Demo: MPBond – Efficient Network-level Collaboration Among Personal Mobile Devices

Xiao Zhu
University of Michigan
shawnzhu@umich.edu

Jiachen Sun
University of Michigan
jiachens@umich.edu

Xumiao Zhang
University of Michigan
xumiao@umich.edu

Yihua Ethan Guo
Uber Technologies, Inc.
yhguo@umich.edu

Feng Qian
University of Minnesota
fengqian@umn.edu

Z. Morley Mao
University of Michigan
zmao@umich.edu

ABSTRACT

We demo MPBond, a novel multipath transport system allowing multiple personal mobile devices to collaboratively fetch content from the Internet. Inspired by the success of MPTCP, MPBond applies the concept of *distributed multipath transport* where multiple subflows can traverse different devices. Other key design aspects of MPBond include a device/connection management scheme, a buffering strategy, a packet scheduling algorithm, and a policy framework tailored to MPBond's architecture. We install MPBond on commodity mobile devices and show how easy it is to configure the usage of MPBond for unmodified apps. We visualize the runtime behavior of MPBond to further illustrate its design. We also demonstrate the download time and energy reduction of file download, as well as the video streaming QoE improvement with MPBond.

CCS CONCEPTS

• **Networks** → **Network protocol design**; **Transport protocols**; • **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**.

ACM Reference Format:
Xiao Zhu, Jiachen Sun, Xumiao Zhang, Yihua Ethan Guo, Feng Qian, and Z. Morley Mao. 2020. Demo: MPBond – Efficient Network-level Collaboration Among Personal Mobile Devices. In *The 18th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '20), June 15–19, 2020, Toronto, ON, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3386901.3396600

1 INTRODUCTION

It is increasingly common that a user possesses multiple mobile devices [1, 2]. For example, despite being a full-fledged computer, a smartwatch naturally needs to pair with a smartphone; business people oftentimes carry two phones, one for work and the other for personal tasks; tablets bear large screens and reasonable portability, making them good companions of smartphones.

From the networking perspective, smart mobile devices are equipped with diverse network interfaces such as cellular, WiFi, and Bluetooth (BT), making them capable of communicating with remote Internet servers as well as other local devices. We make a key observation that despite such a mature wireless *hardware* support, the potential of the devices' network interfaces that can operate collaboratively is far from being fully exploited. In this paper, we bridge this critical gap by bringing networking *software* innovations to the smart mobile device ecosystem. Specifically, we develop MPBond, a multipath transport system allowing multiple personal mobile devices to collaboratively fetch content from the Internet.

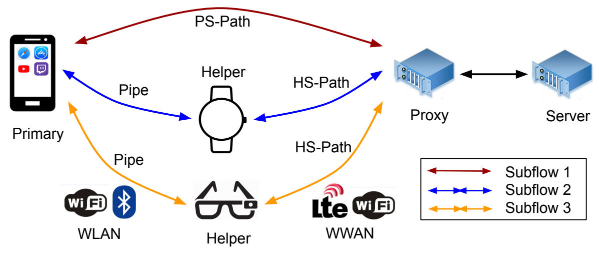

Figure 1: System Architecture of MPBond.

Unlike traditional multipath transport paradigms [5, 9, 10], MPBond needs to support *distributed* multipath where subflows traverse different mobile devices. Specifically, MPBond involves one *primary* device, where the client app runs, and multiple *helper* devices, which boost the primary's network performance. The above scheme provided by MPBond appears to be intuitive. However, we face numerous challenges when designing and implementing the system. We next highlight our key design aspects in §2. Note that our full paper with all technical details can be found at [11].

2 DESIGN OVERVIEW

As shown in Figure 1, MPBond involves two types of devices: one *primary* device and one or more *helper* devices. The client application, such as a file downloader or a video player, only runs on the primary. The TCP traffic from the app are transparently intercepted by the MPBond service and scheduled to transmit either over primary's own interface or through helpers with forwarding, i.e., different *subflows* shown in Figure 1. The reverse direction works similarly. To be fully transparent to Internet servers, the system can introduce an MPBond-capable proxy which hides MPBond from remote servers by establishing single-path connections with them. Also, deployed as an OS service on the primary and helpers, MPBond is transparent to client-side apps as well.

2.1 Subflow Management

We call the data channels between the primary and helpers *pipes*. We also denote the network paths between helpers and the server *HS-Paths* (Helper-server Paths), and the network path between the primary and the server (without a helper) the *PS-Path* (Primary-server Path). An end-to-end subflow therefore traverses through either a PS-Path, or an HS-Path and a pipe. The pipe is established by connecting the helper through WLAN to the primary which acts as a WiFi AP, or pairing the helper to the primary through Bluetooth. Similar to MPTCP's subflows, the pipes can be flexibly torn down or established, and they are loosely coupled with a user TCP connection, allowing seamless migration among pipes without interrupting the data transfer.

2.2 Buffer Management and Helper-side Connection Split

MPBond maintains buffers at both end points (the primary and the server) to absorb the network jitters and to accommodate the subflows' heterogeneous characteristics. Besides having these buffers, we make an important design decision of setting up buffers on helpers. Specifically, MPBond splits any subflow into two TCP (sub)flows, one between the primary and the helper, and the other between the helper and the server. The two flows thus cover the pipe and the HS-Path, respectively. When a helper is involved, the WWAN and WLAN usually exhibit vastly different link characteristics. TCP splitting can effectively improve the performance in such a scenario by shortening the TCP control loop. More importantly, it allows buffers to be set up between the two flows. Such buffers effectively mitigate the negative performance impact caused by the bottleneck shift on a subflow.

2.3 Pipe-aware Multipath Scheduler

As a critical component of a multipath transport system, a scheduler determines how to distribute the traffic onto multiple paths. There are several studies that improve the scheduler design in wireless settings for traditional multipath transport paradigms [6, 7]. However, directly applying them to MPBond is difficult. First, most existing mobile multipath schedulers only deal with two paths (WiFi and cellular), and many solutions such as [6] are inherently difficult to scale to more than two paths, which MPBond needs to handle. Second, none of prior studies considers the pipes, which are unique in MPBond.

We design a Pipe-aware Multipath Scheduler (PAMS) for MPBond. It differs from existing mobile multipath schedulers in two aspects: PAMS is capable of scheduling an arbitrary number of subflows, and it takes MPBond's TCP splitting and helper-side buffering mechanism into consideration when performing scheduling.

2.4 User/App Interfaces and Policy Engine

MPBond has a built-in console on the primary. The console allows users to pair/unpair with helpers, manage the pipes, grant apps permission to use MPBond, monitor the devices' runtime status, and configure the following various policies: (1) *per-app whitelist.* users need to explicitly grant permissions to apps and specify a (super)set of devices/pipes that the app can access through MPBond. Typically this is a one-time effort, provides good flexibility, and boosts security. (2) *Resource Usage.* MPBond allows disabling a device (either helper or primary) when its battery level drops below a threshold or its monthly cellular data usage reaches a pre-defined cap. (3) *Prioritization.* Users can configure rules that prioritize certain applications.

3 DEMONSTRATION

Our demonstration is composed of two Android phones, one Wear OS smartwatch, and one Linux laptop. The phones and smartwatch require LTE cellular access. The laptop requires access to a WiFi network and power, which visualizes the behaviors of MPBond's key design aspects during the demo and reports the performance metrics of each run. Together they demonstrate the following:

Ease of use. We showcase how MPBond can be easily used by unmodified applications on the primary phone. We provide a simple-to-use interface for conference attendees to configure and experience custom polices described in §2.4.

Visualization of system design. Our laptop draws side-by-side diagrams showing the runtime behaviors of MPBond and existing collaboration systems [4, 8] including the dynamics of the buffer level on helpers, the real-time scheduling decision, and the network utilization of each path, to better illustrate the design of different schemes and understand MPBond's performance and energy benefits brought by the buffering on helpers (§2.2) and the judiciously designed PAMS scheduler (§2.3).

Bulk transfer performance improvement. This shows the improvement of download time and energy consumption for bulk data transfers with MPBond. We show the benefits brought by the increased number of devices. We also compare the performance of MPBond with existing schemes [4, 8] under the same workload and network condition.

Video streaming QoE benefits. We finally show how MPBond favors video streaming, one type of mobile apps that dominates the mobile network traffic. We use Exoplayer [3] to stream different adaptive bitrate (ABR) videos hosted on a commodity server, to study the impact of different schemes on video bitrate and energy efficiency. We demonstrate that MPBond improves these QoE metrics.

REFERENCES

[1] 2014. People for Whom One Cellphone Isn't Enough. https://www.wsj.com/articles/people-who-use-two-cellphones-1396393393.
[2] 2017. 8 Frugal Reasons to Have Two Phones. https://www.thefrugalgene.com/frugal-phones/.
[3] 2019. Exoplayer. https://google.github.io/ExoPlayer.
[4] Ganesh Ananthanarayanan, Venkata N Padmanabhan, Lenin Ravindranath, and Chandramohan A Thekkath. 2007. Combine: leveraging the power of wireless peers through collaborative downloading. In *MobiSys*. ACM.
[5] Quentin De Coninck and Olivier Bonaventure. 2017. Multipath quic: Design and evaluation. In *CoNEXT*. ACM.
[6] Yihua Ethan Guo, Ashkan Nikravesh, Z Morley Mao, Feng Qian, and Subhabrata Sen. 2017. Accelerating multipath transport through balanced subflow completion. In *MobiCom*. ACM.
[7] Yeon-sup Lim, Erich M Nahum, Don Towsley, and Richard J Gibbens. 2017. Ecf: An mptcp path scheduler to manage heterogeneous paths. In *CoNEXT*. ACM.
[8] Cătălin Nicutar, Dragoş Niculescu, and Costin Raiciu. 2014. Using cooperation for low power low latency cellular connectivity. In *CoNEXT*. ACM, 337–348.
[9] Ashkan Nikravesh, Yihua Guo, Xiao Zhu, Feng Qian, and Z Morley Mao. 2019. MP-H2: A Client-only Multipath Solution for HTTP/2. In *MobiCom*. ACM.
[10] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. 2012. How hard can it be? designing and implementing a deployable multipath TCP. In *NSDI*. USENIX.
[11] Xiao Zhu, Jiachen Sun, Xumiao Zhang, Yihua Ethan Guo, Feng Qian, and Z. Morley Mao. 2020. MPBond: Efficient Network-level Collaboration Among Personal Mobile Devices. In *MobiSys*. ACM.