Poster: Can Mobile Hardware Keep Up with Today's Gigabit Wireless Technologies?

Shivang Aggarwal[1], Swetank Kumar Saha[1], Pranab Dash[2], Jiayi Meng[2], Arvind Thirumurugan[1], Dimitrios Koutsonikolas[1], Y. Charlie Hu[2]

[1]University at Buffalo, The State University of New York, [2]Purdue University
{shivanga,swetankk,athirumu,dimitrio}@buffalo.edu,{dashp,meng72,ychu}@purdue.edu

ABSTRACT

With the advent of bandwidth-hungry applications and the new advancements in wireless LAN standards (802.11ad, 802.11ay) and cellular technologies (5G), modern smartphones need to be able support multi-Gbps data rates. In this work, we explore if today's smartphones are capable of handling such high-speed network traffic. Using two high-end smartphones, we show that, contrary to previous beliefs, they can indeed support Gbps data rates without significant strain on their hardware resources. Using projections, we further show that up to 12.8 Gbps could be supported with just 50% CPU utilization. Finally, we explore the factors that make this possible and the contribution of each of them.

ACM Reference Format:
Shivang Aggarwal, Swetank Kumar Saha, Pranab Dash, Jiayi Meng, Arvind Thirumurugan, Dimitrios Koutsonikolas, Y. Charlie Hu. 2019. Poster: Can Mobile Hardware Keep Up with Today's Gigabit Wireless Technologies? In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19), October 21–25, 2019, Los Cabos, Mexico.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3300061.3343398

1 INTRODUCTION

An entire class of smartphone applications is emerging that demands multi-Gbps from the underlying wireless network. Examples of such applications include mobile Virtual reality (VR)/Augmented reality (AR) and live 4K video streaming. This list is bound to grow in the future as app developers race towards providing immersive experiences to smartphone users and as more video content starts becoming available at ultra-high resolutions.

Research in the wireless networking space did predict this need and has come up with solutions and technologies that are indeed capable of providing multi-Gbps data rates. In the context of indoor WLANs, IEEE 802.11ad and the upcoming 802.11ay are the prime technologies that are capable of Gigabit throughput. On the cellular side, with the advent of 5G, there are several candidates (e.g., mmWave around 28/39 GHz) that hold the promise of breaking the one Gbps end-to-end throughput barrier. In fact, consumer devices supporting multi-Gbps throughput are already available in the market including but not limited to laptops (Acer TMP648-MG-789T) and routers (Netgear Nighthawk R9000) with 802.11ad chipsets and smartphones equipped with 802.11ad and 5G interfaces (Asus ROG, Samsung Galaxy S10 5G, Moto Z3 5G Mod).

The smartphone is a critical component of this emerging ecosystem but, surprisingly, it has not received enough attention to assess whether the device hardware can meet the demands that come with supporting multi-Gbps network speeds. In fact, several previous works (e.g., [1, 2]) have projected that CPU processing in smartphones may become a bottleneck even if networking technologies were to support such high data rates. Based on these projections, they have even proposed solutions that favor performing certain computation on-board the smartphone as opposed to offloading to remote servers.

In this work, we examine this assumption and seek to answer the following question: can mobile hardware keep up with today's and future's multi-Gbps wireless technologies? Specifically, we examine two top-tier Android devices, Google Pixel 2 (Oct. 2017) and Asus ROG phone (Oct. 2018), and explore how they would perform under Gigabit-scale network traffic. We not only find that these devices are capable of supporting 1.6 Gbps network traffic but also show using projections that they are future-proof with the Pixel 2 phone capable of supporting up to 2.3 Gbps and the ROG phone up to 12.8 Gbps. Next, we investigate further to understand why our observations are in stark contrast to past projections. Specifically, we focus on two important factors that we believe significantly affect CPU performance: (1) CPU Affinity in the context of the big.LITTLE architecture and (2) impact of TCP/Generic Segmentation Offloading (TSO/GSO) and Generic Receive Offloading (GRO). Lastly, we look at two other factors critical for smartphones, namely Energy Aware Scheduling (EAS) and battery life

MobiCom '19, October 21–25, 2019, Los Cabos, Mexico

ACM ISBN 978-1-4503-6169-9/19/10.
https://doi.org/10.1145/3300061.3343398

and show that these are not prohibitive to supporting Gbps network speeds.

2 EXPERIMENTAL SETUP

Table 1: Phone Specifications.

	Google Pixel 2	ASUS ROG Phone
CPU	Snapdragon 835 (Octa-core, 4x2.46GHz & 4x1.90GHz)	Snapdragon 845 (Octa-core, 4x2.96GHz & 4x1.76GHz)
RAM	4GB	8GB
Battery	2700mAh	4000mAh
WiFi	802.11a/b/g/n/ac	802.11a/b/g/n/ac/ad
OS	Android 9.0 (Pie)	Android 8.1 (Oreo)

Devices. Table 1 lists the two smartphones used in our study along with their hardware specifications. The ROG phone, with its newer processor, supports a higher maximum CPU frequency than the Pixel 2. Both CPUs are built on top of ARM's big.LITTLE architecture with 2 clusters with 4 cores in each. The ROG phone hosts an 802.11ad NIC which is capable of providing data rates up to ~1.65 Gbps, in addition to a legacy 802.11ac NIC present in both devices capable of providing data rates up to ~500 Mbps.

Methodology. We use iperf3 to generate TCP traffic with sources rates varying from 100 Mbps to the maximum supported by the device and log the Total and Soft IRQ (used for processing incoming network packets) CPU utilization on the smartphone from `/proc/stats`. All results reported are averages of several 60 s downlink data transfer sessions. For each device, we use the network interface/technology that provides the maximum throughput (802.11ac for Pixel 2; 802.11ad for ROG phone).

3 CPU PERFORMANCE

3.1 Baseline

To establish a baseline, we measure the Total and Soft IRQ (SIRQ) CPU utilization under backlogged traffic. For Pixel 2, with the 802.11ac interface, we observed a maximum downlink throughput of 542 Mbps with the total CPU utilization around 11.6%. Out of the total CPU utilization, SIRQs account for 8.5%, a significant portion of the total value. The ROG phone using its 802.11ad interface supports a much higher throughput of ~1.7 Gbps while keeping its CPU utilization limited to 17.1% (downlink). SIRQs account for 8.1% out of the total value. Note that maximum frequency for the ROG phone's CPU is slightly higher (2.95 GHz vs. 2.46 GHz) than that of Pixel 2 and given that the CPU governors in the two can behave differently we do not make a direct comparison of the utilization values here.

3.2 Non-backlogged Traffic

To remove any artifacts arising due to the CPU governor we manually fix the CPU frequency to the maximum supported value by each of the two devices. To better understand the CPU utilization trend for different network loads, we vary the source data rate at the sender between 100 Mbps and the maximum throughput supported by each device.

Pixel 2 (Fig. 1(a)) shows a linear trend between CPU utilization (for both Total and SIRQ) and throughput. Since the maximum throughput is limited by the 802.11ac interface to 500 Mbps, we extrapolate the CPU utilization to 1.6 Gbps which can potentially be supported with less than 50% CPU utilization. However, note that to support the 4 Gbps throughput required for wireless AR/VR according to [2], CPU utilization would hit 84%, making it impractical.

For the ROG phone (Fig. 1(b)), we observed a similar linear growth of CPU utilization with increasing network traffic. However, the slope here is rather flat and the CPU utilization is capped at less than 15% even at the maximum supported 802.11ad throughput of 1.7 Gbps. Extrapolating this further, supporting 4 Gbps on the ROG phone would result in only 19% CPU utilization. In fact, the ROG phone can potentially support throughputs of up to even 12.8 Gbps with 50% CPU utilization, still leaving enough unutilized CPU capacity to be used by other applications. This further provides us with proof that smartphone devices already support hardware that can process network traffic at multi-Gbps speeds and network performance will remain limited by the wireless technology itself, at least for the near future.

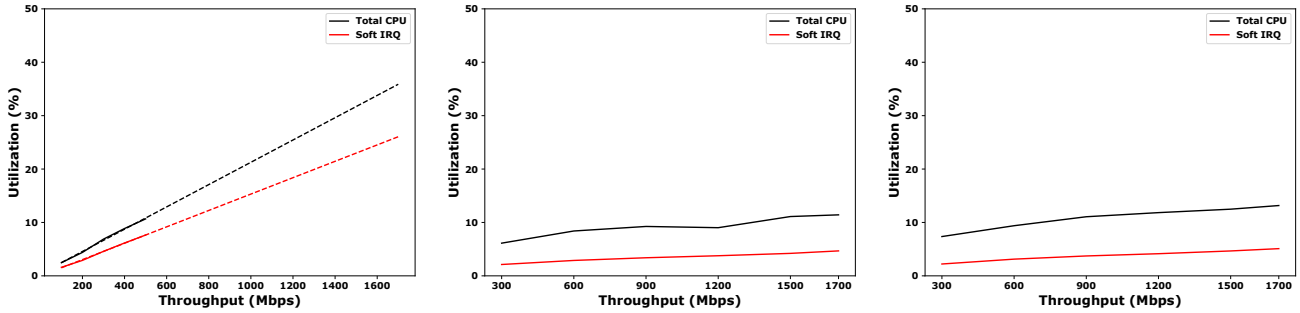
Lastly, in order to do a fair comparison between the two devices, we set the ROG's CPU frequency to be the same as Pixel 2's maximum supported value (2.46 GHz). Interestingly, we observe (Fig. 1(c)) the CPU utilization trend to be almost identical to the case when ROG's CPU was clocked at its maximum frequency (2.95 GHz). This is especially surprising given the SoC used by the two phones is just one generation apart and have CPUs based on the same architecture. We plan to explore this further in the future.

We also repeated the measurements with several different frequencies supported by the big and LITTLE cores of the ROG phone's CPU. We observed that ROG phone can achieve the maximum throughput (1.7 Gbps) at all the big core frequencies and for all the LITTLE core frequencies greater than 826 MHz.

3.3 Understanding CPU Utilization

We investigated further to understand how the smartphones used in our study (especially the ROG phone) are able to process the Gbps network workloads without spending significant CPU resources. Here, we discuss two factors that we believe are significant in this aspect.

3.3.1 Linux Segmentation Offloading. Linux Segmentation Offloading allows for the CPU-intensive task of packet segmentation to be performed at the network interface card (NIC) instead of the CPU. On the sender side, TSO/GSO delays the packet segmentation as much as possible so that one big chunk of data can traverse down the network stack instead of multiple small ones reducing the overhead of processing each packet. On the receiver side, GRO performs the complementary operation. To evaluate the impact of these optimizations, we disable them and run our baseline experiments again with the ROG phone. With GRO off, the



(a) Pixel 2 over 802.11ac at the maximum frequency (2.46GHz).

(b) ROG Phone over 802.11ad at the maximum frequency (2.95GHz).

(c) ROG Phone over 802.11ad at Pixel 2's maximum frequency (2.46GHz).

Figure 1: CPU and SIRQ Utilization at different throughputs.

average throughput is degraded down to 366 Mbps, a sharp drop from the 1.6 Gbps (with GRO on).

Remark: GSO/GRO optimizations are extremely critical for achieving multi-Gbps rates.

3.3.2 CPU Affinity. Upon investigating the big drop in throughput observed in §3.3.1, we realized that CPU1 (which does most of the SIRQ processing) is fully utilized and hence becomes the bottleneck. Note that CPU1 is a LITTLE core (clocked at a maximum of 1.76 GHz). If we move the SIRQ processing from CPU1 to one of the big cores (say, CPU5), we observed the throughput improved significantly to 1.23 Gbps. Although this is lower than the GRO-enabled values (1.7 Gbps), it still amounts to a 4x gain just by moving from the LITTLE to the big core. Note that it is not trivial to split SIRQ over a large number of cores as it can potentially create out-of-ordering, degrading TCP performance. We plan to look at designing solutions for split SIRQ processing.

Remark: Selecting the appropriate CPU core is important to prevent SIRQ processing from becoming the bottleneck.

4 OTHER FACTORS

4.1 Energy Aware Scheduling (EAS)

Another factor that comes into play at these high data rates is the CPU scheduling policy. Recently, Energy Aware Scheduling (EAS) has become popular and more device manufacturers are adopting it. EAS gives the Linux kernel scheduler hints by looking at the performance and the power consumption of the CPUs in order to optimize energy consumption. It is enabled by default in the ROG phone but it is disabled once the phone is connected to a charger. To quantify the impact of the EAS scheduler, we measure the throughput in both charging and discharging states and repeat the measurements with both GRO on and off. With GRO on, the phone can achieve the maximum throughput irrespective of the charging states. For the GRO off case, Table 2 shows the performance with big and LITTLE cores in the two states. With the LITTLE core, a maximum of ~620 Mbps can be achieved while charging and the throughput drops to almost half (366 Mbps) when the phone is discharging. Even with big core, the phone supports a maximum throughput of 1.56 Gbps, close to the maximum value of 1.7 Gbps. However, when discharging, the throughput is degraded to 1.1 Gbps, a drop of ~400 Mbps. We also plan to consider CPU scheduling as one of the aspects to study in our future work.

Table 2: Throughput (Mbps) with GRO off.

	big Core	LITTLE Core
Discharging	1183.56	366.17
Charging	1557.44	620.95

Remark: The EAS scheduler can have a detrimental effect on the network performance if CPU-network optimizations like GRO are not used.

4.2 Battery Life

An important aspect for practically supporting ultra-fast data transfers is whether the battery consumption is within acceptable limits. To test the rate of ROG Phone's battery drain due to Gigabit data transfers, we charge the battery to 100% and run a 20 minute long data transfer in both the uplink and downlink directions separately. We observed that the battery percentage dropped by just 4-5% in both the cases while providing 1.7 Gbps (downlink)/1.4 Gbps (uplink) throughput. In the future, we plan to do a much more extensive study using power monitors to better understand aspects like how much power is consumed by the radio vs. the CPU for network-related processing.

Remark: Battery life is not significantly affected by long running Gbps rate data transfers.

5 ACKNOWLEDGEMENTS

This work was supported in part by NSF grants CNS-1553447, CNS-1422304, and CNS-1719369.

REFERENCES

[1] Andres Garcia-Saavedra, Pablo Serrano, Albert Banchs, and Giuseppe Bianchi. 2012. Energy Consumption Anatomy of 802.11 Devices and Its Implication on Modeling and Design. In *Proc. of the ACM CoNEXT (CoNEXT '12)*.

[2] Zeqi Lai, Y. Charlie Hu, Yong Cui, Linhui Sun, and Ningwei Dai. 2017. Furion: Engineering High-Quality Immersive Virtual Reality on Today's Mobile Devices. In *Proc. of the ACM MobiCom (MobiCom '17)*.