DOI: 10.15439/2014F104
ACSIS, Vol. 2

Characterizing webpage load from the perspective of TCP connections

Luis Miguel Torres, Eduardo Magaña, Mikel Izal and Daniel Morato
Departamento de Automática y Computación, Universidad Pública de Navarra, Pamplona, Spain.
Email: [luismiguel.torres, eduardo.magana, mikel.izal, daniel.morato]@unavarra.es

***Abstract*—Over the last years websites have evolved rapidly incorporating new content types and becoming more and more dynamic. Users today are able to access a wide variety of content and services through their web browsers. As a consequence, web traffic has become increasingly complex and, from a network perspective it can be difficult to ascertain which 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, few works have focused on a connection level perspective even if this kind of data is easily available for network administrators. In this paper we offer a characterization of webpage download using connection level metrics. This description is a first step in developing techniques able to identify individual webpage downloads in real traffic.

We have captured an extensive dataset of more than 20,000 webpage downloads that we study in order to provide different connection level based metrics. We study how these metrics vary between different webpages of different popularity and complexity. In the end, we attempt to provide a general modelling of a normal webpage download.

I. Introduction

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 adding a wide variety of content types (like videos, scripts 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 application has achieved a remarkable flexibility that allows it to provide a huge range of different services aside from traditional web browsing.

All these changes have obviously affected the profile of web traffic. Recent studies [1]–[3] show that its characteristics have greatly changed from the (simpler) ones described thoroughly in the 1990s [4]. This is partially the result of the introduction of HTTP 1.1: 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. 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 some 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 sizeable amount of the content is dynamic, these connections may change if the webpage is accessed at a different time or by a different user.

In this paper we present a study of those sets of connections established by clients during the download of webpages. Although there is extensive work in web traffic characterization the approach has usually been very different to ours. Many proposals center their study on server operation, modelling the behaviour and habits of the users that access the server in order to provide them the best possible service [5], [6]. Others have taken a more user-centric perspective but have focused on application-level operation [7], [8] or the characteristics of the downloaded content [9], [10]. Finally, other works have characterized specific types of web applications, usually with the intention of being able to classify their traffic [11], [12].

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. However, even if we work from a client-side perspective, we only consider data that can be captured directly from the network rather than being "inside" the client monitoring its operation. All in all what we offer is a thorough characterization of the set of connections (and by connections we are referring to bidirectional TCP flows) initiated by the client during the download of a webpage.

This TCP-level characterization is interesting because NetFlow-type records [13] are easy to collect and, specially when compared to full packet-level traces, store and process in any network. 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 concurrently 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 behaviour. 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.

The study we present in this paper 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 very 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.

The remaining of this paper is structured as follows: section II explains the methodology used to capture experimental data and gives an overview of said data; section III discusses the general characteristics of a webpage download from a TCP connection point of view; section IV presents some time-based metrics that describe when the connections that participate in the download are opened and closed; section V focuses on the accessed servers; and, finally, section VI concludes and presents future lines of work.

II. Data collection

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

A. 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 [14]. 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 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 measures from a computer in the Public University of Navarra (Spain) 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 [7] 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 a network sniffer: Tcpdump [15].
- 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 [16], [17]. 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 [3], 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 IV.

B. Preprocessing

In order to obtain connection records from these packet traces, we use Argus [18]. Argus is an open-source audit tool that is able to generate connection reports with the same features (and more) than NetFlow/IPFIX. In particular, aside from the classic TCP/IP 5-tuple (IP addresses, ports and protocol) we consider: timestamps (start and end), total packets, total bytes, application-level bytes (upload and download) and TCP state at the end of the capture. As we said previously, we always consider bidirectional TCP connections. Additionally, we store the first 1,000 bytes of the upstream application data of every connection from which we will extract some HTTP header fields. As we are only interested in web traffic we select connections originated in our PC and with destination ports 80 and 443 (HTTP and HTTPS protocols). However, 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.

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. If the root connection carries HTTPS traffic it is impossible to identify it by checking user data. In this case, the root

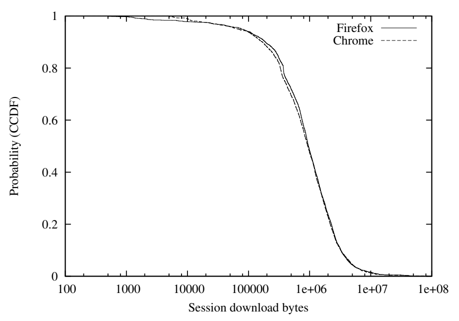


Fig. 1: CCDF of total downloaded bytes in a webpage load

connection will be the first flow opened to an IP address whose related domain name is the name 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 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.

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. We discovered that this browser opens some HTTPS connections to Google servers for different purposes. Some of these connections are related to the webpage (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 connections because, as we said previously, they are related to the browser behaviour and not to the particular websites. We also did not consider some connections opened by Firefox to Mozilla servers.

Now, figure 1 shows the empirical complementary cumulative distribution function (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 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.

III. General characterization

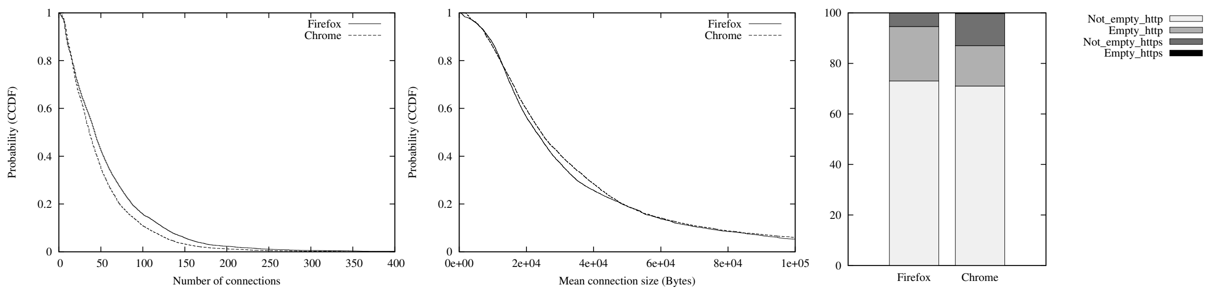
We start by providing some general connection-based metrics of webpage download. As it happened with figure 1 we have aggregated data from all our captures in order to plot the different graphs in this section. Because of this, each of the samples we use to calculate the empirical CCDFs corresponds to 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).

Figure 2a shows the empirical complementary cumulative distribution function of the number of TCP 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% 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. 1) this means that on average, Chrome connections are slightly bigger and more elements of the webpage are grouped in each of them. In any case, in figure 2b we can see that the difference in average connection size is small.

However, if we look at the individual connections (and, specially, the smallest ones) we do find some differences between the browsers' behaviour. In figure 2c 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.

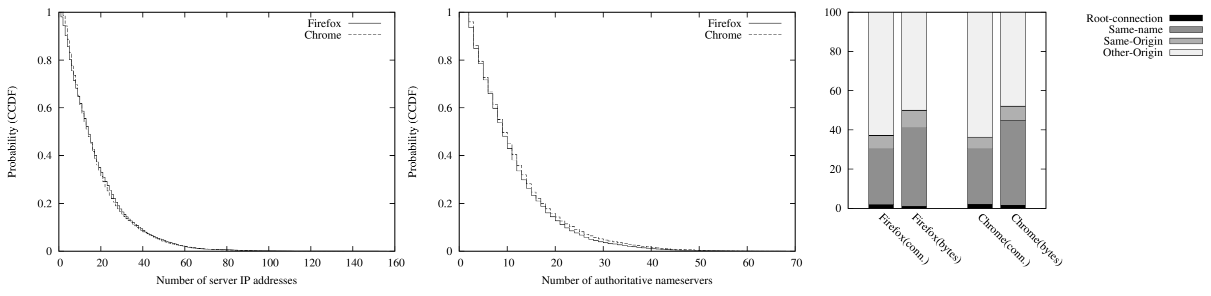
The other kind of connections we consider are empty connections. *Empty connections* are those that, having successfully completed their initial TCP three-way handshake, carry no application data. Most of this connections (around 95%) are also properly terminated although we also consider connections 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 occurrence specially for Firefox in whose traffic they represent around 20% of all connections. In most cases empty connections are a consequence 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 program. Browsers may open multiple connections to a particular server before knowing how much content is going to be downloaded from it because it is faster to have connections prepared in case they can be used to download multiple elements simultaneously. This means that, in some cases, this connections are left unused. Additionally, browsers usually try to open a new connection if the server does not respond to a previous SYN packet within a time limit. This limit is low, again in order to reduce download times.

In figure 3 we focus on the different servers accessed during a webpage download. The differences between both browsers are minimal in this case as browser implementation cannot affect where the elements of a webpage are stored. Figure 3a shows the distribution for the number of different IP



(a) (b) (c)

Fig. 2: CCDF of the (a) number and (b) average size of connections. (c) Percentages of empty and HTTPS connections.



(a) (b) (c)

Fig. 3: CCDF of the (a) number of servers, (b) authoritative nameservers, and (c) percentage of connections by origin.

addresses accessed during each download. The distributions, with medians at 14 IP addresses but long tails, corroborate that modern websites download elements from multiple different servers. As we explained in section II-B we use authoritative nameservers in order to distinguish between different origins for web content. Figure 3b 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 3c we show the percentages of connections and bytes according to their origins. As in figure 2c 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 I we offer a summary of the different per-download metrics presented in this section (number of connections, mean connection size, number of accessed IP addresses, number of authoritative nameservers) providing the median and 10th and 90th percentiles for each of them.

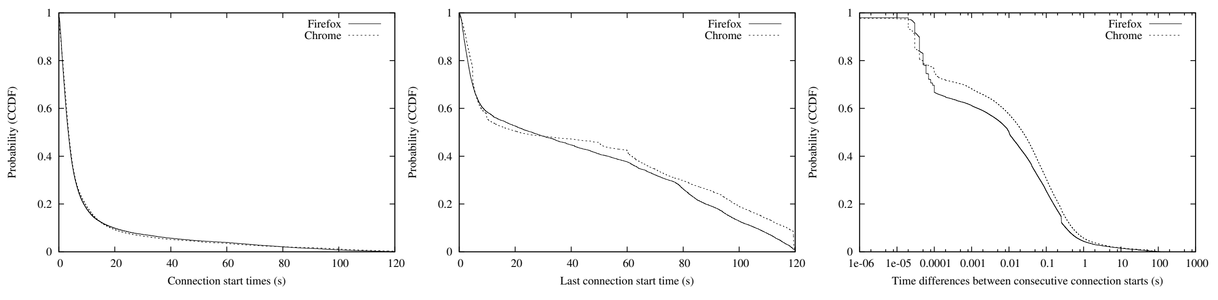
TABLE I: General characterization metrics

	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

IV. TIME METRICS

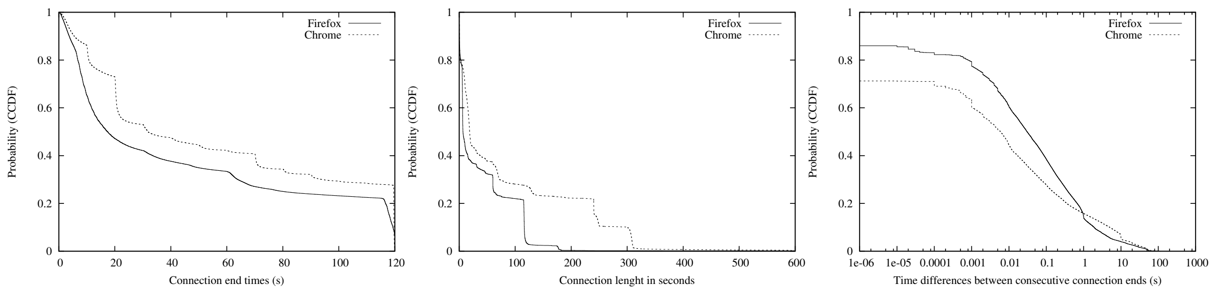
In this section we are going to discuss metrics related to the start and ending timestamps of the flows in the download of a webpage. Again, we consider the aggregated data of every download of every webpage as we want to characterize the behaviour of an average load rather than explore the differences 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 3a with the number of accessed servers per download). In all the figures in this section, the 0 seconds mark corresponds with the start timestamp of the first connection in each webpage download and the rest of connection timestamps in that download are calculated relative to it.

The first parameter that we are going to consider is connection start times which appears in figure 4a. In order to calculate



(a) (b) (c)

Fig. 4: CCDF of (a) connection start times, (b) last connection start time, and (c) time differences between connection starts.



(a) (b) (c)

Fig. 5: CCDFs of (a) connection end times, (b) connection lengths, and (c) time differences between connection ends.

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 [3]. 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. In figure 4b we represent the distributions of the start times of the last connection in each capture and we can confirm that even though most connections are opened in the first seconds, for the sizeable 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 II-B. We expected that the late connections could be specially related to third-party advertisment 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 correspond to third-party content but also to dynamic content hosted in first-party servers.

In figure 4c we look at connection start times from a different perspective by representing 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) and only around 5% have their start times separated more than one second.

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 sizeable 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 5a, both browsers keep more than 20% of the connections opened 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. We realized that 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 5b 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 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).

Figure 5c is equivalent to 4c but this time we are representing time differences between connections that are closed consecutively. We can see that this time 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 II 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 4b. 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 10 and 90 percentiles we realize that we are basically covering the whole length of the capture. 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%). 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 figures 4c and 5c 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 II 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.

TABLE II: 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

V. Server metrics

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 information). 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. Due to space constraints 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 used browser.

In section III we saw that multiple connections are opened to the same servers during each download. On the other hand, in figure 4b 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 6a 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 4b suggesting that a sizeable number of these late connections are opened to servers that have not been previously connected. However, as we said previously, these connections are usually small and the servers that host the main elements of the webpages are accessed in the first seconds of the download.

In section III we also gave information about the total number of different servers accessed in a download and the different authoritative nameservers related to them. However, it is clear that those servers play different 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 6b 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). 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 6b is that, as we know, the content of a webpage is not equally distributed in the different servers accessed during the download. In figure 6c 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 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

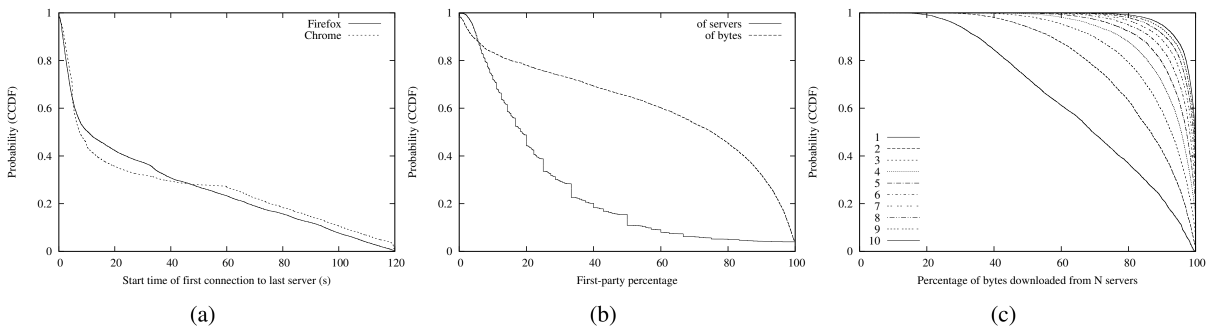


Fig. 6: CCDFs of (a) time of first connection to last server, (b) first-party percentage of servers and bytes, and (c) percentage of bytes downloaded from heaviest servers.

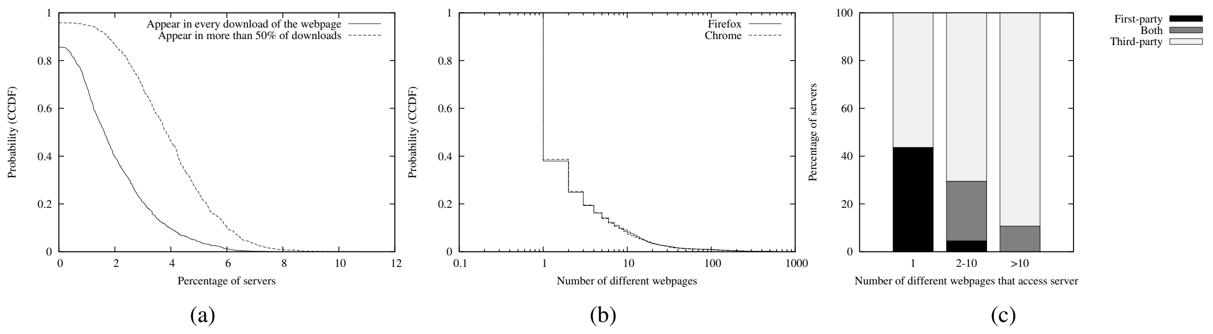


Fig. 7: CCDFs of (a) percentage of servers that appear in all or half the captures of the same webpage, (b) number of different webpages that access each server, and (c) origins of shared servers.

are accessed when downloading different webpages. Figure 7 addresses these situations.

In figure 7a 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 7b 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 services are shared between them. In figure 7c 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 webpage by their name or their authoritative nameserver may host content for other webpages.

In table III we present some server metrics related to the variables we described in figure 6 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.

TABLE III: Server metrics

Metric	Firefox P10	Firefox Median	Firefox P90	Chrome P10	Chrome Median	Chrome 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

VI. Conclusions and future lines of work

The increasing popularity of the web and the interesting complexity of its traffic have made it a popular topic of research for the scientific community. However, little work has been done in order to characterize web traffic at connection level, even though connection level data has the advantage of being easy to store and process in real time and can be collected even if the traffic is encrypted.

In this paper we have presented a thorough characterization of web traffic from the perspective of TCP connections. We have introduced various metrics that describe the set of connections involved in a webpage download 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.

We intend to apply the knowledge obtained in this work into designing a method able to identify webpage downloads in real traffic. We believe that by applying a combination of the metrics described in this paper we will be able to distinguish single webpage downloads in user traffic. A system based on this idea would be lightweight and able to process data in real time giving interesting information to network administrators about the behaviour of their users without accessing sensible information.

We also would like to explore the possibility of using these metrics 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 in this work can help with the tuning of anomaly-based detection systems. These systems that may be able to distinguish between normal web traffic 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).

References

[1] 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*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 5987, pp. 136–150. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-12104-3_12

[2] H. Weinreich, H. Obendorf, E. Herder, and M. Mayer, "Off the beaten tracks: exploring three aspects of web navigation," in *Proceedings of the 15th international conference on World Wide Web*, ser. WWW '06. New York, NY, USA: ACM, 2006, pp. 133–142. [Online]. Available: http://doi.acm.org/10.1145/1135777.1135802

[3] S. Ihm and V. S. Pai, "Towards understanding modern web traffic," in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 295–312. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068845

[4] L. D. Catledge and J. E. Pitkow, "Characterizing browsing strategies in the world-wide web," *Computer Networks and ISDN Systems*, vol. 27, pp. 1065–1073, April 1995. [Online]. Available: http://dx.doi.org/10.1016/0169-7552(95)00043-7

[5] F. M. Facca and P. L. Lanzi, "Mining interesting knowledge from weblogs: A survey," *Data Knowl. Eng.*, vol. 53, no. 3, pp. 225–241, Jun. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.datak.2004.08.001

[6] H. Liu and V. Kešelj, "Combined mining of web server logs and web contents for classifying user navigation patterns and predicting users' future requests," *Data Knowl. Eng.*, vol. 61, no. 2, pp. 304–330, May 2007. [Online]. Available: http://dx.doi.org/10.1016/j.datak.2006.06.001

[7] M. Butkiewicz, H. V. Madhyastha, and V. Sekar, "Understanding website complexity: measurements, metrics, and implications," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 313–328. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068846

[8] G. Xie, M. Iliofotou, T. Karagiannis, M. Faloutsos, and Y. Jin, "Resurf: Reconstructing web-surfing activity from network traffic," in *IFIP Networking Conference, 2013*, 2013, pp. 1–9.

[9] D. Fetterly, M. Manasse, M. Najork, and J. Wiener, "A large-scale study of the evolution of web pages," in *Proceedings of the 12th International Conference on World Wide Web*, ser. WWW '03. New York, NY, USA: ACM, 2003, pp. 669–678. [Online]. Available: http://doi.acm.org/10.1145/775152.775246

[10] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of youtube network traffic at a campus network: measurements, models, and implications," *Computer Networks*, vol. 53, no. 4, pp. 501–514, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2008.09.022

[11] D. Schatzmann, W. Mühlbauer, T. Spyropoulos, and X. Dimitropoulos, "Digging into HTTPS: flow-based classification of webmail traffic," in *Proceedings of the 10th Conference on Internet Measurement (IMC '10)*. New York, NY, USA: ACM, 2010, pp. 322–327. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879184

[12] F. Schneider, S. Agarwal, T. Alpcan, and A. Feldmann, "The new web: Characterizing AJAX traffic," in *Passive and Active Network Measurement*, ser. Lecture Notes in Computer Science. Springer, 2008, vol. 4979, pp. 31–40. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-79232-1_4

[13] E. B. Claise, "RFC 5101: Specification of the IPFIX protocol for the exchange of IP traffic flow information," Jan. 2008.

[14] "Alexa top 1,000,000 sites," Aug. 2013, http://www.alexa.com/.

[15] "Tcpdump & libpcap," Sep. 2013, http://www.tcpdump.org/.

[16] "Statcounter.com," http://gs.statcounter.com/.

[17] "w3counter.com," http://www.w3counter.com/.

[18] "Argus: Audit Record Generation and Usage System," http://www.qosient.com/argus/.