Analyzing IPTV Set-Top Box Crashes

Han Hee Song[§], Zihui Ge[‡], Ajay Mahimkar[‡], Jia Wang[‡], Jennifer Yates[‡], Yin Zhang[§]
The University of Texas at Austin, [§]
Austin, TX, USA
{hhsong,yzhang}@cs.utexas.edu
AT&T Labs – Research,[‡]
Florham Park, NJ, USA
{gezihui,mahimkar,jiawang,jyates}@research.att.com

ABSTRACT

Recent advances in residential broadband access technologies have led to a wave of commercial IPTV deployments. As IPTV services are rolled out at scale, it is essential for IPTV systems to maintain ultra-high reliability and performance. A major issue that disrupts IPTV service is the crash of the set-top box (STB) software. The STB directly resides inside the consumer's home network and provides the essential interface to both the user and the network to deliver rich content that goes well beyond traditional TV. To understand the potential causes of STB crashes, we perform an in-depth statistical analysis focused on the relationships between STB crashes, video stream content, and user activities. Our initial results suggest that (i) impaired video streams may cause STB crashes, and (ii) continuous STB usage may gradually degrade the STB health over time.

Categories and Subject Descriptors

C.4 [**Computer-Performance of Systems**]: Measurement techniques

General Terms

Measurement, Reliability

Keywords

Analysis, IPTV, Set-Top Box, Crash, Video, Home Networks

1. INTRODUCTION

Background and motivations. Although a relatively new technology, Internet Protocol Television (IPTV) has been successfully rolled out within numerous large-scale commercial deployments across the globe [16]. IPTV service providers are constantly striving for service improvements in the deployed IPTV systems because high service quality and strong customer satisfaction are key to commercial success.

An important component of the IPTV system is the IPTV Set-Top Box (STB), which resides directly inside the consumer's home network. The STB provides the essential interface to both the user and the network to enable the delivery of rich content that goes well beyond traditional TV [3]. One of the top issues that IPTV service providers are concerned with is STB software crashes. A STB software crash can seriously impact the consumer's TV viewing experience as it causes noticeable service disruptions (perhaps repeatedly) for minutes. Although individual STBs rarely crash, the aggregate number of crashes across entire service areas could be non-negligible. We are therefore interested in understanding the potential causes of STB software crashes, with the goal that these crashes will be driven out of the deployed IPTV system permanently.

Given that an IPTV STB is in essence a computer interacting with IPTV servers, the IPTV network, and users, analyzing STB crash is far from a simple task. Functioning as the only end-user device in the entire IPTV system, the STB boasts capabilities to run every application that the IPTV service provides including live TV, Video on Demand (VoD), choice program, and even online games. Supporting such diverse capabilities, the software and hardware architecture of an STB is highly intricate by nature [2, 14]. In addition, since an STB is interconnected with interwoven servers and network systems on one side while providing interactive services to subscribers on the other side, the root cause of a given STB crash can lie anywhere within the systems that the STB interfaces with. Analysis of STB crashes thus requires investigation of diverse aspects of the IPTV system: from network components transporting control and data packets, to video stream content fed into the STB, through to users' actions. For diagnosing IPTV service performance degradations, Giza [11] examined the possible correlation between IPTV network events and STB crashes and found no significant relationships. Given that network issues are apparently not the major cause of STB crashes, we focus on the other two aspects of IPTV system in this paper.

Approach and contributions. We take a perspective from the home network and analyze how STB software crashes are related to (i) video stream content inputted and processed by the STBs and (ii) users' interaction with the STBs. We collect a vast set of diverse measurements associated with the IPTV deployment, including logs collected from over four million STBs from across the United States, anonymized usage information from over two million subscribers, and video stream content information fed into the entire set of STBs. We commence by thoroughly characterizing the temporal and spatial properties of STB crashes (Section 3). We then execute an in-depth analysis of STB crashes by applying various statistical techniques for isolating oddities in the correlations and revealing the potential causes of STB crashes.

1. *STB crashes and video stream content* (Section 4). To assess the potential for video stream impairments (*e.g.*, data

corruption, mal-formed codec, noisy signal, etc) to cause STB crashes, we evaluate the STBs that crashed while watching each live TV channel. Using statistical methods finding the mutual dependency between STB crashes and TV channels, we identify crash-prone channels (*i.e.*, the video streams with impairment), suggesting a strong possibility of impaired video streams causing STB crashes.

2. *STB crashes and user activity* (Section 5). To analyze the possibility of user activities triggering STB crashes, we evaluate the effects of user inputs on future crash events as well as the effects of prolonged STB usage on crash events. We find that while accumulation of user input to STB is not closely related to STB crashes, there is evidence that extensive usage over a long period of time can negatively impact the health of STBs.

2. BACKGROUND

In this section, we first present an overview of the IPTV service architecture and the Set-Top-Box (STB) inside the home. We then describe the data set used to analyze STB software crashes.

2.1 Overview of IPTV and Set-Top-Box

An IPTV network typically utilizes a hierarchical structure (similar to the one described in [11]): video programs encoded and packetized at Super Hub Office (SHO) are distributed to Video Hub Offices (VHO) governing each metropolitan areas in the nation. An IP multicast protocol then delivers the video towards millions of home networks. A Set-Top-Box (STB) in an IPTV system works as the user-side end device that receives the packetized video streams, decodes them and sends them to a TV for display. In addition to supporting both standard and high definition live TV channels, STBs also support advanced features such as DVR, VoD, picture-in-picture, online gaming, and chatting. STBs can thus be considered as sophisticated computer devices (equipped with specialized operating systems) [2, 14].

As with other types of computer systems, a STB can crash. Such crashes may involve recovery times of several minutes (similar to rebooting a PC) - disrupting a user's TV viewing and consequently negatively impacting the user's TV experience. Although software crashes are relatively rare events for individual STBs, aggregating across all STBs in a given service area does result in non-negligible numbers of crashes, and thus provide an opportunity for service improvement. In order to ensure a good quality of service, it is thus important to detect, diagnose and mitigate STB crashes in a timely fashion. STB crashes are considered to be one of the key performance related issues faced by IPTV service providers.

In this study, we focus on STB software crashes as opposed to hardware crashes as the former can be diagnosable by inspecting its relation with its inputs while the latter can happen from other causes such as module or device failures which are less likely to be captured by the input logs. As STBs are interfacing with both the IPTV distribution network and the users, we consider the inputs from both sides in determining the cause of STB crashes: video stream content received from the IPTV network, and user activities.

2.2 Dataset Description

We collected data from a large commercial IPTV service provider, with logs collected from over two million home networks with over four million STBs. In addition to the STB crash event logs, we obtained video stream information and user activity logs. To protect privacy, all the logs we consider are anonymized.

STB Crash Logs. We obtained crash event logs from each STB served by this IPTV service provider. In the STB software crash event logs, each message contain the timestamp and one of four crash error codes: managed crash, native crash, watch dog reboot, and out of memory error. A managed crash occurs when code executed inside a protected environment fails. A native crash occurs when a fatality occurs in code executed outside the protected environment (*e.g.*, a device driver or kernel). A watch dog reboot occurs when an STB hangs and the watch dog timer thus expires. And finally - as the name suggests - an "out of memory" error occurs when the STB software runs out of memory. All of our analyses of STB crashes were executed separately for the four different types of crashes. However, our general finding is that the managed and native crashes dominate software crash events and exhibit similar characteristics, while the rarity of watch dog reboot and out of memory error makes them statistically insignificant. Thus, in the interest of brevity, we combine all four crash types in the results presented in this paper.

Video Streaming Information. Software can be vulnerable to errors in its input. Therefore, it is theoretically possible that certain video stream impairments can cause STBs to crash. To investigate possible linkages between crashes and impaired video streams, we collect channel tuning logs from each STB. These tuning logs denote TV viewing sessions that last longer than 20 seconds (*i.e.*, all the channel zapping actions that stay less than 20 seconds on a channel are discarded from the logs).

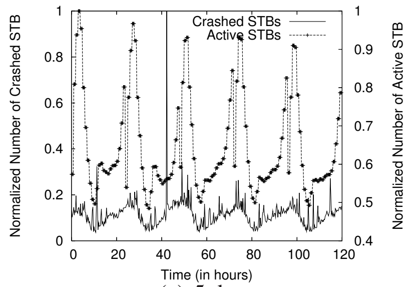
User Activity Data. As a highly user interactive system, certain user activity patterns or habits can increase the chance of the software malfunctioning and causing the STB to crash. We collect the following five types of user activities from all STBs: Power on/off, channel switch, video stream control (*e.g.*, playback, fast forward, rewind, etc), on-screen menu invocation, and application initiation.

3. SPATIO-TEMPORAL ANALYSIS OF STB CRASH

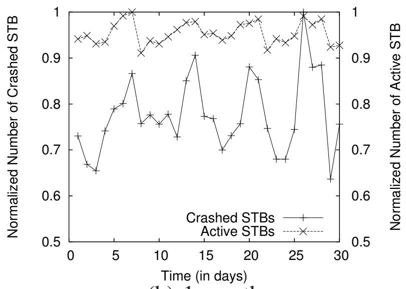
In this section, we present the temporal and spatial analysis of Set-Top-Box (STB) software crashes using data collected over a one month period [1]. We observe diurnal and weekly patterns in the rate of STB crashes as well as correlated crashes across multiple STBs and different VHOs.

Temporal Pattern of STB Crash. Fig. 1 shows the diurnal and weekly pattern of the software crashes aggregated across all of the STBs. The timestamps across the four time-zones are converted to GMT. Fig. 1(a) shows the trends over five consecutive days; the solid curve corresponding to the y axis on the left hand side depicts the hourly STB crash count normalized by the peak hourly STB crash count. As the universe set, we only consider active STBs that are turned on at the time of analysis (as opposed to inactive, turned off STBs). The dashed curve corresponding to the y axis on the right side depicts the normalized hourly active STB counts. From the figures, we observe both periodic behaviors as well as spikes in the STB crashes. The STB crash count is high during evening prime time TV viewing and low during early morning hours. The diurnal pattern of the STB crash counts aligns well with that of the active STB counts, which suggests that the STB crash counts are roughly proportional to the number of active STB . This indicates that the STBs are more likely to crash during their active periods. Further, these crashes could be caused by the user activ-

[1]To protect proprietary information, we normalize some information in the results to the extent that the normalization does not obstruct interpretation of results

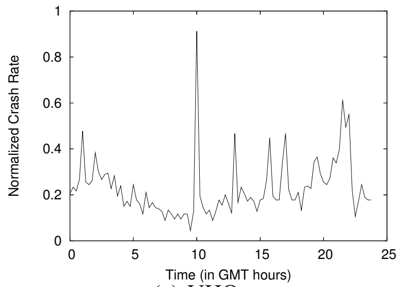


(a) 5 days

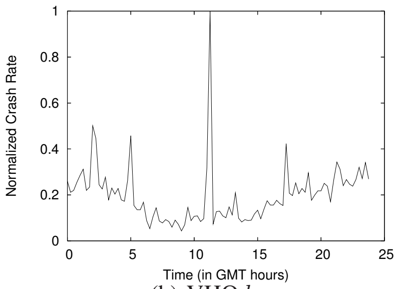


(b) 1 month

Figure 1: Temporal and diurnal patterns of STB crash



(a) VHO a



(b) VHO b

Figure 2: STB crash rate of different VHOs for one day.

ities. The spikes in STB crash counts, on the other hand, have a strong coocurrence that are more likely caused by a common factor (*e.g.*, service outage in an area or corrupted video stream received from external sources). Thus, the diverse temporal patterns (spiky and periodic) motivated us to explore potential root causes of STB crashes from both *video source information* that is common across some STBs at a given time snapshot as well as *user activities* that are less concurrent but diurnal.

Spatial Pattern of STB Crash. Fig. 2 shows the STB crash ratio aggregated over 15 minutes time bin for two different VHOs within a single day. To compare two subfigures under the same scale factor, the plots are normalized by the peak crash rate value appearing in Fig. 2 (b). First, we observe similar diurnal patterns across VHOs, with generally high crash rate around 1AM GMT corresponding to prime time in the U.S. (7 PM Central Time) and low crash rates around 9AM in GMT corresponding to early morning local time (3 AM CST). We also observe sporadic spikes in STB crash ratios for some VHOs as shown in Fig. 2. Moreover, some of these spikes co-occur across multiple VHOs. Figs. 2 (a) and (b) show one such example – spikes occurred between 10 and 11 GMT in VHOs a and b indicating that a significant number of STBs from both VHOs came to crash during the same time interval. This observation suggests that a common root cause is likely to contribute to the spikes in the STB crash ratio across multiple VHOs.

4. DOES VIDEO CONTENT MATTER?

In this section, we focus on identifying the relationship between STB software crashes and the video stream that the IPTV users are watching. It sounds far fetched yet still conceivable that a poorly formatted or impaired video stream may crash the video decoder and even the operating system that the decoder is hosted in. The IPTV provider performs rigorous testing for any newly incorporated content source (*e.g.*, national or regional media company) to ensure the quality of their video stream. However, given the complexity of the encoding and the variety of video data, even trusted sources cannot be prevented from having glitches.

4.1 Methodology

Our approach is operational data-driven, using statistical mining to either accept or reject the hypothesis that some video streams tend to crash STBs at a frequency higher than explainable by simple coincidence. In order to do so, we focus on live TV streams. This is because a live TV channel (that is multicast from the video source) is time synchronized across different STBs within a VHO or a region. There are other types of video streams that are independently fed to each STB (*e.g.*, VoD and DVR). However, we do not have enough information from the logs to determine the exact VoD program being displayed at any given time. Therefore, we only consider STBs that are active (as opposed to in an offline mode) and the ones that are tuned in to receive live TV (as opposed to VoD or DVR) in our analysis.

In order to keep up with the sheer volume of channel tuning events, we first use a simplification process by discretizing time into fixed-length bins. We define a channel-tune-in predicate $T_c(s,t)$ for each channel c such that

$$T_c(s,t) = 1 \text{ iff STB } s \text{ was on channel } c \text{ during time bin } t,$$

and a crash predicate $C(s,t)$ such that

$$C(s,t) = 1 \text{ iff STB } s \text{ crashed in time bin } t.$$

Since we need to exclude the STBs that are offline or in VoD or DVR mode, we define the sample universe as

$$U = \{(s,t) | \sum_{c \in \text{ Live Channels}} T_c(s,t) > 0\}.$$

We should note that, the independence of users choice of TV viewing requires us to add several parameters in the sample universe of STB; because home network devices are not in an always-on mode, we consider the intermittency of live TV viewing. Also because users choose from different video contents, we further categorize the sample universe by program channels.

To investigate whether there is any live TV channel that is more prone to STB crashes, we apply a well known statistic, mutual information [6], on $\{T_c\}$ and $\{C\}$. Mutual information measures the information that two variables share: it measures how much knowing one variable reduces the uncertainty about the other. When two

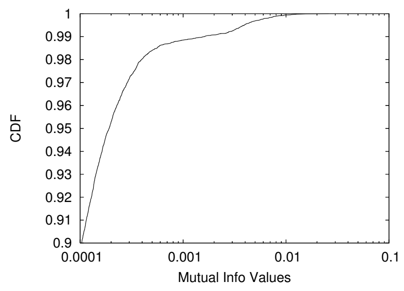


Figure 3: Cumulative distribution of mutual information values of $< channel, VHO >$ pairs

variables always match (or always opposite to each other), their mutual information is high; at the other extreme, if two variables are completely independent, their mutual information is zero.

In our context, we define the joint probability distribution:

$$p(T_c = i, C = j) = \frac{|\{(s,t)|T_c(s,t) = i \text{ and } C(s,t) = j\}|}{|U|}$$

for $i = 0, 1$ and $j = 0, 1$, and the marginal probability distributions:

$$p_{T_c}(T_c = i) = \frac{|\{(s,t)|T_c(s,t) = i\}|}{|U|} \text{ for } i = 0, 1$$

$$p_C(C = i) = \frac{|\{(s,t)|C(s,t) = i\}|}{|U|} \text{ for } i = 0, 1$$

We can now compute the mutual information of T_c and C, $I(T_c; C)$, as follows:

$$\sum_{i=0,1} \sum_{j=0,1} p(T_c = i, C = j) \log \left(\frac{p(T_c = i, C = j)}{p_{T_c}(T_c = i) p_C(C = j)} \right)$$

A high value of the mutual information can be an indication of either the channel being *crash-prone* or *crash-averse*. Since we are particularly interested in isolating the crash-prone channels, we apply a threshold to eliminate the cases with low total crash count ($|\{(s,t)|T_c(s,t) = 1 \text{ and } C(s,t) = 1\}|$). This analysis helps us to quickly focus on a few of the video sources and their VHO presence among an overwhelming number of channels and regions under exploration. We next "zoom in" to these cases and conduct a more fine-grained analysis for them.

4.2 Results

We analyze STB channel tuning logs and crash logs for all four million STBs over a one month period. We first correlate the STB channel tuning events with the STB crash events and identify all the viewing sessions that experience a STB crash. We then compute mutual information between the channel-tune-in and STB crashes for all possible pairs of channels and VHOs to identify possible crash-prone channels and their VHOs. Fig. 3 illustrates the cumulative distribution of the mutual information for hundreds of channels in tens of VHOs for one day. Among all the pairs of $< channel, VHO >$, there are a small number of pairs that have high mutual information values. Using mutual information as a guide to pick out the channels of interest, we rank the pairs by magnitude of mutual information and focus on the top few percentages of them. In our case, we take the top 2% of the pairs, corresponding to those that have mutual information values higher than 0.001 (the average is 0.0001).

Fig. 4 shows one such example with mutual information value of 0.0221. Fig. 4 (a) compares the STB crash ratio (aggregated over 15 minute time bin) of a popular national channel (denoted as channel x) versus that of all other channels in a VHO (denoted as VHO a) during one day period. The y axis is normalized by the peak crash ratios for STBs tuned to channel x and that for STBs tuned to other channels, respectively. We observe a huge spike at 2:00AM in the crash ratio for STBs that are tuned to channel x, while the crash ratio of STBs that are tuned to other channels in the same VHO has a slight increase. This spike indicates that there are a large number of simultaneous STB crashes for STBs tuned to channel x at 2:00AM.

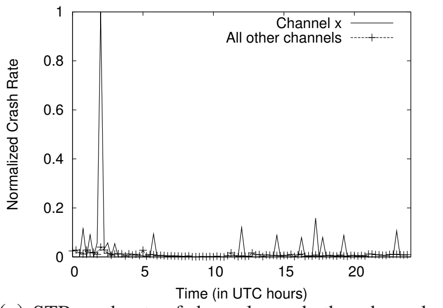
Fig. 4 (b) depicts the normalized counts of crashed STBs (solid curve with left y axis) and uncrashed STBs (dotted curve with right y axis) for all STBs that are tuned to channel x. Figure 4 (c) depicts the normalized count of crashed and uncrashed STBs tuned to other channels. The distribution of STB crash counts for STBs tuned to channel x is clearly very different from that for STBs tuned to other channels. In addition, the close alignment between spikes in crash count and peaks in uncrashed counts suggests that higher STB crash counts are likely to occur during peak watching hours. This observation holds for both STBs tuned to channel x and those tuned to other channels. It is also important to note that the absolute number of uncrashed STBs is orders of magnitude larger than that of crashed STB.

There are several observations suggesting that there is a good chance that the high crash ratio at 2:00AM is caused by a problem at the video stream source. First, we observe from Fig. 4 (a) that there is a small spike at 2:00 AM (*i.e.*, the same time we observe the spike in the crash ratio of STBs tuned to channel x) in the crash ratio of STBs tuned to channels other than channel x. Further analysis reveals that there are several channels with the same source of video content experiencing similar spikes in their STB crash ratios at 2:00AM. For example, Figure 5 (a) shows the crash ratio of STBs tuned to the high definition (HD) version of channel x. In addition to having the HD version, the same content is broadcasted to other metropolitan areas as they are on a nation wide channel. Figure 5 (b) shows the STB crash ratio for channel x at a different VHO (denoted as VHO b). Here, we observe the same spike in the STB crash ratio at 2:00AM. This further suggests that the correlated high STB crash ratio at 2:00AM is highly likely to be caused by video stream issues. Lastly, we add that the frequency of a large number of simultaneous STB crashes (i.e., spikes in STB crash ratio) is small. However, it is not negligible due to their broad impact on user experience of multiple channels across multiple VHOs. Figure 5 (c) shows a separate incidence for channel y and in VHO c, in which y is a local TV channel that is specific to VHO c. Again, we observe synchronized STB crashes that are localized to channel y.

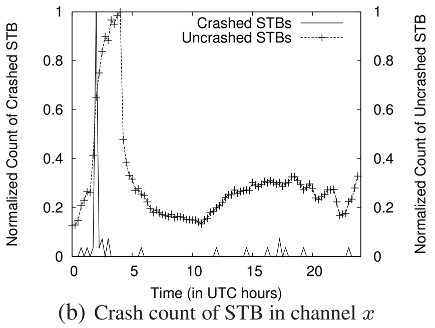
A limitation of our analysis is that at the time of this study, we did not have an extensive archive of video streams in all VHOs. Hence, we could not easily replay the video stream corresponding to the simultaneous STB crashes to reproduce such behaviors. However, in an independent study in a controlled testbed environment, it has been reported that by manipulating the video content stream feeding to the STB (through corrupting the bit-stream in a particular fashion), one can induce a STB crash deterministically. By confirming that this type of video corruption is also taking place in the operational IPTV network, we believe that video stream impairment is likely a contributing factor to STB crashes that users experience in the service.

5. DOES USER ACTIVITY MATTER?

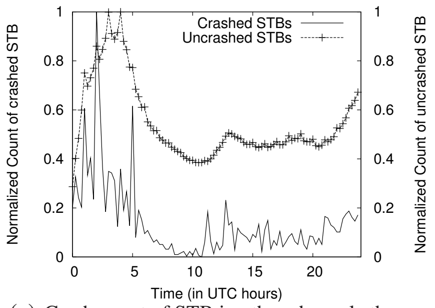
Having established substantial evidence indicating an association between video content and STB crashes, we now turn to investigating the impact of user activities on STB crashes. By user



(a) STB crash rate of channel x and other channels

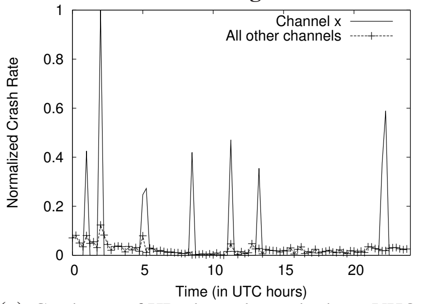


(b) Crash count of STB in channel x

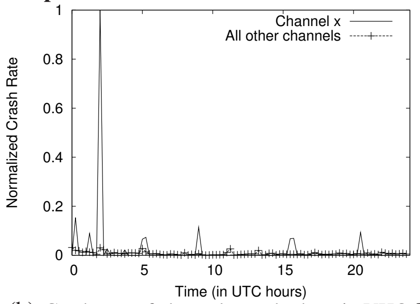


(c) Crash count of STB in other channels than x

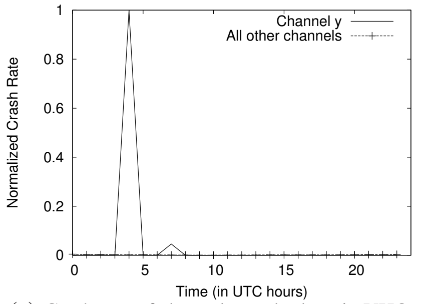
Figure 4: Crash rate comparison between channel x and all other channels in VHO a



(a) Crash rate of HD channel x and others, VHO a



(b) Crash rate of channel x and others in VHO b



(c) Crash rate of channel y and others in VHO c

Figure 5: Additional study on per-channel crash rate in different setups

activity, we are referring to the control actions that a user inputs by operating either the remote control or the STB directly as discussed in Section 2.2.

5.1 Methodology

This analysis is motivated by the type of performance deterioration problems frequently observed in various software systems, such as memory leaks. The intuition is that system resources can be depleted or system state can be corrupted over the accumulation of user activities, or simply over time. We study the relationship between STB crashes with the time duration and the total number control activities since the last reboot or turn on.

In contrast to the analysis in Section 4 in which time correlated STB crashes rates are examined, user activity within different STBs are expected to demonstrate less strict correlation across time, if such correlation exists at all. It is thus assumed to be sufficient to conduct our analysis on an each individual STB basis. Absolute timing across STBs becomes an irrelevant factor, and hence it is sensible to aggregate across STBs by ”time shifting” such that the alignment of the crash events boost hidden signals while suppressing unwanted noise.

We proceed by examining the STB crash rate as a function of (1) the accumulation of user control activities since the last reboot; and (2) the power cycle duration since the last STB reboot. We expect the STB crash rate to be invariant when it is independent of either of the factors.

We begin by defining a power cycle session for STBs. Once a STB is turned on (or recovers from a crash), the “power cycle” session can be terminated by two conditions – turning off the STB or a STB crash. We particularly name the last unintentionally terminated sessions as crashed sessions.

The STB crash rate can then be defined as the number of crashed sessions over the total number of sessions. For the metric (1), the STB crash rate after experiencing n user control actions is defined as

$$\frac{|\{\text{crashed session having control actions} = n\}|}{|\{\text{sessions with control actions} \geq n\}|} \tag{1}$$

and for (2), power cycle metric, the STB crash rate at time t is

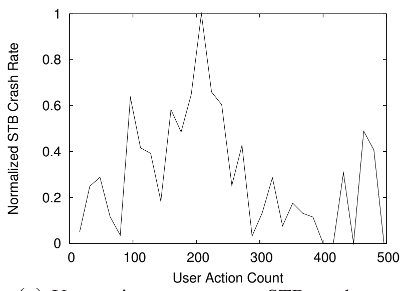
$$\frac{|\{\text{crashed session with length} = t \text{ seconds}\}|}{|\{\text{sessions with length} \geq t \text{ seconds}\}|} \tag{2}$$

We further smooth the functions by applying a bin size of d actions or d seconds to the independent variables.

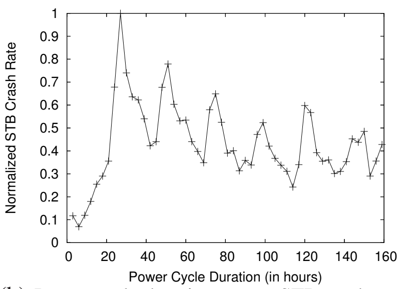
5.2 Results

We first present the results of correlating the accumulation of user control actions with STB crashes over a one month time interval. Figure 6 (a) illustrates the normalized rate of STB crashes as a function of user actions binned in intervals of 10 actions ($d = 10$). For the user action count ranging from 0 to 500 per power-cycle session, we depict the normalized STB crash rate (note that the normalized crash rate of 1 means that the crash rate is the highest among the observations, NOT that STBs always crash). From the trend of the plot, we observe that the STB crash rate does not show a consistent pattern as more user activity is accumulated. In a follow-up analysis where we tested the relationship between the frequency of user actions and the STB crashes, we again confirmed that increased user activities do not lead to higher rates of STB crashes.

Figure 6 (b) shows the trend in STB crash rates over the power cycle duration of STBs binned by every 10 minutes ($d = 10$). Although we use one month’s worth of data, we only plot up to 7 days of time because STB power cycle durations longer than 160 hours do not produce sufficient numbers of STB crashes to conduct a meaningful analysis. From the trend of the plot, we observe that the STBs with prolonged power cycle duration exhibit increased crash rates until the first 24 hours. For the power cycle durations exceeding 24 hours, the crash rate oscillates between two values following the diurnal pattern of the STB crashes. Overall, the trends imply that the crash rate for longer-lived STB “on” ses-



(a) User action count versus STB crash rate



(b) Power cycle duration versus STB crash rate

Figure 6: Comparison of user activity and STB Crash

sions is higher than that of short-lived STB "on" sessions. And the increase in crash rate over the longevity of STB "on" sessions implies that factors (*e.g.*, CPU heat up, non user-level software bug) exist that accumulate and crash the STBs when they are used for extended periods of time.

6. RELATED WORK

There has been an increasing interest in network troubleshooting using statistical analysis [1, 4, 5, 7–13, 17–20]. The goal is to identify the set of root-causes that can best explain a given symptom or set of symptoms. Commercial tools such as HP Openview [15], IBM Tivoli [21] focus on analyzing individual large events. Sherlock [1] uses conditional probabilities and a multi-level approach to look across multiple symptoms of a common type in the inference of dependencies. NICE [13] and Giza [11] focus on detecting and troubleshooting undesirable chronic network conditions using statistical correlations. Mercury [12] uses statistical change detection and correlation to identify the performance impact of network upgrades. URCA [17] uses unsupervised techniques to identify anomalous traffic flows and their root-causes. It uses feedback from the anomaly detector to eliminate flows that exhibit normal behavior. ASTUTE [18] is a recent network traffic anomaly detector that uses the equilibrium property and correlation across anomalous flows to discover a new class of anomalies.

Giza [11] is specific to IPTV, leveraging the hierarchical structure of IPTV networks to localize significant problems. In contrast, this current paper focuses on analyzing software crashes on Set-Top-Boxes using video content and user activities. This is something not explored by previous papers. Our schemes are suitable for handling diversity of video sources and analyzing temporal patterns of user activities identifying causal relationships to STB software crashes.

7. ON-GOING AND FUTURE WORK

In this paper, we run several analyses on STB crashes inside home networks. We start by characterizing the spatial and temporal properties of STB crashes. We then perform an in-depth analysis on the inputs to STBs: video streams and user inputs. For this, we employ various statistical methods most suited for isolating the oddities in the correlations and uncovering the potential causes of STB crashes. Our key findings include: (i) the correlation between STB crashes and video stream content suggests a strong possibility that impaired video streams may cause STB crashes; and (ii) athe ccumulation of frequent user actions does not appear to correlate with increased STB crash rates, whereas extensive usage over a long period of time seems to negatively impact the health of STBs.

One focus of our on-going work is to further analyze the cause of STB crashes, building on our current results. In cooperation with video experts, we are aiming to reveal the types of video impairments that may induce crashes, as well as reveal the reason(s) that prolonged usage may cause STBs to crash. Another aspect of our on-going work is to extend our analysis to other, more sophisticated measures of QoS and user experience beyond simple STB crashes. In an effort to capture subtleties in the users' perception of video quality, we are working on employing user complaint logs filed with customer care centers as a measure of trouble, and are correlating these with network events.

8. REFERENCES

[1] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *Sigcomm*, 2007.

[2] Broadcom Set-top box solutions. `http://www.broadcom.com/press/release.php?id=s407352&industry_id=4/`.

[3] Consumer electronic show - next big thing supersession, 2010. `http://ces.cnet.com/next-big-thing/`.

[4] I. Cohen, J. S. Chase, M. Goldszmidt, T. Kelly, and J. Symons. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In *OSDI*, 2004.

[5] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot. Netdiagnoser: troubleshooting network unreachabilities using end-to-end probes and routing data. In *CoNEXT*, 2007.

[6] S. Guiasu. *Information Theory with Applications*. McGraw-Hill, 1977.

[7] S. Kandula, R. Chandra, and D. Katabi. What's going on? learning communication rules in edge networks. In *Sigcomm*, 2008.

[8] S. Kandula, D. Katabi, and J.-P. Vasseur. Shrink: A tool for failure diagnosis in IP networks. In *MineNet*, 2005.

[9] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. IP fault localization via risk modeling. In *NSDI*, 2005.

[10] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. Detection and localization of network blackholes. In *Infocom*, 2007.

[11] A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao. Towards automated performance diagnosis in a large IPTV network. In *ACM SIGCOMM*, 2009.

[12] A. Mahimkar, H. H. Song, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and J. Emmons. Detecting the performance impact of upgrades in large operational networks. In *ACM SIGCOMM*, 2010.

[13] A. Mahimkar, J. Yates, Y. Zhang, A. Shaikh, J. Wang, Z. Ge, and C. T. Ee. Troubleshooting chronic conditions in large IP networks. In *ACM CoNEXT*, 2008.

[14] Microsoft Media room. `http://www.microsoft.com/media/en/us/media-entertainment-solutions/inte% rnet-protocol-tv-iptv.aspx`.

[15] HP Openview. `http://www.openview.hp.com`.

[16] I. Research. Global market analysis, 2008. `http://www.imsresearch.com`.

[17] F. Silveira and C. Diot. URCA: Pulling out anomalies by their root causes. In *IEEE INFOCOM*, 2010.

[18] F. Silveira, C. Diot, N. Taft, and R. Govindan. ASTUTE: Detecting a different class of traffic anomalies. In *ACM SIGCOMM*, 2010.

[19] M. Steinder and A. S. Sethi. A survey of fault localization techniques in computer networks. *Science of Computer Programming*, 2004.

[20] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar. Answering what-if deployment and configuration questions with WISE. In *SIGCOMM*, 2008.

[21] IBM Tivoli. `http://www-306.ibm.com/software/tivoli`.