELL_DCH state and small packet transmission is possible in CELL_FACH state. A mobile device switches from CELL_DCH to CELL_FACH in absence of data activity for a period of $T1$ s. These timers have static values and they are operator controlled. If the mobile device and network both support Rel 8.0 Fast Dormancy (FD) [7], CELL_DCH $\rightarrow$ CELL_PCH transition happens. For non standard FD, the transition is CELL_DCH $\rightarrow$ IDLE (Fig. 1), which releases the RRC connection altogether.

RRC protocol has a large impact on the energy spending by the smartphones. Fig. 1 also shows that average current consumption in CELL_DCH is 200 mA, in CELL_FACH is 150 mA, and in CELL_PCH is 50 mA approximately. The potential consequence especially with long inactivity timers is high energy consumption at the mobile devices. To learn more about different cellular network configurations and their effect on energy consumption, readers can follow [8].

2.1.3. LTE

The radio resource control protocol for LTE specifies only two states; RRC_IDLE and RRC_CONNECTED. Similar to the HSPA RRC protocol, an inactivity timer ($\mathrm{RRC}_{\mathrm{idle}}$) controls the connected to idle state transition. LTE includes a discontinuous transmission and reception (DTX/DRX) mechanism that enables a mobile device to consume low power even being in the RRC_CONNECTED state. DRX in the connected state is also called connected mode DRX or cDRX, and the associated inactivity timer is $\mathrm{DRX}_{\mathrm{idle}}$. Fig. 2 shows that if there is no data activity for $\mathrm{DRX}_{\mathrm{idle}}$ time, then a DRX cycle, DRX_c, is initiated. The length of a such cycle can vary from 20 ms to few seconds. The device checks data activity during the on period, $\mathrm{DRX}_{\mathrm{on}}$, of the cycle. If the data inactivity continues for a longer time, $\mathrm{RRC}_{\mathrm{idle}}$, the network commands the device to switch from RRC_CONNECTED to RRC_IDLE state. Then the device enters in the channel paging mode in the IDLE state.

From the previous discussion, we can find that the power management techniques of the wireless technologies work in a similar fashion. The wireless interfaces keep their radio powered on for some time after the data transmission. During this inactivity period, the interfaces spend energy doing nothing useful for the sake of user experience. This energy is also known as tail energy [9] and the amount of energy spent depends on the duration of the inactivity timers.

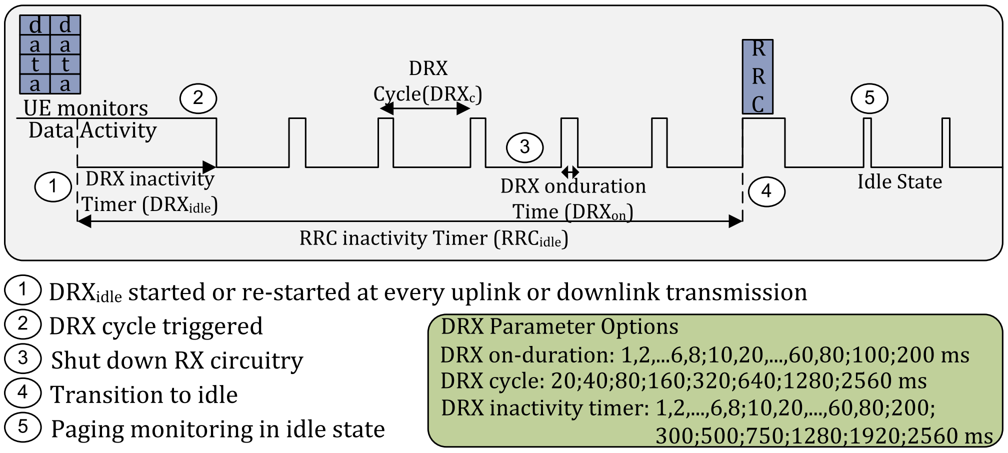

data
data
UE monitors Data Activity
DRX inactivity Timer (DRXidle)
DRX Cycle(DRXc)
DRX onduration Time (DRXon)
RRC
RRC inactivity Timer (RRCidle)
Idle State
1 DRXidle started or re-started at every uplink or downlink transmission
2 DRX cycle triggered
3 Shut down RX circuitry
4 Transition to idle
5 Paging monitoring in idle state
DRX Parameter Options
DRX on-duration: 1,2,...6,8;10,20,...,60,80;100;200 ms
DRX cycle: 20;40;80;160;320;640;1280;2560 ms
DRX inactivity timer: 1,2,...,6,8;10,20,...,60,80;200; 300;500;750;1280;1920;2560 ms

Fig. 2. LTE DRX cycles and timers.

Table 1
Streaming services, the players used by the clients for playback, the quality of the content and the containers to deliver the content.

Streaming services	YouTube, Vimeo, Dailymotion, and Netflix
Players	Native application, Flash, and HTML5
Video quality	LD (240p), SD (270–480p), HD (720–1080p)
Containers	3GPP, MP4, WebM, X-FLV, ismv

2.2. *Mobile video streaming*

Today mobile streaming services deliver content using HTTP over TCP. Smartphone users can access these services using either a native app or a browser. The browser may load a Flash, HTML5 or Microsoft Silverlight player. The quality of the video played is often denoted with a p-notation, such as 240p, which refers to the resolution of the video. 240p usually refers to 360×240 resolution. Different services may use low, standard, and high definition (LD, SD, HD) notations but the resolutions that each one refers to varies between services. Therefore, we define 240p videos as LD, 270–480p videos as SD and 720–1080p or higher resolution videos as HD. MP4, WebM, and X-FLV are the default containers for the players. The native apps of YouTube, Dailymotion and Vimeo also play MP4 and 3GPP videos. Netflix players play ismv videos. WebM and X-FLV are the default containers for the HTML5 and Flash player respectively. Table 1 shows the examples of different video services, the types of video players, video qualities, and the containers.

2.3. *Quality of experience*

The quality of streaming perceived by a user is influenced by the network condition, content quality (e.g. HD or SD), user's preference on the content, and the context in which the user is viewing a video. The network condition translates to network congestion caused by the bottleneck point in between a streaming client and the server. This network congestion is evidenced by the reduced available bandwidth. The impact is realized by the user as long initial playback delay and pauses or playback starvation during playback. In wireless networks, the bottleneck situation can arise when multiple users share the common resources and the throughput per user is reduced so much that user experience is degraded. The bottleneck also can be caused by the radio conditions, i.e. in cell edge the available bit rate is much lower than peak HSPA/LTE bit rates even in an empty cell. The state transition of the WNIs can introduce additional delays.

For dealing with various network conditions, video services apply a number of strategies; (i) Fast caching, (ii) Throttling, (iii) Encoding rate streaming, (iv) Buffer adaptive streaming, and (v) Rate adaptive streaming. A common feature of all streaming services is an initial buffering of multimedia content at the client. This initial buffering is also referred to as Fast Start. The name comes from the fact that a player downloads content using all the available bandwidth. Fast caching is similar to Fast Start, the only difference is that fast caching lasts longer until the whole content is downloaded. These techniques are used by the services for constant bit rate streaming, except rate adaptive streaming. The most prevalent forms of rate adaptive streaming are HTTP live streaming (HLS), Microsoft Smooth Streaming (MSS).

3. Measurement and data collection

3.1. *Properties of the multimedia content*

Compared with our earlier work [10], we excluded previous results for Meego, Symbian, and WP7.5 platforms. We included three latest smartphones; iPhone5, Galaxy S3 LTE (GS3 LTE) and Lumia825. All the video services, YouTube,

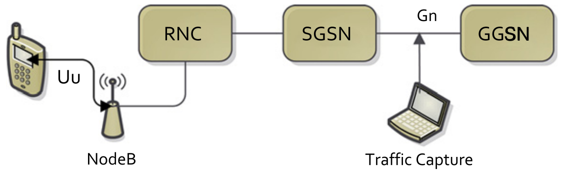


Fig. 3. Capturing traffic at the Gn interface between SGSN and GGSN in the test HSPA Network.

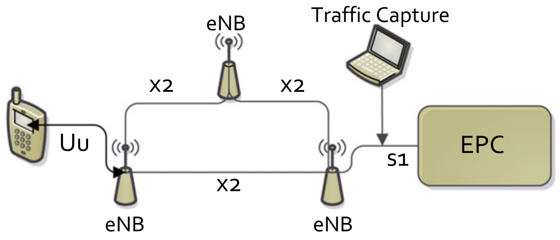


Fig. 4. Capturing traffic at the S1 interface between the eNB (Base Station) and EPC (Evolved Packet Core) in the LTE network.

Table 2
LTE network configurations.

Config name	Parameters
noDRX	RRC_idle = 10 s
$DRX_{80\ ms}$	RRC_{idle} = 10 s, DRX_{cycle} = 80 ms, DRX_{on} = 10 ms,
$DRX_{160\ ms}$	RRC_{idle} = 10 s, DRX_{cycle} = 160 ms, DRX_{on} = 10 ms,
$DRX_{640\ ms}$	RRC_{idle} = 10 s, DRX_{cycle} = 640 ms, DRX_{on} = 10 ms,

Dailymotion, Vimeo and Netflix, have the native applications for the target mobile platforms. The desktop edition of YouTube was used only in the Android platforms as it provides the opportunity to use both Flash and HTML5 players. Our target video services, players, and smartphones are listed in Table 3. Whenever available for the particular smartphone and player, we streamed videos of multiple qualities that range from LD to HD. The average duration of the videos was 10 min.

3.2. Network setup

We watched videos from the video services in the smartphones via Wi-Fi, HSPA, and LTE. In the case of Wi-Fi, a 802.11 b/g access point was used. The access point was connected to the Internet via 100 Mbps Ethernet. AirPcap[1] was used to capture the Wi-Fi traffic. HSPA network measurements were conducted in the Nokia Solutions and Networks test networks. The network parameters, i.e. states and inactivity timers, were configured according to the vendor recommendation. The values of the inactivity timers were from few seconds to few minutes; $T1 = 8$ s, $T2 = 3$ s, $T3 = 29$ min. The CELL_PCH state was enabled in the network. We captured traffic of the streaming clients at the Gn interface between SGSN and GGSN (see Fig. 3). The LTE measurements were conducted with connected mode DRX enabled in the network. Traffic capture is taken at the S1 interface between the eNB and EPC (see Fig. 4). We measured power consumption with three sets of DRX profiles. The DRX profiles are listed in Table 2.

3.3. Power measurement

We used Monsoon[2] and another custom power monitor for measuring the energy consumption of the smartphones during multimedia streaming. We removed the battery of most of the mobile phones and powered them using the measurement devices. Only the iPhones were powered from their batteries. All the devices were in automatic brightness settings during the measurements.

4. Streaming techniques

We identified the streaming techniques used by the constant bit rate and rate-adaptive streaming services. By constant bit rate we refer in this context to constant quality (resolution) as opposed to rate-adaptive streaming. The former may include also variable bit rate encoding that does not change the video resolution. From the captured traffic traces, we inferred the

1 AirPcap: www.cacetech.com/documents/AirPcap%20Nx%20Datasheet.pdf.
2 Monsoon Power Monitor: www.msoon.com.

Table 3
Streaming techniques for popular video streaming services to mobile phones of three major platforms. The selection of a streaming technique does not depend on the wireless interface being used for, rather depends on the player, video quality, device and the video service provider.

	iPhone4S (iOS 5.0)	iPhone5 (iOS 7.0)	Galaxy S3/Galaxy S3 LTE (Android-4.0.4)		Lumia825 (WP8)
YouTube Streaming	*(App)* Throttling Factor = 2.0	*(App)* Throttling Factor = 1.25	*(Flash)* Encoding rate(HD), Throttling(<HD) Factor = 1.25	*(App& HTML5)* ON–OFF-M	(App) Fast Caching
Quality	LD(240p), SD(360p), HD(720p)	LD(240p), SD(360p), HD(720p)	LD(240p), SD(360, 480p), HD(720,1080p)	LD(240p), SD(360, 480p), HD(720p)	SD(270p), HD(720p)
Container	MP4(360,720p)	MP4(360,720p) 3GPP(240p)	XFLV	MP4(>240p) WebM(>240p) 3GPP(270p)	MP4(720p) 3GPP(270p)
Vimeo Streaming	*(App)* HLS Chunk Size = 10s	*(App)* ON–OFF-M	*(App)* ON–OFF-S		*(App)* Fast Caching
Quality	240–720p	SD(270, 480p), HD(720p)	SD(270p), HD(720p)		HD(720p)
Container	MP4	MP4	MP4		MP4
Dailymotion Streaming	*(App)* Throttling Factor = 1.25	*(App)* HSL Chunk Size = 10s	*(App)* Fast Caching(288p), ON–OFF-S(>288p)		*(App)* Throttling Factor = 1.25
Quality	LD(240)	240–720p	SD(288, 480p), HD(720p)		SD(288p)
Container	MP4	MP4	MP4		MP4
Netflix Streaming	*(App)* HLS Chunk Size = 10s	*(App)* HLS Chunk Size = 10s	*(App)* ON–OFF-S		*(App)* MSS Chunk Size = 4s
Quality	240–720p	240–720p	HD(720p)		240–720p
Container	isma, ismv	isma, ismv	MP4		isma, ismv

type of streaming technique used for each of the different combinations of device, service, stream quality, player type, and access network type manually. These findings are summarized in Table 3 and discussed below.

4.1. Fast caching

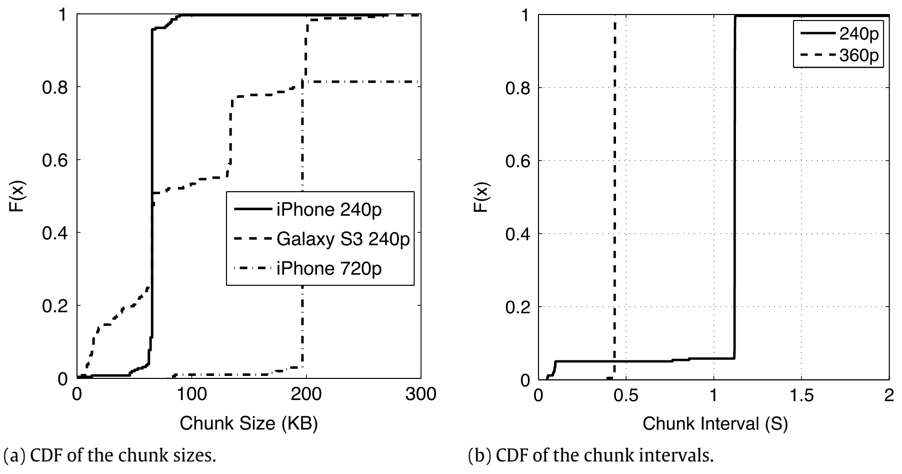
Fast caching is a server applied streaming mechanism, which refers to downloading the whole content as fast as possible at the very beginning of the streaming. The players continue playback and meanwhile they maintain very large playback buffer. YouTube Flash player uses `ratebypass=yes` parameter in the HTTP request to deactivate any rate control at the server side. The YouTube and Vimeo apps in Lumia825 also download videos using fast caching. In Android devices, only the Dailymotion player uses fast caching to download 288p quality videos.

4.2. Throttling

Throttling is another type of server-side streaming technique. In this case, the server sends content at a limited constant rate, which is higher than the encoding rate. The multiple of the encoding rate is referred to as the throttle factor. We observed this technique only with YouTube and Dailymotion. The throttle factor can vary depending on the video service or even on the player type for the same service. For instance, the native YouTube application receives content at a throttle factor of 2.0 in iPhone4S, whereas the Dailymotion application receives at a factor of 1. 25. The Flash player in Android devices and the native app in iPhone5 specify the throttle factor in the request URL (e.g., algorithm = throttle-factor and factor = 1.25) or a service specific default throttle factor is used.

4.2.1. Single TCP connection

In general, throttling is carried over a single TCP connection and the data is sent in small chunks. Fig. 5(a) shows that the YouTube player in iPhone4S receives a LD video in 64 kB chunks. This observation is similar to those explored in [11,3] for YouTube. The chunk size increases to 192 kB when receiving the same video of HD quality. We observed variable chunk sizes when streaming to Samsung Galaxy S3 (see Fig. 5(a)). However, these chunks are sent by the streaming servers at some periodic intervals to the streaming clients. The interval increases as the encoding rate or quality of the video decreases. Fig. 5(b) shows that the chunks are separated by few hundred milliseconds to 1.2 s. This burstiness is independent of the wireless interface being used for streaming. Nevertheless, this kind of burstiness was absent in Dailymotion and Vimeo streaming sessions.



(a) CDF of the chunk sizes.
(b) CDF of the chunk intervals.

Fig. 5. YouTube server sends content in small chunks and periodic manner when throttling the sending rate.

4.2.2. Multiple TCP connections

In iPhone4S, the YouTube player uses a large number of TCP connections to receive HD quality videos. In an example video streaming session, we found that the player downloaded a HD video in 66 connections. The player maintains a 25 MB size playback buffer. At the beginning, the player receives content at the throttled rate of 2. As the playback continues at the encoding rate, there is always some extra content in the buffer. Therefore, this playback buffer becomes full at some point and the player closes the existing TCP connection. Whenever some buffer is freed, the player initiates another HTTP partial content request over a new TCP connection.

In this way, the player actually receives more data from the server than the actual size of the content. Finamore et al. [4] also reported similar observation. From traffic traces, we identified that a YouTube server always sends media content from the beginning of a key frame[3] for any partial content request. The reason is that the player is unable to keep track of the ending position of the current key frame or the beginning of the next key frame. Therefore, it may terminate the connection when receiving a key frame. In addition, the player must support the forward and backward seeking during playback. Subsequently, each time the player requests content from the beginning of a key frame, which it has received partially for the previous request. As a result, the player wastes all the data of the partially received key frame. From traffic traces, we calculated that the player received 160 MB data in total for a 76 MB video.

4.3. Encoding rate streaming

Encoding rate technique is exclusively applied by the streaming clients. The server sends content using fast caching and the player has a small playback buffer. Therefore, the playback buffer and TCP receive buffer become full at the very beginning. Since the player decodes content at the encoding rate, the same amount of buffer is freed from the playback buffer and also from the TCP receive buffer. The client again can receive the same amount of content from the server. The mechanism is illustrated in Fig. 6. From Table 3, we can see that only the Flash player in Android devices receives HD videos from YouTube at the encoding rate.

4.4. Buffer adaptive streaming

Buffer adaptive techniques represent smart player implementations. The players maintain two thresholds of buffer level: a lower and an upper. During a streaming session, the player pauses downloading content when the playback buffer is filled to the upper threshold value. The player resumes downloading, when the buffer drains to the lower threshold. The video players apply buffer adaptation in two different ways and generate ON–OFF traffic pattern. Some video players apply the buffer adaptation over a single TCP connection. We refer this kind as ON–OFF-S. The others use multiple TCP connections and we refer as ON–OFF-M.

4.4.1. Single TCP connection (ON–OFF-S)

The native applications of Dailymotion, Vimeo, and Netflix video services apply buffer adaptation over a single TCP connection in the Android devices (see Table 3). The players pause reading from the TCP socket and an OFF period begins. Fig. 7(a) illustrates that TCP flow control packets are exchanged during an OFF period.

[3] https://www.princeton.edu/achaney/tmve/wiki100k/docs/Key_frame.html.

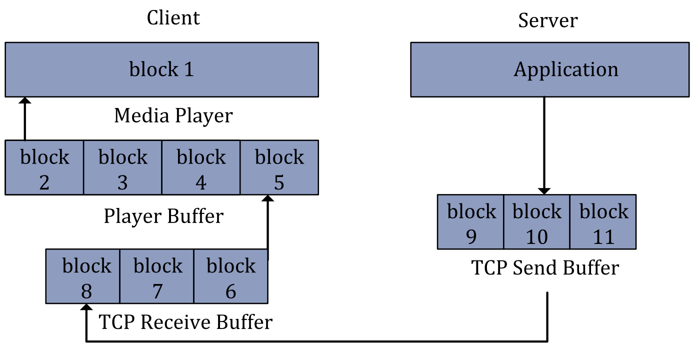
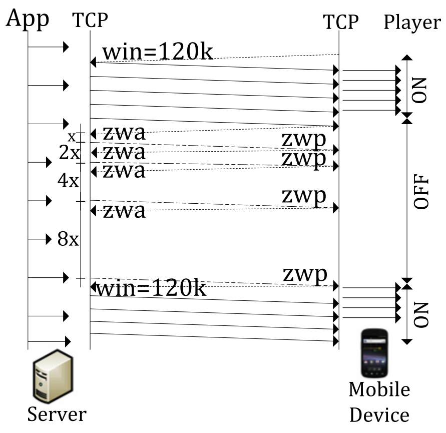
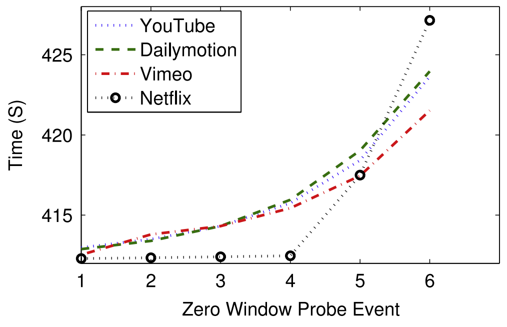


Fig. 6. Interaction between playback buffer and TCP receive buffer for encoding rate streaming.



(a) Buffer adaptive streaming over a single TCP connection activates TCP flow control during an OFF period.



(b) The growth of the TCP persist timer at the streaming servers during an OFF period.

Fig. 7. ON–OFF-S mechanism and interaction with TCP flow control.

The duration of an OFF period can be very long. The older Android devices (e.g., Samsung Nexus S) use an upper threshold of 5 MB [10]. Therefore, the duration of the OFF period is almost equivalent to the $\frac{5\text{ MB}}{\text{Encodingrate}}$ s. On the other hand, in latest devices the duration is $\frac{20\text{ MB}}{\text{Encodingrate}}$ s. However, from traffic traces we found that the TCP persist timer at the server grows only to maximum 5 s. The reason is that the players intentionally reset the persist timer after every 16 s by receiving 64 kB data from the server. This behavior was absent in the case of Netflix. Fig. 7(b) shows how the TCP persist timer values grow at the servers of different video streaming services. In the case of Netflix, the OFF period is always 30 s and the persist timer increases to maximum 10 s. Later in Section 6.3, we show how the TCP flow control messages and TCP persist timer affect the power consumption of smartphones.

4.4.2. Multiple TCP connections (ON–OFF-M)

Only the native app and HTML5 player for YouTube in Android devices use multiple TCP connections for buffer adaptation. The players maintain dynamic lower and upper thresholds of playback buffer. When the playback buffer is filled to the upper threshold, the player closes the TCP connection and an OFF period begins. The ON period begins after a fixed 60 s OFF period (see Fig. 8). The recent version of Vimeo player in iPhone5 also uses multiple TCP connections. Unlike the YouTube player, the Vimeo player downloads 30 MB during the Fast Start and downloads rest of the content in 5 MB chunks. Therefore, the duration of an OFF period is equal to $\frac{5\text{ MB}}{\text{Encodingrate}}$ s.

4.5. Rate adaptive streaming

The streaming techniques we discussed so far are for streaming constant quality content. The players or the servers cannot change the quality on the fly, unless the user interrupts the playback. On the other hand, Dynamic Adaptive Streaming over HTTP (DASH [12])-like rate adaptive mechanisms are able to change the quality on the fly for adapting with bandwidth fluctuations. The quality switching algorithms are implemented in the client players. A player estimates the bandwidth continuously and transitions to a lower or to a higher quality stream if the bandwidth permits. We identified two kinds of rate adaptive streaming; (i) HTTP Live Streaming (HLS) and (ii) Microsoft Smooth Streaming (MSS).

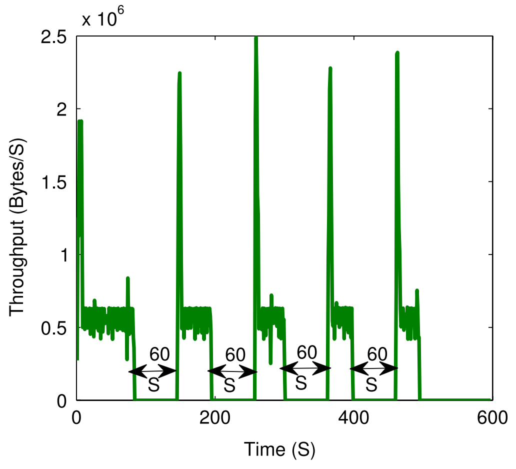


Fig. 8. The YouTube player in Galaxy S3 downloads a video by initiating multiple TCP connections.

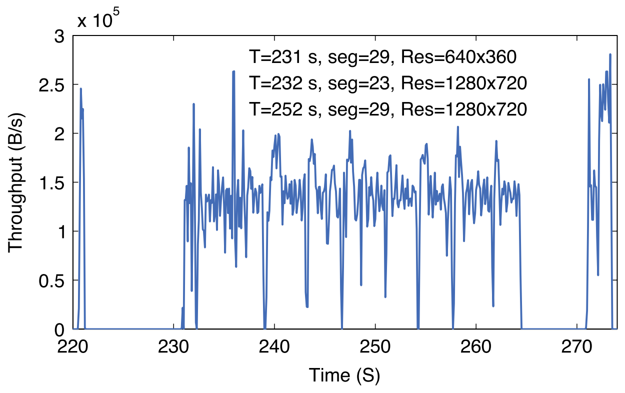


Fig. 9. The Vimeo player in iPhone4S, using HLS, discards content of low quality from the playback buffer upon switching to a higher quality.

4.5.1. *HTTP live streaming*

The Netflix and Vimeo players in iPhone4S, and the Dailymotion player in iPhone5 use HTTP Live Streaming and download content in 10 s chunks. At the beginning, a player receives the media description files, which contain the chunk duration, encoding rates and the bandwidth requirements for the chunk download. The player begins by downloading seven 10 s chunks of the SD quality. After that, the player downloads chunks after every ten seconds. In this way, the player always keeps 60 s playback content in the buffer when streaming via Wi-Fi. In the case of HSPA, the player keeps 20 s equivalent content in the buffer. This observation can change with bandwidth variation. However, in the case of transitioning to a higher quality, the player discards the downloaded lower quality content in order to provide instant response to the quality change to the user. One streaming scenario via Wi-Fi is illustrated in Fig. 9, where the player switches from a SD to HD quality at 232 s and downloads from 23rd to 29th chunks of HD quality.

Similarly, the Netflix player uses HLS in iPhones. However, the Netflix downloads the audio and video chunks separately, where the chunks are of 10 s. From multiple traces, we verified that the audio and video chunk downloading are not synchronized. Fig. 10 shows that after the Fast Start phase, the interval between an audio and a video chunk is approximately five seconds. There were also some cases where an audio chunk appeared very close to the next video chunk. Another interesting observation is that the server specifies its TCP parameters in the HTTP response header, as for example X-TCP-Info:rtt=11625;snd_cwnd=217201;rcv_wnd=1049800. The reason is likely that the streaming server lets the client player to calculate the bandwidth and to decide the quality accordingly.

4.5.2. *Microsoft smooth streaming*

The Netflix player in Lumia825 uses Microsoft's smooth streaming. The player receives video content in 4 s chunks over a single TCP connection. The same connection is used to receive audio content chunks also. However, the audio chunks are received after every sixteen seconds, i.e. after four consecutive video chunks. Unlike the Vimeo and Dailymotion rate adaptive players in iPhones, Netflix is aggressive in providing the highest quality of the stream in Lumia825. In traffic traces, we noticed that the player begins with the lowest quality, and then switches to the maximum quality within the first few seconds of streaming. During this period, the player downloads 60 s playback content. However, unlike the desktop

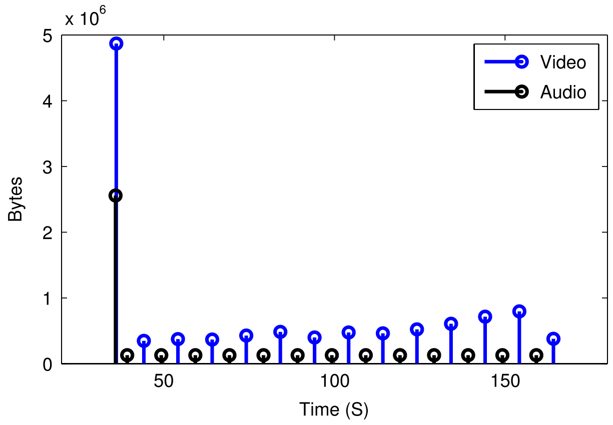


Fig. 10. The Netflix player in iPhone5, using HLS, downloads audio and video chunks asynchronously.

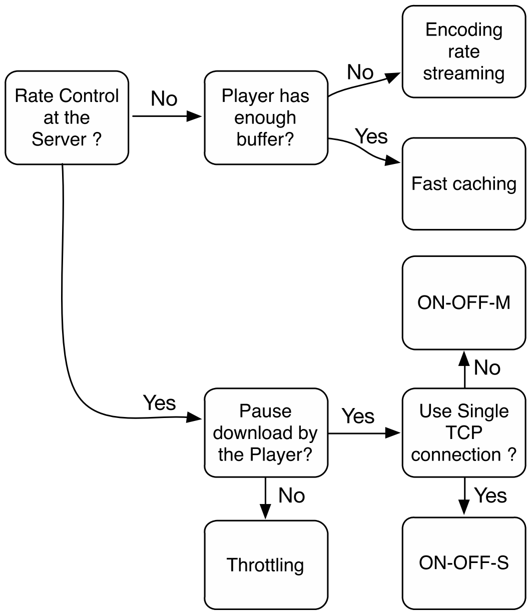

Rate Control at the Server ?
No
Player has enough buffer?
No
Encoding rate streaming
Yes
Fast caching
ON-OFF-M
No
Yes
Pause download by the Player?
Yes
Use Single TCP connection ?
No
Yes
Throttling
ON-OFF-S

Fig. 11. The choice of a streaming technique by the client player for constant bit rate streaming.

player [13], the mobile version requests different filenames for different qualities and specifies the byte range in the URL GET (abc).ismv/range/0-40140/. In response, the server sends the chunks of the corresponding quality. The server also sends a .bif file which contains information about the frames, which is used by the player upon forward or backward seeking by the user. We also found that the Netflix server sends TCP parameters to the player, which is mentioned in the earlier section.

4.6. Summary

Table 3 summarizes our findings on the usage of different techniques in different mobile platforms with four video services. Fig. 11 illustrates how the client app behavior leads to the choice of particular streaming technique for constant quality streaming. We sum up our main observations below:

- Streaming servers use either throttling or fast caching to deliver constant bit rate video to mobile devices. The choice between these two is influenced by client player's request. For instance, YouTube and Vimeo servers use both throttling and fast caching. The Dailymotion servers use throttling. Netflix servers use fast caching for constant bit rate streaming, and MSS or HSL for rate adaptive streaming. Some native mobile apps continuously pause and resume downloading leading to ON–OFF traffic patterns. Encoding rate streaming is the result of small playback buffer at the client buffer and fast caching streaming by the server.
- For constant bit rate streaming, the relevance of a technique depends on the mobile platforms to some extent. Buffer adaptive streaming is commonly used by all the video streaming services in the Android platforms. However, the only

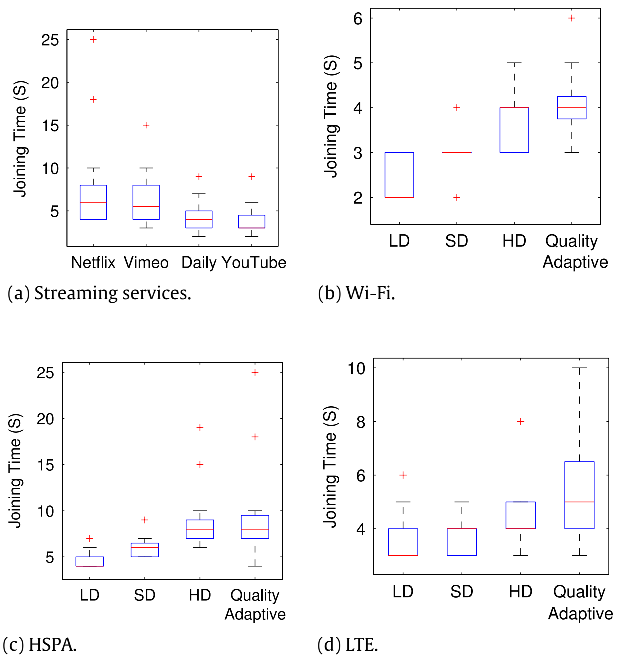


Fig. 12. Joining time observed for the video services and when streaming via wireless network interfaces.

exception is Dailymotion. The reason is that the videos are small in size and the throttling rate is also small compared with YouTube and Vimeo. Therefore, the player does not get enough buffer filled to apply the adaptation. Fast caching is prevalent only in Windows-based devices.

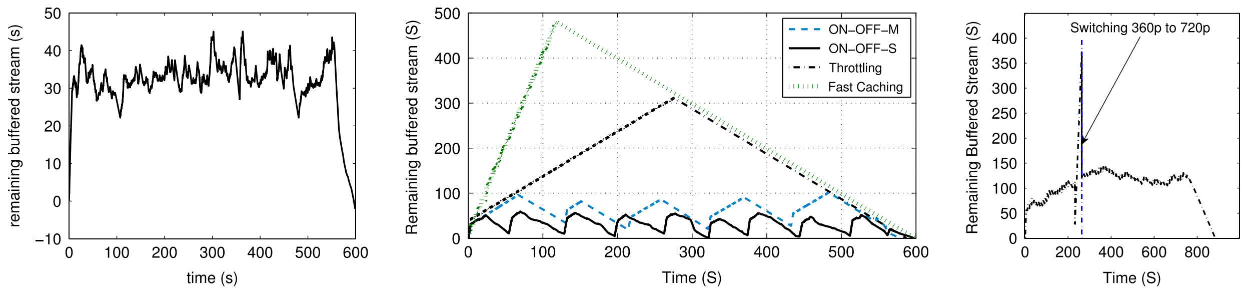
- None of the video services apply rate adaptive streaming in Android mobile devices. The Netflix, Vimeo, and Dailymotion players use HLS in iPhones. In iOS devices, Netflix receives audio and video chunks in separately streams. MSS is used only in Windows phones. These also reflect the influence of platforms on the choice of steaming techniques.
- Although the streaming strategies can vary based on the quality of the video, platforms, and video services, we could not find any evidence that the strategies vary according to the wireless interface being used for streaming.
- The amount of data wasted by the YouTube player in iPhone4S is significant even when a user watches the complete video. This problem could be solved with a smarter player implementation. The YouTube player in the latest iOS version solves this issue by using a smaller throttle factor of 1.25 than 2. As a result, the playback buffer never becomes full, and consequently, there is no data waste. However, potential data waste is also possible when the whole video is downloaded and the user abandons watching earlier.

5. Streaming techniques and quality of experience

The key metrics that characterize the QoE perceived by a user while streaming video are the initial playback delay which is called joining time, and the occurrence and frequency of playback pause events experienced [14,15]. We discussed earlier that the playback pause events are the results of bandwidth variation due to various network conditions. In this section, we take a look at the joining time and the performance of different strategies in providing smooth playback during short/long term bandwidth changes.

Fig. 12 shows the joining time experienced by the players according the video service and the WNI. Although, all the streaming services use fast start at the beginning of streaming, it is shown in Fig. 12(a) that the YouTube players take less time than the other players. The reason is that YouTube caching servers are extensively spread around the globe. Therefore, the content is served from the CDN that is very close to the user. We validated this by measuring the round-trip time from the captured traces. Our observation is similar to [14], in which the authors also proposed to serve content from the nearby CDN to improve the playback experience. However, in the case of Vimeo and Netflix two other facts also contribute in higher joining time. The Vimeo player always receives the HD quality video. The Netflix player maintains larger buffer threshold for initial playback and it receives the lowest video quality at the beginning. Therefore, the player spends longer time initially.

We explained earlier that quality of the stream affects the initial start-up time. The box plots in Fig. 12(b)–(d) illustrate the similar findings. There are two observations. First, streaming via Wi-Fi experiences less joining time than streaming via HSPA and LTE. The joining time is the largest when HSPA is used. We investigated and found that the wireless latency plays the role when streaming via HSPA and LTE. This is because, at the beginning of a streaming session, the HSPA interface transitions from IDLE/CELL_PCH to CELL_DCH state and the LTE interface switches from IDLE to the CONNECTED state. The transition latencies for LTE and HSPA are 120 ms and 2.0 s respectively. In the case of Wi-Fi, the transition latency from



(a) Encoding rate streaming and playback buffer status (in second).

(b) Playback buffer status when using throttling, buffer adaptive and fast caching.

(c) Rate adaptive streaming and playback buffer status.

Fig. 13. Playback buffer status of the streaming clients during multimedia streaming sessions using different techniques.

sleep to active state transition is few milliseconds which is negligible. The other observation is that the rate adaptive players experience more delay in the joining. This observation is biased because of the Netflix.

Next, we looked at the prefetching behavior of the players by studying how much content they maintain in the playback buffer throughout a streaming session. This analysis requires the time series of content consumption and arrival. The arrival time series is computed by extracting timestamps and payload size of received packets from the traffic traces considering the joining time. Although there are findings that YouTube-like video services stream constant bit rate content [16], we found that the video services use variable bit rate encoding for streaming HD videos. Hence, we replayed each video using a VLC player and extracted the instantaneous encoding rate of the content from VLC's web interface module using a shell script. Finally, we compute the amount of buffered content as a function of time by taking the difference of the cumulative sums of the arrival and consumption time series.

Fig. 13 shows the playback buffer status during the streaming sessions using different streaming techniques. Using encoding rate streaming, a player always keeps 30–40 s equivalent content in the playback buffer. Hence, even if the player receives content at the negligible rate after the fast start phase, the player can provide playback for that 30–40 s period. Throttling and fast caching continuously accumulate more content into the buffer and therefore are more robust also towards longer periods of low available bandwidth. From Fig. 13(b), we can see that when the playback is at 50 s, the player already has content for the next 50 s using throttling. In case of fast caching, the player has 200 s worth of content in the buffer. When using the ON–OFF strategies, the buffer is periodically filled up and drained in between. ON–OFF-M begins refilling the buffer 40 s earlier. A surprising result is that ON–OFF-S (in Android 2.3.6) nearly dries the buffer before new content is prefetched. Therefore, the possibility of playback starvation increases, when streaming via HSPA. The rate adaptive players maintain 60–100 s playback buffer, and at the same time they can select to a lower quality (see Fig. 13(c)). Nevertheless, the streaming strategies provide the best effort in guarding short term and long term bandwidth fluctuations.

6. Streaming services and power consumption

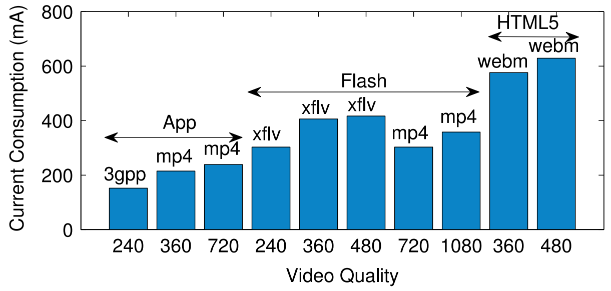
We also measured the total current consumed by the smartphones during video streaming sessions. We separated the total current drawn into the average video playback and streaming current consumption. The playback current includes decoding and display current. We can identify this current draw at the end of the power trace of each streaming session when the content has been fully delivered but playback still continues, since some of content is always buffered at the end regardless of streaming technique used. During this time, the WNIs are in the lowest power consuming states according to their own power savings protocols. We computed the average wireless communication current, which we refer to as streaming current, by subtracting the average playback current from the total current. The results presented in this section are the average of repeated measurements.

6.1. Playback power consumption

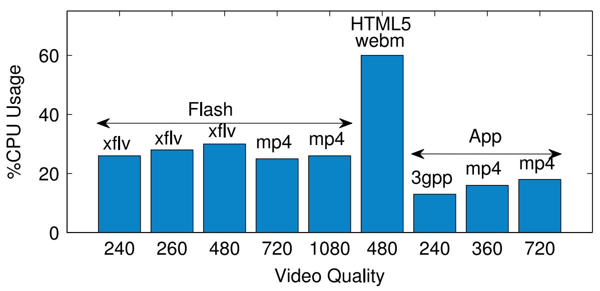
In our earlier work [10], we presented the playback power consumption of the smartphones. Our observation was that playback power consumption for playing a video of the same quality varies among different devices mostly because of the display variation. As the Android devices have all the player types and they can play the videos of all qualities and containers, in this section we discuss the results for GS3.

6.1.1. Video quality

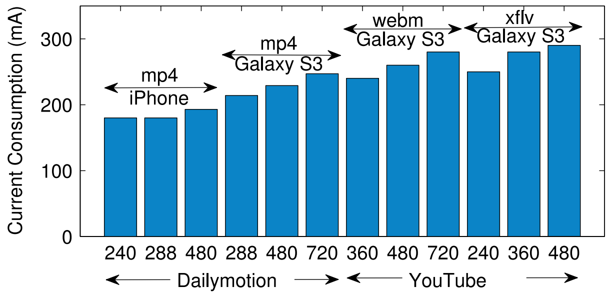
In Fig. 14(a), we can see that playback current drawn by Galaxy S3 increases as the quality of YouTube video increases of the same container type. We also observed similar pattern for watching Dailymotion videos in iPhone4S and Galaxy S3. It is logical that higher quality videos have more information to present than low quality videos and, therefore, more current is drawn. However, in some cases even doubling the resolution adds a relatively small increment to the average playback current.



(a) Avg. playback current draw when streaming 240–1080p YouTube videos to the app and browser in Galaxy S3.



(b) Amount of CPU used by different video players in Galaxy S3 while playing different quality videos of different containers.



(c) Avg. playback current consumption while playing different quality videos of different containers.

Fig. 14. Playback current consumption of Galaxy S3 and CPU usage with different qualities, players and containers.

6.1.2. Video player

For playing YouTube LD, SD and HD videos, the browser loads a Flash player. Flash supports different codecs and containers, such as X-FLV, MP4 and H.264. The browser loads HTML5 player to play WebM videos. Fig. 14(a) compares the energy consumption when using different players for streaming. It is noticeable that the native YouTube application consumes the least amount of energy. In contrast, browser-based players can draw even more than the double current when playing the same video. We discovered that during playback the Flash player does not leverage any native system support to decode the video but consumes a significant amount of more CPU than the native application (see Fig. 14(b)). Although the HTML5 player takes native system support, it consumes 60% of CPU even during the playback of a 480p video. It seems that HTML5 player is required to go through further optimization to be used in mobile platforms.

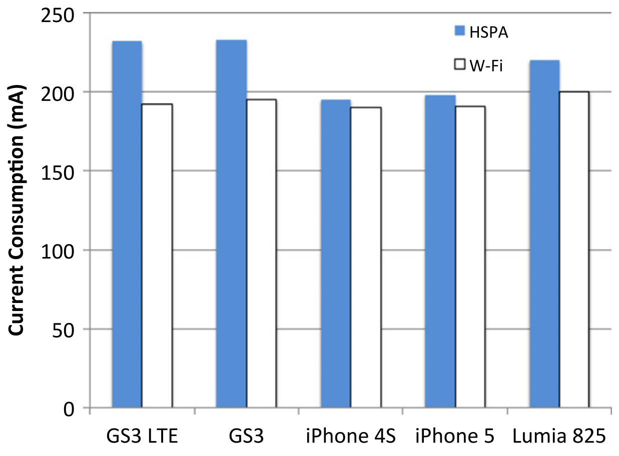
6.1.3. Video container

We already showed how the videos of different quality and different players affect the energy consumption of smartphones. In Fig. 14(a), we can see that playback of a 240p 3GPP video requires less energy than that of an X-FLV video of the same quality. It is also illustrated that the same 240p X-FLV requires more current than a 720p MP4 video. Although from Fig. 14(a) we can infer that 3GPP is the least and WebM is the most energy consuming containers, it is difficult to isolate the effect of the corresponding video containers since some videos can be played only using browsers. Besides, the energy consumption of the browser-based players are very high. Therefore, we downloaded some YouTube videos of X-FLV and WebM formats and then measured energy consumption during playback. The results are shown in Fig. 14(c). This figure also illustrates that playback energy consumption does not change significantly when the quality of video changes with the same container category.

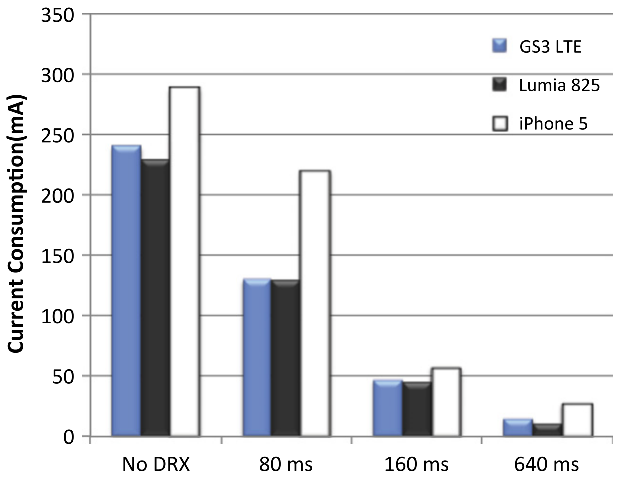
6.2. Device variation

Before discussing the impact of different streaming strategies on the streaming power consumption, we investigate the power consumption of individual WNI in smartphones. In Section 2.1, we described the standard power saving mechanisms applied by different WNIs. We also discussed that there are a number of states and a mobile device consumes different amount of energy in different states. Consequently, we explore what kind of power saving mechanism are applied by our target smartphones and the variation among them in consuming energy.

In Fig. 15(a), we can see that iPhones consume the lowest current when Wi-Fi is active. Android devices consume more current when the Wi-Fi interface is active, whereas the Lumia825 consumes the maximum current. However, all of them use PSM adaptive. iPhones use an aggressive idle period of 50 ms and the remaining devices use 200 ms. The current consumption during these idle periods is half of the active state current consumption. Therefore, iPhones consume the



(a) Wi-Fi and HSPA.



(b) LTE.

Fig. 15. Current consumption of wireless network interfaces in smartphones.

lowest energy and Lumia825 consumes the maximum as Wi-Fi tail energy. Fig. 15(a) shows the current consumption of HSPA interface during data transfer in CELL_DCH state. In this case also, iPhones consume the lowest energy and Lumia825 is the second least. On the other hand, the Android devices consume the maximum energy. However, all the devices use Fast Dormancy with an inactivity timer of 5 s, except iPhone5 which uses an inactivity timer of 8 s. Therefore, iPhone5 spends more tail energy than the others when HSPA is used.

We measured the current consumed by the mobile devices when the LTE interface is active with four different network configurations; DRX is disabled, DRX is enabled with a short DRX cycle (80 ms), with DRX cycles of 160 ms and 640 ms respectively. From the results presented in Fig. 15(b), we find that the smartphones consume the maximum energy when DRX is not enabled in the network. Among the all, iPhone5 consumes the maximum tail energy, because it consumes the highest current when DRX is disabled. If DRX is enabled in the network, the smartphones consume comparatively less energy. This is because the devices periodically wake up to check data activity according to the DRX cycles in the connected state. This Figure also depicts that Lumia825 consumes the lowest current when LTE is active.

Fig. 15(b) also depicts that iPhone5 is the most and Lumia825 is the least energy consuming device when DRX is enabled. From power traces we identified that even though the DRX_{on} was configured to 10 ms, iPhone5 spends 60 ms. On the other hand, Lumia825 and GS3 LTE spend 30 and 45 ms respectively in the on period of the DRX cycle. From Fig. 15(b), we can also see that the devices consume more current when the cycle lengths are shorter. For instance, when DRX cycle is of 80 ms, GS3 LTE and Lumia825 consume around 120 mA current. If the cycle length is increased to 640 ms, the power consumption decreases by a factor of three approximately. The first reason is that when short DRX cycles are in action, a mobile device will spend more time in the on period of the cycles as there will be more cycles when the RRC inactivity timer is active. Second, the LTE chipset is not optimized yet to operate on such small cycles. They cannot efficiently shutdown the power consumption during the DRX sleep phase. Fig. 16(a) shows that current consumption of GS3 LTE is stable at ≈220 mA from 132 to 142 s even though the DRX is active. Current consumption during short DRX cycles does not scale down like when DRX

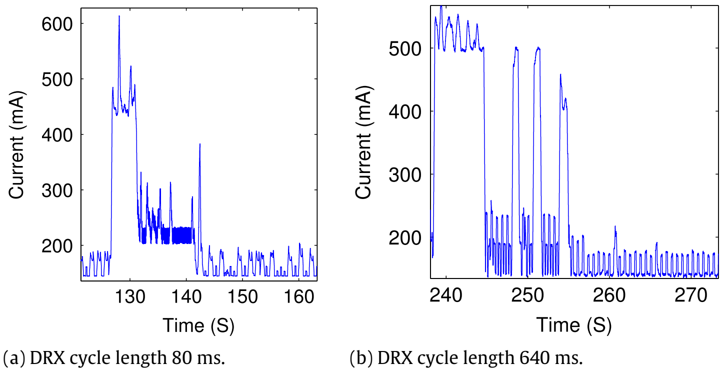


Fig. 16. Current consumption of GS3 LTE with different DRX cycles.

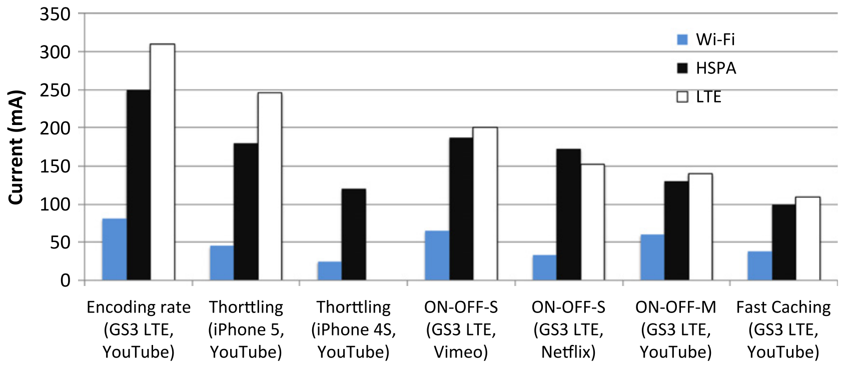


Fig. 17. Avg. streaming current consumption of smartphones when streaming a 600 s long constant bit rate video using the streaming strategies.

cycle is of 640 ms (from 245 to 255 s in Fig. 16(b)). This pattern is also consistent with iPhone5 and Lumia825 (Fig. 15(b)). However, again iPhone5 spends the maximum tail energy, even though the DRX is enabled with any configured DRX cycle length.

6.3. *Impact of streaming techniques*

In the previous section, we showed the basic power consumption characteristics of different WNI. In this section, we discuss the effect of streaming techniques on the energy consumption in smartphones. Since all the techniques are not available in a single mobile platform, it is difficult to compare the energy efficiency of the techniques. Therefore, we compare only the current consumed by the wireless interfaces of the smartphones and exclude the playback current in order to provide a comparison ground. In the case of LTE, the DRX was enabled in the network and we used a single DRX profile with DRX cycle of 80 ms, as this profile is used by the network operators in Finland. We compare them in Fig. 17.

6.3.1. *Encoding rate streaming*

In this case, the content is delivered continuously throughout the entire streaming session and the wireless interface is active all the time. For example, downloading a 6 min video would require approximately six minutes. As a consequence, the average streaming current drawn by Galaxy S3 LTE is very high for the YouTube videos. Fig. 17 also shows that Galaxy S3 LTE (GS3 LTE) consumes around 77 mA for Wi-Fi, 200 mA and 310 mA for HSPA and LTE respectively (HD video using browser). The high current consumption of HSPA/LTE is expected, since these interfaces are constantly in the highest power consuming state. However, power consumption over Wi-Fi is low with respect to the usage of the interface. This is because, the Android devices use Dynamic Voltage and Frequency Scaling (DVFS) when streaming via Wi-Fi.

6.3.2. *Throttling*

In Section 4.2, we discussed that in case of throttling, the throttle factor defines the amount of time is used to deliver the content to the client players. The higher is the throttle factor, the lower is the time required at the client to download the content. Therefore, this factor also determines the amount of time the wireless radio will be powered on and hence it also determines power consumption at smartphones. Energy consumption for two throttled sessions is presented in Fig. 17. In the first case, the server uses the throttle factor 1.25 for iPhone5. The second session is for iPhone4S, where the factor is 2. iPhone5 consumes more current than iPhone4S for streaming via Wi-Fi and 3G. The obvious reason is that iPhone4S

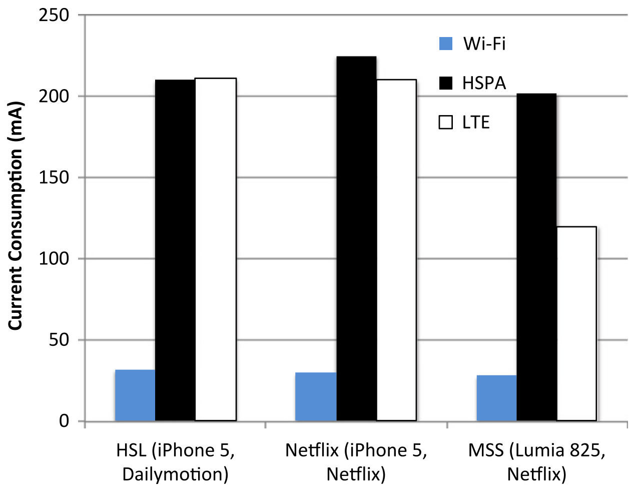


Fig. 18. Avg. streaming current consumption of smartphones for rate adaptive streaming techniques, HTTP live streaming, Microsoft smooth streaming and Netflix's own adaptive mechanism in iPhone5.

downloads at a faster rate. And both smartphones consume less current than the GS3 LTE which downloads video at the encoding rate. Therefore, throttling delivers energy savings over encoding rate streaming as interface usage time is reduced.

6.3.3. *Buffer adaptive streaming*

In Fig. 17, we can see that GS3 LTE consumes more current in streaming a Vimeo video than the Netflix video via any WNI. This is because of the player behavior in resetting TCP persist timer. We described in Section 4.4.1 that the Vimeo player resets TCP persist timer after every 16 s and the maximum interval between TCP control packets from Vimeo can be 5 s. On the other hand, the Netflix player rests after every 30 s and the maximum interval between TCP control packets from Netflix is 10 s. Therefore, the interfaces can spend more time in low power consuming states when streaming from Netflix than streaming from Vimeo. However, the average streaming current consumption is less than the encoding rate streaming.

Fig. 17 also includes a case where GS3 LTE receives content from YouTube in multiple TCP connections. Since the duration of such an OFF period is 60 s, the wireless interfaces can be in sleep or the lowest power consuming states for very long time. As a result, GS3 LTE consumes roughly 50% less energy when using ON–OFF-M than the encoding rate. However, it can be seen that ON–OFF-M does not outperform throttling (iPhone4S) in current consumption as the player receives content at the same throttled rate in each TCP connection.

6.3.4. *Fast caching*

Fast caching is used to download content at the client with as high throughput as possible. As a result the wireless interface is maximally utilized for as little time as possible. Fig. 17 shows that GS3 LTE consumes the least current, if the YouTube player downloads the whole video using fast caching.

6.3.5. *Rate adaptive streaming*

Similar to the ON–OFF-M mechanism, the quality or rate adaptive players also receive content in chunks over a single or multiple TCP connections. The duration of a chunk varies from a minimum four seconds to maximum ten seconds depending on the service. Fig. 18 shows the current consumption of WNIs when streaming Netflix and Dailymotion videos in iPhone5 and Lumia825. Both devices consume about 30 mA streaming current for Wi-Fi. The players in iPhone5 receive content in 10 s chunks. Therefore, the HSPA interface avails the lower states rarely as the FD timer is 8 s and consequently current consumption is high. The LTE interface also consumes significant current even though the DRX is enabled. This is because, the LTE interface in the iPhone5 takes long time in the ON period of the DRX cycle. iPhone5 consumes more current when streaming from Netflix than from Dailymotion via cellular networks. The reason is that the Netflix player downloads audio and video chunks separately and their downloading is not synchronized. Compared with iPhone5, Lumia825 consumes less current when the Netflix player streams via LTE as the interface spends lesser time in the ON state of the DRX cycles when DRX is active.

6.4. *Summary*

From Section 6.1, we learned that native apps are the most energy efficient. Since, HTML5 is an important technology at this moment, optimizing the HTML5-based player implementations would be an important future work. We also noticed that video container/codec also has significant impact on the energy consumption (3GPP seems more efficient than MP4), while video quality has a small impact. Therefore, the focus should be choosing an optimal codec or container.

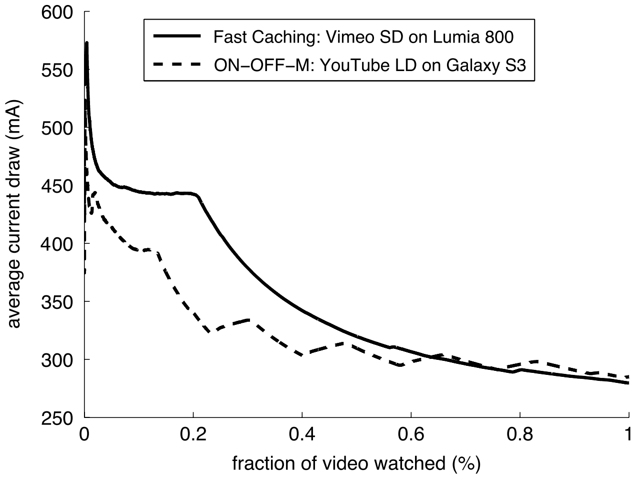


Fig. 19. Average draw of current as a function of viewing time for HSPA access.

Concerning the current consumption of wireless network interfaces, Wi-Fi is the least and LTE is the most energy consuming interface. When using LTE, the smartphones are not optimized yet for 80–160 ms or smaller DRX cycles. Therefore, the network operators should use longer DRX cycles in the network to improve the battery life time of smartphones.

The power consumption of WNI increases as the streaming rate increases [10]. However, the important lesson concerning the different streaming techniques is that encoding rate streaming causes clearly the largest amount of energy consumption. Fast caching is the most energy efficient technique. An effective ON–OFF-M technique should deliver content without any rate control. Although the rate adaptive techniques are similar to ON–OFF-M, higher chunk size and synchronization between audio/video chunks would reduce energy consumption significantly.

7. QoE and energy consumption tradeoffs

In Section 5, our measurement results revealed that most of the video services use optimized methods so that streaming quality does not deteriorate user experience by enabling the players in providing uninterrupted playback as long as possible. From this perspective, fast caching and throttling are the most efficient techniques. However, if the user does not watch the whole video, the downloaded data is wasted. Furthermore, using the cellular access to download unnecessarily content is problematic for users having small quota in their data plan and for the network resources. For example, Finamore et al. analyzed YouTube traffic to desktop computers and iOS devices accessed via Wi-Fi and discovered that 60% of videos were watched for less than 20% of their duration [4]. Therefore, ON–OFF mechanisms are attractive considering the unnecessary content download.

From the energy consumption point of view, the downloading energy is also wasted to retrieve the unwanted content. In Fig. 19, we plot the average current draw for fast caching and ON–OFF-M techniques as a function of percentage of watched video computed out of the complete power traces. We see that abandoning the video watching early on would cause a hefty penalty in terms of wasted energy in both cases but the penalty gets smaller faster with the ON–OFF-M streaming making it a more attractive technique, since it is common not to watch the video completely.

Since ON–OFF-M is the balanced technique in providing both less data waste and less energy consumption, a tradeoff between the buffer thresholds and energy consumption must be understood. Assuming that the upper threshold is fixed, i.e. the player allocates a fixed amount of memory for the playback buffer in the beginning of a streaming session, the lower threshold determines how large chunks of content will be downloaded at a time, i.e. what is the duration and frequency of the ON periods. The lower the threshold, the less frequent are the buffer refill events (ON periods), and the less power is consumed on the average. On the other hand, the lower threshold is set, the higher is also the chance that there is a playback pause event when the buffer refilling begins in case a transient period of low bandwidth happens to coincide. For this reason, there is a tradeoff between risking a buffer underrun event and the power consumption which is controlled by the lower buffer threshold.

In Fig. 20, we plot the average power draw as a function of the dynamic buffer size. The dynamic buffer size is directly determined by the lower threshold if we keep the upper threshold fixed. We notice that if there is plenty of spare bandwidth available compared to the stream encoding rate, then the buffer size should be set at least to a value around 40–50 s, but setting the buffer to a larger value than that no longer reduces the power consumption significantly.

The current YouTube players in Android that use the ON–OFF-M strategy set the upper threshold to a value equaling 100 s $\times$ r_s and the lower one to 40 s $\times$ r_s where r_s is the average encoding rate. These thresholds translate to a 60 s dynamic buffer size which, in light of Fig. 20, strikes a good balance. Those players using ON–OFF-S technique in newer versions of Android use a 20 MB buffer size. Assuming a lower threshold at zero, the dynamic buffer size would translate to 400 s and

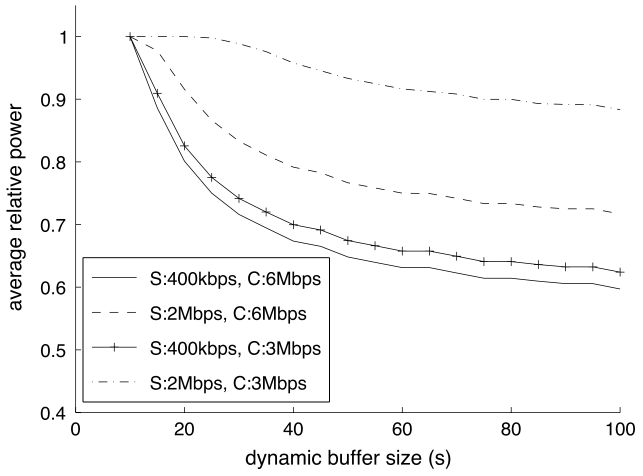


Fig. 20. Relative power draw as a function of dynamic buffer size for HSPA access. S is the stream encoding rate and C is the available bandwidth to download content.

80 s for videos having encoding rate of 400 kbps and 2 Mbps, respectively. With the higher quality video, the lower threshold could be set to 30–40 s $\times$ r_s in order to safeguard from buffer underrun events, and that configuration would still provide good energy efficiency when using HSPA.

8. Related work

The diverse nature of existing popular mobile streaming services in delivering better user experience, and the resulting energy consumption characteristics have so far not been completely uncovered. Krishnan et al. [17] studied the effect of initial joining time and playback pause events on the engagement in watching videos for fixed host users. Their findings were such that users cannot tolerate more than 2 s of joining delay and if a pause event persists more than 1% of total duration of the video the engagement decreases.

Another related work in [18] provides some insights into mapping the considered QoE metrics to user behavior through a utility function. They conclude that join time has relatively little impact on viewing time, whereas buffering ratio is highly correlated. To the best of our knowledge, their dataset did not include mobile clients. Their follow-up work in [15] further develops predictive model. However, based on that work it is impossible to assess the impact of a single QoE metric on the user behavior. Unfortunately, it is not meaningful to directly analyze the buffering ratio in our measurements. Such analysis would require a much richer dataset representing diverse network conditions and client locations in order to allow drawing conclusions concerning a particular video streaming service. This is because the buffering events are naturally often caused by lack of bandwidth rather than server capacity. Hence, our analysis is necessarily qualitative and limited to comparison of the different techniques and their potential to increase or decrease the probability of buffering events.

Many papers have studied the energy efficiency of multimedia streaming over Wi-Fi and developed custom protocols or scheduling mechanisms to optimize the behavior. Examples of such work range from proxy based traffic shaping and scheduling to traffic prediction and adaptive buffer management [2]. However, streaming over HSPA and the specific nature of the streaming services and client apps provide new challenges that these solutions cannot overcome. Balasubramanian et al. [19] studied 3G power characteristics in general and quantified the so called tail energy concept.

The most popular streaming services, especially YouTube, have been subject to numerous measurement studies in recent few years. Xiao et al. [20] measured the energy consumption of different Symbian based Nokia devices while using a YouTube application over both Wi-Fi and 3G access. A similar study was done by Trestian et al. [21] for Android platform. They investigated energy consumption while streaming over Wi-Fi at different network conditions and studied the effect of video quality on energy consumption. However, these studies did not consider the details of traffic patterns and their impact on the energy consumption.

In a measurement study, Rao et al. [3] studied YouTube and Netflix traffic to different smartphones (iOS and Android) and web browsers accessed via Wi-Fi interface. They found three different traffic patterns of YouTube. In a similar passive measurement study, Finamore et al. [4] also analyzed YouTube traffic to PCs and iOS devices accessed via Wi-Fi and demonstrated that iPhone and iPad employ chunk based streaming. Qian et al. [22] explored RRC state machine settings in terms of inactivity timers using real network traces from different operators and proposed a traffic shaping solution for YouTube which closely resembles the ON–OFF streaming technique.

Liu et al. [23] studied power consumption of different streaming services. However, the scope of their study is considerably different from ours. They limit their study to streaming over Wi-Fi and performed experiments with only iPod, while we explored all the major mobile platforms and contrasted Wi-Fi and HSPA energy consumption in [10].

In contrast to these studies, our contributions are the followings. (i) We investigated the traffic pattern of the streaming techniques and the characteristics which influence the choice of a streaming technique. (ii) We measured the initial joining time that varies according to the service, quality of the content and wireless access. (iii) We examined the playback buffer status of the players during playback to understand to which extent they can avoid a playback pause event in case of spurious network condition. (iv) We also studied the impact of the streaming techniques on the energy consumption on different smartphones using Wi-Fi, HSPA and LTE. (v) Finally, we proposed playback buffer configurations for ON–OFF mechanism, which can ensure significant energy savings, reduce data waste, and can tolerate bandwidth fluctuations to some moderate extent.

9. Conclusions

We analyzed the performance of four video streaming services in providing smooth playback at the smartphones. At the same time, we also measured the energy consumption of mobile devices. Based on the measurement results, we identified five different streaming techniques. The used technique depends on the service, client device or mobile platform, player type, and video quality. In general, most of the techniques are efficient in tolerating short term and long term bandwidth fluctuations by prefetching content. Since an interrupted video session can result in significant data and energy waste, ON–OFF-M provides a balance between quality of experience, and data or energy waste. We investigated how the buffer underrun and energy consumption are related and showed the optimal buffer threshold configurations with which a player can tolerate bandwidth fluctuation for 30 s to 1 min, at the same time reducing data waste and saving energy.

References

[1] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck, X. Zhang, Delving into Internet streaming media delivery: a quality and resource utilization perspective, in: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, IMC'06, ACM, New York, NY, USA, 2006, pp. 217–230.

[2] M.A. Hoque, M. Siekkinen, J.K. Nurminen, Energy efficient multimedia streaming to mobile devices—a survey, IEEE Commun. Surv. Tutor. 16 (1) (2014) 579–597.

[3] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, W. Dabbous, Network characteristics of video streaming traffic, in: Proceedings of the Seventh Conference on Emerging Networking Experiments and Technologies, CoNEXT'11, ACM, New York, NY, USA, 2011, pp. 25:1–25:12.

[4] A. Finamore, M. Mellia, M.M. Munafò, R. Torres, S.G. Rao, YouTube everywhere: impact of device and infrastructure synergies on user experience, in: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, IMC'11, ACM, New York, NY, USA, 2011, pp. 345–360.

[5] J. Erman, A. Gerber, K.K. Ramadrishnan, S. Sen, O. Spatscheck, Over the top video: the gorilla in cellular networks, in: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, IMC '11, ACM, New York, NY, USA, 2011, pp. 127–136.

[6] E. Tan, L. Guo, S. Chen, X. Zhang, Psm-throttling: minimizing energy consumption for bulk data communications in wlans, in: Proceedings of the IEEE International Conference on Network Protocols, ICNP2007, IEEE, Beijing, China, 2007, pp. 123–132.

[7] Fast Dormancy, Fast dormancy best practices. GSM association, network efficiency task force, 2010.

[8] M. Siekkinen, M.A. Hoque, J.K. Nurminen, M. Aalto, Streaming over 3G and LTE: how to save smartphone energy in radio access network-friendly way, in: 5th ACM Workshop on Mobile Video, MoVid'13, ACM, 2013, pp. 1–6.

[9] N. Balasubramanian, A. Balasubramanian, A. Venkataramani, Energy consumption in mobile phones: a measurement study and implications for network applications, in: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, IMC'09, ACM, New York, NY, USA, 2009, pp. 280–293.

[10] M. Hoque, M. Siekkinen, J.K. Nurminen, M. Aalto, Dissecting mobile video services: an energy consumption perspective, in: Proceedings of the 14th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM'13, IEEE, 2013.

[11] S. Alcock, R. Nelson, Application flow control in YouTube video streams, SIGCOMM Comput. Commun. Rev. 41 (2) (2011) 24–30.

[12] T. Stockhammer, Dynamic adaptive streaming over http –: standards and design principles, in: Proceedings of the Second Annual ACM Conference on Multimedia Systems, MMSys'11, ACM, New York, NY, USA, 2011, pp. 133–144.

[13] S. Akhshabi, S. Narayanaswamy, A.C. Begen, C. Dovrolis, An experimental evaluation of rate-adaptive video players over http, Image Commun. 27 (4) (2012) 271–287.

[14] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, H. Zhang, A quest for an Internet video quality-of-experience metric, in: Proceedings of the 11th ACM Workshop on Hot Topics in Networks, HotNets-XI, New York, NY, USA, 2012, pp. 97–102.

[15] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, H. Zhang, Developing a predictive model of quality of experience for Internet video, in: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, SIGCOMM'13, ACM, New York, NY, USA, 2013, pp. 339–350.

[16] X. Cheng, J. Liu, C. Dale, Understanding the characteristics of Internet short video sharing: a YouTube-based measurement study, IEEE Trans. Multimedia 15 (5) (2013) 1184–1194.

[17] S.S. Krishnan, R.K. Sitaraman, Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs, in: Proceedings of the 2012 ACM Conference on Internet Measurement Conference, IMC'12, ACM, New York, NY, USA, 2012, pp. 211–224.

[18] F. Dobrian, A. Awan, D. Joseph, A. Ganjam, J. Zhan, V. Sekar, I. Stoica, H. Zhang, Understanding the impact of video quality on user engagement, Commun. ACM 56 (3) (2013) 91–99.

[19] N. Balasubramanian, A. Balasubramanian, A. Venkataramani, Energy consumption in mobile phones: a measurement study and implications for network applications, in: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, IMC'09, ACM, New York, NY, USA, 2009, pp. 280–293. http://dx.doi.org/10.1145/1644893.1644927.

[20] Y. Xiao, R.S. Kalyanaraman, A. Yla-Jaaski, Energy consumption of mobile YouTube: quantitative measurement and analysis, in: Proceedings of the 2008 The Second International Conference on Next Generation Mobile Applications, Services, and Technologies, 2008, pp. 61–69.

[21] R. Trestian, A.-N. Moldovan, O. Ormond, G.-M. Muntean, Energy consumption analysis of video streaming to android mobile devices, in: Proceedings of the Network Operations and Management Symposium (NOMS), IEEE, 2012, pp. 444–452.

[22] F. Qian, Z. Wang, A. Gerber, Z.M. Mao, S. Sen, O. Spatscheck, Characterizing radio resource allocation for 3G networks, in: Proceedings of IMC 2010, ACM, New York, NY, USA, 2010, pp. 137–150.

[23] Y. Liu, L. Guo, F. Li, S. Chen, An empirical evaluation of battery power consumption for streaming data transmission to mobile devices, in: Proceedings of the 19th ACM International Conference on Multimedia, MM'11, ACM, New York, NY, USA, 2011, pp. 473–482.