A Study into the Accuracy of Smartphone-based Mobile Network Measurement

Troy Pandhumsoporn

Xiaosu Ye

Cheng Song

Outline

- Background
- Problem Statement
- Solutions
- Results & Analysis
- Future Steps

BACKGROUND

- Mobile devices have become essential parts of our daily lives
- Recent reports shows that 84% of apps require permission of Internet access from a pool of 55K Android apps randomly picked from the official Android app market
- Monitoring mobile network quality is important
- Measuring and Understanding the performance of mobile network is important.
- Wide variety of many measurement apps exist, but lack of accuracy.

WHAT IS THE PROBLEM?

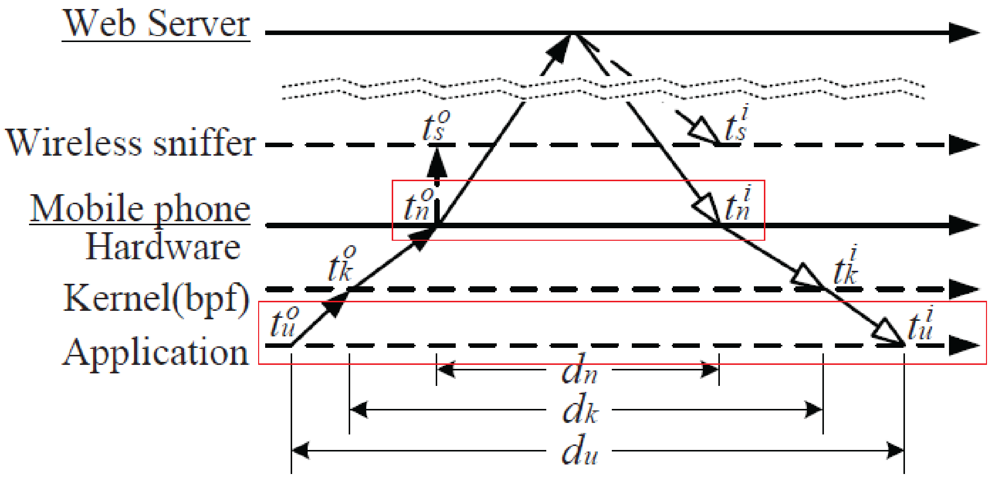
- The measurement accuracy of current apps has not received sufficient scrutiny/ still remain questionable.

- Same mobile devices may have different network quality performance using different apps.

OVERHEAD DELAY MEASUREMENT

- The measurement app uses du as the network RTT (measured network delay)
- The actual RTT is given by dn (actual network delay)
- The delay overhead is given by Δd. (the difference between the measured and actual delay)

$$\Delta d = d_u - d_n = (t_u^i - t_u^o) - (t_n^i - t_n^o).$$

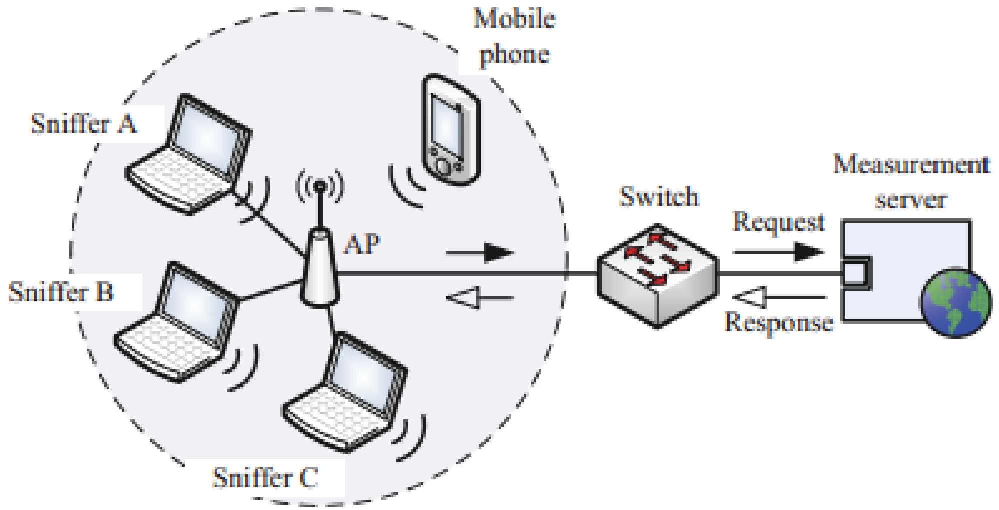


Measurement flow for Android apps.

A probe packet starts at left end $t_u{}^o$ to a web server, eliciting a response packet from the server to arrive at the measurement app at time $t_u{}^i$

Set up a reliable testbed environment

- Measurement server
- Three Android phones
- Three external packet sniffer due to the reliability requirement
- Run three test apps (RTT measurement method) one by one on each phone
- Comparing three RTT measurement models:
  - Native Ping
  - Inet Ping
  - HTTP ping



INTRODUCE 3 RTT MEASUREMENT METHODS

- Native ping

  This method executes external shell commands through a Java Runtime class.

- Internet ping

  The measurement app can also employ the network related classes provided by

  Android or Java, such as class java.net.InetAddress.

- HTTP ping

  HTTP-based classes, such as class java.net.HttpURLConnection, can also be

  used for implementing a measurement app.

Detailed hardware configurations and OS versions

TABLE I: The mobile phones used in the experiment.

Models	OS Ver.	Hardware specifications
Google Nexus 5	4.4.2	Qualia MSM8974 Snapdragon 800 CPU (quad-core 2.26GHz), 2GB MEM
HTC One	4.2.2	Qualia APQ8064T Snapdragon 600 CPU (quad-core 1.7GHz), 2GB MEM
Sony Xperia J	4.0.4	Qualia MSM7227A CPU (1GHz), 512M MEM

EVALUATION

- Most of the delay overheads are observed as RTT-independent
- G phone's delay overheads corretlate with the RTTs:
  - Smaller overhead for short RTTs
  - Larger overhead for long RTTs
- By comparing the results from Inet ping and HTTP ping which use TCP SYN/RST packets and TCP data packets, establishing a new TCP connection may incur a higher delay overhead than processing content in an existing connection.

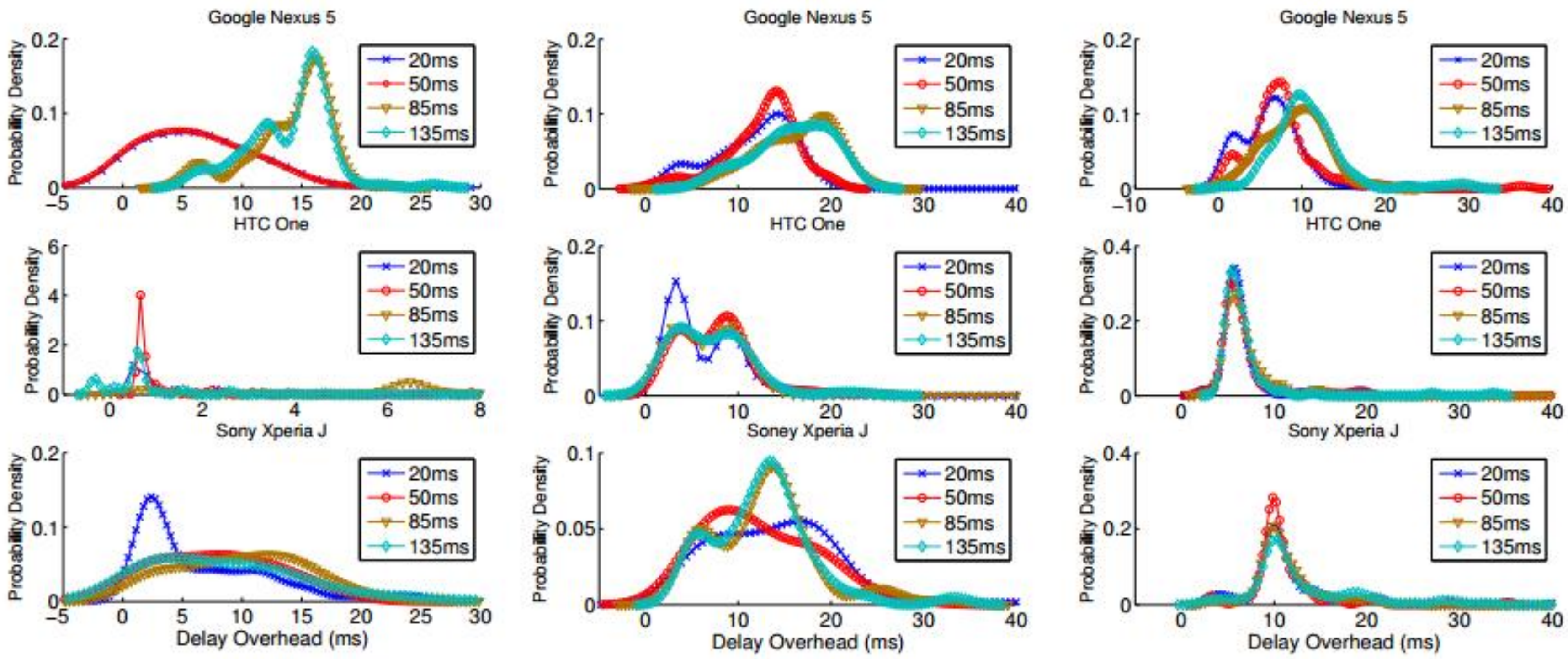
TABLE II: Delay overheads measured when `System.currentTimeMillis()` is used (mean with 95% confidence interval, in ms).

	Phone*	Emulated RTT (ms)			
		20	50	85	135
Native ping	G	7.700 ±2.331	6.028 ±0.811	14.078 ±0.684	13.963 ±0.691
	H	2.108 ±0.726	1.177 ±0.292	5.179 ±0.564	0.849 ±0.281
	S	6.779 ±1.129	7.840 ±0.932	9.999 ±1.039	8.387 ±1.191
Inet ping	G	11.931 ±1.063	12.514 ±0.779	16.211 ±0.833	15.874 ±0.787
	H	7.243 ±1.907	7.470 ±0.815	8.551 ±2.413	7.060 ±0.821
	S	13.822 ±1.327	12.223 ±1.142	12.814 ±1.146	12.511 ±1.055
HTTP ping	G	6.481 ±0.855	7.651 ±0.963	9.156 ±0.703	10.790 ±0.911
	H	6.566 ±0.588	7.151 ±0.957	7.222 ±1.041	6.675 ±0.739
	S	11.206 ±0.947	11.153 ±0.855	11.805 ±0.987	12.987 ±1.312

Note ∗: G for Google Nexus 5, H for HTC One, and S for Sony Xperia J.

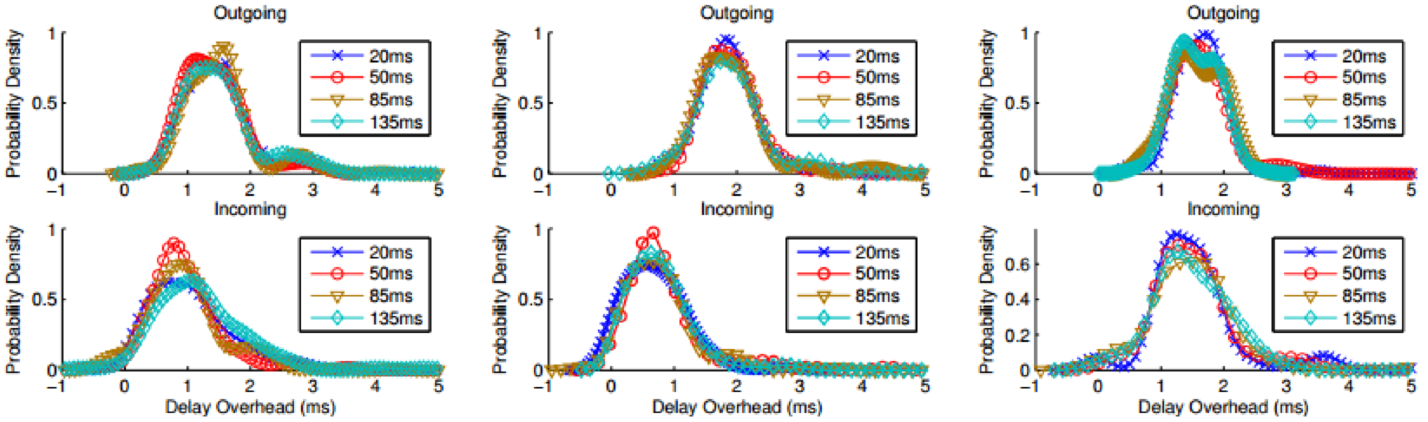
Probability Density Function Plots of the delay overheads (apps and Phones)

Legend represents RTT measurement



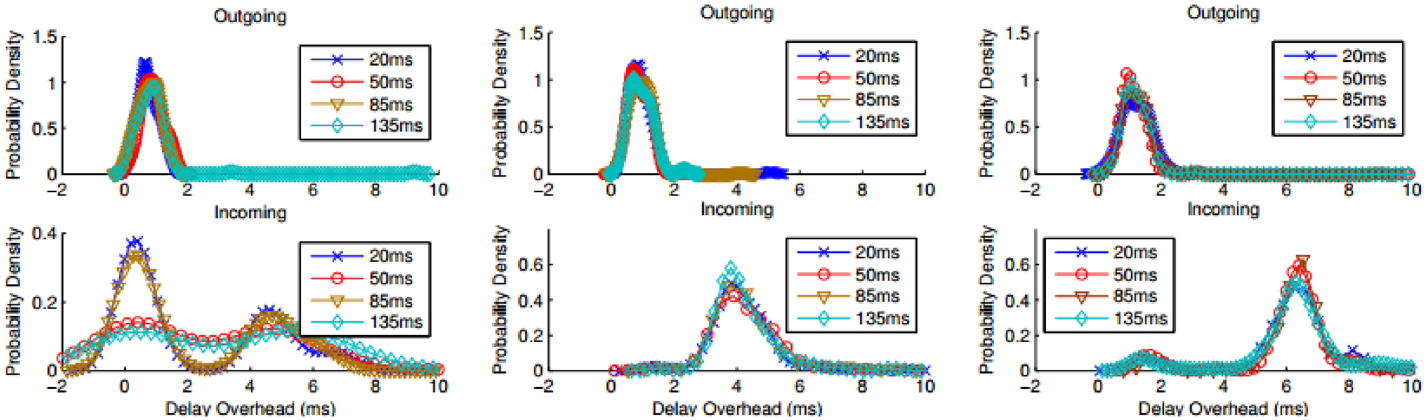
(a) Native ping. (b) Inet ping. (c) HTTP ping.

PDF Plots Of The Delay Overhead Asymmetry (Inet Ping / HTTP Ping)



(a) Google Nexus 5. (b) HTC One. (c) Sony Xperia J.

Fig. 7: PDF plots of the delay overhead asymmetry (Inet ping).



(a) Google Nexus 5. (b) HTC One. (c) Sony Xperia J.

Fig. 8: PDF plots of the delay overhead asymmetry (HTTP ping).

WHAT CAN WE ASSESS FROM THIS TABLE?

- The RTTs measured by the apps are inflated significantly for all three phones.
- The delay overheads can range from a few milliseconds to tens of milliseconds, and the 95% confidence interval can be as high as 2.4ms.
- The same Android phone has different performance for different measurement methods.
- HTTP ping and Native ping exhibit comparatively smaller delay overheads for most of the cases.
- By comparing the results from Inet ping and HTTP ping, we find that establishing a new TCP connection may incur a higher delay overhead than processing content in an existing connection.

A BETTER PRACTICE

TABLE III: A comparison of Δd_u (Ext) for external C socket program and in-DVM measurement (App) (mean with 95% confidence interval, in ms).

	Type	Emulated RTT (ms)			
		20	50	85	135
Inet ping	App	2.946 ±0.695	2.443 ±0.200	2.637 ±0.251	2.828 ±0.236
Inet ping	Ext	0.736 ±0.121	0.794 ±0.139	0.798 ±0.154	0.830 ±0.134
HTTP ping	App	3.312 ±0.663	3.824 ±0.721	3.157 ±0.540	4.542 ±0.834
HTTP ping	Ext	1.095 ±0.075	1.246 ±0.098	1.289 ±0.112	1.365 ±0.186

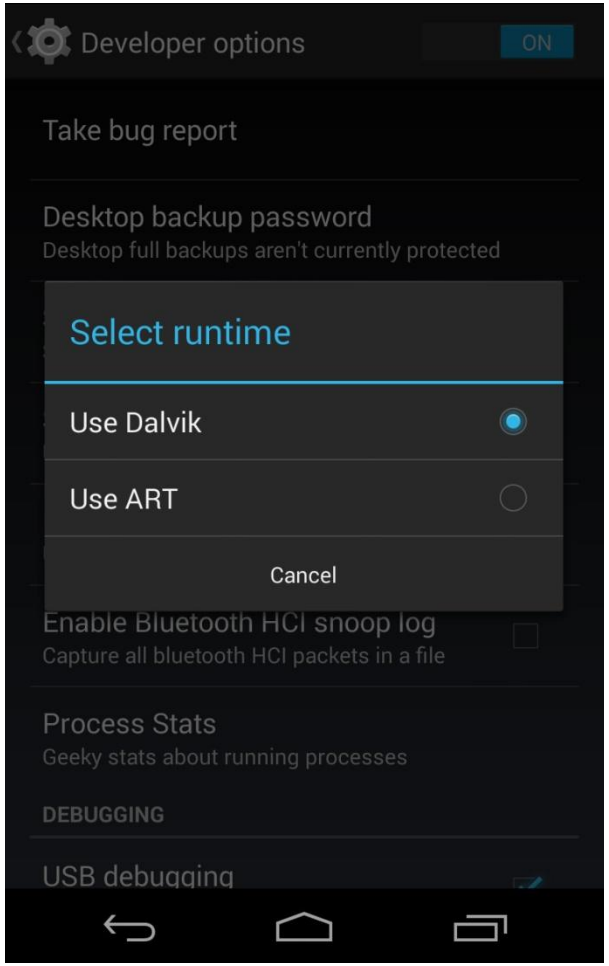
- To validate whether bypassing the DVM can mitigate the delay overhead
  - Implement a simple C socket program for Inet ping
  - Limit the size of messages to no more than 300 bytes for HTTP ping
- The kernel delay overheads for handling HTTP messages are more stable than for TCP control message.

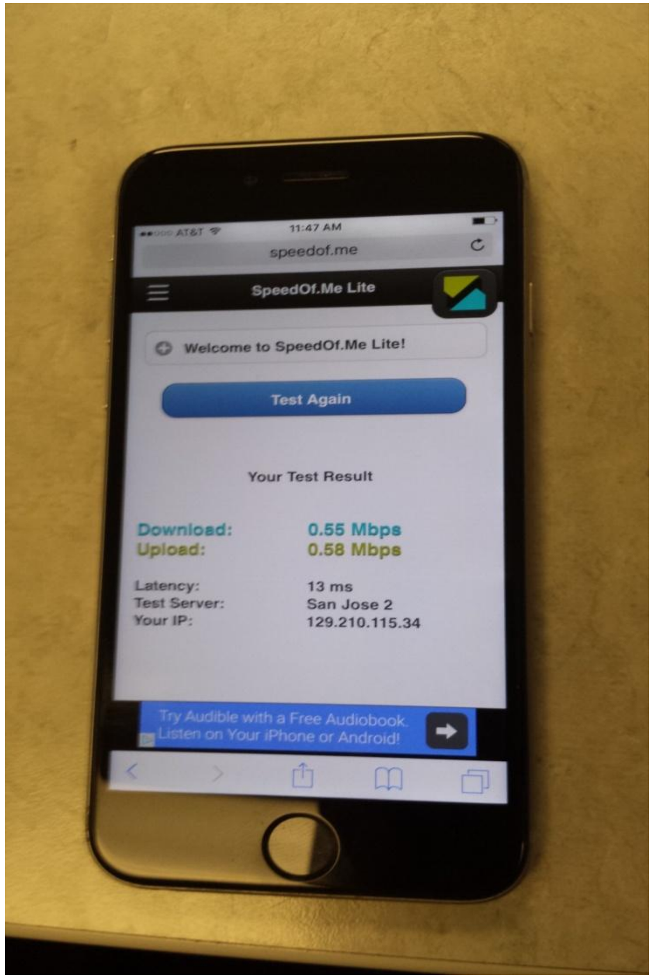
Conclusion

- The RTTs measured by the apps are significantly inflated.
- The delay overhead introduced by the DVM is not negligible and symmetric in the send and receive directions.
- The solution to the delay overhead introduced by the DVM (not negligible) was to implement a native measurement app using HTTP messages for measurement.
- The results yielded a delay overhead reduced to five milliseconds or less.

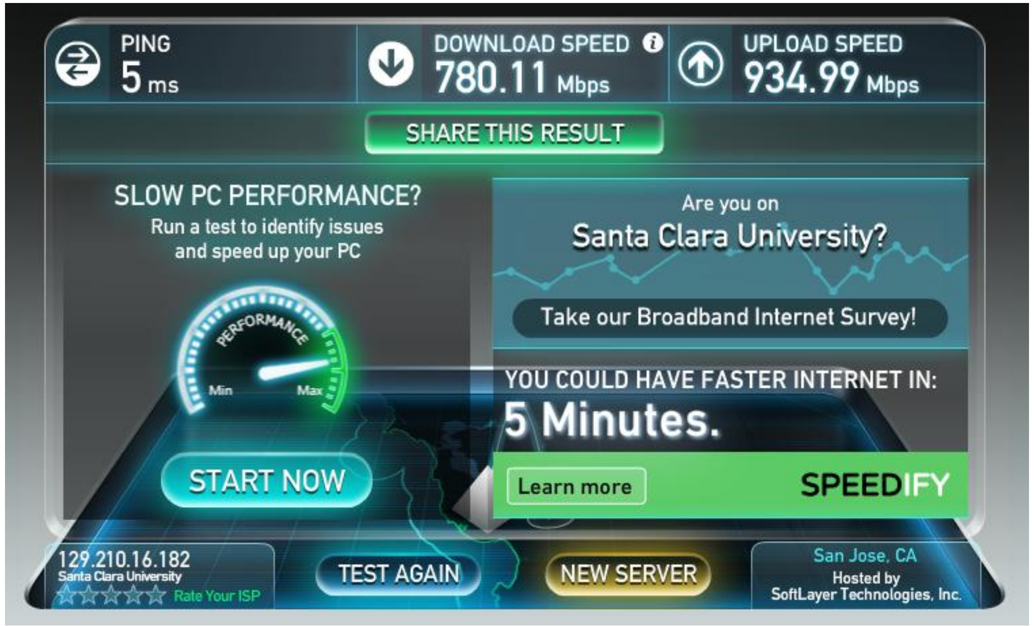
IN THE FUTURE (THE NEXT STEP)

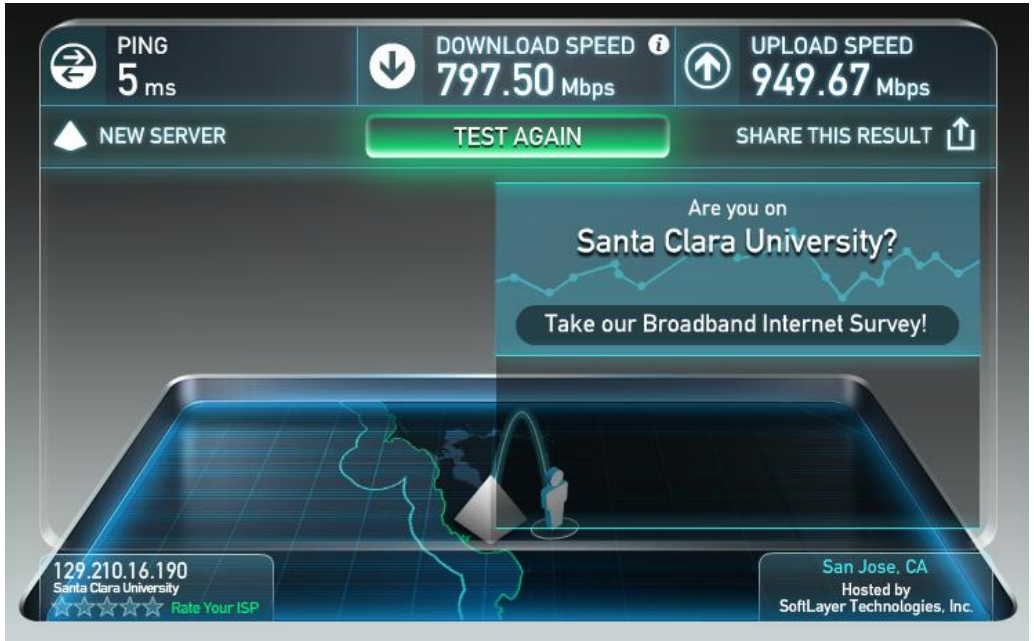
- Similar study for iOS platform (Iphone series)
  - Android vs. iOS platform comparison
- Fixed Network (Desktop) Network Measurements
- Updated Experiment (Testbed Setup):
  - Better Computer Hardware
  - VM (Dalvik vs ART)
  - Selection of Smartphones





Desktop Network Measurement for FUN





Works Cited

- [1] Weichao Li; Mok, R.K.P.; Daoyuan Wu; Chang, R.K.C., "On the accuracy of smartphone-based mobile network measurement," in *Computer Communications (INFOCOM), 2015 IEEE Conference on* , vol., no., pp.370-378, April 26 2015-May 1 2015 doi: 10.1109/INFOCOM.2015.7218402 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7218402&isnumber=7218353

- [2] Anthony, Sebastian. "Android ART: Google Finally Moves to Replace Dalvik, to Boost Performance and Battery Life | ExtremeTech." *ExtremeTech*. Ziff Davis, 8 Nov. 2013. Web. 03 Nov. 2015.

- [3] "ART and Dalvik." *ART and Dalvik*. Android, n.d. Web. 03 Nov. 2015. <https://source.android.com/devices/tech/dalvik/index.html>.