ReSurf: Reconstructing Web-Surfing Activity From Network Traffic

Guowu Xie
UC Riverside

Marios Iliofotou
Narus, Inc.

Thomas Karagiannis
Microsoft Research

Michalis Faloutsos
UC Riverside

Yaohui Jin
Shanghai Jiao Tong University

ABSTRACT

More and more applications and services move to the web and this has led to web traffic amounting to as much as 80% of all network traffic. At the same time, most traffic classification efforts stop once they correctly classified a flow as web or HTTP. In this paper, we focus on understanding what happens "under the hood" of HTTP traffic. One of our key contribution is ReSurf, a systematic approach to reconstruct web-surfing activity starting from raw network data. Even when HTTP traffic is unencrypted, this problem is far from trivial. A key challenge is that websites are complex: a single user request (think user click) creates many network level flows to many different websites. ReSurf overcomes these challenges and reconstructs on average 91% of user requests with more than 95% precision. Our second contribution is an extensive analysis of web activity over four different network traces, including a residential ISP, a large university campus, and mobile data from a cellular provider. By utilizing ReSurf, we study the user behavior in terms of user requests issued and transitions between websites (e.g. the click-through history of following hyperlinks). In terms of user requests explicitly issued towards a site, Facebook dominates in our mobile trace with 38% of user requests compared to 11% for Google. A surprising result is the "shallowness" of the click-through stream with a median of one website transition. Finally, we find that mobile user requests download one third of the objects and generate one tenth of the traffic compared to user requests on the wireline traces.

1. INTRODUCTION

HTTP is the new IP in the Web 2.0 world, and traffic analysis methods need to adapt to this new reality. First, web browsers are being used as the ubiquitous interface to a large number of services and applications, such as Email, gaming, file sharing, video streaming, and social networking sites. Second, today HTTP is the most widely used protocol, contributing up to 80% of the traffic on some networks [12]. One implication of these trends is the limited relevance and applicability of traditional traffic analysis and characterization tools [9, 19]. Assigning flows to an HTTP category today conveys very limited information with regard to the usage of websites/services and web users behaviors.

Given the above trends, it is increasingly important for network administrators to monitor and characterize web traffic for operational and security purposes. First, understanding traffic is important for managing and provisioning one's network. Second, such capabilities are important for security, since modern malware spreads via websites and botnet command & control channels utilize HTTP. Overall, the more information administrators have about the traffic, the more effectively they can manage the network, identify anomalies and prevent attacks. In addition, analyzing web traffic is important for researchers that want to study modern websites and understand their evolution [8, 6].

The overarching problem we address in this paper is the following. Given web traffic collected at a network link, we want to be able to look "under the hood" and reconstruct the user behavior. Here is a list of motivating questions: (a) What websites (e.g., `facebook.com, cnn.com`) are *explicitly* requested by a user as opposed to being accessed automatically in the background? (b) How much traffic is generated by each request? and (c) What are the typical web surfing user patterns and the typical referral relationships across websites? We want to answer these questions starting from raw network traffic, such as a *tcpdump* trace, or web-proxy records [8].

Surprisingly perhaps, answering these questions is challenging even when HTTP headers and payloads are not encrypted. First, users often browse multiple websites at the same time, which causes flows and HTTP requests to intermingle. Second, modern web pages are fairly complex [6]; often rendering a single page generates tens of HTTP requests[1] towards different web servers. Third, many websites, such as content distribution networks (CDNs), web-ad servers, and web analytics services are used by many websites and shared across several services. All the above make the problem of attributing individual HTTP requests to a user request and to the correct primary website quite complex. We discuss the challenges of this problem and the limitations of previous efforts in Sections 2 and 5.

[1]Each HTTP request corresponds to a web-object such as image, video, or javascript. The initial HTTP request is typically a web page (e.g. *html), which can include other objects, which are then acquired by separate HTTP requests.

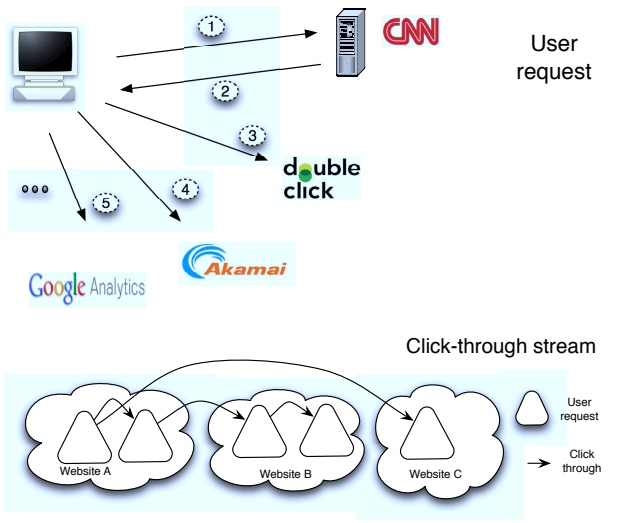

Figure 1: Web-surfing activity: (top) A single user request first generates an HTTP request to `cnn.com`, and then multiple subsequent HTTP requests to other web servers (e.g., `doubleclick.com`); (bottom) An example of click-through stream of user requests over different websites.

To illustrate the complexity of the task, we show visually the technical questions we address through the toy examples of Figure 1. The graph at the top shows a user request towards `www.cnn.com`, which then triggers a sequence of requests towards a CDN server *akamai.com*, a web-ad server *doublecklick.com*, a web analytics sever *google-analytics.com*, and others. The requests and responses are numbered based on how they occur over time. For simplicity, only the first response from the `cnn.com` website is shown. Looking at each of these requests in isolation, it is hard to identify the primary website or service that triggered them. For example, it is hard to say that the request towards the CDN server actually serves the rendering of the `cnn.com` website. In fact, we find that simply using the host name of the server to map requests to websites results in less than 40% accuracy.

Making the problem more specific, we can identify two sub-tasks: (a) we want to group HTTP requests generated by a single user request, such as a click, and associate them with the primary website requested by the user, e.g., `cnn.com` in Figure 1; and (b) we want to reconstruct the **click-through stream**, i.e., the referral relationship of the web-surfing, to capture if a user's request to a website is from a hyperlink clicked on an earlier website or from within the same website. A toy example of a click-through stream is shown at the bottom of Figure 1. Here, the user clicks from website A to website B and she also clicks from website A to website C. The graph also shows the user issuing two clicks inside website A and two clicks inside website B. Understanding web traffic at both the user request and click-through level provides visibility into the user's web-surfing activity.

The problem, as defined here, has not received much attention. Most existing web traffic studies focus on different problems and most traffic classification efforts stop once they identify a flow as web traffic [9, 19], which is our starting point. Several studies [2, 6] focus on understanding the complexity of popular websites by analyzing their homepage and their evolution [8], but but its focus was not real user surfing behavior. Other work [5, 16] just focuses on how users interact with specific online social network websites. We discuss and compare with previous efforts in Sections 3.2 and 5.

In this paper, we make two main contributions: (a) we present ReSurf, an effective approach to reconstruct web-user behavior, and (b) we conduct an extensive measurement study with four real network traces. Our datasets include a residential ISP in Europe, a mobile provider offering 3G and 4G services in the US, a group of users from a research lab in the US, and a large university campus network in China. Our datasets have up to 19 billion HTTP transactions, and range in duration from a three hours to six months. For evaluating and comparing ReSurf, we also use a synthetic data set and a dataset based on accessing the most popular Alexa sites. We explain our two contributions in more detail below.

(a) ReSurf: reconstructing web-surfing activity. We develop a systematic methodology that operates in two steps: (i) we reconstruct user requests (e.g., clicks, refreshes), and (ii) we determine the click-through behavior of the user. To achieve this, we combine information from the HTTP header with timing information at the network level. In a nutshell, ReSurf uses the referrer relationships between related HTTP requests to trace back the user request that generated it. It then uses the size and type of the download objects, as well explicit timing information between successive requests in order to achieve the desirable precision. We provide the details of ReSurf, discuss limitations, and describe our evaluation in Section 3.

(b) Extensive measurements and validation. Our experiments with real data traces and our validation with both real and synthesized data provide the following highlights:

- **ReSurf can reconstruct web-surfing accurately.** We show that our approach can identify and reconstruct user requests with more than 95% precision and 91% recall on all our traces. Our validation further highlights weaknesses in the current state-of-the-art methodologies that rely on web analytics beacons [8].

- **The usual suspects dominate: Google/Baidu, Facebook/Renren, and adult sites.** We quantify the presence of the dominant players of web traffic. In our traces from Europe and the US, adult sites contribute 40% of the traffic. Google is the top referrer website in all our traffic traces, with 34% and 49% of all inter-site referrals in our ISP and mobile trace, respectively. Facebook leads the way with 38% of user requests in our mobile trace, although Google has a small

lead in user request in the ISP trace. In our Chinese trace, Baidu and Renren are dominating, followed by Taobao, an online shopping portal. Interestingly, even in the Chinese dataset, Google is the second most active referrer site. Surprisingly, filesharing sites like `bitshre.com`, `filesonic.com` and `115.com` account for 30-35% of the traffic in volume in both the European residential ISP and the Chinese University campus traces.

- **Web caching reduces the number of downloaded web-objects by three times.** When accessing the same website several times we observe that on average 2/3 of the objects are cached. Therefore, using synthetic traces that accesses popular Alexa websites [6], can overestimate the generated network traffic.

- **Click-through streams are "shallow."** Surprisingly, the median number of websites in a click-through stream is just one. Moreover, we observed that only 5% of the click-through streams have more than three websites.

- **Mobile user requests generate one tenth of the traffic compared to wireline user requests.** On average, the user requests in mobile trace generate one third of the HTTP requests and generate an order of magnitude less web traffic compared to user requests in the wireline trace. This reflects the convergence of online services to mobile equivalent services that are sensitive to the UI limitations of mobile devices and corresponding data charges.

The rest of the paper is structured as follows: In Section 2, we present the problem, provide the necessary background, and describe our data sets. In Section 3, we explain ReSurf, and evaluate its classification performance. The observations extracted from data traces are presented in Section 4. Finally, we discuss related work in Section 5.

2. PROBLEM DEFINITION AND TRACES

The goal of this section is to present the problem in more detail, describe its challenges, and define the terminology. We also present the datasets that we use here.

2.1 Problem definition and background

Web traffic is composed of a sequence of HTTP requests and responses occurring over time. We will refer to an HTTP request and its corresponding response as an **HTTP transaction**. Throughout this paper, we use the terms HTTP request and HTTP transaction interchangeably. When a user requests a website, it causes many HTTP transactions transferred across the network. Each HTTP transaction corresponds to a **web-object**, such as an image, video, HTML file, flash file, javascript, etc. The main website requested by the user is called the **primary** website, e.g., `google.com`, `facebook.com`, `cnn.com` to name a few. We refer to

(1) Initial request to cnn.com

```
GET / HTTP/1.1
Host:  www.cnn.com
Accept-Language:  en-us,en;q=0.5
Connection:  keep-alive
...
```

(2) A advertisement request caused by (1)

```
GET /html.ng/site=cnn&cnn_pagetype=main...
Host:  ads.cnn.com
Referer:  http://www.cnn.com/
...
```

(3) Request to a CDN server caused by (2)

```
GET /cnn/.../advertisement.gif HTTP/1.1
Host:  i.cdn.turner.com
Referer:  ads.cnn.com
...
```

Figure 2: An example of some HTTP requests issued by a web browser during a visit to `www.cnn.com`. For simplicity, only parts of the HTTP headers are shown. The first request directly reflect the action of the user to request (e.g., click on) the `cnn.com` website; and we call this the "head request." The second and third requests are "embedded requests," which are automatically initiated by the user's web browser software.

this first request as the **head HTTP request**, which typically retrieves an HTML or XML file. Usually, this HTML file includes other embedded objects, which are, in turn, acquired by separate HTTP requests initiated automatically by the web browser. We call these subsequent requests **embedded HTTP requests** and they are usually transparent to the user. Ultimately, our goal is to assign each HTTP transaction observed in the network to the user request that generated it. Doing so enables us to measure traffic generated by different user requests and helps us understand web users behavior in the network.

In Figure 2 we present an example of three HTTP requests generated by a visit to `cnn.com`. Following the above terminology, the first HTTP request is the *head request* to the *primary webpage* (i.e., `cnn.com`) and the other two are embedded HTTP requests. For each HTTP request, the domain name of the web server is located in the `host` field of the HTTP header. Even though all three requests are caused by the visit to `cnn.com`, in this example, only one has `www.cnn.com` as the host name. From this example, we see that by looking at an HTTP transaction in isolation, it is hard to know which visit generates the HTTP transaction.

Referrer: The `referrer` field in an HTTP header provides information as to which web-object led to the request for the current web-object. This previous web-object is called the referrer. For example, in Figure 2, we see that the second HTTP request towards the advertising server `ads.cnn.com` has the main web page (`www.cnn.com`)

Name	LAB	ISP-1	MOB	SYN	ALE	CAM
Starting date	Oct 3 2010	Aug 25 2011	Jan 7 2011	Aug 11 2011	Aug 11 2011	Mar 9 2012
Duration	6 mon	24 h	3 h	1 mon	-	2 mon
# of HTTP transactions	1.2 M	1.7 M	22.9 M	186K	973K	19B
Ground truth available	No	No	No	Yes	Yes	No
Payload	Full	Full	Full	Full	Full	HTTP header

Table 1: An overview of the web traffic traces used in our study. The LAB, ISP-1, MOB, and CAM traces contain real-world web activity from thousands of users over different countries. The SYN and ALE traces are traces that were specifically crafted to evaluate ReSurf in a controlled environment.

as its referrer. Similarly, the third HTTP request to the CDN server came from the advertising server and has the name of the advertising server (`ads.cnn.com`) as its referrer. As we explain in Section 3 in more detail, the referrer field can help in our task of attributing HTTP requests to their primary website. The referrer field has one additional key utility. It captures the clicks by the user from one web page that lead to another. For example, when a user visits web page B by clicking a link in web page A, the web browser will place the URL of web page A in the referrer field when generating the HTTP request for web page B. Below, we explain how this helps in identifying the click-through streams of different users in a trace.

User Requests: Typically, a user request happens in the form of clicking hyperlinks, opening and refreshing web pages, writing a URL in a browser's address bar, submitting forms and so on. A user request generates a series of HTTP transactions. As we explained before, the first HTTP request is referred to as the head request. A **click-through stream** is a series of consecutive user requests which have referring relationship between them. Figure 1 shows an example of a user request (top) and a click-through stream (bottom). In the click-through stream of Figure 1, we see two clicks (user request) inside website A, two inside website B, and one in website C. In addition, we see that the user moved from website A to websites B and C by clicking on hyperlinks from inside A.

2.2 Data traces

The six web traffic traces used in our study are summarized in Table 1. Our data cover several thousands of millions HTTP requests, over long periods of time, and at different locations. Details regarding the exact locations and the names of the providers for all our traces are intentionally kept anonymized due to privacy concerns and business agreements. We collected our data from a variety of sources: (a) from monitoring the web activity of users inside a university research lab over the length of six months and over a large university campus over two moths; (b) a 24-hour trace collected at a residential ISP network; (c) a 3 hour trace from a 3G/4G mobile service provider; (d) a trace generated by replaying the browsing history from nine users over a period of one month in a controlled environment; and (e) a trace generated by issuing requests to popular websites in a controlled environment. We collected traces in both controlled and uncontrolled environments, which allows us to both examine user browsing activities in the wild as well as verify the correctness of our methodology. The users in our traces are also diverse, covering academic users in an university lab, residential ADSL users, students and academic staff from a large university campus, as well as mobile device (smartphone and tablet) users. This allows us to compare the browsing pattern difference between different users.

Next,we provide details for each trace. For all traces, we use the same collection methodology: We collected all the IP packets (both header and payload) on TCP ports 80, 443, 8000 and 8080. Due to privacy concerns, the campus (CAM) trace is the only one that contains just the header part of the HTTP requests, without their payload.

LAB: We collected this traffic trace from a research lab in a university in the US. In the lab, there are about 15 graduate students and 20 laptops/desktops. The collection duration covered six non-consecutive months over the period of December 2010 until September 2011.

ISP-1: The trace was collected from an edge link of a European residential ISP. We were given access to only the first five packets of each unidirectional TCP flow. Given that most flows only transfer few HTTP transactions, we did not observe this to be problematic for our study.

MOB: We collected this trace from from a 3G/4G mobile service provider in the US. The vast majority of the traffic is generated by the applications on mobile devices, such as smart-phones and tablets.

CAM: The CAM trace is collected from a university campus in China. Approximately, the trace contains the activity of about 28.2K users. Our monitor point sits on the edge gateway connecting the campus to the public Internet. All downloading and uploading traffic from the whole campus goes through the monitor point. We log all important HTTP header fields for all HTTP transactions on TCP port 80. Specifically, the fields we log are the following: timestamp of each request, client/server IPs, URL, referrer, content-type, content-length, HTTP response code and user-agents. To preserve privacy, client IPs are anonymized. We applied our method to several different days of traffic. The trends extracted from different day of traffic are very similar. So we only show the results for one workday in the rest of paper.

SYN: The trace is generated in a controlled environment for the purpose of evaluating ReSurf. We generated the traf-

fic by replaying a users' browsing history from their Google Chrome browser. We use the web browsing history from nine users that volunteered for our experiments. We extracted the timestamp, referrer and URL field of each visit from their browsing history. Then, we replayed each visit using the procedure described in Figure 3. The replay of a URL works as follows. We remotely instruct the browser (Google Chrome) to open each URL separately. At the same time, we use a packet capturing software (`tcpdump`) to collect all the traffic on TCP port 80, 443, 8000 and 8080. Next, we close the browser after 60 seconds and save the collected HTTP traffic to an individual file. As the final step, we artificially adjust the time stamps and referrer fields to simulate how the traffic would be like if it came directly from the user's surfing activity. Both the timing information and referrer relationships are directly extracted from Chrome's browsing history records (see Figure 3). After replaying all visits, all these individual files are merged to form a complete traffic trace. Since each user activity was collected and stored separately, we effectivity have the ground truth for each HTTP request in the trace.

ALE: The ALE trace is also artificially generated using the same methodology as described above. The only difference is that here we only visit the URLs of the home page of the top 80,000 ranked websites in Alexa [1], and we do not artificially add any referrer fields or modify the timestamps.

3. THE ReSurf APPROACH

In this section, we present our ReSurf methodology, and we evaluate and compare it with existing solutions. In section 3.3, we discuss the practical issues and limitations of our method.

3.1 The ReSurf Methodology

The goal of ReSurf is to group HTTP transactions into user requests (see definition in Section 2.1). Our approach works in two steps. First, we identify the *head HTTP requests* by using different features from each HTTP transaction. These features include: the size of the web-object, the type of the object, the timing between successive requests, and others. Second, we use the referring relationships (see definition in Section 2.1) to assign all the embedded HTTP transactions to their corresponding head request. We explain the methodology in more detail below.

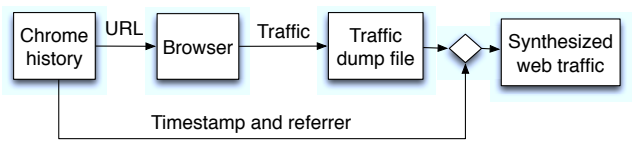

Figure 3: The diagram shows the steps we take to generate synthesized web traffic by replaying a users' browsing history.

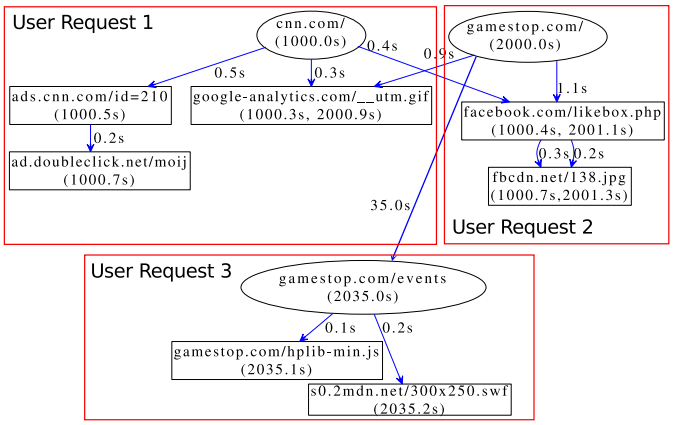

Figure 4: An example of an HTTP referrer graph showing three user requests, one to `cnn.com` and two to `gamestop.com`. We represent the web-objects from head HTTP requests with circles and the web-objects downloaded from embedded HTTP requests with rectangles.

Referrer Graphs: To facilitate our methodology, we first represent all HTTP requests from the same client IP address as a graph. For generating the graph, we use the referrer and host fields from the HTTP requests (see examples in Figure 2). Using these fields, we generate a directed graph that captures the referring and timing relationship between downloaded web-objects. Figure 4 shows an example of the HTTP referrer graph of a single user accessing `cnn.com` and `gamestop.com`. For the purpose of exposition, we simplify the example and keep only a subset of nodes and edges. In reality, even within few minutes the size of the referrer graph reaches several hundreds of nodes. For example, in the CAM trace, the median size of the referrer graph over a ten minute interval is 200 nodes for an IP address. The nodes in the HTTP referrer graph are web-objects annotated with their complete URI. Inside each node we also add the timestamp of it corresponding HTTP transaction to help us understand the ordering of the different requests in our trace. The directed edges capture the referring relationship between nodes. The directed edge from A to B means A is B's referrer. The label of a directed edge represents the timestamp difference between the requests for the two objects.

We provide an intuitive explanation of how our method works using the example of Figure 4. In the figure, we represent head HTTP requests with circles and the web-objects downloaded from embedded HTTP requests with rectangles. In Figure 4, we have three user requests with their corresponding head HTTP request and different embedded HTTP transactions. Note that, initially, the referrer graph does not distinguish between head HTTP requests, and has no information of which groups form User Requests. This information is the outcome of ReSurf. To detect the head requests, ReSurf exploits the following characteristics. First, the head requests of user requests are HTML or XML objects, have

very few incoming edges and many outgoing edges in the referrer graph. At the same time, the nodes in the same user request are very close together in time. On the other hand, HTTP transaction from different user requests are further away from each other in time. Finally, head requests are connected with an edge, if there is referring relationship among them. In more detail, our approach has the following steps:

Step 1. We form the HTTP referrer graph. ReSurf builds an HTTP referrer graph for each host over a period of time (e.g., every five minutes). The creation of the HTTP referrer graph was explained earlier in this section, and an example is shown in Figure 4.

Step 2. We identify all the head HTTP request candidates. ReSurf selects head request candidates according to following rules:

(a) The candidate should be an HTML/XML object. Specifically, its content-type should be one of following: "`text/html`", "`text/xhtml`", "`text/shtml`", and "`text/xml`."

(b) Since most web pages are fairly complex, the size of candidates should be larger than V bytes.

(c) Candidates should have at least K embedded objects.

(d) The time gap between candidates and their referrers should be larger than a predefined threshold T.

(e) As an optional refinement requirement, ReSurf filters out candidates based on the keywords in their URI. That is, candidates' URI should not contain the keywords: `adserver`, `ads`, `widget`, and `banner`.

We observed this keyword heuristic to boost the classification accuracy between 2-3% depending on the trace. We provide specific values for the parameters V, K, T next in our validation section.

Step 3. We finalize the identification of the head requests. We utilize the referring relationship between the head request candidates. Specifically, a candidate is classified as a head request if its referrer is also a head request or if it does not have any referrer. In the referrer graph, nodes with no referrers have no incoming edges. In Figure 4, the cnn.com in "User Request 1" is an example of such a node with no referrer. Such nodes occur when a user, for example, opens a web pages from a browser bookmark or by directly typing the URL in the browser's address bar. If the referrer is not empty, it means that the user navigated to a web page by following the links from a referrer web pages. Following this logic implies that the referrer of a head request should also be a head request itself.

Step 4. We assign embedded HTTP requests to head requests. ReSurf associates embedded HTTP requests to head requests by utilizing the timing information and referring relationship in the referrer graph. In fact, once we know the head request of a user requests, it is easy to attribute the rest of HTTP requests to user requests. For each HTTP transaction (node), we traverse the referrer graph backwards until we reach a head request. If an HTTP transaction (node) has more than one incoming edges, we follow the edge with the smallest time difference (i.e., smaller weight on the edge). In this way, the path will eventually lead back to the head request that was triggered by the user request.

To summarize, the head request of an user request should meet all the requirements in the Table 2.

	Heuristics for detecting head HTTP requests
a	Its content-type should be one of: "text/html", "text/xhtml", "text/xml", "application/xhtml", or "application/xml."
b	Its object size should be larger than V bytes.
c	It should have at least K embedded objects (out-degree in the referrer graph).
d	The time gap between the request under question and its parent request (i.e.g, its referrer) should be more than T.
e	Its URI should not contain any of the keywords: *adserver, ads, widget, embed*, or *banner*.
f	Its referrer is a head request or does not exist.

Table 2: The heuristic rules that ReSurf uses to identify head HTTP requests given a referrer graph. The default values for the above parameters used in ReSurf are $T = 0.5s, V = 3,000\ bytes$, and $K = 2$.

3.2 Evaluation

We evaluate the accuracy of ReSurf using two different sources of ground truth: (a) our synthetically generated trace SYN, and (b) using the web analytics beacons as proposed in *StreamStructure* [8]. We also examine the sensitivity of our approach to its parameters of ReSurf, and justify the values we choose. Finally, we compare ReSurf to the state of the art [8].

Evaluation metrics: We use the standard classification metrics of precision and recall. Precision is the number of true positives (TP) divided by number of TP and false positives (FP), $P = TP/(TP + FP)$. Recall is the number of TP divided by the number of TP and false negatives, $R = TP/(TP + FN)$. We also use the F1 score which is the harmonic mean of P and R, specifically, $F1 = 2 \times \frac{P \times R}{P+R}$.

To evaluate the performance of ReSurf, we ask the following complementary but slightly different questions.

Q.1: How accurately can ReSurf identify head HTTP requests? We want to quantify how effectively ReSurf identifies the head requests from a large set of requests. Given that the number of head requests is usually much less than the total number of requests, this question allows us to focus only on head requests. For example, if out of 100 requests one is a head and the others are embedded, if a classifiers reports all the requests as embedded it would be correct 99% of the times, but would offer limited utility in solving our problem. For this reason, we report the P and R on head requests separately.

Q.2: How accurately can ReSurf classify head and em-

bedded requests? We want to quantify how effectively our approach classifies each HTTP requests as a head or an embedded HTTP request. Unlike Q.1, we report results over all HTTP requests and not only over the head requests. That is, precision represents the number of correctly classified HTTP requests compared to the total number of HTTP requests classified by our algorithm. Note that ReSurf may leave some requests unlabeled (a.k.a unknown). Recall expresses the total number of classified HTTP requests compared to the total number of existing HTTP requests in the trace.

Q.3: How accurately can ReSurf associate HTTP requests to their corresponding user request? This is a more demanding question than the classification for Q.1 and Q.2: we want to associate each HTTP request with the generating user request. This is a **multi-class classification** problem, where each user request is a separate class. For example, if an embedded HTTP request R is correctly identified as embedded, but it is associated with the wrong user request, we will consider it a misclassification. The precision captures the number of correctly classified HTTP requests compared to the total number of HTTP requests classified. The recall reports the correctly classified HTTP requests compared by the total number of HTTP requests in the trace.

We use the following values for the parameters in ReSurf: $T = 0.5$, $V = 3000$ and $K = 2$. We justify this selection later in this section.

A key issue in evaluating any classifier is how to determine the ground truth in the datasets. To address this challenge, we use two different approaches: (a) using a synthesized trace, and (b) using the labels from a classifier that is based on web analytics beacons.

(a) Validation using ground truth from the SYN trace. To evaluate our approach, we created a synthesized trace under a controlled environment. During the creation of this trace, at each point in time we knew exactly which website was being visited, and what requests were generated by the visits to those websites. Details regarding the generation of the SYN trace are given in Section 2.2. Even though we know the ground truth for the ALE trace as well, we do not show results with that trace here since it does not represent real browsing activity. Figure 5 shows the precision, recall and F1 score when we apply ReSurf on the SYN trace, for all three questions, Q1-Q3. As we see, all metrics are above 90%, showing that ReSurf can successfully identify the originating website for the vast majority of HTTP requests. Moreover, we see that the precision of ReSurf is very high, 96% and above, implying high confidence in our classification of requests.

(b) Validation using web analytics beacons as ground truth. For the LAB, ISP-1 CAM, and MOB traces, we do not have the ground truth. Therefore, we compare the classification performance of ReSurf based on the predictions given by the *StreamStructure* [8] method. This method is based on the observation that many websites use web analytics beacons to track their web pages and objects. Intuitively, the web analytics beacons report to the analytics server, and this helps us detect which is the head request and towards which primary website. We consider web analytics beacons from three major services: `google-analytics.com`, `pixel.quantserve.com` and `yieldmanager.com` [8, 10, 11].

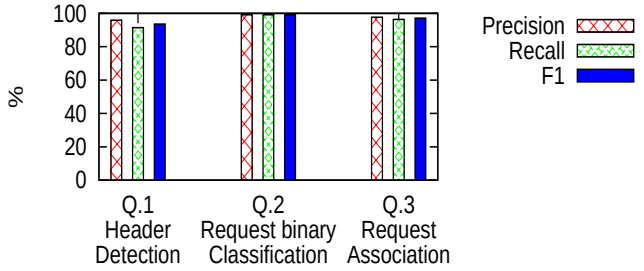

Figure 5: The precision, recall and F1 score in the SYN trace.

Here, we give a more detailed explanation of the beacon method (i.e., StreamStructure), using the google-analytics beacon as an example. Once the object or web page tracked by google-analytics is requested, a beacon is generated based on the requested object's URI and sent to a google-analytics server in the form of "special" HTTP GET request. Unlike regular HTTP GET requests, their URIs encode various information about the user request, including the URI of the requested object/page. Therefore, after some careful parsing of the HTTP requests, we can identify the beacons, and from there we can identify the URL of its main object, which leads us to the primary website of the user request. We refer the reader to [8] for more details about *StreamStructure*.

As we will discuss later in this section, *StreamStructure* can be used for only a fraction of the requests, since only a small percentage of requests use beacons. However, this set of requests can help us determine the effectiveness of ReSurf providing an additional ground truth set. To achieve this, we first use *StreamStructure* to identify as many head requests using beacons. We refer to this set of identified head requests as S. Then, we compare how well ReSurf performs over the known set S. Figure 6 shows the precision, recall and F1 for head detection (Q.1) using beacons as ground truth. We observe that ReSurf achieves above 96% precision in all traces and 91-98% recall. The results show that our approach performs consistently well across all the datasets, which are collected in different continents and during different time periods. Note that we only use web analytics beacons here to establish the ground truth, but **ReSurf does not use beacon information** during its classification process.

Evaluating ReSurf over a different range of parameters: We examine the effect of different parameters on the performance of ReSurf. We only show the plots for Q.1 for brevity; the performance for all questions is qualitatively the same. We use the SYN trace to set our parameters and then apply them to the rest of the traces.

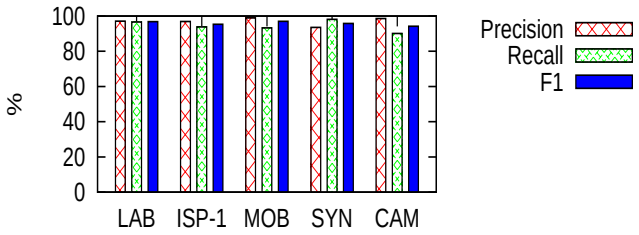

Figure 6: The precision and recall for detecting head requests (Q.1) using web analytics beacons as ground truth.

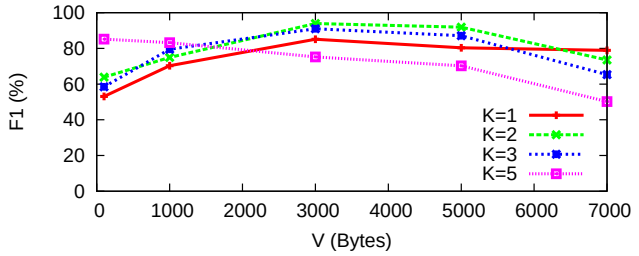

Figure 7: The F1 score of detecting head requests (Q.1) as a function of parameter V and for different values of parameter K in the SYN trace.

Figure 7 shows the F1 metric for detecting head requests (Q.1) using different values for the volume V and the out-degree K over the SYN trace. We observe that the precision increases and the recall decreases as we increase the value of V. Intuitively, large `html/xml` files are more likely to be the web-page of an actual user request compared to shorter ones. Short `html/xml` files typically carry advertising related content and are triggered by embedded requests. At the same time, by further increasing V, we start considering only very large `html/xml` files and we start missing requests, which results to lower recall. As we see from Figure 7, the combined behavior or P and R captured by the F1 score, exhibits good performance for V in the range of 3000 to 5000 bytes. To achieve both good precision and recall, we choose $V = 3000$. In the same figure, different lines show how the out-degree K varies from 1 to 5. We find that the values 2 and 3 gave the best results, with $K = 2$ performing slightly better in the range of parameter V.

Regarding T, we found that our approach exhibits good performance as long as T is less than 1 second and more than 0.1 second. The results are not shown due to space limitations. In the rest of paper, we use this parameter setting: $T = 0.5$, $V = 3000$ and $K = 2$.

Using web-analytic beacons is not enough. A natural question is why we don't just use web analytics beacons exclusively for user request reconstruction. Even though the use of beacons gives good results for those websites that use them we identify several limitations:

The majority of user requests (80%) do not have a beacon in our data traces. We find that less than 20% of the user requests that were found by ReSurf have beacons in the LAB, ISP-1, CAM, and MOB traces. Given the precision and recall of ReSurf in the controlled datasets, we are confident that this percentage is reasonably accurate estimate of requests in the trace. To further verify this, we used the SYN trace, for which we have the ground truth. We plot the results comparing ReSurf with the *StreamStructure* approach [8] in Figure 8. We observe that for the SYN trace, less than 22% of user requests are detected using beacons. To summarize, we observed that using beacons, we can only successfully identify approximately 20% of the user requests, compared to above 91% we achieve with ReSurf. A comparable statistic in the number of head requests that have beacons of 23.9% is also reported in other studio [8].

An additional complication is that sometimes, one user request has multiple beacons and could confuse beacon-based reconstruction solutions.

Figure 8 shows the recall for detecting head requests in the SYN and ALE traces using *StreamStructure* and ReSurf. As we see, with *StreamStructure* the recall is 22% and 60% for the SYN and ALE traces, respectively. The higher recall in the ALE trace is due to the higher popularity of web analytics by very popular websites. By contrast, ReSurf works consistently well in both traces with recall above 92%. Unfortunately, for the LAB, ISP-1, MOB and CAM traces, we cannot repeat the same experiment since we do not have ground truth. Overall, we observed that ReSurf identifies double the number of head requests in these traces compared to *StreamStructure*.

3.3 Discussion

What about encrypted web traffic? ReSurf uses information from the HTTP header, therefore, if the web traffic is encrypted (e.g., using HTTPS) our approach will not classify those flows. However, by analyzing our real-word traces (see Table 1), we observed that the encrypted traffic only amounts for 2% to 8% of the total web traffic. The lowest percentage corresponds to the mobile trace, suggesting that encryption in smartphone applications is not popular. Overall, we observed that unencrypted web traffic is the norm today and we believe it will continue to amount for a significant portion of the traffic in the future. The analysis of encrypted web traffic remains an interesting, open problem.

How is ReSurf affected by users behind network address translation (NAT)? Having users behind NATs is very similar to having users with very high activity. Since referrer graphs are built per IP, NAT users will appear as one "heavy

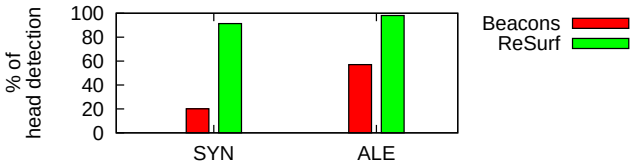

Figure 8: The recall for detecting head HTTP requests (Q.1) using beacons and ReSurf in different traces.

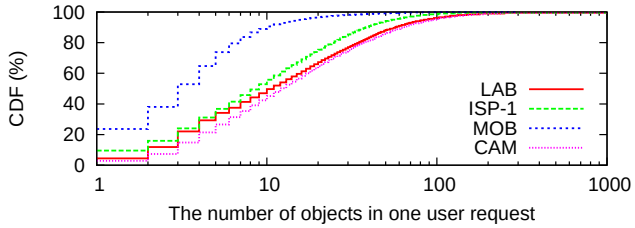

Figure 9: The CDF of the number of downloaded objects per user requests over different traces.

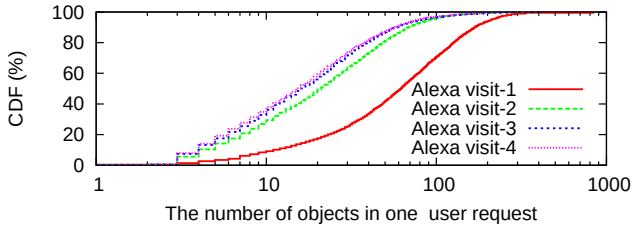

Figure 10: The number of downloaded objects in different visits towards the top 80,000 websites reported by Alexa.

user" with a complex referrer graph, for user accesses within the same time windows. There are two cases here: If different NAT users browse completely different websites, their referrer graphs will not be connected and ReSurf will distinguish different requests. On the contrary, in the worst case where two users request the same web page at the same time, ReSurf will combine them as one large request. However, it will still be able to attribute their traffic to the originating website. Finally, there may be cases where some embedded requests are "multiplexed" between more than one user requests and disambiguating is hard; however, we have not observed that to be a problem in our study. Note that our goal is twofold: i) Group HTTP requests to identify the initial user requested page, and ii) identify the user click-through stream. Hence, having users behind NATs does not affect the first goal, while the second is impacted if users follow the same stream of pages at the same time.

Can ReSurf classify traffic in real-time? Our current implementation does not support real-time classification. In Step 1, we require the collection of traffic for several minutes before we analyze the referrer graph and classify the different requests. Therefore, our approach can classify requests several minutes after their creation. As mentioned earlier, off-line analysis of web traffic is useful to operators that want to understand how their network is being used, as well as for researches that want to study modern trends and changes in web activity. Real-time classification can be important to network operators that want to enforce different policies and achieving this requirement is left as future work.

4. USING ReSurf ON REAL WEB TRAFFIC

In this section, we use ReSurf and analyze the four real-world web traffic traces: LAB, ISP-1, MOB and CAM. First, we group HTTP transactions into user requests with ReSurf. Then, we analyze web traffic at two levels: (a) website, and (b) click-through stream. At the website level, we study how much traffic is caused by different user requests and which are the most popular websites. At the click-through level, we analyze how users behave and how they move from one website to another. Finally, we present differences between mobile and wireline web traffic.

4.1 Analyzing requests at the website level

In Figure 9, we show the CDFs of the number of downloaded objects by all the user requests in the four traces. As a first observation, we see significant differences between mobile and wireline traces. The median number of web-objects per user request is 4 for the mobile trace compared to 11 for the three wireline traces. Similar trends are also observed for the volume of generated traffic, the number of generated TCP flows, contacted IPs, and autonomous system numbers (ASNs) which are not shown here due to limited space. We also observe that approximately 40% of the user requests in the wireline traces results in the download of 20 objects corresponding to 100 KBytes traffic (median). This shows that user requests to modern websites trigger a fairly large number of HTTP requests (web-objects) as well as high traffic volume. Finally, we observed a significant similarity between the CDFs in the LAB, ISP-1, and CAM traces, especially for the lower 50% of user requests. This shows that user requests in wireline traces have similar characteristics, even when the users are in different countries with different websites being popular. The two most similar traces are LAB and CAM. This suggests that even with the small number of users we have in the LAB trace, it captures behaviors that closely match a trace of thousands of users in a large university campus (CAM).

4.1.1 *Caveats of using synthesized data*

To further understand user requests towards popular websites, we compare the properties of the ALE trace with our other traces that represent real user behavior. The use of popular Alexa [1] websites has also been used by other studies [6] that aim to understand the complexity of modern websites. Applying ReSurf on the ALE trace, we observed some significant differences between this trace and our real-world traces. Our hypothesis for this difference is that the browsers' local cache may affect web traffic measurements significantly. To quantify the effect of local caching, we visit the homepage of the top 80,000 website in Alexa [1] four times. The time gap between successive visits is ten minutes.

In Figure 10, we show the CDF of the number of downloaded objects in different visits. On average, the second visit *only downloads one third of the objects of the first visit, and only generates about one third of the network traffic*. In other words, about two thirds of objects are cached locally after the first visit. The third visit requests even less ob-

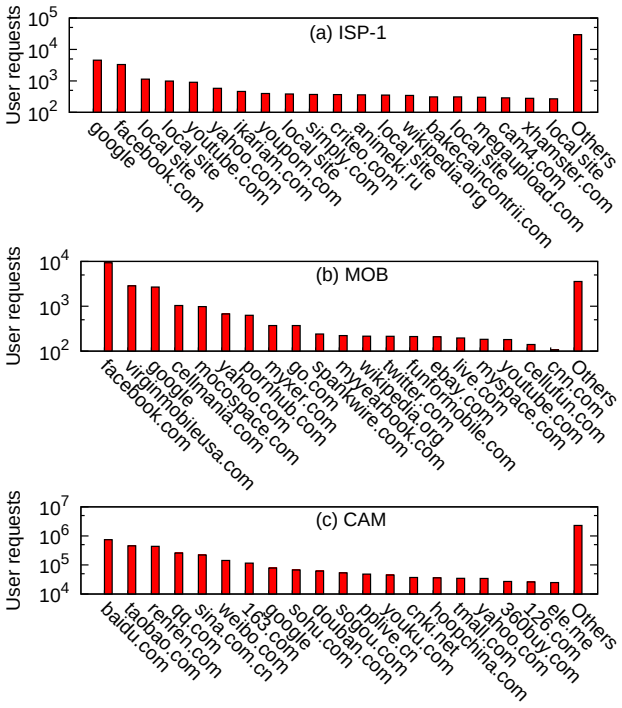

Figure 11: The top websites in term of user request.

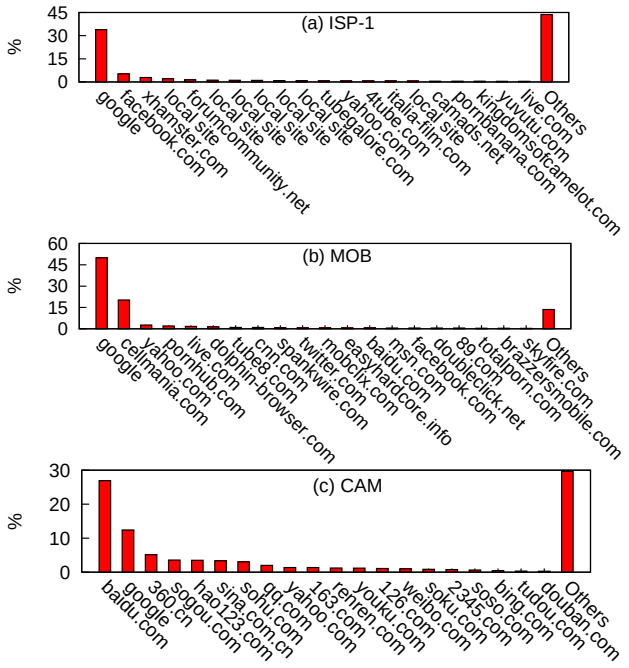

Figure 12: The top external referrers in different traces.

jects, which suggests additional objects being cached by the second visit. Interestingly, there is no significant difference between the third, fourth, and so on, visits in the number of downloaded objects and network traffic. Even though the number of objects decreases significantly between successive visits, we have not observed this to be true about the total number of unique server IPs being contacted. In fact, the total number of contacted IPs in the first visit is just 9% more than the following visits. As expected, since the number of object downloaded decreases, we observe the average number of downloaded objects per server IP to decrease significantly as well. After further investigation, we observed that the different IPs often correspond to third-party analytics and advertising servers, which serve content that is not usually kept in the local cache. Therefore, those IPs are contacted during every visit.

Key takeaway: Overall, *Alexa-based studies seem to overestimate the number of downloaded objects and generated traffic by as much as three times, when compared to actual user traffic*, due to local-cache effects. On the other hand, the number of contacted IPs and domains do not seem to be affected by local caching. It is therefore important to have these two facts in mind when analyzing trends by synthesizing requests to popular websites.

4.1.2 Website popularity

We now turn our attention to website popularity in terms of user requests, external referrers, traffic volume, and network flows. Figure 11 shows the top 20 websites in term of user requests in the ISP-1, MOB, and CAM traces. As expected, the "usual suspects" are in the top places in the ISP-1 and MOB traces, e.g., `Google`, `facebook.com`, `wikipedia.org` and `youtube.com`. The CAM trace is collected in China, which explains the differences in website popularity. Other top websites represent local preferences, such as news sites and portals. This is especially visible in the ISP-1 and CAM traces. The specific local websites in the ISP-1 traces are kept anonymized because of a business agreement. Additionally, Figure 11 shows a large percentage of traffic in to be cause by "Other" websites, which shows that web-traffic does not consists of just a handful popular websites.

Besides popularity, understanding how users reach a particular website is of interest to both website operators and designers. Figure 12 shows the top 20 external referrer websites in the ISP-1, MOB, and CAM traces. A website is considered as an external referrer if it refers users to other websites. It is worth mentioning that we aggregate all international versions of websites into one, e.g., `google.it` and `google.br`, are aggregated together as `Google`. The largest external referrer in all traces collected in Europe and in the US is `Google`. It accounts for over 30% and 45% of all external referrers, respectively. In the LAB trace, the percentage is as high as 80%. The plot for LAB trace is not included here because the user population is too small to draw meaningful conclusions at this level. The second largest external referrer, `cellmania.com`, in the MOB trace, is a portal website for mobile devices. It integrates news, map & weather, wireless search and email for mobile users. Figure 12(c) shows the referrers for the Chinese dataset (CAM). In CAM, the largest referrer website is `baidu.com`, which

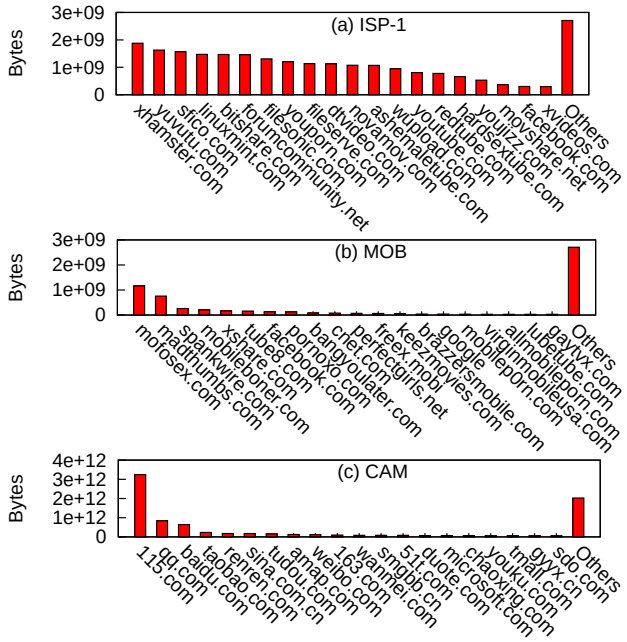

Figure 13: The top websites in term of traffic volume.

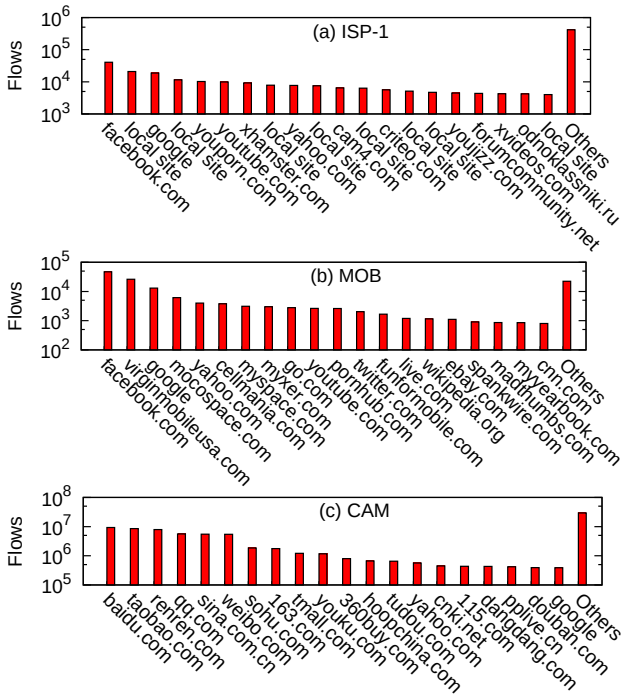

Figure 14: The top websites in term of flows.

is the most popular search engine in China. Interestingly, *Google is the second most popular referrer site in CAM*. In summary, major external referrers are search engine, portal and social networking websites in both wireline and mobile traces.

Figure 13 shows the top 20 websites in term of aggregated traffic volume in the ISP-1, MOB, and CAM traces. Nine out of the top 20 websites in the ISP-1 trace and 16 of 20 in the MOB trace are adult websites. In fact, adult websites in both traces account for more than 40% of total traffic. The ratio of adult websites in the MOB trace (16/20) is much higher than the one (9/20) in the ISP-1 trace. One possible explanation is that people prefer browsing adult websites through personal mobile devices due to privacy reasons. In contrast to the ISP-1 and MOB traces, in the CAM trace, none of the top sites is an adult site. We attribute this to stricter control and censorship in this network. Finally, we see that the second group of big players in the ISP-1 trace are online file sharing websites, e.g., `bitshre.com`, `filesonic.com` and `fileserve.com`, which account for roughly 30% of the total traffic. The site with largest traffic volume in CAM is the file-sharing website `115.com`, which accounts for roughly 35% of total traffic. Online file sharing websites are significant contributors of HTTP traffic [3]. Overall, online file sharing websites and adult websites seem to be the top sources of web traffic in our traces.

Finally, Figure 14 shows the top 20 websites in term of flows. In both ISP-1 and MOB traces, `Facebook` is the number one. There are two possible reasons. First, `Facebook` is highly popular among users. Second, `Facebook` is web 2.0 website and adopts AJAX techniques: `Facebook` pages periodically update themselves even when users leave `Facebook` pages in background. Moreover, the continued interaction of users with `Facebook` and the extensive use of load balancing through CDNs, results in multiple connections being created over time. Finally, the popular `Google` search and `YouTube` video services also rank in the top in both ISP-1 and MOB traces. In Figure 14(c), the CAM trace shows again some differences with the traces from Europe and US. The website with the most flows is `baidu.com`, a popular search engine. Furthermore, the social network sites `renren.com` and shopping portal `taobao.com` (think amazon) are also among the top websites in terms of flows.

Key takeaway: Overall, examining basic popularity trends in this section highlights that different statistics provide different views. For example, looking at user requests versus generated network traffic or network flows results in differences in the popular services, which shows the importance of accurately reconstructing user requests. This highlights the advantage of having a methodology such as ReSurf that can provide such key insights into web traffic.

4.2 Click-through streams

Having examined the basic properties of user requests and website popularity, we now use the referring relationship between web pages to better understand user browsing patterns. We focus on two main questions: (a) How long is a click-through stream in term of user requests, i.e., do user clicks take them through several websites? and (b) what are the typical transitions in these click-through streams, i.e., which websites refer other websites?

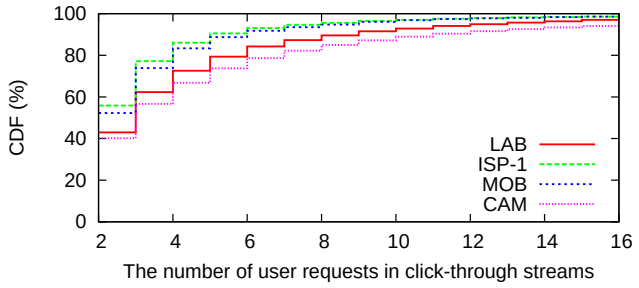

Figure 15: The number of user requests in click-through streams (timeout $T = 1800s$).

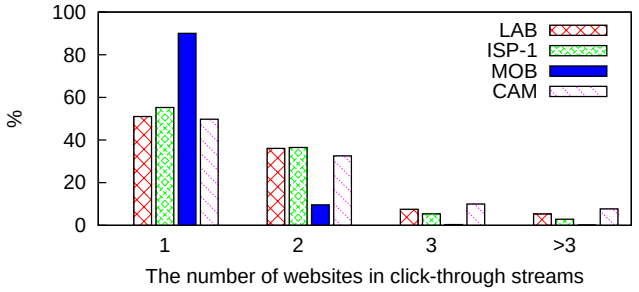

Figure 16: The number of websites in click-through streams.

In Figure 15 we show the number of user requests in click-through streams with a timeout parameter of half an hour. That is, we consider a user request to be a part of a click-through stream if and only if it is less than half an hour away from the previous user request in the stream. We experimented with different timeouts in the range of five minutes up to one hour with very similar results which are not shown where for brevity. Figure 15 shows that the mean (median) number of user requests in click-through streams is 4.5 (3). The 95th percentile of the number of user requests is about 11. This implies that, *typically, click-through streams are short with the users giving up browsing after a small number of user requests.*

In Figure 16, we show the distribution of the number of websites that are being visited during a single click-through stream. The median number of websites in a click-through streams is two for the wireline users and one for the mobile users. This observation suggests that mobile users are less likely to click on links that take them to different websites, which might be due to lower available download rates. Intuitively, this suggests that mobile users have an application in mind when using the Internet and are less likely to "surf around." Finally, it is interesting to observe that for all traces the 95th percentile is only three. Perhaps the use of specialized web services, such as social networking and search engines, explains this behavior.

4.2.1 *Transitions between websites*

Click-through streams also allow us to track the transition across different websites. To this end, Figure 17 shows the most frequent 20 website transitions in different traces. The figure shows that transitions depend on the trace examined. This probably reflects both the different type of collection environments and locality characteristics. Specifically, in the LAB trace (not shown due to space limitation), most transitions are from `Google` to academic and programming related websites reflecting users that are graduate students in a research lab. As shown in Figure 17, website transitions in the ISP-1, MOB, and CAM traces are much more diverse because of the larger user populations. In the ISP-1 trace, most website transitions are from `Google`, `Facebook` and online forums to video and game websites. In MOB, most website transitions are from search engine and portal websites to video and news/blog websites. In CAM, there is a considerable percentage (roughly 10%) of website transitions from portal websites like `360.cn`, `hao123.com` and `2345.com` to other websites, which we did not observe in the other traces.

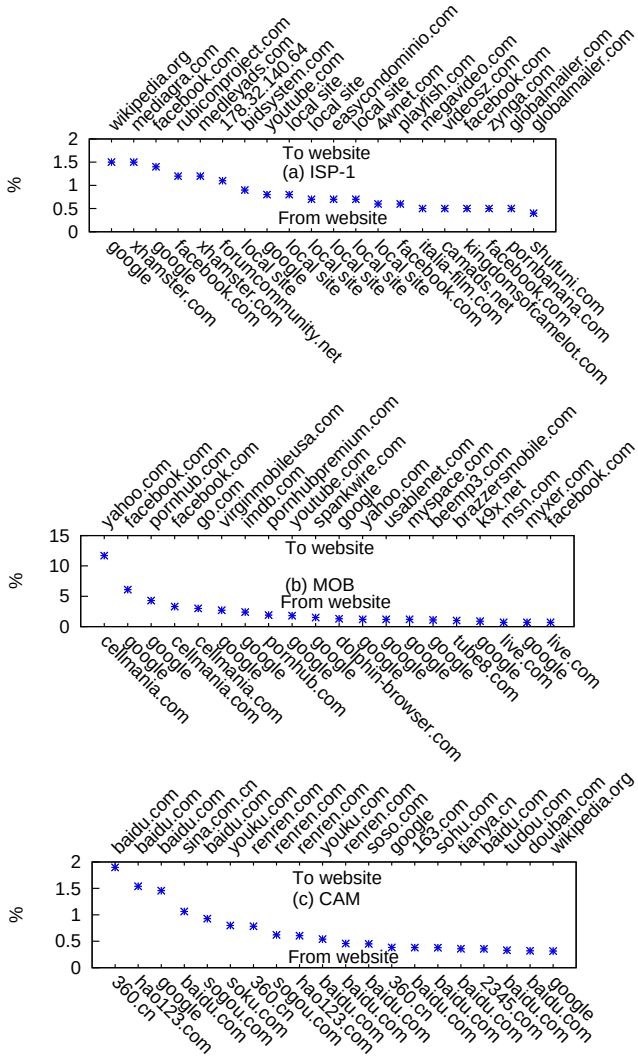

Figure 17: The top website transitions over three traces.

Not surprisingly, our measurements show that "referrer" websites are usually search engines, social networking and online forum websites, while "referred-to" websites are content providers, like `wikipedia`, `YouTube` and News/blog websites.

4.3 Characteristics of mobile web traffic

Here, we focus on web traffic generated by mobile users and further emphasize on websites that offer dedicated smartphone applications for their users [20]. With these proprietary applications, mobile device users can access websites without traditional web browsers. As we explain below, identifying the traffic from those applications requires a less complicated approach compared to ReSurf.

To analyze mobile traffic, we further decomposed the MOB trace into three categories: (a) traffic generated by user requests towards regular web pages using web browsers on mobile devices, (b) traffic generated by user requests towards customized mobile web pages using web browsers, and (c) traffic generated by user requests using smartphone applications (e.g., FacebookTouch). We distinguish the traffic generated by smartphone applications by examining the `user-agent` field in the HTTP headers. Further, to distinguish the traffic generated by (a) and (b), we first reconstruct user requests by applying ReSurf. Then, we match the head request's URL against a list of keywords. If a head request's URL matches one of the following patterns: `m.*`, `*.mobi`, `*/wap`, or `mobile.*`, we considered the user request to be towards a mobile customized web page. Otherwise, it is a regular web page. For example, `facebook.com` is towards a regular web page while `m.facebook.com` is the customized version. We will refer to them as "Browsers→Regular" and "Browsers→Customized" through the remaining of the paper.

Identifying the traffic from smartphone applications. We observed that HTTP headers generated by these applications only include the user agent, host, URI, content-type, and length fields. That is, other important fields like the referrer and cache control information are missing for the majority of cases. Therefore ReSurf is not applicable here. In a nutshell, we classify the flows generated by smartphone applications using the user agent information in the HTTP headers. We first extract all user agents present in the MOB trace. In total, there are 534 different user agents after removing version numbers. Then, we manually compile a keyword list for all the smartphone applications, which covers 118 smartphone applications' user agents. Finally, we classify all HTTP flows into applications by searching for these keywords in user agents. The most popular applications in terms of the number of networks flows are Facebook, YouTube and Pandora. Finally, we simply use timing information to group HTTP transactions into user requests. A user request expires if it is idle for more than 30 seconds. We observed qualitatively similar results by using different timeouts in the range of few second to few minutes.

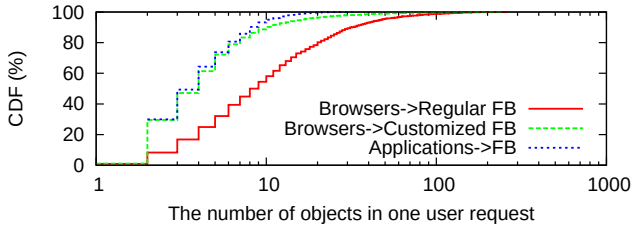

Figure 18: The number of objects downloaded by user requests initiated by traditional web browsers towards regular and customized Facebook web pages, as well as by Facebook smartphone applications.

For presentation purposes, we focus on the comparison of the traffic generated by Facebook smartphone applications and traditional web browsers, both for the "Browsers→Regular" and "Browsers→Customized". We observed qualitatively similar results for other popular services and we use Facebook here as a representative example. Figure 18 shows the CDF of downloaded objects in user requests initiated by smartphone applications and browsers in the MOB trace. We observed that user requests initiated by smartphone applications (Applications→FB) and by traditional browsers (Browsers→Customized FB) towards to customized Facebook pages are similar in term of downloaded objects. However, user requests initiated by web browsers (Browsers→Regular FB) towards regular Facebook web pages, on average download almost twice more objects, and generate four times more network traffic. This shows how mobile users can benefit from the use of such applications when accessing their favorite web service by downloading only relevant content.

5. RELATED WORK

The recent trend of network services over HTTP attracted the interest of the research community. Labovitz et al. [12] brought to light the fact that most inter-domain traffic is HTTP. Schatzman et al. [15] present a methodology to identify web-based mail servers, and distinguishing between services, such as Gmail and Yahoo mail. Erman et al. [7] analyze traffic from residential users and find that a significant part of HTTP traffic is generated by hand-held devices and home appliances, while a large fraction is machine generated (e.g., OS/Anti-virus updates, ads). Li et al. [13] present methods to identify the type of the object transferred over HTTP (e.g., video, xml, jpeg). The recent work from Schneider et al. [17] characterize the inconsistencies between observed HTTP traffic and what is advertised in its HTTP header. All this previous work is complementary to our work, as they focus on different problems, and not the reconstruction of web-surfing at the user requests level and the target websites.

There are three categories of user requests reconstruction methods in the existing literature. The first category assumes

that any HTTP request for an HTML object is considered as the head HTTP request of a user request. The second category is based on the timing information of HTTP requests [18, 4, 14]: if the idle time between two HTTP requests is smaller than a predefined threshold, they belong to the same user request. Both these two categories of methods were effective in the early days of the web, but are not longer effective due to the complexity of the web 2.0 world. The most relevant work to ours is the one that focuses on the evolution of web traffic [8] starting from logs of web proxy servers. To understand modern web traffic and measure web page complexity, they propose a method, `StreamStructure`, to detect "primary" web pages requested by users. A key limitation of `StreamStructure` is its dependence on Google Analytics beacons, which seem to form the basis of their reconstruction algorithm without which accuracy drops significantly. Recall that about 80% of user requests do not have web analytics beacons.

In tangentially related work, [10, 11] study the privacy issues arising from the use of web analytics beacons in web pages. Finally, Xu et al. [20] study the diverse usage patterns of general smartphones applications from mobile network traces. Beisdes the different focus of this work, our study further compares various web-traffic characteristics across mobile and wireline traffic.

6. CONCLUSIONS

Web traffic dominates current network traffic, with HTTP being ubiquitous across different applications. We frame and address a relatively novel problem: reconstructing web-surfing behavior from network data. The problem is far from trivial given the complex and interconnected websites of today. As a key contribution, we develop ReSurf which can reconstruct user requests with more than 95% precision and 91% recall. As our second contribution, we showcase interesting results that one can obtain from raw network traffic by analyzing a number of network traces including a residential ISP and mobile user data. A surprising result is the "shallowness" of the click-through stream of users accessing websites with a median of two transitions. Considering the recent trends of web browsing through custom applications, we expect this shallowness to be the norm of user browsing patterns in the future. We also quantify differences between mobile and wireline web-access patterns: mobile user requests download one third of the objects and generate one tenth of the traffic compared to user requests on the wired trace. Such findings are just a sample of the results and analysis that ReSurf can enable.

In the big scheme of things, ReSurf and similar future algorithms, represent an enabling capability for ISPs and network administrators, that want to manage their networks effectively, as well as for network researchers that want to analyze and study modern web traffic.

7. REFERENCES

[1] http://www.alexa.com/topsites.

[2] AGER, B., MÜHLBAUER, W., SMARAGDAKIS, G., AND UHLIG, S. Web content cartography. In *ACM IMC* (2011).

[3] ANTONIADES, D., MARKATOS, E., AND DOVROLIS, C. One-click hosting services: a file-sharing hideout. In *ACM IMC* (2009).

[4] BARFORD, P., AND CROVELLA, M. Generating representative web workloads for network and server performance evaluation. In *ACM SIGMETRICS Performance Evaluation Review* (1998).

[5] BENEVENUTO, F., RODRIGUES, T., CHA, M., AND ALMEIDA, V. Characterizing user behavior in online social networks. In *ACM IMC* (2009).

[6] BUTKIEWICZ, M., MADHYASTHA, H., AND SEKAR, V. Understanding website complexity: Measurements, metrics, and implications. In *ACM IMC* (2011).

[7] ERMAN, J., GERBER, A., AND SEN, S. HTTP in the home: it is not just about PCs. In *ACM SIGCOMM Computer Communication Review* (2011).

[8] IHM, S., AND PAI, V. Towards understanding modern web traffic. In *ACM IMC* (2011).

[9] KARAGIANNIS, T., PAPAGIANNAKI, K., AND FALOUTSOS, M. Blinc: multilevel traffic classification in the dark. In *ACM SIGCOMM* (2005).

[10] KRISHNAMURTHY, B., AND WILLS, C. Generating a privacy footprint on the internet. In *ACM IMC* (2006).

[11] KRISHNAMURTHY, B., AND WILLS, C. Privacy diffusion on the web: A longitudinal perspective. In *ACM WWW* (2009).

[12] LABOVITZ, C., LEKEL-JOHNSON, S., OBERHEIDE, J., AND JAHANIAN, F. Internet Inter-Domain Traffic. In *ACM SIGCOMM* (2010).

[13] LI, W., MOORE, A., AND CANINI, M. Classifying HTTP traffic in the new age. In *ACM SIGCOMM Poster* (2008).

[14] MAH, B. An empirical model of http network traffic. In *IEEE INFOCOM* (1997).

[15] SCHATZMANN, D., MÜHLBAUER, W., SPYROPOULOS, T., AND DIMITROPOULOS, X. Digging into HTTPS : Flow-Based Classification of Webmail Traffic. In *ACM IMC* (2010).

[16] SCHNEIDER, F., FELDMANN, A., KRISHNAMURTHY, B., AND WILLINGER, W. Understanding online social network usage from a network perspective. In *ACM IMC* (2009).

[17] SCHNEIDER1I2I, F., AGER, B., AND MAIER2I3, G. Pitfalls in http traffic measurements and analysis. In *International Conference on Passive and Active Measurement (PAM)* (2012).

[18] SMITH, F., CAMPOS, F., JEFFAY, K., AND OTT, D. What tcp/ip protocol headers can tell us about the web. In *ACM SIGMETRICS Performance Evaluation Review* (2001).

[19] TRESTIAN, I., RANJAN, S., KUZMANOVI, A., AND NUCCI, A. Unconstrained endpoint profiling (googling the internet). In *ACM SIGCOMM* (2008).

[20] XU, Q., ERMAN, J., GERBER, A., MAO, Z., PANG, J., AND VENKATARAMAN, S. Identifying diverse usage behaviors of smartphone apps. In *ACM IMC* (2011).