Network and Application Performance Measurement Challenges on Android Devices

Mohammad A. Hoque
University of Helsinki, Finland
mohammad.a.hoque@helsinki.fi

Ashwin Rao
University of Helsinki, Finland
ashwin.rao@helsinki.fi

Sasu Tarkoma
University of Helsinki, Finland
sasu.tarkoma@helsinki.fi

ABSTRACT

Modern mobile systems are optimized for energy-efficient computation and communications, and these optimizations affect the way they use the network, and thus the performance of the applications. Therefore, understanding network and application performance are essential for debugging, improving user experience, and performance comparison. In recent years, several tools have emerged that analyze network performance of mobile applications in situ with the help of the VPN service. However, there is a limited understanding of how these measurement tools and system optimizations affect the network and application performance. This paper first demonstrates that mobile systems employ energy-aware system hardware tuning, affecting network latency and throughput. We next show that the VPN-based tools, such as Lumen, PrivacyGuard, and Video Optimizer, aid in ambiguous network performance measurements and degrade the application performance. Our findings suggest that sound Internet traffic measurement on Android devices requires a good understanding of the device, networks, measurement tools, and applications.

CCS CONCEPTS

• **Networks → Network performance evaluation**; • **Computer systems organization → Embedded systems**.

KEYWORDS

delay, traffic measurement, measurement anomaly, energy optimization.

ACM Reference Format:
Mohammad A. Hoque, Ashwin Rao, and Sasu Tarkoma. 2020. Network and Application Performance Measurement Challenges on Android Devices. In *Proceedings of 38th International Symposium on Computer Performance, Modeling, Measurements and Evaluation (IFIP WG 7.3 Performance'2020).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In situ measurement tools shed light on the network and application performance on mobile devices, however, they may provide imperfect results. The sources for these imperfections can be the implementation of the tools and hardware optimization, such as mobile power management. The former can affect application performance by changing the network flow characteristics, such as introducing delay and changing the protocol headers. The latter is independent of any measurement tool, but it can also affect application performance. Nevertheless, these imperfections can lead to ambiguous results because we may measure something we have not intended to measure [22]. Therefore, it is crucial to associate power management with traffic measurements and set up the measurement procedures accordingly.

In situ Internet traffic measurement tools, such as Video Optimizer (VoP) [23], Lumen [25], PrivacyGuard (PvG) [27], and MopEye [30], have proven to be useful in debugging, improving user experience, and performance comparison of mobile applications. VoP [6], formerly known as ARO [23], is a popular open-source tool for collecting traffic from mobile devices without rooting the device, and it also enables various diagnosis and optimization of applications, network, CPU and GPU through offline analysis [6]. MopEye monitors the networking performance of mobile applications. In contrast, Lumen and PvG are two online traffic analysis tools that help users find privacy leaking incidents. Lumen also provides insights on the TLS usage [25], the CDN usage [20], and the DNS usage of mobile applications [8]. The alternative to these tools is rooting the device and using *tcpdump* for offline analysis.

Our goal for this work is to demonstrate that sound and reproducible measurement on mobile devices requires a thorough understanding of the device. We, therefore, investigate the performance of VoP, Lumen, and PvG. We were unable to study MopEye because we were unable to find it in the Google Play Store and public source code hosting websites, such as GitHub. We focus our attention on these three tools because they exemplify state-of-the-art in situ traffic measurement and analysis tools for Android devices. Note that we do not aim to establish which tool is the best or the worst for a given task. Although each of these tools has different high-level goals, we show that they are good candidates for traffic capturing and can be extended for traffic analysis. Our key observations are as follows.

(1) We show that Android imposes different optimization techniques, such as CPU hot-plugging and dynamic frequency scaling. These two mostly affect network I/O, and thus the network throughput. Similarly, the WiFi optimization, i.e., dynamic modulation scheme, affects the uplink throughput. Therefore, one must be aware of the adaptive performance characteristics of mobile devices when conducting experiments (§2).

(2) VoP delays the outgoing traffic, and PvG delays the incoming traffic in the range of 40-100 ms. Therefore, the corresponding pcap traces from VoP have anomalies in the measurements. To avoid such pitfalls in network and application performance measurements, one must have a good understanding of these applications and tools. In contrast, Lumen is free from such abnormal behavior.

IFIP WG 7.3 Performance'2020, Nov 2–6, 2020, Milan, Italy

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

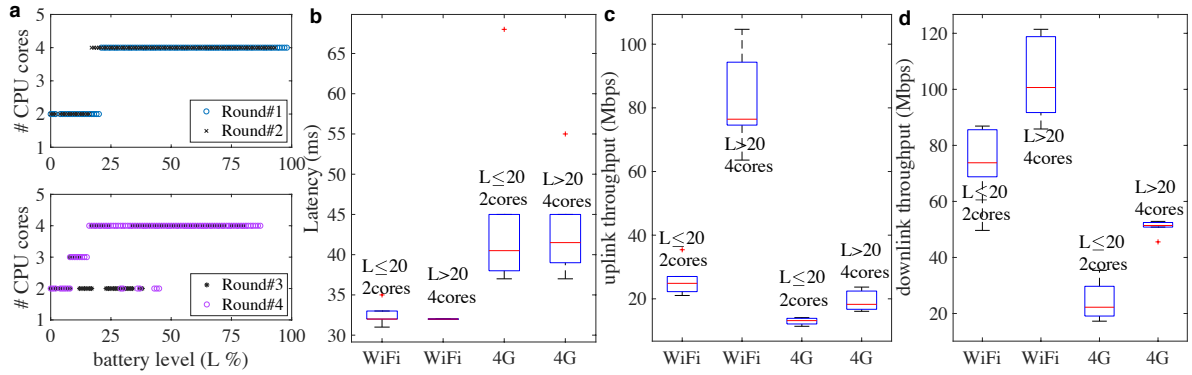

Figure 1: Impact of system optimization on CPU usage, WiFi (W) and LTE (4G) network performance on Nexus 6. *Round#1,2 were taken one year earlier than Round#3,&4. The network performance measurements in (b), (c), (d) were conducted for Round#1&2.*

(3) The VPN-based applications behave like an in-situ middle box. Like most middle-boxes, they fail to recognize the application's intended optimizations expressed via socket options. As a consequence, these tools are unable to capture the intended behavior of applications in their traffic traces (§3).

(4) Various applications, such as multimedia streaming, rely on on-device latency and throughput measurements to optimize the streaming or live broadcasting [13]. These applications may estimate ambiguous latency and throughput in the presence of the VPN-based tools, as we demonstrate with the latency and throughput measurement tools. In the presence of PvG, SpeedCheck [1] estimates on-device latency instead of the network latency. Similarly, VoP doubles the uplink throughput estimates (§4).

We are the first to highlight the presence and effect of battery performance aware system optimization and the impact of measurement tools on mobile network and application performance. We summarize the sources of the above ambiguous or imperfect measurement results and discuss how these tools can be extended for more accurate measurements (§5).

2 IMPACT OF SYSTEM OPTIMIZATION

The power management of modern mobile systems relies heavily on OS modules, i.e., governors. These modules optimize the energy consumption of a device by throttling CPU usage or trading the performance for longer battery life [21, 24]. An individual hardware component also may sleep and use some inactivity timers to avoid unnecessary energy consumption [18].

During the network throughput measurements on Nexus 6, we observed throughput values with a large spread. Our further investigation revealed that the sources of such inconsistency had been the Android system's following optimizations.

Figure 1 (a) shows 4 discharge rounds of Nexus 6 running Android 7.0. The top two rounds were conducted in early 2019, and we notice that Nexus 6 uses only two cores when the battery level is below 20%. Such observations were previously reported for other Nexus devices [4, 5]. Further investigation revealed that the active cores operate at the maximum frequency of 1.73 GHz when the battery level is below 20%. When the battery level is above 20%, all the four cores become active and operate at their maximum frequency of 2.65 GHz. We also observed that the utilization of core differs as the battery ages. Specifically, we observe a lower number of available CPU cores, even with a higher battery level in Round#3 and Round#4. These rounds were conducted one year later than the other rounds. During these rounds, we randomly used Nexus 6 for browsing with Chrome, which caused the transitions to lower CPU cores when the battery levels were below 50%, as shown in the figure.

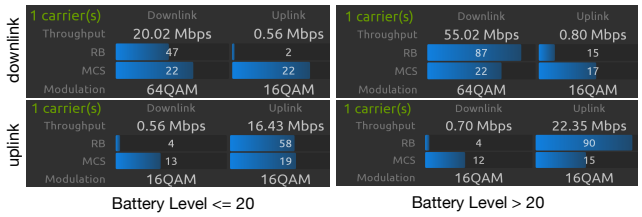

Figure 2: Impact of battery level on LTE modulation scheme. *Four samples from Network Signal Guru during the throughput measurements.*

During the first two rounds, we quantified the impact of these optimizations on network performance. Specifically, we used SpeedCheck [1] (paid) and measured the latency and throughput on Nexus 6 (Android 7.0) with both WiFi and LTE. Each of the above four scenarios was repeated ten times, and the results are presented in Figure 1. Figure 1 (b) shows that while hot unplugging of CPU cores on Android has a negligible impact on the latency, its impacts on throughput is significant. The availability of additional CPU cores, when the battery level is above 20%, improves the I/O performance across both WiFi and LTE.

Furthermore, WiFi uplink throughput improves almost four times when the battery level is above 20% compared to when it is below 20% (Figure 1 (c)). In contrast, the downlink throughput does not degrade significantly with the lower battery level (Figure 1 (d)). The closer inspections of the MAC layer frames revealed that the WiFi radio of the Nexus 6 switches from *802.11ac* to *802.11g* mode when the battery level drops below 20%. This implies that modern Android devices adapt dynamic modulation schemes limiting the WiFi uplink throughput.

Similar to WiFi, we further looked into the physical layer modulation scheme used by the mobile device in the LTE network. We rooted the Nexus 6 and installed Network Signal Guru [7] that samples LTE physical layer parameters every 500 ms. Figure 2 shows that the modulation schemes are 16QAM (Quadrature Amplitude Modulation) and 64QAM for uplink and downlink, respectively.

We observed the usage of the same modulation schemes, and the allocation of Physical Resource Blocks depends on the bitrates for both uplink and downlink in Figure 1. Note that the selection of 16QAM for uplink is since Nexus 6 uses an earlier 3GPP release, as uplink 64QAM was introduced in release 13.

We also observe that connecting the device to a charger after the cores have become inactive does not improve the throughput either on WiFi or LTE. Apart from workload characteristics, the devices may also consider the *battery health* to employ the CPU cores. This is because the performance of the battery in delivering the current and power demands decreases as the battery ages. When the battery voltage reduces to the cut off voltage due to the discharge load, the device shuts down as the battery cannot meet the demand [28].

Nexus 6 enforces the device's reliability by reducing the load on the battery when a small amount of charge is left. The discovered optimizations reduce the discharge load and avoid such sudden shutdown as the battery ages [28]. Apple recently announced such optimizations for the latest iPhones to prevent unexpected shutdown [9]. In order to have reproducible measurements of applications or system performance, one must be aware of such optimization. It is also essential to report the presence or absence of such optimizations as they, directly and indirectly, impact the network performance.

3 IMPACT OF TOOLS ON TRAFFIC

Our above analysis and findings suggest that analyzing the traffic traces that were collected when system optimization is being used will provide incorrect results. Collecting and analyzing the traffic traces is essential for various activities, including investigating traffic patterns [16, 26], studying the flow properties of different mobile applications and their implications for energy consumption [13], analyzing the privacy of individuals [29], and quantifying data waste [19]. On iOS devices, it is possible to collect traffic via a remote virtual interface [10]. Such an interface currently does not exist on Android, and several studies used tcpdump on rooted Android phones [11, 14]. Rooting voids the device warranty, and this can be avoided by using VPN-based tools such as VoP, Lumen, PvG, or MopEye. This section investigates how these applications impact the application's intended flow properties, i.e., packet-gap, packet size, while doing their operations.

3.1 In-situ Traffic Measurement Tools

The forwarder and the packet inspector are two components of the VPN-based in situ traffic measurement tools exemplified by VoP, Lumen, and PvG, as shown in Figure 3.

The forwarder's primary role is to forward (i) the packets received from Android applications to the Internet and (ii) the packets received from the Internet to the Android apps. The forwarder also copies those packets to the inspection queue to decouple the traffic analysis from the packets' path. For TCP flows, the forwarder in Lumen and VoP establishes a socket connection with the remote server using `connect()` API before sending SYN-ACK to the application. In contrast, PvG establishes socket connection after replying with SYN-ACK. Later, we demonstrate how these implementations affect network performance measurements. For UDP flows, the forwarder creates a new UDP socket when it detects a new flow.

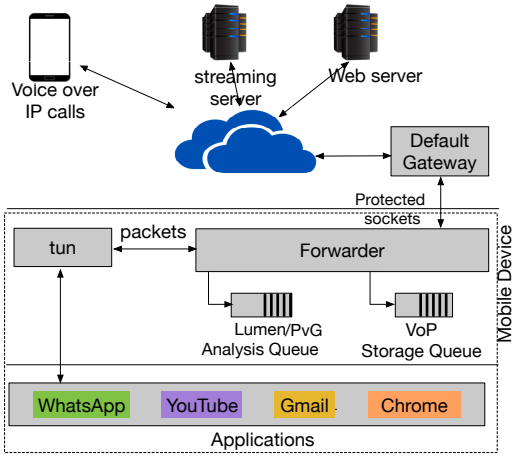

Figure 3: The system components of VoP, Lumen, and PvG. *The newly created sockets are protected so that the packets generated by the Forwarder do not come back to the Forwarder.*

These newly created TCP and UDP sockets are protected so that packets from those do not loop the *tun* interface [3].

A packet inspector is responsible for inspecting the packets in its queue. In Lumen and PvG, the packet inspector performs the privacy analysis on the packets, whereas the VoP's inspector sends packets to the desktop application.

3.2 Addressing Biases

We took the following steps to ensure that the measurement results presented in the upcoming sections are not the artifacts of misconfigured tools and the measurement setup. *(i) Battery level:* based on our observations in 2, we ensured that the devices had more than 80% charge during the experiments. *(ii) Throughput throttling:* because VoP also offers to throttle downlink and uplink traffic, all the measurements in this section were conducted with throttling disabled. *(iii) Software Auto Update.* During the experiments, application and the auto system updates were disabled on mobile devices. *(iv) Advertisements.* We have purchased without ad subscriptions of SpeedCheck and SpeedTest to avoid advertisements.

3.3 Impact on TCP Traffic

We used Periscope to study the impact of VoP and Lumen on real-time TCP flows. Periscope broadcasts over LTE across three different scenarios; (i) baseline, with Lumen, and with VoP. Periscope's live broadcast did not work in the presence of PvG. We capture traffic on Nexus 6 using *tcpdump*. In the case of VoP, we use both VoP and *tcpdump* to capture traffic.

Figure 4 (left) shows the distributions of inter-packet gaps. The distribution for VoP is generated from the pcap traces collected by VoP, and we notice a fixed 100ms delay, whereas such a pattern is absent with Lumen and Baseline distributions. From the distribution of packet size in Figure 4 (right), we notice that more than 70% packets captured by VoP are larger than 1500 bytes. From Traffic traces, we have identified that VoP creates packets of a maximum of 65549 bytes for Periscope. However, the *tcpdump* captures showed that such large packets were fragmented.

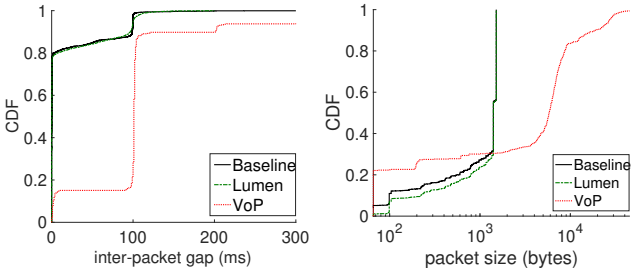

Figure 4: Properties of uplink Periscope TCP flows.

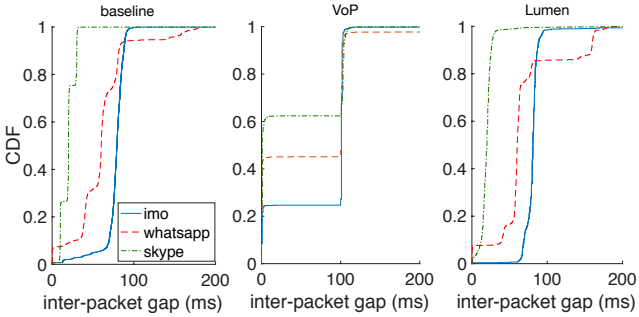

Figure 5: Inter-packet gaps of the VoIP applications. *Baseline refers to the measurements without any localhost VPN.*

Application	Baseline (in/out)	VoP (in/out)	Lumen (in/out)
WhatsApp (v2.18)	21/24 kbps	23/16 kbps	20/22 kbps
IMO (v9.8)	14/15 kbps	14/13 kbps	13/14 kbps
Skype (v8.41)	60/50 kbps	55/44 kbps	48/44 kbps

Table 1: Average bitrates of VoIP applications.

From its source code in Github, we have identified that VoP forwarder implements the maximum segment of 65535 bytes for the TCP flows. With very high bitrate traffic, the buffer gets filled very quickly, contributing to large TCP segments. VoP exports these large segments to the pacp file. The forwarder also writes the same data to the socket. Later in section 4, we show that this aids in higher uplink throughput measurements presented. However, the distributions of the inter-packet gap and packet size from Lumen follow the Baseline measurements.

3.4 Impact on UDP Traffic

In this section, we investigate traffic from three VoIP applications—IMO, WhatsApp, and Skype—that exchange bi-directional encrypted UDP traffic. While these applications fall into the category of VoIP applications, their varying traffic characteristics help us to study the impact of the design of VoP and Lumen. We could not use these applications in the presence of PvG in several trials. We used a rooted Nexus 6 (Android 7.0) and a non-rooted LG G5 (Android 8.0) for these measurements.

For our analysis, we consider three iterations of two-minute conversations over LTE in the following scenarios. As the baseline, we initiated conversations between Nexus 6 and LG G5 using these apps without VoP or Lumen and captured traffic using *tcpdump* on Nexus 6. We then repeated the experiments with VoP running on Nexus 6 and collected traffic from VoP. Finally, we used Lumen. Since Lumen does not store traffic, we captured traffic with *tcpdump*

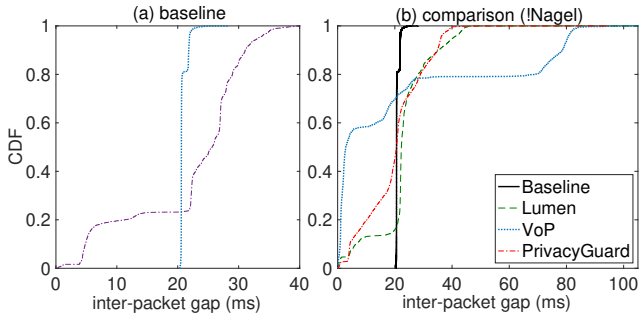

Figure 6: Distributions of the outgoing packet gaps observed at the network interface.

on Nexus 6. For each scenario, we investigate the inter-packet gaps and bitrates.

Baseline Results. The baseline plots in Figure 5 shows that IMO has the highest inter-packet gaps, and Skype packets have the smallest gaps. These apps also have distinct data rates due to the underlying codec [12], with Skype having the highest data rate, as shown in Table 1.

Impact of VoP. Compared to the baseline packet-gaps in Figure Figure 5, VoP significantly alters the inter-packet gaps of outgoing UDP packets. Most of the outgoing packets across all applications have an inter-packet gap of about 100 ms. In contrast, the incoming packets have had similar distributions to the baseline. We have also experienced distorted voice using these applications in the presence of VoP. Table 1 shows that the outgoing data rates of Skype and Whatsapp reduce significantly, which we speculate to be a consequence of the delays introduced by VoP.

Impact of Lumen. Figure 5 shows that with Lumen, the outgoing packets are similarly spaced to the baseline measurements. Besides, the applications experience similar bitrates to the baseline and when using Lumen as shown in Table 1.

3.5 Analysis with Socket Options

This section investigates the performance of the VPN-based tools in processing the flows with TCP_NODELAY (Nagel's algorithm) socket option on Nexus 6. We specifically look into this option because it directly impacts the delay and the performance of web browsing and other real-time applications on mobile devices, such as live broadcasting. We developed a separate traffic generating application that creates two blocking TCP sockets, one where Nagle's algorithm is enabled and the other where it is disabled. The application periodically sends 1300 bytes of data over LTE every 20 ms to a remote server running in our university network. The application also periodically receives data from this remote server every 20 ms in separate TCP sessions. Since the applications experience an additional delay due to Dalvik VM [17], we use the native sockets API.

Performance of VPN-based Tools. Figure 6(a) presents the outgoing inter-packet gap of the application flows, having Nagel's algorithm enabled and disabled. We observe that when Nagel's algorithm is enabled, more than 70% of the packets sent from the application have more than 20 ms inter-packet gaps at the network layer. In the presence of VPN applications, disabling Nagel's algorithm by the application does not reduce the packet-gaps compared to the

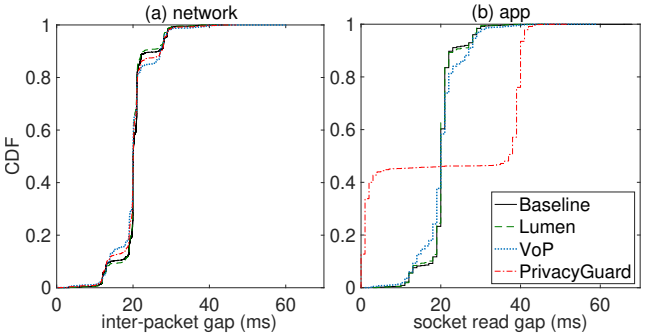

Figure 7: Distributions of incoming packet gaps observed at the network interface and application.

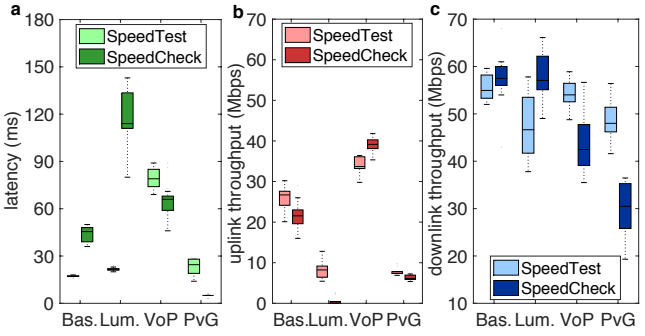

Figure 8: Impact on LTE network latency and throughput. *We used SpeedCheck and SpeedTest on Nexus 6 in the presence of Lumen (Lum.), VoP, PvG, and Baseline (Bas.).*

baseline (Figure 6(b)). Interestingly, VoP's packet gaps reduce, as it receives packets from the local TCP/IP stack without delay. From traffic traces, we have identified that these VPN-based tools do not disable Nagel's algorithm, i.e., do not set the socket option while establishing socket connections.

Figure 7 shows the performance of the VPN applications for incoming traffic. The application receives data at almost similar gaps observed at the network interface. However, in the presence of PvG, approximately 40% packets of the packets received by our application are delayed. The packet-gaps patterns suggest that it uses a fixed interval to read the VPN interface similar to VoP.

3.6 Summary

The results presented in this section reveal the following. **(1)** VoP introduces a fixed delay on 100 ms in the uplink direction and generates huge packets, whereas PvG introduces delay for the applications in the downlink direction. Such delays degrade application performance and can misguide the researchers and developers using these tools. In contrast, Lumen is free from such limitations in both directions. **(2)** These VPN-based tools do not set the TCP/IP socket options as intended by the user applications. Consequently, they can degrade application performance and affect measurement results. Findings in this section explain the high latency we observe in the results presented in §4 when using SpeedTest.

4 IMPACTS ON APP PERFORMANCE

VoP captures traffic on mobile devices for offline analysis, whereas Lumen and PvG track user application performance online. The additional delay introduced by VoP causes poor quality of experience for the voice of over IP applications. Some user applications may rely on active network performance measurements, such as latency and throughput. The prime examples are multimedia applications, such as Netflix and YouTube, which actively measure these metrics for applying various rate control algorithms [15]. To emulate applications that conduct these measurements, we use SpeedCheck [1] and SpeedTest [2]. These two applications also apply different methods to estimate latency and throughput. In this section, we explore the network performance of these two applications in the presence of VoP, Lumen, and PvG; the measurements were repeated ten times.

(1) Latency. Figure 8 (a) compares the network latency reported by the two applications in the presence of the VPN-based tools. In the baseline *tcpdump* traces, we observe that SpeedTest uses 10-12 requests/responses of few bytes (less than 100 Bytes) over a TCP connection to estimate the latency. In our tests, SpeedTest estimated the baseline latency of 16-18 ms. It experiences 3-5 ms additional latency in the presence of Lumen and PvG, whereas VoP increases the latency by three-fold. This is due to the optimization strategy adopted by VoP, which we discussed earlier.

In contrast, SpeedCheck reports the median baseline network latency of about 45 ms. In the corresponding *tcpdump* traces, we observed ten consecutive TCP flows without any data exchange for each latency measurements. These flows suggest that SpeedCheck uses the TCP `connect()` API to measure the latency. Both VoP and Lumen increase the median latency significantly. We speculate that these two take more time to set up new TCP flows. However, SpeedCheck underestimates the latency in the presence of PvG, which is the consequence of the sending SYN-ACK by the PvG forwarder before the connection is established with the remote server, as discussed in §3.1.

(2) Uplink Throughput. In Figure 8 (b), we observe that SpeedTest estimates higher uplink baseline throughput, as its server is in our LTE operator's network. It uses multiple parallel TCP connections to estimate the throughput. Both Lumen and PvG reduce the throughput of SpeedTest and SpeedCheck by half compared to the baseline measurements. However, Lumen severely affects the uplink throughput measurements of the SpeedCheck. On closer inspection, we observed that SpeedCheck uses a single TCP connection and sends a large amount of data. From an exception in Lumen's debug log, we characterized that Lumen's forwarder cannot handle such large volumes of traffic. Interestingly, VoP doubles the uplink throughput of both applications.

(3) Downlink Throughput. In Figure 8 (c), we observe that SpeedTest measures similar downlink throughput in the presence of the VPN tools to the baseline. Lumen aids the highest throughput measurements with SpeedCheck. However, VoP and PvG degrade the throughput of SpeedCheck significantly.

5 SOURCES OF IMPERFECTION

Mobile system optimizations affect downlink and uplink throughput, whereas the VPN-based tools we studied mostly affect the uplink throughput and latency, i.e., the outgoing traffic.

Energy-Aware Optimization. Energy-aware system optimization can affect the network performance by limiting the network I/O

and applying adaptive modulation schemes. Therefore, it is wise to perform such measurements when the battery is fully charged. Such optimization comes as the combined effort of the system on chip and the operating system. For example, Nexus 6 and Nexus 6P are powered by Snapdragon chips. Apple uses similar optimization with iOS 12.1 and the latest iPhones of 8 and higher models [9]. Regardless, VoP, Lumen, and PvG leverage different sleeping techniques for optimizing their energy usage.

The additional latency introduced by VoP on outgoing packets is the artifact of using a fixed sleep interval of 100 ms in its main VPN thread. This delay further contributes to large outgoing packets for higher bitrate uplink traffic and increased energy consumption for fragmentation. PvG also introduces a fixed delay for the incoming traffic. These delays affect the quality of the measurements and the quality of experience when using other user applications.

Forwarder. In situ VPN-based measurement tools are in situ middleboxes that tap the packets using the VPN interface. Therefore, these applications implement a forwarder which primarily consists of three threads: the main VPN thread, and two socket reader/writer threads. The reader/writer threads continuously iterate through a list of live sockets, which contributes to the delays. The forwarder also implements a flow state machine for each flow and constructs/de-constructs the packets. The implementation of the forwarder thus affects the latency and throughput measurements. We have also shown that the characteristics of the newly created flows and their packet headers might not be the same as those generated by the applications. The reason is that the corresponding socket options must be set before the connection establishment, and the forwarder currently has no way of detecting the socket options used by the applications.

6 CONCLUSIONS

In this paper, we investigated the challenges in measuring network and application performance in the presence of system optimization and state-of-the-art measurement tools on Android devices. There is still room for improvement in all the tools. For instance, VoP and PvG can follow Lumen's adaptive sleeping algorithm for reducing the gaps in the outgoing and incoming packets, respectively. All of them can adopt some default socket options to mitigate the performance issues with the outgoing TCP traffic. It can be argued that VoP is mostly for the developers, and therefore, incurring higher delays should not be a problem. Similarly, frequent massive content uploading is rare, and 3-4 ms additional latency is acceptable for many applications. Nevertheless, these imperfections can significantly affect applications' performance and the outcome of traffic measurement studies that use these tools. Although Lumen appears to be the best alternative candidate, it can be extended for traffic capture and analysis. Along with the measurement tools, it is necessary to understand the presence of various system optimization techniques, which may affect network performance. Our findings would assist the researchers and developers in performing reproducible measurements on mobile devices.

REFERENCES

[1] SPEEDCHECK - Speed Test. https://play.google.com/store/apps/details?id=org.speedspot.speedanalytics. [Online; accessed 7-August-2019].

[2] Speedtest by Ookla. https://play.google.com/store/apps/details?id=org.zwanoo.android.speedtest.gworld. [Online; accessed 11-August-2019].

[3] VPN - Android Developers. https://developer.android.com/guide/topics/connectivity/vpn. [Online; accessed 23-January-2019].

[4] 4 cores always stopped, what's going on? https://forum.xda-developers.com/nexus-6p/help/4-cores-allways-stopped-whats-t3390543, 2016. [Online; accessed 15-March-2020].

[5] Low Battery Disabling CPU Cores. https://www.reddit.com/r/nexus6/comments/4k4ut1/low_battery_disabling_cpu_cores/, 2016. [Online; accessed 7-May-2020].

[6] AT&T Video Optimizer. https://developer.att.com/video-optimizer, 2019.

[7] Network Signal Guru. https://play.google.com/store/apps/details?id=com.qtrun.QuickTest, 2019.

[8] Mario Almeida, Alessandro Finamore, Diego Perino, Narseo Vallina-Rodriguez, and Matteo Varvello. Dissecting DNS Stakeholders in Mobile Networks. In *Proceedings of CoNEXT '17*, pages 28–34. ACM, 2017.

[9] Apple. iPhone Battery and Performance. https://support.apple.com/en-us/HT208387, 2020. [Online; accessed 21-July-2029].

[10] Apple. Recording a Packet Trace. https://developer.apple.com/documentation/network/recording_a_packet_trace, 2020. [Online; accessed 7-April-2020].

[11] Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, and Deborah Estrin. A First Look at Traffic on Smartphones. In *Proceedings of IMC '10*, pages 281–287. ACM, 2010.

[12] Mohammad A. Hoque, Petteri Nurmi, Matti Siekkinen, Pan Hui, and Sasu Tarkoma. The Bits of Silence: Redundant Traffic in VoIP. In *Proceedings of the 11th ACM Multimedia Systems Conference*, MMSys '20, page 1–14, New York, NY, USA, 2020. Association for Computing Machinery.

[13] Mohammad Ashraful Hoque, Matti Siekkinen, Jukka K. Nurminen, Mika Aalto, and Sasu Tarkoma. Mobile multimedia streaming techniques: QoE and energy saving perspective. *Pervasive and Mobile Computing*, 16:96 – 114, 2015.

[14] Junxian Huang, Feng Qian, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. Screen-off Traffic Characterization and Optimization in 3G/4G Networks. In *Proceedings of IMC '12*, pages 357–364. ACM, 2012.

[15] Bart Jansen, Timothy Goodwin, Varun Gupta, Fernando A. Kuipers, and Gil Zussman. Performance evaluation of webrtc-based video conferencing. *SIGMETRICS Performance Evaluation Review*, 45:56–68, 2017.

[16] Feng Li, Jae Won Chung, and Mark Claypool. Silhouette: Identifying youtube video flows from encrypted traffic. In *Proceedings of NOSSDAV*, pages 19–24. ACM, 2018.

[17] W. Li, D. Wu, R. K. C. Chang, and R. K. P. Mok. Toward accurate network delay measurement on android phones. *IEEE Transactions on Mobile Computing*, 17(3):717–732, 2018.

[18] Weichao Li, Daoyuan Wu, Rocky K.C. Chang, and Ricky K.P. Mok. Demystifying and puncturing the inflated delay in smartphone-based wifi network measurement. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '16, page 497–504, New York, NY, USA, 2016. Association for Computing Machinery.

[19] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen. Effectively minimizing redundant internet streaming traffic to ios devices. In *2013 Proceedings IEEE INFOCOM*, pages 250–254, 2013.

[20] F. Michclinakis, H. Doroud, A. Razaghpanah, A. Lutu, N. Vallina-Rodriguez, P. Gill, and J. Widmer. The Cloud that Runs the Mobile Internet: A Measurement Study of Mobile Cloud Services. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 1619–1627, April 2018.

[21] Tanuj Mittal, Lokesh Singhal, and Divyashikha Sethia. Optimized cpu frequency scaling on android devices based on foreground running application. *Lecture Notes in Electrical Engineering*, 131:827–834, 01 2013.

[22] Vern Paxson. Strategies for sound internet measurement. In *Proceedings of IMC*, IMC '04, page 263–271. ACM, 2004.

[23] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. Profiling Resource Usage for Mobile Applications: A Cross-layer Approach. In *Proceedings of MobiSys'11*, pages 321–334. ACM, 2011.

[24] K. Rao, J. Wang, S. Yalamanchili, Y. Wardi, and H. Ye. Application-Specific Performance-Aware Energy Optimization on Android Mobile Devices. In *Proceedings of HPCA*, pages 169–180, 2017.

[25] Abbas Razaghpanah, Arian Akhavan Niaki, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Johanna Amann, and Phillipa Gill. Studying TLS Usage in Android Apps. In *Proceedings of CoNEXT '17*, pages 350–362. ACM, 2017.

[26] Andrew Reed and Michael Kranch. Identifying HTTPS-Protected Netflix Videos in Real-Time. In *Proceedings of CODASPY*, pages 361–368. ACM, 2017.

[27] Yihang Song and Urs Hengartner. PrivacyGuard: A VPN-based Platform to Detect Information Leakage on Android Devices. In *Proceedings of SPSM*, pages 15–26. ACM, 2015.

[28] Y. Sun, L. Kong, H. Abbas Khan, and M. G. Pecht. Li-ion battery reliability – a case study of the apple iphone®. *IEEE Access*, 7:71131–71141, 2019.

[29] Tavish Vaidya, Tim Walsh, and Micah Sherr. Whisper: A Unilateral Defense Against VoIP Traffic Re-identification Attacks. In *Proceedings of ACSAC*, pages 286–296. ACM, 2019.

[30] Daoyuan Wu, Rocky K. C. Chang, Weichao Li, Eric K. T. Cheng, and Debin Gao. MopEye: Opportunistic Monitoring of Per-app Mobile Network Performance. In *Proceedings of USENIX ATC '17*, pages 445–457. USENIX Association, 2017.