A System for Crowdsourcing Passive Mobile Network Measurements

Emmanouil Alimpertis
Ph.D. Student, UC Irvine

Athina Markopoulou
Associate Prof., UC Irvine

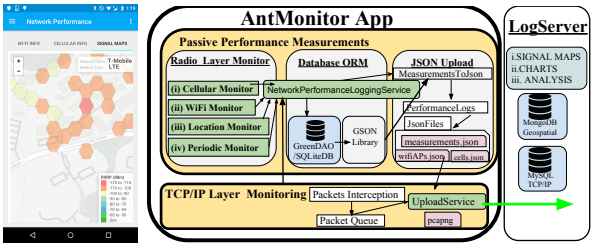
Abstract

In this poster, we build on top of the AntMonitor system and we extend it to add a new module for passive monitoring network performance on mobile devices (w/o middle server). We show that this design is efficient yet powerful, in terms of the performance achieved (compared to alternatives today), realistic measurements (passive monitoring of actual traffic on devices of real end-users), richness of information (fine-grained measurements and rich contextual information available on the devices). We present the system design, a preliminary evaluation, and example applications.

1 Introduction

Crowdsourcing wireless (both cellular and WiFi) network performance measurements (*e.g.* cellular received signal strength, RSS, WiFi RSS, TCP/IP throughput *etc.*) can enable a vast number of network monitoring applications (*e.g.* mobile coverage maps and localization [1]). It can be of great interest to individual users, network operators (who often outsource these measurements to third parties) and researchers and can be part of applications and solutions for next generation wireless networks. In this poster, we propose a system for network performance measurements from the vantage point of mobile devices. It passively monitors network properties and activity only on the device (without redirecting traffic to a middle server), runs as a user-space app in the background seamless to the user, and provides a powerful tool for crowdsourcing such measurements from the mobile devices to a LogServer for processing.

Relevant prior art can be categorized in passive or active monitoring either at network infrastructure or network edge. Monitoring at network infrastructure [2] offer granular information in large scale, however, it misses the network edge. Passive measurements (*e.g.*OpenSignal) at the mobile capture granular performance metrics of the wireless link but they do not record detailed stats per TCP/IP flow. Finally, active measurements (*e.g.* throughput or ping with Speedtest.net) introduce data overhead and must be triggered by the users. Most closely related to our work is Mobilyzer [3], which offers a library with controllable active and passive measurements. However, it offers neither passive throughput measurements nor monitoring per each distinct TCP/IP flow, which are possible with our work along with a whole range of fine-grained metrics.



(a) Android App (b) Network performance monitoring system architecture.

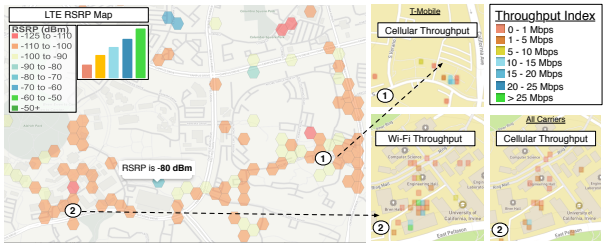
Figure 1: Network performance monitoring is built as part of AntMonitor [4] and it can be a powerful tool for crowdsourcing rich, fine-grained, large-scale, network performance measurements.

2 Network Monitoring System

This work builds on and extends AntMonitor[1] [4] – an Android app we have previously developed for capturing and analyzing network traffic in and out of mobile devices. AntMonitor is VPN-based and thus can be installed without rooting the phone, and run in the background. In prior work [4], we focused on the system design of AntMonitor, we demonstrated its excellent performance in terms of throughput, battery consumption and other resource usage, and we focused on the particular application of privacy leaks detection; we refer to [4] for details. This poster extends AntMonitor by adding a new module to perform network performance measurements, as shown in Fig. 1, thus it inherits its good system properties mentioned (user-space app, seamless user experience, on-device monitoring without redirecting traffic to a server), which makes it ideal for crowdsourcing.

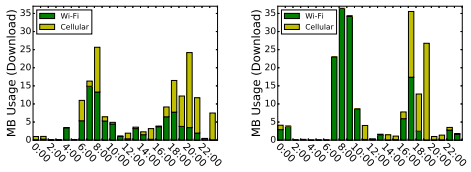
Collecting Measurements on the Device: First, since AntMonitor can intercept and analyze in real-time every packet in and out of a mobile's networks interfaces, it is naturally positioned to perform passive TCP/IP-layer measurements. For example, we compute throughput per flow by counting the number of bytes per timestamp. Capturing and logging packets and computing TCP/IP flow statistics is achieved in real-time with minimal performance impact [4]. Second, we record the radio layer and network performance metrics obtained from Android APIs including but not limited to: cellular & WiFi RSS, network carrier, radio access technology (RAT), frequency bands use in WiFi *etc.* Measurements' recording is triggered by Android's notifications/callbacks for network and location changes (*e.g.* RSS or cell status change), therefore, monitoring comes with minimal battery overhead. Third, we record rich contextual informa-

[1]The AntMonitor Project: antmonitor.calit2.uci.edu



(a) T-Mobile LTE Signal Strenth, Wi-Fi & Cellular Throughput Maps

Figure 2: Performance maps from the university campus. Low RSRP (loc. 1) does not necessarily mean low cellular throughput (for the same carrier). Visualizaitons using open source libraries `leaflet.js` and `D3.js`



(a) Week Typical Day (b) Weekend Typical Day

Figure 3: **Single user's Pattern:** #MB downloaded, averaged over all (a) week or (b) weekend days (one user for one month). Daily pattern differ between week and weekend.

tion at the time of the measurement, including: location (Google Location API provides mechanisms for low energy footprint), timestamp, and apps running.

Storing, Uploading and Processing Measurements: The measurements are saved locally in a SQLite database (DB) by utilizing an object-relational mapping (ORM) library. ORM is an middleware which automatically maps runtime Java objects to Sqlite relational tables, thus provides easy and efficient data manipulation, without complex SQL statements. The recorded measurements are put in Javascript Object Notation (JSON) format[2] (see Fig. 1) and are uploaded on the `LogServer` (per user's request or when the phone is charging and on WiFi). We use noSQL DB (MongoDB) because it scales better than traditional database systems, allows schema-less storage and MongoDB particularly supports spatio-temporal operations. Off-line processing and visualization (*e.g.* RSS maps) are provided at the `LogServer`. Please note that the `LogServer` is used only for storing measurements and the collection is 100% done on the device.

3 Preliminary Results

Performance: First, we utilize our module to compute *passively* the smartphone's throughput and we compare it to a state-of-the-art *active* monitoring tool (`Speedtest`). Table 1 shows that the values are very close, but our passive approach does not incur any data overhead. Resources usage by these two methods is shown in Table 2.

[2]A JSON payload example is: {"cellID":141705489, "rat":"LTE", "carrier": "AT&T", "location": { "lat":33.644794, "lng":−117.82986 ··· }, "lteInfo" : {"rsrp":-94, ··· }, "wifiInfo": { "SSID": "10:fe:···", "rssi":-49, "freq":2412, ··· }, ··· }.

Exp #	1	2	3	4	5	6	7
AM: W=5	21.24	29.26	22.83	27.01	30.75	26.84	26.14
Speedtest	19.96	28.42	22.39	28.74	31.66	26.98	27.22

Table 1: **Throughput (Download Mbps): Active (using `Speedtest`) vs Passive (using `AntMonitor`: AM) measurements**. First, we ran multiple Speedtests, with 5 min gaps, from the same location, and we list the throughput mentioned by `Speedtest`. Second, we computed the throughput using AntMonitor logs, over a window of 5 sec. Our approach is close to `Speedtest` but does not incur any measurement overhead. For a fair comparison in this table, we passively monitored the Speedtest packets using AntMonitor. In the wild, throughput computations can be made by counting the bytes of actual traffic sent over time.

Metric	Data Overhead	Memory	CPU	Battery
`Speedtest`	50 MB	116 MB	14.7%	−0.5%
`AntMonitor`	0 MB	134 MB	43.4%	−0.7%

Table 2: **Resources Utilization** for `AntMonitor` and `Speedtest` per Exp.

Applications: Second, in order to showcase the applications and versatility of our tool, we report measurements collected on our campus (for a period of 8 months and among approx. 10 people in our group) including: reference signal received power (RSRP) for LTE network and device throughput (both WiFi and cellular). Fig. 2(a) shows the LTE RSRP for one cellular provider on UCI campus, for one month. LTE reception has large spatial variations and the RSRP would be lower than the same link budget RSS in GSM, because they are measured in different bandwidth range. Fig. 2(a) also reports the average throughput of WiFi and LTE networks and compares it to LTE RSRP. Interestingly, we observe that low RSRP does not necessarily result in low throughput. Furthermore, Fig. 3 shows the data (MB) used by one user throughout a typical day per network (WiFi or Cellular). `AntMonitor`'s fine granularity (per flow, per app, per location, over time *etc.*) can enable a vast number of monitoring applications, troubleshooting, SDN operations *etc.* Due to lack of space, we omit other available metrics' visualizations such as WiFi's RSS and frequency channel use per region.

Prototype. A prototype of the system, as shown in Fig. 1(a), is currently in alpha testing[3]. As part of ongoing work, we are planning to include our monitoring module on beta test in `AntMonitor` on GooglePlay (currently focuses on privacy leaks), and to collect and analyze campus-wide measurements.

References

[1] E. Alimpertis, N. Fasarakis-Hilliard, and A. Bletsas. "Community RF Sensing for Source Localization". *IEEE Wireless Commun. Lett.*, 3(4):393–396, Aug 2014.

[2] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk. Gigascope: A Stream Database for Network Applications. In *Proc. of the ACM SIGMOD*, pages 647–651, June 2003.

[3] A. Nikravesh, H. Yao, S. Xu, D. Choffnes, and Z. M. Mao. Mobilyzer: An Open Platform for Controllable Mobile Network Measurements. In *Proc. of the 13th ACM MobiSys*, pages 389–404. May 2015.

[4] A. Shuba, A. Le, E. Alimpertis, M. Gjoka, A. Markopoulou. AntMonitor: System and Applications. *arXiv:1611.04268*.

[3]`zeus.calit2.uci.edu:8080/manos/apk.php`