Networking 2013 1569696597

ReSurf: Reconstructing Web-Surfing Activity From Network Traffic

Guowu Xie*, Marios Iliofotou†, Thomas Karagiannis‡, Michalis Faloutsos* and Yaohui Jin§
*UC Riverside, {xieg, michalis}@cs.ucr.edu
†Narus, Inc., miliofotou@narus.com
‡Microsoft Research, thomas.karagiannis@microsoft.com
§Shanghai Jiao Tong University, jinyh@sjtu.edu.cn

***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 label a flow as web or HTTP. In this paper, we focus on understanding what happens "under the hood" of HTTP traffic. Our first contribution is ReSurf, a systematic approach to reconstruct web-surfing activity starting from raw network data with more than 91% recall and 95% precision over four real network traces. Our second contribution is an extensive analysis of web activity across these traces. By utilizing ReSurf, we study web-surfing behaviors in terms of user requests and transitions between websites (e.g. the click-through history of following hyperlinks). A surprising result is the prevalence of advertising and tracking services that are being accessed during web-surfing that are without the user's explicit consent. In our traces, we found that with 90% chance a user will access such a service after just three user requests (or "clicks"). We believe that our methodology and findings provide valuable insights into modern traffic that can allow: (a) network administrators to better manage and protect their networks, (b) traffic regulators to protect the rights of on-line users, and (c) researchers to better understand the evolution of the traffic from modern websites.**

I. 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 widely used as the interface to a large number of services and applications, such as Email, gaming, file sharing, and video streaming. Second, today HTTP is the most widely used protocol, contributing up to 80% of the traffic on some networks [1]. One implication of these trends is the limited relevance and applicability of traditional traffic analysis and characterization tools [2], [3]. 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 more and more 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. At the same time, extracting information from network traffic is needed by regulators who aim to protect the rights of the consumers and allow a healthy competition between content providers and between ISPs. In addition, analyzing web traffic is important for researchers who want to study modern websites and their evolution [4], [5].

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 behaviors. Here is a list of motivating questions: (a) What websites (e.g., `google.com`, `cnn.com`) are *explicitly* requested by a user as opposed to being accessed automatically by his browser 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.

Making the problem more specific, we can identify two subtasks: (a) group together HTTP requests generated by a single **user request**, such as a click, and associate them with the primary website requested by the user; and (b) reconstruct the **click-through stream**, i.e., the referral relationship between user requests, to identify whether a user's request to a website is from a hyperlink clicked on an earlier website or from within the same website. Figure 1 illustrates the above concepts. Understanding web traffic at both the user request and click-through levels provides insights into the user's web-surfing activity.

The problem of understanding web-surfing activity from network traffic has been studied before in [6], [4], [7], but not to the extent we do here. Schneider et al. [6] focus on reconstructing the browsing activity in social networking websites by using specialized features. The primary goal of those studies is to understand user behaviors in social networking websites and not to provide a generic methodology for reconstructing web-surfing activity from different sites. Closest to our work is the StreamStructure methodology proposed in [4], which utilizes the web analytics beacons generated by tracking services. By relying on tracking services, their approach can identify websites that do send beacons, which on average decreases the coverage by 40% (see Figure 5). We extensively compare ReSurf with StreamStructure in Sec-

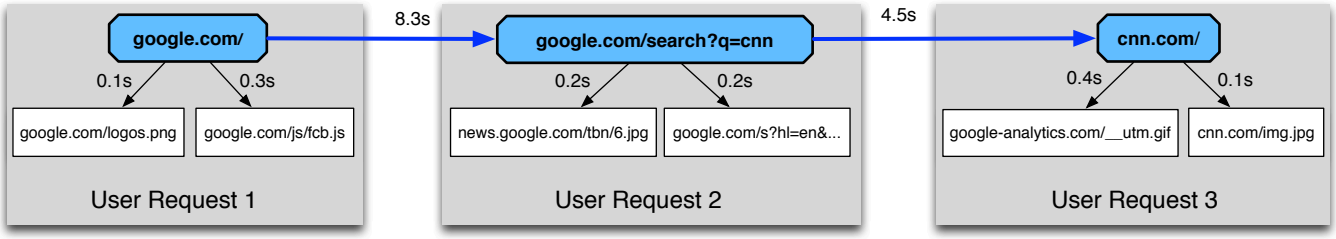

Fig. 1: An example of a click-through stream with three user requests (large grey boxes): (a) visiting `google.com`, (b) querying Google for CNN, and (c) following `cnn.com` from the results of the query. Each user request generates one head HTTP request, and each such request generates zero or more embedded HTTP requests, shown as two requests in this example.

tion III and discuss related work in more detail in Section V.

In this paper, we make two main contributions:

(A) ReSurf: reconstructing web-surfing activity. We develop a systematic method to reconstruct user requests and their relationship in click-through streams. First, we create a graph that represents the referral relationships between HTTP requests. Second, we identify the HTTP requests that are generated explicitly by the user, such as clicking on a link or typing a URL in a browser's address bar. Finally, we reconstruct the user request by grouping "subordinate" HTTP requests that are generated due to other requests, such as the acquisition of a video, an image or a web advertisement.

(B) Extensive measurements and validation. Our experiments with real data traces and our validation with both real and synthesized data provide the following highlights: (i) *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, while we highlights the limitations of the state-of-the-art methods that rely on web analytics beacons [4]; (ii) *Web users are continuously being exposed to advertising and tracking services*. We observed in our traces that 50-60% of user requests (i.e., clicks) trigger an interaction to a tracking or advertising service. In fact, we observed that the chance of a user triggering such as service after just three user requests is close to 90%; (iii) *Click-through streams are surprisingly "shallow"*, as the median number of websites in a click-through stream is one or two, and only 5% of the click-through streams have more than three websites.

II. Terminology, Problem Definition, and Datasets

We present the necessary background, previous work, and the web traffic traces used in our study.

Terminology: We use the term **user request (UR)** to describe a single user action, such as clicking on a hyperlink, visiting a page from one's bookmarks, or submitting a web-form, which includes a search engine query. Figure 1 depicts three user requests represented by big gray boxes: one to Google homepage, a Google query, and one to CNN, via clicking a hyperlink returned by the query. A user request generates one main HTTP request, which we refer to as the **head HTTP request** indicated by the colored octagons in the figure, and it usually accesses an HTML or XML file. In practice, the head HTTP request will often generate more HTTP requests that acquire different **web-objects**, such as images, videos, or javascripts. We call these subsequent requests **embedded HTTP requests**, which happen without the user explicitly requesting them. In the example, we indicate these requests as white boxes (i.e. `google.com/logos.png`). Note that embedded HTTP requests can obtain objects from different websites, such as ad servers and content distribution networks (CDNs). We use the term **primary website** to indicate the website requested by the head HTTP request, which also represents the website that the user intends to visit.

A **click-through stream** (CTS) is a series of consecutive user requests, where the later user requests in the stream are follow-ups of the earlier user requests ones. Figure 1 shows an example of a click-through stream with three user requests, as we discussed above. The Google query, which is the second user request, would not have been possible without the user having visited Google first, in the first user request. Subsequently, the third user request is generated by the user's clicking on the returned results of the Google query.

Problem definition: Given web traffic collected at a network link (think HTTP packet headers), we want to reconstruct the web-surfing behavior. We want to identify head HTTP requests, and correctly group their associated embedded requests. Simply put, in our example in Figure 1, we want to identify the "grey boxes". Once we do this correctly, we can identify which websites users visit explicitly and how much traffic is generated by each visit.

When analyzing web traffic, we have to work with information available in the HTTP requests and responses. In Figure 2 we give an example of only 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 triggered 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 HTTP headers in isolation, it is hard to track which visits they originated from.

An important piece of information is the **referrer** field in an HTTP header (visible in Figure 2). This field shows which

previous HTTP request triggered the current HTTP request. Moving back to the graph of Figure 1, in "user request 1" the request for `google.com/logo.png` has `google.com` in its referrer field. In the same example, when a user visits `cnn.com` by clicking a link in the Google search results, the referrer field of the resulting HTTP request indicates the Google search result page as the origin. In Section III, we show how we can carefully combine the HTTP requests as a graph and use it to identify head HTTP requests.

Challenges: Correctly attributing individual HTTP requests to a user request and to the primary website is quite complex. First, when looking at HTTP transactions in isolation, it is hard to know which website the user intentionally visits. In our traces, if we use the `host` field in HTTP headers to identify the primary website, it results in only 20-40% accuracy. We can see this from Figure 2 where only one of the three requests has the intended website (`www.cnn.com`) as the host name. Second, users often browse multiple websites at the same time, which causes flows and HTTP requests to intermingle. Modern web pages are fairly complex [5]; rendering a single page may generate tens of HTTP requests towards different web servers. In Figure 1, for clarity, we keep only a subset of web objects. In reality, even within few minutes the size of the referrer graph reaches several hundreds of nodes. In our traces, the median size of the referrer graph over a ten minute interval is 200 nodes for an IP address. Third, many websites, such as CDNs, web-ad servers, and web analytics services are used by many websites and shared across several services.

Data sets: The web traffic traces used in our study are summarized in Table I. Our traces cover several thousands of millions HTTP requests over long periods of time. They include an ISP link trace, two university traces of different sizes, a mobile traffic trace and a synthesized trace in a control enviroment. 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: researchers in a 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 patterns between different users. Details regarding the exact locations and the names of the providers for all our traces are intentionally kept anonymized due to privacy concerns.

We use the same traffic collection methodology for all the traces and capture all the IP packets on TCP ports 80, 8000 and 8080 in both directions. More details can be found below.

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 spanned 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.

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 mobile devices, such as smart-phones and tablets.

CAM: The CAM trace was collected from a university campus in China, containing the browsing activities of about 28.2K users. Our monitor device sits on the edge gateway connecting the campus to the public Internet. All download and upload traffic from the whole campus goes through the monitor point. Due to the amount of traffic, we did not store raw IP packets, instead logged all important HTTP header fields for all HTTP transactions. Specifically, the fields includes: 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 days of traffic are very similar. So we only show the results for one weekday in the paper.

SYN: The trace was generated in a controlled environment for the purpose of evaluating ReSurf. We generated the traffic by replaying nine volunteers' Google Chrome browsing history. We first extracted the timestamp, referrer and URL field of each visit from their browsing history. Then, to establish the ground truth, we replayed each visit using the following procedure. We instructed Google Chrome to open each URL in isolation. At the same time, we collected all the traffic on TCP port 80, 443, 8000 and 8080 using a packet capturing software (tcpdump). After 60 seconds, we closed the browser and saved the captured HTTP traffic to an individual file. To emulate how the traffic would be if it came directly from the user's surfing activity, we carefully adjusted the time stamps and referrer fields of HTTP traffic according to Chrome's browsing history. After replaying all visits, we merge all these individual files to form a complete traffic trace. Since each visit was collected and stored separately, we effectivity have the ground truth for each HTTP request in the trace.

III. The ReSurf Approach

Here, we present the ReSurf methodology, evaluate and compare it with existing solutions. Finally, we discuss the practical issues and limitations of our method.

A. The ReSurf Methodology. The goal of ReSurf is to group HTTP requests into user requests (see definition in Section II). Our approach works in two steps. First, we identify the *head HTTP requests* by using different features from each HTTP request. 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 referral relationships (see definition in Section II) to assign all the embedded HTTP requests to their corresponding head request. We explain the methodology step by step below.

Step 1. We form the HTTP referrer graph. We represent the HTTP requests from the same client IP address as a `referrer graph`. ReSurf builds such a graph for each IP address over a period of time (e.g., every ten minutes). An example of such a graph is shown in Figure 1, with the

(1) Head request to cnn.com

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

(2) An embedded request caused by (1)

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

(3) An embedded request caused by (2)

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

Fig. 2: A simplified example of HTTP request headers issued by a browser during a visit to `cnn.com`.

Name	CAM	LAB	ISP-1	MOB	SYN
Starting date	Mar 9 2012	Oct 3 2010	Aug 25 2011	Jan 7 2011	Aug 11 2011
Duration	2 mon	6 mon	24 h	3 h	1 mon
# of HTTP transactions	19B	1.2 M	1.7 M	22.9 M	186K
# of Clients (IPs)	28.2K	21	359	3,521	9
Ground truth available	No	No	No	No	Yes
Payload	HTTP header	Full	Full	Full	Full
Users	Students and staff in an university	Graduate students in a CS lab	Residential users	Smartphone, tablet users in 3G networks	-

TABLE I: An overview of the web traffic traces used in our study.

exception that we don't know which HTTP requests are in which user requests (i.e. shown as grey boxes), until we use our method. We provide a very high level description of this graph creation. For generating the graph, we use the `referer` field from each HTTP request to identify the previous request that triggered the request to the current website (as indicated by the `host` field). We generate a directed graph that captures these referral relationships and we enrich it with edge weights that represent the time difference between the two requests. Figure 1 shows an example of the HTTP referrer graph of a user accessing `google.com` and `cnn.com`. The nodes in the HTTP referrer graph are web-objects annotated with their complete URL. The directed edges capture the referral relationship between nodes where a directed edge from A to B means A is B's referrer.

In practice, the construction of the referrer graph hides many subtleties. First, character case, URL encoding and the presence or absence of trailing slashes are very common in HTTP headers. For the ease of string matching between referrer and URL, we unify character case, unquote URL encoding and strip all trailing slashes. Second, HTTP redirection (HTTP status code 302) complicates the construction of the referrer graph. We remove 302 HTTP redirections by combining the endpoints of the HTTP redirection in referrer graph into a new super node. Third, some web objects may have empty referrers. From our experiments, we learn there are two major reasons for such cases. (a) A Flash player plugin in Firefox has a well-known bug that does not append the referrer while requesting flash objects. (b) For some objects whose URL are dynamically generated by javascripts, the referrer fields in their HTTP requests are empty. Parsing the javascript file can prove useful for identifying the referrer, but only if payload is available. Without using payload, there are two possible solutions for handling these HTTP requests with empty referrer. The conservative one is to simply label them as unknown. The aggressive one is to attach these to their closest HTTP requests. In this paper, we use the conservative strategy and opt for precision.

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

(a) The candidate should be an HTML/XML object.

(b) Since most modern web-pages are fairly complex, the size of candidates should be larger than V bytes and candidates also should have at least K embedded objects.

(c) The time gap between candidates and their referrers should be larger than a threshold T. The reason is that head requests are usually further away from their referrers in time since they are initiated by users. By contrast, embedded requests are very close to their referrers because they are automatically initiated by browsers.

Step 3. We finalize the identification of the head requests. We utilize the referral 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 has no referrer. In the referrer graph, nodes with no referrers have no incoming edges. In Figure 1, the `google.com/` in "User Request 1" is an example of such a node with no referrer. Such nodes are formed when a user, say, opens a web pages from a browser bookmark or by directly typing the URL in the browser. If the referrer is not empty, it means the user navigated to a web page by following the links from a previous web page. This implies that the referrer of a head request should also be a head request.

Step 4. We assign embedded HTTP requests to head requests. ReSurf associates embedded HTTP requests to head requests by utilizing the timing information and referral 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. If a node has no referrer and is not a head request, it is labeled "unknown."

Below we will show that ReSurf outperforms the current state-of-the-art [4], and provides high classification precision and recall.

B. Evaluation. 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 its precision is 99%, 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 embedded 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 seconds, V=3000 bytes and K=2 embedded objects. 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 SYN, and (b) using the labels from a classifier that is based on web analytics beacons.

(a) Validation using ground truth from the SYN trace. In SYN 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 II. Figure 3 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.

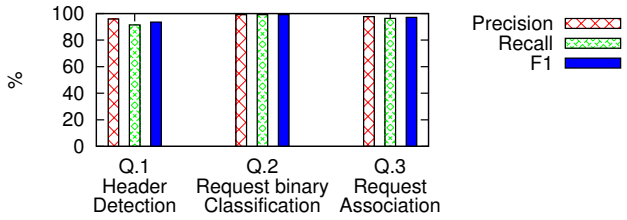

Fig. 3: The precision, recall and F1 score in the SYN trace.

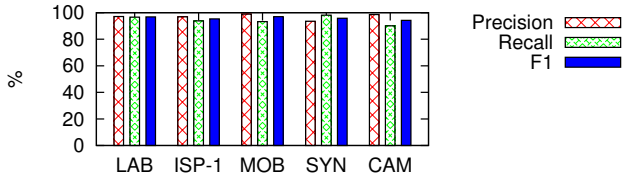

Fig. 4: The precision and recall for detecting head requests (Q.1)

(b) Validation using web analytics beacons as ground truth. For the CAM, LAB, ISP-1 and MOB traces, we do not have the ground truth. Therefore, we evaluate ReSurf based on the predictions given by the *StreamStructure* [4] 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 which page is visited. 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`.

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, beacons' URIs encode the URI of the tracked object/page. Therefore, after some careful parsing of beacon's URI, we can identify the primary website of the user request. We refer the reader to [4] 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 beacons to identify as many head requests as possible. We refer to this set of identified head requests as S. Then, we compare how well ReSurf performs over the known

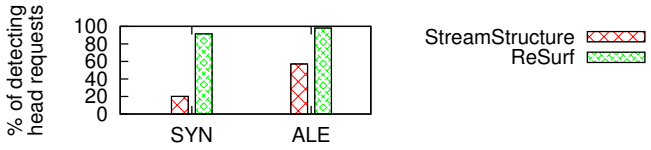

Fig. 5: The recall for detecting head HTTP requests (Q.1) using StreamStructure and ReSurf.

set S. Figure 4 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.

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 discuss one of identified limitations here. **The majority of user requests (∼80%) do not have a beacon in our data traces.** In all our traces, we find that less than 21% of the user requests that were found by ReSurf have beacons. Given the precision and recall of ReSurf in the controlled dataset SYN, we are confident that this percentage is reasonably accurate estimate of requests in the other traces. To further verify this, we used the SYN trace, for which we have the ground truth, and observe that only 23.9% of them carry a beacon. In the ALE trace, the ratio of user requests with beacons is roughly 60%. However, note that ALE does not represent surfing patterns from real-users, and only covers the popular website homepages as reported by Alexa. To summarize, we observed that beacons can only successfully identify approximately 21% of the user requests, compared to above 91% we achieve with ReSurf.

Figure 5 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 CAM, LAB, ISP-1 and MOB 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*.

Using *StreamStructure* and ReSurf on the same trace results in different result. In Figure 6 we plot the distribution of user requests to the top websites for the CAM trace. We see that the reduced number of identified requests by *StreamStructure* leads to different results. For instance, the top website with ReSurf corresponds to 18% of all user requests, whereas the same value for *StreamStructure* is 8%.

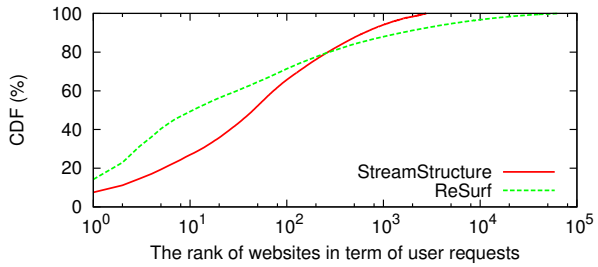

Fig. 6: The user requests to top websites using StreamStructure and ReSurf in the CAM trace.

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 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 primary web-site 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 as head 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 F1 score changes when 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.

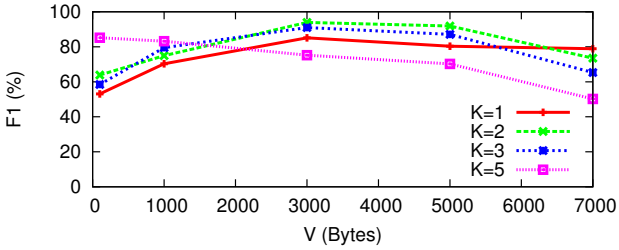

Fig. 7: The F1 score of detecting head requests (Q.1) with different parameters.

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 these setting: T=0.5 seconds, V=3000 bytes and K=2 embedded objects. Finally, our observation interval for generating the referrer graphs and performing our classification is 10 minutes. We experimented with different intervals in the range of few seconds up to 30

minutes and we observe better results and faster computations with windows in the range of 5 to 15 minutes.

C. 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 cannot classify those flows. However, by analyzing our real-word traces (see Table I), we observed that the encrypted traffic only amounts for 2% to 8% of the total web traffic. 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 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 request 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? We have not applied and tested ReSurf in real-time classification. From the offline experiments with our current implementation, ReSurf can classify ten minutes of one thousand users' HTTP traffic in 5-8 minutes. However, Step 1 requires the collection of traffic for several minutes before ReSurf can analyze the referrer graph and classify the requests. Therefore, ReSurf can only classify requests a few 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.

IV. Using ReSurf on real web traffic

We now use ReSurf to group HTTP requests into user requests and analyze how users behave in our four web traffic traces. We focus on three main directions:

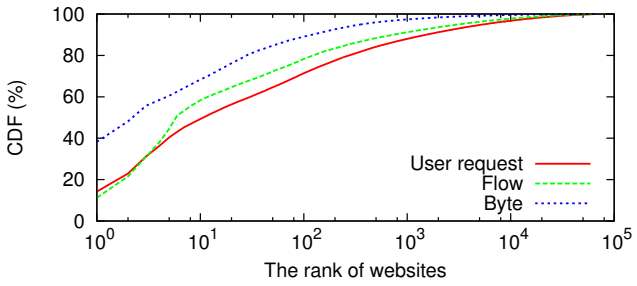

Fig. 8: The top websites in the CAM trace in terms of user requests, flows, and bytes. The x-axis (rank) is in logarithmic scale.

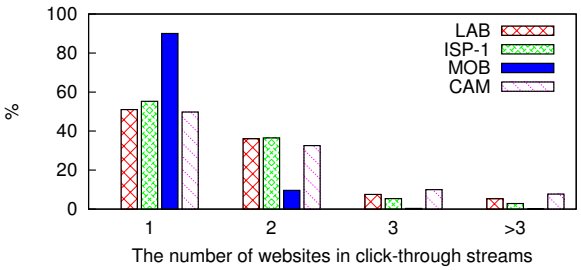

Fig. 9: The number of user requests in click-through streams (CTSs).

A. We study the popularity of websites in terms of user requests.

As expected, in all our traffic traces we observed that some websites are more popular than others. Figure 8 shows the cumulative share of user requests, bytes, and flows for the top websites in the CAM trace. We observe that the byte-curve is more skewed compared to the others, with the traffic to the top website (`115.com, an online file sharing website`) accounting for 40% of the total traffic in bytes. The corresponding cumulative share of flows and user requests for this top site is lower at 16% and 17% respectively. In our traces, we found the flow and user request rankings to be similar; which is also supported by how close the two curves are together in Figure 8. That is, the top websites in terms of flows are usually the websites that have the most user requests and vice versa. Intuitively, the more user requests a website receives, the more flows will be generated.

We also observed the traffic volume of a website depends on the content it distributes. That is, the top websites in terms of bytes usually are multimedia streaming websites, such as `youtube.com`, and sites featuring adult-content video. On the other hand, the top websites in terms of user requests and flows correspond to social networking and search sites, such as `facebook.com`, `baidu.com`, and `google.com`. To give an example of how different the byte ranking and user request ranking for different website are, in the CAM trace `115.com` ranks the first in terms of bytes and 16th in terms of user requests. It covers a remarkable 40% of the byte volume and only 2% of the number of user requests. Our observations suggest the existence of two categories of websites in terms of the traffic they generate: (a) high traffic volume sites, and (b) high flow/user request sites.

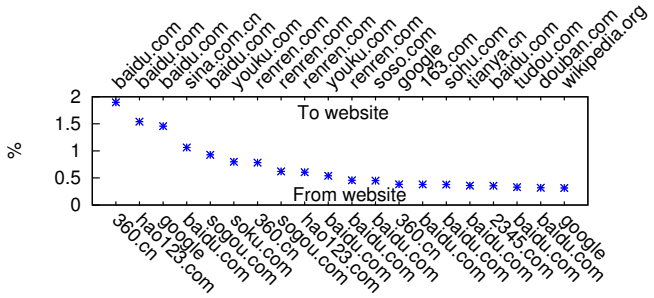

Fig. 10: The top website transitions for the CAM trace.

B. *We follow the click-through streams of users and study modern web browsing behaviors.*

In Figure 9 we show the distribution of the number of websites traversed in a single click-through stream (CTS). The definition of CTS is given in Section II. Here we assume that a CTS ends after an inactivity period of 30 minutes [1]. Surprisingly, the vast majority of CTSs span only up to two different websites. In fact, the percentage of CTSs with more than three websites is less than 5% in all our traces. These observations suggest a browsing behavior that is "focused" on a particular task. For example, the user starts with a particular goal in mind, searches for something on a popular search engine, and stops when he clicks on the correct link that takes them to that website they are looking for. This behavior is further supported by the most popular external referrals shown in Figure 10. Our measurements show that "referrer" websites are usually web portals, search engines, and social networking, while "referred-to" websites are content providers, like news sites, online video, and information sites (e.g., wikipedia).

Mobile traffic: Finally, we want to highlight that the "focused" browsing is more prevalent in mobile traffic where we see in Figure 9 that 98% of CTSs have a maximum length of two. We believe this is due the fact that the lower bandwidth available to these users deters them from initiating long browsing sessions that span over multiple sites.

C. *We study the exposure of users to advertising (ad) and tracking services*

Our goal is to determine the percentage of user requests in our traces that involve: (i) advertising (ad) services, (ii) tracking services (i.e., web analytics beacons), or (iii) either an ad or a beacon. We summarize the results of this study in Figure 11, where we show the percentage of user requests to web pages that have at least one ad or trigger at least one analytic beacon to a tracking site. In order to identify popular ad services we use the popular keywords and patterns compiled by open source Ad-blocking software [8]. To identify beacons, we use the approach described earlier in Section III. From Figure 11, we see that 18-36% of all user requests, depending on the trace, involve tracking services, 40-50% of them directly access at least one ad, and 50-60% access either

[1]We experimented with inactivity periods of 5-60 minutes and we observe similar results.

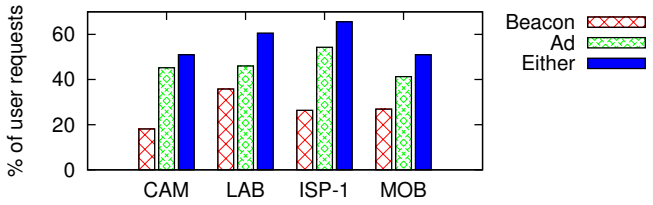

Fig. 11: The percentage of user requests to web pages with beacon, ad, and either in our four traces.

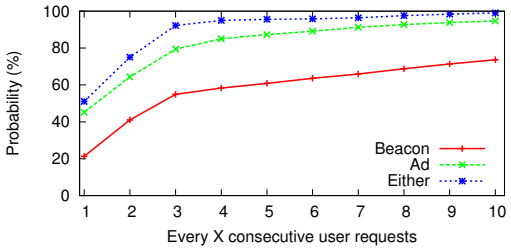

Fig. 12: The probability of encountering at least one beacon, ad, and either in every X consecutive user requests in the CAM trace.

a beacon or an ad. This suggests that more than half of the user requests (e.g., clicks) in the web traffic involve some form of advertisement or tracking!

We take this study one step further and try to understand the probability of a user encountering ads, beacons and either as a function of the number of user requests (e.g., clicks) that he makes. For this experiment, we treat each distinct IP address as a single user and use ReSurf to create a sequence of consecutive user requests for each IP. In the CAM trace, IT regulations enforce that no NAT is being used, which increases our confidence about IPs used by a single user at any time. Then, we randomly select X consecutive user requests from all IPs and caculate the probability of them containing at least one ad, beacon, and either. We summarize these results for the CAM trace in Figure 12. We see that, the probability of encountering an online ad or beacon after three clicks is close to 90%! We repeated the same experiment for the other traces and observed very similar trends even for our mobile trace. We also want to stress the fact that Figure 11 represents user requests from all users (IPs) grouped together, whereas Figure 12 focuses on what happens to different users.

V. Related Work

The recent growth of network services provided over HTTP has been attracting the interest of the research community. Labovitz et al. [1] brought to light the fact that most inter-domain traffic is HTTP. Schatzman et al. [9] present a methodology to identify web-based mail servers, and distinguishing between services, such as Gmail and Yahoo mail. Erman et al. [10] 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. [11] present methods to identify the type of the object transferred over HTTP (e.g., video, xml, jpeg). The recent work from

Schneider et al. [12] characterize the inconsistencies between observed HTTP traffic and what is advertised in its HTTP header. Bermudez et al. [7] understand the tangle between the content, content providers, and content hosts based on DNS traffic. All this previous work try to understand HTTP traffic from different perspectives, but they do not focus on the reconstruction of web-surfing at the user request level from HTTP traffic as we do here.

Regarding the problem of reconstructing web-surfing activity, existing work usually fall into one of four categories. In the first category, people assumes that any HTTP request for an HTML object is the head HTTP request of a user request [13]. The second category of methodologies are based on the timing information of HTTP requests: if the idle time between two HTTP requests is smaller than a predefined threshold, they belong to the same user request [14], [15]. Both methods were effective in the early days of the web, but are no longer due to the complexity of modern WWW. The third category is based on web analytics beacons. A representative example is `StreamStructure` in [4]. To understand the evolution of modern web traffic and web-pages, Ihm et al. proposed a web analytics beacon based method to detect "primary" web pages requested by users from web proxy server logs. A key limitation of `StreamStructure` is its dependence on web analytics beacons, which seem to form the basis of their reconstruction algorithm without which accuracy drops significantly. Recall that as high as 80% of user requests may not contain any analytics beacons (see Figure 5). The last category relies on the patterns in the requested URI. E.g., Schneider et al. [6] proprosed to reconstruct user requests (user actions) in several online social networking sites by matching previously compiled patterns with URI. This kind of method is customized for target websites and hard to be generalized because of various website architectures.

The click-through streams are also studied in the previous work. To understand the behavior of searching the Web, Kammenhuber et al. in [16] extract issued search term, search results returned from Google and the subsequent clicks on results from network traffic traces. The reconstructed sequences of clicks after the query is used for profiling users prevalent search patterns. Schneider et al. [6] extract click streams within online social networks from HTTP traffic to characterize the interaction between users and social network websites. However, both studies characterize user browsing behaviors on a specific kind of websites. Instead, our study shows user browsing behaviors in general.

Web analytics beacon is pervasive in modern web pages [17]. Krishnamurthy et al. [18], [19] study the privacy issues arising from the wide use of web analytics beacons in web pages. Ihm et al. in [4] show web analytics beacon can be used to detect pages and understand modern traffic.

VI. Conclusions

Network traffic has been increasingly dominated by web traffic and HTTP protocol has become the most prevalent means for applications to provide their services. In this paper, we framed and addressed a relatively novel problem: reconstructing web-surfing behavior from web traffic. The problem is far from trivial given the complex and interconnected websites of today. We made two key contributions. The first is developping a systematic approach 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 using ReSurf. We observed that web users are continuously being exposed to advertising and tracking services and that in our traces 50-60% of user requests (think clicks) interacts with tracking or advertising services. Another surprising result is the "shallowness" of the click-through stream of users accessing websites. The the majority of streams have the maximum length of two. This behavior suggests a more focused usage of the web, where users have a specific goal in mind and are less likely to click on links that take them to irrelevant websites. In conclusion, we believe that ReSurf represents an enabling capability for ISPs, network administrators, and researchers that want to model and understand how users surf the web.

References

[1] C. Labovitz and et al., "Internet Inter-Domain Traffic," in *ACM SIGCOMM*, 2010.

[2] T. Karagiannis and et al., "BLINC: multilevel traffic classification in the dark," in *ACM SIGCOMM*, 2005.

[3] I. Trestian and et al., "Unconstrained endpoint profiling (googling the internet)," in *ACM SIGCOMM*, 2008.

[4] S. Ihm and et al., "Towards understanding modern web traffic," in *ACM IMC*, 2011.

[5] M. Butkiewicz and et al., "Understanding website complexity: Measurements, metrics, and implications," in *ACM IMC*, 2011.

[6] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger, "Understanding online social network usage from a network perspective," in *ACM IMC*, 2009.

[7] I. Bermudez, M. Mellia, M. Munafò, R. Keralapura, and A. Nucci, "Dns to the rescue: Discerning content and services in a tangled web," in *ACM IMC*, 2012.

[8] "http://easylist.adblockplus.org/en/."

[9] D. Schatzmann, W. Mühlbauer, T. Spyropoulos, and X. Dimitropoulos, "Digging into HTTPS : Flow-Based Classification of Webmail Traffic," in *ACM IMC*, 2010.

[10] J. Erman, A. Gerber, and S. Sen, "HTTP in the home: it is not just about PCs," in *ACM SIGCOMM Computer Communication Review*, 2011. [Online]. Available: http://portal.acm.org/citation.cfm?id=1925876

[11] W. Li, A. Moore, and M. Canini, "Classifying HTTP traffic in the new age," in *ACM SIGCOMM Poster*, 2008.

[12] F. Schneider, B. Ager, G. Maier, A. Feldmann, and S. Uhlig, "Pitfalls in HTTP traffic measurements and analysis," in *International Conference on Passive and Active Measurement (PAM)*, 2012.

[13] P. Barford and et al., "Generating representative web workloads for network and server performance evaluation," in *ACM SIGMETRICS Performance Evaluation Review*, 1998.

[14] B. Mah, "An empirical model of http network traffic," in *IEEE INFOCOM*, 1997.

[15] F. Smith and et al., "What TCP/IP protocol headers can tell us about the web," in *ACM SIGMETRICS Performance Evaluation Review*, 2001.

[16] N. Kammenhuber, J. Luxenburger, A. Feldmann, and G. Weikum, "Web search clickstreams," in *ACM IMC*, 2006.

[17] D. Martin, H. Wu, and A. Alsaid, "Hidden surveillance by web sites: Web bugs in contemporary use," *Communications of the ACM*, vol. 46, no. 12, pp. 258–264, 2003.

[18] B. Krishnamurthy and C. Wills, "Generating a privacy footprint on the internet," in *ACM IMC*, 2006.

[19] ——, "Privacy diffusion on the web: A longitudinal perspective," in *ACM WWW*, 2009.