JUNCTION: A Scalable Multi-access Solution using Programmable Switches

Xin Zhe Khooi[1], Cha Hwan Song[2], Satis Kumar Permal[1], Nishant Budhdev[3], Levente Csikor[4], Raj Joshi[5], Mun Choon Chan[1]
[1]National University of Singapore, Singapore [2]Samsung Electronics, Republic of Korea [3]Nokia Bell Labs, Belgium
[4]Institute for Infocomm Research (I²R), A*STAR, Singapore [5]Harvard University, USA

***Abstract*—Multi-access networks are increasingly important for reliable end-to-end connectivity and enhanced throughput performance. A scalable multi-access solution is required to roll out multi-access networks at scale. However, existing CPU-based solutions can no longer scale sustainably, as network traffic has outgrown the CPU performance growth. Consequently, hardware accelerators offer a compelling alternative. This paper introduces JUNCTION, a scalable multi-access solution designed using programmable switches. JUNCTION features a multipath protocol tailored to the hardware constraints and optimized for efficient memory utilization, enabling it to handle a large number of multipath sessions. We validate JUNCTION on a 5G-WiFi multi-access testbed. Our analysis demonstrates that it can scale an order of magnitude better than existing solutions.**

***Index Terms*—multi-access networks, multipath protocol, programmable switches, P4, 5G ATSSS**

I. INTRODUCTION

As demand for high-bandwidth and reliable connectivity from applications like HD video streaming, online gaming, and mission-critical services grows [1], network operators are turning to multi-access networks. These networks increase throughput and reliability by using multiple access networks concurrently [2]–[6]. Multi-homed end-user devices, such as mobile devices combining LTE/5G with WiFi or residential gateways bundling LTE/5G with xDSL/fiber, enable this capability. Fig. 1 depicts an overview of multi-access networks.

A multi-access network deployment requires two primary components: the Multi-Access Gateway (MAG) and the multi-access client on user equipment (UE), e.g., mobile phone. The MAG serves as the convergence point for various access networks and is typically deployed in the central part of a telecommunications network. The multi-access client enables seamless utilization of multiple network interfaces on UEs.

Multipath protocols are essential in multi-access networks, enabling seamless connectivity and aggregated bandwidth by masking the multiple accesses from the user and upper-layer applications. These protocols determine the usage of multiple access networks, whether using a primary path, switching between paths, or using both paths simultaneously. Effective multipath protocols must perform continuous path measurements, dynamically select paths, and reorder out-of-order packets [7, Sec. 2.1].

Most multipath protocols are designed to run on CPUs in commodity servers. To support multi-access at scale, operators today horizontally scale the multi-access gateways (i.e., scale-out) by increasing the number of servers to handle the increased traffic loads. However, such a scale-out approach is becoming unsustainable due to the rapid growth in Internet traffic coupled with the slowdown of CPU performance growth [8]. Consequently, there are efforts to look for alternative scaling solutions beyond CPUs, such as hardware accelerators, to enable the widespread deployment of multi-access networks.

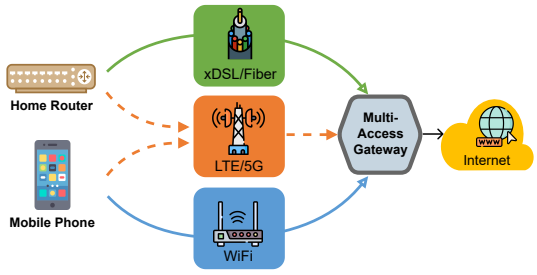


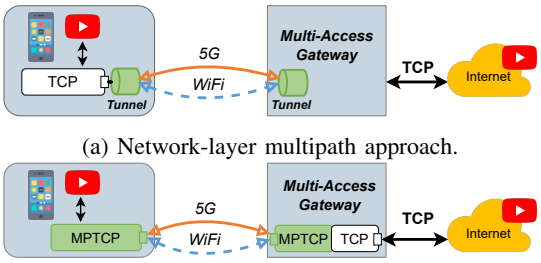
Figure 1: Overview of Multi-Access Networks.

Programmable switches with multi-Tbps line rate packet processing capabilities offer a promising platform for realizing a performant and scalable multi-access solution. This aligns with recent research and industrial trends [8]–[11], which highlight the capabilities of programmable switches for handling large volumes of traffic for customized use cases at scale. However, the restrictive programming model of programmable switches makes it challenging to implement existing multipath protocols designed for CPUs. Additionally, the limited hardware resources make it difficult to maintain large numbers of multipath protocol sessions at the MAG.

In this paper, we present JUNCTION, a scalable multi-access solution using programmable switches to support large-scale multi-access networks. JUNCTION carefully considers the restrictive programming model of programmable switches in realizing a hardware-friendly multipath protocol, which operates at the network layer, together with its programmable switch-based MAG.

Our approaches include:

1) Efficient path measurement (§IV-A): JUNCTION avoids expensive per-user probe generation on programmable switches by using only the UE for active measurements.
2) Packet reordering (§IV-B): JUNCTION uses pause-able hardware FIFO queues to restore packet order when there is path switching and manages path switches so that there are always sufficient buffer resources to store out-of-order



(a) Network-layer multipath approach.

(b) Transport-layer multipath approach.

Figure 2: Overview of existing multipath approaches.

packets.

3) State maintenance (§V): JUNCTION compresses multipath session data and uses a shared data structure to optimize memory use.

JUNCTION consists of three components: JUNCTION-UE, JUNCTION-GW, and JUNCTION-CP. Implemented on an Intel Tofino2 switch, JUNCTION is tested with 5G and WiFi multi-access testbed. Our analysis shows that it achieves up to 46× cost reduction and 93.5× power savings compared to a CPU-based solution. We also discuss its alignment with current telecommunication standards for practical deployment (§IX).

Paper structure: §II reviews the current state of multi-access network deployments and the need for a programmable switch-based solution for large-scale implementations. The challenges in designing such a solution are presented in §III, with our approaches detailed in §IV and §V. We then describe our end-to-end prototype (§VI) and analyze JUNCTION's scalability (§VII). The evaluation of JUNCTION is covered in §VIII. We discuss limitations, and the relationship to 3GPP standards in §IX, and conclude in §XI. Related work is addressed in §X.

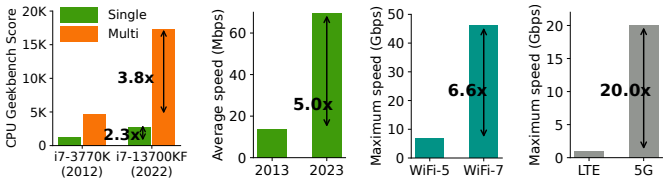
II. BACKGROUND AND MOTIVATION

We provide the background surrounding multi-access networks and the need for a scalable multi-access solution.

A. Standards for multi-access networks

Multi-access networks have grown in popularity over the past decade [2], [6] to provide robust connectivity in areas lacking high-speed fiber. This is achieved by combining copper-based xDSL with LTE in an over-the-top (OTT) approach [2], requiring specialized home routers and a separate MAG at the operator end. The Broadband Forum sets the standards for these hybrid access deployments [12], [13].

Recently, Access Traffic Steering, Switching, and Splitting (ATSSS) has been standardized in 3GPP Release 16 [14]. This allows the integration of non-3GPP networks (e.g., WiFi, fiber, satellite) with 3GPP networks (e.g., 5G) for simultaneous use. ATSSS provides a unified framework for multi-access deployment, eliminating the need for previous OTT methods. It integrates with the 5G core network and supports multi-access natively on end-user devices. Major carriers are actively exploring ATSSS for multi-access rollouts [4], [5], [15].



(a) CPU (b) Broadband (c) WiFi (d) LTE/5G

Figure 3: The growth in CPU performance, average broadband speed [19], [20], and peak throughput of WiFi [21] and cellular technologies [22] over the past decade.

B. Multipath protocols for multi-access networks

Multipath protocols are crucial for multi-access solutions [12], [14], with common examples being bonded-GRE [16], MPTCP [17], and MPQUIC [18]. These protocols fall into two categories based on their TCP/IP model layer: (1) network-layer (e.g., GRE), which uses overlay tunneling to mask multiple access networks from applications, and (2) transport-layer (e.g., MPTCP, MPQUIC), which requires applications to use multipath sockets. Overlay tunnels can also be implemented using transport-layer protocols, but we categorized them as network-layer approaches, similar to [7, Table 1]. These approaches are illustrated in Fig. 2.

The main differences between network-layer and transport-layer multipath solutions are: (1) transport-layer solutions are more complex due to their inclusion of reliable delivery and congestion control, while network-layer solutions delegate these tasks to upper layers (e.g., TCP), and (2) transport-layer approaches require application modifications, whereas network-layer approaches do not.

Multipath protocols perform dynamic path selection. To do so, multipath protocols conduct continuous path measurements to monitor metrics such as path latency. The path selection mechanism can be broadly categorized into three schemes: (1) using one path as the primary path and the other as a failover, such as active-standby or cheapest-path-first [7]; (2) using one path at a time, but continuously switching between available paths based on which is better, such as lowest-latency-first; and (3) using both paths simultaneously, where traffic is sent over both paths to combine their throughput, or load balance traffic. Finally, multipath protocols also require packet reordering to handle out-of-order arrivals and ensure application performance [7, Sec. 2.1].

C. Need for scalable approach for multi-access

Over the past decade, peak throughputs for WiFi and LTE/5G have increased by over 6.6× and 20×, respectively (Fig. 3c and Fig. 3d). Mobile device density has risen by 41% per 100 people [23], and connected users have increased by over 35% [24]. Broadband speeds have grown by more than 5× over ten years [19], [20], [25] (Fig. 3b).

In contrast, single-core and multi-core CPU performance has only increased by 2.3× and 3.8×, respectively (Fig. 3a). This slower growth rate is insufficient to meet the demands of increased user throughput and traffic volume, leading to a

need for more CPU cores and servers to handle the traffic. As highlighted by Pan et al. [8, Sec. 2.3], this disparity in growth has made current CPU performance a bottleneck for handling network traffic at scale. Thus, scaling up or out compute resources is no longer a sustainable and practical solution for deploying multi-access solutions at large-scale.

D. The case for programmable switches

Hardware accelerators offer a scalable solution beyond traditional CPUs. We select programmable switches due to their multi-Tbps packet processing capabilities. Programmable switches can execute data plane programs at line rate with low, deterministic latency [26]. Network functions on programmable switches are more power- and cost-efficient compared to CPU-based software implementations [8], [9].

Previous studies highlight the advantages of programmable switches: SilkRoad [9] reported up to 500× and 250× of power and cost reduction, while SailFish [8] showed a 95% latency improvement. This has led to the growing use of programmable switches in data centers and mobile networks [8]–[11]. Thus, we choose programmable switches to realize a scalable multi-access solution given its scalability potential.

III. JUNCTION: A SCALABLE MULTI-ACCESS SOLUTION

Programmable switches provide excellent throughput, cost, and power efficiency but have restrictive pipeline programming models, rigid (stage-local) memory access patterns, and limited hardware resources (e.g., memory) [27, Sec. 2]. This section outlines the challenges of implementing a scalable multi-access solution and provides an overview of our approach.

A. Challenges

Multipath protocol must be amenable to programmable switches: To devise a multipath protocol suitable for programmable switches, we adopt a network-layer approach with three essential core components: (1) continuous path measurements, (2) packet reordering, and (3) dynamic path selection. This approach avoids transport-layer mechanisms like congestion control and retransmission, which are impractical due to the limited buffer space and restrictive memory access patterns in programmable switches. Despite so, it is non-trivial to realize (1) and (2) on a programmable switch.

Continuous path measurements: Continuous path measurements require probing, but generating probe packets on programmable switches is inefficient. This inefficiency arises from: (1) memory overhead: maintaining individual timers for each user consumes additional memory, and (2) inefficient memory accesses: memory scans to determine which multipath session or user to probe are slow, as memory regions can only be accessed once by each packet, thus requiring N packets to iterate through an array of size N users. Additionally, generating dedicated probes consumes extra bandwidth. To minimize overhead, probes must be frequent enough to capture path-level changes, such as link availability or latency.

Handling out-of-order packets: Packets in a multipath session can arrive out-of-order and must be reordered before forwarding. This requires buffering and sorting out-of-order packets. However, programmable switches lack native support for selective buffering and sorting in the data plane [28], and their limited buffer capacity further complicates this process.

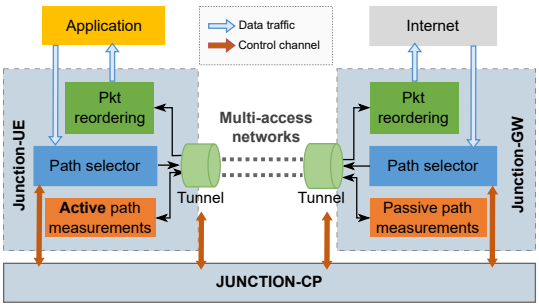


Figure 4: JUNCTION end-to-end solution overview. The JUNCTION-UE and JUNCTION-GW communicate using the JUNCTION multipath protocol using an overlay tunnel. Each tunnel can have several multipath sessions (not shown here).

MAG must be scalable: The programmable switch serving as the MAG must manage large number of users and multipath sessions. The limited memory on the switch must be used efficiently to maintain per-session states, such as tunnel endpoint information, sequence numbers, and path measurement data likew availability and latency. This careful memory management is crucial to ensure the solution scales effectively, minimizing the number of switches needed to support a given user base and traffic volume.

B. Overview of JUNCTION

We propose JUNCTION, a scalable multipath solution using programmable switches. A JUNCTION end-to-end solution consists of the following components: (1) JUNCTION-UE, (2) JUNCTION-GW, and (3) JUNCTION-CP. The JUNCTION multipath protocol runs between the JUNCTION-UE and JUNCTION-GW. An overview of JUNCTION's components and their corresponding responsibilities is shown in Fig. 4.

Scope: As our focus is on the realization of a scalable multipath solution, we emphasize the programmable switch hardware-specific design choices for the JUNCTION multipath protocol (§IV) and the JUNCTION-GW (§V), addressing the challenges as mentioned earlier in §III-A.

Henceforth, we will not cover the control plane procedures and the design details of JUNCTION-UE and JUNCTION-CP, which are CPU-based. At a high level, JUNCTION adapts the multipath tunnel setup procedures from [16, Sec. 5], where the JUNCTION-UE interacts with the JUNCTION-CP to retrieve the network configuration such as tunnel endpoint addresses and path selection schemes. At the same time, the JUNCTION-CP manages the JUNCTION-GW data plane rules.

IV. HARDWARE-FRIENDLY MULTIPATH PROTOCOL

JUNCTION is a network-layer multipath solution, and thus the JUNCTION multipath protocol uses an overlay tunneling approach to tunnel network traffic between the JUNCTION-UE and JUNCTION-GW over multiple access networks.

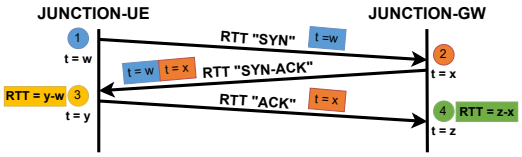


Figure 5: RTT measurement process.

The JUNCTION multipath protocol manages path measurements, dynamic path selection, and packet reordering on both JUNCTION-UE and JUNCTION-GW.

Our design aligns with the capabilities of programmable switches. Specifically, we depend on the JUNCTION-UE to actively conduct path measurements instead of the programmable switch (§IV-A), managing path switching at the JUNCTION-UE and using pause-able FIFO queues to reorder out-of-order packets at the JUNCTION-GW (§IV-B).

A. Path measurements and dynamic path selection

1) Path measurements: The path measurement process involves: (a) RTT measurements, and (b) link loss notification.

a) RTT measurements: As illustrated in Fig. 5, the RTT measurements resemble the TCP three-way handshake. It is conducted for all available access networks. Initiated by the JUNCTION-UE, this process minimizes switch processing by having the switch react only to probe packets sent by the JUNCTION-UE. This approach avoids the need for maintaining per-session timers and generating probes on the switch.

The JUNCTION-UE uses a configurable timer to set the frequency of path measurements. It piggybacks measurements on data packets whenever possible. If no data packets are available and the network is idle, the JUNCTION-UE generates dedicated RTT probes. For each timer interval, the JUNCTION-UE checks for outgoing packets in the buffer; if present, it tags them with an RTT "SYN" for measurement. If no packets are available, a dedicated RTT probe is sent.

When the JUNCTION-GW detects an RTT "SYN", it responds with an RTT "SYN-ACK" and its timestamp. The JUNCTION-UE calculates the RTT at ③, while the JUNCTION-GW computes the RTT at ④ upon receiving the JUNCTION-UE's RTT "ACK" response. The measured path conditions are updated directly in the data plane.

b) Link loss notification: The JUNCTION-UE detects link failures by monitoring RTT measurement timeouts. A link is deemed unavailable if it experiences more than three RTT timeouts. Upon detecting a failure, the JUNCTION-GW updates the data plane to stop using the affected link immediately. This failure information is also communicated from the JUNCTION-UE to the JUNCTION-CP (§VI-C).

2) Dynamic path selection: Based on path measurements, both JUNCTION-UE and JUNCTION-GW enforce the appropriate path selection scheme for packets in the uplink and downlink, respectively. Examples of these schemes include selecting the primary path, choosing the best path, or utilizing both available paths.

B. Preserving packet order

Implementing packet reordering in the downlink on the JUNCTION-UE is straightforward since it runs on CPUs. However, in the uplink, programmable switches face challenges due to their lack of support for selective buffering and limited buffer memory, making packet reordering more difficult.

The JUNCTION protocol addresses these challenges with the following techniques: (1) FIFO queue pausing: the JUNCTION-GW uses pause-able FIFO queues to buffer and restore order for out-of-order packets effectively, and (2) temporary buffering: when FIFO queues at the JUNCTION-GW are unavailable, the JUNCTION-UE temporarily buffers outgoing packets to manage transitions between high-latency and low-latency links, reducing out-of-order packets.

This reordering mechanism is limited to scenarios without arbitrary out-of-order arrivals (§IV-B1). Consequently, JUNCTION only supports the use of one path in the uplink. This limitation is acceptable given the typically lower bandwidth requirements in the uplink compared to the downlink [1].

We now discuss the rationale behind this design and provide a detailed explanation of the chosen techniques.

1) Putting packets back in order: Packet recirculation, as discussed in literature such as [29], is a straightforward method for holding packets on programmable switches until the next expected packet arrives. However, it has several drawbacks: (1) inefficiency: packet recirculation consumes valuable packet processing capacity, and (2) indeterminate ordering: packets are inserted randomly during recirculation, leading to non-deterministic dequeuing due to their scattered arrangement. This can cause delays as the next expected packet may need to wait for recirculation.

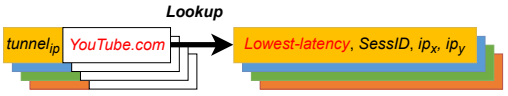
a) Packet reordering with FIFO queues: Drawing from insights in ConWeave [28, Sec. 2], we implement a packet reordering mechanism using pause-able FIFO queues on modern programmable switches. We reorder packets based on whether they were transmitted *before* ($path_{old}$) or *after* ($path_{new}$) a path switch, rather than by their sequence numbers. This approach avoids expensive packet recirculations.

When a path switch occurs, packets are redirected from $path_{old}$ to $path_{new}$. Out-of-order arrival can only happen when packets on $path_{new}$ reach JUNCTION-GW before those on $path_{old}$. Thus, we use a paused FIFO queue to buffer the $path_{new}$ packets until all packets from $path_{old}$ have arrived.

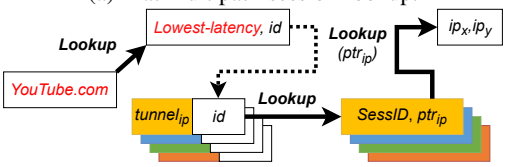
To determine when all packets have arrived, JUNCTION-UE tags the last packet sent over $path_{old}$ as the TAIL. Once the JUNCTION-GW receives the TAIL, the paused FIFO queue releases the buffered packets from $path_{new}$.

How to reserve a FIFO queue? Before initiating a path switch, the JUNCTION-UE reserves a FIFO queue from the JUNCTION-GW using a 1-RTT exchange. The JUNCTION-UE piggybacks a data packet with a S_REQ flag and awaits a S_RESP from the JUNCTION-GW. The path switch proceeds only after receiving the S_RESP.

How to handle TAIL *losses?* To handle TAIL losses, we implement a reordering timeout. If the TAIL packet does



(a) Flat multipath session lookup.



(b) Multi-stage multipath session lookup.

Figure 6: Flat versus multi-stage multipath session lookups.

not arrive within this timeout, the buffered packets from $path_{new}$ are released to maintain JUNCTION's functionality. The timeout duration adapts to the measured path latency.

2) Delayed path switching: While statistical multiplexing ensures that the required number of queues does not scale linearly with the number of active user sessions, FIFO queues for reordering may not be available given that the JUNCTION-GW has a limited number of FIFO queues available. If no queues are available or if the S_RESP is lost, the JUNCTION-UE estimates whether $path_{new}$ packets will arrive at the switch before $path_{old}$. If so, the JUNCTION-UE buffers the packets for approximately half the RTT of the old path before releasing them on the new path. This approach leverages path status information obtained from the path measurement process (see §IV-B1a) to minimize out-of-order packets.

V. Efficient Memory Usage on Hardware

To efficiently support a large number of JUNCTION multipath sessions at the JUNCTION-GW within the limited memory of programmable switches, we use a multi-stage lookup structure and a common representation.

A. Multi-stage multipath session lookup table

We use downlink packet lookup to illustrate our approach for managing multipath sessions efficiently. In JUNCTION, each multipath tunnel supports multiple sessions for different application types, often sharing path selection schemes.

A naive approach would require $N \times M$ entries for N sessions and M applications, which is memory-inefficient (see Fig. 6a). Instead, we reduce memory usage by decoupling [30] application traffic types from tunnel endpoint addresses, using shorter representations like id and ptr_{ip}.

We first look up the application traffic type to determine the path selection scheme and session ID, then use this ID to retrieve tunnel endpoint addresses. This multi-stage lookup fits well with the programmable switch's pipeline model. By adopting this method, we significantly reduce memory requirements. We quantify the memory savings later in §VIII-D2. For sessions that cannot be decoupled, we use a small supplementary lookup table (e.g., provisioned for 10% of the total number of users) to manage exceptions efficiently.

B. Common representation different path selection schemes

To minimize memory usage, we use a shared data structure with a common representation for the different path selection schemes. For binary schemes (e.g., failover), a simple 0 or 1 suffices, while schemes requiring RTT path selection use non-zero integers. Since RTT measurements assume network availability, one set of path states per user is sufficient. We utilize two integer arrays to maintain path states for available access networks. The multipath session lookup retrieves the index, ptr_{ip}, to retrieve the path states. This approach reduces memory usage from $N \times M$ entries to N.

VI. End-to-end Junction Prototype

Packet header layout: As JUNCTION employs a network-layer approach with overlay tunneling, we encapsulate the user packet with an IPv4 (20 bytes) and JUNCTION header (12 bytes), adding a total overhead of 32 bytes. The JUNCTION header includes a 4-bit flag for packet type, a 4-bit tunnel session ID, a 1-byte protocol field, a 2-byte length field, a 4-byte timestamp, and a 4-byte sequence number.

A. JUNCTION-UE

JUNCTION-UE is as a multi-threaded userspace client application (∼800 lines of C++) using Linux TUN/TAP. JUNCTION-UE manages the multipath tunnel and individual sessions, with multiplexing achieved via the tunnel session ID.

B. JUNCTION-GW

We prototype[1] the JUNCTION-GW on an Intel Tofino2 switch using ∼1300 lines of P4 [31], built using the Intel P4 Studio v9.11.1. We use match-action tables to implement the multi-stage lookup table (§V-A) and two 8-bit register arrays for the data structure (§V-B) to represent the path states. We update the computed path RTTs and link loss notifications directly to the register arrays (§IV-A). This enables the JUNCTION-GW to quickly react to network events. Lastly, we use 32-bit registers to keep track of the indirection pointers and sequence numbers.

As path measurements and link loss notifications are directly updated in the JUNCTION-GW's data plane, situations may arise where the measured RTT of the links experiences "flip-flopping" leading to frequent path switching. To address this, the JUNCTION-GW forwards the path measurements to the JUNCTION-CP to monitor the received measurement and notification packets (§VI-C). If instability is detected, the JUNCTION-CP suspends direct updates in the data plane for the specific multipath session until the network stabilizes. This approach ensures a more reliable path selection.

For dynamic path selection, the common path state representation and sequence number are used as follows:

1) single link usage: for active-standby failover, a primary and backup path is defined. Traffic continues on the primary path if its RTT is non-zero; otherwise, it switches

[1]https://github.com/NUS-CIR/JUNCTION.

Table I: Power and cost analysis against a CPU-based system used in [33]. We assume a per-user throughput of 100 Mbps.

	# servers required	cost ratio	power ratio
FlexCP [33]	92	46	93.5
L3-IDEAL	64	32	65.1

to the backup path. For lowest latency-first, traffic is forwarded on the path with the smallest non-zero RTT.

2) dual link usage: packets are routed based on the rightmost digit of their sequence numbers. For instance, to equally split packets over both links, packets with digits 0-7 go over the first path, and those with digits 8-15 go over the other. Path availability is checked before forwarding.

To reorder packets, we utilize the queue pause/resume feature on the Intel Tofino2. We reserve $N-1$ queues out of the N available queues from an upstream port (e.g., 127 out of 128 queues [28] for a 400 Gbps port). We use two-way associative hash tables with register arrays to track and manage these queues. When a queue is flushed, it is updated and freed for future use. To handle lost TAIL packets, a continuously recirculated "timer" packet is used. The "timer" packet is generated upon the receipt of the first out-of-order packet, it is discarded upon queue resumption or expiry.

C. JUNCTION-CP

JUNCTION-CP is implemented in ∼300 lines of Python, managing JUNCTION-specific components and interacting with the JUNCTION-UE for multipath session management and JUNCTION-GW data plane configuration (§V-A). It monitors RTT "ACKs" to detect and address frequent path switching due to instability by suspending data plane updates until conditions stabilize (§VI-B). JUNCTION-CP can be further optimized with fast control channels such as [32].

VII. SCALABILITY ANALYSIS

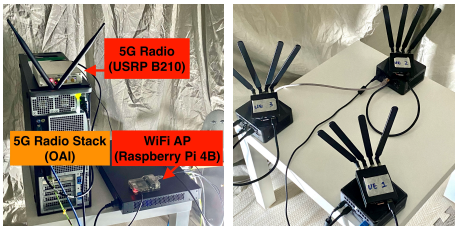
We analyze the scalability of a JUNCTION-based solution in terms of throughput.

Comparisons: We use FlexCP [33], a state-of-the-art hardware-assisted MPTCP proxy, as a reference. FlexCP supports up to 140 Gbps (281 Gbps bidirectional) with a dual-port 100 Gbps NVIDIA BlueField2 DPU [34]. Despite FlexCP being a transport-layer solution and different from the network-layer JUNCTION, our goal here is to demonstrate the significant scalability benefits of a simpler network-layer solution that can be implemented fully on a programmable switch as compared to CPU-based solutions. We also consider an ideal case (L3-IDEAL) where the same system in [33] achieves full 200 Gbps throughput with a network-layer implementation.

Analysis: First, a single Intel Tofino2 switch (12.8 Tbps) running JUNCTION achieves raw throughput comparable to 92 servers running FlexCP and 64 servers running L3-IDEAL. This illustrates the baseline scalability of JUNCTION compared to CPU-based solutions. Assuming a per-user throughput of 100 Mbps [25], a single Tofino2 switch can support up to 128K users. We perform a power-cost analysis comparing the CPU-based systems used in FlexCP and L3-IDEAL to the Tofino2-based JUNCTION. The CPU-based system, consisting of a 24-core Intel Xeon Gold 6342 [35] and an NVIDIA BlueField2 DPU [34], costs approximately 5K USD and consumes 305 Watts. In contrast, the Tofino2 switch costs about 10K USD and consumes 300 Watts[2]. From Table I, we can see that the cost to support the same number of users would be up to 92 times higher for FlexCP and 64 times higher for L3-IDEAL compared to the Tofino2 switch. This translates to 46× (32×) more in the hardware acquisition cost for the Tofino2 switch alone, and 93.5× (65.1×) increase in power consumption, not to mention additional rack space required to host them in contrast to the 1 rack-unit required by the Tofino2 switch in JUNCTION. This analysis highlights the significant potential for cost and power savings of JUNCTION.

Table II: Testbed hardware and software configuration.

5G Basestation	USRP B210; 20MHz SISO @ 3.5GHz
WiFi Access Point	Rasp. Pi 4B; WiFi5 20MHz SISO @ 5GHz
Multi-homed UEs (3x)	Quectel RM500Q-GL & Intel AX211
Servers (2x)	Intel Xeon Gold 6326;256GB RAM
Ethernet NICs (2x)	NVIDIA ConnectX-6 DX 100GbE NIC
Ethernet Switch	Intel Tofino2 [37] switch
Linux Kernel	Linux Kernel v5.15-realtime
5G Radio Stack	OpenAirInterface5G (tag 2024.w15)
5G Core	OpenAirInterface5G SA Core (v1.5.1)



(a) 5G gNB, WiFi AP. (b) Multi-homed UEs.

Figure 7: Our multi-access testbed in an EMI/RFI-shielded tent. The UEs are 1m away from the 5G gNB and WiFi AP.

VIII. EVALUATION

We validate JUNCTION using a 5G and WiFi multi-access testbed. The testbed configuration is summarized in Table II, and Fig. 7 shows a picture of the testbed. In this testbed, the end-to-end median latency with minimal traffic is 20 ms for 5G and 5 ms for WiFi accesses. The path measurement interval on the JUNCTION-UE is set to 200 ms.

Overview: We validate JUNCTION across various use cases, demonstrating its ability to leverage multiple accesses with different path selection schemes (see §VIII-A to §VIII-C). We also conduct ablation studies to highlight the importance of the design decisions incorporated in JUNCTION (§VIII-D).

A. Transparent Path Migration

Experiment setup: We demonstrate that JUNCTION can maintain user Quality of Experience (QoE) for live video

[2]Pricing and power consumption details are based on publicly available sources [9], [35]–[37].

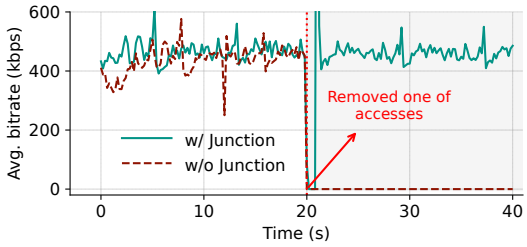


Figure 8: Average DL bitrate measured with a video-conferencing app. At $t = 20s$, we disable one of the accesses.

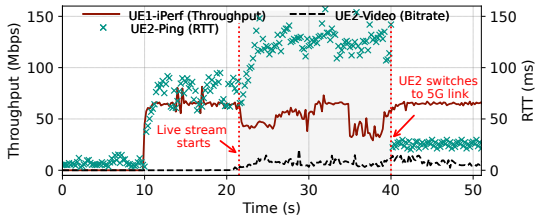


Figure 9: Throughput reported by UE1 and UE2 for iPerf3 and live video streaming, respectively.

conferencing by seamlessly transitioning between access links using the UDP-based ringmaster [38] framework.

In our setup, a video sender transmits a compressed talking-head video at a targeted bitrate of 500 kbps to the UE, and we assess the average video bitrate at the receiver, which directly impacts QoE and recovery times [39]. The UE is connected to both 5G and WiFi accesses, with WiFi as the primary access during the video session.

We compare the video bitrate variations when the UE loses its primary access network *with and without* JUNCTION. The video bitrate is recorded every 200 ms.

Results: Fig. 8 shows the measured video bitrate in the downlink direction. At $t = 20$ s, the WiFi network is disabled.

Without JUNCTION, the UE loses the connection. The application would have to be restarted to reconnect. In contrast, JUNCTION causes only a brief bitrate dip of approximately 600 ms before recovery, reflecting the link failure detection time at the JUNCTION-UE (see §IV-A). During this time, the JUNCTION-GW switches to the 5G access.

This experiment highlights JUNCTION's ability to maintain application resilience during network events. Similar results are observed when the 5G network is primary and in the uplink direction, but are omitted for brevity.

B. Dynamic Path Selection

Experiment setup: Next, we demonstrate JUNCTION's responsiveness to changing path conditions using two UEs: UE1 and UE2, both connected to the 5G and WiFi networks. As the control, UE1 runs an iPerf3 downlink session using the WiFi network. We run a background ping session on UE2. Then, we start a *live* high-bitrate video stream using FFMPEG and VLC in the downlink on UE2 using WiFi to compete with UE1. We later set the path selection scheme for UE2 in the downlink to dynamically choose the path with the lowest latency irrespective of the application traffic. This setup illustrates JUNCTION's ability to adaptively select the optimal path based on real-time latency measurements.

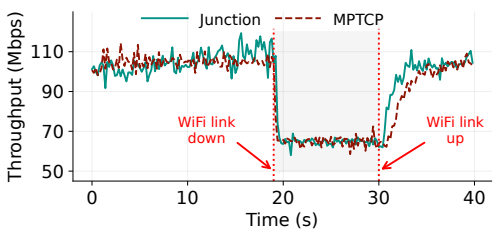


Figure 10: Throughput reported by iPerf3. Both accesses are used to achieve aggregated throughput.

Results: We illustrate JUNCTION's dynamic path selection capabilities in Fig. 9, showing reported throughput for both UE1 and UE2 and ping latency from UE2:

- before competition ($t < 21s$): UE2 experiences latency of $<$10 ms, before UE1 starts the iPerf session at $t = 10s$. Latency increases to $>$50 ms as UE1's traffic begins.
- traffic competition ($21s \leq t < 40s$): when UE2 starts its high-bitrate video stream at around $t = 21s$, UE1's throughput drops due to increased competition. The WiFi path's latency spikes to $>$100 ms due to congestion.
- path selection activation and switching ($t \geq 40s$): at $t = 40s$, we enable the lowest latency path selection for UE2. The JUNCTION-GW switches UE2's traffic to the 5G network based on real-time latency measurements. This results in an immediate recovery of UE1's throughput and a significant decrease in UE2's latency to $\sim$30 ms.

This demonstrates JUNCTION 's ability to adaptively switch paths to maintain performance and minimize latency.

C. Bandwidth Aggregation

Experiment setup: We demonstrate JUNCTION's ability to aggregate multiple accesses for bandwidth-intensive applications. We use iPerf3 to simulate large file downloads. We test by disabling and re-enabling one access network to observe JUNCTION-GW's response. For comparison, we contrast JUNCTION with the widely used MPTCP [17].

Results: From Fig. 10, we observe when both the accesses are used individually, each reaches a throughput of approximately 65 Mbps in the downlink. When both networks are available, the aggregated throughput for JUNCTION and MPTCP is roughly 105 Mbps. When WiFi is disabled between $t = 20 - 30s$, throughput drops to about 65 Mbps. Upon re-enabling WiFi at $t = 30s$, throughput recovers to 105 Mbps for both JUNCTION and MPTCP. This illustrates that JUNCTION achieves aggregated bandwidth comparable to MPTCP.

D. Ablation Study

1) Uplink packet reordering at JUNCTION-GW*:* We demonstrate the importance and effectiveness of JUNCTION's hardware-friendly packet reordering mechanism (§IV-B) for the uplink. We generate uplink traffic by uploading 7 MB

Table III: The number of TCP retransmissions w/ and w/o the uplink packet reordering engine (§IV-B) at JUNCTION-GW.

	average	99th-%ile
w/o Packet Reordering Engine	186.8	219.0
w/ Packet Reordering Engine	1.2	3.0

Table IV: Normalized total number of supported users with and without our proposed optimizations in §V-A and §V-B.

Per-user multipath sessions	Naive	MMLT only	CR only	JUNCTION
1	1.00	1.22	1.00	1.23
2	1.00	1.22	1.13	1.45
4	1.00	1.22	1.21	1.56
8	1.00	1.22	1.25	1.63

short videos. For this experiment, we disable the dynamic path selection mechanism in JUNCTION and configure the JUNCTION-UE to switch between accesses every 2000 packets. The experiment is repeated 100 times.

We measure TCP retransmissions, as severe packet reordering can cause packets to be misinterpreted as lost, leading to reduced sending rates and degraded application performance. Our experiment results in Table III show that enabling the packet reordering mechanism in JUNCTION-GW reduces TCP retransmissions by $155.7\times$.

2) Memory Savings: We quantify how the multi-stage multipath lookup table (MMLT) (§V-A) and common representation (CR) (§V-B) impact the total number of users that can be supported on the JUNCTION-GW. Using a simplified data plane program to maximize user support while saturating available memory, we vary the number of per-user multipath sessions (up to 8) along with the proposed optimizations.

Results in Table IV show that our optimizations enable JUNCTION to support at least $23\,\%$ more users than a naive approach, regardless of the number of multipath sessions per user. For configurations with multiple multipath sessions per user, improvements of up to $63\,\%$ are observed.

IX. DISCUSSION AND FUTURE WORK

Limitations: The JUNCTION multipath protocol can affect upper transport layer congestion control (CC) estimations, particularly when switching between access networks with varying characteristics. Future work will explore tuning per-path CC behavior, inspired by TD-TCP [40]. Besides, given the lack of support for cryptographic primitives on our prototyping platform (Intel Tofino2), our current prototype assumes that the access networks are secure and trusted (e.g., 5G PDCP ciphering [41]). While cryptographic algorithms can be implemented on programmable switches such as [42], they require substantial hardware resources. We expect that future hardware will support such features, as suggested by [43].

Mapping JUNCTION to 3GPP 5G ATSSS: As a network-layer approach, the JUNCTION multipath protocol falls under the ATSSS-LL realm (see Fig. 11). ATSSS-LL can be deployed standalone or used to complement existing standard-

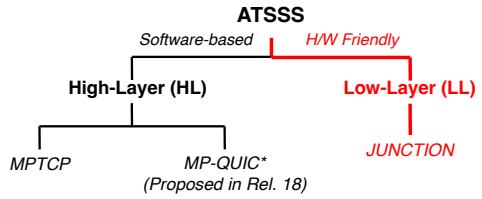


Figure 11: Approaches in 3GPP ATSSS – ATSSS-HL and -LL. JUNCTION falls under the realm of ATSSS-LL.

Table V: JUNCTION-UPF's hardware resource consumption on the Intel Tofino2. We extend P4UPF [10].

Resource	P4UPF [10]	JUNCTION-UPF
SRAM	38.4%	52.8%
TCAM	10.4%	10.4%
Stateful ALUs	21.3%	26.3%
VLIW Ins.	11.7%	12.2%

ized ATSSS-HL approaches, such as MPTCP. JUNCTION's design supports key traffic steering modes in ATSSS, including *active-standby*, *smallest-delay*, and *load-balance* (downlink only, see §IV-B1). Additionally, JUNCTION's continuous path measurement mechanism (§IV-A) aligns with the Performance Measurement Function in ATSSS as outlined in [14].

As the ATSSS function is part of the 5G user-plane function (UPF), we integrated JUNCTION-GW with P4UPF [10] to demonstrate feasibility, calling it JUNCTION-UPF. Table V shows the hardware resource utilization of JUNCTION-UPF on Intel Tofino2 [37], compared to P4UPF [10]. This setup supports 512K GTP sessions, translating to 256K users with up to 2 ATSSS multipath sessions each, connected over 5G and one additional non-3GPP network.

JUNCTION can be deployed as an over-the-top (OTT) solution outside the 5G core. For wider adoption, JUNCTION needs to be standardized and supported in UE modem firmware, reducing the barrier to entry with native support. We hope JUNCTION will raise interest in ATSSS-LL approaches.

X. RELATED WORK

Multi-access solutions: Various existing known multi-access solutions and proposals [33], [44]–[46] are built using existing multipath protocols (such as MPTCP [17], MPQUIC [18]) for which they are designed to run on commodity servers. JUNCTION differs from them and is the first multi-access solution designed using highly scalable programmable switches with a programmable-switch-friendly multipath protocol.

Using multiple accesses to improve performance: [7], [47]–[49] proposed new schemes for path selection over multiple access networks to improve application performance. These works are orthogonal to JUNCTION which focuses on the scalability of the system as a whole. Nevertheless, JUNCTION presents a platform to explore and design new path selection schemes that can then be deployed at scale.

WiFi offloading: Beyond hybrid access networks in §II, 3GPP has leveraged multiple accesses since 3G [50] and

LTE [51], primarily for offloading cellular traffic via WiFi, which required specialized hardware. JUNCTION expands path selection options beyond offloading and can be deployed without hardware changes, regardless of network type. It also aligns with emerging 3GPP 5G ATSSS requirements for potential deployment (see §IX).

XI. CONCLUSION

We demonstrate a hardware-first approach to designing a scalable multi-access solution, JUNCTION. By co-designing a multipath protocol aligned with hardware constraints, it enables the realization of a scalable multi-access gateway. Validated on a 5G and WiFi-based multi-access testbed, JUNCTION enables reliable connectivity and aggregated throughput. As a programmable switch-based system, JUNCTION significantly reduces CapEx and OpEx, showcasing benefits that outweigh its current limitations. We believe JUNCTION can be pivotal in enabling large-scale multi-access deployments.

ACKNOWLEDGEMENT

We thank the reviewers for their invaluable feedback. We also thank Archit Bhatnagar and Biqing Qiu for their suggestions on earlier drafts. This research is supported by the National Research Foundation, Singapore, and Infocomm Media Development Authority under its Future Communications Research & Development Programme.

REFERENCES

[1] Ericsson, "Ericsson Mobility Report June 2023," Telefonaktiebolaget LM Ericsson, Tech. Rep., June 2023.
[2] Proximus, "Tessares-Proximus' Access Bonding," https://tinyurl.com/24fwm6bj [Accessed: Apr 2024], 2017.
[3] Telecom, "Korea Telecom and Tessares claim 5G Low Latency Multi-Radio Access Technology first," https://tinyurl.com/448847z3 [Accessed: July 2023], 2019.
[4] Deutsche Telekom AG, "Deutsche Telekom demonstrates Multipath for Fixed Mobile Convergence on Campus," https://tinyurl.com/mr39w2w8 [Accessed: July 2023], May 2021.
[5] Deutsche Telekom AG, "ATSSS: Seamless Multi-Access Connectivity for Converged 5G/WIFI," https://tinyurl.com/57th7vcd [Accessed: 2024], 2024.
[6] N. Keukeleire *et al.*, "Increasing Broadband Reach with Hybrid Access Networks," *IEEE Comm. Standards Magazine*, vol. 4, no. 1, 2020.
[7] M. Amend *et al.*, "Cost-efficient Multipath Scheduling of Video-on-demand Traffic for the 5G ATSSS Splitting Function," *Computer Networks*, vol. 242, 2024.
[8] T. Pan *et al.*, "Sailfish: Accelerating Cloud-scale Multi-tenant Multi-service GW with Programmable Switches," in *ACM SIGCOMM*, 2021.
[9] R. Miao *et al.*, "Silkroad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching Asics," in *ACM SIGCOMM*, 2017.
[10] R. MacDavid *et al.*, "A P4-Based 5G User Plane Function," in *ACM SOSR*, 2021.
[11] T. Pan *et al.*, "LuoShen: A Hyper-Converged Programmable Gateway for Multi-Tenant Multi-Service Edge Clouds," in *USENIX NSDI*, 2024.
[12] Broadband Forum, "TR-348 Hybrid Access Broadband Network Architecture," Tech. Rep., 2016, https://tinyurl.com/3mtkdyu6.
[13] Broadband Forum, "TR-378 Nodal Requirements for Hybrid Access Broadband Networks," Tech. Rep., 2018, https://tinyurl.com/mtypey9b.
[14] 3GPP, "TS 24.193 v16.4.0: 5G System;Access Traffic Steering, Switching and Splitting (ATSSS); Stage 3," 2021.
[15] MediaTek Inc., "MediaTek to Showcase 5G, Satellite Communications, Computing and Connectivity Technology Advancements at MWC," https://tinyurl.com/txn5ehhs [Accessed: July 2023], February 2023.
[16] N. Leymann *et al.*, "Huawei's GRE Tunnel Bonding Protocol," RFC 8157, May 2017.
[17] A. Ford *et al.*, "TCP Extensions for Multipath Operation with Multiple Addresses," RFC 8684, Mar. 2020.
[18] Q. De Coninck and O. Bonaventure, "Multipath QUIC: Design and Evaluation," in *ACM CoNEXT*, 2017.
[19] OFCOM, "UK Fixed-line Broadband Performance, November 2013," https://tinyurl.com/ynhn7nry [Accessed: May 2024], 2013.
[20] OFCOM, "UK Fixed-line Broadband Performance, September 2023," https://tinyurl.com/3ujjhauc [Accessed: May 2024], 2023.
[21] E. Oughton *et al.*, "Reviewing Wireless Broadband Technologies in the Peak Smartphone Era: 6G versus Wi-Fi 7 and 8," *Telecommunications Policy*, vol. 48, no. 6, p. 102766, 2024.
[22] ITU, "ITU-R FAQ on INTERNATIONAL TELECOMMUNICATIONS (IMT)," https://tinyurl.com/yf7x5bpu [Accessed: Mar. 2023].
[23] T. W. Bank, "Mobile Cellular Subscriptions (per 100 people)," https://data.worldbank.org/indicator/IT.CEL.SETS.P2, [Accessed: July 2023].
[24] Cisco, "Cisco Annual Internet Report (2018–2023)," Cisco Systems Inc., Tech. Rep., 9 March 2020.
[25] FCC, "FCC INCREASES BROADBAND SPEED BENCHMARK," https://tinyurl.com/bdef8f8j [Accessed: May 2024], 2024.
[26] X. Jin *et al.*, "Netcache: Balancing Key-Value Stores with Fast In-Network Caching," in *ACM SOSP*, 2017.
[27] X. Z. Khooi *et al.*, "Revisiting Heavy-Hitter Detection on Commodity Programmable Switches," in *IEEE NetSoft*, 2021.
[28] C. Song *et al.*, "Network Load Balancing with In-network Reordering Support for RDMA," in *ACM SIGCOMM*, 2023.
[29] R. Joshi *et al.*, "Masking Corruption Packet Losses in Datacenter Networks with Link-local Retransmission," in *ACM SIGCOMM*, 2023.
[30] C. Jung, S. Kim, R. Jang, D. Mohaisen, and D. Nyang, "A Scalable and Dynamic ACL System for In-Network Defense," in *ACM CCS*, 2022.
[31] P. Bosshart *et al.*, "P4: Programming Protocol-Independent Packet Processors," *ACM SIGCOMM CCR*, 2014.
[32] C. H. Song *et al.*, "DySO: Enhancing Application Offload Efficiency on Programmable Switches," *Computer Networks*, vol. 224, 2023.
[33] D. Kim *et al.*, "FlexCP: A Scalable Multipath TCP Proxy for Cellular Networks," *Proc. ACM Netw.*, vol. 1, no. CoNEXT3, Nov 2023.
[34] NVIDIA, "Mellanox BlueField2 DPU SmartNICs," n.d., https://store.mellanox.com/categories/dpu.html [Accessed: Jan 2024].
[35] Intel Corporation, "Intel® Xeon® Gold 6342 Processor," 2021, https://tinyurl.com/yc6z8ex7 [Accessed: July 2023].
[36] Supermicro, "Supermicro NVIDIA BlueField-2 DPU 2x100GbE," 2024, https://tinyurl.com/2azy2sya [Accessed: May 2024].
[37] "Intel Tofino2 – A 12.9 Tbps P4-Programmable Ethernet Switch," https://ieeexplore.ieee.org/document/9220636 [Accessed: Mar. 2023], 2020.
[38] M. Rudow *et al.*, "Tambur: Efficient Loss Recovery for Videoconferencing via Streaming Codes," in *USENIX NSDI*, 2023.
[39] K. MacMillan *et al.*, "Measuring the Performance and Network Utilization of Popular Video Conferencing Applications," in *ACM IMC*, 2021.
[40] S. S. Chen *et al.*, "Time-division TCP for Reconfigurable Data Center Networks," in *ACM SIGCOMM*, 2022.
[41] 3GPP, "TS 38.323 v17.4.0: NR; Packet Data Convergence Protocol (PDCP) specification," 2023.
[42] A. Bhatnagar *et al.*, "P4EAD: Securing the In-band Control Channels on Commodity Programmable Switches," in *EuroP4*, 2023.
[43] Broadcom Inc., "Broadcom Delivers Industry's First Dual 400G MACSec PHY for Hyper-scale Data Center and Cloud Infrastructure," https://t.ly/S5mMD [Accessed: Apr 2024], 2019.
[44] Tessares, "5G ATSSS," https://www.tessares.net/solutions/5g-atsss-solution/ [Accessed: March 2023].
[45] Y. Chabanois, "OpenMPTCProuter – Internet connection bonding," https://www.openmptcprouter.com/ [Accessed: May 2024].
[46] M. a. Baerts, "Leveraging the 0-RTT Convert Protocol to Improve Wi-Fi/Cellular Convergence," in *ACM/IRTF ANRW*, 2021.
[47] M. Bednarek, G. B. Kobas, M. Kühlewind, and B. Trammell, "Multipath Bonding at Layer 3," in *ANRW*, 2016.
[48] M. Pieska *et al.*, "Low-delay Cost-aware Multipath Scheduling over Dynamic Links for Access Traffic Steering, Switching, and Splitting," *Computer Networks*, vol. 241, 2024.
[49] W. Sentosa *et al.*, "DChannel: Accelerating Mobile Apps With Parallel High-bandwidth and Low-latency Channels," in *USENIX NSDI*, 2023.
[50] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting Mobile 3G Using WiFi," in *ACM MobiSys*, 2010.
[51] D. Laselva *et al.*, "3GPP LTE-WLAN Aggregation Technologies: Functionalities and Performance Comparison," *IEEE Comm. Magazine*, 2018.