When Should We Surf the Mobile Web Using Both WiFi and Cellular?

Bo Han
AT&T Labs – Research
Bedminster, NJ
bohan@research.att.com

Feng Qian
Indiana University
Bloomington, IN
fengqian@indiana.edu

Lusheng Ji
AT&T Labs – Research
Bedminster, NJ
lji@research.att.com

ABSTRACT

In this paper, we investigate when to browse the web using WiFi and cellular simultaneously on mobile devices. Our observation, based on empirical measurements, is that mobile web may not always benefit from multipath, which motivates a cost-benefit analysis. However, it is challenging to analyze the benefits (*i.e.,* improved user experience) and costs (*e.g.,* energy consumption) of web browsing, due to HTTP's resource fetching model. We propose to use server push, a standard feature in HTTP/2, to provide an ideal framework for the cost-benefit analysis. We then design a practical system that reduces resource footprint for mobile web over multipath by providing adaptive multipath support.

CCS Concepts

•**Networks** → **Cross-layer protocols;** *Transport protocols; Application layer protocols; Network measurement;* Mobile networks;

Keywords

Multipath TCP; Mobile Web; HTTP/2; Server Push; Cost-Benefit Analysis

1. INTRODUCTION

Given the increasing popularity of smartphones, the industry and research community have spent numerous efforts on improving the performance and quality of experience (QoE) of web browsing on mobile devices. For example, more than 60% of Alexa top websites have mobile versions [26]; mobile-friendly browsers are in wide use on all major platforms [4]; dedicated commercial proxies have been deployed to reduce mobile data usage [5]; also advanced in-cloud services have been developed for accelerating mobile web [7, 28]. Most mobile devices have multiple network interfaces, such as WiFi and cellular, that can be leveraged together to improve web performance through multipath TCP (MPTCP [15]). MPTCP transparently splits an end-to-end TCP connection onto multiple interfaces. It stripes data onto subflows at the sender and reassembles them from each paths at the receiver. MPTCP has already been implemented in the Linux kernel [24].

AllThingsCellular'16, October 03-07, 2016, New York City, NY, USA

DOI: http://dx.doi.org/10.1145/2980055.2980060

In this paper, we aim to answer an under-explored yet important question: *should we surf the mobile web using MPTCP all the time?* Multipath on mobile devices has been studied in diverse aspects, such as accelerating file download [10, 13] and facilitating mobility [11]. However, to the best of our knowledge, little work has been done to understand its interplay with mobile web. We recently conducted a first measurement of mobile web performance over MPTCP [17]. The key finding is, MPTCP brings more benefits to SPDY than to HTTP/1.1, due to SPDY's long-lived multiplexing connection that uses both paths effectively. However, as we will show in this paper, the performance gain of MPTCP for HTTP/2[1] may diminish, *e.g.,* when the page size is small, when the network characteristics of a path differs significantly from the other, or when network transfer is not the bottleneck. Thus, blindly enabling multipath may increase cellular data usage and energy consumption, while providing no or little QoE improvement to mobile users.

Based on the above insights, we argue that multipath should be used *adaptively*, *i.e.,* only when its offered benefits outweigh its incurred costs. Unfortunately, we found it is fundamentally difficult to perform the cost-benefit analysis, due to HTTP's default "request-response" resource fetching model that iteratively discovers and fetches resources. As a result, at the very beginning of loading a page, neither the client nor the server has the knowledge of all contents to be transferred, let alone performing any analysis based on them.

In order to overcome this limitation, we propose to strategically use HTTP/2's *server push* feature. Server push allows early resource discovery and separates network transfer and local computation, thus making the cost-benefit analysis, a key prerequisite of adaptive multipath, much easier. We sketch a system design that makes mobile web over multipath practical by reducing its resource footprint. We summarize our contributions as follows.

- We conduct experiments in diverse settings to understand the interplay between multipath and HTTP, as well as the importance of adaptive multipath (§3).
- We reveal the key challenge of using multipath adaptively for HTTP's resource fetching scheme (§4).
- To address the above challenge, we propose a novel server-push based framework that performs the cost-benefit analysis to intelligently determine when to use multipath, and present a high-level system design (§4).

2. EXPERIMENTAL SETUP

We set up a multipath testbed consisting of a Linux laptop client and a server machine, which runs a web server and a proxy. The

[1]HTTP/2 was developed based on the SPDY protocol.

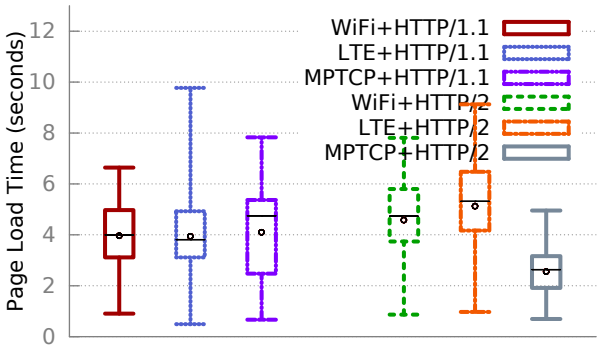

Figure 1: Experiments E1 (left) and E2 (right): PLT of HTTP/1.1 vs HTTP/2 over single path and multipath. The box plots show the min, 25-th percentile, median, 75-th percentile, and the max, across the 24 sites. The open dot is the mean.

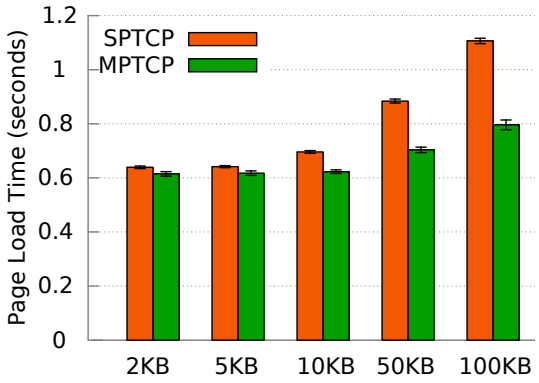

Figure 2: Loading small pages over SPTCP and MPTCP (Experiment E3).

laptop communicates with the server using the WiFi (through an 802.11n AP) and LTE (over a cellular carrier) paths simultaneously. Based on recent measurement studies, we impose additional delay and bandwidth limit on both paths using Dummynet [8], to emulate typical link characteristics experienced by urban residents. For WiFi, the throughput is 7.0Mbps for downlink and 2.0Mbps for uplink, and the RTT is 50ms [29]. For LTE, they are 9.2Mbps (downlink), 2.3Mbps (uplink) and 70ms, respectively [19]. We use this setup by default, unless otherwise mentioned. The client runs an off-the-shelf Chrome browser (version 43). Each page is loaded with cold-cache 100 times for statistically meaningful results. As we set up our own web server, there is also no in-network caching. We conduct four types of page loading experiments E1 to E4:

E1. HTTP/1.1 Replay. The client fetches pages from our replay server (Google Page Replay [1]) that hosts pre-recorded landing pages of 24 popular websites selected from Alexa top 100 websites.

E2. HTTP/2 Replay. This is similar to E1 except the web protocol is HTTP/2. Since Google Page Replay does not yet support HTTP/2, we configure an HTTP/2 proxy (nghttpx 1.0.5 [2]) colocating with the replay server. The proxy thus incurs negligible forwarding overhead.

E3. HTTP/2 Synthetic Page Fetching. The client downloads synthetic pages with different sizes from the web server (nginx 1.4.6 [3]), using the nghttpx proxy[2].

E4. HTTP/2 Server Push. The client loads landing pages of five mirrored websites from the nginx web server. The nghttpx proxy uses the server push feature in HTTP/2 to push all resources.

3. WHY ADAPTIVE MULTIPATH?

We study two representative web protocols, HTTP/1.1 and HTTP/2, over MPTCP. HTTP/1.1 is the *de facto* protocol used by millions of web servers today. As the successor of HTTP/1.1, HTTP/2 [6] was standardized in 2015. It provides new features such as multiplexing and server push.

3.1 Background: HTTP/1.1 vs. HTTP/2

The left side of Figure 1 compares the page load time (PLT) of HTTP/1.1 over single-path TCP (SPTCP) and MPTCP. Since WiFi is usually free (or cheaper than LTE) and incurs less energy footprint on mobile devices, we select it as the primary path to initiate MPTCP connections, and use LTE as the secondary one. The results indicate that MPTCP often does not improve (and may even hurt) the performance of HTTP/1.1. This is attributed to the fact that in HTTP/1.1's paradigm, browsers issue a large number of *short-lived* connections, many of which terminate before the secondary path's handshake is completed. The chance of using the secondary path is thus significantly reduced.

We next repeat the experiments for HTTP/2. As shown in the right side of Figure 1, MPTCP significantly improves the performance of HTTP/2, with the average reductions of PLT ranging from 18.22% to 57.19%, compared to the best single-path PLT among WiFi and cellular. Unlike HTTP/1.1, HTTP/2 supports multiple outstanding requests by multiplexing many objects onto a *single* connection, thus creating sufficient opportunities to utilize both WiFi and cellular paths. In our MPTCP experiments for HTTP/2, the flow duration increases significantly, making all connections use the secondary path.

Take-away 1: *HTTP/1.1 does not work well with MPTCP, due to its short flow duration. MPTCP can significantly improve the performance of HTTP/2 which employs long-lived multiplexing connections.*

3.2 Does MPTCP Always Help HTTP/2?

The results of HTTP/2 over multipath look promising. However, one question remains: *are these PLT reductions always achievable?* In fact, the performance gain of MPTCP diminishes in many scenarios, for example:

- **When the page size is small or a warm-cache loading is performed**: In this case, page fetching may finish right after the secondary subflow is established. Figure 2 shows the results for synthetic pages with sizes from 2KB to 100KB (Experiment E3). As the page size becomes smaller, the PLT difference between SPTCP and MPTCP diminishes. Note that prior study shows the median size of warm-cache loading of a mobile page is around 50KB [26].

- **When the secondary path has a higher latency and lower throughput than the primary**: In this case, MPTCP's default scheduler routes most packets to the primary path. To verify it, we repeat E2 by changing the cellular path's characteristics to emulate a typical 3G link (970 Kbps downlink, 331 Kbps uplink, 160 ms RTT [19]). As shown in Figure 3, MPTCP and SPTCP over WiFi yield similar performance.

- **When network transfer is not the bottleneck**: Due to local computation (*e.g.,* JavaScript evaluation and page rendering) interleaved with network activities, the web page transfer may exhibit

[2]For E3 and E4, we use a proxy because the support of HTTP/2, especially for server push, by nginx was still experimental, as of the time of running the experiments.

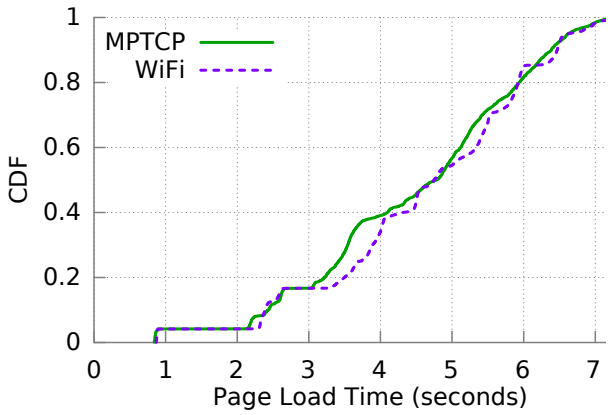

Figure 3: Repeating E2 with the cellular path emulated as a 3G link.

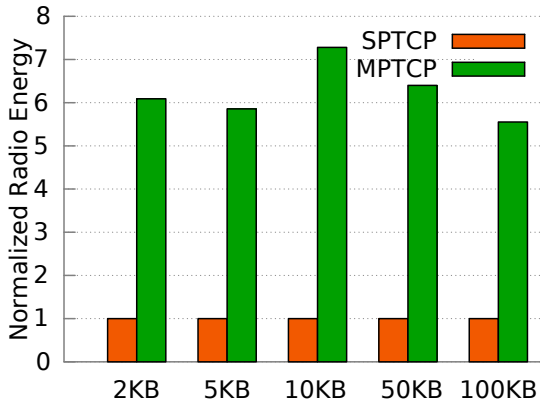

Figure 4: Normalized radio energy consumption for loading small pages (E3).

an intermittent traffic pattern [31]. In this case, CPU becomes bottleneck, making MPTCP potentially not beneficial since a single path may already provide sufficient bandwidth. This can happen more frequently on complex or poorly designed pages.

Take-away 2: *HTTP/2's multiplexing nature makes it outperform HTTP/1.1 when working over MPTCP. However, MPTCP's benefits may diminish due to several factors such as small page size and poor secondary path.*

3.3 Energy Impact of MPTCP

We learn from §3.2 that MPTCP may have limited performance gain for HTTP/2. Therefore, blindly using multipath causes unnecessary cellular data usage and potentially significant energy drain. Cellular interfaces are a major battery consumer of mobile devices: 3G radio accounts for 1/3 to 1/2 of the overall device energy consumption [27], and LTE consumes at least 50% of the overall energy [23]. For the experiments of fetching small pages, we use the recently proposed single- and multipath power models for Samsung Galaxy Note [23] to quantify the radio energy consumption in Figure 4, by conducting trace-driven simulations. Comparing Figure 2 and Figure 4, we can see that MPTCP consumes more energy (5.5x to 7.3x) than SPTCP does, but provides marginal performance improvement, when loading small pages. Note Figure 4 does not include the *tail* energy that can be significant for cellular on some devices. We also conduct a simulation for the experiments of Figure 3, using a 3G UMTS power model [27]. The results indicate that under our configured network conditions (§3.2), using MPTCP over WiFi and 3G increases the radio energy consumption by 7.5x on average while providing negligible PLT boost as shown in Figure 3.

We emphasize that the energy consumption of MPTCP may not always be higher than SPTCP. When MPTCP significantly reduces the PLT, making radios stay up for a much shorter period of time, the overall radio energy consumption may also be reduced. This further motivates our proposal of adaptively performing multipath only when MPTCP brings in performance gains.

Take-away 3: *Compared to SPTCP, MPTCP may incur additional radio energy consumption, which will be wasted if it cannot improve the QoE for mobile web.*

4. TOWARD ADAPTIVE MULTIPATH

The above findings suggest we should use multipath *adaptively* for mobile web: predicting the *costs* (additional cellular data usage and energy consumption) and the *benefits* (improved QoE), and enabling multipath only if the benefits outweigh the costs.

Unfortunately, conducting such a cost-benefit analysis is very difficult, if not impossible. Both the costs and benefits are related to how big the to-be-fetched-objects are, and the order and interleaving of the fetching of these objects. HTTP's default "request-response" resource fetching model, which iteratively discovers and fetches resources, makes it difficult to predict the costs and benefits. As a result, at the beginning of loading a page, *neither the client nor the server has knowledge of all contents to be fetched, let alone making any prediction based on them.* Thus, we believe changes to the page fetching model are needed to allow early resource discovery, and to enable the client or the server to determine the best content delivery strategy (single- or multipath).

4.1 Leveraging Server Push

We propose to employ *server push*, a standard feature in HTTP/2 [6], for the cost-benefit analysis. Server push lets the server proactively send resources to clients. Compared to the conventional paradigm of iteratively fetching resources, it offers three benefits:

- Server push reduces the difficulty of predicting the *costs* of multipath. The ideal usage scenario of server push is, upon the reception of an initial page request, the server gathers most (if not all) resources associated with the page and pushes them in a single bundle to the client. Unlike the conventional iterative resource fetching, network transfer will ideally never be hindered by local computation. Thus, fetching a web page essentially becomes bulk file downloading with a known size. Then we can estimate the costs, *i.e.,* the additional data usage and energy consumption incurred by the secondary path, through either simulations or theoretical modeling (by extending existing TCP models [9, 25] for MPTCP).

- Server push also makes predicting the *benefits* of multipath easy, by separating network transfer and local computation as much as possible. A recent study [7] indicates that after clicking a link, mobile users have a typical tolerance limit of 3 to 5 seconds before most contents of a page are loaded. It suggests that the number of high-utility resources delivered in the first few seconds is a good metric for quantifying the QoE for web browsing[3]. We therefore use $B^T_{\text{MPTCP}} - B^T_{\text{SPTCP}}$ to quantify the QoE improvements brought by MPTCP, where B^T_{MPTCP} and B^T_{SPTCP} are the bytes delivered by MPTCP and SPTCP, respectively, in the first T seconds (a pre-defined threshold, *e.g.,* T=3). B^T_{MPTCP} and B^T_{SPTCP} can also be estimated by simulation or modeling. In contrast, without server

[3]This assumes the resources are prioritized, which can be achieved by profiling resources dependencies using approaches proposed in the literature, either statically [21, 31] or dynamically [7].

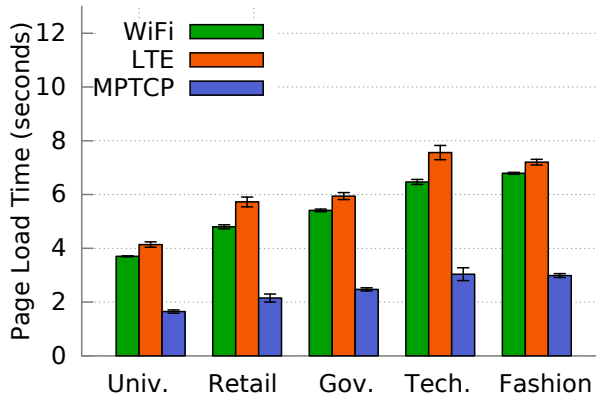

Figure 5: HTTP/2 server push: SPTCP vs. MPTCP (E4).

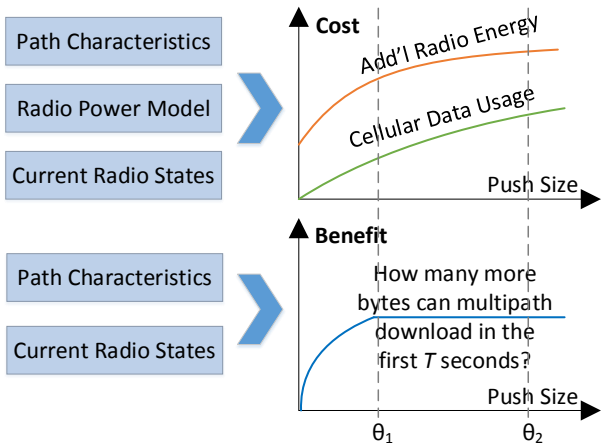

Figure 6: Illustration of the proposed cost-benefit analysis.

push, the local computation and network transfer are tightly coupled, leading to potentially intermittent traffic patterns that make B^T_{SPTCP} and B^T_{MPTCP} much more difficult to estimate.

- Server push also improves cellular energy efficiency, by maximizing bandwidth utilization and minimizing the "radio-on-but-idle" time [12]. Without server push, the potentially intermittent traffic patterns force the cellular radio to stay at the high-power state for a longer period.

Realization of the above advantages assumes the majority of resources can be pushed at the beginning of loading a page. In §5.1 we argue this assumption is promisingly valid for most of today's web pages.

To demonstrate MPTCP can indeed help server push when transfer size is large, we compare the PLT of server push over SPTCP and MPTCP using five mirrored landing pages of popular websites (Experiment E4). For each resource, we add a corresponding `LINK` field to the response header of the initial page request, which makes the resource pushed by nghttpx. Figure 5 clearly indicates that when server push is enabled, the PLT over MPTCP is around only half of that over SPTCP. Note that the performance of server push should be better than non-push due to the reduced network transfer time (e.g., by up to 45% over cellular networks [16]).

Take-away 4: *Server push provides two key features: early resource discovery and separation between network transfer and local computation. They make it possible to perform cost-benefit analysis for multipath.*

4.2 Sketch of System Design

Based on the above analysis, we present a high-level design of using multipath adaptively for mobile web.

1. The client sends the initial page request to the server over the primary WiFi path and describes the cost-benefit policies of multipath. A policy can be, for example, multipath should be used only if at least 200KB data will be transferred. It is derived from the cost-benefit analysis below. We expect the policy to be very concise (*e.g.,* one line in the HTTP header).

2. The server gathers as many resources associated with the page as possible, and then pushes them over either SPTCP or MPTCP, depending on the client's policy.

3. For the remaining resources that will not be pushed by the server, the client can either conservatively fetch them over a single path, or use the same policy as server push. We expect there are none or not many of such resources, and they incur limited impact on user's experience (*e.g.,* they may be advertisements and periodic background pings).

Next, we describe how to conduct the cost-benefit analysis. As shown in Figure 6, before sending a page request, the client derives the *cost* functions of additional cellular data usage and energy consumption (if any, compared to SPTCP over WiFi). Several factors affect the costs: both paths' characteristics (latency, bandwidth, loss, *etc.*), the power models of both interfaces, and their current radio states (turning on the radios costs energy). We can obtain such information from mobile devices (*e.g.,* RRC states [30] and signal strength) and by lightweight passive measurement on the client. The client also derives a *benefit* function $B^T_{\text{MPTCP}} - B^T_{\text{SPTCP}}$, which is the additional bytes MPTCP can deliver in the first T seconds compared to SPTCP, under the current network conditions and radio states (§4.1). With external factors (network conditions, power models, and radio states) known, the only variable in both the cost and benefit functions is the size of pushed resources. We exemplify an empirically derived model in Figure 7, which visualizes the percentage of bytes transferred over LTE as a function of WiFi and LTE latencies, for downloading a 1MB file. The baseline is our multipath testbed configuration (§2) without any additional latency. X and Y axes are the additional latency added to each path. Except for few outliers, we see clearly predictable patterns: more bytes traverse the LTE path as the RTT ratio between WiFi and LTE becomes larger. Developing detailed methodologies is part of our ongoing work.

Subsequently, based on the functions and user-specified high-level requirements, the client generates a compact cost-benefit policy and embeds it into the initial page request. The simplest policy consists of a threshold θ_1, dictating the server to use multipath only when the push size is larger than θ_1. If the client concerns excessive energy and/or monetary cost, it may send another threshold θ_2, asking the server to limit the cellular data usage. The values of θ_1 and θ_2 depend on not only the characteristics of both paths, but also user's preference. For example, if the phone is charging and the user has sufficient quota for her cellular data, a small θ_1 and a large θ_2 can be applied. The server will then simply follow the policy and use multipath accordingly.

We summarize the adaptive nature of our scheme by revisiting the three scenarios in §3.2, where MPTCP may not be able to improve the performance of mobile web. The small page size and the poor quality of the secondary path are taken into account by θ_1 and θ_2, which are dynamically determined based on network conditions. The third one (interleaved computation and data transfer) is addressed by server push that decouples both factors.

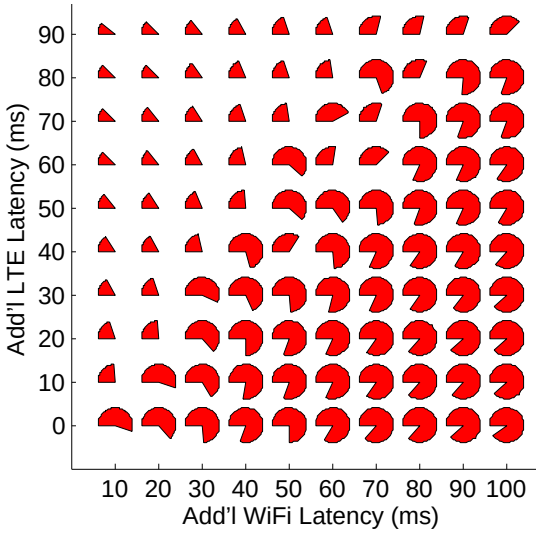

Figure 7: Example of an empirical model based on WiFi/LTE latencies.

5. DISCUSSION

We discuss several practical issues in this section.

5.1 Determining What to Push

Determining what resources to push is a non-trivial task. Pushing unwanted data incurs bandwidth overhead, while pushing less data than needed reduces the performance gain. This problem is not specific to multipath, since server push is a general optimization approach.

We did not find any literature that provides detailed methods of how to select the resources to push. We thus give some guidelines here. First, usually a majority of resources have their URLs embedded in the HTML page (the page itself can be either statically or dynamically generated). Based on our measurements of popular sites' pages, the fraction of bytes of statically embedded resources ranges from less than 50% to 100%. Second, we can generate the set of resources requested dynamically by JavaScript *without per-user customization* using a simple black-box approach, such as having several bots load the same page and taking an intersection of what they get. Third, the remaining resources are dynamically generated for a specific client, such as advertisements generated from a cookie. Usually large objects (*e.g.,* images and fonts) are not likely to be customized, and we thus expect pushing the resources described above can already achieve most of the performance gain for many sites. Finally, the server can obtain the set of resources already cached by the clients using existing cache synchronization mechanisms, such as MetaPush [16] and Cache Hints [20]. These resources will not be pushed. Thus, we believe for websites without substantial per-user customization, most of their resources can be accurately identified and pushed.

5.2 Working with MPTCP

Our system can use the existing MPTCP protocol implementation [24] with minor changes.

Secondary Subflow Establishment. In the default MPTCP, shortly after establishing the primary subflow, the client will start the secondary subflow. In our case, the client needs to hold off, and wait for the server to determine whether to establish the secondary subflow. However, this generally useful mechanism, which also allows the server to manage subflows, is missing in today's MPTCP. It can be realized by slightly modifying MPTCP as follows. (1) The client uses a socket option to inform its MPTCP stack to hold off establishing the secondary subflow. (2) When establishing the primary subflow, the client informs the server about its cellular interface's address ID through `ADD_ADDR`. (3) When the server decides to establish the secondary subflow, it sends `MP_JOIN+`, a new message similar to `MP_JOIN` but including the client's cellular address ID, over the *primary* subflow. Note that directly sending a downlink `MP_JOIN` to the client's cellular interface may not work due to NAT and firewalls that prevail in today's cellular networks. Subsequent secondary subflow handshake remains unchanged.

Congestion Control (CC). When multipath is used in datacenter networks, due to fairness considerations, subflows' CC algorithms are coupled [32]. In mobile world, since WiFi and cellular are heterogeneous access networks, their CC can be decoupled (*e.g.,* each independently runs standard TCP CUBIC) to maximize both paths' bandwidth utilization [18]. MPTCP also supports decoupled CC.

5.3 Deployment using a MPTCP Proxy

Since most of today's web servers do not support MPTCP or use server push, one way to quickly deploy our scheme is to use a proxy. In this configuration, MPTCP and HTTP/2 server push are used between the proxy and the client, and regular HTTP/1.1 or non-push HTTP/2 over a single path is used between the proxy and remote server. Upon the reception of a page request, the proxy iteratively fetches contents (including parsing JavaScript) on behalf of the client (which only issues the initial request). Similar approaches already exist in several cloud-based browsing solutions [4, 28]. The proxy further needs to measure the overall page size, based on which it decides whether to use multipath or not, and then pushes the resources to the client accordingly. It may take time for the proxy to fetch and parse the resources before pushing them out in a bundle, leading to additional wait time on the client. We expect these proxies to be computationally powerful, and have low-latency links to web servers, making the additional delay small. The issue can be further mitigated by proxy-side caching and other optimizations.

5.4 Other Issues

System Overhead. In our design, to make the scheme scalable, all measurements and the cost-benefit analysis are performed on the client side. One potential concern is its incurred overhead. We expect it to be low due to several reasons: a client needs to perform only passive and lightweight measurements; the multipath policy is very simple; and we can generate the models offline and reuse them. We plan to thoroughly evaluate the client-side overhead in our future work.

Policy and primary interface. We have not explicitly considered the data usage of LTE services, which can be incorporated into the policies for determining θ_1 and θ_2 as described in §4.2. Also, in this study we always use WiFi as the primary interface. Changing it to cellular may improve the file download time [13]. We leave the evaluation of its impact on web performance in our future work.

6. RELATED WORK

Mobile Web. Mobile web performance has been extensively investigated. For instance, Erman *et al.* [14] measured the performance of HTTP and SPDY over cellular networks and studied their interactions with TCP. PARCEL [28] reduces energy consumption and page load time for mobile devices through "bundled push". Klotski [7] improves the quality of user experience by dynamically reprioritizing mobile web resources. Flywheel [5] is Google's data

compression HTTP proxy for the mobile web, which has been a production system for years. However, none of them studies how mobile web should perform over multipath.

MPTCP in Mobile Networks. Mobile devices can benefit from MPTCP for various applications. For file downloading, Chen *et al.* [10] and Deng *et al.* [13] evaluated its performance when using SPTCP and MPTCP over WiFi and cellular networks. The mobile kibbutz system [22] consolidates cellular traffic from multiple users onto fewer links through MPTCP to improve energy efficiency. Leveraging MPTCP, Croitoru *et al.* [11] studied seamless WiFi mobility among multiple APs. In our recent work [17], we evaluated the performance of HTTP/1.1 and SPDY over MPTCP. Partially motivated by the findings, this paper investigates the question of when we should use MPTCP for mobile web.

7. CONCLUDING REMARKS

To the best of our knowledge, this is the first proposal of adaptively using MPTCP for mobile web. We have found mobile web does not always benefit from MPTCP, and we argue a cost-benefit analysis is needed to reduce the resource footprint of MPTCP. Our proposed cost-benefit analysis is enabled by HTTP/2's server push feature. We are implementing this adaptive MPTCP for mobile web and evaluating its performance. We believe the same lesson also applies to other mobile applications, such as video streaming.

Acknowledgements

We thank the anonymous reviewers for their helpful feedback. We thank Vijay Gopalakrishnan for his valuable comments and suggestions. Feng Qian's research was supported in part by NSF under CNS-1566331.

8. REFERENCES

[1] Google Web Page Replay Tool. https://github.com/chromium/web-page-replay.

[2] HTTP/2 C Library and Tools. https://nghttp2.org/.

[3] Nginx: a web and reverse proxy server. http://nginx.org/.

[4] Opera Mini Browser. http://www.opera.com/mobile/mini.

[5] V. Agababov, M. Buettner, V. Chudnovsky, M. Cogan, B. Greenstein, S. McDaniel, M. Piatek, C. Scott, M. Welsh, and B. Yin. Flywheel: Google's Data Compression Proxy for the Mobile Web. In *NSDI*, 2015.

[6] M. Belshe, R. Peon, and M. Thomson. Hypertext Transfer Protocol Version 2 (HTTP/2). RFC 7540, 2015.

[7] M. Butkiewicz, D. Wang, Z. Wu, H. V. Madhyastha, and V. Sekar. Klotski: Reprioritizing Web Content to Improve User Experience on Mobile Devices. In *NSDI*, 2015.

[8] M. Carbone and L. Rizzo. Dummynet Revisited. *ACM SIGCOMM Computer Communication Review*, 40(2):12–20, Apr. 2010.

[9] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP Latency. In *INFOCOM*, 2000.

[10] Y.-C. Chen, Y.-S. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley. A Measurement-based Study of MultiPath TCP Performance over Wireless Networks. In *IMC*, 2013.

[11] A. Croitoru, D. Niculescu, and C. Raiciu. Towards WiFi Mobility without Fast Handover. In *NSDI*, 2015.

[12] S. Deng and H. Balakrishnan. Traffic-Aware Techniques to Reduce 3G/LTE Wireless Energy Consumption. In *CoNEXT*, 2012.

[13] S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan. WiFi, LTE, or Both? Measuring Multi-homed Wireless Internet Performance. In *IMC*, 2014.

[14] J. Erman, V. Gopalakrishnan, R. Jana, and K. Ramakrishnan. Towards a SPDY'ier Mobile Web? In *CoNEXT*, 2013.

[15] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824, 2013.

[16] B. Han, S. Hao, and F. Qian. MetaPush: Cellular-Friendly Server Push For HTTP/2. In *All Things Cellular Workshop*, 2015.

[17] B. Han, F. Qian, S. Hao, and L. Ji. An Anatomy of Mobile Web Performance over Multipath TCP (Short Paper). In *CoNEXT*, 2015.

[18] H.-Y. Hsieh and R. Sivakumar. A Transport Layer Approach for Achieving Aggregate Bandwidths on Multi-homed Mobile Hosts. In *MobiCom*, 2002.

[19] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. In *SIGCOMM*, 2013.

[20] J. Khalid, S. Agarwal, A. Akella, and J. Padhye. Improving the performance of SPDY for mobile devices. In *HotMobile (Poster Session)*, 2015.

[21] Z. Li, M. Zhang, Z. Zhu, Y. Chen, A. Greenberg, and Y.-M. Wang. WebProphet: Automating Performance Prediction for Web Services. In *NSDI*, 2010.

[22] C. Nicutar, D. Niculescu, and C. Raiciu. Using Cooperation for Low Power Low Latency Cellular Connectivity. In *CoNEXT*, 2014.

[23] A. Nika, Y. Zhu, N. Ding, A. Jindal, Y. C. Hu, X. Zhou, B. Y. Zhao, and H. Zheng. Energy and Performance of Smartphone Radio Bundling in Outdoor Environments. In *WWW*, 2015.

[24] C. Paasch, S. Barré, et al. Multipath TCP in the Linux Kernel. http://www.multipath-tcp.org.

[25] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *SIGCOMM*, 1998.

[26] F. Qian, S. Sen, and O. Spatscheck. Characterizing Resource Usage for Mobile Web Browsing. In *MobiSys*, 2014.

[27] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Profiling Resource Usage for Mobile Applications: a Cross-layer Approach. In *Mobisys*, 2011.

[28] A. Sivakumar, S. P. Narayanan, V. Gopalakrishnan, S. Lee, S. Rao, and S. Sen. PARCEL: Proxy Assisted BRowsing in Cellular networks for Energy and Latency reduction. In *CoNEXT*, 2014.

[29] J. Sommers and P. Barford. Cell vs. WiFi: On the Performance of Metro Area Mobile Connections. In *IMC*, 2012.

[30] N. Vallina-Rodriguez, A. Aucinas, M. Almeida, Y. Grunenberger, K. Papagiannaki, and J. Crowcroft. RILAnalyzer: a Comprehensive 3G Monitor On Your Phone. In *IMC*, 2013.

[31] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall. Demystifying Page Load Performance with WProf. In *NSDI*, 2013.

[32] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *NSDI*, 2011.