How's My Network? Predicting Performance From Within a Web Browser Sandbox

Murad Kaplan, Mihajlo Zeljkovic, Mark Claypool and Craig Wills
Computer Science Department
Worcester Polytechnic Institute
Worcester, Massachusetts 01609, USA
{mkaplan,mihajlo,claypool,cew}@cs.wpi.edu

***Abstract*—Measuring Internet performance for home users can provide useful information for improving networks. Unfortunately, Internet measurements typically require users to install special software, a major impediment to use. To overcome this impediment, we designed and implemented several scripting techniques to predict Internet performance from within the tightly constrained sandbox environment of a Web browser. Our techniques are integrated into a Web site project called "How's My Network" that provides predictions for common Internet activities, with this paper concentrating on the performance of online news, social networks, and online shopping. The contributions of this work include the design, implementation and evaluation of new approaches for predicting Web browser performance.**

I. Introduction

As Internet use has grown, so has the importance of understanding network performance for end-users. Network researchers care about network statistics, such as throughput and round-trip time, while end-users care about how effectively their network connection supports the applications they want to use. What is needed are techniques that provide researchers with low-level network data, while providing end-users with effective, easy-to-access methods of understanding their network performance.

Unfortunately, most current measurement tools and platforms have high impediments to use for the typical home user [9]. Dimes [7] and DipZoom [3] provide good network data, but require users to install software on their machines to run measurements. PlanetLab [14] and Archipelago [11] are even more flexible in the data that can be gathered, but require participants to permanently contribute to the measurement infrastructure. Tools such Speedtest [1] have low impediments to use by being available through a standard Web browser, but the measurement methods are not transparent and the low-level data gathered is not available for network research. Other tools such as Gomez [4] go a step further by providing incentives (e.g. money) for use, but are closed in that the data returned is not available to the users or to researchers. Moreover, current network tools and platforms do not provide incentives for use in that the information returned is not understandable or even relevant to most end-users for the applications they care about. For example, Speedtest provides ping times, download speeds and upload speeds to the nearest server, but most home users are unable to map such low-level performance data to the applications they care about, such as playing a network game, talking in a VoIP session, or reading news online. What are needed are performance tools targeted towards the end user, while also providing valuable, low-level data for network researchers.

The *How's My Network* (*HMN*) project seeks to provide a measurement platform with low impediments to use by running within a Web browser without the need of additional installed software, while providing users with incentives in the form of meaningful performance data on the applications users care about. Through several Javascript techniques [18], users' browsers, upon visiting HMN,[1] collect accurate, low-level network data that is mapped to the applications in which they have indicated interest. Thus, through the HMN Web site, users select Internet activities in which they have interest (e.g. reading online news), and are provided with their predicted performance (e.g. four "star" performance) for those activities, while network researchers obtain valuable, low-level performance data on the users' networks.

This paper presents the results of making HMN predictions for Web browsing, concentrating on the domain of online news. Online news is our initial focus since reading news online via a Web browser is an increasingly important Internet activity. For example, in Korea, more than half the population reads news online [5]. Moreover, the number of people who read online news is likely to grow with new generations that get early exposure to the Internet. The Internet is already the main source of news for 16-24 year olds, and about 5 percent of all Internet visits are related to reading news online [15].

We characterize news Web sites and analyze browser download mechanisms to provide a basis for techniques to predict performance, both low-level (e.g. download time) as well as user-centric (e.g. "star") performance. The prediction methods are realized using our Javascript techniques, allowing us to compare predicted to actual performance for a range of news sites. Beyond news, our prediction techniques are applied to social networks and online shopping in order to test the generality of the methods deployed. The predictions allow users to assess performance much more quickly (about 10x faster) and with far less data (about 50x less) than downloading Web pages.

[1] http://hmn.cs.wpi.edu/

Evaluation shows that using the number of objects from the "dominant domain," the one providing the most number of objects, improves prediction performance over using the total number of objects. Predictions done by downloading objects in parallel, as is supported by all modern browsers, is also an improvement over predictions done by downloading objects serially. About 40% of predictions based on parallel downloads from the dominant domain have no noticeable error as far as users are concerned, and nearly 90% of all predictions are within one "star" on a five-star performance scale for news, social networks and shopping.

The rest of this paper is organized as follows: Section II describes our approach to predict the performance of Web browsing. Section III describes our experimental setup. Section IV highlights results and evaluations from the experiments. Section V shows how news performance prediction methods can be used on shopping and social sites. Finally, Section VI summarizes our conclusions and presents possible future work.

II. Approach

Ascertaining the download time of a single main page from one news site can be done fairly quickly, often in less than 10 seconds. However, the typical news site has tens of pages and there are dozens or even hundreds of potential news sites that may interest a user. Downloading many such pages to provide an overall measure of performance is prohibitive for most users. Our approach is to provide an estimate of Web page download time quickly, in less than a second, yet accurately in terms of assessing performance relevant to the user. A key aspect of our work is to measure the performance of downloading an object from an arbitrary server on the Web without the need for software outside the users Web browser. Our approach uses the following steps to provide predictions for Web page download performance:

- *Characterization:* We characterize news Web sites based on object sizes and domains from which the objects are retrieved. We also investigate three popular Web browsers, studying the mechanisms they use for retrieving and rendering Web objects. While such characterization and analysis has been done in the past, to the best of our knowledge there are no relevant, recent characterizations and the Web's structure and retrieval methods change rapidly.
- *Prediction Methods:* After characterization, we use the sites' structures and the browsers' behaviors to inform methods designed to predict Web browser performance. We construct four different prediction methods to match the differences observed among the sites and Web browsers.
- *Testing and Evaluation:* We experimentally test the prediction methods with different Web browsers and different Web sites. We setup environments to control bandwidth and delay and compare the predicted performance times to actual performance times over a range of controlled network settings.

A. Characterization

For characterization, the four most visited news sites in the U.S. [13] were selected:

- CNN (http://www.cnn.com/) (CNN)
- MSNBC (http://www.msnbc.msn.com/) (MSN)
- The New York Times (http://www.nytimes.com/) (NYT)
- The LA Times (http://www.latimes.com/) (LAT)

Characterization items of interest are those that may affect the page load time [17]: number of objects, sizes of objects, size of overall page, and number and location of servers.

For each site, Web page data was gathered by completely downloading each page using PageStats [10], a Firefox plug-in that automatically fetches complete Web pages, recording the URLs and sizes of embedded objects. The URL provides the domain used to serve the object, including both the original domain and third-party servers that are often used for images or advertisements. Web page characterization data was gathered in February and March, 2011.

With the advent of asynchronous technologies such as AJAX, some Web pages never "finish" loading, instead downloading small objects every few minutes. In our experiments, these pages are considered downloaded when the container and all objects were downloaded the first time.

Table I shows summary results, with each news home page indicated in the first column. The second column provides the total number of objects in the page. The next three columns are, in Kbytes, the average object size, median object size, and maximum object size, respectively. The sixth column is the total size for all objects on the page. The last two columns are the number of different domains used to serve the objects and the fraction of those objects that come from a single (or dominant) domain, respectively.

The number of objects varies across news sites, with the fewest (CNN and NYT) having about half those of the most (LAT). This proportion of 2 to 1 holds for the average and maximum sized objects, also, although not necessarily for the same sites. The median sizes vary considerably more, with the smallest median (NYT) being only one-tenth the size of the largest median (LAT). The total sizes follow a similar, but less dramatic, difference of about 4 to 1 for the LAT to the NYT. The number of domains varies from 10 to about 20, with about 25% to 75% of the objects being served from a single domain at the most.

Also of critical importance is the mechanisms used by browsers for retrieving Web pages. We focused on the three most popular Web browsers [2]: Chrome (v12), Internet Explorer (IE, v8), and Firefox (v4). Figure 1 shows a screenshot from Fiddler's[2] timeline for the CNN home page for each browser. The vertical axis indicates the individual objects downloaded by the Web browser, with the horizontal axis representing time. The left column shows each object's URL, and the colored bars for each object show the time it was

[2]Fiddler is a Web debugging proxy that logs all HTTP traffic between client and server, http://www.fiddler2.com/fiddler2/

TABLE I
CHARACTERIZATION OF NEWS SITES

News Site	Objects (Count)	Container Size (Kbytes)	Avg Size (Kbytes)	Median Size (Kbytes)	Max Size (Kbytes)	Total Size (Kbytes)	Domains (Count)	Dominant Domain (% Objects)
CNN	96	22.3	8.7	2.5	61.4	813.1	10	61%
LAT	203	54.8	10.3	7.0	65.7	2049.5	21	78%
MSN	161	47.6	4.8	2.0	90.8	748.9	22	24%
NYT	96	3	5.1	0.7	54.7	474.5	19	31%
Average	139	32	7.2	3.1	66	1021.5	18	49%

TABLE II
WEB BROWSER CHARACTERISTICS

Browser	Connection per Server	Maximum Connections	DL Script Image	Style Sheet
Chrome	6	35	Yes	No
Internet Explorer	6	30	Yes	Yes
Firefox	6	35	No	No

downloading. Fiddler colors objects according to their types: light-green for images, dark-green for Javascript, purple for style sheets and blue otherwise. The second object in the Internet Explorer timeline has a conditional comment[3] which causes IE to block the download of the images (light green) until all the Javascript files (dark green) have been downloaded.

The order and means by which objects are downloaded depends upon the Web browser used. The download mechanisms used by the three Web browsers are provided in Table II. The first column is the maximum number of connections that can be simultaneously opened per server host. The second column is the maximum number of connections a browser opens across all server hosts. The third column indicates whether or not the browser downloads scripts in parallel with other images in the page. The last column shows whether or not the browser blocks objects from downloading if there is a conditional comment in the style sheet.

All three browsers make up to 6 persistent connections per host, with about 30 connections maximum. Both Firefox and Chrome download objects while downloading scripts and style sheets, while IE blocks other downloads while downloading a script or downloading a style sheet with a comment.

B. Prediction Methods

The data on the structure of news sites and browser download mechanisms form the basis for methods that can be used to predict Web browser performance.

Preliminary investigation with Fiddler showed the container page (i.e. `index.html`) is always downloaded first, followed by embedded objects, sometimes downloaded serially, sometimes downloaded in parallel. Based on these observations, we explore two primary aspects of Web page retrieval that affect performance: 1) the total number of objects (*T*) versus the objects from the dominant domain (*D*), and 2) downloading Web objects serially (*S*) or in parallel (*P*).

[3]Conditional comments are typically used to give IE-specific instructions to the browser.

For each type of news page (the home page, the section page, (e.g. Sports), and an article page) for each news site, we choose two representative objects based on data obtained in Section II-A. The first object's size is approximately equal to the container object's size and the second object's size is approximately equal to the average size of all objects in the page. Our methods download each object using previously developed Javascript techniques [18], downloading the object in the browser's `iframe` and measuring the time it takes to be loaded using the `onload` event as shown in the following example:

```
[language=Javascript]
var object_load_start = new Date().getTime();
var URL = "http://www.reuters.com/images/sprite-core.gif";
iframe_avgObject.onload = log(object_load_start, URL);

function log(object_load_start, file_URL) {
  var object_load_end = system.getTime();
  report = ((object_load_end - object_load_start) + "ms\n");
  // report results
}
```

Combinations of both aspects (T/D and S/P) provide four possible methods that can be used to predict Web browser performance (ST, SD, PT, PD) shown in Table III. In the Formula column, T_c is the time to download the container, T_o is the time to download an average-size object, N_a is the number of all objects on the page, N_d is the number of objects from the dominant domain and p is the number of downloads in parallel (6, as in Table II for all browsers tested). There is an associated "cost" in using each method to predict performance, defined here as the number of downloads. Using the Javascript techniques above, the serial methods download the container page plus one object while the parallel methods download the container page plus the number of objects downloaded in parallel.

III. EXPERIMENT SETUP

The effectiveness of our methods to predict Web browser performance was measured through a set of experiments, depicted in Figure 2. A PC with a 2.93 GHz Intel Core i7 CPU, 8 GBytes of RAM, was setup, running Windows 7 Service Pack 1 and Web browsers Chrome v12, Internet Explorer v8, and Firefox v4. The PC sent network traffic by way of a Linux PC acting as a router, running netem configured to provide a network environment of a standard U.S. residential ADSL link: 1 Mb/s download and 256 Kb/s upload with

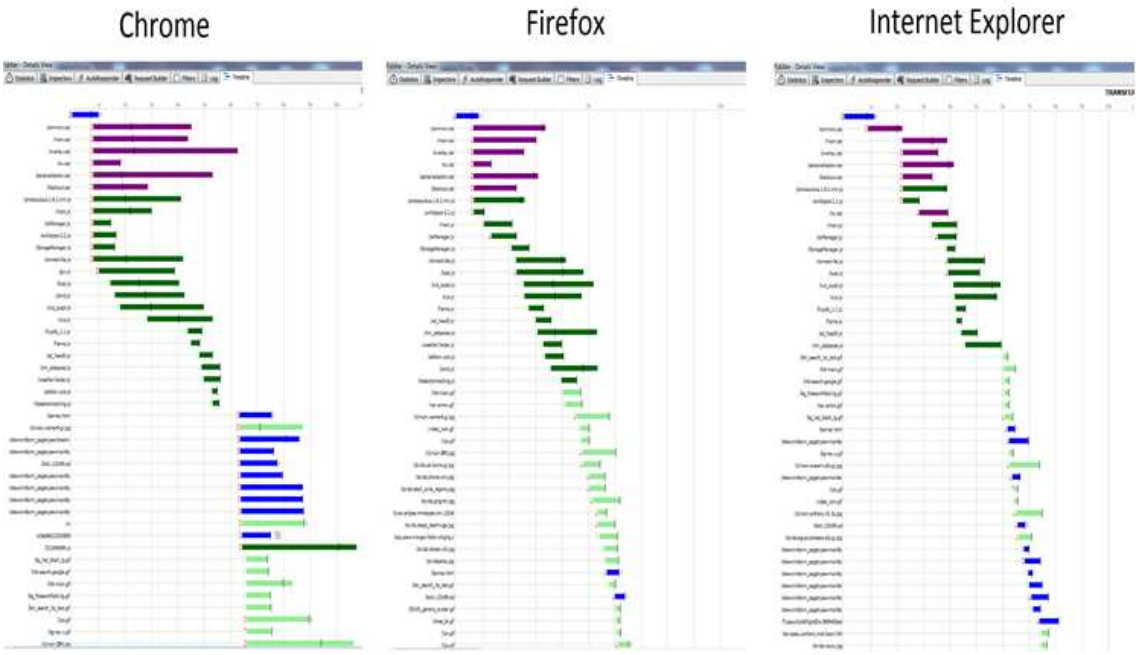

Fig. 1. Screenshot of Fiddler, showing download timeline for Web browsers (CNN)

TABLE III
SUMMARY OF METHODS FOR PREDICTING WEB PAGE DOWNLOAD TIME

Method	Formula	Cost
Serial-Total (ST)	$T = T_c + T_o \times \frac{N_a}{1}$	2
Serial-Dominant (SD)	$T = T_c + T_o \times \frac{N_d}{1}$	2
Parallel-Total (PT)	$T = T_c + T_o \times \frac{N_a}{p}$	1 + p
Parallel-Dominant (PD)	$T = T_c + T_o \times \frac{N_d}{p}$	1 + p

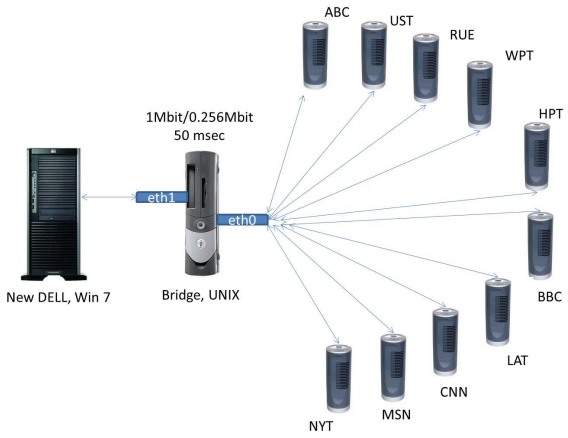

Fig. 2. Experiment setup

100 milliseconds of round-trip time. For testing, our initial four news sites were expanded to the ten most popular news sites in the U.S. [13]: ABC News (ABC), the BBC (BBC), CNN (CNN), the Huffington Post (HPT), the Los Angeles Times (LAT), MSNBC (MSN), the New York Times (NYT), Reuters (REU), USA Today (USA), and the Washington Post (WPT). The tests were run on the weekends from May to June, 2011. The experiments measured the time to retrieve a Web page, from the initial request until the Javascript `onload` event was received. Also, during each run, the objects that represent the `index.html` files and the average size objects were downloaded and timed using the Javascript technique from Section II-B. For each browser type, each Web site was measured five times based on levels of variance observed, with the browser cache cleared between runs.

IV. RESULTS

Figure 3 shows, for each Web browser, the download time averaged across five measurements for each of the ten news pages, with the standard deviation shown by error bars. The time needed to download each page varies by news site and browser, with the largest difference between the NYT in Chrome to the LAT in IE. These time differences are largely due to the differences in the number and sizes of objects. The download times vary somewhat less, but still by as much as by 50%, across browsers to the same news site, with the largest difference to the USA Today for the home page. This performance difference depends on the type of objects the site retrieves and the browser functionality as in Table II. Overall, the home and sports pages have similar page load times across browsers. The article times vary somewhat more, differing by up to 60% across browsers to the same news site, the WPT.

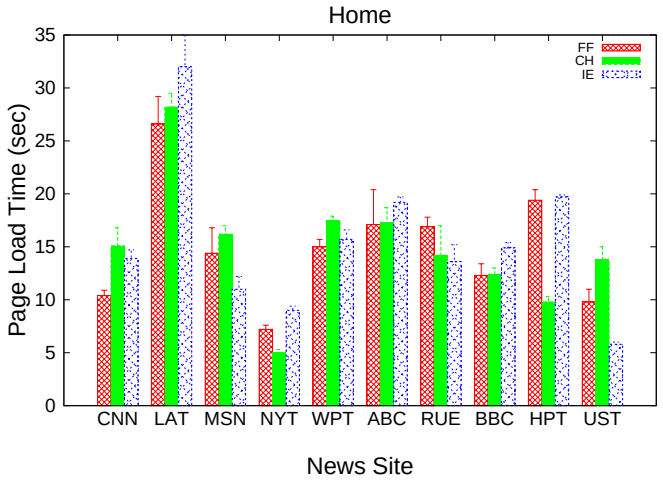

Fig. 3. Average download times for news sites

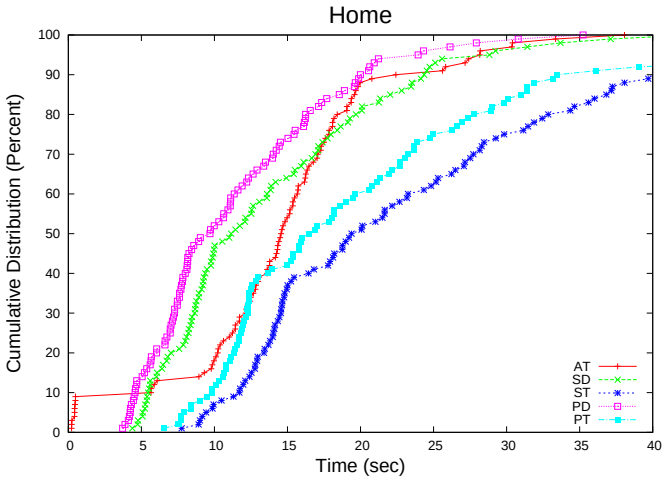

Fig. 5. Distribution of prediction times and page download times for news pages

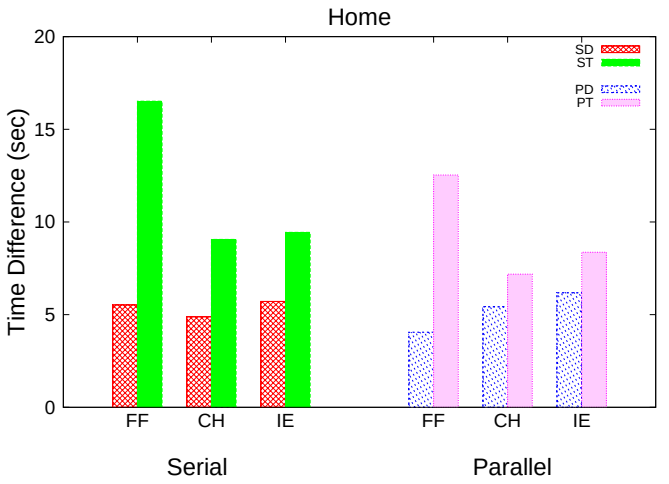

Fig. 6. Average time difference between predicted and measured for news pages

Figure 4 shows the average time difference in seconds for the four prediction methods across the three browsers for all downloads. The vertical axis is the difference between the measured download time and the predicted download time broken down by browser. The average times are depicted by AVG on the far right of each graph. Visually, Firefox tends to have the highest differences in predicted versus actual times and Chrome the lowest.

Figure 5 depicts a cumulative distribution of the predicted times for all runs, with the horizontal axis being the time difference in seconds. The trendlines are separated by prediction method: serial-dominant (SD), serial-total (ST), parallel-dominant (PD) and parallel-total (PT). The AT trendline shows the actual load times. Distributions to the left of AT underestimate the download time while distributions to the right overestimate the download time. The prediction method distributions clearly separate, with PD being the farthest to the left, followed by SD, PT and ST, in that order. Using the total number of objects tends to overestimate the prediction time, while using the number of objects from the dominant domain underestimates prediction times for some news sites.

Figure 6 shows the average time difference in seconds for the four prediction methods among the three browsers. The vertical axis is the average difference between the measured download time and the predicted download time. The horizontal axis has two main clusters, with the left side showing the serial predictions and the right side showing the parallel predictions. Within each side, there is a separate cluster for measurements done on Firefox (FF), Chrome (CH) and Internet Explorer (IE). In general, the SD bars are lower (better predictions) than the ST bars and the PD bars are lower than the PT bars for all browsers. These results indicate that using the number of objects served from the dominant domain provides better predictions than using the total number of objects.

The time to download the container (i.e. `index.html`) is analyzed in order to see the container's contribution to the prediction methods. Figure 7 shows the distribution of container download time and the rest of the time calculated by each method. For all runs, the container download has a time of 0.5 to 2 seconds. This suggests accurate predictions can eliminate downloading the container, merely using a small constant for each, reducing the cost of downloading objects in a user's browser and time needed to complete the run.

While the measured time differences may be of interest to network researchers, many users will not notice the impact of an additional few seconds of page load time to their browsing experience. In order to reflect performance relevant to users, measured download and predicted times are mapped to a 5-star scale, which can be shown to users when tests are complete. The thresholds for stars was informed by previous

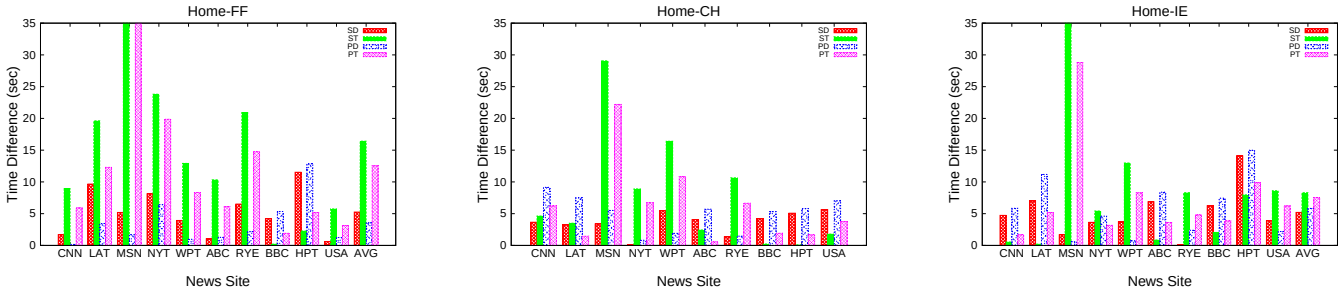

Fig. 4. Difference in times between predicted and measured in news sites for each browser

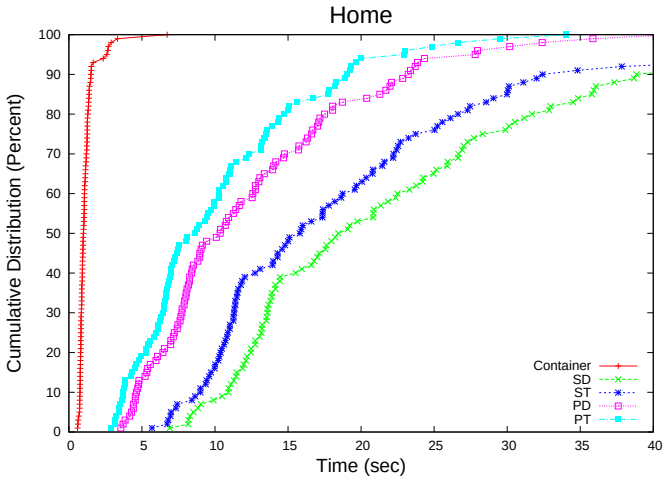

Fig. 7. Distribution of trendline of container download time and prediction times for news sites

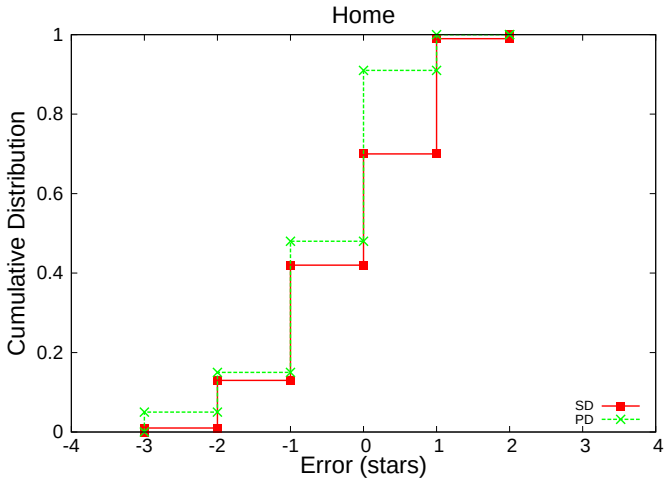

Fig. 9. CDF of prediction errors for news sites across browsers

TABLE IV
STARS SCALE

Time (seconds):	[0-5]	(5-10]	(10-15]	(15-20]	(20+
Stars:	⋆ ⋆ ⋆ ⋆ ⋆	⋆ ⋆ ⋆⋆	⋆ ⋆ ⋆	⋆⋆	⋆

work on Web site response times [12], [16] where response times within 10 seconds are generally satisfactory while longer response times lead to more dissatisfaction. The mapping of performance to stars is shown in Table IV. Stars are only provided as in integer number of stars (e.g. no 1/2 star ratings).

The difference between the download time and the prediction time in stars provides a user-centric prediction error. Figure 8 shows the error in stars for each news Web site. The horizontal axis indicates the site while the vertical axis indicates the error, in stars. Errors less than zero indicate the prediction time was greater than the measured time. Since using the number of objects from the dominant domain gives a better prediction than using the total number of objects, here and for all subsequent graphs only analysis of predictions using the number of objects in the dominant domain is shown. In many cases, for home pages (CNN, LAT and ABC) both SD and PD provide "accurate" predictions that effectively predict browser performance with no error. In most other cases, the error is only 1 star and the only 2 star error is for HPT. Comparing PD to SD, in all cases PD does as well as SD, except for RUE and USA where SD is worse since it overestimates the download times in FF and IE.

To evaluate the effectiveness of the PD predictions compared to the SD predictions, Figure 9 shows the cumulative distribution functions (CDFs) of prediction errors for all news sites and browsers. The horizontal axis is the prediction error (in stars) and the vertical axis is the cumulative distribution. There are two trendlines, one for SD and one for PD. The graphs indicate that PD provides "accurate" predictions just over 40% of the time, while SD is somewhat worse at just under 30%. At the right of the distribution, for about 3% of the predictions, PD has nearly 3 stars in error, compared to only 0.5% for SD.

V. SHOPPING AND SOCIAL SITES

After vetting our prediction methods with online news, our approach is applied to social networks and online shopping to briefly investigate the efficacy of the prediction methods to other domains. The five most popular sites were chosen for

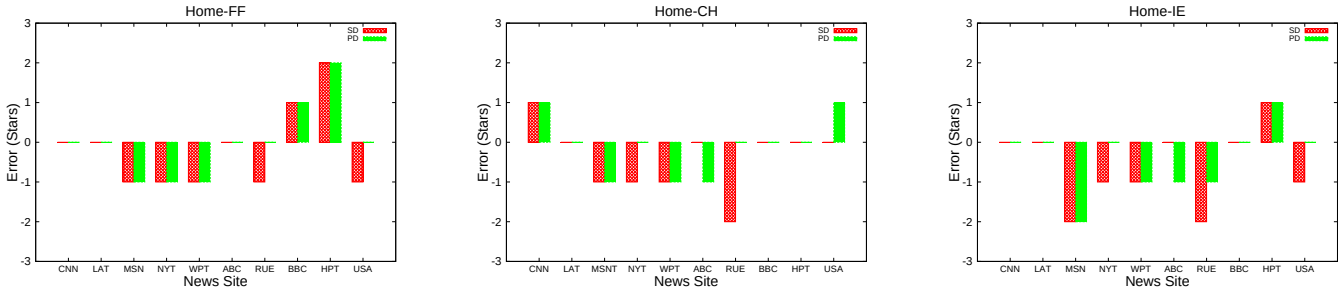

Fig. 8. Prediction error for news sites across browsers

shopping [8]: Amazon (AZ), Ebay (EB), Bizrate (BZ), Giftag (GT), and SmartBargains (SB); and for social networks [6]: Facebook (FB), Twitter (TW), Myspace (MS), Friendster (FR), and Linkedin (LD). An analysis of the comparative distribution for PD and SD across all browsers is provided.

For characterization, overall, the shopping page size is relatively small compared to news sites, about 200 Kbytes, with a maximum page size of 889 Kbytes for Amazon. While GT has the smallest average page size of 215.8 Kbytes, GT has the highest percentage of dominant domain objects at 93% and only 4 domains. Although SB has the highest number of objects, it has the smallest container page, and smallest average, median and maximum object sizes.

Characterization of social networking sites shows them all to have similar page sizes, about 400 Kbytes, compared to shopping and news sites. Also social networking page sizes are about double those of shopping sites, except for Amazon.

Figure 10 shows the error in stars for each shopping and social networking site for each browser. Our prediction methods give better predictions for shopping and social sites than for news with the worst prediction of two stars for both methods for social networking in Chrome. Both SD and PD provide accurate predictions for Amazon and Ebay for all browsers except for Amazon for shopping using Chrome.

Figure 11 shows the CDFs of prediction errors for shopping and social sites for all three browsers, with the three trendlines representing online news, social networks and online shopping in the lower graph. PD provides predictions with effectively 0 stars of error 80% of the time for shopping and 60% of the time for social networking. PD is about as effective for social networking as it is for news, while PD provides even better predictions for online shopping. For online shopping, about 65% of the predictions have effectively no error, and no predictions are worse than 2 stars of error.

VI. Conclusion and Future Work

Platforms that provide network measurements are needed by researchers to better understand and improve networks. End users can provide valuable network data if there are low impediments to participating in measurements studies and if there are incentives that encourage them to provide data.

Fig. 11. CDF of prediction errors for shopping, social and news across browsers

The *How's My Network (HMN)* project seeks to provide low impediments to user participation by enabling end-host measurements from within the constrained sandbox of a Web browser, thereby not requiring the user to install any additional software. Incentives for users are provided in the form of meaningful feedback data for the applications they care about, such as quickly and accurately predicting the number of "stars" a network can provide in support of Web browsing. This paper presents an approach for predicting the performance for Web browsing from within a browser sandbox by using the structure of Web sites and download techniques for typical browsers to provide predictions based on the number and sizes of objects and their methods of download.

Detailed experimental results for online news find that using the number of objects from the dominant domain is better for predictions than using the total number of objects, being over twice as effective for some Web browsers. Assuming objects download in parallel rather than serially provides generally better predictions, with about 15% better accuracy for online news. Applying our methods to social networks and online shopping shows our approach works for other browser activities too, with nearly half of all predictions having

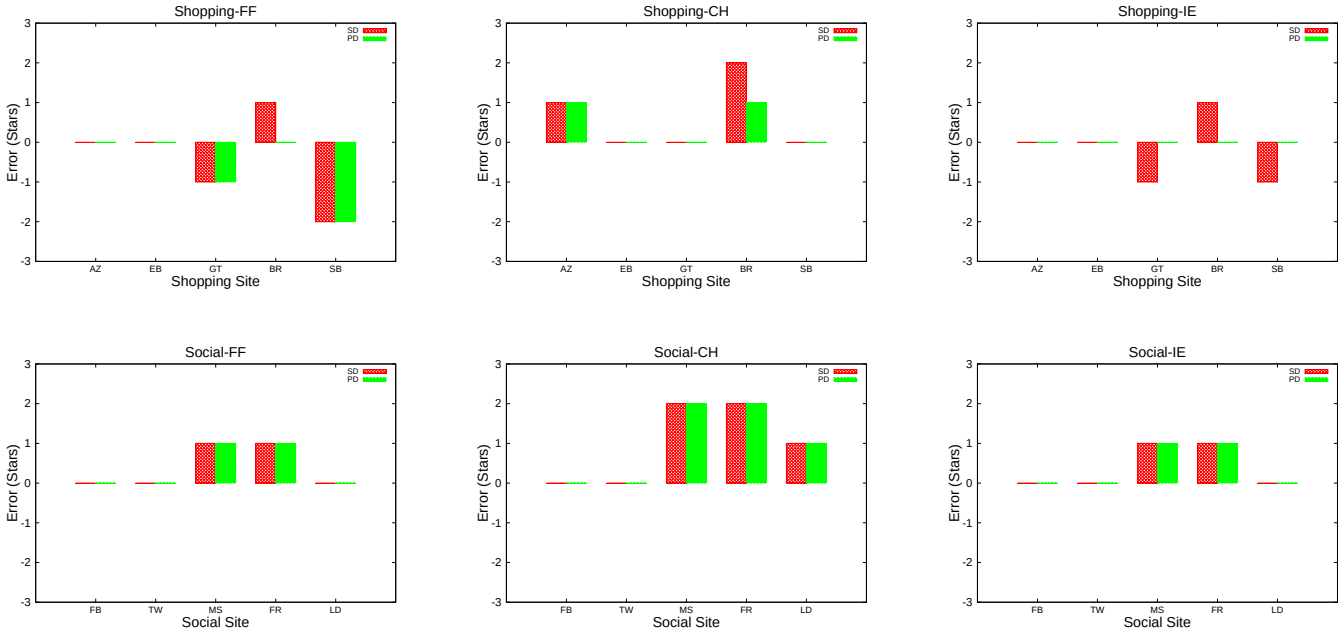

Fig. 10. Prediction error for shopping and social sites across browsers

effectively no error.

Future work includes extensions to other Web sites such as online banking and blogs. Our method would be to characterize the Web sites, followed by applying our prediction methods as in this paper. Methods to include other object types, such as streaming media, may provide additional predictive power for a broader range of user tasks. Repeated measurements and predictions over time may discover if there are systematic errors in prediction for certain Web sites. The work assumes the dominant factor in determining Web browser performance is the object download times, a reasonable assumption for many end-devices. However, future work could adjust the prediction methodology for systems in which end-host processing and page rendering are the bottleneck. The HMN project continues to be developed, providing light-weight, but accurate, performance predictions for other Internet applications such as VoIP, online games, and video streaming. Beyond download times, additional metrics being developed include packet loss, round trip-time and delay jitter.

REFERENCES

[1] "Broadband reports.com Speed Test," http://www.dslreports.com/stest/.
[2] "BrowserScope," http://www.browserscope.org.
[3] "Deep Internet Performance Zoom," http://dipzoom.case.edu/.
[4] "Gomez Peer Community," http://www.porivo.com/.
[5] "OECD examines the future of news and the Internet," http://tinyurl.com/3n26e8j.
[6] "Social Networking Websites Review," http://social-networking-websites-review.toptenreviews.com.
[7] "The DIMES project," http://www.netdimes.org/.
[8] "Top 15 Most Popular Comparison Shopping Websites," http://www.ebizmba.com/articles/shopping-websites.
[9] k. claffy, M. Crovella, T. Friedman, C. Shannon, and N. Spring, "Community-oriented network measurement infrastructure (CONMI) workshop report," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 2, pp. 41–48, 2006.
[10] S. DeDeo, "Pagestats," May 2006, http://www.cs.wpi.edu/ cew/pagestats/.
[11] Y. Hyun, "Archipelago Meas. Infrastructure," in *CAIDA-WIDE Workshop*, 2006.
[12] J. Nielsen, "Website Response Times," June 2009, http://www.useit.com/alertbox/response-times.html.
[13] E. O'Neill, B. Lavoie, and R. Bennett, "Trends in the evolution of the public web," *D-Lib Magazine*, vol. 9, no. 4, 2003.
[14] L. Peterson, A. Bavier, M. Fiuczynski, and S. Muir, "Experiences Building PlanetLab," in *USENIX Symposium on OSDI*, Seattle, WA, 2006.
[15] T. Rogers, "What is Web Journalism," http://journalism.about.com/od/trends/a/whatwebjour.htm.
[16] P. J. Sevcik, "Understanding How Users View Application Performance," July 2002, http://www.netforecast.com/Articles/BCR C22 Performance Zones.pdf.
[17] S. Souders, *High Performance Web Sites*. O'Reilly Media, 2007.
[18] M. Zeljkovic, M. Kaplan, C. Wills, and M. Claypool, "Javascript and flash overhead in the web browser sandbox," Computer Science Department, Worcester Polytechnic Institute, Tech. Rep. WPI-CS-TR-10-14, 2010.