Summary Review Documentation for

"Fathom: A Browser-based Network Measurement Platform"

Authors: M. Dhawan, J. Samuel, R. Teixeira, C. Kreibich, M. Allman, N. Weaver, and V. Paxson

Reviewer #1

Summary: This paper describes the design, implementation, and evaluation of Fathom, a Firefox extension that enables measurements from the Internet's edge. The main idea is that, in order to gather measurements from end-hosts, a browser extension is more portable than applications. And once the Fathom browser extension is installed by a user, websites that wish to perform measurements from the user's computer can include appropriate javascript in their site's code such that this javascript can invoke methods inside Fathom to gather either active or passive measurements. Fathom's architecture takes several measures to address the security and privacy of the user.

Strengths: Attempts to solve an important problem. Thorough evaluation. Real implementation available for public use. Well written.

Weaknesses:

- Road to deployment is by subversion of users.
- Limitations of architecture are not fully discussed.

Comments to authors: Fathom certainly solves an important problem and I would be interested to see how much traction this gets.

My main concern is that your envisioned path to deployment is effectively by conning users. You state that you expect that users can be incentivized to install Fathom to avail of its "diagnose web connectivity" feature, and thereafter Fathom can be used for measurements by all websites. How is this different from all those random browser toolbars that get you install them for useful features like search, shortcut buttons, etc. and then end up gathering passive measurements? It is hard for me to distinguish Fathom from spyware! I know you state that users can selectively grant permission to each site to gather measurements. But, to get widespread deployment at the scale you envision, you surely need users who will not understand anything about what they are being asked to grant permission for, at which point they will either uninstall Fathom or blindly click "yes", which again leaves me wondering: how is this different from spyware ...

Its unclear to me that portability is really the primary motivation for in-browser measurements. What is the problem with Java standalone applications? Java-based programs can run on most systems. In fact, I would expect Java programs to run on more systems than the number of machines with a given browser (for which the extension is written). With the browser market getting more and more fragmented, portability of the extension across browsers may be a bigger concern than the portability of Java-based applications.

In fact, the current Fathom architecture has two significant limitations compared to stand-alone applications:

1. a website can perform a measurement for only as long as that site's page is open. If I visit news.google.com and close the page before the javascript on that page finishes gathering relevant measurements, too bad for google. In general, Fathom does not support scheduling of (possibly a batch of) measurements for the future.

2. measurements can only be gathered by web pages that the user visits. So, even though you state this as part of your motivation, it is unclear how researchers can benefit from this platform.

It appears to me that the motivation for in-browser measurements lie elsewhere from portability: 1) a browser extension enables passive measurements (such as load times to websites), whereas a stand-alone application will require root privileges to do the same, and 2) an extension is better at being "invisible" to the user (which goes back to my point above of trying to trick users).

By the way, the paper would be significantly stronger if it elaborated on the types of measurements that websites cannot perform themselves today (by embedding appropriate javascript in their web pages), but is enabled by Fathom. You allude to this in your examples, but a more direct treatment would help validate your point about offering "rich measurement capabilities".

Reviewer #2

Summary: This paper presents Fathom, a browser extension for Firefox that enables in-browser access to network and system resources for the purposes of network experiments, measurements, and debugging. Web site operators can include Fathom scripts (written in Javascript with special access to a Fathom object) in their pages, and can receive the results back from end users. The authors implement Fathom, present microbenchmarks showing that Fathom imposes low overhead, and use Fathom in three case-studies to demonstrate different use cases.

Strengths: The paper presents a nice, complete, usable system for performing network measurements, experiments, and debugging in browsers. The paper is well-written and is a pleasure to read.

Weaknesses: As the authors admit, using a browser extension greatly reduces the possible user base (due to user action required, and the Firefox-specific nature). I am somewhat skeptical of the security/privacy story, as most end users would not likely be able to understand the implications of the various permissions.

Comments to authors: Overall, I enjoyed reading this paper. I like the novel use of the browser as a measurement tool, and of the

API that you've developed to enable third-party sites to use user browsers. I have a few comments, divided into high-level and low-level ones below.

High-level:

- My primary concern is over the incentives for users to install the extension (to be fair, this is a topic that you discuss at some length). Unfortunately, I wasn't swayed by the argument that the extensions significantly aids users through "built-in troubleshooting capabilities". There exist a number of alternative troubleshooting facilities (e.g., Apple's Network Diagnostics, Netalyzer, etc), and Fathom would not seem to provide a substantial benefit beyond these existing systems (at least, not enough to convince random users to install it). It seems that, overall, the primary direct beneficiaries of Fathom are the web site operators who are able to gain significantly more information about the end systems their web page is rendered on (esp. since Fathom's implementation strategy is an in-browser framework). It therefore might be a more convincing case to argue that web site operators could offer incentives for users to install the extension, possibly in response to requests for troubleshooting assistance.
- My secondary concern is over the user interface issues that Fathom introduces. From your description in 4.3, it would seem that the user must configure the local policy, and then potentially make decisions along the lines of "should the Fathom script at page X be able to access resource Y?". As you admit, this is likely to be difficult for many users. However, I'm somewhat skeptical that the signed code would provide much assistance – the signed code would only verify that (say) cnn.com is the author of the code, but wouldn't tell me much about what the implications are of enabling access to the extension. Moreover, I suspect that most users will no modify the local policy from the default, making the choice of the defaults incredibly important. But, I don't see any discussion of what the defaults are.

Lower-level:

- In 5.1, you note that Java applets have poor code portability, citing Android as an example where the full Java API is not available. I found this a bit odd, as Fathom would seem to have little hope of running under Android (no Firefox, to begin with). Given that I was already convinced about the downsides of Java applets (lowering supporting for Java runtimes on end hosts, to begin with), I would recommend reworking this comparison.
- I don't quite understand while Plugin (runtime) [representing Java et al.] gets an "X" for both Browser and OS portability. For most of the stuff that Fathom supports, the Java runtime would seem to work just fine (with the caveat that the user has to approve the applet being run with full privileges).
- At the end of 5, you argue that Fathom code writers are discouraged from using too many resources, as the user may simply close their browser tab in response. But, I don't quite see how a user would figure out which tab was causing the problem – most browsers that I know of don't give you much insight into how many resources different tabs are consuming, and even looking at the "ps aux" output (or equivalent) would only show that it was the browser using the (say) CPU.

Reviewer #3

Summary: A measurement plug-in for the Firefox web browser to support user performance measurements as well as researcher experimentation. Seems like the kind of paper IMC was designed for, will generate a lot of interesting conversation at conference.

Strengths: The platform design goals, and its implementation, are spelled out clearly, likewise Fathom's design. Much of the section about this is taken up with detailed comments about the programming and execution enviroments inside various browsers - that's certainly interesting, and will be new territory for many readers.

Section 6, performance evaluation, shows that Fathom is capable of high network throughput (I hope it warns users that testing available bandwidth by transferring big files will impact other network users), and that it adds little overhead to normal browser operations.

The examples given to demonstrate how Fathom can be used, e.g. implementing Netalyzer, are interesting too, and do demonstrate its strengths. Paper weaknesses The only thing that could usefully be added to the paper would be some brief comments on what a user has to do to install it, and some experience by real users.

Weaknesses: The only thing that could usefully be added to the paper would be some brief comments on what a user has to do to install it, and some experience by real users.

Reviewer #4

Summary: This paper describes an extension to Javascript, that will let researchers run latency and throughput experiments directly from the browser, rather than asking users to install new software to run end-user experiments Paper strengths The authors have designed a Javascript extension to expose various APIs to the web page developer. The extensions not only allow browser-based measurement experiments, but also allows ping and traceroute-like experiments.

To provide security, the authors request the user to confirm the set of APIs that the user will allow, and for accountability, requires the experimenters to sign the code. The users can also allow a set of IP addresses (although, they don't seem to have a blacklist of IP addresses, but that may be trivial to do)

The authors show that measurements performed from the browser are accurate, and implement three case studies using Fathom—a Netalyzer alternative, a connection debugger application, and a web services debugging application. The main point of these case studies is to show that these capabilities can easily be implemented using Javascript without a lot of effort (e.g., 400 loc for the debugging application)

Weaknesses: Fathom has been designed to allow experimenters to easily deploy their experiments (see intro). Yet, I wasn't sure how Fathom will be used by experimenters. Should the user visit a particular web page to run an experiment. If so, how will it be used to do continuous (or at least near continuous) measurements, without requiring the user to continually visit a site. Injecting Fathom into a popular webpage such as Google Maps is not a viable option, as that may need changes to the server.

While I think allowing measurements from the browser (similar to Netalyzer) itself is useful, I am not sure I buy the motivation about making deployability easier.

Comments to authors: Other researchers have explored the topic of extending javascript APIs to have more control. Please see reference "Rivet: Browser-agnostic Remote Debugging for Web Application". It will be useful to explain why Fathom is architecturally different from Rivet.

It appears that some of the more interesting aspects of Fathom, which allows it to do low level socket I/O can be currently realized only on Firefox. Can the authors explain how much effort will go into making this possible on other browsers?

The authors claim that the Fathom's overhead is not additive as javascript execution does not block resource loading. While this is true, javascript execution is infact a huge bottleneck. It stalls the changes to the DOM tree and newer resources cannot be fetched until the script is executed. It appears that this does not affect the overhead of the web pages that the authors test, but I don't know that this will not affect the overhead of *any* web page. Specifically, although the authors claim that they test Fathom with web pages with diverse javascript complexity, they do not quantify/explain this complexity diversity. In fact, the websites chosen are popular web pages that are likely optimized.

In Section 5.4, paragraph 2, the authors mention the use of Chrome threads But I thought Fathom is implemented in Firefox.

I am not sure how easy it is for users to verify the code that experimenters install. Is there an easy way to let users know all of the privileges that that javascript or the web page can access. I realize that this problem exists even when users download experiment software. But since the authors discuss a lot about security, I will be interested in seeing how exactly the user privileges are presented, for it to be easily understandable Similarly, how does a client set its policies?

The writing is often grandiose and long-winded. This makes reading harder. For example, "We begin my framing desiderata for a ..", "Fathom has a potential to flourish where other measurement systems have floundered", etc. Comments to PC(hidden from authors) between 2 and 3. I said 3 because I am not very familiar with this topic. My main concern was with respect how experimenters will use Fathom to conduct experiments.

Reviewer #5

Summary: The paper introduces a browser-based measurement platform called Fathom for Firefox which provides an API to program common measurement tasks in JavaScript. The authors have implemented several measurement capabilities and have designed the system with emphasis on security and privacy. For example, fathom uses code signatures to trust executed scripts. Most of the paper is about the low-level design and implementation details of the platform. In addition, the paper shows that fathom imposes less than 3% overhead in page load times. Finally, the paper analyzes a number of case studies. In particular, they report that it was much easier to implement Netalyzr scripts in fathom than in JavaScript. The authors also implemented a feature for debugging web access failures only in 400 LOC and a useful extension for Google maps that uses fathom to debug connectivity problems.

Strengths: Fathom is indeed a very useful tool that the authors to their credit made available to the community.

Weaknesses: Almost the entire paper is about low-level design and implementation details of a Firefox extension, which I found boring.

Comments to authors: I liked the code signing feature of Fathom. It was a bit disappointing to see later on that you have not implemented this feature yet.

How much cross-traffic does iperf generate? It was interesting to see that cross-traffic has a significant impact on the accuracy of the timestamps.

How is the baseline value used in the Google maps case study determined? It seems that the authors use the metrics from the 20 first sessions, but this was not entirely clear.

PC discussion summary: The reviewers unanimously supported accepting this paper, but wanted the authors to adjust the text of the paper to accommodate concerns mentioned in the reviews, e.g., list of measurements that websites cannot perform themselves today but that Fathom allows, how the default config supports user security/privacy, incentives for users to install the extension, limitations of architecture.

Response from the Authors

In response to the reviewer's comments, we have made the following changes:

- We have added the evaluation specifics the reviewers have requested.
- We have expanded the Discussion with a treatment of incentives for adoption and support for long-running experiments.
- We have stressed the limitations the Fathom API puts in place to prevent deeply invasive monitoring as suggested by reviewer #1.
- We have strengthened the web site operator use case by clarifying specific capabilities Fathom enables in this context.
- We have clarified various technical details e.g. regarding Chrome workers, Firefox availability on Android, etc.