An Empirical Evaluation of Battery Power Consumption for Streaming Data Transmission to Mobile Devices*

Yao Liu[1] Lei Guo[2] Fei Li[1] Songqing Chen[1]
[1]Department of Computer Science George Mason University
[2]Microsoft Corporation Mountain View, CA, USA
{yliud, lifei, sqchen}@cs.gmu.edu leguo@microsoft.com

ABSTRACT

Internet streaming applications are becoming increasingly popular on mobile devices. However, receiving streaming services on mobile devices is often constrained by their limited battery power supply. Various techniques have been proposed to save battery power consumption on mobile devices, mainly focusing on how much data to transmit and how to transmit.

In this paper, we conduct an experiment-based study with 11 Internet streaming applications using different streaming protocols. Our goal is to empirically investigate the battery power consumption on the wireless network interface for receiving streaming data via different approaches. Through measurement and analysis, we find that (1) the Chunk-based streaming is widely used in practice and it is most power-efficient because the traffic shaping technique is adopted to utilize PSM on mobile devices to save battery power consumption; however, it may cause quality degradation from time to time; (2) reducing streaming data transmission (by switching to a lower streaming quality) can marginally help save battery power consumption in RTSP, Pseudo streaming, and Chunk-based streaming applications; but it is effective for P2P streaming applications; (3) P2P streaming to mobile devices is not power-efficient because of the additional transmission of control traffic and uploading traffic; and reducing upload alone does not help for battery power saving. Our investigation provides new insights and some guidelines for the current Internet mobile streaming services and calls for further research on more power-efficient and scalable Internet mobile streaming protocols.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications

General Terms

Measurement, Experimentation

Keywords

Internet Mobile Streaming, Battery Power Consumption, Power Saving, Data Transmission

*Area chair: Wei-Tsang Ooi

MM'11, November 28–December 1, 2011, Scottsdale, Arizona, USA.

1. INTRODUCTION

Internet streaming to mobile devices is gaining increasingly popularity in practice. For example, CTV [1], CCTV [2], WTV [3], ImgoTV [4], and NHKworld [5] all allow iOS users to access their live TV programming content via 3G or WiFi network. SPBtv [6] supports more diverse platforms including iOS, Android, Windows Mobile, and Blackberry. It also provides a great variety of live TV channels from different countries. Orb [7] and AirVideo [8] allow users to access media content stored on their home computers. Justin.tv [9] even enables users to watch live streaming content broadcasted by other users. Many cellular network providers in the US today also provide streaming services to their subscribers, including Verizon's VCast [10] and AT&T's Mobile TV [11].

Different from their counterpart of streaming to desktop computers, Internet streaming to mobile devices is even more challenging because of the variety of the mobile platforms and operating systems, less reliable wireless connections, slower CPU, limited battery supply, etc. Among these factors, the limited battery supply is a critical constraint because most of the time mobile devices are battery-powered. Since the network transmission could consume a significant portion of the battery power, power saving on the wireless network interface card (WNIC) has been a major direction of many research efforts. The battery consumption of the WNIC mainly depends on the following two aspects: (1) how much data is transmitted during the streaming session? (2) how is such data transmitted?

The first aspect is very intuitive. Under the same circumstances of all other factors, transmitting more data would make the WNIC work longer and thus consume more battery power. In practice, a user can choose a low quality version of the same video clip and expects less battery power consumption. This is easy to do because the current practice from the industry often provides multiple versions of the same video clip with different quality (encoding rate) for users to choose from.

While choosing a low quality version is simple for end-users and only requires server support in advance, how to transmit data in order to reduce battery power consumption is much more complicated. Since today most of the commodity network interface card is supported with the IEEE 802.11 power saving mode (PSM) by default, it is straightforward that the PSM should be utilized to minimize the battery power consumption during streaming services. However, the standard PSM has limited effectiveness for streaming applications since streaming applications often involve bulk data transmission in a continuous fashion, which would not provide many opportunities for the WNIC to switch into PSM. Thus, plenty of research has been conducted in order to make the WNIC to sleep as much as possible for streaming to mobile devices and a variety of power saving schemes have been proposed. For example, PSM-

throttling [12] proposes a client-centric scheme to enable buffering at the server so that streaming data could be sent in burst in order to maximize the sleep time of the WNIC. Proxy buffering [13] suggests to buffer packets at a local intermediate proxy and send them in client-specific intervals. The key idea of these schemes is to shape the traffic so that they can be sent in burst without degrading the streaming quality. They mainly target client/server based streaming services. With the popularity of P2P based Internet streaming, SCAP [14] has also been designed to leverage the temporal locality of uploading and downloading data at the access point so that less data would be required to upload from a mobile client.

In this study, we aim to evaluate the battery power consumption for Internet streaming data transmission to mobile devices in the current practice. For this purpose, we conduct Internet experiments with 11 existing Internet streaming services. These experiments examine the impact of how streaming data is transmitted and how much data is transmitted during these streaming sessions. Our results indicate that (1) the Chunk-based streaming is widely used in practice and it is most power-efficient because the traffic shaping technique is adopted to utilize the PSM on mobile devices to save battery power consumption; however, it may cause quality degradation from time to time; (2) reducing streaming data transmission (by switching to a lower streaming quality) can marginally help save battery power consumption in RTSP, Pseudo streaming, and Chunk-based streaming applications; but it is effective for P2P streaming applications; (3) P2P streaming to mobile devices is not power-efficient because of the additional transmission of control traffic and uploading traffic; and reducing upload alone does not help for battery power saving.

Our measurement results indicate that there is no panacea for power saving in receiving Internet streaming on mobile devices and further research is desired to improve the power efficiency in receiving Internet streaming applications. A particular challenge comes from P2P based Internet streaming, in which neither traffic shaping helps, nor P2P specific schemes can provide sufficient battery power saving benefit.

In summary, we have made the following contributions:

- Through measurement and analysis, we have provided an update of the battery power efficiency on mobile devices in receiving Internet streaming services.
- We have qualitatively and quantitatively investigated the four approaches for streaming to mobile devices, compared their power consumption characteristics, and analyzed the underlying reasons.
- Our results provide new insights for future research on power saving schemes for Internet mobile streaming and call for a power-efficient, high quality, and scalable protocol for streaming to mobile devices.

The reminder of the paper is organized as follows. We briefly describe the background and our measurement methodology in Section 2 and Section 3. We study the impact of how data is transmitted on battery power efficiency in Section 4, and we investigate the impact of reduced transmission in Section 5. Some related work is described in Section 6 and we make concluding remarks in Section 7.

2. BACKGROUND

In this section, we briefly illustrate the streaming protocols that are currently being used by existing Internet mobile streaming applications.

2.1 RTSP Streaming

Real-time streaming protocol (RTSP) [15] is a standard Internet streaming protocol for the communication between a media player and a media server. RTSP is a stateful protocol that establishes a media session between the client and the server, and allows the client to execute VCR-like commands, such as play and pause. The client maintains a small buffer to absorb network jitter and ensure smooth playback. With RTSP, the streaming server delivers streaming data at the rate equivalent to the object encoding rate.

In practice, today YouTube [16] and Vuclip [17] use RTSP to deliver streaming content to mobile devices such as Blackberry, Nokia, and SonyEricssion mobile phones.

2.2 Pseudo Streaming

While RTSP is a standard streaming protocol, Internet streaming services today also widely use HTTP for streaming content delivery, which we call Pseudo streaming or progressive downloading. With Pseudo streaming, a client downloads the media file from a HTTP server. The playback can start before the entire file is downloaded. Pseudo streaming is used in many popular Internet streaming services today, including YouTube and Vuclip. For example, a mobile version of YouTube allows iOS and Android users to use Pseudo streaming to watch videos in their native browsers [16].

Fast Start [18] is often used in pseudo streaming. Fast Start allows the server to deliver the data at a higher speed than the encoding rate at the beginning of the session and then uses a low speed to deliver the rest of the file. This is to reduce the startup delay perceived by the users as well as smoothing some network jitter during the playback.

2.3 Chunk-Based Streaming

Chunk-based streaming also uses HTTP to deliver streaming content. However, unlike Pseudo streaming that downloads a single streaming file, the media content is segmented into many chunks in Chunk-based streaming, and a client periodically queries the streaming server for new file chunks.

Many streaming services today are Chunk-based, including Apple HTTP Live Streaming [19], Flash HTTP Streaming [20], and Microsoft Smooth Streaming [21]. In this study, we focus on Chunk-based streaming implemented by Apple Inc, which was made available on iOS 3.0 in July 2009. CTV [1], CCTV [2], W.TV [3], ImgoTV [4], Justin.tv [9], NHKworld [5], SPBtv [6], and AirVideo [8] all use Chunk-based streaming. Among them, AirVideo is different from other seven applications in that it allows a user to set up streaming server at a desktop computer, which transcodes the video content and delivers to the mobile device.

Typically, in Chunk-based streaming, the media content is encoded as MPEG-4 audio and video, and transported via TCP using MPEG-2 Transport Stream (MPEG2-TS). They are assembled into `.ts` files using File Segmenter at the server side. A `.ts` file typically contains 10 second media content. For streaming access, a client queries the server for index files (`.m3u8`), which provides information about availability and location of `.ts` files. Then the client requests the listed media (`.ts`) files in order. The client usually queries the server for streaming content about every 10 seconds and the server replies with links to several `.ts` files, each of which contains media content for the next a few (typically 10) seconds. These files are reassembled after being downloaded and fed into the Media Player for playback.

2.4 Peer-to-Peer (P2P) Streaming

Peer-to-Peer (P2P) has demonstrated in practice to be scalable for file sharing and Internet streaming. Lots of Internet streaming

Table 1: Measured Applications

Service	Protocol	Transport	Format
YouTube	RTSP	UDP	3GP/MP4
Vuclip	RTSP	UDP	3GP
YouTube	Pseudo Streaming	TCP	MP4
Vuclip	Pseudo Streaming	TCP	3GP
AirVideo	Chunk-Based	TCP	MPEG-TS
CTV	Chunk-Based	TCP	MPEG-TS
CCTV	Chunk-Based	TCP	MPEG-TS
W.TV	Chunk-Based	TCP	MPEG-TS
ImgoTV	Chunk-Based	TCP	MPEG-TS
Justin.tv	Chunk-Based	TCP	MPEG-TS
SPBtv	Chunk-Based	TCP	MPEG-TS
NHKworld	Chunk-Based	TCP	MPEG-TS
TVUPlayer	P2P Streaming	UDP	ASF

systems have adopted P2P techniques. For example, both PPLive and PPStream are large scale P2P assisted multi-channel Internet streaming systems. Within a P2P system, a peer downloads the streaming content for the playback from other users and uploads the downloaded content to other peers simultaneously. While different protocols have been extensively studied, in practice, most of deployed Internet P2P streaming systems today are mesh-based and they often use a pull approach for data acquisition. That is, peers in these systems need to request data chunks from a dynamically changing set of neighbors. To pull/request data from neighbors, a peer first needs to exchange data chunk availability information (e.g., buffermap) with neighbors. After knowing which data chunks available at which neighbors, the peer then sends *Streaming Data Chunk Requests* to request missing chunks from different neighbors. Typically, the unit of a data chunk is small (e.g., 1280 bytes) compared to the video file size. Thus such data chunk requests are fine-grained, and are highly frequent. Such buffermap messages and fine-grained data chunk requests are referred to as control messages [22].

TVUPlayer [23] uses a Peer-to-Peer protocol to distribute live streaming content. It is available on different platforms, targeting both desktop/laptop users (Windows, Mac OS) and mobile device users (iOS). Instead of TCP, it uses UDP for streaming delivery.

3. METHODOLOGY

To investigate the energy performance of WNIC under the four streaming protocols as we have discussed in Section 2, we conduct experiments with the streaming services shown in Table 1. In our experiments, we use a 2nd generation iPod Touch (iTouch) running iOS 3.1.2 to receive streaming services. Among these services, YouTube and Vuclip provide streaming services using both RTSP and Pseudo streaming. TVUPlayer is the only P2P streaming application available on iTouch at the time of this measurement. The rest 8 applications use Chunk-based streaming through HTTP on iTouch.

For streaming services from YouTube, Vuclip, and AirVideo, we first access video clips from both YouTube and Vuclip via the mobile browser on iTouch. Then we download the video clips to a desktop running a local AirVideo server, and access it using the AirVideo App on iTouch. For CTV, CCTV, W.TV, ImgoTV, Justin.tv, SPBtv, NHKWorld, and TVUPlayer, we watch a streaming channel from each service.

In the experiments, for streaming services delivered via Pseudo-streaming, Chunk-based streaming, and P2P streaming protocols, we use iTouch to receive these streaming services. Since RTSP is not implemented on iOS, we were not able to access YouTube or Vuclip via RTSP directly on our iTouch. Instead, we use a netbook running Windows 7 with `Maximum WiFi Power Saving` enabled. The netbook declares itself as a mobile device, and receives streaming data via RTSP from YouTube and Vuclip servers.

In order to record all the incoming and outgoing packets, we set up Wireshark to listen on the same channel as the iTouch/netbook in a promiscuous mode and capture all packets received/delivered by our testing device.

In this study, we focus on the battery power consumed by the streaming data transmission. In receiving streaming services on a mobile device, there are three major power drainage sources: the WNIC for streaming data transmission, the CPU for data decoding, and the display for data rendering. Among them, the battery power consumed by the WNIC is significant. For example, it has been reported that the WNIC accounts for about 40% total energy consumed in video streaming [24]. We also have confirmed this on iTouch. Basically, we have compared the battery lifetime when playing the locally stored video clip on the iTouch (the WNIC is turned off) and when playing the same video clip streamed from a local server. In a typical test, the battery can last for 8 hours and 20 minutes without network transmission, but it can only last for 5 hours 40 minutes with streaming from AirVideo. This means that the WNIC consumes about 32% battery power in the streaming session. Note that in the above experiment, the WNIC spends 73.5% of the session time in the PSM when receiving streaming data from the local server. If the WNIC spends less time in the PSM, it would consume more battery power.

To estimate the power consumed by the WNIC for streaming data transmission, we rely on the `Pwr Mgt` flag in the frame control information from IEEE 802.11 headers to determine the Power Management mode that the WiFi interface will switch to after the transmission of the current frame. Basically, we count the total time the WNIC spends in the sleep mode. So instead of giving the absolute value of energy consumed, we report and compare the percentage of time that the WNIC spends in the sleep mode. Please note that the sleep time of the WNIC is not necessarily proportional to the power consumption of WNIC. For example, as shown in [25], 30%, 50%, and 70% WNIC sleep time would translate to 0.67 (1/1.5), 0.57 (1/1.75), and 0.5 (1/2) WNIC battery power consumption (compared to when the WNIC is always on). Nevertheless, the sleep time is still a good indicator of the WNIC power consumption without using any external instrument.

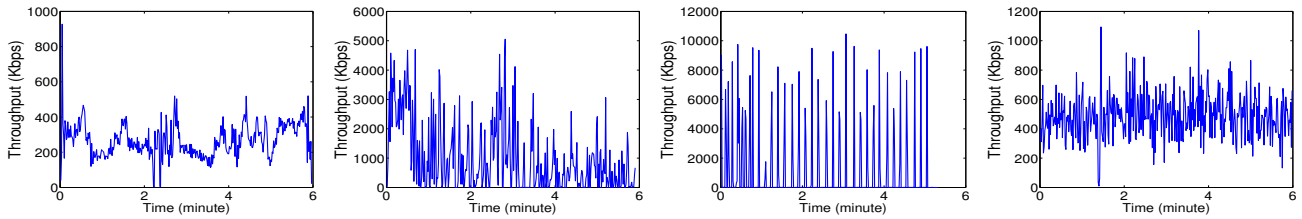
As we focus on the battery power consumption at the WNIC for data transmission in the streaming sessions under the four different delivery protocols, we will investigate battery power efficiency under these four different approaches, centering around the two fundamental aspects, namely (1) how data is transmitted and (2) how much data is transmitted, in the following two sections.

4. IMPACT OF HOW TO TRANSMIT

As we discussed before, how the streaming data is transmitted directly affects how long the WNIC can stay in PSM. In this section, we thus first examine how data is transmitted under these different protocols, and the consequence of such transmission on the battery power consumption on mobile devices.

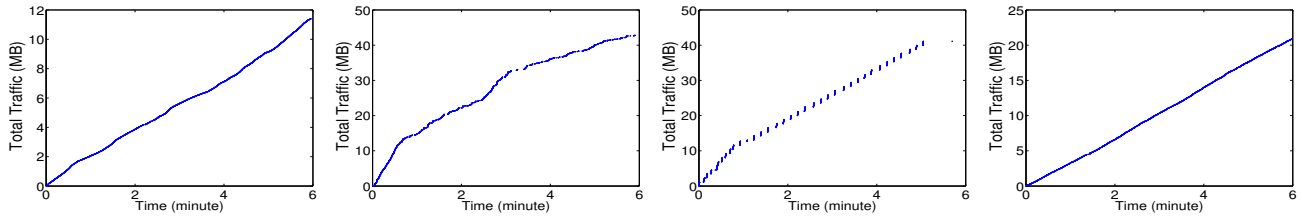
4.1 Chunk-based Streaming is Most Power-Efficient with Traffic Shaping

Figure 1 shows the throughput of traffic delivered to/from our testing device when YouTube is accessed via RTSP and Pseudo streaming protocols, and AirVideo is used with Chunk-based



(a) RTSP (b) Pseudo Streaming (c) Chunk-based Streaming (d) P2P Streaming

Figure 1: Throughput (Kbps) in Delivering High Quality Video



(a) RTSP (b) Pseudo Streaming (c) Chunk-based Streaming (d) P2P Streaming

Figure 2: Traffic Pattern in Delivering High Quality Video

streaming, as well as TVUPlayer is used with P2P streaming in one of their experiments. All these sessions last for 6 minutes. Figure 2 shows the corresponding traffic pattern during the 6-minute playback.

For RTSP as shown in Figure 1(a), the throughput varies between 200 Kbps and 400 Kbps, which is close to the delivered video's encoding rate of 246 Kbps. As a result, the accumulated traffic pattern shown in Figure 2(a) is close to a straight line.

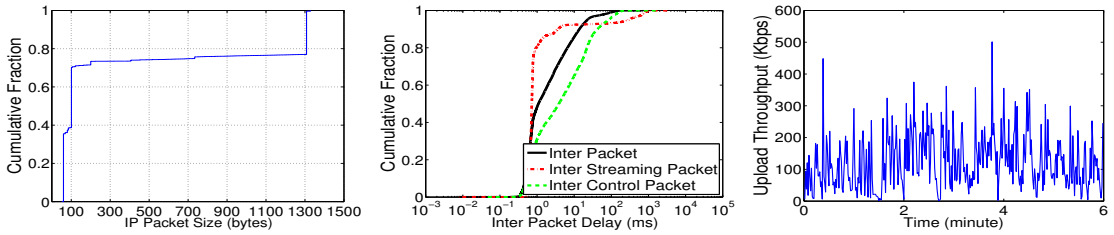
Compared to RTSP that delivers YouTube video according to the video encoding rate, Figure 1(b) shows that when Pseudo streaming is used, the data is delivered at a much higher speed on average although the throughput decreases to the level close to the object encoding rate (654 Kbps) later. As shown in the figure, the average throughput at the beginning 30 seconds reaches 3 Mbps, which is 5 times of the object encoding rate. This would lead more data to be delivered ahead of the playback need. Correspondingly, Figure 2(b) confirms this result. Such a delivery method is called Fast Start as we have discussed in Section 2. The benefit of such delivering is to provide sufficient data to the player at the client slide to reduce the startup delay as well as smoothing out some network jitter.

For Chunk-based streaming, Figure 1(c) and Figure 2(c) show very different patterns from RTSP and Pseudo streaming when the same video clip (886 Kbps) is delivered via AirVideo. As we can observe from Figure 1(c), in AirVideo, the throughput has peaks peridocially, and the peak throughput could reach 11 Mbps (about 12 times of the object encoding rate). As a result, the traffic pattern is very bursty with a regular pattern as shown in Figure 2(c).

Figure 1(d) shows the throughput of P2P based TVUPlayer is about 450 Kbps on average while the object encoding rate is 281 Kbps. The throughput also varies from time to time, but there is no clear pattern as shown in Figures 1(b) or 1(c). Such a traffic pattern is due to additional traffic involved in P2P streaming applications since in P2P streaming, a peer needs to dynamically communicate with multiple neighbors to exchange buffermap information, request streaming data, and upload to neighbors as we have discussed in Section 2.4. Such control traffic and uploading traffic are highly dynamic, depending on the number of neighbors the client is communicating with at each moment.

To verify the impact of control traffic and uploading traffic in P2P streaming, we further plot packet size distribution (CDF) and upload throughput during this 6-minute session. Figure 3(a) shows that the packet sizes have a bi-modal pattern. Further packet inspection shows that all small packets are control packets with a size of 300 bytes or less, while all the larger packets are data packets containing file chunks. As shown in the figure, control packets account for more than 75% of total packets transferred. Such a large number of control packets significantly reduce the inter-packet delay and thus the idle time of the WNIC, resulting in more battery power consumption. Figure 3(b) shows the inter packet delay distribution. `Inter Streaming Packet` delay considers only ingress streaming data packets. `Inter Control Packet` delay considers both incoming and outgoing control packets, and `Inter Packet` delay takes all packets transmitted into account. While over 5% successive streaming packets of TVUPlayer arrive with a delay larger than 100 ms (meaning opportunities to sleep), about 65% control packets are sent/received within 10 ms from the preceding one. This causes the overall Inter Packet delay to significantly deviate from Inter Streaming Packet delay patterns. Apart from the extremely frequent control packets, Figure 3(c) shows that the uploading throughput also reaches as high as 300 Kbps. As a result, while the encoding rate of the video is 281 Kbps, the total throughput varies between 300 Kbps and 800 Kbps as shown in Figure 1(d). On the other hand, even with the control traffic and uploading traffic, Figure 2(d) shows that its traffic pattern is close to a straight line, similar to that of RTSP.

These results show that the streaming data are delivered differently under these four protocols. This results in significantly different battery power consumption on our mobile device: the WNIC sleeps the most in Chunk-based streaming (86.7%) and sleeps the least in RTSP streaming (23.1%) and P2P streaming (30.6%). In Pseudo streaming, the WNIC also sleeps for more than half of the session time (71.4%). Note that the video bit rates in RTSP and P2P streaming are 246 Kbps and 281 Kbps, much lower than these for Chunk-based and Pseudo streaming with the bit rates of 886 Kbps and 654 Kbps.



(a) Packet Size (CDF) (b) Inter Packet Delay (CDF) (c) Upload Throughput (Kbps)

Figure 3: Detailed Results of TVUPlayer (P2P streaming)

Table 2: Average WNIC Sleep Time (%) from Multiple Runs

RTSP	Pseudo	Chunk-based	P2P
23.2	72.2	86.4	29.7

While we only show detailed traffic patterns from one typical run (median of multiple runs), Table 2 shows the average WNIC sleep time over multiple experiments (5 runs) for different protocols. These results are consistent with the findings on the single runs we have presented.

Considering these results with the traffic throughput and traffic patterns we have observed in Figure 1 and Figure 2, we find that

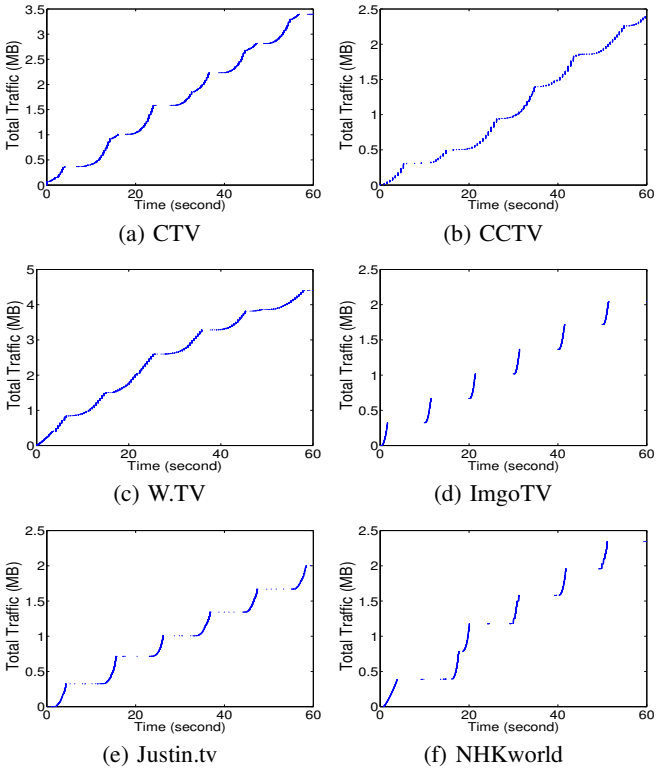
- In Chunk-based streaming such as AirVideo, traffic shaping techniques are being used to deliver the data to mobile devices in burst, which leaves the WNIC on the mobile device into sleep mode for most of the time.
- In Pseudo streaming, the Fast Start technique can help transmit a large amount of data to the user at the beginning, which provides more opportunities for the WNIC to sleep when receiving the rest of data.
- In RTSP streaming, since the server delivers data according to the object encoding rate, it results in the least amount of sleep time of the WNIC.
- In P2P streaming, because a user has to frequently exchange control information and upload data to other peers, the WNIC cannot sleep for long time as in the Pseudo streaming or Chunk-based streaming.

4.2 Traffic Shaping is Widely Used, but it may Cause Quality Degradation

To investigate whether traffic shaping has been used in other services, we have conducted experiments with all other services. The results show that traffic shaping is being used in all these Chunk-based streaming applications: CTV, CCTV, W.TV, ImgoTV, Justin.tv, NHKworld, and SPBtv. Figure 4 shows the traffic pattern of 1 minute snapshot during a 6-minute streaming session. As indicated by these figures, traffic shaping is used in all of these applications for bursty data delivery. However, the burst length varies from application to application.

Correspondingly, Table 3 shows the sleep time of the WNIC in these sessions. As shown in this figure, the sleep time of the WNIC varies between 77% and 87% in all of these sessions, confirming the effectiveness of the traffic shaping on the battery power saving.

Although traffic shaping is very effective in Chunk-based streaming for saving battery power consumption on mobile devices, it has to be used carefully. Otherwise, it could affect the streaming



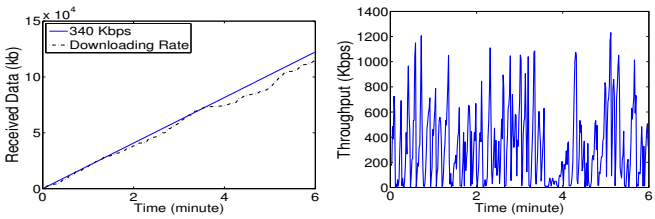
(a) CTV (b) CCTV (c) W.TV (d) ImgoTV (e) Justin.tv (f) NHKworld

Figure 4: Traffic Shaping is Used in All Chunk-based Streaming Applications

Table 3: WNIC Sleep Time (%)

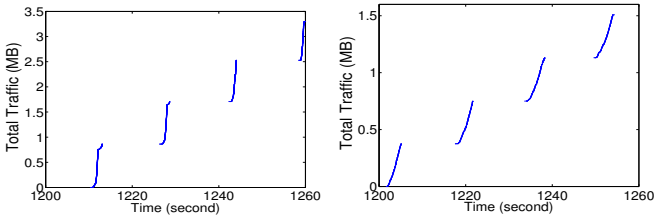
AirVideo	CTV	CCTV	WTV
86.7	78.1	85.8	86.0
ImgoTV	Justin.tv	SPBtv	NHKworld
77.5	82.9	86.9	84.8

quality perceived by end users. Figure 5 shows such a case in our experiments. The data receiving speed shown in Figure 5(a) fluctuates from time to time, and sometimes cannot catch up with the playback (for example, between 100 seconds to 200 seconds), resulting in playback freezing. This is not due to the bandwidth shortage because the thoughput shown in Figure 5(b) can reach higher than the video encoding rate (340 Kbps) even when the playback freezing happens. Therefore, how to choose the right burst length to use in the traffic shaping is important and deserves careful study. We leave it for our future study.



(a) Data Receiving vs. Playback (b) Throughput

Figure 5: CCTV at 340 Kbps



(a) SPBtv - 464 Kbps (b) SPBtv - 174 Kbps

Figure 8: SPBtv: Traffic Pattern

Table 4: WNIC Sleep Time (%) Comparison in High and Low Quality Streaming

	HighQ (Kbps)	Sleep Time (%)	LowQ (Kbps)	Sleep Time (%)
RTSP	246	23.1	81	33.0
Pesudo	654	71.4	247	80.3
Chunk	886	86.7	200	95.6
P2P	281	30.6	156	74.4

4.3 Summary

Through experiments and analysis on how data is transmitted under different protocols, our study shows traffic shaping is very helpful in Chunk-based streaming. However, due to protocol nature of constant bit rate sending, RTSP streaming can hardly apply traffic shaping to reduce battery power consumption on mobile devices.

Pseudo streaming could improve its power efficiency in two ways. One is to use Fast Start to deliver the entire file. The risk, however, is that a user may terminate the playback after viewing the clip for a few seconds. Another concern is the server capacity as the server may wish to reserve some bandwidth to serve other incoming requests. Besides fast streaming, the other option is to adopt traffic shaping technique, which practically turns into Chunk-based streaming.

On the other hand, P2P streaming is affected by both control traffic and uploading traffic. Whether such traffic can be reduced for power saving remains unclear and we will investigate in the next section.

5. IMPACT OF HOW MUCH TO TRANSMIT

As we have studied, how streaming is transmitted could significantly affect the power efficiency on mobile devices when receiving streaming with different approaches. Next we investigate whether reduced traffic amount would significantly reduce battery power consumption on mobile devices. Intuitively, a user may opt to a lower streaming quality with the expectation of less battery power consumption.

5.1 Reducing Streaming Quality Helps Little for All But P2P

To study the impact of reduced traffic amount, we instruct our device to access these streaming services again, but to video clips with lower quality in terms of the object encoding rate. Figure 6 and Figure 7 show the corresponding throughput and traffic patterns in these tests. Table 4 shows the corresponding power saving effectiveness.

Table 5: SPBtv – Impact of Reduced Streaming Data Transmission in Chunk-based Streaming

Quality	Encoding Rate (Kbps)	Total # of Packets	Sleep Time (%)
High	464	184689	86.9
Low	174	106756	90.2

For RTSP based streaming, as the streaming rate is reduced from 246 Kbps to 81 Kbps, Figure 6(a) shows the throughput of data transmission is also reduced from 200-400 Kbps to 50-150 Kbps, without taking full advantage of available bandwidth. The sleep time increases to 33%, and it is mostly because the downloading finishes 1 minute earlier than the viewing session as shown in Figure 2(a).

For Pseudo streaming, the YouTube server still sends streaming data at a very high rate at the beginning (about first 12 MB) as shown in Figures 1(b) and 2(b). However, because of reduced file size due to reduced video encoding rate (from 654 Kbps to 247 Kbps), only about 2 MB more content for the rest of playback is remaining to be downloaded. Thus, in this test, instead of switching to sleep mode opportunistically between successive transmissions as that in the High Quality streaming test, the WNIC can exploit larger intervals between bursts to sleep. As a result, the WNIC spends 9% more time in sleep mode.

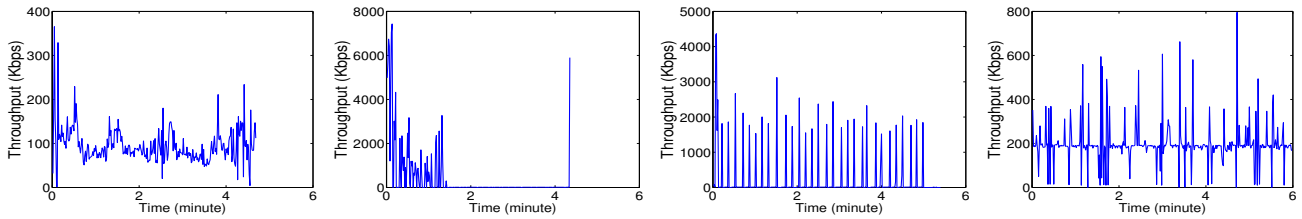
For Chunk-based streaming using AirVideo, similar to its previous experiment with the High Quality video clip, a video chunk is delivered every 10 seconds, and packets are delivered in burst. Because of the reduced chunk size, the time spent on downloading each chunk is also reduced, and the WNIC can sleep longer after downloading each chunk. This is reflected with about 9% increase of the WNIC sleep time in Table 4.

However, surprisingly, in P2P streaming, although Figure 6(d) and Figure 7(d) do not show much difference from Figure 1(d) and Figure 2(d) except the decrease of the average throughput and the accumulated traffic amount, the increase of the power saving is significant: the sleep time of WNIC increases from 30% to 74%.

Overall, the sleep time increase of the WNIC due to the reduced streaming data amount is marginal or moderate (between 5% and 10%) for RTSP, Pseudo streaming, and Chunk-based streaming. But it is very effective for P2P based streaming.

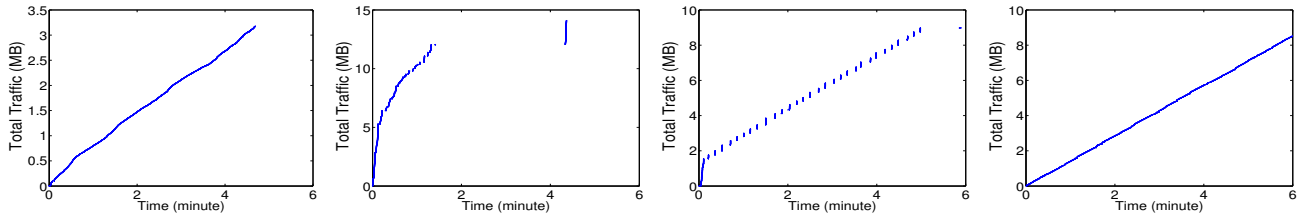
While it is relatively intuitive for such a result in RTSP and Pseudo streaming due to their protocol nature of constant streaming (data sending) rate or Fast Start at the beginning, we further investigate why the impact of the reduced traffic amount is trivial in Chunk-based streaming and is significant in P2P streaming.

To verify the trivial power saving benefit in Chunk-based streaming by reducing streaming data transmission, we further conduct experiments with all other Chunk-based streaming services. Figure 8 shows the typical results with SPBtv. SPBtv allows users to ex-



(a) RTSP (b) Pseudo Streaming (c) Chunk-based Streaming (d) P2P Streaming

Figure 6: Throughput (Kbps) in Delivering Lower Quality Video



(a) RTSP (b) Progressive Download (c) Chunk-based Streaming (d) P2P Streaming

Figure 7: Traffic Pattern in Delivering Lower Quality Video

plicitly choose between High Quality and Low Quality. Through reverse-engineering, we find that: (1) for High Quality, the video bitrate is 400 Kbps and audio bitrate is 64 Kbps; (2) for Low Quality, the video bitrate is 110 Kbps and audio bitrate is also 64 Kbps. As we have described in Section 2, in HTTP streaming, a client would first query the server for index files (playlists). To support multiple quality levels of the same content, such index files can select appropriate files to request based on bandwidth conditions or the quality the client desires. Figure 8 compares the two streaming tests viewing High Quality (464 Kbps) and Low Quality (174 Kbps) live streaming content, respectively. Figure 8(b) indicates that when Low Quality video is accessed, the server also reduces its delivery rate. Given the access is done in non-peak hours and two tests were conducted consecutively, it is less likely due to the network congestion or the client side bandwidth limit. Our conjecture is that the server tends to reserve/allocate more resources to serve other/incoming High Quality streaming requests. Nevertheless, the server still uses traffic shaping when delivering the lower quality content. On the other hand, receiving lower quality streaming only makes the WNIC sleep about 3% more as shown in Table 5, thus confirming very limited effectiveness of reducing the streaming quality on battery power saving.

While the trivial power saving impact in Chunk-based streaming is mainly because of a reduced delivering rate from the server after the user switches to a lower quality video, P2P streaming is more complicated since it involves both control traffic and uploading traffic.

To study how this power saving happens in the lower quality P2P streaming, we further conduct experiments with TVUPlayer. Figure 9 shows more detailed results of TVUPlayer when it is used to access the Low Quality channel of 156 Kbps for one hour, while Figure 10 shows the results of watching a High Quality channel of 281 Kbps for one hour. Both of these two channels are unpopular. Figures 9(d) and 10(d) show the bandwidth spent on uploading to neighboring peers. For most of the time, there is little uploading to other peers in both sessions. Both sessions have similar traffic patterns and inter packet delays as shown in Figures 9(b) (c) and 10(b) (c).

Table 6: TVUPlayer – Impact of Reduced Streaming Data Transmission in P2P Streaming

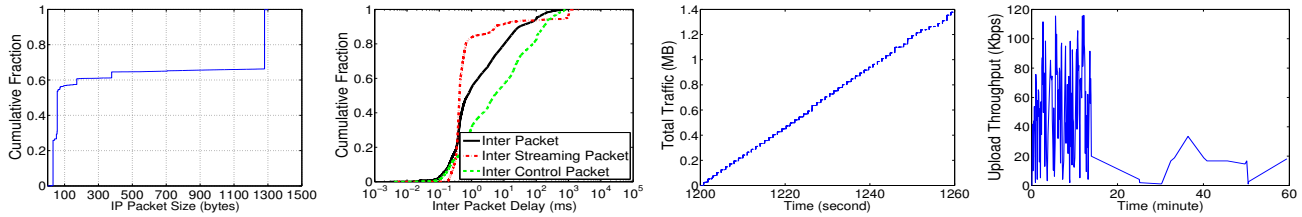
Quality	Encoding Rate (Kbps)	Total # of Packets	Sleep Time (%)
High	281	502987	40.6
Low	156	220524	71.6

However, Figure 10(a) shows a different result from that in Figure 9(a). With a lower streaming quality, the percentage of control packets (these with small sizes of 300 bytes or less) is reduced by more than 15%. This is because control traffic also depends on the number of neighbors the client is communicating with during the streaming session even if both experiments are done with unpopular channels.

Table 6 lists the total number of data packets transmitted in these two sessions. With a reduced streaming quality session, the total number of packets is reduced by more than 50%, among which control packets are reduced even more as shown in Figure 10(a). Thus, when playing the video of 156 Kbps, the WNIC was able to sleep as much as 71.6% of the session time, while it can only sleep about 40.6% when watching the video of 281 Kbps. Clearly, the reduced total number of packets, including the control packets, in the lower quality streaming, leads to a longer sleep time of the WNIC.

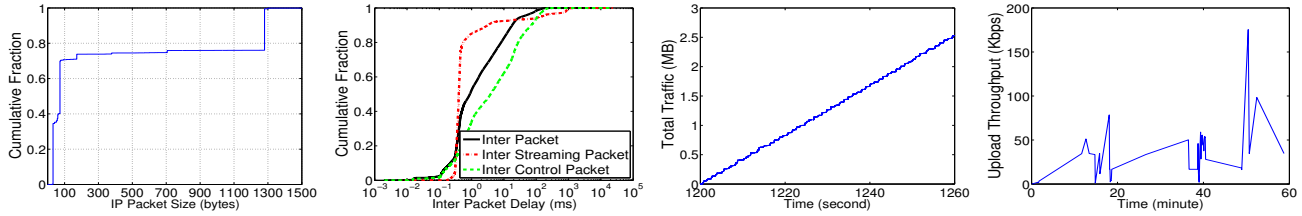
5.2 Reducing Uploading Traffic in P2P Streaming Sometimes Helps

We have shown in the previous experiments that by switching into a lower quality video version, the mobile device in P2P streaming could save more battery power because of reduced data packets and control packets. For P2P streaming, data transmission also includes uploading to other peers. A peer in P2P systems has to upload to other peers for system scalability. But uploading to other peers is a heavy burden to mobile devices. We are wondering if we can reduce the uploading traffic amount to gain some power sav-



(a) Packet Size (CDF) (b) Inter Packet Delay (CDF) (c) Traffic Pattern (d) Upload Throughput

Figure 9: TVUPlayer - P2P Streaming for One Hour (156 Kbps) - Unpopular Channel



(a) Packet Size (CDF) (b) Inter Packet Delay (CDF) (c) Traffic Pattern (d) Upload Throughput

Figure 10: TVUPlayer - P2P Streaming for One Hour (281 Kbps) - Unpopular Channel

ing without switching to a lower quality video, as it is often not desirable by end-users.

5.2.1 *Upload Throttling Can Help*

Due to the concern of free-riders, P2P streaming systems today do not implement *tit-for-tat* as BitTorrent does [26]. As a result, most P2P applications would saturate the available uploading bandwidth when the channel has a large population. This is also true in TVUPlayer. Therefore, we want to study whether throttling uploading traffic in the P2P mobile streaming would reduce the power consumption.

Upload throttling is to throttle the uploading traffic to other peers. Since uploading to others is essential in P2P streaming systems and is the key to its scalability, throttling upload may help preserve battery power, but may be harmful to the entire P2P system. Because TVUPlayer does not allow us to set an uploading cap, we conduct the following experiments to test its effectiveness.

We instruct iTouch to watch a popular channel and an unpopular channel with a same streaming rate. The two experiments are conducted with the same network settings. The popular channel is accessed from 8 PM to 9 PM, and the unpopular one is accessed from midnight to 1 AM in the morning. Figure 10 shows the results of watching the unpopular channel of 281 Kbps, while Figure 11 shows the results of watching the popular channel of 281 Kbps. Since both channels are encoded with the same streaming rate, the CDF of packet size, as shown in Figure 10(a) and Figure 11(a), are largely similar. However, Figure 10(d) and Figure 11(d) show that when watching the popular channel, the device uploads much more traffic than when watching the unpopular one. Such uploading traffic results in a smaller percentage of large inter packet delay as shown in Figure 11(b).

As a result, Table 7 shows that while the WNIC can sleep about 40% of time when watching the unpopular channel, it can only sleep about 25% of time when watching the popular channel. The uploading overhead can result in 15% less sleep time. Note that in this popular channel session, the uploading is not excessive, as on average the uploading bandwidth is around 175 Kbps. If the uploading is higher, more battery power would have been consumed.

Table 7: Uploading Throttling (P2P)

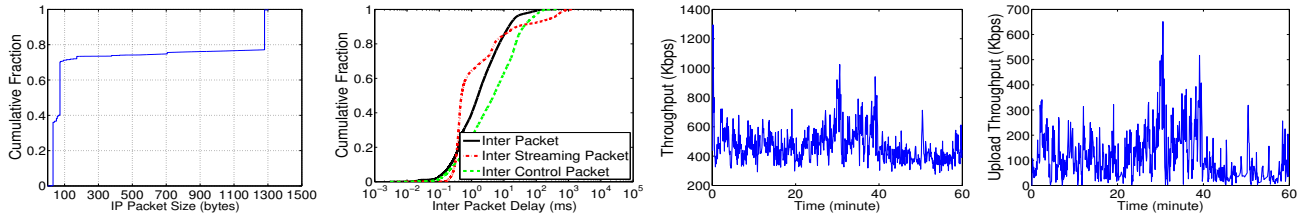
Channel	Encoding Rate (Kbps)	Total # of Packets	Sleep Time (%)
Unpopular	281	502987	40.6
Popular	281	634527	25.9

These experiments show upload throttling can be effective, particular because these P2P streaming systems do not enforce *tit-for-tat*. Otherwise, the received streaming quality might be affected. In practice, these results imply that a user in P2P streaming can choose to watch the unpopular channels in order to preserve more battery power.

5.2.2 *Smart Caching at Access Point Does Not Help*

While upload throttling may not be fair to other peers in a P2P system, and potentially can affect the user's perceived streaming quality if *tit-for-tat*-like mechanisms are enforced, smart caching can alleviate this concern. Smart caching at access points (SCAP) [14] has been proposed to reduce/eliminate uploading in P2P applications by caching the downloaded data in the AP in case they need to be uploaded to other peers soon. For P2P streaming applications, a peer needs to quickly upload the downloaded content to others, and thus the AP only needs to maintain a small buffer to temporarily cache downstream data. When uploading is needed, the mobile device does not need to send the packet by itself. Instead, it sends a compressed packet telling the AP to do that for it. The AP will detect which cached packet to send and send the packet on behalf of the mobile device. Compared with upload throttling, SCAP could be transparent to the application. However, the mobile device still needs to receive requests from neighbors, and thus the control traffic generated by uploading is not eliminated.

We did not implement SCAP on our iTouch. Instead, to evaluate the potential benefit of using SCAP, we assume the best case in which all uploading traffic from the iTouch is eliminated to estimate the maximum power saving. We re-evaluate the trace shown in Figure 11 for watching a popular channel and we calculate the



(a) Packet Size (CDF) (b) Inter Packet Delay (CDF) (c) Throughput (d) Upload Throughput

Figure 11: TVUPlayer - P2P Streaming for One Hour (281 Kbps) - Popular Channel

Table 8: Estimated Effectiveness of SCAP

	Encoding Rate (Kbps)	Total # of Packets	Sleep Time (%)
w/o SCAP	281	634527	25.9
SCAP	281	597746	26.4

potential power saving by counting all inter packet delays that are larger than the idle timeout interval as the sleep time of the WNIC. We have validated our method on our iTouch based on the data link layer control information for streaming sessions shown in previous sections. The error is less than 1%.

As shown in Table 8, compared to the case without using SCAP, SCAP only allows the WNIC to sleep 18 seconds more (less than 1 % of the session time). This surprising result, however, has some underlying reasons. Table 8 shows that the number of uploading packets saved by SCAP is relatively small (less than 40000), about 6% of the total number of packets. However, as shown in Table 7, twice more packets can be saved by watching an unpopular channel. This indicates that there is a significant portion of control traffic packets involved in uploading. Without removing these control packets, SCAP cannot increase many more opportunities for the WNIC to switch into the power saving mode.

These results on smart caching indicate that reducing uploading alone is not necessarily very helpful. The control traffic plays a very important role in P2P streaming and must be taken into consideration. This is why reducing the streaming quality or upload throttling are more effective, in which control packets are reduced proportionally.

5.3 Summary

The experiments in this section show that reducing transmission has trivial impact on power saving for mobile devices in RTSP, Pseudo streaming, and Chunk-based streaming, mainly because the server also reduces the sending rate when a lower quality streaming session is accessed although there is additional bandwidth available.

On the other hand, while reducing streaming quality is helpful for power saving in P2P streaming, it may not be desirable for end-users. Without sacrificing the streaming quality, simply reducing upload alone such as smart caching does not help, because the control traffic is not reduced. However, upload throttling can reduce both the uploading and control traffic, and thus can reduce the battery power consumption on the mobile device.

Considering both aspects, i.e., how to transmit and how much to transmit, during the streaming data transmission, in the current practice, we find Chunk-based streaming is more power-efficient than others. That is, a mobile user who wants to save battery power may always want to choose Chunk-based streaming delivery first if it is available. Pseudo streaming could be the next choice. On the other hand, different delivery approaches are not always available for the same video clip. Thus, Internet streaming service providers utilizing other approaches should adopt more power-saving techniques in order to better serve mobile users.

6. RELATED WORK

Along the popularity of wireless Internet accesses, a lot of research has been conducted on battery power saving on mobile devices. For example, Chandra studied typical Internet streaming services over wireless connections and concluded that 802.11 PSM does not save energy if the streaming rate is over 56 Kbps [27]. Anand et al. considered a self-tuning approach to adapt the WNIC's behavior to the access pattern and the intent of applications [28]. A power saving technique to reduce `energy × delay` product was proposed in [29]. Qiao et al. [30] proposed a smart power saving mode. Guo et al. designed a cooperative relay service in order to exploit the idle communication power of the WNIC for network throughput improvement [31]. Targeting streaming services delivered over the Internet to wireless devices, PSM-throttling leverages traffic shaping from the client side in order to allow the WNIC to sleep for a longer time [12]. This client-centric scheme eliminates the demand of infrasfrasture support as required by many previous schemes. For P2P applications such as P2P streaming running on wireless devices, Tan et al. designed SCAP [14] in order to reduce the power consumption on wireless devices by caching the uploading traffic at the AP. In SCAP, the upload from the wireless device can be performed directly from the AP instead of the wireless device, which could save both uploading bandwidth and battery power consumption on the wireless device. More recently, Xiao et al. studied the power consumption of mobile YouTube [32] and we have also examined resource utilization in different Internet mobile streaming architectures [33].

On the other hand, since mobile devices these days commonly have multiple network interfaces, plenty of work has considered to leverage the different power consumption rate of these interfaces [34, 35, 36, 37]. For example, CoolSpots [34] was proposed to intelligently switch between the Bluetooth and the WiFi interfaces. Agarwal et al. proposed to wake up the WiFi interface for incoming calls using the GSM radio [37]. Most recently, authors in [38] proposed Wiffler to reduce power consumption of mobile devices by augmenting 3G using WiFi.

While a lot of aforementioned efforts had aimed to reduce the battery power consumption by the wireless network interfaces on mobile devices, little is known about their feasibility and effectiveness for streaming to mobile devices. Furthermore, in the current practice, how streaming data is transmitted to mobile devices and how much data is transmitted under different protocols have not been thoroughly investigated. Our work in this study measures, analyzes, and compares the battery power efficiencies on data transmission under different delivery approaches. We have also explored potentials of various power saving techniques. Our results

provide new insights for further improving the battery utilization on mobile devices when receiving Internet streaming services.

7. CONCLUSION

Along the pervasive usage of all kinds of mobile devices for Internet accesses, more and more streaming services are provided over the Internet to mobile users. However, receiving streaming services on mobile devices can consume a significant portion of the limited battery power supply. In this paper, we set to examine the battery power consumption on mobile devices in receiving such streaming services, focusing on streaming data transmission. Our measurement and analysis provide some new insights for providing power-efficient Internet mobile streaming services in the current practice. In particular, we show that Chunk-based streaming delivery is currently the most power-efficient approach with appropriate traffic shaping techniques. On the other hand, while P2P streaming is scalable and very promising, it is challenging to reduce the battery power consumption without sacrificing the streaming quality. Besides adding to the state-of-the-art, our results also provide some basis for next step research for more battery power-efficient and scalable Internet mobile streaming services.

8. ACKNOWLEDGEMENT

We appreciate constructive comments from our shepherd Wei-Tsang Ooi and anonymous referees. The work is partially supported by NSF under grants CNS-0746649, CNS-1117300, CCF-0915681, CCF-1146578, and AFOSR under grant FA9550-09-1-0071.

9. REFERENCES

[1] "CTV," http://itunes.apple.com/app/id340381556.
[2] "CCTV," http://itunes.apple.com/app/id331259725.
[3] "W.TV," http://itunes.apple.com/app/id318676629.
[4] "ImgoTV," http://itunes.apple.com/app/id349448995.
[5] "NHKworld," http://itunes.apple.com/app/id350732480.
[6] "SPBtv," http://itunes.apple.com/app/id356830174.
[7] "OrbLive," http://itunes.apple.com/app/id290195003.
[8] "Air Video," http://itunes.apple.com/app/id306550020.
[9] "Justin.tv," http://itunes.apple.com/app/id358612216.
[10] "Verizon Wireless: V CAST," http://products.vzw.com/index.aspx?id=video.
[11] "AT&T Mobile TV," http://www.att-mobiletv.com.
[12] E. Tan, L. Guo, S. Chen, and X. Zhang, "PSM-throttling: Minimizing Energy Consumption for Bulk Data Communications in WLANs," in *Proc. of IEEE ICNP*, Beijing, China, October 2007.
[13] S.Chandra and A.Vahdat, "Application-specific network management for energy-aware streaming of popular multimedia formats," in *Proc. of USENIX ATC*, 2002.
[14] E. Tan, L. Guo, S. Chen, and X. Zhang, "SCAP: Smart Caching in Wireless Access Points to Improve P2P Streaming," in *Proc. of IEEE ICDCS*, June 2007.
[15] "Real Time Streaming Protocol (RTSP)," http://www.ietf.org/rfc/rfc2326.txt.
[16] "Mobile YouTube," http://m.youtube.com.
[17] "Vuclip," http://m.vuclip.com.
[18] "Fast Start," `http://www.microsoft.com/windows/windowsmedia/technologies/bettertogether.aspx#faststart`.
[19] "Apple HTTP Live Streaming," http://tools.ietf.org/html/draft-pantos-http-live-streaming.
[20] "Flash HTTP Streaming," http://www.adobe.com/products/httpdynamicstreaming/.
[21] "Microsoft Smooth Streaming," http://www.iis.net/download/SmoothStreaming.
[22] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "Insights into PPLive: A Measurement Study of a Large-Scale P2P IPTV System," in *Proc. of IPTV Workshop*, Edinburgh, Scotland, UK, May 2006.
[23] "TVUPlayer," http://itunes.apple.com/app/id323640984.
[24] J. Adams and G. Muntean, "Adaptive-Buffer Power Save Mechanism for Mobile Multimedia Streaming," in *Proc. of IEEE ICC*, 2007.
[25] F. Dogar, P. Steenkiste, and K. Papagiannaki, "Catnap: Exploiting High Bandwidth Wireless Interfaces to Save Energy for Mobile Devices," in *Proc. of ACM MobiSys*, 2010.
[26] Y. Huang, T. Fu, D. Chiu, J. Lui, and C.Huang, "Challenges, Design and Analysis of a Large-scale P2P-VoD System," in *Proc. of ACM SIGCOMM*, 2008.
[27] S. Chandra, "Wireless network interface energy consumption implications of popular streaming formats," in *Proc. of MMCN*, 2002.
[28] M. Anand, E. Nightingale, and J. Flinn, "Self-Tuning Wireless Network Power Management," in *Proc. of ACM MOBICOM*, Sept. 2003.
[29] H. Yan, R. Krishnan, S. A. Watterson, D. K. Lowenthal, and K. Li, "Client-Centered Energy and Delay Analysis for TCP Downloads," in *Proc. of IWQoS*, June 2004.
[30] D. Qiao and Kang G. Shin, "Smart Power-Saving Mode for IEEE 802.11 Wireless LANs," in *Proc. of IEEE INFOCOM*, 2005.
[31] L. Guo, X. Ding, H. Wang, Q. Li, S. Chen, and X. Zhang, "Exploiting Idle Communication Power to Improve Wireless Network Performance and Energy Efficiency," in *Proc. of IEEE INFOCOM*, April 2006.
[32] Y. Xiao, R. Kalyanaraman, and A. Yla-Jaaski, "Energy Consumption of Mobile YouTube: Quantitative Measurement and Analysis," in *Proc. of NGMAST*, Washington, DC, September 2008.
[33] Y. Liu, F. Li, L. Guo, and S. Chen, "A Measurement Study of Resource Utilization in Internet Mobile Streaming," in *Proc. of ACM NOSSDAV*, 2011.
[34] T. Pering, Y. Agarwal, R. Gupta, and R. Want, "CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio interfaces," in *Proc. of ACM MobiSys*, Uppsala, Sweden, 2006.
[35] T. Armstrong, O. Trescases, C. Amza, and E. de Lara, "Efficient and Transparent Dynamic Content Updates for Mobile Clients," in *Proc. of ACM MobiSys*, Uppsala, Sweden, 2006.
[36] A. Rahmati and L. Zhong, "Context-for-wireless: Context-sensistive energy-efficient wireless data transfer," in *Proc. of ACM MobiSys*, 2007.
[37] Y. Agarwal, R. Chandra, A Wolman, P. Bahl, K. Chin, and R. Gupta, "Wireless Wakeups Revisited: Energy Management for Voip over Wi-Fi Smartphones," in *Proc. of MobiSys*, San Juan, Puerto Rico, 2007.
[38] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting Mobile 3G Using WiFi: Measurement, Design, and Implementation," in *Proc. of ACM MobiSys*, 2010.