POLITECNICO DI TORINO

SCUOLA DI DOTTORATO

Dottorato in Ingegneria Elettronica e delle Comunicazioni – XXV ciclo

Tesi di Dottorato

Inferring Network Usage from Passive Measurements in ISP Networks

Bringing Visibility of the Network to the Internet Operators

Ignacio Bermudez Corrales

Tutore
prof. Marco Mellia

Coordinatore del corso di dottorato
prof. Ivo Montrosset

Febbraio 2013

Abstract

The Internet is evolving with us along the time, nowadays people are more dependent of it, being used for most of the simple activities of their lives. It is not uncommon use the Internet for voice and video communications, social networking, banking and shopping.

Current trends in Internet applications such as Web 2.0, cloud computing, and the internet of things are bound to bring higher traffic volume and more heterogeneous traffic. In addition, privacy concerns and network security traits have widely promoted the usage of encryption on the network communications. All these factors make network management an evolving environment that becomes every day more difficult.

This thesis focuses on helping to keep track on some of these changes, observing the Internet from an ISP viewpoint and exploring several aspects of the visibility of a network, giving insights on what contents or services are retrieved by customers and how these contents are provided to them. Generally, inferring these information, it is done by means of characterization and analysis of data collected using passive traffic monitoring tools on operative networks.

As said, analysis and characterization of traffic collected passively is challenging. Internet end-users are not controlled on the network traffic they generate. Moreover, this traffic in the network might be encrypted or coded in a way that is unfeasible to decode, creating the need for reverse engineering for providing a good picture to the Internet operator. In spite of the challenges, it is presented a characterization of P2PÂTV usage of a commercial, proprietary and closed application, that encrypts or encodes its traffic, making quite difficult discerning what is going on by just observing the data carried by the protocol. Then it is presented DN-Hunter, which is an application for rendering visible a great part of the network traffic even when encryption or encoding is available. Finally, it is presented a case study of DN-Hunter for understanding Amazon Web Services, the most prominent cloud provider that offers computing, storage, and content delivery platforms. In this paper is unveiled the infrastructure, the pervasiveness of content and their traffic allocation policies. Findings reveal that most of the content residing on cloud computing and Internet storage infrastructures is served by one single Amazon datacenter located

in Virginia despite it appears to be the worst performing one for Italian users. This causes traffic to take long and expensive paths in the network. Since no automatic migration and load-balancing policies are offered by AWS among different locations, content is exposed to outages, as it is observed in the datasets presented.

Acknowledgements

I would like to express my gratitude to everyone who has helped me along the way with my studies at Policenico di Torino. Firstly, I would like to say thanks to all people in the networking group that I have been in touch during these last 3 years. In particular, I am thankful for all the thing I have learnt from my advisor, Marco. I appreciate his efforts on giving opportunities to his students, his contribution on TSTAT and his networking which made possible to get all the datasets used throughout in this thesis. Also, I would like to mention Maurizio who is actively working on TSTAT and provided highly technical support. Finally, from the networking group professor staff, I appreciate all the scientific advices provided by Michela in my first year of Ph.D. From the student side of the networking group, I would like to acknowledge Miquel who was more than an office-mate, Alessandro for these contributions to TSTAT, technical chit-chatting and friendship, Edoardo who shared office during the first year of Ph.D and I have got a lot of fun, Ricca for sharing bunch of thoughts and Stefano for making this group not just a working place.

During my visiting periods in the CTO group of Narus, I have got the opportunity to meet quite valuable people. Among them, I would like to mention Marios, who help me a lot on technical and scientific issues, and fortunately I got the chance to work close to him during these periods. I appreciate the work being carried out by Ram, the boss in all the projects that I was involved with, Antonio for giving prospectives to Ph.D students outside the academia, and finally, to Ruben for this time shared outside Narus.

Finally, I dedicate this thesis to all these people “behind” of what I am, my family, mother, father, sister and now Macarena, my life mate. She is the one that stayed and shared everything with me, from the time to thoughts of an ideal world we want to construct. With her support and patience, things are quite easier and life happier.

Contents

Chapter 1

Introduction

The Internet have been with us for several years and will stay for longer. Along all this time, Internet has evolved drastically as culture trend, market and politics evolve as well. The changes on the Internet are reflected on several levels, from how the end-users interacts on the network, to how content is delivered and the transmission capacity is getting available to everybody. Considering just the technological part, Internet has changed because of end-users now can buy cheap and powerful computing devices, wireless connections are in anywhere providing connectivity to anything, video transmission, which represents tons of data, is quite popular thanks to cheap devices able to capture high resolution digital video, cheap storage devices which allows to save tons of data, high-speed networks which makes feasible content sharing and infrastructures, like Youtube, which can host unlimited audiovisual content for free almost no matter the duration of the film, and thanks to high-res displays which make users being more interested on better video content. Therefore understanding what end-users do when they are online and the impact of their actions in a network, from the point of view of an operator, or commonly known ISP (Internet Service Provider), is essentially for providing Internet access to end-users. This knowledge empowers ISP permitting a better administration of the network, the creation of new business and to react faster to Internet changes as well as failures in the network. It also helps to software engineers, which develop on-line applications, to develop better applications that are able to coexist with the rest.

This thesis explores several aspects of the visibility of a network, giving insights on what contents or services are retrieved by customers and how these contents are provided to them. Generally, inferring these information, it is done by means of passive network monitoring of operative networks, from where traffic is collected to then being characterized and analysed. From the point of view of who analyse the Internet, this is not a simple task. Internet is a quite tangled and confusing place in which many actors are involved, and the information available about it is not complete, e.g. some Internet traffic is encrypted or encoded in a obscure way. Particularly, the core of this thesis is organized in three different chapters: (i) in chapter 2 is studied the human factor and its impact on the network, by unveiling the usage P2P-TV applications over the Internet, (ii) in chapter 3 is presented DN-Hunter a generic tool that permits association of network connections to content/service, even in cases which there is encryption or unknown coding, finally (iii) in chapter 4 is presented a study of Amazon infrastructure, which is one the big players providing content/service to end-users.

Among the most prominent findings covered in this thesis, in chapter 2 is shown evidence on why users in an European country select one channel instead of other, which is the most popular content to watch on P2P television and how these combined with the application design, impact together the network in terms of traffic locality. Moreover it is demonstrated that without high speed peers in the P2P overlay, broadcasting of real-time content becomes difficult. In 3 is presented a tool, completely projected and coded by the author, which allows to associate the real used domain name with a network connection in real-time. Together with this, there is demonstrated how vague is reverse lookup for detecting the service or content of a given connection and the amount of information that an operator can access for free from DNS traffic in its network. Last, in chapter 4 a study case of DN-Hunter is presented, putting particular attention to Amazon Web Services, which offer dedicated Cloud Computing, Internet storage and a Content Delivery Network. This chapter also introduces a generic methodology to assess the geographical location of an IP address which becomes useful to determine the location of the different Amazon datacenters, permitting in this way understand from where the traffic is coming, to where is exported, which are the datacenters performing worst, and thanks to DN-Hunter understand which content performs worst and where are located the different contents. Among all the findings, it is interesting that most of content generated by the Cloud Computing infrastructure is imported from America, being available datacenters for cloud-computing in Europe.

Finally, this thesis is concluded with chapter 5, which briefly describes the experience during three years of monitoring actively the Internet at the edge, recalling the difficulties on characterizing the interactions in the Internet and commenting some toughs of the author.

Methodology and Collaborations

All the works presented through this thesis are based mostly on traces of network traffic collected from passive measurements on operative ISP networks. Basically, the analysis an characterization of traffic from passive measurements has the advantage that what is being observed is not bias and represents normal behavior of real end-users. On the other hand, virtual hosting of content or service mixed with privacy and security concerns make this task difficult. E.g. consider someone is interested on Facebook traffic, most of it is encrypted using TLS/SSL and it is hosted using several IP addresses. This is the same that happens with other not very well-known applications generating traffic, e.g. traffic generated by a botnet, could you tell what is going on when traffic protocol is obscured intentionally?

In this work, *Tstat* DPI developed in Politecnico di Torino has been used extensively. It has capabilities of classification, generates statistics about TCP/UDP connections providing hundred of features of a flow, as well as other statistics related to some specific protocols and content. Signatures for *Tstat* classifier are written by hand and are the result of hard reverse engineering of the protocols. In this context *Tstat* has been placed in several vantage points spread over Europe. Thanks to Network-Aware P2P-TV Application over Wise Networks (NAPAWINE) a Seventh Framework Programme project and the collaboration with of some partners, with the scope of understanding P2P-TV applications and the attitude of end-users towards it, some *Tstat* probes has been placed in Telekomunikacja Polska (now Orange Polska, Poland), Magyar Telekom (Hungary), in Fastweb (Italy) and in Politecnico di Torino (Italy). Then thousands of end-users have been monitored with the scope to unveil and characterize P2P-TV usage.

Given the second problem of generic traffic being obscured or coded in an unknown fashion, there is a collaboration with Narus Inc. (USA), in which DN-Hunter has been developed. This collaboration has as scope the creation of a labeler for automatic generated signatures of protocols, but comes out that DN-Hunter was more than a labeler for this specific application, providing visibility to the encrypted or coded traffic to the network operator. DN-Hunter has been included in one private version of *Tstat* for research purposes. Therefore using some available probes able to run *Tstat*, it was possible to understand better such as big infrastructures as Amazon Web Services, providing insights of its overall performance and the way this company distribute its content to end-users.

Main Contributions

Throughout the chapter 2 you may find evidence that supports that P2P-TV activity in Europe is high correlated with sports events, particularly football matches being broadcast on paid TV. It is noted that are cultural biases on channel selection, in which end-users running P2P-TV applications prefer to retrieve the same content from channels with audio in particular languages, even though alternatives, broadcasting essentially the same content, are available. This has implications on traffic that remains naturally in a particular geographical region, notwithstanding P2P-TV applications do not favor or promote any kind geographical traffic enclosing. On the other hand, it happens when content is not available in the local language end-users, they retrieve content from another channel broadcasting the content. This have negative implications for geographical traffic allocation, for instance when channel content is being transmitted in English, which is the language most spoken as a secondary language, the spatial distribution of peers is sparse, then end-users the same happen to traffic allocation, where a lot of the traffic is being exported/imported from abroad. It is observed that there is a constant churning rate, which means peers or end-users leaving the channels. This is important to have in mind for scaling P2P-TV infrastructure, since P2P applications rely on end-user capability to replicate the information over the P2P overlay. Moreover, the behavior of peers is quite synchronized. That is an effect of flashÂcrowd when events starts and high churning when broadcast event, e.g. football match ends. In spite of generic P2P applications, there are two particular methodologies which allowed to find all the already described results. The first is a methodology that permits the isolation of P2P-TV traffic by channels, and the second one is a methodology to estimate the the size of a P2P swarm in a scenario in which the peer discovery is not forced, that is, with the same probability each peer in the swarm may exchange some traffic with any other peer in the swarm.

On chapter 3 is presented DN-Hunter, which allows the association any TCP/UDP flow with its corresponding Fully Qualified Domain Names (FQDN). A FQDN may provide granular information about the service or content associated with a network connection, even though a network connection is encrypted. It has been implemented as part of DPI solutions, as *Tstat* a open-source DPI developed at Politectino di Torino and it is inside Narus Insight, a commercial DPI solution developed at Narus Inc. In this thesis contains a brief description of DN-Hunter, which includes internals of DN-Hunter and algorithms for giving visibility of the network using the FQDN annotated connections. Among the algorithms presented here, an automatic port labeling is described which mixes flow features as a destination TCP/UDP port number, with the domain names associated to the connections using some particular ports, in a way that DN-Hunter provides labels for unknown ports being used in the network, pointing out information for a network administrator. Moreover, it has been used to label kind of traffic which share a particular signature, in the context of an automatic application signature learner, DN-Hunter provides labels for the new protocols it founds in the network. Heuristics for content discovery and how different content is being hosted on different infrastructures is shown, presenting findings related to web services which operate differently according to the geographical location they are serving. Also, it is shown that DN-Hunter allows to a network administrator to have a grasp on the kind of content being provided by a general purpose cloud. An example of Google Appspot, a general purpose cloud for running applications for free on Google machines, shows that with DN-Hunter is possible to determine that this cloud is abused by BitTorrent trackers for sharing probably copyrighted content. This is particularly astonishing since content is served by Google IP addresses which are well known to serve "bad" content, moreover other legacy service run on the same IP addresses, then blocking trackers by their IP address would break the whole connectivity to other innocuous applications.

Chapter 4 is a complete study of one big player of the Internet nowadays as observed from the point of view of an ISP. This is the case of Amazon Web Services (AWS), from which traffic directed to it is passively gathered from all the traffic. One of the main contributions is a heuristics for determining the location of an Internet address as observed from a particular location, bypassing in this way any particular routing configuration which could trick some other accurate methodologies like [1], which uses several probes spread on different geographical locations. Using DN-Hunter, this chapter provides insights on performance of the different datacenters, performance of the different services hosted on AWS cloud as well as content hosted on their Content Delivery Network, hosting policies and pricing for customers with implications on geographical traffic allocation and eventually on QoS for end-users, which have to import traffic from long distances when it could be available from nearest datacenters. In spite of this, it is found that Italian end-users import near the 85% of traffic volume from America, when a datacenter is available in Ireland. This impairs the quality of communication with the servers specially for the case of TCP connections. In addition, it is shown which kind of content/service is responsible for such traffic importing, finding that mostly online social games are quite popular among Italian end-users and that those applications are mostly hosted in Amazon datacenter located in North Virginia, near to Washington.

Chapter 2

Inferring Internet end-user attitude

2.1 Introduction

As the Internet changes, in the recent years we have witnessed the success of P2P-TV applications, bringing TV channels, some of which live, to the users' home through the Internet. Several commercial P2P-TV systems are available and some are popular among users because they feature cheaper video broadcasting than other solutions, e.g., IPTV or pay-TV. Unfortunately, most of the successful P2P-TV applications rely on proprietary protocols and unknown algorithms, so that the understanding of such systems is intrinsically complex. Thus P2P-TV traffic characterization has become a topic of great interest for the research community and for network operators. Both are interested in understanding the positive and negative aspects of P2P-TV applications, to understand how these complex systems work and to improve their design and effectiveness.

Service providers, network operators and designers, are interested in assessing the potential impact of this traffic on the network of today, impact that might turn out to be disruptive, given the possible large number of users and high bandwidth requirement combined with the traffic being loosely controlled with respect to network conditions. Researchers are interested in the investigation of end-users attitude towards these new services to foresee new trends in the future usage of the Internet, and to augment the design of their application. A deep understanding of P2P-TV traffic and its characterization is therefore an important task that can contribute to the design of network elements, including traffic engineering mechanisms, component dimensioning, resource management strategies.

In P2P-TV systems, three different graphs can be identified. The first graph represents the users that run the applications forming a "social network graph". Where are the users? When and for how long do they run the application? Is churning relevant for P2P-TV systems? These and others are all relevant questions whose answer allows researchers to design more robust applications, e.g., by exploiting natural localization properties of users.

Peers form then an “overlay topology”, a second graph where peers are interconnected by logical links. Peers interested in the same channel then form a “swarm”; independent overlay topologies are built for different swarms. Which are the properties of the swarms? Are the peer neighbors carefully selected or are they randomly chosen? By understanding the overlay topology graph properties it is possible to understand the P2P-TV system properties, its robustness to churning or its scalability with respect to the number of peers.

Finally, the subset of the overlay links that are used by peers to exchange the video traffic forms the third graph, the “distribution graph”. Is the video data being downloaded from neighbors in the same Autonomous System to reduce the network provider cost? Are neighbors with larger upload capacity preferentially selected to download content from? What is the fraction of high capacity peers in a swarm? Recall indeed that the total available upload capacity plays a key role in the success of P2P-TV content distribution since the video stream must be downloaded at an almost constant rate by each peer.

To answer most of the latter questions, this chapter contributes to the characterization of P2P-TV traffic by analyzing the traffic due to popular applications (SopCast, TV-Ants and PPLive), in the operative links of four networks in operation in Europe, three of which provide ADSL access, the forth one employs FTTH (Fiber-To-The-Home) technology. Differently from the measurement works present in the literature, here there is adopted a pure passive methodology to observe normal usage of P2P-TV applications by customers. Collecting traffic for more than one year, it is found that SopCast is the largely preferred application by customers in these networks. Furthermore, the usage of these applications is still very much discontinuous and often associated to events, such as sport events, that are popular but expensive to retrieve through normal TV broadcasting systems.

As a study case, then focus on two months during which the UEFA Champions League 2009 final matches were held. Investigating deeper into the SopCast traces, it is reported traffic and peer volumes, swarm evolution, peers geo-localization and lifetime, and their contribution to the video distribution. Results suggest that the implications of traffic burstiness, the peer population and their evolution might become challenging for the network, should these applications become widely popular.

Methodologically, it is proposed a general heuristic to identify swarms corresponding to TV channels; observing churning associated to SopCast events, finding out that users stay connected to the P2P-TV system for the whole duration of the event, but they can frequently change swarm seeking for better channels broadcasting the same event.

In spite of the peer discovery process in the overlay topology, it is found out that SopCast implements a simple random discovery which is very robust. Conversely, the distribution graph is severely biased by peer upload capacity and by the Autonomous System a peer belongs to. Results suggest that the implications of traffic burstiness, peer population and their evolution over time might become challenging to face.

Some key aspects highlighted in this chapter include:

• Despite the average bandwidth usage of P2P-TV applications is not significant, it can be substantial during periods in which popular events are shown. Today, a few tens of users can contribute to 15% of total aggregate traffic generated by more than 20,000 users on a network access link.

• Node churning during the lifetime of a stream is not significant, but there is a flash crowd entering the system at the beginning of the event and a rush towards exit at the end. This clearly has an impact on the design of P2P-TV applications.

• Evidence shows that often high-speed residential networks and University networks altruistically serve content to residential peers with highly asymmetric bandwidth. Without the contribution of those peers, the P2P-TV system would not sustain the service at all.

• Geo-locality of social network graphs is deeply affected by cultural and language trait of customers. This biases the traffic distribution graph that is inherently geographically localized.

The latter two facts clearly impact the ability to localize P2P traffic, a theme that is currently debated in the research community.

2.2 P2P-TV Usage

In this section usage and popularity of P2P-TV among Internet end-users. Fig. 2.1 report the P2P-TV average incoming bitrate versus time, for different timescales observed at the EU2-ADSL vantage point. Results are qualitatively similar in other monitored PoPs. On average, the traffic generated by these applications is marginal, but the *burstiness* of traffic reflects P2P-TV usage that is concentrated during short periods of time. This is when the amount of traffic generated can reach very high and possibly disruptive peaks. Moreover, it is observed that P2P-TV activity typically coincides with the transmission of popular sport events, e.g. UEFA Champions League during Wednesday and Thursday or Premier League (England First Division) on Saturday and Sunday. It is observed that even the most popular events reach as much as a hundred of same PoP end-users, which correspond to less than 0.5% of all monitored end-users connected to the vantage point. Still, the download bitrate often exceeds 15% of total PoP incoming traffic during those events. P2P-TV bitrate during peaks is larger than the aggregated YouTube bitrate consumed by customers in the same network. Nevertheless P2P-TV usage is disruptive and consume lot of network resources, only a few end-users connected to PoPs run these P2P-TV applications. This "bursty" user behavior, which can be pretty difficult to handle, is also very different from normal TV and IPTV usage pattern, that is typically smoother and more evenly distributed during the day. Notice also the abrupt drop of traffic that happens after 20:30, i.e., after the event ends. This hints that flash crowd phenomena are not negligible in P2P-TV systems, as it discussed in next sections.

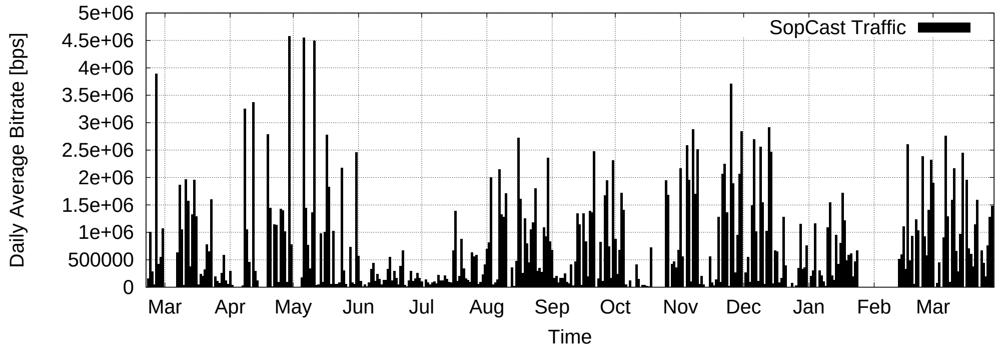
On all the monitored vantage points it is observed at the time in which the measurements were collected, that in these European networks SopCast is by large the most popular P2P-TV application among others like PPLive and TV-Ants, which traffic signatures can be identified using Tstat DPI. Therefore, in the following, the analysis of P2P-TV is restricted to some of the largest traces of SopCast traffic, which remains statistically more relevant. Since SopCast adopts a proprietary protocol and relies on encryption mechanisms, reverse engineering of the application protocol and algorithms are avoided. Instead, all the characterization of the application and its usage is devised using simple methodology that can be leveraged to study other P2P applications too.

2.3 Channel Identification

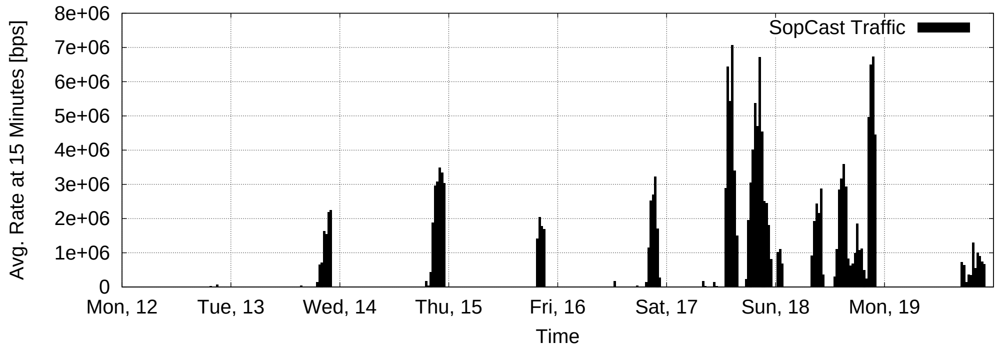
The information on whether the observed peers are watching the same "*channel*", i.e. peers watching different channels belong to different disjoint swarms, is interesting since it can be leveraged to better characterize and categorize the different channels and users. Unfortunately, identifying the channel turns out to be complex from passive monitoring of uncontrolled peers. Worst, SopCast adopts its own proprietary protocol and uses encryption mechanisms, that makes harder the channel identification by inspection of protocol messaging.

In order to avoid complex (and questionable) reverse engineering of the SopCast protocol, it is defined a methodology that allows to cluster peers in swarms, that is a group of peers watching the same channel. This methodology is generic and can be leveraged for most P2P systems as well. The intuition at the base of this solution is that peers in a given channel contact other peers that are part of the same channel too. Then it is defined the concept of *neighborhood* of a peer, as the set of peers that have established any connection with the peer. Formally it is claimed that peers with similar neighborhoods belongs to the same channel. For instance consider two peers a and b which have a lot of neighbors in common, both belongs to the same channel. On the other hand, consider a third peer c which have only a few neighbors in common, it is said that c belongs to a different channel.

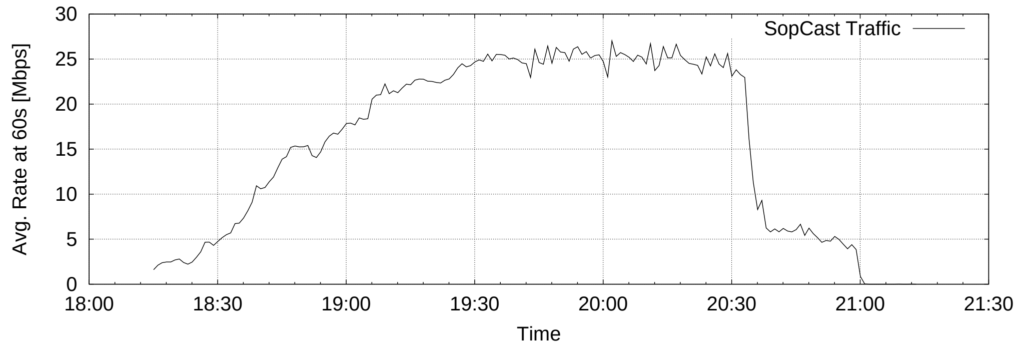
Let a and b denote two internal peers and let $P(a)$ be the set of peers *contacted* by a, i.e., peers which a sent a packet to. The amount of common peers among



(a) Average daily bitrate over one year.



(b) Average 15min bitrate over one week.



(c) Average 60s bitrate over one evening.

Figure 2.1: P2P-TV Traffic at Different Time-scales in the TP PoP.

a and b is then $C(a,b) = |P(a) \cap P(b)|$, where $|\cdot|$ is the cardinality operator. It is defined then the common peer matrix M, as a matrix in which element (i,j) is $M_{ij} = C(i,j)$.

If each row (column) on the M matrix represents a monitored peer, then by sorting peers, so that two adjacent rows (columns) in M refer to peers that have a large number of common peers and their neighborhoods are similar as well, then it is easy to identify the different swarms by just looking at the rows (columns) of the matrix. Neighborhood similarity is computed using the number of contacted peers in common, in a way that it gives similarity score if two peers a and b have external peers in common, but it gives an even higher score if there is a third peer c which also shares peers with a and b, and a forth d which do not have peers in common neither with a, b and c. Let V_a be the vector of common peers of a with all other monitored internal peers, i.e., the a-th row of M. Denote by V_a^T the transposed of V_a, i.e., the a-th column of M. The product,

$$S(a,b) = 2\frac{V_a V_b^T}{V_a V_a^T + V_b V_b^T} \tag{2.1}$$

is a measure of the similarity between the neighborhoods of a and b. By iteratively sorting the list of peers and moving closer those with larger similarity, it is obtained the *swarming matrix*, i.e., an ordered common peer matrix M' that depicts in a clear way how peers are clustered together.

Fig. 2.2 reports the swarming matrix considering 133 peers active for more than 600 s during 2-hours event on the 5^{th} of May 2009. Each cell is colored according to the amount of common peers it represents. The numbers along the main diagonal correspond to the total number of contacted peers, $P(a)$. The swarming matrix shows that there are several groups of peers that share a large fraction of common peers, identified by the darker blocks. The largest block includes peers from 0 to 70 (named swarm A), the second group corresponds to peers from 105 to 126 (swarm B), then peers from 90 and 105 (swarm C). The magnitude of well defined clusters of peers with intersections of neighborhood suggests that they correspond to different swarms, or channels. Moreover it is possible to understand the swarming matrix as an *adjacency matrix*, where each peer is a node in a graph and dark cells represent connections between peers of the same channel. In the same way, dark blocks represent very well connected components or in P2P-TV jargon, a channel. It is inferred that during the 5^{th} of May event users were watching different channels that were possibly broadcasting the same event. Then, each identified swarm is also characterized by very different properties, which corroborate this claim.

Swarming matrices allow grouping of users watching the same channel. As a first result, it clarifies that some channels are more popular in the monitored network. Interestingly, it is possible to observe from the matrix that during the same short period of time, several channels were active, probably transmitting same content. This assertion is enforced by the observation of peers changing from one channel to another, as you can read in section Sec. 2.3.1. Probably these channels provide essentially same content, but some features such as video quality, sound quality, channel stability or even channel, could be different.

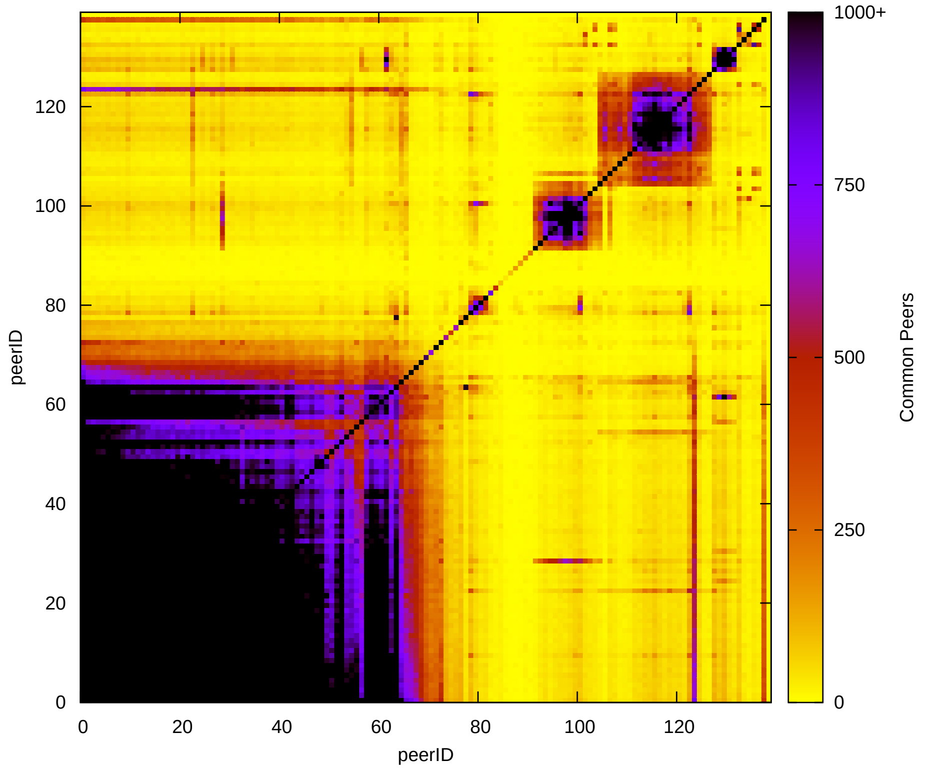


Figure 2.2: Swarming matrix for 5th of May trace. Darker blocks refer to peers in the same swarm.

The swarm analysis is repeated over all the traces, identifying several swarms. In this section the analysis and results are restricted to the subset of swarms reported in Tab. 2.1, which are the largest swarms in terms of number of present internal peers. The table summarizes prominent swarm characteristics: the number of internal peers observed in the channel and the estimated amount of external peers in the channel, the total amount of received (RX) and transmitted (TX) data, estimated video rate, probe country code (CC) location and the portion of external peers that belong to the same Autonomous System (AS) the probe was located in. Note that channels are sorted by decreasing values of the last metric. Note that all the largest swarms were observed in the EU2-ADSL traces, being P2P-TV usage more popular in EU-CC1 than in the other two European countries. Nonetheless, in swarm 11 and swarm 14 is identified one peer that was monitored in EU3-ADSL and EU1-ADSL2, respectively. These are listed in the two bottom rows of the Table.

The number of external peers in the channels are estimated using a metric explained in detail in Sec. 2.4.3. Video rate, is computed considering the number of peers watching the channel and the total amount of video data observed, during a window time of 1 minute. The video data is discriminated from the signaling data by taking advantage of the very biased distribution of packet size of SopCast. Furthermore packets which hold more than 1000 bytes in their payloads are considered as video packets. Results show that the video rate is typically lower than 480kbps, i.e., low quality video. Other metrics reported in the table are straightforward computed from traces.

Table 2.1: List of the largest swarms

Swarm ID	Internal Peers	External Peers	RX [GB]	TX [GB]	Video Rate [kbps]	CC	PL AS %
0	35	15489	8	2	330	PL	32
1	29	19701	8	2	400	PL	31
2	50	32757	15	3	450	PL	28
3	33	25575	9	2	400	PL	27
4	41	33320	15	3	400	PL	27
5	69	60502	23	5	420	PL	25
6	66	76416	24	5	470	PL	23
7	19	20662	8	2	350	PL	21
8	77	68264	30	6	400	PL	19
9	11	10371	3	1	440	PL	10
10	5	12288	1.5	0.2	320	PL	9
11	12	18684	5	1	430	PL	9
12	10	20930	2	0.2	370	PL	8
13	8	13591	2.6	0.4	450	PL	8
14	16	39948	4.3	0.4	380	PL	6
15	13	39718	4	0.3	450	PL	5
16	25	54830	5	1	470	PL	4
17	8	23333	2.7	0.3	480	PL	3
18	10	30026	3	0.5	330	PL	2
19	9	27195	2.1	0.5	400	PL	2
11	1	8049	0.5	0.2	430	HU	N/A
14	1	5122	0.3	0.1	380	IT	N/A

2.3.1 Detecting Channel Switching

Usually when someone watches TV use to change channel often. This behavior is commonly known as zapping, which is using the remote control device to switch across channels or TV programs. Motivated on finding differences between P2P-TV and normal TV, the user satisfaction is studied in this section, measuring how much are like the users to change the channel that they were watching. From the swarming matrix, there are still some peers which share some common peers with others in disjoint channels. For instance, peer 123 at Fig. 2.2 as a large number of common peers with both, swarm A and swarm B. Has this peer watched both channels at the same time, or somehow it switched from channel A to B or vice-versa during the 2 hours of monitoring?

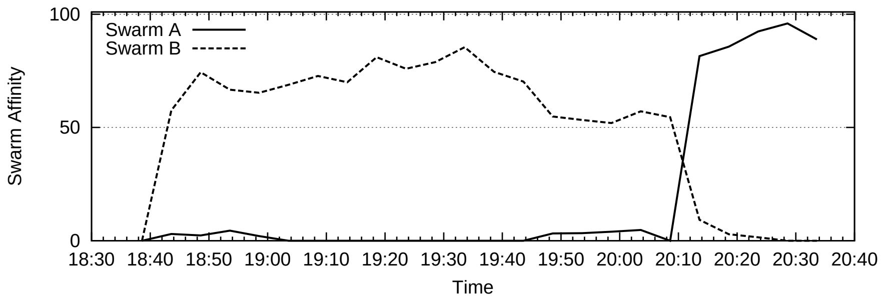


Figure 2.3: Affinity of peer 123 to swarm A and B during the 5th of May trace.

To support the hypothesis that a peer had jumped from one channel to another, instead of watching both contemporary, it is defined the following test. Let X and Y be the sets of internal peers that belong to different swarms. Then,

$$P(X,\bar{Y}) = \cup_{x\in X} P(x) \setminus \cup_{y\in Y} P(y) \tag{2.2}$$

is the set of unique external peers contacted by peers of swarm X but not by peers of swarm Y. Let $P_{\Delta T}(a)$ be the set of peers contacted by the internal peer a during time ΔT. It is defined the *swarm affinity* of peer a to swarm X and not to Y as

$$A_{\Delta T}(a,X,\bar{Y}) = 100\frac{|P(X,\bar{Y}) \cap P_{\Delta T}(a)|}{|P_{\Delta T}(a)|} \tag{2.3}$$

Fig. 2.3 shows the affinity of peer 123 to swarm A and swarm B, considering $\Delta T = 5$ minutes moving along the event duration. The plot shows that peer 123 exhibits a high swarm affinity towards swarm B from 18:30 until 20:10, time at which its affinity to swarm B drops and the one to swarm A increases. Notice peer 123 left swarm B to join swarm A at 20:10. Identical results are obtained when considering other peers showing same connection pattern. Finally there is evidence of channel switching when there are other available channels providing same content with different features.

2.3.2 Natural Churning and Flash Crowds

Differently to normal TV, on P2P-TV is observed a very like event driven behavior among P2P-TV users Fig. 2.1. When someone runs SopCast it is probably that some particular TV show or event is being broadcast at that time, so that a flash crowd effect is present when an event starts. Users then keep running SopCast for the whole event duration, and then suddenly stop using it at the event end. This is confirmed by Fig. 2.4 which reports the Cumulative Distribution Function (CDF) of the time which internal peers are active using P2P-TV during an event. This time is called the lifetime of the peer. Both separated and aggregated CDFs are reported. Results show that users lifetime is very similar for different events, and it is rather long, e.g., 90% of users have a lifetime longer than 30 minutes. As highlighted by the horizontal lines, 50% of users have a lifetime within 90 min and 130 min, which corresponds to the typical duration of a soccer event. Only less than 10% of users run SopCast for more than 150 min.

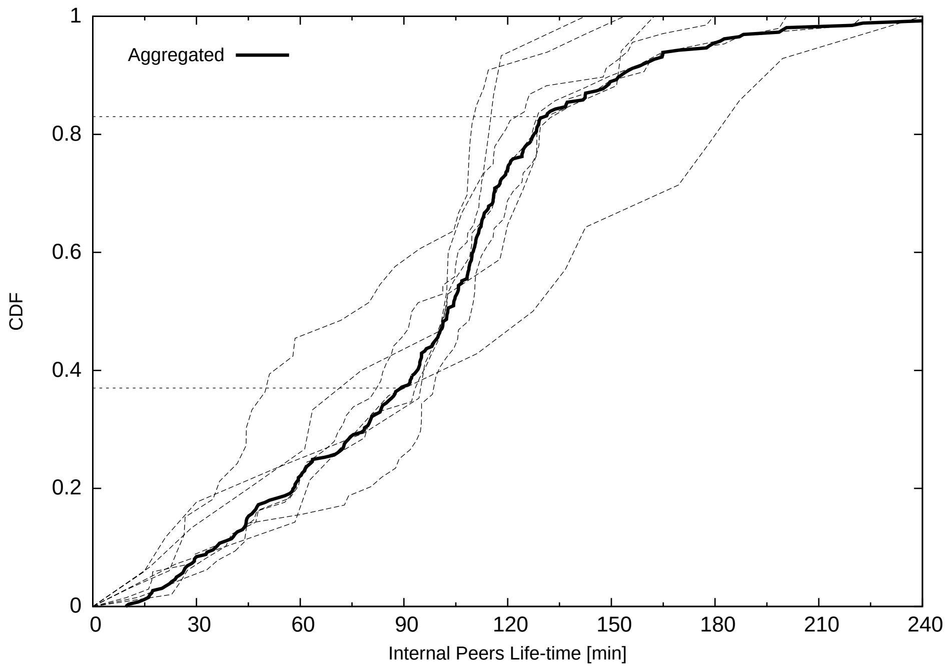
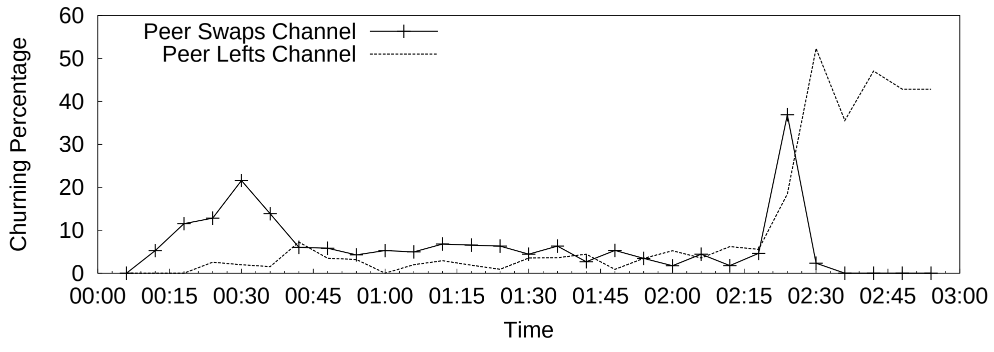


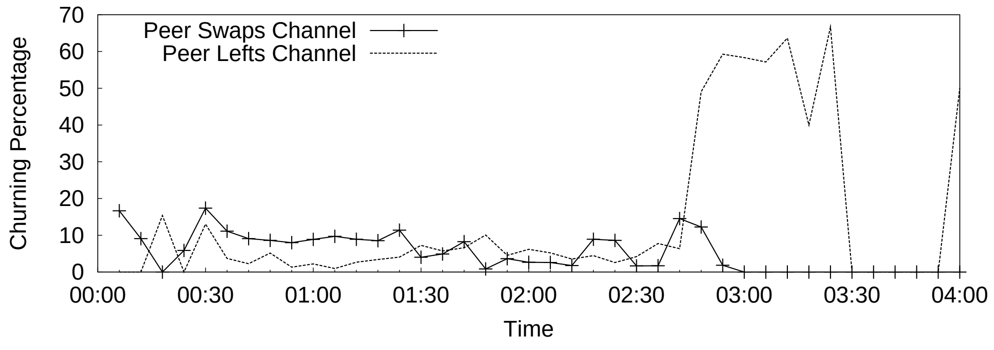
Figure 2.4: User lifetime CDF; the thick line refers to the aggregate statistic.

It has already addressed that some users switch channel during their lifetime, but how much often this happen during the lifetime of an event? To answer this question, Fig. 2.5 shows the churning percentage over time. Both peers that change channel and peers that close the application are considered. Two different events are considered in top and bottom plots. Results are obtained by computing the swarming matrix for different time slots. In detail, $M(t_n)$ is computed every time period t_n of 5 min; then $M(t_n)$ and $M(t_{n-1})$ are compared to count peers entering or leaving a channel and peers switching channel. Finally, the churning percentage is computed with respect to the number of active peers at time t_n.

Fig. 2.5 shows that the number of users switching channel is not negligible, e.g., around 8-10% of users changes channel every 5 min. Interestingly, the percentage of peers that change channel is higher at the beginning, when possibly users are seeking for a good channel to follow the whole event which they are interested in. The churning percentage of users leaving the system is small, but it suddenly increases at the event end when more than 50% of users leave the system at the same time.



(a) Event A



(b) Event B

Figure 2.5: Percentage of peers changing channel and leaving the system during two soccer matches.

These results highlight the "human-factor" implication when designing a P2P-TV system. Both flash crowd and sudden peer departures are not negligible, so that algorithms must explicitly deal with them.

2.4 Peer Discovery Process

In this section, is of interest to understand some properties regarding with the overlay graph underneath a P2P-TV channel. Most of P2P-TV systems (SopCast included) implement a peer discovery algorithm based on gossiping protocols [2]. In SopCast a continuous discovery process is carried out by peers that look for new peers at a practically constant rate. In the overlay graph, an edge is created every time a peer exchanges information with another peer. To assess what are the properties of the SopCast overlay graph, the peer discovery process is observed.

2.4.1 Peer Discovery by Access Technology

The first question that arise regarding to peer discovery is, do all peers discovery others with the same rate? To answer this question, the number of peers contacted by internal peers in a period of time ΔT is measured during the events. The discovery rate depends on the rate with which the peer is contacted by or contacts other peers. Fig. 2.6 shows an example of measurement for peers in swarm 14. The figure shows the average discovery rate of peers for TP and IT-FTTH data sets. It is chosen a $\Delta T = 60s$. Focusing first on the TP peers, all peers in this data set perform the same kind of discovery process. In contrast, the discovery rate of the IT-FTTH peer is much larger than the TP peer rate. It also shows a significant variation during peer lifetime. Similar results are observed when comparing low and high upload capacity peers: the formers exhibit smaller discovery rate than the latter. This suggests that SopCast implements some algorithms to exploit the upload bandwidth of high capacity peers.

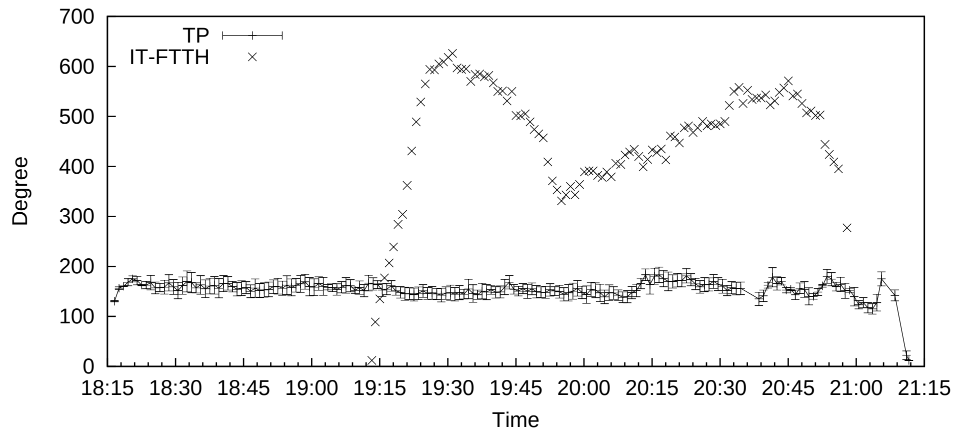


Figure 2.6: Number of peers contacted every 60s by internal peers.

To corroborate this hypothesis, a controlled test-bed experiment is set in the

University Campus network. The setup of the experiment consists on two peers watching the same channel at the same time. Both peers are connected to the Internet via the same router which limits the upload rate to 256kb/s and 32Mb/s, respectively. Note that the download rate should be limited by the video rate, in the case that protocol signaling overhead is not considered. The evolution over time of the discovering rate of the two peers is observed, showing identical results on Fig. 2.6: the peer with higher upload capacity exhibits a much higher peer discovery rate. Therefore, there is enough evidence to say that the discovery rate on SopCast depends on the peer upload capacity.

2.4.2 Bias of Peer Discovery Process

The scope of this section, is to unveil if there is any preference in the discovering process, based on the peer geo-location, distance or any other property. Fig. 2.7 shows the geographical breakdown of peers contacted by different internal peers during their whole lifetime. Left and right groups of bars refer to swarm 11 and 14, respectively. For each swarm, it is reported the breakdown for i) two internal peers selected at random from the TP dataset, ii) all peers in the TP dataset, and iii) one peer in the MT and IT-FTTH dataset. Results show that there is no statistically significant difference, even if peers are located in different countries (PL or HU for swarm 11, PL or IT for swarm 14) or are connected through different access technologies (PL-ADSL vs IT-FTTH for swarm 14). However, the breakdown changes in the two events. This suggests that the SopCast peer discovery mechanism is not driven by any preference related to any peer property, but it reflects only the natural distribution of users around the world. That is, the peer discovery mechanism follows a random process in which the probability of contacting (or being contacted) by a peer is independent from other peers. This is a very robust choice which allows SopCast to deal with the high churning rate seen in the previous section.

2.4.3 Channel Size Identification

Leveraging on the SopCast peer discovery process, in the following it is exploited a simple model to estimate the channel size, that is the number of peers actually watching the channel across the world. Let consider independent observation periods of duration ΔT. Over each period, the internal peers watching the same channel are monitored. The discovering process can be modeled as a random walk at constant rate defined by the access technology of each internal peer. Then each internal peer in the same network discover a random set of external peers every ΔT. Thus given any two internal peers a and b, the amount of common external peers $C(a, b)$ they discover follows a Poisson distribution with mean value determined by the degree

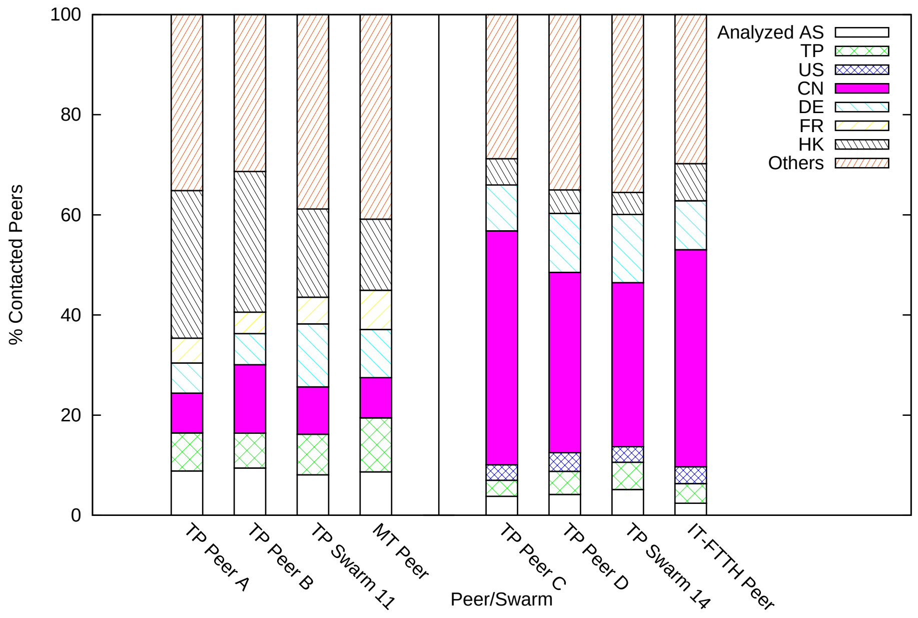


Figure 2.7: External peer spatial distribution.

$|P(a)|$ and $|P(b)|$ of a and b, as well the size N of the channel:

$$C(a,b) = \frac{|P(a)||P(b)|}{N} \tag{2.4}$$

Given a channel and an observation time ΔT, it is measured $C(a,b), P(a), P(b)$ for each possible pair of internal peers a, b. From (2.4) it is possible to estimate the total number of peers in the swarm; let this estimation be denoted by $\hat{N}(a,b)$.

$$\hat{N}(a,b) = \frac{C_a C_b}{C_{a,b}} \tag{2.5}$$

Since there are several internal peer pairs, it is computed the average and standard deviation among the estimations $\hat{N} = E[\hat{N}(a,b)]$ and $\sigma_N = std[\hat{N}(a,b)]$. $\hat{N}$ is the estimated swarm size, i.e., the number of vertexes in the overlay graph.

The sampling time ΔT plays a key role, since it defines the result of the discovery process: on the one hand, ΔT should be large enough to allow a correct estimation of $C(a,b)$; on the other hand, ΔT should be small to minimize the impact of peers churning and channel switching. Sensitivity analysis on the impact of ΔT throws that a good trade off is obtained at ΔT larger than 2 min and shorter than 10 min. In the following, it is choosen $\Delta T = 360$s. Fig. 2.8 reports the CDF of $C(a,b)$ for all the pairs a, b of internal peers in a swarm during a single observation period.

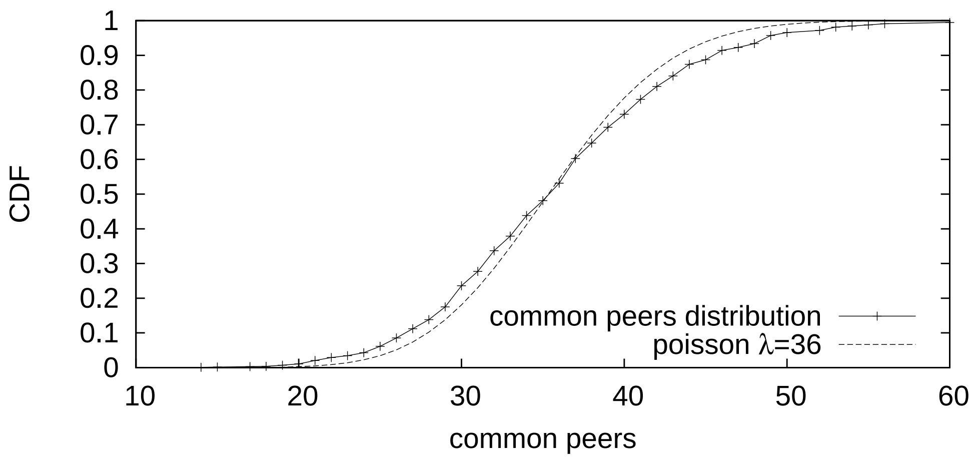


Figure 2.8: Common peer distribution as observed during a $\Delta T = 360$s time interval.

$C(a,b)$ closely follows a Poisson distribution with mean value 36, confirming that (2.4) offers a good approximation.

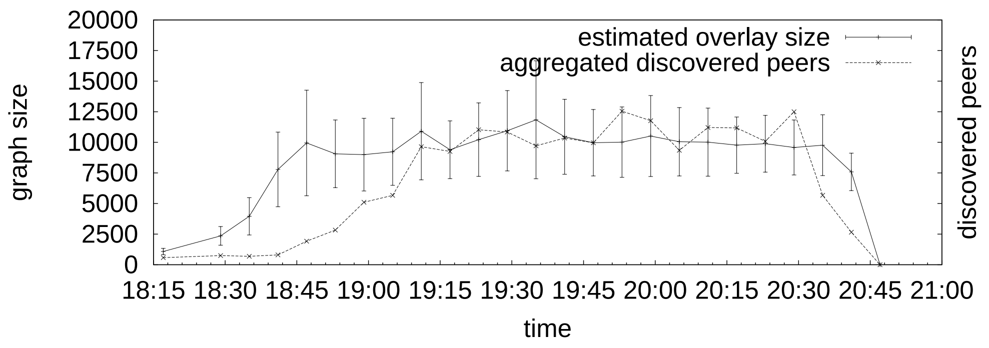


Figure 2.9: Estimated overlay graph size and total number of discovered peers. $\Delta T = 360$s.

Fig. 2.9 depicts the estimated average and standard deviation of the estimated channel size. It is contrasted with the amount of unique peers M discovered by all internal peers during the same observation time ΔT. This figure reports the evolution of swarm 6, which was previously reported in bottom plot of Fig. 2.1. $\hat{N}$ quickly grows at the beginning when the flash crowd phenomenon starts. During the event, $\hat{N}$ is then stable since the swarm population remains constant. Finally, at the event end an abrupt departure of peers is observed. This findings are coherent with results regarding the user habits on P2P-TV, which has been already noticed in Fig. 2.1. Comparing $\hat{N}$ with M, it is observe that the latter does not provide a good estimation of the swarm size during the initial transient. In regime situation,

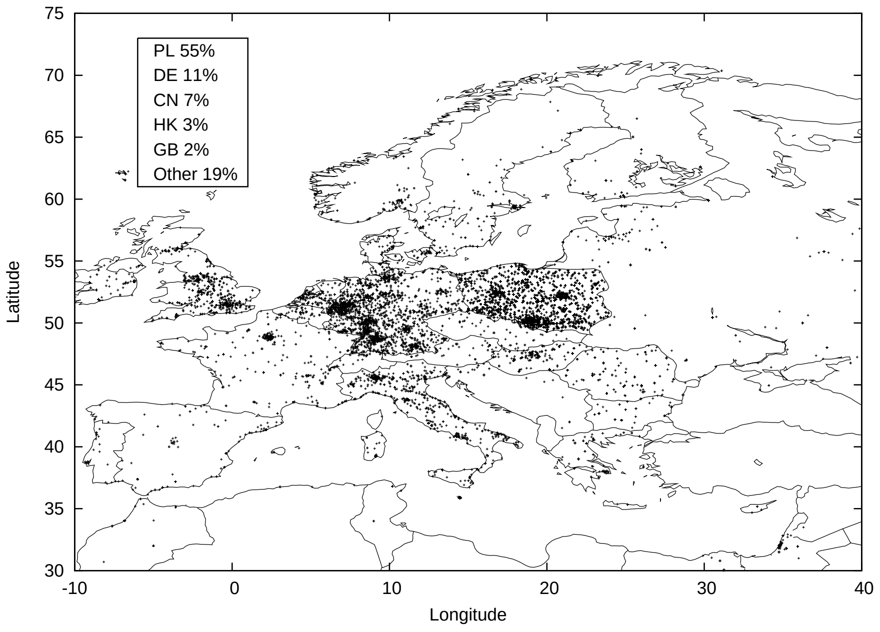
M is comparable with $\hat{N}$. Because more than 60 internal peers are present, the aggregated discovery process they perform allows to practically find all peers in the swarm in 360 s only. However, when the number of internal peers is small, M provides a lower bound to N.

2.5 Spatial Analysis

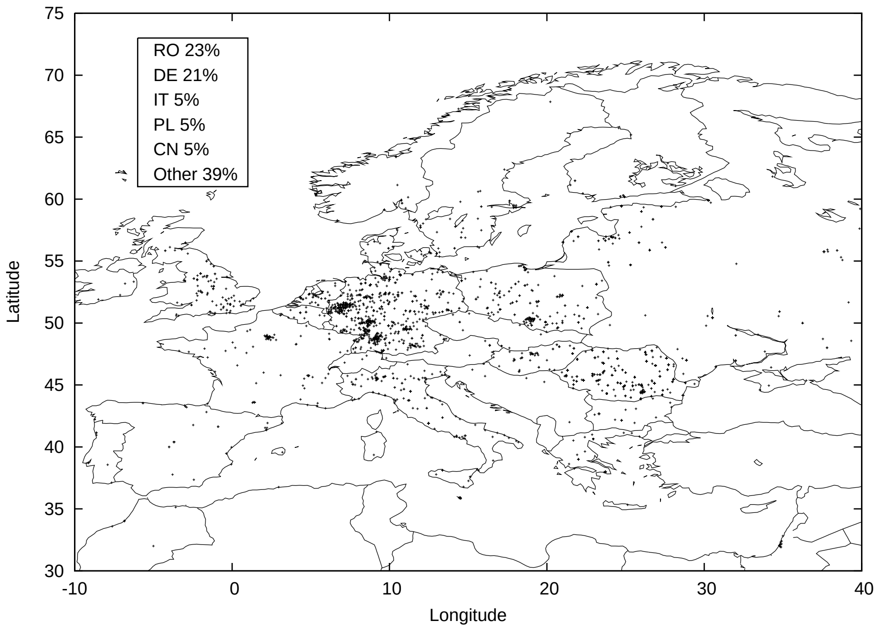
In this section the spatial characteristics of external peers are investigated, in order to discover whether there is any localization mechanism that drives peer discovery, selection process, or whether or not any cultural bias influences the P2P-TV overall distribution characteristics. Each observed external peer is geographically located using information provided by MaxMind GeoIP lite databases [3] which are coherent in date to the traces used. Note that even if MaxMind GeoIP database has been controversial because its accuracy, for the case of residential IP addresses this database remains accurate.

2.5.1 Peer Discovery by Channel

Fig. 2.10 shows the geographical location of contacted peers during a given event for two different ISPs. Note that the results depicted in the figures correspond to the traffic reported for the event, that is all the traffic from channels has been aggregated. Top plot refers to a trace collected in EU-CC1, while bottom plot refers to a trace collected in EU-CC2. As it can be observed, the countries of peers interested in the event are very different. For instance lot of peers are found in EU-CC1, Germany and U.K. in top plot, while bottom plot shows very few peers in EU-CC1 and U.K. Regardless subdivision of channels, from a global point of view seems that users from different countries will choose different channels.



(a) EU2-ADSL



(b) EU1-ADSL2

Figure 2.10: Contributing peers in Europe by ISP.

Now consider the case of different channels regardless the location of the vantage point. The goal is to understand how different is the spatial distribution of peers on different channels. Fig. 2.11 reports a breakdown of all external peers according to their origin country, identified by the Country Code. The set of countries has been selected among the most frequent ones. To easy readability, in this figure swarms are sorted according to the fraction of peers that belong to the same Polish Autonomous System (labeled "same AS") where the probe is located in.

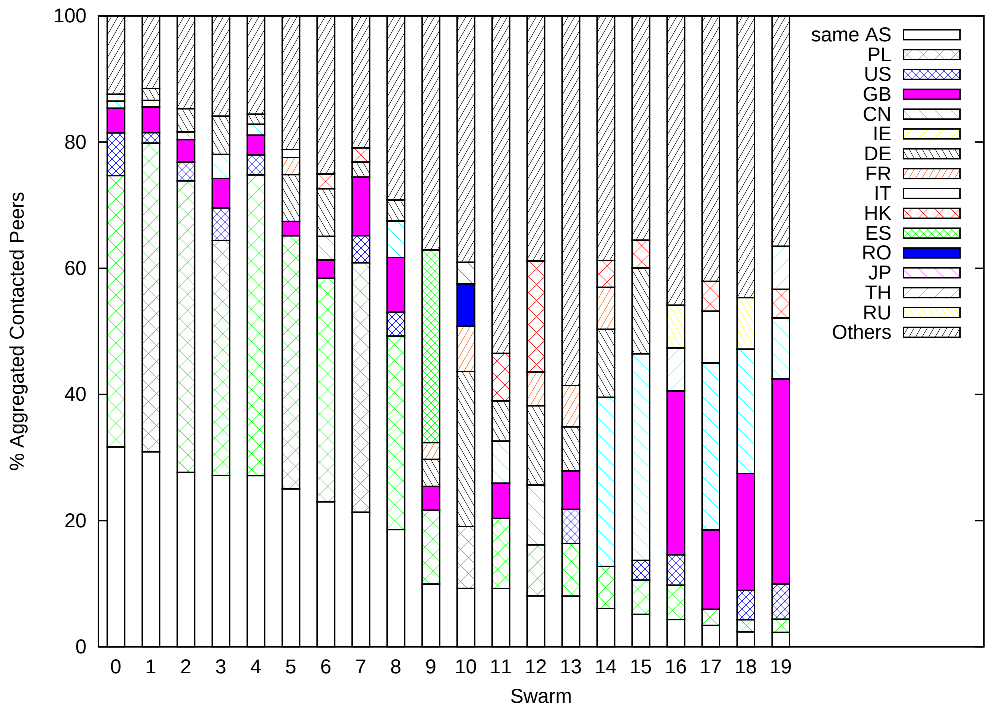


Figure 2.11: External peer spatial distribution by swarm.

Several considerations hold. First, the fraction of peers located in EU-CC1 is larger than 50% up to Swarm 8. Then, it suddenly drops to less than 20%. Note that the portion of peers in same AS with respect to the total Polish peers is very similar across all channels. This fact reflects the market share of the ISP that this probe was located in. Second, the fraction of peers found in other countries is variable. For instance, channel 9 has a large fraction of users in Spain (and indeed over this channel the broadcast content was a football match involving a Spanish team), while channels from 16 to 19 have a clear predominance of United Kingdom users (and they all correspond to "Premier League" events). his clearly shows that there is a bias in the external peers contacted during each event, but it is naturally induced by the actual distribution of users and not by the application; this, in its turn, is highly dominated by cultural and language traits.

2.5.2 Channel popularity

It was already found that some channels are dominated by the presence of some geographical location peers because of the content, but what makes local users to select one channel over another, when the content shown during the event on most channels is the same. Let start showing the evolution of internal peers among contemporary different channels observed in the same vantage point. Fig. 2.12 shows the peer evolution for three different contemporary channels in the EU2-ADSL probe. Channel 5 is much more popular in the monitored PoP, reaching 60 coexisting internal peers (left y-axis); channels 10 and 14 are much less popular (right y-axis). Nonetheless, the peers evolution is very similar, suggesting that users are watching different channels, but the same event. Recall indeed that SopCast (and P2P-TV systems in general) typically offers several channels that broadcast the same event.

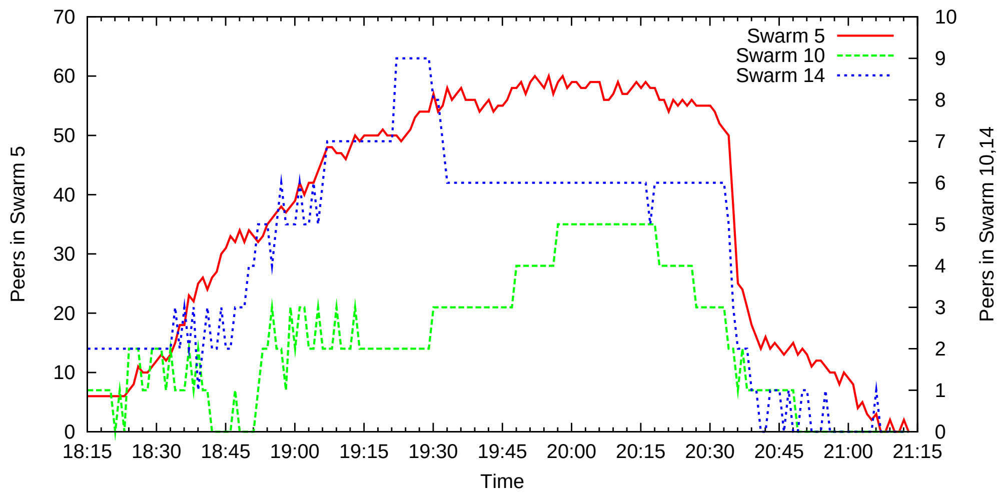


Figure 2.12: Peer evolution for three swarms present at the same time.

Up to now there are evidences from the channeling matrix and the peer evolution, that at the same moment during an event there are contemporary channels broadcasting the same. Now what makes users being more interested to one channel instead of another?. The hypothesis is that channel popularity is biased by cultural traits of a given country or community of people. Let P_l be "local channel popularity", i.e., the fraction of internal peers watching a specific channel over all internal peers that were alive during an event. Let P_c be "Polish channel popularity", i.e., the fraction of peers in Poland over the entire peer population that joined a channel. P_l is a measure of how popular is a channel in the vantage point (in Poland). P_c instead is a measure of how popular a channel is among Polish with respect to worldwide population, i.e. P_c measures how biased the peer distribution of a channel is towards Poland.

Fig. 2.13 shows P_c versus P_l for each channel. Interestingly there are two main clusters of points: channels which are locally popular (large P_l) and mostly popular in Poland (large P_c), and channels which are not locally popular (small P_l) but popular worldwide (small P_c). The first subset corresponds to channels that exhibit a high bias toward Polish interests, so that they are mainly popular in Poland. The second subset, on the contrary, corresponds to channels which are less interesting for Polish and more popular outside Poland. This hints that cultural and language traits affect the channel selection.

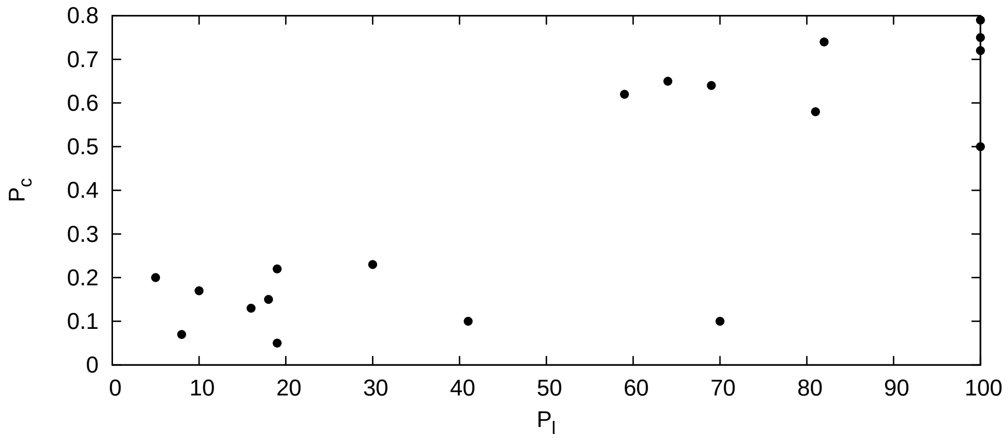


Figure 2.13: EU2-ADSL Popularity versus Country Popularity and Video Rate

However there is an exception to the rule. One channel shows high P_l but low P_c (local channel popularity of 70%, Polish channel popularity 10%). This particular channel results to be interesting for global audience, so that the fraction of Polish over all peers is not predominant, despite the large interest of Polish toward the content (indeed the event being broadcast at that time was the match of Liverpool v.s. Arsenal in April 21^{th} 2009).

2.6 Content retrieval

In this section is of interest the characterization of the content distribution graph, that is characterize, regardless how peers are connected, from where the traffic is coming or going. In this scope, it is investigated if there is any preference in choosing from which contributing peer, the video content is downloaded. First, it is investigated if any preference is given to contributing peers within the same Autonomous System, then it is investigated contribution at country level, finishing with peers in terms of their contributing capacity. This is a timely topic being investigated for both P2P and P2P-TV systems [4,5]; indeed, preference to peers in the same AS would allow ISP to reduce peering costs.

2.6.1 Preference to same AS

For each channel, the set of all external peers are partitioned according to the AS they belong to. By focusing on a particular AS, it is compared the fraction of external peers belonging to the AS with the fraction of data transmitted/received from internal peers. Any difference in these fractions is interpreted as an indication of some form of preference given to internal peers. This analysis is made over prominent ASes, that is ASes that contribute or receive more data from the internal peers.

The results on Fig. 2.14(a) are derived focusing on peers that belong to the same EU2-ADSL AS. Channels are sorted in decreasing order, considering as key value the fraction of external peers in the same AS, as in Tab. 2.1. Focusing on transmitted data: internal peers are likely to transmit a big portion of data to peers which are located in the same AS. For example, for channel 0, more than 50% of transmitted traffic goes to only the 32% of peers. Considering instead received data, it is observed the opposite: in channel 0, 32% of peers can only provide less than 18% of traffic to internal peers. Indeed, customers of the EU2-ADSL ISP are offered ADSL lines and have small upload capacity. Since low-capacity peers can only upload a fraction of the traffic they download, the rest of the traffic has to come from other ASes, in which high upload capacity peers are present.

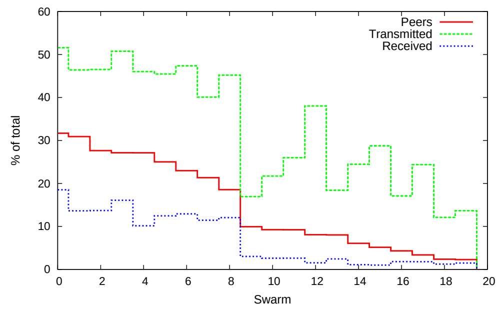
2.6.2 Preference to same CC

To quantify if there is a preference at the country level, another AS located in Poland is considered. For this experiment it is called PL2 AS, which also its customers are offered ADSL access lines. Results are reported in Fig. 2.14(b). Comparing the fraction of peers in PL2 and the fraction of traffic transmitted to PL2, it is noticed that this time the two curves are clearly similar. This shows that peers in EU2-ADSL send an amount of traffic to peers in PL2 which is proportional to the number of contacted PL2 peers, i.e., a peer in PL2 is selected with the same probability of peers in any other AS. Considering the amount of traffic received from PL2 peers, it is observed that it follows exactly the same trend seen on Fig. 2.14(a). This is expected, since PL2 customers have the same upload capacity as EU2-ADSL customers, so that they can provide only a fraction of the traffic. Repeating the analysis considering other countries ADSL providers, similar results are obtained. These results suggest that there is no preference based on the country the peers belong to.

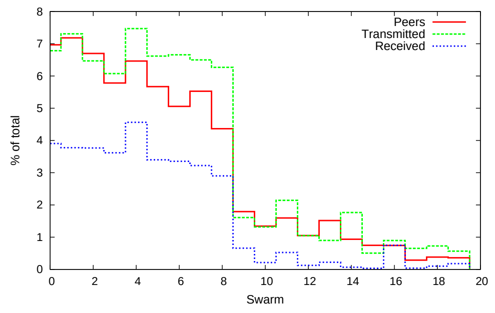
2.6.3 Preference to faster peers

Fig. 2.14(c) and Fig. 2.14(d) refer to results considering two ASes with a large fraction of high upload capacity peers. The first one is an educational Polish AS - PL3, the second one is a commercial AS in Russia - RU1. It is clear that when these high capacity peers are present, internal peers download from them a large fraction of traffic. For example, on channel 1, peers from PL3 contribute to more than 10% of downloaded traffic, despite they represent only 0.3% of external peers. For the channels on the left, i.e., popular channels in Poland, it is easy to find peers with high capacity in PL3 AS. On the contrary, for rightmost channels that are unpopular channels in Poland, traffic is received from any AS, provided that peers there have large upload capacity. This is the case of RU1.

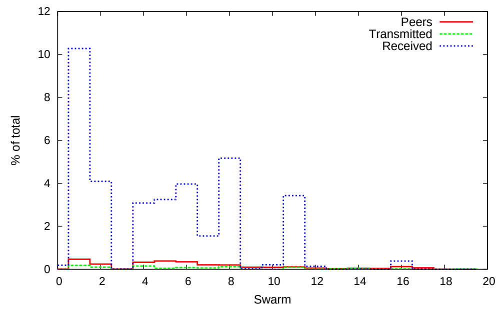
Thus, it is found that SopCast peers tend to transmit traffic to peers in the same AS they belong to. But if bandwidth availability in the AS is not sufficient, peers fetch the content from any high bandwidth peer, whatever AS it belongs to.



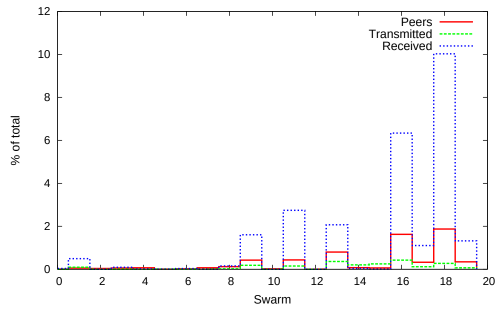
(a) Contribution of Same AS



(b) Contribution Same Country AS



(c) Contribution of a Same Country High Capacity AS



(d) Contribution of a Foreign High Capacity AS

Figure 2.14: EU2-ADSL Aggregated AS Contribution to Channels

As realized on Sec. 2.6.3, uplink capacity of peers is a key factor for content distribution over the P2P overlay. Moreover, it allows to understand the feasibility and scalability of P2P-TV services. Indeed, the total available upload capacity offered by peers should be equal or larger to the total download capacity dictated by the video rate and the number of peers in a channel. Today, ADSL peers lack of upload capacity to sustain the video content diffusion, and high upload capacity peers are required to supply it. What is then the access capacity of contributing peers in SopCast overlay? To the best of the knowledge this characterization has never been carried out.

To estimate path capacity between internal peers and external ones, it is used the so called *Probe Gap Model* (PGM) [6]. The intuition is to observe the minimum Inter-Packet Gap (IPG) of packets received by a given node, which, in case of packets sent back-to-back at the source, provides an estimation of the available capacity C along the path. Several assumptions must be verified to obtain reliable measurements: i) packets must be sent back-to-back by the transmitter; ii) measurement must be repeated several times to estimate the minimum IPG; iii) time-stamping at the receiver must be very accurate, and packets must not be artificially delayed at the receiver by some buffering. To this extent, large packets are preferred, so that the packet transmission time is large (a 40 Bytes long packet lasts only 320 ns on a 1 Gb/s link).

For each channel, an estimation of the upload capacity for all external contributing peers is obtained. Contributing peers that have transmitted at least 10 packets larger than 1300B are considered, otherwise the estimation is considered not valid. Such amount of data in a single data packet, suggest that those packets contain video or audio data. Particularly, these video or audio packets are sent in bursts by the transmitter. Given the preference already seen, in which the peers are prone to download content from high capacity peers, it is expected that the estimation can be performed only for a limited subset of contacted peers. Hence there is a bias to assess higher capacity peers. Fig. 2.15 reports the estimated capacity versus normalized peer ID; peers are ordered in decreasing estimated capacity. Log scale is used on the y-axis. The aggregate distribution is also shown. For all channels, it emerges that about 10% of peers have 1 Gb/s upload capacity, while [60,80]% of them have a capacity smaller than 2 Mb/s, with more than 50% of peers having less than 1 Mb/s capacity. This reflects the intuition that nowadays P2P-TV systems rely on high capacity peers, which act as amplifiers, and help in re-distributing the video stream to several peers. By trying to identify the heavy contributing peers, it is verified by hand that that the majority of them are actually peers at University campuses around the world.

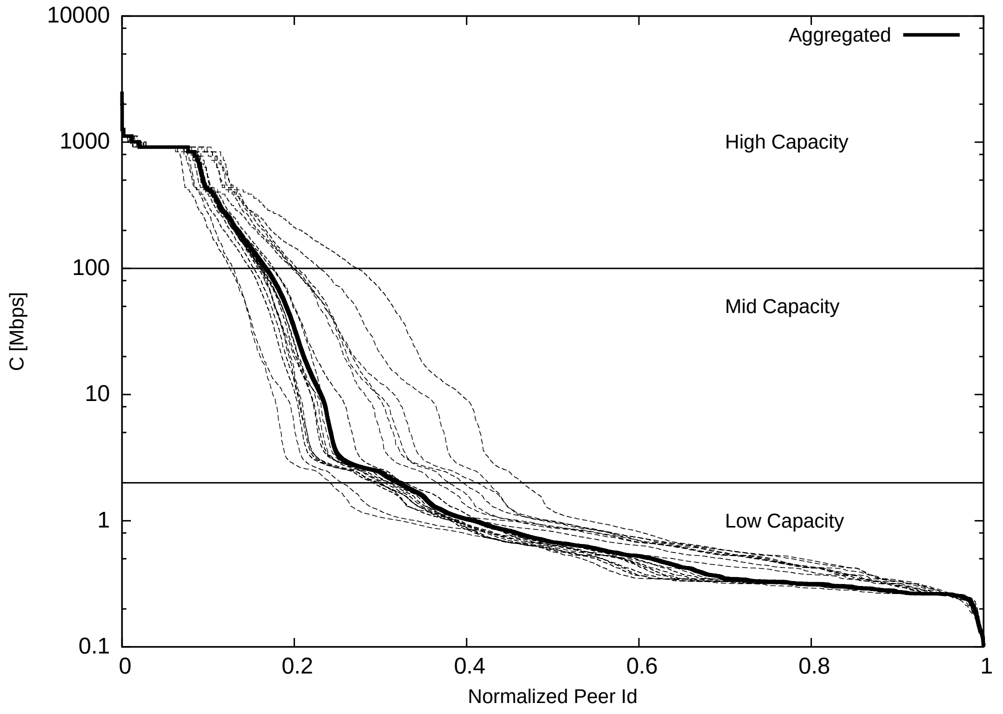


Figure 2.15: Contributing peer capacity for every swarm; the thickest line refers to the aggregate statistic.

2.7 Conclusions

This chapter covered an extensive analysis and characterization of P2P-TV traffic, as observed from the viewpoint of an ISP. Among all P2P-TV applications found in the wild, SopCast is the more popular among Internet end-users in the monitored in Europe, surpassing in traffic and audience other P2P-TV applications like PPLive and TVAnts at the time in which the analysis was performed. It is important to clarify that *Tstat*, the DPI monitoring tool used for classifying traffic in the network studied, was able to detect traffic for all these already mentioned P2P-TV application.

Therefore experiments shown results considering basically end-users habits when they watch TV using SopCast. ISP customers use it to watch sport events that are difficult or expensive to retrieve using traditional means. While the average traffic due to these applications is overall marginal, few tens of users generate 15% of traffic in the monitored PoP where more than 20,000 users are aggregated during popular events.

Then it was investigated SopCast swarm formation, focusing on peer discovery and data delivery processes. Findings indicate mainly two things. First, given the nature of P2P-TV, the distribution of peers, and, hence, traffic, has significant locality properties deriving from users' cultural and language traits. While this was already noticed in the case of file-sharing applications, for P2P-TV systems this phenomenon is even more visible. Second, data are mainly provided by high capacity peers and P2P-TV provision would not be feasible without their support, indeed ADSL connected peers can only contribute to about 1/5 of the required traffic.

This means that, even if some network aware peer selection mechanism is enforced to localize traffic to any subset of peers, its effectiveness might be limited by i) the already strong localization of peers enforced by users' preference and ii) the absence of sufficient capacity provided by the subset of peers. For example, ISPs providing ADSL lines to their users that desire strong locality of traffic, might need to deploy a few high capacity peers, acting as super peers, inside their network.

Chapter 3

Passively bringing visibility of content/service of network traffic

3.1 Introduction

In the past few years, the Internet has witnessed an explosion of cloud-based services and video streaming applications. In both cases, content delivery networks (CDN) and/or cloud computing services are used to meet both scalability and availability requirements. An undesirable side-effect of this is that it decouples the owner of the content and the organization serving it. For example, CNN or YouTube videos can be served by Akamai or Google CDN, and Farmville game can be accessed from Facebook while running on Amazon EC2 cloud computing platform, with static content being retrieved from a CDN. This may be even more complicated since various CDNs and content owners implement their own optimization mechanisms to ensure "spatial" and "temporal" diversity for load distribution. In addition, several popular sites like Twitter, Facebook, and Google have started adopting encryption (TLS/SSL) to deliver content to their users [7]. This trend is expected to gain more momentum in the next few years. While this helps to protect end-users' privacy, it can be a big impediment for effective security operations since network/security administrators now lack the required traffic visibility. The above factors have resulted in "tangled" world wide web which is hard to understand, discern, and control.

In the face of this tangled web, network/security administrators seek answers for several questions in order to manage their networks: (*i*) What are the various services/applications that contribute to the traffic mix on the network? (*ii*) How to block or provide certain Quality of Service (QoS) guarantees to select services?

While the above questions seem simple, the answers to these questions are non-trivial. There are no existing mechanisms that can provide comprehensive solutions to address the above issues. Consider the first question above. A typical approach currently used by network administrators is to rely on DPI (deep packet inspection) technology to identify traffic based on packet-content signatures. Although this approach is very effective in identifying unencrypted traffic, it severely falls short when the traffic is encrypted. Given the popularity of TLS in major application/content providers, this problem will amplify over time, thus rendering typical DPI technology for traffic visibility ineffective. A simple approach that can augment a DPI device to identify encrypted traffic is to inspect the certificate during the initial handshake, since during TLS negotiation, the server certificate contains a plain text string with the name being signed. Although this approach gives some visibility into the applications/services, it still cannot help in identifying specific services. For instance, inspecting a certificate from Google will only reveal that it is Google service, but cannot differentiate between Google Mail, Google Docs, Blogger, and Youtube. Thus administrators need a solution that will provide fine-grained traffic visibility even when the traffic is encrypted.

Focus now on the second question which is even more complex. Consider the scenario where the network administrator wants to block all traffic to Zynga games, but prioritize traffic for the DropBox service. Notice that both of these services are encrypted, thus severely impairing a DPI-based solution. Furthermore, both of these services use the Amazon EC2 cloud. In other words, the server IP-address for both of these services can be the same. Thus using IP-address filtering does not accomplish the task either. In addition the IP-address can change over time according to CDN optimization policies. Another approach that can be used in this context is to introduce certain policies directly into the local name servers. For example, the name server does not resolve the DNS query for zynga.com in the above example, thus blocking all traffic to Zynga. Although this approach can work effectively for blocking certain services, it does not help when administrators are interested in prioritizing traffic to certain services. Administrators face the same situation when they want to prioritize traffic to *mail.Google.com* and *docs.google.com*, while deprioritizing traffic *blogspot.com* and *youtube.com* since all of these services can run over HTTPS on the same Google platform.

This chapter describes DN-Hunter, a novel traffic monitoring system that addresses all of the above issues in a completely automated way. The main intuition behind DN-Hunter is to correlate the DNS queries and responses with the actual data flows in order to effectively *identify* and *label* the data flows, thus providing a very fine grained visibility of traffic on a network. It helps network administrators to keep track of the mapping between users, content owners, and the hosts serving the content even when this mapping is changing over time, thus enabling them to enforce policies on the traffic at any time with no manual intervention. In addition, network administrators could use DN-Hunter to dynamically reroute traffic in order to use more cost-effective links (or high bandwidth links as the policies might dictate) even as the content providers change the hosts serving the content over time for load balancing or other economic reasons.

At a high level, the methodology used in DN-Hunter seems to be achievable by performing a simple reverse DNS lookup using the server IP-addresses seen in traffic flows. However, using reverse DNS lookup does not help since it does not return accurate domain (or the sub-domain) names used in traffic flows.

Some key aspects highlighted in this chapter include:

• DN-Hunteris a novel tool, that can provide *fine-grained traffic visibility* to network administrators for effective policy controls and network management. Unlike DPI technology, using experiments on real traces, it is demonstrated that DN-Hunter is very effective even when the traffic is encrypted clearly highlighting its advantages when compared to the current approaches. DN-Hunter can be used either for active or passive monitoring, and can run either as a stand-alone tool or can easily be integrated into existing monitoring systems, depending on the final intent.

• A key property of DN-Hunter is its ability to identify traffic *even before the data flow starts*. In other words, the information extracted from the DNS responses can help a network management tool to *foresee* what kind of flows will traverse the network. This unique ability can empower proactive traffic management policies, e.g., prioritizing all TCP packets in a flow (including the critical three-way-handshake), not just those packets that follow a positive DPI match.

• DN-Hunter can be used not only for providing real-time traffic visibility and policy controls, but also for helping to gain better understanding of how the dynamic web is organized and evolving today. In other words, many other applications of DN-Hunter are presented throughout this chapter including: (*i*) *Spatial Discovery:* Mapping a particular content to the servers that actually deliver them at any point in time. the video service from Dailymotion uses Akamai CDN in Europe, but uses LimeLight in the US. (*ii*) *Content Discovery:* Mapping all the content delivered by different CDNs and cloud providers by aggregating the information based on server IP-addresses. (*iii*) *Service Tag Extraction:* Associating a layer-4 port number to the most popular service seen on the port with no a-priori information.

• Extensive experiments are conducted using five traffic traces collected from large ISPs in Europe and North America. The traces contain full packets including the application payload, and range from 3h to 24h. These ISPs use several different access technologies (ADSL, FTTH, and 3G/4G) to provide service to their customers, thus showing that DN-Hunter is effective in several different contexts. Furthermore, DN-Hunter has been implemented and currently deployed in three operative vantage points since March 2012.

Although DN-Hunter is a very effective tool in any network administrator arsenal to address issues that do not have a standard solution today, there are some limitations as well. First, the effectiveness of DN-Hunter depends on the visibility into the DNS traffic of the ISP/enterprise. In other words, DN-Hunter will be rendered useless if it does not have visibility into the DNS queries and responses along with the data flows from the end-users. Second, DN-Hunter does not help in providing visibility into applications/services that do not depend on DNS. For instance, some peer-to-peer applications are designed to work with just IP-addresses and DN-Hunter will be unable to label these flows. Third, automatic and smart algorithms must be devised to dig into the information exposed by DN-Hunter. In this chapter the reader will find some examples of how the information extracted from the DNS traffic can be used. The applications of DN-Hunter are not limited to the ones presented in this thesis, and other novel applications can leverage the information exposed by DN-Hunter.

The rest of this chapter is organized as follows: Sec. 3.2 introduces the datasets used through this chapter. In Sec. 3.4 describes the architecture and design details of DN-Hunter. Sec. 3.5 presents some of our advanced analytics modules while Sec. 3.6 provides extensive experimental results. Also, it is discussed correct dimensioning and deployment issues in Sec. 3.7.

3.2 Experimental Datasets

Table 3.1: Dataset description.

Trace	**Start [GMT]**	**Duration**	**Peak DNS Responses Rate**	**#Flows TCP**
US-3G	15:30	3h	7.5k/min	4M
EU2-ADSL	14:50	6h	22k/min	16M
EU1-ADSL1	8:00	24h	35k/min	38M
EU1-ADSL2	8:40	5h	12k/min	5M
EU1-FTTH	17:00	3h	3k/min	1M

As in the previous chapter, all the experiments and further analysis are based on datasets collected Points-of-Presence (PoP) of large ISPs where the end customers are connected to the Internet. In addition to the set of vantage points used in the last chapter, in this one it is included a dataset gathered in a cellular network in the US. Used datasets are reported in Tab. 3.1. Again, in all of these traces activities from several thousands of customers are monitored. These 5 datasets corresponds to full packet traces including the application payload without any packet losses. For the sake of brevity, Tab. 3.1 only reports the start time and trace duration, the peak hour DNS response rate, and the number of TCP flows that were tracked. All traces have been captured on sparse periods of 2011. The first dataset is a trace collected from a large North American 3G/4G mobile operator GGSN aggregating traffic from a citywide area. The second dataset originates from a European ISP

(EU2) which has about 10K customers connected via ADSL technology. The last three datasets correspond to traffic collected from different vantage points in the same European ISP (EU1). The vantage points are located in three different cities - two ADSL PoPs and one Fiber-To-The-Home (FTTH) access PoP.

Currently, DN-Hunter has been implemented in a commercial tool as well as in *Tstat* [8]. The latter has been deployed in all the three vantage points in EU1 and has been successfully labeling flows since March 2012. Some of the results in this paper are derived from this deployment.

3.3 DNS Terminology

DNS is a hierarchical distributed naming system for computers connected to the Internet. It translates "domain names" that are meaningful to humans into IP-addresses required for routing. A DNS name server stores the DNS records for different domain names.

A domain name consists of two or more "labels" that are conventionally concatenated, and delimited by dots, e.g., *www.example.com*. These names provide meaningful information to the end user. Therefore labels naturally convey information about the service, content, and information offered by a given domain name. The labels in the domain name are organized in a hierarchical fashion. The Top-Level Domain (TLD) is the last part of the domain name - *.com* in the above example; and sub-domains are then pre-pended to the TLD. Thus, *example.com* is a subdomain of *.com*, and *www.example.com* is a subdomain of *example.com*. From here and onwards it is referred to the first sub-domain after the TLD as "second level domain"; it generally refers to the organization that owns the domain name (e.g., *example.com*). Finally *Fully Qualified Domain Name* (FQDN) is the domain name complete with all the labels that unambiguously identifies a resource, e.g., *www.example.com*.

When an application needs to access a resource, a query is sent to the local DNS server. This server responds back with the resolution if it already has one, else it invokes an iterative address resolution mechanism until it can resolve the domain name (or determine that it cannot be resolved). The responses from the DNS server carry a *list of answers*, i.e., a list of $serverIP$ addresses that can serve the content for the requested resource.

Local caching of DNS responses at the end-hosts is commonly used to avoid initiating new requests to the DNS server for every resolution. The time for which a local cache stores a DNS record is determined by the Time-To-Live (TTL) value associated with every record. It is set by the authoritative DNS name server, and varies from few seconds (e.g., for CDNs and highly dynamic services) to days. Also, memory limits and timeout deletion policies can affect local caching at the client OS.

However, as we will see later, in practice, clients cache DNS responses for typically less than 1 hour.

3.4 DN-Hunter Architecture

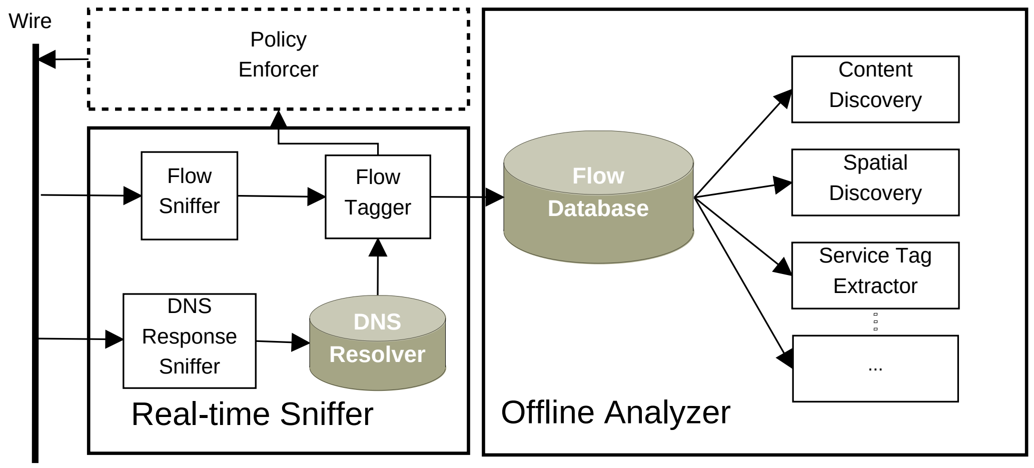


Figure 3.1: DN-Hunter architecture overview

A high level overview of DN-Hunter architecture is shown in Fig. 3.1. It consists of two main components: *real-time sniffer* and *off-line analyzer*. As the name indicates, the sniffer labels/tags all the incoming data flows in *real time*. The output from the sniffer can be used for online policy enforcement (using any available policy enforcing tool) and/or can be stored in a database for off-line analysis by the analyzer component. Note that the sniffer can be a passive component instead of being active if the policy enforcer is not implemented. During this, DN-Hunter will be treated as a completely passive real-time domain sniffer, instead of being a policy enforcer.

3.4.1 Real-Time Sniffer Component

The sniffer has two low-level sniffing blocks: (*i*) *Flow sniffer* which reconstructs layer-4 flows by aggregating packets based on the 5-tuple $Fid = (clientIP, serverIP, sPort, dPort, protocol)$, and (*ii*) *DNS response sniffer* which decodes the DNS responses, and maintains a local data structure called the *DNS Resolver*. The DNS resolver maintains a mapping between client IP, domain names queried, and the server IP(s) included in the DNS response. In particular, for each response, it stores the set of $serverIP$ addresses returned for the fully qualified domain name (FQDN) queried, associating them to the $clientIP$ that generated the query.

All data flows reconstructed by the flow sniffer is passed on to the *Flow Tagger* module. The flow tagger module queries the DNS resolver to tag the incoming $clientIP, serverIP$ pair. The flow tagger will tag the incoming flow with the "label" (i.e., the FQDN) and sends the flow to the policy enforcer (to enforce any policy on the flow including blocking, redirection, rate limiting, etc.) and/or the database for off-line analysis.

DNS Resolver Design

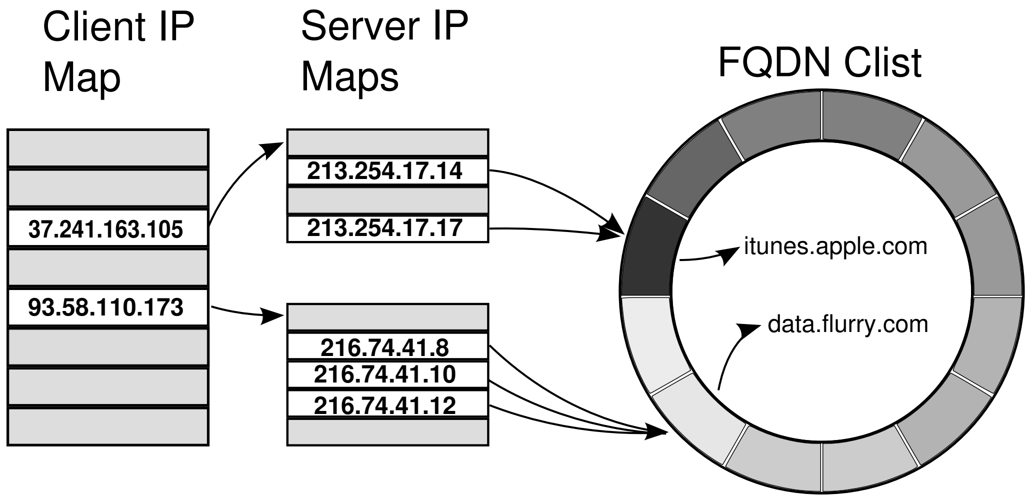


Figure 3.2: DNS Resolver data structures

The key block in the real-time sniffer component is the DNS Resolver. Its engineering is not trivial since it has to meet real-time constraints. The goal of the DNS Resolver is to build a replica of the client DNS cache by sniffing DNS responses from the DNS server. Each entry in the cache stores the $FQDN$ and uses the $serverIP$ and $clientIP$ as look-up keys. To avoid garbage collection, $FQDN$s are stored in a first-in-first-out FIFO circular list, *Clist*, of size L; a pointer identifies the next available location where an entry can be inserted. L limits the cache entry lifetime and has to properly match the local resolver cache in the monitored hosts.

Lookup is performed using two sets of tables. The first table uses the $clientIP$ as key to find a second table, from where the $serverIP$ key points to the most recent $FQDN$ entries in the *Clist* that was queried by $clientIP$. Tables are implemented using C++ maps[1], in which the elements are sorted from lower to higher key value

[1]Unordered maps, i.e., hash tables, can be used as well to further reduce the computational costs

following a specific strict weak ordering criterion based on IP addresses. Let N_C and $N_S(c)$ represent the number of monitored clients and the number of servers that client c contacts, respectively. Assuming L is well-dimensioned, the look-up complexity is $O(\log(N_C) + \log(N_S(c)))$. N_C depends on the number of hosts in the monitored network. $N_S(c)$ depends on the traffic generated by clients. In general, $N_S(c)$ is in the order of a few hundreds. Note that when the number of monitored clients increase, several load balancing strategies can be used. For instance, two resolvers can be maintained for odd and even fourth octet value in the client IP-address.

Fig. 3.2 depicts the internal data structures in the DNS resolver. Alg. 1 provides the pseudo code of the "insert()" and "lookup()" functions that access the data structures in Fig. 3.2. Since DNS responses carry a list of possible $serverIP$ addresses, more than one $serverIP$ can point to the same $FQDN$ entry (line 11-22). When a new DNS response is observed, the information is inserted in the $Clist$, eventually removing old entries (line 12-15). In the actual implementation of DN-Hunter, the information about the old FQDN is lost and may create some ambiguity. Then, when an entry in the DNS circular array is overwritten, the old $clientIP$ and $serverIP$ keys are removed from the maps before inserting the new one (line 25).

Despite the actual scenario of DN-Hunter, it could be devised to have some ambiguity under the scenario of a pair of IP addresses associated to several names, i.e. in cases in which a monitored IP address, contemporary, is connecting to virtual servers running behind the same IP address. For further details regarding characteristics of DNS on the wire see Sec. 3.7 for more details.

DNS Traffic Characteristics

Using the Alg. 1 for tagging (or labeling) incoming data flows, some experiments are conducted in order to accomplish the following goals: (i) Understand how much information DNS traffic can expose in enabling traffic visibility, and (ii) Understand how to correctly dimension the DNS resolver data structures.

To address the first goal, it is computed the DNS hit ratio. In other words, DNS hit ratio represents the fraction of data flows that can be successfully associated with a FQDN. The higher is the hit ratio, the more successful is DN-Hunter in enabling traffic visibility. Intuition suggests that all client-server services/applications rely on the DNS infrastructure and hence DN-Hunter will be able to accurately identify them. However, certain peer-to-peer services/applications do not use the DNS infrastructure and thus are not detected by DN-Hunter. Some statistics shown in Tab. 3.2 confirm this intuition. This table reports for each trace the number of DNS hits and the corresponding percentage of flows that were resolved, considering different classes of traffic such as HTTP, TLS, and P2P traffic (classification of traffic is done by *Tstat* DPI). Particularly, for this experiment is considered a warm-up time

Algorithm 1 DNS Resolver pseudo-code

```
 1: **INSERT**(DNSresponse)
 2: *Input*: DNSresponse
 3: $(FQDN, ClientIP, answerList) \leftarrow decode(DNSresponse)$
 4: $DNEntry \leftarrow newDNEntry(FQDN)$
 5: $mapServer \leftarrow mapClient.get(clientIP)$
 6: if $mapServer = null$ then
 7:    $mapServer \leftarrow$ new $MapServer()$
 8:    $mapClient.put(clientIP, mapServer)$
 9: end if
10: for all $serverIP$ in $answerList$ do
11:    /* replace old references */
12:    if exists $mapSever.get(serverIP)$ then
13:       $OLDEntry \leftarrow mapSever.get(serverIP)$
14:       $OLDEntry.removeOldReferences()$
15:    end if
16:    /* Link back and forth
17:    references to the new DNSEntry */
18:    $mapServer.put(serverIP, DNEntry)$
19:    $MSEntry \leftarrow mapServer.get(serverIP)$
20:    $DNEntry.insert(MSEntry)$
21: end for
22: /* insert next entry in circular array */
23: $OldDNEntry \leftarrow Clist.nextEntry()$
24: $OldDNEntry.deleteBackreferences()$
25: $Clist.nextEntry \leftarrow DNEntry$
26:
27: LOOKUP(ClientIP, ServerIP)
28: Input: $ClientIP$ and $ServerIP$ of a flow
29: Output: $FQDN$ of $ServerIP$ as requested by $ClientIP$
30: $mapServer \leftarrow mapClient.get(clientIP)$
31: if $mapServer$ contains $serverIP$ then
32:    $DNEntry \leftarrow mapServer.get(serverIP)$
33:    return $DNEntry.FQDN$
34: end if
```

of 5 minutes, i.e., all flows are tracked, but the statistics contributed by the flows in the first 5 mins of the trace are ignored. The reason of doing such a warm-up, is because for some of the first flows their associated FQDN request, could take place before starting the monitoring, and then impairing the quality of the statistics.

As expected, HTTP and TLS flows show a very high hit ratio, with the majority of cache-miss occurring in the initial part of the trace when the end host operating system local resolver cache resolves the query locally and limits the queries to the DNS server. P2P data flows are hardly preceded by DNS resolutions, and hence it results in a very low hit ratio. In this case P2P hits are related to BitTorrent tracker traffic classified in this way by *Tstat*.

When considering only HTTP and TLS data flows, the hit ratio mostly exceeds 90% for all traces except US-3G. When considering only the last hour of each trace, the DNS hit ratio increases further close to 100% in all traces but US-3G. For the US-3G case, it has been verified a particular configuration when traces were obtained. This configuration involved a load balancer through more than one interface, therefore traffic has been gathered from not all of the interfaces, from which, traffic has been split. Nevertheless the configuration, on device mobility scenario may also affect these results, e.g. when DN-Hunter observes flows from devices entering the coverage area after performing a DNS resolution outside the visibility of our monitoring point. Thus DN-Hunter might miss the DNS response resulting in a cache-miss. More details about the DNS traffic characteristics that affects DN-Hunter dimensioning is provided in Sec. 3.7.

Table 3.2: DNS Resolver hit ratio

Protocol	**EU1-ADSL1**	**EU1-ADSL2**	**EU1-FTTH**
HTTP	92% (4.4M)	90% (2.7M)	91% (683k)
TLS	92% (0.4M)	86% (196k)	84% (50k)
P2P	1% (6k)	1% (1.3k)	0% (48)
	EU2-ADSL	**US-3G**	
HTTP	97% (5.8M)	75% (445k)	
TLS	96% (279k)	74% (83k)	
P2P	1% (4.2k)	8% (8k)	

DN-Hunter vs. DNS Reverse Lookup

The information that the sniffer component extracts is much more valuable than the one that can be obtained by performing active DNS reverse lookup of $serverIP$ addresses. Recall that the reverse lookup returns only the designated domain name record, which is intended to be used for administrative purposes mainly. Consider Tab. 3.3 and EU1-ADSL2 dataset, 1,000 $serverIP$ have been randomly selected from the set of IP addresses already associated to a FQDN. An active DNS reverse lookup iteration over the randomly selected *serverIP* is performed and results are compared with the returned FQDN that DN-Hunter was able to provide. In 29% of cases, no answer was returned by the reverse lookup while in 26% of the lookups the two answers were totally different from each other. All the other queries had at least had a partial match. In fact, only 9% of the reverse lookups completely matched the results from the sniffer while the rest of the 36% only matched the second-level domain name. These results are not surprising since single servers are typically serving several FQDNs (see Sec. 3.6). In addition to this, reverse lookup poses scalability issues as well, since it is much more efficient to use information already generated by users, than querying a DNS server each time a flow is monitored.

Table 3.3: DN-Hunter vs. reverse lookup

Same FQDN	9%
Same 2nd-level domain	36%
Totally different	26%
No-answer	29%

3.4.2 Off-Line Analyzer Component

Although the sniffer module provides deep visibility into the services/applications on the wire in real-time, some analytics cannot be performed in real-time. In other words, dissecting and analyzing the data in different ways can expose very interesting insights about the traffic. The off-line analyzer component does exactly this. It contains several intelligent analytics that can extract information from the flows database by mining its content. Sample of possible analytics are presented in Sec. 3.5, then real examples applying these implemented methodologies are shown in Sec. 3.6.

3.5 Advanced Analytics

In this section some advanced analytics are described, using information stored in the labeled flows database to automatically discover interesting information and discern the tangled web.

3.5.1 Spatial Discovery of Servers

Today, CDNs and distributed cloud-based infrastructures are used to meet both scalability and reliability requirements, decoupling the owner of the content and the organization serving it. In this context some interesting questions arise: (*i*) Given a particular resource (i.e., a FQDN) what are all the servers or hosts that deliver the required content?, (*ii*) Do these servers belong to the same or different CDNs?, and (*iii*) Do CDNs catering to the resource change over time and geography? (*iv*) Are other resources belonging to the same organization served by the same or different set of CDNs?

DN-Hunter can easily answer all of the above questions. Alg. 2 shows the pseudo-code for the Spatial Discovery functionality in DN-Hunter. The spatial discovery module first extracts the second-level domain name from the FQDN (line 4), and then queries the labeled flows database (line 5) to retrieve all $serverIP$ addresses in flows directed towards the second-level domain (i.e., the organization). Then, for every FQDN that belongs to the organization, the spatial discovery module will extract the $serverIP$ addresses that can serve the request (line 6-9) based on the DNS responses. This enables the module to: (*i*) Discover the information about the structure of servers (single server, or one/many CDNs) that handle all queries for the organization, (*ii*) Discover which servers handle a more specific resource. For example, different data centres/hosts may be serving the content for *mail.Google.com* and *scholar.google.com*, and (*iii*) Automatically keep track of any changes in $serverIP$ addresses that satisfy a given FQDN over time. Note that the ability of DN-Hunter to easily track temporal and spatial changes in the FQDN-$serverIP$ address mapping also enables some basic anomaly detection. While out of scope of this paper, consider the case of DNS cache poisoning where a response for certain FQDN suddenly changes and is different from what was seen by DN-Hunter in the the past. This scenario can be flagged as an anomaly, enabling the security operator to take some action if required.

Algorithm 2 Spatial Discovery Analytics Algorithm

```
1: SPATIAL DISCOVERY(FQDN)
2: Input: The targeted FQDN
3: Output: ranked list of $serverIP$ addresses
4: $2ndDomain \leftarrow FQDN.split()$
5: $ServerSet \leftarrow FlowDB.queryByDomainName(2ndDomain)$
6: $FQDNset \leftarrow 2ndDomain.query()$
7: for all $FQDN$ in $FQDNSet$ do
8:     $FQDN.ServerSet \leftarrow FlowDB.queryByDomainName(FQDN)$
9: end for
10: Return($FQDN.ServerSet.sort(), ServerSet.sort()$)
```

3.5.2 Content Discovery

As previously observed in Sec. 3.5.1, a particular resource can be served by one or more CDNs or cloud infrastructures, and the spatial discovery analytics module provides deep insights into this. However, it is also important to understand tangle from another perspective. In other words, we need to answer the following questions: (*i*) Given a particular CDN what are the different resources that they host/serve? (*ii*) What is the popularity of particular CDNs in different geographies? (*iii*) Given two CDNs, what are the common resources that they both host?, and (*iv*) Does a given CDN focus on hosting content for certain types of services (like real-time multimedia streaming, mail, etc.)?

Once again DN-Hunter can answer the above questions easily based on the mapping stored in the flows database and using the whois database to associate IP addresses to CDNs. The complete algorithm for the content discovery module is shown in Alg. 3. The algorithm takes a $ServerIPSet$, i.e., the set of $serverIP$ addresses belonging to one or more CDNs, and extracts all the FQDNs associated with them (line 4-7). Depending on the desired granularity level, either the complete FQDN or only part of the FQDN (say, the second-level domain) can be considered. If only the second-level domains are considered, then the algorithm will return all the organizations served by the set of $serverIP$ addresses provided as input. However, if only service tokens are used (we will discuss this in the next sub-section), then the algorithm will return which popular services are hosted by the input $serverIP$ addresses.

Algorithm 3 Content Discovery Analytics Algorithm

```
 1: CONTENT DISCOVERY(ServerIPSet)
 2: Input: The list of targeted $serverIP$
 3: Output: The list of handled FQDNs
 4: $DomainNameSet \leftarrow FlowDB.query(ServerIPSet)$
 5: for all $FQDN$ in $DomainNameSet$ do
 6:     $TokenSet \leftarrow DomainName.split(FQDN)$
 7: end for
 8: for all $Token$ in $TokenSet$ do
 9:     $Token.score.update()$
10: end for
11: Return($Tokens.sort()$)
```

3.5.3 Automatic Service Tag Extraction

Identifying all the services/applications running on a particular layer-4 port number is a legacy problem that network administrators encounter. Even today there are no existing solutions that can identify all applications on any given layer-4 port number. In fact, the network administrators depend on DPI solutions to address this problem. DPI technology can only provide a partial solution to this problem due to two reasons: (1) Several services/applications use encryption and hence bypass DPIs, and (2) DPI devices can only identify those services/applications for which they already have a signature, thus severely limiting the coverage.

DN-Hunter provides a simple and automated way to address the above issue. The algorithm for extracting service tags on any layer-4 port number is shown in Algorithm 4. The input to the algorithm are the target port number and the k value for the top-k services to be identified. The algorithm first retrieves all FQDNs associated to flows that are directed to $dPort$ (line 4). Each FQDN is then tokenized to extract all the sub-domains except for the TLD and second-level domain. The tokens are further split by considering non-alphanumeric characters as separators. Numbers are replaced by a generic N character (lines 5-7). For instance, *smtp2.mail.Google.com* generates the list of tokens {smtpN, mail}.

The frequency (support) of tokens is used as a metric of "relevance" of the token for the targeted port (lines 8-10). To mitigate the bias due to some clients generating a lot of connections to a FQDN having the same token X, it is used a logarithmic score. Mathematically, let $N_X(c)$ be the number of flows originated by $clientIP\ c$ having the token X. Then the score of X is:

$$score(X) = \sum_c \log(N_X(c) + 1) \tag{3.1}$$

Tokens are then ranked by score and the top-k tokens are returned to the users (line 11). Depending on the final goal, different criteria can be applied to limit the list of returned tokens. For instance, the list can simply be limited to the top 5%, or to the subset that sums to the n-th percentile. Typically, the score distribution is very skewed, as it is shown in Sec. 3.6.

Algorithm 4 Service Tag Extraction Analytics Algorithm

```
 1: TAG EXTRACTION(dPort, $k$)
 2: Input: targeted $dPort$, $k$ of tags to return
 3: Output: The ranked list of tags
 4: $DomainNameSet \leftarrow FlowDB.query(dPort)$
 5: for all $FQDN$ in $DomainNameSet$ do
 6:     $TokenSet \leftarrow DomainName.split(NoTLD|NoSLD)$
 7: end for
 8: for all $Token$ in $TokenSet$ do
 9:     $Token.score.update()$
10: end for
11: Return($Tokens.sort(k)$)
```

3.6 Experimental Results

Previously methodologies on Sec. 3.5 intelligent analytics have been presented for leveraging better information from FQDN. Now, this section presents results obtained applying the analytics using DN-Hunter on the traces mentioned in Sec. 3.2. The discussion is started by showing evidence of how tangled is the web today in terms of content, providers, and hosts serving what end-users consume. Then some results are selected, to show the current advantages of using DN-Hunter in an operational network compared to the existing solutions for traffic visibility and policy enforcement. As a fact, DN-Hunter is currently implemented as module of two different DPI tools, providing more traffic visibility to network operators. In the second half of this section,results from our advanced analytics modules are presented, demonstrating in this way the wide applicability and usefulness of DN-Hunter.

3.6.1 The Tangled Web

The basic hypothesis of this chapter is that the web today is intertwined with content, content providers, and hosts serving the content that are continually changing over time and space. Hence we need a methodology that can assist in restoring clarity to operators regarding their network traffic. The top plot of Fig. 3.3 reports, for each FQDN, the overall number of $serverIP$ addresses that serve it. In the bottom plot of Fig. 3.3 we show the opposite - the number of different FQDNs a single $serverIP$ address serves. Fig. 3.3 was generated using the EU2-ADSL dataset, however, all the other datasets produced very similar result. From the figures one can see that one single $serverIP$ is associated to a single FQDN for 73% of $serverIP$s, and 82% of FQDNs map to just one $serverIP$. But more important to note is that there are FQDNs that are served by hundreds of different $serverIP$ addresses. Similarly a large number of FQDNs are served by one $serverIP$. Notice the x-axis in this figure is presented in log scale.

Just looking at the one-to-many mapping between FQDN and $serverIP$ addresses reveals only a small part of the complexity. But what if time is added into the mix?. Fig. 3.4 shows the number of $serverIP$ addresses that have been observed responding to some selected well-known second-level domains. Here, time bins of 10min are considered, covering a 24h period from EU1-ADSL2 dataset. For some of the domains (like fbcdn.net and youtube.com) a clearly diurnal pattern is spotted, with more $serverIP$s being used during late evening as compared to early morning. Indeed, for *youtube.com* there is a big and sudden jump in the number of $serverIP$s between 17:00 and 20:30. This reflects changes in the YouTube policies, triggered by the peak-time load. The domain *fbcdn.net* (owned by Akamai and serving Facebook static content) shows similar characteristics with more than 600 different $serverIP$ addresses serving content in every 10min interval between 18:00

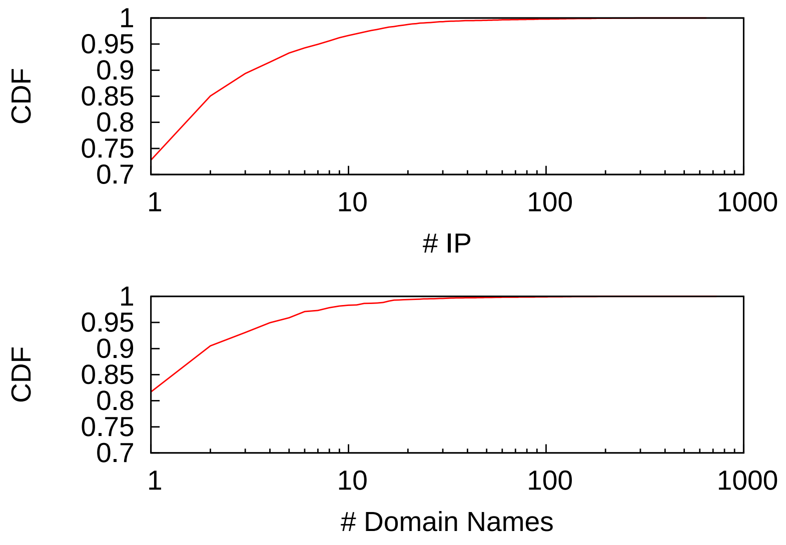


Figure 3.3: Number of $serverIP$ addresses associated to a FQDN (top) and number of FQDN associated to a $ServerIP$ (bottom). EU2-ADSL.

and 20:00. Finally, some of the other domains like *blogspot.com* (aggregating more than 4,500 total FQDN) are served by less than 20 $serverIP$s even during peak traffic hours.

Fig. 3.5 reports the number of different FQDNs that were served every 10min by different CDNs and cloud providers over a period of 24h. The MaxMind organization database is used to associate $serverIP$ addresses to organization (WHOIS database could be used as well, since both databases provide coherent information at organization level). First, it is observed that Amazon serves more than 600 distinct FQDN in every 10 min interval during peak hours (11:00 to 21:00). Totally Amazon served 7995 FQDNs during a day. While Akamai and Microsoft also serve significant number of FQDNs during peak hours, other CDNs like EdgeCast are more specific or less popular, serving less than 20 FQDNs a day.

Another aspect worth noting here is that association between FQDNs and CDNs change over time and space (i.e., geography). However, all of the above results clearly show why it is very hard to discern and control the traffic in today's networks. In fact, there is clear need for a solution like DN-Hunter that can track these changes seamlessly to ensure traffic visibility at any point in time. Surprisingly, the results presented in this section for motivating the need for a solution like DN-Hunter could not have been produced without DN-Hunter.

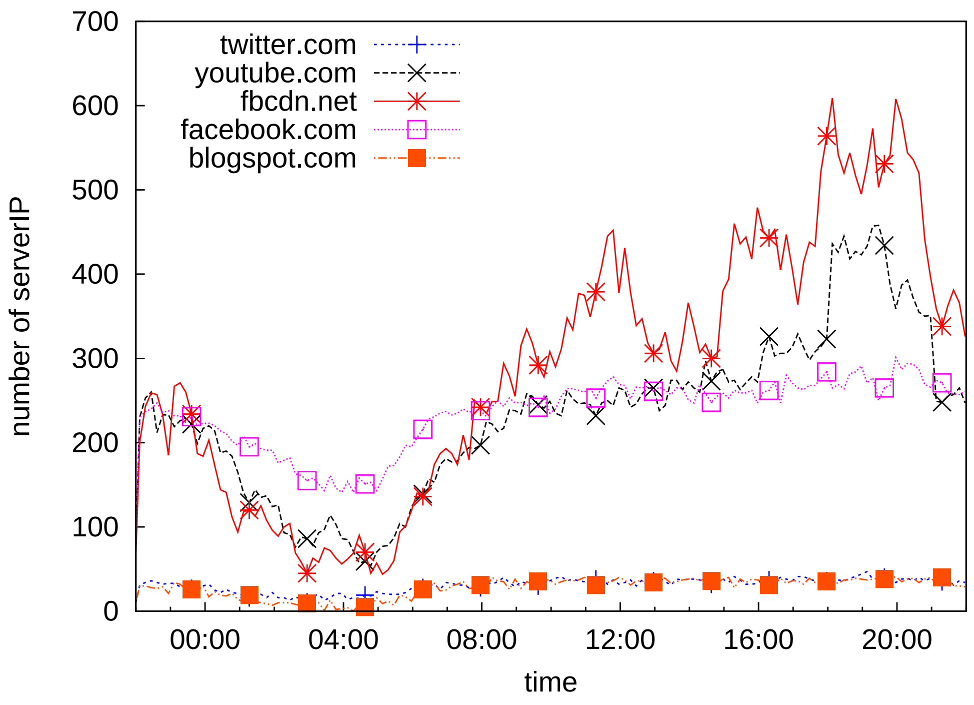


Figure 3.4: Number of IP addresses serving some particular 2nd-level domain name. EU1-ADSL2.

3.6.2 Traffic Visibility and Policy Enforcement

The key feature of DN-Hunter is to provide a “label” (i.e., the FQDN that the client was contacting) to every flow in the network automatically. To show how this labeling evolves over time, Fig. 3.6 shows the growth of unique entities - FQDNs, second-level domain names, and *serverIP* over a 18 days as observed from EU1-ADSL2 vantage point, which is one of the vantage points used to generate the datasets. Once again, it clearly shows a diurnal pattern where the increase in unique entities is much higher during the day than the night. After a steep growth during the first few days, the number of unique *serverIP* addresses and second-level domains reach a saturation point and do not grow much. This result basically indicates that the same *serverIP* addresses are used to serve the contents for the same organizations (i.e., second-level domains). However, a surprising result is regarding the unique FQDNs. As it can be noticed, the number of unique FQDNs keeps increasing even after 18 days of observation. In 18 days we saw more than 1.5M unique FQDNs and it was still growing at the rate of about 100K per day. This reflects the fact that the content being accessed on the Internet keeps growing, with new services popping up regularly. The main take away point is that in order to get fine-grained traffic

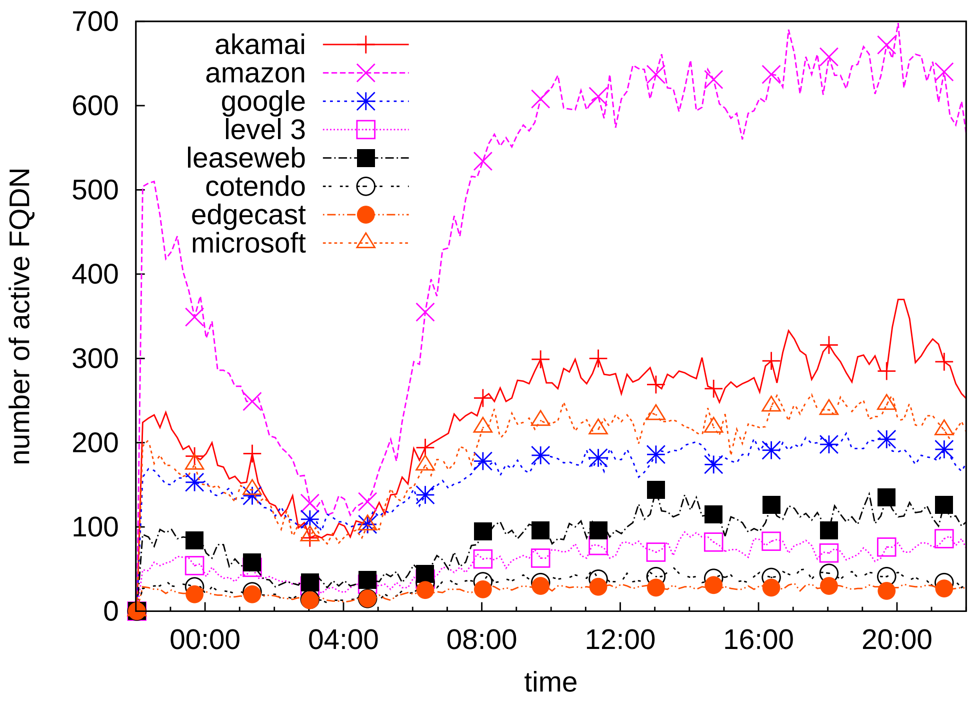


Figure 3.5: Number of FQDN served by CDNs through a day. EU1-ADSL2.

visibility (and thus be applied for policy enforcement), it is critical to use a tool like DN-Hunter that can dynamically keep track of the content and their association with content providers and the hosts serving the content.

3.6.3 The Case of Encrypted Traffic

One of the main advantages of DN-Hunter, when compared to traditional DPI solutions, is its ability to label encrypted (TLS/SSL) flows. Traditional DPI solutions cannot identify encrypted traffic by inspecting the packet content and matching it against a signature. However, a DPI solution would eventually inspect for information in the certificates exchanged during the TLS/SSL handshake, then somehow it would figure out the FQDN associated to the connection as well as the organization that will provide the content.

Then seems it appears so simple to extract FQDN using inspection on the TLS/SSL certificates, it is devised an experiment to compare the certificate inspection approach with DN-Hunter. Thus, certificate inspection functionality is implemented on *Tstat*. Tab. 3.4 compares certificate inspection approach with DN-Hunter for all TLS flows in the EU1-ADSL2 dataset. Results show that DN-Hunter clearly outperforms the certificate inspection approach. For 23% of the flows in

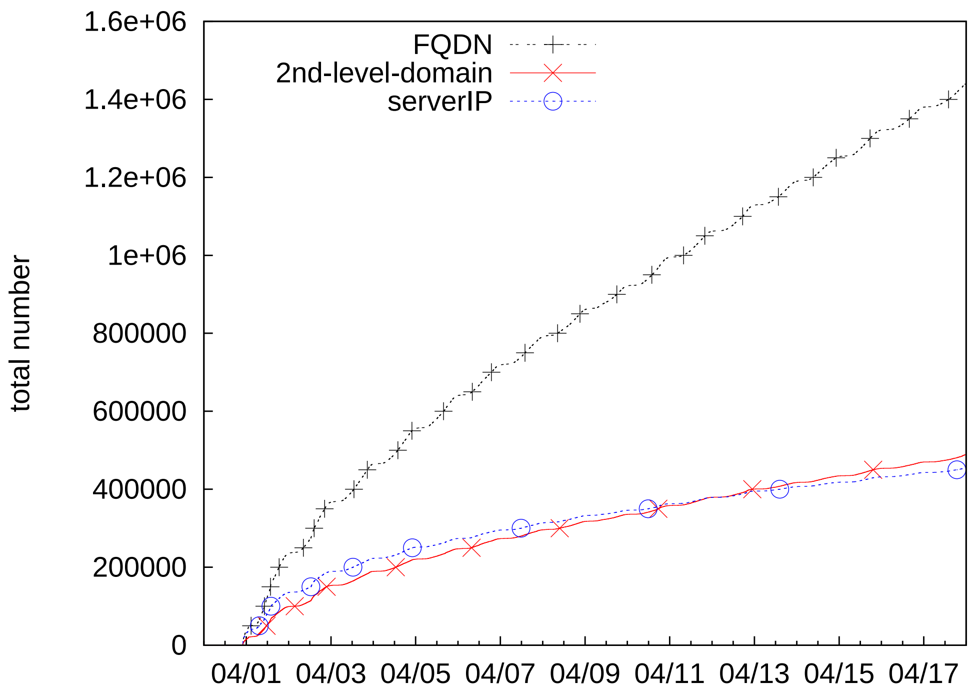


Figure 3.6: Unique FQDN, 2nd level domain names and IP birth processes. EU1-ADSL2 live.

the trace there was no certificate, while for 40% of the flows the server name in the certificate was totally different from the FQDN. For the other 37% of the flows that matched the second-level domain name in the FQDN, only 18% matched the complete FQDN. The main problems with the certificate inspection approach are three-fold: (*i*) The server name can be "generic", e.g., $*.Google.com$, thus not giving the fine-grained visibility into the actual services. (*ii*) The server name may indicate the server used by the hosting CDN and may not reflect anything about the service, e.g., $a248.akamai.net$ in the certificate for providing Zynga content, and (*iii*) Certificate exchange might happen only the first time a TLS/SSL server is contacted and all other flows following that will share the trust. Thus using such an approach is almost infeasible.

3.6.4 Spatial Discovery of Servers

The main goal of the spatial discovery module is to track a particular resource (FQDN or second-level domain) to understand which $serverIP$s and CDNs provides the requested content. For the ease of exposition, in this section, two specific second-level domains, LinkedIn and Zynga, are analyzed. Fig. 3.7 shows the mapping

Table 3.4: Comparison between the server name extracted from TLS certificate-inspection and the FQDN using DN-Hunter. EU1-ADSL2.

Certificate equal FQDN	18%
Generic certificate	19%
Totally different certificate	40%
No certificate	23%

between the various FQDNs of LinkedIn and the CDNs serving the content in US-3G dataset. The oval nodes represent DNS tokens extracted from the FQDNs, while arcs connect the tokens to reconstruct the FQDN. The numbers in these tokens are represented as a generic letter, N. The rectangular nodes group tokens by the CDN hosting them based on the information from the MaxMind database. To illustrate the concept better let us consider the leftmost branch in Fig. 3.7. The complete FQDN is the concatenation of all the tokens, i.e., *mediaN.linkedin.com*. These FQDNs are served by Akamai CDN using 2 servers and accounts for 17% of the total flows destined to *linkedin.com*. In order to limit the size of the figure, 7 different tokens are hidden in the rightmost branch of the tree given their marginal traffic generated.

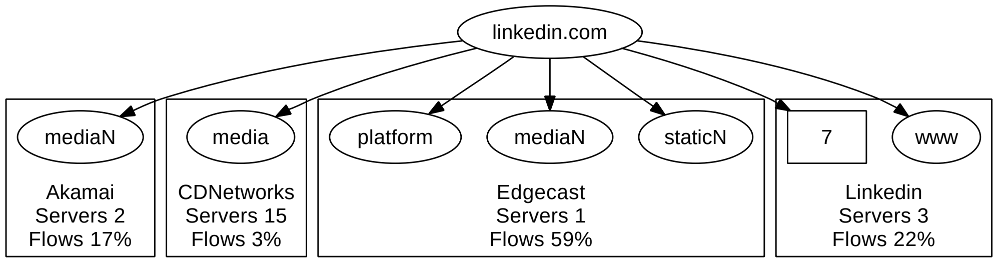


Figure 3.7: LinkedIn domain structure served by two CDNs seen from US-3G.

From the figure, it is easy to see that LinkedIn relies on the service offered by several CDN providers. Only the *www.linkedin.com* FQDN along with 7 other FQDNs are served by LinkedIn managed servers. Most of the static content is served by hosts in three different CDNs - Akamai, CDNetwork, and Edgecast. In fact, EdgeCast serves 59% of all flows with a single $serverIP$ address. On the contrary, CDNetworks, serves only 3% of flows with 15 different $serverIP$ addresses.

Consider now the second sample domain - Zynga (see Fig. 3.8). Here, some Amazon EC2 cloud service machines are spotted providing computational resource for gaming service, while Akamai CDN machines provide static content to end-users. Some services/games like MafiaWars are served directly by Zynga owned servers. Interestingly, around 500 Amazon $serverIP$ addresses are contacted and they handle 86% of all Zynga flows. Akamai serves fewer requests (7%); yet, 30 different *serverIP* are spotted.

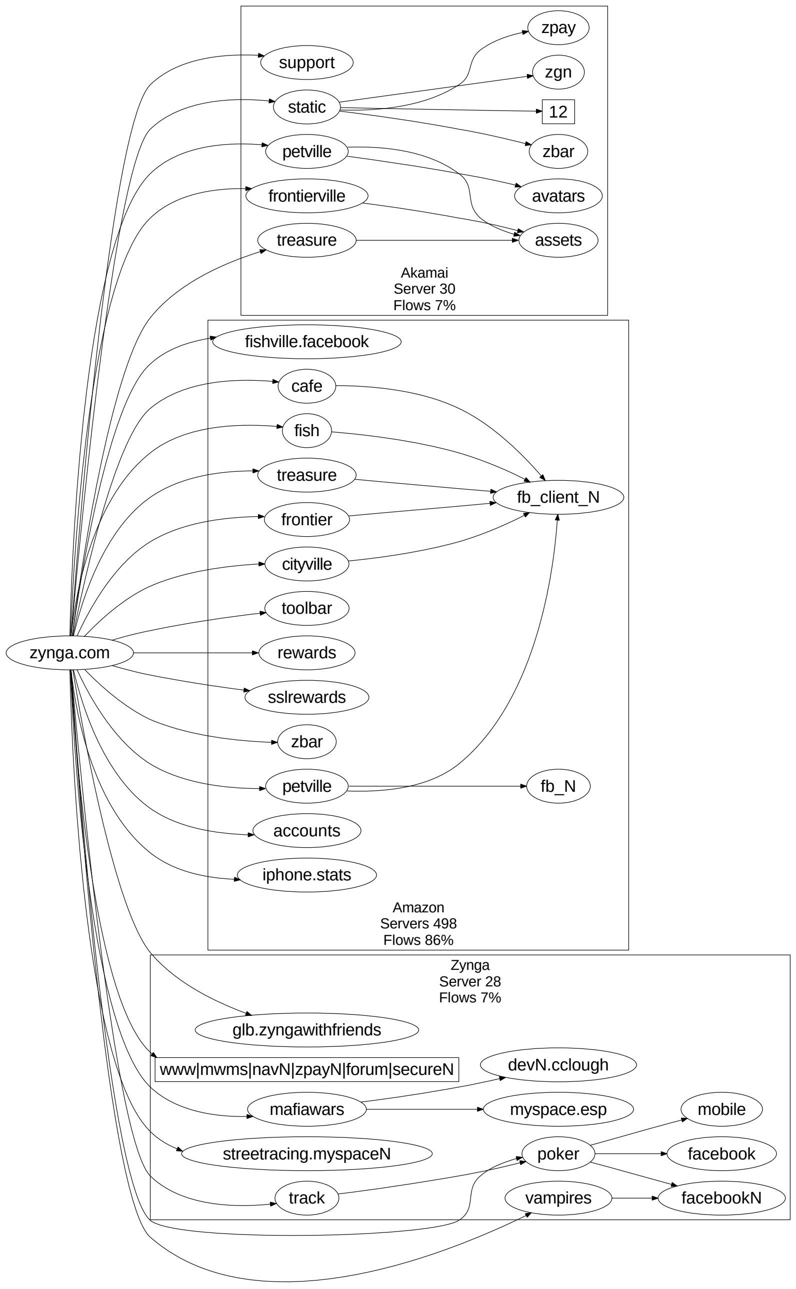


Figure 3.8: Zynga domain structure served by two CDNs seen from US-3G.

Given that the off-line analyzer relies on actual measurement of network traffic, it is able to capture the service popularity among the monitored customers, the bias induced by the server selection and load balancing mechanisms. To elaborate this further, let consider Fig. 3.9. Each of the three sub-figures corresponds to a different content provider (i.e., the second-level domain name). For each of these content providers we plot the access patterns in three of our traces (EU1-ADSL1, US-3G, and EU2-ADSL). In other words, the x-axis in each of these graphs are the CDNs hosting the content and the y-axis represents different traces. Notice that for every CDN, the figure shows all the accessed *serverIP* addresses that belong to the CDN. Hence the width of the column representing each CDN is different. Also, the gray scale of each block in the graph represents the frequency of access; the darker is a cell, the larger is the fraction of flows that a particular *serverIP* was responsible for. The "SELF" column reports cases in which the content providers and content hosts are the same organization.

The top graph in Fig. 3.9 shows the access pattern for Facebook. Observe that in all the dataset most of the Facebook content or services are provided by Facebook servers. In addition, Facebook also relies on Akamai for providing content. Akamai uses different *serverIP* in different geographical regions. On the other hand, the second figure related to Twitter shows an access pattern a little different, regarding on how the infrastructure relies on CDNs according to the spatial location of end-users. Although, Twitter relies heavily on its own servers to provide content, equally they also rely on Akamai to serve content to users in Europe. Nevertheless, the dependence on Akamai is significantly less in the US. The bottom figure shows the access patterns for Dailymotion, a video streaming site. Dailymotion heavily relies on Dedibox to host content both in Europe and US. But it does not use any of its servers to provide content to European end-users, as it does with American end-users. Moreover, if other CDNs are considered, the distribution pattern change according the location. In the USD Dailymotion relies on Meta and NTT, whereas in Europe it relies a little on Edgecast.

3.6.5 Content Discovery

Although the spatial discovery module provides invaluable insight into how a particular resource is hosted on various CDNs, it does not help in understanding the complete behavior of CDNs. In the content discovery module our goal is to understand the content distribution from the perspective of CDNs and cloud service providers. Tab. 3.5 shows the top-10 second-level domains served by the Amazon EC2 cloud in EU1-ADSL1and US-3G. Notice that one dataset is from Europe and the other from US, and clearly the top-10 in the two datasets do not match. In fact, some of the popular domains hosted on Amazon for US users like admarvel, mobclix, and andomedia are not accessed by European end-users, while other domains like cloudfront, invitemedia, and rubiconproject are popular in both datasets. It is

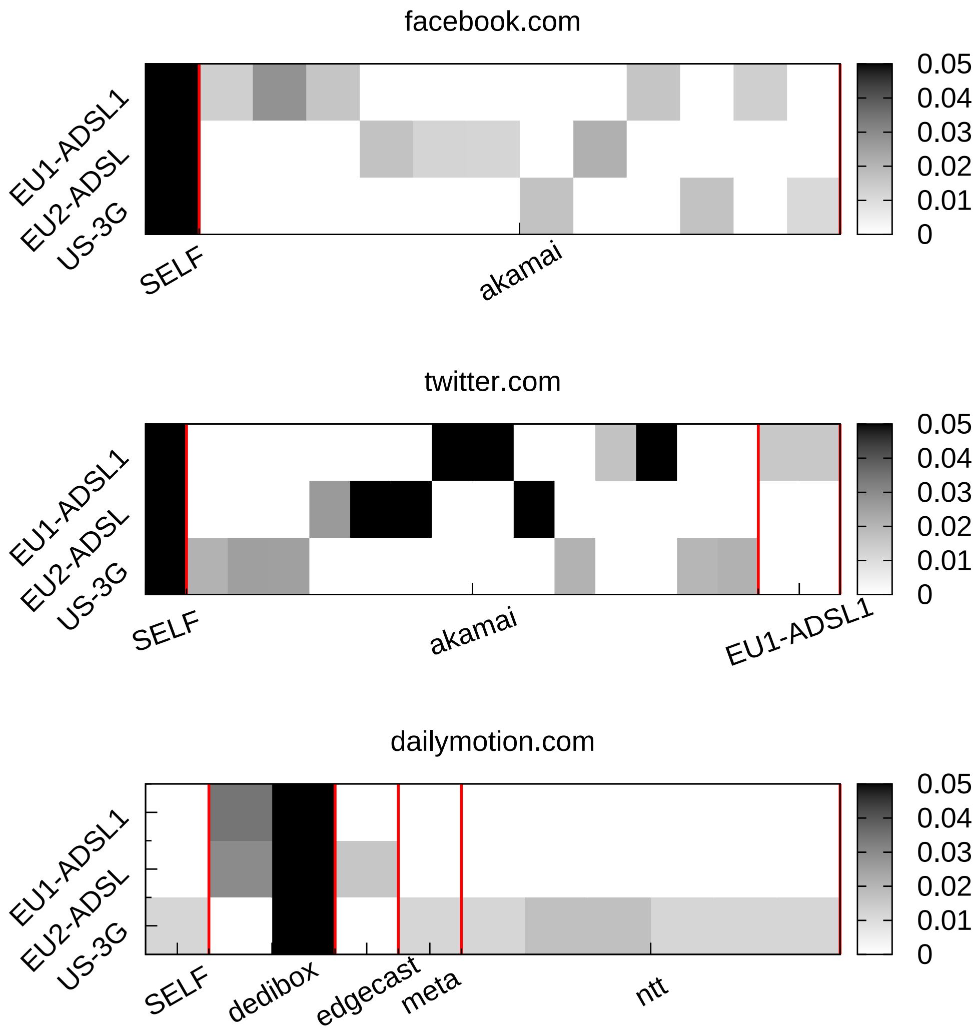


Figure 3.9: Organizations served by several CDN according to viewpoint.

found that popularity and access patterns of content/service for a cloud provider, changes depending on geographical spatial location of end-users; therefore extrapolating results from one geography to another might result in incorrect conclusions.

These variation in the patterns could be explained by two different phenomena. First as the previous section pointed out, it could be a decision on the company hiring different CDNs or clouds to reach end-users according to their location. Or second reason, it could be that some content or services might be more interesting for some particular population.

Table 3.5: Top-10 domains hosted on the Amazon EC2 cloud.

Rank	**US-3G**	%	**EU1-ADSL1**	%
1	cloudfront.net	10	cloudfront.net	20
2	invitemedia.com	10	playfish.com	16
3	amazon.com	7	sharethis.com	5
4	rubiconproject.com	7	twimg.com	4
5	andomedia.com	5	amazonaws.com	4
6	sharethis.com	5	zynga.com	4
7	mobclix.com	4	invitemedia.com	2
8	zynga.com	3	rubiconproject.com	2
9	admarvel.com	3	amazon.com	2
10	amazonaws.com	3	imdb.com	1

3.6.6 Automatic Service Tag Extraction

Another interesting application of DN-Hunter is in identifying all the services/applications running on a particular layer-4 port number. This application is only feasible due to the fined grained traffic visibility provided by DN-Hunter. To keep the tables small, there are shown just the results extracted on a few selected layer-4 ports for two data sets - EU1-FTTH (Tab. 3.6) and US-3G(Tab. 3.7). These tables show the lists of terms along with the weights returned by the Service Tag Extraction Analytics algorithm (Alg. 4). The last column in each of these tables is the ground truth obtained using Tstat DPI and augmented by Google searches and our domain knowledge.

Clearly from the both tables, it is observed that the that the most popular terms extracted, in fact represents the application/service on the port. Some of them like pop3, imap, and smtp are very obvious by looking at the top keyword. However, some of the other are not very obvious, but can be derived very easily. For example, consider the port 1337. TCP port 1337 is not a standard port for any service and even a Google search for TCP port 1337 does not yield straight forward results. However by adding “exodus” and “genesis”, the main keywords extracted in DN-Hunter, to the Google search along with TCP port 1337 immediately shows that this port in US-3G dataset is related to www.1337x.org BitTorrent tracker.

3.6.7 Case Study - appspot.com Tracking

In this section, it is presented a surprising phenomenon discovered using DN-Hunter's ability to track domains. Consider the domain *appspot.com. Appspot* is a free web-apps hosting service provided by Google. The number of applications, CPU time and server bandwidth that can be used for free are limited. Using the labels for

Table 3.6: Keyword extraction example considering well-known ports. EU1-FTTH.

Port	Keywords	GT
25	(91)smtp, (37)mail, (22)mxN, (19)mailN, (18)com, (17)altn, (14)mailin, (13)aspmx, (13)gmail	SMTP
110	(240)pop, (151)mail, (68)popM, (33)mailbus	POP3
143	(25)imap, (22)mail, (12)pop, (3)apple	IMAP
554	(1)streaming	RTSP
587	(10)smtp, (3)pop, (1)imap	SMTP
995	(101)pop, (37)popN, (31)mail, (20)glbdns (20)hot, (17)pec	POP3S
1863	(21)messenger, (5)relay, (5)edge, (5)voice, (2)msn, (2)com, (2)emea	MSN

Table 3.7: Keyword Extraction for frequently used ports; Well-known ports are omitted. US-3G.

Port	Keywords	GT
1080	(51)opera, (51)miniN	Opera Browser
1337	(83)exodus, (41)genesis	BT Tracker
2710	(62)tracker, (9)www	BT Tracker
5050	(137)msg, (137)webcs, (58)sip, (43)voipa	Yahoo Messager
5190	(27)americaonline	AOL ICQ
5222	(1170)chat	Gtalk
5223	(191)courier, (191)push	Apple push services
5228	(15022)mtalk	Android Market
6969	(88)tracker, (19)trackerN, (11)torrent, (10)exodus	BT Tracker
12043	(32)simN, (32)agni	Second Life
12046	(20)simN, (20)agni	Second Life
18182	(92)useful, (88)broker	BT Tracker

various flows in the labeled flows database, all traffic associated with services is extracted. This allows to understand the kind of applications and services that are hosted in the Appspot cloud.

Fig. 3.10 shows the most relevant applications, in terms of connections, hosted on *appspot* as a word cloud where the larger/darker fonts represent more popular applications. Although *appspot* is intended to host legacy applications, it is easy to see that users host applications like “open-tracker”, “rlskingbt”, and the like. A little more investigation reveals that these applications actually host *BitTorrent* trackers for free. With the help of the information from DN-Hunter and also the Tstat DPI deployed at the European ISP, it is found that there are several trackers and other legacy applications running in the *appspot.com* site. These findings are presented in Tab. 3.8 as well. As we can see, BitTorrent trackers only represent 7% of the applications but constitute for more flows than the other applications. Also, when considering the total bytes exchanged for each of these services, the traffic from client-to-server generated by the trackers is a significantly large percentage of the overall traffic.

Fig. 3.11 depicts a timeline (considering bins of 4 hours interval) of 18 days long. This timeline shows when the trackers were active over that period. A dot represents a tracker active on that time bin of 4 hours. On the Y-axis, trackers are identified by an arbitrary ID, starting at 1 and increasing based on the time when the tracker was first observed. Out of the 45 trackers seen in the 18 day period, about 33% (colored in red, ids 1-15) stayed active for all the 18 days. Trackers with ids 26-31 (colored in blue) exhibit a unique pattern of on-off periods. In other words, all of these trackers were accessed during the same time intervals. Such a synchronized behavior indicates, with high probability, that one BitTorrent client was part of a swarm, so that when the client was active, the tracker has been monitored by DN-Hunter. Similar considerations hold for the trackers with ids 33-34 (colored in green), which was seen accessed on May 5th for the first time.

Finally, trackers with ids larger than 33 highlight a birth process according to which a new tracker is born and accessed by only a few BitTorrent clients. This is due to the particular environment the trackers live: since Appspot limits the amount of resource an application can use for free, new trackers are born once they run out to resources. Indeed, checking the status of the trackers during May 15th 2012, it has been verified that most of them, while still existing as FQDN, run out of resources and made unavailable by Google. They live as *zombies*, and some BitTorrent clients are still trying to access them.

Table 3.8: Appspot services. EU1-ADSL2 live.

Service Type	**Services**	**Flows**	**C2S**	**S2C**
Bittorrent Trackers	56	186K	202MB	370MB
General Services	824	77K	320MB	5GB

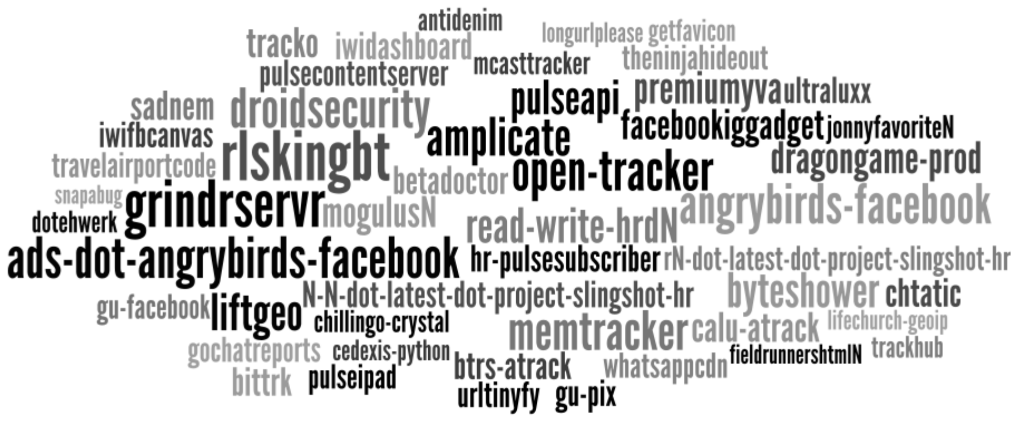


Figure 3.10: Cloud tag of services offered by Google Appspot. EU1-ADSL2 during live deployment.

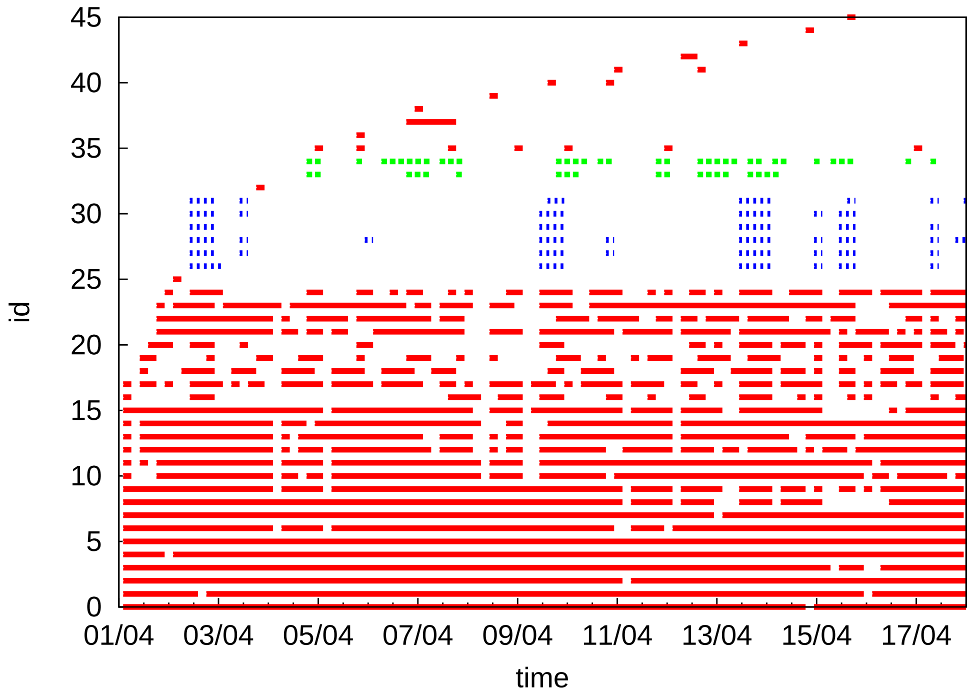


Figure 3.11: Temporal evolution of the BitTorrent trackers running on Appspot.com. EU1-ADSL2 during live deployment.

3.7 Dimensioning the FQDN Clist

In Sec. 3.4, it is presented the design of the DNS resolver. One of the key data structures of the DNS resolver is the FQDN Clist. Choosing the correct size for the

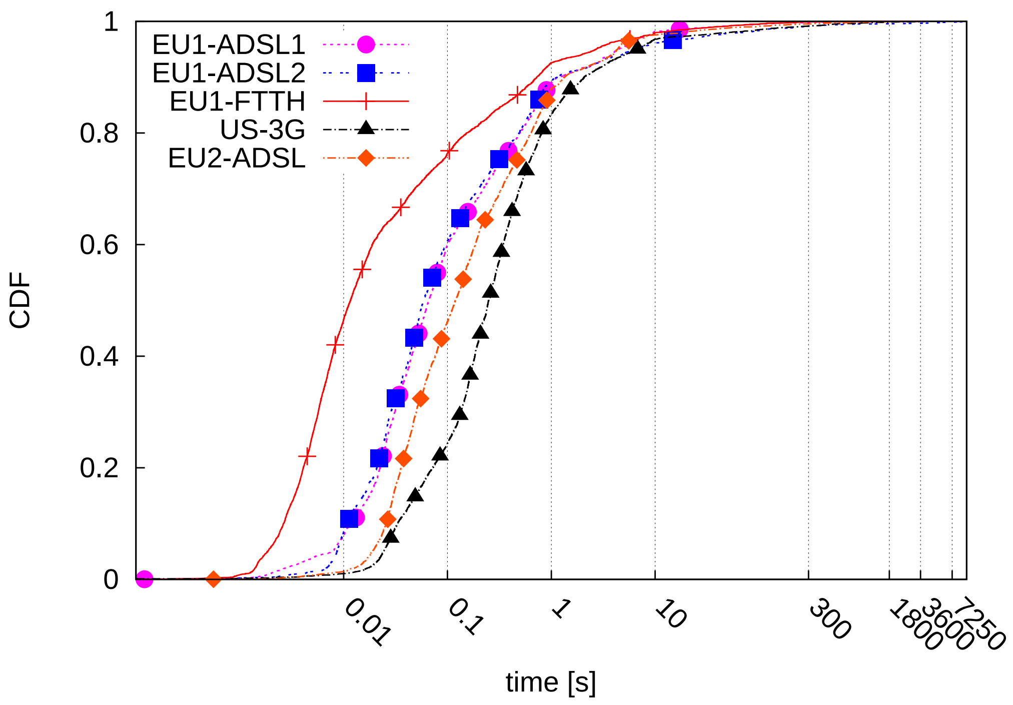


Figure 3.12: Time elapsed between DNS response and the *first* TCP flow associated to it.

Clist is critical to the success of DN-Hunter. This section presents a methodology to choose the correct value of L (size of the Clist) and the real-time constraint implication.

Fig. 3.12 shows the Cumulative Distribution Function (CDF) of the "first flow delay", i.e., the time elapsed between the observation of the DNS response directed to $clientIP$ and the first packet of the *first* flow directed to one of the $serverIP$ addresses in the answer list. Semilog scale is used for the sake of clarity. In all datasets, the first TCP flow is observed after less than 1s in about 90% of cases. Access technology and sniffer placement impact this measurement; for instance, FTTH exhibits smaller delays, while the 3G technology suffers the largest values.

Interestingly, in all traces, for about 5% of cases the first flow delay is higher than 10s, with some cases larger than 300s. This is usually a result of aggressive pre-fetching performed by applications (e.g., web browsers) that resolve all FQDNs found in the HTML content before a new resource is actually accessed. Tab. 3.9 quantifies the fraction of "useless" DNS responses, i.e., DNS queries that were not followed by any TCP flow. Surprisingly, about half of DNS resolutions are useless. Mobile terminals are less aggressive thus resulting in lower percentage of useless responses.

Table 3.9: Fraction of useless DNS resolution.

Trace	**Useless DNS**
EU1-ADSL1	46%
EU1-ADSL2	47%
EU1-FTTH	50%
EU2-ADSL	47%
US-3G	30%

Fig. 3.13 shows the CDF of the time elapsed between the DNS response and *any* subsequent TCP flow the client establishes to any of the $serverIP$ addresses that appeared in the answer list. It reflects the impact of caching lifetime at the local DNS resolver at clients. The initial part of the CDF is strictly related to the first flow delay (Fig. 3.12); subsequent flows directed to the same FQDN exhibit larger time gaps. Results show that the local resolver caching lifetime can be up to few hours. For instance, to resolve about 98% of flows for which a DNS request is seen, $Clist$ must handle an equivalent caching time of about 1 hour.

Fig. 3.14 shows the total number of DNS responses observed in 10m time bins. As it can be point out, at the peak time about 350,000 requests in EU1-ADSL1 dataset. In this scenario, considering a desired caching time of 1h, L should be about 2.1M entries to guarantee that the DNS resolver has an efficiency of 98%.

Also it is checked the number of $serverIP$ addresses returned in each DNS response. Since the $clientIP$ can choose (in most cases randomly) any one of the $serverIP$ addresses to open the data connection, all of the $serverIP$ addresses must be stored in the DNS resolver. The results from all the datasets are very similar with about 40% of responses returning more than one $serverIP$ address. About 20-25% of responses include 2-10 different ip-addresses. Most of these are related to servers managed by large CDNs and organizations. For example, up to 16 $serverIP$s are returned when querying any Google FQDN. The maximum number exceeds 30 in very few cases.

3.7.1 Collision Probability in Practice

Now it is considered the possibility of collisions, that is when when the same $clientIP$ is accessing two or more FQDNs hosted at the same $serverIP$. DN-Hunter actually returns the last observed FQDN, thus possibly returning incorrect labels. This event is called by the author as a "collision". To assess such collision probability, traces are examined to see frequently such situation occurs. It is observed that the most common reason for this is due to HTTP redirection, e.g., *google.com* being redirected to *www.google.com* and then to *www.google.it*, all FQDNs being served by the same $serverIP$. Hence, two types of collisions are possible: (i) generic collision, and

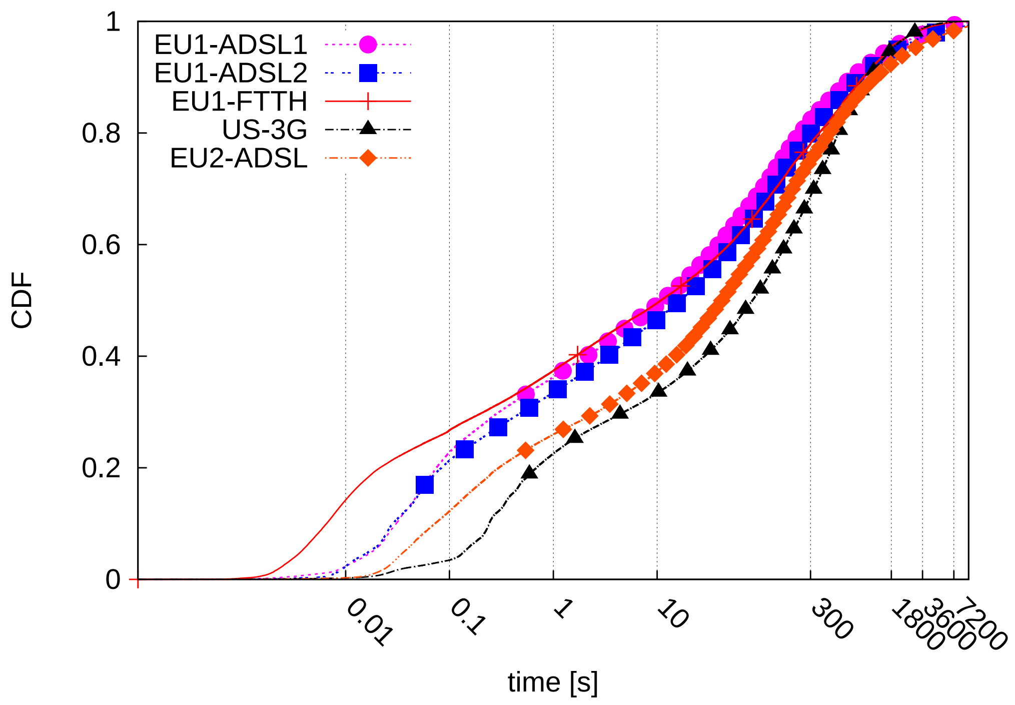


Figure 3.13: Time elapsed between a DNS response and *any* TCP flow associated to it.

(*ii*) severe collision. The latter accounts for the cases in which the two conflicting FQDNs have different second level domain names, e.g., *google.com* and *youtube.com*.

Table 3.10: DN-Hunter Collision Ratio

Trace	**Total Test**	**Generic coll.**	**Severe Coll**
US-3G	1.3M	0.7%	0.6%
EU2-ADSL	11.9M	0.7%	0.3%
EU1-ADSL1	4.9M	0.01%	0.004%
EU1-ADSL2	2.6M	0.1%	0.05%
EU1-FTTH	680k	1.6%	0.9%

Collisions can occur in DN-Hunter when the same *clientIP* address contacts the same *serverIP* address several times during the period of our traces. In the second column of Tab. 3.10 it is reported the total number of flows that are potential candidates for collision by DN-Hunter. Consider the case when there is no collision; then two successive flows from the same <*clientIP*, *serverIP*> are associated to the same FQDN. But, if the associated domain name is different, then it is declared that a collision has occurred. Once a collision occurs, it is examined the domain names

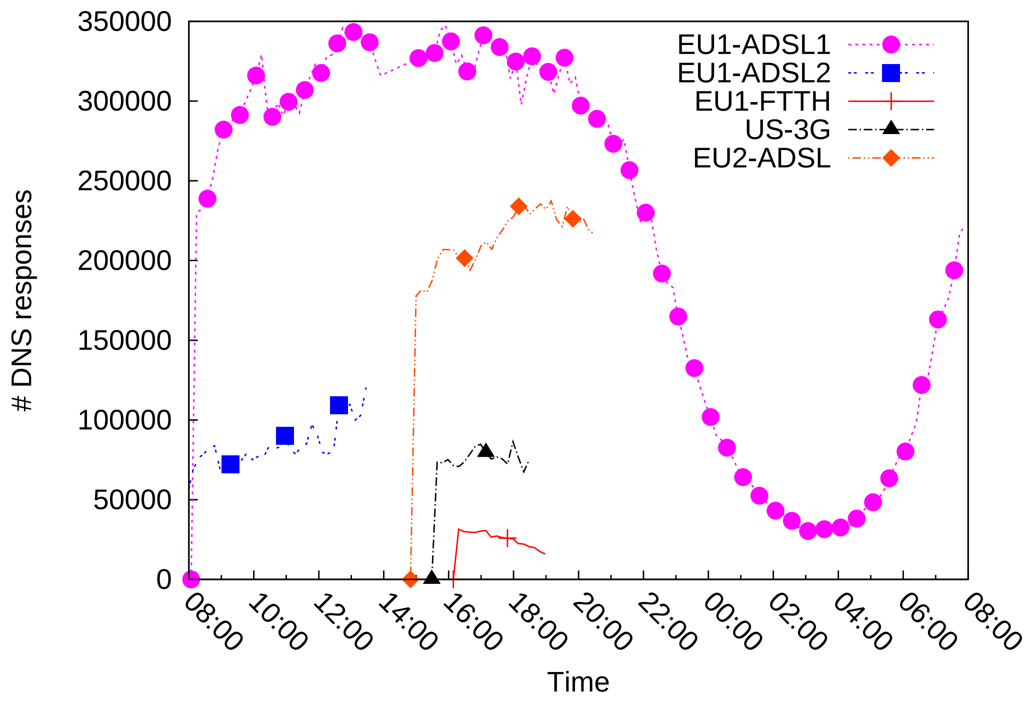


Figure 3.14: DNS responses observed during a day by intervals of 10 minutes in EU1-ADSL1.

to determine if the collision is a generic one or a severe one. The second column of Tab. 3.10 reports the total number of potential candidate flows for collision. The third and the fourth columns report the percentage of the potential collision candidates that translate into generic and severe collisions respectively. Clearly, the collision rate is extremely low in all the traces used for evaluation. In fact, observe that more than 70% of collisions occur within a 1 second interval, suggesting that these collisions could be a result of HTTP redirection. Although, these are collisions that might not impact the accuracy of DN-Hunter, in Tab. 3.10 still are considered them to be collisions and give a conservative estimate. In summary, the problem of collisions has minimal impact on the operational value of DN-Hunter.

Finally, note that it is possible to extend DN-Hunter to return all possible labels associated to a flow instead of only the latest one, thus giving the network administrator the ability to resolve the collisions using more advanced policies than by strictly using the latest FQDN.

3.7.2 Implementation Issues

DN-Hunter is a passive sniffer which assumes to observe DNS and data traffic generated by the end-users. The natural placement of the sniffer is at the network boundaries, where end-users' traffic can be observed. The Flow Sniffer and the DNS Response Sniffer may also be placed at different vantage points, e.g., the latter may be located in front of (or integrated into) the internal DNS server to intercept all DNS queries. Considering DNS traffic sniffing, DNSSEC [9] poses no challenge since it does not provide confidentiality to DNS traffic. DNSCrypt [10], a recent proposal to encrypt DNS traffic, on the contrary, would make the DNS Response Sniffer ineffective. DNSCrypt is not yet widely deployed and it requires significant DNS infrastructure [10] changes to be pragmatic in the near future [11].

DN-Hunter labels TCP flows using $(clientIP, serverIP)$ pair as the key. In scenarios where the same $clientIP$ is shared by multiple end-hosts via NAT/connection sharing tools, DN-Hunter may associate the wrong FQDN to the flow,causing a possible collision. Notice that this can happen only if the end-hosts are initiating connection at almost the same time to the same $serverIP$ (which has been returned when requesting two different FQDN). This can also happen when a end-user is using multiple applications (e.g., multiple browsers windows/tabs opened at the same time) to access different services running on the same $serverIP$ at the same time. It is not feasible to quantify how probable these events are in practice in the traces, and given the very low collision probability (see Sec. 3.7.1), it is expected to be a marginal problem in the considered scenarios. Even though in scenarios where single households are allowed to share the internet connection among few terminals, the collision probability is still marginal. However, in the scenario where there are several hundreds (or thousands) of users behind NAT and the monitoring point for DN-Hunter is after the NAT, DN-Hunter will be severely impaired. Deploying DN-Hunter before the NAT will result in accurate results.

3.8 Conclusions

Throughout this chapter DN-Hunter has been introduced. It is a novel tool that links the information found in DNS responses to traffic flows generated during normal Internet usage. Explicitly aimed at discerning the tangle between the content, content providers, and content hosts (CDNs and cloud providers), DN-Hunter unveils how the DNS information can be used to paint a very clear picture, providing invaluable information to network/security operators. Moreover several applications and heuristics of DN-Hunter are presented, ranging from automated network service classification to dissecting content delivery infrastructures.

As notable examples, it is shown shown how DN-Hunter is helpful (*i*) in providing a fine-grained traffic visibility even when the traffic is encrypted (i.e., TLS/SSL flows), (*ii*) in identifying flows even before the flows begin, thus providing superior network management capabilities to administrators, (*iii*) in observing and tracking how CDNs and cloud providers host content for a particular resource over time, and (*iv*) in discerning the services/contents hosted by a CDN or cloud provider. Applications presented in this chapter are limited within the range of examples, indeed DN-Hunter is not limited to the applications presented in this work.

DN-Hunter has been deployed in actual ISP networks, providing network administrators enhanced traffic visibility and the ability to perform detailed forensics if required. A key challenge for the future is to devise automatic algorithms to mine the amount of data exposed by DN-Hunter and by network monitoring tools in general. In fact, this challenge is common to the network monitoring and measurement community and modern tools have to be devised to dig into this enormous volume of data.

Chapter 4

Web services from the ISP viewpoint

4.1 Introduction

Last years witnessed the growth of cloud-based services that provide computing, storage and offloading capabilities on remote datacenters, offering the opportunity to customers to reduce costs by virtualizing hardware management. The leading position in this panorama is taken by Amazon, which offers a large gamma of cloud-based services, named Amazon Web Services (AWS). Active since July 2002, they offer computing and storage cloud solutions. The most well-know Amazon cloud services are “Elastic Compute Cloud” (EC2), and “Simple Storage Service” (S3), with “CloudFront” (CF), the Content Delivery Network (CDN), that has augmented Amazon’s portfolio in late November 2008.

Following the definitions provided in [12], AWS represents an *Infrastructure Provider*, and EC2 and S3 correspond to *Infrastructure as Service* products. In other words, through virtualization, a large set of computing resources, such as storing and processing capacities can be split, assigned, and dynamically sized to satisfy customers’ demand. *Customers* are represented by companies aiming at offering their *content* without carrying on costs and risks of building and managing their own hardware and infrastructure. Given the great scalability and extremely low costs of pay-as-you-go cloud services, many successful companies like Dropbox, Zynga and Netflix to name a few, have been attracted by AWS and successfully relies on them. Amazon indeed represents today one of the largest source of traffic in nowadays Internet, accounting for about 3.1% of the total HTTP/HTTPS traffic flows, i.e., one fifth of Google’s [13].

Given its success, AWS has gained a large interest within the research community. In particular, many works specifically focus on the possibility of exploiting AWS

EC2 for research purposes [14, 15]. Other works instead focus on evaluating the performance of AWS computing and networking virtual resources [16–18]. However, to the best of our knowledge, all the previous works focus on the benchmarking of AWS services and infrastructure, and they all rely on "active" probing. What is missing is the characterization of Amazon Web Services as perceived by the end-users, where the actual workload and performances can be evaluated by means of "passive" observation of traffic. Only by the passive characterization of services it is possible to discover eventual performance degradation and, most of all, to gauge their impact on end-users.

The goal of this paper is thus to provide an extensive study of AWS through the passive analysis of network traffic collected from our University campus and from three large Points of Presence (PoP) of an Italian national-wide Internet Service Provider (ISP). Our datasets span more than 60 days during which the traffic generated by more than 50.000 end-users has been observed.

In this work, we dig into a one week long portion of our dataset with a twofold goal: first, we shed light on the AWS infrastructure itself, proposing a simple yet accurate methodology to reveal the number of datacenters, their locations, and resulting traffic allocation policies. Second, we evaluate which are the services that run on AWS, and how they are accessed by end-users. Notice that providing such characterizations is a challenging task due to the nature of cloud services, where encryption schemes and proprietary solutions are very common, and virtualization allows to share resources and dynamically move content and services over time.

This paper represents the first attempt of observing the Amazon cloud from passive measures. We follow the direction suggested by other works that provide a characterization of popular services such as social networks [19, 20] or YouTube [21]. Our main findings are:

- Among the seven EC2 and S3 datacenters, the one placed in Virginia, close to Washington DC, is the most used one, hosting more than 6.000 EC2 virtual machines and 120 S3 nodes regularly accessed by end-users. It handles alone 85% of total traffic generated by EC2 and more than 64% for S3 – serving daily more than 15TB of data to the ISP end-users in Italy. Surprisingly, the datacenter in Ireland is not the preferred one, and it serves only about 20% of AWS traffic to Italian end-users.
- Web companies that offer their contents from EC2 and S3 systems tend to rely upon one datacenter only. This makes the network to pay the large cost of carrying information to far away end-users. Moreover, it represents a large risk in case of failures, since no automatic load-balancing and migration are offered by AWS. This is confirmed by the results we provide about the outage happened on the 30th of June 2012, in Virginia.
- Performance of datacenters in terms of response time (for EC2) and goodput (for S3) show that the most popular datacenter is also the worst performing one.

Evidence shows that some contents suffer because of under-provisioned instances, but we cannot exclude that the whole infrastructure may be overloaded.
• Considering CloudFront, 24 out of 33 different world-wide caches that build the CDN infrastructure have been spotted in our traces. However, the cache selection policies adopted by CloudFront wisely serve 98% of traffic from the cache placed in Milan, the closest to Italian end-user. The remaining 2% of traffic comes from worldwide caches. This happens to be due to CloudFront directing traffic to other caches for load-balancing purpose. A minimal fraction can be due to incorrect DNS configuration of end-user Clients [22].

We believe this paper provides useful insights about the infrastructure offered by AWS, helping in understanding the properties of services relying on cloud-based platform EC2, S3 and CloudFront. Provided information may result worthwhile for developers aiming at entrust AWS to deploy their contents.

The rest of this paper is organized as follows: Sec. 4.2 describes products offered by AWS, and Sec. 4.3 overviews the data collection procedure and the datasets we study. In Sec. 4.4 and Sec. 4.5 we present the techniques and the metrics, respectively, we employed for the analysis of our datasets. We then show the results of our study, starting with a spatial characterization of AWS infrastructure (Sec. 4.6), moving on the analysis of properties of hosted contents (Sec. 4.7) and ending with the performance evaluation of AWS platform (Sec. 4.8). Finally, Sec. 4.9 concludes the paper.

4.2 Amazon Web Services Primer

Amazon Web Services offers on-demand cloud computing services to its customers, which are mostly Web companies. Often AWS are accessed through HTTP, REST and SOAP protocols (of course other type of protocols are not discarded), which in case it is necessary guaranteeing privacy on the network communications, TLS/SSL can be implemented to encrypt the communications. This chapter is dedicated to the three most popular AWS services which are used by Amazon's customers to reach Internet end-users:
•**EC2** is an on-demand virtual computing environment supported by Xen virtualization [23]; EC2 lets customers rent free-to-use virtual machines, called *instances*, to run any application. At the time in which the all the Amazon monitoring has performed, EC2 instances could be allocated on seven datacenters (called also *datacenters* by AWS terminology[1]) world-widely distributed. Customers can choose any datacenter as support for their instances. Still, Amazon provides different pricing

[1]http://aws.amazon.com/about-aws/globalinfrastructure/

according to the processing power of the instance rented, the time the instance will run and also, on which datacenter is selected by the customer to set the instance. Inside an datacenter, AWS provides an optional DNS-based load-balancing service, called Elastic Load Balancing, whose task is to uniform the workload over rented instances. To the best of our knowledge, no automatic tool for migrating instances among different datacenters has been implemented yet.

Examples of web applications relying on EC2 are social-gaming applications like Zynga and Playfish, or social networks like FourSquare.

• **S3** offers storage services through standard interfaces (REST, SOAP, and BitTorrent). Data are stored by means of "objects" whose size can span from 1B to 5TB each. S3 is reported to store more than a trillion objects as of June 2012[2]. EC2 and S3 are co-located in each datacenter.

• **CloudFront** is the Amazon's Content Delivery Network (CDN) for distributing content from locations near to end-users, thus guaranteeing low latency access and high data transfer speeds. CloudFront can deliver dynamic, static as well as streaming content, using a unknown number of caches placed at Internet Exchange Points around the globe. An example of application supported by CloudFront is Instagram, which exploits CloudFront network to distribute user-generated content among end-users.

4.3 Experimental Datasets

As in the previous chapters, traces obtained by means of passively monitoring operational networks using *Tstat*, the open-source traffic monitoring tool developed at Politecnico di Torino. It analyses packets exchanged by actual end-users (inside monitored vantage points [24]) and Internet servers.

Tstat was installed in four different vantage points where it has been collecting measures from April to June 2012, observing more than 50,000 end-users normally accessing the Internet. The resulting dataset is large enough to reveal significant information about AWS infrastructure as well as about end-users habits and performance of the content served by the AWS cloud.

Tab. 3.1 shows some details about the vantage points: its conventional name, the type of access technologies, the daily number of TCP flows directed AWS, and the number of unique IP address on monitored clients. EU1-ADSL2, EU1-ADSL1 and EU1-FTTH datasets represent end-users of a same ISP collected from three regions in the same European Country. EU1-ADSL2 and EU1-ADSL1 offer connectivity to end-users via ADSL interfaces. Customers in the EU1-FTTH PoP are offered

[2]http://aws.typepad.com/aws/2012/06/amazon-s3-the-first-trillion-objects.html

a Fiber to the Home (FTTH) interface instead. Given the ISP assign a static IP address to each household, the number of observed IP addresses is a lower bound on the number of monitored end-users.

EU1-CAMPUS refers instead to measurements performed at border routers of the University Campus, including traffic from a campus-wide wireless access points and student houses, where NAT and HTTP proxy-ing are often applied. About 10,000 end-users regularly access the Internet via a 1Gb/s link in this case.

For the sake of brevity, being all Italian datasets and the goal of this chapter the understanding of AWS, the analysis of measurements is restricted to those gathered from EU1-ADSL2 PoP which provides the largest dataset. EU1-ADSL2 is an ISP PoP in a large city in Italy, where about 15,000 ADSL lines are active. The entire week starting from April 1st, 2012 is considered, during which 6M TCP connections where directed to AWS servers, for a total of more than 340GB of data exchanged overall. The analysis is repeated considering traffic from other two ISP PoPs and from EU1-CAMPUS, and over different periods of time. Findings on EU1-ADSL2 are general and not biased, being the results collected from the other datasets practically identical. Nevertheless, results presented in this thesis regarding this topic are unfortunately biased, since traffic is observed from one single country. Then some generalizations should be considered with prudence. Naturally, it is expected that some of the findings change based on different cultural influence and geographical location of the vantage point.

4.4 Analysis Methodology

Several challenges have to be faced when passively monitoring cloud services: the separation between *content* and *server* makes it hard to identify which content is actually accessed by the end-users. Formerly content could be easy associated to a set of servers by their IP addresses, but as stated in the previous chapter associations one-to-one between content and server are less common than before. In spite of AWS which provides also cloud services, virtualization, load-balancing and migration of resources do not allow to simply rely on information collected from the Network Layer to identify the content, i.e., the server IP address and its official owner (retrieved from *whois*) give no information on who is really using the server and the content or service being provided.

In addition, the increasing adoption of TLS/SSL encryption by large content providers (including most of those running in the AWS platforms [13]) makes classic Deep Packet Inspection (DPI) useless.

For example, consider the same AWS server hosting *www.acmegame.com* and *www.acmeshop.com* content over HTTPS. The IP addresses of instances hosting them allow only to identify that both are handled by AWS (being the IPs registered to Amazon). Hence, no indication about handled contents can be achieved looking at information from Network Layer. Furthermore, given that *Amazon Web Services* is an infrastructure that comprises several datacenters around the Globe, identifying which datacenter is being used for singular tasks or for hosting content is of interest. Public available databases for geographical location of IPs have been controversial on their accuracy. For instance MaXMind database used in the first chapter to locate users of an P2P network, for the case of locating IP addresses of huge infrastructures which provide content as Google, or Amazon in this case, provides the location of the company headquarters instead of the datacenter location, which is completely wrong. Understanding from/to traffic comes/goes is important from the point of view of end-user as well as for Internet providers. Hence in this section a novel technique is presented, overpass issues related with IP multicast, that would make other accurate methodologies for IP geo-location fail.

4.4.1 Content Discovery

To identify the content retrieved by end-users from a AWS server, Tstat has been augmented to implement DN-Hunter, as presented in the previous chapter, in such a way that the improved Tstat snoops DNS queries performed by clients to resolve URLs, and uses this information to tag network flows extracting the right Fully Qualified Domain Name (FQDN). In the example above, end-user's client has first to resolve the *www.acmegame.com* hostname into a list of IP addresses by contacting the DNS server. Once the list of IP addresses is obtained, the client contacts one of them to fetch the actual content.

DN-Hunter is able to cache all DNS responses, and associates the original hostname used by the end-user to the actual server IP addresses being contacted, associating *www.acmegame.com* to the observed TCP flows. This key feature allows Tstat to recover the original content name requested by the client and being served by an AWS server.

4.4.2 Datacenter and Service Discovery

Since AWS offers different services to its customers, that is *Amazon S3*, *Amazon EC2* and *Amazon CloudFront*; to better understand AWS, it is necessary to separate and individuate which part of traffic belongs to each one. This is analogous to the process of channel identification of the first chapter. First of all, traffic owned by *Amazon* is isolated based on the IP addresses retrieved from an organization database. For this particular scope, WHOIS and MaxMind organization databases are suitable and well accurate. Once Amazon traffic is separated, EC2, S3 and *Amazon CloudFront* traffic is isolated. For some of the traffic going to these three infrastructures, provided information by DNS becomes useful. AWS indeed follow a strict naming rule for EC2: the instance IP address *a.b.c.d* is registered with a Type-A DNS record as **ec2**-*a-b-c-d.XXXXX.amazonaws.com*, where XXXXX is a variable string. A simple DNS reverse lookup from the IP address allows to discover that *a.b.c.d* hosts an EC2 instance.

Unfortunately, the Type-A record for an IP address of S3 and CloudFront does not always reveal which service that server is providing. Hence, a quite simple technique called *HTTP-knocking* is proposed in this chapter is used to identify such services. HTTP-knocking send an HTTP HEAD request to the inspect the *Server* field returned in the HTTP response by the server; either `AmazonS3` or `CloudFront` are always returned in the HTTP response, allowing easy classification of the service hosted by that IP address.

The first finding is that the address spaced We have found that the address spaces used by CloudFront and S3 servers are separated. Furthermore, it is verified that the allocation of IP addresses to these services is "static", allowing HTTP-knocking to be performed once for each IP address.

Notice that HTTP-knocking cannot be employed for EC2 since instances are managed by Amazon's customers who are free to run different Operating Systems as well as different flavours of HTTP servers.

HTTP-knocking and name reversing have been performed over any detected Amazon IP address in the dataset, letting us identify every IP address associated to EC2, S3 and CloudFront, observed from the vantage points.

4.4.3 Server Geolocation

As said previously, IP geographical location is difficult when the target are IP addresses which belong to huge organizations, which are able to play around with route policies globally. Geolocation of IP addresses represents a well-known problem. Common public databases, such as RIPE[3], ARIN[4] or MaxMind, do not represent reliable sources when seeking for information about the physical location of a machine associated to a given IP address [25]. CBG [1] geolocation technique is far a very accurate technique to determine the location of an endpoint, but in this context, since it relies on RTT between spread globally probes and the target IP, it could be not accurate in cases that the same IP address belongs to two different geographical locations according from where it is being contacted.

A simple, yet accurate, approach is to exploit again information provided by the Type-A DNS record assigned to servers by Amazon administrators. Indeed, the Type-A record returned when performing the reverse lookup of AWS IP addresses

[3]http://www.ripe.net/db/index.html

[4]http://www.arin.net

often unveils information about server placement. In particular, the standard International Air Transport Association (IATA) airport location identifiers are used by Amazon in the form *server-a-b-c-d.**AIR**.r.cloudfront.net*, where AIR is the IATA airport code of the nearest large airport. Unfortunately, not all Type-A entry of AWS servers include the IATA code, e.g., *s3-1.amazonaws.com*.

To geolocate IP addresses for which no IATA code is found, it is proposed a technique called *Geolocation by Network Pivots* (see Alg. 5). This technique is an active measurement that uses `traceroute` towards the target IP address to discover machines along the path. These machines along the path are called *pivots* and some information about location can be leveraged, for instance using the domain names associated to the interfaces of the machines. Similar to some Amazon domain names, some of the pivots have domain names which can be easy associated to geographical locations. This technique focus on the last in the path, machine domain name that provides information about the location. By convenience this last machine is called *The Pivot*. The difference of RTT from the Pivot to the targeted IP address is a measure of the geographical distance among the two nodes, so that the smaller is the delay, the smaller their distance. This allows to position IP addresses close to well-known locations, using a single probe located in the same monitored network or geographically near to the monitored network, avoiding in this way any strange routing policy that could trick a geolocation technique as ANYCAST policies used by some CDNs. The approximation obtained by running this simple algorithm have been proven to be almost perfect on those AWS servers whose position has been cross-checked via the IATA code (RTT errors below 1ms on average).

As convention, throughout the rest of this chapter datacenters will be identified by it corresponding IATA codes instead to the conventional names assigned by Amazon, as it is published in their web site.

4.5 Measurements Definition

4.5.1 Per-flow Metrics

When running on the vantage point, Tstat observes packets, rebuilds each TCP flow, tracks it, and, at the flow end, logs more than 100 metrics [24]. Among the different measurements, in this chapter is consider the server IP address, its FQDN as retrieved by DN-Hunter, the flow RTT, the amount of bytes exchanged at the Application Layer, and the presence of TLS/SSL at the Presentation Layer. These metrics are straight-forward to monitor and we direct the reader to [24] and references therein for more details.

More complicated metrics can be extracted from the log files. In particular, it is defined:

Algorithm 5 Path based IP Geolocator

```
GeoLocation(IP)
$IP$: Target IP Address
$(Names, Delays) \leftarrow traceroute(IP)$
$H \leftarrow sizeof(Delays)$
$TargetDelay \leftarrow Delays[H]$
while $H > 0$ do
    $name \leftarrow Name[H]$
    if $getLocation(name)$ not null then
        /* Return Pivot's location and
        delay difference between target
        and Pivot as measure of confidence */
        $PivotLocation \leftarrow getLocation(name)$
        $PivotDistance \leftarrow TargetDelay - Delays[H]$
        return $(PivotLocation, PivotDistance)$
    end if
    $H \leftarrow H - 1$
end while
return $null$
```

Response Time

it is the time a server employs to start sending the reply after receiving the first request from a client. Let T_{Ack} be the timestamp of the first TCP ACK message sent by server with relative ACK number greater than 1, i.e., acknowledging the reception of some data sent by the client. Let T_{Reply} be the timestamp of the first TCP segment sent by the server and carrying application data.The response time is defined as

$$\Delta R = T_{Reply} - T_{Ack} \tag{4.1}$$

For HTTP flows, it represents an estimation of the time the server takes to elaborate and to transmit the response to the first HTTP request [5] (e.g. an HTTP response). For HTTPS flows, ΔR represents the time taken by the server during the SSL handshake to send the first SSL message.

[5]The response time estimation can be affected by client requests that are longer than 1 TCP segment. It is assumed that these cases are independent from the server, thus they do not bias the comparison.

Flow Goodput

it is defined as the rate at which information generated at Application Layer by the server is delivered to the client. Let T_{First} and T_{Last} be the timestamps of the first and the last packet sent by the server and carrying data. Let D be the size of effective data sent by the server. The server goodput is thus defined as

$$G = \frac{D}{T_{Last} - T_{First}} \tag{4.2}$$

To avoid the bias of short-lived flows and of Persistent-HTTP requests, the server goodput is evaluated only on flows in which the client sent exactly one data packet, and for which $D > 500kB$. Notice that HTTPS flows are automatically filtered out (requiring more than 1 data packet on the client side to complete the SSL handshake).

4.5.2 Network Cost

This metric aims at evaluating the cost sustained by the Internet to transport data generated by AWS servers to the monitored end-users. To this extent, it is defined the Network Cost as the weighted average of the distance traveled by information units. Formally, given a flow, let $b(c,s)$ be the number of Application Layer bytes a client c exchanges with a server s, and let $d(c,s)$ be the distance between client c and server s. The resulting network cost $\beta(s)$ for a given server s is computed as

$$\beta(s) = \frac{\sum_c d(c,s)b(c,s)}{\sum_c b(c,s)}. \tag{4.3}$$

The average network cost of servers in a datacenter $\mathcal{S}$ results

$$\beta = E[\beta(s)|s \in \mathcal{S}]. \tag{4.4}$$

Observe that distance $d(c,s)$ can be defined in different ways, e.g., considering i) the average RTT, ii) the number of traversed AS on the path[6] or iii) the geodetic physical distance, leading respectively to $d^{RTT}(c,s)$, $d^{AS}(c,s)$, $d^{km}(c,s)$. Given these different definitions, we can obtain different network cost metrics β^{RTT}, β^{AS}, β^{km}, respectively.

[6]The number of traversed AS is obtained running a traceroute and checking the AS of returned routers.

Table 4.1: Summary of Amazon's datacenters hosting EC2, S3 services (top) and top 14 CloudFront caches we located (bottom).

	ID	#IPs		Exchanged Data (%)		Avg. RTT [ms]	β^{RTT} [ms]		β^{AS}	β^{km} [km]
		EC2	S3	EC2	S3		EC2	S3		
Datacenters	IAD	6429	121	85.31%	64.22%	132.13	113.97	116.18	3	6709
	DUB	1167	24	12.65%	35.14%	45.10	48.73	43.77	3	1365
	SJC	632	12	1.71%	–	203.06	182.14	174.81	4	9556
	NAR	18	0	–	–	298.67	–	–	4	9843
	SIN	71	0	0.03%	–	235.60	228.10	–	3	10390
	SEA	0	32	–	0.02%	196.04	–	214.79	4	8617
				97.26GB	37.13GB					

	ID	#IPs	Exchanged Data (%)	Avg. RTT [ms]	β^{RTT} [ms]	β^{AS}	$\beta^{km}[km]$
Caches	IAD	2	–	132.13	102.75	3	6709
	DUB	222	0.05%	45.10	49.76	3	1365
	SJC	–	–	–	–	4	9556
	NAR	115	–	298.67	–	4	9843
	SIN	51	–	235.60	–	3	10390
	SEA	64	–	196.04	–	4	8617
	SFO	253	0.83%	172.80	175.21	4	9537
	CDG	246	0.13%	32.09	38.43	3	584
	FRA	245	0.17%	19.56	21.87	2	566
	MXP	232	98.03%	18.34	21.26	3	124
	EWR	208	–	105.28	109.53	3	6380
	AMS	205	0.04%	23.94	29.88	3	837
	LHR	182	0.17%	31.84	31.60	3	920
	ANR	151	0.56%	41.02	41.48	3	1734
			104.19GB				

4.6 Spatial Characterization

This section provides some aggregate information about the spatial distribution of datacenters, the traffic they generated toward monitored end-users, and its cost for the network.

Thanks to Geolocation by Network Pivots Sec. 4.4.3, Tab. 4.1 provides the breakdown of the AWS traffic distinguishing the identified datacenters.

4.6.1 EC2 and S3

The top part of the table reports the list of locations where both EC2, S3 services were detected. Those located in Virginia (IAD), Ireland (DUB) and California (SJC) appear to be the largest datacenters hosting AWS from the point of view of our vantage points placed in Italy.

Several observations hold. Focusing on the number of IP addresses associated to each datacenter, the number of detected IP addresses associated to EC2 service is much larger than to any other service. This is due to the nature of EC2 service itself, that, thanks to virtualization, is capable of allocating, re-sizing and switching on/off independent EC2 instances, in general each provided and reachable by a different public IP address.

S3 and CloudFront, instead, offer storage services that are implemented at application level, and data are made accessible through URI pathnames. The pool of IP addresses needed to keep the service alive is much smaller then, as confirmed by values in Tab. 4.1.

The large unbalance in the number of instances (number of IP addresses in EC2 column) suggests that somehow servers in IAD are the preferred among Amazon customers for providing content to end-users, or the content and services which are popular among European end-users, accidentally, is hosted on that datacenter. This could suggest that allocate services on IAD datacenter is cheaper and that Amazon does not promote that services must be allocated near to the end-users, which in this particular case, are located in Europe. The column reporting the fractions of data generated by EC2 services further confirms this, being the IAD datacenter responsible for generating more than 85% of the total amount of data produced by EC2. It is 7 times higher than the volume handled by the DUB (Dublin) datacenter, the second largest in our ranking.

Interestingly, IAD EC2 (S3) generates more than 80GB (23GB) of data traffic in one day. Considering the user population of the monitored PoP, it is possible to extrapolate that the IAD datacenter serves about 15TB of data per day to the all ISP end-users.

Surprisingly, such large amounts of data are exchanged with such a distant location. Given that Ireland is much closer to Italy than US, indeed, one may expect to be DUB the best candidate to host EC2/S3 instances for serving Italian (and European) end-users. All but β^{AS} network cost metrics, indeed, look sizeable for IAD, from 233% to 491% more expensive than the DUB datacenter. This may suggest that AWS customers, for the sake of a simple management, are more oriented to set their servers on one datacenter. IAD may represent the first choice for AWS customers because of its lower price[7].

AWS offers load-balancing-based forwarders for incoming traffic to enhance performance of instances, but no location-aware policy is offered. Furthermore, recall that EC2 and S3 services are statically allocated to datacenters that are chosen by customers, and no automatic migration policy for instances/objects among datacenter is provided.

This at the expenses of network cost, and, possibly, user experience. Observe how β^{AS} looks comparable for all datacenters, suggesting that Amazon (and the ISP) have good peering agreements with many providers.

Even though Tab. 4.1 presents an overall picture of how EC2 distribute content among end-user, Fig. 4.1 enrich the vision, showing its evolution over time of the volume of data traffic (top) and of the number of flows (bottom) seen from the top

[7] http://aws.amazon.com/ec2/spot-instances/

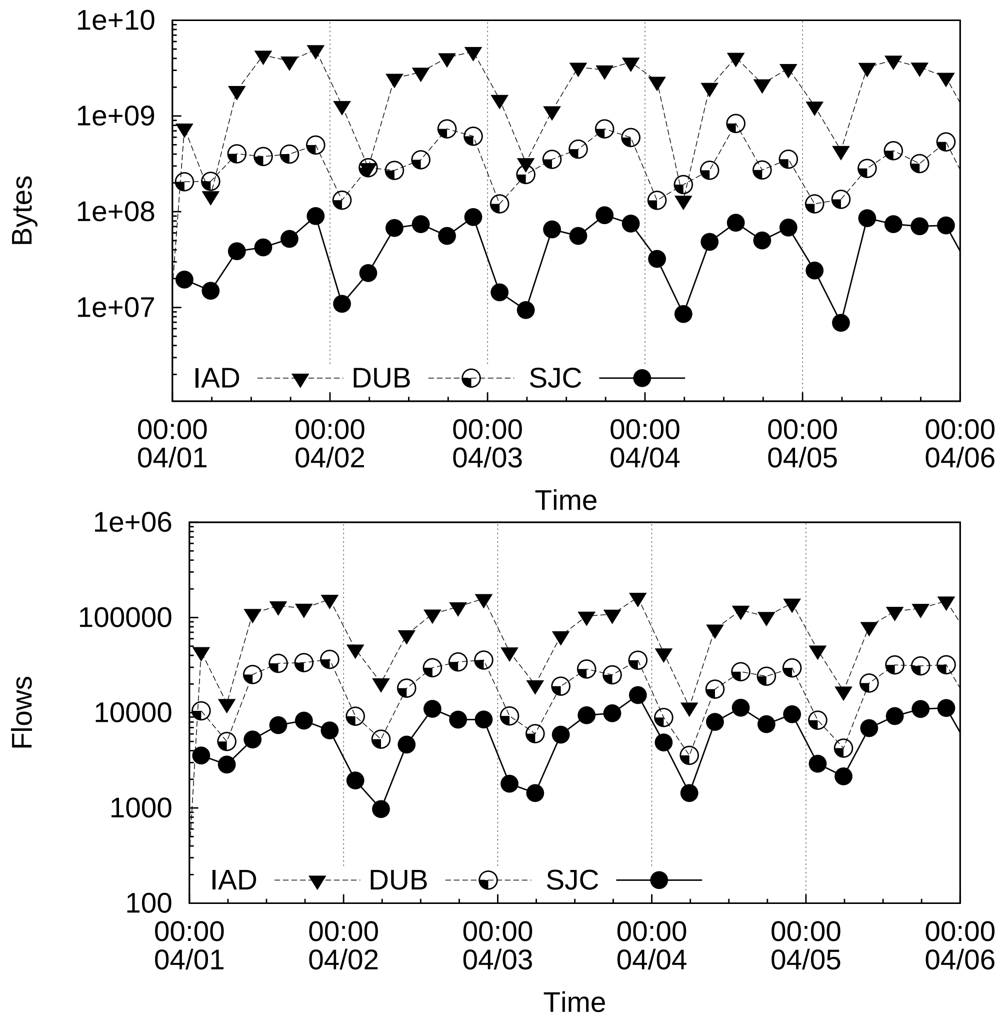


Figure 4.1: Evolution over time of data traffic volume (top) and traffic flows (bottom) for EC2 service.

three EC2 datacenters. One point refers to a 4h long time interval; the first five days of the dataset, starting from Sunday, April 1st, 2012, are reported. Other datasets and periods of time show very similar trends: a very periodic pattern that follows busy period of end-users. IAD datacenter is consistently responsible for handling a much larger amount of traffic with respect to DUB and SJC. This is consistent with values presented in the top part of Table 4.1, and it confirms the static allocation of EC2 instances to datacenters.

Same observation holds for S3 service, as reported in Fig. 4.2. In this case, DUB exchanges an amount of data slightly lower than IAD (notice the log scale that flattens differences).

Comparing the number of flows end-users exchange with EC2 and S3 (bottom

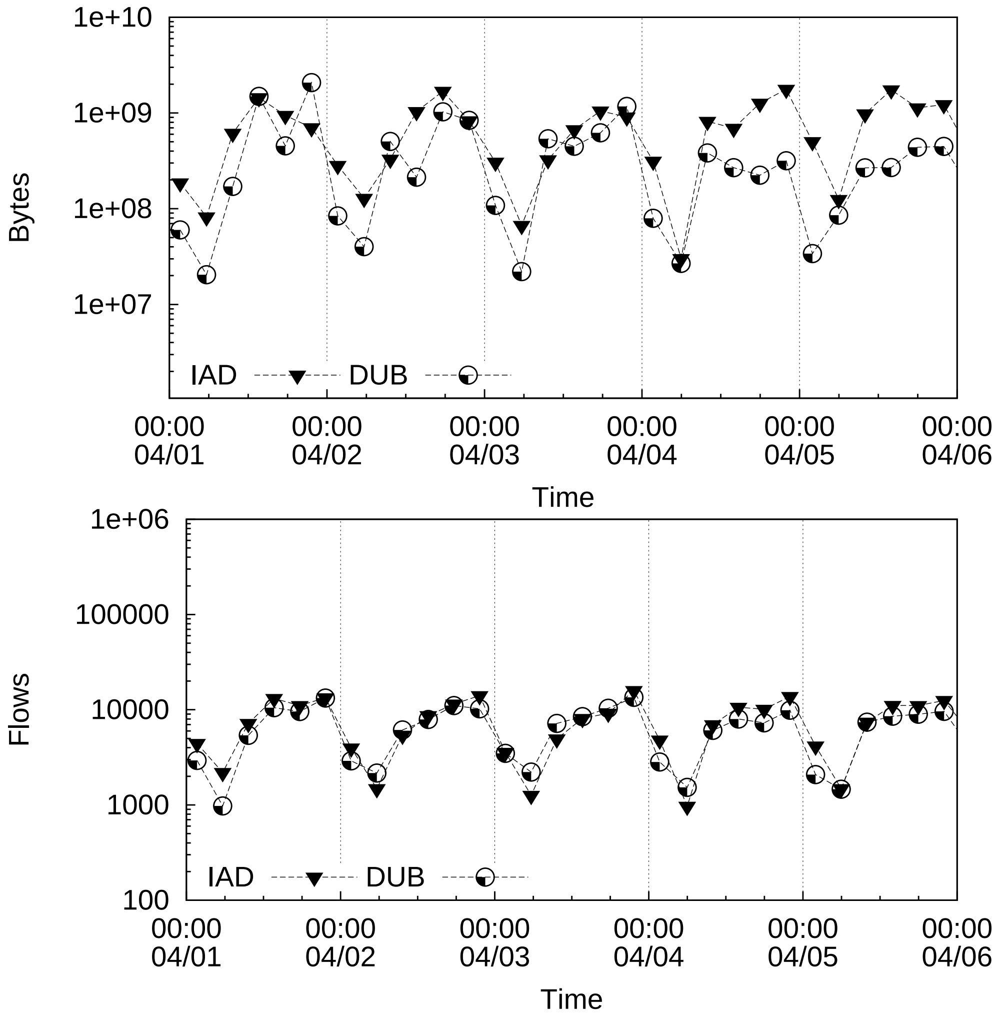


Figure 4.2: Evolution over time of data traffic volume (top) and traffic flows (bottom) for S3 service.

plot of Fig. 4.1 and Fig. 4.2, respectively), it is possible to notice that S3 traffic is made of flows that carry more data than EC2 flows, i.e., more elephants than mice in S3, and vice-versa for EC2. This is confirmed by observing the flow size Cumulative Density Function (CDF), not reported here due to lack of space. Interestingly, comparing the conditional CDF of different datacenters, marginal differences are seen.

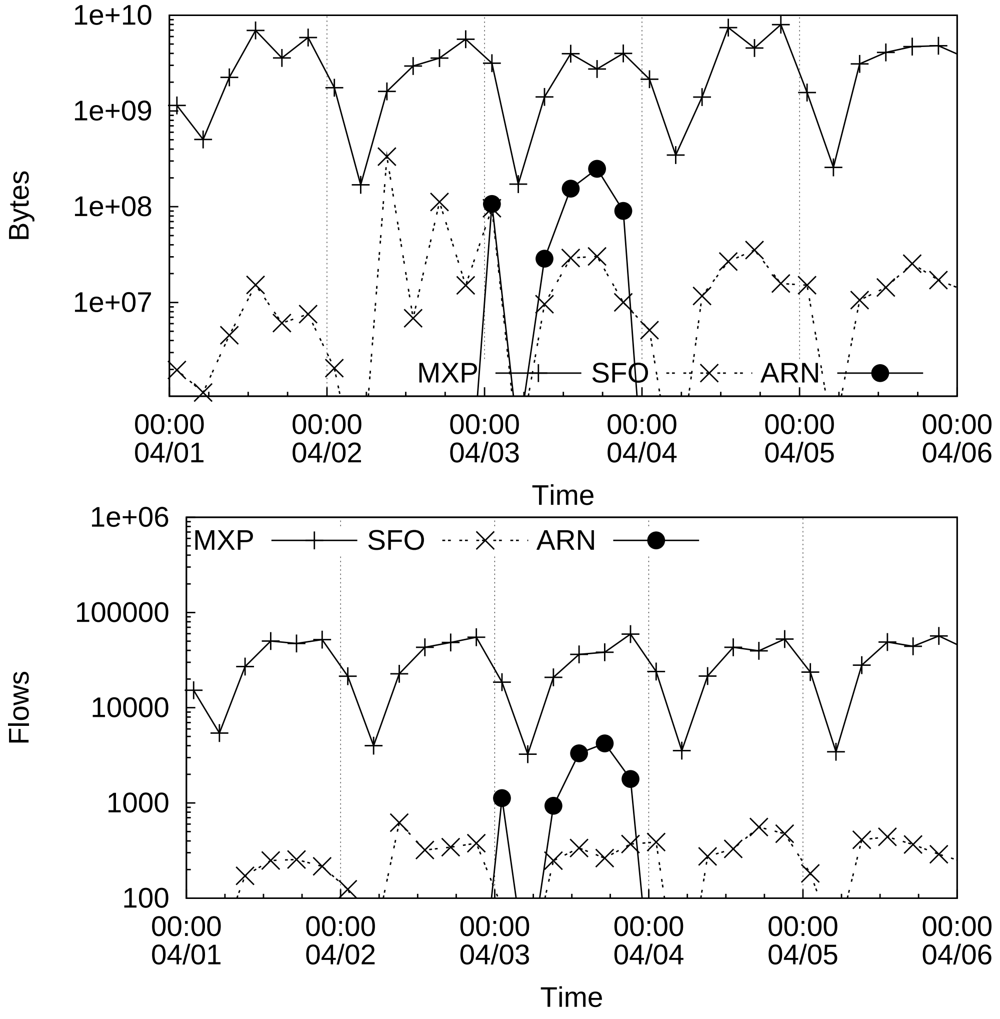


Figure 4.3: Evolution over time of data traffic volume (top) and traffic flows (bottom) for CloudFront service.

4.6.2 CloudFront

Let focus now on CloudFront traffic breakdown, reported in the second part of Tab. 4.1. As expected from normal operation of a CDN, it is observed how biased is the preference toward the MXP cache, located in Milan at less than 200 kilometers from all the monitored end-users. This results to be the best cache considering any definition of network costs. As for most CDN, this confirms that CloudFront relies on DNS load-balancing to direct the user to the closer cache. Simply, whenever a client queries the local DNS server for a CloudFront IP address, the local DNS server forwards the query to the authoritative CloudFront's DNS server, whose reply will direct the client to the geographically closer cache. This happens regardless the type of retrieved content.

This has been confirmed by running an active experiment in which 100 different FQDN are resolved. These FQDN are associated to CloudFront and for resolving more than 2000 DNS servers scattered worldwide were considered. As a side discovery of this process, at least 33 different CloudFront caches were identified, each hosting a /24 subnet. The bottom part of Tab. 4.1 reports the top CloudFront caches whose servers were detected in during the monitoring.

Focusing in the column reporting the location of detected caches, it can be seen that EC2/S3 datacenters host also CloudFront caches, even if ISP end-users are seldom directed to any of these.

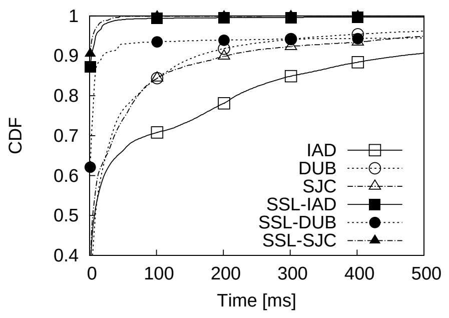
Overall, CloudFront behaves as any normal CDN and performs as expected, allowing cache selection effectively in a way that end-users are directed to the closest cache in Milan. Of course some inefficiencies are observed. Near to 2% of traffic was delivered from caches far away from end-users position. This can be explained by the fact that some end-users could rely on alternative DNS servers different from those provided by their ISP; the Amazon Authoritative DNS would reply directing traffic to caches located close to the used DNS server, but eventually far from the ISP. For instance, both OpenDNS and Google's DNS servers causes requests from the ISP end-users to be directed to FRA, since their servers are located in Amsterdam and Frankfurt respectively (confirmed using the Pivot Geolocation Technique presented in Sec. 4.4.3). This is consistent with findings in [22].

Fig. 4.3 reports the evolution over time of the volume of data traffic (top) and of the number of flows (bottom) for the top three caches, i.e. MXP, SFO and ANR. Regular patterns are present for caches placed in Milan and San Francisco. However this does not hold for ANR, in Stockholm, where it presents an unusual peak on the third day of measurements, precisely from 10pm of April 2 to 6pm of April 3. Investigating further, it has been verified that this was not due to some unusual DNS end-users' setting, but to an intentional change in the Amazon DNS policies. Indeed, many end-users and contents that were typically available from MXP had been redirected to ARN during that period. This is similar to what has been observed for YouTube CDN [21].

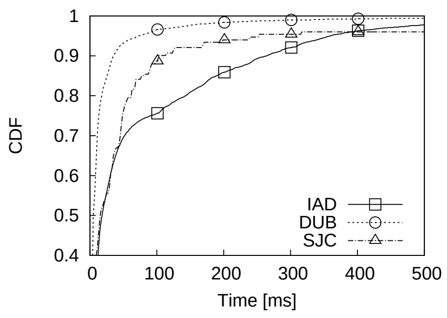
While it is impossible to know why this happens, it is observed that CloudFront policies are dynamic, in contrast with the static allocation of the EC2/S3 content.

4.7 Content Analysis

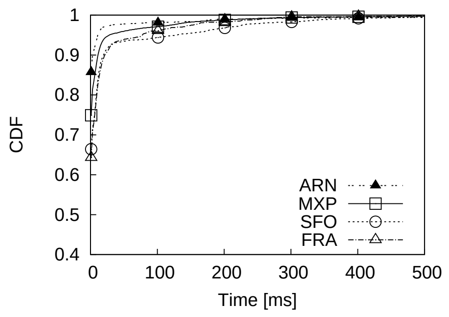
Regarding the type of content hosted by the 3 different infrastructures, Fig. 4.5 depicts how different are the objects associated to AWS. Around 50% of objects observed on CloudFront presents a size smaller than 10kB, mostly probably being CSS or JavaScript files employed for the rendering of web pages. Other 20% of contents, which present a size larger than 100kB, is binary data, e.g., Flash objects



(a) ΔR EC2



(b) ΔR S3



(c) ΔR CF

Figure 4.4: Distribution of response time for AWS infrastructure.

or images. Differences between sizes of contents served by different caches are negligible. Files hosted on S3 show in general the same size distribution of files hosted by CloudFront. The average size of contents distributed by CloudFront and S3 is 78kB for both. Flows directed to EC2 carry smaller data in general, being the 60% of them smaller than 1kB. These could be small XML files or messages directed to APIs. For these files, the TCP three-way-handshake and tear-down procedures last longer than the data transfer. Even for EC2, no significant difference among different locations is evident. The average size of contents served by EC2 flows is 26kB.

Table 4.2: The top contents hosted by EC2 by number of flows or volume.

IAD			DUB		
Service	%Flows	%Vol.	Service	%Flows	%Vol.
zynga	14	5	wooga	24	4
farmville	13	6	invitemedia	18	3
playfish	9	3	cdn.com	6	0.2
widdit	6	1	360yield	6	0.1
chartbeat	3	>0.1	mydlink	2	>0.1
dropbox	0.4	59	wetransfer	>0.1	16

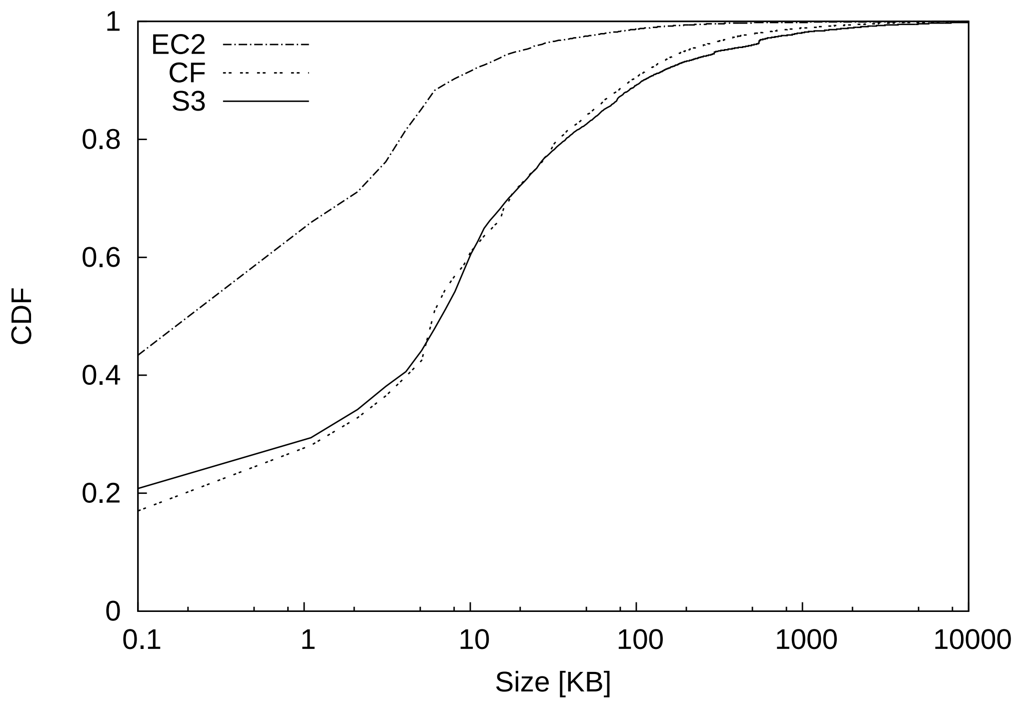


Figure 4.5: Distribution of size of content/flow.

In Tab. 4.2 it is reported the most popular kind of contents served by EC2, i.e., those contents which are generating the largest number of flows, or volume. The type of content/service is quite heterogeneous, with the larger portion of flows generated by social games, such as Zynga, Farmville, Wooga and DigitalChocolate, and by advertising companies, e.g., InviteMedia, 360yield. Notice how none of these services seem to be replicated among different datacenters, that is their traffic is coming exclusively from a single datacenter. The only notable service found to be present on both IAD and SJC datacenters is DigitalChocolate; interestingly, digitalchocolate relies on IAD for the distribution and execution of games, while it relies on SJC for the management of system logins and subscriptions. This level of detail of the analysis is reached observing the less hierarchical domains associated to digitalchocolate and running on EC2.

Considering the top services in terms of volume, the important presence of storage services like Dropbox and WeTransfer. The former is responsible for 59% of volume of data handled by EC2 alone, and it is only available from IAD!

Contents hosted by S3 presents similar properties of EC2 (Tab. 4.3). S3 contents are in general services for storage, advertisement and social games, but even in this case none of them results distributed over different datacenters.

Table 4.3: The 5 services hosted by S3 generating the largest amounts of data traffic.

IAD		DUB	
Service	%	Service	%
s3.amazonaws	24	wetransfer	33
castlemania-productions	11	s3-3-w.amazonaws	20
uploadus	5	adagiobanner	13
dotandad	4	village-it	4
softonic	3	fanpage	3

4.8 Performance Evaluation of AWS

Throughout this section, an evaluation of the performance of AWS based on datacenter and content is performed using information obtained from completely passive monitoring.

4.8.1 Datacenters and Caches Performance Evaluation

Fig. 4.4 depicts the distribution of the estimated response time ΔR for EC2, S3 and CloudFront on left, center and right plot, respectively. Top popular datacenters are shown. Data refer to a single day of April 2012.

Focusing on the performance of different locations, EC2 in IAD shows values of ΔR larger than 100ms in 30% of the cases, resulting the worst performing datacenter from the point of view of Italian end-users. These plots combined with information about the large number of IPs allocated on IAD might suggest that popular instances hosted in IAD are overloaded because of the large number of hosted EC2 instances, or the instances hosted there are low profile in terms of processing power. Hence the average bad performance of IAD could be caused by popular and poorly performing contents running on congested instances.

DUB appears to be the best choice among datacenters for S3, while it competes with SJC in the case of EC2.

Since a sizable part of content hosted on EC2 is encrypted using TLS/SSL (14% of flows), it is reported the response time for HTTPS flows in left Fig. 4.4. Recall that ΔR is a measure of the server reactiveness in the SSL handshake in this case. Both IAD and SJC look very reactive, with 93% of the cases responding to SSL initial negotiation almost with no delay. This conflicts with the hypothesis of IAD being congested. On the other hand, for the 10% of the flows, DUB presents large values of ΔR. It is found out that these impairing in the performance is related to a single application or content hosted on that datacenter. This application is called *proxy.eu.mydlink.com* and flows directed to it suffer from more than 10 seconds of response time.

This finding supports the idea that in general it is not the entire EC2 datacenter to be congested, but rather some instances running on it.

Additionaly, Fig. 4.6 complements the findings already presented, reporting the evolution over time of $E[\Delta R]$ for EC2 for a period of one day for IAD and DUB. From the figure it is observed that IAD consistently performing worse on average than DUB. Notice that the average is i) a strongly non-stationary measure (being it biased by the different contents retrieved at different times), and ii) practically independent on the datacenter load.

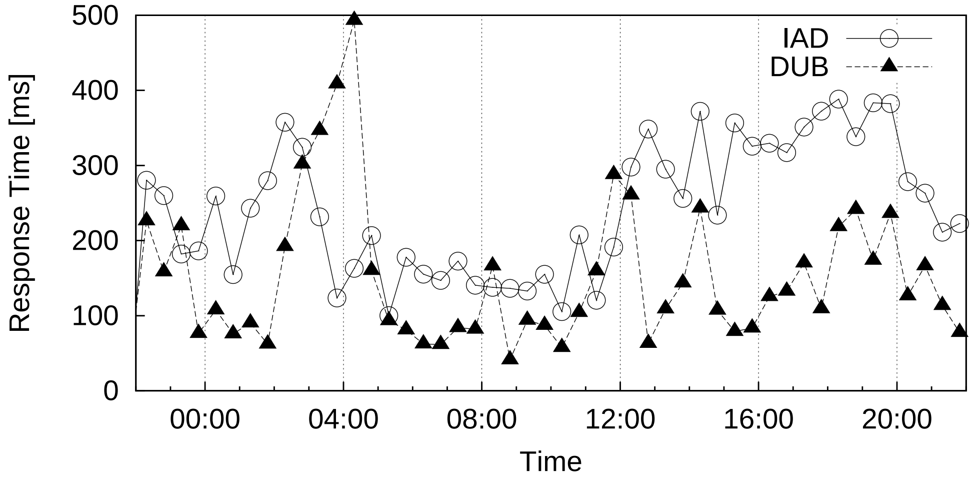
Now considering the performance of the AWS CDN, depicted in Fig. 4.4 shows in general very good performance, being 83% of requests satisfied in less than 20ms in the worst case, i.e., FRA. MXP and ARN caches elaborate 80% of requests in less than 3ms; SJC and FRA serve only 65% and 55% of request in less than 3ms, respectively.

Fig. 4.7 compares the distributions of goodput G. Here, it is compared the performance of two main datacenters for S3, IAD and DUB, together with CloudFront MXP cache. The plot shows that more than 50% of flows get a goodput larger than 2Mbit/s for S3 in DUB and CloudFront in MXP. For S3 in IAD, only 21% of flows can achieve a goodput larger than 2Mbit/s. This difference can be due to the larger RTT running from our vantage point to IAD, that affects the TCP congestion control, thus, reducing achievable goodput.

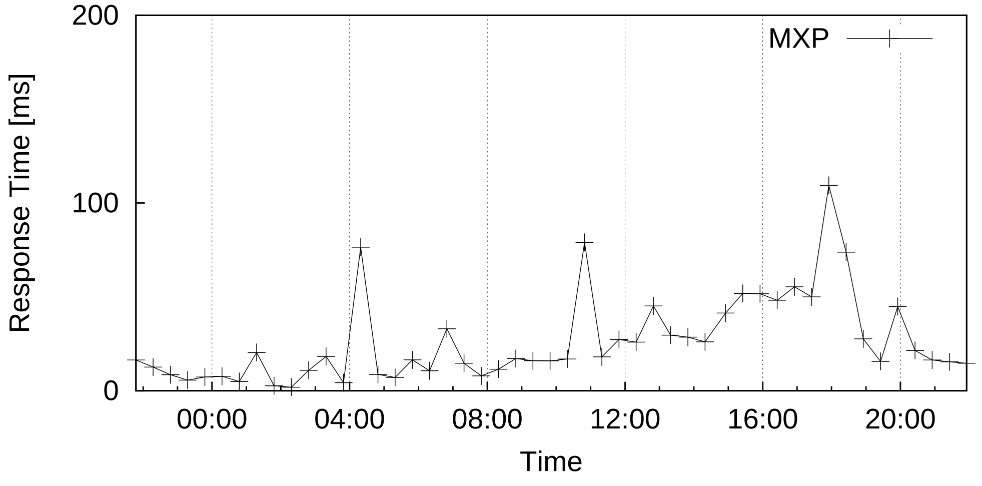
4.8.2 Per-content Performance Evaluation

Fig. 4.8 reports the distribution of the response time ΔR for different social gaming services hosted on different datacenters. Notice that all social games hosted by IAD present poor performance with respect to those hosted by DUB and SJC. This suggests again that IAD datacenter suffers somehow from congestion or the rented instances for performing game tasks are of lesser processing power. Observe how the instances of Farmville, a popular game, are indeed performing poorly. Notice that, this lack in performance could make the service not suitable for real-time gaming for instance, which requires broadcasting of data among players with as less latency as possible.

Congestion may affect single instances. For example, Fig. 4.9 reports two examples of applications hosted by EC2 in DUB that suffer large average ΔR. These two applications, SamsungMobile and MyDlink, show really impaired performance, with average response time higher than 2s. Recall that DUB is the best performing datacenter in our measurements (Fig. 4.4). This suggests that such poor performance is due to a bad dimensioning of the instance or bad software design, and not due to datacenter issues.



(a) ΔR EC2



(b) ΔR CF

Figure 4.6: Changes of response time over the day on different datacenters.

At last, it is shown the impaired performance following the outage of IAD datacenter on June 30th, 2012[8]. Selecting two contents hosted at IAD, and a third one at DUB, Fig. 4.10 shows that i) not all contents where affected by the IAD outage, ii) it had no impact on DUB, iii) affected instances suffer a 100 fold worse performance during the failure, and iv) they kept suffering for performance issues for several hours after the fault.

Focusing on the performance of CloudFront service, in Fig. 4.11 it is reported

[8]http://aws.amazon.com/message/67457/

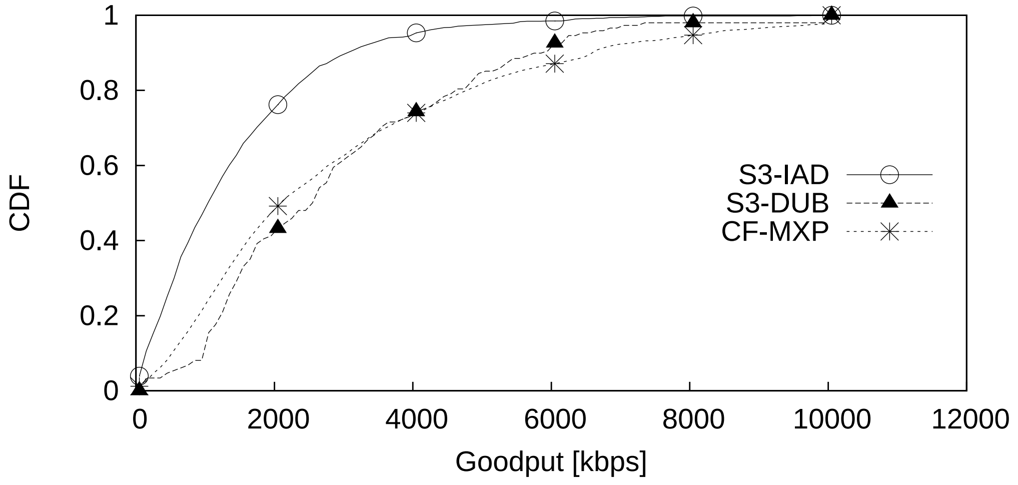


Figure 4.7: Cumulative distribution function of goodput G for the two most used S3 datacenter, IAD and DUB, and for MXP CloudFront cache.

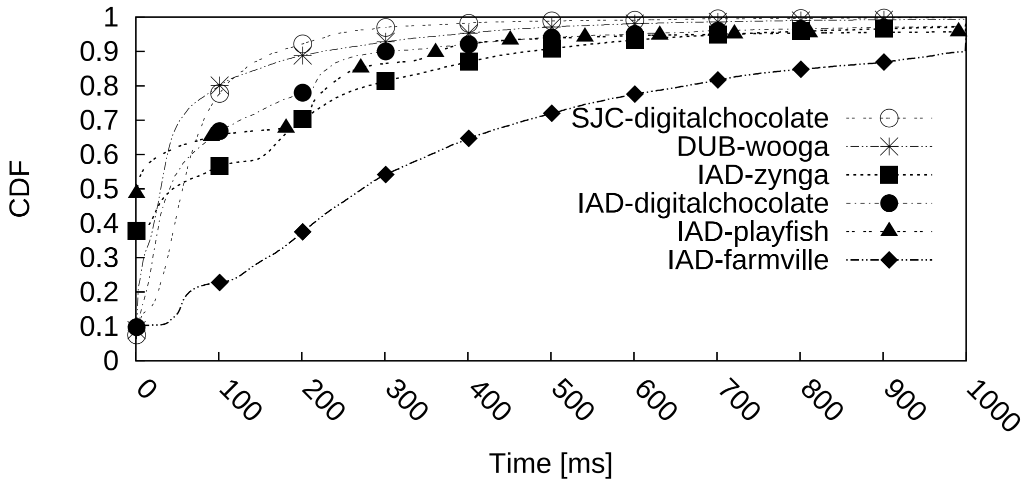


Figure 4.8: Distribution of response time ΔR for EC2 social gaming services.

the distribution of ΔR for different kinds of contents that end-users downloaded from MXP cache. *Static* refers to static content for web pages (e.g. HTML files), *js* represents JavaScript files, *img* refers to binary data such as images and *Instagram* is referred to contents related to the well-known photo-sharing service. *Aggregate* reports the behavior of all services together. As previously noticed, CloudFront shows really good performance, being able to process 50% of requests in less than 2ms, independently on the kind of content.

However, ΔR is consistently better on average for static and JavaScript files, whereas images and Instagram contents show larger response time. This may be

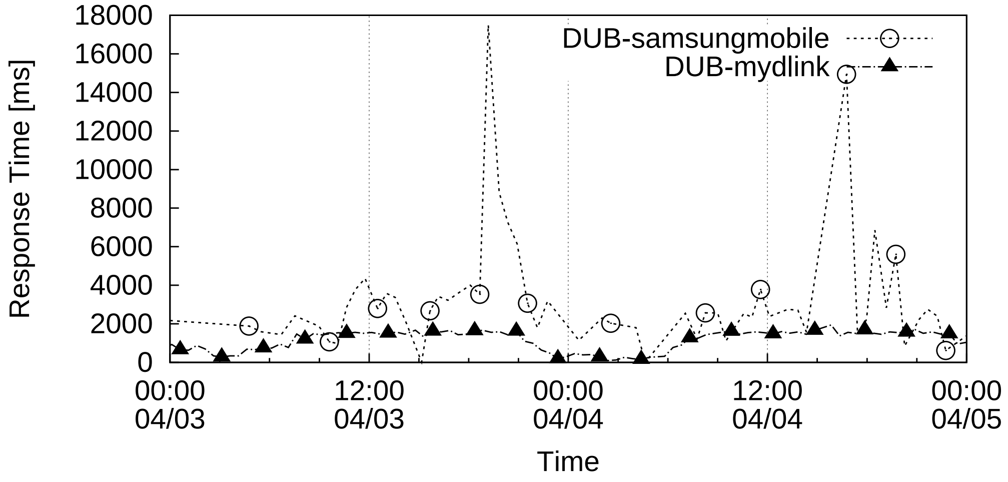


Figure 4.9: Evolution during time of ΔR for two poorly performing contents hosted on EC2.

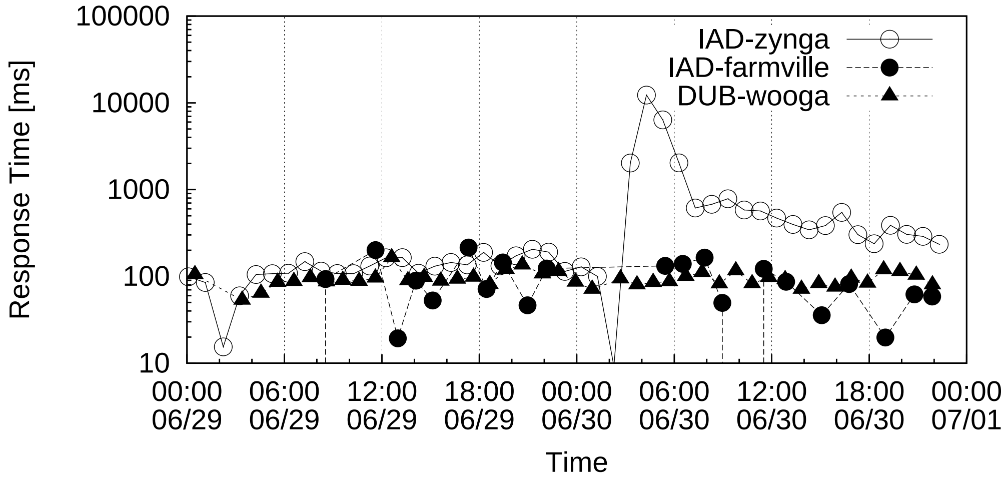


Figure 4.10: Evolution over time of ΔR during the IAD outage.

due to the nature of the user-generated contents that are the most critical to manage for content delivery services, because of the size of the catalog, and of the small popularity of each single content [26]. Finally, SSL flows show excellent ΔR, suggesting that the cache, the path and the peering points are not congested.

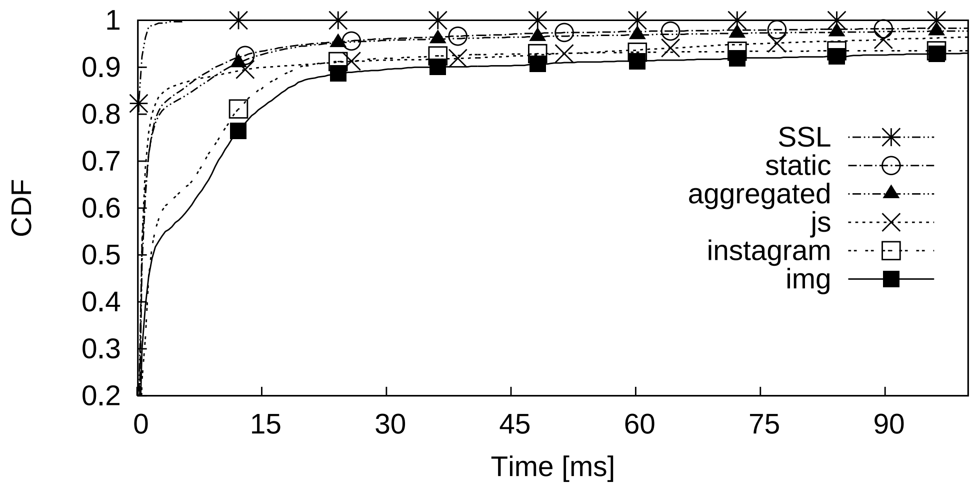


Figure 4.11: Distribution of response time ΔR for different kinds of contents provided by CloudFront cache located in MXP.

4.9 Conclusions

This chapter presents the first study of the traffic of *Amazon Web Services* (AWS) in the wild on the Internet. This analysis assessed the growth of interest for cloud-based platforms, which, thanks to virtualization technology, represents a scalable and inexpensive solution for many web companies.

Through the study of traces captured from live networks, it was confirmed that AWS represents a big player in nowadays Internet, being responsible for the generation of a sizeable portion of traffic.

It was shown an extensive characterization of AWS offerings, in particular for EC2, S3 and Amazon's CDN CloudFront. Results presented in this chapter show that there is a big workload unbalance among different datacenters hosting both EC2 and S3 products; in particular, the datacenter in Virginia was responsible for 85% of the total traffic sent to Italian end-users and it handled seven times the traffic served by the datacenter in Ireland. Companies which relied on EC2 and S3 tended to concentrate their content on one datacenter, with the drawbacks of i) increasing the cost sustained by the network to carry data to faraway end-users and, ii) increasing risk in case of failures. It was evaluated the performance of contents hosted by EC2 and S3, in terms of servers reactiveness and network goodput, providing comparison among datacenters and among different contents. The results shown that the datacenter in Virginia presented in general poorer performance, but was difficult to identify if it was due to an actual overloading caused by the large population of EC2 instances, to congestion or under-dimensioned instances offering particular content.

It is found that CloudFront performed as any other CDN, being able to serve 98% of traffic to the best possible cache. However, it presented issues that are typical of CDN systems: i) generic DNS servers returning caches far from end-users, that lower perceived QoS and system's efficiency; ii) lower performance when processing unpopular user-generated contents.

Chapter 5

Conclusions

During these three years of Internet traffic analysis, I observed changes on the way the content or services are delivered to end-users. People studying Internet traffic should realized a change in the paradigm on how content is delivered to end-users. There was and increase of encrypted traffic over TLS/SSL and the extensive usage of WEB 2.0, that made quite hard unveiling what was already unveiled. In spite of this, it was presented DN-Hunter, which brings a little light into the visibility of the things happening in the network, providing clues about the content or service associated to a network flow. In this thesis were presented very particular heuristics to understand traffic collected passively that somehow was obscured, getting very detailed information about the protocols and infrastructures studied. However, these heuristics presented do not scale up. An open research could be extending these, formalizing and automatizing these processes for having a better picture of the evolution of the Internet as seen from the viewpoint of an ISP. In this context I would like to recognize the work started on the mPlane Seventh Framework Programme project, which aims to build an "intelligent measurement plane for the Internet".

Bibliography

[1] Bamba Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. Constraint-based Geolocation of Internet Hosts. In *Proceedings of the 4th ACM SIGCOMM on Internet Measurement Conference*, IMC '04, pages 288–293, New York, NY, USA, 2004. ACM.

[2] D. Ciullo, M.A. Garcia, A. Horvath, E. Leonardi, M. Mellia, D. Rossi, M. Telek, and P. Veglia. Network Awareness of P2P Live Streaming Applications: A Measurement Study. *Multimedia, IEEE Transactions on*, 12(1):54–63, 2009.

[3] http://www.maxmind.com. Geoip Database.

[4] H. Xie, Y.R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz. P4P: Provider portal for (P2P) applications. In *Proc. of ACM SIGCOMM*, 2008.

[5] E. Leonardi, M. Mellia, A. Horvath, L. Muscariello, S. Niccolini, and D. Rossi. Building a cooperative P2P-TV application over a wise network: the approach of the European FP-7 strep NAPA-WINE. *Communications Magazine, IEEE*, 46(4):20–22, 2008.

[6] L. Lao, C. Dovrolis, and MY Sanadidi. The probe gap model can underestimate the available bandwidth of multihop paths. *ACM SIGCOMM Computer Communication Review*, 36(5):29–34, 2006.

[7] V. Gehlen, A. Finamore, M. Mellia, and M. Munafò. Uncovering the big players of the web. In *Traffic Monitoring and Analysis workshop, COST-TMA*, pages 15–28, 2012.

[8] M. Mellia, R. Lo Cigno, and F. Neri. Measuring IP and TCP behavior on edge nodes with Tstat. *Computer Networks*, 47(1):1–21, 2005.

[9] The DNS Security Extensions, February 2011.

[10] Introducing DNSCrypt (Preview Release), February 2011.

[11] B. Ager, H. Dreger, and A. Feldmann. Predicting the DNSSEC Overhead using DNS Traces. In *40th Annual Conference on Information Sciences and Systems*, pages 1484–1489. IEEE, 2006.

[12] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A Break in the Clouds: Towards a Cloud Definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, December 2008.

[13] Alessandro Finamore, Vinicius Gehlen, Marco Mellia, Maurizio Munafò, and

Saverio Nicolini. The Need for an Intelligent Measurement Plane: the Example of Time-Variant CDN Policies. In *IEEE Networks*, pages –, October 2012.

[14] Edward Walker. Benchmarking Amazon EC2 for High-Performance Scientic Computing. *USENIX ;login: Magazine*, 2008.

[15] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good. The Cost of Doing Science on the Cloud: The Montage Example. In *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1 –12, nov. 2008.

[16] Simson L. Garfinkel. An Evaluation of Amazon's Grid Computing Services: EC2, S3, and SQS. Technical report, Center for Research on Computation and Society, School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA, 2007.

[17] Guohui Wang and T.S.E. Ng. The Impact of Virtualization on Network Performance of Amazon EC2 Data Center. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, March 2010.

[18] Sipat Triukose, Zhihua Wen, and Michael Rabinovich. Measuring a Commercial Content Delivery Network. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 467–476, New York, NY, USA, 2011. ACM.

[19] Minas Gjoka, Michael Sirivianos, Athina Markopoulou, and Xiaowei Yang. Poking Facebook: Characterization of OSN Applications. In *Proceedings of the First Workshop on Online Social Networks*, WOSN '08, pages 31–36, New York, NY, USA, 2008. ACM.

[20] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and Analysis of Online Social Networks. In *Proceedings of the 7th ACM SIGCOMM on Internet Measurement Conference*, IMC '07, pages 29–42, New York, NY, USA, 2007. ACM.

[21] Alessandro Finamore, Marco Mellia, Maurizio M. Munafò, Ruben Torres, and Sanjay G. Rao. YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience. In *Proceedings of the 2011 ACM SIGCOMM on Internet Measurement Conference*, IMC '11, pages 345–360, New York, NY, USA, 2011. ACM.

[22] Bernhard Ager, Wolfgang Mühlbauer, Georgios Smaragdakis, and Steve Uhlig. Comparing DNS Resolvers in the Wild. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 15–21, New York, NY, USA, 2010. ACM.

[23] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the Art of Virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5):164–177, October 2003.

[24] Alessandro Finamore, Marco Mellia, Michela Meo, Maurizio M. Munafò, and Dario Rossi. Experiences of Internet Traffic Monitoring with Tstat. *IEEE*

Network, 25(3):8–14, 2011.

[25] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. IP Geolocation Databases: Unreliable? *SIGCOMM Comput. Commun. Rev.*, 41(2):53–56, April 2011.

[26] B. Ager, F. Schneider, Juhoon Kim, and A. Feldmann. Revisiting cacheability in times of user generated content. In *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, pages 1 –6, march 2010.