Universidad
Pública de Navarra
Nafarroako
Unibertsitate Publikoa

E.T.S. de Ingenieros Industriales y de Telecomunicación
Departamento de Automática y Computación

Tesis doctoral:

Web browsing interactions inferred from a flow-level perspective

Autor: Luis Miguel Torres García
Director: Dr. Eduardo Magaña Lizarrondo

Pamplona, Marzo de 2015

TESIS DOCTORAL:	Web browsing interactions inferred from a flow-level perspective
AUTOR:	Luis Miguel Torres García
DIRECTOR DE TESIS:	Dr. Eduardo Magaña Lizarrondo

El tribunal para la defensa de esta tesis está formado por:

PRESIDENTE:	Dr. Javier Aracil Rico
SECRETARIO:	Dr. Mikel Izal Azcárate
VOCALES:	Dr. Jorge E. López de Vergara Méndez
	Dr. Francisco José Gómez Arribas (suplente)
REVISORES EXTERNOS:	Dr. József Stéger
	Dr. Juan Jose Unzilla Galán
	Dr. Ignacio de Miguel Jiménez (suplente)

Resumen

Desde que su uso se extendiera a mediados de los noventa, la web ha sido probablemente el servicio de Internet más popular. De hecho, muchos usuarios la utilizan prácticamente como sinónimo de Internet. Hoy en día los usuarios de la web utilizan una gran cantidad dispositivos distintos para acceder a ella desde ordenadores tradicionales a teléfonos móviles, tabletas, lectores de libros electrónicos o, incluso, relojes inteligentes. Además, los usuarios se han acostumbrado a acceder a diferentes servicios a través de sus navegadores web en vez de utilizar aplicaciones dedicadas a ello. Este es el caso, por ejemplo del correo electrónico, del *streaming* de vídeo o de suites ofimáticas (como la proporcionada por Google Docs). Como consecuencia de todo esto, hoy en día el tráfico web es muy complejo y el efecto que tiene en las redes es muy importante. La comunidad científica ha reaccionado a esta situación impulsando muchos estudios que caracterizan la web y su tráfico y que proponen maneras de mejorar su funcionamiento.

Sin embargo, muchos estudios centrados en el tráfico web han considerado el tráfico de los clientes o los servidores en su totalidad con el objetivo de describirlo estadísticamente. En otros casos, se han introducido en el nivel de aplicación al centrarse en los mensajes HTTP. Pocos trabajos han buscado describir el efecto que las sesiones de un sitio web y las visitas a páginas web tienen en el tráfico de un usuario. No obstante, esas interacciones son las que el usuario experimenta al navegar y, por tanto, son las que mejor representan su comportamiento. El trabajo que se presenta en esta tesis gira alrededor de esas interacciones y se enfoca especialmente en identificarlas en el tráfico de los usuarios.

Esta tesis aborda el problema desde una perspectiva a nivel de flujo. En otras palabras, el estudio que se presenta se centra en una caracterización del tráfico web obtenida para cada conexión mediante datos de los niveles de transporte y red, nunca mediante datos de aplicación. La perspectiva a nivel de flujo introduce ciertas limitaciones en las propuestas desarrolladas, pero lo compensa al permitir desarrollar sistemas escalables, fáciles de instalar en cualquier red y que evitan acceder a información de usuario que podría ser sensible.

En los capítulos de este documento se introducen varios métodos para identificar sesiones a sitios web y descargas de páginas web en el tráfico de los usuarios. Para desarrollar dichos métodos se ha caracterizado tráfico web capturado de varias formas: accediendo a páginas automáticamente, con la ayuda de voluntarios en un entorno controlado y en el enlace de la Universidad Pública de Navarra. Los métodos que presentamos se basan en parámetros a nivel de conexión como los tiempos de inicio y final de los flujos o las direcciones IP de servidor. Estos parámetros se emplean para encontrar conexiones relacionadas en el tráfico de los usuarios.

La validación de los resultados obtenidos con los distintos métodos ha sido complicada al no disponer de trazas etiquetadas correctamente que puedan usarse para verificar que las clasificaciones se han realizado de forma correcta. Además, al no haber propuestas similares en la literatura científica ha sido imposible comparar los resultados obtenidos con los de otros autores. Por todo esto ha sido necesario diseñar métodos específicos de validación que también se describen en este documento.

Ser capaces de identificar sesiones a sitios web y descargas de páginas web tiene aplicaciones inmediatas para administradores de red y proveedores de servicio ya que les permitiría recoger datos sobre el perfil de navegación de sus usuarios e incluso bloquear tráfico indeseado y dar prioridad al importante. Además, las ventajas de trabajar a nivel de conexión se aplican especialmente en su caso. Por último, los resultados obtenidos a través de los métodos presentados en esta tesis podrían emplearse en diseñar esquemas capaces de clasificar el tráfico web dependiendo del servicio que lo haya producido ya que se podrían utilizar como parámetros de entrada las características de múltiples conexiones relacionadas.

Web browsing interactions inferred from a flow-level perspective

HC SVNT DRACONES

Inscription on the Hunt-Lenox Globe (ca. 1510)

Summary

Since its use became widespread during the mid 1990s, the web has probably been the most popular Internet service. In fact, for many lay users, the web is almost a synonym for the Internet. Web users today access it from a myriad of different devices from traditional computers to smartphones, tablets, ebook readers and even smartwatches. Moreover, users have become accustomed to accessing multiple different services through their web browsers instead of through dedicated applications. This is the case, for example, of e-mail, video-streaming or office suites (such as the one provided by Google Docs). As a consequence, web traffic nowadays is complex and its effect on the networks is very important. The scientific community has reacted to this providing many works that characterize the web and its traffic and propose ways of improving its operation.

Nevertheless, studies focused on web traffic have often considered the traffic of web clients or servers as a whole in order to describe their particular performance, or have delved into the application level by focusing on HTTP messages. Few works have attempted to describe the effect of website sessions and webpage visits on web traffic. Those web browsing interactions are, however, the elements of web operation that the user actually experiences and thus are the most representative of his behavior. The work presented in this thesis revolves around these web interactions with the special focus of identifying them in user traffic.

This thesis offers a distinctive approach in that the problem at hand is faced from a flow-level perspective. That is, the study presented here centers on a characterization of web traffic obtained on a per connection basis and using information from the transport and network levels rather than relying on deep packet inspection. This flow-level perspective introduces various constraints to the proposals developed, but pays off by offering scalability, ease of deployment, and by avoiding the need to access potentially sensitive application data.

In the chapters of this document, different methods for identifying website sessions and webpage downloads in user traffic are introduced. In order to develop those methods, web traffic is characterized from a connection perspective using traces captured by accessing the web automatically, with the help of voluntary users in a controlled environment, and captured in the wild from users of the Public University of Navarre. The methods rely on connection-level parameters such as start and end timestamps or server IP addresses in order to find related connections in the traffic of web users.

Evaluating the performance of the different methods has been problematic because of the absence of ground truth (labeled web traffic traces are hard to obtain and the labeling process is very complex) and the lack of similar research which could be used for comparison purposes. As a consequence, specific validation methods have been designed and they are also described in this document.

Identifying website sessions and webpage downloads in user traffic has multiple immediate applications for network administrators and Internet service providers as it would allow them to gather additional insight into their users browsing behavior and even block undesired traffic or prioritize important one. Moreover, the advantages of a connection-level perspective would be specially interesting for them. Finally, this work could also help in research directed to classifying thee services provided through the web as grouping the connections related to the same website session may offer additional information for the classification process.

Acknowledgments

First of all I would like to express my gratitude to my supervisor, Dr. Eduardo Magaña who has been a constant source of support and encouragement during the work on this thesis. I am fortunate that he has always found time for our frequent meetings in which he has shared his knowledge and expertise. His guidance has been truly invaluable. I would also like to thank Dr. Mikel Izal and Dr. Daniel Morató for their availability, help and useful advice during these years.

I acknowledge that this research would not have been possible without the funding of the Government of Navarre (through its doctoral fellowship program) and the collaboration of the Public University of Navarre and its department of Automatics and Computation. I express my gratitude to those agencies.

On a personal note I must, of course, thank my parents for their constant support not only during these last five years, but during the previous twenty-five. I would not be here without you. Thanks. [1]

My coworkers at the GRRST group (Iria, Olga, Santi, Félix and many more that have been here through the years) are responsible for the wonderful work environment that I have enjoyed while writing this thesis. Even though I never seemed to want to get coffee with them, I am truly thankful for their help, sympathy and laughs.

I would like to give credit to the guys and girls of *Jazzy Leap* for giving me a great reason to keep working until late in the evenings and to the H4/B3 group

[1]Desde un punto de vista más personal quiero agradecer a mis padres su apoyo constante no sólo durante estos últimos cinco años sino durante los anteriores venticinco. No estaría aquí sin vosotros. Gracias

for providing a needed solace after a hard day’s work.

Last, but definitely not least, I would like to thank Mikel, Coro, Mikel, Jesús and Victoria —friends with whom I have shared the better part of these years, both at the university and outside of it— for being their stupendously awesome selves.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Since its inception, the web has received considerable attention from the scientific community which has worked towards describing, testing and improving many aspects of its operation. In this vein, web traffic has been thoroughly characterized, as it deserves by accounting for a substantial share of all Internet traffic. However, this characterization has often considered the traffic of web clients or servers as a whole in order to describe their particular performance, or has delved into the application level by focusing on HTTP messages. Nevertheless, the choices a user makes while browsing the web, such as visiting certain websites and navigating through the webpages that constitute them, have an obvious effect on said user's traffic. Less effort has been put into studying these web browsing interactions although being able to identify them in the traffic of the users would have multiple applications. This thesis focuses on these interactions and does so from a flow-level perspective. That is, the study presented here centers on a characterization of web traffic obtained on a per connection basis and using information from the transport and network levels rather than relying on deep packet inspection. This flow-level perspective introduces various constraints to the proposals we develop but pays off by offering scalability, ease of deployment, and by avoiding the need to access potentially sensitive application data.

1.1 Background

The web is probably the classic Internet application that has grown and evolved the most during the past two decades. The simple and mostly static webpages of the 1990s have given way to much more complex sites. This complexity is represented, in the first place, by the addition of a wide variety of content types (such as videos or interactive media) to the text and images that classic webpages traditionally hosted. Nevertheless, modern websites not only offer these new content types, but they do so in a dynamic way, keeping their content current and tailoring their offer to each specific visitor.

The network requirements introduced by all this and the ever-increasing popularity of the web have also pushed for improvements in the web application protocols and the development of new techniques, like content distribution networks (CDNs) or analytics services, that help in its operation. As a consequence, the web has achieved a remarkable flexibility which allows it to provide a huge range of different services aside from traditional web browsing. Services such as e-mail, video streaming, on-line games or e-learning are, in a lot of cases, provided through the web, taking advantage of the facts that web clients (browsers) are present in almost any network-enabled device and that web traffic usually faces few network restrictions.

All these changes have obviously affected the profile of web traffic with recent studies showing that its characteristics have greatly changed from the (simpler) ones described in the 1990s. This is partially the result of the introduction of HTTP/1.1 as persistent connections and pipelining have made obsolete the notion that every connection comprises a single request-response pair. But, the truth is that the profile of web traffic has been specially affected by the new contents and services provided by the application.

As of today, the traffic of a user browsing the web usually comprehends a large number of connections directed to many different servers and of vastly different characteristics (consider, for example, a small and short connection that downloads a small icon versus a large one responsible for the download of a Youtube video). Although the problem of distinguishing web traffic from the traffic of other applications has been tackled by many authors, few have ventured into disentangling that set of connections in order to make sense of which part each of them plays in the web interactions of the user. Even fewer have approached that problem without relying on application data. This is, precisely, the objective of this thesis: inferring web browsing interactions from a flow-level perspective (*i.e.* without using application-level data).

1.2 Approach

In this thesis, we take an user-centric perspective and we study the connections initiated by the user and directed to different servers in order to identify web browsing interactions such as webpage visits or website sessions. That is, we use data extracted from web traffic at a connection-level in order to infer information about application-level operation. In this section we explain our approach to this problem in detail.

First of all, our user-centric perspective is given by where we capture web traffic, which is always in or close to the web clients. A client vantage point ensures the capture of all the connections between the client and all the different servers it connects to. This provides all the information needed in order to characterize and study user traffic. Details about our vantage point can be found in chapter 3.

Now that we can consider the complete traffic of a web user, we can aggregate it at different levels which correspond to parts of the web application operation. These are the web browsing interactions that we would like to infer by studying the traffic of the user. From lower to higher level of aggregation we distinguish:

Request-response pair: HTTP (and, by extension, HTTPS) is a request-response protocol in the client-server computing model. In the normal operation of the protocol, the client sends request messages to the server asking for specific resources (*i.e.* content such as HTML documents, images, etc.). The server, in turn, answers the client's requests with response messages that carry the solicited resources. Therefore, we could say that a request-response pair is the minimal element inside web communication.

Connection: In order to request and receive content, clients establish connections with the servers. HTTP relies on TCP connections and, since HTTP/1.1, these connections can be kept open for more than one request-response pair. As a consequence, a single HTTP connection can be used to request and download multiple elements of a webpage or even of multiple different webpages (which host part of their content in the same server and which are accessed during a limited time span). However, in order to expedite the transfer of data it is common that clients open multiple concurrent connections with the same servers in order to request and download different resources at the same time.

Webpage download: Request-response pairs and connections are transparent to the user who experiences browsing the web as visiting a set of different webpages. In most cases, the action of visiting a webpage is triggered by the user by following a hyperlink, typing the address or opening a bookmark. Modern webpages can be heavy and contain complex content types (such as multimedia or web applications). Therefore, in the vast majority of cases, multiple connections are needed in order to download the content of a webpage. Moreover, as a lot of this content is distributed by CDNs and an also sizable fraction of it comes from third parties (*e.g.* an embedded Youtube video), the connections responsible for the download of a webpage can be directed to many different servers.

Website session: Websites are usually composed of multiple webpages. Users that visit a webpage will often visit other webpages of the same website. For example, a Facebook user may visit his profile and that of some of his friends, a Youtube user may watch multiple videos, or a CNN user may read multiple news articles. A website session comprehends the different webpages of that site visited by a user during a period of time. In many cases, the content of these webpages is heavily related and some of it may be hosted in the same servers. However, it is also possible for two webpages of the same website to have entirely different content. With the advent of tab-based browsers, concurrent sessions to two or more websites have become a common occurrence and some sessions can be very long (for example, a user may access his webmail account and leave it open in a tab while he browses through other websites).

Browsing session: Finally, a browsing session comprises all the web interactions of a user inside a period of time during which the user is actively browsing the web. Browsing sessions may contain one or more website sessions and can be very variable in length. The use of the word *session* both for website and browsing sessions can be confusing but both examples are present in the literature (compare [Tho14] and [BMM+09]). Due to the scope of this thesis we will mostly refer to website sessions but we will specify as often as possible.

From a capture of user traffic we can isolate request-response pairs by simply checking application data. Web traffic has been thoroughly studied and modeled at this level of aggregation [CJO+01; IP11; SAM+12; XIK+13]. Studying request-response pairs offers insight into the nature of web resources and allows gathering detailed information about HTTP operation. Likewise, many studies have characterized web connections as they can be isolated just by looking at IP and TCP headers [SCJ+01; CCW+07; SMS+10]. Monitoring web traffic at a connection level is specially interesting for Internet service providers (ISPs) and network administrators in order to correctly model the effect of this traffic in their networks.

However, few studies have focused on higher web traffic aggregation levels. Intervals of web activity in the traffic of a user (browsing sessions) are mentioned in studies about web traffic but few works have been centered around them [BMM+09]. Even less proposals have ventured into studying webpage downloads and website sessions which is our objective.

The most evident way of inferring webpage downloads and website sessions from network traffic would be through deep packet inspection, that is, studying the application-level information of each connection. We would be going back to the request-response pair level in order to study the dependencies between the downloaded resources (*e.g.* checking an HTML document to see which images are included in it or using the HTTP referer field to figure out which webpage has prompted the request). Then we would use these dependencies to reconstruct the whole HTTP conversation finding related request-response pairs and, subsequently, related connections and webpage downloads.

Nevertheless, deep packet inspection has multiple drawbacks. On the one hand, it requires processing a lot of information for each connection hindering real-time operation. On the other, accessing user data raises privacy concerns and, in fact, depending on the local legislation, may be illegal [MFWRG+12]. Moreover, a sizable fraction of web traffic nowadays is HTTPS for which application-level data will not be accessible [Wired14] and encrypted traffic is indeed expected to become more and more prevalent in the future with HTTP/2 using transport layer security (TLS) by default [BPT14].

Taking this into account we approach our problem from a connection-level perspective. That is, rather than relying on application data, we only consider information that can be extracted from the network and transport levels. This kind of information is widely available on the Internet (in most cases in the form of NetFlow records [Cis12]) and, due to its summarized nature, it is easy to store and process. Moreover, connection-level data is unaffected by encryption and using it is far less invasive on user privacy. Nevertheless, the information available in flow records is far from complete which transforms identifying webpage visits and website sessions from a simple —if laborious— problem into a challenging one. The advantages and disadvantages of working with connection-level data are further discussed in chapter 3.

1.3 Objectives

Once we have defined the problem at hand and have decided to build our study around connection-level information, we lay out the following objectives:

- To characterize web traffic focusing on connection-level parameters. In particular, we will center our study in webpage downloads and website sessions. In order to carry out this characterization it will be necessary to research and develop capture methods that identify the connections according to the specific webpage download or website session that generated them.

- To study the connection-level parameters that can be used to find related web connections and classify them according to the webpage download or website session to which they belong. Although multiple parameters are taken into account throughout the thesis, we primarily focus on time-based parameters and server IP addresses. The former are interesting because the connections involved in a webpage download are usually very close in time and the latter because a sizable part of the content of websites is hosted in a limited number of dedicated servers.

- To use these parameters to develop methods able to find webpage downloads and website sessions in the traffic of web users. These methods are the ultimate goal of the thesis as there are very limited similar proposals in the literature in spite of the interesting applications they could have: inferring webpage downloads and user sessions from web traffic could help network administrators in gathering information about their users and their web usage, and in prioritizing important services and blocking undesired ones.

- To test the developed methods with traces of real web traffic from multiple users with the objective of assessing their performance under real-world conditions. For this purpose it will be necessary to research ways of labeling web traffic traces in order to obtain the ground truth necessary for testing our methods.

We also introduced an additional related objective that became interesting during the research of labeling methods for web traffic:

- To study the server IP addresses accessed during sessions of different websites to identify servers closely related to them. The acquired knowledge could be used to label web traffic traces or to identify users accessing those particular websites and characterize their traffic.

1.4 Document organization

The rest of this thesis is organized as follows. Chapter 2 presents the state of the art focusing, in the first place, on the changes the web application has experimented since its creation and their effect on web traffic. The evolution of the web is reviewed from the point of view of web protocols, infrastructure, content, web browsers and the services provided through the web. Traffic classification techniques have been an inspiration for this thesis which shares some of their methodology and nomenclature. A collection of these techniques is also provided in chapter 2 before centering on web traffic classification, where proposals closer to the work presented here are compiled.

Chapter 3 describes how the traffic captures that are used in throughout the thesis are captured and processed. Different capture vantage points are discussed and the traffic monitoring infrastructure in the Public University of Navarre is introduced. The chapter also explains how traffic is converted to (or directly captured as) flow records, which flow-level parameters are considered and the advantages and disadvantages of working from a flow-level perspective.

Chapter 4 provides an initial approach to the problem of identifying distinct elements inside web traffic. It focuses on finding website sessions in captured web traffic using only flow-level statistics. Information about connection start and end times and server IP addresses are used to develop an algorithm capable of clustering the connections of the same website session in a small number of groups.

Chapter 5 introduces a method for finding server IP addresses related to specific websites through the study of a traffic trace. Again, the study relies on flow-level statistics to design a method that could work with NetfFlow-type records. The basis of the method is the hypothesis that certain IP addresses are closely related to specific websites and that they could be identified by studying multiple sessions of those websites.

Chapter 6 offers a thorough characterization of webpage downloads using connection level metrics. A data set of more than 20,000 webpage downloads is captured and thoroughly studied in order to provide different metrics that model a normal webpage download.

Chapter 7 takes into account the study of the previous chapter to present a method for clustering together the connections involved in the same webpage download. Different approaches to the problem are considered and a time-based method inspired by a well-known clustering algorithm is designed.

Finally, chapter 8 concludes the thesis and presents possible lines of future work both directed to extending and improving the different methods presented in the previous chapters and to applying their results into other fields of web traffic study.

CHAPTER 2

State of the art

This chapter provides background in the topics related with this thesis. We start by describing the web application and giving a historical perspective of the changes it has experimented through the last twenty years. We focus on changes on the web protocols, infrastructure and content and how they have contributed to the appearance of new services provided through the web. Moreover, we review the evolution of the capabilities and features of web browsers. All these changes have had a profound effect on the profile of web traffic and we gather different experiences in the scientific literature about how to collect and characterize said traffic.

Traffic classification techniques have been an inspiration for the work presented in this thesis which shares some of their methodology and terminology. We offer a summary of the state of the art on Internet traffic classification with a special focus on statistics-based techniques that attempt to classify individual connections according to the application that generated them.

Finally, we have reviewed the —somewhat limited— scientific literature on web traffic classification: both those works that attempt to identify specific elements in web traffic (such as specific services or types of content) and those that seek to classify it according to the individual websites that generated it.

2.1 The web

The World Wide Web (often referred simply as the web) was developed by CERN scientists Tim Berners-Lee and Robert Cailliau in the early 1990s. A proposal published by them on November 1990 [BLC90] introduced the World Wide Web as a hypertext-based scheme by which clients (web browsers) would be able to access information stored in different CERN servers through a computer network. The first published version of the HTTP protocol (HTTP v0.9 [BL91]) dates from 1991 and the first draft for the HTML standard, from 1993 [BLC93]. On that same year the first popular web browser, Mosaic, released its first version with Netscape and Internet Explorer appearing in 1994 and 1995 respectively.

By 1994 the web was gaining around 150,000 users each month [PR94] and received a growing interest from the scientific community. Work was done in order to characterize user navigation patterns [CP95], server workload [AW96] and web traffic [CB97; Sed95; SCJ+01]. In the web these articles described most of the content was static and consisted primarily on text (HTML documents) and relatively small images. Most, if not all, of the content of each webpage was stored in only one server. Early web users would use the application sparingly averaging one browsing session a day. Saving or printing visited webpages was not a strange occurrence happening around 2% of the time (for the authors in [CP95] this was a surprisingly low percentage and they inferred from it that web users had a minimal potential of copyright infringements). All in all, it was a very different situation to the one the web experiments today.

A study of Internet traffic in the first half of 2014 [Sand14] shows that web browsing and web-based services (such as Youtube) account for more than 37% of the total traffic from devices in non-mobile networks in Europe. This more than doubles the share of the second most popular application, BitTorrent, which is responsible for around 15%. In North America, the increasing popularity of real time entertainment services (particularly video streaming) has relegated the web to a second place with 24% of the total traffic (Netflix taking the top spot with 31%). This trend may repeat itself in Europe where video streaming services are still building their market but, in any case, it seems difficult that the web will lose its position as one of the main Internet applications worldwide.

The web has enjoyed a high popularity over the last two decades thanks to its adaptability and flexibility. The relatively simple web protocols have been easily adapted to provide increasingly complex content. A wide variety of content types (such as videos, scripts or interactive media) have been added to the text and images that classic webpages traditionally hosted. Moreover, modern websites not only offer these new content types, but they do so in a dynamic way, keeping their content current and tailoring their offer to each specific visitor. The challenges introduced by these changes have been met with the development of multiple complementary tools (such as cookies or JavaScript) and of network infrastructure (such as web proxies or content distribution networks). As a consequence, the web become a very flexible application able to provide a huge range of different services aside from traditional web browsing. Nowadays, services like e-mail, file-sharing, document edition, video streaming, on-line games or e-learning are, in a lot of cases, provided through the web. The ability to access these services has made the web a must-have application and web browsers are present in almost every network enabled device from desktop computers to mobile phones, e-readers, or even smart watches.

In the following subsections we will explore the evolution of the web during the last twenty years and the characteristics of current web traffic.

2.1.1 Evolution of web protocols

The first official version of the HTTP protocol was HTTP/1.0 introduced in 1996 [BLFF96]. Much more detailed than v0.9, it vastly expanded the protocol with extended operations and negotiation, and additional methods and header fields. However, by 1996, the HTTP working group was already developing HTTP/1.1 and, in fact, many web browsers of the time were already compliant with it before it was officially released in early 1997 [FGM+97]. The new standard introduced two important improvements over HTTP/1.0 with the objective of boosting its performance: persistent connections and pipelining. In HTTP/1.1, all connections are considered persistent unless declared otherwise. This way, connections can be reused for more than one client request, making an efficient use of server and network resources and avoiding latency introduced by TCP handshakes. This makes obsolete the notion that every connection comprises a single request/response pair. Persistent connections also allow for the introduction of pipelining, a technique in which multiple HTTP requests are sent on a single TCP connection without waiting for the corresponding responses. Pipelining improves webpage loading times (specially in high latency connections) as the server can send multiple responses immediately one after the other without having to wait for the requests. HTTP/1.1 was updated by RFC 2616 in June 1999 which in turn was obsoleted by RFCs 7230-7235 published in 2014 with the objective of revising and clarifying the protocol.

The effects of HTTP/1.1 on network performance were studied early on [NGBS+97]. However, even today when all web traffic is HTTP/1.1, the way in which different browsers and servers take advantage of HTTP/1.1 characteristics is very variable. For example, persistent connections depend on a keep-alive timeout that decides for how much time an idle connection is kept open. For a specific connection, this timeout will depend on the ones specified by the web browser and the web server (it will be the shortest of the two). Values for the keep-alive timeout are not standardized and can usually be configured by the user. Popular modern web browsers have by default intervals of one minute or more (Internet Explorer, 60 seconds; Firefox, 115 seconds; Google Chrome, not documented but around 300 seconds) while servers usually have much shorter ones (*e.g.* Apache 2.0, 15 seconds). Moreover, the maximum number of total concurrent persistent connections and concurrent persistent connections per server is also different for different browsers as are the policies on how many connections should be opened to the same servers depending on the content that is going to be downloaded from them.

Up to HTTP/1.1, the HTTP protocol has privacy and security issues as it is vulnerable to wiretapping and man-in-the-middle attacks. In order to provide a secure version of HTTP, Netscape Communications created HTTPS in 1994 by layering HTTP on top of SSL (Secure Sockets Layer, a cryptographic protocol designed to provide communication security over the Internet). After SSL was superseded by TLS (Transport Layer Security), the current version of HTTPS was formally specified by RFC 2818 in May 2000 [Res00]. Since then, HTTPS has been more and more used as the popularity of services that required the exchange of sensitive information increased (such services are prevalent today: webmail, social networks, e-commerce, online banking, etc.) and malicious users willing to take advantage of unsecured connections became more common. Moreover, recent discoveries about the eavesdropping practices of various governmental agencies of powerful countries have pushed for a more widespread use of HTTPS connections [Wired14; tim]. As a consequence, the future version of the HTTP protocol, HTTP/2, which is currently under development, will use TLS by default [BPT14]. The basis of HTTP/2 is the Google-sponsored protocol SPDY that in addition to improving web security includes new mechanisms for reducing webpage load and latency.

In addition to HTTP/2, new protocols have been developed in the last years to meet the new requirements of the web. WebSocket [FM11] is a protocol that makes it possible to open an interactive communication session between the user's browser and a server. It provides a full-duplex communication channel over a TCP connection in which the server is able to send content to the browser without the need of a previous request from the user. This bidirectional channel can be used for streaming live content (*e.g.* in video conferences) and for real time applications like games. Another new proposal is the QUIC protocol which is designed to improve latency in certain web applications by using UDP instead of TCP at the transport layer [QUIC].

2.1.2 Evolution of web infrastructure

In the previous subsection we have described how the evolution of HTTP contributed to the optimization of client-server interactions. However, as the web was more and more used, bandwidth usage and server load became problematic. Moreover, lag also became an issue as web servers and web clients were located all around the world forcing queries and responses to travel large distances passing through a considerable number of routers. The main way the web has tackled these challenges has been through caching. A web cache is a mechanism for the temporary storage (or caching) of web documents so that subsequent requests for said content can be served from the cache rather than from the original server that stored it. Caches can contain bandwidth usage and server load by reducing the number of queries to the original web servers. They also can help with lag by moving content closer to the clients that request it. HTTP/1.0 already included some considerations to make caching possible such as, for example, the *expires* header field while the HTTP/1.1 standard devoted a whole section to caching considerations.

Web caches can be used in multiple situations. In the client, browser caches store static content of recently viewed webpages so that it can be reused if a webpage is revisited (for example, when clicking the back button). Proxy caches work in a similar way but instead of providing content for a single user, they support hundreds or thousands of them. Proxy servers usually are placed on the edge of an organization's network. Users of that network have their browser requests routed through the proxy which stores content of the webpages they visit. This way, multiple users accessing the same popular webpages can share the benefits of the cache in a more efficient manner. A gateway cache is also called a reverse proxy as instead of residing in front of a network of browser users, it resides in front of a web server. It stores server responses as they pass through it and answers subsequent requests with said cached responses reducing server load in the process.

Aside from caching, another technique to reduce server load is load balancing which consists in distributing HTTP requests among two or more servers where web content is duplicated. There are several techniques to implement a load balancing scheme of which the simplest is DNS load balancing. With this technique a single host name (the website's) is associated to multiple IP addresses each of them belonging to a different web server. When a DNS server resolves the host name it provides the different IP addresses in a round-robin manner achieving load balancing in this way.

Both techniques have their shortcomings. Caches, particularly those close to the clients (proxy caches), can be problematic in a web that has become more and more dynamic, interactive and tailored to specific users. Cached content has to be refreshed very fast nowadays and the content of the same website may not be identical for different users. On the other hand, load balancing reduces server load problems but coherence has to be maintained for the content of the different servers. Moreover, it does not address the problem of lag caused by large distances between clients and servers.

To solve these issues, content distribution (or delivery) networks were designed as a combination of caching and load balancing which takes into account the need for a geographical distribution of web content. A Content Distribution Network (CDN) can be described as a group of geographically dispersed servers deployed to facilitate the distribution of information generated by web publishers in a timely and efficient manner [Hel11]. The geographical dispersion of the servers in the CDN reduces latency between client and server by moving the content closer to the users [CCY00]. Moreover, CDNs implement load balancing considerations in order to distribute load among the servers in an fair manner.

Very big web content providers (*e.g.* Google) have deployed their own CDNs all around the globe. However, this is unfeasible for most organizations and, because of that, they rely on the services provided by third party CDNs (such as Akamai Technologies or Limelight Networks). Moreover, CDNs owned by Internet service providers are a growing trend in the last years. These CDNs make sense because they allow to reduce the demands on the ISP network backbone and also because some ISPs are becoming content providers (especially of video in triple play models). ISP-owned CDNs have the additional advantage of being able to deliver content even closer to the user as the ISP controls the last mile. Moreover, ISPs can offer multicast support and QoS in their networks.

Nowadays, when accessing a particular webpage, the amount of content coming from servers in a CDN is very variable. On the one hand, simple and not very visited webpages may still be fully hosted on their server but, on the other, popular modern websites can completely rely on CDN infrastructure with even multiple CDN servers fully dedicated to hosting them. In many cases, the situation falls somewhat in the middle: some of the content is hosted in a website-owned server and other —ancillary— elements in CDN servers. This is especially common when the secondary elements of the webpage are either very heavy and latency sensitive (*i.e* video) or third-party content common to different webpages (*e.g.* advertisements). In those cases, hosting that content in a CDN has clear advantages.

Aside from improving latency, server load and network usage, the effects of CDNs in modern web traffic are notable. Whereas in the past most, if not all, of the content of a webpage came from the same server, now, content is usually distributed through many servers that may belong to different organizations. As a consequence a webpage download which consisted of connections to usually one server IP address now consists of connections to multiple IP addresses on different networks. Moreover, two different, and relatively close in time, downloads of the same webpage may involve connections to mainly different IP addresses because of load balancing inside the CDNs. The opposite case can also happen: two connections to the same IP address may be related to the download of different webpages which host part of their content in the same CDN.

The importance of CDNs to web operations have been addressed by the scientific community and there are many works in the literature characterizing their structure [HWL+08], their ability to improve the service provided [BRV+06], and how much they are used by popular websites [Cha10].

2.1.3 Evolution of web content

Even though content in the very first web pages was mostly static, it soon became interesting to offer some kind of personalization or interactivity to the users. This came in the form of "shopping carts" for early e-commerce services or the ability of users to log into the website and keep a profile were their information and preferences were stored. In order to offer this capabilities, cookies were introduced in the mid 90s as a mechanism for session management able to keep state information in HTTP [KM97]. Cookies are downloaded from the web server into the client and sent back to the server each time the client loads the website. Cookies can keep user information during a browsing session (session cookies) or store it for a longer period of time spanning multiple sessions to the same website (persistent cookies). Aside from session management and personalization, cookies can also be used to track users as they browse through the webpages in a website.

Cookies are still widely used today although their tracking capabilities have risen many privacy concerns requiring strong legal regulations [MM12]. The main issue is that providers of third-party content, such as advertising companies, can use persistent cookies to track users through any websites where their content is integrated. On the best case, this information is only used to tailor website contents (especially ads) to the browsing history of the user. Tailoring of website content can be achieved by other means like, for example, inspection of HTTP referer fields, geolocation (of client IP addresses) or employing user information collected through the use of certain websites (such as search engines or social networks). Moreover, many websites today use third-party web analytics services to gather information about their users. The most popular today is Google Analytics [Tech12] which uses a web beacon (also called web bug or pixel tag) to track users as they browse through tagged webpages.

Dynamic web content and interactivity have motivated the creation and development of programming environments able to offer those capabilities. Both server-side (PHP, MySQL, ASP, etc.) and client-side (primarily JavaScript) processing are ubiquitous today. Using these tools, many web applications have been created and have evolved during the years. The most complex ones, sometimes called Rich Internet Applications, traditionally have required the use of browser plug-ins (Adobe Flash, Microsoft Silverlight, etc.). They often offer interactive multimedia content such as browser based games.

Aside from the change from static to dynamic content, the other main difference with early webpages is that multimedia has become a staple of modern web content. If older webpages usually consisted of text accompanied by some images, many websites today are primarily composed of multimedia content. Recent studies show that video streaming takes more than half of the total HTTP traffic in Europe and North America [Sand14]. Some of this multimedia content is also provided by means of browser plug-ins (especially Flash).

In order to address these new trends of web content, the most recent revision of HTML has introduced multiple tools to provide native support to many types of multimedia and dynamic content, avoiding the need of external plug-ins. HTML5 [HTML5], which was officially released in 2014 but has been in development since 2004 and is already widely used, includes audio, video and canvas elements (the last of which is used for scriptable rendering of 2D graphics). It also integrates scalable vector graphics (SVG) and mathematical formulas (through MathML).

The effects of new content in web traffic have been varied [SFK+09; BMS11]. The prevalence of multimedia elements have made modern webpages much heavier than they were in the past putting strain on network resources and explaining the popularity of CDNs as presented earlier. The dynamic nature of modern web content makes it difficult to predict how many elements are going to be downloaded when accessing a webpage, how big they are going to be or from where they are going to be downloaded, even if we have accessed the same webpage recently. Tailoring makes possible that the same webpage has very different content for different users. Advanced web applications usually involve longer webpage dwell times while the user interacts with the application, which usually runs in the web browser but may interchange data with the server depending on its specific operation.

2.1.4 Evolution of web browsers

Of the first three popular web browsers that we mentioned at the beginning of this section, only Internet Explorer survives today. During the late 90s and early 2000s it enjoyed a position of undisputed dominance reaching percentages of usage share over 95%. The release of Mozilla Firefox (somewhat of an heir of the by-then-disappeared Netscape) in 2004 and, specially, Google Chrome in 2008 reverted this trend. Today, Chrome is preferred by around half of the web users, with Internet Explorer and Firefox competing for the second spot with usage shares close to 20% [SCounter]. Other popular browsers include Safari and Opera. The increased competition has pushed browsers to include new attractive features on top of improving their rendering capabilities and adapting to new web protocols and standards. These features influence the way modern users access the web and the traffic they generate.

From the point of view of the user interface, many functionalities have been proposed during the last twenty years. Some, such as bookmarks of password management, have become must-haves for any browser while others, such as mouse gestures, have remained anecdotal. Perhaps, one of the most influential evolutions has been the introduction of tabs. A tabbed interface has important effects on the user's browsing habits [VSG+06; ZZ11]. Tabs allow users to keep open webpages that they may want to revisit later, open multiple interesting links at the same time, or browse through a webpage while another is loading. Because of this, it is common today that users leave certain webpages open for hours, if not days, or that they download multiple webpages concurrently. Tabs also provide a faster navigation experience as users can switch between open webpages seamlessly. Moreover, most modern browsers also include an easy access to search engines (by means of a dedicated search bar or integrated in the address bar) allowing users to find new interesting webpages rapidly.

A common way to introduce and test new features has been through browser extensions. Most modern browsers allow the installation of plug-ins that add new functionalities (some of which have been included afterwards in their vanilla versions, *e.g.* pop-up blockers). Many of these extensions change how webpages are rendered in different ways, from blocking ads or flash elements and translating text, to providing a full-fledged scripting interface to modify web content on the fly (*e.g.* Greasemonkey [Pil05]). Extensions can also be used by analytics services to gather information about web usage (*e.g.* Alexa toolbar [Alexa]). Although extensions usually rely on client-side processing they also have an effect on web traffic as they may prevent some content from being downloaded. Additionally, they often require downloading data from servers that did not host elements of the original webpages (for example, safe browsing extensions may need to download information about blacklisted websites in order to point out dangerous hyperlinks in a webpage).

All this describes the evolution of web browsers designed for PC systems. Nevertheless, in the last few years, web browsers have been installed in many different network enabled devices, especially, mobile ones [Her09]. These devices usually rely heavily on web content and it has been necessary to adapt traditional browsers to the specific capabilities of each device. In turn, many webpages have adapted to these mobile browsers (specially in order to consider new screen sizes). In fact, web content has become so central to the operation of mobile devices that many apps designed for tablets or mobile phones are little more than a simplified web browser specifically designed for a concrete website.

2.1.5 Services provided through the web

The evolutions presented in the previous sections have been both cause and consequence of the appearance of new services provided through the web. That is to say, new services introduced demands on the application that forced it to improve and, at the same time, those improvements paved the way for newer and more complex services. As we said at the beginning of this chapter, the web was originally conceived as a means for CERN workers to access static information stored in their servers. However, today, multiple different services are provided through the web aside from traditional web browsing. Many of these services are highly interactive and rely on content provided by their users [O'r07]. We will review some of them in the following paragraphs.

In the first place, the web has become an interface for classical Internet applications: Most e-mail providers have a web interface that allows their users to access their accounts through a web browser. This interface is often the only way users check their e-mails through. Chat and instant messaging services are also provided through the web with many added functionalities (up to full-fledged tele and videoconference). Web-based file hosting (Dropbox, Google Drive, etc.) and sharing (Rapidshare, Megaupload, etc.) services have enjoyed massive popularity over the last years outshining FTP and even P2P applications, which were, until then, the more common ways of transferring files between hosts. Many file sharing services have disappeared recently as a consequence of stronger anti-piracy regulation but, at the height of their usage they had very apparent effects on web traffic [SCBRSP12].

Many popular websites have been built around multimedia content, be it images (Flikr, Instagram, etc.), audio (Last.fm, Soundcloud, etc.) or video (Youtube, Vimeo, etc.). Of these, video streaming is by far the most disruptive to the characteristics of web traffic and the requirements it demands from the networks. It is also the most common, with Youtube traffic amounting to close of half of the total web traffic in certain areas [Sand14]. Because of this, Youtube.com has gathered a lot of attention from the scientific community [GAL+07; YF08; ZSG+09]. However, although some studies focus on the effect of these services on web traffic (specially on the influence of video codecs in per-packet statistics), many of them focus on the social characteristics of the website (such as the popularity of particular videos). In fact, most media-centered websites have a social network structure in which the users themselves provide the multimedia content.

Nowadays, social networks, those multimedia-centered and those of a more general nature (*e.g.* Facebook), are among the most visited websites [MMG+07;

Alexa]. In social networks, the content is not only provided by the users but tailored to their interests, social relationships and many other factors such as date and time, recorded past behavior or information about the users obtained through third parties. All this gives traffic from social networks many distinctive characteristics [SFK+09].

Although interactivity is common in many modern websites, some services are specially characterized by it. This is the case of browser-based gaming, web-mapping services (*e.g.* Google Maps) and web-based office applications (*e.g.* Google Docs) among others. The traffic of websites that provide these services is influenced by their interactivity which also results in characteristic profiles [SAA+08].

All in all, given the vast number of different services that can be accessed through the web, most users interact with the Internet only through their web browser. Moreover, web-based applications have started to substitute desktop applications in many cases, with the most remarkable example being Chrome OS based systems in which the whole operating system is built around the web browser and web applications [Wri09]. This trend has promoted the development of new protocols and standards that allow the web to provide services for which it was not designed and thus was not well suited. For example, webRTC [BBJ+15] is an API specification under development that offers native support for browser-to-browser applications (videoconference, P2P file-sharing) leaving behind the client-server paradigm typical of the web.

2.1.6 Modern web traffic characterization

The evolution of the web, as presented throughout this section has obsoleted the characterizations of its traffic carried out in the nineties. Studies published in the last years [FMN+03; WOH+08; IP11; NJA13] paint a changing picture in which the profile of web traffic has been increasingly affected by the new contents and services provided by the application. Nowadays, from a network perspective, accessing a webpage may imply establishing multiple connections to different servers while the elements of the webpage (often coming from third parties) are downloaded and, in many cases, user information is collected. The result is a set of a variable number of connections of different durations and sizes to multiple server IP addresses. Moreover, as a sizable amount of the content is dynamic, these connections may change if the webpage is accessed at a different time or by a different user. The complexity of modern web traffic has called for extensive characterization work from the scientific community. This characterization has been attempted from different perspectives [FL05].

Some proposals focus on server operation modeling server load and the behaviour and habits of the users that access the server [LK07; BRV+06; KHW+14]. These works are useful to improve server and website designs in order to provide the best possible service to the users. They also allow gathering information about the users as they browse through the webpages hosted in the server. However, as content in modern webpages usually comes from multiple different sources, a server-side perspective offers a limited picture of the interactions of a user with the web.

Works with a user-centric perspective are also common. In them, data is gathered in the client hosts which allows capturing all the different elements that form a webpage. When the objective is to describe the content of individual webpages, data may be captured by accessing them with automatic programs. With this methodology the content of different webpages, where it was hosted and the influence of this in webpage load times was described in [BMS11]. In [FMN+03], a similar setup was used to check how the content of modern webpages changes over time. If, on the other hand, the objective of the study is to analyze real user traffic, it may be captured at the clients while users access the web. In [WOH+08] this was used to describe and relate certain webpage characteristics and user behaviors.

Finally, a network perspective can be adopted in which traffic is captured in a middle point between clients and servers. Works that use this kind of data [SAM+12; ZSG+09] often focus on general statistics of the traffic and on information that can be extracted from HTTP and TCP header fields. A network vantage point usually limits the study to the characteristics of individual TCP connections rather than of complete webpage downloads or user sessions in a website as it is not easy to find the relationships between the connections. In order to avoid this, additional application-level information can be obtained if the traffic is captured in a proxy [IP11].

2.2 Internet traffic classification

The use of the Internet has grown exponentially over the last years and communication infrastructures have been improved accordingly. This has opened the door for the diversification of the services provided through the Internet. As we have seen in the previous section, the web, a classic Internet application, has evolved in order to provide a lot of those services. However, many of them rely primarily on specific applications. This is the case of services such as P2P file sharing (BitTorrent, eDonkey), audio streaming (Spotify, Pandora), tele and video conference (Skype), online gaming, telnet, remote desktops and many more. From the perspective of an ISP or a network administrator, it is very interesting to be able to distinguish between the traffic of different applications. This allows prioritizing certain services, blocking undesired ones and, in general, gathering information about their users' habits and needs.

Traditionally, application identification has been possible simply by checking transport port numbers. The Service Name and Transport Protocol Port Number Registry maintained by the IANA [CET+11] offers a list of port numbers associated with specific applications. However, applications are not forced to use those port numbers. In fact, many modern applications either use ephemeral ports (with numbers higher than 1024) or ports assigned to other more traditional applications (specially, port 80 assigned to HTTP). The former strategy allows certain applications (such as P2P file sharing) to avoid detection by employing different user-configured port numbers. The latter can be used to benefit from the lack of restrictions that firewalls offer to the traffic of known applications such as the web. Because of this, when dealing with modern Internet traffic, port numbers are no longer a reliable way of identifying applications.

In order to overcome this issue, in the last years, techniques have been developed that attempt to identify applications through other properties of their traffic. The initial approach was to rely on signature based systems [CKY+04; HSS+05]. These systems study the payload of the packets searching for strings characteristic of each application. This incurs into heavy computational loads as the volume of data to be processed can be very high. However, signature based systems are very reliable as long as two requirements are met: traffic must not be encrypted and the application to detect must be known beforehand allowing its behavior to be thoroughly characterized in order to generate the signatures. Sadly, these requirements are not fulfilled in many cases today. On one hand, new applications appear often, sometimes using ad-hoc application-level protocols that may change over time. This makes it difficult to keep updated signatures. On the other, as we have seen previously, encrypted traffic is increasing and probably will be the norm in the near future.

Therefore, new behavior-based techniques have been developed that seek to identify the traffic of applications using characteristics more difficult to disguise. In most cases, they focus on the study of traffic statistics rather than on the specific information contained by the packets. Ultimately, traffic patterns generated by an application depend on the service that it provides and, given that the application should try to make the most of the available network resources, they cannot be modified without negative effects to its performance. Many different traffic variables are studied in these methods (perhaps the more widely used are packet size and time between packets) with different models depending on if they are calculated for each packet, group of packets, flow, group of flows between two hosts or for all the traffic generated by a host. Nevertheless, in most cases, flow-level metrics are used. These flow-level metrics are usually fed to machine learning methods that carry out the classification.

In this section we present a review of behaviour-based methods for Internet traffic classification. They are relevant to the work in this thesis for two different reasons. The first is that the web traffic classification techniques we present in the following chapters are in some ways inspired by these proposals and share some of their standpoints and nomenclature. The second is that the work in this thesis prepares web traffic to be classified according to the different services provided through the web. This can be achieved using techniques similar to these.

2.2.1 General application classification

In the first place we present methods that seek a "general" classification. That is, they attempt to identify a wide spectrum of applications from the more classic ones (web, e-mail, FTP, telnet, etc.) to others that have become popular more recently (P2P, streaming, etc.). These methods, rather than focusing in specific characteristics of each application, take into account general characteristics of the traffic that concern all of them and that can be used to differentiate them. We use as a basis the comparative survey carried out by Nguyen and Armitage in 2008 [NA08] which we extend with some newer proposals. As they do, we focus on machine learning techniques that classify traffic flows according to the application that generated them.

Machine learning techniques are algorithms that receive their input data as a group of instances characterized by the value of various attributes and, by means of a training process, they establish rules that allow classifying the instances in different classes. Here, an instance is a traffic flow (whether we consider full bi-directional flows, one of the directions or even just some of the packets); attributes are those characteristics of the flow that are considered to be relevant in order to decide which application is responsible for it; and, finally, classes are the applications that we want to identify. According to the algorithm employed, machine learning techniques can be divided in supervised and clustering techniques.

When comparing two different techniques, multiple factors have to be taken into account: the number of applications to identify, the fineness of the classification, the complexity of calculating the selected attributes (and whether it can be done in real time) and the complexity of the technique in itself. However, the most important point is whether the techniques offer an accurate classification. In order to represent this, the following concepts are usually employed.

- **True positives (TP):** are those flows that are correctly identified as caused by an application.
- **True negatives (TN):** are those flows that are correctly identified as not caused by an application.
- **False positives (FP):** are those flows that are identified as caused by an application when they are not. False positives are also called type I errors.
- **False negatives (FN):** are those flows that are identified as not caused by an application when they are. False positives are also called type II errors.

Other metrics are defined using these concepts:

- **Recall, sensitivity** or **true positive rate (TPR):** the rate of flows correctly labeled as being caused by and application out of the total flows of said application.

$$TPR = \frac{TP}{TP + FN} \tag{2.1}$$

- **Specificity** or **true negative rate (TNR):** the rate of flows correctly labeled as not being caused by and application out of the total flows not caused by said application.

$$TPR = \frac{TN}{TN + FP} \tag{2.2}$$

- **Precision** or **positive predictive value (PPV):** the rate of flows that actually are caused by an application out of the total flows labeled as caused by the application.

$$PPV = \frac{TP}{TP + FP} \tag{2.3}$$

- **Accuracy (ACC):** the rate of correctly labeled flows out of the total of considered flows.

$$ACC = \frac{TP + TN}{Total flows} \tag{2.4}$$

As we said previously, machine learning methods can be divided in supervised and clustering techniques. We now present some examples.

2.2.1.1 Supervised techniques (classification)

Supervised techniques classify instances into a predefined number of classes. That is to say, they model the input-output relationships mapping instances with specific attributes and the classes that should contain them. Typical examples of supervised techniques are naive Bayes classifiers and supervised decision trees. These techniques are divided in two phases:

- Training: in which a training data set is examined in order to build the classification model.

- Testing: in which said model is used to classify new instances.

In order to classify Internet traffic it is necessary, for the training phase, to have a traffic trace in which the flows are correctly labeled according to the applications that generated them. This trace is used to build a model for each application that will be in turn used to classify more traffic. As a consequence, these systems can only identify applications for which they have been previously trained. Moreover, labeling training sets is an arduous and time-consuming process that often has to be done manually in order to get accurate results.

One of the first uses of supervised techniques for classifying Internet traffic was published in 2004 [RSS+04] in which two different machine learning algorithms, Nearest Neighbor (NN) and Linear Discriminant Analysis (LDA), are used to classify flows into four categories: interactive (telnet), massive data transfer (FTP and P2P), streaming (RealMedia) and transactional (DNS and HTTP). Although the authors consider multiple attributes calculated at packet and flow levels, they conclude that the more interesting ones are average packet size (for each flow) and flow length. They reach accuracy results over 90% but this value decreases if a finer classification (*i.e.* with more classes) is attempted.

Naive Bayes classifiers and bayesian neural networks have also been used with good results [MZ05; AMG07]. For these proposals an interesting previous work was done in order to define possible flow-level attributes [MZC05] and design an analytic way of selecting those that are especially relevant for traffic classification purposes.

Support vector machines are algorithms that divide a set of instances in two classes and that can be applied iteratively. One of these algorithms was tested in [LYG07] reaching an accuracy over 97% for seven different classes (file transfer, interactive, www, services, P2P, mail, others). In this case, the considered attributes were general flow variables (length, size), packet size statistics and TCP header field values.

Decision trees have been used to classify traffic flows using attributes obtained from the first five packets in each flow [VG08; RV09]. Studying only the first packets of a flow was originally proposed in [BTA+06] which presents a clustering method and to which we will refer in the following section. The attributes used for the decision trees are packet size and time between packets, obtaining low false positive and negative rates (less than 1% and 10% respectively) when distinguishing between five applications: DNS, FTP, Telnet, SMTP and HTTP.

Finally, in [KHW+14], the authors raise concerns about the difficulty of comparing some of the techniques previously presented as they have been tested in very different conditions. In order to solve this problem they test multiple supervised techniques with the same data set (which contains HTTP, DNS, e-mail, chat, FTP, P2P, streaming and gaming traffic). They compare the classification results and the computational costs of the different techniques among themselves and with other approaches: port-based classification (by means of CoralReef [CoralReef] and the BLINC system (see section 2.2.1.3). Out of the tested techniques, suport vector machines give the best accuracy results with affordable computational cost. An interesting find is that port-based classification is still mostly accurate for certain applications (HTTP, DNS, e-mail and chat).

2.2.1.2 Unsupervised techniques (clustering)

As opposed to supervised techniques, clustering algorithms do not need a list of predefined classes. They are designed to find natural groups (clusters) of instances whose attributes are similar and which are thus assigned to the same class. When applied to identifying Internet traffic, these algorithms return groups of flows with similar statistic characteristics. These groups will then need to be assigned to specific applications. Although this last step can be complex, clustering algorithms have the advantage that they can find groups that reveal the existence of new applications. This is impossible with supervised techniques.

Clustering has been used for Internet traffic classification at least since also 2004 [MHL+04]. In that paper, an expectation-maximization algorithm is used for clustering bi-directional flows of HTTP, FTP, SMTP, IMAP, NTP and DNS traffic in groups of applications with similar behaviors. They use multiple flow-level attributes but some of them are calculated at a packet-level (maximum or minimum packet size inside a flow, for example).

An attempt to select the most interesting attributes for clustering is presented in [ZNA05] where the AutoClass clustering algorithm is applied over traces of a thousand randomly selected flows of different applications. By applying iteratively the clustering algorithm with different attributes, the authors conclude that the variance of packet size is the most relevant. They are able to distinguish among the traffic of eight applications reaching an average accuracy of 86%.

In [BTA+06], a classic clustering algorithm (K-means) is used to classify traffic flows using only the size and direction of the first five packets of each flow (without taking into account the initial handshake or ACKs without payload). The reasoning behind this is that those first packets contain a negotiation messages characteristic of each application and that they are, as a consequence, more interesting when it comes to classifying the complete flow. This technique has the advantage of providing an early identification of the flow's content. However, it will fail if the system is not able to capture the flow's beginning. Testing is carried out with traces of traffic from ten different applications (eDonkey, FTP, HTTP, Kazaa, NNTP, POP3, SMTP, SSH, HTTPS, POP3s) and it consists on two phases. The first step is applying the algorithm over a training data set in order to generate groups of flows with similar characteristics. These groups are then mapped to applications by means of deep packet inspection. The second step is classifying real traffic using the knowledge gathered from the training phase to map the generated clusters and the applications. The system is able to cluster flows with an accuracy close to 80%. This value can be raised up to 85% introducing different improvements [BTS06; BT07].

The K-means clustering algorithm is also used in [EMA+07] where the attributes used are calculated for just one of the directions of the flow allowing classification of traffic captured at vantage points in which the upstream and downstream directions are not available at the same time. They are able to reach accuracies in the 80-90% range but classify applications in relatively wide classes.

2.2.1.3 Techniques not based on machine learning

Up until now we have presented machine learning techniques that can be used to classify individual flows according to the application that generated them. However, there are proposals that allow the use of information from multiple flows (aggregated for each host) in order to carry out the classification. One of the most interesting works with this idea is the BLINC system presented in [KPF05].

BLINC studies the traffic of the hosts at three different levels. At the social level, hosts are characterized by the number of other hosts (different IP addresses) with which they communicate. This can be used to detect certain applications such as P2P in which hosts are part of large application-level networks. At the functional level, the system attempts to distinguish between hosts that work as clients, servers or as peers in a collaborative network. This is done studying the variability of transport ports and IP addresses. Finally, at the application level, the data obtained from the other levels is combined with flow level statistics to build models of different applications.

The resulting models are tested with traces containing web, P2P, FTP, services (DNS, SMB...), mail, NNTP, chat, attacks, streaming and gaming traffic. The system is able to identify 90% of the flows with 95% accuracy.

2.2.2 Identification of specific applications

Because of their special interest, their weight in the total load of a network or because other factors, it is very interesting to know if a host is using some specific applications by analyzing its traffic. In these cases flows are not considered independently. Rather than that, the traffic of the application to detect is thoroughly studied (at packet, flow and host levels) searching for specific characteristics that can be used to identify it.

For example, work has been done to identify Skype traffic (something useful because of the special latency requirements of the application). In [BMM+07], Skype is detected by studying two different characteristics of its traffic. On one hand, Skype encrypts payloads in a characteristic way (specially over UDP) that can be used to distinguish it from the traffic of other applications. On the other, the fact that Skype connections usually carry audio streams has effects over its traffic profile that make it identifiable. Other works [PGD+07] use study Skype signaling messages in order to identify hosts running the application by checking to which servers they connect.

A lot of work has also been carried out in order to identify P2P applications [KBF+04; CM06; WCC+09]. These applications have been often targeted by ISPs and network administrators because of their high bandwidth usage and because they are used to exchange copyrighted content. P2P applications have responded by using variable ports, obfuscating their protocols and encrypting their traffic. In order to detect P2P traffic, multiple techniques have been proposed although most works rely on detecting hosts acting as peers (rather than as clients of servers) and on studying the differences between the signaling traffic and the connections that actually carry the transferred files.

2.3 Web traffic classification

The popularity of the web and the many different services that today are provided through it have made of web traffic an attractive research topic for the scientific community. Many proposals have worked towards classifying web traffic from different perspectives. Those works, some of which we compile in this section, are of special interest to us as they are closer to the idea of this thesis.

In the first place, website sessions have often been studied from the perspective of the servers [PHMA+00; LK07; DT09]. These proposals reconstruct website sessions using information from web server logs and they offer interesting insight into the actions users take while visiting a particular website. This information can be used to adapt the website's design in order to improve the experience of the users. However, as we have said previously, we consider a user-centric perspective and the challenges in identifying website sessions in user traffic are very different to the ones faced in those studies.

Some effort has been directed to identify specific services in user web traffic. For example, in [ALC+11] statistics for packet size and packet inter-arrival times are used to distinguish between connections that belong to Gmail, Facebook and Youtube. Although the proposed method offers good results after being trained with traffic of the selected websites, it is unclear if it would achieve a similar performance if more websites of not so different characteristics were considered. A similar idea is presented in [LCL14] where multiple classification methods are tested in order to distinguish between the traffic of different AJAX-based web applications.

Rather than services, some authors have tried to identify the download of specific webpages. Using only connection-level data, the authors in [CCW+07] present a method able to detect the download of a set of webpages in NetFlow records. However, it has the disadvantage of being able to detect only webpages that have been previously characterized (something troublesome given the very dynamic nature of modern webpages). In this last case the authors rely on a simple heuristic to delimit webpage downloads in user traffic: an interval of inactivity of 10 seconds. However, they do not validate this assumption properly (or at all). Neither do the authors in [KAA06], who propose a similar scheme. On the other hand, in [MFWRG+12] the authors use a minimum interval between user clicks of 1 second and they do justify it with experimental data (even though the validation they offer is far from exhaustive). The method proposed in this last article uses both TCP and application-level data to offer a more precise detection. However, their approach is directed to identifying specific webpages for advertising purposes and still suffers the problem of working only with pre-characterized webpages.

If we focus on website sessions from the user perspective, elaborate methods for reconstructing them have relied on monitoring application level data. For example, in [KHM+13] the authors present a "cobbling" (*i.e.* clustering) method that groups together the HTTP request-response pairs related to the same website session. Their method depends heavily on the referer field which they use to identify the webpage responsible for each request. However, using the referer, they find difficulties in distinguishing between requests originated by the user following a link to a different website (that should be part of a new session) and those requesting content hosted in CDNs or embedded third-party content. In order to solve this they offer methodologies for detecting CDNs associated to specific websites and for detecting embedded content with the help of file-types and simple time-based heuristics. Finally, they test the complete method for a hundred popular websites using traffic from a university network. They exclude HTTPS based websites such as Gmail as their proposal does not work with encrypted traffic. For the websites considered they obtain good results with false positive and negative rates of about 5%.

A similar proposal, this time searching to delimit individual webpage downloads, is presented in [XIK+13]. Again, the introduced algorithm relies mostly on the relationships established between request-response pairs by the HTTP referer field. Nevertheless, it introduces an interesting methodology for detecting the first request of a webpage download based on the resource requested (HTML or XML), its size and other time-based considerations. Although the system obtains good identification results (precision and recall over 90%) it is unable by design of working with HTTPS connections.

A different approach that resorts to data obtained outside of web traffic is presented in [BMM+12]. In this case, the authors use DNS information to "untangle" web traffic. They introduce DN-Hunter, a tool able to relate traffic flows with content providers on the fly by analysing DNS responses. It is an interesting concept that yields good results in identifying webpage downloads and website sessions. Still, it requires capturing all DNS traffic directed to the individual users (something that, for example in our case was not possible, see section 3.1) and can only relate a particular server to a website if there is a relationship between them in the DNS information.

Some work has also been carried out with the objective of delimiting browsing sessions. In [BMM+09] clustering techniques are used to identify user sessions in web traffic (defining a session as a set of TCP connections generated by a user while browsing through multiple webpages during a period of time).

In brief, the state of the art on web traffic classification is relatively limited, specially considering the apparent interest of this field. Although we have presented some works related to the one we describe in this thesis, we have not, alas, been able to find any proposals similar to the ones we present in the following chapters. This gives interest to our work, but precludes us from comparing our results with those of other authors.

CHAPTER 3

Capturing web traffic as flow records from a client vantage point

Information about web usage can be gathered in different ways, from different vantage points and in different formats. In the experiments we present in this thesis we rely on captures of web traffic in order to study and identify website sessions and webpage downloads. Both usually comprehend multiple connections between the client and different servers. As a consequence, our study takes a client point of view in which all the web traffic generated by the clients is monitored. Our captures are primarily in the form of flow records because we focus our study on connection-level parameters of the traffic. In this chapter we describe how these captures are gathered in our university network. We also discuss flow records explaining the connection-level parameters we consider and the advantages and challenges derived of working with them.

3.1 Capture vantage points

When gathering information about web usage it is crucial to select an appropriate vantage point as it will heavily influence the data we are able to collect. We can distinguish three different vantage points: client, Internet link and server. We show them in figure 3.1.

For a client vantage point information is captured in or near the client. Lets consider network A in figure 3.1 which is inhabited by two web clients and a web server. Application level data about web usage can be collected via software installed in the client host or using browser extensions (capture point 1). An example of this would be Alexa Toolbar [Alexa] which gathers information about the webpages visited by the users that have it installed. This information is then used for various purposes from calculating traffic rankings to establishing relationships between interlinked websites. When capturing in the client host, interesting information can also be gathered about the user's behavior by, for example, monitoring the movement of the mouse or the time spent reading each section of a webpage. In some cases, the user can even be prompted to provide feedback about his experience (see section 4.2).

Also for a client vantage point, web traffic can be captured in the client's network card (capture point 2). The information that can be extracted from network traffic may be less extensive that the one captured at the application level but the process can be totally transparent to the users ensuring that their behavior is not conditioned by the fact that they are being monitored. Another option is capturing on the access link of the local area (or campus) network that the client belongs to (capture point 3). In this case, web traffic from multiple clients can be captured by a network administrator without the need of running any type of software in the clients. However, this does not include traffic directed to web servers inside the LAN or CAN. In our example, capturing at point 3 would allow monitoring the connections between the clients in network A and web servers in networks B, C, D or, in fact, any other server on the Internet but the web server in network A.

Client vantage points are specially useful in order to characterize web user habits: they offer a complete picture of the user's traffic as they are able to cap-

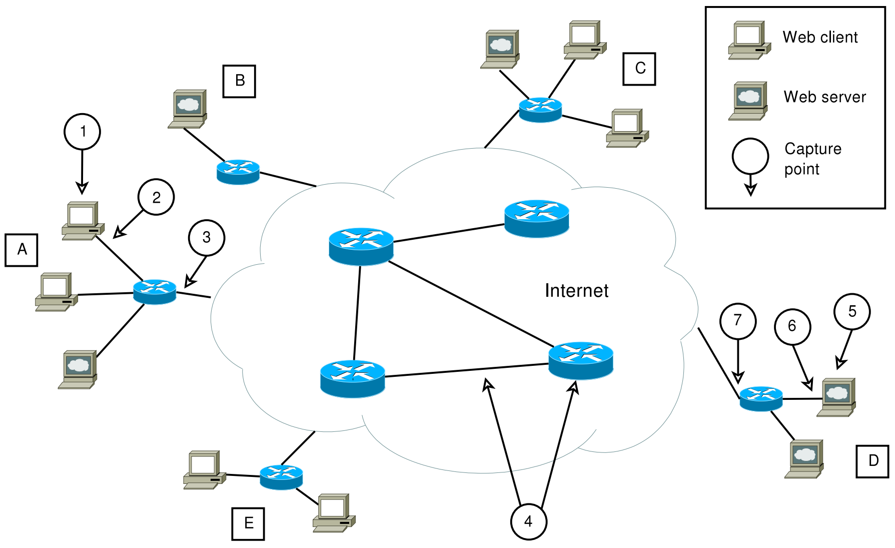


Figure 3.1: Capture vantage points.

ture all the interactions with the different web servers he accesses.

Traffic can also be captured in metropolitan or wide area networks as it travels through links (using a network tap) or in networking devices such as routers (capture point 4). The volume of traffic that can be captured at these vantage points is often very big and sampling or summarizing it is usually required before processing or storage. In most cases, network vantage points are employed by ISPs because the data obtained from them can help in optimizing their network infrastructure. In the case of web traffic they are able to monitor the connections between multiple clients and servers. However, the information provided is incomplete if we want to characterize either of them as parts of their traffic may also be routed through other links or routing devices.

For a server vantage point, information is captured in or near the server. In most cases this is done at the application level in the form of server logs that store data about server operation (capture point 5). These logs can be reviewed afterwards in order to find ways of improving the performance of the servers, detect configuration errors or study their load and the characteristics of the clients that access them. However, it is also possible to capture network traf-

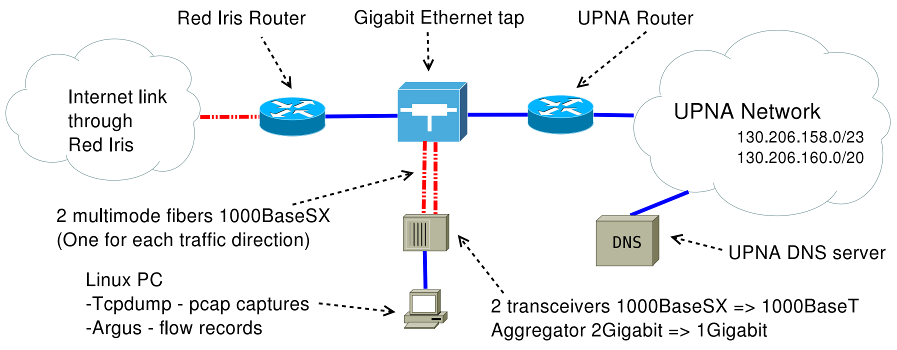


Figure 3.2: Capturing traffic at the access link of the UPNA.

fic in as it leaves/arrives to the server or the server's network (capture points 6 and 7). For servers hosting popular websites, captured traffic may also need to be summarized in this case in order to be manageable. Server vantage points (specially when capturing data at the application level) allow gathering detailed information about the actions users take while visiting the particular website they host and they have the advantage of capturing the traffic from all the different users accessing the servers.

Our study focuses on web traffic from a client point of view and, as a consequence, we will work with network traffic captured close to the clients. For us it is important that we are able to capture the interactions of each client and all the servers it connects to. This is specially important if we consider that the content of modern webpages is often hosted in multiple different servers. More precisely, in chapters 4 and 6 we capture data in the client's network card (the specific setups are explained in sections 4.2 and 6.2). In chapters 5 and 7, on the other hand, we capture data in the Internet link that serves the campus network of the Public University of Navarre. As we previously said, the interactions between hosts inside the campus network are not captured but as web traffic is mainly directed to servers outside of the network this does not have an effect on our study.

The network of the Public University of Navarre (UPNA) serves a community of close to 10.000 people of which more than 8.000 are students, around are 1.000 professors and researchers and the rest, administrative personnel. How-

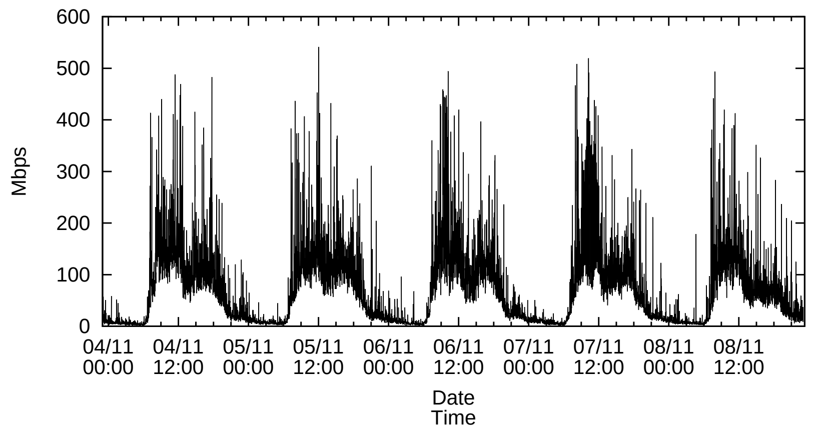


Figure 3.3: Web traffic in the access link of the UPNA during a work week.

ever, most of these users (those in computer labs and those using the university's WiFi network) connect to the Internet through NAT routers, which complicates capturing their individual traffic, leaving around 1.000 users with public IP addresses. Traffic leaving and arriving to the university network can be captured in its access link as shown in figure 3.2. A Gigabit Ethernet network tap is used to extract the traffic from the link which is then sent through a multimode fiber for each direction (fibers are used because the network tap and the sniffer are far away from each other). Both directions of the traffic are again merged in a Gigabit Ethernet link that reaches the sniffer.

We give an idea of the quantity of web traffic generated in the UPNA network in figure 3.3. In the figure we use a capture of web traffic from late 2013 that spans a work week, from Monday 4/11 to Friday 9/11, and which will be used in chapter 7. For that week, we represent the web traffic data rate in the link in Mbps. As we can see, although the average rate is close to 70 Mbps, web traffic is concentrated during work hours each day where it can reach spikes of 500 Mbps.

In chapters 5 and 7, DNS traffic in pcap format is captured in the Internet link at the same time of the web traffic captures. As it can be seen in figure 3.2 the location of the network tap and the university DNS server precludes capturing the individual DNS traffic of the users of the university network. We are able to capture, however, the DNS traffic between the university server and other external DNS servers.

3.2 Working with flow records

Once a vantage point has been chosen, it is necessary to decide what kind of traffic records to use. Records captured in pcap format with programs such as tcpdump [pcap] have the advantage of being able to store all the information of the captured traffic. In these records, both packets headers and payloads are available from which we can access a multitude of characteristics of the traffic from per packet statistics (inter-arrival times, sizes...) to payload information obtained through deep packet inspection. However, pcap traces have their disadvantages. The sheer volume of data captured makes them difficult to store and process if the traffic of a big number of hosts is considered. Moreover, the advantage of being able to extract information from the packet payloads is lost when working with encrypted traffic (which is increasingly popular as we have said in previous chapters).

A different way to store traffic information is in flow record format. Since Cisco introduced NetFlow in its routers in 1990 [Cis12], flow records have been available in many network devices. In 2004, the IETF standardized the export of flow information through the IPFIX protocol [QZC+04] which is heavily based on NetFlow v9. In NetFlow/IPFIX records, a flow is a group of packets observed during a time interval and which share a common source, destination and transport protocol (source and destination are identified by IP addresses and transport ports). Multiple statistics can be calculated and stored for each flow such as start and finish timestamps or certain header information (*e.g.* observed TCP flags). The resulting records offer a summary of the traffic that is much easier to store and process and because of that they are widely used for measurement, accounting and billing by service providers and network administrators throughout the world.

For our objective of finding related connections in web traffic, it is flow wide statistics and not per packet ones which are mainly interesting. This has driven us to consider working with flow records as they offer many advantages:

- NetFlow information is widely available in almost any network: most modern routers are able to generate flow records and no additional dedicated hardware is required. For vendors other than Cisco, the flow records generated by their network devices may be in a different IPFIX-compliant format (*e.g.* J-Flow for Juniper Networks devices).
- Flow records offer a concise summary of the traffic in a network that is easy to store and process.
- The information contained in flow records is extracted from the network and transport levels which protects them from confidentiality issues and makes them impervious to application-level encryption.

Therefore, a system developed using flow records benefits of these advantages in different ways: it can be easily deployed anywhere on the Internet as the input data is usually available and, because of its summarized nature, processing it will be fast and computationally inexpensive. Moreover, the system will be able to gather information about web usage without violating confidentiality as it does not access user data. For the same reason, it will be able to process HTTP and HTTPS traffic seamlessly.

As a consequence, in the following chapters we primarily work with flow records. In order to obtain them we use Argus rather than NetFlow. Argus [ARGUS] is an open source audit tool able to generate NetFlow-type records. It is, however, much more configurable which allows calculating many different flow-level statistics. It can collect flow records by directly listening on a network interface or summarize already captured pcap traces. It also includes many utilities to process flow records and translate them into different formats. In our case we use it to store at least the following information for each flow (in cases in which additional data is needed it will be specified):

- Source and destination IP address.
- Source and destination ports.
- Transport protocol (although this is irrelevant as we only work with TCP flows).

- Start and end timestamps.
- Number of total bytes and packets.
- TCP state at the end of capture (which specifies if the flow was still open or closed and, if closed, whether it was reseted or properly finished).

All these parameters are easy to calculate and, although some of them describe the whole bidirectional connection, they could be inferred or, at least, estimated from either the downstream or upstream data if only one of the directions is available for capture from a specific vantage point.

Nevertheless, in our captures, we always consider bidirectional flows. This is a difference from traditional NetFlow. Argus identifies clients (sources) and servers (destinations) by checking which host initiated the connection. Another difference is that Argus checks TCP messages in order to identify the three-way handshake at the beginning of the flow and the FIN or RST packets at is ending. This way, even in the unlikely occurrence of two connections with the same IP addresses and ports happening close in time, they will be considered separate flows. Taking these precautions each HTTP connection can be fully mapped to a bidirectional TCP flow in our records allowing us to use the terms *flow* and *connection* interchangeably.

As we are only interested in (outbound) web traffic we filter all non-TCP flows and those TCP flows whose destination port is not 80 or 443 as it is widely assumed that they represent the majority of HTTP and HTTPS traffic [Bla12]. When capturing traffic from our university network, we also eliminate flows from IP addresses that we know are NAT routers so we can make the assumption that each IP address we see from our network represents a single user.

3.3 Conclusions

In this chapter we have explained the different vantage points that can be considered for capturing web traffic. We have selected a client perspective as it is important for us to capture all the interactions of the client with the different servers it connects to. We have also decided to work with flow records instead of full pcap traces as they offer a manageable summary of web traffic which is not affected by encryption, does not break confidentiality, and is available in many network devices.

CHAPTER 4

Identifying website sessions

This chapter provides an initial approach to the problem of identifying distinct elements inside web traffic. We focus on finding website sessions in captured web traffic using only flow-level statistics. This is a complex problem because website sessions have very variable characteristics depending on the specific website and on the behavior of the user that visits it. However, a system able to find website sessions without relying on deep packet inspection would have immediate applications in accounting and characterization of websites and web users.

In the different sections of this chapter we introduce the problem at hand and some basic concepts. We explain the methodology used to capture the experimental data used throughout the chapter: we review the experiences of other authors in capturing web traffic and we present the portable test bed that we have designed for this purpose. Using real web traffic, we consider different useful parameters for the clustering process we propose, which we finally decide to base on connection start and end times and on server IP addresses. Finally, we test our clustering method obtaining promising results.

4.1 Introduction

We have previously seen that web traffic can be aggregated in different ways: from a simple request-response transaction to a complete browsing session that spans all the activity of a user during a relatively long period of time. One of the highest levels of aggregation is a *website session* which comprehends all the traffic generated by a user while browsing through webpages of the same website. To provide an example, lets consider a newspaper website. A typical website session may involve the user accessing the front page of the newspaper and then following links to one or more articles that he finds interesting (returning —or not— to the front page in between). The session will end when the user stops browsing through the newspaper's webpages whether because he accesses a webpage of a different website or because he ends his browsing session altogether.

In this chapter we present a method for identifying individual website sessions in web traffic. From our perspective, this means clustering together the connections related to a website session. When we started working on identifying distinct elements inside web traffic, we thought web sessions were an interesting candidate for two different reasons. On one hand, the traffic of a website session should span a limited period of time and involve an also limited number of servers (if we assume that the content of the different webpages of a website must be hosted in the same servers). This gives us two parameters to work with: time and server IP addresses. On the other, if we consider the different services provided today through the web, identifying website sessions has an special interest because they may be mapped to sessions of said services. For example, a Gmail session can be seen as a webmail session or a Facebook session as a social-network one. Being able to identify website sessions would thus allow gaining a lot of insight into the traffic generated by these services.

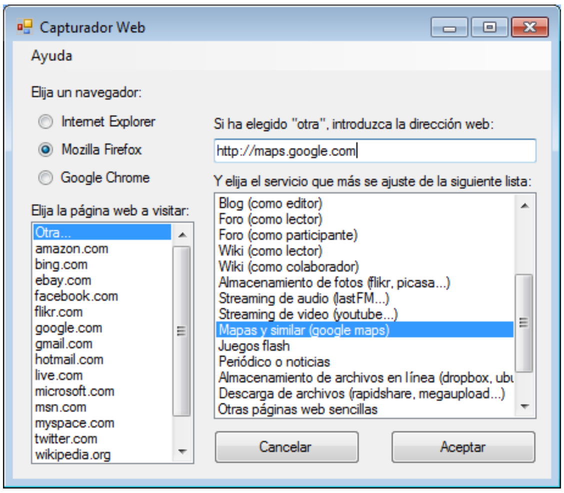
However, identifying complete website sessions is far from an easy task. It can be done —except for HTTPS connections— by carefully studying application data in order to identify the origin of every HTTP request. However, as explained in chapter 3 we work at flow level and that information is not accessible to us. Using only flow level information, website sessions are difficult to pinpoint. They are very variable in length depending on the service accessed and on the behavior of each particular user. Moreover, two sessions of two different websites can occur at the same time, for example, in different tabs of the same browser. This complicates establishing clear bounds for whole website sessions. As a consequence, the objective we lay out in this chapter is not to cluster together all the flows in a website session but, rather than that, to achieve a clustering in which the connections of a website session are grouped into the fewest (hence biggest) clusters while avoiding flows of different sessions being clustered together.

4.2 Capturing user sessions

As we stated previously, the objective we lay out in this chapter is to provide a method able to group HTTP connections that belong to the same web session. In order to study the features that these connections share and test our clustering method, it is necessary to use real web traffic in which every connection is labeled as belonging to a session. Even though a massive amount of Internet traffic (specially in pcap or NetFlow formats) is widely available to researchers around the world, traces this thoroughly labeled are hard to come by. In fact, even labeling a trace manually (by inspecting the payloads of the packets) might be impossible as the payloads are usually anonymized and, in those cases where they are not, they might not contain enough information to assign the connections to a specific website session.

Because of this, we have decided to prepare a traffic capture system that also provides labeling information (*i.e.* which websites the user visits) and use it to gather the traces required for the study. Capturing information about web user actions can be a challenging task [KHI+08]. Different authors have developed different techniques depending on the nature of the information they wanted to gather:

- In some cases, spyware applications have been installed in the users' computers (with their consent) in order to log their actions [KB04; KA02]. However, this provides a huge amount of data that has to be carefully studied to extract the interesting information about web behavior.



Elija un navegador:

- Internet Explorer
- [✓ Mozilla Firefox]
- Google Chrome

Elija la página web a visitar:

- [✓ Otra...]
- amazon.com
- bing.com
- ebay.com
- facebook.com
- flikr.com
- google.com
- gmail.com
- hotmail.com
- live.com
- microsoft.com
- msn.com
- myspace.com
- twitter.com
- wikipedia.org

Si ha elegido "otra", introduzca la dirección web: http://maps.google.com

Y elija el servicio que más se ajuste de la siguiente lista:

- Blog (como editor)
- Foro (como lector)
- Foro (como participante)
- Wiki (como lector)
- Wiki (como colaborador)
- Almacenamiento de fotos (flikr, picasa...)
- Streaming de audio (lastFM...)
- Streaming de video (youtube...)
- [✓ Mapas y similar (google maps)]
- Juegos flash
- Periódico o noticias
- Almacenamiento de archivos en línea (dropbox, ubu
- Descarga de archivos (rapidshare, megaupload...)
- Otras páginas web sencillas

Figure 4.1: Capture system user interface.

- Other authors have modified web browsers with extensions that add logging capabilities [Cap11] or even programmed a full modified browser from scratch [CLW+01].

- Finally, custom built logging tools have also been used [OWH04] which run alongside the browser and collect information from the user.

In our case, we have designed a traffic capture and labeling system that can be easily distributed among multiple test users in order to obtain the traces we need. This system involves a portable virtual machine (Windows XP) installed on a USB flash drive that can be executed in any Windows system. The virtual machine offers a controlled environment where any traffic captured should be directly related with web navigation. A C# program acts as an interface (see figure 4.1) in which the user must introduce:

- The URL of the website the user wants to visit.

- The desired web browser (Mozilla Firefox, Google Chrome or Internet Explorer).

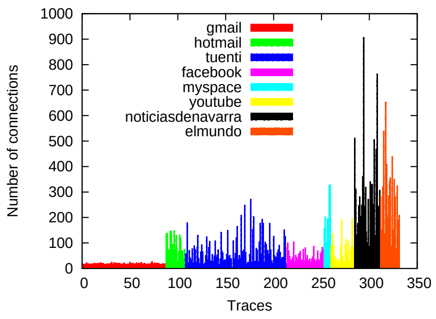
- The type of service that the user considers the website offers. This is chosen from a list (e.g. webmail, video streaming, social network, wiki, forum, etc.)

The program then launches the selected web browser, starts a windump process and waits for the user to close the browser window. The user is instructed to only follow internal links and not to open any new tabs or different websites. When the browser window is closed, the program sends the captured pcap data and a file with the labeling information to a server via SFTP. In order to keep the capture system as simple as possible, we capture the traces in pcap format. Argus is then run on the server to translate them into flow records calculating the parameters explained in chapter 3. Capturing the data in pcap also allows us to keep the additional information for other possible uses.

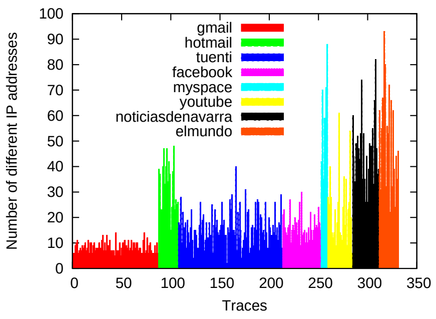
We are aware that this approach has a weakness in that it considerably limits the behavior of the users so the captured traffic may not always correspond to the one they would generate in normal conditions. This is specially true for advanced users who switch faster between websites and usually have more than one tab or browser window open at the same time. Nevertheless, we believe that this is the only way we can correctly assign every connection to a specific web session.

Using this methodology we have captured more than 300 sessions of different websites and services. For this study we have selected four services: web mail (hotmail and gmail, 107 traces), video streaming (youtube, 24 traces), social networks (facebook, tuenti and myspace, 151 traces) and online local newspapers (elmundo and noticiasdenavarra, 46 traces). Each of the traces contains an individual website session. The total number of captured sessions is not high but it is difficult to get web users to collaborate in a study of these characteristics without some compensation [KB04; KA02].

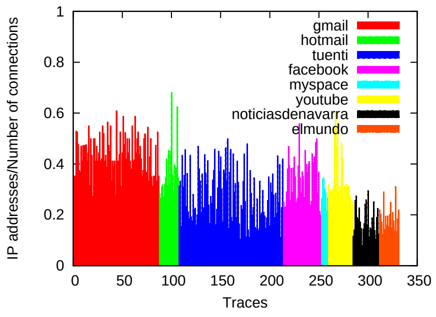
In order to provide a general outlook of the traces, figure 4.2 shows general parameters of each one. Traces have been ordered by service and website and a different color has been assigned for the traces of each website in order to facilitate comparison. In figures 4.2(a) and 4.2(b) the variability in the number of connections and different accessed IP addresses for each session is high. Nevertheless, it is possible to glimpse differences between the traces of different web-



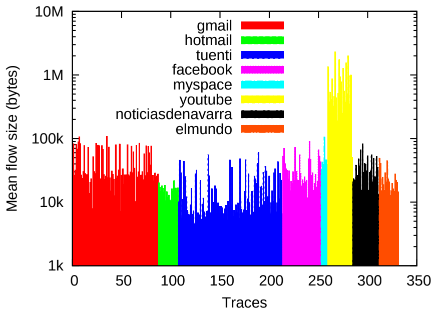
(a) Number of connections.



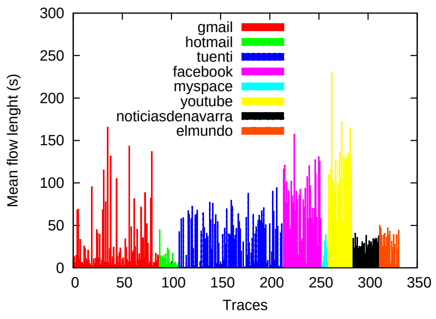
(b) Number of different IP addresses.



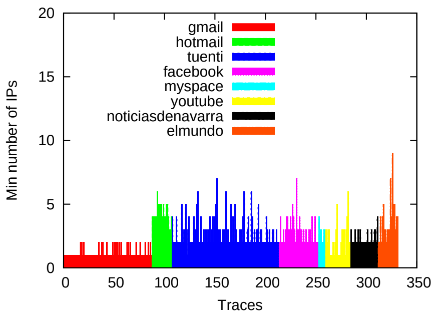
(c) #IP/#conn. ratio.



(d) Mean connection size.



(e) Mean connection length.



(f) Number of heavy IP addresses.

Figure 4.2: General description of the captured traces.

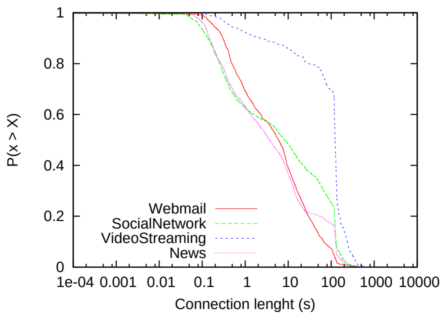
sites and services. For example, Gmail sessions tend to open fewer connections and access a more limited number of servers than the others and, among the social networks, Facebook and Tuenti have similar behaviors but not Myspace. Figure 4.2(c) shows the ratio of these two variables. When it is closer to one, it means that destination IP addresses are rarely repeated in the connections of the trace; when it is closer to zero, a small number of web servers are responsible for the majority of the captured connections.

The average size of the connections in each session is represented in figure 4.2(d). Youtube sessions stand out (even more considering that the scale is logarithmic) as they tend to produce bigger connections. In figure 4.2(e), which focuses on average connection length, the variability is high as connection length depends in many cases on when the user finishes the session (some connections may be kept open, albeit idle, until then by web browsers).

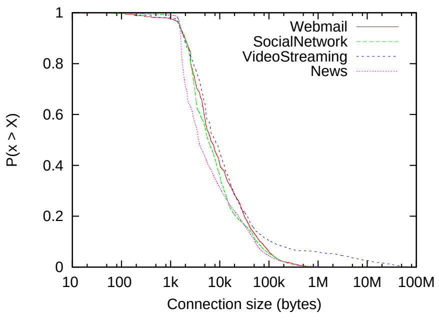
Finally, figure 4.2(f) shows the minimum number of different IP addresses which concentrate at least 80% of the total traffic of each session. In general, this number is low (mean = 2.5, median = 2) and its dependency on the total number of IP addresses present in the trace is not strong.

Some of the parameters shown in figure 4.2 show differences for sessions of different services. Therefore, they may be used in order to classify traffic depending on the service that generated it. However, these parameters refer to whole sessions not individual connections. As a consequence, a system able to cluster together connections of the same website session as the one we present here may be useful in order to produce identifiable clusters.

Now, focusing on per connection statistics, figure 4.3 shows the distributions of connection size (in bytes) and connection length (in seconds) for the experimental traces. Data from the different websites has been grouped in four bigger service categories (webmail, social network, video streaming and news). The complementary cumulative distribution functions (CCDF) of the two parameters are calculated for each trace and then aggregated by service. Even though most connections are relatively small both in size and length, there is a sizeable number of long and massive connections. This is partially a consequence of HTTP/1.1 (persistent connections stay open longer and are used for the download of multiple elements) but, the main cause are the different traffic profiles of



(a)



(b)

Figure 4.3: CCDFs of (a) connection length and (b) connection size for different services.

new services. For example, video streaming sessions and connections are longer than usual and the flows that carry the actual video are very heavy.

4.3 Clustering parameters

In this section we present some parameters that we believe can be useful in the task of clustering connections that belong to the same web session. In figure 4.3 the distributions of connection size and length were shown. They are, obviously, two very important parameters to describe a connection. Nevertheless, they are hardly useful in this task, as website sessions usually include multiple connections of varied sizes and lengths.

4.3.1 Connection interarrival times

It seems intuitive that connection interarrival times should be useful for clustering connection from the same sessions. When the user accesses a webpage, the web browser opens all the connections needed to download the different objects automatically and thus, rapidly. Later the user may follow a link or open another website prompting another group of automatically-opened connections. We expect, as a consequence, to find groups of connections that belong to the same session whose opening timestamps are very near and which are separated from other groups by the longer periods of time related to user actions.

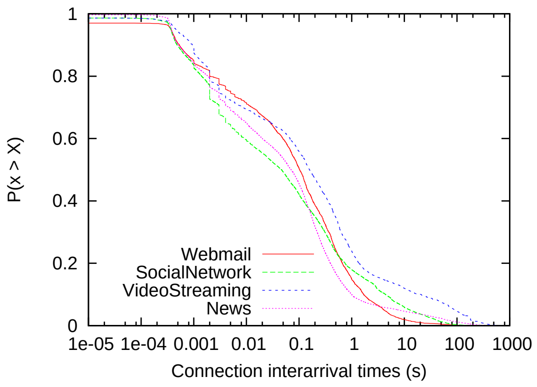
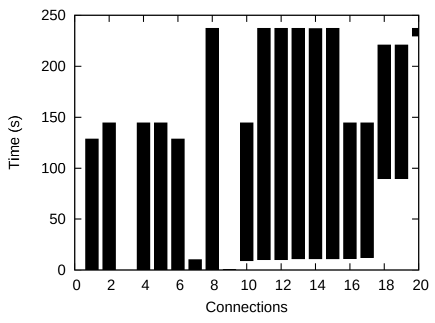


Figure 4.4: CCDF of connection interarrival times for different services.

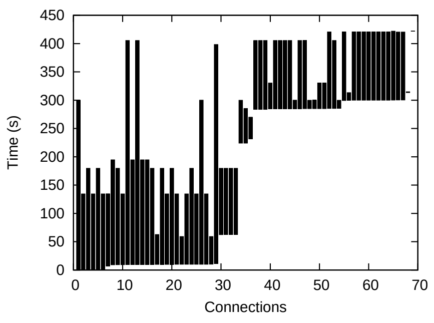
In order to validate this assumption, figure 4.4 shows the CCDF of the difference between starting timestamps of consecutive connections for the different services. We can see that the interarrival times of almost 50% of consecutive connection pairs that belong to same session are under 100ms. This is a very small period of time to think that both connections in the pair do not belong to the same session. It is also good that the distribution is similar for all the services as it indicates that this parameter does not depend on the type of webpage being loaded.

4.3.2 Connection overlap

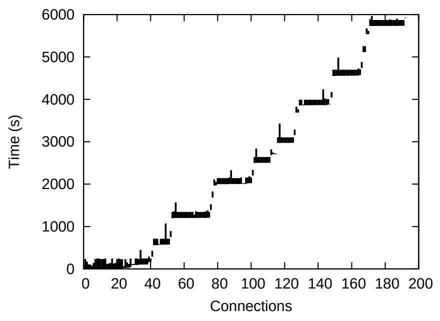
In the previous subsection we have stated that the interarrival times of a sizable number of connections in a website session are small. This is also true, in some cases, for the difference between the timestamps of the last packet of the connections. This happens because, with persistent connections, web browsers tend to leave some connections opened until the user closes the browser, the tab or loads a different website in the same tab. In order to show this behavior we provide three example figures —4.5(a), 4.5(b), 4.5(c)— of Gmail, Facebook and Youtube sessions where connections are represented as vertical bars with their bottom at the start timestamp of the connection and the top at its ending. Most traces yield graphics similar to these. In shorter sessions (in this case, Gmail), most connections are opened immediately as the session starts and closed when



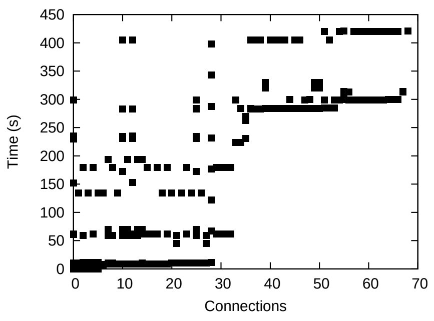
(a) Connection overlap in a (short) Gmail trace.



(b) Connection overlap in a Facebook trace.



(c) Connection overlap in a Youtube trace.

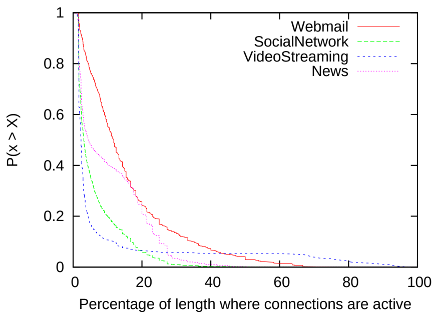


(d) Connection activity in a Facebook trace.

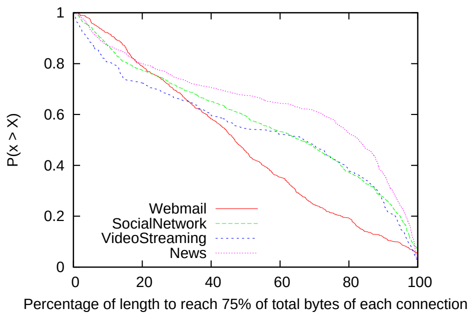
Figure 4.5: Connection overlap and activity for different websites.

it ends. Some are closed before (probably because their persistence timeouts expire) but, even then, their closing timestamps tend to coincide. In longer sessions this behavior repeats itself for each of the new webpages the user opens within the website he is visiting. However, even then, some connections may be kept open for all the website session's duration.

Anyway, even though browsers may leave connections open, this does not mean that they are active all the time. In order to study this, we have instructed Argus to inform about the activity of the connections every second (*i.e.* whether traffic is sent during that second or not). We will not, however, use this information as we want our clustering method to depend only on parameters calculated for the whole connections. As an example, figure 4.5(d) shows the same Facebook trace as figure 4.5(b). In this case we have represented every connection as



(a) Connection activity.



(b) Connection activity concentration.

Figure 4.6: Connection activity for different services.

a collection of points in the one second intervals where traffic is sent or received. If we compare both figures we can see that connection activity is concentrated on few intervals of their total length. Figure 4.6(a) shows this for all the traces divided by services like in previous cases. More than 60% of the connections are inactive in more than 80% of the one-second intervals of their length. This is specially extreme for video sessions where most connections are inactive during much of their length but a few (the ones that carry the actual videos) are active throughout most of it. In figure 4.6(b) we try to show if traffic is specially concentrated at the beginning or ending of the connection. In order to do this we represent the CCDF of the percentage of the length of each connection necessary to reach 75% of the total traffic of the flow. If traffic in the connections were uniformly distributed (i.e. we would reach x% of the traffic at x% of the length) the CCDF should fall sharply around the 75% mark. This does not happen as traffic in bigger connections tends to be concentrated at their beginning. It is worth mentioning the stark difference between the different services even though, as previously stated, we do not intend to use this information for identification purposes.

In brief, we have seen that connections that belong to the same session tend to overlap in time but, as they are not active during most of their length, the overlap is not really inherent to web traffic. As it will be discussed in section 4.4, we do use ending timestamps of connections in our clustering method although not in a direct way. It should be noted that when the user closes the web browser,

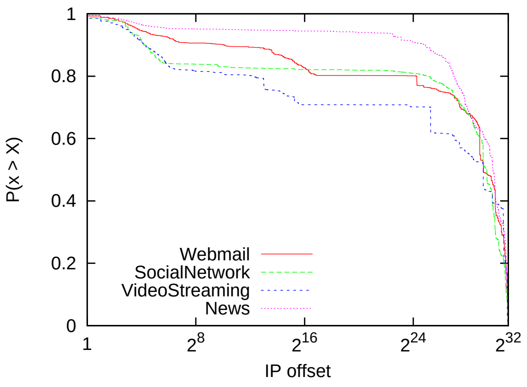


Figure 4.7: IP offsets.

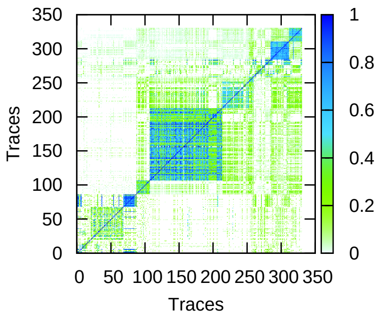
connections related to different websites may also be closed at the same time leading to them being added to an incorrect cluster.

4.3.3 Shared and near IP addresses

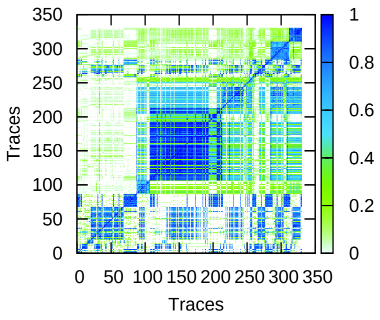
Server IP addresses are also intuitively a good parameter for flow clustering. During the load of a webpage multiple connections to the same servers may be opened and is easy to cluster them by their shared IP address. Moreover, even when objects are downloaded from different servers it is to be expected that, in some cases those servers will be in the same networks (if they belong to the same company or institution) so their IP addresses will be near. Nevertheless, there are situations where IP addresses may be misleading (shared content, CDNs, etc.) so we should be careful when using them.

We saw how multiple connections are opened to the same IP addresses for each session in figure 4.2(c). To represent near IP addresses in a session we calculate the *IP offset* as the difference between a pair of IP addresses expressed as unsigned integers. In figure 4.7 the IP offset between pairs of destination IP addresses present in the traces is represented. The IP offset is calculated for all pairs of each trace and then the results are aggregated. As we see, for video streaming and social networks, around 20% of IP pairs have an offset lower than 256, the size of a class C network.

It is necessary, now, to study if IP addresses (shared or near) happen in dif-



(a) Ratio of connections that share server IP.



(b) Ratio of connections with near server IP.

Figure 4.8: Shared and near IP addresses.

ferent sessions. In figure 4.8(a), the color of each dot represents how many of the connections of a particular trace (divided by the total number of connections in that trace) share their server IP address with connections in other trace. Traces are ordered as they were in figure 4.2. As expected, traces of the same website share a lot of IP addresses. However, it also happens between traces of different websites as connections of certain services (e.g. Google Analytics) may be present in both. Moreover, some websites may download content from CDNs or be hosted in shared servers.

Figure 4.8(b) is similar to its companion but in this case server IP addresses do not need to be equal but near. We have chosen 256 as a nearness threshold as it is, again, the size of a class C network. Given this figure, although we still believe that IP nearness is useful to cluster flows that belong to the same session, it must, as we said, be used with caution as it is present in many pairs of connections of different websites (this seems to be also related to content distribution networks).

4.4 Clustering method

In this section we explain the operation of our clustering method. Our objective is to group the connections that belong to the same website sessions in the smallest number of clusters while trying to avoid mixing connections from different sessions in the same cluster. The only information used for each connection is the times of capture of its first and last packet, its size in bytes, the total number of packets in the connection and the server IP address (we only consider connections opened by the client). All this data, as stated previously is collected by Argus from pcap traces and is normally provided by NetFlow.

In order to carry out the clustering, we have designed a two-step procedure represented in figure 4.9. In the first step (in white in the figure), we only attempt to cluster contiguous connections. We consider that two contiguous connections should be in the same cluster if they fulfill at least one of two conditions:

1. Their start timestamps are very near (so near that it is improbable that they are related to different user actions).

2. They share the same destination IP address.

Given this, in this part of the procedure clusters only grow while one of this conditions is met by the last connection in the current cluster and the new candidate. When this does not happen, a new group is created for the new connection. The next one will attempt to join that new group.

The second part of our system (in grey in figure 4.9) is a process that is called whenever there are clusters in memory that are considered old. A cluster is old if the last packet captured that belongs to the cluster was captured more than a number of seconds ago. When a cluster is old, our system tries to join it with a newer cluster. We use two parameters to do this. On the one hand, we define the length of a cluster as the time elapsed between the capture of the first and the last packet in the cluster (i.e. the smallest of the start times of the flows in the cluster and the biggest of the end times). Consequently, we define the *time overlap* of two clusters as two times the period of time both of them exist divided by the sum of both their lengths. Therefore, the time overlap varies from 1 when the clusters' start and end times coincide exactly to 0 when they are never present at the same time. On the other hand, we say that two IP addresses are near if, when expressed as unsigned integers, the difference between them is smaller than a certain value. We now consider the amount of bytes in each cluster that belong to connections whose destination IP addresses are near to those of some

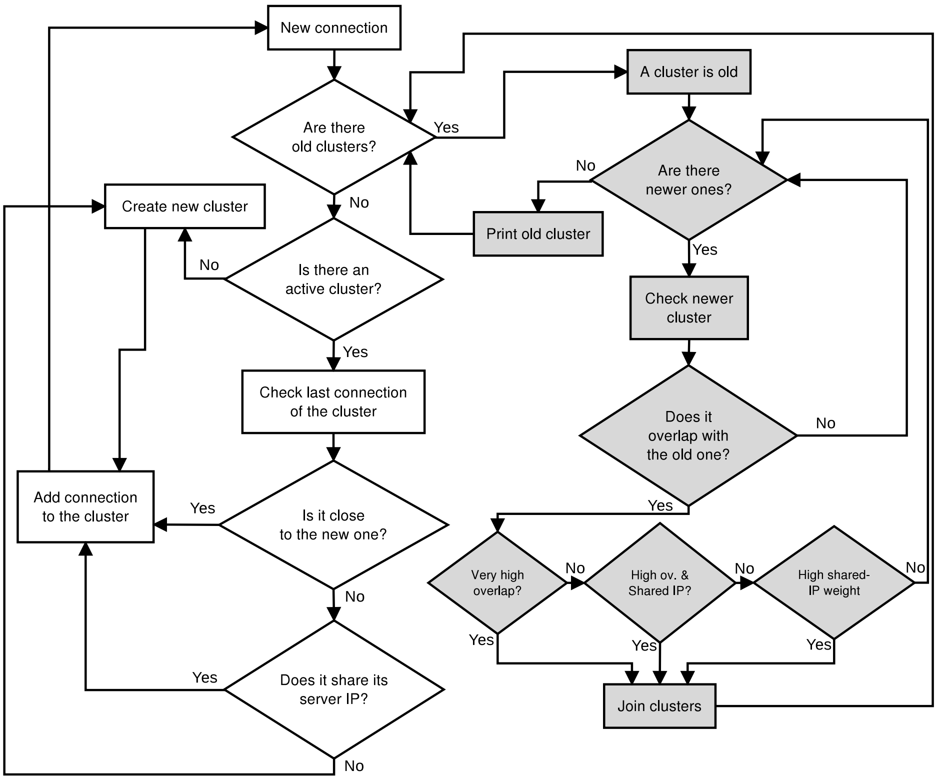


Figure 4.9: Clustering method.

of the connections in the other cluster. We define the *shared-IP weight* for each cluster as that amount of bytes divided by the size of each cluster. Using these two parameters, we join two clusters if:

- The time overlap between both of them is very high.
- The time overlap is high and the clusters share at least one IP address (or have near IP addresses).
- The time overlap is low but the shared-IP weight is high.

Finally, if an old cluster could not be incorporated into a newer one, it is taken out of the system.

As it can be inferred from the previous paragraphs, our method depends on a series of thresholds that need to be tuned in order to achieve a correct operation. These are:

- The time interval after which a cluster is old so the grouping process is carried out for it. This interval is related to the time elapsed between user actions. If a cluster is older than the average time between user actions it should be grouped with newer clusters or taken out of the system. Various studies have tried to obtain a reference value for the interval between user actions (through clickstream analysis). We will use 30 seconds as suggested in [BMM+09].
- The maximum difference between the start timestamps of two connections that are considered very near. In figure 4.4 we saw that the inter-arrival time of consecutive connections in the traces was less than 100ms for around 50% of connection pairs. As this is a time interval short enough to avoid confusion with different user actions we will use it.
- The maximum difference between IPs (expressed as integers) which are considered near. We use 256 as it is the size of a class C network.
- The minimum time overlap value considered very high (95%).
- The minimum time overlap value considered high (50%).
- The minimum IP overlap value considered high (50%).

4.5 Results and discussion

4.5.1 Individual traces

Our first approach to testing our clustering method has been to use the collection of session traces described in section 4.2. Using the thresholds specified previously, we have obtained the results that compose table 4.1. We provide, for each service, mean values of the number of clusters the method found in each trace and the number of clusters that include, at least 90% of the traffic of

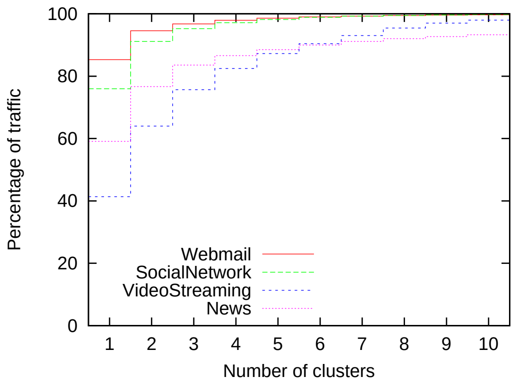


Figure 4.10: Aggregated traffic in n clusters (%).

each trace. This is interesting as small clusters (most of them with just one connection) occur frequently but are not very significative. As we see, most of the traffic is grouped in few clusters specially considering that the captured sessions can be very long and comprise hundreds of different connections.

Service	**Number of clusters**	**Number of clusters for 90% traffic**
Webmail	5.88	1.65
Social network	7.00	2.63
Video streaming	13.20	3.20
Newspapers	39.32	6.46

Table 4.1: Mean numbers of clusters by service.

Results for the same traces are shown in a different way in figure 4.10. In it, we have aggregated, again by service, the traffic of the different traces in order to show how much of it is included in a specific number of clusters. Except for the video streaming service, the biggest of the clusters discovered in each trace contain, in average, more than 50% of the total traffic of the session. The case of video streaming is slightly different as during long youtube sessions, users normally watch multiple videos. If the connections carrying two videos (which are responsible for the biggest part of the load) are grouped in two different clusters, both will have a significant weight even if one of them is composed by a lot more (smaller) connections than the other. Anyway, the four curves grow fast and in average 90% of the traffic is grouped in 6 or 7 clusters in the worst case. It is worth noting that although table 1 and figure 4.10 show the same results, they are not calculated in the same way and may seem inconsistent. This happens because in table 1 all traces are given the same weight when calculating the means regardless of the total traffic in each of them. In figure 4.10 as the total traffic of the service is considered, bigger traces have a bigger effect than smaller ones.

4.5.2 Normal web navigation

The next step is to test the method with traces of normal web traffic that contain multiple sessions of different services in order to check if the algorithm is able to distinguish between them. As stated in section 4.2, the biggest challenge here is labeling the traces indicating to which session each connection belongs. We have captured two traces of about thirty minutes duration each with traffic of the websites previously introduced. In these traces some different sessions are kept open at the same time as it will happen normally. We have labeled the connections manually using Whois information, analyzing previous DNS requests and looking into HTTP headers. Nevertheless, even giving this much attention to each connection, in some cases the classification is doubtful. For example, it is difficult to know if some secure connections to Google servers belong to Gmail or Youtube sessions.

Table 4.2 shows the performance of our method when classifying traffic from those traces. We say that a connection is correctly classified when it has been aggregated into a cluster where the majority of connections belong to its same website. We calculate percentages of recall (the rate of connections correctly labeled as being caused by and application out of the total connections of said application) and precision (the rate of connections that actually are caused by an application out of the total connections labeled as caused by the application) as defined in equations 2.1 and 2.3 respectively.

As we can see, results are generally good. There are some problems between gmail.com and youtube.com sessions as both of these services are hosted

Trace 1

Website	Recall (%)	Precision (%)
gmail.com	43.2	100
facebook.com	100	94.3
youtube.com	100	79.2
noticiasdenavarra.com	100	100

Trace 2

Website	Recall (%)	Precision (%)
gmail.com	36.6	100
facebook.com	98.4	93.8
youtube.com	100	62.9
elmundo.es	97.6	100

Table 4.2: Clustering for two normal web traffic traces.

in Google servers. If a Gmail session is open while visiting Youtube, its connections tend to be included in Youtube clusters. These problems are almost unavoidable with services that are so interrelated.

4.6 Conclusions

In this chapter we have presented a method to cluster connections that belong to the same web session using only the information provided by flow records. We believe that this is interesting as the resulting clusters will contain more information about the session than the individual connections and thus will be more easily classified into the different services that are provided through the web.

In order to design our method we have previously captured and studied multiple sessions to four popular web services. We have prepared a capturing and labeling system that can be easily distributed but has some limitations as it can only gather traces of individual website sessions. We find that labeling the connections of a trace with multiple different sessions is complex even when doing it manually.

We have selected flow start and end timestamps and server IP addresses as the most interesting parameters in order to cluster the connections. Nevertheless, they all have their drawbacks which our method attempts to minimize: connections of concurrent sessions may start at the same time, when the connections are closed depends heavily on the web browser implementation and server IP addresses may be shared by connections of different services if, for example, they are provided by the same company.

Experimental results show that connections are grouped in a relatively small number of big clusters for each session and that recall and precision are generally good for the services tested.

CHAPTER 5

Finding server IP addresses related to specific websites

This chapter provides a method for finding server IP addresses related to specific websites through the study of a traffic trace. Again, our study relies on flow-level statistics to design a method that could work with NetfFlow-type records. We hypothesize that certain IP addresses are closely related to specific websites and that we should be able to identify them by studying multiple sessions of those websites. The method we present in this chapter could have applications in identifying the traffic of particular websites (and prioritize, filter or simply characterize it) and could be used to label flow records of web traffic.

In the different sections of this chapter we describe our approach to the problem we face. After carrying out a preliminary testing with traces of real web traffic, we introduce a method based on the concentration of appearances of an IP address in sessions of a website and on the appearance of IP addresses of the same subnetworks. We test said method and we validate it by manually checking a sample of the identified connections. Finally, we introduce various improvements from the website selection process, to the definition of website sessions and the validation procedure.

5.1 Introduction

When working with web traffic, many connection-level statistics may be informative about the kind of websites that are being accessed and the content hosted in them. For example, heavy connections can be related to embedded video or long ones may be caused by websites that users tend to leave open for extended periods of time (such as webmail or social networks). However, the only connection-level parameter that can be used in order to identify the specific website responsible for a connection is the server IP address. In the early days of the web, this identification was straightforward as websites were usually hosted in one or a few dedicated servers. Nowadays the situation is more complex because websites often include a sizable share of third-party content and even first-party content may be hosted in third-party servers (*e.g.* CDNs). As a consequence, the IP addresses that are accessed when loading a website change over time and some of them may be used by more than one site. Nevertheless, even taking this into consideration, many servers are still primarily dedicated to hosting specific websites, at least if we consider their operation during a limited time frame.

Our objective in this chapter is to develop a method able to extract, from a trace of captured web traffic, a list of server IP addresses that are strongly related to a predefined set of websites. This problem could be approached for unencrypted traffic by studying HTTP headers in order to gather information about the servers accessed in the trace. However, our connection-level perspective does not allow us to access packet payloads. Rather than that, our approach is to gather multiple sessions of the websites under study in the traffic trace and identify related servers as those that are frequently connected during those sessions. Intuitively, if we capture enough sessions of the same websites, a number of server IP addresses are bound to start appearing repeatedly. Also intuitively, these IP addresses must belong to servers that store the fundamental content of the site as opposed to some images, videos, advertisements and other rapidly-changing or third party content.

Having a list of server IP addresses associated with a specific website can be useful in different ways. In the first place, this information is interesting for accounting purposes as it allows network administrators and ISP operators to easily know which websites are being accessed by their users. Instead of focusing on user habits, this information can be used to isolate the traffic of particular websites in order to characterize and model it. Finally, we have seen in chapter 4 that, when designing any type of traffic classification system, one of the biggest challenges to overcome is the difficulty of finding correctly labeled traffic traces that can be used as ground truth in order to tune or test the system. These traces are so hard to come by because the labeling process is not trivial. In cases where traffic must be labeled according to the application that generated it, some simple (albeit somewhat unreliable) techniques exist such as port mapping or signature-based classification (see section 2.2). However, in chapter 4 we needed traces in which web connections were labeled according to the website sessions they belonged to. A list of popular websites and their associated server IP addresses could be used to tackle the labeling process in that case.

5.2 Problem statement

We have stated that our objective in this chapter is to develop a method that, studying a traffic trace, finds server IP addresses related to specific websites. Our approach to the problem is to study different sessions of the same websites and decide which of the servers accessed during these sessions are related to said websites. We could divide this problem into two smaller ones: finding website sessions and defining a relationship criterion for IP addresses.

5.2.1 Rough website sessions

In chapter 4 we presented a method for identifying sessions to websites in real traffic. However, although the results were promising we were unable to validate it adequately because we did not have access to properly labeled traces (which partially prompted this research). Nevertheless, in that case, we were more concerned with preventing connections of different sessions from mixing rather than identifying the whole sessions. Here, that concern is not so important and we consider website sessions in a different way.

We define a *rough website session* as the set of connections opened by a user while accessing a website. That is to say, a rough website session of a particular website includes the connections opened to download the content of the visited webpages inside the website and other web connections that the user may have opened in the same time frame (some of which may belong to concurrent sessions to other websites). For example, a session of a webmail site would ideally span all the connections opened by the web browser from the moment the user opened the login webpage of the mail service until he closes the browser, the tab or opens a different website in the same tab. However, it is possible that the user accessed a different website in another tab while checking his e-mail and the resulting connections will also be part of the first rough website session. It is, thus, the time interval what defines a rough website session and it reduces the problem of finding them to deciding when they start and how long they are.

Even with this simplification, detecting start times of rough website sessions is not easy just by inspecting captured traffic. However, we seek to identify IP addresses related to a predefined set of websites and we can use this to our advantage. We use connections to specific server IP addresses as marks of the beginning of a session. We call them *main IP addresses* and they are associated to the websites under study through DNS information. Main IP addresses are associated to the ″main″ domain name of each website (i.e. www.facebook.com rather than, for example, s-static.ak.facebook.com) and, because of this, we expect that connections to them will happen at the beginning of the sessions of said websites.

Once main IP addresses identify the first connection of a rough website session, we define *session length* as the interval of time during which we consider that the user is still visiting the same website. We will set an adequate value for session length after studying our datasets in section 5.4.1.

For a simple graphical representation of rough website sessions and the concepts defined in this section see figure 5.1 in which each impulse represents the start of a different connection. Throughout this chapter, we will often refer to rough website sessions simply as ″sessions″ for the sake of brevity.

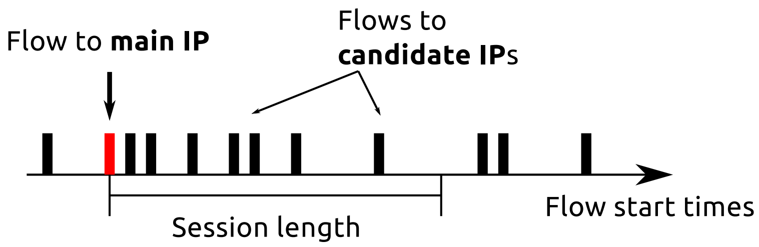


Figure 5.1: A rough website session.

5.2.2 Candidate IP addresses

Once we find a number of rough website sessions of a specific website we consider all the server IP addresses accessed during those sessions as *candidate IP addresses*. Candidate IP addresses can be related to the websites or not. As we said, rough website sessions do not ensure that every connection in the session is related to the website; they only offer a time frame during which the session is taking place. Because of this, candidate IP addresses may be accessed as a consequence of a concurrent session to another website or even because of a different application using web ports.

It is thus necessary to establish a criterion that decides if a candidate IP address is truly related to a website or not. Our reasoning is that, if we study enough sessions of the same websites, servers that are closely related to them will be accessed often. Therefore, we should be able to distinguish them from unrelated servers that are accessed only when another session is coincidentally open at the same time. This idea is the basis of the different decision parameters presented in section 5.4.2.

5.3 Data collection

5.3.1 Traffic traces

For the different tests presented in this chapter, we use three traffic traces captured in the Internet link of the Public University of Navarre. The traces were captured respectively during June, October and December 2011 and basic information about them is shown in table 5.1. We use Argus to obtain flow records

Table 5.1: Traffic traces.

Trace	Date of capture	# Flows	# Users
Trace 1	Jun 5 - 20	90M	1022
Trace 2	Sep 29 - Oct 10	85M	916
Trace 3	Dic 14 - 27	76M	857

with the information described in chapter 3. As we explained in that chapter, we only consider connections originated in clients inside our network and directed to outside server TCP ports 80 and 443 (HTTP and HTTPS traffic). We also eliminate flows from IP addresses that we know are NAT routers so we can make the assumption that each IP address we see from our network represents a single user.

5.3.2 DNS records

As stated in section 5.2.1, we use the fact that we are working with a predefined set of websites to find main IP addresses that mark the beginning of sessions. We have selected 40 sites, some of them worldwide known and others which are popular in Spain and Navarre (e.g. Tuenti, a Spanish social network or a number of local newspapers such as El País or Diario de Navarra). At the same time of the traffic capture, we use a PC in our network to resolve the domain names of these sites with automatic hourly DNS requests. We store the IP addresses obtained in the DNS responses which will be our main IP addresses.

We observed that, for websites that belong to the same company, sometimes servers that offered a website would later offer another. In fact, in those cases, the sites usually shared a lot of content making it difficult to differentiate them with our method. We decided to put the websites from the same companies together in groups reducing our list to 28 websites or groups of websites. This affects Google websites, Microsoft's and groups national and international versions of others. In table 5.2, as an example, we show a list of some of the websites with the number of IP addresses obtained with the DNS requests for the October data set (for briefness we only show international websites omitting locally popular ones).

Table 5.2: A selection of the websites under study.

Website or group	Number of main IPs	Domains
Amazon	6	amazon.com amazon.co.uk
Google	47	blogger.com docs.google.com gmail.com google.com youtube.com youtube.es
Ebay	6	ebay.com ebay.es
Facebook	8	facebook.com
Flikr	2	flikr.com
MSN	12	hotmail.com hotmail.es live.com live.es msn.com msn.es
Megaupload	10	megaupload.com megavideo.com
Tumblr	2	tumblr.com
Twitter	4	twitter.com
Wikipedia	1	wikipedia.org
Wordpress	3	wordpress.com
Wordreference	6	wordreference.com
Yahoo	7	yahoo.com yahoo.es

5.4 Preliminary testing

In section 5.2 we described our approach to the problem and defined rough website sessions. However, in order to find a good value for session length and test different parameters that check if a candidate IP address is truly related to a website, we rely on measurements obtained from our captured traffic. The experimental data we present in this section corresponds to the September-October traffic trace which we have used to tune our system. The results for the other two traces are similar.

5.4.1 Finding rough website sessions

We have defined a rough web session as the set of connections generated by the web browser while the user is accessing a specific website. We should remember that, even if we know when a user is accessing a website, we have no way to assess if the connections are truly caused by the load of that website. Users may open concurrent sessions (a frequent happening given the widespread tab-based design of web browsers) and may use other applications that use the ports normally associated to HTTP(S). Even web browsers may open certain connections unrelated to the visited websites. Because of that, our web sessions will comprise all the connections opened during a website access whether they are related to it or not.

We use the IP addresses that we obtained from the domain names of the websites as main IP addresses which signal the beginning of a session. In other words, when we find a connection from a user to one of the main IP addresses, we consider that the user is visiting the corresponding website. The following connections initiated by that user will be considered part of the same session. Their server IP addresses will be then candidate IP addresses that, in the end, may or may not be associated with the website.

Choosing an indicator for session ending is, however, not simple. Sessions are variable in length depending on the type of service accessed and on user behavior. We have chosen to set one fixed value for session length as setting one for each website and user profile would be too complex and hardly scalable. Choosing a big value for session length will result in long website sessions. Intuitively

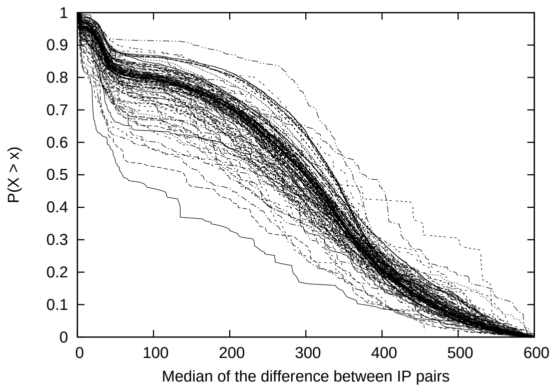


Figure 5.2: Time differences between IP pairs.

this will allow a better labeling of the IP addresses of websites that usually are related to long sessions (like, for example, online newspapers). However, there is a drawback: as we increase the value of session length, the probability of overlapping sessions of different websites also increases.

To set this value we have studied the time differences between the connections to a main IP and the next connections to candidate IP addresses made by the same user. We expect that connections to candidate IP addresses that do belong to a certain website will be on average closer to the main IP than those related to other websites. In this case the mean of the time differences proved to be too affected by extreme values so we opted to calculate the median for every main/candidate IP pair.

In figure 5.2 each curve represents the distribution of these medians for a main IP limited to differences smaller than ten minutes. The flat area around 100 seconds suggests that connections to candidate IP addresses related to the main IP happen before, while the points to the right of the graphic correspond to IP addresses that belong to other sessions. In view of this, we have chosen 120 seconds as a value for session length. In any case, as we are studying multiple sessions for each website, it is not necessary that each session comprehends all the related connections. As a consequence, the exact value we choose for session length is not a critical factor in the identification process.

The process by which rough website sessions are assembled from the connec-

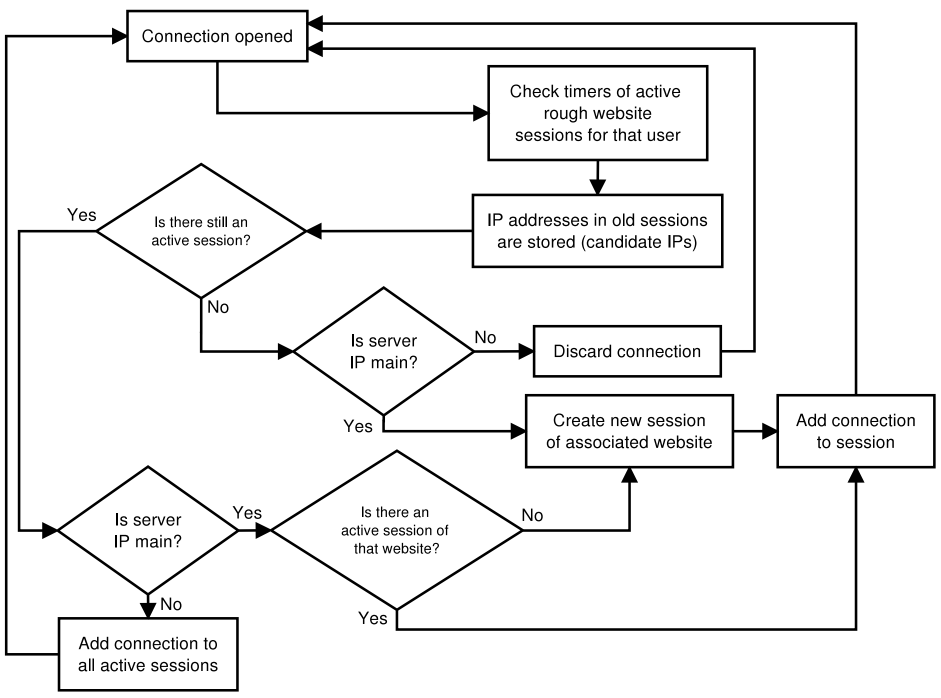

Connection opened
Check timers of active rough website sessions for that user
IP addresses in old sessions are stored (candidate IPs)
Yes
Is there still an active session?
No
Is server IP main?
No
Discard connection
Yes
Create new session of associated website
Add connection to session
Is server IP main?
Yes
Is there an active session of that website?
No
No
Yes
Add connection to all active sessions

Figure 5.3: Grouping connections in rough website sessions.

tions in the traffic traces is explained in figure 5.3. As it can be seen in the figure, connections to main IP addresses create new sessions unless there is already an active session of the same website. In that case, they are simply aggregated to that session. Connections to other IP addresses, on the other hand, are assigned to all the currently active sessions unless there are none, in which case they are discarded. When the session length timer expires for an active session it is considered "old" and it exists the program. The IP addresses in that session are recorded as candidate IP addresses of the associated website.

5.4.2 Decision parameters

Using the definition of rough website session, the main IP addresses obtained via DNS in section 5.3.2 and the session length calculated in the previous subsection, we assemble a set of rough website sessions from the traffic trace. Each session is related to a user which, as defined, is the source IP address in every connection. Each session is also related to a website depending on the main IP address of the connection that started the session. By considering all the sessions related to a website, we gather a list of candidate IP addresses for that website.

Now, we need to decide which of these candidate IP addresses are truly related to the websites. As a first step, for each website, we will only consider candidate IP addresses that appear in two different sessions of two different users. This allows us to eliminate a big number of candidate IP addresses that are not strongly related to the website or may appear in its sessions because of the specific behavior of a single user. In order to make this requirement fair for sites with a very different number of sessions we add the condition that the candidate IP address must appear in, at least, 1% of the total sessions of the website. With this step, in the September-October trace we obtain a total of 2011 candidate IP addresses for all the 28 websites. Out of these, 1355 appear only in sessions of one website and 656 appear in sessions of more than one website.

It is clear that, at this stage, we cannot assign the doubtful 656 IP addresses to any website. But we cannot do it either for the 1355 candidate IP addresses that appear only in sessions of one website. Some of them will be related to that website but others may belong to websites that we did not consider in 5.3.2. As a consequence, we need to find decision parameters that allow us to distinguish between these two cases. Ideally, with these parameters we will also be able to assign some of the doubtful IP addresses to the correct website if they appeared in the sessions of another just because of incidental overlapping of the sessions.

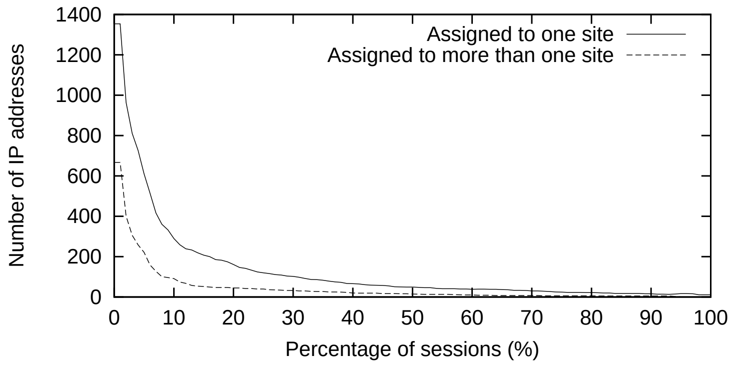
We hypothesize that the doubtful IP addresses can act as an indicator of how good the chosen decision parameters are. We want to minimize that number as it is reasonable to think that reducing the number of IP addresses that are doubtful between the websites under study will also reduce the number of doubtful IP addresses with other websites.

We now present three different decision parameters:

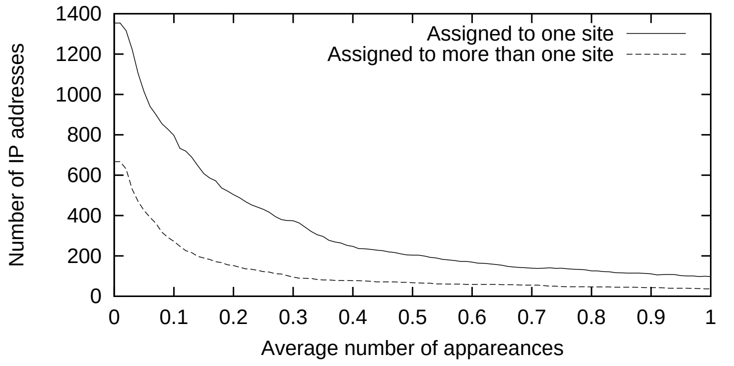
Percentage of sessions: The first decision parameter we consider is the percentage of sessions of the website in which each candidate IP address appears. As stated previously, we expect IP addresses that host important content of a website to appear in most of the sessions of that website. We will, in this case, assign a candidate IP address to a website if it appears in more than x% of the sessions of that website. Those that do not meet the threshold for a website will then be eliminated from the website list. A high percentage should avoid confusion as it is extraordinary that users always open concurrent sessions to the same websites. In figure 5.4(a) we represent a sweep of the percentage of sessions in which candidate IP addresses appear. The solid line represents the number of candidate IP addresses only assigned to one website and the dotted one, the number of doubtful candidate IP addresses. The results are unexpected as IP addresses that appear in most sessions of a website are rare. In fact, normally only one or two addresses will appear in more than 50% of the sessions. Most of the content of the sites must be either dynamic or hosted dynamically. As a consequence, this is not a good parameter for our purposes as the number of related IP addresses it is able to find is very small. Moreover, this parameter is not useful if we want to minimize the number of doubtful IP addresses because the number of IP addresses assigned to more than one website does not decrease fast enough as we increase the assignation threshold. By analyzing the trace, it becomes clear that most of these doubtful IP addresses are related to web tracking services for which connections are opened during sessions of many different websites.

Average number of appearances in sessions: Still working with the same idea we propose a slightly different parameter. Connections to IP addresses related to a website not only may appear in multiple sessions of that website but may appear multiple times in each session. Figure 5.4(b) is similar to the previous one but now the parameter is the average number of appearances of a candidate IP per session. We sweep the threshold from 0 to 1 although its value could be higher. In any case, we see little improvement.

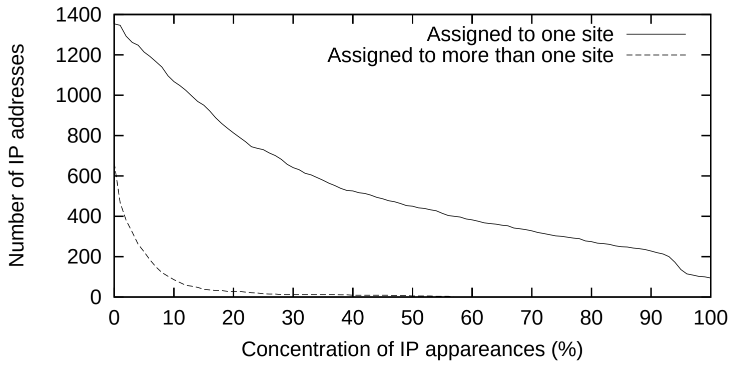
Concentration of IP appearances: Figure 5.4(c) represents a different approach. For every candidate IP in a website list, we have gone through the complete flow record and counted how many connections were made to that IP and how many of them were opened during the sessions of the website. We de-



(a)



(b)



(c)

Figure 5.4: Three decision parameters.

fine the *concentration of IP appearances* for a candidate IP address of a website as the ratio between the number of appearances (i.e. TCP flows) of the candidate IP in sessions of the website and the total number of appearances of the IP in the trace. It is important to notice that connections to a candidate IP may be opened outside the corresponding website sessions. For one thing, sessions, as they have been defined, may not encompass all the actual connections. Furthermore, a website may be accessed without a previous connection to a corresponding main IP when, for example, following a hyperlink. Nevertheless, a high value of the concentration of IP appearances strongly suggests that the candidate IP is related to the website. As shown in the figure, this parameter proves to be better suited to our purposes: the number of doubtful IP addresses decreases rapidly as we increase the threshold while the number of IP addresses assigned to only one site is still acceptable.

5.4.3 Candidate IP subnetworks

During the study of the candidate IP addresses of each website it became clear to us that a sizable number of them belonged to the same IP subnetworks. This is something to be expected as the websites host part of their content in servers that belong to the same company and the IP space that each company uses is limited. In the previous subsection we have seen that the percentage of sessions of the website in which the candidate IP appears is not a good decision parameter because very few IP addresses are identified with it. However, we hypothesize that this parameter may be more useful if we consider not the individual IP addresses but groups of them that belong to the same IP subnetworks.

In order to group candidate IP addresses into networks, we consider them as 32-bit unsigned integers and we define the *distance* between two of them as the subtraction of those integers. In figure 5.5 we represent the distribution of the distances between each pair of candidate IP addresses with one curve for each website. The scale in the abscissa axis is logarithmic and we have zoomed into the left side in order to show the relevant (i.e. small) distances. As we can see, most of the distances between IP addresses are very big (in the majority of cases, they are bigger than a class A network size). This is not strange as we are now working with a huge number of, mostly uninteresting, candidate IP addresses

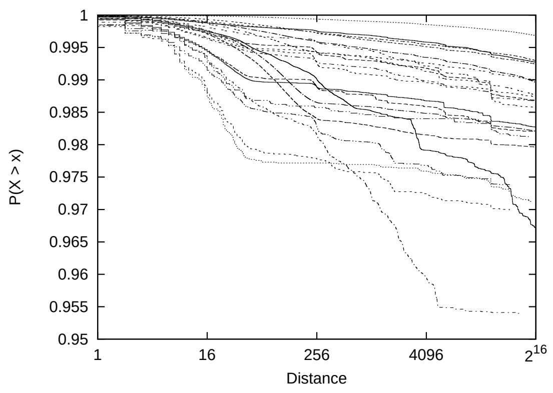


Figure 5.5: IP distances distribution.

(e.g. for Google, more than forty thousand IP addresses). However, if we focus on the small distances, we can see that, for most websites, there are flat areas in the distribution of the distances between IP addresses that may indicate the maximum distance between IP addresses of the same network. Studying the figure we have decided to use candidate networks of class C size so that the maximum distance between IP addresses in them is 255. This is fairly restrictive as usually no more than 2% of the distances are that small.

Using that threshold we proceed in the following way:

- We group the candidate IP addresses of a website in IP subnetworks. In this case, we do not eliminate candidate IP addresses that do not appear in, at least, 1% of the sessions of each website as the filtering will come after, when we consider the whole subnetworks. We only consider a class-C subnetwork if there are connections to, at least, two different server IP addresses that belong to said network in sessions of the site.

- We calculate in how many sessions of the website a connection to an IP from the subnetwork appears.

- We filter the candidate subnetworks according to a *subnetworks session threshold*. We will assign a subnetwork to a website if connections to it appear in a percentage of its sessions that is higher than the threshold.

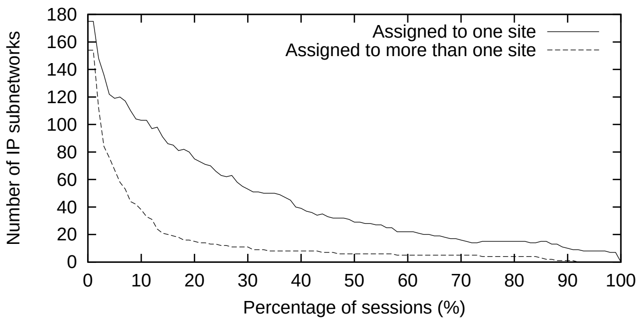


Figure 5.6: Percentage of sessions for IP subnetworks.

- If a subnetwork has been assigned to only one website, we will label the IP addresses in that network as belonging to the website even if some of them did not appear in its sessions. If, on the other hand, a subnetwork has been assigned to more than one website we will discard it.

A subnetwork will be assigned to more than one website primarily because of two reasons: it is a CDN subnetwork or it is a subnetwork of a site accessed very often (i.e. Google) the connections of which may appear in sessions of less popular websites. As we cannot distinguish between these two situations we do not assign the IP addresses to any website as it may lead to incorrect assignation.

In figure 5.6 we sweep the subnetworks session threshold. In this case, the number of networks assigned to multiple websites falls rapidly as the parameter increases, something that makes this technique more appealing. Moreover, if we assign a network to a website we are not only relating the candidate IP addresses we grouped into the network but the whole network gaining additional information about the IP space used by the website.

5.5 Results and discussion

Taking into consideration the preliminary testing carried out in section 5.4 we use both the concentration of IP appearances decision parameter for candidate IP addresses and the percentage of sessions parameter for candidate networks in order to find IP addresses related to websites in our traces. For brevity's sake, we will call them **concentration method** and **subnetworks method** respectively.

In the first case, we have chosen 50% as the assignation threshold. In other words, candidate IP addresses are assigned to a website if at least 50% of their appearances happen in sessions of that website. A 50% threshold ensures that the ratio between the candidate IP addresses assigned to more than one site and the candidate IP addresses assigned to only one site is under 0.05 for the three traffic traces we have studied. Although the doubtful IP addresses are simply not assigned and do not suppose a problem, this value gives an estimation of the probability of incorrect assignations of IP addresses that belong to websites which we are not studying. These mistakes will be, in any case, infrequent as the candidate IP would not only need to be equally popular in the sessions of the other website but the sessions of both websites would need to overlap or the candidate IP would not meet the threshold in both cases.

For the subnetworks method we have chosen a subnetworks session threshold of 30%. This is a purely empirical decision as it yields good results. With this threshold, the number of networks assigned to more than one website is still considerable. These networks meet the threshold for several of the studied websites because connections to them appear throughout the trace. However, we find that they are almost always related to content distribution networks, web tracking companies and other services that do not really belong to any website. Because of this, we believe that it is safe to just ignore them.

Once the two methods are applied we obtain a list of IP addresses that have been assigned to each website. We now have to check if this assignation is correct.

5.5.1 Assignation results

The results of the assignation process for the three traffic traces appear in table 5.3. If we look at the rows of the table, in the first ones we present the individual results of both methods (specifying both the number of subnetworks and IP addresses assigned by the subnetworks method). Then we show the combined number of assigned IP addresses, how many of them were assigned to the same website by both methods (agreements) and how many were assigned differently (disagreements).

Table 5.3: Assignation results.

		Trace 1	Trace 2	Trace 3
Concentration method	IPs	532	446	654
Subnetworks method	Subnets	53	50	54
	IPs	667	652	643
Both methods	IPs	1059	954	1127
	Aggree	133	142	157
	Disagree	7	2	13

The behavior of the assignation system seems consistent enough for the different data sets. The number of related IP addresses is similar for the two methods which are complementary. As only 12-15% of the resulting IP addresses are shared by the two methods, applying both of them allows finding a much bigger number of related IP addresses. Addresses assigned differently are a rare occurrence (around 1% in the worst case) which is a promising indicator of the precision of the system.

In table 5.4 we show how many IP addresses were assigned to each website (again, for briefness we have omitted locally popular websites as we did in table 5.2). In some cases (Flikr, Tumblr) there were not enough sessions in all the traces to obtain results. We set a minimum of ten sessions per trace as we fear that the information obtained with very few sessions may not be representative.

5.5.2 Validation

Validating the obtained results is a challenging process. Identifying the assigned IP addresses manually is a time consuming task that may prove to be impossible in some cases. In order to do it, the tools that we can use are limited:

- Simply trying to access the web server (e.g. by typing the IP address in a web browser address bar) gives no information in most of the cases as

Table 5.4: Assigned IP addresses by website.

Website	**Trace 1**	**Trace 2**	**Trace 3**
Amazon	55	32	33
Google	95	85	113
Ebay	68	133	38
Facebook	31	106	77
Flikr	0	0	15
MSN	123	58	58
Megaupload	78	74	89
Tumblr	2	3	0
Twitter	23	8	5
Wikipedia	3	12	0
Wordpress	57	52	54
Wordreference	17	12	14
Yahoo	48	44	23

many servers expect to be asked for specific content and will provide a standard error page or simply reject the connection.

- Studying the application data of the packets of the connections to the IP address may sometimes help. In our case, the sniffer that captures the Internet traffic of our University limits the capture size to 100 bytes per packet. Because of that, we rarely see past the HTTP GET field and we have not found it very useful for identification purposes.

- In the end, we have primarily relied on information obtained via DNS and WHOIS protocols. We have used the tools provided by some websites [Robtex; revIP] that gather this information and complete it with data from BGP feeds and from other parties such as the Routing Assets Database (RADb [RADb]), analytics companies (e.g. Alexa) or domain name providers.

- When everything else fails, it is worth a try to use a web search engine to lookup the IP address but this is not usually very useful.

In any case, trying all of these strategies to check the thousands of labeled IP addresses is a daunting task. As a consequence, in order to validate our method we have applied a simple accuracy testing method inspired on the ones commonly used to validate classifications in which, as it is our case, it is difficult or time consuming to check if each particular element is correctly labeled [CG99]. Our approach is to sample around 10% of the labeled IP addresses, check if the assignation is correct for them and use the results to infer the accuracy of our classification.

We have validated the two proposed assignation methods separately as we describe in the following subsections.

5.5.2.1 Concentration method

The results for the individual IP addresses identified by the concentration of appearances parameter are shown in table 5.5. As we can see, of the sampled IP addresses, around 70% are correctly labeled in the three traces. This means that we have been able to establish a clear relationship between the IP address and the website to which it was assigned. A very small percentage of IP addresses are incorrectly labeled: of the five cases in which this happens, two are IP addresses related to malware, other two belong to web tracking services and the last one belongs to a different normal website.

Table 5.5: Validation of individual IP addresses.

Traffic	Sampled	Correct		Incorrect		Unknown	
Trace	IPs #	#	%	#	%	#	%
Trace 1	55	39	71	1	1.8	15	27.2
Trace 2	44	30	68.2	2	4.5	12	27.3
Trace 3	64	44	68.8	2	3.1	18	28.1

However, a sizable percentage of IP addresses is marked as unknown. As we have previously stated, the labeling process is far from easy even if carried out manually and for these IP addresses it was impossible to know whether they were related to the website or were incorrectly classified. Most of these IP addresses belong to content distribution networks (Akamai, Edgecast, etc.)

or remote computing services (Amazon web services). We expect the majority of these unknown IP addresses to be related to their assigned websites as our thresholds are very restrictive and we have been unable to link them to any other website. However, we have no way of knowing for sure.

5.5.2.2 Subnetworks method

For the subnetworks, instead of sampling 10% of the labeled addresses randomly we have chosen one random address of each subnetwork as we believe this is more representative of the total data. The results are shown in table 5.6. Again, the percentages of correct assignation are near 70%. In this case nevertheless, no IP address is incorrectly assigned. There is, again, a sizable percentage of unknown addresses. In the majority of cases these do not belong to CDNs as before but are related to on-line marketing companies whose servers probably host advertisements integrated in the assigned websites.

Table 5.6: Validation of IP subnetworks.

Traffic	**Sampled**	**Correct**		**Incorrect**		**Unknown**	
Trace	**IPs #**	**#**	**%**	**#**	**%**	**#**	**%**
Trace 1	53	39	73.6	0	0	14	26.4
Trace 2	50	35	70	0	0	15	30
Trace 3	54	40	74.1	0	0	14	25.9

5.6 Improving the system

In the previous sections of this chapter we have presented a method able to identify server IP addresses related to specific websites which we have afterwards tested. In this final section we introduce multiple improvements to the method and its validation process. Some of them are:

- A website selection process that allows us to choose the most popular websites in a traffic trace instead of using a predefined list. Through this new method, main IP addresses are inferred from passively captured DNS traffic instead of by automatic requests. Using DNS data to gather information about web connections has been done before, for example in [BMM+12].

- Changes in the way we find rough website sessions that take into account the different popularity of the websites under study.
- A more refined tuning and validation process in which DNS information is taken into account.

These improvements are detailed in the following subsections.

5.6.1 New website selection process

In section 5.3 we selected a list of popular websites based on our knowledge about the browsing habits of the users in our network. However, in order to extract the most possible information of a trace, it would be preferable to select the websites that are especially popular in it. Obviously, this has to be done after capturing the trace so we will not be able to gather the main IP addresses at the same time with automatic DNS requests. Rather than that, we will capture DNS traffic in our network and use it to obtain the main IP addresses. This forces us to capture new traces that we will use in the tests presented in this section.

We use two traffic traces captured, again, in the Internet link of the Public University of Navarre with the same methodology described in section 5.3. The traces were captured respectively during January and April 2013 and basic information about them is shown in table 5.7.

Table 5.7: New traffic traces.

Trace	Date of capture	# Flows	# Users
Trace 4	Jan 14 - 23, 2013	11M	1096
Trace 5	Apr 5 - 19, 2013	16M	967

In addition to the flow records, we captured all DNS traffic between our DNS server and the Internet. Our vantage point did not allow the capture of DNS traffic from the users to the DNS server which would have been more convenient (see figure 3.2). In the case of DNS records we capture full packets in pcap format as we will extract information from fields in the DNS payload. This DNS information is, again, not necessary for the identification process we present here. Our method does not require deep packet inspection, we only use it for tuning and validating our system and for choosing the list of relevant websites.

Using the captured flow records and information from DNS traffic, we select the most popular websites in the trace and obtain their main IP addresses following these steps:

- We separate the flows in the trace by user and, considering their start times, in intervals of 120s (which was previously chosen as a good higher threshold for session length).
- A server IP address will be popular depending on how many of these intervals it appears in. Defining popularity like this instead of just considering the number of flows for each IP address minimizes the effect of websites which open multiple connections to the same addresses during a session. In those cases, an IP address could have a big number of flows with a small number of sessions for the associated website.
- Following this definition, we have selected the 2,000 most popular server IP addresses in each trace. Of these popular addresses we consider the ones that, in the DNS capture, have an associated domain name composed of top and second-level domain names (or third for cases such as example.co.uk or www.example.com). That is, an IP address with example.com (or www.example.com) as an associated name will be selected while an IP address associated to another.example.com will not. Websites with different top-level domain name but the same same second-level one are grouped together (e.g. twitter.com, twitter.es and www.twitter.com).
- We manually select the interesting websites filtering advertising servers or web tracking services.

With this procedure we have obtained a list of 66 sites for trace 4. Figure 5.7 shows an estimation of the number of sessions for each site in trace 4 (i.e. the number of different 120s intervals in which the popular IP addresses related to

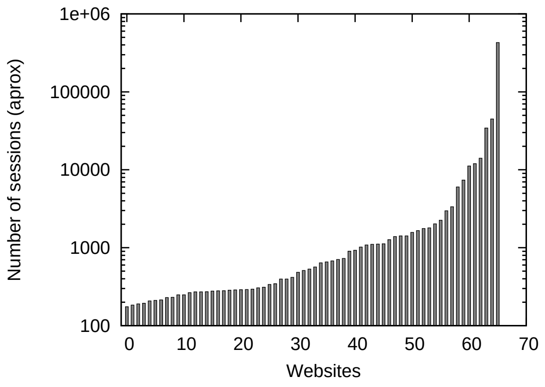


Figure 5.7: Estimation of the number of sessions per website.

the site appear in the trace). Note that the scale in the ordinate axis is logarithmic so the differences between the popularity of the sites is very big. Information for the ten most popular sites for trace 4 is also shown in table 5.8. Elpais.com, elmundo.es and diariodenavarra.es are popular Spanish newspapers (the latter specifically in Navarre); eltiempo.es is a local weather forecast service. For trace 5 we obtained a list of 73 websites. Both traces share the same very popular websites although there are some differences in the less popular ones.

Each of the selected websites has one (or more) associated main IP address that we will use in order to identify their sessions. As we can see in table 5.8, for trace 4, there are 51 main IP addresses related to Google which is, by far, the most popular website. In our case, Google encompasses not only the search engine but Youtube, Blogspot, Google Docs and other Google services as we could not distinguish between them using DNS information. We have found that Google provides these websites in our area from a pool of servers used for multiple services. Because of that, for example, users trying to access www.youtube.com and docs.google.com may be directed to the same server IP. The only Google service that seems to use independent servers is Gmail.

5.6.2 Popularity-aware sessions

The original definition of rough website sessions acknowledged the possibility of sessions from different websites occurring at the same time and that the re-

Table 5.8: Most popular sites in trace 4.

Website	**# sessions**	**# popular IPs**
Google	429,449	51
facebook	44,977	5
ElPais	34,311	2
Twitter	14,093	5
DiariodeNavarra	12000	2
Gmail	11,186	4
ElMundo	7,387	1
Linkedin	6,017	2
Wikipedia	3,352	2
eltiempo	2,976	1

sulting rough sessions would include connections from more than one website. This was not worrisome because we were considering multiple sessions of the same websites and these occurrences should be punctual. However, as we have seen in figure 5.7, even if we take into account the most popular websites in our network, the differences in number of sessions are very big. Because of this, if we consider a very popular site against one of the less accessed, it is possible that most of the sessions of the latter happen during sessions of the former. In this case it would be difficult to assign correctly the IP addresses of the less popular site as they also always appear in sessions of the more popular one. Moreover, if we consider the less popular websites that have not made it into our list (for the sake of brevity we will call them *unknown websites* from now on), we have no way of knowing if there is a concurrent session to one of them at any given moment so their IP addresses could be assigned incorrectly.

Taking this into consideration we modify the way we defined the sessions in section 5.2 introducing measures that help in the identification of these IP addresses. Sessions in a trace are now created as follows:

- We use the popular IP addresses described in 5.6.1 as main IP addresses. In other words, when we find a connection from a user to one of the main IP addresses, we consider that the user is visiting the corresponding website.
- The following connections initiated by that user during a period of time

(session length) will be considered part of the same session. Their server IP addresses will be then candidate IP addresses that, in the end, may or may not be associated with the website.

- If the user accesses a main IP address during a session of a different website a new session will start. The method takes now into account the popularity of both sites. If the new session is associated to a more popular site, the following connections will still be assigned to the older session until it finishes. On the other hand, if the new session is associated to a less popular site, the following connections will be assigned to it, except for those whose server IP are already candidate IP addresses in the older session.

- If there are no active sessions and the user opens a connection to an unknown (not main) IP address, a session to an unknown site is created. If a session to a known site starts while this session is active, all new connections will be assigned to the new session except those whose server IP addresses are already assigned to the unknown session. The only purpose of this unknown session is to protect the addresses of the sites we are not considering from being incorrectly assigned to known sites.

The modified process by which rough website sessions are assembled from the connections in the traffic traces is explained in figure 5.8.

5.6.3 Improved tuning

If we want to use the concentration and subnetworks methods over the candidate IP addresses of our new popularity-aware sessions there are three parameters that must be tuned. In previous sections we empirically chose a value of 120 seconds for session length and values of 50% and 30% for the thresholds of the concentration and subnetworks methods. Here we present a more refined method for tuning these parameters.

For the tuning process we will use data from trace 4. Because of this, all figures in this subsection present data from that trace. In section 5.6.4 we will check if the selected values also yield good results with trace 5.

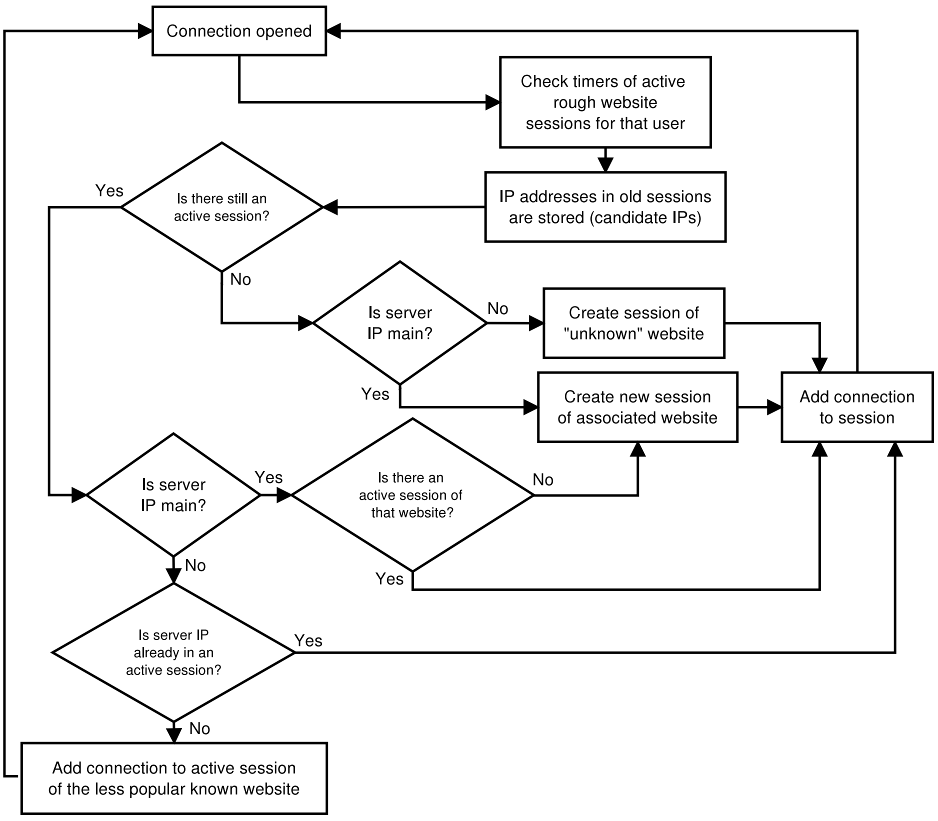
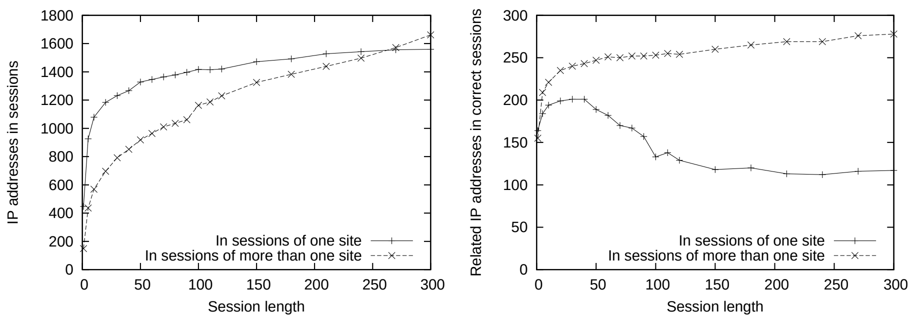

Connection opened
Check timers of active rough website sessions for that user
IP addresses in old sessions are stored (candidate IPs)
Is there still an active session?
Yes
No
Is server IP main?
No
Yes
Create session of "unknown" website
Create new session of associated website
Add connection to session
Is server IP main?
Yes
Is there an active session of that website?
No
Yes
No
Is server IP already in an active session?
Yes
No
Add connection to active session of the less popular known website

Figure 5.8: Grouping connections in popularity-aware rough website sessions.

5.6.3.1 Choosing web session length

Due to the popularity-aware nature of the new sessions, overlapping of sessions is not so critical for our considered websites: the sessions of the least popular ones are protected from overlapping while the most popular will appear elsewhere in the trace anyway. Nevertheless, the new definition of sessions can do little to protect the IP addresses of unknown websites and if we use very big sessions we are bound to make some mistakes with them. Moreover, longer session lengths imply higher processing requirements as more information must be kept in memory during the execution of the system. Therefore, we still need to choose an value for session length that balances these considerations.

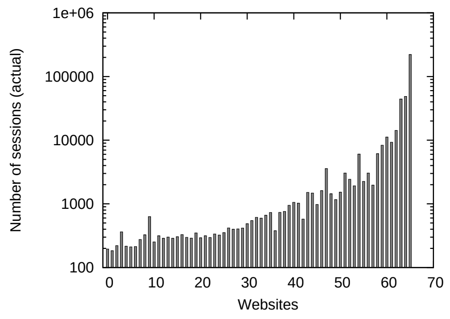


(a) Candidate IP addresses in sessions. (b) Related IP addresses in sessions.

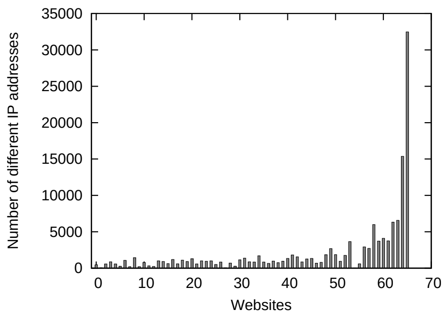
Figure 5.9: Candidate and related IP addresses in sessions.

Taking this into account we first consider figure 5.9(a). In this figure we represent, for different session lengths, the number of candidate IP addresses that appear in all the sessions of all the considered websites combined. We are not applying the concentration or subnetworks method yet. The continuous line represents IP addresses that appear only in sessions of one website; the dashed one, IP addresses that appear in sessions of more than one website. It seems interesting to get a big number of IP addresses that appear only in sessions of one website as they are good candidates for labelling. However, as we can see, that curve ceases to grow for small values of session length. On the other hand, the growth of the number of IP addresses that appear in sessions of various websites suggest, as predicted, that increasing the length of the sessions results in more overlapping between websites. It is important to note, however, that even if an IP address appears in sessions of more than one website, it may appear in a lot of sessions of one of them and in only a few of the others. The concentration of IP appearances parameter will then be able to assign it to the correct site.

Figure 5.9(a) is interesting in that it shows that a high value for session length is unnecessary. However, the fact that an IP address appears only in sessions of one of the considered websites does not imply that this address truly belongs to the website. In order to shed some light on this, we consider a subset of the candidate IP addresses that we will call *DNS-related IP addresses*. An IP address will be DNS-related to a website if it has an associated domain name that contains the name of the website (e.g. an IP that appeared in a DNS response for



(a) Sessions per website.

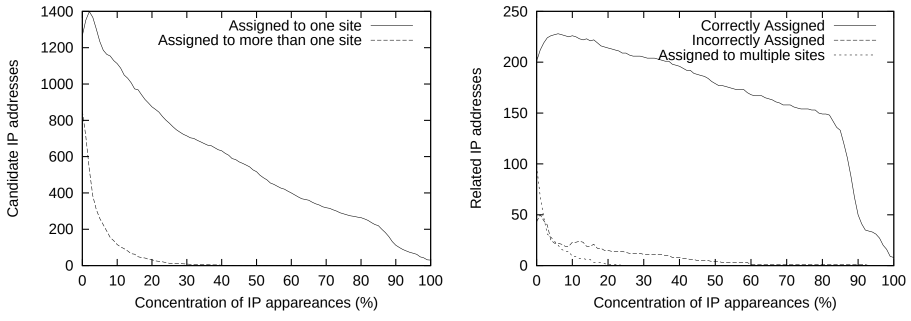


(b) Different IPs in website sessions.

Figure 5.10: Sessions for 40s of session length.

profile.ak.facebook.com.edgesuite.net will be DNS-related to Facebook). Not all IP addresses that belong to a website will be DNS-related (companies have different naming policies for their servers and some websites may host some of their content in third-party servers). In figure 5.9(b), the dashed line represents the number of DNS-related IP addresses that appear in sessions of the correct website. Of those, the ones that appear only in sessions of the correct website are represented by the continuous line. Again, increasing session length past 50 seconds does little in order to increase the number of DNS-related IP addresses that appear in sessions of their websites. Also, the number of related IP addresses that appear only in sessions of the correct website decreases after reaching a maximum in 40 seconds. In view of these results, we choose 40 seconds as the value for session length.

In order to describe the sessions created, figures 5.10(a) and 5.10(b) show, respectively, the number of sessions and candidate IP addresses for the considered websites with sessions of 40 seconds. With the new session length there are some differences between figures 5.7 and 5.10(a) but the difference between the most popular websites and the rest is still considerable. The differences in the number of candidate IP addresses shown in figure 5.10(b) are very high. This is to be expected as Google sessions, for example, overlap with a lot of sessions of other websites (some of which are unknown). Most of these candidate IP addresses will be discarded by the thresholds of the concentration and subnetworks methods.



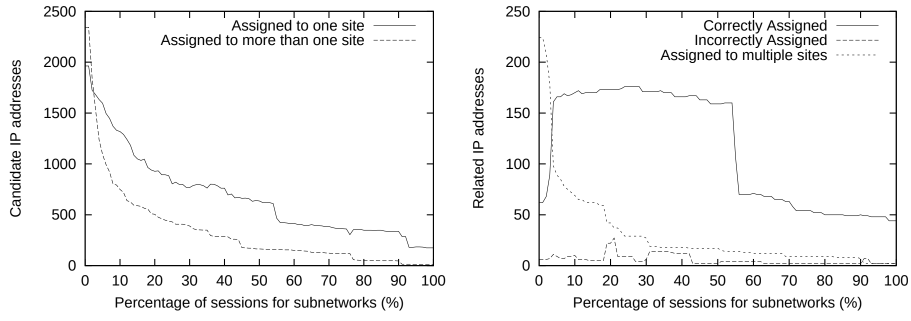
(a) Assigned candidate IP addresses. (b) Assigned DNS-related IP addresses.

Figure 5.11: Assigned candidate and DNS-related IP addresses for the concentration method.

5.6.3.2 Choosing a concentration threshold

With 40 seconds as session length and filtering the IP addresses that do not appear in, at least, two sessions (or 1% of the total sessions of the website) of two different users, we obtain a total of 2,119 candidate IP addresses for the 66 websites combined for trace 4. Of these, 1,267 appear only in sessions of a website and 852 appear in sessions of more than one website. The fact that some candidate IP addresses appear in sessions of more than one website suggests that some of the candidate IP addresses that appear only in one site may actually appear also in sessions of unknown sites. Because of this, if we want the concentration threshold to be able to identify the candidate IP addresses that truly belong to a site, the chosen value should, in the first place, ensure that no candidate IP address is assigned to more than one of the considered websites.

As we see in figure 5.11(a) (which is analogous to figure 5.4(c) but this time with the data from trace 4), the concentration method still works well with the popularity-aware sessions. In this figure we represent, for different values of the concentration of IP appearances threshold, the assigned candidate IP addresses for all the websites. The continuous line represents the candidate IP addresses assigned to only one site and the dashed one, the candidate IP addresses assigned to multiple sites. Increasing the threshold produces a sharp decrease of the latter that nears zero for values higher than 30%.



(a) Assigned candidate IP addresses. (b) Assigned DNS-related IP addresses.

Figure 5.12: Assigned candidate and DNS-related IP addresses for the subnetworks method.

However, as it happened with session length we do not know if the candidate IP addresses are being correctly or incorrectly assigned. To clarify this, lets consider DNS-related IP addresses again. In figure 5.11(b), the continuous line represents the number of DNS-related IP addresses correctly assigned to the websites; the long-dashed one, the number of incorrectly assigned DNS-related IP addresses; and the short-dashed one, the DNS-related addresses assigned to multiple sites. As it can be seen in the figure, addresses assigned to multiple sites do not suppose a problem as they disappear quickly. Moreover, as we increment the threshold, the number of addresses assigned wrongly also decreases. For values of the concentration of IP appearances parameter over 40% the error ratio is less than 5% and therefore we will choose this value for the concentration threshold.

5.6.3.3 Choosing a subnetworks threshold

Analyzing the 40 second sessions, we obtain 565 class-C subnetworks of which 246 appear in sessions of only one website. Figure 5.12 is analogous to 5.11 but with data from the subnetworks method assignation. As we can see in figure 5.12(a), the number of IP addresses assigned to various websites is higher in this case and increasing the subnetworks threshold does not lower it much. As we predicted, most of these IP addresses belong either to very popular websites (especially Google) or to CDNs. However, figure 5.12(b) shows that the number of

DNS-related IP addresses assigned wrongly is low (lower than 5% for values of the threshold above 50%). We will set 50% as the value of the subnetwork sessions threshold. Another reason to choose a 50% threshold is the sharp decrease in the number of assigned IP addresses that happens between 50% and 60%. In fact, at 50% 95 subnetworks are assigned to the sites but 29 of them appear in less than 60% of the sessions of their websites and will disappear if we increase the threshold. The fact that there is a sharp decrease both in the total assigned IP addresses and the DNS-related IP addresses suggest that these networks were correctly assigned.

5.6.4 Automatic validation

Once we have selected the values of the different thresholds we test our improved system with traces 4 and 5. We want to check if once the tuning is performed for our network with one of the traces, the results will still be good for the other. In table 5.9 we show a breakdown of the assignation results for both traces. It is a similar table as the one shown in section 5.5.1: the first rows show the individual results of both methods and then we show the combined number of assigned IP addresses, how many of them were assigned to the same website by both methods (agreements) and how many were assigned differently (disagreements). Sadly, we cannot compare the performance of the original method and the improved one because the data sets used are different but this was necessary in order to capture the DNS traffic needed in this section.

Table 5.9: Assignation results

		Trace 4	Trace 5
Concentration method	IPs	632	574
Subnetworks method	IPs	636	591
Both methods	IPs	1268	1140
	Aggree	182	103
	Disagree	6	4

Checking table 5.9, the behavior of the labeling system seems, again, consistent for the two traces. The methods still appear complementary with only

9-15% of the assigned IP addresses shared between them. Addresses labeled differently are 0.5% in the worst case.

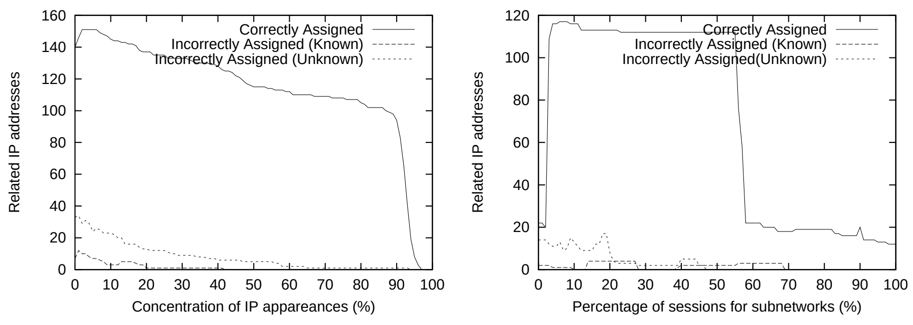
In section 5.5.2 we stated that validating the results was a complex process and we chose to extract a sample of the assigned IP addresses and tackle their validation manually. We realized that the most useful font of information for this manual validation process was DNS data. Because of that, here, we have designed an automatic validation process using DNS-related IP addresses.

Table 5.10: Assignation results for DNS-related IP addresses

		Trace 4	Trace 5
Concentration method	A. IPs	197	183
	of total	31%	32%
	P.IP	95.0%	93.4%
	P.Flow	94.0%	90.7%
	P.Byte	99.5%	98.7%
Subnetworks method	A. IPs	159	173
	of total	25%	29%
	P.IP	98.7%	92.4%
	P.Flow	99.0%	86.2%
	P.Byte	99.9%	97.9%

In table 5.10 we show the results of the assignation for the DNS-related IP addresses. For both methods we present: the number of DNS-related IP addresses assigned (A. IPs), the ratio of the DNS-related IP addresses against the total assigned IP addresses, and the precision in IP addresses, flows and bytes (P.IP, P.Flow and P.Byte). For example, a precision ratio of 95% for IP addresses means that out of 100 DNS-related IP addresses, 95 were assigned to the correct website. Taking into account the number of flows and the bytes in those flows for each IP we can also obtain the flow and byte precisions.

The assignation results are generally very good. For trace 4 they are better as the tuning of the system was made in order to minimize the wrong assignations in that trace. However, all values of precision remain near 90% for trace 5. It must be noted that, although trace 5 was captured in the same network and under the same conditions, it was captured months after trace 4, it is longer and



(a) Concentration method. (b) Subnetworks method.

Figure 5.13: Assigned DNS-related IP addresses with unknown sites.

the selected websites for it are not the same. Even with these differences, the performance of the system for the DNS-related IP addresses remains acceptable.

Given that a representative amount (25%-32%) of the assigned IP addresses are DNS-related IP addresses it is tempting to extended these results for all the assigned IP addresses. However, there are two main concerns: (i) the DNS-related IP addresses are not selected randomly from the total assigned IP addresses so their behavior may not be extensible for all of them. (ii) the DNS-related IP addresses only consider the preselected websites so we have no way of proving that we are not assigning IP addresses from unknown websites wrongly. Both concerns are related as the possible mistakes in the assignation of non DNS-related IP addresses primarily affect unknown sites.

In order to address this problem, we have taken half of the 66 considered sites out of the system and used them as a control group. We have taken both very popular and less popular sites (in fact, we have ordered them by popularity and omitted the odd ones leaving the even ones). For all purposes we treat them as unknown sites except that we now can check if their DNS-related IP addresses are assigned to other sites. Using the control group we have obtained the results shown in figure 5.13 which includes two subfigures similar to 5.11(b) and 5.12(b). Now we distinguish two types of incorrectly assigned IP addresses: those that are DNS-related to a site under study (known) and those that are DNS-related to a site from the control group (unknown). As we see, the system still works well. Some IP addresses from control group sites are assigned wrongly but the IP precision remains over 90% for both methods with the selected thresholds. All this suggests that the non DNS-related assigned IP addresses are correct in most cases.

5.7 Conclusions

In this chapter we have presented a method able to identify server IP addresses closely related to a predefined list of websites in a traffic trace. This is a far from trivial problem as users often access more than one website at the same time.

Our system labels individual IP addresses based on the number of times connections to them appear in sessions of a particular website against the total number of apparitions in the trace. We have found that this parameter is more representative of the relationship of an IP address and a website than the percentage of sessions of the website in which the IP address appears (which *a priori* seemed more intuitive). It also labels IP subnetworks if various IP addresses that belong to them appear repeatedly in sessions of a website.

We have tested our system with three traffic traces of, at least, a duration of ten days. We have identified an average of more than 30 IP addresses per website in each of the traces. We have validated those results manually but the difficulty in doing so only allows us to give a lower bound for the accuracy of our system at around 70% although it probably is higher.

Afterwards, we have introduced some modifications to the system and a new validation procedure. We have tested the new system with two traffic traces of similar characteristics. With this new system, we have obtained an average of more than 20 IP addresses per website in each of the traces. We have validated these results by considering the assignation of IP addresses that are related to the websites by DNS information obtaining precision values over 90%.

A system such as the one we present here can be used for multiple purposes. Identifying the IP addresses associated with particular websites allows gathering information about the browsing behavior of the users and about the websites themselves. Our initial motivation was, however, to obtain labeled traffic traces that could be used to tune and test a web traffic classification system. Nevertheless, even though the system is able to identify a sizable number of IP addresses for selected popular websites, it has limited success when trying to label a complete traffic trace where the variability of accessed websites is very high.

CHAPTER 6

Characterizing webpage downloads from a flow-level perspective

In this chapter we offer a thorough characterization of webpage downloads using connection level metrics. Although there is an extensive literature on the characteristics of web traffic, few works have focused on connection level measurements even if this kind of data is easily available for network administrators. We have captured a data set of more than 20,000 webpage downloads that we study in order to provide different connection level based metrics. We describe how these metrics vary between different webpages of different popularity and complexity. In the end, we attempt to provide a general modeling of a normal webpage download. This is a first step in developing techniques able to identify webpage downloads in real traffic.

6.1 Introduction

We have previously described different levels of web traffic aggregation, from a single request-response pair to a browsing session. Up until now we have worked at a website session level considering the relationship between the connections established while a user visits a specific website. We have found that the traffic of different website sessions usually has very different characteristics. In particular, website sessions are very variable in length depending on the type of website and on the behavior of the user that visits it. This makes it difficult to use time-based metrics to group the connections of the same website session. Moreover, we have studied the servers accessed during website sessions of different websites and we have found that even though some servers are closely related to individual websites, most of them are not. In fact, because of CDNs and third-party content (in the form of advertisements, embedded multimedia, widgets, etc.) it is nowadays relatively common that some of the servers accessed during sessions of different websites are the same. As a consequence, checking server IP addresses is not either a reliable method for grouping together the connections of a website session.

We have now decided to focus on a different level of aggregation: webpage downloads. Considering individual webpage downloads instead of whole website sessions simplifies the problem under study. Although webpages can be very different from one another, they have in common that the web protocols are designed to expedite the download of the content as much as possible in order to allow the webpage to be rendered rapidly. Because of this, the connections involved in the download of a webpage are usually opened very close in time and the time intervals that should be considered in this case are less variable. Moreover, user behavior has much less impact as even though concurrent sessions of different websites are somewhat common, downloading two different webpages at the exact same time is a rarer occurrence.

A further advantage of studying individual webpage downloads is that we can obtain samples of their traffic by accessing them automatically. Precisely, in this chapter we describe the traffic generated by automatic webpage downloads focusing on different flow-level characteristics. Although there is extensive work in web traffic characterization (see section 2.1.6) the approach of other authors has usually been very different to ours either because they take a server perspective or because their studies work with application level data. In our case, we consider our research from the client's point of view in the sense that we study the connections between the client and multiple servers through Internet and we do not have any information about the relationships between those servers or the content they host. As in previous chapters we work with network traffic in the form of flow records captured close to the client (in this case, in its own network card). All in all what we offer in this chapter is a thorough characterization of the set of connections initiated by the client during the download of a webpage.

Having a good description of the connections involved in the download of a webpage can be very useful for multiple purposes. On one hand current users access multiple webpages in short periods of time, often of different websites thanks to tab-based browsers, so it is far from trivial to guess which webpages (or even how many different ones) a user visits. This characterization may allow the development of techniques that help identify each individual webpage download, offering insight into the user's behavior (this is, in fact, the focus of chapter 7). On the other hand, nowadays multiple applications mask their traffic in order to pass it off as HTTP and avoid certain restrictions that network administrators may want to enforce. Characterizing normal webpage downloads could help in designing anomaly-based detection systems able to identify that kind of applications.

6.2 Data collection

In this section we describe the data set we are going to use for our analysis in this chapter. The traffic captures that integrate it were made during the months from August to October 2013.

6.2.1 Website selection and measurement setup

In order to collect a representative sample of webpage loads, we have selected 1,000 sites from the top 100,000 websites of the Alexa global ranking [Alexa].

We have chosen the 100 most popular websites, 300 websites selected randomly from the 100-1,000 most popular ones, another 300 from the 1,000-10,000 range and the last 300 from the 10,000-100,000 range. With this, we ensure that the most popular (and hence interesting) sites, like Google, Facebook, or Amazon are well represented in the sample while also collecting data from a wide variety of less popular sites from all around the world.

We have gathered our measurements from a computer in the Public University of Navarre's network. This PC has a public IP address and runs Ubuntu Linux (version 13.04). We felt unnecessary to set more than one vantage point as the authors in [BMS11], for example, found few differences when collecting the traffic of the same websites from different locations around the world. In this PC we run an automated script which follows these steps for each webpage under study:

- Launch Tcpdump [pcap], a network sniffer.
- Open a web browser to the selected website. We have collected measurements for both Mozilla Firefox (version 22.0) and Google Chrome (version 29.0.1547) which are the most popular browsers for Linux systems and together are responsible for a big percentage of the global web traffic [SCounter; W3Counter]. Plug-ins such as Adobe Flash player were installed in order to ensure that websites render properly but, aside from that, we use clean installations of both browsers with no ad or pop-up blockers.
- Wait for two minutes. Although webpages usually load in a few seconds [IP11], we capture traffic while the browser is idle for longer in order to study data transfers that happen even after the webpage has been fully rendered (when, for example, refreshing dynamic content).
- Close the web browser and close Tcpdump (we leave a small guard interval before closing Tcpdump in order to capture the ending of the pending connections).

We have repeated this procedure gathering twenty captures of each webpage download in pcap format (ten for each browser). We also captured one additional 10 minute long load for each page and browser in order to study flow end times as we will explain in section 6.4.

The study we present in this chapter is focused on website landing pages (*i.e.* the page served when the user inputs the domain name of the website). The characteristics of landing pages can be different to those of internal pages (*i.e* accesed via links from the landing page) of the same websites. However, we are studying a wide variety of landing pages from 1,000 websites of different popularity. We believe that this is a sample with enough diversity to be representative of the characteristics of most webpages.

6.2.2 Preprocessing

As we have done in previous chapters, in order to obtain flow records from these packet traces, we use Argus. As we are only interested in web traffic we select flows originated in our PC and with destination ports 80 and 443 (HTTP and HTTPS connections). For each flow, we collect the information we described in section 3.2 and, additionally, we store:

- The number of upstream and downstream application-level bytes: we will use this information to calculate the size of the downloaded objects in a webpage.
- The first 1,000 bytes of the upstream application data: from which we will extract some HTTP header fields. We will use this application-level data to obtain additional information about the nature of the connections involved in a webpage download (see subsection 6.2.3). However, this is not central to our study which, as previously stated, focuses on flow-level statistics.

Moreover, we also extract DNS information from the pcap traces: we consider all the different server IP addresses accessed during the load of the webpage and, by studying the DNS query responses captured, we obtain a list of related domain names and authoritative nameservers for each IP address. This DNS information will allow us to better understand the part each connection plays in the load of the webpage as explained in the following subsection. Nevertheless, again, the information obtained through DNS is not a central part of the study we present here.

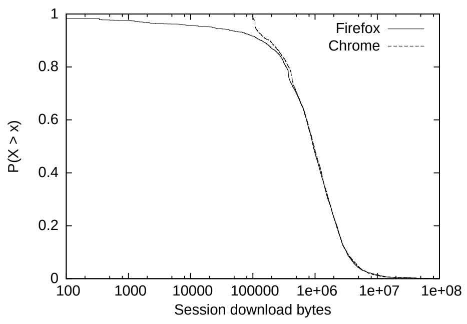
6.2.3 Connection origins

We parse the HTTP data captured for each connection so we are able to extract the name of the accessed server, the URI of the first element requested and the HTTP method. With this information we identify the *root connection* of the webpage load. The root connection uses the GET method and requests the server root ("/") of a host with the same name as the website name. We then label connections according to their *origin*: those connections whose server name is related to (*i.e.* contains) the site name are classified as *shared name connections* and, from the rest of connections, we distinguish between *same origin* and *other origin* connections by checking if the domain name of the related server comes from the same authoritative nameserver as the root connection's or not.

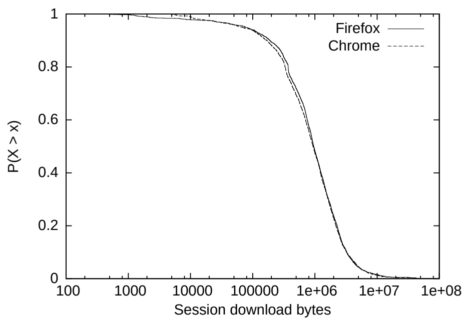
Authoritative name servers are a name servers that give answers in response to questions asked about names in a zone. Popular websites with a relatively big web infrastructure usually administrate their own DNS zone and the associated authoritative nameservers. Those websites are more liable to host the content of their webpages in multiple different servers which will be identifiable because they have the same associated authoritative nameservers. Smaller websites may not maintain their own authoritative nameservers but if they have a smaller and localized server infrastructure, their servers will probably belong to the same DNS zone too.

In any case, if the root connection carries HTTPS traffic, it is impossible to identify it by checking user data. In this case, the root connection will be the first flow opened to an IP address whose related domain name is the name of the website. If we look back to chapter 5, we could say that the root connection is the first connection opened to a main IP address of the website.

If we are unable to identify a root connection (i.e. there is no connection opened to an IP address related to the domain name of the website) or if we are able to identify it but it carries no application data, we discard the capture as a failed webpage load. For the sake of simplicity in processing the data, we only



(a) With browser-related traffic



(b) Without browser-related traffic

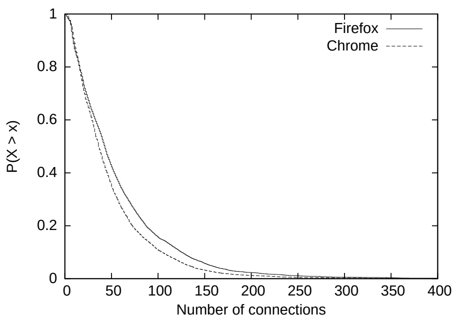
Figure 6.1: CCDF of total downloaded bytes in a webpage load.

consider websites that loaded successfully at every captured attempt. With this we reduce the list of considered websites from 1,000 to 912 resulting in a total of 18,240 flow records.

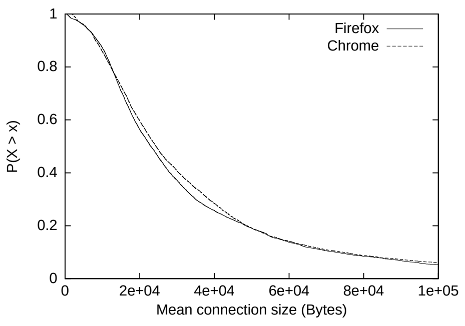
6.2.4 Browser-related traffic

When comparing the total downloaded bytes for the same websites we observed discrepancies between the two web browsers. In particular, around 100KB of additional content were downloaded by Google Chrome in each capture. This is particularly apparent for small webpage downloads in figure 6.1(a). We discovered that this browser opens some HTTPS connections to Google servers for different purposes. Some of these connections are related to the webpage and could be considered part of its download (for example, services like Google translate or Adsense) but others are automatic connections that are part of the browser operation and happen at fixed intervals from the start of the process. We decided to not consider these second type of connections because, as we said previously, they are related to the browser behavior and not to the particular websites. We also eliminate from our data sets some similar connections opened by Firefox to Mozilla servers.

After filtering browser-related traffic, figure 6.1(b) shows the empirical CCDF for the total bytes of downloaded application data in every one of the captures of 120 seconds. This represents the total size of the different elements



(a)



(b)

Figure 6.2: CCDF of the (a) number and (b) average size of connections.

of each webpage. The distributions are very similar for both browsers as the effect of the browser in the elements downloaded from the webpage should be minimum.

6.3 General characterization

We start by providing some general connection-based metrics of webpage download. As it happened with figure 6.1 we have aggregated data from all our captures in order to plot the different graphs in this section. Because of this, each of the samples we use to calculate the empirical CCDFs corresponds to a different traffic capture (multiple downloads of the same webpages are present but there are the same number of them for every webpage so they are evenly represented).

6.3.1 Number of connections and connection sizes

In the first place we focus on two simple connection-level statistics: the number of connections in each webpage download and the average size of those connections. Figure 6.2(a) shows the empirical CCDF of the number of connections initiated by the client during each webpage download. The distributions are, again quite similar: for both browsers the median is close to 40 connections while the 10 percentile sits around 10 connections (which means that for 90%

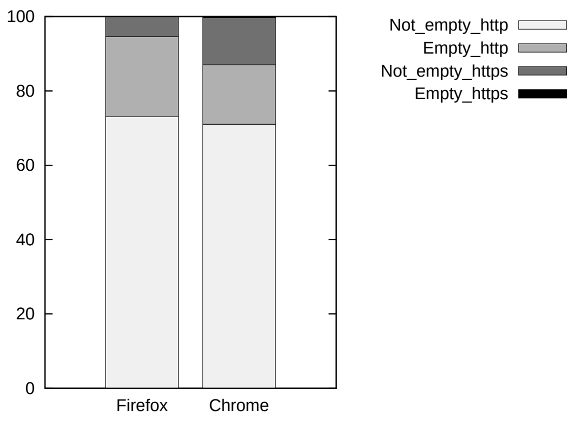


Figure 6.3: Percentages of empty and HTTPS connections.

of the studied webpages, at least 10 connections were used in the download). However, the tails of both distributions are long and, as we consider downloads with more connections, Firefox starts opening more of them (the 90 percentile is at 104 connections for Google Chrome and 125 for Firefox).

As the total bytes downloaded by both browsers are very similar (fig. 6.1) this means that on average, Chrome connections are slightly bigger and that more elements of the webpage are grouped in each of them. In any case, in figure 6.2(b) we can see that the difference in average connection size is small.

However, if we look at the individual connections (and, especially, the smallest ones) we do find some differences between the browsers' behavior. In figure 6.3 we show the percentages of HTTPS and empty connections. These percentages have been calculated by aggregating all flows from all traffic captures of each web browser. As we can see, the percentage of normal HTTP connections is similar for both browsers however, the rest of connections are divided differently. Google Chrome has a higher percentage of HTTPS connections. By studying the servers this connections were established with, we realized that a lot of them were small connections related to services Google provides through Chrome to help the navigation process like, for example, Google Translate. However, the number of HTTPS connections initiated by Chrome to Twitter or Facebook servers is also higher.

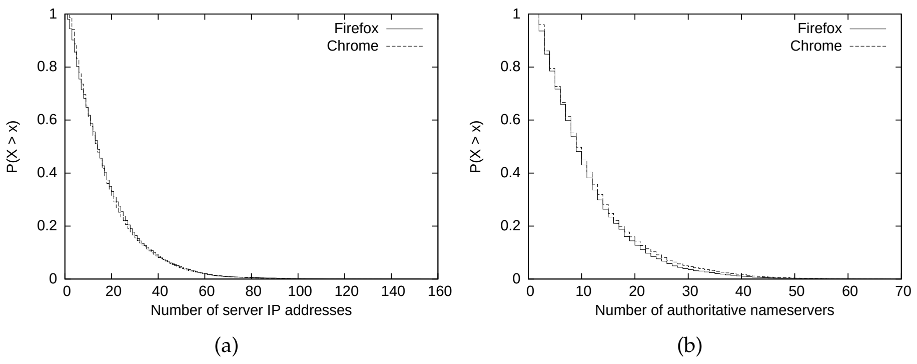


Figure 6.4: CCDF of the (a) number of servers, (b) authoritative nameservers.

The other kind of connections we consider are empty connections. *Empty connections* correspond to flows that, having successfully completed their initial TCP three-way handshake, carry no application data. Most of this flows (around 95%) are also properly terminated although we also consider flows ended with a reset message or still open at the end of the capture. HTTPS empty connections are a very rare occurrence for both browsers but HTTP empty connections are a quite common event specially for Firefox in whose traffic they represent around 20% of all connections.

In most cases empty connections are a result of strategies employed by browsers with the objective of reducing webpage load times and, because of that, their number depends on the particular implementation of the browser (see section 2.1.1). On one hand, browsers may open multiple connections to a particular server before knowing how much content is going to be downloaded from it. This behavior has its root in that it is faster to have connections prepared, in case they can be used to download multiple elements simultaneously, than opening them as they are needed and having to wait for the TCP handshakes. On the other hand, browsers usually wait for the answer of a SYN packet for a very short time before sending a new one. This short wait interval is designed to reduce the delay in the download when the first SYN packet is lost. However, in many cases, the first SYN packet was not lost, the SYN-ACK from the server does arrive a little later and, as a consequence, two connections are opened. In both cases, some of these connections may be left unused.

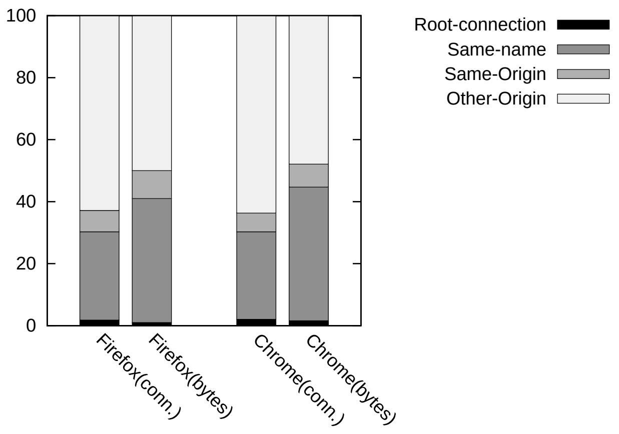


Figure 6.5: Percentage of connections by origin.

6.3.2 Servers and origins

We now focus on the web servers accessed during a webpage download. The differences between both browsers should be minimal in this respect (beyond some browser-specific services mentioned before) as browser implementation cannot affect where the elements of a webpage are stored. We start by calculating the distributions of the number of different IP addresses accessed during each download which is shown in figure 6.4(a). The distributions, both with medians at 14 IP addresses but long tails, corroborate that modern websites download elements from multiple different servers.

As we explained in section 6.2.2 we use authoritative nameservers in order to distinguish between different origins for web content. Figure 6.4(b) shows the distributions of the number of different authoritative nameservers seen on the DNS responses of the server names associated with each webpage download. Two authoritative nameservers are considered different if they have a different second-level domain name (third-level for some second-level domains like "co", *e.g.* "google.co.uk"). As we can see most webpages download content from servers of different origins (medians are 9 different origins).

Finally, in figure 6.5 we show the percentages of connections and bytes according to their origins. As in figure 6.3 we have considered every individual connection from the different traffic captures. The figure shows that, on average, more than 60% of the connections made during a webpage download are directed to third-party (other origin) servers. The most popular of them include, among others: analytics, social networking, image hosting, content distribution networks or video streaming. However, when considering downloaded bytes we can see that first-party content (root-connection, shared-name and same-origin servers) represents more than 50% of the total download suggesting that connections to first-party servers are bigger on average.

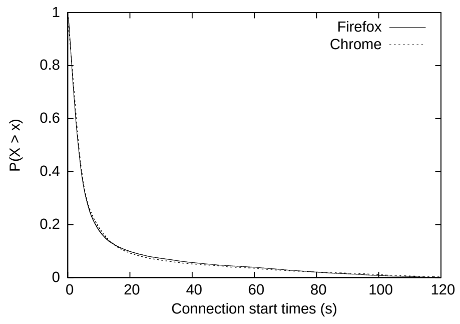
In table 6.1 we offer a summary of the different per-download metrics presented in this section. These are: number of connections, mean connection size, number of accessed IP addresses and number of authoritative nameservers. We also provide the median and 10th and 90th percentiles for each of them in order to give some quantitative measurements of their usual values.

Table 6.1: General characterization metrics

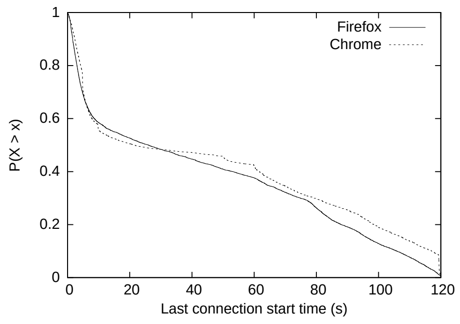
	Firefox			**Chrome**		
Metric	**P10**	**Median**	**P90**	**P10**	**Median**	**P90**
N. conn.	6	43	125	10	36	104
C. size (KB)	8.7	23.0	72.6	8.1	24.4	74.4
N. IPs	4	14	39	4	14	37
N. A.NS.	3	9	22	3	9	24

6.4 Time metrics

In this section we are going to discuss metrics related to the start and end timestamps of the flows in the download of a webpage. Again, we consider the aggregated data of every download of every webpage as we want to characterize the profile of an average load rather than explore the differences between webpages. However, in this case, we are going to represent some parameters that are related to the individual connections rather than to the complete captures (as it was the case, for example, in figure 6.4(a) with the number of accessed servers per download). In all the figures in this section, the 0 seconds mark corresponds to the start timestamp of the first connection in each webpage download. The



(a)



(b)

Figure 6.6: CCDF of (a) connection start times, (b) last connection start time.

rest of connection timestamps in each download are calculated relative to that first one.

The first parameter that we are going to study is connection start times which appears in figure 6.6(a). In order to calculate these CCDFs we have considered the start timestamps of all the connections in every capture. As their timestamps are calculated relative to the first connection in each capture, we can compare them. We see that the majority of connections are opened during the first seconds of the download of a webpage (around 80% of connections in the first 10 seconds). This makes sense as this is an upper threshold for the time a webpage takes to load [IP11]. However, the tail of the distribution is long and a considerable amount of connections are opened later which suggests that even after the webpage is fully rendered some information is still exchanged.

We explore these connections that are opened late in the download in figure 6.6(b) where we represent the distributions of the start times of the last connection in each capture. We can confirm that even though most connections are opened in the first seconds, for a sizable number of captures the last connections are opened much later. To shed some light about these late connections we used origins as defined in section 6.2. We expected that the late connections could be specially related to third-party advertisement or analytics services. However, there is very little difference in the distributions of connection start times according to the different origins (aside from root connections happening always early in each capture). This suggests that the connections opened late do not only cor-

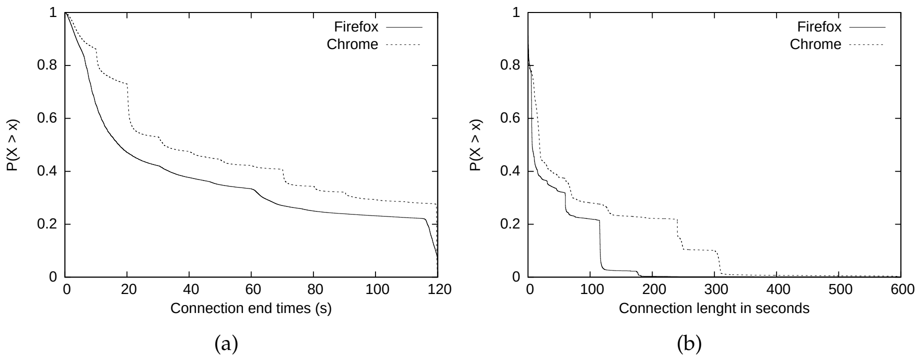
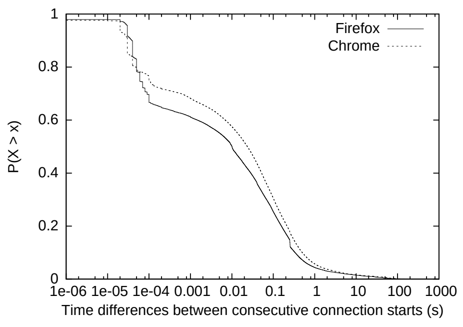


Figure 6.7: CCDFs of (a) connection end times, (b) connection lengths.

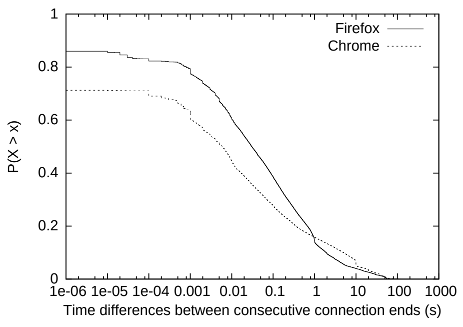
respond to third-party content but also to dynamic content hosted in first-party servers.

Lets now look at connection endings. Intuitively, we may think that connections are closed as soon as the elements they were opened to download are received by the client. However, this is not the case for a sizable amount of them. Because servers and browsers implement persistent connections (all the HTTP connections observed were HTTP/1.1) some connections are kept open for longer in case they are needed for an additional download. As shown in figure 6.7(a), both browsers keep more than 20% of the connections open for all the duration of the capture and close them simultaneously as the browsers are closed. Nevertheless, studying the figure we saw that Firefox started ending some connections a few seconds before we closed the browser.

As we explained in section 2.1.1, Firefox implements a persistence timeout for HTTP connections that, by default, has a value of 115 seconds but can be tuned by the user via the configuration utility in about:config. The value of this timeout for Google Chrome is not documented. In order to properly study the effects of these timeouts we captured one additional traffic trace of ten minutes for each webpage and browser. In figure 6.7(b) we show the distributions of connection length in seconds for these longer traces. For Firefox, the 115 seconds timeout is evident because most persistent connections have that length. The cause of the other step in the distribution (around 60 seconds) is more difficult to pinpoint but should be related to a server timeout because it also appears in the



(a)



(b)

Figure 6.8: CCDF of (a) time differences between connection starts and (b) time differences between connection ends.

Chrome distribution. For Google Chrome the default persistence timeout could be around 250-300 seconds. In any case, for both browsers around 60% of the connections are shorter than 20 seconds, either because they are not persistent or because of timeouts in the servers (Apache 2.0 has a default timeout of 15 seconds, for example).

In figure 6.8 we look at connection start and end times from a different perspective. In figure 6.8(a) we represent the distribution of the time differences between consecutive connection starts. As expected, consecutive connections of the same webpage download are generally opened very close in time. Around 30% of consecutive connections are opened with less than a tenth of a millisecond between them (for smaller values, some precision/rounding artifacts appear). Furthermore, only around 5% of the connections have their start times separated more than one second. Figure 6.8(b) is equivalent to 6.8(a) but this time we are representing the difference in end times of connections that are closed consecutively. We can see that, again, differences are usually very small. Because most connections are opened very close in time at the beginning of the download of the webpage, the effect of the persistence timeouts is very apparent when comparing their ending times. This, together with the fact that connections are ended immediately when the browsers are closed (note that in the figure around 20% of differences are 0), implies that the connections of the same webpage download usually end in almost simultaneous groups.

For table 6.2 we wanted to offer per-download metrics that describe the start and end timestamps of a webpage download. Again, we provide the median, 10th and 90th percentiles. The first metric we consider is the time of start of the last connection in the download (T. Last) as seen in figure 6.6(b). However, this time may not, in some cases, represent the time interval during which most of the webpage is downloaded. In fact, if we consider the 10th and 90th percentiles we realize that we are basically covering the whole length of the captures. To give a better idea of the busiest time interval we consider the connections that carry the first 90% of the total data downloaded in each capture and provide the start time of the last of these connections (T. 90%).

Table 6.2: Time metrics (all values in seconds).

	Firefox			**Chrome**		
Metric	**P10**	**Median**	**P90**	**P10**	**Median**	**P90**
T. Last	1.62	25.96	105.82	2.34	21.29	117.10
T. 90%	0.61	2.67	12.51	0.76	3.36	17.85
T. Starts	0.00	0.01	0.12	0.00	0.02	0.13
T. Ends	0.00	0.04	0.32	0.00	0.01	0.22

With this we eliminate the effect of small connections opened late in the download and give a better approximation of how close the connections of a webpage download are in time. For the time differences between flow starts and ends, in figure 6.8 we considered all flow pairs in every download but here we want to provide a per-download metric. We have calculated the median value for each capture and, in table 6.2 we show the median and percentiles of the distribution of said medians (T. Starts and T. Ends). As expected, the values of these two statistics are very low for almost every webpage download.

6.5 Server metrics

As we saw in chapter 5, of the classic 5-element tuple that traditionally describes an IP connection, the most interesting parameter when studying web traffic from the client point of view is the server IP address (protocol is always TCP, server port is 80 or 443 and client port is ephemeral and will not give us any infor-

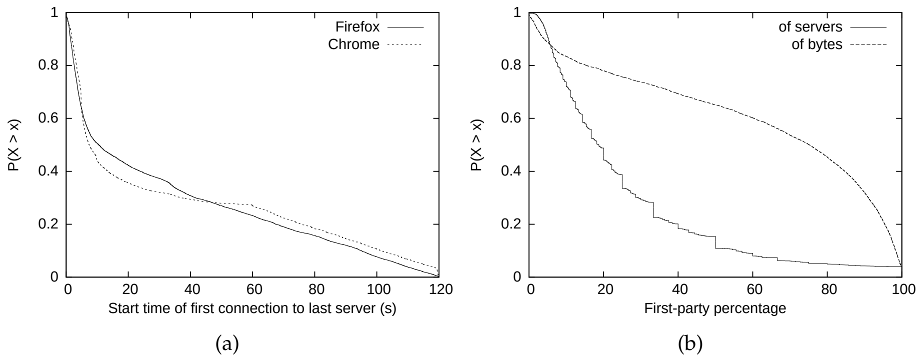


Figure 6.9: CCDFs of (a) time of first connection to last server and (b) first-party percentage of servers and bytes.

mation). Because of this, in this section we are going to center our study in the different servers accessed during the download of a webpage, considering that each of them corresponds to a different server IP address. For the sake of brevity, we only use Google Chrome data in most of the figures of this section. The results for Firefox are very similar because, aside from very specific browser services, the servers accessed during the download of a webpage should not vary depending on the browser that we use.

In section 6.3 we saw that multiple connections are opened to the same servers during each download. On the other hand, in figure 6.6(b) we realized that some of these connections happen very late in the captures. We wonder if these late connections are opened to servers that have already been accessed or to new ones. Figure 6.9(a) addresses this question by representing the CCDF of the start timestamp of the first connection to the last server that appears in each capture. The results are similar to the ones in figure 6.6(b) suggesting that a sizable number of these late connections are opened to servers that have not been previously connected. However, as we said, these connections are usually of small size and the servers that host the main elements of the webpages are accessed in the first seconds of the download.

In section 6.3 we also gave information about the total number of different web servers accessed in a download and the different DNS authoritative nameservers related to them. However, it is clear that those servers play different

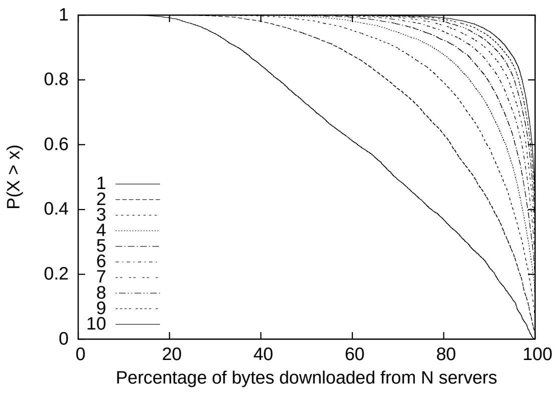


Figure 6.10: CCDF of the percentage of bytes downloaded from heaviest servers.

roles in the webpage downloads depending on the elements they host or the services they provide so it should be interesting to study them individually. In figure 6.9(b) we show the distribution of the percentage of first-party servers for each webpage download (that is, the server the root connection is directed to, shared-name servers and same origin servers; see section 6.2.3). We can see that, for most webpages, this percentage is low implying that most servers accessed during a download are third-party servers. However, the percentage of bytes downloaded from these first-party servers is much higher so, even if fewer first-party servers are accessed during a webpage download, they usually host a bigger part of the webpage content than the third-party ones.

Another consequence of figure 6.9(b) is that, as we know, the content of a webpage is not equally distributed in the different servers accessed during the download. In figure 6.10 each CCDF represents the percentage of content downloaded from the "heaviest" server, the two heaviest servers and so on, of each capture. Even if many servers are accessed to download certain webpages, most of the content is hosted by few of them. For example, for 90% of the webpages, more than 80% of the downloaded content comes from only 5 different servers.

Until now, we have considered the servers in each webpage download independently. However, the content of many webpages is hosted in distribution networks or multiple hosts (for load balancing purposes) and because of this, connections to the same IP addresses are not always opened when accessing

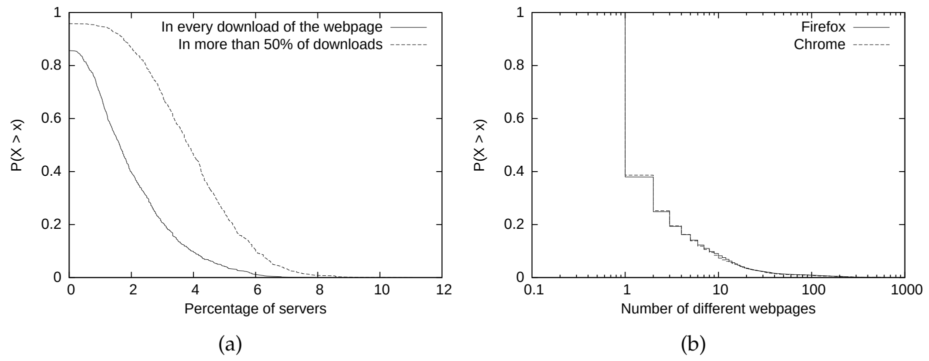


Figure 6.11: CCDFs of (a) percentage of servers that appear in all or half the captures of the same webpage and (b) number of different webpages that access each server.

the same webpages. On the other hand, a lot of webpages use third-party services (*e.g.* analytics, image hosting, advertising, etc.) and, as a consequence, the same third party servers are accessed when downloading different webpages. We now address these situations.

In figure 6.11(a) we consider all the IP addresses accessed in the ten captures we made for each website. We represent the CCDF of the percentage of servers that appear in every capture and that appear in, at least, half of the captures of each website. As we can see, both percentages are quite low. This means that most of the content in the webpages is dynamic or hosted dynamically and, because of that, the servers involved in the download change rapidly over time making very difficult to identify a particular IP address with a particular webpage.

Lets now compare all the servers accessed during the download of different webpages. Figure 6.11(b) represents the number of different webpages in whose downloads a particular IP address appears. For this figure we do not consider as different some of the webpages under study like, for example, amazon.com and amazon.co.uk, because they probably share an important part of their hosting infrastructure. A very high percentage (around 40%) of all the IP addresses appear in downloads of more than one webpage suggesting that a lot of content is shared between them.

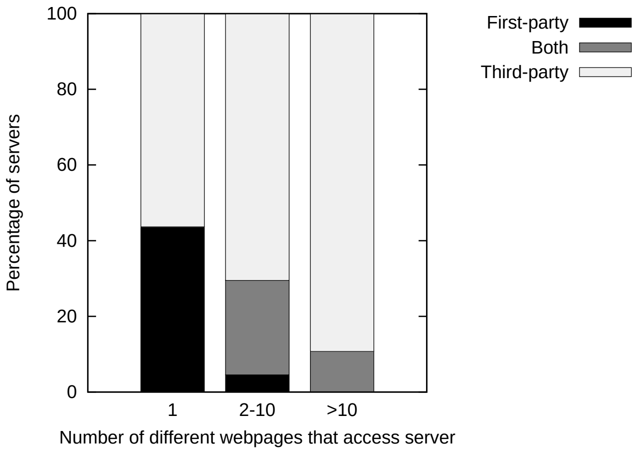


Figure 6.12: Origins of shared servers.

In figure 6.12 we have divided all the server IP addresses in three groups according to how many webpages download content from them. The servers of the first group only host content of one of the studied webpages, the ones in the second group host content of two to ten different webpages, and the servers in the third group host content of more than ten different webpages. For each group, we represent the percentage of first and third-party servers and servers that are both first and third-party depending on the webpage. As expected, most only first-party servers appear in downloads of just one webpage (the few of them that appear in 2-10 webpages are probably related to webpages that share a hosting platform like blogs). However, in a considerable amount of cases, servers that are considered first-party for a webpage appear in downloads of another one as third-party servers.

A consequence of all this is that even though IP addresses are an interesting parameter in order to group the connections of the same webpage download (as multiple connections are usually opened to the same servers) they should be used very carefully to relate different downloads of the same webpages. On one hand, the same content may be downloaded from different servers (or even the content itself may change in short periods of time). On the other, many servers are accessed by different webpages and even servers closely associated to a website by their name or their authoritative nameserver may host content for webpages of other websites. We have experienced this situation in chapter 5 where we were able to find only a relatively small number of related IP addresses for each website under study for these same reasons.

Table 6.3: Server metrics.

	Firefox			Chrome		
Metric	**P10**	**Median**	**P90**	**P10**	**Median**	**P90**
T. Last S.	1.42	10.48	95.01	1.89	7.42	101.86
T. 90 S.	0.15	2.03	7.25	0.30	2.40	10.54
% F.P. (S)	5.26	20.0	78.57	5.26	18.75	55.56
% F.P. (B)	5.02	69.79	99.62	4.08	75.18	98.34
% 1 Serv.	30.00	61.14	95.63	34.97	69.42	95.97
% 5 Serv.	75.77	96.05	100	82.92	97.28	100
% 10 Serv.	90.43	99.72	100	93.75	99.81	100

In table 6.3 we present some server metrics related to the variables we described in this section providing, again, median values and percentiles. As it happened in the previous section, the time of the first connection to the last server (T. Last S.) is not very representative so we have calculated the time of appearance of the last server that, together with the ones that have already appeared is responsible for 90% of the total download (T. 90 S.). This interval represents how long it takes for the servers that are responsible for the majority of the download to be contacted by the client. We also show the percentages of first-party servers and bytes (F.P. (S) and F.P. (B)) and the percentage of bytes downloaded from the heaviest 1, 5 and 10 servers.

6.6 Combining the metrics

In the past sections we have studied different aspects of the group of connections that compose a webpage download. We have provided some metrics about them and the normal range of values in which they usually move (defined by the 10th and 90th percentiles). Those ranges are, in most cases, quite broad and, because of that, any of the metrics is not enough to characterize a webpage load

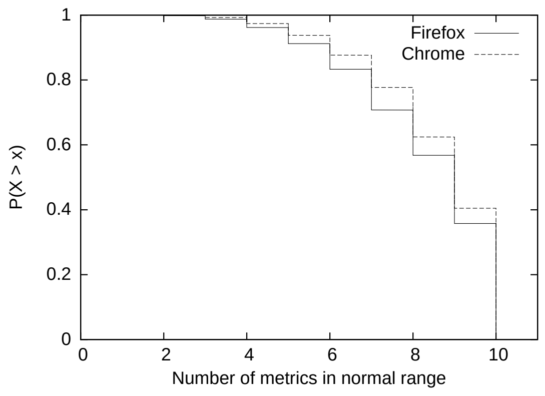


Figure 6.13: CCDF of the number of metrics in range for each capture.

by itself. Nevertheless, if we combine some of them we will have a thorough description that takes into account multiple aspects of the download.

Out of the fifteen metrics we have introduced in tables 6.1, 6.2 and 6.3 we have selected ten for figure 6.13. We do not consider the start times of the last connection (T. Last) or the first connection to the last server (T. Last S.) because the very broad ranges the percentiles provide for them suggest that they depend on capture length rather than on the characteristics of the download. In fact, we checked with the ten-minute-long captures that the client kept opening connections past the 120 seconds mark, some of them to new servers. We also ignore the number of authoritative nameservers (N. A.NS.) and the percentage of first-party servers and bytes (% F.P. (S) and % F.P. (B)) because, although they are descriptive of the origin of the elements that compose the webpage, they cannot be calculated using only connection level information which is our main focus in this thesis.

With these considerations, figure 6.13 shows the distributions of the number of metrics that fall into the normal ranges for each webpage download. Rather than considering the separate ranges provided for each of the browsers we have calculated the mean of the limits and used this average range for both of them. We do this in order to simulate an in-the-wild implementation in which the browser used by the client is not known. Because of how the normal ranges have been defined, around 20% of the captures will fall outside of them for each metric. However, when combining them we see that they are complementary in the sense that even if a webpage download fails to be inside the normal range for one metric it will be inside it for a different one. In fact, all of the captures fall into the range of, at least, two different metrics with more than 50% of the captures falling in, at least, nine out of ten. The results of Google Chrome are slightly better because the ranges of some of the metrics were more restrictive in its case, and it benefits from the new average ranges while Firefox captures meet tougher requirements.

6.7 Conclusions

In this chapter we have presented a thorough characterization of web traffic from a connection-level perspective. In order to do so we have collected a sample of landing webpages of a thousand popular websites. We have introduced various metrics that describe the set of connections involved in those webpage downloads focusing on their general characteristics, their distribution in time and the servers they reach. For each of these metrics we have shown its probability distribution and given some statistics to describe it.

Taking into account the very limited nature of the information available in Netflow-type records, we have painted an accurate picture of the average webpage download. More than 50% of the more than 18,000 captures studied fulfill at least nine out of the ten connection level characteristics considered, with all of them fulfilling at least two.

Because of the variability in the considered webpages, the ranges in which the different metrics move are often wide. However, focusing on time metrics, we have found that, even though it is common that new connections are opened late in the webpage download, most of the content of the webpages is downloaded in connections opened during a short period of time. Furthermore, most consecutive connections in the same webpage download are very close in time.

The knowledge obtained in this work will be applied in chapter 7 that covers the design of a method able to identify webpage downloads in real traffic. However, these metrics may also be used in order to distinguish between websites of different categories (like social networks, news portals, etc.) or which offer different services (video streaming, games, etc.). As the normal range of the metrics is quite broad, the variability in them suggests that information about the characteristics of a webpage (indicative of the related website category or service provided) can be extracted from this kind of connection level data.

Other possible applications of this work are more related to network security as a thorough characterization as the one provided here can help with the tuning of anomaly-based detection systems. These systems may be able to distinguish between normal web browsing and other applications that masquerade their traffic in order to avoid restrictions imposed by network administrators (or, in the case of malicious applications, in order to blend in and avoid detection).

CHAPTER 7

Identifying webpage downloads

In this chapter we present a method for clustering together the connections involved in the same webpage download without the need of deep packet inspection. As far as we know this has not been attempted before, although it is useful for characterization and accounting purposes. We have considered different approaches to the problem and designed a method based on a well-known clustering algorithm. Our method relies on the start timestamps of the connections as we have seen that it is a reliable parameter in order to find connections related to the same webpage download. In order to develop and test our proposal we use both real traffic from our university's Internet link and the experimental captures presented in chapter 6. We have also defined metrics based on DNS and HTTP information that describe how well formed our clusters are. Using these metrics we have validated our method obtaining results close to the achievable maximums.

7.1 Introduction

In chapter 6 we captured a large number of webpage downloads in a controlled setting and we described them through different connection-level statistics. In this chapter we apply that characterization for developing a method able to cluster together the connections related to each individual webpage download. In that sense, the problem we face here is similar to the one described in chapter 4, where we presented a method for clustering connections that belong to the same website session, but considering a different aggregation level.

As we have previously said, webpage downloads are easier to work with than website sessions as they span shorter periods of time which are independent of user actions. However, from a network perspective, if we consider the set of connections opened by a web client during a period of time, it is still complex to ascertain the download of which webpage has originated each one of the connections.

As far as we know, this problem has not been tackled by the scientific community although resolving it could have a variety of applications. On the one hand, network administrators would be able to collect information about the habits of their web users: how many webpages they visit, how many times they visit the same ones, how much time they spend visiting them, etc. This kind of information can now be obtained by installing software in the users' computers (e.g. Alexa Toolbar [Alexa]) but we would be able to do it just by analyzing their traffic. On the other hand, the group of connections related to a webpage download offers a lot of information about said webpage that may be used to characterize it and identify the service it provides (webmail, social networks, video streaming, etc.). This would allow network administrators to prioritize important traffic and block undesired connections.

As in previous chapters, we work with flow records which offer a very summarized description of each connection that does not include sensitive user information. We do use application-level data but only to validate our results; our clustering algorithm does not use it in any way. This makes it possible for our method to run in real time and independently of encryption.

In order to design, test and validate our clustering method we have used both real traffic traces and the automatic captures of webpage downloads described in chapter 6. After testing multiple approaches to our problem, we present a method based on the DBSCAN clustering algorithm [EKS+96] which was designed for density-based clustering in noisy databases. We believe that it is a good option for our purposes as most of the connections of a webpage download are usually very close in time but there is a lot of noise in the form of automatic web connections initiated by the web browser and other programs (*e.g.* antivirus and system updates, online storage, etc.).

7.2 Data collection

In this section we describe the experimental data we use in order to design and test our clustering method. We use traffic traces from two different sources: the automatic captures of webpage downloads in a controlled setup that we used in chapter 6, and traces of real traffic from web users in our university network.

In chapter 6 we presented and described a data set of more than 20,000 downloads of the landing pages of a thousand different popular websites. We extracted them from the the top 100,000 websites of the Alexa global ranking taking care that the most popular and interesting sites, like Google, Facebook, or Amazon were well represented while also collecting data from a wide variety of less popular sites from all around the world. We accessed the landing pages of these websites automatically from a computer in our university network and captured the resulting traffic during a time interval of two minutes. In this way, each capture contained the download of an individual webpage. We used both Firefox and Google Chrome as web browsers. We refer to the characterization of these captures during the design of our clustering system and we come back to them in order to test our validation method.

Nevertheless, testing the clustering method we present here requires real traffic from real web users. For this purpose we have captured web traffic from the Internet link of the Public University of Navarre. Traffic is captured directly with Argus and, as we did in the previous chapter, in addition to the parameters listed in chapter 3, we calculate the number of downstream and upstream application-level bytes and store the first 1,000 bytes of the upstream application data of every connection. HTTP header data will not be used in our clustering method but will help us in validating it.

We consider the traffic of around 500 unique users for which we have captured their web traffic during two weeks starting on Monday, October 28th, 2013. Out of this time interval we have selected only work days (9 days total as November 1 is a national holiday in Spain) from 8:00 to 21:00 and we have eliminated the traffic of users that (1) start less than 500 connections a day, (2) have a mean connection size of less than 10KB, or (3) have a median time between consecutive connections of more than 1 second. These thresholds are meant to discard users that are not useful for our study. Users that do not meet the requirements either do not have enough traffic to be significative (500 connections represent around 10-15 webpage downloads) or have a traffic profile that is not typical of a user actively browsing the web and which may be related to other applications using web ports (connections too small and too far in time from each other). The experimental measures behind the choosing of the thresholds have been calculated from the automatic webpage downloads data set and are shown in table 7.1. As we can see, the thresholds are far from restrictive and, after this final filtering step, we are left with 250-300 active users depending on the day of the capture for a total of 439 different users for the considered 9 days.

In addition to the flow records, we have captured all DNS traffic between our DNS server and the Internet during the two weeks under study (as shown

Table 7.1: Thresholds for considered users and related experimental measures

Threshold	**Experimental measure**
Connections per day 500	*Median number of connections per download* 43 (Firefox), 36 (Chrome)
Mean connection size 10KB	*Median mean connection size per download* 23.0KB (Firefox), 24.4KB (Chrome)
Median time between consecutive connections 1 second	*Median mean time between consecutive connections* 0.04s (Firefox), 0.01s (Chrome)

in figure 3.2, our vantage point does not allow the capture of DNS traffic from each user). In this case we capture full packets in pcap format as we will extract information from fields in the DNS payload. We only use DNS data in order to obtain information about HTTPS connections for validation purposes.

7.3 Identifying webpage downloads from a flow-level perspective

In this section we describe webpage downloads from a flow-level perspective but focusing on connection parameters that can be used to identify which connections belong to the same webpage download. We consider that a webpage download comprehends all the data exchanged between a web browser and one or multiple web servers in order to render a specific webpage in the former. From a flow-level perspective, a webpage download comprehends the connections opened by the web browser after a user types a URL address or follows a hyperlink to a new webpage and which carry the content of said webpage or are somehow triggered by accessing it (*e.g.* Google Safe Browsing).

Throughout the section we use data from the automatic webpage captures presented in chapter 6. As we said previously, we use Argus in order to extract a number of flow-level parameters from traffic traces. The first thing we do is ask ourselves which of these parameters can be used to cluster the connections that belong to the same webpage download:

- **Client IP address:** we use client IP addresses to identify users in our network. Obviously, connections of the same webpage download will share the same client IP address.
- **Server IP address:** server IP addresses host the content of the webpages. Usually, multiple connections to the same server IP addresses are opened during the download of a webpage. Some of these servers are related to the organization that owns the webpage (e.g. a Facebook server providing Facebook content) but, with the increasing popularity of CDNs and third-party content, a sizable amount of them are not. Nevertheless, server IP

addresses are interesting for clustering connections of the same webpage download and we discuss them in detail in section 7.3.2.

- **Protocol:** useless as we only capture TCP traffic.
- **Client TCP port:** useless as browsers use ephemeral client TCP ports that are chosen by the operating system.
- **Server TCP port:** as we only consider traffic to ports 80 and 443, the server TCP port only allows us to distinguish between HTTP and HTTPS connections. However, many modern websites use a combination of HTTP and HTTPS and the portion of encrypted traffic in webpage downloads is very variable.
- **Start timestamp:** as the content of a webpage is downloaded as fast as possible in order to provide the best user experience, the connections that take part in the download are usually opened very close in time. Because of this, start timestamps are very interesting for our purposes and we will discuss them in section 7.3.1.
- **End timestamp:** as shown in figure 6.8(b) two connections of the same webpage download have a higher probability of ending close in time. However, in most cases this is an effect of them also starting close in time. Non-persistent connections are usually very short and persistent connections have lengths fixed by persistence timeouts in the client (the same timeout for all connections of the same browser) or in the server (usually the same —default— timeout for connections to the same server software, *e.g.* Apache HTTP server 2.2: 5 seconds). Because of this, it is probable that two connections of the same webpage download start close in time and have a similar duration, thus ending also close in time. The only case in which end timestamps offer non-redundant information is when a user closes a webpage by closing the web browser or the specific tab. All connections related to the webpage that are still open will be closed at this time. However, it is difficult to know how many webpages the user is closing at the same time.

- **Connection size (in bytes or packets):** useless as connections of the same webpage download are very variable in size and, although some services tend to produce bigger connections (e.g. video streaming), they will also use smaller ones to download accessory content.

Taking all this into account, we believe that start timestamps and server IP addresses are the most promising connection-level parameters for clustering connections that belong to the same webpage download. In the following subsections we discuss these parameters in detail.

7.3.1 Start timestamps

In order to illustrate the start timestamps of connections involved in different webpage downloads, we consider a real traffic capture of an example browsing session of around one minute of length. In this session, a user accesses his Facebook profile at the beginning of the capture; after browsing through it for 20 seconds, he follows a link to an article in a local newspaper website (diariodenavarra.es); and after reading it, he follows a link to another article in the same newspaper website. All in all, the session comprises two website sessions and three individual webpage downloads. In figure 7.1(a), the top sub-figure shows the evolution of the number of active connections during the browsing session and the bottom sub-figure, the instants in which connections are opened. As expected we can see that the connections of the same webpage download are very close in time with a short interval after the start of the download concentrating most of the connections. In this case, most of the connections are persistent and, in fact, some of those opened for the first newspaper webpage are used in the second to download new content.

At the end of the capture, the user closes the browser and connections still open are terminated at the same time. It should be noted that some of these connections have been used in downloading different webpages. As we said previously, this is one of the reasons why we do not consider end timestamps for our clustering method.

In order to study connection start timestamps in the download of webpages from a variety of websites, we use the dataset of automatic webpage downloads

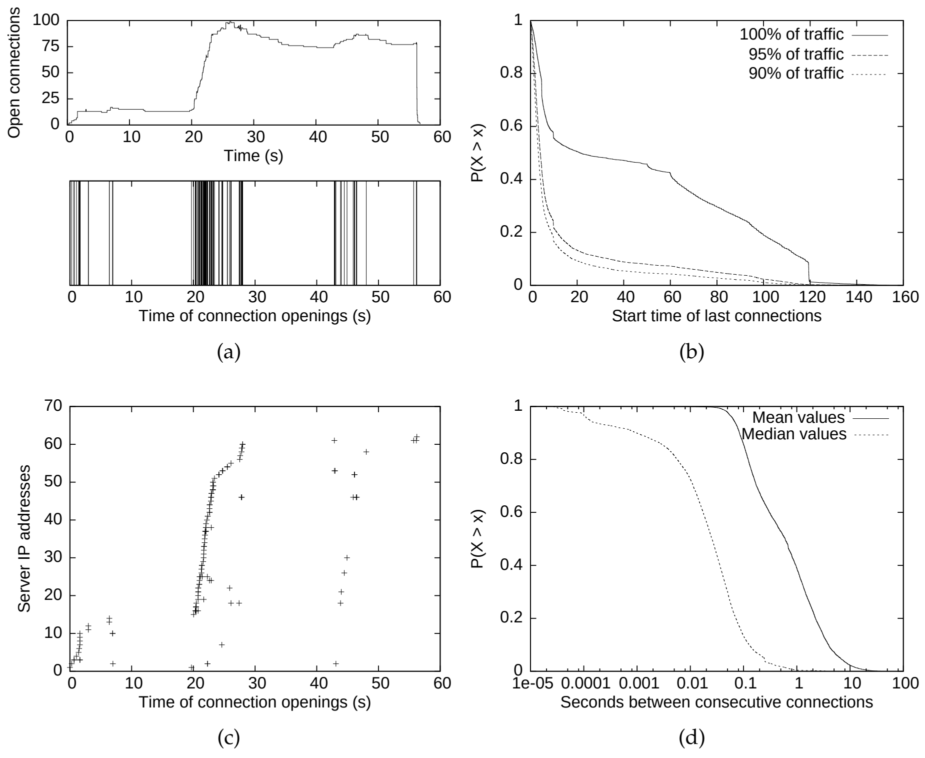


Figure 7.1: (a) Start timestamps for an example web session, (b) start time of last connections in experimental webpage downloads, (c) connections to different servers for an example web session, and (d) seconds between consecutive connections in experimental webpage downloads.

described in chapter 6 (we have chosen to use Chrome data for brevity as both browsers performed similarly). In figure 6.6(b) we represented the complementary cumulative distribution function of the start timestamps of the last connection for each capture. For figure 7.1(b) we add two CCDFs: we consider the set of connections that carry the first 90% and 95% of the traffic in the captures and represent the CCDFs of the start timestamps of the last connection of those sets.

This gives a better idea of the time interval during which most of the webpage is downloaded. As we can see, although connections opening late in the capture are a relatively common occurrence (our captures were limited to 120 seconds), most of the traffic is concentrated in the connections opened during the first seconds. We can say that 90% of the traffic is carried by connections opened in the first 20 seconds for more than 90% of the captured webpage downloads (P90 = 17.85s) although it is a heavy-tailed distribution and the median is in fact much lower (P50 = 3.36s).

In figure 7.1(d) we consider time differences between consecutive connection starts (we skip figure 7.1(c) that deals with server IP addresses and to which we will come back in the following section). For figure 7.1(d) we have calculated the median and mean value of the time differences between consecutive connection starts in each capture and represented the CCDF of both statistics. The median CCDF shows that connections of the same webpage download are usually very close from each other although, as evidenced by the mean CCDF which is far less robust against extreme values, some differences are bigger. If we consider the 90th percentiles of both distributions we obtain values of 0.13 and 3.95 seconds respectively. These values are small enough if we compare them with the time a user usually spends in a webpage.

Webpage dwell times are difficult to model as they depend on the user's navigating habits, the interest and complexity of the webpage and its actual content. For example, a user will take some time in reading a news article or watching a video but may follow a link to another webpage rapidly after using a search engine. Nevertheless, a minimum dwell time of 30 seconds has often been used for webpages that are of interest to the user [KHW+14] although very simple or uninteresting webpages may experience lower dwell times.

In any case, it should be noted that, in some cases, a user may open two (or more) webpages at almost the same time. This is relatively common today for advanced users that queue webpages they are going to visit in the tabs of their web browser. A limitation of a clustering scheme based on start timestamps is that it will be unable to distinguish between the different webpage downloads in those cases.

In brief, we can describe a typical webpage download as a group of connections whose start timestamps span a limited time interval and leave small gaps between them. We have quantified both intervals by calculating the 90th percentile from our experimental dataset obtaining values of 17.85s and 0.13s (for per download medians) respectively. This description suggests that we can use start timestamps to group the connections of the same webpage download.

7.3.2 Server IP addresses

Figure 7.1(c) shows information about IP addresses from the same example web session of the previous section. We present it below figure 7.1(a) for comparison purposes. In this figure we show the start timestamps of the connections in the example capture but this time we represent the accessed server IP addresses in the ordinate axis. We assign a number to each IP address according to their order of appearance in the capture. Three facts stand out from the figure: (1) although, intuitively, multiple connections should be opened to the same servers during a webpage download, in this example, the number of servers contacted is of the same order of magnitude as the number of connections; (2) webpages of the same website require connections to the same servers as it happens with the two different webpages of the newspaper site; and (3) there are some common servers that are accessed when downloading webpages of different websites.

In order to study how many connections are opened to the same servers are opened during a webpage download, we present figure 7.2. In it, we show the CCDF of the ratio of accessed server IP addresses against the number of connections per download. By default, modern browsers usually limit to 6 the number of concurrent open connections to the same server. If 6 connections were opened to each server in the download of a webpage, the obtained ratio would be 0.17 (smaller values are possible because all connections to the same server are not necessarily open at the same time). However, for the vast majority of webpage downloads, this ratio is much bigger. This suggests that the content in a sizable number of webpages is distributed through multiple different servers, some of them hosting a small fraction of it.

These servers will usually receive only one or two connections during the download. Since the number of connections required for downloading most webpages is considerable (from our experimental data, in table 6.1, we calculated a median value of around 40), only a relatively small fraction of them will share the same server IP addresses.

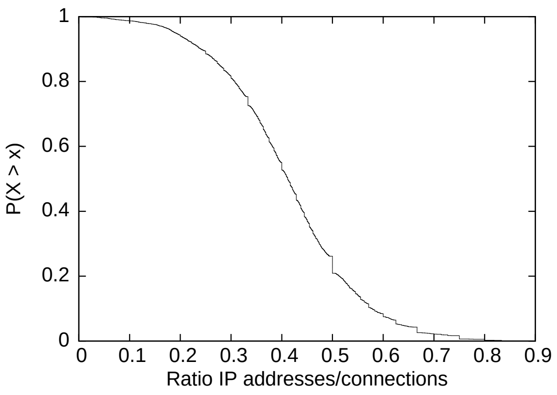


Figure 7.2: CCDFs of the ratio of server IP addresses/connections for each webpage download.

However, the biggest issue with using server IP addresses for our clustering method was shown in figure 6.11(b). There, we saw that there is an important number of shared IP addresses in the downloads of webpages and that two different webpages hosting some of their content in the same servers is a common occurrence.

Because of these findings, we believe that server IP addresses are not as useful for our purposes as they may seem intuitively. Modern sites download content from a big number of different servers, a substantial fraction of which belong to third-party services that may be shared with other websites. In fact, in chapter 5 we attempted to discover server IP addresses related to specific websites by studying multiple sessions to them and we were only able to obtain a very limited number of IP addresses per website. This suggests that modern websites are less and less centralized in easily identifiable servers. As a consequence we have decided to center our proposal around connection start timestamps and we use server IP addresses only for validation purposes.

7.4 Time-based clustering: two naive approaches

In the previous section we decided to center our study around connection start timestamps because we discovered that connections of the same webpage download usually happen in a short interval of time and very close to each other. Here, we attempt to use these findings to propose two different clustering approaches that attempt to find clusters of connections in user traffic that correspond to individual webpage downloads. In the following subsections we will describe these approaches and evaluate their operation by testing them using our 9-day real-traffic trace. For comparison purposes we will refer to two metrics shown in tables 6.1 (number of connections per webpage download, N. conn.) and 6.2 (seconds for 90% of bytes, T. 90%). We remind their experimental values in table 7.2.

Table 7.2: 10th, 50th and 90th percentiles of number of connections and webpage download time (for 90% of bytes).

	Firefox			**Chrome**		
Metric	**P10**	**Median**	**P90**	**P10**	**Median**	**P90**
N. conn.	6	43	125	10	36	104
T. 90%	0.61	2.67	12.51	0.76	3.36	17.85

7.4.1 Download-wide timer clustering

The first clustering method takes advantage of the fact that most connections of a webpage download are concentrated in a short time interval. In this scheme, we monitorize the traffic of each user. When, after a period of inactivity, a user initiates a connection, we set up a timer. This first connection and those initiated before the timer expires are considered part of the same cluster (clusters must, at least, contain two connections; isolated connections are discarded as noise). A careful selection of the value of the timer is key for the correct operation of this method: a timer too short will split webpage downloads in multiple clusters; a timer too long will jumble up separate webpage downloads.

In order to set the value of this timer, we consider figure 7.1(b) and table 7.2. We have decided to work, again, with the group of connections that concentrate the first 90% of the total downloaded bytes because we believe it is more representative and we would be satisfied if our clustering method was able to group this set of connections. Table 7.2 shows the 10th, 50th (median) and 90th per-

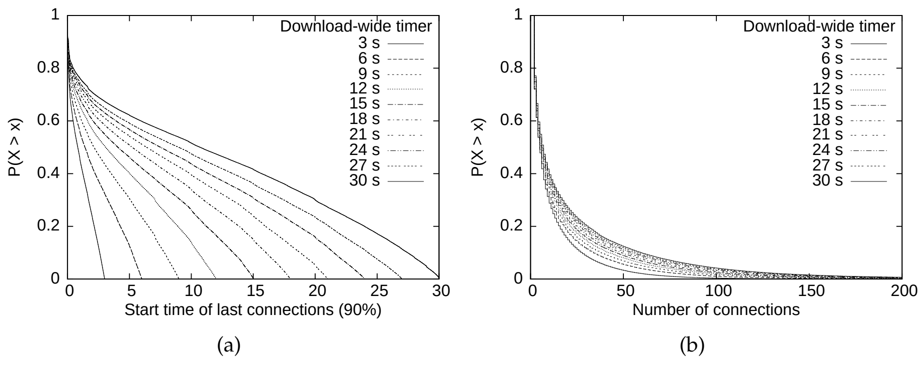


Figure 7.3: Download-wide timer clustering results for different timer values: CCDFs of (a) cluster length and (b) number of connections per cluster.

centiles of this 90% length. As we said previously, although the 90th percentile is close to 20s for Google Chrome webpage downloads, we see that median values are much lower. In the end we decided to sweep the timer from 3s to 30s hoping that we would be able to find an operation point that fitted most webpage downloads. The results of this sweep are presented in figure 7.3.

We expected that, as we increased the timer value, we would find a point where the obtained clusters would stop growing because most of the webpage download would be already inside of the considered time interval. Then, if we continued increasing the timer, the clusters would start to grow again as some of them started to comprise more than one webpage download. However, the results we obtained (shown in figure 7.3(a)) are quite different. We see that the length of a considerable fraction of the obtained clusters is very dependent on the chosen timer. This should not happen if the traffic of the users was composed of concentrated webpage downloads and idle connection-free intervals between them.

The reasons for this dependency on the timer are multiple: (1) even though most of the download is concentrated in a short interval, some connections are opened later while visiting a webpage (*e.g.* dynamic content, analytic services and others), (2) a sizable fraction of users run applications that use web ports and open connections regularly (*e.g.* some update services), and (3) in this simple scheme, the first connection of a cluster may not be the first connection of a webpage download but one of the “noise connections” described in (1) and (2). Starting the download-wide timer with this “noise connection” may divide in half a following webpage download. Additionally, figure 7.3(b) shows that a big fraction of the clusters created are very small (when compared with the expected results in table 7.2) most of them comprising either parts of webpage downloads or just noise connections. Because of this, we believe this scheme is flawed: a download-wide timer is not flexible enough to identify the webpage downloads in the traffic of a user.

7.4.2 Inter-connection interval clustering

The second clustering method uses the fact that consecutive connections of a webpage download are very close in time. Again, we monitorize the traffic of a user and when, after a period of inactivity, the user initiates a connection, we start collecting a cluster. This cluster will grow with new connections as long as the time between the start timestamps of the last connection in the cluster and a new one is smaller than a preset interval (clusters must, again, have at least two connections). As it happened with the previous approach, setting this interval is a crucial step in order to gather complete webpage downloads and avoid mixing different ones. By reviewing the experimental data presented in the previous sections, we have decided to sweep this threshold between 0.5s and 5s. The results are shown in figure 7.4 which is analogous to figure 7.3.

Figure 7.4(a) shows that cluster length is not as influenced by the chosen threshold. This makes sense because the noise connections that disrupted the operation of the previous method will not be taken into account here unless they are very close to the connections of the webpage download. In fact, it is now less problematic if the first connection of a cluster is a “noise connection” as there is no time limit from the start of this connection and the end of the cluster and webpage downloads will not be splitted because of this reason. However, this approach still suffers important issues: it still creates a big number of small clusters (figure 7.4(b)) which are, in most cases, composed of noise connections; and, for higher values of the inter-connection interval, it can generate massive clusters where multiple downloads are mixed. This is specially a problem with very active users or those that run background applications using web ports.

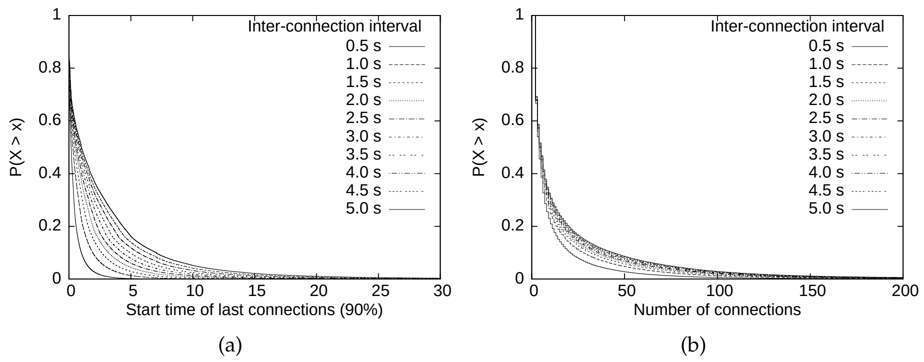


Figure 7.4: Inter-connection interval clustering results for different interval values: CCDFs of (a) cluster length and (b) number of connections per cluster.

7.5 Time-based density clustering

Our experience with the two naive approaches presented in the previous section tells us that this is a problem that cannot be solved so simply. Even though it is true that connections of the same webpage download are close to each other and happen during short intervals of time, it is difficult to take advantage of these facts when dealing with real web traffic. The variability of user behaviors and the disruptions introduced by other applications using web ports generate complex traffic from which webpage downloads can be difficult to select. We have tested combinations of the two approaches and obtained slightly better results but, in the end, we have relied on a modification of a well-known and tested method that could adapt to our needs.

Grouping one-dimensional data is a different problem than grouping multi-dimensional data because the former can be sorted simplifying the process. As a consequence, most clustering algorithms are designed for multidimensional data and using them in one dimension is inadvisable. Nevertheless, there are some specific methods designed for grouping one-dimensional data.

A classical one, *Jenks natural breaks optimization* [Jen67], is a method of statistical data classification that partitions one-dimensional data into classes. It seeks to reduce variance within the classes and maximize variance between them. However, it requires, as an input parameter, the number of classes in which the data will be divided (the same restriction as the K-means clustering algorithm which is a generalization of it). As we do not know beforehand how many webpages a user accesses during a period of time, we cannot use this method.

Another approach is estimating the probability density function (PDF) of the variable. In our case, studying the shape of the density function of the connection start times could allow us to identify intervals of high density of connections and therefore, the webpage downloads. However, estimating the PDF is complicated. We could use simple histograms but the information they provide can be very misleading depending on the selected box width. On the other hand, complex approaches as *Kernel density estimation* [Par62] provide a more faithful estimation but add taxing computational requirements. In any case, the PDF cannot be estimated on the fly preventing the system from working in real time. Moreover, even with a good PDF estimation, using it to identify the high density intervals is a far from trivial task.

Taking all this into consideration, we have decided on an adaptation of the DBSCAN clustering algorithm [EKS+96], which was specially designed to work in large databases with noise and is nowadays one of the most popular clustering algorithms. Even though it was designed for multidimensional data, we have modified it in order to work in one dimension taking advantage of the opportunity of sorting the data. These modifications allow our algorithm to operate in real time (i.e. clustering connections as they are captured) rather than process a database of already captured traffic.

The method we propose here operates in a similar way to the inter-connection interval clustering approach that we presented in the last section as, again, the time between consecutive connections is central to the algorithm. However, this new method introduces a mechanism to avoid forming too small clusters and the density considerations control cluster growth in very active or noisy users. Those were, precisely, two of the biggest drawbacks of the previous proposal.

In the following subsections we describe the time-based density clusters (in our case each cluster will ideally comprehend a webpage download), provide a scheme of our algorithm, explain the tuning process and present some preliminary results.

7.5.1 Describing the clusters

Like DBSCAN, our algorithm depends on two parameters: a time interval between connection starts Tbc and a number of connections Nc. With these parameters and considering U as the set of connections opened by a certain user, we present the following definitions:

Definition 1: given two connections, x and y, and their respective start timestamps, T_x and T_y, the *distance* between the connections is defined as:

$$dist(x, y) = |T_x - T_y| \tag{7.1}$$

Definition 2: the *Tbc-neighborhood* of a connection x, denoted by $N_{Tbc}(x)$, is defined as:

$$N_{Tbc}(x) = \{y \in U \mid dist(x, y) \leq Tbc\} \tag{7.2}$$

Therefore, x and y are *neighbor connections* if $y \in N_{Tbc}(x)$.

Definition 3: a connection x is a *core connection* if it verifies that $|N_{Tbc}(x)| \geq Nc$.

Definition 4: a *cluster core* is a set of consecutive core connections $c_1, ..., c_n, n \geq 1$ for which $\forall i < n, c_i \in N_{Tbc}(c_{i+1})$.

Definition 5: a *cluster border* is the set of the neighbors of the connections in a cluster core that are not core connections.

Definition 6: a *cluster* is a set of consecutive connections formed by a cluster core and its cluster border.

Definition 7: connections that do not belong to a cluster, that is, they are neither core connections nor in the neighborhood of one, are considered *noise*.

With these definitions, the clusters that we create have a core with a high density of connections and a less dense border. If for a cluster, T_{c1} is the start time of the first core connection and T_{cn} is the start time of the last, the *total cluster length* (TCL, the time difference between the start times of the first and last connections in the cluster) has an upper limit: $TCL \leq T_{cn} - T_{c1} + 2 * Tbc$. This means that total cluster length is limited by the length of the cluster core and thus clusters only “grow” through periods of time where the density of connections opened by the user is high.

Given the previous definitions, a connection may belong to the cluster border of two different clusters. If this happens, our algorithm will assign the connection to the oldest one. This makes sense because, as we have seen previously, the density of connections is usually higher at the beginning of a webpage download so we expect to find more connections in the cluster border after the core than before.

7.5.2 Time-based density clustering algorithm

Figure 7.5 shows a flow diagram of our clustering algorithm. In order to explain its operation, lets consider the traffic of a particular user (the algorithm treats each user individually).

When a connection is opened by the user the algorithm checks if there are neighbors in a connection array that it keeps for each user. If the array is empty there are, obviously, no neighbors and the connection is simply added to it. If the array is not empty but none of the connections are neighbors of the new one, the array is emptied and the new connection added to it afterwards. In this array, connections are identified by the classic 5-tuple and the timestamp of their first packet and a neighborhood size counter is kept for each one of them with the number of neighbors they have in any given moment.

If, on the other hand, there are neighbors in the array, the connection is added to it and neighborhood size counters are updated in the array for the new connection and its neighbors. We then check the neighborhood size of the older neighbor of the new connection. If this neighbor is not a core connection, none of the more recent ones can be one (because their neighborhoods are equal or smaller). In this case, we do nothing more and wait for the next connection. However, if the oldest neighbor is a core connection it has to be part of a cluster. In this case, we check if it already is part of a cluster (it may have been a core connection or part of a cluster border before the new connection was opened). If it is, we make all its neighbors part of the same cluster. If it is not, we create a new cluster and make all its neighbors part of it. An exception to this occurs if a neighbor already belongs to a different cluster in which case we leave it as it is. This is the case of border connections that could belong to two different clusters which, as we said in the previous subsection, are assigned to the oldest one.

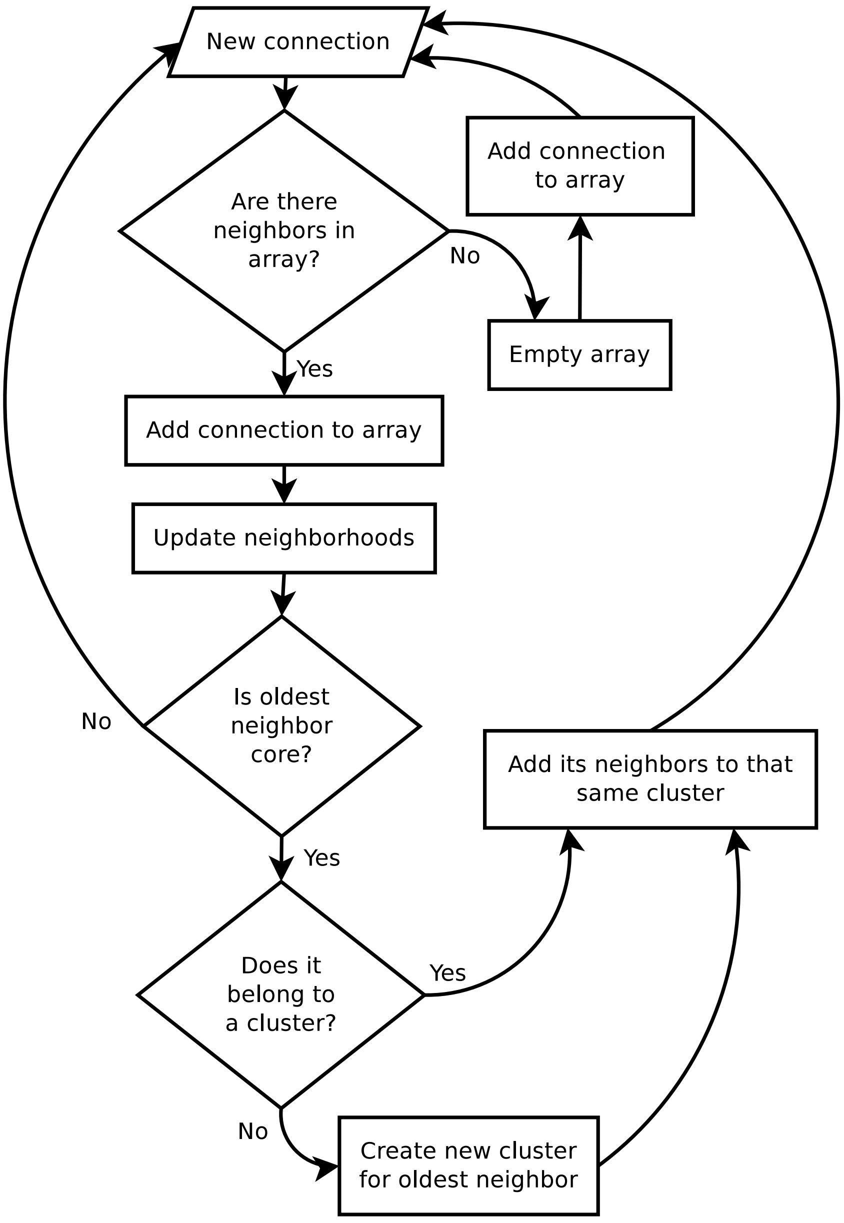

New connection
Are there neighbors in array?
No
Yes
Add connection to array
Empty array
Add connection to array
Update neighborhoods
Is oldest neighbor core?
No
Yes
Does it belong to a cluster?
Yes
No
Add its neighbors to that same cluster
Create new cluster for oldest neighbor

Figure 7.5: Flow diagram of the clustering algorithm.

As we have explained, the array of connections that the algorithm uses for each user is emptied when the new connection has no neighbors in it. That is, when the time interval between the new connection and the most recent connection in the array is bigger than Tbc. In this case, the new connection cannot be part of the same clusters as any of the ones in the array, and it is not necessary to keep them in memory. When the array is emptied, the connections in it that do not belong to a cluster are considered noise and discarded.

7.5.3 Tuning the parameters

Our algorithm depends on two parameters: a time interval between connection starts Tbc and a number of connections Nc. As with any clustering algorithm, selecting an appropriate value for these parameters is key for a correct operation. With data as complex and variable as web traffic, we do not expect to find exact values of the parameters that offer the best results in any possible setting. Rather than that, we would like to find ranges of parameter values for which the algorithm offers good results. A more precise tuning would be advisable on a network basis (or even taking into account the characteristics of different users).

Nc is the minimum number of neighbors required for a connection to be core. Because of this, it defines minimum cluster size and, in our case, it will be related to the minimum number of connections in a webpage download. In [EKS+96], a default value for Nc of 4 is proposed arguing that the DBSCAN algorithm behaves similarly with higher values while adding computational complexity. Nc=4 produces clusters of a minimum size of 5 connections which seems consistent with the experimental results from our automatic captures (for Firefox, the 10th percentile of the number of connections per webpage download is 6). However, this default value of Nc is given for 2-dimensional databases so we need to study how it behaves with 1-dimensional data.

On the other hand, Tbc is the maximum time between neighbor connection starts. In our case, its value has to be chosen carefully in order to avoid dividing a webpage download in multiple clusters while also avoiding clustering together connections of different webpage downloads. A *k-distances graph* can help in choosing a correct value for Tbc. In this graph we represent the time distances between connections and their kth nearest neighbors. According to

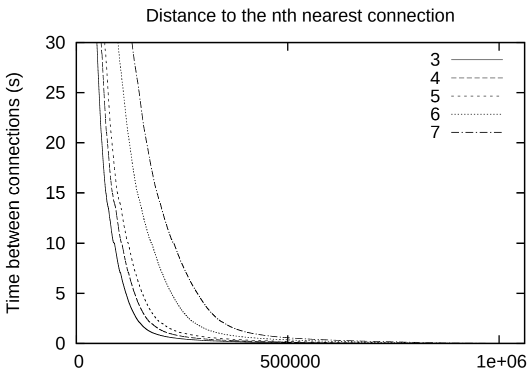
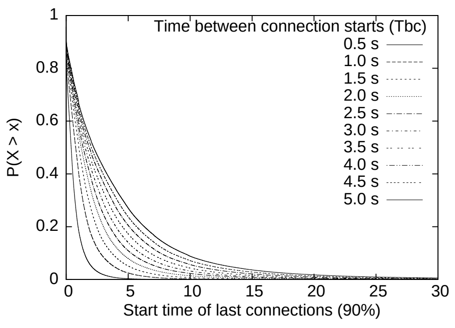


Figure 7.6: K-distances graph.

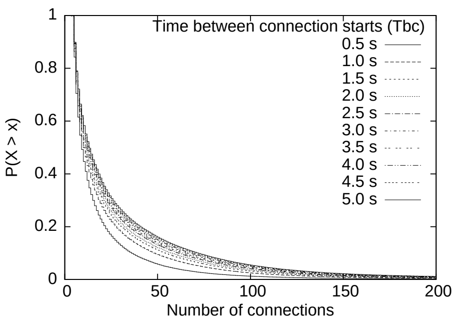
[EKS+96], if k=Nc, a "knee" in the k-distances graph marks a good value for Tbc distinguishing between core and non-core connections.

For figure 7.6 we have selected a workday from our two-weeks capture and calculated the k-distances (with k=3,4,5,6,7) for every one of the over a million connections opened during the day. We have then aggregated the data of every user and sorted all the distances from highest to lowest. We set a top limit of 30 seconds for the ordinate axis in order to focus on the interesting part of the plot. We can see that there is not a big difference between the lines for $k \leq 5$ while the 6-distances and 7-distances are higher. This suggests that a top limit for Nc should be 5 as with bigger values the smallest webpage downloads will not be considered clusters. As for Tbc, if we focus on $k \leq 5$, we can see a "knee" that marks that around 80% of the k-distances are smaller than 2 seconds.

In brief, for Nc we will follow the recommendations for tuning the algorithm and use Nc=4. This value is consistent with our experimental knowledge of the problem and the k-distances graph suggests that choosing Nc=3 or Nc=5 would not produce very different results. Choosing a exact value for Tbc is more complicated although we have seen that it should be in the vicinity of 2 seconds. In the next subsection we will test the algorithm with different values of Tbc in order to understand how it affects the clustering process.



(a)



(b)

Figure 7.7: Time-based density clustering: CCDFs of (a) cluster length and (b) per cluster number of connections for different parameter values.

7.5.4 Preliminary testing

As we did in section 7.4 with the two naive approaches, we have tested the density clustering algorithm with our 9-day real-traffic trace obtaining the results shown in figure 7.7. Comparing it with figure 7.4, we can see that the density clustering algorithm offers results similar to the inter-connection interval proposal (said interval and Tbc work in a similar way and, because of this, it makes sense to sweep them over the same range). However the density clustering algorithm has two immediate advantages over the simpler approach. As clusters only grow through core connections that have a minimum number of close neighbors, this method usually avoids creating massive clusters for very active users as it sometimes happened with the inter-connection interval approach. Moreover, as Nc is the minimum number of connections for a cluster, the method does not create very small clusters (less than 5 connections) that usually do not contain a full webpage download.

7.6 Testing and validation

In sections 7.4 and 7.5 we have proposed three different methods to tackle the problem of clustering connections that belong to the same webpage download. Those methods depend on parameters for which we have given approximate working ranges based on different experimental observations. In this section we will compare the operation of the three approaches and validate their results in order to choose the better one and tune its related parameter to an optimal value. For the sake of brevity we will refer to each method by an acronym: **DWTC** (download-wide timer clustering), **ICIC** (inter-connection interval clustering) and **TBDC** (time-based density clustering). As some of the validation carried out here is computationally expensive, we have chosen to work with data of only one day out of the 9-day traffic trace (in particular, October 29th). This will also allow us to test the final tuned system with the full trace and check if it still operates correctly.

In the figures of this section we offer different statistics about the clusters created by the three methods for different values of their respective parameters. We have swept the parameters in the same ranges discussed in sections 7.4 and 7.5. Although the parameters are different for each method, we have selected approximate ranges of optimal operation and, as a consequence, we believe it makes sense to compare the results. In fact, as we will see in the following figures, the behavior of the different statistics is similar for the three methods in the selected parameter ranges.

In order to offer a fair comparison of the three methods we have filtered out clusters of less than 5 connections from the results of the DWTC and ICIC approaches. In the vast majority of cases, these very small clusters do not comprehend full webpage downloads and it would be unfair to include them in a direct comparison with the TBDC method which would discard them as noise. Because of this, we have decided that the connections in these small clusters will also be considered noise for the two naive approaches.

In the first place, in figure 7.8, we offer a general comparison of the clustering results of the three approaches. Figure 7.8(a) shows the percentage of connections that have been included in clusters —with 5 or more connections— for the three methods and figure 7.8(b) shows the average length of those clusters. The values of the Tbc parameter (TBDC) and the inter-connection interval (ICIC) are shown in the bottom abscissa axis while the values of the download-wide timer (DWTC) are shown in the top one. The values of the percentage of clustered connections fall in the 70-90% interval depending on the method and the parameter values. This has to be interpreted taking into account that we are working in a

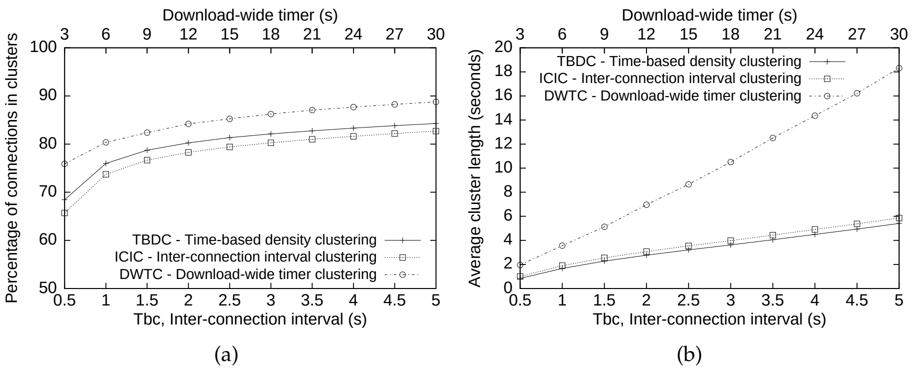


Figure 7.8: Comparing the three methods: (a) percentage of connections (in clusters of more than 5 connections) and (b) average cluster length.

very noisy environment where a sizable percentage of the connections to ports 80 and 443 belong to applications other than the web (such as application updates as mentioned previously).

As predicted, the ICIC and TBDC methods yield similar results as the latter is based on the former. However, the TBCD method is able to cluster more connections in clusters that are, on average, smaller which highlights its ability to more accurately select the time intervals with a high density of connections. On the other hand, the DWTC approach clusters the most connections of the three but does so in the longest clusters with cluster length being highly dependent on the timer value (as seen in figure 7.3(b)).

In any case, figure 7.8 does not allow us to check how well formed the resulting clusters are. We need to perform a validation process that checks if each cluster comprehends a full webpage download. This validation process is complex because we have to obtain some kind of ground truth that relates connections and webpage downloads. The most thorough way of tackling this problem is to monitorize the full HTTP conversation in every connection, learning which content is included in each webpage and which connections are responsible for the download of said content. However, in order to do this, we would need to capture and analyze the full payload of every connection: an almost impracticable task in a relatively big network. Moreover, this approach would not work with HTTPS traffic.

In our case, as we explained in section 7.2 we stored only the first 1000 bytes of the upstream application data of every connection in our captures with the intent of using this information to validate our methods. From these application bytes, in the vast majority of cases we are able to extract the following fields of the first HTTP request in each connection:

- Request line (*e.g.* GET /images/logo.png HTTP/1.1): which contains the requested resource.
- Host: the domain name of the server.
- Referer: identifies the address of the webpage that linked to the resource being requested. That is, for the connection that starts a webpage download, the referer is the address of the previous webpage from which a link to the current webpage was followed. But, in the case of the following connections which download content of the current webpage the referer is the address of the current webpage.

For HTTPS connections this is obviously impossible as user data is encrypted. In those cases we use DNS information captured at the same time to assign a host name to the server IP address of the HTTPS connection although we will not have information about the resources requested or the referer. In order to measure how well formed our clusters are, we use this information to define two concepts: consistency and completeness.

The **consistency** of a cluster is a measure of how many of the connections in the cluster are related. Ideally, we could find a criterion by which two connections would be related if they belonged to the same webpage download and unrelated if they did not. Using that ideal criterion, a consistency of a 100% would mean that all the connections inside a cluster are related and belong to the same webpage download. However, as we said previously from a network perspective it is sometimes very difficult or even impossible to decide if two connections belong to the same webpage download so we have settled for a representative but imperfect relationship criterion. We say that two connections are related if they are connected by the same 2nd level domain name. That is, the same 2nd level domain name appears in the host name, request line or referer

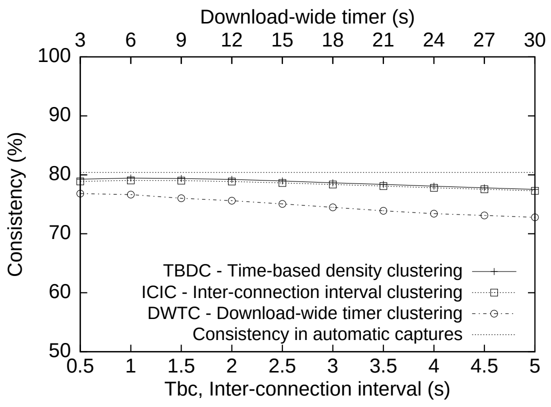


Figure 7.9: Consistency of the methods.

of the first HTTP request in each connection (or in the host name inferred via DNS for HTTPS connections). For registries were the 2nd level domain name is used for classification (for example in the United Kingdom: .co.uk, .org.uk, etc.) we consider the 3rd level domain name. Using this relationship criterion we define the consistency of a cluster as the percentage of connections related by the same 2nd (3rd) level domain name. For example, in a cluster that contains a Facebook webpage download, and where facebook.com is the most popular 2nd level domain, the consistency of the cluster would be the percentage of connections that contain "facebook.com" in the previously specified fields. Of course, even if a cluster only contains connections of a single webpage download, some of these connections might not be related in this way (*e.g.* HTTPS connections to third-party servers) and consistency will be less than 100%.

In order to see which values of consistency we could expect for "correct" clusters (those that contain a single webpage download), we applied this definition of consistency to the captures of automatic webpage downloads from chapter 6 and we obtained an average value of 80.4%. We offer this value for comparison in figure 7.9 where the average consistency of the clusters obtained with each method and parameter value is represented. As we can see, for values up to 2s of their respective parameters, the ICIC and the TBDC methods reach consistency averages very near to the expected value with the former performing around 0.5% better. The DWTC method yields significantly worse results.

However, internal consistency is not enough to validate our clustering methods. We also need to ascertain if the connections that belong to the same webpage download are divided in multiple clusters. We cannot use the same relationship criterion based on domain names as different webpages of the same websites will usually share the same domain names and these webpages are often opened consecutively. Because of this, we opt for a different approach based on the referers.

As we said previously, the referer of the requests for content in a webpage is the address of the webpage. As a consequence, in most cases, the most popular referer of a webpage download, that is, the one that appears in the most connections, will be the address of that webpage. In our case, we calculate the most popular referer for each cluster and we check how many of the following noise connections and of the connections of the following cluster share that referer. Taking all these connections into consideration, we define the **completeness** of a cluster as the percentage of connections of the most popular referer of the cluster that are inside said cluster. Thus defined, completeness only takes into account HTTP traffic as we do not have a referer available for HTTPS connections. Another limitation is that completeness will not usually reach 100% as the referer of the first connection of the next webpage download will match the previous one if the user has followed a link from the first webpage to the second. Also, some websites use gallery-like webpages in which the user cycles through different content (for our purposes, different webpages) but the referer remains the same.

Nevertheless, we believe that completeness still is a representative measurement because, even taking into account these issues, its average stabilizes for the three methods in values close to 85% when the parameters take values bigger than 2 seconds for DBTC and ICIC, or 12 seconds for DWTC (figure 7.10). The fact that it stabilizes is important because it shows that if we increase the parameter values (aggregating more connections to the clusters) most of these new connections will not belong to the same webpage download as consistency falls but completeness stays almost constant.

In brief, reviewing the different figures shown in this section, the ICIC and TBDC methods perform similarly and are generally better than the DWTC

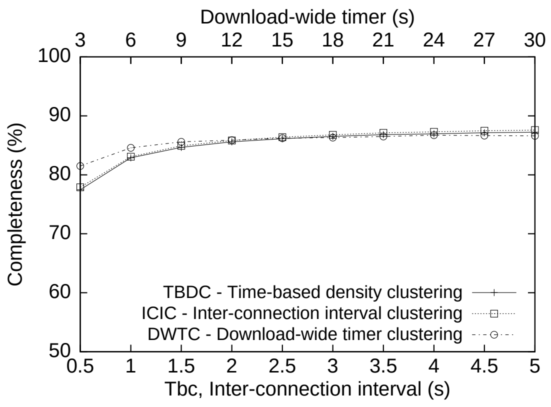


Figure 7.10: Completeness of the methods.

method. Between the two, the TBDC method yields better consistency results which we believe is the most critical and representative metric. Moreover, we should take into consideration that a lot of users may be more active than the university users with whom we are testing the methods. In those cases, the ability of the TBDC method to find the time intervals with a high density of connections will give it an additional advantage over the ICIC method which will be more prone to mix connections of different webpage downloads. Because of this, we believe that the TBDC method is the better option among the three.

As for the optimal value of the Tbc interval, studying figure 7.9 we see that consistency values are more or less stable up to 2 seconds; from that point, they start to decrease. On the other hand, in figure 7.10, completeness grows rapidly while we increase Tbc up to, again, 2 seconds and then changes much more slowly. Because of this, we believe that 2 seconds is a good value for Tbc in our network. In the following section we will use the full 9-day traffic trace to test the algorithm with Tbc=2s and check if consistency and completeness values remain in acceptable values.

7.7 Results and discussion

We have tested the time-based density method (Nc=5; Tbc=2s) for the rest of the days of the 9-day web traffic trace. The algorithm is very fast being able

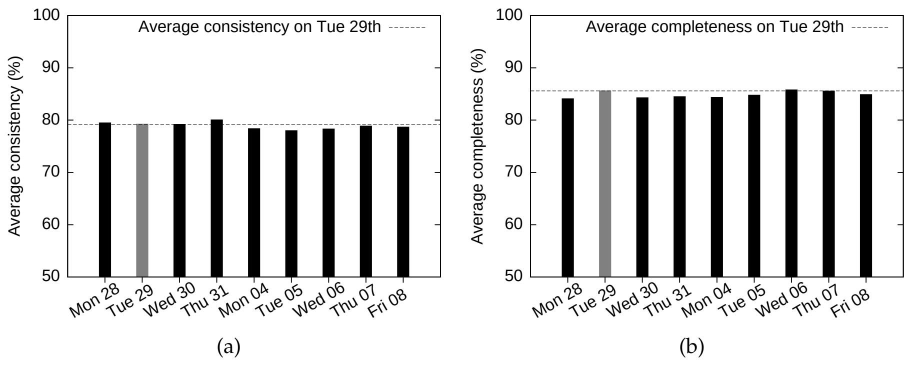
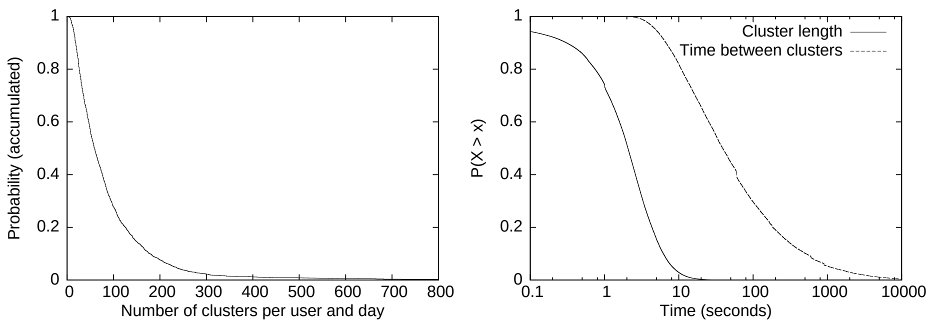


Figure 7.11: Testing the time-based density method with Tbc=2s: (a) daily average consistency and (b) daily average completeness.

to process the whole 9-day trace in less than a minute (using only one core in an Intel Xeon workstation). This makes it suitable for real-time operation. The average daily values for consistency and completeness are shown in figure 7.11. As we can see, both parameters remain close to the ones obtained for Tue 29th for which the algorithm was tuned (shown in the figure in a lighter shade). This shows that once tuned with a sample of the network traffic, the algorithm offers a stable performance.

In figure 7.12 we present some additional parameters that describe the clusters created by the algorithm. Figure 7.12(a) shows the CCDF of the daily number of clusters per user. With a median of 59 clusters and 10th and 90th percentiles at 18 and 179 clusters respectively, we believe that the number of clusters moves in a reasonable range if we consider that it should be closely related to the number of webpages visited daily by each user. For comparison, in [WOH+08] an study of 25 users found that the average number of webpages visited per user and day ranged from 25 to 284.

Although rare, some of our users have a very high daily number of clusters (P99 = 426). We have checked these cases manually and we have found that most of the clusters are related to non web applications using web ports. These are mainly antivirus and system updates and Dropbox clients (which use port 80). In one case, a user with a massive number of daily clusters had a malfunctioning web browser attempting to download favicons over and over. Although the



(a) (b)

Figure 7.12: Testing the time-based density method with Tbc=2s: CCDFs of (a) number of clusters per user and day (b) cluster length and time between consecutive cluster starts.

time-based density algorithm filters most of this traffic as noise, some of these applications do open enough close connections to be clustered together. Distinguishing between these and webpage downloads is one of our future lines of work.

On the other hand, figure 7.12(b) shows the distributions of cluster length in seconds and time between consecutive cluster starts. Values for both parameters are close to what we expected. With a median length of 2.1 seconds, our clusters are shorter than the experimental webpage downloads (see section 7.3.1 and chapter 6) but this is understandable because those were only landing pages (which usually are more complex) and because in real traffic some of the content may be already cached in the host as users usually revisit the same webpages. Median time between clusters is 39 seconds which seems a reasonable webpage dwell time (again, see section 7.3.1).

Another interesting consideration arises if we consider extreme percentiles for both distributions. The 90th percentile of cluster length is 6.2 seconds while the 10th percentile of the time between consecutive cluster starts is 6.7 seconds. This shows that even the longest clusters are shorter than the time between the closest ones which makes the possibility of overlapping two different webpage downloads in the same cluster a remote one.

7.8 Conclusions

In this chapter we have discussed methods for grouping connections that belong to the same webpage download. We have centered our study in connection start times as we have seen that it is the most representative connection-level parameter for our purposes. We have tried out simple approaches based on total download length and the time interval between connections. However, in the end, we have chosen a clustering algorithm based on DBSCAN which is specially suited to our noisy environment.

We have tuned and tested said method using both automatically collected samples of webpage downloads and real web traffic from the Internet link of our university. We have defined metrics (based on HTTP header data and DNS information) that describe the goodness of the results obtained by our method and we have also compared it with the simpler approaches. We have been able to obtain clusters of average internal consistency of around 80% (very close to the achievable maximum) and average completeness of near 85%. These results have been shown to be stable over time once the method is tuned with a sample of traffic from our network.

The proposed method is simple and fast which makes it suitable for real-time operation. This gives it multiple applications such as accounting, filtering and/or prioritizing certain services, or characterization of user browsing habits.

CHAPTER 8

Conclusions

Over the last years, websites have evolved rapidly incorporating new content types and becoming more and more complex and dynamic. Users today are able to access a wide variety of different services through their web browsers which are now present not only in computers but in almost every network-enabled device. As a consequence, web traffic has become increasingly complex and, from a network perspective, it can be difficult to ascertain which webpages or websites are being visited by a user, let alone which part of the user's traffic each of them is responsible for.

Although there is an extensive literature on the new characteristics of web traffic and many studies have researched ways of distinguishing it from the traffic of other applications, few works have focused on identifying web browsing interactions through traffic analysis. Even fewer have attempted to tackle this problem from a connection perspective, that is, without relying on application-level data. This has been the focus of this thesis and in this section we present the conclusions to our work, our main contributions and some future lines of research.

8.1 Conclusions

The main objective of this thesis has been to develop methods able to group connections that belong to the same webpage download or website session. It is easy to underestimate the difficulty of this task but the truth is that, without relying on application-level data, the information that can be used for this purpose is very limited and the complexity of modern web traffic makes it a complicated endeavor. Modern web users browse through websites offering a multitude of different services and often, because of tab-based browsers, the sessions of those websites overlap. The webpages that form those websites may also have very different characteristics and comprise various content types hosted in a multitude of servers. Moreover, other applications using web ports often run along the web browser in the users' computers.

As a consequence, grouping connections related to the same web interactions is a far from trivial task. In fact, previous studies have found difficulties in resolving this problem even when using deep packet inspection to monitor HTTP data. In our case we have worked with parameters calculated at the connection level such as the ones that can be found in NetFlow records. Using these kind of parameters is very interesting because they are widely available, processing them is computationally inexpensive and they avoid monitoring application data which may be encrypted and, besides, is legally problematic. However, as we have said, they offer a much more limited information to work with.

Throughout the thesis we have characterized web traffic from a connection-level perspective. We have seen that two parameters are specially interesting in order to find related connections: time and server IP addresses. Time is interesting because related connections tend to happen closer in time and because the times elapsed between connections triggered automatically are much shorter than those between connections triggered by user actions. Server IP addresses can help in finding related connections because each server hosting a webpage often provides more than one resource and because the same servers usually provide content for the different webpages of a website. We have used this to our advantage to identify server IP addresses closely related to websites. However, we have found that the content of webpages is more and more dispersed in multiple servers (some belonging to the website, some to CDNs and some to third-parties). As a consequence, the number of servers closely related to a website is usually small and using IP addresses to find related connections must be done carefully.

A related effect of the complexity of the modern web is that we have found very difficult to validate the results of the methods we have presented in the thesis. On one hand, it is often complicated to asses if a connection is related to a specific webpage download or website session even when studying it individually (especially for encrypted connections). Therefore it is complex to design automatic labeling schemes that would provide the ground truth necessary to validate our results. This has forced us to develop different validation schemes depending on the nature of the results of each of the proposed methods. For this purpose we have mainly used DNS traffic captures and HTTP header information (referer field). The absence of ground truth has also prevented us from applying algorithms that require a pre-labeled data set for training (as it is the case with machine learning techniques). On the other hand, the lack of similar works in the scientific literature has made impossible to compare the performance of our methods with those proposed by other authors, further hindering the validation process.

8.2 Main contributions

In order to present the main contributions of this thesis we look back to the objectives we set in section 1.3 and to the conclusions of the different chapters of this document.

Characterizing web traffic focusing on connection-level parameters: Throughout this thesis we have characterized the traffic of isolated website sessions (chapter 4) and webpage downloads (chapter 6). We have also thoroughly studied the traffic of web users in our university network (chapters 5 and 7).

In order to capture website sessions we prepared an easily-distributable capturing and labeling system and we requested the help of voluntary test users.

Although the system did not allow for a natural web browsing experience it was the only way of capturing complete sessions given the difficulty of labeling them manually on a trace of network traffic. We characterized different aspects of the captured website sessions showing the differences between the sessions of different services and focusing on parameters that could help in relating connections of the same session. This characterization appears in the following publication:

- Luis Miguel Torres, Eduardo Magaña, Mikel Izal and Daniel Morato. Identifying sessions to websites as an aggregation of related flows. In *proceedings of the 15th International Telecommunications Network Strategy and Planning Symposium (NETWORKS)*, pages 1-6, 2012.

We have also collected an extensive data set of popular webpage downloads which we captured by accessing them automatically. Chapter 6 offered a general characterization of these downloads before focusing on time and server-based parameters. In the end we selected ten metrics that describe a normal webpage download. Our data set of webpage downloads is presented in the following publication:

- Luis Miguel Torres, Eduardo Magaña, Mikel Izal, and Daniel Morato. Characterizing webpage load from the perspective of TCP connections. In *proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 977–984, 2014.

Studying the connection-level parameters that can be used to find related web connections and using them to develop methods that classify the connections according to the webpage download or website session to which they belong: The objective of chapter 4 was to develop a method able to cluster the connections of the same website sessions. We found that flow start timestamps, end timestamps and server IP addresses were the most interesting parameters in order to cluster the connections. Nevertheless, they all had their drawbacks which the proposed method attempted to minimize. Experimental results showed that connections were grouped in a relatively small number of big clusters for each session and that recall and precision were generally good for the services tested. However, we found difficult to properly validate our results given the lack of ground truth in the form of labeled web traffic traces. The method for clustering the connections of the same website sessions was presented in:

- Luis Miguel Torres, Eduardo Magaña, Mikel Izal and Daniel Morato. Identifying sessions to websites as an aggregation of related flows. In *proceedings of the 15th International Telecommunications Network Strategy and Planning Symposium (NETWORKS)*, pages 1-6, 2012.

On the other hand, in chapter 7 we focused on finding individual webpage downloads in user traffic. This problem proved to be more manageable as there is less user-induced variability than with website sessions. We centered our study in connection start times as we saw that it was the most representative connection-level parameter for our purposes. We tried out simple approaches based on total download length and the time interval between connections. However, in the end, we chose a clustering algorithm based on DBSCAN which was specially suited to the noisy environment.

The method was tuned and tested with automatically collected samples of webpage downloads and real web traffic from the Internet link of our university. In order to validate the method we defined metrics that described the goodness of the results obtained and we compared it with the simpler approaches. We were able to obtain clusters of average internal consistency of around 80% (very close to the achievable maximum) and average completeness of near 85%. These results were shown to be stable over time once the method is tuned with a sample of traffic from our network. The results collected in chapter 7 appear in:

- Luis Miguel Torres, Eduardo Magaña, Mikel Izal, and Daniel Morato. Time based clustering methods for identifying webpage downloads in TCP flow records. Sent to *ACM Transactions on the web.*

Both methods were simple and fast which makes them suitable for real-time operation. This gave them multiple applications such as accounting, filtering and/or prioritizing certain services or characterization of user browsing habits.

Studying the server IP addresses accessed during sessions of different websites to identify servers closely related to them: In chapter 5 we presented a method able to identify server IP addresses related to a predefined list of websites in a traffic trace. The proposed system labeled individual IP addresses based on the number of times connections to them appeared in sessions of a particular website against the total number of apparitions in the trace. It also labeled IP subnetworks if various IP addresses that belonged to them appeared repeatedly in sessions of a website.

The system was tested with traffic traces from our university network and was able to identify an average of more than 30 IP addresses per website. The results were validated manually obtaining a lower bound for the accuracy of the system at around 70% (the difficulty in the manual validation did not allow us to offer a more precise estimation of the accuracy of the system).

Afterwards, we introduced some modifications to the system and a new validation procedure. With this new system, we identified an average of more than 20 IP addresses per website. We validated these results by considering the assignation of IP addresses that are related to the websites by DNS information obtaining precision values over 90%.

The work presented in chapter 5 gave way to the following publications:

- Luis Miguel Torres, Eduardo Magaña, Mikel Izal and Daniel Morato. Strategies for automatic labelling of web traffic traces. In *proceedings of the 37th IEEE Conference on Local Computer Networks (LCN)*, pages 196–199, 2012.

- Luis Miguel Torres, Eduardo Magaña, Mikel Izal, and Daniel Morato. A popularity-aware method for discovering server IP addresses related to websites. In *proceedings of the 5th Global Information Infrastructure Symposium (GIIS)*, pages 1–8, 2013.

A system able to identify IP addresses related to websites can be used for multiple purposes as it allows gathering information about the browsing behavior of the users and about the websites themselves.

8.3 Future work

The results collected in this thesis suggest new research ideas for future work in web traffic characterization and classification. We present some of them in this section.

An immediate and interesting work would be to further compare the results obtained in chapters 6 and 7. Although the method presented in the later depends heavily on the characterization performed in the former, it makes sense to test if the reconstructed webpage downloads have characteristics similar to the ones described for the data set of automatic webpage downloads.

It would also be interesting to extend the webpage download detection method presented in chapter 7 so that it is able to group webpages of the same website session. Server IP addresses may prove useful in this task although, as we have previously said, they have to be used carefully. If we managed to do this we could compare the performance of the new extended method and the one presented in chapter 4.

The webpage downloads and website sessions reconstructed in chapters 4 and 7 can be characterized in order to gain useful knowledge about the behavior of web users (how much they access the web, how many different websites they visit, etc.) and about the traffic generated by different websites. The method for identifying IP addresses related to specific websites presented in chapter 5 can also be interesting for characterization purposes as it allows detecting website sessions in the traffic of the users.

In section 2.2 we collected different proposals for classifying Internet traffic. Some of them used connection level metrics for this purpose. There is a more limited literature in distinguishing different services (*e.g.* webmail, video streaming, maps, etc.) in web traffic. The few authors that have attempted this have relied on packet statistics because the information offered in a flow record is very limited if we consider the connections individually. However, using our reconstructed sessions and webpage downloads, the characteristics of multiple related connections may be distinctive enough to classify them according to the service that originated them. We believe that this can be one of the most interesting applications of the work carried out in this thesis.

Bibliography

[ALC+11] R. Archibald, Y. Liu, C. Corbett, and D. Ghosal. "Disambiguating HTTP: classifying web applications". In: *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*. IEEE. 2011, pp. 1808–1813. DOI: `10.1109/IWCMC.2011.5982809`.

[Alexa] *Alexa - Actionable Analytics for the web*. URL: `http://www.alexa.com/` (Retrieved 09/2014).

[AMG07] T. Auld, A. Moore, and S. Gull. "Bayesian Neural Networks for Internet Traffic Classification". In: *Neural Networks, IEEE Transactions on* 18.1 (2007), pp. 223 –239. DOI: `10.1109/TNN.2006.883010`.

[ARGUS] *ARGUS: Audit Record Generation and Usage System*. URL: `http://www.qosient.com/argus/` (Retrieved 09/2014).

[AW96] M. F. Arlitt and C. L. Williamson. "Web Server Workload Characterization: The Search for Invariants". In: *ACM SIGMETRICS Performance Evaluation Review* 24.1 (1996), pp. 126–137. DOI: `10.1145/233008.233034`.

[BBJ+15] A. Bergkvist, D. Burnett, C. Jennings, and A. Narayanan. *WebRTC 1.0: Real-time Communication Between Browsers*. Editor's Draft. World Wide Web Consortium, 2015. URL: `http://www.w3.org/TR/webrtc/` (Retrieved 03/2015).

[BL91] T. Berners-Lee. *HTTP V0.9*. World Wide Web Consortium, 1991. URL: `http://www.w3.org/Protocols/HTTP/AsImplemented.html` (Retrieved 12/2014).

[Bla12] M. Blanchet. *Implications of Blocking Outgoing Ports Except Ports 80 and 443*. Internet-Draft. Internet Engineering Task Force (IETF), 2012. URL: `http://tools.ietf.org/id/draft-blanchet-iab-internetoverport443-01.html` (Retrieved 03/2015).

[BLC90] T. Berners-Lee and R. Cailliau. *WorldWideWeb: Proposal for a HyperTexts Project*. World Wide Web Consortium, 1990. URL: `http://www.w3.org/Proposal.html` (Retrieved 12/2014).

[BLC93] T. Berners-Lee and D. Connolly. *HyperText Markup Language HTML*. Internet-Draft. Internet Engineering Task Force (IETF), 1993. URL: `http://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt` (Retrieved 12/2014).

[BLFF96] T. Berners-Lee, R. Fielding, and H. Frystyk. *Hypertext Transfer Protocol – HTTP/1.0*. RFC 1945. Internet Engineering Task Force (IETF), 1996. URL: `http://tools.ietf.org/html/rfc1945` (Retrieved 01/2015).

[BMM+07] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli. "Revealing Skype traffic: when randomness plays with you". In: *SIGCOMM Computer Communications Review* 37.4 (2007), pp. 37–48. DOI: `10.1145/1282427.1282386`.

[BMM+09] A. Bianco, G. Mardente, M. Mellia, M. Munafo, and L. Muscariello. "Web User-Session Inference by Means of Clustering Techniques". In: *Networking, IEEE/ACM Transactions on* 17.2 (2009), pp. 405 –416. DOI: `10.1109/TNET.2008.927009`.

[BMM+12] I. N. Bermudez, M. Mellia, M. M. Munafo, R. Keralapura, and A. Nucci. "DNS to the rescue: discerning content and services in a tangled web". In: *ACM conference on Internet Measurement*. IMC '12. 2012, pp. 413–426. DOI: `10.1145/2398776.2398819`.

[BMS11] M. Butkiewicz, H. V. Madhyastha, and V. Sekar. "Understanding website complexity: measurements, metrics, and implications". In: *ACM conference on Internet Measurement*. IMC '11. 2011, pp. 313–328. DOI: 10.1145/2068816.2068846.

[BPT14] M. Belshe, R. Peon, and M. Thomson. *Hypertext Transfer Protocol version 2*. Internet-Draft. Internet Engineering Task Force (IETF), 2014. URL: https://datatracker.ietf.org/doc/draft-ietf-httpbis-http2 (Retrieved 03/2015).

[BRV+06] L. Bent, M. Rabinovich, G. M. Voelker, and Z. Xiao. "Characterization of a large web site population with implications for content delivery". In: *World Wide Web* 9.4 (2006), pp. 505–536. DOI: 10.1007/s11280-006-0224-x.

[BT07] L. Bernaille and R. Teixeira. "Early recognition of encrypted applications". In: *Passive and Active Network Measurement*. Lecture Notes in Computer Science. Springer, 2007, pp. 165–175. DOI: 10.1007/978-3-540-71617-4_17.

[BTA+06] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian. "Traffic classification on the fly". In: *SIGCOMM Computer Communications Review* 36.2 (2006), pp. 23–26. DOI: 10.1145/1129582.1129589.

[BTS06] L. Bernaille, R. Teixeira, and K. Salamatian. "Early application identification". In: *ACM CoNEXT conference*. CoNEXT '06. New York, NY, USA, 2006, pp. 1–12. DOI: 10.1145/1368436.1368445.

[Cap11] R. Capra. "HCI Browser: A Tool for Administration and Data Collection for Studies of Web Search Behaviors". In: *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*. Ed. by A. Marcus. Vol. 6770. Lecture Notes in Computer Science. Springer, 2011, pp. 259–268. DOI: 10.1007/978-3-642-21708-1_30.

[CB97] M. E. Crovella and A. Bestavros. "Self-similarity in World Wide Web Traffic: Evidence and Possible Causes". In: *IEEE/ACM Transactions on Networking* 5.6 (1997), pp. 835–846. DOI: 10.1109/90.650143.

[CCW+07] S. E. Coull, M. P. Collins, C. V. Wright, F. Monrose, and M. K. Reiter. "On Web Browsing Privacy in Anonymized NetFlows". In: *USENIX Security Symposium*. SS'07. 2007, 23:1–23:14. URL: http://dl.acm.org/citation.cfm?id=1362903.1362926.

[CCY00] V. Cardellini, M. Colajanni, and P. S. Yu. "Geographic Load Balancing for Scalable Distributed Web Systems". In: *International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. MASCOTS '00. IEEE Computer Society, 2000, pp. 20–. DOI: 10.1109/MASCOT.2000.876425.

[CET+11] M. Cotton, L. Eggert, J. Touch, M. Westerlund, and S. Cheshire. *Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry*. RFC 6335. Internet Engineering Task Force (IETF), 2011. URL: https://tools.ietf.org/html/rfc6335 (Retrieved 01/2015).

[CG99] R. G. Congalton and K. Green. *Assessing the accuracy of remotely sensed data: Principles and practices*. Lewis Publishers, Boca Raton, 1999.

[Cha10] J. Charzinski. "Traffic Properties, Client Side Cachability and CDN Usage of Popular Web Sites". In: *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*. Vol. 5987. Lecture Notes in Computer Science. Springer, 2010, pp. 136–150. DOI: 10.1007/978-3-642-12104-3_12.

[Cis12] *Introduction to Cisco IOS NetFlow*. White Paper. Cisco, 2012. URL: http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.pdf.

[CJO+01] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith. “Tuning RED for Web Traffic”. In: *IEEE/ACM Transactions on Networking* 9.3 (2001), pp. 249–264. DOI: 10.1109/90.929849.

[CKY+04] T. Choi, C. Kim, S. Yoon, J. Park, B. Lee, H. Kim, H. Chung, and T. Jeong. “Content-aware Internet application traffic measurement and analysis”. In: *IEEE/IFIP Network Operations and Management Symposium*. Vol. 1. NOMS ’04. 2004, pp. 511–524. DOI: 10.1109/NOMS.2004.1317737.

[CLW+01] M. Claypool, P. Le, M. Wased, and D. Brown. “Implicit Interest Indicators”. In: *International Conference on Intelligent User Interfaces*. IUI ’01. ACM, 2001, pp. 33–40. DOI: 10.1145/359784.359836.

[CM06] F. Constantinou and P. Mavrommatis. “Identifying Known and Unknown Peer-to-Peer Traffic”. In: *IEEE International Symposium on Network Computing and Applications*. NCA ’06. 2006, pp. 93–102. DOI: 10.1109/NCA.2006.34.

[CoralReef] *CoralReef Software Suite*. URL: http://www.caida.org/tools/measurement/coralreef/ (Retrieved 03/2014).

[CP95] L. D. Catledge and J. E. Pitkow. “Characterizing Browsing Strategies in the World-Wide Web”. In: *Computer Networks and ISDN Systems* 27.6 (1995), pp. 1065–1073. DOI: 10.1016/0169-7552(95)00043-7.

[DT09] R. Das and I. Turkoglu. “Creating meaningful data from web logs for improving the impressiveness of a website by using path analysis method”. In: *Expert Systems with Applications* 36.3 (2009), pp. 6635–6644. DOI: 10.1016/j.eswa.2008.08.067.

[EKS+96] M. Ester, H. Kriegel, J. Sander, and X. Xu. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: *International Conference on Knowledge Discovery and Data Mining*. KDD ’96. AAAI Press, 1996, pp. 226–231. URL: https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf.

[EMA+07] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson. "Identifying and Discriminating Between Web and Peer-to-peer Traffic in the Network Core". In: *International Conference on World Wide Web*. WWW '07. ACM, 2007, pp. 883–892. DOI: `10.1145/1242572.1242692`.

[FGM+97] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2068. Internet Engineering Task Force (IETF), 1997. URL: `https://www.ietf.org/rfc/rfc2068.txt` (Retrieved 01/2015).

[FL05] F. M. Facca and P. L. Lanzi. "Mining Interesting Knowledge from Weblogs: A Survey". In: *Data & Knowledge Engineering* 53.3 (2005), pp. 225–241. DOI: `10.1016/j.datak.2004.08.001`.

[FM11] I. Fette and A. Melnikov. *The WebSocket Protocol*. RFC 6455. Internet Engineering Task Force (IETF), 2011. URL: `https://tools.ietf.org/html/rfc6455` (Retrieved 01/2015).

[FMN+03] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. "A Large-scale Study of the Evolution of Web Pages". In: *International Conference on World Wide Web*. WWW '03. ACM, 2003, pp. 669–678. DOI: `10.1145/775152.775246`.

[GAL+07] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. "Youtube Traffic Characterization: A View from the Edge". In: *ACM Conference on Internet Measurement*. IMC '07. San Diego, California, USA, 2007, pp. 15–28. ISBN: 978-1-59593-908-1. DOI: `10.1145/1298306.1298310`.

[Hel11] G. Held. *A Guide to Content Delivery Networks*. second. CRC Press, 2011. ISBN: 9781439835883.

[Her09] E. Hernandez. "War of the Mobile Browsers". In: *Pervasive Computing* 8.1 (2009), pp. 82–85. DOI: `10.1109/MPRV.2009.21`.

[HSS+05] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. "ACAS: Automated Construction of Application Signatures". In: *ACM SIG-*

COMM Workshop on Mining Network Data. MineNet '05. 2005, pp. 197–202. DOI: 10.1145/1080173.1080183.

[HTML5] *HTML5: A vocabulary and associated APIs for HTML and XHTML*. Recomendation. World Wide Web Consortium, 2014. URL: http://www.w3.org/TR/html5/.

[HWL+08] C. Huang, A. Wang, J. Li, and K. W. Ross. "Measuring and Evaluating Large-scale CDNs". In: *ACM SIGCOMM Conference on Internet Measurement*. IMC '08. 2008, pp. 15–29. URL: http://dl.acm.org/citation.cfm?id=1452520.1455517.

[IP11] S. Ihm and V. S. Pai. "Towards Understanding Modern Web Traffic". In: *ACM SIGCOMM Conference on Internet Measurement*. IMC '11. 2011, pp. 295–312. DOI: 10.1145/2068816.2068845.

[Jen67] G. Jenks. "The Data Model Concept in Statistical Mapping". In: *International Yearbook of Cartography*. 7. 1967, pp. 186–190.

[KA02] K.-S. Kim and B. Allen. "Cognitive and task influences on Web searching behavior". In: *Journal of the American Society for Information Science and Technology* 53.2 (2002), pp. 109–119. DOI: 10.1002/asi.10014.

[KAA06] D. Koukis, S. Antonatos, and K. G. Anagnostakis. "On the privacy risks of publishing anonymized IP network traces". In: *Communications and Multimedia Security*. Springer, 2006, pp. 22–32. DOI: 10.1007/11909033_3.

[KB04] D. Kelly and N. J. Belkin. "Display Time As Implicit Feedback: Understanding Task Effects". In: *ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '04. 2004, pp. 377–384. DOI: 10.1145/1008992.1009057.

[KBF+04] T. Karagiannis, A. Broido, M. Faloutsos, and K. claffy. "Transport Layer Identification of P2P Traffic". In: *ACM SIGCOMM Conference on Internet Measurement*. IMC '04. 2004, pp. 121–134. DOI: 10.1145/1028788.1028804.

[KHI+08] M. Kellar, K. Hawkey, K. M. Inkpen, and C. Watters. "Challenges of capturing natural Web-based user behaviors". In: *International Journal of Human–Computer Interaction* 24.4 (2008), pp. 385–409. DOI: 10.1080/10447310801973739.

[KHM+13] H. Khandelwal, F. Hao, S. Mukherjee, R. R. Kompella, and T. Lakshman. "CobWeb: In-network cobbling of web traffic". In: *IFIP Networking Conference*. IEEE. 2013, pp. 1–9. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6663509&isnumber=6663488.

[KHW+14] Y. Kim, A. Hassan, R. W. White, and I. Zitouni. "Modeling Dwell Time to Predict Click-level Satisfaction". In: *ACM International Conference on Web Search and Data Mining*. WSDM '14. 2014, pp. 193–202. DOI: 10.1145/2556195.2556220.

[KM97] D. Kristol and L. Montulli. *HTTP State Management Mechanism*. RFC 2109. Internet Engineering Task Force (IETF), 1997. URL: https://www.ietf.org/rfc/rfc2109.txt (Retrieved 01/2015).

[KPF05] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. "BLINC: Multilevel Traffic Classification in the Dark". In: *SIGCOMM Computer Communications Review* 35.4 (2005), pp. 229–240. DOI: 10.1145/1090191.1080119.

[LCL14] P.-C. Lin, S.-Y. Chen, and C.-H. Lin. "Towards fine-grained traffic classification for web applications". In: *Australasian Telecommunication Networks and Applications Conference (ATNAC)*. IEEE. 2014, pp. 28–33. DOI: 10.1109/ATNAC.2014.7020869.

[LK07] H. Liu and V. Kešelj. "Combined Mining of Web Server Logs and Web Contents for Classifying User Navigation Patterns and Predicting Users' Future Requests". In: *Data & Knowledge Engineering* 61.2 (2007), pp. 304–330. DOI: 10.1016/j.datak.2006.06.001.

[LYG07] Z. Li, R. Yuan, and X. Guan. “Accurate Classification of the Internet Traffic Based on the SVM Method”. In: *IEEE International Conference on Communication*. ICC ’07. 2007, pp. 1373–1378. DOI: 10.1109/ICC.2007.231.

[MFWRG+12] G. Maciá-Fernández, Y. Wang, R. A. Rodríguez-Gómez, and A. Kuzmanovic. “Extracting user web browsing patterns from non-content network traces: The online advertising case study”. In: *Computer Networks* 56.2 (2012), pp. 598 –614. DOI: http://dx.doi.org/10.1016/j.comnet.2011.10.012.

[MHL+04] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. “Flow Clustering Using Machine Learning Techniques”. In: *Passive and Active Network Measurement*. Vol. 3015. Lecture Notes in Computer Science. Springer, 2004, pp. 205–214. DOI: 10.1007/978-3-540-24668-8_21.

[MM12] J. R. Mayer and J. C. Mitchell. “Third-Party Web Tracking: Policy and Technology”. In: *Symposium on Security and Privacy*. SP ’12. IEEE Computer Society, 2012, pp. 413–427. DOI: 10.1109/SP.2012.47.

[MMG+07] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. “Measurement and Analysis of Online Social Networks”. In: *ACM SIGCOMM Conference on Internet Measurement*. IMC ’07. 2007, pp. 29–42. DOI: 10.1145/1298306.1298311.

[MZ05] A. W. Moore and D. Zuev. “Internet traffic classification using bayesian analysis techniques”. In: *SIGMETRICS Performance Evaluation Review* 33.1 (2005), pp. 50–60. DOI: 10.1145/1064212.1064220.

[MZC05] A. Moore, D. Zuev, and M. Crogan. *Discriminators for use in flow-based classification*. Queen Mary and Westfield College, Department of Computer Science, 2005. URL: http://www.cl.cam.ac.uk/~awm22/publications/RR-05-13.pdf.

[NA08] T. T. T. Nguyen and G. Armitage. "A survey of techniques for internet traffic classification using machine learning". In: *Communications Surveys & Tutorials, IEEE* 10.4 (2008), pp. 56–76. DOI: `10.1109/SURV.2008.080406`.

[NGBS+97] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. W. Lie, and C. Lilley. "Network Performance Effects of HTTP/1.1, CSS1, and PNG". In: *SIGCOMM Computer Communication Review* 27.4 (1997), pp. 155–166. DOI: `10.1145/263109.263157`.

[NJA13] B. Newton, K. Jeffay, and J. Aikat. "The Continued Evolution of Web Traffic". In: *International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems*. MASCOTS '13. IEEE, 2013, pp. 80–89. DOI: `10.1109/MASCOTS.2013.16`.

[O'r07] T. O'reilly. "What is Web 2.0: Design patterns and business models for the next generation of software". In: *Communications and Strategies* 65.1 (2007), pp. 17–37. URL: `http://ssrn.com/abstract=1008839`.

[OWH04] H. Obendorf, H. Weinreich, and T. Hass. "Automatic Support for Web User Studies with SCONE and TEA". In: *Extended Abstracts on Human Factors in Computing Systems*. CHI EA '04. 2004, pp. 1135–1138. ISBN: 1-58113-703-6. DOI: `10.1145/985921.986007`.

[Par62] E. Parzen. "On Estimation of a Probability Density Function and Mode". In: *The Annals of Mathematical Statistics* 33.3 (1962), pp. 1065–1076. DOI: `10.1214/aoms/1177704472`.

[pcap] *Tcpdump & Libpcap*. URL: `http://www.tcpdump.org/` (Retrieved 09/2014).

[PGD+07] M. Perenyi, A. Gefferth, T. D. Dang, and S. Molnar. "Skype Traffic Identification". In: *Global Telecommunications Conference*. GLOBECOM '07. IEEE, 2007, pp. 399 –404. DOI: `10.1109/GLOCOM.2007.81`.

[PHMA+00] J. Pei, J. Han, B. Mortazavi-Asl, and H. Zhu. "Mining access patterns efficiently from web logs". In: *Knowledge Discovery and Data Mining. Current Issues and New Applications*. Springer, 2000, pp. 396–407. DOI: 10.1007/3-540-45571-X_47.

[Pil05] M. Pilgrim. *Greasemonkey Hacks: Tips & Tools for Remixing the Web with Firefox*. O'Reilly Media, 2005. ISBN: 0596101651.

[PR94] J. Pitkow and M. M. Recker. "Results From The First World-Wide Web User Survey". In: *Computer Networks and ISDN Systems* 27.2 (1994), pp. 243–254. DOI: 10.1016/0169-7552(94)90138-4.

[QUIC] *QUIC, a multiplexed stream transport over UDP*. URL: http://www.chromium.org/quic (Retrieved 01/13/2015).

[QZC+04] J. Quittek, T. Zseby, B. Claise, and S. Zander. *Requirements for IP Flow Information Export (IPFIX)*. RFC 3917. Internet Engineering Task Force (IETF), 2004. URL: http://www.ietf.org/rfc/rfc3917.txt (Retrieved 01/2015).

[RADb] *Merit RADb. Routing Assets Database*. URL: http://www.radb.net/ (Retrieved 09/2014).

[Res00] E. Rescorla. *HTTP over TLS*. RFC 2818. Internet Engineering Task Force (IETF), 2000. URL: http://tools.ietf.org/html/rfc2818 (Retrieved 01/2015).

[revIP] *revIP.info (now Binary monkey)*. URL: http://revip.info (Retrieved 04/2012).

[Robtex] *Robtex Swiss Army Knife Internet Tool*. URL: http://www.robtex.com (Retrieved 04/2012).

[RSS+04] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. "Class-of-service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification". In: *ACM SIGCOMM Conference on Internet Measurement*. IMC '04. 2004, pp. 135–148. DOI: 10.1145/1028788.1028805.

[RV09] F. Raineri and G. Verticale. “Early Internet Application Identification with Machine Learning Techniques”. In: *International Conference on Evolving Internet*. IEEE Computer Society, 2009, pp. 60–64. DOI: 10.1109/INTERNET.2009.16.

[SAA+08] F. Schneider, S. Agarwal, T. Alpcan, and A. Feldmann. “The New Web: Characterizing AJAX Traffic”. In: *Passive and Active Network Measurement*. Vol. 4979. Lecture Notes in Computer Science. Springer, 2008, pp. 31–40. DOI: 10.1007/978-3-540-79232-1_4.

[SAM+12] F. Schneider, B. Ager, G. Maier, A. Feldmann, and S. Uhlig. “Pitfalls in HTTP Traffic Measurements and Analysis”. In: *International Conference on Passive and Active Measurement*. PAM’12. Springer, 2012, pp. 242–251. DOI: 10.1007/978-3-642-28537-0_24.

[Sand14] *Global Internet Phenomena Report 1H 2014*. Tech Report. Sandvine, 2014. URL: https://www.sandvine.com/downloads/general/global-internet-phenomena/2014/1h-2014-global-internet-phenomena-report.pdf (Retrieved 12/2014).

[SCBRSP12] J. Sanjuàs-Cuxart, P. Barlet-Ros, and J. Solé-Pareta. “Measurement Based Analysis of One-Click File Hosting Services”. In: *Journal of Network and Systems Management* 20.2 (2012), pp. 276–301. DOI: 10.1007/s10922-011-9202-4.

[SCJ+01] F. D. Smith, F. H. Campos, K. Jeffay, and D. Ott. “What TCP/IP protocol headers can tell us about the web”. In: *ACM SIGMETRICS Performance Evaluation Review*. Vol. 29. 1. 2001, pp. 245–256. DOI: 10.1145/384268.378789.

[SCounter] *StatCounter Global Stats*. URL: http://gs.statcounter.com/ (Retrieved 03/2015).

[Sed95] J. Sedayao. “World Wide Web network traffic patterns”. In: *Compcon ’95 Technologies for the Information Superhighway, Digest of Papers*. 1995, pp. 8–12. DOI: 10.1109/CMPCON.1995.512356.

[SFK+09] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger. “Understanding Online Social Network Usage from a Network Perspective”. In: *ACM SIGCOMM Conference on Internet Measurement*. IMC ’09. 2009, pp. 35–48. DOI: `10.1145/1644893.1644899`.

[SMS+10] D. Schatzmann, W. Mühlbauer, T. Spyropoulos, and X. Dimitropoulos. “Digging into HTTPS: Flow-based Classification of Webmail Traffic”. In: *ACM SIGCOMM Conference on Internet Measurement*. IMC ’10. 2010, pp. 322–327. DOI: `10.1145/1879141.1879184`.

[Tech12] *Google Biz Chief: Over 10M Websites Now Using Google Analytics*. URL: `http://techcrunch.com/2012/04/12/google-analytics-officially-at-10m/` (Retrieved 01/2015).

[Tho14] P. Thomas. “Using Interaction Data to Explain Difficulty Navigating Online”. In: *ACM Transactions on the Web* 8.4 (2014), 24:1–24:41. DOI: `10.1145/2656343`.

[tim] *Trustworthy Internet Movement*. URL: `https://www.trustworthyinternet.org` (Retrieved 01/2015).

[VG08] G. Verticale and P. Giacomazzi. “Performance evaluation of a machine learning algorithm for early application identification”. In: *International Multiconference on Computer Science and Information Technology*. IMCSIT ’08. 2008, pp. 845 –849. DOI: `10.1109/IMCSIT.2008.4747340`.

[VSG+06] M. Viermetz, C. Stolz, V. Gedov, and M. Skubacz. “Relevance and Impact of Tabbed Browsing Behavior on Web Usage Mining”. In: *IEEE/WIC/ACM International Conference on Web Intelligence*. 2006, pp. 262–269. DOI: `10.1109/WI.2006.147`.

[W3Counter] *w3counter.com*. URL: `http://www.w3counter.com/` (Retrieved 03/2015).

[WCC+09] C.-C. Wu, K.-T. Chen, Y.-C. Chang, and C.-L. Lei. "Peer-to-peer application recognition based on signaling activity". In: *International Conference on Communications*. ICC'09. IEEE Press, 2009, pp. 2174–2178. DOI: `10.1109/ICC.2009.5199305`.

[Wired14] *Encrypted Web Traffic More Than Doubles After NSA Revelations*. URL: `http://www.wired.com/2014/05/sandvine-report/` (Retrieved 10/2014).

[WOH+08] H. Weinreich, H. Obendorf, E. Herder, and M. Mayer. "Not Quite the Average: An Empirical Study of Web Use". In: *ACM Transactions on the Web* 2.1 (2008), pp. 1–31. DOI: `10.1145/1326561.1326566`.

[Wri09] A. Wright. "Ready for a Web OS?" In: *Communications of the ACM* 52.12 (2009), pp. 16–17. ISSN: 0001-0782. DOI: `10.1145/1610252.1610260`.

[XIK+13] G. Xie, M. Iliofotou, T. Karagiannis, M. Faloutsos, and Y. Jin. "Resurf: Reconstructing web-surfing activity from network traffic". In: *IFIP Networking Conference*. IEEE. 2013, pp. 1–9. URL: `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6663499`.

[YF08] B. Yu and H. Fei. "Multiscale analysis and modeling of user session traffic in social networks". In: *IEEE International Conference on Communication Technology*. ICCT '08. 2008, pp. 85 –88. DOI: `10.1109/ICCT.2008.4716125`.

[ZNA05] S. Zander, T. Nguyen, and G. Armitage. "Automated Traffic Classification and Application Identification using Machine Learning". In: *Conference on Local Computer Networks*. LCN '05. IEEE Computer Society, 2005, pp. 250–257. DOI: `10.1109/LCN.2005.35`.

[ZSG+09] M. Zink, K. Suh, Y. Gu, and J. Kurose. "Characteristics of YouTube network traffic at a campus network: measurements, models, and implications". In: *Computer Networks* 53.4 (2009), pp. 501–514. DOI: `10.1016/j.comnet.2008.09.022`.

[ZZ11] H. Zhang and S. Zhao. “Measuring Web Page Revisitation in Tabbed Browsing”. In: *SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. ACM, 2011, pp. 1831–1834. DOI: `10.1145/1978942.1979207`.