Consolidated Review of

Appraising the Delay Accuracy in Browser-based Network Measurement

1. Strengths:

I found this paper to be very interesting. This paper sheds light on the inaccuracy of round-trip time measurement of ten HTTP-based and TCP socket-based tools. The paper does a systematic and extensive study of a large set of methods, browsers, and operating systems and explores several different combinations to characterize the overall delay. It finds that several existing methods can give varying delay measurements depending on the OS+browser combination. I especially found it interesting that two back-to-back measurements can be negatively correlated. It does make sense because of the complexity of browser implementation/DATE implementation etc.

2. Weaknesses

This paper does not include experiments on Mac systems and mobile Internet devices. Many methods have recently been developed to use browsers on mobile devices to measure round-trip time.

Not sure how novel this finding is, because there seems to be related work on this topic, but it is definitely comprehensive. Qualitative results are already known in previous work (e.g., [9,11]).

It does not address the problem of precision in the results and provides limited discussion of what it means to be "accurate". Some measurements and conclusions appear dubious. In addition to visual graphs, are there statistical test to indicate WebSocket is superior across all methods?

The writing can be significantly improved

3. Comments

This paper performs a thorough study on the accuracy of round-trip delay measurements by a variety of browser-based network measurement tools. I especially like that the paper comprehensively presents taxonomy of existing systems and compares the results across many different systems and browsers. As many end-uses rely on such tools to estimate network performance of broadband or wireless connections, it is important to gain an in-depth understanding of the accuracy of such measurements. This paper could consider including another major operating system: Mac OS in the experiments. In addition, this paper could expand the study with mobile Internet devices such as smart phones and tablets, for which several browser-based measurement tools are available today to measure round-trip time.

I am not sure what you mean by delay overhead? How do you measure this? Is it simply the additional overhead on top of the 50ms RTT (assuming the two servers are close enough that they should barely add any additional RTT)? It isn't clear from the presentation. You claim that the Java Applet will underestimate the RTT. How? If anything RTT can be inflated due to overheads. As a ground truth, it would have been useful to compare this with a non-browser socket based rtt measurement (or a ping). It will also be useful to understand how these delays add up. If I send multiple packets to measure throughput, are these delays going to add up. If so, by how much? (Again use a non-browser based measurement for ground truth.) Have you considered the interaction between the computational unit and the load unit in the overhead? Is there a computational overhead because of the execution of Javascript, for example, or thread scheduling?

The authors need to deal with precision and accuracy systematically. The paper does not discuss the ranges seen in the box and whisker plots and the possible system conditions that lead to them. The correlation discussion concludes that compile time optimizations may lead to reordering of the measurements in their data. Hence while the result indicates how such measurement methodologies need to be carefully designed otherwise they could provide misleading results. It does not provide any additional insight into true and accurate delay experienced by browsers. The paper does not include any OS or process scheduling information (/proc on Linux), which would provide a more holistic picture about the total delay experienced by the browsers and the interaction between the various processes and threads. Do the network measurements from pcap have low variance across experiments?

I have concerns about the validity of some of the results. First, Fig 2(h-j) show that many of the windows browser results have negative overheads. This doesn't make sense unless the timestamps within the browser are not accurate (as the browser's timestamps should occur strictly before and strictly after the timestamps recorded by the network stack in the pcap trace). I suspect that in those scenarios the granularity of the Java applet timestamps may be 10ms rather than 1ms. Second, the authors conjecture that the reason behind the negative correlation between the two measurements is due to potential instruction reordering in the java measurements. However, I find this unlikely as the timestamp and send/receive operations both invoke system calls and it is unlikely that the JVM would reorder system calls since they can have side effects. Isn't it more likely that this is due to a time resolution issue? I would check your timestamp resolution in the java measurements. After discussion with other reviewers, I feel that these explanatory problems can be fixed with (at least) more discussion of the hypothesis or (in the best case) additional validation experiments. I would very much appreciate it if the authors could verify that the negative results they observed are not solely due to timestamp granularity (or if so, discuss the implications of that and whether it was just an artifact of their experimental setup). Note that there has also been some passive measurement work comparing socket and HTTP based RTT measurement methods that have found similar variability in HTTP methods even when no RTT variability exists: Can you GET me now?: estimating the time-to-first-byte of HTTP transactions with passive measurements. IMC 2012.

The graphs are extremely hard to read. The y-axis ranges in Figure 2 vary significantly and hence do not allow the reader to visually compare two graph results directly. Figure 2 presents a number of box plots to show the delay overheads incurred by different browsers/OSes for each method. It is also interesting if the paper shows the results by browser types (draw one plot for every browser) and compares the delay overheads of different methods on the same type of browsers. Also, Figure 3, the CDF plots overlay. The plots in Figure 4 are hard to read. The authors could possibly reduce the number of results plotted on the same graphs to make the plots clearer.

4. Summary from PC Discussion

We like the fact that the study was comprehensive and was a well-done comparison of techniques. The paper could be improved two ways: First, the authors should clarify what is meant to be "accurate." Second, they should look into or discuss alternative explanations for their counter-intuitive findings (see the reviews).

5. Authors' Response

In our camera-ready version, we clarify the meaning of "accuracy" and "delay overhead." We follow the ISO 5725 standard and define the accuracy of a network measurement as how the measurement results deviate from the real network performance. There are two meanings for accuracy: trueness and precision. The former refers to how close the measurement result is compared with the actual value. The latter is related to the repeatability of the measurement, that is, whether the measurement can get a consistent result when being repeated. A network measurement is considered more accurate if its produced results are closer to the actual values (trueness) and are more consistent (precision or repeatability). Generally, network measurement accuracy depends on a number of factors, including the correctness of adopted methodology, time resolution, system load, and so on.

We also clarify the definition of "delay overhead" in the Introduction section. It is the difference between the measured value and the actual value. For browser-based measurement, the effect of the overhead on the client depends on how the rendering engine (e.g., JavaScript engine) interprets the measurement code and invokes system function calls. In our experiments, we use equation (1) to estimate the delay overhead Δd:

$$\Delta d = (t_r^B - t_s^B) - (t_r^N - t_s^N),$$

where t_r^B and t_s^B are the timestamps recorded by browser, and t_r^N and t_s^N by tcpdump/Windump. During the measurement, we also made sure that there were no cross traffic, packet loss, and retransmissions. Although the web server could bias the RTT, the subtraction of $t_r^B - t_s^B$ and $t_r^N - t_s^N$ in the same measurement round can mitigate the bias, if any.

For the counter-intuitive findings of the Java applet cases, we performed a new set of experiments to identify the root causes. We first used appletviewer provided by Java Development Kit (JDK) to load the applets directly, instead of running them within the browsers. This setup can eliminate the influence of browsers and their corresponding Java Plug-ins on the measurement results. Similar under-estimation and discrete levels of values can still be observed in this set of experiments. Hence, we can conclude that the source for the counter-intuitive findings comes from the Java applet, instead of the browser. We then tested the real granularity of the timing function (`Date.getTime()`) with a piece of simple loop code. The results show that the granularity is not a constant value. It can be 1 ms or ~15 ms. Each possible value will last for a period of time (several minutes) and then change to other values. Both 32-bit JRE and 64-bit JRE exhibit the same behavior. To further validate our findings, we analyzed the data of the delay overhead experiments. The time resolution obtained from our analysis concurs with the timestamp granularity obtained from the test codes. As a last step, we replaced the timing function with `System.nanoTime()`. The under-estimation and variation of RTT disappeared after the replacement. Therefore, we conclude that the bizarre delay overheads in Windows are caused only by the timestamp granularity of the `Date.getTime()` function. Our inspection of some implementations shows that many of Java applet tools, for example, Netalyzr, NDT, AuditMyPc, and so on, are still using the function `Date.getTime()` or `System.currentTimeMillis()`. Switching to the more precise function `System.nanoTime()` can greatly improve their accuracy in Windows.

Another issue is what could have led to the variation in the measurements, given that system conditions were tightly controlled. Although we did not record the system load, we made sure that all the necessary processes (e.g., explorer.exe in Windows, init in Linux, and so on) were running in the background. However, there were still some other programs, such as packet capturing program and automation scripts, need to be dynamically invoked during the measurement procedure. Moreover, the browsers themselves need to consume resources to render the measurement objects. As a result, the delay overheads may still vary, depending on how sensitive the measurement methods are to the system load.

Due to the page limit, we cannot expand our work further. But we totally agree that expanding the work to include mobile devices is very interesting and including Mac OS can make this paper more comprehensive. We will study them in our future work.