What is Wrecking Your Data Plan? A Measurement Study of Mobile Web Overhead

Abner Mendoza
SUCCESS Lab
Texas A&M University
mendoza@cse.tamu.edu

Kapil Singh
IBM Research
kapil@us.ibm.com

Guofei Gu
SUCCESS Lab
Texas A&M University
guofei@cse.tamu.edu

***Abstract*—The growing popularity of smartphones and continuous user demand for a rich web experience has resulted in an exponential surge in cellular bandwidth requirements. Cellular providers have struggled to keep pace with the new requirements while users often face a monetary cost associated with the data downloaded to their device. While many modern websites have adapted to the new mobile habitat, they often take shortcuts to transition from their desktop to mobile versions, many times carrying redundant content that is never utilized. Moreover, mobile users are effectively paying for certain undesirable content, such as advertisements, in the form of their bandwidth costs.**

In this paper, we study the composition and complexity of modern websites, from both a mobile and desktop perspective, to identify sources of wasted bandwidth. We developed a custom crawler-based framework to perform an in-depth analysis of the top 100,000 popular sites ranked by Alexa. Our results show that 23% or more of the content size on an average website is unnecessary, unused or redundant. Our results serve as a motivation for developing optimized websites and enhancing the web infrastructure to better suit the mobile environment with emphasis on reducing bandwidth costs, while also improving performance and efficiency.

I. Introduction

As mobile devices become ubiquitous and smartphones gain more popularity, the demand for bandwidth on the cellular infrastructure has risen to unprecedented levels. This surge in bandwidth demand is largely dominated by HTTP traffic from mobile browsers and mobile applications [1].

The increasing demand and consumption of higher bandwidth causes cellular networks to operate under severe resource constraints [2], which often translates into higher costs for users. This effectively means that the cost of each byte to the user must be increased in order to alleviate the increased burden on the cellular infrastructure. From the perspective of both users and service providers, there is a mutual best interest in reducing the overall bandwidth consumption.

The goal of this work is to have a deep understanding of the website components that contribute to useless, unnecessary bytes to be transmitted over the network and to develop a large-scale perspective of the amount of useless content that exist on the Web today. To achieve this goal, we perform a measurement-driven study to analyze a large set of popular websites from both a mobile and desktop browser perspective, to identify development flaws in today's real web pages that impose additional and often unnecessary data transfers resulting in higher bandwidth surcharge for users and the mobile infrastructure as a whole. For example, our measurements show that the average mobile website imposes an average total of about 923 kilobytes of data on each initial page load, with an excess of about 221 kilobytes of data consisting of content either unused or unnecessary for loading the web page. This represents a 24% overhead of additional bytes that have an adverse and significant impact to all stakeholders, both in terms of costs and performance.

While websites have evolved to cater to the mobile platform, their development has often happened in a piecemeal and ad-hoc fashion, in effect inheriting considerable amount of legacy code (such as JavaScript and CSS code) from their desktop counterparts. Moreover, developers have largely focused on quick development and rich user experience for mobile sites, without much consideration of the bandwidth cost for the end user. As a result, even though bandwidth-saving solutions such as code compression [3]–[5] are known, we find that they are not effectively leveraged by a large majority of popular sites. In this paper, we emphasize on the amplitude of the problem by showing that even popular websites are culprits for not filtering unnecessary content, resulting in considerable overhead for the users and cellular providers.

In our evaluation we crawled the top 100,000 websites as ranked by Alexa [6], using a custom mobile browser and desktop browser. Overall, our findings confirm that mobile websites are becoming increasingly complex, and there is a huge need for increased code optimization and localized caching mechanisms in the face of disappearing unlimited data plans and popularity of pay-as-you-go plans in several countries. We believe that different stakeholders, such as the web developers and advertisers, must take greater responsibility in reducing the burden on the end users and the cellular infrastructure. Unfortunately, we found several reputed websites, such as `facebook.com` and `amazon.com`, serving content with substantial amount of unnecessary overhead at 29% and 38% respectively.

In summary, this paper makes the following contributions:

- Analysis of multiple sources of unnecessary content overhead in website designs.
- Large-scale measurement on top 100,000 Alexa-ranked sites to understand the prevalence of unnecessary web content on popular sites.
- Provides in-depth analysis of the content overhead on

desktop and mobile websites.

We show that desktop and mobile websites, while different in total size, are relatively similar in terms of the overhead of unused content downloaded by the client browser. While the overhead burden may be more severe on the mobile platform, the benefits of reducing overhead can apply for all websites, exclusive of the form factor for which they are targeted.

II. An Analysis of Unnecessary Overhead Sources

First, we present our methodology for identifying unnecessary content of a web page and enumerate page resources that contribute to such overhead.

A. Methodology

For a systematic analysis, we establish the following principle to guide our overhead measurements:

A piece of content is considered unnecessary overhead if it satisfies at least one of the following conditions: (1) it is not rendered by the browser, *or* (2) it has no visual or functional impact on the offerings of the web page, *or* (3) the content is not directly related to the page and presents limited to no value to the user.

We evaluated page resources against the three conditions and marked them as unnecessary if one of the conditions are satisfied. The next subsection enumerates the major resource categories identified as unnecessary by our analysis.

We use content size as the metric for our measurements, where size is calculated at the user's browser once the content is downloaded. A similar approach is taken when deciding how to optimize websites for performance [7]. Indeed, our work is related to performance measurement studies, but with several key differences. Most significantly, we do not consider timing and latency related measurements due to the nature of our crawling infrastructure (Section III). Latency measurements in our context would not truly simulate real user experiences on cellular networks. Instead, we focus on the content size of assets transferred over the network and analyze the extent of the usage by the browser of all downloaded assets. Our goal is to identify the total added bandwidth consumed by data that is redundant, often unnecessary, and nonetheless costly to the user. We use the term bandwidth loosely in this paper to refer to the data size, but largely ignoring the rate measurement usually considered in bandwidth measurements.

B. Resources contributing to Unnecessary Overhead

Modern web pages include a rich variety of content types that include both static data (such as images) and executable code (such as JavaScript). For our analysis, we focus on six major components of a website – Images, JavaScript, Fonts, CSS, HTML and Cookies – since our evaluation shows that, on average, they make up to 98% of the total size on mobile websites, and 92% of the total size on desktop websites. We enumerate various sources of unnecessary content based on analysis of these major page components.

1) Image Overhead: Images are the major contributors to the total page size with contributions of approximately 63% and 64% of the total page weight for desktop and mobile sites respectively (Section IV). Unfortunately, there is a major disconnect between the quality of images that websites are offering and what the end devices can effectively render for enhanced visual experience for the user. Even though several image compression techniques exist, they are often not utilized by web developers and images continue to embed unnecessary meta-data that is transferred to the browser over the network.

2) Unreachable code in JavaScript and CSS: Development of many websites is often done in a piecemeal and ad-hoc fashion with new code modules being added while old ones are maintained for backward compatibility. Additionally, many websites leverage the same code base for both desktop and mobile versions that often results in code that remains included but never used. Moreover, they include conditional code based on device type, OS, browser, etc. A piece of code that is not meant for a particular device or condition would not be consumed, so it is useless for that particular device or condition.

Code coverage usually refers to the amount of code that has been tested prior to deployment and represents the amount of application code that is reachable. In the context of this study, we use the term code coverage to describe the amount of code that is executed by the browser in the process of rendering a web page. In our measurements, code coverage is analyzed for CSS code, and for JavaScript code, which in turn enables us to identify unreachable (and unnecessary) code.

3) "Double-Taxed" Advertising: Advertisements (Ads) are a special case in the context of identifying overhead in the Web. Advertisements are an important element of the web ecosystem, and help to sustain the availability of web services to users. Publishers rely on advertising revenue to maintain their websites. However, advertisements also introduces additional bytes that must be downloaded for each website, which effectively represents an additional cost to the end users. This is double taxing for the users especially in cases where undesirable ads are forced on them. Additionally, cellular providers usually have no extra benefit from such advertising, while advertising inherently places additional pressure on their infrastructure.

4) Unused Fonts: Many websites use web fonts as a means to ensure consistent user experience across platforms and devices. Web fonts are included on a page as an external font file that is downloaded by the browser. These font files typically include a large set of possible fonts, or variations of the same font, even though the site does not consume all of them.

5) Comments and Whitespaces in HTML, JavaScript and CSS: Comments and whitespaces are important components for the web developer to maintain clean and well-understood code. While they are critical for the development stage, they have no value for the browser or to the end user, thus satisfying our first and second overhead condition.

6) Improper Use of Cookies: Modern websites leverage cookies for maintaining states across multiple requests. By default, cookies belonging to a website origin are automatically sent with any requests to that origin. Unfortunately, in many cases the server does not require all the cookies in order to serve the requested content. For example, if the request is for a static content such as an image file, session cookies are not required. Even if the cookie content size is small, the total data overhead due to cookie can be considerably large if the number of requests to the origin are high (Section IV-G).

III. Measurement Framework

We utilized a *web-centric* measurement approach in which we actively crawled the Web to measure unnecessary content hosted on popular websites. In order to extract the true representation of each website close to what would be seen by a real user, our custom crawler utilized the popular PhantomJS headless browser [8] to render websites including all dynamically loaded content. We ran our crawler using user-agent and viewport settings for both a mobile browser and a desktop browser, resulting in several million individual HTTP requests captured. Similar to previous works [9], [10], we focused our data collection to the landing page of each website. In addition, we simulated a few user actions to invoke additional content and functionality that may be hidden behind the landing pages, especially focusing on code coverage. We downloaded all assets loaded by each website, along with a record of all request and response details saved in an HTTP archive record (HAR) file [11]. Our dataset was accumulated over a span of several days in April of 2014.

A. Architecture

We built a customized framework used to crawl and capture all requests and responses, and subsequently analyze the raw data. A high-level architecture of the framework is shown in Figure 1. On each successful page load, an HTTP Archive (HAR) formatted file is saved that includes all the request and response details for the loaded page. The information within the HAR file is a comprehensive representation of the required transactions (request-response pairs) used for downloading the given page. In addition to the traditional HAR file contents, we modify our HAR capture code to include additional details, such as the nested frames and the actual page's HTML code. This gives us a better holistic representation of the entire page in a single file rather than multiple files. For a large-scale analysis, this facilitates easier parsing and automatic analysis of page measurements on a per-page basis.

Additionally, for each HTTP GET request, we downloaded each target file separately to disk. This is necessary because the PhantomJS browser itself does not provide a means for saving the downloaded assets. After each web page is processed, or timed out, we start a new instance of the browser from a clean state to crawl the next page.

The last major component in the framework is the JSCover proxy [12], which is a tool that enables us to measure code coverage for JavaScript. We performed our overhead measurement and analysis offline using various data files and reports collected during the crawling stage.

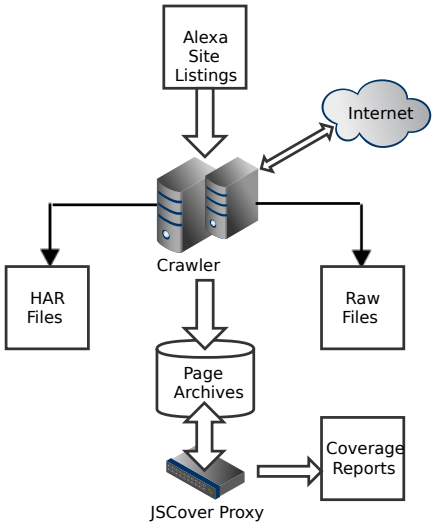

Fig. 1. Measurement Framework Architecture: Measurements are calculated from analysis of HAR files, raw resource files, and coverage report files.

User Simulation: In order to be more comprehensive in our measurements, and account for code execution triggered by user interaction on a website, our crawling framework analysis included automated user simulation. The automated user simulation actions includes code that performs scrolling, mouse events such as clicks and hover, link navigation, and other actions that may expose page's characteristics, such as triggering JavaScript execution. We were able to closely imitate user interactions and achieve good coverage using these simple simulations (Section IV-H).

B. Measurement Approach

We analyzed each web page with respect to various metrics that measures the impact on data usage on desktop and mobile devices. Some measurements were performed at the time the web pages were crawled, and saved into the HAR file. For example, we calculated content length at the time of crawling since HTTP headers sometimes report incorrect content length for a given response. This ensured that the HAR file is saved with the correct content length for each request. We also measured the content size online across different MIME types such as Image, JavaScript, CSS, HTML and Fonts. Other measurements, such as those resulting from code coverage and image optimization, are done offline after the websites are crawled. The framework is designed to automatically produce various sets of result files, as well as the downloaded website component files that can be analyzed offline.

1) Code Coverage: For JavaScript coverage, we used the JSCover tool [12] in order to facilitate our measurement of unused JavaScript code. JSCover modifies the JavaScript code that is served to the browser with code that records measurements for code that is executed. JSCover saves the

resulting coverage report to disk, and we are then able to parse these reports in order to infer and measure the lines of code not covered. For CSS, we measure the amount of unused CSS by extracting unused rules and subsequently analyzing the remaining CSS rules according the general CSS specificity rules that determine the rules applied by the browser.

2) Images and Fonts: For Images and Web Fonts measurements, we use compression tools that optimize the files by applying lossless compression. We measure the total overhead as the amount of file size reduction obtained by using compression tools to optimize the files while maintaining the fidelity of the files. For JavaScript, CSS, and HTML, we apply both compression tools that strips out whitespace and comments, and we perform dynamic analysis in the form of code coverage measurements to determine the specific portions of the code not utilized by the browser.

3) Advertisements: In order to measure advertisements, we first needed to detect what portion of each website represented an advertisement. We used the popular filter list used by browser ad blockers, known as EasyList [13]. Utilizing these filters, we are able to scan a website loaded in our browser, and extract the segments that are identified as advertisements to be able to measure and analyze their size overhead. In this sense, we treat Ads as an entity all on its own.

C. Crawling Data Set

We crawled the set of 100,000 most popular sites ranked by Alexa in order to collect a data corpus with enough coverage of representative websites. We crawled each website using both a mobile and desktop user-agent string as discussed previously, resulting in distinct datasets for mobile and for desktop websites. For each website, we set a connection timeout of 120 seconds and only captured the final landing site for any sites that had automatic redirection. In total, we captured a total of 95,718 websites each for both data sets, and a total of 12,142,743 individual requests for desktop websites, and 8,042,122 individual requests for mobile websites. Websites that were not captured included those that have no landing page, such as those from content delivery networks and advertising network servers. Additionally, some websites timed out or were not available at the time we attempted to crawl them. Overall, our data set represents 95.7% of the total site listing that we used.

D. Data Variance

One limitation of our study is the temporal invariance of our data set, which might not fully capture the dynamic nature of a web page. This is especially true when considering advertisements loaded on a page. On each load of a page, the final representation varies depending on several factors, including time, advertisements, HTTP headers, and other parameters. Website statistics show a steady growth trend in the overall page sizes over time [10]. Our study, however, is based on a snapshot in time of these web pages, with a goal of presenting a view of the general state of mobile and desktop sites.

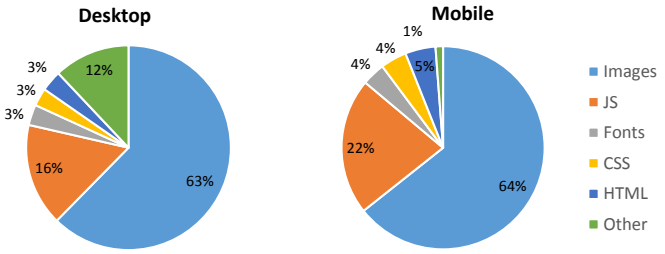

Fig. 2. Average object sizes per page page for mobile and desktop websites

IV. Experimental Results

A. Baseline Measurements

Analysis of the total average sizes of major website components gives us a general picture of the Web landscape in terms of website composition and content sizes. These measurements establish a baseline against which we measure the overhead percentage for each component.

While the focus of our work is mobile due to its higher per-byte data cost, the insights from this study can also equally benefit desktop websites design practices. Therefore, we compare mobile and desktop measurements to illustrate the correlation between the two platforms. Our measurements show that the average size of a mobile website is *923 kilobytes*, and the average size of a desktop website is *1741 kilobytes*. These measurement as similar to those gathered by HTTPArchive.org [10].

Previous studies have shown that images make up the majority of the total page weight on the average website [9], [14], [15]. This holds true for both mobile and desktop website as can be seen from Figure 2. We also see the prominence of JavaScript code in both mobile and desktop sites. Ihm et al. [16] attributes the increased size of JavaScript and CSS over the years to the increased use of Ajax functionality on modern websites. An interesting observation in comparing between mobile and desktop is that the size of the average JavaScript code on both platform is almost the same. Our results show that for desktop websites, JavaScript code accounts for about 283 kilobytes (16%), while on mobile websites we see an average of about 201 kilobytes (22%) of JavaScript code. While the prominence of JavaScript is to be expected [16], it is interesting to note that mobile websites use relatively more JavaScript code, which could give rise to concerns related to performance bottlenecks and overhead.

B. Image Analysis

Since images are the majority contributors to the page weight, their impact on data usage is significant. As can be observed from Figure 2, images take up about 63% and 64% of the total page weight for desktop and mobile sites respectively. We found that entertainment websites use a larger volume of images on both mobile and desktop websites, and finance websites have the least average image size. We measure the impact of applying compression and optimization on images to reduce the total footprint while maintaining quality. Despite

best practice recommendations, our analysis shows that images are rarely optimized by most websites.

Image types also play a considerable role in the overall image size. We found that three formats – GIF, PNG, JPG – tend to be the most popular, with average total size measurement (of all images) for each format per page being 97, 222 and 646 kilobytes for desktops and 42, 159 and 387 kilobytes for mobile. There is a negligible amount of images which we ignore that utilize other formats such as BMP, TIFF, etc. Each of the three major image formats utilize different compression techniques in order to optimize their size and quality and hence could be suitable for different target devices.

1) Overhead in Image Content: Web images are usually created using graphics tools that most likely do not save the final images in the most efficient manner. For example, many graphics tools include metadata information such as the tool name, author's name, etc. These snippets of extraneous information add to the total image size while having no effect on the actual image representation. Additionally, varying graphics tools may not use the most optimized algorithms to save the final image on disk, resulting in images that are not fully optimized. For example, PNG optimization usually requires selection of the best neighboring pixel depending on the actual image, but graphics tools may not fully implement such optimizations.

In terms of reducing bytes, one intuition is that converting GIF images to 8-bit PNG can result in reduced image sizes while still preserving quality, due to the optimized algorithm used for PNG images. We performed an analysis on all GIF images to get an idea of how much size savings could be realized by simply converting GIF images to PNG images. We used the OptiPNG image compression tool [17] to convert GIF images to PNG, and compared the size difference. We performed this analysis for a sampling of both mobile website images and desktop website images. On a straight conversion to PNG from GIF, we saw an average of 13% file size reduction. This represents a potential saving of about 13 kilobytes on desktop pages, and 5.5 kilobytes on mobile sites.

Next, we ran the compression tool on a sampling of existing PNG images and found a significant reduction in file sizes at about 31% on average. This applied equally for both mobile and desktop web images. The results show that a potential reduction of 69 kilobytes and 49 kilobytes can be realized on an average desktop and average mobile page, respectively, by simply performing PNG image optimization.

We performed a similar analysis on JPG images to compare the size reduction that can be realized by applying image optimization. The results on optimized JPG images show that a file size reduction of about 15% on average can be realized. Considering our average JPG content sizes per page, we can potentially reduce the size by an average of 97 kilobytes for desktop sites, and 58 kilobytes for mobile sites. These are significant savings on the total page weight that can be realized by simple image optimization.

2) Screen Quality Analysis: Mobile devices differ significantly in their screen quality. We use pixel density as a measure of the screen quality of the device and perform a preliminary evaluation of the available image quality compared to the screen quality. Our hypothesis is that a large number of high-quality images are served to mobile devices unnecessarily, resulting in wasted resources if the device cannot display the full quality of the image, or if the high-quality is not discernible from a lower-quality image on the same display.

Considering image dimensions compared to popular screen resolutions, we found that most images on a desktop website fit within the 1024x768 screen resolution. That is, we found that less than 1% of the images had a dimension that exceeded the 1024-pixel width. However, on mobile sites we found that a significant number of images exceeded the most popular 640x960 screen resolution. Specifically, we measured the width of all images on mobile websites and found that out of a total of 3.5 million images that were downloaded, 732,023 images (20.7%) had dimensions that exceeded the popular 640-pixel width for mobile displays. We further analyzed the reduction in file sizes of these identified images when the resolution was reduced to a minimum of 640 pixels while till maintaining the aspect ratio. This image resizing reduced the size of these images from an average 32.6 kilobytes to 23.2 kilobytes, i.e., an overall reduction of 28.8%. While there are a large number of varying display dimensions and resolutions in the mobile space, we present this simple analysis to illustrate the need for more targeted image-resolution management. For example, 320x480 display resolutions are still common place for mobile phones, which present an opportunity for further reduction of image file sizes.

Summary Our image overhead analysis showed that image optimization is lacking on most websites. From our average size measurements, we observe that a total of saving of about 263 kilobytes can be realized on desktop sites, and 121 kilobytes for mobile websites. These are significant size reductions that alone can reduce mobile website sizes by an average of 13.1%, and desktop website sizes by an average of 15.1%. The size reduction can be further enhanced if the image quality is appropriately optimized for the target screen quality.

C. Analysis of Code Usage

1) JavaScript Coverage: Based on our code coverage analysis, we found that up to 33% of JavaScript code remains unused on mobile web pages (Figure 3). This represents an average of 67 kilobytes for mobile web pages. On desktop pages, we found a higher average size of unused JavaScript code, at 82 kilobytes, which represents about 28% of the total JavaScript weight on desktop pages.

What contributes to unused code? From our analysis, we noted that in many instances unused code is attributed to blocks of code from JavaScript framework libraries. In our mobile dataset, we found that 58% of sites made use of some form of the jQuery framework. For example, a number of mobile sites use the jQueryMobile library, which includes pre-built functionality for a number of user interface layouts,

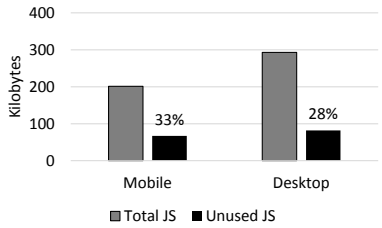

Fig. 3. JavaScript coverage results showing size of unused JavaScript code with user simulation

TABLE I
JAVASCRIPT LIBRARY INCLUSION OVER ALL SITES

Framework	Sites
jQuery	58%
Prototype	1%
Angular	1%
Scriptaculous	0.25%
Dojo	0.19%

events, animations, and other pre-built utilities. Of the average 67 kilobytes of unused JavaScript code, we found that around 75%, or roughly 50 kilobytes, can be attributed to the inclusion of JavaScript framework libraries.

The top 5 libraries we identified are shown in table I. Overall, we found that 60% of the total sites made use of one or more of these frameworks.

2) CSS Usage: CSS files account for an average of 38 kilobytes on mobile websites and 51 kilobytes on desktop websites. While CSS accounts for the least bytes in comparison with the other major components, there are still opportunities to identify redundant data. CSS in particular is notorious for being bloated with unused CSS rules. In terms of website performance, it is particularly important to reduce the size, and therefore the latency, of CSS files. The reason is that browsers typically do not begin to render a page until it has downloaded the CSS files and parsed all the rules. There are two main methods of reducing the size of CSS files. First, the CSS files can be compressed similar to JS files by using tools that strip out all comments, and whitespace. In our analysis we found that desktop CSS files achieve the most size reduction by simple compression at an average of 4 kilobytes. For mobile websites, we found the average reduction in size to be close to 1 kilobyte. One possible reason for this is that mobile website developers tend to use optimization techniques more extensively for performance reasons.

The second method of size reduction for CSS is through analysis of CSS specificity rules to determine the unused CSS rules. We performed an extensive analysis on CSS files through our instrumented browser to identify the unused CSS rules. On desktop sites, we found that an average of 78% of the total rules were not used by the websites. On mobile sites, we found that the average was 71% of the rules not used. Measuring the size of the unused rules we found that CSS files can be reduced to an average of 24 kilobytes on mobile websites, and 30 kilobytes for desktop sites. This is a significant size reduction, representing 38% reduction for mobile sites, and 41% reduction for desktop sites.

Summary With regard to the total overhead for unused code (JavaScript and CSS), our results show that a total savings of 85 kilobytes (9.2%) can be achieved on mobile websites, and a total savings of 104 kilobytes (6%) can be achieved for desktop sites.

D. Advertisements

We also quantify the contribution of Ad-related content to the total weight of a page rendered in a browser. In order to identify Ads, we used the same mechanisms used by browser extensions to block advertisements. These include regular expressions and other filtering mechanisms that allows us to extract Ads from within a page and measure the total size of all its components. Overall, we found that advertisements account for about 10% of the total page weight.

We found that ads consume an average of 75 kilobytes (8.1%) on mobile web pages, and 153 kilobytes (8.8%) on desktop pages. While these numbers represent a measurement based on a snapshot in time of each website, they nonetheless give us a broad picture of the Ad landscape in terms of how much of the page content may be reserved for Ads. While we do not advocate that advertisement are unnecessary, it is also important to consider the additional cost to the user for downloading advertisements when such downloads are not the main purpose of visiting a given website. For example, on mobile devices, advertisements contribute about 19 of the total 201 kilobytes of JavaScript code. This represents a significant 9% of the total JavaScript downloaded, which increases the load on the JavaScript engine and also contributes to degraded battery life [18]. If we consider the additive effects of browsing several pages, we can start to understand the adverse effects of ads from the user's perspective both in terms of data usage and energy consumption.

E. Web Fonts

Recent trends point to increasing adoption of web fonts across the web. Google, for example, provides a web service that serves free open source web fonts and reports several million font downloads across different font families. Web fonts consist of external font files downloaded by the browser and utilized on a website through CSS rules. In our dataset, we found that 31% of the websites downloaded at least one font file. In total, our mobile dataset contained 86,221 font files, and our desktop dataset contained 137,799 font files. The average size of all font files per site was 57 kilobytes on desktop sites, and 34 kilobytes on mobile sites.

We found that over half of all font files are not utilized by the browser in rendering the landing page, meaning that they present significant overhead in terms of page size.

We further analyzed the font files to determine possible optimizations that could be applied in order to reduce the total file sizes. We used the open source *sfntly* library from Google to apply compression to our font files. After optimization using the sfntly tool, font file were reduced by an average of 23

kilobytes (40.3%) for desktop sites, and 10 kilobytes (29.4%) for mobile sites. These represent significant overhead in terms of data usage.

F. HTML

While HTML code does not contribute as much to the page size as other components, there are still opportunities for optimization. Our analysis of HTML is limited to measuring the unused portions of HTML code, such as those not displayed due to CSS rules, or those tags that were empty or included no content even after user simulation actions.

In analyzing HTML code, we found that the average mobile site contained 44 kilobytes of HTML code, while the average desktop site contains a slightly larger 57 kilobytes. We found that on mobile sites, just about 4 kilobytes of HTML was unnecessary, either because they were hidden by CSS rules, or they were empty tags, such as empty div tags. On desktop sites, we found a higher size of unused HTML code at 11 kilobytes.

G. Additional Extraneous Data

1) Comments and Whitespace: In our analysis, we also measured the size of comments and whitespaces to determine their corresponding overhead. For comments, we measured all instances in JavaScript files, CSS, and HTML. For whitespaces, we measured all whitespace in JavaScript and CSS, as well as all whitespaces between tags in HTML code. We measured an average of 1.8 kilobytes of comments and whitespaces on desktop sites, and 0.78 kilobytes of comments for mobile sites. While not significant to the overall page size, comments and whitespaces contribute enough unnecessary bytes to have an adverse effect on the data consumption for end users.

2) Cookies: Similarly, we found that cookies can also negatively impact data consumption. In our measurements, we found that average cookie data usage measured per web page was 1.2 kilobytes for an average mobile website, and 15 kilobytes for an average desktop site. Our measurement shows that cookies are used much more extensively on desktop sites. This may be due to the added number of ad-related content on desktop sites as opposed to mobile sites. Cookies are most often used as a means of identifying users and building profiles for advertising and other purposes, and have been the subject of several privacy-related studies over the years [19] [20].

Since cookies are sent on each request to the same origin, a single byte of cookie originating from a given server can account for several bytes of data usage during page rendering. For example, in an extreme case of a desktop webpage, we measured a total of 360 requests on `lolnexus.com`, which is a site for video game related content. The site contains a number of ads and social networking widgets, which contribute significantly to the number of cookies loaded. In total, we measured 396 kilobytes of cookie data sent in request headers, and 290 kilobytes of cookie data received in response headers, for a total of 686 kilobytes of total data usage. This website does not have a mobile-optimized website, which means that a user on a mobile device would download the same amount of data just from cookies. On manual inspection of the cookie values for this particular site, we found that the vast majority were tracking cookies related to advertisements and social networking widgets. This extreme case illustrates the potential total data overhead imposed by ads and other third party content loaded on a website.

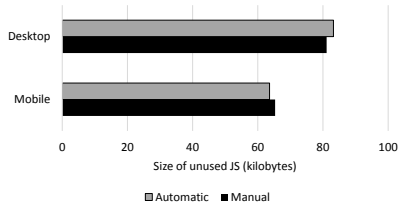

Fig. 4. Comparison of user-driven analysis vs. automatic simulation for 100 Alexa sites.

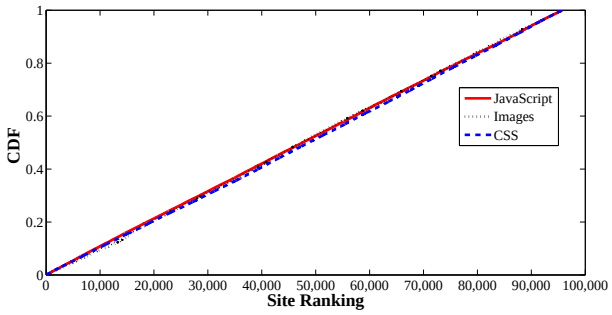

Fig. 5. CDF of total overhead measurement by site popularity ranking for JavaScript, CSS, and images on mobile sites

H. Methodology validation using user-driven analysis

In order to validate the automated simulations as being almost as effective as real user actions, we conducted a user-driven examination of 100 randomly chosen Alexa websites. To do this, one of the authors manually browsed and interacted with these sites using a browser proxied through the JSCover proxy. This was repeated for each website using both a mobile and desktop user-agent string, each time using a new private browser session to eliminate the influence of caching.

We then compared the results obtained through this manual analysis to those obtained using automated simulation for the same websites. Figure 4 shows the results of our comparison for the representative case of JavaScript coverage analysis. We observe that the JavaScript coverage results are almost comparable for both desktop and mobile, providing confidence that our automated simulation works well in practice.

I. Measurements Insights

The aggregated measurement results show that websites carry a significant overhead of redundant data. The average overhead measurements for each of the major website components are shown in Table II. Not surprisingly, images and JavaScript carry the most overhead of all components, contributing a total of 19.8% overhead for desktop websites,

TABLE II
TOTAL OVERHEAD MEASUREMENTS RELATIVE TO TOTAL PAGE WEIGHT OF 923 KB FOR MOBILE AND 1741 KB FOR DESKTOP

Component	Mobile		Desktop	
	Absolute	*Percentage*	*Absolute*	*Percentage*
Images	121 kb	13.1 %	263 kb	15.1 %
JavaScript	67 kb	7.3 %	82 kb	4.7 %
Fonts	10 kb	1.1 %	23 kb	1.3 %
CSS	18 kb	2.0 %	22 kb	1.3 %
HTML	4 kb	0.4 %	11 kb	0.6 %
Other	1 kb	0.1 %	2 kb	0.1 %
TOTAL	**221 kb**	**24 %**	**403 kb**	**23.1 %**

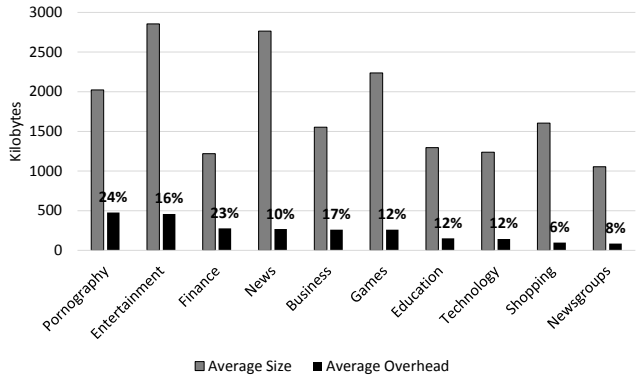

Fig. 6. Total overhead for mobile websites for top 10 categories

and 20.4% overhead for mobile websites. Overall, our results show that on an average websites suffer from a substantial overhead of *more than 23%*, thus motivating the urgent need to employ data optimization techniques.

1) Correlating overhead and site popularity: We consider how the popularity of sites correlates with the total overhead. The cost of overhead is higher for the cellular networks if higher ranked sites exhibit such overhead, as more people would visit them and download unnecessary content. Figure 5 shows a CDF of the overhead for JavaScript, CSS and images on mobile sites according to the sites' ranking. Interestingly, the site ranking seem to have little bearing on the overhead measurement, meaning that both higher and lower ranked sites have almost equal tendency to exhibit similar overhead measurements.

We also examined the overhead numbers for specific popular representative sites. Not surprisingly, we found that even highly-reputed websites are more-or-less equal culprits in serving web content with substantial overhead. While sites like `ebay.com` and `google.com` have overheads lower than our average measurements at 11% and 18% respectively, other popular ones such as `facebook.com` and `amazon.com` exhibit much higher numbers at 29% and 38% respectively.

2) Correlating overhead and site category: We analyzed the total overhead measurements across different website categories for the top 10 categories as shown in Figure 6. While there is not a clear relationship between site category and overhead, we observe that categories which are known to contain more images, such as Entertainment, carry a higher overhead.

V. RELATED WORK

Several previous studies have been done to characterize websites, both at the network level [14], [16] and the client level [9]. Most previous work focus on measuring the performance of web pages in terms of latency rather than data usage based on the downloaded content sizes. Our focus is to measure overhead from the client-side perspective.

Qian et al. [14] considers several strategies, including packet-level redundancy elimination as a means of reducing cellular traffic. Our work is focused on eliminating unnecessary content, as defined in Section II-A, in terms of what is necessary for the functionality of a web application, which in turn also serves to reduce redundancy at the network level to some extent.

Butkiewicz et al. [9] measured the complexity of web pages by characterizing the different components of web pages and measuring the performance impact based on latency metrics. Their measurements are based solely on desktop websites, and evaluated on a smaller number of sites. Industry initiatives such as the HTTP Archive project [10] is similar to our work in measuring and analyzing the size of different website components. Our work differentiates from these in that we make comparisons with mobile and desktop sites, and we also present a deeper analysis of each website's composition in terms of code coverage for JavaScript, CSS, advertisements, and other metrics not previously analyzed. Khan et al. [21] measured advertisement traffic associated with mobile applications and proposed a solution to mitigate the costs to consumers.

Longitudinal studies have previously characterized the dynamic nature of the web over extended periods of time [22] [15]. Ihm et al. [16] investigated the changing nature of web traffic over a span of five years, showing that Ajax and Flash traffic have steadily increased. We take these findings into account, especially with regard to traffic generated by client-side interactions, by simulating user interactions.

[16] also showed that loading times of web pages improved due to enhanced caching mechanisms and increased concurrent connections initiated by modern browsers. Our results show that although caching has improved over time, there is still limited cache capacity, and there is no value gained by caching extraneous data.

Thiagarajan et al. [18] presented an analysis of mobile phone battery consumption as a result of web browsing. They proposed optimizations similar to our work. Our results further highlights the urgent need for more aggressive optimization practices to increase mobile device energy efficiency, among other concerns.

Our work is motivated by the insight of previous work as we seek to fill the gap into understanding the overhead imposed by unoptimized code and data. Our evaluation of mobile websites and comparisons to their mobile counterparts extends the insight of the changing nature of the web landscape and the need for optimization on mobile and desktop sites.

VI. Discussion & Future Work

Our insight on the overhead imposed by websites are a clear indication that there is a dire need for more robust solutions aimed at web front-end optimization. It has long been known that the browser presents a significant bottleneck in terms of page load time and performance [7]. However, previous work in performance optimization is often at odds with the best interest of material cost of moving data across the network. For example, solutions such as SPDY [23], and HTTP2 [24] place an emphasis reducing latency by minimizing the number of HTTP requests, among other things. This often leads developers to concatenate several code files into one file, thereby reducing the number of HTTP requests, but without further consideration of any overhead in the code. Our results show that most of the included JavaScript functionality and CSS rules within these frameworks are never used by the web page and thus present a significant overhead both in terms of performance and data usage.

In the Apache web server, for example, the mod_concat module is used to concatenate files dynamically at run-time. From a purely performance viewpoint, this practice improves latency, page load time, and ultimately improves the user experience. On the other hand, we argue that this practice has a real cost to users who are inevitably forced to download a larger set of files, in terms of content size, while a large percentage of that downloaded content is actually left unused.

While some optimization tools, such as text and image compression tools, are available today, our results indicate that such tools are not being used often. There is a need for an comprehensive, automated solution that would optimize websites without solely relying on the websites. The insights gained from our measurement study could be leveraged while developing such a system. We plan to continue our work in this direction. We have developed a preliminary blueprint of such a solution that leverages a suite of optimization tools to automatically customize the best possible version of a website for the device and server conditions without sacrificing functionality or user experience. We are evaluating the suitable form (such as an adaptive, in-network cache) and location in the network path (such as at the ISP or the web hosting server) where such a system would be most effective. We believe that our proposed solution would complement previous works on web content adaptation: while most of the existing solutions focus on content transformation to adapt to the user agent properties such as display size and resolution, the primary contribution of our solution is to remove unused and redundant data that is unnecessary for the functionality of the website.

VII. Conclusion

In this study, we presented an analysis of the top 100,000 websites as ranked by Alexa, using a simulated mobile browser and desktop browser. We built a framework to perform an extensive analysis of the file sizes and overhead of the various website components. We compared and contrasted the differences in content sizes between mobile and desktop websites at a very granular level. For each website, we measured the size of assets downloaded to the browser and analyzed the size reduction that could be realized through various optimization tools and techniques without affecting the functionality of the site. Our analysis show that mobile websites impose an average of 221 kilobytes of data to the end user's browser, while desktop websites impose up to 403 kilobytes of data. We performed various optimization techniques and used various tools to measure the added overhead of the different components. We built a framework using these tools and techniques, which can effectively reduced a website's total size by a minimum of about 23% without compromising usability or functionality. We showed that there is a dire need for such optimization on the current web landscape, especially in the mobile space where data usage come at a premium for both end users and network operators.

References

[1] F. Qian, K. S. Quah, J. Huang, J. Erman, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Web caching on smartphones: ideal vs. reality," in *MobiSys*, Lake District, UK, 2012.

[2] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "Characterizing radio resource allocation for 3g networks," in *IMC*, Melbourne, Australia, 2010.

[3] "ShrinkSafe," http://shrinksafe.dojotoolkit.org/.

[4] D. Crockford, "JSMin," http://crockford.com/javascript/jsmin.

[5] "Yui compressor," http://yui.github.io/yuicompressor/.

[6] "Alexa," http://www.alexa.com/.

[7] S. Souders, "High-performance web sites," *Communications of the ACM*, vol. 51, no. 12, pp. 36–41, 2008.

[8] "PhantomJS," http://phantomjs.org/.

[9] M. Butkiewicz, H. V. Madhyastha, and V. Sekar, "Understanding website complexity: Measurements, metrics, and implications," in *IMC*, Berlin, Germany, 2011.

[10] HttpArchive.org, "HTTPArchive," https://www.httparchive.org/.

[11] "HTTP Archive Format," https://dvcs.w3.org/hg/webperf/raw-file/tip/specs/HAR/Overview.html.

[12] "JSCover - JavaScript code coverage tool," http://tntim96.github.io/JSCover/.

[13] Ad Block, "EasyList," https://easylist.adblockplus.org/en/.

[14] F. Qian, J. Huang, J. Erman, Z. M. Mao, S. Sen, and O. Spatscheck, "How to reduce smartphone traffic volume by 30 percent?" in *PAM*, Hong Kong, 2013.

[15] R. Pries, Z. Magyari, and P. Tran-Gia, "An http web traffic model based on the top one million visited web pages," in *Next Generation Internet (NGI), 2012 8th EURO-NGI Conference on*. IEEE, 2012, pp. 133–139.

[16] S. Ihm and V. S. Pai, "Towards understanding modern web traffic," in *IMC*, Berlin, Germany, 2011.

[17] "OptiPNG: Advanced Image Optimizer tool," http://optipng.sourceforge.net/.

[18] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. P. Singh, "Who killed my battery?: Analyzing mobile browser energy consumption," in *WWW*, Lyon, France, 2012.

[19] B. Krishnamurthy and C. Wills, "Privacy diffusion on the web: a longitudinal perspective," in *WWW*, Madrid, Spain, 2009.

[20] N. Vallina-Rodriguez, J. Shah, A. Finamore, Y. Grunenberger, K. Papagiannaki, H. Haddadi, and J. Crowcroft, "Breaking for commercials: characterizing mobile advertising," in *IMC*, Boston, USA, 2012.

[21] A. J. Khan, V. Subbaraju, A. Misra, and S. Seshan, "Mitigating the true cost of advertisement-supported free mobile applications," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM, 2012, p. 1.

[22] T. Callahan, M. Allman, and V. Paxson, "A longitudinal view of http traffic," in *PAM*, 2010.

[23] J. Erman, V. Gopalakrishnan, R. Jana, and K. Ramakrishnan, "Towards a spdy'ier mobile web?" in *CoNEXT*, Nice, France, 2013.

[24] "HTTP/2," http://http2.github.io.