Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 5, May 2014, pg.176 – 183

RESEARCH ARTICLE

MOBILE MULTIMEDIA TRAFFIC ANALYSIS: CLIENTS CAN WASTE NETWORK BANDWIDTH

Miss. Karishma Talan
Department of Computer Science & Engineering
Prof. Ram Meghe Institute Of Technology & Research
Badnera, Maharashtra, India
Email: karishmatalan@gmail.com

Dr. G.R. Bamnote
Department of Computer Science & Engineering
Prof. Ram Meghe Institute Of Technology & Research
Badnera, Maharashtra, India
Email: grbamnote@rediffmail.com

Abstract:
Video is increasingly becoming one of the most pervasive technologies in terms of everyday usage, both for entertainment and in the enterprise environments. Mobile video is responsible for a majority of the growth seen in mobile broadband data volume. Video streaming on mobile devices is on the rise. According to recent reports, mobile video streaming traffic accounted for 52.8% of total mobile data traffic in 2011, and it is forecast to reach 66.4% in 2015. The network traffic behaviors of the two most popular HTTP-based video streaming services: YouTube and Netflix. The research indicates that the network traffic behavior depends on factors such as the type of device, multimedia applications in use and network conditions. Furthermore, it is found that a large part of the downloaded video content can be unaccepted by a video player even though it is successfully delivered to a client. This unwanted behavior often occurs when the video player changes the resolution in a fluctuating network condition and the play out buffer is full while downloading a video. Some of the measurements show that the discarded data may exceed 35% of the total video content. The spike in mobile video streaming has come about as a result of the development of the smartphone. Smartphones registered a 5% to 40% market penetration between 2007 and 2010 in the United States. In the third quarter of 2011, smartphone sales increased by 42% from 2010. Mobile Operators are facing an explosion in wireless data use, which is projected to grow 18-fold from 2011 to 2016 per the latest Cisco VNI forecast. With the use of mobile devices increasing so rapidly, and almost half of the traffic on mobile internet networks being accounted for by video sessions, mobile service providers have begun to recognize the need to provide higher quality video access while using the lowest possible bandwidth.[With the release of the iPhone 5 in September 2012, it has been predicted that LTE networks might experience decreased data speeds as streaming multimedia begins to tax the 4G network. Cloud-based content optimizers that reduce the strain of over-the-top multimedia streaming could provide potential relief to mobile providers.

Keywords: HTTP adaptive bit-rate streaming, HTTP-based video streaming, Video Streaming, Mobile Wireless

INTRODUCTION

Today's popular video streaming services such as YouTube and Netflix use HTTP adaptive bit-rate streaming. Adaptive bitrate streaming is a technique used in streaming multimedia over computer networks. While in the past most video streaming technologies utilized streaming protocols such RTP with RTSP, today's adaptive streaming technologies are almost exclusively based on HTTP and designed to work efficiently over large distributed HTTP networks such as the Internet.[1]

It works by detecting a user's bandwidth and CPU capacity in real time and adjusting the quality of a video stream accordingly. It requires the use of an encoder which can encode a single source video at multiple bit rates. The player client switches between streaming the different encodings depending on available resources. "The result: very little buffering, fast start time and a good experience for both high-end and low-end connections."[1][5]

More specifically, and as the implementations in use today are, adaptive bitrate streaming is method of video streaming over HTTP where the source content is encoded at multiple bit rates, then each of the different bit rate streams are segmented into small multi-second parts. The streaming client is made aware of the available streams at differing bit rates, and segments of the streams by a manifest file. When starting, the client requests the segments from the lowest bit rate stream. If the client finds the download speed is greater than the bit rate of the segment downloaded, then it will request the next higher bit rate segments. Later, if the client finds the download speed for a segment is lower than the bit rate for the segment, and therefore the network throughput has deteriorated, then it will request a lower bit rate segment. The segment size can vary depending on the particular implementation, but they are typically between two (2) and ten (10) seconds.

Regardless of the file format and size of the video, the video server pushes the requested video content to the client as network conditions permit. The video content is buffered locally on the device and played back. Hence, if the network bandwidth available to the client is smaller than the encoded data rate of the video, the client has to wait until there is sufficient space in the buffer. Regardless of whether the client pauses or not while playing a video, some video content providers such as YouTube and Netflix continue to push the requested video content to the client. A part of the downloaded video content can be discarded without being watched if the client chooses to quit the video before it ends.

This paper focuses on HTTP adaptive bit-rate streaming; [*Hyunwoo Nam*]analyze YouTube and Netflix video streaming while watching the videos on mobile devices (iOS and Android) over wireless networks (Wi-Fi, 3G and LTE) under varying network conditions. As shown in Figure 1, [*Adam Back]* have designed the Video Streaming Packet Collector (VSPC) to capture and analyze TCP/IP and HTTP packets on video streaming between a client and a video server. Table I shows the hardware specifications of the selected iOS and Android mobile devices that [*Saamer Akhshabi*] used in the experiments.

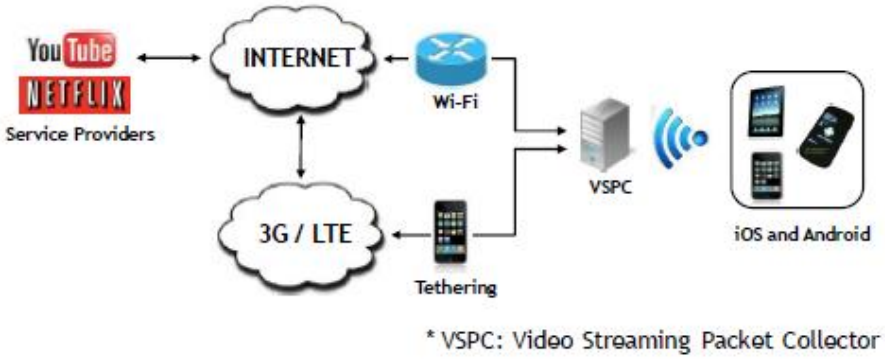

Fig.1Mobile Application Traffic testbed

Devices	OS versions	Resolutions	Memory
iPad 3	iOS 6.1.2	1920 x 1080	1024 MB
iPhone 4S	iOS 6.1.2	640 x 960	512 MB
iPhone 3G	iOS 4.1.2	320 x 480	128 MB
Nexus 7	Android 4.2.1	1280 x 800	1 GB
Nexus S 4G	Android 4.1.1	480 x 800	512 MB

Table 1. iOS and Android Mobile Devices

After analyzing HTTP-based video streaming, [*S. Lederer*] found that a video player establishes a sequence of TCP connections by sending HTTP GET messages while playing a video. The behavior of downloading video contents varies depending on the operating system (OS), the hardware performance of the client device and the network condition. Compared to Android, for example, YouTube video player for iOS sends more HTTP GET messages via new TCP connections to download the duplicate video content for the possible re-play activities by the client. *[K. Balachandran*] also found that Netflix video player for iOS periodically requests a small chunk of the video (every 10 sec in measurements) while the video player for Android aggressively downloads the entire video at one go. The analysis indicates that a significant amount of video content can be discarded by the video player, even without being stored in the video play out buffer while the video is being played. One of the measurements shows that over 35% of the total video content can be lost by the video player. [*T.-Y. Huang*] found that the undesirable behavior often occurred when the video player established new TCP connections in the middle of downloading the video that was requested previously. Once it opened a new TCP connection, it rejected all the incoming video packets via the previous connection. Based on the measurements *[C. Müller]* found out that the client received the unnecessary video data when the video player changed resolutions and the video play out buffer was full while downloading the video.

RELATED WORK

Videos are specialized content that comprise of three basic elements. They are:

1) Moving video component: This section actually contains the moving images that when played in quick succession gives the feeling of watching a continuous video stream. Some popular video codecs used in the industry today are H.264, VP6, VP8, etc.

2) Synchronized audio component: This section has the audio content corresponding to the video being played back. Some popular audio codecs used in the industry are AAC+, AAC, mp3, etc.

3) Video container: Both the video and audio components are wrapped around a media object called a video container. This has additional information about the audio and video components like, the codecs used and the frame offsets. This information is used by the player to play back the video smoothly and also to handle video seek requests from the user. Some popular video containers used in the industry are MPEG4, Flash Video and WebM. [*Adam Back,Ulf Möeller and Anton Stiglic]*

A. Video Coding

Video coding using the H.264 / MPEG-4 standard is achieved by removing the redundancy or similarities between the neighboring frames in a video sequence. There are two kinds of redundancies present in video data: spatial and temporal sampling. The characteristics of a natural video scene, that are relevant for video processing and compression are spatial characteristics, e.g., texture variation within the scene, number and shape of objects, color , and temporal characteristics, e.g., object motion, changes in illumination, movement of the camera or viewpoint.

Temporal quality refers to the number of frames per second. The motion in a scene appears smoother if more frames are transmitted and received within each time interval. Spatial quality can be expressed as the number of pixels in the video frame, which is the product of its horizontal and vertical resolution.

Both temporal and spatial resolutions are usually determined before video encoding, preventing any dynamic trade-offs during the encoding process.

The H.264/AVC codec is an attractive candidate for wireless applications including video streaming, due to the available features for achieving higher compressions efficiency than previous video standards. The H.264/AVC is widely used in real time video transmission.

B. Video Streaming Analysis

Using (Figure 1), [*C. Müller*] analyzed the network traffic behaviors of YouTube and Netflix while playing several videos on mobile devices (iOS and Android) over wireless networks (Wi-Fi, 3G and LTE). The videos were played using the video players provided by the content providers, not using a Web browser installed in the user device. As a baseline analysis,*[S. Lederer*] performed the following three experiments.

_ Analyze and contrast the resulting resolution of the same video contents delivered to several mobile devices over Wi-Fi (Figure 2): These experiments indicate that a device capable of higher performance receives a video with higher resolution.
_ Analyze and contrast the resulting resolution of the same video contents delivered through iOS and Android over Wi-Fi (Figure 3): These experiments indicate that the video content size remains the same regardless of the operating system.
_ Analyze and contrast the resulting resolution of the same video contents delivered to an iPhone 3G via different access networks, namely Wi-Fi, 3G and LTE (Figure 4): These experiments indicate that the video content size remains the same regardless of the network type. These baseline experiments indicate that a video quality is highly dependent on the hardware specification of a client's device when a client requests a video. The device specification can be obtained from user-agent information in the HTTP GET messages. Hence, such device considerations directly impact Over The Top (OTT) resource consumption, as well as the Quality of Experience (QoE) for the end users.
This poses more significant challenges on the wireless capacity planning, which traditionally has been ignoring the capacity consumption per device type, and as it relates to an enforced QoE in a wireless network.

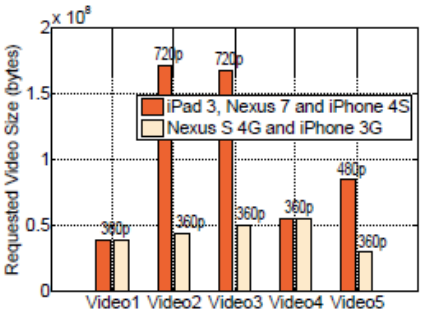

Fig.2 Video Resolution with mobile Devices over wi-fi networks

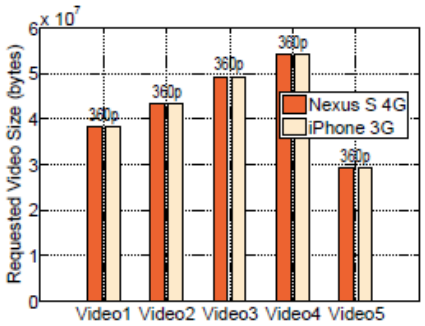

Fig.3 Video Resolution With iOS & android over wi-fi network

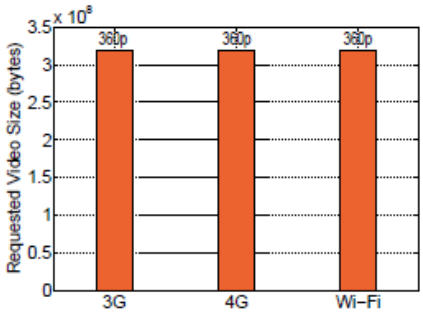

Fig 4. Video Resolution with an iphone 3G over wi-fi, 3G & LTE

[Bong Ho Kimy] found that video players send multiple HTTP GET messages to the video content servers while playing YouTube and Netflix videos. For Netflix video streaming, the video player for iOS generates periodic HTTP GET messages while maintaining a single TCP connection. Each spike in Figure 5 corresponds to the video packet transmission from Netflix after the periodic HTTP GET messages (10 sec on average) from the client's device. Unlike iOS, the Netflix video player running on Android devices requests the whole video at one go. Each time it establishes a TCP connection, it uses a different TCP port number from the previous connection.

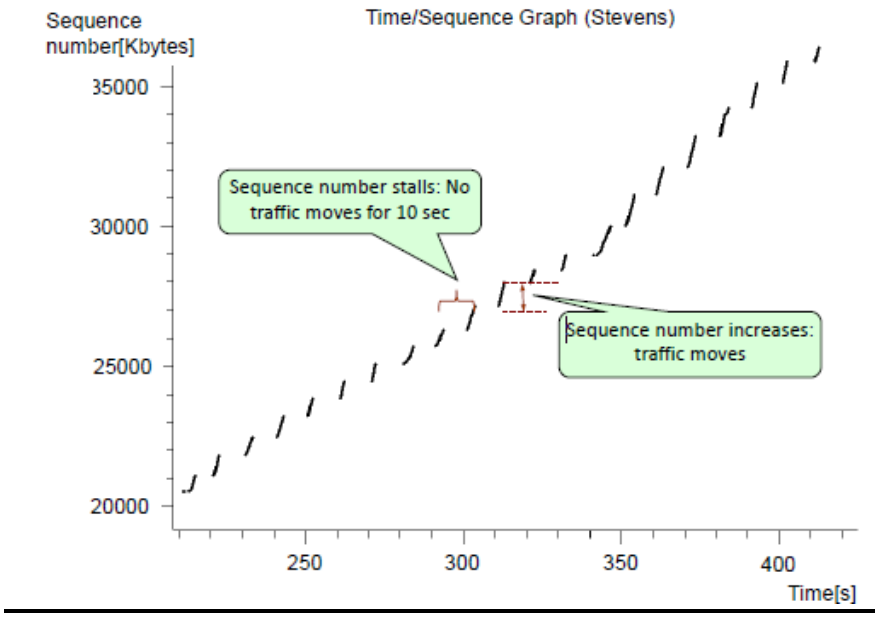

Fig. 5. TCP sequence number with Netflix video trace over LTE

Unlike Netflix, the traffic behavior is quite dynamic with YouTube video servers. The dependency of HTTP GET messages varies depending on the operating systems (OSs). For example, experimental results show that the YouTube video player for iOS typically sends more HTTP GET messages via new TCP connections than the video player for Android while playing a video. As investigated by[*Yao Liu et al]*. One of the reasons is that the YouTube video player for iOS sends additional HTTP GET messages to download duplicate video data for the possible re-play activities after completely downloading the video content. [*Bong Ho Kimy]* see the additional traffic on iOS devices, but do not find it on Android devices. Using PCs, Anton Stiglic established multiple video sessions via the same wireless connection to load the network. Conditions beyond the access point are unknown, since the testbed is established over the public Internet. Under loaded network conditions, the clients for YouTube and Netflix often experienced buffer underflow (downloading rate < video encoded rate) and the display froze from time to time. Throughout the measurements, *[Anton Stiglic]* found that the YouTube video player for iOS sent more HTTP GET messages than the video player for Android. The measurements indicate that the video player for iOS established multiple TCP connections in parallel to download small chunks of the video content, while the video player for Android maintained a single TCP connection under the congested network conditions.

C. Problem Finding

Traditional streaming uses special media protocols, such as RTSP (Real-Time Streaming Protocol), RTMP (Real-Time Messaging Protocol) and RDT (Real Data Protocol). With RTSP, traditional streaming will send the video as a series of small packets in sequence in RTP packets. The RTP packet size is around 1 Mb for 1 Mbps quality video and these packets will get transmitted over UDP transport (UDP RFC 768). The protocol is designed so that it is better to have a momentary glitch in audio or video than for the playback to stop altogether and wait for missing data. The packets are streamed at a specified bit rate. The client can tell the server this rate using RTSP or derive it from the encoding rate of the content.
Forward Error Correction (FEC) can be applied to the UDP stream to help mask packet loss. Alternatively, the client can request the retransmission of the missing packets. This is done using RTP Control Protocol (RTCP RFC3611). RTCP can also be used to indicate the quality of the communication such as jitter and round-trip-time delay (RTT). This can be used to help decide whether to change the codec or bit rate.
RTP/RTSP video is most often used in wireline IPTV deployments. Strict admission control before starting a new stream ensures there will be sufficient bandwidth dedicated for the stream. It also works best over low loss networks so the FEC or packet retransmission overheads are minimized.
In contrast to progressive download described, during a streaming media session, the file is never completely downloaded to a local storage device as it is with progressive download. This difference makes streaming media a more secure option that reduces the risk of piracy.*[K. Balachandran, D. Calin, E. Kim, and K. Rege]*
Issues with traditional streaming that motivate the need for HTTP adaptive streaming include the following:

- Since dedicated media servers are needed, existing web servers cannot be leveraged. The transmission of media packets is through UDP protocols.
- Once the video is encoded at certain bitrates, the client will receive that quality of video content regardless what kind of CPU condition and bandwidth network the client has. This is similar to progressive download.
- Traditional streaming is harder to scale since, as a stateful protocol, traditional media servers maintain a one-to-one persistent connection with each client.

D. HTTP Adaptive Streaming (HAS)

The available bandwidth over the Internet is highly variable. Wireless technology adds to this variability due to the time and spatially varying radio propagation characteristics of the Radio Access Network (RAN). To handle this variability in bandwidth and to ensure best image quality when bandwidth is good, an adaptive streaming technology is required. For professional content streamed over the Internet, a common approach for streamed content is HTTP Adaptive Streamed (HAS). There are several variants including MPEG DASH, Apple HLS, Microsoft Smooth Streaming but they all work in the same way. Figure 5 shows the operation of HAS and Figure 6 shows the message exchanges to execute HAS.[1]

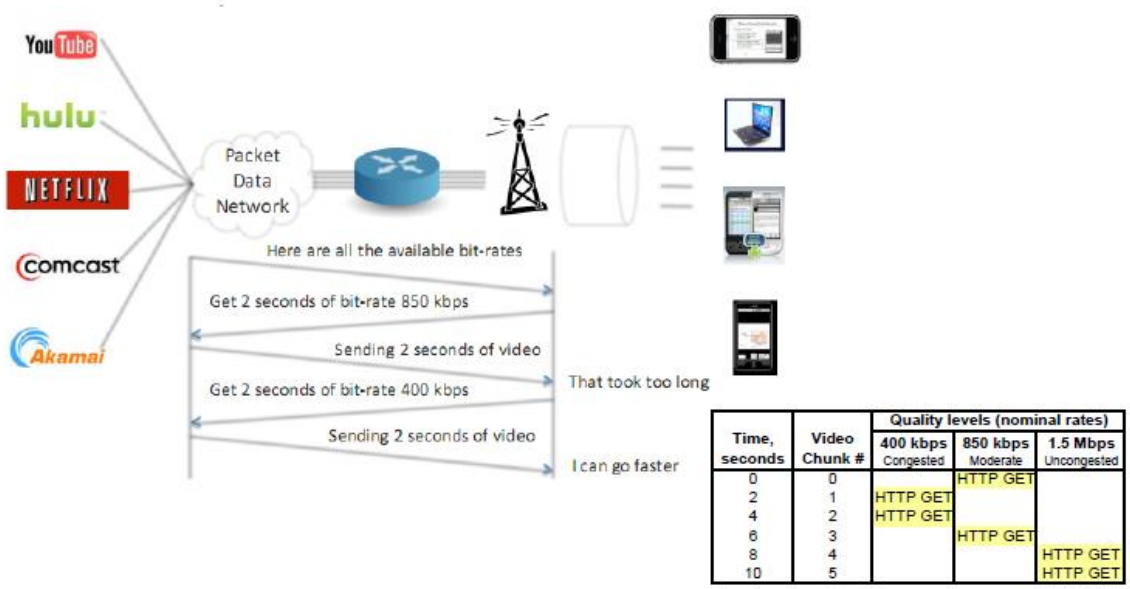

Fig. 6: Message exchanges during HTTP Adaptive Streaming

Video gets encoded into different formats and video qualities. HTTP Adaptive Streaming uses normal web servers and transmits data via HTTP protocol. Video files are divided into small non-overlapping video fragments (or intervals) in sequence, between 1 and 10 seconds long. Each interval is then encoded in multiple bit rates. The bit strings associated with these encodings are referred to as chunks (also called segments in MPEG DASH). The HAS client uses HTTP to request chunks – the first one usually at a low bit rate. If chunks are consistently delivered in a time shorter than the interval length, the Rate Determining Algorithm (RDA) in the client selects a higher bit rate for the next chunk (see Figure 5). In that way, the RDA continually senses the available throughput and adapts the video rate accordingly. In order to absorb mismatches between video rate and throughput, the video client maintains a play-out buffer. A larger playout buffer means a longer period of poor radio conditions can be tolerated. But it also increases the startup time. Assuming that the RDA is working properly, there should be no stalls while it automatically adjusts to the available bandwidth. *.[Saamer Akhshabi, Ali C. Begen, Constantine Dovrolis]*

Using HTTP has several advantages. It allows reuse of standard web technologies in the distribution and delivery of video. This helps minimize costs and avoids the need for special network technologies or configurations. Furthermore, HTTP operates over TCP, which automatically provides flow control and recovery from packet loss. Another advantage of HTTP-based video delivery is its ability to traverse firewalls through port 80. Finally, HTTP uses caching mechanisms on the web. Cached data is generally closer to the viewer and hence more easily retrievable. This capability should both decrease the total bandwidth costs associated with delivering the video, since more data can be served from web-based caches rather than the origin server, and improve quality of service.
Adaptive streaming enables viewers to start video play-back while the content is being downloaded. Consider two common sources for video streaming traffic in the Internet namely, Netflix and YouTube. Users can view Netflix and YouTube videos either on PCs, using a Web browser, or on mobile devices, using a Web browser or a mobile application. A mobile application is the native Netflix or YouTube application running on mobile devices.[*Müller, S. Lederer, and C. Timmerer*]

Netflix and YouTube use TCP to stream videos. A number of professional video content providers, such as Netflix, Hulu and BBC iPlayer, currently use HAS to deliver video content over unmanaged networks. HAS allows video delivery to adapt dynamically to the available bandwidth of a network while providing adequate quality of experience (QoE) for subscribers.

E. FUTURE WORK

Towards the goal of improving the state of the art, we plan to conduct more detailed measurements and analysis of the behaviors of existing video players. We will identify and categorize the shortcomings in using the plain HTTP protocol that the video players are trying to overcome. We aim at proposing a comprehensive solution for mobile video streaming over wireless

networks. We envision that proposed work will combine the best practices of the existing video players, possibly augmented by novel approaches.

CONCLUSION

This paper explored and analyzed the two most popular HTTP-based video streaming services (YouTube and Netflix) in terms of video traffic behavior in the network, while playing the videos on mobile devices (iOS and Android) over wireless networks (Wi-Fi, 3G and LTE). In the experiments, we point out that the network traffic behavior of downloading videos on-line depend on hardware performance, software running on clients' devices and network conditions between clients and video content servers. The measurements show that when a client requests a video, a video resolution is selected based on the device types regardless of OSs on clients' devices or network interfaces that clients access

While delivering a video to a client over HTTP, we also observed that a noticeable amount of video content is being discarded without being stored in the video playout buffer, after the successful delivery to the client. The content discarding occurs when a TCP connection is repeatedly terminated and established. In such cases, the video packets arrived at the client through the terminated TCP connection are discarded. The experimental results indicate that the average of video packet loss is 10.1%, and it may exceed 35% of its complete content. It causes additional mobile data usage paid by consumers and misuse of the limited network resources. Considering the increasing tendency of watching videos by

the mobile users and the scarcity of the network bandwidth, understanding the application traffic behavior is very important in order to develop an effective video delivery mechanism.

Playing videos on mobile devices depend on hardware performance, video players running on the devices and network conditions. While delivering a video to a client over HTTP, we also observed that a noticeable amount of video content gets discarded without being stored in the video playout buffer, after the successful delivery to the client device. The discarded video content occurs when a TCP connection is repeatedly terminated and established. In such cases, the video packets that arrived at the client through the terminated TCP connection get discarded.

REFERENCES

[1] http://en.wikipedia.org/wiki/Traffic_analysis
[2] Hyunwoo Nam, Bong Ho Kimy, Doru Caliny and Henning Schulzrinn, Columbia University, New York, NyBell Laboratories, Alcatel-Lucent, Murray Hill, NJ
[3] Adam Back; Ulf Möeller and Anton Stiglic (2001). "Traffic Analysis Attacks and Trade-Offs in Anonymity Providing systems". Springer Proceedings - 4th International Workshop Information Hiding.
[4] Saamer Akhshabi, Ali C. Begen, Constantine Dovrolis (2011). "An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP". In Proceedings of the second annual ACM conference on Multimedia systems (MMSys '11). New York, NY, USA: ACM.
[5] http://blogs.adobe.com/digitalmarketing/analytics/mobile-solution-series-advanced-mobile-traffic-analysis/
[6] Müller, S. Lederer, and C. Timmerer, "An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments," in Proceedings of the 4th Workshop on Mobile Video, ser. MoVid, Chapel Hill, North Carolina, USA, Feb. 2012.
[7] K. Balachandran, D. Calin, E. Kim, and K. Rege, "Clearmedia: A Proxy-based Architecture for Streaming Media Services over Wireless Networks," in Personal, Indoor and Mobile Radio Communications, PIMRC. IEEE 18th International Symposium on, Athens, Greece, Sep.2007.
[8] Cisco, "Visual Networking Index: Forecast and Methodology, 2012-2017," Cisco, Tech. Rep., May 2013.
[9] T.-Y. Huang, R. Johari, and N. McKeown, "Downton Abbey Without the Hiccups: Buffer-based Rate Adaptation for HTTP Video Streaming," in Proceedings of the 2013 ACM SIGCOMM Workshop on Future Humancentric Multimedia Networking, Hong Kong, China, Aug. 2013.
[10] C. Müller, S. Lederer, and C. Timmerer, "An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments," in Proceedings of the 4th Workshop on Mobile Video, ser. MoVid, Chapel Hill, Nort Carolina, USA, Feb. 2012.

AUTHOR PROFILE

Karishma Talan persuing M.E degree in Computer Science And Engineering from Prof. Ram Meghe Institute Of Technology and Research Badnera , Maharashtra , India

Dr. G.R. Bamnote Prof. & Head Of Computer Science And Engineering at Prof. Ram Meghe Institute Of Technology and Reasearch Badnera, Maharashtra, India

