A First Look at Performance of TV Streaming Sticks

Ayon Chakraborty*, Arani Bhattacharya*, Santosh Ghosh*, Samir R. Das*
Stony Brook University
Email: {aychakrabort, arbhattachar, sghosh, samir}@cs.stonybrook.edu

***Abstract*—Recent measurements show that more than half of the peak time Internet traffic is due to video streaming. Recent trends also suggest that consumers are increasingly receiving their TV content over the Internet via streaming appliances that are connected to the TV. We present the first systematic measurement study of a popular class of such devices that have the 'stick' form factor. The study covers streaming and network related performance using a widely used content server on the Internet (Netflix) and a local instrumented media server. On the user-end, we use three widely available mediasticks in the US – `Chromecast` , `Amazon Fire` and `Roku` . We observe that there are significant performance differentials across the streaming sticks. Our experiments show that `Amazon Fire` and `Chromecast` provide better user experience in the presence of varying network conditions, whereas `Roku` performs best at high stable bandwidth.**

I. Introduction

Much of the explosion of Internet traffic load in recent times is due to video streaming. According to the Sandvine study [1] more than 50% of the total Internet load in North America generated by fixed access networks during peak periods in 2014 came from video streaming – dominated by Netflix and Youtube. Interestingly, often such video traffic is not initiated by a computer or tablet class device – rather by embedded platforms that are either embedded inside the TV (so called 'smart TV') or attached to the TV via an audio/video input port. Examples of the latter type are Roku, Apple TV, Amazon Fire, Chromecast, etc. In the last few years these latter type of platforms have been increasing in popularity [2]. There are several reasons for this. First, they are fairly general purpose in the sense that they can be attached to any TV or display. Second, they are inexpensive with some models costing an order of magnitude less than a TV set. Third, the application/channel offerings are improving rapidly. There are several media reports describing how these devices are encouraging 'cord-cutting' [3] – the subscribers are increasingly using such platforms to get their TV programming from the Internet, rather than relying on more traditional means such as the cable or satellite. This trend is likely to continue and the day is not far when the Internet will be the primary delivery vehicle of television content.

With increasing popularity of such embedded platforms for Internet TV, consumer concerns abound. Popular discussion forums are showing a sharp increase in posts related to such platforms discussing various performance issues including device/hardware, streaming, network consumption and user experience (more on this in Section II). However, while there is a large body of literature on video streaming performance on the Internet (see, e.g., [4]), there is yet no systematic study of performance of these embedded TV platforms. Our specific goal in this work is to do a systematic study of their performance related to streaming and load on the network. The broader goal is to discover general or specific limitations, explore diversity of behaviors, discover possible trends.

While a broad range of embedded platforms exist that are either part of a TV or attached to a TV, we limit this study to 'streaming stick' platforms only. They are of the form factor of a regular USB flash drive, are USB-powered and carry an HDMI port that directly attach the device to the TV. Low cost (street price $\leq$ US$50) and ease of portability make these sticks very attractive to consumers. On-board processing and memory limitations limit programmability of these devices and also make them prone to performance issues. Thus, they make interesting case studies. Three of these sticks are popular in US market, viz., Roku stick [5], Chromecast [6] and Amazon Fire TV stick [7]. We use all three in our study.

Our major findings are as follows:

- The devices vary widely in their bit rate adaptation behavior. `Roku` is fairly aggressive in the choice of bit rate to be played. It adapts quickly but suffers from a long start up delay. In contrast, `Amazon Fire` is very conservative and adapts slowly, but offers a quicker startup. But overall all devices have poor startup delay.
- The devices also vary widely in their prefetching behavior. `Roku` and `Chromecast` tend to prefetch continuously. `Amazon Fire` on the other hand uses periodic prefetching. `Roku` is somewhat aggressive in prefetching and wastes a significant amount of data on viewer abandonment. In contrast, `Amazon Fire` incurs only modest loss in case of abandonment.
- The devices react differently to competing network flows. `Roku` adapts best and `Amazon Fire` worst.

The rest of the paper is organized as follows. In Section 2, we explain user concerns about performance by analyzing forum discussions.. In Sections 3, 4 and 5 we describe experimental setup, streaming and network related performance analyses respectively. We present related works in Section 6 and conclude in Section 7.

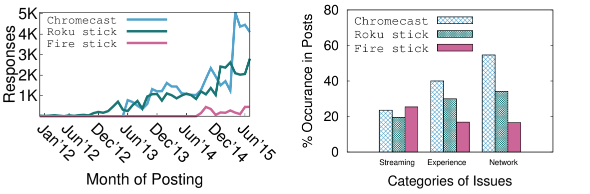
II. Study Background

A. *Streaming Sticks*

Our study uses all the three currently available USB-powered streaming sticks in the US market, viz., `Chromecast`, `Amazon Fire TV stick` and `Roku` (3500R). These devices have a form-factor of a USB flash drive and connect to the TV using a HDMI port. They are

"Internet speeds drop dramatically after ROKU plugged in"
"Netflix buffering and stalling recently?"
"HBO Go Loading Issues ..."
"FireTV Stick Freezes"
"Is Plex ever getting updated on the Fire TV Stick? 1080p constantly buffering."

Fig. 1: Titles of sample posts in Reddit showing user concerns



(a) Popularity Trends (b) Issues based on keyword analysis

Fig. 2: Analyzing public forum dataset

USB powered and are powered directly from a USB port on the TV. They connect to the Internet via WiFi.

Just like the smartphones the streaming sticks also have an application ecosystem built around it. Every streaming service or vendor (e.g., Netflix, Hulu, HBO Go, Youtube, Sling TV, Plex etc.) provides an application ('channel' in `Roku`) for the streaming stick platform. All the devices expose SDKs to facilitate development of media streaming applications. While `Roku` uses the Brightscript language [8], `Amazon Fire` and `Chromecast` support two methods for application development, viz., Android apps and HTML5 web apps. The SDKs provide a rich set of APIs to develop media applications fetching content from remote CDNs or the local network. Applications developed for these devices can be published via the appstore of the respective devices.

B. Popularity Trends and User Complaints

In order to drive the measurement study we need to have at least a broad idea of performance issues that might arise in the streaming sticks we consider. To develop this understanding we crawl public forums for data related to reviews or user complaints concerning these sticks. Our data set consists of approximately 50K posts/comments spanning from January, 2012 through August, 2015 on forums such as `reddit.com`, `forums.roku.com` and Google's chromecast forum. More than half of the data set is from `reddit.com` which is one of the most popular discussion/Q&A forums on the Internet. Analysis of this data set reveals a few interesting insights. First, number of posts related to these devices are increasing very fast. See Fig. 2a. `Chromecast` 's popularity has risen significantly in recent months relative to the other two. We also verified that this trend matches quite well with the Google search trends (as per `https://www.google.com/trends/`) for the terms `Chromecast` , `Roku` and `Amazon Fire` .

We apply simple text mining approaches to understand the nature of the concerns presented in the forum posts. Broadly we look for three specific categories while analyzing the posts. First, we look into issues concerning overall experience related problems, that can include application crashes, non-responsive

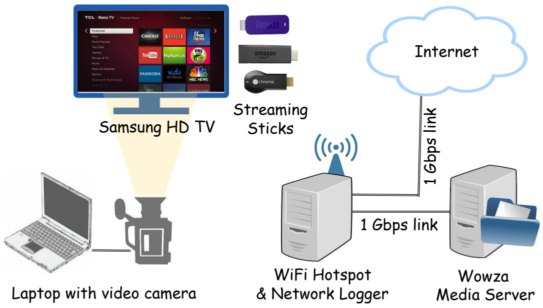


Fig. 3: Schematic of our testbed setup

UIs, setup problem etc. The second category of problems are related to purely streaming related concerns, for example, video buffering problems or playing low resolution video etc. Third, we look into concerns regarding the network that includes WiFi or Internet connectivity problems, aggressive prefetching, significant amount of broadband data consumption etc. Figure 1 show some samples of user concerns.

Our text mining technique is based on keyword matching. For each category (streaming, experience and network) we list a set of words related to that category and look for the occurrence of some word in the set in the post's text. Note that a post may belong to multiple categories or it may not match to any category. The keywords for each category were chosen manually. For example, some keywords for category network are *'connection problem', 'qos', 'slow', 'speed', 'bandwidth'* etc. For streaming they are *'buffering', 'freeze', 'stall', 'resolution'* etc, and for general experience they are *'crash', 'frustrated', 'sucks', 'bad experience'* etc.

Figure 2b shows these individual categories as a fraction of the total number of posts for each of the streaming sticks. For both `Roku` and `Chromecast` network related issues frequent more than general experience or streaming problems. For `Chromecast` approximately half of the posts relate to network issues while for `Roku` it is roughly 30%. However, majority of the posts for `Amazon Fire` point to streaming related problems. Also, for all of the three sticks streaming related problems account for roughly 20% of the posts. In our evaluation we limit ourselves to streaming and network-related performance issues and draw comparisons across the three platforms.

III. Experimental Setup

Our testbed consists of the following: a) a Samsung HD TV that supports a maximum resolution of 1080p and has a HDMI port, b) three streaming sticks considered and c) a WiFi access point in our lab connected to a 1 Gbps campus backbone. The streaming sticks are powered through USB connection from the TV itself. The WiFi access point is created using a hotspot in a moderately well provisioned (Core2Duo processor with 6GB memory) desktop computer with a 1 Gbps wired Ethernet connection to the campus backbone (for Internet-hosted services) or to another, similarly equipped desktop hosting the Wowza [9] media server (for locally hosted experiments). The computer serving as the AP runs Ubuntu Linux 14.04. We use an Atheros-C9 WiFi card (2.4GHz-802.11n, running ath9k

Resolution	Average Bitrate		Resolution	Average Bitrate
320 x 240	235Kbps		720 x 480	1750Kbps
384 x 288	375Kpps		1280 x 720	2350Kbps
512 x 384	560Kbps		1280 x 720	3000Kbps
512 x 384	750Kbps		1920 x 1080	4300Kbps
640 x 480	1050Kbps		1920 x 1080	5800Kbps

TABLE I: Video resolutions and average bitrates used in the study

wireless driver) in the desktop's mini-PCI slot and configure it in the master mode to serve as an access point. The benefit of such software AP solution is that it enables traffic sniffing and logging directly at the AP (i.e., the desktop). Also, it allows for changing the parameters of the wireless link to do controlled experiments. For example we can emulate a wireless link with a certain throughput or introduce desired latencies or losses. This is done with the help of Linux utilities `tc` and `netem`. We configure all the three streaming sticks to connect to our hotspot. A schematic of our testbed is shown in Figure 3. We have used popular 'speedtest' applications for all the three streaming platforms – *Speed4Cast* for `Chromecast` , *SpeedTest* channel for `Roku` and *Ookla Speed Test* app for `Amazon Fire` and obtained an average network bandwidth of 30 Mbps. This is more than enough for streaming contents we use in this study. Thus, the Internet connection is not a bottleneck.

In the study, part of our focus will be the performance of adaptive bitrate streaming particularly HTTP Live Streaming(HLS) [10], for the chosen platforms and services. This requires us to learn the actual resolution the video is being streamed at every instant. However, determining the resolution requires some work. This is because the coding rate of the video varies even at the same resolution and thus network load on the backhaul is not always a good indicator of the quality of the video being played. We developed a separate system to track the video resolution and average video bit rates. The basic idea is to overlay the video resolution and bit rate on the video itself and use an external video camera to record the video being played (Fig. 3). Post-processing of the recording reveals the resolutions/bit rates played at every instant as well as whether the video is stalling.

Netflix hosts a video called *'Example Short 23.976'* (11:04 mins) to help users to do exactly the above. This video displays resolution and bit rate on-screen. We use Netflix and this video extensively in our rate adaptation study. A part of our study also requires us to have server-side instrumentation (e.g., segment fetch, request timings etc.) that is not possible with a hosted service like Netflix. For this we use a locally hosted media streaming server (Wowza [9])[1] that supports adaptive streaming. For locally hosted experiments, we use the standard *Big Buck Bunny* video [11] (9:56 mins) and `ffmpeg` tool to create the different bitrate/resolution versions for the video exactly as in Table I. Similar to Netflix, we put a watermark on the video mentioning bitrate and resolution.

[1]The app store for the streaming sticks do not have the client-side app for Wowza . So we developed our own app for this study.

IV. Streaming Performance

We present the performance study in two parts. In this section we present results related to streaming performance and viewer's quality of experience (QoE). In the following section we will present network related performance measures.

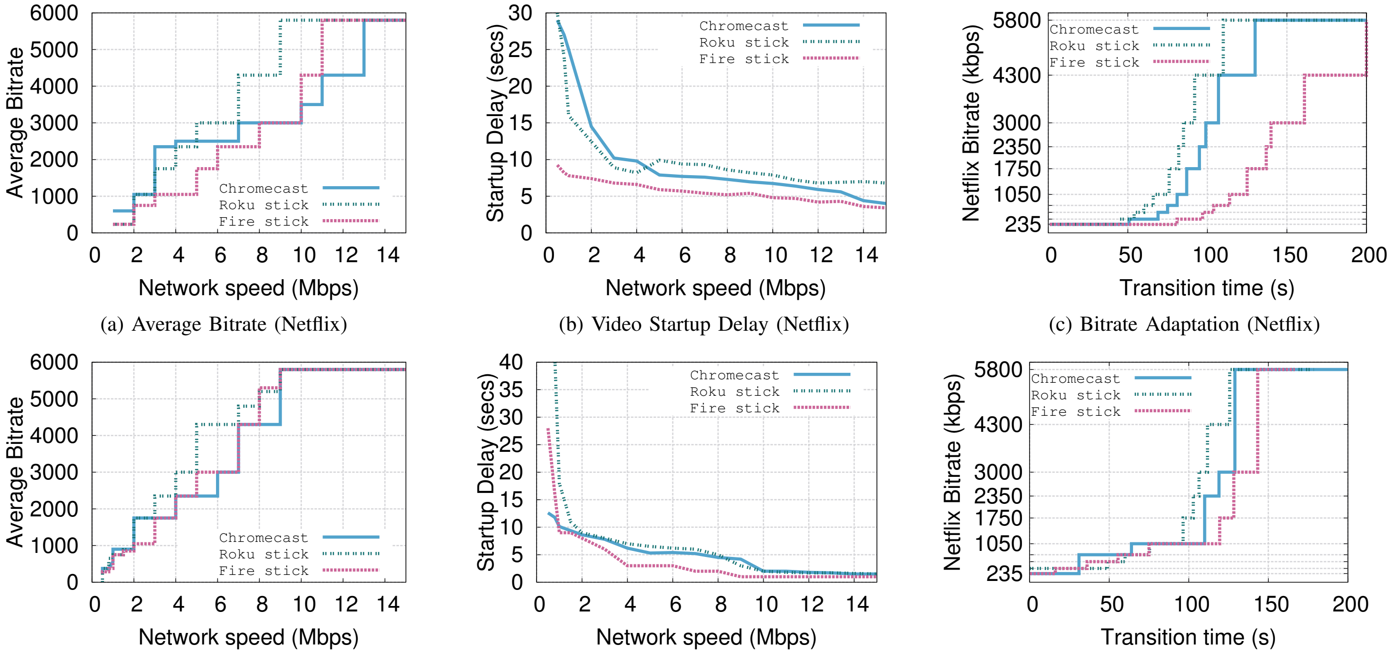
To study streaming performance we evaluate four QoE metrics of common interest: i) *average bit rate* played for a given available network bandwidth, ii) *video startup delay*, iii) *video stalls* during playback – frequency and duration, iv) *bit rate adaptation time* – how long it takes to adapt to a different bit rate when the network condition changes. All experiments use the specific video clips mentioned in Section III and Netflix or Wowza as the video server. The main results are summarized in Fig. 4.

Average Bitrate: Plots in Fig. 4a and 4d show the average bit rate played (see Table I for the rates) for the three streaming sticks for different network conditions. Here, network condition refers to available WiFi link bandwidth. We first play each video at bandwidths where the link bandwidth ranges from 0.5 Mbps to 15 Mbps. This link is the bottleneck link given the backhaul bandwidth is significantly higher (Section III). The average bit rate here is simply the time average of different video bit rates played out in the TV. We note that `Roku` always tries to play a better resolution video over `Chromecast` or `Amazon Fire` sticks. Thus, Roku offers the best Quality of Experience (QoE) Both Netflix and Wowza are roughly similar in behavior, though Wowza offers higher video quality for the same network condition. This is due to additional latency over the Internet for Netflix that is not emulated for Wowza in our experiments.

Video Startup Delay: This delay is the time elapsed between the events when the video is requested to be played and the time when the video is rendered on the screen. This happens because the streaming stick fills up its video buffer with a certain length of video before it actually starts the playback. Fig. 4b and 4e show the startup delays for different available network bandwidths.[2] At speeds $\geq$ 5 Mbps, `Roku` requesting higher resolution videos faces a higher startup delay. This trend is pretty much the same for both Netflix and Wowza. The startup delay largely depends on the player's bitrate adaptation algorithm that attempts to match the video bit rate with the available network bandwidth. Overestimation of the available bandwidth can end up in a higher startup delay. For example, at 1 Mbps, `Roku` is unable to start the Wowza video. On the other hand, the `Amazon Fire` has a lower startup delay of 18 s at 1 Mbps bandwidth.

We have analyzed Wowza server's logs for `Amazon Fire` to understand this better. The logs show that it requests the first segment of the video at a much lower resolution that the link could potentially support. This provides the user a more interactive experience however such conservative estimate creates a delay in achieving the optimal resolution.

[2]When the video is requested and yet to render, the screen is blank with a 'loading' icon. The screen record is used to estimate the time when the application is launched and the time when the first video frame is rendered.



(a) Average Bitrate (Netflix) (b) Video Startup Delay (Netflix) (c) Bitrate Adaptation (Netflix)

(d) Average Bitrate (Wowza) (e) Video Startup Delay (Wowza) (f) Bitrate Adaptation (Wowza)

Fig. 4: QoE metrics for the three streaming sticks using Netflix and Wowza media streaming servers

Overall the startup delays for the devices can be considered poor enough to hurt user engagement. Even in the best network condition the smallest delay is ≈ 4 sec for Netflix. A recent measurement study [12] concludes that with this level of delay a non-negligible fraction (upto $\approx$ 15%) of viewers would abandon the video. The same study also concludes that viewer impatience is more significant for better connected players. The streaming sticks must do more to reduce startup delay.

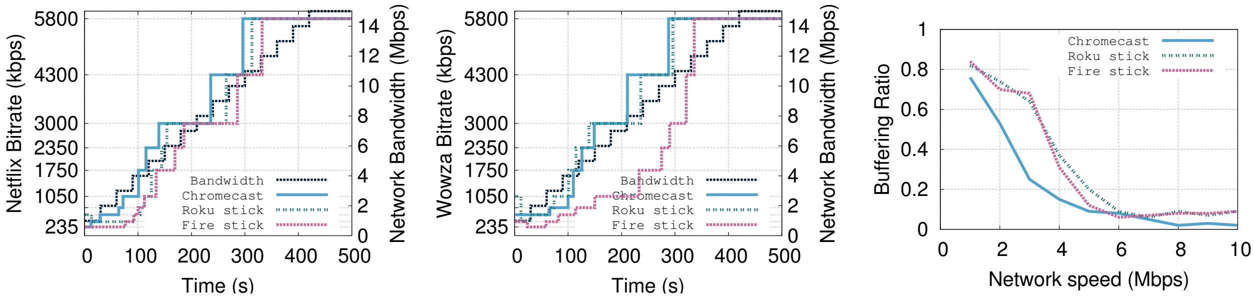
Bitrate Adaptation Time: This refers to the time the player requires to estimate the optimal video resolution that could be played without stall. For understanding the adaptation time we do two experiments. In the first experiment, we start with a poor network bandwidth (1 Mbps) and continue until the players stabilize the video resolution and bit rate. At this point of time we make the make the network faster – bandwidth changed to 20 Mbps. Assume that this is time = 0. The trace of video bit rate versus time is shown in Fig. 4c and 4f. Note that `Amazon Fire` adapts very conservatively relative to the rest ($\approx 2\times$ slower than `Roku` for Netflix). In a second experiment we increase the network bandwidth from 1 Mbps to 8 Mbps in steps of 1 Mbps at every 60 seconds interval. In this case their performances are somewhat similar (and video is adapted at a faster pace) though `Amazon Fire` still lags behind relative to the rest (Fig. 5a and 5b).

Video Stalls: The adaptive streaming strikes a tradeoff between video bit rate/resolution and streaming interruptions (freezes, stalls). We have seen a large number of stalling issues when fixed rate videos are used in the HD mode. Such stalls happen across all the streaming sticks. The *buffering ratio*, or the percentage of the video session duration spent in buffering state, is shown in Fig. 5c for all three sticks while streaming HD videos over Wowza. Here, `Chromecast` works somewhat better specifically in poor network conditions. Note that in our study in Section II we have encountered a number of mentions of video stall/buffering related issues. On closer look we find that a majority of them refer to the Plex media server [13] that does not support adaptive streaming. Hence attempting to play HD videos invariably faces stalls in congested networks.

Discussion: A recent study on bitrate adaptation has shown that the quality of experience of a streaming service can be improved in an improving network by increasing the bitrate as soon as possible [14]. In contrast, good quality of experience in a deteriorating network is obtained by making the bitrate reduction gradual. Since `Roku` aims to provide a higher bitrate, it provides the best user experience when the network is stable. However, `Roku` is slow to adapt to both increasing and decreasing bandwidths. In contrast, while `Amazon Fire` and `Chromecast` provide a poorer experience at lower bandwidths, they adapt to a deteriorating bandwidth condition much more smoothly. Thus, in the presence of varying network conditions, `Amazon Fire` and `Chromecast` provide better user experience.

V. Network Load

In this section we evaluate how video data is prefetched by the streaming sticks and related traffic load on the network. All adaptive streaming players prefetch and buffer video segments before playing. More aggressive prefetching may provide a better viewing experience by preventing stalls. However, there are potential costs for aggressive prefetching. First, if the



(a) Bitrate Adaptation (Netflix) (b) Bitrate Adaptation (Wowza) (c) Buffering Ratio (Wowza)

Fig. 5: Bitrate adaptation timeline on a gradually changing network and impact of segment size.

viewer abandons the video in between (this is a relatively common occurrence particularly for short videos [12]), prefetched segments are wasted. This leads to wastage of bandwidth. Second, if the network condition improves it can lead to one of two inefficiencies. Either viewing experience suffers as the prefetched segments could be of lower resolution than the current bandwidth allows, or bandwidth is wasted as these segments are to be fetched again at a higher resolution abandoning the previously fetched segments [15]. In addition, impact of competing flows in the network on video streaming is of interest [16].

Prefetching Behavior: To study the prefetching behavior, we keep the network bandwidth fixed and stream the *Big Buck Bunny* video using each of the three sticks using Wowza. We show sample traces of the download speed in Mbps observed on the backhaul. See Fig. 6. These specific plots are samples when the bandwidth is fixed at 20 Mbps. But the qualitative nature of the plot is similar for other bandwidths and not shown for brevity. Note that the data fetching behavior is very different in the three streaming sticks. `Roku` and `Chromecast` fetch almost continuously and `Roku` somewhat more aggressively, especially at the beginning. Recall the high startup delay for `Roku` and generally better quality viewing. `Amazon Fire` , in contrast, exhibits a periodic *on-off behavior* [17]. It fetches in short spurts. We hypothesize that the continuous prefetching of `Roku` and `Chromecast` enable them to do a better bandwidth estimation for rate adaptation relative to `Amazon Fire` . We repeated similar studies for longer (30–60 mins) videos. The general nature of the behavior remains unchanged.

Data Wastage on Abandonment: Given the diversity of the prefetching behavior we also study how much video is wasted if the viewer abandons the video in midstream. See Fig. 7a. This plot is derived from the same experiment in Fig. 6 by noting at every time instant how much of the video is delivered and actually played. The difference is plotted across a normalized timeline. As expected, `Roku` produces a very significant wastage of the viewer abandons around the midpoint. `Amazon Fire` , on the other hand, has only a modest waste that is independent of playing time. While this study is specifically done for a short video, note that the viewer abandonment is more frequent for short videos [12].

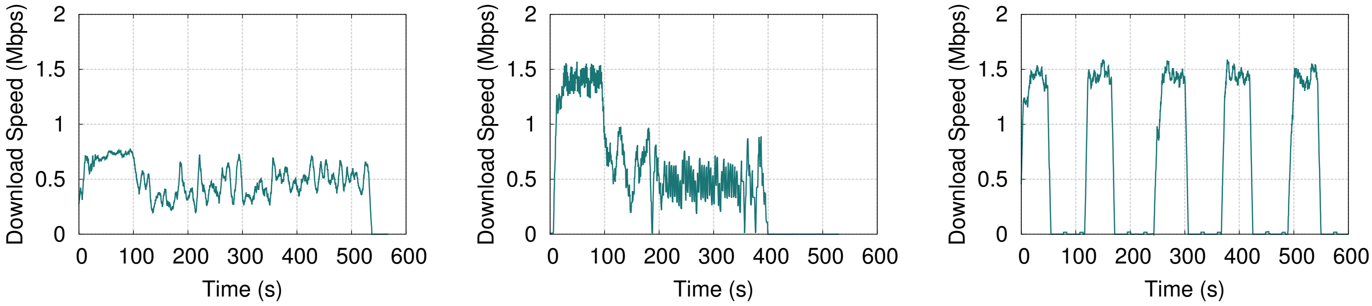
Effect of Competing Flows: Here, we study the performance of the streaming sticks in presence of background traffic. We choose two types of background traffic: a) HTTP download and b) video streaming. In the first case we use a laptop connected to our access point to download a 100 MB file from the local network via HTTP. In the second case we use a similar setup to stream a HD video from Youtube. These are indicative of typical background traffic present in homes. See Fig. 7b and 7c for the throughput and video bit rate plots respectively. With file download as background traffic, we see conservative bitrate adaptation behavior in the streaming sticks, particularly in case of `Amazon Fire` . Here, the sticks should be able to play the video at the highest bit rate possible (tops at about ≈ 6 Mbps) given fair sharing and network capacity of about ≈30 Mbps. However, all sticks play the video at a far poorer quality indicative of poor adaptation behavior. Similar adaptation issues are also reported in [16]. With video streaming as background traffic the adaptation is better as perhaps the background load is lower. Note also that `Amazon Fire` is very conservative about the bandwidth estimation and plays a significantly poorer quality video both kinds of background traffic relative to `Roku` and `Chromecast` . `Roku` remains the most aggressive player.

VI. Related Works

Media streaming over the Internet and its key challenges have been extensively studied in literature [18]. While one direction of work addresses issues like improving bitrate adaptation [19] or improving fairness [20], others explore issues related to performance measurement of video streaming [21] and related user experience/engagement [22]. The latter type of work includes real applications as well as new platforms and environments (e.g., home networks). For example, there are studies that report performance difference among various video players depending on whether it is played on a PC or mobile client [23]. Dimopoulos et al. [24] identify bottlenecks that affect user experience in mobile video streaming applications. Though media streaming sticks are extremely popular, user concerns abound. To the best of our knowledge this is the first work to study streaming performance of such sticks.

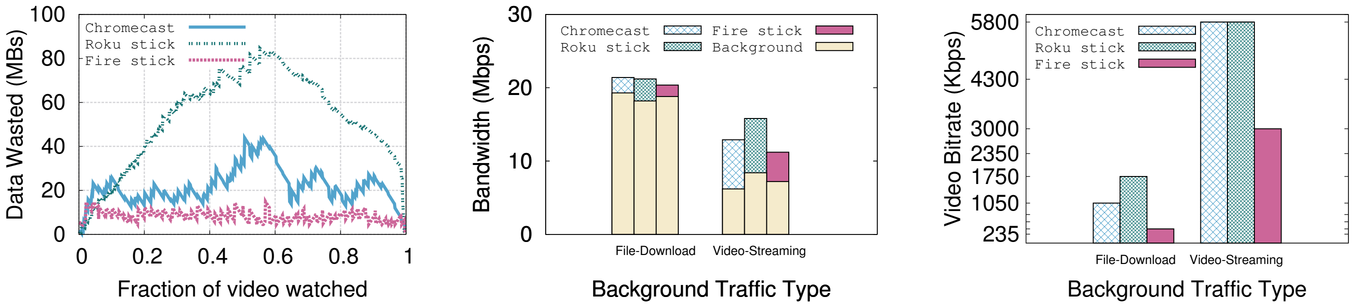
VII. Conclusions

In this paper, we presented a first attempt to study the performance of TV streaming sticks. Following up on posts



(a) Chromecast (b) Roku stick (c) Fire stick

Fig. 6: Data streaming behavior for the three devices for a short video on Wowza



(a) Data wastage on abandonment (b) Video Throughput (c) Video Bitrate

Fig. 7: Bandwidth wastage and effect of competing flow

in various forums regarding the type of performance issues typical users are concerned with, we analyzed two categories of performance issues – streaming performance and network loading. Our analysis shows that significant performance differentials exist across the streaming sticks. They could be aggressive or conservative in adapting to changing network conditions and experience significantly different amounts of startup delays. Aggressive prefetching leads to significant amount of data wastage in case of viewer abandonment. We hope that our work will encourage deeper studies across a larger class of TV streaming devices and a broader understanding of performance tradeoffs, as well as improvements in streaming and network performance of such devices.

Acknowledgment

This research was (partially) supported by the Ministry of Science, ICT and Future Planning (MSIP), Korea, under the "ICT Consilience Creative Program" (reference number IITP-2015-R0346-15-1007) supervised by the Institute for Information and Communications Technology Promotion (IITP).

References

[1] "Sandvine, Global Internet Phenomena," http://bit.ly/1zRaB0K.
[2] "Thinking of buying a smart TV? Think Roku instead. (News Article) ," http://bit.ly/1QzrSnn.
[3] "Roku 4 is Continuing to Help Cord Cutters – Here's How, (News Article)," http://bit.ly/1VEzVWo.
[4] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming video over the internet: approaches and directions," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 3, 2001.
[5] "Roku Streaming Stick." https://www.roku.com/products/streaming-stick.
[6] "Google Chromecast." https://www.google.com/intl/en_us/chromecast/.
[7] "Amazon Fire TV Stick." http://www.amazon.com/fire-tv-stick.
[8] "BrightScript Language Reference," http://bit.ly/1LSnqSJ.
[9] "Wowza Streaming Engine." http://www.wowza.com/products/streaming-engine.
[10] "HTTP Live Streaming." https://developer.apple.com/streaming/.
[11] "Big Buck Bunny," https://peach.blender.org/download/.
[12] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs," *Networking, IEEE/ACM Transactions on*, 2013.
[13] "Plex Media Server," https://plex.tv/ad/everywhere.
[14] S. Tavakoli, S. Egger, M. Seufert, R. Schatz, K. Brunnström, and N. García, "Perceptual quality of http adaptive streaming strategies: Cross-experimental analysis of multi-laboratory and crowdsourced subjective studies," *IEEE JSAC*, Aug 2016.
[15] A. Mansy, M. Ammar, J. Chandrashekar, and A. Sheth, "Characterizing client behavior of commercial mobile video streaming services," in *Proc. ACM MobVid 2014.*
[16] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: picking a video streaming rate is hard," in *Proc. ACM IMC 2012.*
[17] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proc. ACM CoNEXT 2011.*
[18] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of http adaptive streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.
[19] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," *ACM SIGCOMM Computer Communication Review*, 2015.
[20] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proc. ACM CoNEXT 2012.*
[21] L. Plissonneau and E. Biersack, "A longitudinal view of http video streaming performance," in *Proc. ACM MMSys 2012.*
[22] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *Proc. ACM SIGCOMM 2011.*
[23] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "Youtube everywhere: Impact of device and infrastructure synergies on user experience," in *Proc. ACM IMC 2011.*
[24] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, K. Papagiannaki, and P. Steenkiste, "Identifying the root cause of video streaming issues on mobile devices," in *Proceedings of 2015 ACM CoNEXT*, 2015.