Multipath TCP in Smartphones Equipped with Millimeter Wave Radios

Imran Khan[1], Moinak Ghoshal[1], Shivang Aggarwal[1], Dimitrios Koutsonikolas[1], Joerg Widmer[2]
[1]Northeastern University, USA, [2]IMDEA Networks, Spain

ABSTRACT

The well-known susceptibility of millimeter wave links to human blockage and client mobility has recently motivated researchers to propose approaches that leverage both 802.11ad radios (operating in the 60 GHz band) and legacy 802.11ac radios (operating in the 5 GHz band) in dual-band commercial off-the-shelf devices to simultaneously provide Gbps throughput and reliability. One such approach is via Multipath TCP (MPTCP), a transport layer protocol that is transparent to applications and requires no changes to the underlying wireless drivers. However, MPTCP (as well as other bundling approaches) have only been evaluated to date in 60 GHz WLANs with laptop clients.

In this work, we port for first time the MPTCP source code to a dual-band smartphone equipped with an 802.11ad and an 802.11ac radio. We discuss the challenges we face and the system-level optimizations required to enable the phone to support Gbps data rates and yield optimal MPTCP throughput (i.e., the sum of the individual throughputs of the two radios) under ideal conditions. We also evaluate for first time the power consumption of MPTCP in a dual-band 802.11ad/ac smartphone and provide recommendations towards the design of an energy-aware MPTCP scheduler. We make our source code publicly available to enable other researchers to experiment with MPTCP in smartphones equipped with millimeter wave radios.

CCS CONCEPTS

• **Networks** → **Transport protocols**; **Wireless local area networks**; • **Human-centered computing** → **Smartphones**.

ACM Reference Format:
Imran Khan, Moinak Ghoshal, Shivang Aggarwal, Dimitrios Koutsonikolas, Joerg Widmer. 2022. Multipath TCP in Smartphones Equipped with Millimeter Wave Radios. In *The 15th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterization (WiNTECH'21), January 31-February 4 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3477086.3480839

1 INTRODUCTION

An emerging class of smartphone applications, such as mobile Augmented/Virtual reality (AR/VR), Miracast, and UHD video streaming, demand Gbps data rates from the underlying wireless network. For example, 8K resolution VR demands 1.2 Gbps [29] in order to satisfy the 20ms photon-to-motion latency, while live 4K video streaming at 30 FPS demands 1.8 Gbps [13] for good user QoE. While such stringent wireless network performance demands could not be supported in the past, the advent of mmWave technologies in recent years has brought Gbps wireless data rates within reach and holds the promise to enable these demanding applications. For example, the IEEE 802.11ad WLAN standard [25] governs the use of the unlicensed spectrum around 60 GHz and supports 2 GHz wide channels to provide PHY data rates of up to 6.7 Gbps. Multiple 802.11ad-compliant commercial of-the-shelf (COTS) devices have been released over the past few years including APs [6, 8], laptops [1], and more recently smartphones [2, 7]. Similarly, a number of smartphones equipped with 5G NR interfaces operating at 28 GHz and 39 GHz have been launched on the market over the past year and recent studies [30–32] report speeds up to 3 Gbps over commercial cellular networks.

Nonetheless, communication at mmWave frequencies faces fundamental challenges due to the high propagation and penetration loss and the use of directional transmissions makes links susceptible to disruption by human blockage and client mobility. Even though a number of PHY and MAC layer enhancements have been proposed (e.g., [22, 23, 47, 48]) to improve beam steering and reduce re-connection times, the data rate of mmWave links has been shown to fluctuate widely and unpredictably [9, 10, 13, 40]. Hence, it is unrealistic to expect ubiquitous mmWave coverage, similar to that offered by sub-6 GHz technologies such as WiFi or LTE.

Consequently, researchers have recently proposed bundling mmWave and legacy WiFi interfaces [13, 39] to simultaneously offer both multi-Gbps data rates (by jointly using both interfaces when they are available) and reliability (by falling back to the legacy WiFi interface when mmWave connectivity becomes unavailable). While such bundling can be implemented at different layers of the protocol stack, in our previous work [39] we explored a generic transport layer solution via Multipath TCP (MPTCP), a standard transport layer protocol, which works with unmodified applications that run over TCP. Compared to solutions that try to achieve a similar functionality at the MAC layer, e.g., via 802.11ad's Fast Session Transfer (FST) [46] or the Linux bonding driver [13], MPTCP is transparent to applications and already provides mechanisms to re-order packets from different interfaces at the receiver in order to provide an in-order data stream, similar to TCP. As such, it requires no modifications to applications or to underlying wireless drivers.

In [39], we showed that MPTCP with the default *minRTT* scheduler can achieve near optimal performance in dual-band 5/60 GHz WLANs under static scenarios, but it fails to provide the sum-rate of the two interfaces under dynamic scenarios involving interference, mobility, or blockage. We then designed *MuSher*, an MPTCP scheduler that addresses the root-cause of the performance degradation via throughput-ratio-based scheduling, and allows MPTCP to perform near optimally under a wide variety of use cases. Nonetheless,

WiNTECH'21, January 31-February 4 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-8703-3/22/01…$15.00
https://doi.org/10.1145/3477086.3480839

the performance over MPTCP over mmWave links has only been evaluated using laptops as clients.

Given the prevalence of smartphones in today's WLANs and the fact that most of the bandwidth demanding applications (AR, VR, Miracast) target smartphone users, it is important to evaluate the performance of MPTCP in smartphones equipped with mmWave radios. In fact, the concerns about mmWave performance in dynamic scenarios are more pronounced in the case of smartphones [9, 10] due to the small form factor, which increases the probability of self-blockage and limits the number of phased array elements, resulting in lower antenna gains and reduced transmission range. In addition, there have been concerns related to the capability of resource-constrained mobile devices to handle Gbps traffic rates [27]. Although in our previous work [9, 10] we showed that modern smartphones equipped with 802.11ad radios can support up to 1.6 Gbps, the use of MPTCP will result in even higher rates, as a result of jointly using two radios. In addition, power consumption is a serious concern in the case of smartphones, as the joint use of two radios can quickly deplete the device's battery. Consequently, bundling solutions designed for laptops [13, 39] that only focus on throughput maximization may not be suitable for smartphones.

In this work, we take a first step towards filling this gap by porting for first time the MPTCP source code to a smartphone equipped with an 802.11ad radio. We discuss the challenges we faced and the steps we took to ensure that the phone yields optimal performance (i.e., the sum of the individual throughputs of the two radios) under ideal conditions. We make the MPTCP source code publicly available[1] to enable other researchers to experiment with MPTCP in mmWave WLANs. We also evaluate for first time the power consumption of MPTCP in a dual-band 802.11ad/ac smartphone and provide recommendations towards the design of an energy-aware MPTCP scheduler.

2 DEVICES AND EXPERIMENTAL METHODOLOGY

We ported MPTCP to an ASUS ROG Phone II [7]. This phone is the successor of the ASUS ROG Phone, which was the first commercially available smartphone with an 802.11ad chipset. The ROG Phone II is powered by a Snapdragon 855+ octa-core processor with a 12 GB RAM and a 6000 mAh battery. It supports all the 802.11ad single-carrier (SC) MCSs (1-12), which correspond to PHY data rates from 385 Mbps up to 4.6 Gbps. However, in practice, the maximum TCP throughput is limited to about 1.6 Gbps [9, 10]. The phone also has an 802.11ac chipset from Qualcomm, which supports all the 802.11ac MCSs (0-9), channel widths of 20/40/80 MHz, and 2 spatial streams (SS), yielding PHY data rates from 6.5 Mbps up to 866 Mbps.

For our evaluation, we use a dual-band 802.11ad/ac Netgear Nighthawk X10 Smart WiFi router [6] as access point. It is connected through a 10G LAN SFP+ interface to a Dell Precision Tower 3620, which acts as the MPTCP server. We use iperf3 to generate TCP traffic and log throughput every 100 ms. We measure power by logging the voltage and current drawn by the phone from the `/sys/class/power supply/battery` directory every 1 s. All the power measurements are taken with the screen on and minimal background activity. This ensures that the base power (defined as the power consumed by the phone when the screen is on but all radios are off) is low and stable over time. The power results are relative to the base power. All experiments are performed at night to avoid interference from other devices. They are done with a fully charged battery and the battery does not drop by more than 5% during the experiments.

[1] https://github.com/NUWiNS/ROGII_MPTCP/tree/master

3 PORTING MPTCP TO ROG PHONE II

In this section, we describe the implementation of MPTCP on an ASUS ROG Phone II and the challenges we had to overcome to achieve optimal performance.

3.1 Implementation

The ASUS ROG Phone II runs Android OS 10 based on Linux kernel version 4.14.191. We ported MPTCP v0.94 [5] for Linux kernel version 4.14.184 to the Android kernel on the ROG Phone II. We used a publicly available version of the kernel source code [3] for the ASUS ROG Phone II as our base. To successfully enable MPTCP on this kernel, we needed to port two different parts of the code from the MPTCP and generic Linux kernels to the Android kernel.

First, we added the main MPTCP tree to our base kernel (the entire `net/mptcp/` directory). However, apart from that, MPTCP also requires changes in many of the existing TCP modules that form the TCP networking stack for the Linux kernel (in `net/ipv4/`, `net/core/`, etc.). Hence, we had to carefully go through all the structures and functions modified/added for MPTCP and make changes accordingly in our Android Linux kernel.

Second, due to the Linux kernel version mismatch between the Android kernel and the MPTCP kernel, we had to make changes in other non-TCP/MPTCP parts of the kernel as well. The challenge here comes from the fact that there are some differences in the Android version of the Linux kernel and the generic Linux kernel [4]. Hence, we had to again carefully check each file in the kernel with differences between the two kernel versions and apply those changes to our kernel while making sure we do not unintentionally tamper with the Android related parts of the code.

For a first evaluation, we implemented our *FixedRatio* MPTCP scheduler from [39], which assigns packets to the two interfaces based on a user-defined ratio. This choice allows us to (i) remove the impact of the scheduling decisions on the MPTCP performance and focus on smartphone-specific challenges that affect the performance (Sections 3.3, 3.4) and (ii) to experiment with different packet assignment ratios and evaluate the impact of the packet assignment ratio on the energy consumption (Section 4.2). In summary, after all the aforementioned changes, we imported around 15000+ lines of code to the Android Linux kernel.

3.2 Baseline evaluation

We measure the uplink and downlink throughput with the phone kept static 1 m away from the AP with the 802.11ad phased array facing the AP. In the downlink direction (from the AP to the smartphone), the throughput with single path TCP (SPTCP) in this setting is 1600 Gbps over the 802.11ad interface and 600

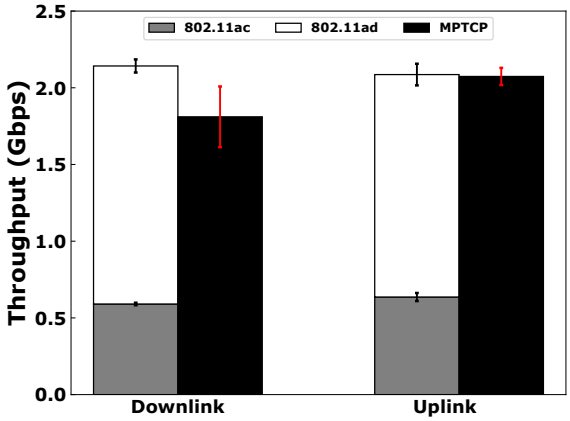


Figure 1: MPTCP throughput comparison.

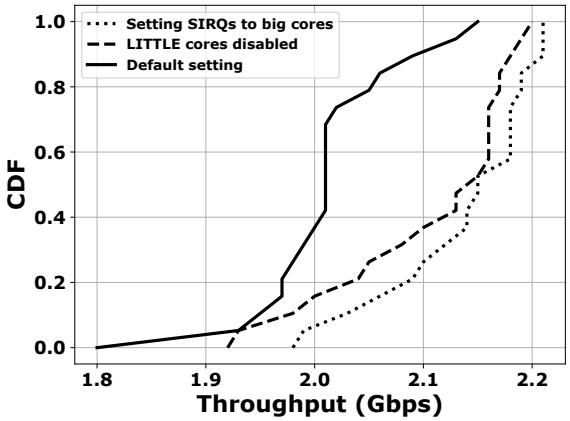


Figure 2: CDF of MPTCP throughput for various CPU settings.

Mbps over the 802.11ac interface. In the uplink direction, the corresponding values are 1450 Gbps and 600 Mbps. Hence, we expect a downlink MPTCP throughput of around 2200 Gbps and an uplink throughput of around 2050 Gbps. In [39], we showed that there exists an optimal packet assignment ratio to the two interfaces that maximizes the throughput, given by the ratio of the individual throughputs over the two interfaces. For example, in our set up for the downlink direction, where the ratio of the two throughputs is $Thput_{ad} : Thput_{ac} = 1600 : 600 \approx 2.7$, this ratio is $Pkts_{ad} : Pkts_{ac} = 73 : 27 = 2.7$, i.e., for every 100 packets, 77 packets should be sent over the 802.11ad interface and the remaining 23 packets should be sent over the 802.11ac interface. For the uplink direction, the optimal ratio is 70:30, since the uplink 802.11ad throughput is lower (1450 Gbps vs. 1600 Gbps). We set these two ratios in our *FixedRatio* scheduler to evaluate the performance of MPTCP.

Fig. 1 shows the average MPTCP throughput over 20 runs as well as the average individual throughput over each interface in the uplink and downlink direction. The error bars in this figure and all the following figures show the standard deviations. We observe that MPTCP can indeed achieve optimal performance in the uplink direction. However, in the downlink, MPTCP only achieves an average throughput of about 1810 Mbps (82% of the optimal). In the following, we discuss two challenges we had to overcome for MPTCP to achieve optimal downlink performance.

3.3 Generic Receive Offloading

We found that the reason for the suboptimal downlink performance was the fact that MPTCP by default disables the Linux Generic Receive Offloading (GRO) feature on the 802.11ac interface. In our previous work [11], we showed that GRO is critical to enable Gbps data rates in smartphones equipped with 802.11ad interfaces. While rates up to 600 Mbps over the 802.11ac interface alone can be supported without GRO using SPTCP, it is important to enable GRO over both interfaces for optimal MPTCP performance. Before performing GRO for a new TCP flow (or MPTCP subflow), the 802.11ac driver checks if the chipset supports GRO by calling the function `hdd_can_handle_receive_offload`. This function checks that the flow is a TCP flow, as GRO is not supported for UDP flows. Strangely, on the ROG Phone II, this check (through function `QDF_NBUF_CB_RX_TCP_PROTO`) fails and thus, GRO is not used. Thus, we had to bypass this check to enable GRO for the 802.11ac interface for MPTCP subflows. With GRO enabled over the 802.11ac interface, the average throughput is 2 Gbps, a 10% improvement compared to the GRO off case.

3.4 SIRQ Processing

While enabling GRO on the 802.11ac interface improves the performance, throughput remains suboptimal. Fig. 2 shows the CDF of 20 downlink MPTCP throughput measurements with GRO ON (Default setting). We observe a median throughput of 2.01 Gbps (91% of the optimal); in addition, 7/20 runs results in throughput values below 2 Gbps, with a min value of 1.8 Gbps.

To explore the root cause of the suboptimal performance, we looked at the Soft IRQ (SIRQ) processing, which accounts for a large part of the total CPU utilization under backlogged traffic at Gbps rates [11]. The ASUS ROG Phone II has a customized version of the Snapdragon 855 processor known as Kryo 485. The chipset has 8 CPU cores in total. Among them, there is a prime core that operates at 2.84 GHz and three performance cores that operate at 2.42 GHz. Together, we call them `big` cores. There are also four efficiency cores that operate at 1.8 GHz, which we call `LITTLE` cores. The `msm_irqbalance` process is responsible for scheduling the interrupt requests (SIRQs) to CPUs.

We found that, whenever SIRQs are scheduled to `LITTLE` cores, the throughput does not exceed 2 Gbps. To quickly test the hypothesis that scheduling SIRQs to `LITTLE` cores results in lower performance, we manually disabled the 4 `LITTLE` cores by setting the `/sys/devices/system/cpu/cpu*/online` flag to 0, where $* = 0, 1, 2, 3$ corresponds to the indexes of the `LITTLE` cores. The result in Fig. 2 shows a substantial improvement; the median throughput is now 2.13 Gbps (96.8% of the optimal) and the max value is 2.2 Gbps. Further, only 3/20 values are below 2 Gbps and the min throughput also increases to 1.92 Gbps, confirming that CPU affinity is indeed the key to achieving optimal performance.

Instead of manually turning off 4 out of 8 cores, a more practical solution is to stop the `msm_irqbalance` process and control the CPU affinity by setting `/proc/ irq/X/smp_affinity` to the corresponding bit map "f0" indicating `big` cores. Here, "X" is the SIRQ number we found by looking at the SIRQ database located at `/proc/interrupts`. This approach also allows for other, less CPU

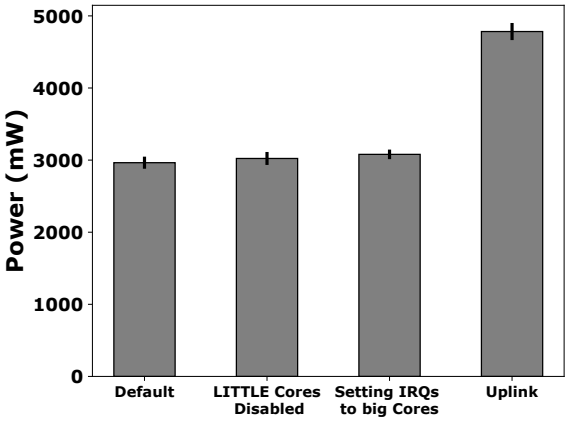


Figure 3: MPTCP power consumption.

demanding tasks, to be processed by LITTLE cores, and results in an additional performance improvement, as shown in Fig. 2. The median throughput is now 2.15 Gbps (97.7% of the optimal) and the min throughput is 1.98 Gbps (90% of the optimal).

4 ENERGY EFFICIENCY

In this section, we evaluate the power consumption and energy efficiency of MPTCP and make recommendations for an energy-aware MPTCP scheduler.

4.1 Overall power consumption

Fig. 3 shows the total power consumption in the uplink and downlink direction. In the downlink direction, we consider 3 different CPU configurations: default, with the `LITTLE` cores disabled, and with all cores enabled but the SIRQ processing assigned to `big` cores. The results are averaged over 5 runs. The downlink power consumption is comparable in all three configurations. Interestingly, distributing the SIRQs to the `big` CPU cores only causes only a minor increase in power consumption when compared to the default configuration. On the other hand, the uplink power consumption is 56% higher than the downlink power consumption, mainly due to the very high Tx power consumption of the 802.11ac radio [10]. Under backlogged traffic, the 802.11ac radio in the ROG Phone II consumes 2800 mW in Tx mode, while the 802.11ad ratio consumes only 1600 mW, in spite of the much higher data rates. In contrast, the power consumption in Rx mode is about 1100 mW for the 802.11ac radio and 2200 mW for the 802.11ad radio.

4.2 Energy efficiency

In this section, we explore whether the packet scheduling ratio that maximizes the throughput is also the most energy-efficient ratio. We again use the *FixedRatio* scheduler to measure energy per bit (in nJ/bit), defined as the ratio of the power consumption over throughput, for different packet scheduling ratios.

Backlogged traffic. The results in Figs. 4a, 4b confirm that the ratio that maximizes the throughput also results in the minimum energy cost, among all the configurations that schedule packets over both interfaces, in the downlink direction. Although using only the 802.11ad interface (i.e., a ratio of 100:0) results in slightly lower energy cost (Fig. 4b), this configuration results in only 1.6 Gbps (Fig. 4a). Utilizing both interfaces with the right packet scheduling ratio yields a 32% throughput increase with only a 1.4% increase in the energy cost.

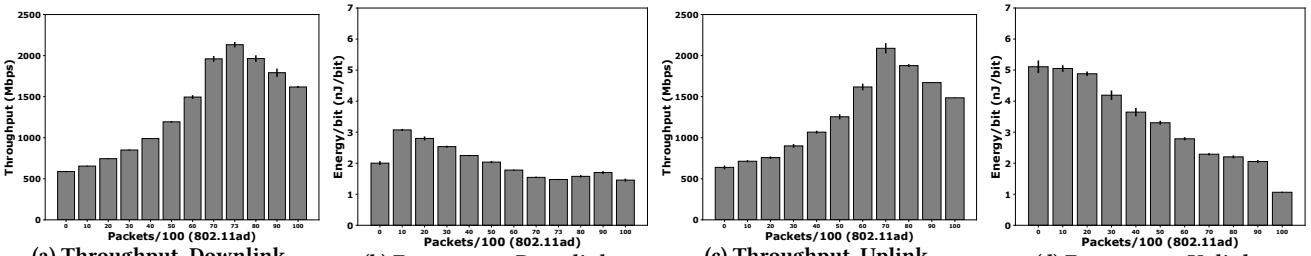
However, this finding is not true for the uplink direction, as we observe in Figs. 4c, 4d. Here, the throughput-optimal ratio is 70:30, but the energy-optimal ratio is 90:10, due to the much lower Tx power consumption of the 802.11ad radio, as we explained in Section 4.1. Nonetheless, the difference in the energy cost between the throughput-optimal and the energy-optimal ratio is small; the throughput optimal ratio results in 25% higher throughput at only 12% higher energy cost. We conclude that under backlogged traffic, the throughput optimal ratio should be selected even in an energy-aware design, as it also results in near-optimal energy cost.

Non-backlogged traffic. The work in [39] focused on backlogged traffic only. We now explore the case of non-backlogged traffic using three representative source data rates in each direction: 1750/1500/500 Mbps in the downlink case; 1750/1400/500 Mbps in the uplink case. The first data rate requires the use of both interfaces, the second one can be satisfied using the 802.11ad interface alone, and the third one can be satisfied using 802.11ac alone. Figs. 5, 6, and 7 show the throughput and energy cost for varying packet scheduling ratios for each of the three source data rates, respectively.

Figs. 5a and 5c, 6a and 6c, 7a and 7c show that, for non-backlogged traffic, there are more than one ratio that satisfy the traffic demand. In particular, in the case of traffic demands lower than the bandwidth of the slower of the two interfaces (Figs. 7a and 7c), the traffic demand can be satisfied with any ratio. However, the energy cost of each ratio is different, as shown in Figs. 5b and 5d, 6b and 6d, 7b and 7d.

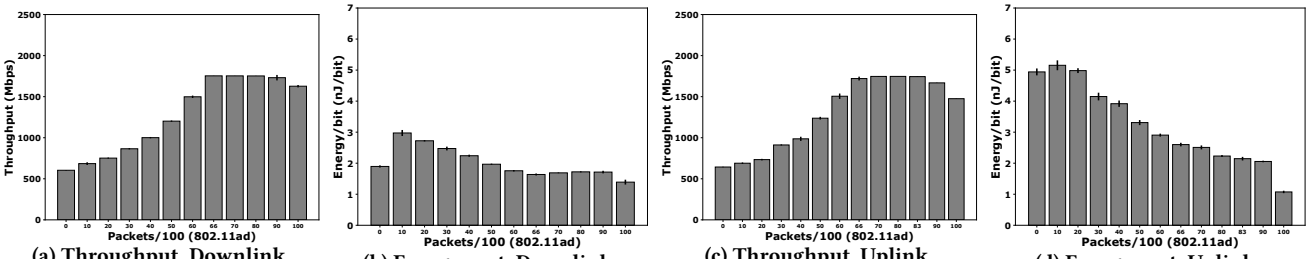
In the downlink case, we found that, when the traffic demand is higher than the bandwidth of the faster interface, the most energy efficient packet ratio is the one that fully utilizes the 802.11ac interface. This is because the 802.11ac radio is less power hungry than the 802.11ad radio in Rx mode [10]. For example, for a traffic demand of 1750 Mbps (Figs. 5a, 5b), the most energy efficient ratio is 66:34, which results in 600 Mbps downloaded via the 802.11ac interface and the remaining 1750 - 600 = 1150 Mbps via the 802.11ad interface. On the other hand, when the traffic demand can be satisfied by a single interface (Figs. 6a, 6b and 7a, 7b), the most energy efficient option is to use a single interface, and let the other interface remain in a low power (sleep) state via the Power Saving Mode (PSM). With a source data rate of 1500 Mbps, only the 802.11ad interface can satisfy the traffic demand alone and the most energy-efficient ratio is 100:0 (Fig. 6b). In contrast, with a source data rate of 500 Mbps, both interfaces can satisfy the traffic demand, but the most energy efficient option is to use only 802.11ac (Ratio 0:100 in Fig. 7b), which is less power hungry.

The situation is different in the uplink case, due to the much lower Tx power consumption of the 802.11ad radio compared to the 802.11ac radio. When the traffic demand is higher than the bandwidth of the faster interface, the most energy efficient packet ratio is the one that fully utilizes the 802.11ad interface. For example, for a traffic demand of 1750 Mbps (Figs. 5c, 5d), the most energy efficient ratio is 83:17, which results in 1450 Mbps downloaded via the 802.11ad interface and the remaining 1750 - 1450 = 300 Mbps via the 802.11ac interface. In contrast, when the traffic demand can



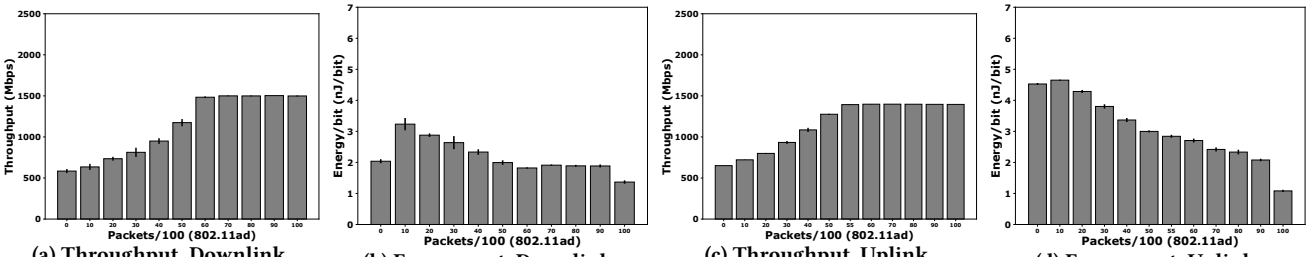
(a) Throughput, Downlink. (b) Energy cost, Downlink. (c) Throughput, Uplink. (d) Energy cost, Uplink.

Figure 4: MPTCP Throughput and Energy/bit with different ratios for backlogged traffic.



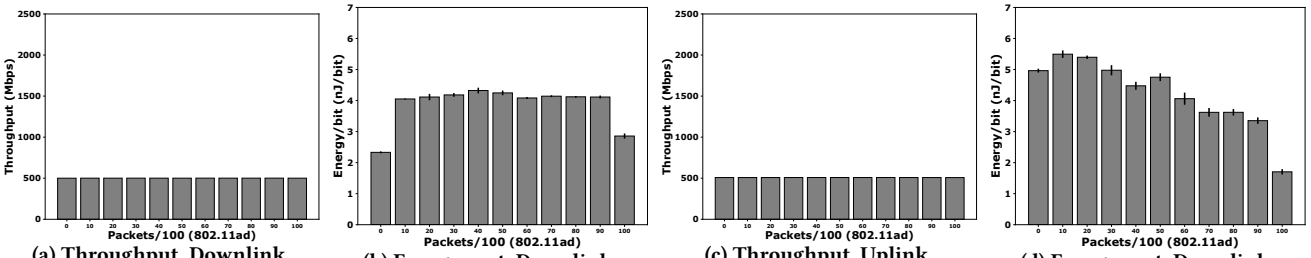
(a) Throughput, Downlink. (b) Energy cost, Downlink. (c) Throughput, Uplink. (d) Energy cost, Uplink.

Figure 5: MPTCP Throughput and Energy/bit with different ratios for source data rate 1750 Mbps. The source data rate requires use of both interfaces.



(a) Throughput, Downlink. (b) Energy cost, Downlink. (c) Throughput, Uplink. (d) Energy cost, Uplink.

Figure 6: MPTCP Throughput and Energy/bit with different ratios for source data rate 1500 Mbps (DL) and 1400 Mbps (UL). The source data rate can be satisfied with 802.11ad alone.



(a) Throughput, Downlink. (b) Energy cost, Downlink. (c) Throughput, Uplink. (d) Energy cost, Downlink.

Figure 7: MPTCP Throughput and Energy/bit with different ratios for source data rate 500 Mbps. The source data rate can be satisfied with 802.11ac alone.

be satisfied by a single interface, the most energy efficient option is to use only the 802.11ad radio (Figs. 6d and 7d).

MPTCP backup mode. When the traffic demand can be satisfied by one interface, one can either use a ratio-based scheduler to assign zero packets to the other interface, or use the MPTCP backup mode, which uses only one interface and falls back to the other interface, only if the connection over the first one breaks. Fig. 8 compares the total Rx power consumption in case of backlogged traffic over each of the two interfaces when the other interface is (i) assigned 0 packets and (ii) set to backup mode, for varying source data rates. We observe that the power consumption is very similar with both approaches, hence, either of them can be used in the design of an energy-aware scheduler.

Recommendations. The flow charts in Fig. 9 show a set of guidelines for the design of an an energy-aware MPTCP scheduler for smartphones combining 802.11ad and 802.11ac interfaces. The scheduler needs to know the application traffic demand and the maximum supported data rate over each interface. The application traffic demand can be estimated online, e.g., by monitoring the number of bytes sent over each interface, as in [39]. One potential design could be to always start with the MuSher scheduler [39],

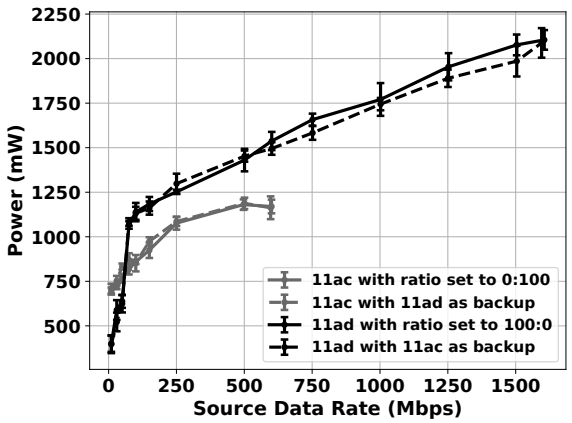
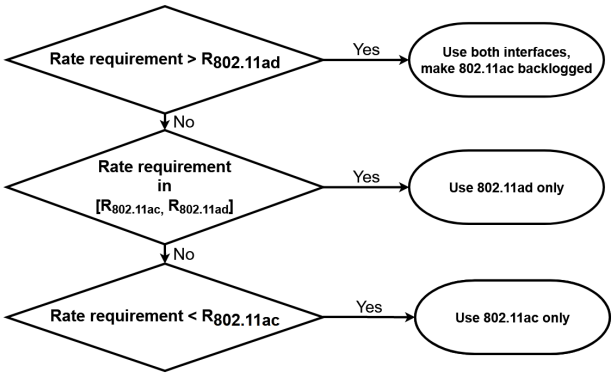
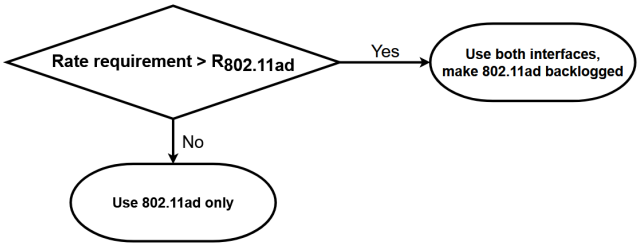


Figure 8: Rx power consumption of an interface for varying source data rates when the other interface is (i) assigned 0 packets or (ii) set to backup mode.



(a) Downlink traffic.



(b) Uplink traffic.

Figure 9: Design guidelines for an energy-aware MPTCP scheduler. $R_{802.11ad}$, $R_{802.11ac}$ is the max supported rate for 802.11ad and 802.11ac, respectively.

which searches online for the packet assignment ratio that maximizes the throughput, and compare the throughput achieved with MuSher in the steady state (after the search discovers the optimal ratio) with the max data rate supported by each interface, in order to select the most energy efficient configuration. The max data rate over each interface can either be hard-coded (all COTS 60 GHz devices support a maximum data rate of 1.6 Gbps and the majority of 802.11ac smartphones support 2x2 MIMO and 80 MHz channel width resulting in a maximum data rate of 600 Mbps, similar to the ROG Phone II) or estimated via a probing-based technique (e.g., [28]).

5 RELATED WORK

A number of works have evaluated different aspects of MPTCP performance under various scenarios, e.g., [14, 18, 20, 37, 38, 44]. In addition, researchers have proposed a large number of MPTCP schedulers, e.g., [12, 19, 24, 26, 34, 35, 42, 45]. All these works use simulation/emulation or experiments on desktops/laptops in their evaluation and they do not deal with the challenges that arise when implementing MPTCP on resource-constrained mobile devices. For example, all these works (with the exception of [18]) focus on performance benefits of MPTCP and ignore energy consumption.

A smaller set of works have studied MPTCP on smartphones [15–18, 21, 36, 41, 43]. All these works focus on bundling a WiFi interface and a cellular interface (3G or LTE), a scenario very different from the one we consider in this work (bundling legacy WiFi with 60 GHz). Cellular and WiFi radios have very heterogeneous characteristics in terms of RTT and power consumption (e.g., the existence of a *tail* state in 3G/LTE), which affect the design of both performance- and energy-aware MPTCP schedulers. Such heterogeneity is not present in 802.11ac and 802.11ad interfaces, making the design of energy-aware schedulers in this case much simpler (see Section 4.2). On the other hand, the combined data rates of older 802.11 standards (802.11b/a/g/n) and 3G/LTE standards are low enough to be handled even by older generations of mobile phones. In contrast, the Gbps data rates introduced by 802.11ad pose a challenge even for modern smartphones and require OS optimizations (Sections 3.3, 3.4) for MPTCP to yield optimal performance.

Very little work has been done towards leveraging MPTCP in networks involving mmWave links. A few works [33, 46] briefly explored the use of MPTCP in dual band 5/60 GHz WLANs and showed that it often results in lower performance than using the 802.11ad interface alone. In contrast, our previous work [39] as well as the work in [13] showed that 5 GHz and 60 GHz radios can be effectively bundled together to yield optimal performance under intelligent scheduling at the transport [39] or the link layer [13]. All these works were evaluated using dual-band laptops. To our first knowledge, this is the first work to explore the use of MPTCP in dual band 5/60 GHz smartphones.

6 CONCLUSION

In this paper, we ported for first time MPTCP to a smartphone equipped with an 802.11ad radio and performed a preliminary performance evaluation using a *FixedRatio* MPTCP scheduler. We identified two system level optimizations required for optimal performance in a resource-constrained mobile device, that are not needed in more powerful laptops: enabling GRO on the 802.11ac interface and scheduling SIRQ processing to `big` CPU cores. We also evaluated for first time the power consumption of MPTCP in a dual-band 802.11ad/ac smartphone and provided a set of guidelines towards the design of an energy-aware MPTCP scheduler.

As part of our future work, we plan to perform an extensive evaluation of the MPTCP performance and energy consumption on the ROG Phone II using different MPTCP schedulers, including the default *minRTT* scheduler and our throughput-ratio-based MuSher scheduler [39], which was designed specifically for

bundling 802.11ad and 802.11ac radios, in a variety of dynamic scenarios involving interference, human blockage, and realistic smartphone user mobility patterns. We also plan to evaluate MPTCP with mobile applications that require Gbps throughput, such as AR, VR, Miracast, and HD video streaming.

ACKNOWLEDGMENTS

This research work was sponsored in part by the NSF grant CNS-1553447, the Spanish Ministry of Science and Innovation (MICIU) grant RTI2018-094313-B-I00 (PinPoint5G+), and the Region of Madrid through TAPIR-CM (S2018/TCS-4496).

REFERENCES

[1] [n.d.]. Acer TravelMate P446-M. https://www.acer.com/ac/en/US/content/professional-series/travelmatep4.
[2] [n.d.]. *ASUS Republic of Gamers (ROG) Phone.* https://www.asus.com/us/Phone/ROG-Phone/
[3] [n.d.]. *ASUS ROG Phone II Kirisakura Kernel.* https://github.com/freak07/Kirisakura_Yoda/tree/master_q_exp_14
[4] [n.d.]. *Linux Kernel Source.* https://github.com/torvalds/linux
[5] [n.d.]. *MPTCP v0.94.* https://github.com/multipath-tcp/mptcp/tree/mptcp_v0.94
[6] [n.d.]. *Netgear Nighthawk® X10.* https://www.netgear.com/landings/ad7200
[7] Online. *ASUS Republic of Gamers (ROG) Phone II.* https://www.asus.com/us/Phone/ROG-Phone-II/
[8] [Online]. TP-Link Talon AD7200 Multi-Band Wi-Fi Router. http://www.tp-link.com/us/products/details/cat-5506_AD7200.html.
[9] Shivang Aggarwal, Moinak Ghoshal, Piyali Banerjee, and Dimitrios Koutsonikolas. 2021. An Experimental Study of the Performance of IEEE 802.11ad in Smartphones. *Elsevier Computer Communications* 169 (2021), 220–231.
[10] Shivang Aggarwal, Moinak Ghoshal, Piyali Banerjee, Dimitrios Koutsonikolas, and Joerg Widmer. 2021. 802.11ad in Smartphones: Energy Efficiency, Spatial Reuse, and Impact on Applications. In *Proc. of IEEE INFOCOM.*
[11] Shivang Aggarwal, Swetank Kumar Saha, Pranab Dash, Jiayi Meng, Arvind Thirumurugan, Dimitrios Koutsonikolas, and Y. Charlie Hu. 2019. Poster: Can Mobile Hardware Keep Up with Today's Gigabit Wireless Technologies?. In *Proc. of ACM MobiCom.*
[12] Sabur Hassan Baidya and Ravi Prakash. 2014. Improving the performance of Multipath TCP over Heterogeneous Paths using Slow Path Adaptation. In *Proc. of IEEE ICC.*
[13] Ghufran Baig, Jian He, Mubashir Adnan Qureshi, Lili Qiu, Guohai Chen, Peng Chen, and Yinliang Hu. 2019. Jigsaw: Robust Live 4K Video Streaming. In *Proc. of ACM MobiCom.*
[14] Yung-Chih Chen, Yeon sup Lim, Richard J. Gibbens, Erich M. Nahum, Ramin Khalili, and Don Towsley. 2013. A Measurement-based Study of MultiPath TCP Performance over Wireless Networks. In *Proc. of ACM IMC.*
[15] Quentin De Coninck, Matthieu Baerts, Benjamin Hesmans, and Olivier Bonaventure. 2015. Poster: Evaluating Android Applications with Multipath TCP. In *Proc. of ACM MobiCom.*
[16] Quentin De Coninck, Matthieu Baerts, Benjamin Hesmans, and Olivier Bonaventure. 2016. A First Analysis of Multipath TCP on Smartphones. In *Proc. of PAM.*
[17] Quentin De Coninck, Matthieu Baerts, Benjamin Hesmans, and Olivier Bonaventure. 2016. Observing Real Smartphone Applications over Multipath TCP. *IEEE Communications Magazine, Network Testing Series,* 54, 3 (March 2016), 88–93.
[18] Shuo Deng, Ravi Netravali, Anirudh Sivaraman, and Hari Balakrishnan. 2014. WiFi, LTE, or Both? Measuring Multi-Homed Wireless Internet Performance. In *Proc. of ACM IMC.*
[19] Simone Ferlin, Ozgu Alay, Olivier Mehani, and Roksana Boreli. 2016. BLEST: Blocking Estimation-based MPTCP Scheduler for Heterogeneous Networks. In *Proc. of IFIP Networking.*
[20] Simone Ferlin, Thomas Dreibholz, and Özgu Alay. 2014. Multi-Path Transport Over Heterogeneous Wireless Networks: Does It Really Pay Off?. In *Proc. of IEEE GLOBECOM.*
[21] Yihua Ethan Guo, Ashkan Nikravesh, Z. Morley Mao, Feng Qian, and Subhabrata Sen. 2017. Accelerating Multipath Transport Through Balanced Subflow Completion. In *Proc. of ACM MobiCom.*
[22] Muhammad Kumail Haider, Yasaman Ghasempour, Dimitrios Koutsonikolas, and Edward Knightly. 2018. LiSteer: mmWave Beam Acquisition and Steering by Tracking Indicator LEDs on Wireless APs. In *Proc. of ACM MobiCom.*
[23] Muhammad Kumail Haider and Edward W. Knightly. 2016. Mobility Resilience and Overhead Constrained Adaptation in Directional 60 GHz WLANs: Protocol Design and System Implementation. In *Proc. of ACM MobiHoc.*
[24] Bo Han, Feng Qian, Lusheng Ji, and Vijay Gopalakrishnan. 2016. MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath. In *Proc. of ACM CoNEXT.*
[25] IEEE 802.11 Working Group. 2012. IEEE 802.11ad, Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band. (2012).
[26] Nicolas Kuhn, Emmanuel Lochin, Ahlem Mifdaoui, Golam Sarwar, Olivier Mehani, and Roksana Boreli. 2014. DAPS: Intelligent Delay-Aware Packet Scheduling For Multipath Transport. In *Proc. of IEEE ICC.*
[27] 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 ACM MobiCom.*
[28] Mingzhe Li, Mark Claypool, and Robert Kinicki. 2008. WBest: a Bandwidth Estimation Tool for IEEE 802.11 Wireless Networks. In *Proc. of IEEE LCN.*
[29] Simone Mangiante, Guenter Klas, Amit Navon, Zhuang GuanHua, Ju Ran, and Marco Dias Silva. 2017. VR is on the Edge: How to Deliver 360°Videos in Mobile Networks. In *Proc. of VR/AR Network.*
[30] Arvind Narayanan, Eman Ramadan, Jason Carpenter, Qingxu Liu, Yu Liu, Feng Qian, and Zhi-Li Zhang. 2020. A First Look at Commercial 5G Performance on Smartphones. In *Proc. of ACM WWW.*
[31] Arvind Narayanan, Eman Ramadan, Rishabh Mehta, Xinyue Hu, Qingxu Liu, Rostand A. K. Fezeu, Udhaya Kumar Dayalan, Saurabh Verma, Peiqi Ji, Tao Li, Feng Qian, and Zhi-Li Zhang. 2020. Lumos5G: Mapping and Predicting Commercial MmWave 5G Throughput. In *Proc. of ACM IMC.*
[32] Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuowei Jin, Xiao Zhu, Dustin Zhang, Denis Rybkin, Michael Yang, Z. Morley Mao, Feng Qian, and Zhi-Li Zhang. 2021. A Variegated Look at 5G in the Wild: Performance, Power, and QoE Implications. In *Proc. of ACM SIGCOM.*
[33] Kien Nguyen, Mirza Golam Kibria, Kentaro Ishizu, and Fumihide Kojima. 2017. Feasibility Study of Providing Backward Compatibility with MPTCP to WiGig/IEEE 802.11ad. In *Proc. of VTC-Fall.*
[34] Dan Ni, Kaiping Xue, Peilin Hong, and Sean Shen. 2014. Fine-grained Forward Prediction based Dynamic Packet Scheduling Mechanism for multipath TCP in lossy networks. In *Proc. of ICCCN.*
[35] Dan Ni, Kaiping Xue, Peilin Hong, Hong Zhang, and Hao Lu. 2015. OCPS: Offset Compensation based Packet Scheduling mechanism for multipath TCP. In *Proc. of IEEE ICC.*
[36] Ashkan Nikravesh, Yihua Guo, Feng Qian, Z. Morley Mao, and Subhabrata Sen. 2016. An In-depth Understanding of Multipath TCP on Mobile Devices: Measurement and System Design. In *Proc. of ACM MobiCom.*
[37] Christoph Paasch, Ramin Khalili, and Olivier Bonaventure. 2013. On the Benefits of Applying Experimental Design to Improve Multipath TCP. In *Proc. of ACM CoNEXT.*
[38] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. 2012. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *Proc. of USENIX NSDI.*
[39] Swetank Kumar Saha, Shivang Aggarwal, Rohan Pathak, Dimitrios Koutsonikolas, and Joerg Widmer. 2019. MuSher: An Agile Multipath-TCP Scheduler for Dual-Band 802.11ad/ac Wireless LANs. In *Proc. of ACM MobiCom.*
[40] S. K. Saha, H. Assasa, A. Loch, N. M. Prakash, R. Shyamsunder, S. Aggarwal, D. Steinmetzer, D. Koutsonikolas, J. Widmer, and M. Hollick. 2018. Fast and Infuriating: Performance and Pitfalls of 60 GHz WLANs Based on Consumer-Grade Hardware. In *Proc. of IEEE SECON.*
[41] Swetank Kumar Saha, Abhishek Kannan, Geunhyung Lee, Nishant Ravichandran, Parag Kamalakar Medhe, Naved Merchant, and Dimitrios Koutsonikolas. 2017. Multipath TCP in Smartphones: Impact on Performance, Energy, and CPU Utilization. In *Proc. of ACM MobiWac.*
[42] Tanya Shreedhar, Nitinder Mohan, Sanji K. Kaul, and Jussi Kangasharju. 2018. QAware: A Cross-Layer Approach to MPTCP Scheduling. In *Proc. of IFIP Networking.*
[43] Yeon sup Lim, Yung-Chih Chen, Erich M. Nahum, Don Towsley, Richard J. Gibbens, and Emmanuel Cecchet. 2015. Design, Implementation and Evaluation of Energy-Aware Multi-Path TCP. In *Proc. of ACM CoNEXT.*
[44] Yeon sup Lim, Yung-Chih Chen, Erich M. Nahum, Donald F. Towsley, and Kang-Won Lee. 2014. Cross-layer path management in multi-path transport protocol for mobile devices. In *Proc. of IEEE INFOCOM.*
[45] Yeon sup Lim, Erich M. Nahum, Don Towsley, and Richard J. Gibbens. 2017. ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths. In *Proc. of ACM CoNEXT.*
[46] Sanjib Sur, Ioannis Pefkianakis, Xinyu Zhang, and Kyu-Han Kim. 2017. WiFi-Assisted 60 GHz Wireless Networks. In *Proc. of ACM MobiCom.*
[47] Sanjib Sur, Xinyu Zhang, Parameswaran Ramanathan, and Ranveer Chandra. 2016. BeamSpy: Enabling Robust 60 GHz Links Under Blockage. In *Proc. of USENIX NSDI.*
[48] Teng Wei and Xinyu Zhang. 2017. Pose Information Assisted 60 GHz Networks: Towards Seamless Coverage and Mobility Support. In *Proc. of ACM MobiCom.*