GazeTube: Gaze-Based Adaptive Video Playback for Bandwidth and Power Optimizations

Shruti Patil*†, Yu Chen*‡ and Tajana Simunic Rosing†
†University of California San Diego, ‡ Boston University
†patil@ucsd.edu, ‡chenyua@bu.edu, †tajana@ucsd.edu

***Abstract*—With the popularity of mobile devices for personal entertainment, video streaming over mobile networks has seen a dramatic increase. We observe that users are often interested in just the supporting audio within the videos rather than the visual content, for example, when users play videos to listen to music. New mobile device capabilities such as gaze tracking enable easy detection of such use-cases. We propose to exploit such capabilities to selectively stream audio/video content based on user gaze. This eliminates unnecessary streaming, with two-fold advantages: first, it saves precious network bandwidth for ISPs, and second, reduction in video processing decreases device power. Using an android-based gaze-aware media application, we evaluate three strategies: a maximal saving scheme that cuts off video stream at source when user gaze is directed away from the screen; a conservative scheme that reduces video resolution, aimed at maintaining user experience, and a third, intermediate scheme that streams the videos at a reduced frame rate. Our evaluation shows that gaze awareness during video streaming can lead to high bandwidth savings that are linearly proportional to Gaze-OFF ratios, and up to 46% power savings on device (up to 28% power savings considering gaze tracking overheads).**

I. Introduction

A recent study shows that 50% of YouTube's [1] traffic now stems from mobile devices, compared to just 25% last year and 6% only two years ago. 23% of all Netflix subscribers watch videos on smartphones, and 15% use iPads [2]. VEVO music video platform's mobile and TV app audience ballooned by 184% in 2014, with half of its views coming from mobile devices. Further, Amazon has about 16.7 million Prime subscribers that get unlimited video streaming on Kindle devices and via Amazon's mobile apps. Similar studies have been undertaken in [3], [4], [5]. Given this fast growing need for high quality on-demand video streaming with limited bandwidth infrastructure, techniques to optimize bandwidth without reducing user experience are critical for video providers, as well as for clients that are charged as per their data usage.

In smart devices, a new class of user interfaces called 'Attentive User Interfaces' is gaining traction to enable responsive and user aware device operation [6]. Speech and gaze awareness are the primary features provided by these interfaces. Accessibility to gaze detection features on mobile devices provides an opportunity to exploit user awareness during video streaming applications. In particular, tracking the "eye contact" between user and mobile device can enable video bandwidth optimizations. During video streaming, when the user's gaze trails away from the screen, precious network bandwidth and device computation resources are wasted. However, in many cases, the audio content may continue to be of interest. Such situations occur often for distracted users with inactive gaze at the screen. Typical examples are when users resort to playing music videos primarily for their music content instead of the video, or when users watch news or sports videos in the background while working inside a different application.

We propose *GazeTube*, gaze driven video streaming optimization schemes that reduce streaming overheads of video traffic, without interrupting the audio stream. When user gaze moves away from the screen, we assume the video stream is no longer of primary interest, allowing for crucial bandwidth optimizations. We propose and evaluate three strategies. First, the video transmission is blocked at the source, allowing audio packets to continue streaming. This is aimed at *maximal* savings (MaxSav). To evaluate the bandwidth and power savings enabled by MaxSav, we implement a gaze-aware media player application in Android OS, that tracks user gaze and transmits a 'gaze-on' and 'gaze-away' signal to the streaming server. The server executes the selective video streaming strategy, conserving network bandwidth for the duration of time the gaze is away. Our evaluation seeks to study the overheads in terms of user experience, i.e. the duration of time video content is absent when gaze returns. To minimize the discontinuity experienced by the user, we implement a second strategy where the video quality is degraded to minimal acceptable resolution (ResSav). This is a conservative strategy that reduces video quality instead of halting the video stream. However, in maintaining acceptable user experience, the power benefits of the optimization are reduced. To decrease the power consumption on the device, we evaluate a third strategy that streams the low resolution video at lower frame rate (FpsSav). This scheme strikes a middle ground between MaxSav and ResSav, providing a trade-off between user experience and the metrics of bandwidth and power.

Our work makes the following contributions:

- To the best of our knowledge, this is the first investigation of application of gaze detection for network bandwidth reduction. Data usage savings are proportional to the gaze-away ratios. On the device side, up to 46% power savings are possible by stalling video stream during user inattention.
- We measure the degradation of user experience due to video discontinuity. Our evaluation revealed that while the network communication delay is low, application playback settings can introduce a perceptible delay before the new video frames are displayed on the phone. To reduce the delay experienced by the user, we evaluate two additional schemes that reduce video resolution instead of stalling video stream, while offering comparable bandwidth savings.
- We illustrate the possible data savings through two case studies by inferring average user attention during video streaming. Our analytical calculations reveal considerable savings for ISPs.

*Both authors have contributed equally to the paper.

In existing streaming protocols, data is already provided in differentiated packets of audio and video, making the schemes easy to implement without significant changes. The largest overhead of implementing the schemes is imposed by gaze tracking on the device. We expect that custom gaze tracking features will soon become available on future phones, in much similar manner as custom ultra-low power always-on speech-aware circuitry has become available in existing phones. In this paper, to provide a proof-of-concept, we implement a software-based gaze detection using the front camera in the phones.

The paper is organized as follows: Section II discusses related work. Section III describes the strategies proposed in this paper. Section IV describes the experimental framework. Section V presents our results and case-studies. Finally, Section VI concludes the paper.

II. Related Work

New sensors are being added to smartphones to support functionalities that enhance user experience. Addition of front facing cameras in smartphones has enabled applications such as Samsung's 'Smart Scroll'/'Smart Pause' [7], and Amazon Fire Phone's Head and Gaze Tracking APIs [8] to track user gaze and customize application behavior. Apple has also shown interest in gaze detection for their product lines[9].

Gaze Detection: Extensive research has been conducted in gaze detection algorithms. Some techniques use a single camera [10], [11], [12], [13], [14], while other schemes need additional hardware, such as auxiliary infra-red LEDs [15], stereo camera [16]. Use of such additional hardware may lead to better detection results, but they are not yet available in current smartphones. A low-power hardware implementation is proposed in [17], which achieves gaze detection within a 70mW power budget. We use the results from this paper to analyze our gaze detection overheads.

Video Bandwidth Optimizations: To save bandwidth, [18] optimizes video broadcasting protocol by removing redundant data. Similarly, [19] and [20] share video data among cellphones within the same WiFi network through Bluetooth to save cellular bandwidth. Granola [21] is a protocol designed for on-demand video streaming, where only one stream of video data is sent to cellphones within the same subnet, which then share the data through WiFi to save server bandwidth.

Video Power Optimizations: In reducing power of video streaming, downloads are smartly scheduled on 3G/4G radio leveraging user viewing history [22]. Screen luminosity is varied according to different types of videos in [23] , while cross-layer optimizations are performed in [24]. Power optimization has also been studied with respect to video data, by analyzing the stream and reducing redundant computations [25]. Our work focuses on point-to-point protocol, and switches the video and audio stream using gaze detection. Video decoding power reduces due to absence of video stream at the source.

So far, bandwidth and power saving techniques in existing video playback applications have not taken advantage of gaze detection capabilities. Mobile vendors have identified possible device power optimizations with the proposed gaze detection capabilities, such as screen dimming when user gaze is away[9]. Samsung uses gaze detection to pause videos when user gaze is away[7]. However, our observation and case study illustrates that the video content is not requested for some of the use cases when user gaze is away. We therefore exploit gaze detection to propose optimizations for device power, WiFi/cellular data transmission, and network bandwidth, in video streaming applications.

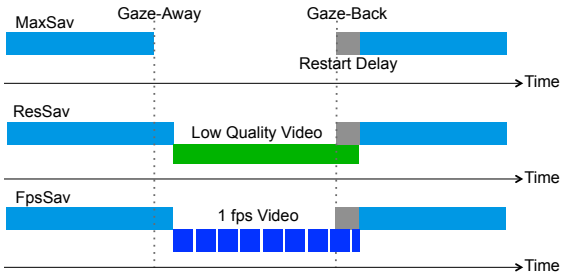


Fig. 1: MaxSav, ResSav and FpsSav schemes

III. Background and Approach

The goal of our work is to reduce bandwidth and power usage for video content that is not being consumed by the user. However, for gaze-away scenarios, the audio content may be of interest to the user and hence must continue to play. Compared to video size, audio files are considerably smaller in size (in our experiments, the video data rate is $\sim$1.5 Mbps, while audio data rate is 64kbps i.e. $\sim$23X lower) and incur lower bandwidth and processing overheads. Thus, significant amount of network bandwidth and mobile device battery can be conserved by reactively controlling the video data at source. We first describe how video streaming is accomplished in mobile networks and then introduce our optimization schemes.

A. Video Streaming

Numerous protocols exist for streaming multimedia from Content Delivery Networks (CDNs) to mobile users. Android OS natively supports Real-Time Streaming Protocol (RTSP) for multimedia streaming. It provides for communication between Client and Server with a 'VCR-style' remote control over the stream of multimedia data over an IP network [26]. The Client can issue requests such as play, pause, fast forward, and seek within the stream. On receiving the request, the Server begins sending the audio and video stream over Real-Time Transport Protocol (RTP), a widely-used communication protocol for IP audio and video packets.

Another popular streaming protocol with CDNs is the Dynamic Adaptive Streaming over HTTP (DASH)[27]. It allows dynamic switching between videos that differ in bit rates based on network load and congestion. This protocol is widely used by streaming providers such as Netflix to ensure continuous playback without frequent buffering[28]. Videos are split into multiple segments (of 2-10sec typically) and get buffered segment-by-segment as video content is consumed. While our underlying protocol is RTSP, we dynamically switch between video streams of varying resolution through app-level control, triggered by user gaze feedback. Similar to DASH, the Server hosts multiple video streams, and streams the one requested by the Client gaze engine. We also propose two schemes to tackle the video restart delay experienced upon gaze return for RTSP and buffering based protocols like DASH.

B. Gaze aware optimizations

1) Maximal Savings Scheme (MaxSav): Maximum bandwidth and power savings can be achieved by completely blocking the video stream when gaze is directed away. When gaze returns, the video computation must seek to the correct video frame, synchronize it with the audio and restart playback.

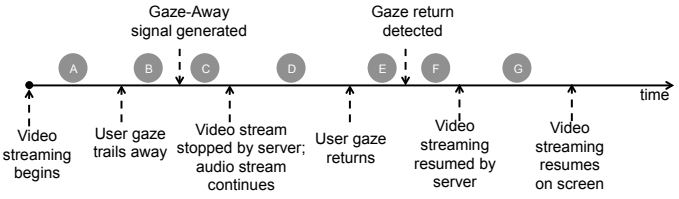


Fig. 2: Delays involved during playback

The gaze feedback control signaling between Client and Server for MaxSav introduces a delay. Figure 2 illustrates the delays involved during playback. We find that the regular operations of *play* and *pause* appear to be instantaneous, however, the delay of restarting the video (E+F+G) when gaze returns is not negligible. Restart delay is due to signaling latency, *seek* overhead on the Server, Client playback settings for processing new data frames, and Client video pipeline refill. The actions of *play* and *pause* only incur signaling latencies, and both Server and Client instantly resume operation from their point of halt. We discuss the restart latency in Section IV.

2) Reduced Resolution Scheme (ResSav): A critical performance metric in interactive applications is user experience. High video restart latency degrades user experience by affecting the continuity of video playback for the user. To address this, ResSav reduces the video quality by lowering resolution instead of stopping video packets completely. Similar to MaxSav, the delay of restarting or refilling the pipeline still exists when switching between the video streams. However, the media player can continue to play the low-res video, while simultaneously processing the new high-res frames, offering a seamless switch. This scheme trades-off bandwidth and power savings for un-interrupted user experience.

3) Reduced FPS (FpsSav): To improve power gains in ResSav, we exploit the trade-off of user experience to power and bandwidth by controlling video frame-rate. Reducing the frame-per-second (FPS) decreases the video decoder computation, while user continues to get periodic updates on the screen. In our experiments, we set the FPS to 1 during gaze-away, so that the user receives content that is at most 1sec old when gaze returns.

Figure 1 illustrates the proposed schemes. These require a signaling between Client-Server that supports switching between streams of varying resolutions or frame rates. Provisions for such signaling already exist in CDNs as they support adaptive protocols like DASH. While DASH stream is selected based on network load measurements by the Client, our schemes select the stream type based on user gaze. Although reduced number of video packets decreases computation power on the mobile device, some gains are lost to the overheads introduced by gaze detection, discussed in Section V-D.

IV. Experimental Setup

For evaluation of the three schemes, we implement an Android-based media player application for smartphones. We first describe the hardware and software used in the experiments. We then describe our implementation of gaze-detection algorithm. Lastly, we explain our benchmarking and overhead measurement methodology.

Hardware: We use Snapdragon MSM8960 smartphone (Table I). A desktop acts as a Server delivering video content to the phone. The phone communicates with the Server through WiFi within same subnet.

TABLE I: Snapdragon S4 MSM8960 hardware parameters

Processor	Snapdragon S4 MSM8960 with asynchronous dual Krait CPU cores at 1.5GHz each
Graphics	Adreno 225 graphics processing unit (GPU)
OS	Android Version: 4.1.1; Kernel Version: 3.4.0-g6ad319f-dirty
Display	4.0in WSVGA 1024x600 multi-touch display
Video	1080 High-definition video recording and playback up to 30 frames per second; Stereoscopic 3D playback via HDMI output
Camera	13MP main camera with flash; 2MP front facing camera
Memory	1GB LPDDR2; 16 GB eMMC

Software: To emulate realistic streaming environment, we use the *Wowza Streaming Engine* as video stream server [29]. Wowza is designed for streaming of live and on-demand multimedia through IP networks to network-connected devices. It supports a wide variety of video streams, including RTSP, DASH, Apple HLS and Adobe RTMP. Due to native support for RTSP in Android, we implemented an RTSP client application using the *mediaplayer* class provided in Android 4.0. It first establishes a connection using the Server URL, sends RTSP request signals, receives RTP video stream, and plays the video stream. We use Apache web server to separately stream the audio data using an MP3 audio file. The Android device loads the entire audio file when the video is initially loaded. A synchronization logic monitoring audio and video timestamps is used to keep the audio-video in sync during display.

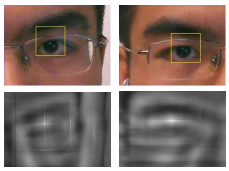
Our Gaze Detection Algorithm: To provide proof-of-concept, we implement software-based gaze detection with the front camera in our media player application. We use Android OpenCV SDK [12] to trace the relative position of the iris to determine 'gaze-on' or 'gaze-away'. Our algorithm is implemented as follows:

(a) First, we use the Cascades algorithm with the training set *lbpcascade-frontalface*[30] to track the face.The eyes are assumed to be located at relative dimensions to the face, within a bounding box of $\frac{2}{9}$ to $\frac{4}{9}th$ fractions of the face height, and $\frac{1}{8}$ to $\frac{7}{8}th$ fraction of the face width [14].

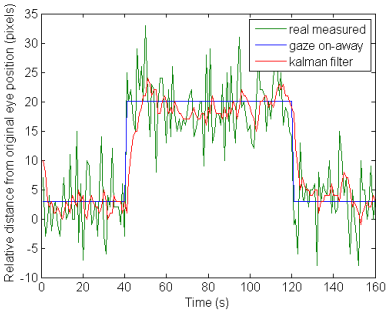
(b) Using templates of left and right iris, the eye area is scanned. Template matching requires two images: the *source* image (I), in which we hope to find the object; and the *template* image of the iris (T), which is captured for the user at the start. To detect the area with the highest match, the mask is moved over the image one pixel at a time, from top left to the bottom right. The correlation is calculated as follows:

$$R(x,y) = \frac{\sum_{x',y'} T'(x',y') \cdot I'(x+x',y+y')}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

where (x,y) is the position of source pixel and (x',y') is the position of mask pixel. For each location of T over I, it



(a) Result matrix R matching eye areas.



(b) Relative iris positions detected

Fig. 3: Detecting gaze on and away signals

generates the metric of one point. After scanning the source image, the result matrix R is produced (Figure 3a). The brightest locations indicate the highest matches.

(c) We use Kalman filter on the X and Y positions of the iris to obtain a stable output.

The above steps are repeated for each frame captured at 1s interval. In our experiments, the computation executed within 0.143s on average. Thus, delays B and E in Fig. 2 are at most 1.143s. Iris templates are captured at the beginning of the program, or when the user loses the tracking of eyes. If no face can be found, or if the eyes' relative position is larger than a threshold, then gaze-away is signaled. Figure 3b shows the gaze signals determined from distance calculations.

User behavior model: To simulate user gaze changes, we model normally distributed gaze-on and gaze-away intervals, targeting a specific proportion of on and away time. Figure 4 shows an example schedule with 80% gaze-on ratio.

Benchmark: The BigBuckBunny video benchmark[31] encoded with H.264 video codec is used for our experiments. High-res video is 480p at data rate of 1.5Mbps, and low-res video is 144p, at data rate of 0.103Mbps. The audio stream is encoded at 64kbps. While we experiment with two videos at fixed high and low resolution, the schemes can be extended to multiple levels of resolution if desired.

Bandwidth and Data Usage: Linux-based Android offers several tools for network monitoring. We use *tcpdump* to monitor real-time data rate for the Client-Server channel.

Power: We measure power using the Trepn Profiler[32], a diagnostic tool designed for Snapdragon-based phones to measure the performance of Android applications. The Trepn Profiler uses exclusively designed hardware hooks embedded in the Snapdragon MDP to isolate and measure real power, CPU usage, and other data points specific to certain blocks of the hardware, providing a breakdown of power and performance measurements, along with total battery energy.

Measuring video restart latency: We break down the video restart latency into two parts - the Server turnaround delay and video player delay. Server turnaround delay is the time between when the client sends the 'gaze-back' signal and when the first packet of video stream arrives at the client. This delay happens in any RTSP operation. Video player delay is the time between when the first video packet arrives at the client and when the player resumes displaying the new

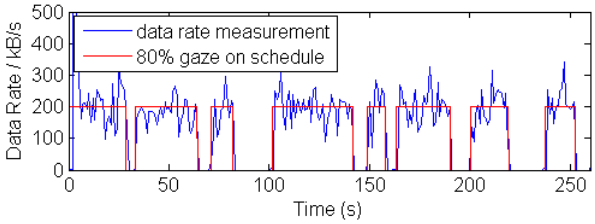


Fig. 4: Bandwidth trace for MaxSav with 80% gaze-on ratio

content on the screen. Server turnaround time is profiled in the application through instrumentation. To measure video player delay, we record a video of the smartphone screen. Although our overhead results are specific to the RTSP setup, the measurement approach is applicable to other protocols.

V. Evaluation

A. Bandwidth and Power Savings

Figure 4 shows an example of real time bandwidth plot for MaxSav with 80% average gaze-on ratio. Bandwidth usage over a range of gaze-on ratios (0-100%) for proposed schemes is compared in Figure 5. Average bandwidth is normalized to 100% gaze-on ratio, which is the same for the three schemes since there is no switching in the video playback. ResSav consumes an average of 18% more data than MaxSav. Results show that the bandwidth reduces by an average 19% as the proportion of gaze-on ratio reduces in 20% steps with the maximal savings scheme. The reduction is 17% on an average with ResSav. This shows that most of the bandwidth savings ($\sim$90%) of MaxSav can be retained in ResSav while maintaining an uninterrupted user experience. Quantitatively, the data usage and savings for the two schemes can be expressed with the following equations:

$Used_{Max} = \alpha \cdot v_h \cdot t + A;$	$Saved_{Max} = (1-\alpha) \cdot v_h \cdot t$
$Used_{Res} = \alpha \cdot v_h \cdot t + v_l(1-\alpha)t + A;$	$Saved_{Res} = (v_h - v_l)(1-\alpha)t$

$DataUsage_{Max}$ and $DataUsage_{Res}$ is the the total data transmitted with the MaxSav and ResSav schemes respectively. α is the average gaze-on ratio. t is the total video playback time at high resolution. v_h and v_l are the average video data rates. In the experiment, v_h = 1.49 Mbps and v_l = 0.103Mbps, as set for the benchmark. A is the size of the audio file. The results expected analytically match our experimental results closely. In both schemes, data savings are proportional to the gaze-away ratio $(1-\alpha)$. With ResSav, the data savings are also linearly proportional to the relative difference between v_l and v_h. Although actual data savings increase if 720p or more high-res video is used, the normalized bandwidth gains stay comparable. This is because the differences in data savings in the two schemes stem from the term $v_l \cdot (1-\alpha) \cdot t$. With 144p low-res video, this amounts to a small additional bandwidth consumption compared to MaxSav. Bandwidth savings with FpsSav fall between those of MaxSav and ResSav.

Figure 6 compares the power consumption of the whole cellphone for the three schemes at 90% confidence intervals. The empirical measurements show 5-15% reduction in power consumption with MaxSav as user attentiveness reduces in steps of 20%. Power decreases by about 42% compared to baseline of 100% gaze-on, when user gaze is present for 20% time. On the other hand, only 6% savings are observed with ResSav at 20% user gaze. FpsSav comes up as middle ground

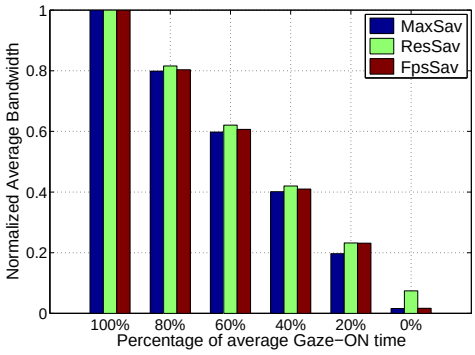


Fig. 5: Average bandwidth for schedules with 0-100% gaze-on.

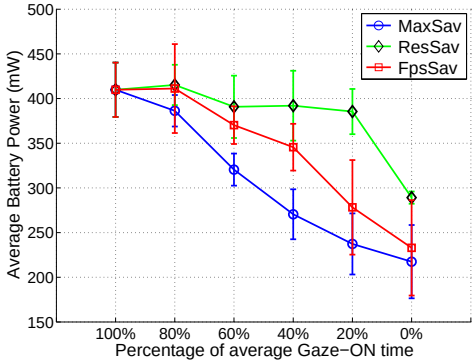


Fig. 6: Device power for schedules with 0-100% gaze-on

between the two schemes, with 32% power savings at 20% gaze-on. Thus, while the bandwidth savings are comparable with the three schemes, trading off user experience for frame-rate allows for achieving significant power gains. We expect realistic scenarios to fall within the 60-80% gaze-on ratios. For example, we found 18-44% advertisement time (Table II) accompanying popular TV shows when streamed on the mobile device, during which user gaze is likely to trail away. At 60% gaze-on ratios, we can expect to get 21.8%, 4.6% and 9.7% power savings with MaxSav, ResSav and FpsSav respectively. With MaxSav at 0% gaze-on ratio, up to 46% maximum power savings are observed owing to reduced computation.

TABLE II: Analyzing Advertisement-to-Episode time ratio when streaming TV shows on mobile devices

Six Award-winning TV Shows	Episode Time (s)	Total Ad Time (s)	Ad-to-Episode Ratio (%)
Fox-Family Guy (S13E12)	1297	243	18.7
Fox-Gotham (S1E18)	2630	438	16.7
ABC-Modern Family (S6E17)	1291	576	44.6
ABC-Agents of S.h.i.e.l.d (S2E13)	2583	942	36.5
NBC-Saturday Night Live (S40E15)	3431	904	26.3
NBC-The Tonight Show Jimmy Fallon (E233)	2484	660	26.6

B. Video restart delay

We measure the video restart delay in two parts as described in Section IV. On an average, the Server turnaround time is 47.5ms, while the videoplayer delay time is 2.28s. Latter delay is sensitive to video player application settings. For e.g., applications typically wait for 60 frames to arrive at the device before processing the frames. This ensures uninterrupted video display on the screen. This is also necessary to ensure that a new video keyframe is obtained in order to refresh the contents faithfully. For DASH like protocols, the restart latency may arise due to the amount of buffering requirements. These delays degrade user experience. Prior studies recommend latencies less than 250ms for video conferencing[33]. This is the main motivation for the conservative low-res schemes. Alternatively, the user experience degradation can be avoided by predicting the gaze-back signal, which is still an interesting problem. In general, if duration of 'gaze-away' is less than 2s, then the video stream player is desired to continue playing instead of being stopped. Predicting expected gaze-away durations is a part of our future investigations.

TABLE III: Youtube Data Savings for audio-focused videos

	Len (s)	Number of Views (as of Mar-15-2015)	Video BW (kbps)	BW-Savings-per-view (kbps)	Total Data Saved (TB)	LinkID
Beck-BlueMoon	241	3,582,521	239.00	109.54	4.73	WIWbgR4vYiw
Coldplay-SkyFullOfStars	274	48,895,276	338.69	210.22	140.82	v=zp7NtW_hKJI
Coldplay-Magic	285	29,070,048	305.96	176.84	73.26	1PvBc2TOpE4
EdSheeran-Afire	323	17,025,841	274.92	146.13	40.18	JznXx1Ns374
EdSheeran-ThinkingOutLoud	292	23,365,988	334.25	205.48	70.10	WpyfrixXBqU
LanaDelRey-BigEyes	283	2,946,903	534.28	404.24	16.86	Col9Av1ydS4
Maroon5-Maps	189	44,695,400	355.56	224.34	94.75	xj6fHiF8Osg
TheBlackKeys-Fever	246	7,930,868	806.50	676.42	65.98	iZZUY32iCzU
				269.15 (Avg)	506.68 (Total)	

C. Case Studies - Expected Network Data Savings

The benefits of gaze-aware schemes for bandwidth reduction are significant considering the massive scale of video streaming over the internet. We describe two analytical case studies to illustrate this.

1) Audio-only videos: YouTube is a popular application for watching videos[1]. While majority of the YouTube content is video, a significant number of videos are audio-only and use static or neutral video frames. Since they carry no useful video content, we conclude that these videos are accessed primarily for their audio content, and expect 0-20% gaze-on. We pick ten such videos[1] uploaded in 2014 by official music channels with more than a million views. Table III analyzes the possible bandwidth savings by assuming 20% gaze-on. The videos are downloaded at 360p resolution. Savings are analytically calculated as data size * 80% gaze-away * number of views * % views from mobile platforms. We use 2014 statistic of 50% video views from mobile platforms[1]. With the eight videos, more than 500TB mobile network data savings were possible in 2014. These numbers are compelling when we consider the large number of youtube uploads that are aimed at audio-focused access, such as 'playlist mixes'.

2) Streaming talks: As second case study, we study bandwidth savings for TED talks. These videos are typically 15mins or longer in duration, and elicit a vast audience. The length of these talks makes gaze awareness schemes highly relevant when streaming to mobile devices. We analytically model the possible data savings using 80%, 50% and 20% gaze-on ratios. Table IV shows the data savings considering that the optimization is applicable to at least 10% of the viewers. The calculations show the possibility of over 290TB data savings for five popular videos.

[1]Videos can be accessed at *https://www.youtube.com/watch?v=[LinkID]*

TABLE IV: Analytical Data Savings for five TED videos

Title Talk	Video Size (MB)	Audio Size (MB)	Num of views (as of Mar-25-2015)	Data Savings with X% Gaze (MB) X=20 / 50 / 80	Savings for 10% users (TB) 20	50	80
Ken Robinson - How schools kill creativity	136.6	7.4	32,120,098	103.36 64.6 25.84	332.0	207.5	83.0
Amy Cuddy - Your body language shapes who you are	153.2	15.5	24,629,660	110.16 68.85 27.54	271.3	169.6	67.8
Simon Sinek - How great leaders inspire action	126.3	11.4	21,673,662	91.92 57.45 22.98	199.2	124.5	49.8
Brené Brown - The power of vulnerability	145	12.8	19,279,475	105.76 66.1 26.44	203.9	127.4	51.0
Jill Bolte Taylor - My stroke of insight	124	8.4	16,956,622	92.48 57.8 23.12	156.8	98.0	39.2
Total					1163	727	290

D. Gaze Detection Overhead

Continuous gaze detection on mobile systems is challenged by power consumption. A recent effort in addressing this challenge on mobile platforms has yielded a low-power solution that consumes about 70mW power [17] which is about 18% that of the maximum power consumption in our experiments. This implies that if user gaze-on is 60% or lower, the proposed *MaxSav* scheme has the potential to reduce power consumption in the device by up to 28%. Our paper is based on the premise that power efficient hardware gaze tracking circuitry will soon become mainstream on mobile devices to facilitate power optimizations as well as improved user experience. Recent introduction of such hardware in the Amazon Fire phone[8] and Samsung devices[7] are promising steps towards the availability of such sensors in future devices.

These strategies can also be easily extended to desktop/laptop class multimedia player. Most PCs are equipped with front camera, and likely to encounter gaze-away scenarios more often due to their static location, for example, when users use video playlists for listening to music, which is not always available on-demand on audio streaming websites. Besides signals from the active gaze tracking circuitry, other indicators are also useful to initiate the bandwidth saving schedule, for instance when user switches to other windows or locks the screen. These can be detected using software-driven events, without incurring gaze tracking overheads.

VI. Conclusion

Video consumption by mobile users has dramatically risen over the past few years, and is expected to become a dominant part of wireless network traffic in the near future. New smart features being added to phones provide an opportunity to reduce wasteful consumption of the network and device resources. We exploit the ability of gaze detection on a smartphone to monitor user attention during video streaming. When durations of inattention are detected, the video stream can either be stopped or reduced in resolution until gaze returns. Our experiments show that network bandwidth savings are proportional to the gaze-away ratio when video streaming is completely stopped, but results in degraded user experience due to video restart delay on gaze return. Alternatively, conservative schemes that reduce video resolution quality maintain the user experience, providing trade-offs between user experience, and savings in bandwidth and power.

References

[1] YouTube, https://www.youtube.com/yt/press/statistics.html, 2015.

[2] Business Insider, http://www.businessinsider.com/mobile-video-statistics-and-growth-2013-12, [accessed 05-Jan-2014].

[3] eMarketer, [2013]http://www.emarketer.com/Article/DoPeopleWatchVideoDifferentlyonMobilePhonesvsTablets/1009733.

[4] MarketingLand, http://marketingland.com/nielsen-time-accessing-internet-smartphones-pcs-73683.

[5] salesforce, http://www.exacttarget.com/sites/exacttarget/files/deliverables/etmc-2014mobilebehaviorreport.pdf, 2014.

[6] P. Maglio *et al.*, "Gaze and speech in attentive user interfaces," in *ICMI 2000*, 2000, vol. 1948, pp. 1–7.

[7] Samsung, "Galaxy S4," http://www.samsung.com/global/microsite/galaxys4/lifetask.html, 2013, [Online; accessed 15-Mar-2015].

[8] Amazon, https://developer.amazon.com/public/binaries/content/assets/javadoc/fire-phone/reference/com/amazon/headtracking/package-summary.html, 2015, [Online; accessed 01-Jan-2015].

[9] A. Hodge and M. Rosenblatt, "Electronic devices with gaze detection capabilities," May 2013, US Patent 20130135198.

[10] N. H. Cuong and H. T. Hoang, "Eye-gaze detection with a single webcam based on geometry features extraction," in *ICARCV*, 2010.

[11] C. Li *et al.*, "The indirect keyboard control system by using the gaze tracing based on haar classifier in opencv," in *IFITA*, 2009, pp. 362–366.

[12] OpenCV, http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html, 2014, [Online; accessed 01-Feb-2015].

[13] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, vol. 1. IEEE, 2001, pp. I–511.

[14] F. Timm and E. Barth, "Accurate eye centre localisation by means of gradients," in *VISAPP'11*, 2011, pp. 125–130.

[15] Q. Ji and X. Yang, "Real-time eye, gaze, and face pose tracking for monitoring driver vigilance," *Real-Time Imaging*, vol. 8, no. 5, 2002.

[16] Y. Matsumoto and A. Zelinsky, "An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement," in *Automatic Face and Gesture Recognition*, 2000, pp. 499–504.

[17] A. Mayberry *et al.*, "iShadow: Design of a wearable, real-time mobile gaze tracker," in *MobiSys '14*, 2014, pp. 82–94.

[18] J.-F. Paris, "A simple low-bandwidth broadcasting protocol for video-on-demand," in *ICCCN*, 1999, pp. 118–123.

[19] T. Pering *et al.*, "Coolspots: Reducing the power consumption of wireless mobile devices with multiple radio interfaces," in *MobiSys '06*.

[20] M. Ramadan, L. El Zein, and Z. Dawy, "Implementation and evaluation of cooperative video streaming for mobile devices," in *PIMRC*, 2008.

[21] D. Santos and A. Perkusich, "Granola: A location and bandwidth aware protocol for mobile video on-demand systems," in *SoftCOM*, Sept 2008.

[22] X. Li *et al.*, "Greentube: Power optimization for mobile videostreaming via dynamic cache management," in *MM '12*, 2012, pp. 279–288.

[23] S. Pasricha *et al.*, "Reducing backlight power consumption for streaming video applications on mobile handheld devices." 2003, pp. 11–17.

[24] S. Mohapatra *et al.*, "Integrated power management for video streaming to mobile handheld devices," in *MULTIMEDIA*, 2003, pp. 582–591.

[25] Y. Huang *et al.*, "Using offline bitstream analysis for power-aware video decoding in portable devices," in *MULTIMEDIA*, 2005, pp. 299–302.

[26] C. Liu, "Handbook of emerging communications technologies," 2000, ch. Multimedia over IP: RSVP, RTP, RTCP, RTSP, pp. 29–46.

[27] MPEG DASH Protocol, http://dashif.org/mpeg-dash/.

[28] V. Adhikari *et al.*, "Unreeling netflix: Understanding and improving multi-cdn movie delivery," in *INFOCOM*, March 2012, pp. 1620–1628.

[29] Wowza, http://www.wowza.com/, 2015, [accessed 01-Feb-2015].

[30] Itseez, https://github.com/Itseez/opencv/tree/master/data/lbpcascades.

[31] Peach Open Movie Project, "Big buck bunny," https://peach.blender.org.

[32] Qualcomm, https://developer.qualcomm.com/mobile-development/increase-app-performance/trepn-profiler, [accessed 01-Feb-2015].

[33] D. Ferrari, "Client requirements for real-time communication services," *Communications Magazine, IEEE*, vol. 28, no. 11, pp. 65–72, Nov 1990.