Experience: A Three-Year Retrospective of Large-scale Multipath Deployment for Mobile Applications

Chengke Wang[ℙ], Hao Wang[ℍ], Feng Qian[𝕄], Kai Zheng[ℍ], Chenglu Wang[ℍ]
Fangzhu Mao[ℍ], Xingming Guo[ℍ], Chenren Xu[ℙℤ𝕂✉*]

[ℙ]Peking University [ℍ]Huawei [𝕄]University of Minnesota – Twin Cities [ℤ]Zhongguancun Laboratory
[𝕂]Key Laboratory of High Confidence Software Technologies, Ministry of Education (PKU)

ABSTRACT

Multipath transport allows the simultaneous use of diverse paths on mobile devices to maximize mobile resource usage. Over the years, we have witnessed several mobile multipath deployment examples by network operators and mobile app providers. However, existing deployment methods require modifications to either the network infrastructure or both endpoints. To lower the bar of the deployment, we present Fleety, a mobile system service that provides the multipath transport capability with client-only modification. To the best of our knowledge, we are the first to carry out a large-scale mobile multipath deployment that can support hundreds of mobile applications in the cross-ISP setting. This paper is a retrospective of our experience in building and deploying multipath transport for mobile applications. We reveal several practical deployment challenges and share our experience in dealing with them.

CCS CONCEPTS

• **Networks** → **Mobile networks**; **Network protocols**.

KEYWORDS

MPTCP, Multipath HTTP, Nationwide Deployment

ACM Reference Format:
Chengke Wang, Hao Wang, Feng Qian, Kai Zheng, Chenglu Wang, Fangzhu Mao, Xingming Guo, Chenren Xu. 2023. Experience: A Three-Year Retrospective of Large-scale Multipath Deployment for Mobile Applications. In *The 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23), October 2–6, 2023, Madrid, Spain.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3570361.3592506

*✉: chenren@pku.edu.cn.

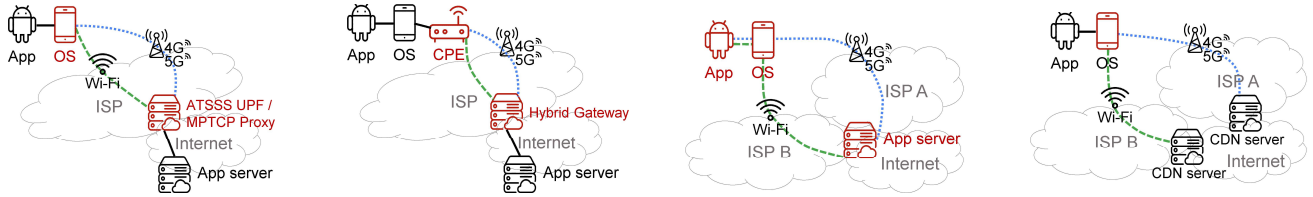
ACM MobiCom '23, October 2–6, 2023, Madrid, Spain

ACM ISBN 978-1-4503-9990-6/23/10…$15.00
https://doi.org/10.1145/3570361.3592506

1 INTRODUCTION

As today's smartphones are equipped with multiple connectivity technologies such as Wi-Fi and cellular, mobile multipath transport can automatically switch to the better quality network path to provide reliability, or simultaneously use both paths to improve overall throughput [1, 2]. The most common use case among consumers is to concurrently use cellular data when Wi-Fi exhibits poor performance, or to opportunistically leverage surrounding Wi-Fi to reduce data usage when the user is primarily using cellular [3–5].

In order to improve the consumer experience, the industry has made significant efforts to deploy mobile multipath transport since 2014. These deployments fall into two categories: relay-based solutions and end-to-end solutions, depending on where the multiple subflows are created and rendezvous. In the *relay-based solution*, illustrated in Fig. 1a and Fig. 1b, a modified mobile OS or customer premises equipment (CPE) creates subflows over multiple network paths, which are aggregated by the relay server and relayed to a legacy plain TCP connection. This solution requires modifying the client and deploying a multipath relay inside the access network, and is typically deployed by the Internet service provider (ISP). Examples of relay-based solutions include GiGA's multipath TCP (MPTCP) [4], hybrid access network [6], and access traffic steering, switching, and splitting (ATSSS) [7]. On the other hand, the *end-to-end solution* involves adopting multipath transport at both endpoints, as shown in Fig. 1c. The multipath transport protocol needs to be customized and integrated into each application and its corresponding servers. As modifications to the client and the server are necessary, these solutions are typically employed by the app service providers. Examples include iOS's MPTCP [5] and Alibaba's multipath QUIC (MPQUIC) [3]. Despite the availability of various solutions, there are only a limited number of news articles that report on the deployment experience [3, 4, 7–11], and even fewer that report on the cost and difficulties associated with large-scale commercial deployment.



(a) The phone vendor and the ISP cooperate in deploying the multipath proxy.

(b) The network operator deploys CPEs and gateways to avoid changes to end-user devices and the access network.

(c) The app service provider makes changes to both the app and the server.

(d) The phone vendor makes changes to the OS and deploys client-only solutions.

Figure 1: Different solutions to mobile multipath deployment. The red components are the ones that require modifications for the deployment. The blue and green dotted lines are two subflows over different network paths.

To fill this gap, we collaborated with a major phone vendor to understand the commercial deployment cost of mobile multipath transport. The vendor's primary goal is to enable cross-ISP multipath transport for top mobile applications in China; this goal is also aligned with end-user requirements. We did not adopt the relay-based solutions due to the expense and impracticality of upgrading the access network infrastructure to support cross-ISP access, especially with three primary ISPs in China. Instead, we opted for the end-to-end solution, specifically MPTCP [12], to conduct a feasibility study. MPTCP is the state-of-the-art multipath transport solution attractive for its application transparency and path robustness. Previous research and deployment have shown MPTCP to have superior performance. We also collaborated with a major online video platform that holds 31% of the online video market share in China [13, 14] to enable multipath transport for video traffic, which accounts for the majority of global mobile network traffic (71% in 2022 [15]). In December 2018, we launched the deployment in Beijing, Shanghai, and Shenzhen, intending to enhance video streaming quality by providing aggregated throughput and robust path handover.

Our MPTCP deployment has undergone three phases, and we found that it has turned out to be much more difficult than expected, primarily due to business considerations rather than technical difficulties. *In the development phase*, we attempted to address several well-known issues with the vanilla MPTCP by patching the MPTCP path scheduler inside the kernel. However, the CDN company that provides the MPTCP servers rejected out-of-tree kernel patches due to security and reliability concerns. *In the deployment phase*, we were only able to deploy MPTCP for video traffic but not for other traffic, such as video manifest files, comments, and ads. This is because different stakeholders manage these types of traffic, and it is hard to convince the entire community that MPTCP meets everyone's interests. We afterward discovered that only accelerating video traffic leads to a limited improvement in user experience because other types of traffic could stall the video loading process. *In the operation phase*, MPTCP incurred a nearly three-fold bandwidth budget cost because the server needed to join multiple ISPs. We failed to find a clear business model to make a profit. Additionally, we discovered that some sub-tier ISPs redirected the connection to their local cache servers by performing DNS poisoning, preventing users from connecting to the MPTCP servers.

After a careful examination of our experience with MPTCP, we found that the primary difficulty is that the end-to-end solution needs to coordinate multiple network parties to collaboratively support MPTCP, which is challenging because different parties have diverse business incentives. For instance, the advertising department of a company prioritizes the conversion rate[1] over the network quality; ISPs aim to reduce their networking traffic cost rather than to support multipath transport; although app providers can adopt standardized technology and protocols, they are slow to adopt custom optimizations. This situation motivates us to explore a new direction for mobile multipath transport deployment. As we collaborate with the phone vendor, we seek a solution that only requires changes to the client and is transparent to other network parties, including app service providers, ISPs, and CDN operators.

We have observed that HTTP is the primary protocol used for mobile application traffic on the network we studied in China. HTTP is highly compatible with the existing infrastructure, such as CDNs and web caches [16]. Moreover, the three most popular Chinese streaming platforms, namely iQiyi, Tencent Video, and Youku, which collectively hold 79% of market shares [17], currently use unencrypted HTTP to stream videos for mobile devices. This unencrypted traffic allows us to perform multipath actions at the HTTP level. Therefore, we have shifted our approach from MPTCP to multipath HTTP (MPHTTP) [18–20], which introduces

[1]Conversion rate refers to the percentage of people who click on an ad and then take a desired action, such as making a purchase.

Solution	App	Mobile OS	CPE	Gateway	Server
∗ GiGA [4]	•		•		•
∗ Hybrid [6]	•	•			•
∗ ATSSS [7]	•		•		•
∘ Apple [5]			•	•	
∘ XLINK [3]		•	•	•	
Fleety (Ours)	•		•	•	•

Table 1: Comparison of Fleety and related solutions. "•" means the solution does not require modification to the existing equipment. "∗ / ∘" denotes relay-based and end-to-end solutions respectively.

multipath functionality on top of HTTP to simplify the deployment process across various stakeholders. MPHTTP uses HTTP byte-range-request, a standard HTTP feature [21], to fetch data simultaneously from multiple paths. As the MPHTTP subflows are decoupled as regular TCP flows, MPHTTP is fully compatible with today's network infrastructure and inherently works in the cross-ISP setting, as illustrated in Fig. 1d.

In this work, we present a mobile system service named Fleety that transparently enables multipath transport for all other Internet parties. We summarize the necessary modifications for Fleety and other mobile multipath solutions in Tab. 1. Additionally, we consider other factors that are missed in previous work on MPHTTP [18–20], including path prioritization, buffer overhead, and consistency verification. Fleety comprises four building blocks that reside entirely on the client side to facilitate multipath transport. First, Fleety includes a flow classifier to identify HTTP flows from the application traffic for further processing. While MPHTTP is compatible with any plaintext HTTP traffic, the classifier selects requests for medium-to-large files only due to performance considerations. Second, there is an MPHTTP proxy that transparently processes the HTTP request forwarded by the flow classifier. The proxy transparently splits one request into multiple requests with smaller byte ranges, assigns the requests to different network paths, reassembles the responses, and replies to the application. Third, Fleety incorporates a lightweight consistency verifier that ensures *all* data fetched from different paths corresponds to the original content that the URL refers to. This is necessary because network middleboxes may inspect and modify the HTTP traffic. The verifier detects data inconsistencies across paths by sampling a small byte range as a "fingerprint". Finally, we implement a path selector that provides link reliability for flows that are not covered by the MPHTTP proxy. The Quality of Experience (QoE) for an application can be affected by both non-HTTP and HTTPS traffic. For such traffic, although it is difficult to achieve client-only multipath, we therefore perform path selection to boost their performance.

Applying multipath to mobile devices incurs additional energy consumption and cellular data charges. Fleety is designed to be cost-aware, a feature not covered by previous work [18–20]. Specifically, we prioritize the Wi-Fi path over the cellular path. When the Wi-Fi path quality suffices to meet the application QoS requirement, Fleety behaves the same as using a single path, except that it continuously monitors the path quality and takes action when the Wi-Fi becomes worse. Our in-lab evaluation shows that the energy overhead of such a setting is 6.43% and 1.40% for web browsing and real-time gaming, respectively. Considering that Fleety provides a better network quality, the energy overhead is practically acceptable.

We deployed Fleety in September 2019. Nowadays, Fleety has supported 142 device models (including smartphones and tablets) and 156 popular applications (such as social media, video, gaming, news, and cloud disk) in China. To evaluate Fleety, we collected data from opt-in user devices. As of January 2022, at least 9.96 million users have opted-in and used multipath transport for one or more applications. Fleety provides additional throughput of 4.43 Mbps on average when users use a slow-speed single-path network. We also identified several factors that influence user acceptance of mobile multipath capability. These factors include the application category, device model, and Wi-Fi quality. Understanding these factors helps us tailor multipath deployment to meet user preferences and demands better.

Contributions.

- We have successfully deployed MPTCP on a major online video platform. We share our experiences and lessons learned from the deployment.
- We present a mobile system service called Fleety for multipath transport. To the best of our knowledge, we are the first to present an MPHTTP design that not only works transparently to the application, the server, and the middlebox but also takes into account both path overhead and data consistency.
- To the best of our knowledge, we are the first to deploy a large-scale mobile multipath solution for hundreds of mobile applications across multiple ISPs, demonstrating the feasibility and ease of deployment of multipath transport with minimal involvement from other network parties.

Ethical Concerns. All analysis and experimental tasks in this study comply with the agreement established between the users and us. The users who participated in the study opted-in with informed consent, the analysis was conducted under a well-established IRB, and no personally identifiable information (*e.g.,* phone number, IMEI, and IMSI) was collected. We never (and have no way to) link the collected information to the users' true identities.

2 BACKGROUND

The current commercial multipath deployment approaches can be classified into relay-based solutions and end-to-end solutions. Network traffic is forwarded through a relay in relay-based solutions, while in end-to-end solutions, subflows are created directly between endpoints.

Relay-based Solution. The principle of relay-based solutions is to deploy relays inside the network to reduce endpoint modifications. The relay translates between the multipath protocol and the legacy TCP/UDP flow. These solutions also require changes on the client side to create connections on multiple network paths, as shown in Fig. 1a. For example, Korea's KT network released the GiGA service in 2015 [4, 22, 23]. They collaborated with the phone vendor Samsung to enable MPTCP on mobile devices. They deployed MPTCP proxies within KT's access network, reaching a throughput of about 800 Mbps out of a theoretical maximum of 1.17 Gbps. Another example is the access traffic steering, switching, and splitting (ATSSS), a new feature on the user equipment (UE) and the 5G core network. ATSSS provides finer-grained multiple accesses by introducing the notion of the multi-access PDU session (MAPS), enabling data traffic to be served over one or more concurrent accesses. Since ATSSS technology has been standardized in 3GPP R16 in 2019 [24], KT and vendor Tessares have successfully deployed and tested ATSSS [7, 8] and German Deutsche Telekom has demonstrated the ATSSS benefits on the automated guided vehicle [25]. There are also solutions that do not require mobile device changes but require modifications to the residential gateway. As shown in Fig. 1b, a customer premises equipment (CPE) deployed at the residential gateway converts user traffic into two MPTCP subflows, which are then converted back into a single connection by a hybrid access gateway. Vendor Tessares has started deploying such hybrid access networks to address the bandwidth limitations of xDSL links in rural areas, which may not provide enough bandwidth to support bandwidth-demanding applications such as video conferencing [6, 26].

End-to-end Solution. As the name suggests, the key idea of end-to-end solutions is to upgrade the two endpoints (the application and the server) so that they communicate directly with the multipath transport protocol as illustrated in Fig. 1c. As an example, nine months after the publication of the MPTCP RFC in 2013, Apple adopted the technology on iOS 7 for its voice control product Siri. Following Siri, Apple also expanded the multipath services to other applications, including iCloud, Maps, and Apple Music [5, 27]. A recent evaluation shows that MPTCP improves the time to the first word by 20% in the 95-th percentile for Siri and reduces music stalls by 13% for Apple Music [28, 29]. As another example, Alibaba deployed multipath QUIC for its e-commerce product short-video plays in 2021 and achieved 19 to 50% improvement in the 99-th percentile video-chunk request completion time [3].

Deployment Requirements. These two solutions have different requirements that limit their deployment options. The relay-based solution requires a relay, typically located within the access network, to maintain transparency to other layers of individual networks. As a result, this solution can only provide services to users who subscribe to that access network. It is usually adopted by ISPs, enterprises, or factories and is not suitable for a cross-network setting. Although it is technically possible to deploy the relay in a public cloud accessible from different access networks, we have not seen any large-scale commercial deployments. Our experience suggests that a cross-ISP cloud service for multipath transport does not currently have a clear business model. The end-to-end solution, on the other hand, requires changes to both communication endpoints. Even if the phone vendor integrates the multipath capability into the mobile operating system and releases the public API [29, 30], the application service provider still needs to make changes to the application and upgrade relevant servers. Therefore, the end-to-end solution is suitable for application service providers to adopt. If the solution requires changes to the mobile OS, the deployment will also need support from the device vendor. From our perspective, the relay-based solution is impractical, considering there are three major ISPs in China. It is expensive to upgrade their infrastructure and support cross-ISP access. Therefore, we opted for the end-to-end solution.

3 MPTCP DEPLOYMENT EXPERIENCE

We used MPTCP, the de-facto standardized solution, as a starting point to roll out the multipath transport service. Our initial plan was to provide cross-ISP access for one application and then extend the deployment to other application platforms. To begin with, we collaborated with a major video streaming platform (holding 31% of online video China market shares in 2021 [13, 14]), with the goal of improving video streaming quality by providing aggregated bandwidth and path reliability. The deployment turned out to be more difficult than expected. This section highlights the unexpected problems and findings we encountered during the development, deployment, and operation.

3.1 Development Phase

We revisited three performance issues of the vanilla MPTCP in the mobile scenario. Although there are sophisticated solutions to these problems, we found them impractical from a business standpoint.

Performance suffers from path heterogeneity. It is well known that path heterogeneity, where available bandwidths or round trip times (RTT) of the paths differ considerably, has

a negative impact on the MPTCP performance [31–35]. Path heterogeneity is common in mobile devices with multiple interfaces. However, the default MPTCP scheduler prefers to distribute data on the path with smaller RTT and is unaware of the heterogeneity. This behavior can cause bandwidth under-utilization, and the performance can be worse than using the single-path TCP. Prvious work [31, 33, 34] addresses this problem by designing new MPTCP path schedulers that adjust the subflow-level send window. However, installing new schedulers needs to change the kernel or to load a loadable kernel module (LKM). Such changes are primarily on the MPTCP servers rather than the client side because the downlink dominates the wireless traffic. The deployment constraint is that our partner, who operates the servers, rejects our out-of-tree kernel patches, which could lead to severe security problems in production. When a server operation accident occurs, it is difficult to address the responsibility attribution problem because the accident is caused by two different interest groups (*i.e.,* we provide the code patch, and our partner operates the servers). This motivates us to find a solution that requires the least server modification.

Default scheduler is unaware of path overhead. For mobile devices, the cellular link usually has a higher energy consumption and usage cost than the Wi-Fi link [2, 36]. The default MPTCP scheduling strategy is unaware of the path overhead and can cause excessive cellular usage [1, 37–39]. In practice, unnecessary cellular usage is a critical problem that stops users from enabling the multipath transport feature. One of the most common questions about the network connectivity we have received after the deployment is "why does my phone still consume mobile data when connected to a Wi-Fi network?". Unfortunately, we have the same constraint as in the path heterogeneity problem. Existing work proposes changes to the kernel module, but our partner is unwilling to accept out-of-tree modifications. We thus need a client-only alternative approach.

The default initial path can break. When establishing an MPTCP session, MPTCP utilizes a unique TCP three-way handshake to negotiate the multi-path functionality on the initial path and then initiates TCP subflow handshakes on additional paths to join the same MPTCP session. However, MPTCP always uses the configured default initial path, regardless of its quality [40, 41]. This can cause a significant video startup delay if the quality of the default initial path deteriorates or the link is broken. To address this issue, applications must decide which path is to be used for the connection establishment. We requested that our video platform partner modify their application to allow for path selection based on the network quality when establishing the MPTCP connection. Although they initially accepted our request, they soon discovered that implementing this feature would be costly. Path selection is not an orthogonal function, and to choose the optimal path, they would need to monitor path signal strength, maintain real-time path quality, and handle interface events when they go up or down. All of these functions are beyond the scope of a video player. In the end, our partner declined to make the requested modification, and we had to address this problem independently.

Our Development Efforts. We need to address the aforementioned problems before the deployment. To avoid modifications to standard MPTCP servers and minimize application modifications, we propose performance- and cost-aware state management in the mobile OS and provide a software development kit (SDK) to enable multipath capability in our partner's application. On the one hand, we use backup mode [42] to address path heterogeneity and path overhead issues. In backup mode, the backup path is not selected by the MPTCP scheduler to transfer data until all non-backup paths become unavailable. To address the path heterogeneity problem, we put the worse path into the backup mode when the throughput (or RTT) ratio between the better and worse paths is larger than a threshold. To address the path overhead problem, we first determine whether the Wi-Fi link can support the application's throughput demand based on the difference between estimated real-time throughput and the demand reported by the application via our service API. We then set the cellular link to backup mode if Wi-Fi is sufficient. On the other hand, we use path historical performance to address the path initialization issue. We maintain a path information base that contains measured performance metrics about each path. The performance metrics include signal strength, throughput, RTT, and historical time-to-first-byte time. We then select the path with the best transmission performance as the initial path. With these state management methods, we were able to deploy MPTCP successfully. Note that our solutions are suboptimal compared to methods that change kernel code and application logic. Although we addressed technical problems in the development phase, it is still difficult to handle business problems during the deployment (§3.2) and the operation (§3.3) phases.

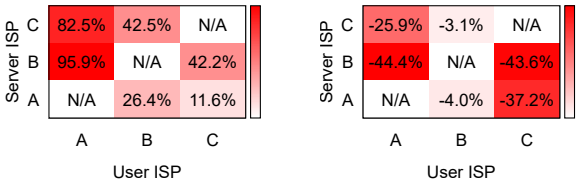
Lessons Learned. One may expect that MPTCP works out of the box as it uses standard sockets with a few options and sysctls. However, the vanilla MPTCP yields suboptimal performance for mobile scenarios, and it is hard to fix them by letting other parties merge the out-of-tree patch or make major changes to the application network library. It is hard to adopt a solution that involves multiple interest parties. This motivates us to find a solution that is transparent to both the application and the server.

3.2 Deployment Phase

In December 2018, we launched an incremental deployment for MPTCP in Beijing, Shanghai, and Shenzhen. Our deployment required modifications to both the client and the server. On the client side, we merged the stable implementation from the Linux Kernel Multipath TCP project [43] into our vendor-specific mobile OS. We also provide an SDK for our video platform partner to glue the app and the multipath capability. Users can activate the multipath transport feature in the application settings. On the server side, we deployed MPTCP servers and obtained the app provider's consent to redirect the app traffic to these servers.

Our deployment covers only video chunk files. While it is technically feasible to redirect all application traffic to the deployed MPTCP servers, which would forward the traffic to the associated origin servers, such redirection is commercially difficult, even for a single application. The reason is that different network flows are actually managed by different interest groups (both inside and outside the video platform), most of which refuse to proxy their traffic through our MPTCP servers, as multipath transport does not align with their interests. *Inside the video platform*, not all departments have an incentive to use multipath transport because network quality does not relate to their performance indicators. For example, the sales department aims at improving the ad conversion rate but does not care about the potential benefit of reducing the ad load time. *Outside the video platform*, the servers are managed by different Internet parties, and it is hard to convince the entire community that MPTCP meets everyone's interests. For example, video application uses the captcha, CDN, and cloud computing services provided by other companies. These companies currently don't have a strong motivation to adopt MPTCP just for a single video application. Finally, we only collaborated with the video player development and maintenance team. Our MPTCP servers only cover video chunk files.

Covering only video chunks yields limited benefits. During our deployment, we discovered that MPTCP's QoE improvement was limited when our MPTCP servers only worked for video files. This is because the non-MPTCP traffic can hold up the application response. First, our deployment can hardly lower the video startup delay. To load the video playback, the HTTP-based adaptive video streaming application first needs to get the manifest file, which defines the available video resolutions and the locations of all the files required to play that video. However, the manifest file requires user authentication and thus is not served by the MPTCP-capable CDN servers because of security concerns. Therefore, the manifest file can block the playback loading process. Second, we cannot reduce the application page load time because pages include user comments, advertisements, and video thumbnails, none of which are served by our deployed MPTCP servers. Without reducing the page load time, it is hard to provide a satisfactory QoE even if video playback is smooth. To avoid the blocking caused by non-MPTCP traffic, we modify the client to automatically select the network with better quality. When users turn on the multipath feature, the mobile OS automatically switches to cellular data when the Wi-Fi is too weak to offer enough throughput.



(a) Delay increase. **(b) Throughput decrease.**

Figure 2: Cross-ISP TCP performance degradation.

Lesson Learned. Ideally, an MPTCP deployment can be incremental, starting with enabling multipath transport solely on the video servers and gradually expanding to other servers. However, only covering video chunk files results in a limited QoE improvement because non-MPTCP traffic can still stall video playback and page loading. It is challenging to deploy MPTCP on all the required servers, even for a single application, as the outreach will need to be massive and involve upgrading multiple sites simultaneously. These commercial challenges motivates us to find a solution that reduces the cross-industry outreach expense.

3.3 Operation Phase

After the deployment, we identified two roadblocks during the operation phase, *i.e.,* the deployed MPTCP relay causes high bandwidth budget, and the network middlebox is not always compatible with the MPTCP flows. After one year of operation, our deployment retired in December 2019 because we failed to find a clear business model to make a profit.

Cross-ISP access leads to a high bandwidth budget. The deployed MPTCP servers need to join multiple ISPs to prevent performance degradation because it is known in the literature that cross-ISP access increases path delay [3, 32, 47]. To confirm the conclusion, we measure the cross-ISP performance degradation over the LTE path with good signal quality (RSSI > -65 dBm). The result in Fig. 2 shows that the delay could inflate up to 1.96x and the throughput could be 44% worse than accessing the servers within the same ISP network. However, joining multiple ISPs causes high bandwidth budget at the internet exchange points. We utilize a metric-based approach to estimate the MPTCP capital expense by Eq. (1), which includes bandwidth cost (C_{bw}), cloud

Symbol	Meaning	Value
N_{user}	# MPTCP users	704k
N_{ISP}	# ISPs in China	3
β_{inject}	Bandwidth overhead	1%
B_{user}	User data consumption [44]	3.4 GB/user/mo
U_{bw}	Bandwidth cost [45]	$123.1/TB
R_{vm}	Packet process speed [46]	1M pps/VM
U_{vm}	Cloud computing cost [46]	$632.7/VM
C_{mtn}	Maintenance cost	$3000/mo
C_{bw}	Bandwidth cost	$0.096/user/mo
C	Overall budget cost	$0.100/user/mo

Table 2: Bandwidth budget for cross-ISP access. "VM" denotes the virtual machine and "pps" denotes packets per second.

computing cost (C_{vm}), and maintenance cost (C_{mtn}):

$$C = C_{\text{bw}} + C_{\text{vm}} + C_{\text{mtn}} \tag{1}$$

$$B_{\text{bw}} = N_{\text{user}} \cdot (N_{\text{ISP}} + \beta_{\text{inject}}) \cdot B_{\text{user}} \tag{2}$$

$$C_{\text{bw}} = B_{\text{bw}} \cdot U_{\text{bw}} \tag{3}$$

$$C_{\text{vm}} = \frac{B_{\text{bw}}}{R_{\text{vm}}} \cdot U_{\text{vm}}$$

The symbol meaning and their estimated values are listed in Tab. 2. Our estimation is the lower bound of the actual cost. We estimated C_{bw} by Eq. (2) and Eq. (3). The coefficient $(N_{\text{ISP}} + \beta_{\text{inject}})$ in Eq. (2) has the following interpretations. Generally, the bandwidth charge increases in proportion to not only the actual amount of traffic but also the number of joined ISP access networks and therefore we approximate the actual cost by multiplying N_{ISP}. The β_{inject} is the bandwidth overhead of the opportunistic retransmission [9]. In brief, the bandwidth cost takes up 96% of the overall budget and is N_{ISP}-times higher than the cost of single-path servers. To avoid unnecessary cross-ISP traffic, we let the client falls back to the legacy TCP connection and connect to the traditional single-path server when Wi-Fi can provide stable latency and enough throughput.

Middleboxes may not be compatible with MPTCP. The sub-tier ISP redirects HTTP requests for static content to its transparent caching middlebox to save bandwidth budget. It has been shown that the caching saves 15.6% of the data volume [48]. During the operation phase, we found that some ISPs perform DNS poisoning for the HTTP redirection, which prevents the client from connecting to the deployed MPTCP servers. In practice, we disable the MPTCP feature for the ISP that performs caching. We thus avoid undesired traffic expenses by preventing the caching middlebox from fetching data from the MPTCP servers.

Lesson Learned. MPTCP connection requires that the multi-homed client connects to the same server. This implies high bandwidth budget and possible packet mangling by a middlebox. This situation motivates us to find a solution that is transparent to the network middlebox.

3.4 Summary

We examined our experience with MPTCP. The primary difficulty is that the end-to-end solution needs to coordinate multiple network parties to collaboratively support MPTCP, which is challenging because different parties have diverse and non-overlapping business incentives. This situation motivates us to turn to a new direction for mobile multipath transport deployment. We propose a solution that is transparent to other network stakeholders, including app providers, ISPs, and CDN operators.

4 FLEETY DESIGN

Our observation is that video traffic in China is primarily delivered over unencrypted HTTP. Additionally, HTTP is compatible with the existing infrastructure, such as network middleboxes and CDN servers, lowering the deployment bar. Therefore, we make a shift from MPTCP to MPHTTP, which uses HTTP byte-range-request to fetch data simultaneously from multiple paths. Based on MPHTTP, we design a mobile system service called Fleety for mobile multipath transport. We implement Fleety as a transparent shim layer in the vendor-specific OS. Unlike previous work that requires application modifications [19, 20], Fleety transparently intercepts the application traffic and performs the multipath-related operations. We also consider other factors that are missed in previous work, including path prioritization, buffer overhead, and consistency verification. The high-level design and the overall workflow are illustrated in Fig. 3. There are four building blocks in Fleety. When the application initiates a new connection via the standard socket API, the flow classifier identifies the flow type and forwards the data for subsequent processing (§4.1). The MPHTTP proxy will then fetch HTTP objects over multiple network interfaces (§4.2). The consistency verifier checks if servers on different paths return the same object (§4.3). To avoid non-HTTP and HTTPS flows blocking the HTTP traffic, the path selector proactively migrates network traffic on the poor network path to other paths (§4.4).

4.1 Flow Classifier

The flow classifier inspects the flow payload, classifies the flows, and takes corresponding actions. There are three flow types. The first type is the HTTP request, which will be forwarded to the MPHTTP proxy. Note that for small HTTP objects, we do not use MPHTTP due to its short flow duration and insignificant performance benefit [2]. The second is the DNS queries. The classifier duplicates the queries, sends them on all paths, and saves the responses for later use. All other flows are classified as the third type. These flows are

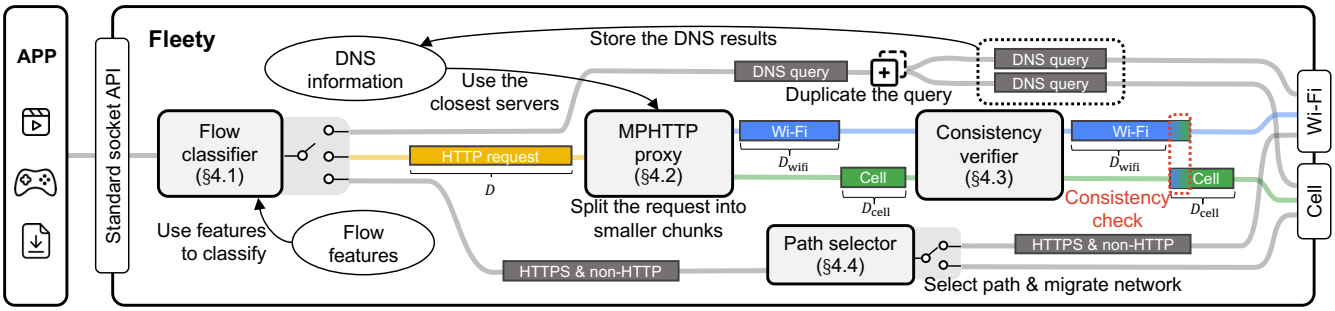


Figure 3: The Fleety design. The thick lines without arrows are two-way data paths (*i.e.,* read and write).

processed the same way as the default network stack, except they might be assigned to different network paths. There are two databases that the flow classifier needs to maintain.

DNS Information Database. Using a single CDN server for both paths impairs the performance as the closest server varies for different paths and the cross-ISP access has a negative impact on throughput and delay [19, 20, 47]. Therefore, we need to perform separate DNS lookups on both paths and fetch the content from different CDN servers. To facilitate this process, the flow classifier maintains a DNS information database, which records the closest servers for both paths. For example, when the application performs a DNS query for domain `d.com` and the responses on two paths are IP addresses `A` and `B`, a record `(d.com, A, B)` is stored in the database. Afterward, when the application connects to the address `A`, we know that the corresponding server on the other path has the address `B`. The classifier also parses HTTP DNS requests (*i.e.,* DNS resolution over HTTP rather than UDP). Our experience is that video streaming platforms use HTTP DNS, and the server IP address can be encapsulated in JSON, URL link, or other proprietary formats.

Flow Feature Database. The flow classifier relies on the flow payload features for classification. The features are used to decide whether MPHTTP should be enabled. For example, we can determine that an HTTP request is for the video chunk by checking whether the URL format or request header values are in the database. The features are hardcoded in a database, which will be updated regularly.

4.2 MPHTTP Proxy

The MPHTTP proxy fetches data simultaneously from multiple paths by using HTTP byte-range-request. The proxy first parses the HTTP request sent by the application. The proxy then splits the request into two smaller chunks and assigns the chunks to different network paths. When the servers return different portions of the requested objects, the proxy reassembles them into one HTTP response and returns to the application. The application is completely unaware of the multipath process behind the scenes.

Path Scheduler. The key design of the HTTP proxy is its scheduling algorithm, which determines the sizes of the chunks assigned to different paths so that the paths complete their transfers simultaneously. Suppose the application requests a chunk of size D, and the Wi-Fi path and the cellular path download the chunk size of D_{wifi} and D_{cell} respectively, as shown in Fig. 3. We have $D = D_{\text{wifi}} + D_{\text{cell}}$ and the transfer complete time for Wi-Fi path is $T_{\text{wifi}} = RTT_{\text{wifi}} + \frac{D_{\text{wifi}}}{R_{\text{wifi}}}$, where RTT_{wifi} and R_{wifi} are the delay and the throughput of Wi-Fi path. Similarly, we have $T_{\text{cell}} = RTT_{\text{cell}} + \frac{D_{\text{cell}}}{R_{\text{cell}}}$. We then solve the equation $T_{\text{wifi}} = T_{\text{cell}}$, which means both paths complete transfer at the same time (we denote this time as $\hat{T}$), and calculate the chunk sizes assigned to both paths and achieve a theoretical maximum path utilization. Unfortunately, the file size D is not always available before the path scheduler sends the request. In this case, we first send the request for the whole object on one path (denoted as *Path 1*). As soon as the HTTP response header returns, we get the file size and send a byte-range request over the other path (denoted as *Path 2*). When we receive the planned amount of data from over *Path 1*, we reset the connection on *Path 1*.

Path Prioritization. Generally, prioritization over Wi-Fi is desirable for most of our users. To this end, we set a tolerance time T_{tol} and try to use Wi-Fi for the duration of at least T_{tol}. Specifically, we first check whether $RTT_{\text{wifi}} + \frac{D}{R_{\text{wifi}}}$ is less than T_{tol}. If it is, we use only the Wi-Fi path. Otherwise, i) if $\hat{T} > T_{\text{tol}}$, we use the path scheduler mentioned above; ii) if $\hat{T} \leq T_{\text{tol}}$, we re-calculate the chunk sizes by solving the equation $T_{\text{wifi}} = T_{\text{tol}}$. In practice, we set T_{tol} to one second. Fleety thus avoids unnecessary data usage when the Wi-Fi is of good quality. Fleety at the moment always prioritizes the Wi-Fi path. In the future design, Fleety can allow users to decide their preferences.

Buffer Overhead. Fleety divides an HTTP request into two sub-requests and sends them on different paths. When the HTTP responses are returned, the chunk with the smaller byte range number can be returned to the application at once; in contrast, Fleety needs to buffer the other chunk until the

first chunk is all read by the application. This buffering could cause a large memory overhead for the file-downloading use case. Also, buffering a large amount of data wastes data usage when the application aborts the connection. To minimize the buffer overhead, if the requested file is larger than a threshold, we split the request into several portions and then schedule each portion sequentially. The threshold is set as $\min\{M, (R_{\text{wifi}} + R_{\text{cell}}) \cdot T\}$ where M is the available system resource and T is a configurable parameter for performance trade-off. As a result, the maximum memory overhead will not exceed the threshold.

4.3 Consistency Verifier

Fleety includes a consistency verifier to ensure the objects returned on different paths are identical. Ideally, the CDN servers should replicate the content from the original server and offer identical copies of that content. However, content inconsistency might occur in the wild due to caching middleboxes, which are commonly used by ISPs to cache HTTP objects. The cached data may be outdated, and this can cause the objects returned on different paths to differ. In such a case, the MPHTTP proxy splices two different objects and returns to the application. The application is unable to interpret the result, and Fleety is unaware of the failure. Therefore, Fleety needs to reliably detect the inconsistency and *fall back to single path transport if the inconsistency is detected.*

Inspecting the `Last-Modified` or `Etag` fields in the HTTP response headers is a straightforward idea to identify a specific version of a resource, as these fields can be considered as "file checksums" and are used to indicate the status of a resource [49]. Unfortunately, our experience shows that there is no broad consensus for using these optional headers to specify the resource version. Site operators or some major CDN service providers often denote the resource status in their own private way, such as using hash functions to generate a globally unique ID for the resource.

To address this problem, the verifier detects an inconsistency by comparing a small portion of the object. This idea derives from the assumption that if there are two versions of an object, the samples of a small portion likely do not match. In theory, such a consistency check has zero false positives but may have false negatives when the file difference does not belong to the portion we sample. Such false negatives are inevitable unless we use the entire file to compare. The key problem here is which portion is used for consistency check so that we can minimize the overhead. For example, we can always use the first 100 bytes of the object. However, this approach incurs perceivable overhead because we need to send additional HTTP requests to fetch this portion. To avoid the overhead of checking a pre-configured portion byte-rage, the consistency verifier examines the intersection of two byte-range requests. As illustrated in Fig. 3, the start byte position of D_{cell} follows the end byte position of D_{wifi}. We now extend D_{wifi} end byte position so that the two ranges overlap. Therefore, the verifier can perform the consistency check with low overhead by comparing the overlapped parts on two paths. During our extensive in-lab experiments and the subsequent deployment of Fleety to ~10 million users, we did not encounter any consistency check failures or receive any customer complaints about video players playing incorrect content. This suggests that the consistency verifier empirically works well. However, it is still necessary to include this system module as a precautionary measure to ensure the continued reliability of Fleety.

4.4 Path Selector

The path selector provides link reliability for the non-HTTP and HTTPS flows by binding the flow to the path that meets the flow's QoS requirement. To accomplish this, we follow the existing common practice of automatically selecting the optimal available network. This method is often referred to as Wi-Fi Assist, Adaptive Wi-Fi, or Wi-Fi+ [50–54]. Fleety monitors the network condition, determines the QoS requirement, and finds the best path. The network condition is characterized by throughput, latency, and jitter. The QoS requirement is determined by the specific needs of the application, which are hardcoded in a configurable file. For example, gaming demands low latency, file downloading prefers a high throughput, and instant messaging does not have a strict requirement. By comparing traffic analysis results with the QoS requirements, the path selector is able to select the path with better quality when the flow starts. The selector also proactively terminates the flow in the middle of a flow if the path becomes worse or down and the other path can meet the QoS requirement. When the selector terminates the flow, the application will then try to re-create the flow, and the selector will select the suitable path.

5 IMPLEMENTATION

Fleety is implemented as a system service within the mobile operating system. Fleety is designed as a shim layer to allow the application to use the standard socket API to transmit data across multiple paths. To achieve this, we leverage the Linux feature Netfilter [55] for intercepting traffic initiated by specific applications that the user enables Fleety for. The intercepted traffic is redirected to Fleety, which acts as a proxy and handles all related multipath operations.

To improve the performance of MPTCP path management (§3.1), path scheduler (§4.2), and path selector (§4.4), Fleety requires accurate and real-time network metrics such as throughput, delay, jitter, and packet loss. To obtain precise estimates, we use kernel hooks [56] to collect packet-level information. We also develop a kernel module that gathers these fine-grained measurements every 200 ms and stores the results in a database of historical path performance. All

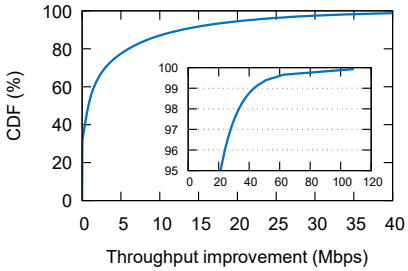


Figure 4: Throughput improvement.

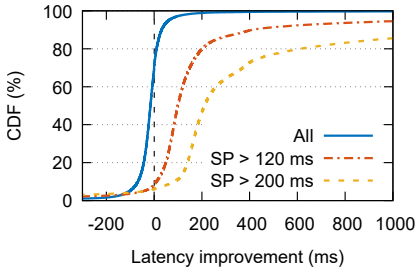


Figure 5: Latency improvement.

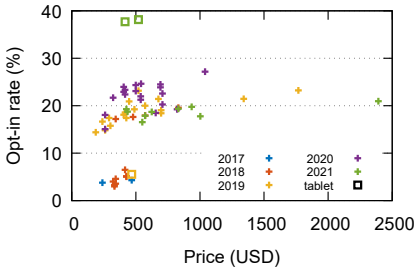


Figure 6: Opt-in rate on each device model.

Fleety modules can access this database to obtain network performance characteristics, helping to facilitate more informed decisions when managing network paths.

6 EVALUATION

We deployed Fleety in September 2019. Nowadays, Fleety supports 142 device models (including smartphones and tablets) and 156 popular applications (such as social media, video, gaming, news, and cloud disk) in China. To evaluate the Fleety performance in the wild, we collect data from user devices and share the analysis result. All the data were collected with the informed consent of opt-in users, and no personally identifiable information (PII) was collected during the measurement. The Fleety is not enabled by default. We use gentle prompts to inform users of the new feature. In addition, Fleety can be enabled only for some selected applications but not all. In January 2022, there were 9.96 million opted-in users that used multipath transport for one or more applications. The opt-in rate is 20.63%.

6.1 Performance Measurement

Throughput Improvement. To measure the bandwidth benefit of MPHTTP, we compared the performance of single-path throughput and multipath throughput. By default, the device uses only single-path transport (Wi-Fi). Fleety monitors the real-time throughput for each TCP flow. When the single-path fails to meet the application QoE requirement, Fleety turns on the cellular connectivity and enables multipath transport. We instrument Fleety to record the single-path throughput and multipath throughput as a pair when transitioning from single-path to multipath. The throughput pair is then uploaded to our data center. Finally, we collected over two billion pairs from September 2021 to January 2022. The throughput improvement and its long tail above the 95 percentile are shown in Fig. 4. Note that the improvement is less than the potential bandwidth gain of multipath transport because Fleety prefers using the Wi-Fi path and distributes less data on the cellular path. This design results in lower throughput improvement from a statistical view. Moreover, the app-required throughput often does not exceed the single path throughput. This observation is consistent with a prior study [1] on multipath video streaming. The shortage of data to transmit over multiple paths also lowers the median improvement. The median and average improvements are 82 Kbps and 4.43 Mbps, respectively, demonstrating Fleety's advantage for bandwidth-demanding applications.

Latency Improvement. Fleety includes a path selector that automatically selects the best available path for non-HTTP flows. To evaluate the latency benefit of Fleety, we compare the flow RTT before and after the time when Fleety migrates the network. We collect two billion network switching events, and the result is shown in Fig. 5. *From the viewpoint of all the data*, Fleety has no significant impact (*i.e.*, the latency difference is less than 30 ms) on the network latency for 49.97% of network switching events. For 5.32% of the events, Fleety increases the latency by more than 100 ms. We attribute the increase in latency to two facts. First, before changing the network path, the path selector can only be aware of the path signal strength instead of the actual path RTT. A good signal quality does not always mean there is no congestion in the network. Second, the path selector decides to switch the network not only based on the RTT but also on other factors including bandwidth and packet loss rate. *From the viewpoint of the flows that experience a high latency before the network switch*, Fleety significantly lowers the network latency. For the flows using the single path and with a latency of more than 120 ms and 200 ms, Fleety lowers the latency by 99.30 ms and 210.91 ms in the median. This illustrates the Fleety's ability to maintain low latency by leveraging multipath transport and to avoid the HTTP flows from being blocked by the non-HTTP ones.

6.2 User Behavior Measurement

Based on our data collection, we have observed multifold factors that impact users' willingness to use the mobile multipath capability provided by Fleety.

Application	Normalized opt-in rate	# Opt-in users	Category
WeChat	1.00	6,687,806	Instant Messaging
Arena of Valor	0.57	601,962	Game
App Market	0.50	3,478,005	App Downloader
Huya Live	0.44	106,826	Video
Game for Peace	0.41	238,982	Game
Tencent Video	0.30	628,948	Video
Taobao	0.30	1,653,067	E-commerce
QQ	0.28	972,334	Instant Messaging
Baidu Web Drive	0.27	209,832	Cloud Storage
QQ Downloader	0.27	390,870	File Downloader

Table 3: Top-10 apps ordered by opt-in rate.

Product Model. Our measurement reveals that the user's willingness to enable multipath is related to the device model. We plot each device model's price and opt-in rate in Fig. 6. The numbers in the legend represent the device release dates, and the cross symbols and squares represent smartphones and tablets, respectively. From a global view, there exists a positive correlation. Flagship model users are more likely to opt-in. From the view of the point cluster, there are two outliers in the figure. The first outlier is the models with less than 10% opt-in rates. These devices were released before the development of Fleety in September 2019, and we do not send prompts to encourage users to enable the multipath feature. The second one is the tablets released in 2021, which have opt-in rates of nearly 40%. Users that buy tablets equipped with cellular connectivity are naturally ready to pay for high-data mobile plans. In brief, high-end device users are more likely to use the multipath feature.

Mobile Applications. Fleety supports 154 popular mobile applications in China. The vendor OS allows users selectively turn on Fleety for different applications. The top 10 applications with the highest opt-in rate are listed in Tab. 3. We normalize the opt-in rate by dividing by the maximum, and we sort them by the opt-in rate. The first (WeChat) and the third (App Market) have the most users because they are extremely popular in users' daily life. Amongst the ten applications, two are used for video streaming, two for real-time gaming, three for file downloading (*i.e.,* app store, cloud storage, and downloader), and the remaining three for instant messaging and browsing. Overall, although the numbers of users vary drastically across different applications, interactive-sensitive applications such as video streaming and gaming tend to have a high opt-in rate because multipath transport can automatically select the better network path to ensure the user experience. Another finding is that file-downloading applications have a lower opt-in rate than instant messaging. This is because some users have concerns about their data plan and prefer only using Wi-Fi despite a longer download time. Note that the specific applications differ in different countries but we believe that our findings still apply. We suggest that from the perspective of consumers, multipath transport is more useful for interactive-sensitive applications than throughput-demanding ones.

Where to enable Fleety	(%)
Heads-up notification when using the app	80.04
Notification drawer and lock screen notification	14.25
"WLAN - Network acceleration" in the setting	5.71

Table 4: Proportions of the methods by which users enable Fleety.

Feature Adoption and Discovery. Due to the energy and data usage overhead associated with multipath transport, Fleety is disabled by default. To encourage users to take advantage of multiple network interfaces, we designed heads-up prompts and phone notifications that direct users' attention to the multipath feature. Experienced users who appreciate the Fleety enhancements can also enable it in the system settings. Tab. 4 lists the proportion of ways users enable Fleety, showing that 80% of users adopt Fleety by clicking the "enable" button on the heads-up notification while using the application. Less than 6% of opt-in users are aware of the multipath feature before the feature promotion. In brief, multipath transport is not a well-known feature among consumers. We suggest that in the future, we need to more efficiently promote multipath transport (*e.g.,* pop up a prompt when the single path network becomes unstable).

Wi-Fi Quality. Contrary to our previous belief that users with poor Wi-Fi coverage are more likely to use Fleety, our data shows that the Wi-Fi signal generally does not impact the opt-in willingness. As indicated in Fig. 7, we calculate each user's average Wi-Fi RSSI samples. No statistical difference exists between the distribution of opt-in users and all users (p-value = 0.59 > 0.05, using Kolmogorov–Smirnov test). We then measure the Wi-Fi session persistence time for each user, which is defined as the average duration when the user is connected to the same access point. From Fig. 8, we can see that opt-in users have a shorter Wi-Fi persistence time (1.64 hours) than normal users (1.98 hours). To investigate the impact of persistence time on opt-in willingness, we use Bayes' theorem to calculate the posterior probability of opt-in rate given the persistence time. As illustrated in Fig. 9, the opt-in rate tends to increase with a shorter Wi-Fi persistence time. A possible hypothesis for the reason why a poor Wi-Fi signal does not apparently affect the opt-in rate is as follows: applications usually have the ability (*e.g.,* adaptive bitrate streaming algorithm) to adapt to the poor Wi-Fi performance; in contrast, when a path becomes unavailable, it causes a complete network blackout which is more noticeable and more disrupting. This finding suggests that path reliability is one of the most important factors in offering a better network experience.

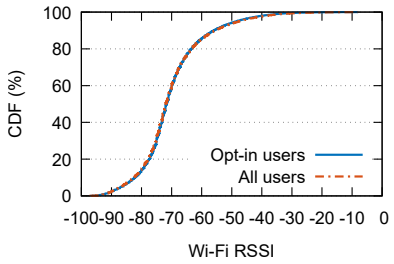


Figure 7: Distribution of the Wi-Fi signal strength.

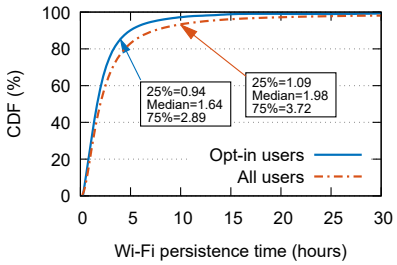


Figure 8: Distribution of the Wi-Fi session persistence time.

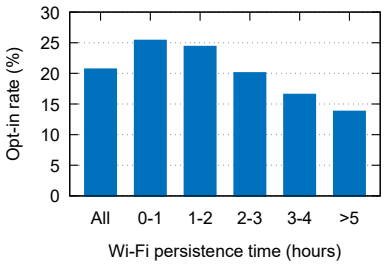


Figure 9: Posterior probability of opt-in rate.

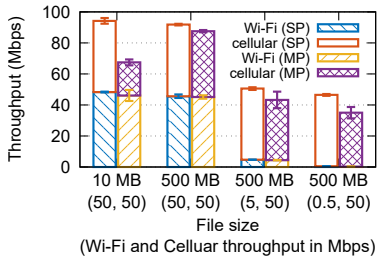


Figure 10: MPHTTP throughput aggregation.

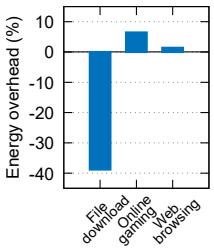


Figure 11: Energy consumption.

6.3 Micro-benchmark

MPHTTP Bandwidth Aggregation. We evaluate how well MPHTTP aggregates the multipath bandwidth and prioritizes the Wi-Fi path. Fig. 10 plots the throughput of single-path and multipath transport in four settings with the same RTT (50 ms) but different throughputs, which are imposed by the Linux `tc` [57]. For each setting, we first test the throughput of two separate paths. We then test the MPHTTP throughput and calculate how much traffic is distributed over two paths. Each experiment is repeated ten times. We define the "aggregation rate" as the ratio between MPHTTP throughput and the ideal throughput (*i.e.,* the throughput summation of all paths). For the medium file size of 10 MB (approximate to the size of a 10-second 1080p video chunk), Fleety achieves an aggregation rate of 71.58%, with low cellular usage (46.46%) and almost fully utilized Wi-Fi path (95.41%) due to its preference for Wi-Fi. For the large file size of 500 MB, we repeat the experiment with the same cellular network but different Wi-Fi throughputs, which are limited to 50, 5, and 0.5 Mbps, respectively. The aggregation rates are 91.42%, 85.53%, and 75.35%. In all three cases, the Wi-Fi path is always fully utilized. In contrast, Fleety conservatively assigns data to the cellular path. This experiment shows that Fleety efficiently aggregates the bandwidth of multiple network paths and is aware of the cost of cellular usage.

Energy Consumption. We also evaluate the energy overhead of Fleety compared with only using the Wi-Fi single-path network. The result is shown in Fig. 11. For bandwidth-demanding flows (*i.e.,* file downloading) on paths with good quality (Wi-Fi RSSI is -70 dBm and LTE RSRP is -105 dBm), Fleety lowers the energy consumption by 38.79% because of the reduction in download time. For online gaming and web browsing, we conduct the experiment with the same cellular connectivity but with poor Wi-Fi (RSSI is -85 dBm) to encourage Fleety to use the cellular path. The energy overheads are 6.43% and 1.40%, respectively. Considering that multipath transport provides a better network quality, the energy overhead of Fleety is practically acceptable.

7 DISCUSSION

MPHTTP for Encrypted Traffic. The majority of the world is transitioning to HTTPS for securing clients and servers from potential threats [58]. Currently, the MPHTTP proxy only supports plaintext HTTP. This design choice was made due to the dominance of video content on the Internet [15], with the top three video streaming platforms in China using HTTP to distribute content. We consulted the teams of these streaming platforms to understand the reasons behind HTTP's widespread use. The encryption principle is actually shaped by the balance between security and economic interests. From the perspective of security, all personal and proprietary information, such as watch and search history, video comments, and manifest files, is transferred over HTTPS to ensure their confidentiality. Only video chunks on the streaming platforms, which do not contain sensitive content, are served over HTTP. It is important to note that these video chunks can still be encrypted, with the app holding the decryption key to play the videos, preventing middlemen such as ISPs and device manufacturers from accessing the content. From the perspective of economic interests, HTTP does not require computing resources for encryption, resulting in a 36% savings in server expenses compared to HTTPS [59]. Moreover, HTTP allows ISP caching, which could save 15.6% of the data volume [48]. Although the MPHTTP proxy

cannot handle HTTPS traffic, this is not a fundamental limitation. We only need to examine <0.1% of HTTP traffic, including a small number of HTTP header fields for scheduling and a tiny fraction of the HTTP body for integrity check. With dedicated APIs, applications can still use HTTPS while having the above information securely exposed to Fleety. Our ongoing work is cooperating with the app provider and deploying the HTTPS-compatible version of Fleety. It is worth noting that HTTPS support raises the deployment and maintenance bar since it requires application changes. This is why we first rolled out HTTP-only Fleety, which can immediately benefit the applications at scale.

Evaluation Limitation. The system evaluation follows a non-standard approach instead of the conventional A/B testing method. In the ideal scenario, all opt-in users would be randomly divided into a control group and a treatment group. The control group would use a single path (either Wi-Fi or cellular only), while the treatment group would utilize Fleety to leverage multiple paths. However, this method may negatively impact the performance of users in the control group and does not align with our current version of the user agreement, as it intentionally disables the multipath functionality that users have explicitly requested. In this paper, we collect performance metrics before and after the point at which Fleety aggregates bandwidth or switches paths to reduce latency spikes. This approach may introduce potential biases, as Fleety only takes actions when single-path performance is inadequate, and we do not consider single-path results when it is already performing well.

Performance Overhead. Compared with MPTCP which can schedule the data at the packet level [31, 60], MPHTTP works at the chunk level and the scheduling is much more coarse-grained. Fleety uses MPHTTP to trade the performance for deployability and flexibility. The MPHTTP proxy at the moment only works for medium-to-large files due to performance considerations. For apps that generate low traffic volume, the benefit of Fleety may be limited.

8 RELATED WORK

Multipath Transport in Industry. Besides large data centers [12], there exists multipath transport deployment for the mobile scenario. A common practice is to deploy the relay to avoid modifications to content providers. Examples include KT's GiGA [4], Tessares' hybrid access service [6], and ATSSS access [8]. To address the head-of-line blocking problem of MPTCP, Deutsche Telekom proposes MP-DCCP for latency-sensitive applications [61]. In addition, application providers can also enable multipath transport using end-to-end solutions, such as MPTCP [5] and MPQUIC [3]. All previous solutions require modifications to the client and the server (or the relay), while Fleety is a client-only solution and uses MPHTTP to maintain transparency to other network parties. The industry also proposes physical-layer solutions. Dual connectivity (DC) maintains physical-layer connections to LTE and NR cells simultaneously [62–64]. Similar to DC, LTE-WLAN aggregation (LWA) aggregates LTE and Wi-Fi at the physical layer [65]. Carrier aggregation (CA) can improve the throughput by assigning more frequency blocks to a user [62]. These solutions need to upgrade the base station and cannot work in a cross-ISP setting.

Multipath Transport in Academia. Numerous studies have explored multipath solutions at different protocol layers [66]. At the IP layer, there exists work using the multi-homed intermediate router [67, 68] or IP tunneling [69] to create multipath routing for aggregated bandwidth and link robustness. Another approach is Multihoming by IPv6 intermediation (SHIM6) [70], which is a shim layer between the IP layer and the transport layer that facilitates switching between different IP addresses. At the transport layer, MPTCP [12], concurrent multipath transfer for SCTP (CMT-SCTP) [71], MP-DCCP [61], MPRTP [72], and MPQUIC [73, 74] make multipath extensions for transport protocols. At the application layer, mHTTP [18], MSPlayer [19], MP-H2 [20], and the work in [75] present HTTP-based multipath solutions for adaptive streaming. Fleety also uses HTTP for multipath transport but is different from the existing work [18–20, 75]. First, Fleety as a shim layer does not require modifications to the application, the server, and the middlebox. Second, Fleety prefers using the Wi-Fi path to save data usage. Third, Fleety incorporates a lightweight verifier to ensure data consistency across the network paths.

9 CONCLUSION

Multipath transport can improve mobile users' experience in terms of throughput, latency, or reliability. Despite the vast interest from research, large-scale deployments of multipath transport have been slow over the past decade on the public Internet. We have presented a new way to push forward the mobile multipath deployment from the perspective of phone vendors. We developed and deployed Fleety, a mobile system service that supports multipath while maintaining transparency with other Internet parties. We believe our design can lower the deployment bar and improve the penetration rate of multipath transport.

ACKNOWLEDGMENTS

We are grateful to the anonymous MobiCom reviewers for their constructive critique and valuable comments, all of which have greatly helped us improve this paper. This work is supported in part by the National Key Research and Development Plan, China (Grant No. 2020YFB1710900) and the National Natural Science Foundation of China (Grant No. 62022005, 62272010 and 62061146001). Chenren Xu is the corresponding author.

REFERENCES

[1] Bo Han, Feng Qian, Lusheng Ji, and Vijay Gopalakrishnan. MP-DASH: Adaptive video streaming over preference-aware multipath. In *ACM CoNEXT*, 2016.

[2] Bo Han, Feng Qian, and Lusheng Ji. When should we surf the mobile web using both Wi-Fi and cellular? In *ACM AllThingsCellular*, 2016.

[3] Zhilong Zheng, Yunfei Ma, Yanmei Liu, Furong Yang, Zhenyu Li, Yuanbo Zhang, Jiuhai Zhang, Wei Shi, Wentao Chen, Ding Li, et al. XLINK: QoE-driven multi-path QUIC transport in large-scale video services. In *ACM SIGCOMM*, 2021.

[4] In Korean, multipath TCP is pronounced GIGA path. http://blog.multipath-tcp.org/blog/html/2015/07/24/korea.html. (Accessed on 02/17/2022).

[5] Use multipath TCP to create backup connections for iOS - Apple Support. https://support.apple.com/en-us/HT201373. (Accessed on 02/17/2022).

[6] Hybrid access solution - Tessares. https://www.tessares.net/hybrid-access-solution/. (Accessed on 07/14/2022).

[7] KT, Tessares claim first '5G low latency multi-radio access technology test' on live network. https://www.commsupdate.com/articles/2019/09/05/kt-tessares-claim-first-5g-low-latency-multi-radio-access-technology-test-on-live-network/. (Accessed on 03/13/2022).

[8] Opening the way to 4G / 5G Wi-Fi convergence (new) - Tessares. https://www.tessares.net/new-white-paper-opening-the-way-to-5g-convergence-september-2020/. (Accessed on 03/12/2022).

[9] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. How hard can it be? Designing and implementing a deployable multipath TCP. In *USENIX NSDI*, 2012.

[10] Fabien Duchene and Olivier Bonaventure. Making multipath TCP friendlier to load balancers and anycast. In *IEEE ICNP*, 2017.

[11] Recent articles and software — MPTCP. http://blog.multipath-tcp.org/blog/html/archive.html. (Accessed on 02/15/2023).

[12] Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. Improving datacenter performance and robustness with multipath TCP. *ACM SIGCOMM CCR*, 2011.

[13] Annual reports | iQIYI, Inc. https://ir.iqiyi.com/financial-information/annual-reports. (Accessed on 02/09/2023).

[14] China: video streaming revenue per user 2017-2027 - Statista. https://www.statista.com/forecasts/1270445/video-streaming-svod-revenue-per-user-china. (Accessed on 02/09/2023).

[15] Ericsson mobility report. https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/november-2022. (Accessed on 02/08/2023).

[16] Lucian Popa, Ali Ghodsi, and Ion Stoica. HTTP as the narrow waist of the future Internet. In *ACM HotNets*, 2010.

[17] Chinese streaming platforms: 6 trending apps - Tenba Group. https://tenbagroup.com/6-chinese-streaming-platforms-to-amplify-your-business/. (Accessed on 08/19/2022).

[18] Juhoon Kim, Yung-Chih Chen, Ramin Khalili, Don Towsley, and Anja Feldmann. Multi-source multipath HTTP (mHTTP): a proposal. In *ACM SIGMETRICS*, 2014.

[19] Yung-Chih Chen, Don Towsley, and Ramin Khalili. MSPlayer: Multi-source and multi-path video streaming. *IEEE Journal on Selected Areas in Communications*, 2016.

[20] Ashkan Nikravesh, Yihua Guo, Xiao Zhu, Feng Qian, and Z Morley Mao. MP-H2: a client-only multipath solution for HTTP/2. In *ACM MobiCom*, 2019.

[21] RFC 7233 - hypertext transfer protocol (HTTP/1.1): Range requests. https://datatracker.ietf.org/doc/html/rfc7233. (Accessed on 08/13/2022).

[22] KT's GiGA LTE: Commercial mobile mptcp proxy service launch; collaboration with handset manufacturers. https://www.ietf.org/proceedings/93/slides/slides-93-mptcp-3.pdf. (Accessed on 02/17/2022).

[23] KT's GiGA LTE: Mobile mptcp proxy deployment. https://datatracker.ietf.org/meeting/97/materials/slides-97-banana-kt-giga-lte-mobile-mptcp-proxy-development-01. (Accessed on 03/11/2022).

[24] 3GPP; technical specification group services and system aspects; system architecture for the 5G system (5Gs); stage 2 (release 16). https://www.3gpp.org/ftp/Specs/archive/23_series/23.501/23501-g60.zip.

[25] Deutsche Telekom demonstrates multipath for fixed mobile convergence on campus - Deutsche Telekom. https://www.telekom.com/en/company/details/deutsche-telekom-demonstrates-multipath-for-fixed-mobile-convergence-on-campus-625838. (Accessed on 08/01/2022).

[26] Innovating with multipath TCP – Olivier Bonaventure. https://perso.uclouvain.be/olivier.bonaventure/blog/html/2020/06/22/deploying_new_multipath_tcp_use_cases.html. (Accessed on 07/13/2022).

[27] iOS & Linux implementation updates. https://datatracker.ietf.org/meeting/99/materials/slides-99-mptcp-sessa-ios-linux-implementation-updates/. (Accessed on 02/18/2022).

[28] Boost performance and security with modern networking - WWDC20 - videos - Apple Developer. https://developer.apple.com/videos/play/wwdc2020/10111. (Accessed on 07/13/2022).

[29] Advances in networking, part 1 - WWDC17 - videos - Apple Developer. https://developer.apple.com/videos/play/wwdc2017/707/. (Accessed on 03/03/2022).

[30] Improving network reliability using multipath TCP - apple developer documentation. https://developer.apple.com/documentation/foundation/urlsessionconfiguration/improving_network_reliability_using_multipath_tcp. (Accessed on 03/03/2022).

[31] Hang Shi, Yong Cui, Xin Wang, Yuming Hu, Minglong Dai, Fanzhao Wang, and Kai Zheng. STMS: Improving MPTCP throughput under heterogeneous networks. In *USENIX ATC*, 2018.

[32] Li Li, Ke Xu, Tong Li, Kai Zheng, Chunyi Peng, Dan Wang, Xiangxiang Wang, Meng Shen, and Rashid Mijumbi. A measurement study on multi-path TCP with multiple cellular carriers on high speed rails. In *ACM SIGCOMM*, 2018.

[33] Yeon-sup Lim, Erich M Nahum, Don Towsley, and Richard J Gibbens. ECF: An MPTCP path scheduler to manage heterogeneous paths. In *ACM CoNEXT*, 2017.

[34] Imran Khan, Moinak Ghoshal, Shivang Aggarwal, Dimitrios Koutsonikolas, and Joerg Widmer. Multipath TCP in smartphones equipped with millimeter wave radios. In *ACM WiNTECH*, 2022.

[35] Vivek Adarsh, Paul Schmitt, and Elizabeth Belding. MPTCP performance over heterogenous subpaths. In *IEEE ICCCN*, 2019.

[36] Swetank Kumar Saha, Abhishek Kannan, Geunhyung Lee, Nishant Ravichandran, Parag Kamalakar Medhe, Naved Merchant, and Dimitrios Koutsonikolas. Multipath TCP in smartphones: Impact on performance, energy, and cpu utilization. In *ACM MobiWac*, 2017.

[37] Junxian Huang, Feng Qian, Alexandre Gerber, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A close examination of performance and power characteristics of 4G LTE networks. In *ACM MobiSys*, 2012.

[38] Duc Hoang Bui, Kilho Lee, Sangeun Oh, Insik Shin, Hyojeong Shin, Honguk Woo, and Daehyun Ban. Greenbag: Energy-efficient bandwidth aggregation for real-time streaming in heterogeneous mobile wireless networks. In *IEEE RTSS*, 2013.

[39] Yeon-sup Lim, Yung-Chih Chen, Erich M Nahum, Don Towsley, and Richard J Gibbens. How green is multipath TCP for mobile devices? In *ACM AllThingsCellular*, 2014.

[40] An improvement of MPTCP initialization. https://www.ietf.org/archive/id/draft-kien-mptcp-init-00.txt. (Accessed on 06/21/2022).

[41] Markus Amend, Veselin Rakocevic, Andreas Philipp Matz, and Eckard Bogenfeld. RobE: Robust connection establishment for multipath TCP. In *ACM/IRTF ANRW*, 2018.

[42] RFC 6824 - TCP extensions for multipath operation with multiple addresses. https://datatracker.ietf.org/doc/html/rfc6824. (Accessed on 03/07/2022).

[43] C. Paasch, S. Barre, et al. Multipath TCP in the Linux Kernel. https://multipath-tcp.org/. (Accessed on 02/17/2022).

[44] Streaming video - Mbs to GBs - mobility report - Ericsson. https://www.ericsson.com/en/reports-and-papers/mobility-report/articles/streaming-video. (Accessed on 06/22/2022).

[45] How much does data really cost an ISP? - BroadbandNow. https://broadbandnow.com/report/much-data-really-cost-isps/. (Accessed on 08/08/2022).

[46] Alibaba cloud: Cloud computing services. https://www.alibabacloud.com/. (Accessed on 08/08/2022).

[47] Ashkan Nikravesh, Yihua Guo, Feng Qian, Z Morley Mao, and Subhabrata Sen. An in-depth understanding of multipath TCP on mobile devices: Measurement and system design. In *ACM MobiCom*, 2016.

[48] David Naylor, Alessandro Finamore, Ilias Leontiadis, Yan Grunenberger, Marco Mellia, Maurizio Munafò, Konstantina Papagiannaki, and Peter Steenkiste. The cost of the "S" in HTTPS. In *ACM CoNEXT*, 2014.

[49] RFC 7232 - hypertext transfer protocol (HTTP/1.1): Conditional requests. https://datatracker.ietf.org/doc/html/rfc7232. (Accessed on 03/07/2022).

[50] About Wi-Fi Assist - Apple Support. https://support.apple.com/en-us/HT205296. (Accessed on 03/03/2022).

[51] Activate Adaptive Wi-Fi on my Samsung device - Samsung Australia. https://www.samsung.com/au/support/mobile-devices/enable-adaptive-wi-fi/. (Accessed on 03/03/2022).

[52] Wi-Fi Assistant - OPPO Global. https://support.oppo.com/en/answer/?aid=neu4992. (Accessed on 03/03/2022).

[53] How to activate the Wi-Fi Assistant on your Xiaomi and improve Internet speed. https://www.xiaomist.com/2020/11/how-to-activate-wifi-assistant-on-your.html. (Accessed on 03/04/2022).

[54] Turn Wi-Fi+ on or off; how to use wi-fi+; how do I enable Wi-Fi+ - HUAWEI Support Global. https://consumer.huawei.com/en/support/content/en-us00449892/. (Accessed on 03/04/2022).

[55] netfilter/iptables project homepage. https://www.netfilter.org/. (Accessed on 08/19/2022).

[56] Networking — The Linux Kernel documentation. https://linux-kernel-labs.github.io/refs/heads/master/labs/networking.html#netfilter-1. (Accessed on 02/05/2023).

[57] tc(8) - linux manual page. https://man7.org/linux/man-pages/man8/tc.8.html. (Accessed on 02/11/2023).

[58] HTTPS encryption on the web – Google Transparency report. https://transparencyreport.google.com/https/overview. (Accessed on 01/12/2023).

[59] Amazon CloudFront pricing - Amazon CDN. https://www.amazonaws.cn/en/cloudfront/pricing/. (Accessed on 02/09/2023).

[60] Swetank Kumar Saha, Shivang Aggarwal, Rohan Pathak, Dimitrios Koutsonikolas, and Joerg Widmer. Musher: An agile multipath-TCP scheduler for dual-band 802.11 ad/ac wireless LANs. In *ACM MobiCom*, 2019.

[61] MP-DCCP: Multipath extension for DCCP. https://multipath-dccp.org/. (Accessed on 07/18/2022).

[62] T Uchino, K Teshima, and K Takeda. Carrier aggregation enhancement and dual connectivity promising higher throughput and capacity. *NTT Docomo technical Journal*, 2015.

[63] Michele Polese, Marco Giordani, Marco Mezzavilla, Sundeep Rangan, and Michele Zorzi. Improved handover through dual connectivity in 5G mmwave mobile networks. *IEEE Journal on Selected Areas in Communications*, 2017.

[64] Subramanya Chandrashekar, Andreas Maeder, Cinzia Sartori, Thomas Höhne, Benny Vejlgaard, and Devaki Chandramouli. 5G multi-RAT multi-connectivity architecture. In *IEEE ICC*, 2016.

[65] Daniela Laselva, David Lopez-Perez, Mika Rinne, and Tero Henttonen. 3GPP LTE-WLAN aggregation technologies: Functionalities and performance comparison. *IEEE Communications Magazine*, 2018.

[66] Ming Li, Andrey Lukyanenko, Zhonghong Ou, Antti Ylä-Jääski, Sasu Tarkoma, Matthieu Coudron, and Stefano Secci. Multipath transmission for the internet: A survey. *IEEE Communications Surveys & Tutorials*, 2016.

[67] Kun-chan Lan and Chen-Yuan Li. Improving TCP performance over an on-board multi-homed network. In *IEEE WCNC*, 2012.

[68] Pablo Rodriguez, Rajiv Chakravorty, Julian Chesterfield, Ian Pratt, and Suman Banerjee. MAR: A commuter router infrastructure for the mobile Internet. In *ACM MobiSys*, 2004.

[69] Dhananjay S Phatak and Tom Goff. A novel mechanism for data streaming across multiple IP links for improving throughput and reliability in mobile environments. In *IEEE Computer and Communications Societies*, 2002.

[70] RFC 5533: Shim6: Level 3 multihoming shim protocol for IPv6. https://www.rfc-editor.org/rfc/rfc5533.html. (Accessed on 08/19/2022).

[71] Load sharing for the stream control transmission protocol (SCTP). https://datatracker.ietf.org/doc/html/draft-tuexen-tsvwg-sctp-multipath-00. (Accessed on 03/03/2022).

[72] Varun Singh, Saba Ahsan, and Jörg Ott. MPRTP: multipath considerations for real-time media. In *ACM MMSys*, 2013.

[73] Quentin De Coninck and Olivier Bonaventure. Multipath QUIC: Design and evaluation. In *ACM CoNEXT*, 2017.

[74] Tobias Viernickel, Alexander Froemmgen, Amr Rizk, Boris Koldehofe, and Ralf Steinmetz. Multipath QUIC: A deployable multipath transport protocol. In *IEEE ICC*, 2018.

[75] Heekwang Kim and Kwangsue Chung. Multipath-based HTTP adaptive streaming scheme for the 5G network. *IEEE Access*, 2020.