Energy and Performance of Smartphone Radio Bundling in Outdoor Environments

[†]Ana Nika, [†]Yibo Zhu, [§]Ning Ding, [§]Abhilash Jindal, [§]Y. Charlie Hu, [*]Xia Zhou
[†]Ben Y. Zhao and [†]Haitao Zheng
[†]UC Santa Barbara, [*]Dartmouth College, [§]Purdue University
[†]*{anika, yibo, ravenben, htzheng}@cs.ucsb.edu*, [*]*xia@cs.dartmouth.edu*, [§]*{ding34,jindal0,ychu}@purdue.edu*

ABSTRACT

Most of today's mobile devices come equipped with both cellular LTE and WiFi wireless radios, making radio bundling (simultaneous data transfers over multiple interfaces) both appealing and practical. Despite recent studies documenting the benefits of radio bundling with MPTCP, many fundamental questions remain about potential gains from radio bundling, or the relationship between performance and energy consumption in these scenarios.

In this study, we seek to answer these questions using extensive measurements to empirically characterize both energy and performance for radio bundling approaches. In doing so, we quantify potential gains of bundling using MPTCP versus an ideal protocol. We study the links between traffic partitioning and bundling performance, and use a novel componentized energy model to quantify the energy consumed by CPUs (and radios) during traffic management. Our results show that MPTCP achieves only a fraction of the total performance gain possible, and that its energy-agnostic design leads to considerable power consumption by the CPU. We conclude that not only there is room for improved bundling performance, but an energy-aware bundling protocol is likely to achieve a much better tradeoff between performance and power consumption.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Measurement techniques, Modeling techniques; C.2.3 [**Network Operations**]: Network management

Keywords

Radio Bundling; Energy Consumption; Throughput Performance

1. INTRODUCTION

Connectivity options for mobile devices have dramatically evolved in the last decade. First, for today's mobile devices, multiple wireless interfaces *e.g.* cellular, WiFi and Bluetooth, are the norm, not the exception. Second, cellular LTE is not only widely deployed, but offers peak download rates that match or surpass modern WiFi speeds. In a world of ever-increasing demands for improved network performance, these factors make the concept of *radio bundling* both practical and highly appealing.

A flow using radio bundling transfers data simultaneously over multiple wireless networks [1, 2, 19, 25, 35, 26], and works best at locations where WiFi and LTE performance is comparable. Today's mobile data access is dominated by public or outdoor locations, where WiFi cannot match the performance of residential/private environments due to range limitations and uncontrolled contention between public WiFi users. Our own measurements across five US cities shows that smartphones' outdoor WiFi throughput varies significantly between 0.21Mbps and 14.5Mbps, while LTE throughput at the same locations mostly range between 1Mbps to as high as 16Mbps. Another measurement study has shown that the median LTE throughput is 4-5 times higher than that of WiFi while the performance of the 3G family largely lags behind LTE [20].

While it is tempting to draw early conclusions about the efficacy and practicality of radio bundling for mobile devices, many fundamental questions remain. First, what are the performance limits of radio bundling under constraints of realistic environments? What is the *maximum* benefit that can be provided by bundling on today's devices, assuming we had an ideal transport protocol to take advantage of available bandwidth? Second, what are the energy costs of achieving that performance gain, given the increasing power demands of today's mobile applications? What role, if any, does traffic management and the CPU play in the energy profile of radio bundling protocols? Answers to these fundamental questions will shed light on the true relationship between energy consumption and bundling performance, and enable us not only to understand the performance and energy profile of current protocols, but to predict the behavior of future protocols.

The goals of our work sets it apart from prior work in several ways. First, where we seek to understand the fundamental limits of radio bundling, prior work on radio bundling consistently rely on experiments with MPTCP [30] as the default transport protocol. These studies provide a single data point in the performance vs. energy tradeoff space. And since MPTCP was not designed for wireless interfaces or considering energy constraints, these results often underestimate the potential benefits of bundling. Second, we seek to build a detailed and accurate *componentized* energy model for mobile devices, which would allow us to attribute energy consumption to specific components like the CPU. This is critical to a true understanding of the CPU's role in energy consumption in bundling scenarios. Finally, given the close relationship between energy consumption and performance, we consider it critical to characterize energy and performance on full protocol implementations on actual mobile devices.

Measurement Methodology. Our measurement study includes three key steps. *First*, we instrumented multiple LTE-capable An-

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741635.

droid smartphones to operate simultaneously on both LTE and WiFi networks, and developed an Android application that enables data transfers using bundling. We performed measurements at 63 outdoor locations in 5 US cities, characterizing energy and network performance for both radio bundling and single radio methods. *Second*, for both bundling and single radio methods, we developed accurate power models to characterize total and individual contributions of both the dual-core CPU and network interfaces (WiFi and LTE). *Finally*, we performed detailed analysis to identify critical issues in implementing smartphone radio bundling, *e.g.* traffic partitioning. We also implemented and compared a protocol-level solution (MPTCP [30]) against an application-level solution on both energy and network performance.

We summarize our key findings below.

- *Performance*: We empirically characterize energy and performance of radio bundling protocols in the wild, and compare two different bundling implementations (optimal bundling that maximizes throughput and MPTCP). Bundling always outperforms single radio access methods (*e.g.*, best radio, radio switching) in network throughput. More importantly, performance gain varies significantly across different RF environment conditions, and is sensitive to the relative throughput between WiFi and LTE links.
- *Energy Consumption*: We provide the first detailed power model on radio bundling that characterizes contributions of individual components (dual-core CPU and two radio interfaces). Bundling dual radios increases instantaneous power draw, but total energy is lowered by reduction in transfer times. More importantly, we show that the CPU can be a key contributor to energy consumption, and must be carefully considered in any energy-aware bundling solution. When modeling energy consumption in bundling protocols, our energy profiler achieves $<8\%$ error rate, compared to 17% in the state of the art [22].
- *Performance vs. Energy Tradeoff*: Finally, our measurements and analysis shows that performance gains from bundling are heavily dependent on traffic partitioning algorithms, and naive approaches can actually perform worse than single radio operations. Most importantly, MPTCP achieves only between 40%-85% of the total gains possible from bundling, and incurs a considerable energy cost in its CPU utilization. Clearly, there is both a need and an opportunity to design an energy-aware radio bundling protocol with an improved energy-performance tradeoff.

Limitations and Future Work. Our measurement study has several limitations that we plan to address in future work. *First*, because bundling is rarely used today, we only study its performance from a single user perspective without considering its network-level impact in case of wide adoption. We believe that our results can stimulate development and adoption of bundling systems, and build the groundwork for studies of network impact. *Second*, we did not consider network service costs, which can affect a user's decision on bundling and limit the amount of data transfer on LTE. We plan to investigate how bundling performs subject to this constraint in a separate study. *Finally*, while our methodology is general, our study focused on the two most popular Android phone models. We believe conclusions from our energy and performance analyses are indicative of typical smartphones today, and we are planning to expand our study to other phone/OS models.

The rest of our paper is organized as follows. First, we describe in Section 2 our methodology for enabling radio bundling, as well as our two implementations for traffic partitioning (optimal-throughput and MPTCP). Next in Section 3, we describe in detail our novel energy model for decomposing energy consumption between the CPU and network interfaces, and validate it using experimental measurements. In Section 4, we describe our data collection and dataset. This is followed by detailed analysis of performance and energy consumption in Sections 5 and 6. Finally, we describe related work in Section 7 and conclude (Section 8).

2. ENABLING & OPTIMIZING BUNDLING

Our goal is to empirically examine the performance of smartphone radio bundling at scale. In this section, we highlight the two key components of radio bundling, *turning on both radios simultaneously* and *partitioning traffic between the two radios*. We also describe our Android experimentations that evaluate radio bundling with different traffic partitioning approaches.

2.1 Turning on Both Radios

By default, modern smartphone operating systems (Android, iOS, and Windows Mobile) do not allow simultaneous use of cellular and WiFi networks. Even when the cellular connection is set to "always enabled," a smartphone will always automatically stop forwarding data packets to the cellular interface when it establishes a WiFi connection.

Using Android phones, we overcome this restriction by leveraging a small set of undocumented Android APIs, *i.e.* core API calls implemented in the *ConnectivityManager* class. The key is an API call that forces the cellular network to remain on even after a WiFi connection is made. Using these APIs, we reconfigured the Android operating system to split the routing table and forward data traffic to both WiFi and cellular radio interfaces. Depending on which source IP address they use, packets are forwarded through either the WiFi interface or the cellular interface.

On top of this modification, we developed an Android application that enables bundling-based data transfers using parallel HTTP[1] connections over both WiFi and LTE. We use the Apache HTTP Java library to open HTTP connections, since it allows us to specify a local network interface when opening an HTTP connection. We create two threads to manage the two parallel HTTP connections. To download (or upload) files concurrently on both interfaces, we set one local address to the cellular network IP and the other to the WiFi network IP. We partition HTTP downloads (or uploads) on the granularity of bytes, by specifying the start and end points of the requested segment in the HTTP header. To monitor each transfer, we also add an extra monitoring thread that continuously logs, every 100ms, LTE and WiFi signal strength, CPU frequency and usage; we also record downloaded (and uploaded) bytes by running two parallel *tcpdump* commands that listen to the two radio interfaces respectively. Later in §3 we show that these traces are used to project energy consumption of the data transfer.

2.2 Traffic Partitioning

The second component is partitioning data traffic between the two radio interfaces. This can be implemented as a stand-alone scheduler, or integrated into the transport protocol. In our experiments, we use the stand-alone scheduler to evaluate bundling with the optimal traffic partitioning that maximizes throughput, and compare it to MPTCP, which embeds traffic partitioning in TCP.

Optimal. The stand-alone scheduler determines the amount of data to transfer on each radio. The simplest approach is to equally partition the transfer amount between the two radios. But the data transfer ends when the weaker radio finishes its transmission. Instead, to maximize bundling throughput, an optimal sched-

[1]The majority of mobile applications and streaming services (*e.g.* YouTube and Netflix) use HTTP [21, 24].

uler should partition traffic such that both radios finish their assigned transfer at the same time.

Using the bundling application described in the above, we can emulate bundling with optimal traffic partitioning. Specifically, in each measurement we transfer a large chunk of data on each radio, and record the corresponding packet traces. We then "play-back" these traces based on the traffic partitioning algorithm till the total downloaded (or uploaded) bytes from both radios reaches the target transfer size. This play-back analysis enables us to identify, for each data transfer, the throughput (and delay) performance as well as the energy drain. We will discuss this further in §4.1.

MPTCP. An alternative is to integrate traffic partitioning into the transport layer protocol. Existing works [14, 22] have studied MPTCP [30] as a method to using multiple radios simultaneously. MPTCP is a transport layer protocol for multipath communications. It partitions data transfer on the fly by observing each radio's buffer status. In §5 and §6 we compare the performance and energy of MPTCP to those of optimal bundling described in the above.

To do so, we must first port MPTCP to smartphones. While prior efforts have ported MPTCP to two Android phone models [3], both models are outdated and do not support LTE. Instead, we were able to port MPTCP to the popular Galaxy Note phone by configuring the phone to Android 4.0.4, Linux kernel 3.0.8 and instrumenting all the required changes to realize MPTCP. To support data transfer in MPTCP, we set up an HTTP server on port 8080 because cellular carriers (AT&T in our case) remove all TCP header options on port 80. Finally, we note that recent measurement study [14] has shown that the choice of the radio for the primary subflow, *i.e.* the radio to turn on first, can affect MPTCP performance. However, there is no existing mechanism on how to choose this radio optimally. Thus, for our experiments we follow the default configuration in the original MPTCP implementation [3].

3. AN ACCURATE ENERGY MODEL

Another goal of our study is to understand the energy drain of radio bundling and radio selection. In this section, we describe the methodology we used to capture the energy drain of each data transfer using either radio bundling or selection.

3.1 Key Features

Accurately measuring the energy drain of radio bundling on phones in the wild is challenging. One option is to use a power meter (as in [14]). However, such an approach suffers two drawbacks: (a) the power meter is difficult to carry in field experiments and (b) it can only report the energy drain of the entire phone. Instead, we develop an accurate power model that captures the energy drain of critical phone components used in radio bundling, *i.e.*, the CPU and network interfaces, during each data transfer. Using this model, we can accurately project energy consumption of each data transfer from the CPU log and packet trace. Our projection achieves an error rate of $<$8%, compared to 17% reported by [22].

Compared with previous works studying radio bundling or radio selection, our proposed energy model includes two key new features that significantly boost the model's accuracy.

Accurately Accounting for CPU Drain. In all of the recent work on modeling power draw of cellular network interfaces (*e.g.*, [7, 20, 23, 29]) and MPTCP [22], none identified the portion of the power consumed by the CPU during data transfers, *i.e.* from interrupt handling and network stack processing. As we will show in our experiments, the CPU power draw accounts for a significant portion of the total energy consumption during data transfers (up to 39% for radio bundling and even 80% for WiFi-only), and hence not modeling the CPU power drain during data transfer can lead to significant error in power modeling[2].

We develop accurate power models for the dual-core CPU and WiFi and LTE interfaces for the phones used in our experiments. These models allow us to not only accurately derive the total energy drain of data transfers but also study the energy-throughput tradeoffs of radio bundling versus using a single radio. Our model validation below shows that incorporating CPU power modeling improves energy estimation error of data transfers to be within 8.3%.

Incorporating Signal Strength. A recent work [15] showed that the power draw of cellular and WiFi interfaces is significantly affected by the wireless signal strength. But, none of the recent work on radio selection or bundling [22] took into consideration the impact of wireless signal strength on the accuracy of power modeling. In contrast, we develop accurate signal-strength-aware power models for LTE and WiFi for the phones used in our experiments and use them in our study of energy drain of data transfers in the wild.

3.2 Model Details

CPU Power Modeling. To capture the CPU power draw during data transfers we first developed a power model for the dual-core CPUs used in the two smartphones. In training the model, we used the power meter to measure the power draw of the CPU under different frequencies with only one core turned on, while running microbenchmarks. We then repeated the process with both cores turned on. Table 2 in Appendix A shows the CPU power draw at 100% CPU utilization for both phones under a range of frequencies. Single-core results are shown with Core1 turned off.

To use the CPU model in data transfer experiments, we logged the frequencies of the two cores as well as each core's utilization once every 100ms during data transfers. Then in post-processing, we predicted the CPU power over each 100ms time interval based on the logged CPU frequency and utilization, *i.e.* as the power draw at that frequency under 100% utilization weighted by its actual utilization. Finally, we summed up the energy consumed by the CPU in each 100ms bin to arrive at the total CPU energy consumption for the entire data transfer.

WiFi and LTE Power Modeling. WiFi and LTE interfaces have multiple power states (see Appendix A) and the power draw and duration at each state and state transitions are affected by the wireless signal strength in significant ways [15]. To develop signal-strength-aware WiFi and LTE power models for our phones, we followed the procedure in [15]. Specifically, in training the models, we connected the phone to the power meter and ran data transfer microbenchmarks. While the power meter collects the power profile, we also recorded the traffic via tcpdump alongside signal strength values via Android APIs as well as core frequencies and the CPU utilization. We varied the signal strength received by the phone by adjusting the distance between the phone and the AP for WiFi experiments and changing the location of the phone for LTE experiments. In post-processing, we synchronized the power profile from the power meter, tcpdump and signal strength traces. We derived the power draw by the radio interface(s) by subtracting the CPU power from the total power. We inferred the different power states of LTE and WiFi following the procedure in [20, 15] and derived the various parameters of the signal-strength-aware power state machine for each interface (see Appendix A).

To use the power models to estimate the power draw by different interfaces during data transfer experiments, we collected *tcpdump* data and signal strength traces on each interface during data transfers. Then in post-processing, we drove our power state machines

[2]In fact, the authors of [22] reported 17% error in power modeling.

Table 1: Average prediction error of total energy on Galaxy S III

Transfer Size (MB)	0.5	1	2	4	5
Error under					
WiFi	5.5%	4.0%	7.2%	8.1%	5.4%
LTE	7.6%	1.0%	8.1%	2.6%	3.0%
Bundling	7.0%	1.7%	3.0%	3.9%	8.3%

using our tcpdump traces, and recorded the total energy consumed by each interface as it transitions between its power states.

3.3 Model Validation

We validated the above CPU/WiFi/LTE power models as follows. We use the same app as used in our measurement experiments to perform 0.5MB, 1MB, 2MB, 4MB, and 5MB data transfers, under WiFi only, LTE only, and bundling the two, respectively, at 10 random locations on the two measurement phones. During each download (or upload), the power meter is connected to the phone to read the actual power draw, and the CPU frequencies, tcpdump and signal strength traces are collected. We calculate the accuracy of the power models by comparing the predicted total energy drain against that of the power meter reading.

Table 1 shows the results for Galaxy SIII phones. The results for Galaxy Note are similar and omitted. We observe that the average prediction errors vary between 4–8.1%, 1–8.1%, and 1.7–8.3%, under the three radio scenarios respectively.

Difference from Existing Power Models. Our power models for individual radios differ from those in [20]. One key cause is the difference in phones used in the measurement: [20] used an HTC phone (Qualcomm Snapdragon S2 MSM8655 SoC, Android 2.2.1 OS) while we use two newer models: Samsung Galaxy Note (Qualcomm Snapdragon S3 MSM8660 SoC, Android 4.0.4 OS) and Samsung Galaxy SIII (Snapdragon S4 MSM8960 SoC, Android 4.0.4 OS). The other cause is the newer LTE deployment. For example, an author of [20] has confirmed with us they now also observe zero tail power for LTE on their HTC phone (which was 1000mW in [20]), same as in our LTE model (Table 3).

4. DATA COLLECTION

Before presenting the results of our throughput and energy analysis, we first describe the underlying data collection process that allow us to explore and compare radio bundling and selection efficiently. We then describe the resulting dataset used for our analysis.

4.1 Efficient Data Collection

To evaluate radio bundling/selection, we seek to compare six different options in radio usage: *LTE-only*, *WiFi-only*, *Best Radio* [13] (*i.e.* using the better radio), *Radio Switching* [25], MPTCP, and *Bundling* (with optimal data partitioning).

To do so, we face two immediate challenges. *First*, we need to ensure a fair, consistent comparison across all six options. One might use six collocated smartphones, one for each option, to perform data transfer simultaneously. This, however, introduces local traffic contention in both LTE and WiFi networks that pollutes the results. An alternative is to sequentially test each option in isolation, which can also introduce discrepancy due to temporal variations in both signal quality and network traffic. *Second*, it is important to understand the impact of transfer file size, which has been shown to largely affect the MPTCP performance [22, 14]. Yet experimenting with individual file sizes is extremely time consuming and faces the same temporal discrepancy problem.

Coupling Measurements with Replay Analysis. We address these challenges by combining sequential measurements with packet trace "replay" analysis. Specifically, we perform multiple rounds of data transfer experiments in each location. In each round of experiments, we configure the smartphones to operate in one of the three modes sequentially ("Bundling", "MPTCP", "single-radio") and record detailed packet traces. For "single-radio" we use two identical and collocated smartphones, each with only one radio on and performing data transfer (download or upload) of size xMB in total. For "MPTCP" we turn on both radios and perform data transfer of xMB in total. For "Bundling" we turn on both radios and run our bundling application to download (or upload) xMB on each individual radio[3]. For our study, we chose $x = 5MB$ so that the maximum transfer size is $5MB$. For all the experiments/locations, we observed that the three sequential experiments were separated by at most $5s$-$10s$. This time lag is small and thus the variation in network traffic condition should be small. We also examined signal strength variability during our experiments to confirm that signal strength patterns for both radio interfaces remained nearly identical across the three experiments.

Next, we perform "replay" analysis on the collected packet traces, from which we derive throughput and energy performance at varying transfer sizes. Given a target file transfer size y ($y \leq 5MB$), we identify the corresponding segment of packet traces by terminating the "replay" at the time point where the total transfer reaches y. Applying the same methodology, we characterize performance of radio switching [25] and radio selection [13] by determining the target file transfer size on each radio accordingly. For these two options, we consider their ideal realization that maximizes the average throughput. That is, we assume they all make the optimal decision on the choice of radio or traffic partitioning between the two radios. For Best Radio we compare the average throughput of WiFi-only and LTE-only and pick the radio with the better throughput. For Radio Switching we monitor the instantaneous throughput of both radios continuously to make the optimal switching decision. We use the switching overhead measured by [25].

4.2 Dataset

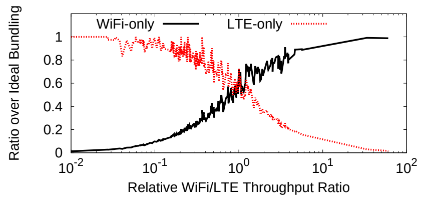
Our efforts produced data transfer measurements at 63 *outdoor* locations in 5 US cities, Boston (MA), Chicago (IL), Lafayette (IN), Miami (FL), Santa Barbara (CA), between March and October 2013. Our experiments used two Android phones: Galaxy Note and Galaxy SIII[4]. Among all the measurement locations, 32 locations were using Galaxy Note and 31 using Galaxy SIII. At each location, we performed 6 rounds of data transfer measurements, 3 for download and 3 for upload. This produced a total of 378 measurement instances where each instance contains the packet traces for all six radio usage options.

All measurement locations had LTE cellular services and public WiFi connections, *i.e.* outdoor malls, coffee shops and campus networks. We used the AT&T LTE network in all the experiments. While both phones support 802.11a/b/g/n in 2.4GHz and 5GHz, we found that all WiFi connections in our experiments used 802.11b/g/n in 2.4GHz. We performed all experiments with only our measurement applications running and no applications running in the background.

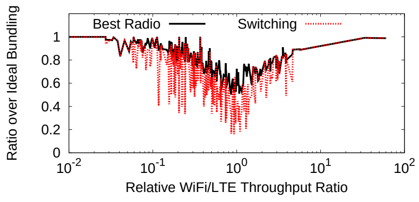
Representing RF conditions. Our measurements at locations across multiple cities allow us to evaluate radio bundling/selection

[3]The extra data transfer for "Bundling" is to ensure that we can always emulate bundling with optimal traffic partitioning.

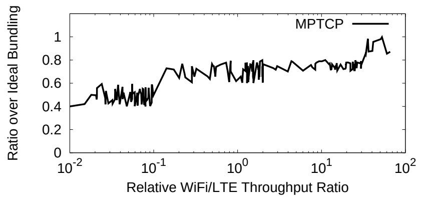
[4]While there are differet varieties of Android LTE-capable phones, our experiments used the above two smartphones because at the time of our experiments they were the two *most popular Android phones* and their radio chipsets were the two *most used chipsets* among all LTE-capable Android phones [4].



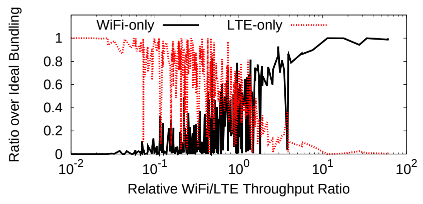
(a) Static selection vs. Bundling, avg throughput



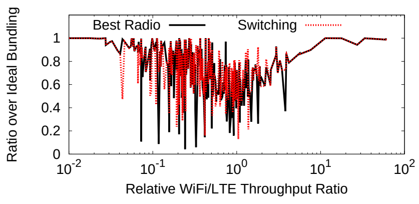
(b) Online selection vs. Bundling, avg throughput



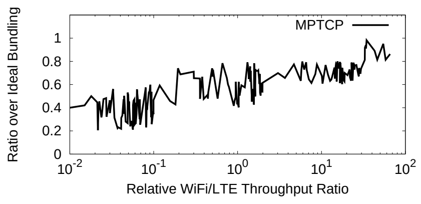
(c) MPTCP vs. Bundling, avg throughput



(d) Static selection vs. Bundling, tail Throughput



(e) Online selection vs. Bundling, tail Throughput



(f) MPTCP vs. Bundling, Tail Throughput

Figure 1: Comparing the five radio usage options with Bundling in terms of average and tail throughput. The result is shown as the ratio of throughput over Bundling R_X for option X vs. the relative WiFi/LTE throughput q.

under different RF conditions. We capture these conditions in terms of the *relative quality* of the two radios. That is, we compute the average throughput of WiFi and LTE, and define the relative quality as $q = \frac{thpt_{WiFi}}{thpt_{LTE}}$. For example, $q = 10^{-2}$ means that WiFi throughput is 100x weaker than LTE, and $q = 10^2$ means that WiFi is 100x faster than LTE in average.

Next, we perform detailed analysis on this dataset to understand the throughput and energy performance of radio bundling over radio selection and MPTCP.

5. THROUGHPUT ANALYSIS

We now present our analysis on bundling throughput. Our goal is to understand whether radio bundling provides large advantages in data throughput over radio selection, and if so, where the gain comes from. In addition, we also seek to identify whether today's bundling implementation, *i.e.* MPTCP, can fulfill this potential.

Throughput (or Delay) Metrics. For this, we evaluate the six radio usage options based on two throughput metrics. The first metric is *average throughput*, computed as the total transfer amount divided by the total transfer time. The second metric is *tail throughput*, computed as the bottom 10 percentile value of the instantaneous throughput during the data transfer, where the instantaneous throughput is computed every $100ms$. Essentially, the average throughput directly reflects the overall download (or upload) delay, while the tail throughput reflects the maximum instantaneous delay in each direction.

Using the dataset described in §4, we compare the five radio usage options (LTE-only, WiFi-only, Best Radio, Radio Switching, MPTCP) to Bundling (with optimal data partitioning) in terms of the normalized ratio over Bundling:

$$R_X = \frac{THOUGHPUT_X}{THOUGHPUT_{bundling}}, \qquad (1)$$

where X is one of the five options. As mentioned in §4, we apply sequential measurements with replay data analysis such that the comparison across different options is fair and consistent. We mix the results of the two smartphones as well as download and upload scenarios because they lead to the same pattern in R_X.

5.1 Bundling vs. Radio Selection

We first compare Bundling to static radio selection, *i.e.* LTE-only and WiFi-only, and online radio selection, *i.e.* Best Radio and Radio Switching. In Figure 1 we plot the R_X results against q, the relative WiFi/LTE quality, for both the average and tail throughput. We have tested different data transfer sizes between 512K and 5MB, and found that they led to similar trends (discussed later in §5.3 and Figure 3). For brevity we only show the results for 5MB.

Improvement over Static Radio Selection. Figure 1(a) shows that in terms of average throughput, the R_X of the two static radio selection options are both strongly correlated with q, *i.e.* $R_{LTE-only} \approx \frac{1}{1+q}$ and $R_{WiFi-only} \approx \frac{q}{1+q}$. This is because the throughput of Bundling is pretty much the sum of the LTE radio throughput and the WiFi throughput.

On the other hand, Figure 1(d) shows that in terms of tail throughput, the gain of Bundling is much more evident and displays a larger variance over q. For example, the gain can reach more than 19x when LTE is 5-10x stronger than WiFi. This is because during data transfer, each individual radio often experiences temporal variations in signal quality, *e.g.* due to channel fading or interference. Even after averaging over 100ms, the corresponding "instantaneous" throughput trace can still contain deep holes. Bundling, on the other hand, can effectively suppress these deep holes by using both radios simultaneously.

Improvement over Online Radio Selection. For *Best Radio* that always picks the stronger radio for data transfer, the R_X of average throughput is the upper bound of WiFi-only and LTE-only: $R_X \approx \max(\frac{1}{1+q}, \frac{q}{1+q})$. In this case, Bundling is the most beneficial (*i.e.* with 2x throughput gain) when the two radios have similar throughput ($q = 1$). In terms of tail throughput, the gain of Bundling can be significant (>10x) regardless of q. This is because Best Radio still uses the same radio for the entire data transfer who can experience deep fades over time.

When it comes to *Radio Switching* which selects the stronger radio on the fly, the gain of Bundling can reach 5x for average throughput and 10x for tail throughput (see Figure 1(b)(e)). This is somewhat surprising since Radio Switching should perform better than Best Radio. We found that this is due to the overhead of Radio Switching. As shown by [25], the minimum time to switch from cellular to WiFi is 1212ms and from WiFi to cellular 196ms, during which data transmission is disabled. As a reference, we also compared Bundling to instantaneous Radio Switching, *i.e.* zero switching delay, and found that Bundling still achieves up to 2x throughput gain.

Source of Bundling Gain. Our results show that radio Bundling can significantly boost smartphone throughput. Next we dig deeper in our results to understand the key triggers for such improvement. Specifically, we seek to understand whether Bundling can fully utilize both radios. One concern is that Bundling could negatively affect the performance of each single radio, *e.g.* due to CPU competition or radio interference.

To answer this question, we compare each radio's individual throughput with and without bundling using our measurements on WiFi-only, LTE-only and Bundling[5]. Our results (omitted for brevity) confirm that for both WiFi and LTE radios, throughput remains similar with and without bundling. This shows that at least for two popular Android phones [5, 6], bundling LTE and WiFi radios does not negatively impact each radio's performance. That is, the throughput achieved by Bundling is the sum of each radio's throughput operating in isolation.

We suspect that the observed radio independence is due to two factors. *First*, the carrier frequencies of LTE (700/1700MHz) and WiFi (2.4GHz) are widely separated, so the two co-located radios do not physically interfere with each other. *Second*, modern smartphones have two CPU cores, and multi-threaded applications such as our bundling application minimize CPU contention by spreading load across cores. We confirm this via observing the CPU utilization on the two cores (Core 0 and Core 1) of Galaxy Note and Galaxy SIII phones. Specifically, we observe that for all the experiments without bundling, Core 0 is utilized while Core 1 is rarely used. Yet during bundling, Core 1's utilization increases largely while that of Core 0 remains unaffected.

5.2 Bundling vs. MPTCP

Figure 1(c) and (f) compare MPTCP to Bundling with optimal traffic partitioning. In terms of average throughput, MPTCP is able to achieve 50%-80% of the optimal bundling performance. The gap is slightly larger for tail throughput. A more detailed analysis on the traces reveals that the performance gap is mostly caused by the fact that MPTCP often "sacrifices" its throughput to maintain fairness between MPTCP and non-MPTCP clients. For example, if one radio path becomes congested, MPTCP immediately lowers the data transfer on this path more than that of conventional TCP designs. Thus, the current MPTCP design is unable to fully utilize both radios during bundling.

We also observe that compared to radio selection, MPTCP is insensitive to the relative RF condition between WiFi and LTE. This comes from the fact that MPTCP can utilize two radios simultaneously who compensate each other during deep fades. However, when the difference between LTE and WiFi radios is large ($q < 10^{-1}$ and $q > 10^{1}$), online radio selection outperforms MPTCP largely. This is partially because our online radio selection assumes optimal choice of radio to use while MPTCP uses a sub-optimal traffic partitioning algorithm.

Importance of Traffic Partitioning. To further understand the impact of traffic partitioning on Bundling performance, we repeat our analysis using three sub-optimal traffic partitioning algorithms:

- *Even*: It simply partitions the data transfer into two equal parts, one for each radio.
- *Naive*: It first sends a small amount of data, *e.g.* 100KB[6] on the two radios to estimate their throughputs. It then partitions data based on the ratio of the two throughput values.

[5]As discussed earlier in §4, while these measurements take place sequentially, we only consider data where the signal strength pattern remains similar across all three measurements.

[6]We tried 200KB and 300KB and they led to similar results.

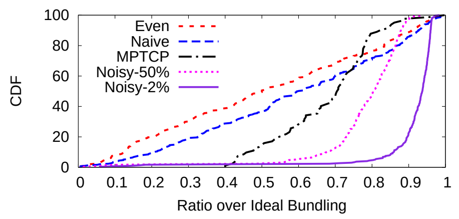


Figure 2: Impact of traffic partitioning in bundling implementation.

- *Noisy*: It assumes a *noisy* estimate of the average throughput for each radio over time, and uses this metric to estimate the transfer amount for each interface. For this, we add zero-mean Gaussian noise on the actual average throughput and set its standard deviation to be 2% (and 50%) of the actual average throughput.

Figure 2 compares the above algorithms and MPTCP to the Bundling with optimal traffic partitioning. We make three key observations. *First*, algorithms like "even" that do not consider the current radio quality perform poorly. In fact it is often worse than single radio methods. *Second*, the bundling performance is sensitive to errors in projecting each radio's throughput. Among Naive, Noisy-2% and Noisy-50%, Naive performs the worst because it uses the instantaneous throughput measured in the first 400-600ms to estimate the average throughput over time, leading to the largest prediction error. For 90% of the measurement instances, Noisy-2% achieves at least 85% of the optimal bundling throughput, while Noisy-50% achieves at least 70%. This again demonstrates the importance of accurate throughput estimation in implementing bundling. Finally, by partitioning traffic on the fly based on buffer status, MPTCP lies in between Naive and Noisy-50%. Thus, there is definitely room for further improvement.

5.3 Impact of Data Transfer Size

We also carry out our throughput analysis under transfer data size between 512KB to 5MB. We chose these file sizes because recent measurement studies have shown that smartphone traffic is dominated by large media flows that consist of data chunks of 1MB or larger [21]. For example, typical YouTube files are divided in blocks of size 512KB, while Netflix transfers blocks of either 1.8MB or 5.2MB in size [12].

Figure 3 plots, for the five transfer sizes between 512KB and 5MB, the CDF of R_X (for average throughput) across all the measurement instances. We observe similar trends in tail throughput and thus omit the results for brevity. For both MPTCP and the four configurations of radio selection, the statistics of R_X remain consistent across all transfer sizes. The values of R_X are slightly higher for 512KB and 1MB transfers, but quickly converge as the transfer size grows. This is mostly because TCP slow start prevents the bundling link to be fully utilized and the effect is much more visible for small transfers.

5.4 Summary of Results

Overall, our results indicate that the gain of radio Bundling over radio selection and MPTCP is consistent across transfer sizes above 512KB. Thus, Bundling, with optimal traffic partitioning, can consistently boost throughput (and delay) for most smartphone media applications that demand high throughput. The bundling gain comes from two factors, *radio independence*, *i.e.* the LTE and WiFi operate independently from each other, and *optimal traffic partitioning* that accurately projects radio throughput over time in order to fully utilize both radios.

6. ENERGY ANALYSIS

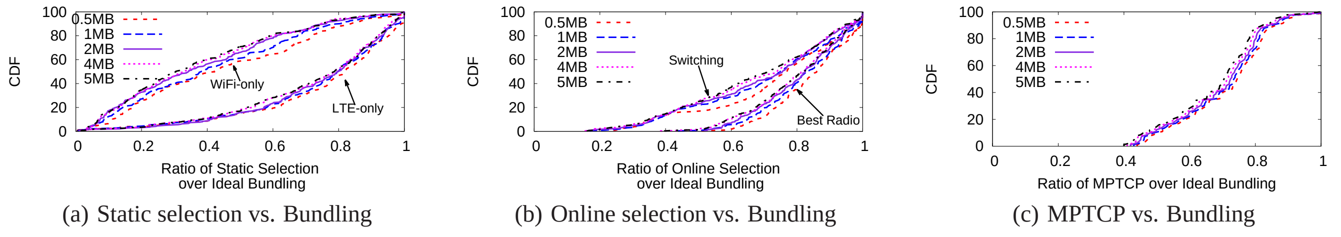


Figure 3: CDF of the ratio of throughput over Bundling R_X for option X (LTE-only, WiFi-only, Radio Switching, Best Radio, and MPTCP) for five different file sizes: 0.5MB, 1MB, 2MB, 4MB, 5MB.

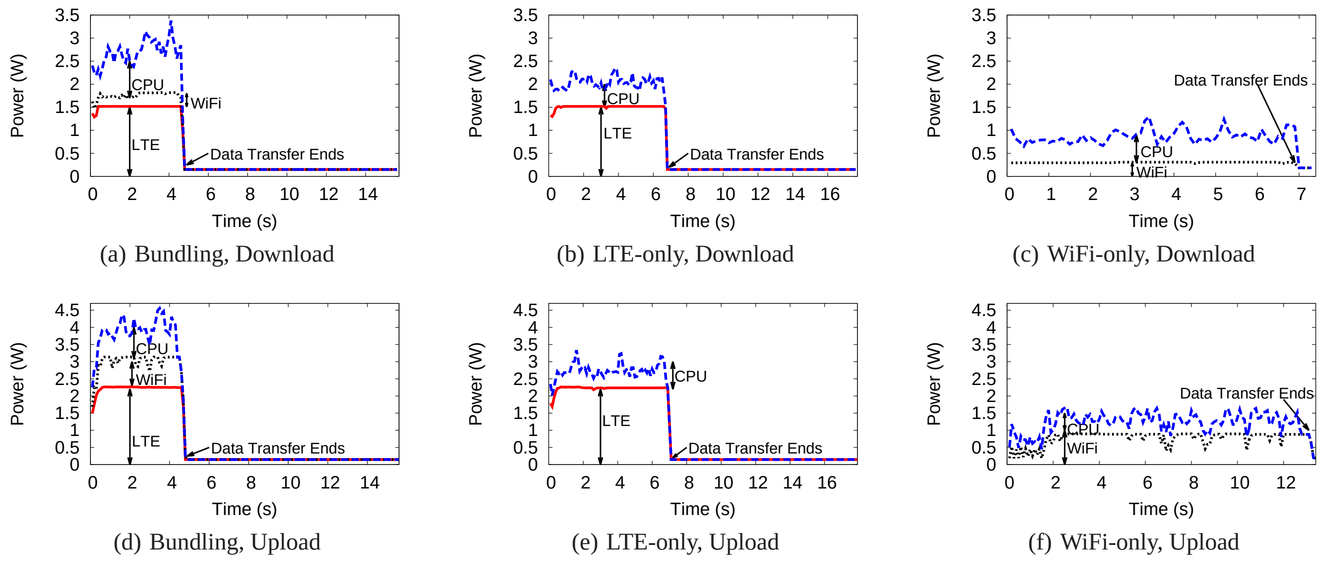


Figure 4: A sample plot of power draw of Bundling, LTE-only and WiFi-only in terms of WiFi radio, LTE radio and CPU for a random measurement location while downloading/uploading 5MB. We stack the power draw of the three components to illustrate both the individual contributions and the total power draw value during each transfer. The spikes of LTE and WiFi energy tails do not appear in these figures because they were averaged out in each 100ms interval.

Our measurements in §5 show that bundling LTE and WiFi radios can largely boost smartphone throughput. Yet one may wonder whether such improvement comes with a heavy cost in energy consumed by activating one additional radio. In this section, we perform a detailed analysis of energy consumption on radio bundling using the energy model described in §3. Different from existing models [20, 22], our model separates the energy consumed by CPU and WiFi/LTE radios. This not only improves accuracy of energy estimation (error rate $<$ 8% compared to 17% in [22]), but also allows fine-grained analysis of radio bundling and selection.

We present results on both instantaneous power draw and total energy drain per transfer for five radio usage options: LTE-only, WiFi-only, Best Radio, MPTCP, and Bundling (with optimal data partitioning). We are unable to examine Radio Switching because there is no available implementation for real-time switching between LTE and WiFi[7]. We leave this to future work. The logging needed for energy estimation was collected along with all the throughput measurements.

6.1 Componentized Power Draw

To provide context for our discussion on the total energy consumption per transfer, we first study how different smartphone components contribute to energy consumption. Using the energy profiler described in §3, we break down energy consumption into three major components related to bundling: *WiFi radio*, *LTE radio* and *CPU*. We then extract the instantaneous power draw (energy consumption in each 100ms slot) per component during each data transfer. We do not include any result on screen energy cost since the data transfer can be done with screen on or off. Figure 4 presents a sample power draw result for bundling with optimal traffic partitioning, LTE-only and WiFi-only, obtained by transferring 5MB data using a Galaxy Note phone at a randomly chosen location.

Next we analyze the three components in details.

LTE vs. WiFi Radio. Across our experiments with both phones we found that the LTE radio consumes at least 50% of the total power draw (see Figure 5(a) that plots the normalized contribution of the LTE radio). On average, it consumes 5+ times more power than the WiFi radio. For example, in the download example of Figure 4, the LTE radio takes 1.5W power draw while WiFi takes only 0.25W, and for uploads, LTE takes 2.2W while WiFi uses 0.75W. Therefore, *LTE is the heaviest energy drainer across the three components.* This observation aligns with existing works [15, 20].

CPU Contribution. For both dual-core smartphones, CPU consumes a significant amount of energy. Figure 5(b) plots the CDF of the normalized CPU energy cost over all the measurement instances. We mix the results of the two smartphones since they are similar. Overall, the median value is 20% for LTE-only and

[7]To the best of our knowledge, the Android version of radio switching [25] only applies to 3G/WiFi and is not available publically.

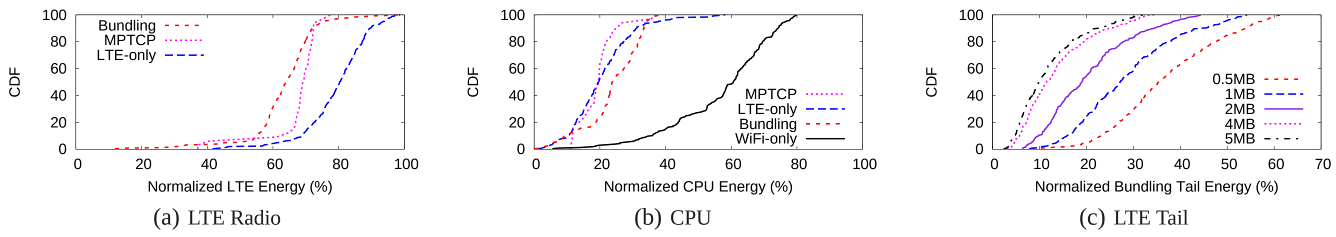


Figure 5: Contribution of LTE radio, CPU and LTE tail to the overall energy drain.

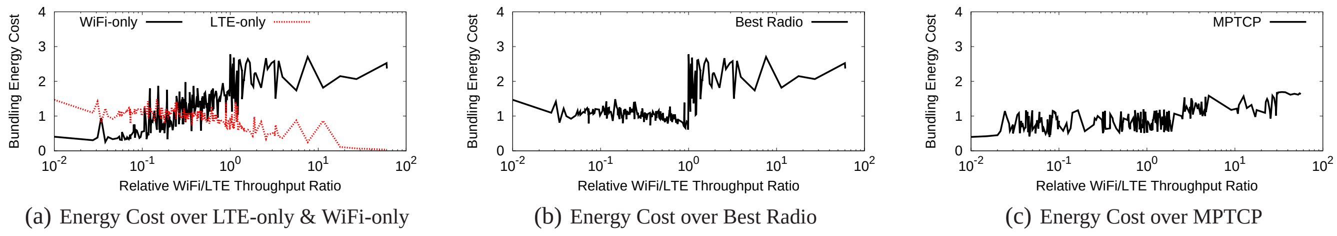


Figure 6: Energy cost of radio bundling over LTE-only, WiFi-only, Best Radio, as well as MPTCP across all measurement instances.

MPTCP, 23% for Bundling, and rises to 60% for WiFi-only, while the maximum value can reach 80% for WiFi-only. To the best of our knowledge, this is the *first* field measurement over many locations that shows data transfers can consume significant CPU energy, using popular multi-core smartphones. More importantly, our result shows that *an accurate energy model must not ignore the CPU contribution*.

We also observe that bundling's CPU power draw is approximately the sum of those of WiFi-only and LTE-only. This can be explained by our measurements on CPU utilization (in §5.1). When only a single radio is active, CPU Core 0 is utilized while Core 1 is rarely used. During bundling, Core 1 becomes active while Core 0's utilization remains unaffected.

Radio Energy Tail. Both LTE and WiFi radios stay in moderate power state for a period of time after the data transfer finishes [27]. This is the tail state and lasts 11s for LTE and 210ms for WiFi [20]. The power draw displays sudden spikes (of length 40ms) during the tail state. They are averaged out by the 100ms display interval and thus not visible in Figure 4.

As shown by Figure 5(c), our key finding here is that *the energy contribution of these tails, especially the LTE tail, depends on transfer size*. For small files ($\leq$1MB), the LTE tail contributes to up to 60% of energy in the bundling mode (with a median of 25-35%). As the data size increases, the contribution reduces to no more than 30% (with a median of 10%)[8].

6.2 Energy Consumption per Transfer

Next we study the total energy consumption of a data transfer, by first comparing bundling to radio selection and then to MPTCP. We use the metric of the *normalized energy cost of bundling over non-bundling*:

$$Cost_X = \frac{ENERGY_{Bundling}}{ENERGY_X} \quad (2)$$

where X is a non-bundling strategy. If $Cost_X > 1$, then bundling consumes more energy than strategy X.

Bundling vs. Radio Selection. Figure 6(a)(b) plot the value of $Cost_X$ vs. the ratio of WiFi and LTE average throughput, for 5MB transfers. Later we will show the impact of transfer size in Figure 7. We mix the results of the two phones as well as download and upload scenarios since they are similar.

We make three key observations. *First*, bundling consumes less energy compared with LTE-only, despite turning on the extra WiFi radio. This is because bundling improves the transfer throughput so that the transfer ends earlier. As a result, the energy saving due to reduced transmission time compensates the extra power draw of the WiFi radio. To further illustrate this, we plot in Figure 8 bundling's energy cost as a function of its throughput gain (inverse of the throughput ratio R_X defined in §5). We see that in general the energy cost scales inversely with the throughput gain.

Second, compared to WiFi-only, bundling's energy cost fluctuates around 1 when the WiFi-radio is weaker, but increases to more than 2 when the WiFi radio achieves higher throughput than the LTE radio. Again this is due to the fact that bundling's throughput gain over WiFi-only is higher when WiFi is weaker. The resulting shortened transfer time effectively reduces energy drain despite the fact that bundling turns on the energy-hungry LTE radio (Figure 8).

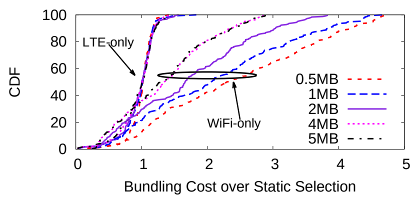
Finally, the energy cost over Best radio follows a bimodal curve: it equals to the cost over LTE-only when the WiFi radio is weak and jumps to that of WiFi-only once the WiFi radio becomes stronger than the LTE radio.

Overall, these results show that the energy consumption of radio bundling is on par with that of radio selection. Although bundling uses an extra radio and thus consumes more power draw (radio+CPU), its throughput improvement reduces data transfer time and effectively compensates the increase in power draw.

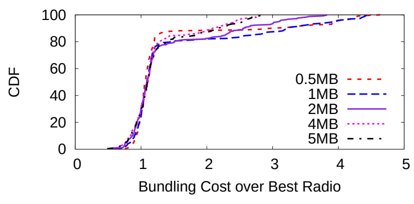
Bundling vs. MPTCP. Next we compare the energy consumption of optimal bundling and MPTCP. We note that by partitioning traffic at the transport layer, MPTCP only uses one HTTP connection while optimal bundling uses two parallel HTTP connections, one for each radio.

Figure 7(c) plots the bundling energy cost over MPTCP for 5MB data download. Interestingly, the value fluctuates between 0.4 and 1.6, indicating that MPTCP can sometimes save energy! This is unexpected since our throughput results in §5 shows that bundling always outperforms MPTCP in throughput, so MPTCP must run longer to complete the transfer. After studying the energy traces carefully, we find that the key cause is that bundling consumes more CPU energy by maintaining two parallel HTTP connections with two threads, which activates two CPU cores. MPTCP, on the other

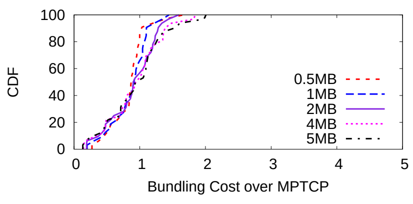
[8]As discussed in §3, this result, coming from the new LTE deployment, differs from [20] which observed a much larger energy tail.



(a) Energy Cost over Static Selection



(b) Energy Cost over Best Radio



(c) Energy Cost over MPTCP

Figure 7: CDF of bundling energy cost over LTE-only, WiFi-only, Best Radio, and MPTCP for different file sizes.

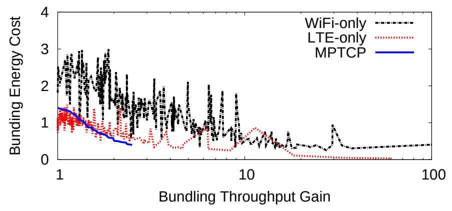


Figure 8: Bundling's energy cost vs. its throughput gain.

hand, only manages a single thread and runs in a single core. We confirm this by comparing the CPU power draw of the two modes. So while MPTCP turns on both radios like bundling, its reduction in CPU usage can lead to savings in the overall energy consumption. As shown in Figure 8, this saving becomes visible when bundling's throughput gain is small, *i.e.* between 1 and 2, but gets wiped out as bundling's throughput gain increases.

Impact of Data Transfer Size. We repeated the above energy analysis for different data transfer sizes: 512KB, 1MB, 2MB, 4MB, and 5MB. Figure 7 plots the CDF of energy cost across all the measurements. Overall, the values of energy cost remain consistent across these data transfer sizes, except for the cost of bundling over WiFi-only. This is because the impact of LTE energy tail rises as the transfer size becomes smaller. Compared to WiFi-only, bundling's extra LTE energy tail becomes a significant overhead in energy. Yet even for small transfers of 512KB, bundling consumes at most 4.5x energy over WiFi-only (with a median of 2.5).

6.3 Summary of Results

Our analysis on energy consumption shows that bundling, by using two radios, has a higher instantaneous power draw than single radio methods. Yet such increase in power draw is effectively compensated by the shortened data transfer time (due to bundling's throughput gain). As a result, bundling achieves a similar total energy consumption per transfer compared to LTE-only and at most 3.5x more energy than WiFi-only and Best Radio (due to the high energy cost of the LTE radio).

MPTCP and bundling achieve similar energy consumption since they both use two radios simultaneously. However, MPTCP, by using a single thread implementation, reduces CPU usage and thus CPU power draw. This often turns into energy savings per data transfer. This result suggests that one must account for the CPU contribution when evaluating and designing radio bundling. We are able make this observation because our energy model can accurately project the energy consumed by CPU and radio interfaces separately.

Designing Energy-efficient Bundling. Our goal in this paper is to understand the energy cost behind bundling's throughput improvement. Thus we target an optimal bundling design that maximizes throughput without considering energy consumption. However, we believe that the insights from our study will be valuable in the design of an energy-aware radio bundling protocol with an improved energy-performance tradeoff. We leave this to future work.

7. RELATED WORK

Multi-Radio Usage. Existing works have studied how to select the best network for each flow when multiple networks are available [8, 9, 10, 34]. A recent survey summarizes techniques on multi-radio aggregation [17]. To select the best interface for each application, existing works have considered application requirements [18, 13] and energy [25]. Recent works have studied concurrent use of 3G/WiMAX and WiFi radios on laptops [30, 19, 35, 36] and smartphones [35, 33]. [35] considers bundling WiFi/WiMAX, while [33] proposes to offload TCP ACKs from WiFi to 3G to improve TCP performance. There are also two Android apps that use both cellular 3G/4G and WiFi interfaces [1, 2].

Our work leverages insights from existing studies, but differs significantly from them by measuring and characterizing energy and performance of bundling smartphone LTE/WiFi radios at many locations. To our best knowledge, this is the first empirical study on this subject. Finally, our work also provides the first detailed, validated power model on various components during radio bundling.

MPTCP. Recent works have evaluated MPTCP throughput and energy performance. [26] studied MPTCP handover between 3G/WiFi on a Nokia N950 smartphone and measures energy consumption at one location. Chen et al. [12] analyzed MPTCP performance for different flow sizes using laptops. Deng et. al [14] studied flow-level MPTCP performance under different traffic patterns. They ran MPTCP on a laptop that sends flows to two tethered smartphones and measured energy using a power meter. Lim et al. [22] proposed a power model for MPTCP and examined MPTCP energy consumption for different file sizes. The error rate of the proposed power model is 17%.

Our work differs from these prior works in three key aspects. First, we target bundling with optimal traffic partitioning to explore the full benefit of bundling over radio selection. We also examine whether MPTCP can fulfill the bunding gain. Second, we ported MPTCP to smartphones and performed measurements at scale. Third, we developed an accurate energy model that captures the contribution of individual components (CPU and radio interfaces). Our energy model achieves an error rate of 8%.

Smartphone Energy Characterization. Prior works have examined smartphone energy consumption in a single radio operation mode [11, 32, 37, 23, 28, 29]. Recent work also developed specific power models for WiFi, 3G, GSM, and LTE [7, 15, 20, 31, 16]. Our study leverages insights from these existing studies on power measurements and profiles. The unique contribution of our work is to develop (and verify) detailed power models for both CPU and individual radio components that apply to both radio bundling and single radio operations. Our work considers the two of today's most used Android phone models, and reveals power usage patterns that differ significantly from prior works.

8. CONCLUSION

In this paper, we re-assess the relationship between energy consumption and performance in smartphone radio bundling scenarios. Instead of characterizing bundling performance with MPTCP, we study the more general problem of maximal performance gain in ideal scenarios. Our results show that MPTCP achieves only a portion of the total performance gain possible. In addition, we build an accurate componentized power model that identifies the significant role mobile CPUs play in energy consumption. Most importantly, our study demonstrates a clear tension between network performance and power consumption (from both radio transmissions and traffic partitioning at the CPU).

There are two key takeaways from our study. First, an energy-agnostic MPTCP achieves only a fraction of the possible performance gains in a radio bundling scenario. Second, given our insights into the role of CPU in power consumption for radio bundling systems, there is ample room for a new bundling protocol that provides a better tradeoff between performance and energy consumption.

Appendix A: Details of the Power Models

Dual-Core CPU Power Model. Table 2 shows the power model for the dual-core CPUs on the two phones used in our study, *i.e.*, the power draw when the two cores are at different combinations of frequencies, derived by following the procedure described in §3.

Table 2: Dual-core CPU power model for Galaxy Note and Galaxy SIII, sampled on 6 frequencies for each core. The unit of power is mW.

Galaxy Note							
Core0 (Mhz)	Core1 (Mhz)						
	0	384	594	810	1026	1242	1512
384	263	363	440	577	677	907	1129
594	322	366	492	629	655	966	1184
810	470	574	629	710	729	1047	1273
1026	562	670	729	810	847	1132	1365
1242	781	903	958	1051	1132	1262	1480
1512	781	1117	1177	1269	1354	1476	1680

Galaxy SIII							
Core0 (Mhz)	Core1 (Mhz)						
	0	384	594	810	1026	1242	1512
384	296	744	766	818	873	977	1047
594	359	766	814	866	921	1036	1103
810	411	818	866	918	973	1080	1154
1026	455	873	921	977	1029	1136	1217
1242	555	981	1029	1084	1140	1199	1277
1512	633	1062	1106	1158	1221	1273	1351

LTE and WiFi Power Models. The LTE interface on smartphones has four power states. The power states and their transitions are shown in Figure 9(a): (1) *IDLE:* The interface is in idle states when the User Equipment (UE) does not send or receive any data. It consumes little power and periodically wakes up to check whether there are incoming data buffered at the network. (2) *CR:* When the UE sends or receives any data, the interface enters the Continuous Reception (CR) state and consumes high power. (3) *Short DRX:* After the UE finishes data transfer and becomes idle for 200ms, the interface enters the Short DRX state, consumes little power but wakes up frequently to check for incoming traffic. (3) *Long DRX:* The interface enters this state after staying in Short DRX for 400ms without receiving any data. Long DRX is similar to Short DRX except that the wakeup interval becomes longer. Finally, if the UE stays in Long DRX for 11s without receiving any data, the interface returns to the IDLE state; otherwise, any data transfer in Short or Long DRX states will trigger the interface to enter the CR state.

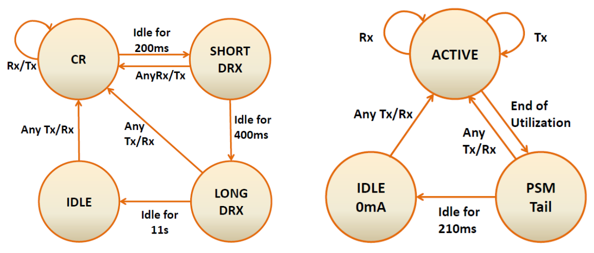
Table 3: Parameters of signal-strength-aware power models for WiFi and LTE on Galaxy Note and SIII. The unit of Tx/Rx/Tail power is mW.

WiFi						
RSSI (dBm)	Galaxy Note			Galaxy SIII		
	Tx	Rx	Tail	Tx	Rx	Tail
-50	373.7	214.6	185.0	333.0	262.7	185.0
-60	862.1	292.3	185.0	444.0	281.2	185.0
-70	906.5	292.3	185.0	518.0	314.5	185.0
-80	869.5	358.9	185.0	662.3	373.7	185.0
-90	884.3	314.5	185.0	643.8	355.2	185.0
The duration of WiFi tail for both phones is 210ms.						

Galaxy Note LTE			
	Power(mW)	Duration(ms)	Periodicity(ms)
LTE promotion	950.0	300	N/A
Short DRX	947.2	36	100
Long DRX	932.4	40	320
LTE tail base	0	11000	N/A
DRX in IDLE	780.7	25	1280

Galaxy SIII LTE			
	Power(mW)	Duration(ms)	Periodicity(ms)
LTE promotion	1200.0	300	N/A
Short DRX	788.1	41	100
Long DRX	788.1	45	320
LTE tail base	0	11000	N/A
DRX in IDLE	569.8	32	1280

	Note LTE		SIII LTE	
RSRP (dBm)	Tx (mW)	Rx (mW)	Tx (mW)	Rx (mW)
-90	2160.8	1391.2	2264.4	1280.2
-100	2312.5	1646.5	2275.5	1535.5
-110	1861.1	1949.9	2282.9	2327.3



(a) LTE (b) WiFi

Figure 9: WiFi and LTE state machines for Galaxy Note and SIII.

Figure 10 plots the LTE power states on Galaxy SIII in a 100KB download under good signal strength (-90dBm).

The WiFi interface also has four power states: Tx, Rx, Tail, and Idle (Figure 9(b)). The interface is in the Idle state when there is no traffic, and enters the Tx (Rx) state when it starts sending (receiving) data. After data transfer, the interface stays in the Tail state for 210ms before returning to the Idle state. The interface consumes very little power in the Idle state, moderate power in the Tail state, and high power in the Tx and Rx states.

For both LTE and WiFi interfaces, the power draw and duration at each state and state transitions are affected by the wireless signal strength [15]. We derive all the parameters in the power state machines for the WiFi and LTE interfaces for the Galaxy Note and Samsung Galaxy SIII phones, shown in Table 3.

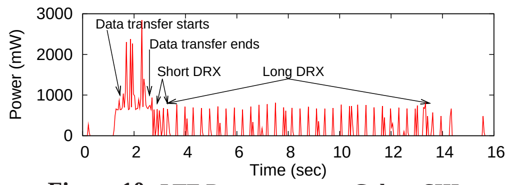


Figure 10: LTE Power states on Galaxy SIII

9. REFERENCES

[1] `https://play.google.com/store/apps/details?id=it.opbyte.superdownload_lite&hl=en`.

[2] `http://www.shoelacewireless.com/pages/videobee`.

[3] `http://multipath-tcp.org/pmwiki.php/Users/Android`.

[4] `http://androidandme.com/2012/08/smartphones-2/qualcomm-explains-the-system-of-tiers-for-snapdragon-s4/`.

[5] `http://www.theverge.com/2013/1/14/3874576/samsung-sells-over-100-million-galaxy-s-devices`.

[6] `http://news.cnet.com/8301-1035_3-57493718-94/samsung-10m-galaxy-notes-sold-in-nine-months/`.

[7] BALASUBRAMANIAN, N., BALASUBRAMANIAN, A., AND VENKATARAMANI, A. Energy consumption in mobile phones: a measurement study and implications for network applications. In *IMC* (2009).

[8] BHULAI, S., ET AL. Optimal concurrent access strategies in mobile communication networks. In *ITC* (2010).

[9] BHULAI, S., ET AL. Dynamic traffic splitting to parallel wireless networks with partial information: A bayesian approach. *Perform. Eval. 69*, 1 (2012), 41–52.

[10] BOSMAN, J. W., HOEKSTRA, G. J., VAN DER MEI, R. D., AND BHULAI, S. A simple index rule for efficient traffic splitting over parallel wireless networks with partial information. *Perform. Eval. 70*, 10 (2013).

[11] CARROLL, A., AND HEISER, G. An analysis of power consumption in a smartphone. In *ATC* (2010).

[12] CHEN, Y.-C., ET AL. A measurement-based study of multipath TCP performance over wireless networks. In *IMC* (2013).

[13] DENG, S., ET AL. All your network are belong to us: A transport framework for mobile network selection. In *HotMobile* (2014).

[14] DENG, S., ET AL. WiFi, LTE or both? measuring mulit-homed wireless internet performance. In *IMC* (2014).

[15] DING, N., ET AL. Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. In *SIGMETRICS* (2013).

[16] GARCIA-SAAVEDRA, A., ET AL. Energy consumption anatomy of 802.11 devices and its implication on modeling and design. In *CoNEXT* (2012).

[17] HABAK, K., ET AL. Bandwidth aggregation techniques in heterogeneous multi-homed devices: A survey. *CoRR* (2013).

[18] HIGGINS, B. D., ET AL. Intentional networking: Opportunistic exploitation of mobile network diversity. In *MobiCom* (2010).

[19] HOU, X., DESHPANDE, P., AND DAS, S. R. Moving bits from 3G to metro-scale WiFi for vehicular network access: an integrated transport layer solution. In *ICNP* (2011).

[20] HUANG, J., ET AL. A close examination of performance and power characteristics of 4G LTE networks. In *MobiSys* (2012).

[21] HUANG, J., ET AL. An in-depth study of LTE: effect of network protocol and application behavior on performance. In *SIGCOMM* (2013).

[22] LIM, Y.-S., ET AL. How green is multipath tcp for mobile devices. In *AllThingsCellular* (2014).

[23] MITTAL, R., KANSAL, A., AND CHANDRA, R. Empowering developers to estimate app energy consumption. In *MobiCom* (2012).

[24] NAM, H., ET AL. Mobile video is inefficient: A traffic analysis. Tech. rep., Department of Computer Science, Columbia University, 2013.

[25] NIRJON, S., ET AL. Multinets: A system for real-time switching between multiple network interfaces on mobile devices. In *TECS* (2013).

[26] PAASCH, C., ET AL. Exploring mobile/wifi handover with multipath tcp. In *CellNet* (2012).

[27] PATHAK, A., ET AL. Fine-grained power modeling for smartphones using system call tracing. In *EuroSys* (2011).

[28] PATHAK, A., HU, Y. C., AND ZHANG, M. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In *EuroSys* (2012).

[29] QIAN, F., ET AL. Profiling resource usage for mobile applications: a cross-layer approach. In *MobiSys* (2011).

[30] RAICIU, C., ET AL. How hard can it be? designing and implementing a deployable multipath tcp. In *NSDI* (2012).

[31] SCHULMAN, A., ET AL. Bartendr: A practical approach to energy-aware cellular data scheduling. In *MobiCom* (2010).

[32] SHYE, A., SCHOLBROK, B., AND GOKHAN, M. Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures. In *MICRO 42* (2009).

[33] TSAO, C.-L., AND SIVAKUMAR, R. On effectively exploiting multiple wireless interfaces in mobile hosts. In *CoNEXT* (2009).

[34] VAN DER MEI, R. D., ET AL. Efficient traffic splitting in parallel tcp-based networks: modeling and experimental validation. In *ITC* (2013).

[35] YAP, K.-K., ET AL. Making use of all the networks around us: A case study in android. In *CellNet* (2012).

[36] YAP, K.-K., ET AL. Scheduling packets over multiple interfaces while respecting user preferences. In *CoNEXT* (2013).

[37] ZHANG, L., ET AL. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *CODES+ISSS* (2010).