Computer Network
Yingcong Li, S1616245

Survey for the Accuracy of Network Performance Tools

Recent years, more and more companies and applications are offering their services to end-users by the network, measuring and understanding the performance of the network becomes more important. This leads to a growth in network measurement tools in different platforms and gains a large popularity. But the accuracy of these tools has not been sufficiently tested and little known by end users. To test the (in)accuracy of these network performance tools, Li et al. performed a series of experiments to appraise the performance of browser-based tools and smartphone-based tools. In addition, they also tried to explore the reason for the delay overflow introduced by these tools. On the other hand, Pelsser et al. turned their focus on the accuracy of a commonly used tool in PC -- *ping*. The purpose of the survey is to appraise the suitability of these network measurement tools to measure the network delay, or how they are close to reflecting the actual network delay. In this survey, the results of their works will be given, finally, we try to get a general conclusion and point out the opportunities for future work.

Pelsser et al. (2013) pointed out that the widely used network measurement tool--*ping*--is not accurate. A set of fields--Type, Code, Checksum--are combined to identify a flow (denoted as *flow-id*). Probes with the same flow-id can decrease the influence incurred by the forward and return path. They used *tokyo-ping* (modified version of *paris-traceroute*) to get the RTTs for each flow-id and compared the results with ping's results. Results show that ping is able to report the upper and lower bound of RTTs, but significantly overestimates the jitter. The RTT variability in each individual flow (captured by *tokyo-ping*) is much smaller than ping's results (4ms vs. 0.8ms). More rigorous experiment with 32 different flow-ids shows that the result from ping is actually a collection of each individual flow. Main causes of the delay diversity between each flow can be the LAG and ECMP, but a more detailed diagnosis cannot be achieved due to some restrictions. This inaccuracy in *ping* may influence the correctness of other researches such the accuracy of some smartphone-based network measurement tools which we will talk about later.

Browsers is a common entry for us to get connected to with the Internet, so many network measurement tools are built in browser platform. Experiments conducted by Li et al. (2013) compared the performance of some browser-based network measurement tools in different browsers, operating systems and different approaches (HTTP-based and Socket-based). The results show that all these approaches experience delay overhead in different degrees, ranging from minus 5 ms to over 120 ms, these significant delay overheads (or underestimates) are too large to ignore. But delay overheads by Socket-bash approach methods are much smaller and more constant than HTTP-based approach. The median overheads are mostly less than 1 ms. As a result, POST method suffers the delay caused by connection rebuilding and Java timing function *Date.getTime()* will introduce a delay underestimate in Java applet. Java socket method shows a comparable performance with tcpdump/Windump when *System.nanoTime()* is adapted.

Due to the performance and storage of smartphones, many apps provide their service rely on the network. Both wifi and cellular network measurement apps can be downloaded in all app stores. Li et al. (2015) also performed a performance measurement to appraise the accuracy of some smartphone-based network performance measurement tools in Android system. Three methods--Native ping, Inet ping and HTTP ping--are tested. As the result, the

RTT measured by apps are inflated significantly among three phones, and these three methods also cause various delay overheads and asymmetry between sending and receiving in the same smartphone. The delay overheads range from 1 to 15 ms. Some delay underestimates are incurred by the inaccuracy of *ping* as we talked about in the second paragraph. Parts of the reason for the delay and asymmetry are incurred by the DVM--the running environment of an app, more instructions have to be executed than a native Linux program. Bypassing the DVM by using a pre-compiled C program which supports RTT measurements with HTTP messages, the delay overhead introduced in application-kernel can be mitigated to ~1 ms in both Inet ping and HTTP ping. And total delay overhead can be controlled under 5 ms. But the delay incurred in kernel-hardware part is still unknown.

Li et al. (2016) conducted other experiments focusing on the delay caused in the kernel-hardware by measuring the nRTTs on Google Nexus 4 and 5 in two different sending intervals (10ms and 1000ms). Results show that when the sending interval is small (10ms), both two phones report a small kernel-hardware overhead, which is smaller than 4 ms. While turning into long sending interval (1000ms), delay overheads in Nexus 5 grow to ~18ms, nearly two times more than Nexus 4 (around ~6ms). On the other hand, the application-kernel delay is close to 0 on both phones. Also, some delay underestimates in application-kernel is due to the low resolution of *ping* results. Looking into the source code of the WNIC driver. They found that the driver puts the SDIO bus to the sleeping mode when the data rate is not high, more time is needed to bring the SDIO bus to wake up. In a similar way, the PSM (Power Saving Mode, PSM) also could inflate the nRTT, for example, if the actual nRTT is longer than the PSM timeout, a smartphone will turn into PSM mode before receiving the reply message, this will introduce additional delay. They modified AcuteMon to send background traffics to keep the smartphone and SDIO bus active. In this approach, the overall median delay overheads are controlled within 3ms using their AcuteMon. This is the most accurate delay measurement that can be achieved on Android platform so far without hacking the system.

To sum up, all these network measurement tools in different platforms will introduce a delay overhead on RTT in different degrees. It is obvious that native-call method always can decrease the delay overheads than other built-in tools on all these platforms. Timing functions in different platforms are also another important reason for the inaccuracy, and they always introduce some underestimates to the results. But it is good to see that researchers' approaches have significantly decreased the delay overheads in all platforms. Java socket method has the same performance with tcpdump in browser platform and modified AcuteMon only overestimates delay for ~3ms. In the future, browser-based measurement can be extended to smartphone browser. These accuracy measurements can be conducted in the cellular network. And the accuracy of tcpdump/Windump is also can be considered since they are the benchmark for browser-based and smartphone-based measurements.