Université pierre et marie curie

Intership Report

MASTER 2 STL SPÉCIALITÉ ALGORITHMIQUE ET PROGRAMMATION

Analyse de la diffusion multi-canal P2P-TV

Author:
Marwan Ghanem

Supervisor:
Fabien Tarissan
Olivier Fourmaux

September 2, 2015

Contents

Chapter 1

Introduction

Today the Internet is the main support of data and voice exchanges, becoming a massive video support offering both live TV and video on demand services. These services were previously confined mainly to video broadcasting infrastructures such as satellite or fiber coax hybrids. Video quality particularly TV streaming has been rapidly improving, becoming High Definition[1] and soon Ultra High Definition, both using enormous amount of communication network resources. Such high requirement demands the development of dedicated technologies which till now are truly local and limited to residential operators (IPTV[2]) due to the high complexity and expenses of global services.

The alternative to these limited or expensive technologies is the Internet, particularly P2P. This technology depends on virtual networks (overlays) to connect its users. The dynamic topology of these overlays depends on many parameters: location of resources, network status, internal mechanisms of peers, the distributed content and the behavior of the user directly involved as consumers of content.

When it came to live TV streams, additional constraints were present which required P2P technology to adapt precisely real-time aspects. Such adaptation produced a new class of applications that realize such task while respecting these extra constraints: P2PTV. The content of the applications consists of audio/video streams, with the goal of real time distribution to a large number of receivers. The large number of streams and their intrinsic real-time characteristics generate timing constraints difficult to guarantee in the considered dynamic environment. Strict compliance with these constraints impact directly on the user's quality of experience and thus on his behavior, which in turn will impact the overlay.

P2PTV applications offer hundreds of channels, each carrying a live audio/video content to thousands of users. Each channel corresponds to an overlay integrating users wishing to receive its content, and these users can change

[1]video image with considerably more than 576 horizontal lines

[2]system through which television services are delivered using the Internet protocol suite such as VOD

channels at any time (usually depending on the contents) adding an extra dynamic factor as they become integrated in a different overlay. These users are called *multi-channel users.*

Such dynamic aspect adds large strain on the network. With the ultimate goal of understanding such dynamism, we attempt to track the presence of these *multi-channel users.* This class of applications, unlike most common P2P applications which use GNU GPL licenses[3], are completely proprietary in other words a black box. Thus the limitation in the possible analysis techniques. Therefore, non-invasive measurement techniques were used, giving only a partial view of the network. To compensate for such handicap, datasets that offer different vantage of points are used.

I present to you the work which was carried out in the past 6 month. Firstly I was able to detect the *multi-channel users* in spite of the disadvantages we faced. Secondly, by creating my own formalism, I was able to detect and highlight different *user behaviors.* Such detections would lead the way to more in-depth investigation on the subject

[3]Licenses that offer the freedom to run, study, share (copy), and modify the software

Chapter 2

State of the art

2.1 P2PTV

P2PTV is the diffusion of live video content using Peer-to-peer (P2P) networks particularly TV channels. These systems appeared after P2P systems have proven to be an excellent way to diffuse files. With the increase offer of video services online as well as the increase in video quality, P2PTV has proven to be able to withstand such demand. This success is due to the fact that such systems do not rely on a dedicated delivery infrastructure but instead on the upload capacity of theirs users whom are known as peers, as they act as client and server at the same time. Hence the low cost rapid deployment since server loads are dramatically reduced or even removed in certain cases.

2.2 Mesh-Pull Structure

Most P2PTV were designed initially to act completely as autonomous systems, relying solely on their peers, which means the higher the demand the better the offer. But as such applications became more commercial, they started using hybrid P2P infrastructures with servers to guarantee a better quality of video to their users. These servers are identified as *super-peers.*

A typical mesh-pull architecture is shown in Figure 1. This structure is deployed for each channel diffused by such applications. A video is divided into media chunks and made available by a super-peer. Afterwards these chunks are disseminated into the network via a certain number of peers who in return re-diffuse the video to other peers using their own upload capacity. For example in Figure 1 the super-peer diffuses the video directly to peers 1, 2 and 3 who would pass along the video to others.

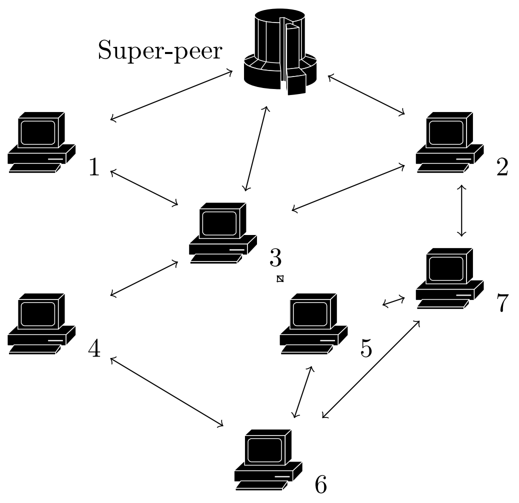


Figure 1: Mesh-based architecture

2.3 Protocol

P2PTV softwares typically have two major communication protocols: Firstly a peer registration, channel and peer discovery protocol and secondly a P2P media chunk distribution protocol. Figure 2 presents the channel and peer discovery protocol. When a peer first connects to the application, the peer downloads the list of channels that are diffused. Once a channel is chosen, the peer downloads an initial list of peers who are watching the same channel to communicate with them (Step 1). Afterwards the peer communicates directly with the other peers to acquire more peers, increasing gradually the list of peers. The list can also increase in size by directly getting contacted from new peers. However, in most cases the list tends to become constant quite fast [7]. It was also discovered that certain applications deploy a memory based list to know which peers are better to communicate with, based on previous experience [4].

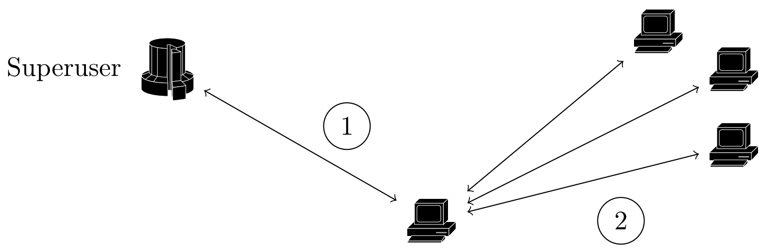


Figure 2: Channel and peer discovery protocol

A description of the video exchange protocol is as follows. A peer will ex-

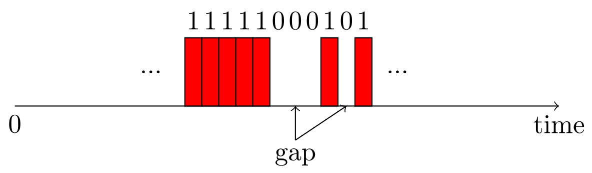


Figure 3: Buffer map

change randomly with other peers buffer maps which represent the video chunks. The buffer map is a sequence of bits, each bit is equal to either 1 or 0, where 1 represents a video chunk that the peer has while 0 represents a missing chunk. Peers will exchange buffer maps randomly during the diffusion, and then will exchange the video chunks depending on the information they acquired from the buffer map exchange. For example if a peer is missing the fifth video chunk and receives a buffer map with the fifth video chunk labeled by 1, the peer will request from the owner of the buffer map that specific chunk of video. Figure 3 presents a graphical representation of the buffer map with chunks of video.

From this we can determine that the traffic generated from such applications can be shared out of two categories. The first is the control (or signaling) traffic which could be either a heart beat signal[1], a peer's list exchange or buffer maps as previously explained. The second kind of traffic is the data (or video) which is the actual video stream in the form of chunks of video.

2.4 Related work

In this section, I will present the literature starting with the contemporary techniques as well the results that have been found to this day. There are two main types of analysis: firstly active analysis where the analysis is based on live information obtained by observing the network locally or globally with the goal of gaining a complete image of the network. Secondly passive analysis where the main analysis is done on network traces that has been collected.

S. Spoto et al. presented an investigation of PPLive using both active and passive measurements [13]. Using a crawler, they were able to by pass the absence of internal access. A crawler is an application that executes queries with the aim to gather a tremendous amount information. Therefore, this requires a reverse engineering process to get access to the protocols used by the application, making it hard to have a generic crawler. They deployed the crawler on 5 PCs each running 1 master and 200 slaves, allowing them to have an accurate snapshots of the network. Such process was able to classify the traffic into three classes as well as proving that only 15% of peers could be considered as active peers. However such process seems to be highly demanding due to the

[1]a heartbeat is a periodic signal generated by hardware or software to indicate normal operation or to synchronize other parts of a system

large number of running slaves, and also the fact that a reverse engineering is required to be able to get the necessary queries.

Some other work has been done on a more quantitative perspective: Hei et al. proposed for instance a large scale measurement study of P2PTV, using also a PPLive dedicated crawler [7]. By running the crawler through different scenarios and collecting a huge amount of data, they showed that P2P IPTV users have the same behavior as that of TV users. They also demonstrated the existence of a small set of super-peers that highly contribute to the video uploading.

Another use of crawlers was done by J. Jia et al. They tried to characterize PPStream [8]. They were able to find certain characteristics such as geographical clustering, arrival/departure patterns and playback quality. Their findings thus help in generating models of such streaming systems.

Unlike the previous studies which mostly involved active studies, D. Rossi et al. proposed a complete framework for comparing P2P applications, analyzing the most used P2PTV applications such as PPLive but also well known file-sharing applications such as BitTorrent [6].They started by defining a set of observable features related to the protocol used by these applications, highlighting the main similarities and differences between each P2P application. In doing so they provided the key elements that open the way to passive analyses to such applications.

T. Silverston et al. passively studied the traffic in P2PTV infrastructures and were able to compare different applications pointing out their similarities and differences [12]. They offer a good base to characterize traffic generated by such applications and built realistic simulation and emulation of such systems. They looked deeply into the traffic, where they discovered that signaling traffic tend to have a large inter-packet time while video traffic have a smaller one. Finally they looked into peer behavior, showing that the vast majority of peers tend to receive data more than they send.

T. Silverston et al. also did a comparison study on four P2P IPTV applications by analyzing the different traffic patterns collected from the four applications during a massive global event. They show that each application has its own download policy, as well as different churn of peers[2] and session durations. [11]

More specific studies exist particularly in studying the user behavior and multi-channel observation: M.Wang et al. analyzed traffic that was characteristic to users switching from one channel to another [15]. Using the most popular P2PTV applications such as PPLive or SOPCast, they monitored a channel for a given period of time and then suddenly changed to another one. They revealed that switching has a huge impact on the network efficiency as it increases the overload and adds a significant overhead. Finally, K. Mitzutani et. al. were able to detect video servers as well as to find new characteristics of PPTV by monitoring multi-channel PPTV traffic [10].

A different type of study was also done by G. Tian et al. to show how P2P

[2]A churn of peers is the effect created from the arrival and departure by thousands or millions of peers independently

streaming has great potentials, using a *cooperative game theory* framework to analyze P2P dynamic streaming [14]. With such insight, they proposed a P2P dynamic streaming system showing its efficiency through a set of simulations.

E. Alessandria et al. were also able to show how such applications are efficient [4]. They ran the applications under adverse network conditions such as lack of bandwidth or packet loss probability. All the applications they tested managed efficiently to react to such conditions; however, in certain situations they became aggressive and potentially harmful to the network.

N. Magharei et al. proposed a study on the structure of networks that most P2PTV applications use [9].They examined key design issues and trade-offs of such structures as well how bottlenecks can appear in such systems through simulations.

Finally Y. Goh compared the different overlay structures in P2P media streaming systems [16]. They have shown that tree-based models have a more stable video delivery quality than mesh-based models. Mesh-based models are more resilient to dynamic peer churn which is the used in P2PTV applications.

Chapter 3

Analysis

In this chapter, my methods of analysis as well as the results will be presented.One reason behind presenting both methods and results is that there was no previous knowledge of the structure or content of the dataset. Most of the analysis was developed while advancing in research and depended highly on what was previously found. There was also some trial and error analysis, thus I will present first the main dataset that was used in the analysis.

3.1 DataSet

Access was given to a dataset that was extracted from a traffic that was measured on December 2013 on PPTV, and was 14 hours long. The key aspect of this dataset is the fact that it measures simultaneously 12 different vantage points. During the measurement 12 different PCs were each running the application on a different channel (the most populars PPTV channels at that moment), hence the 12 vantage points. Every PC had an Internet connection provided by FLET'S HIKARI NEXT, 100 Mbps optical access service via Plata HIKARI Mate as an ISP in Japan. In addition, each PC had Wireshark [3], a well-known packet sniffer running. Therefore, I was given the integrity of the traffic that has been sent and received by the 12 machines. One of the first challenges was to keep the data that is relevant to our goal. Using TCPDUMP [1] and TShark [2], both well-known packet analyzers, I kept only the information that is needed for the analysis. For each packet exchanged I kept the timestamp, source, destination and size, while removing any other information such as transport control and unrelated packets since as previously mentioned I had the totality of the traffic. The final product was simple text files that I later treated using the applications that were developed in the course of the internship.

Figure 4 shows a graphical representation of a line from the files that I generated and used for my analysis.

$$\underbrace{\text{2013-12-18 05:06:15.153450}}_{\text{Time Stamp}}\ \underbrace{\text{221.184.82.26.5829}}_{\text{IP Src}}\ \underbrace{\text{192.168.12.40.5041}}_{\text{IP Dst}}\ \underbrace{\text{UDP}}_{\text{Protocol}}$$

$$\underbrace{\text{38}}_{\text{Size}}$$

Figure 4: One line from text file

Once I had the required data, I started my first basic analysis. Table 1 presents the global properties of the dataset. It shows how a larger amount of data (193.3 GB in total) was exchanged during the measurement. This is due to the presence of 12 different PCs which leads to a vast amount of data in such a short period time while contacting 100 80 peers in total. One other theory might be related to the location where the trace was measured. The trace is measured from Japan which is geographically close to China where such applications are most popular. Therefore, making our machines good peer candidates to a larger pool of peers.

Property	**Value**
Duration	14 hours
Data Size	45 GB
Number of channels	12
Number of peers	100 809
Maximum number of peers per channel	21 518
Average payload [1] size	504 29 B
Total payload size	193.3 GB

Table 1: Properties of the dataset

[1]Payload is the size or weight of a packet

3.2 Global Analysis

In this section I present my global first analysis where I exploited all the information in the collected data in order to start the detection of multi-channel users as well as getting an image of the dynamics of the P2PTV infrastructure. I will quickly present general statistics on the dataset and then focus on the topic of this internship.

3.2.1 Evolution of users and packet size over time

By using the timestamps as well as the IP addresses presented in dataset either as sender or receiver and also the size of packets exchange during the whole measure, I was able to get an idea of the dynamism of the network. Figure 5 presents the evolution of the number of peers detected as well as the sum of the size of payload exchanged. We can easily see how both quantities are similar, and how the amount of data increases when the number of peers increases. One might also notice the peak between 1 pm and 3 pm which can be explained by the fact that this is evening time in Asia a time when people watch TV the most [5]. Another reason for such peak might be related to the broadcasting of certain popular TV programs.

Another interesting information that I have noticed is the amount of data exchanged was uniformly distributed over the 12 channels that was followed. On the other hand, the number of peers was not uniformly distributed; 4 channels of the 12 contributed to more than 60% of the total number of peers.

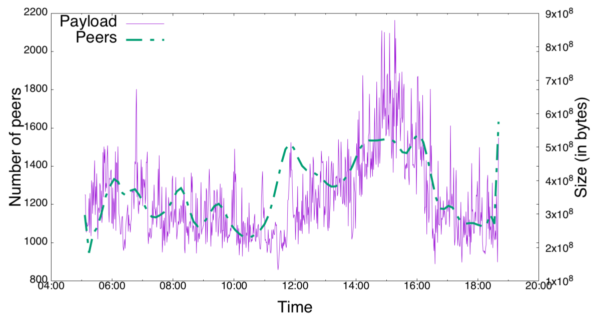


Figure 5: Evolution of number of peers and payload size over time

3.2.2 Tracking exchanges of video content

As I mentioned in Section 2.3, the traffic generated by P2P-TV applications can be categorized into signaling and video traffic. As the goal of the internship was to detect the peers that change channels and therefore, are watching these channels. We can therefore, base our work on the fact that if a peer watches a channel, there must be an exchange of video traffic. Consequently it is vital to distinguish between both types of traffic.

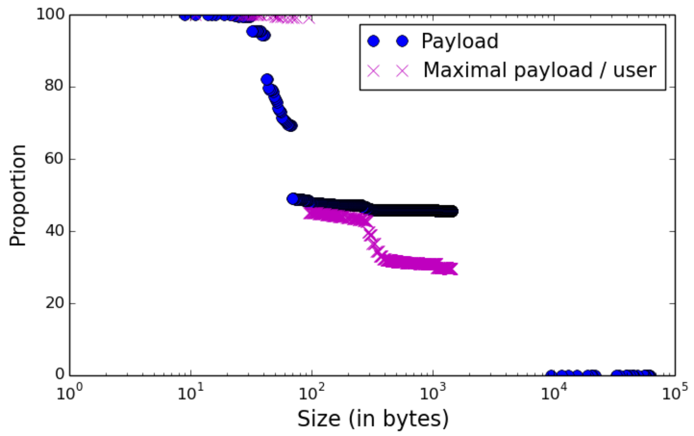


Figure 6: Inverse CDF of payload size and maximal payload size/user

Gathering the sizes of all the packets exchanged and counting the the frequency of each packet size, I was able to get a clear image of the traffic. Figure 6 shows the inverse cumulative distribution of the payload size(plain circles) in log scale. Two regions can be easily spotted: A first group of packets around 49% involving sizes in the range [20 : 90], a second group around 45% involving sizes higher than 1000 bytes. Clearly the first packet is signaling traffic while the second one can be categorized as video exchange, which is coherent with what we can find in the current literature.

From this I was able to conclude that any exchange between 2 peers with a packet of size less than 1000 bytes would not be video traffic. Secondly, I made an assumption that a peer involved in at least one packet of video traffic, will be referred to as *active user*. In other words, an *active user* is a peer who is likely to be watching a channel and therefore, is an interesting candidate for our main goal in this internship.

From Figure 6 we can notice that the number of control packets and video packets is well balanced. However, this can not be applicable when considering the actual quantity due to the huge difference of size between control and video packets, as this doesn't show a lot about the peer behavior. Figure 6 shows the maximal payload size a user has sent or received (cross dots). Using the assumption I made in the previous paragraph, I can start categorizing the peers. This plot reveals that only 30.93% of the users can be considered as an *active user*, which is in accordance with the standard results found in the literature (see [13] for instance).

3.2.3 Presence multi-channel users

Now I started analyzing the presence of multi-channel users. Firstly, I calculated the proportion of users that I can identify in several channels during the whole duration of the measurement. It is necessary to recall that one of the main challenges of this internship or analysis is the fact that we are using partial and independent measurements of 12 channels. So a peer can be present on two channels with one of them only being in the 12 channels or simply be present in 2 channels and not communicating with our machines. Thus, at this moment of the internship I had no guarantee to detect such behavior or at least detect it completely.

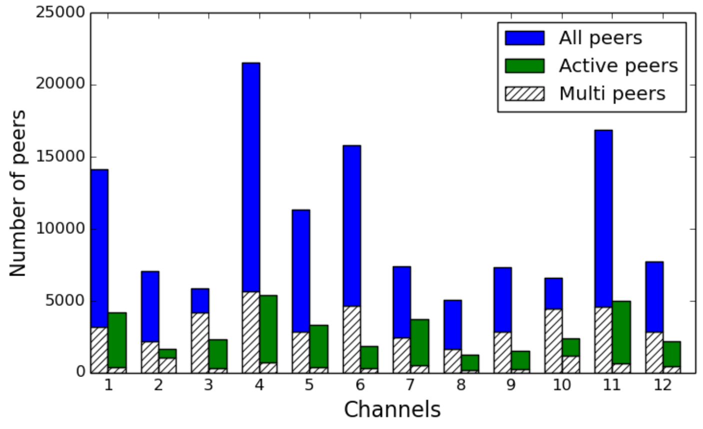


Figure 7: Distribution of the peers over the channels.

Figure 7 introduces each channel more in detail. For each channel, I show the number of peers detected (left bar in blue) and the number of *active users* (right bar in green) as defined in the previous section 3.2.2. Finally, for both bars I show all the fraction of users that who were also detected on at least one other channel (bottom part of the bars with hatched lines).

As I previously noted, four channels had the majority of the peers. This is likely due to the difference of popularity of channels as well as the programs diffused during the measurement. Figure 7 is actually the first confirmation that this different approach of having only a partial point of view is actually up to the task, as it shows that it is possible to detect users that appear on several channels. It turns out that 15.83% of the users are *multi-channel users*. While if we only focus on users who exchange video packets, this ratio drops to 2.83% of the total number of users. Nevertheless, this is still 9.13% of the active users, thus revealing that there is a non negligible fraction of users who exchange video content and are involved in several channels.

3.2.4 Exploiting sliding time windows

The results that I presented till now are interesting as they highlight that through a partial point of view, I was able to detect the different type of traffic, but most importantly the presence of *multi-channel users*. However, the aggregation of information prevents from further refinements. The main problem is that it removes all possibilities to distinguish a user who switches between different channels (referred further as *switching user*) from a user who completely stops watching and disappears from the network and comes back to watch a different channel later in the measurement. Both types would be considered as *multi-channel users* when the information is aggregated.

This is why I propose in this section another approach by relying on a view provided by a sliding time window. I basically sliced the whole dataset into non-overlapping windows of similar temporal size and studied each separately. As expected, the size of the window was the first problem to tackle, as it is the key parameter in this approach. Since the main focus of this internship is to track the presence of *switching users*, the size had to be short enough to discard users that disconnect and came back later, but it has to be long enough to detect the presence of the users in several channels as they change channels. Therefore, I decided to use a 1 minute size window.

The non-overlapping window was simply chosen over a overlapping window due to the high density of the dataset making the analysis time consuming and simply unfeasible. We will see further that the accuracy I get from the non-overlapping window is already high enough and up to the required task.

3.2.5 Different peer behavior

The first task was to look into *multi-channel users*, and to be able to properly differentiate them. By depending on the the 1 minute windows, I can detect users present *simultaneously* on different channels during a window. Technically wise for each peer I count the number of channels the peer is involved in during the one minute. From such information I was able to calculate the average of channels the peer is involved in and plot Figure 8.

Figure 8 presents the inverse cumulative distribution of the average number of channels on which a peer is simultaneously present; for all the peers (plain circles) and for active peers (cross dots). In both cases, there is a large amount of peers involved in 1 channel only. Firstly, the value decreases smoothly between 1 and 2. Then one can clearly observe a breach around the value of 2 and a large gap between 2.5 and higher values.

First notable observation is that some peers averaged at least 4 channel in such a short window, which in theory should not be possible due to application restraints. This is an indication of an unusual behavior and it is reasonable to consider such peers as *super-peers*. This was validated by the fact that all these peers were present at least 90% of the total duration of the measurement. To back this, I also looked into the intervals between two consecutive traffic on the same channel, the maximum was 37 seconds. Unfortunately, there is no

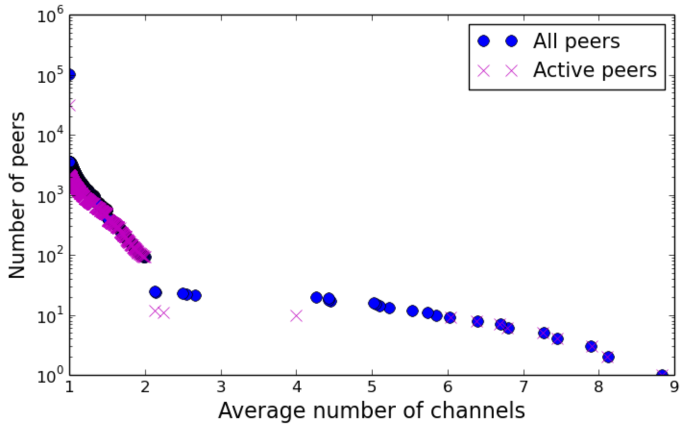


Figure 8: Inverse CDF of the average multi-channel presence.

formal method to prove such information, but it is highly unlikely that a peer would show such behavior; that is being present for such a long duration while being highly active on a large number of channels. Therefore, as mentioned in section 2.1 there are two types of users in the network, which I can manage to distinguish using this method.

3.2.6 Different super-peers behavior

The detection of several super-peers raised a question for me about their role in the infrastructure, which I decided to investigate even through it was not one of the goals of this internship; particularly if they participate actively in the diffusion of video content or simply are present for regulation such as offering the list of peers or list of channels to newly connected users.

In order to answer my curiosity, I used again the average number of channels as well as the maximal size of payload for each user. Figure 9 shows this, where each dot stands for a different user. One can clearly still distinguish the super-peers detected previously. I can now even get a clear image of their roles, where we can see certain peers support only control traffic (Peers under the green line at the 1000 mark, video traffic limit). While other peers who seem to have the highest average of channels, do exchange video packets in the server which can be proven from the fact they have high maximum size of payload. These peers are likely to be the main servers that cut the video stream into chunks, that are afterwards injected into the network.

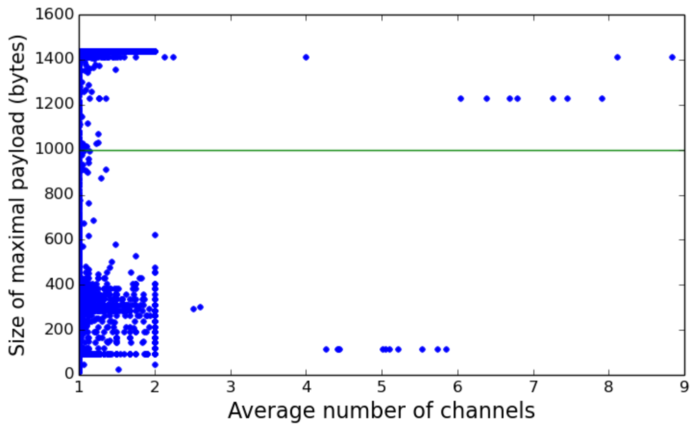


Figure 9: Average multi-channel presence *v.s.* maximal payload size

3.3 Minor Analysis

At this moment of the internship, I was able to detect superusers as well as their roles in the network. In addition I was able to differentiate between the two types of traffic, which allowed me to have a first classifications for the peers in the network. I ll present certain analysis that were done to either get a better understanding of the network or to validate some of our analysis.

3.3.1 Impact of the size of the window

As previously mentioned the key parameter of my analysis was the size of the time window. 1 minute was thought to be an ideal choice, but I still wondered what sort of effect would we have in case I had used a different value.

Figure 10 presents the same results in Figure 8 but using different window sizes (5, 10, 30 and 60 minutes). The quantitative results change of course but the main observation does not change, where all the curves dramatically decrease around the value 2. The increase of values after 2 is due to the fact that large window size manages to regroup channels for multi-peers that change channels. We can also notice that certain peers shifted to 12, thus confirming that certain peers have a different behavior and should be considered as super-peers.

3.3.2 NAT router detection

Another question that I have been posing till now, was what if multi-channel peers in particular super-users were simply nothing more than users behind a NAT router. In other words two users on the application who happen to be in the same household or building and are connected to the same router, therefore sharing the same IP address.

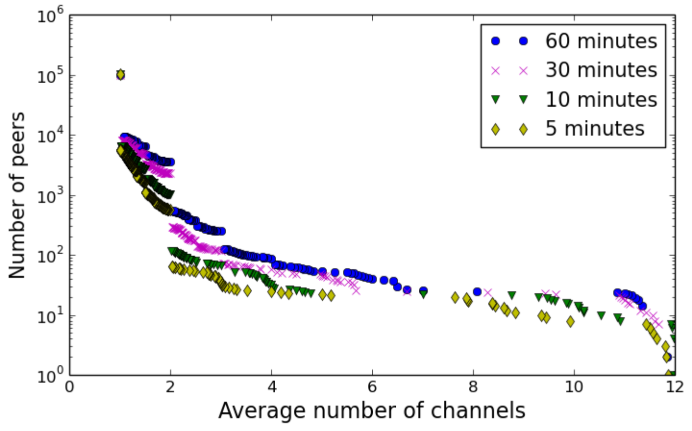


Figure 10: Inverse CDF of the average multi-channel presence for different time windows.

I picked a number of peers that seem to have an activity that can resemble more than one peer behind a router such as active on 2 channels or highly active for a long period. For each peer I got the IP identification number of each packet the peer has sent to me. IP identification is value coded using 16 bits. This value is incremented each time a machine sends out a packet, thus theoretically speaking each packet id our machine receives from a certain peer should be higher than the previous id packet (till the counter is reset at $65,535$). My way of testing was simply to see for the selected peers if the previously mentioned rule was followed. All peers that were tested seemed to be normal users depending on my method. The problem of such method is there is no way to completely be sure. Firstly, due to the partial image we only have. Secondly, certain routers tend to change the identification value. Therefore, I have no way to prove completely that some of our peers happen to be more than 1 active user sharing the same IP address.

Chapter 4

Formalization

At this moment of the internship, I was able to detect the multi-channel users particularly the active ones. To be able to start looking into behaviors over periods of time rather than single packets , I had to start a formalization that would allow me to represent activity over a duration of time.

4.1 Sessions

I started by defining what I call a *session*, which would represent a span of time for a certain user on a certain channel. I should also point out that, from now on I analyze only video traffic as I am interested in active users who truly zap, in particular the download traffic to be truly sure that a peer is present. Since if I take into consideration the upload traffic as well, I can end up with a false impression that the user is still present thus generating sessions longer than they really are; as we have no real proof the user is present when we send a packet to him.

Definition 4.1.1. *Session* is a period where I consider a peer is active and watching a channel. A session is live as long as the peer sends at least one packet every Δ. In other words, a session terminates if a packet is not received after Δ duration or received after a duration longer than Δ. (P, C) represents the couple of active peer and the channel being watched.

Figure 11 shows a graphical representation of a session. The Horizontal black line represents the time lapse of the channel and peer communication, while each vertical blue line represents a packet received from the peer. In red dashed line we can see what I present as the duration of session labeled *Dur* for short.

As previous, the temporal aspect is an important key factor and it had to be picked wisely. The most important criterion in my choice was not to have sessions for a peer on the same channel close to each other. In other terms not having plenty of tiny sessions close to each other that would have been one

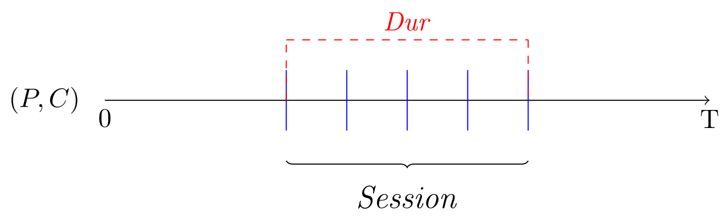


Figure 11: *Session*

large session if the value of Δ was just a bit larger. So for a set of Δ values between 1 and 15 minutes, I calculated all the sessions. Afterwards I calculated the difference in time between every consecutive session having the same peer and channel (P, C). Finally using the differences, I calculated the percentage of differences that are under $2 * \Delta$ and $1.5 * \Delta$.

Figure 12 shows for each of the 15 Δ values measured the percentage of differences smaller than $\Delta * 1.5$ (purple bottom curve) and $\Delta * 2$(green top curve). In other words the percentage of sessions that are close to each other. We can see that both curves increase at first than decreases around the value of 3 then starts to increase again dramatically. The only difference between both curves is that around 5 $\Delta * 2$ curve decreases-slightly again. Combining both curves, I decided that the optimal Δ is 3 minutes.

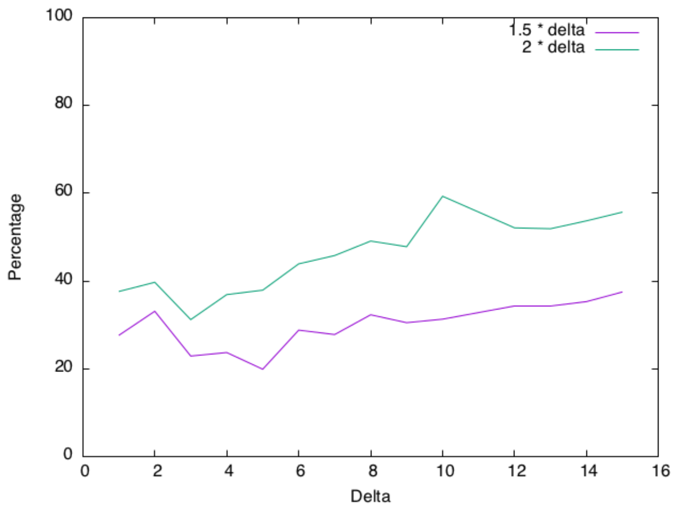


Figure 12: Percentage of difference between sessions less than Delta

4.1.1 Quantitative Analysis

Once the correct Δ was defined, I started by doing certain quantitative analysis. I calculated the number of sessions for each peer. Figure 13 shows the inverse cumulative distribution of number of sessions with log scale on the y axis. We can still notice the *super-peers*, who all have a really high number of sessions (larger than 700). While normal peers do not seem to ever have more than 100 sessions.

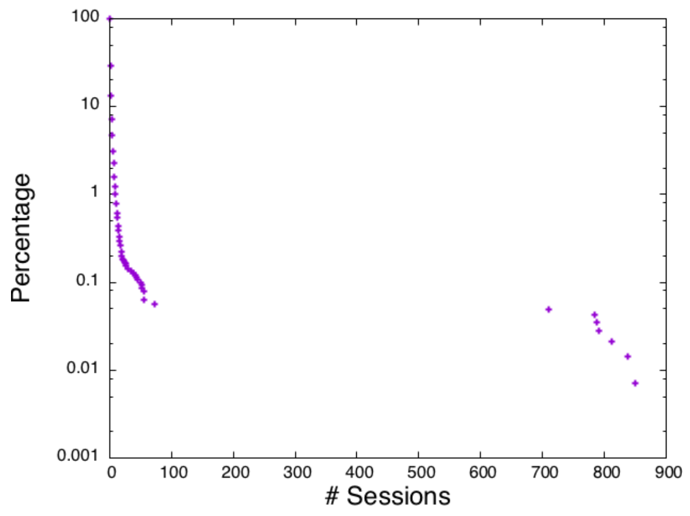


Figure 13: Inverse CDF of number of sessions

4.1.2 Refinement

After the quantitative analysis, I started to filter my data even more to have a fine group of candidates. The first filter was removing all peers that watch only 1 channel as well as the *super-peers*, using the knowledge I previously acquired. This filter reduced my data drastically from 14089 peers to 766 peers. The second filter was eliminating sessions that did not contain enough payload which would correspond to an suitable size of buffer. For this filter the only possible value that could be taken in consideration was the number of packets exchanged during the session. Since I am considering only the video packets, I was able to do the assumption that all packets have a payload of 1000. Secondly I calculated the minimum payload that needs to be exchanged, and due to the lack of information related to the dataset I made the assumption that all channels were diffused with a quality of 240p (Resolution of 426×240). In other words, an upload or download speed of 300 kbs is at least required to actually receive an accepted image; 300 kb represent 1 second of video . Considering the 300 kbs and 1000 byte packets, I concluded that 40 packets in a session would correspond to 1 second of video. Thus I kept only sessions that exchanged at least 40 packets. This decreased the number of users dramatically again, keeping only 233 users.

4.2 Zapping Sessions

After filtering my data and having the periods that were interesting, I had to define a *zapping sessions* which would group more than one session and represent a users behavior over a span of time.

Definition 4.2.1. *Zapping period* is a collection of sessions on at least 2 different channels, with a maximum difference between each session less than or equal to $2 \times \Delta$.

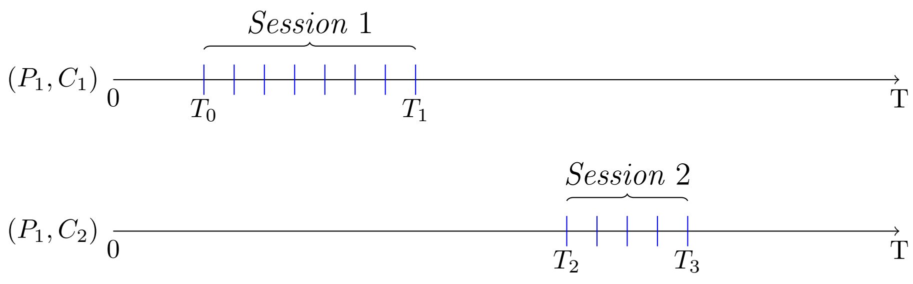


Figure 14: *Zapping Session*

Figure 14 represents a zapping session that has 2 channels. Each horizontal black line represents the time lapse of a channel and peer communication, with the peer being the same on both channels. We can see the sessions on each channel as previously defined in section 4.1. The first session ends at T_1 and the second starts at T_2, thus for this be considered as *zapping session* the equation $T_2 - T_1 <= 2 \times \Delta$ has to be true. I consider the true duration of the period (Dur_{true} for short) to be equal to $T_3 - T_0$, while the actual duration (Dur_{real} for short) to be the sum of duration of the sessions.

Using such formalism, I regrouped the sessions previously detected and filtered. I calculated for each *zapping period* the number of changes, the number of channels that was watched, the number of sessions, the true duration, the actual duration and the longest duration spent on one of the channel as well as the number of switch that is being the number of times the peer switches channels.

The first remark was the presence of periods in which the sessions overlap and for a period of time the peer is truly present on both channels, which are named *overlapping periods.* While other periods had no over lapping sessions thus the peer was never really present on both channels at the same time, which are named *non-over lapping periods.* The difference between both could be easily detected using the equation $Dur_{true} - Dur_{real}$. If the result is smaller than zero then sessions do overlap if not then sessions do not overlap. This helps me to do the first categorization and analyze each separately.

Figure 15 shows an over lapping period on the left and on the right a non-over lapping period.

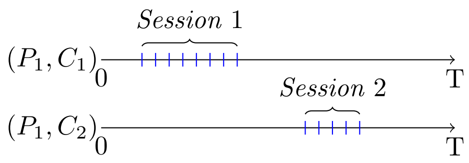


Figure 15: Type of Zapping Sessions

4.2.1 Non-over lapping periods

In this section, I will present the analysis that was done on *non-over lapping periods*. Firstly, I looked at the number of channels that was watched in each period, Figure 16 shows a bar chart representing for each number of channel the number of periods. We can see how over 90% of the periods contain only 2 channels, while the rest contain 3 channels. I use this fact to my advantage and I analyses each sub-class separately.

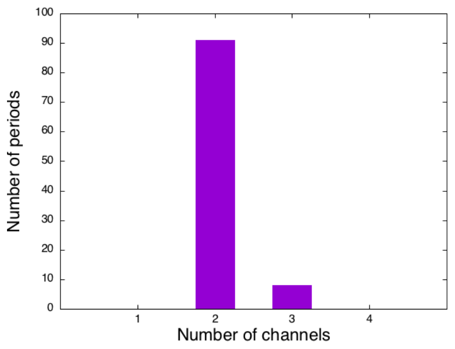


Figure 16: Number of periods for each number of channels

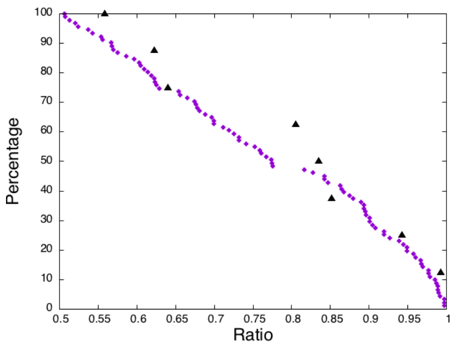


Figure 17: Inverse CDF longest duration/ real duration

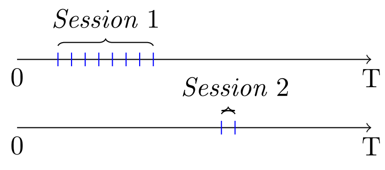
Afterwards I checked the percentage of time spent on each channel, using the real duration Dur_{real} and the longest duration spent on one channel of the two channels, then calculated the ratio between both values (longest duration / real duration).

Figure 17 shows in purple cross the inverse cumulative distribution for periods with 2 channel, while in black triangles for 3 channels. Concentrating on the 2 channel curve, the first remark we can notice is that certain peers were almost purely active on one of the 2 channels. Therefore, I can detect the first *user behavior*, that is just under 50% of peers watch over 80% of the time one of the two channels. Behavior wise it is likely they followed a certain program and after wards simply changed channels or the complementary of such behavior, changing and then finding a certain programming and following it. While A second *user behavior* would be the group of users, who happen to watch a program then switch to a second program since the amount spent on both channels is almost equal.

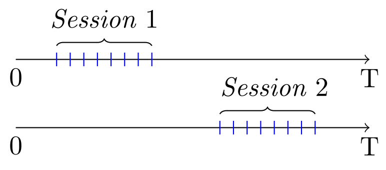
While if we look at the case of 3 channels, all peers spend at least 50% of their time on only one of the three channels which can explain a third *user behavior* of either looking for a certain program through 3 channels then staying on the last or being on a program and then after wards changing channels twice. Finally I looked deeply into these periods using the number of sessions in each case as well as the number of switches. A normal period where a peer changes channel every time would normally have 3 sessions each one on a different channel. However, except for one peer most users had at least 4 sessions. By comparing the number of switches and the number of sessions I was able to identify the pattern of each peer.

When the number of channels is less than the number of sessions, yet the number of changes is one less than the number of sessions, the peer did return to a channel he previously watched at some point during the period; since the peer changed channels every time yet the number of sessions is higher than number of channels. Thus, defining the forth *user behavior* which is changing between channels and returning to a previous channel.

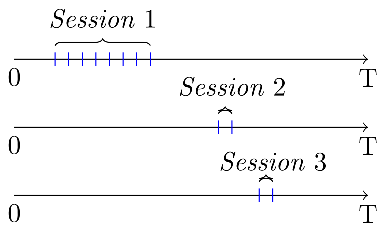
Figure 18 represents the 4 current user behaviors which were detected, using the same representation previously used to represent a *zapping period.*



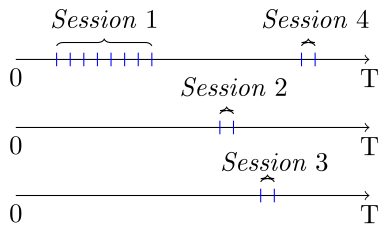
(a) *user behavior* 1



(b) *user behavior* 2



(c) *user behavior* 3



(d) *user behavior* 4

Figure 18: User behaviors

4.2.2 Over lapping period

In this section, I started analyzing the *over lapping period*. I firstly looked into the number of channels watched per period. Figure 19 shows a bar chart representing for each number of channel the number of periods. We can see that like *non over lapping periods* over 90%watch only 2 channels, however certain periods contain up to 4 channels.

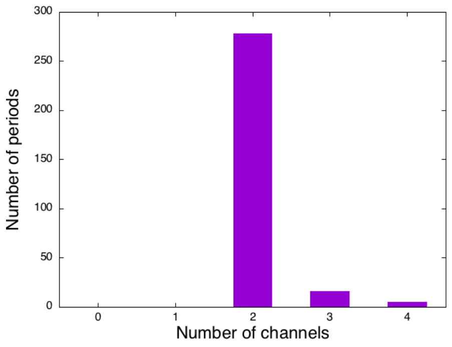


Figure 19: Number of channels

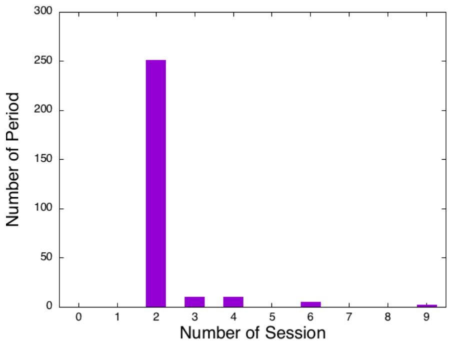


Figure 20: Number of sessions in period

I started by analyzing each sub-class separately; firstly the class where periods have only 2 channels and secondly periods with more than 2 channels. For periods with 2 channels, I started by checking the number of sessions. Figure 20 shows this distribution. We can see how most periods contained only 2 sessions yet others contained up to 9 sessions which indicates different behavior.

For periods with 2 sessions only, I calculated the overlapping duration between the sessions, using the formula $Dur_{real} - Dur_{true}$. Figure 21 shows the inverse distribution cumulative of overlapping durations. First remark is certain periods have sessions overlapping for over 2 Hours, such large duration was another indicator of an irregular behavior. This behavior brought me to the calculation of the overlapping duration divided by the duration of the shorter of the 2 sessions, in other terms the percentage of the shorter duration is being overlapped. Figure 22 shows the inverse cumulative distribution of such ratio, first thing we notice is that over 60% of periods have completely enveloped sessions. Such behavior can only be explained by 2 possible situations. Firstly, a user who manages to open the application twice on the same PC. Secondly, the application uses users with high internet speed to re-diffuse a second channel, which also can not be proven due to lack of access to the application's code. The other 40% have the same characteristics as that of first and second behavior that were found in section 4.2.1, except with the presence of overlapping.

Looking into the periods with 2 channels and more than 2 sessions, I used the same technique previously explained in 4.2.1 of comparing the number of switches with the number of sessions. 50% of the periods had its peer change channels between every session. This behavior would be explained by a user switching between 2 channels constantly, thus I consider this as the fifth *user*

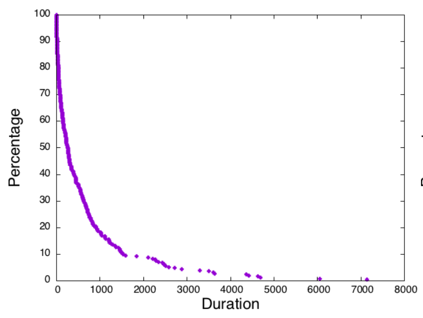


Figure 21: Inverse CDF overlapping duration

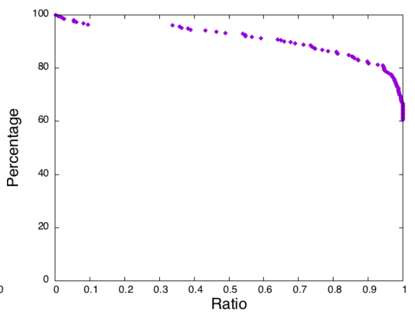
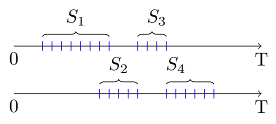


Figure 22: Inverse CDF ratio overlapping/shorter session

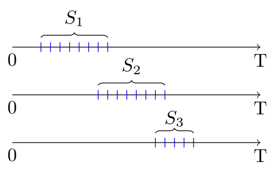
behavior. While the rest had a similar behavior however, at certain moments had 2 consecutive sessions on the same channel. Formally the number of switches was larger however, not large enough to prove that the peer changed channel between each channel.

Now we look into the final sub-class of periods, periods having more than 2 channels with sessions overlapping each other. I compared again the number of switches with the number of sessions. However I took in consideration the number of channels as well. The first *user behavior* in this sub-class and the sixth behavior that was detected, was peers rapidly zapping between 3 or 4 channels. These periods had the number of sessions equal to the number channels thus never being being on the same channel twice while the number of switches being equal to the number of sessions minus one. This behavior clearly characterizes someone looking for a program to watch or simply zapping around channels. The second *user behavior* I was able to detect was users rapidly zapping but who were coming back to a previously seen channel. Such behavior is similar to the forth behavior I detected in section 4.2.1. The difference between this behavior and the previous one is that the number of sessions is higher than the number of channels, yet the peer did change between each session. While the last behavior detected is exactly the same but with periods where the peer stays for at least 2 consecutive sessions on the same channel.

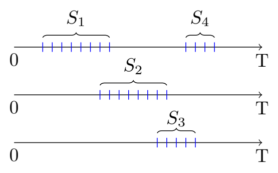
Figure 23 represents the 3 user behaviors that I found in the overlapping periods.



(a) *user behavior* 5



(b) *user behavior* 6



(c) *user behavior* 7

Figure 23: User behaviors

Chapter 5

Other datasets

At the end of the internship, I was given access to a different dataset, it had four main difference. Firstly the dataset was measured in France. Secondly the dataset was much longer; it was 4 days long. Thirdly the dataset contained 10 channels only and not 12 as the previous dataset, thus I had less vantage points. Finally the dataset was not measured at the same time of the previous dataset. The same process that was applied to the main dataset was applied to this dataset. However due to the lack of time, I wasn't able to deeply analyze the trace.

First thing I noticed is that the number of peers contact 329 436 which is a really high number yet not as high as we would expect in comparison to the main dataset; this is likely due to the location of the measurement which makes the PCs not good peer candidates for a larger number of users and we are more likely contacting peers that are present in Europe.

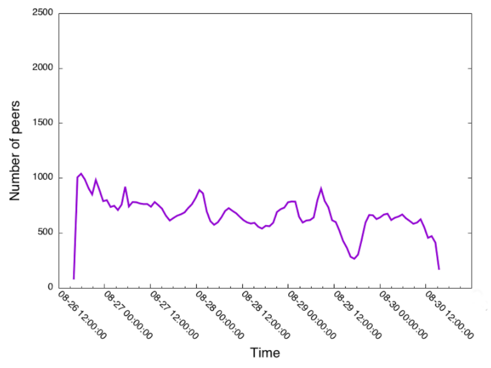


Figure 24: Evolution of number of peers

Figure 24 shows the evolution of number of peers. The first thing we can notice is the number of peers detected is almost always around 1000 except for certain peaks unlike the previous dataset where the number of peers was rarely near 1000. This effect can also be related to the geographic location of the measurement. Afterwards I looked into the number of multi-channel users since this was one of our main goals, the percentage of multi-channel users was 15% which is really close to the 15.86% we find in the previous dataset. I was able also to differentiate between servers and normal peers, where servers all had a large average of number of channels during the 1 minute interval. Finally I was able to check the presence of the same behaviors previously detected. All the behaviors were present but with different ratios.

Chapter 6

Conclusion

At the beginning of this Internship, P2PTV applications were presented briefly, and the interest behind the dynamism was explained. Thus, two challenges were present: First finding a non conventional method of analysis for P2PTV applications, Second detecting peers that zap or change channels since these peers are one of the main reasons behind the dynamism. By analyzing traces measured simultaneously over different channels, the method of depending on many partial points of view was proven to be feasible. This method was able to detect the difference between traffic, the role of peers and detect *multi-channel users* as well as traditional methods which are highly depending. And as we can see from the different datasets, the location of the measurement has a large impact on the number of peers detected, the number of channels that were followed had less of an impact however not much can be proved since the difference was only 2 channels. So the best configuration of duration, location and number of channels has yet to be found. But the fact that we were able to find common points between both datasets, is sort of a prove that the application is a general application rather than application specific for a certain dataset.

Secondly, using a formalization that was developed, different behaviors were detected. All which we can relate to when it comes to normal TV users. These behaviors can help us even to distinguish between what some would call zapping and changing channels. This information would help us in the future validate propositions to solve problems related to this dynamism or even solve bottleneck problems that occur which in theory appear due to dynamism which again is mainly present due to changing users.

Bibliography

[1] Tcpdump. http://www.tcpdump.org.

[2] Tshark. https://www.wireshark.org/docs/man-pages/tshark.html.

[3] Wireshark. http://www.wireshark.org/.

[4] E. Alessandria, M. Gallo, E. Leonardi, M. Mellia, and M. Meo. P2P-TV Systems under Adverse Network Conditions: A Measurement Study. *IEEE INFOCOM 2009*, 2009.

[5] Meeyoung Cha, Pablo Rodriguez, Jon Crowcroft, Sue Moon, and Xavier Amatriain. Watching Television Over an IP Network. In *Proc. ACM Internet Measurement Conference (IMC'08)*, volume 22, pages 71–84, 2008.

[6] D.Rossi, E.Sottile, and P.Veglia. Black-box analysis of internet p2p applications. *Peer-to-Peer Networking and Applications*, 4:146–164, June 2011.

[7] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Kieth W. Ross. A measurement study of a large-scale p2p iptv system. *IEEE Transactions on Multimedia*, 9(8):1672–1687, December 2007.

[8] Jinkang Jia, Chunxi Li, and Changjia Chen. Characterizing ppstream across internet. In *Proceedings of the 2007 IFIP International Conference on Network and Parallel Computing Workshops*, NPC '07, pages 413–418, Washington, DC, USA, 2007. IEEE Computer Society.

[9] Nazanin Magharei and Reza Rejaie. Understanding mesh-based peer-to-peer streaming. In *Proceedings of the 2006 International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '06, pages 10:1–10:6, New York, NY, USA, 2006. ACM.

[10] K Mizutani, T Miyoshi, and Olivier Fourmaux. *Traffic Analysis in Concurrent Multi-Channel Viewing on P2PTV*. Springer Berlin Heidelberg, 2015.

[11] Thomas Silverston and Olivier Fourmaux. P2P IPTV measurement: A comparison study. *CoRR*, abs/cs/0610133, 2006.

[12] Thomas Silverston, Olivier Fourmaux, Alessio Botta, Alberto Dainotti, Antonio Pescapé, Giorgio Ventre, and Kavé Salamatian. Traffic analysis of peer-to-peer iptv communities. *The International Journal of Computer and Telecommunications Networking*, 53(4):470–484, March 2009.

[13] M.Grangetto & M.Sereno S.Spoto, R.Gaeta. Analysis of pplive through active and passive measurements. In *Parallel & Distributed*, 2009.

[14] Guibin Tian, Yang Xu, Yong Liu, and Keith Ross. Mechanism Design for Dynamic P2P Streaming, 2013.

[15] Manxue Wang, Olivier Fourmaux, Yuko Nakamura, and Takumi Miyoshi. Network impact of p2p-tv zapping. IEEE/ACIS-SNDP, July 2013.

[16] Chin Yong Goh, Hui Shyong Yeo, Hyotaek Lim, Poo Kuan Hoong, Jay W Y Lim, and Ian K T Tan. A comparative study of tree-based and mesh-based overlay P2P media streaming. *International Journal of Multimedia and Ubiquitous Engineering*, 8(4):97–105, 2013.

Appendix A

The script that converts pcap files into the required textual format.

```
#! /bin/bash

echo "Converting␣files␣pcap␣to␣txt␣\n"
echo "in␣directory␣$1"

cd $1
for d in *; #first loop into all folders
do
        echo "entering␣$d"
        cd "$d" && ls
        touch $d.txt #Creation of text file
        for i in *.pcap #Looping PCAP files and transfering them
        do
                /usr/sbin/tcpdump -tttt -n -q -r $i
                | awk '{print $1,$2,$4,$6,$7,$9,$8}'
                | sed  's/\(.*\):/\1/'
                |  sed 's/length//g' >>  $d.txt
        done
        echo "exiting␣$d"
        cd ..
done

cd -
```

The two main functions responsible for aggergated analysis as well as sliding window analysis

```
//MAIN FUNCTION FOR AGGREGATE ANALYSIS
void file2data_PCAP_batch(string name,vector<string> channels,Graph * g){
        ifstream file(name.c_str());
        string str,t,b,time_str,tmp,hours;
        struct tm tm;
        time_t t1 = 4;
        int size_pack = 0;
        int z =  0;
        while (getline(file, str)){ //loop around each file
        z++;
        istringstream iss(str);
        iss >> time_str;
        iss >> hours;
        time_str.append("␣" + hours);
        iss >> b;
        iss >> t;
        iss >> tmp;
        iss >> tmp;
        if(tmp.compare("ip-proto-17")==0){
                continue;
        }
        size_pack = atoi(tmp.c_str());
        if(size_pack == 0){
                continue;
        }
        size_t n = count(b.begin(), b.end(), '.');// Removing port numbers
        if(n==4 && find(channels.begin(), channels.end(), b) == channels.end()){
                unsigned found = b.find_last_of(".");
                b = b.substr(0,found);
        }
        n = std::count(t.begin(), t.end(), '.');
        if(n==4 && find(channels.begin(), channels.end(), t) == channels.end()){
                unsigned found = t.find_last_of(".");
                t = t.substr(0,found);
        }
        }// Differentiating between Machine IP and Peers IP
        if(my_own_regex(b)&&find(channels.begin(), channels.end(), b) == channels.end()){
                continue;
        }
        if(my_own_regex(t)&&find(channels.begin(), channels.end(), t) == channels.end()){
                continue;
        }
        // Saving the communication by calling the function addlink
        if(find(channels.begin(), channels.end(), b)!=channels.end()
                && find(channels.begin(), channels.end(), t)==channels.end()){
                addlink(g,b,t,&time_str,-size_pack);
        }else if(find(channels.begin(), channels.end(), t)!=channels.end()
                        && find(channels.begin(), channels.end(), b)==channels.end()){
                        addlink(g,t,b,&time_str,size_pack);
                }
    }
    g->density = g->links / (float)(g->tops.size()*g->bots.size());
    file.close();
}
//MAIN FUNCTION FOR INTERVAL ANALYSIS
void file2dataPCAP_interval(ifstream * file,vector<string> channels,int interval,Graph *g){
        string str,t,b,time_str,tmp,hours;
        struct tm tm;
        time_t t1 = 4;
        time_t t2 = 4;
        int size_pack = 0;
        while (getline(*file, str)){
        istringstream iss(str);
        iss >> time_str;
        iss >> tmp;
        time_str.append("␣" + tmp);
        iss >> b;
        iss >> t;
        iss >> tmp2;
        iss >> tmp2;
        // Filters
        if(tmp2.compare("ip-proto-17")==0){
                continue;
        }
        size_pack = atoi(tmp2.c_str());
        if(size_pack == 0){
                continue;
        }
        // testing if out side of window
        if(count(b.begin(), b.end(), '.') > 2 &&  count(t.begin(), t.end(), '.') > 2){
                if(t1 == 4){
                        t1 = timestamp_to_ctime(time_str.c_str());
                        g->set_time(time_str);
                }else{
                        t2 = timestamp_to_ctime(time_str.c_str());
                        double diff = difftime(t2,t1);
                        if(diff > interval){
                                break;
                        }
                }
                // Removing Port
                size_t n = count(b.begin(), b.end(), '.');
                if(n==4 && find(channels.begin(), channels.end(), b) == channels.end()){
                        unsigned found = b.find_last_of(".");
                        b = b.substr(0,found);
                }
                n = std::count(t.begin(), t.end(), '.');
                        if(n==4 && find(channels.begin(), channels.end(), t) == channels.end()){
                                unsigned found = t.find_last_of(".");
                        t = t.substr(0,found);
                        }
                        // Differentiating between Machine IP and Peers IP
                        if(my_own_regex(b)&&find(channels.begin(), channels.end(), b) == channels.end()){
                continue;
                        }
                        if(my_own_regex(t)&&find(channels.begin(), channels.end(), t) == channels.end()){
                                continue;
                        }
                        // Saving the communication by calling the function addlink
                    if(find(channels.begin(), channels.end(), b)!=channels.end()
                        && find(channels.begin(), channels.end(), t)==channels.end()){
                                addlink(g,b,t,&time_str,-size_pack);
                    }else{
                            if(find(channels.begin(), channels.end(), t)!=channels.end()
                                && find(channels.begin(), channels.end(), b)==channels.end()){
                                        addlink(g,t,b,&time_str,size_pack);
                            }
                        }
            }
        }
        g->density = g->links / (float)(g->tops.size()*g->bots.size());
}
```