MmWave RAT Optimization: MAC Layer Initial Access Design and Transport Layer Integration

by

Suresh Srinivasan

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Computer Science

Dissertation Committee:
Ehsan Aryafar, Chair
Wu-Chi Feng
Feng Liu
Jacob Chakareski

Portland State University
2024

Abstract

MmWave Radio Access Technology (RAT) is a promising technology for wireless communication due its large bandwidth and is already being deployed in 5G cellular and emerging WiFi technologies. MmWave systems use highly directional beams with narrow beamwidths to overcome the high path loss associated with their frequency bands. A mmWave radio can be used either in a standalone mode (where all radios use the same technology) or simultaneously with other technologies such as LTE and low frequency WiFi in a communication mode commonly referred to as integrated mode. This thesis proposes two methods to optimize mmWave RAT performance in both standalone and integrated modes. First, we address the problem of mmWave link establishment in the standalone mode, which is part of the initial access of the MAC layer. We show that when multiple clients try to establish a link, there exists a severe power imbalance among competing clients' beams, as clients naturally have different orientations and are at different distances from the same access point. This beam power imbalance coupled with poor contention protocols results in poor fairness in dynamic systems with multiple clients. We then introduce a joint power control and contention adaptation mechanism to enhance

fairness. Second, we focus on integrated mode and show that existing transport layer solutions such as MPTCP that utilize multiple links simultaneously can have very poor performance when a mmWave RAT is deployed due to susceptibility of mmWave links to mobility and blockages such as human, objects (vehicles, walls, vehicles, animals etc.,). We then propose the design of Forward and Backward Data Transmission (FBDT) protocol, a novel multi-path transport layer solution that can get the summation of individual rates across RATs despite system dynamics. We have implemented FBDT in the Linux kernel and show substantial improvement in throughput against state-of-the-art schemes, e.g, 2.5x gain in a dual-RAT scenario (WiFi and WiGig) when the client is mobile. Further, we extend FBDT to more than two radios and demonstrate that its throughput performance scales linearly with the number of RATs, in contrast to multi-path TCP, whose performance degrades with an increase in the number of RATs. We evaluate the performance of FBDT on different traffic classes and demonstrate (i) 2-3 times shorter file download times, (ii) up to 10 times shorter streaming times and 10 dB higher video quality for progressive download video applications, and (iii) up to 9 dB higher viewport quality for interactive mobile VR applications, when our viewport maximization framework is employed along with FBDT.

Dedicated to my Wife and Son - Nalinadevi and Srivatsan

If four things are followed - having a great aim, acquiring knowledge, hard work, and perseverance - then anything can be achieved. by A. P. J. Abdul Kalam

Acknowledgments

I extend my sincerest gratitude to Prof. Ehsan Aryafar, my advisor, whose invaluable guidance and unwavering support were instrumental throughout my Ph.D. journey. His counsel emphasized the significance of addressing research problems that profoundly impact the field, a lesson that remains pivotal in my academic pursuits. Prof. Aryafar's understanding and encouragement during personal and professional hurdles were pivotal in my successful completion of the Ph.D. program. Under his mentorship, I honed the necessary research skills to yield impactful results, primed for competitive conference presentations. The technical expertise imparted by my advisor now serves as a cornerstone as I embark on my professional career. This collaborative journey with Prof. Aryafar has been both unforgettable and profoundly enlightening, shaping my academic and technical growth in profound ways.

I express my heartfelt appreciation to Prof. Jacob Chakareski, whose timely guidance and unwavering encouragement propelled me towards achieving our research objectives. I am profoundly grateful to my Ph.D. committee members , Prof. Wu-Chi Feng and Prof. Feng Liu, for their invaluable insights as members of

my thesis committee. Their consistent availability and valuable advice throughout my Ph.D. journey have been pivotal.

I wish to extend my heartfelt gratitude to Prof. Mark P. Jones, the department chair, for his unwavering support throughout my Ph.D. journey. His support was instrumental in making my Ph.D. program possible.

I am immensely grateful to my wife, Nalinadevi, my son, Srivatsan, and my mother, Selva Kumari, for their unwavering and enduring support. Their encouragement was indispensable, without which my Ph.D. journey would not have been possible.

I extend special gratitude to my managers at Intel, Puneet Jain, Geng Wu, Wendell T Scruggs, and Cristina Beldica, whose unwavering support was instrumental in facilitating my Ph.D. research endeavors. Their continuous assistance was indispensable, enabling me to pursue and successfully complete my Ph.D. program.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

MmWave communication is one of the essential components of next generation wireless networks to support extremely high data rate services. The mmWave frequency bands provide an order of magnitude more spectrum than already congested sub-6 GHz bands, which can be used to boost the communication capacity. However, mmWave systems suffer from high path loss, high noise power, and susceptibility to blockages such as humans[1] [2]. To address these challenges, mmWave systems use an array of antennas and form highly directional beams[2] at both the transmitter (Tx) and receiver (Rx) to increase the SNR. These directional beams reduce interference, boost capacity, and increase the security of communication[3], however, the plurality of narrow beams at the Tx and Rx makes the initial access (IA) and adaptation to mobility much harder. IA is a process by which an access point (AP) and a client device establish a physical connection, after which

[1] e.g., the human body alone can reduce the signal strength of a mmWave signal by more than 20 dB [1].

[2] **We use the words “beams" and “sectors" interchangeably.**

[3] Omni-directional transmission is susceptible to interception, which can be alleviated by the use of directional beams.

the data communication can start. In mmWave systems, the Tx and Rx need to find appropriate beams before they can communicate. This beam training procedure in the existing standards (e.g., 3GPP 5G and IEEE 802.11 ad/ay) is handled during the IA. In this thesis, we delve into the intricacies of next-generation wireless networks, particularly focusing on millimeter-wave (mmWave) communication and the diverse range of applications driving the need for multi-RAT wireless communication.

Simultaneously, the evolution of social media, video platform, cloud-based communication and advanced AR/VR interactive applications has led to a diverse range of mobile devices such as smartphones, VR/AR headsets, and laptops. These devices cater to multifaceted applications that demand varying bandwidth and data rates. Different wireless networks like Wigig, WiFi, 5G, 6G, LTE, each serving distinct purposes, ranging from high bandwidth with limited range to long-range communication with lower bandwidth, highlight the need for multiple wireless networks to meet specific application Quality of Service (QoS) requirements. Utilizing a singular network for these multifunctional devices falls short in delivering the necessary QoS parameters like low/high latency and high data rates. In addressing this challenge, it becomes crucial to categorize these multifunctional applications to ensure the fulfillment of desired QoS. These applications can be effectively classified into four primary categories: Conversational, Streaming, Interactive, and Background, as depicted in Figure 1. This categorization serves as a fundamental step towards ensuring the tailored provisioning of network resources to meet the diverse QoS demands of these multifaceted applications. There are four main

classes as per 3GPP specifications[[3]]: Conversational, Streaming, Interactive, and Background, as illustrated in Fig. 1.1. Among these, Conversational and Streaming traffic are real-time and highly time-sensitive. Conversational traffic, utilized by applications like metaverse, AR/VR communication, video conferencing, and gaming, is the most sensitive to delays. Streaming traffic, which includes platforms like YouTube, Netflix, and ESPN, is also real-time but less sensitive to delays, often involving one-way transport aimed at live human audiences.

On the other hand, Interactive and Background traffic have looser delay requirements and lower bandwidth needs. Interactive traffic is generated by applications like interactive chat, file download and web browsing. Meanwhile, Background traffic is associated with social media apps running in the background, such as Meta, Twitter, Facebook, etc., and simple mail transfer protocol (SMTP). These distinctions in traffic classes are essential for optimizing network performance and ensuring seamless user experiences in various mobile applications. The rising need for cloud storage applications, including services for photos, backups, videos, and more, has led to the emergence of a Background class of applications. These applications involve data transfers typically ranging from 10 to 100 gigabytes [4]. This increasing demand highlights the significance of accommodating large-scale data transfers over wireless network.

An exciting and highly anticipated addition to the conversational class of traffic is the metaverse, which is broadly described as a network of 3D virtual worlds facilitated by virtual reality (VR) and augmented reality (AR) headsets. The Metaverse places a strong emphasis on social interaction and holds the promise

of diverse future applications [5]. The Metaverse and VR/AR applications in general have been recently identified by 3GPP as key emerging application settings for 5G+/6G technologies [6]. Mobile 360° video VR streaming, i.e., observing high-quality 360° video streams on unthetered mobile VR headsets, is anticipated to be one of the key enabling technologies of the Metaverse, and has attracted considerable attention recently [7].

While mobile cloud storage applications are relatively less sensitive to delays, they significantly consume wireless network bandwidth, which can impact the performance of other mobile applications. In contrast, studies indicate that streaming and conversational class applications have demanding requirements. According to several recent studies [8], [9], the required wireless data rates are about 30 Mbps for a 2K H.264-encoded stream, 800 Mbps for an 8K H.266 encoded stream, and up to a few Gbps with higher resolution or frame rate. The uplink rates are insignificant, i.e., well below 2Mbps, as only headset orientation and some other client-generated control need to be transmitted. Existing radio access technologies (RATs) cannot consistently provide high downlink data rates. For example, LTE speeds are typically 10 Mbps [10], which is far lower than the required data rates. Recent sub-6 GHz (11 ac/ax) and 60 GHz (11 ad/ay) WiFi can provide anywhere from 100 Mbps to a few Gbps but sub-6 GHz WiFi can suffer from interference and bad channel conditions (e.g., low rank MIMO) whereas 60 GHz communication is highly susceptible to client mobility and blockages. Each of these conditions can drastically reduce the RAT throughput. This susceptibility to blockages and mobility is also prevalent in existing mmWave 5G solutions, which

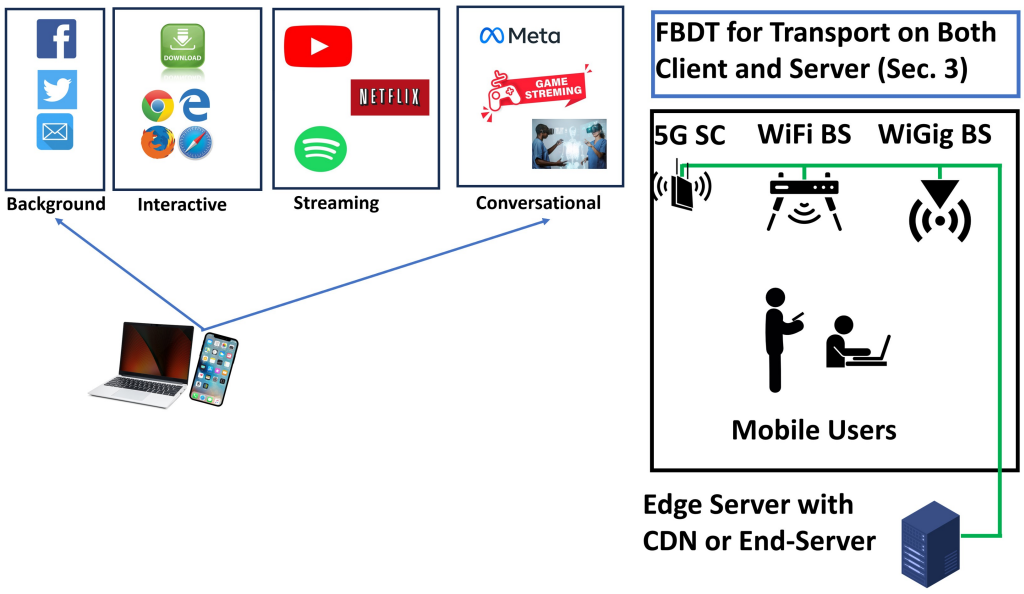


Figure 1.1: FBDT - Low Latency and High throughput. Left: Client with Interactive, Background and Conversational class of application Right: An edge server is connected to different Base Stations (BSs) including WiFi, WiGig, and 5G small cell. FBDT is implemented at the transport layer on both the client and server.

creates large fluctuations in throughput. Second, the high variability in wireless network bandwidth can cause unexpected re-buffering, which drastically reduces the client Quality of Experience (QoE) [11]. This reduced QoE due to either low quality video frames or re-buffering can even lead to client disorientation or nausea [12].

Simultaneous traffic aggregation across multiple RATs such as sub-6 GHz WiFi, mmWave WiFi (WiGig), LTE, and/or 5G can be a key solution to address the aforementioned challenges by boosting the wireless capacity and providing high minimum data rates across all channel conditions and in presence of client/network dynamics. However, as we will show later in this thesis, existing transport layer

multi-RAT traffic aggregation schemes can suffer from Head-of-Line (HoL) blocking and sub-optimal traffic splitting across RATs, particularly in the presence of system dynamics such as when the channel condition frequently changes from Line-of-Sight (LoS) to non-Line-of-Sight (nLoS) and vice versa. As a result, in many practical situations, the overall throughput of a Multi-Path TCP (MPTCP) protocol can even be less than the throughput that would be achieved by using only a single RAT at all times.

1.1 Summary of Thesis Contribution

In lieu of optimizing the mmWave mac layer, we focus on the 802.11 ad/ay standard and design the JPOC protocol to improve the IA fairness. While we focus on WiFi, the methods proposed in this thesis are also applicable to the 5G cellular standard. First, we conduct extensive experiments with commercial-off-the-shelf (COTS) devices to identify the root cause of contention imbalance in real mmWave networks. We show that *poor contention protocols* and *power imbalance among competing beams* result in poor IA fairness among clients. The problem becomes worse in presence of system dynamics such as mobility and blockages.

In this thesis, we present the design JPOC to address the contention unfairness during IA. JPOC's design is complementary to several existing beam search, mobility, and blockage management protocols, and can be used to enhance their fairness in presence of system dynamics. At its core, JPOC introduces a new *open-loop* and *client-side* power control mechanism and a new contention protocol that improve fairness and reduce the overhead of IA.

We extensively evaluate JPOC's performance through experiments using COTS devices. We show that compared to the standard 802.11 ad/ay protocols, JPOC substantially reduces the contention overhead and increases the IA fairness.

Though MAC layer optimization leads to better mmwave communication but it fails to overcome blockages and nLoS issues. To overcome such issues, we propose a new transport layer multi-RAT traffic aggregation scheme that achieves *summation* of individual RAT data rates in all channel conditions. FBDT eliminates HoL blocking and sub-optimal traffic splitting problems by removing retransmission schemes employed by MPTCP and relying on individual Single-Path TCP (SPTCP) mechanisms for reliable delivery. Further, for a given queue of packets to be transmitted, FBDT optimally places the pointers to fetch traffic for each RAT by taking into account the historical reliability of each RAT. Finally, FBDT serves packets from both forward and backward directions in the queue so that each RAT can independently transmit packets without being blocked by the status of other RATs.

We implemented FBDT for multi-RAT communication as a daemon in the user space in Linux on both client and server side. This allowed us to implement our protocols and algorithms for multi-RAT communication in Commercial-off-the-shelf (COTS) hardware (routers, laptops) and evaluate its performance in real-world settings. Prior to Journal publication, we will open-source the updated software and publicly release it on GitHub so that other researchers in the community can benefit from our work and expand on it.

We have conducted extensive experiments to evaluate the performance of FBDT

and the induced Quality of Experience (QoE) of the receiving client. We show that FBDT can achieve an aggregate throughput that equals the sum of the data rates of the individual RAT network paths, for all channel conditions, i.e., when the client is in LoS, nLoS, or mobile. We also show that compared to a state-of-the-art MPTCP scheduler, FBDT increases the aggregate throughput by a factor of 2.5x in a dual radio setup with one WiFi and one WiGig (802.11 ad) RAT. This throughput gap increases when using a higher number of RATs. Moreover, compared to the MPTCP scheduler, FBDT provides a gain of up to 30% for the Interactive and Background classes of traffic, and up to 18X gain in the case of Streaming traffic. Finally, for the Conversional traffic class, we show that FBDT, coupled with a 360-degree video tile rate allocation and scheduling optimization method, provides an average of 9 dB increase in VR user viewport PSNR quality.

1.2 Thesis Overview

The thesis proceeds as follows. We discuss the background for this thesis in chapter 2. In chapter 3, we present design and evaluation of Fair Initial Access Design for mmWave wireless. In chapter 4, we present design, Implementation and evaluation of Forward and Backward Data Transmission (FBDT) Across Multi-RAT. In chapter 5, related work is discussed on both the topics. Finally, In chapter 6, We conclude by delving into the implications arising from the thesis findings.

Chapter 2

Background

In this section, we begin by outlining the common types of antenna arrays employed in mmWave radios and detail the beamforming training process utilized in 802.11 ad/ay. Following this, we delve into Multi RAT communication and subsequently explore multipath TCP along with its limitations in the context of multi-RAT communication.

2.1 MmWave and IEEE802.11ad/ay

MmWave Radios. Fig. 2.1(a) depicts the various components of a commercial mmWave radio. Here a Tx/Rx RF chain, which is responsible for creating a single Tx/Rx signal stream, is connected to an array of antennas. These antenna arrays are referred to as phased arrays. The phase shifter on each antenna element i shifts the phase of the signal that passes through it by multiplying the time domain RF signal by a complex coefficient ω_i. By an appropriate setting of the phase shift variables (i.e., ω_is), we can (i) realize a beam pattern with a maximum beamforming gain (i.e., main lobe) in a particular direction and (ii) steer the beam in the 3D space.

Commercial mmWave radios also have access to an omni (or quasi-omni) antenna pattern, which is used in the 802.11 ad/ay standard.

MmWave Initial Access. Initial access in mmWave systems is a procedure that allows a client device to discover a cell, helps the AP and clients to find appropriate beams to communicate with each other, and allows the AP to send management and control information to all the clients.

In 802.11 ad/ay, this process is handled in the beginning of each beacon interval (BI) [13], [14]. The length of a BI is typically 100 ms, i.e., the BI is repeated every 100 ms. The BI is composed of two parts: (i) the beacon header interval (BHI), which helps with AP discovery, beam training, and control and management information exchange, and (ii) the data transmission interval (DTI), which is used for data communication and can support different types of medium access protocols. The format of a BI is depicted in Fig. 2.1(d).

The BHI consists of three sub-intervals:

- **Beacon Transmission Interval (BTI):** The BTI comprises multiple beacon frames, each transmitted sequentially by the AP on a different sector (beam) to cover the desired directions. This process is referred to as AP sector sweep and is used for network announcement and beamforming training of the AP's sectors. During the AP sector sweep, all clients stay in reception mode using an omni (or quasi-omni) antenna pattern. Each client records the signal strength and beam ID of every sector sweep frame (SSW frame) received from the AP. Fig. 2.1(c) shows this operation.
- **Association Beamforming Training (A-BFT):** This interval is used by the

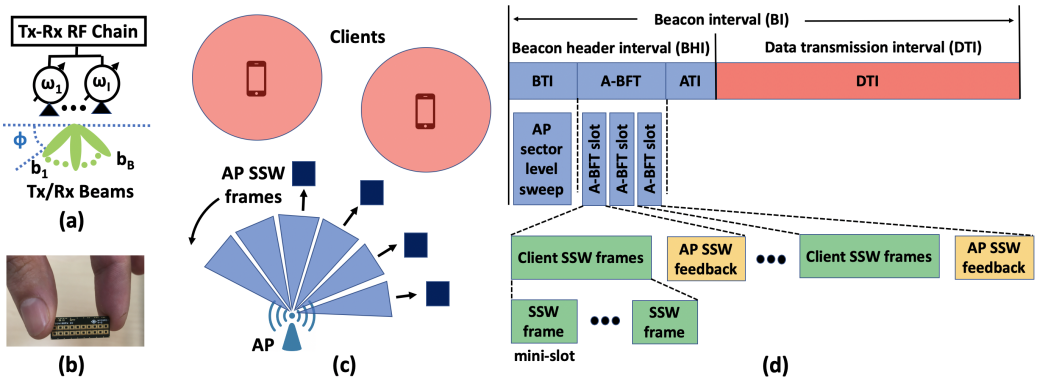


Figure 2.1: (a): In a conventional mmWave radio, a single RF chain is connected to an array of antennas. There are B sectors (beams) that cover the desired 3D space; (b) A real 802.11 ad radio; (c) During the beacon transmission interval (BTI), AP sequentially sends sector sweep (SSW) frames on each of its sectors. Each client uses an omni antenna pattern and records the beam ID and signal strength of all the received SSW frames; (d): Association beamforming training (A-BFT) is composed of a few slots. A client randomly chooses an A-BFT slot to conduct its sector sweep.

client devices to train their sectors for communication with the AP. Upon completion of a successful A-BFT communication, the AP would determine and inform the clients about the beam that each client should choose for communication with the AP. To allow multiple clients to respond to an AP sector sweep, the A-BFT interval implements a contention-based response period. The A-BFT interval reserves time for multiple client sector sweeps (i.e., A-BFT slots). An overview of the A-BFT procedure is shown in Fig. 2.1(d). Each A-BFT slot consists of a fixed time allocation (i.e., a fixed number of mini-slots) for a number of SSW frames (transmitted by the connecting client) and one SSW feedback frame sent by the AP. Each contending client randomly selects an A-BFT slot and performs its sector sweep in that slot. Each SSW frame sent by a client contains the client

information and AP beam ID that resulted in the maximum SNR at the client during the AP sector sweep (i.e., BTI). Note that during A-BFT, the AP remains in reception mode leveraging an omni antenna pattern. The AP simply records the signal strength, beam ID, and client ID for each successfully received SSW frame and then informs the client about the beam ID it should use to communicate with the AP. The contention process during A-BFT does not apply carrier sensing. Instead, a collision is detected by a missing SSW frame from the AP. Further, a client device may not be able to finish its sweep in one A-BFT slot, e.g., because the number of its sectors exceed the number of SSW frames per slot. In this scenario, a client may randomly choose another A-BFT slot in the same BHI or a later one. Finally, the 802.11 ad and ay standards only specify the maximum allowed number of A-BFT slots and leave the exact number of slots/mini-slots to vendors.

- **Announcement Transmission Interval (ATI):** During ATI, the AP exchanges management information with associated and beam-trained client devices. While communication during BTI and A-BFT uses the lowest modulation and coding scheme (MCS) to increase range for untrained beams, communication during ATI happens with trained beams and thus can use a higher order MCS for increased efficiency.

Medium Access Control (MAC). 802.11 ad/ay standard support three types of MAC protocols: contention based access, scheduled channel time allocation, and dynamic channel time allocation. The latter two mechanisms uses time division multiple access (TDMA) and polling (similar to 802.11 PCF) to share resources

among the clients.

2.2 Multiple User RAT Communication

In IEEE 802.11 ad and ay standards [13], [14], IA (and beam search) is done in the beginning of every beacon interval. In particular, initially an AP sequentially sends sector sweep frames across its sectors, while all the clients record the signal strength of the received beams. In the next phase, each client randomly chooses a beam training slot and performs a sector sweep in that slot. Several research works have proposed alternative methods that find better beams and/or reduce the beam search overhead. These works can be broadly divided into three classes: (i) *exhaustive sweeping* [15]–[17]: narrow spatial beams are used to scan all the directions exhaustively; (ii) *hierarchical sweeping* [18]–[20]: hierarchical codebooks are used to sweep all the directions; and (iii) *random sweeping* [21]–[25]: several random beamforming vectors are used to find the directions. A key missing piece in all these works (including the 802.11 ad/ay standard) is fairness in IA. In particular, IA provides an opportunity for all the clients to train their beams and most existing protocols rely on contention between clients as they sweep their beams. However, mmWave systems suffer from the near-far problem, in which the beams of a client that is near to the AP would have a much higher power at the AP than the beams of a competing far-client. This power imbalance can create a significant IA unfairness among competing clients, which can delay or even deny far-clients from being admitted to the network. To the best of our knowledge, this is the first thesis that addresses the IA fairness problem in mmWave networks. In

particular, we identify *received power imbalance* and *poor contention protocols* as the key reasons behind poor IA fairness in multi-client mmWave networks. Note that IA fairness is different from throughput fairness commonly studied in networking problems [26]–[28].

Other works have proposed protocols to better address mmWave's mobility and blockages. Existing mmWave standards respond to these events by re-trigerring the sector sweep procedure and finding a new set of beams. Recent works have proposed several solutions to optimize the standard beam adaptation methods, e.g., (i) out-of-band beam search methods [29], [30] exploit the channel information from a co-existing low-frequency radio to speed up the beam adaptation; (ii) environmental sensing solutions [31], [32] sense the reflective environment and leverage the sensed information to facilitate beam adaptation, and (iii) pro-active beam adaptation [33]–[36] uses model-driven methods to adjust the beams before blockages happen. These solutions alleviate the impact of mobility and blockages, and ensure a smoother data communication. However, they cannot necessarily find the optimal Tx-Rx beams in presence of mobility or blockages. As a result, a client may continue to contend for beam training slots in continuously recurring beam search intervals. Thus, system dynamics (e.g., mobility, blockages) can exacerbate the IA contention unfairness problem as clients frequently compete to re-train their beams.

2.3 Multi Path Transmission Control Protocol (MPTCP)

MPTCP Architecture. MPTCP [37]–[39] increases the performance for the

application by pooling resources from multiple RATs. Fig. 2.2 depicts the main components of MPTCP. At the sender, a single TCP socket is exposed to the application and outgoing application segments are copied to a send queue at the meta-level. A separate re-injected queue is used at the meta-level for segments that needs to be retransmitted. Below this MPTCP interface, TCP subflows are created for each RAT (path). The pooling of the subflow's resources is achieved by multiplexing individual segments across the different subflows. When multiplexing individual segments, MPTCP uses a *scheduler* to decide on which subflow to schedule each segment. The scheduler has access to the state of each TCP subflow, including congestion window (*cwnd*) and round-trip time (RTT). The rate at which segments are sent out on each subflow is determined by *cwnd*. On the receiver side, the segments arrive first at the receive queue on each subflow and then delivered in-order at the subflow level to the common receive queue at the meta-level. Segments arriving out-of-order are placed on an out-of-order (OOO) queue at the meta-level. Note that the receive and OOO queues at the meta-level are shared among all subflows. The size of the required buffer is critical to allow high MPTCP throughput and the amount of the buffer left in the shared buffer is advertised by the receiver to the sender as the receive window (*rwnd*).

Head-of-Line (HoL) Blocking. MPTCP leverages multiple paths with different delay and loss profiles. As packets are multiplexed across the different subflows, the paths' delay and loss differences might cause OOO delivery at the receiver. Note that MPTCP ensures in-order packet delivery. As a result, packets scheduled on the low-delay path might have to wait for the high delay path packets in the OOO queue.

This phenomenon is known as the HoL blocking. Fig. 2.2 depicts an example in which packets 4, 5 and 6 have to wait at the OOO queue as packet 3 from RAT1 is lost. The severity of HoL blocking depends on the delay, throughput, stability of the links, and the size of the buffers (e.g., switch buffers) along the different paths. Wireless networks generally suffer from more unstable links (as compared to wired networks) due to substantial variations in link quality due to mobility, interference, and changes in the multi-path environment. Moreover, emerging wireless technologies such as 5G or 802.11 ad/ay operate in the higher frequency mmWave bands. These bands introduce additional challenges, e.g., mmWave bands are known to suffer from blockages [40], [41], which can suddenly bring down the rate of a few Gbps link to zero. Additionally, these technologies use highly directional beams for communication, which can cause significant performance degradation with mobility. Even if future MAC and PHY improvements (e.g., [42]–[45]) result in faster beam steering and link recovery, many realistic wireless setups would contain a very high number of disconnection events, which can significantly degrade the MPTCP and application level performance.

MPTCP Evaluation. Several works [46]–[50] have studied MPTCP performance but these works consider scenarios that consist of Internet paths or wireless setups with only sub-6 GHz RATs. Other works have studied MPTCP performance in networks that use mmWave RATs, e.g., [51], [52] studied dual WLANs with 802.11 ac+ad and show that MPTCP can get a lower performance than using WiGig only, [53] explores MPTCP in 5G+LTE through simulations, and MuSher [54] explores dual WiFi 802.11 ac+ad through implementation. FBDT is implemented

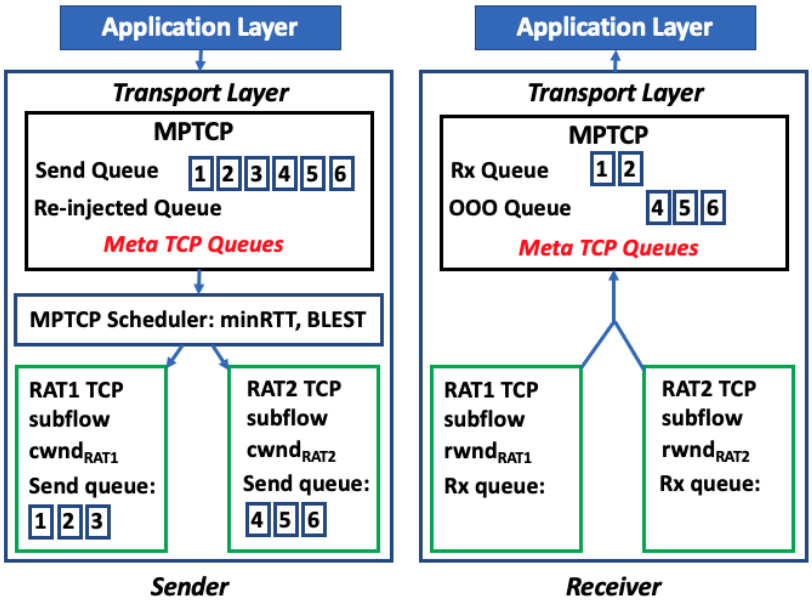


Figure 2.2: MPTCP architecture. Packet 3 from RAT1 is lost. Packets 4, 5, and 6 are received at the receiver but moved to the out-of-order buffer. The loss stops subsequent packet deliveries to the Meta Rx queue and the application.

in Linux kernel, supports any number of RATs, and significantly outperforms MuSher when client frequently switches between LoS and nLoS channels.

Schedulers. A wrong scheduling decision might result in HoL blocking, OOO packet arrivals, or receive excess buffering (buffer bloats). Accurately scheduling data across multiple paths while trying to avoid these issues is challenging, especially if different paths have very different delay/loss/rate profiles, which is the case in today's heterogeneous wireless networks. To minimize these issues, several schedulers have been proposed in the literature (for full discussion refer to subsequent MPTCP_Schedulers). The most important schedulers that are used by Linux are minRTT and BLEST. minRTT starts by filling the congestion windows

of the subflow with the lowest RTT before advancing to other subflows with higher RTTs. When one of these subflows blocks the connection, the scheduler retransmits the segments blocking the connection on the lowest delay path and penalizes the paths that caused the issue [55]. This can result in under-utilization of paths with burstiness in their rate, leading to suboptimal capacity aggregation. BLEST [56] is designed to increase MPTCP's performance over heterogeneous paths. BLEST monitors the MPTCP send window to reduce the time where the faster subflow cannot send a packet due to insufficient space. We have observed in our experiments that BLEST has superior performance to minRTT. Thus, unless otherwise specified, we use BLEST as the default MPTCP scheduler. MuSher [54] is a recently developed MPTCP scheduler that is designed and evaluated for 802.11ac/802.11ad dual WiFi setups. Their key finding is that the optimal MPTCP performance is achieved if the packet assignment ratio matches the throughput ratio of the two subflows. MuSher is recently shown to achieve low throughout with small queues [57]. We will additionally show in Section 4.4 that MuSher can get very low throughput performance when the client frequently switches between LoS and nLoS channels.

MPTCP Schedulers. In addition to the schedulers discussed above, several schedulers have been proposed in the community, including schedulers that try to: (ii) address the challenges associated with heterogeneous paths [56], [58]–[60], (ii) leverage the differences in subflow RTTs [56], [58], [60]–[63], (iii) improve MPTCP performance for special use cases [64]–[66], and (iv) require modifications to the application [67]. FBDT can address multi-RAT scenarios with vastly different characteristics across RATs, has superior performance in

static/mobile and LoS/nLoS scenarios, and does not require explicit information from the lower layers or modifications to the application.

2.4 Multiple Radio access Technologies (RAT)

Simultaneous traffic aggregation across multiple RATs such as sub-6 GHz WiFi, mmWave WiFi (WiGig), LTE, and/or 5G can be a key solution to address the aforementioned challenges by boosting the wireless capacity and providing high minimum data rates across all channel conditions and in presence of client/network dynamics. However, as we will show later in this thesis, existing transport layer multi-RAT traffic aggregation schemes can suffer from Head-of-Line (HoL) blocking and sub-optimal traffic splitting across RATs, particularly in presence of system dynamics such as when the channel condition frequently changes from Line-of-Sight (LoS) to non-Line-of-Sight (nLoS) and vice versa. As a result, in many practical situations, the overall throughput of a Multi-Path TCP (MPTCP) protocol can even be less than the throughput that would be achieved by using only a single RAT at all times.

Chapter 3

Fair Intial Access Design for mmWave Wireless

3.1 Motivation

In this section, we present experimental results to motivate the existence and prevalence of the contention unfairness problem in mmWave networks with commercial-off-the-shelf (COTS) 802.11 ad devices. We first conduct experiments to characterize the variations in SNR as a function of beam ID and client location. We show that there is a significant imbalance in the received power of clients' beams as a function of client location, which can cause severe contention unfariness during the A-BFT interval. Next, we show how network dynamics such as mobility and blockages can re-trigger client participation in the A-BFT contention.

Experiment Setup. We setup a single-cell mmWave network in an indoor office environment using three Talon AD-7200 routers and an Acer TravelMate laptop. We use one router as an AP, one router as a packet sniffer, and one router configured as a client device. We use the TravelMate laptop as a different client device. Fig. 3.1(a) shows a picture of our equipment. All of our devices use Qualcomm's QCA9500 802.11 ad chip, which uses a 32-antenna phased array

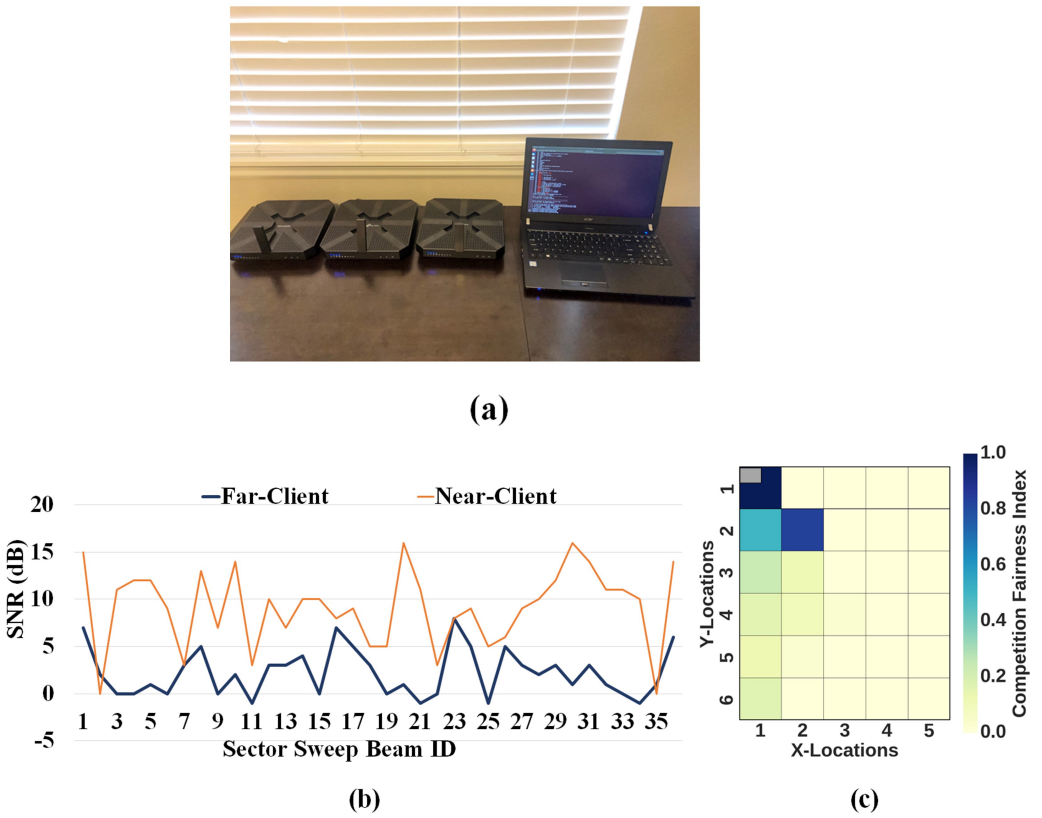


Figure 3.1: (a): We conduct experiments leveraging three Talon AD-7200 routers and an Acer TravelMate laptop; (b): SNR of near- and far- clients' beams versus sector ID; (c): Fairness heatmap. Near-client location is fixed. The other client is placed in different cells to derive the fairness value. The gray box at the top left corner shows the AP location. Without power control, the competition fairness index at many locations is close to zero.

with a single Tx-Rx RF chain. The default firmware for the AD-7200 router neither supports A-BFT SNR dump nor sniffer mode. To enable these features, we modified the default firmware using Nexmon framework [68] and installed that on the sniffer router to gather low-level signal statistics. This framework serves as a jailbreak into the default firmware of 802.11 ad, facilitating the integration of patches in C language rather than assembly. Additionally, it introduces new

attributes and programs, including a GCC plugin. Talon AD-7200 routers with Nexmon firmware can be configured as either an AP, client or sniffer [36], [68]. We place the sniffer router next to the AP. In all of our experiments, we let the clients get connected to the AP but measure their packets' signal strengths at the sniffer. This is because, as mentioned, the default AP firmware (which should be used for client connectivity) does not provide signal statistics.

Sample Beam SNR values for two Different Client Locations. We first conduct an experiment to see how distance between clients and AP affects the SNR of the clients' beams at the AP, and hence their contention during the A-BFT interval. We place one client in front of the AP at 1 m distance ((x,y) location (1,1) in Fig. 3.1(c)). We refer to this client as the near-client. We let the client conduct its sector sweep and get associated to the AP. The orange graph in Fig. 3.1(b) shows the SNR of the near-client's beams as a function of its beam IDs. We observe that all beams achieve a minimum SNR of 0 dB with a few beams achieving SNR values as high as 16 dB. We next use a second client and place it at 5 m distance from the AP ((x,y) location (3,3) in Fig. 3.1(c)). We let the client get associated to the AP and plot the resulting beams' SNR values on the same figure. We refer to this client as far-client. We observe that almost all of the far-client beams achieve a lower SNR than the near-client beams. As a results, whenever the two clients choose the same A-BFT slot to conduct their sector sweeps, most of the near-client's SSW frames would be captured[1] by the AP, whereas the far-client would not be heard

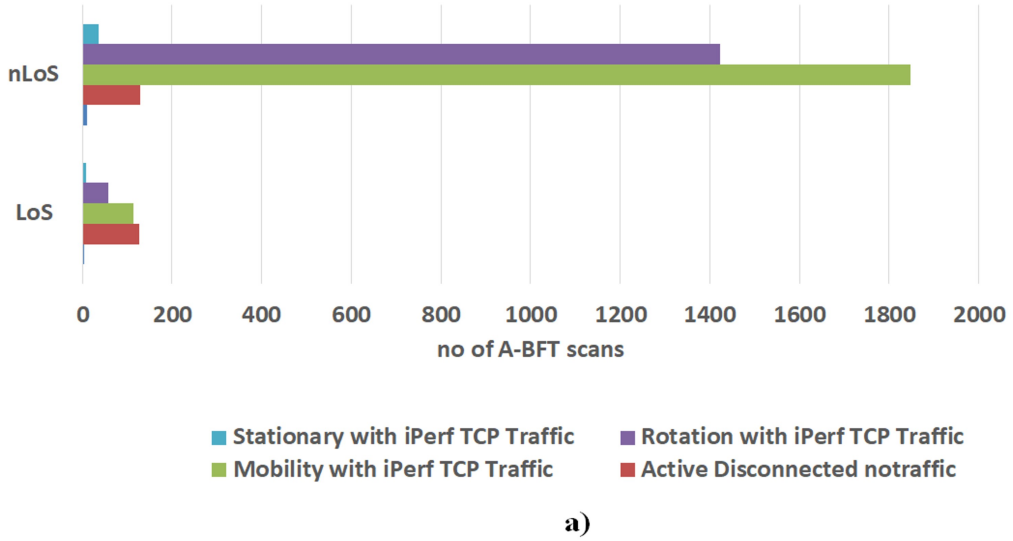
[1]When two frames arrive at the same time at the AP, the frame with an x dB or more signal strength can be correctly decoded, whereas the other frame would be lost. The exact value of x depends on the frames' MCS values, e.g., 3 dB with the lowest MCS, which is used in SSW frames.

or acknowledged. The far-client can only get associated to the AP if it chooses a different A-BFT slot in the same or next beacon interval.

Prevalence of Contention Imbalance. Our next goal is to characterize the prevalence of contention unfairness caused by received beam power imbalance across different locations in a typical indoor environment. Fig. 3.1(c) shows the layout of our experiment setup and the corresponding fairness heatmap. Our AP is located at the top left corner (gray box in Fig. 3.1(c)). We place the near-client in (x,y) location (1,1) and let it associate to the AP. We next record the SNR values of its beams at the AP. We next place the second client in every other (x,y) location, allow it to associate to the AP, and record the SNR values of its beams at the AP. Finally, we take the following steps to characterize the competition fairness among the two clients from all of our recorded data. Let SNR_i^{near} denote the near-client SNR value (in dB) for beam index i, and $SNR_i^{other(j)}$ show the SNR value of the other client at location j as it uses beam index i. Let $S^{other(j)}$ (success rate of the other client at location j be defined as the total number of beams (is) for which $SNR_i^{other(j)} - SNR_i^{near}$ is $\geq$ 3 dB. Assuming that the two clients send their SSW frames in the same A-BFT slot, these are the beams that can be decoded at the AP due to the capture effect. Similarly, we define S^{near} (success rate of the near client) as the total number of beams for which $SNR_i^{near} - SNR_i^{other(j)}$ is $\geq$ 3 dB. We then define the competition fairness index at other client location j as $\frac{S^{other(j)}}{S^{near}}$. Fig. 3.1(c) shows the heatmap of this index. We observe that sector SNR imbalance is quite prevalent in typical indoor environments. Only when the other client is located in

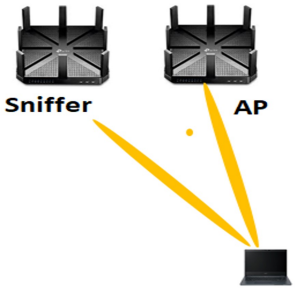
This phenomenon is referred to as the “capture" effect [69].

the same (x,y) cell as the near-client the fairness index is close to 1. As its moves to other locations, all of its beams' SNR values quickly drop to below the SNR values of the near-client's beams. In other words, it will lose contention in a majority of other locations, if both clients choose the same A-BFT slot to conduct their sector sweeps.



a)

(b)



(c)

Figure 3.2: (a): Frequency of client's participation in A-BFT during different mode of operation. (b): Travelmate laptop on a kubuki robot, which automates rotation and mobility (c): LAN setup: RoG/Travelmate connected to the AP with TCP upload/downloading TCP traffic to the server.

Frequency of Client Participation in A-BFT. The degree of throughput unfairness depends on the frequency of clients' participation in A-BFT, which itself depends on systems dynamics and parameters. For example, assume only a single A-BFT slot, one near-client and one far-client. If the near-client participates in A-BFT in each beacon interval, then the far-client would continuously lose the contention and would achieve a zero throughput. We have conducted several experiments to characterize what can re-trigger client participation in A-BFT. We have omitted the experimental results due to the page limitations, but provide a summary here: **(i) Mobility:** when a client moves, the initially selected sectors would not be optimal any longer. Hence, the client may participate in the next A-BFT to re-train its beams; **(ii) Blockages:** when blockage happens, the client would completely lose the connection to the AP. In an uplink scenario (e.g., uplink UDP), the client would wait until the next beacon interval to receive the AP SSW frames and then would participate in A-BFT to find an alternative path to the AP; and **(iii) Association to a New AP:** when a client moves, it can discover a new AP with a better channel quality. The client would then participate in A-BFT to find beams to the new AP.

We extensively evaluate client's frequency of participation using commercial of the shelf devices such as AD-7200, Travelmate and AsusRoG phone. We setup up a mmwave Local area network (LAN) using AD-720 as Access point and Travelmate or ASUSRoG phone as clients shown in Fig. 3.2(c)). We extensively evaluate client's frequency of participation in ABFT scans under stationary, rotation, mobile and disconnected setups as shown in Fig. 3.2(c)) and discussed these in detail in

the subsequent subsections.

Stationary with iPerf TCP Trafic. In this setup, the client- Travelmate- is placed at a fixed LoS location and it is connected to the access point (AP). To evaluate client's participation in this setup, the client generates iPerf TCP traffic for 10min over mmwave in LoS and nLoS orientation. In LoS, we observe that the client participates in ABFT for every minute as shown in Fig. 3.2(a)). but, whereas in the nLoS the client participates in the A-BFT whenever the QoS or RSSI drops below the threshold, in the hope to find a better beam as shown in Fig. 3.2(a)).

Rotation with iPerf TCP Traffic. Users with mmwave device, commonly undergo an alternate LoS and nLoS due to varying user mobility, mobile blockage etc., For such scenario, we evaluate client's frequency of participation under rotation. To emulate rotation, we place the client- Travelmate- on the kubuki robot, which is configured to rotate at 37rotation/sec as shown in Fig. 3.2(b)). Travelmate was connected to the AP with iperf TCP traffic over the mmwave link. The client, undergoing rotation, periodically switches between Line of Sight (LoS) and non-Line of Sight (nLoS). When the client gets an LoS with the AP, the client participates in the A-BFT periodically for every 1 min but when the clients are in nLoS, then the client preemptively participates in the A-BFT whenever the QoS or RSSI drops below the threshold, in the hope to find a better beam. We emulated rotation for about 10min and evaluated the frequency of client's participation in the A-BFT as shown in Fig. 3.2(a)).

Mobility with iPerf TCP Traffic. We evaluate client's frequency of participation in A-BFT under mobility. To emulate mobility, we place the client- Travelmate

– on the kubuki robot and it was programmed to move in a rectangular pattern of 6 ft by 2ft at a known distance from the AP. The client was in LoS and connected to the AP with iperf TCP traffic over the mmwave link. In this case, we observed that the client participated in the A-BFT more often as compared to afore mentioned stationary or mobility scenario. This is due to longer alternating duration of nLoS and LoS pattern at any given location. For example consider LoS location, with rectangular pattern mobility, the client undergoes an alternating pattern of LoS and nLoS with the AP. whenever the client is in LoS with the AP, we observed that it switches to the regular one minute A-BFT participation whereas when it is switched to nLoS due to rectangular pattern mobility, the client participates in the A-BFT whenever the QoS/RSSI falls below certain threshold in the hope to find a better beam as show in Fig. 3.2(a)). We further evaluate client's frequency of participation due to mobility at a nLoS location by placing the client – Travelmate- with the aforementioned robot setup at a nLoS location with the rectangular movement, and the client was connected to the AP with iperf TCP traffic over the mmwave link. we observed that the degree of client's A-BFT participation was much worse since the client was in a nLoS location with rectangular mobility. Due to nLoS, the client's QoS/RSSI signal was most of the time below the threshold and due to which the client used participate in the A-BFT on an average of about 15-20sec in the hope to find a better beam.

Active Disconnected no traffic. Interestingly, we discovered that disconnected clients also participate in A-BFT at the rate of 10sec as show in Fig. 3.2(a)). Consider a client, Travelmate, or RoG, with its mmWave radio turned on but

not connected to the access point (AP). Even under this scenario, the clients still participate in A-BFT scans to obtain the best possible beam. The clients participate in A-BFT scans at this high frequency as part of their periodic scan to keep their available AP(s) list updated. Listening to the BI would give the availability of the APs but the clients go ahead further and frequently participate in A-BFT scan to get the best possible beam at a very high rate of about 10sec. We refer to such clients as chatty clients causing interference to the active connected clients.

Clients participating in the A-BFT comes with a cost associated with it, as the client participates in the A-BFT more and more it worsens the near far issue as explained in section-3. Most of the nLoS clients participate in A-BFT when the QoS/RSSI falls below certain threshold in the hope to find a better beam, which is unnecessary and expensive as well. The clients are unaware of their state such as nLoS, mobility, rotation, disconnected etc., and blindly participating in the A-BFT in the hope to find a better beam, which shows the ignorance of the client. The clients should be wise in participating in the A-BFT either by sniffing the RSSI of BI of the AP or by using LoS and nLoS estimation, which will reduce the unnecessary participation in the A-BFT. Unfortunately, the existing of the shelf clients do not limit their A-BFT participation and increases the probability of near far problem.

3.2 Design of JPOC

We now present our joint power control and contention slot adaptation protocol (JPOC) that addresses the identified challenges. At a high level, JPOC incorporates

three innovations to achieve fairness in mmWave initial access. First, it introduces an uplink power control mechanism that is executed by each client device. This makes sure that for each client, only a few of its beams achieve high enough SNR at the AP so that they can be correctly decoded, whereas the rest of the client beams cause little to no interference at the AP. Second, we suggest a new A-BFT protocol and a mechanism for the AP to predict the optimal number of contention mini-slots (as a function of number of competing clients). This both reduces the overhead of initial access as well as the time it takes for clients to establish connection with the AP. Third, the AP continuously adapts its parameters in order to handle any system dynamics.

3.2.1 Power Control Design

The goal of our power control algorithm is to allow each client choose a transmission power such that only a few of its beams can be received at the AP. These are the client beams that without power control would have achieved the highest SNR values at the AP. We refer to these clients beams as **good-beams**. Now, assume that such a power control mechanism is executed by each client. As a result, when a near and a far client simultaneously perform their sector sweeps, the far client would not be drawn in the interference caused by the near client (Fig. 3.3 shows this operation). This increases the initial access fairness among competing clients and both clients would be able to establish a connection with the AP with a much lower number of required mini-slots. Note that the power control mechanism is used only during the A-BFT interval and would be disabled during data transmission for fast

data communication between the AP and client.

In this section, we discuss how a client can choose an appropriate transmission power. Note that when PC is employed, the client uses the **same** transmission power for all of its SSW frames as it performs sector sweep during A-BFT. We assume that without PC, each client uses the maximum transmission power (which is what is implemented in our COTS devices). Thus, when PC is employed, the selected transmission power is either reduced or unchanged.

We take the following approach. For every client, our goal is to select the client transmit power such that its *best beam* achieves a desired signal strength (and hence SNR) at the AP. Note that the *beam best* is not known prior to sector sweep.

Let RxP^{AP}_{max} denote the desired signal strength at the AP. We use the RxP^{AP}_{max} notation, as for each client, its best beam would achieve this value. Further, as PC is employed many of the client beams would cause little to no interference at the AP (Fig. 3.3). Also, note that RxP^{AP}_{max} is the same across all the clients. Now, a desired maximum received power at the AP is equivalent to aiming for a desired maximum SNR at the AP, as the two values are related to each other as follows

$$\gamma(dB) = RxP^{AP}_{\max}(dBm) - NoisePower^{AP}(dBm) \tag{3.1}$$

Here, γ denotes the desired SNR and $NoisePower^{AP}$ denotes the AP noise power. Noise power is a function of channel bandwidth and AP noise figure, which are all known parameters at the AP. Now, for a sector sweep frame to be decodable at the AP, its SNR should be at least equal to 1 dB. Thus, we choose a γ higher than 1 so

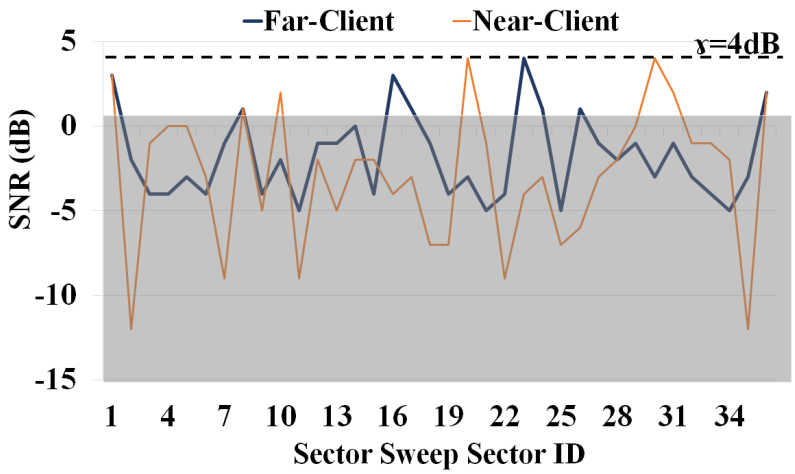


Figure 3.3: SNR of near- and far- clients' sectors when power control (PC) is employed. These are the same two clients that without PC received the SNR values depicted in Fig. 4.1(b). Each client adjusts its transmit power such that its best beam achieves an SNR equal to γ at the AP. Beams with less than 1 dB SNR cannot be decoded at the AP and cause zero to no interference. Clients' beams that fall into the shaded gray box depict such beams. In this example, $\gamma = 4$ dB.

that for each client potentially a few beams achieve high enough SNR at the AP as opposed to just one (e.g., in Fig. 3.3 we set γ equal to 4).

Therefore, for a desired γ, all the client has to do is to *select a transmit power* such that

$$RxP^{AP}_{\max}(dBm) = \gamma(dB) + NoisePower^{AP}(dBm) \tag{3.2}$$

To help clients with their PC, we let the AP announce the value of $RxP^{AP}_{\max}$ by using Eq. (3.2) and choosing a desired γ. The calculated value is included in the SSW frames transmitted by the AP during the BTI interval.

We next proceed to discuss how each client can choose its transmit power such that its maximum received power at the AP (as the client performs sector sweep)

is equal to $RxP^{AP}_{\max}$. Consider a client (CLT) and let i denote the client beam index that achieves the maximum received power at the AP. Then, from the path loss formula we have

$$RxP^{AP}_{\max} = P^{CLT}_T + G^{CLT}_T(i) + G^{AP}_R - PL \tag{3.3}$$

Here, P^{CLT}_T is the client transmission power, $G^{CLT}_T(i)$ is the client beamforming gain as it uses beam index i, G^{AP}_R is the AP's beamforming gain (note that during the client sector sweep, the AP uses a fixed omni beam for reception), and PL is the path loss component between the AP and client[2].

Next, consider the preceding AP sector sweep (i.e., BTI) and let j denote the AP beam index that achieved the highest SNR at the client. Then

$$RxP^{CLT}_{\max} = P^{AP}_T + G^{AP}_T(j) + G^{CLT}_R - PL \tag{3.4}$$

Note that $RxP^{CLT}_{\max}$ (i.e, the maximum received power at the client device as the AP performs sector sweep) is a known value at the client. By subtracting Eq. (3.4) from Eq. (3.3), removing the common PL term, and rearranging the parameters we have

$$P^{CLT}_T = RxP^{AP}_{\max} - RxP^{CLT}_{\max} + (G^{CLT}_R - G^{CLT}_T(i)) + (G^{AP}_T(j) + P^{AP}_T - G^{AP}_R) \tag{3.5}$$

[2]In LTE, clients explicitly estimate the PL with the help from eNB, and use that for uplink PC [70], [71]. We do not explicitly calculate PL.

Now, $(G_T^{CLT}(i) - G_R^{CLT})$ can be approximated as the array gain (or maximum beamforming gain) of the client antenna array[3], which is a known variable at the client device. Similarly, $(G_T^{AP}(j) - G_R^{AP})$ is the maximum array/beamforming gain of the AP's antenna array, which is known at the AP[4]. In JPOC, the AP calculates the value of $(G_T^{AP}(j) + P_T^{AP} - G_R^{AP})$ and adds this information to the message that is sent on each of its SSW frames during the AP sector sweep (i.e, BTI).

As a result, each client would have all the necessary information to use Eq. (3.5) and find the optimal transmission power (P_T^{CLT}) that it should use for its sector sweep. Note that if the calculated P_T^{CLT} is higher than the maximum possible Tx power, the client simply uses the maximum Tx power.

3.2.2 Design and Optimal Mini-Slot Number Determination

As discussed in Chapter 2, the 802.11 ad and ay standards divide the A-BFT time into slots and mini-slots, which as we show later in Section 3.3.2 can unnecessarily increase the A-BFT overhead. We propose to only use mini-slots. In particular, in JPOC, the AP determines the number of mini-slots that would be dedicated to A-BFT interval and announces that during its sector sweep (i.e., BTI interval). Let M denote the number of mini-slots and B the number of client beams. Then, during

[3]The approximation is because a selected beam does not maintain the beamforming gain in its entire beamwidth. In fact, the beamforming gain drops by 3 dB towards the edges of the selected beam. Thus, in all of our evaluations in Section 3.3, we remove 3 dB from the calculated $(G_T^{CLT}(i) - G_R^{CLT})$ and $(G_T^{AP}(j) - G_R^{AP})$ values. This slightly reduces JPOC's performance but ensures that the best beam achieves γ or a slightly higher SNR at the AP.

[4]All of our COTS radios use a fixed number of pre-determined beams for IA, which makes it easy to use Eq. (3.5) and find the Tx power. If dynamic beamforming is employed (i.e., the selected IA beams are continuously changed), then we need to employ a model to determine the variations in beamforming gains (e.g., $(G_T^{CLT}(i) - G_R^{CLT})$) before using Eq. (3.5).

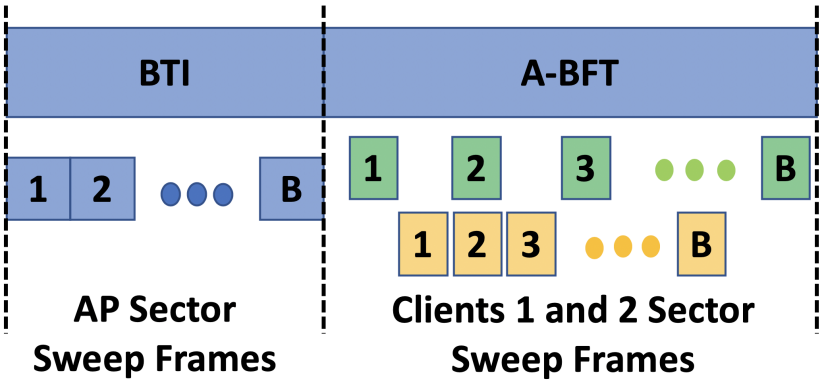


Figure 3.4: There are M mini-slots in the A-BFT interval. Each client randomly chooses B mini-slots and sends its SSW frames in them. The AP can acknowledge all clients during the ATI or as part of the A-BFT.

A-BFT, each client randomly selects B out of these M mini-slots and transmits on a different beam during each mini-slot. Fig. 3.4 shows this operation.

The only remaining task in JPOC's A-BFT design is to answer *how should the AP determine the optimal number of mini-slots?* We conduct theoretical analysis to answer this question. Let K denote the number of *good-beams* (i.e., the number of client beams with decodable SNR at the AP) and N the number of clients that are contending during A-BFT. We assume that K is the same across all clients[5]. We also assume that non good-beams do not collide with good-beams at the AP[6], i.e., only a good-beam transmission can be successful, and only if no other client transmits with a good-beam in the same mini-slot. Then, for a given client, the transmission probability in a mini-slot (p) is $\frac{K}{M}$ (we ignore non good-beam transmissions, as

[5]This is because with PC, the similarity in the number of good-beams across all clients increases. We will show this through experiments in Section 3.3.1.

[6]This is because non good-beams cause very little interference at the AP. For example, see the gray box in Fig. 3.3, which contains the non good-beams.

they would not be detected at the AP).

Now, consider a client that is currently transmitting with a good-beam. The probability of this good-beam transmission to be successful is equal to $(1-p)^{N-1}$.

Let P_0 denote the **targeted** failure probability for a client, i.e., the desired probability that a client that participates in A-BFT cannot establish a connection with the AP. P_0 is a design parameter used in JPOC. Our goal is to select a number of mini-slots (M) such that the probability of a client not being able to establish connection to the AP during A-BFT (i.e., $P[failure]$) is less than P_0. Thus, we have

$$P[failure] = (1-(1-p)^{N-1})^K \leq P_0 \tag{3.6}$$

Replacing p with $\frac{K}{M}$ in Eq. (3.6) and after a series of simplifications we have

$$\boxed{M_{opt} = \lceil \frac{K}{1-(1-P_0{}^{\frac{1}{K}})^{\frac{1}{N-1}}} \rceil} \tag{3.7}$$

Thus, with known K and N, the optimal M is the smallest integer that satisfies Eq. (3.7) ($\lceil\ \rceil$ shows the ceiling operator). In section 3.3, we conduct extensive experiments that show the statistics (e.g., average) number of good-beams with COTS devices. We use these statistics to choose K.

The only unknown parameter in determination of M_{opt} in Eq. (3.7) is the number of clients that will participate in A-BFT contention (i.e., N). We next propose a method so that the AP can estimate N. At a high level, JPOC uses statistics from contention in the previous A-BFT rounds to determine the number of competing clients in those rounds, and then uses those estimates to determine

the number of clients that would compete in the current A-BFT round.

Consider a **previously** completed A-BFT round, e.g., round t', which used a given number of mini-slots ($M^{t'}$). Upon completion of A-BFT in that round, the AP counts the number of A-BFT mini-slots in which the received energy was below the detectable SNR threshold. An empty A-BFT mini-slot means that either no client attempted to transmit in that mini-slot or the selected beam did not produce enough energy at the AP. Thus, we have

$$\frac{\#\ empty\ mini-slots}{M^{t'}} \approx (1-p)^{N^{t'}} = (1-\frac{K}{M^{t'}})^{N^{t'}}$$
$$\implies \boxed{N^{t'}_{\text{est}} \approx \frac{\log(\frac{\#\ empty\ mini-slots}{M^{t'}})}{\log(1-\frac{K}{M^{t'}})}} \tag{3.8}$$

Here, $N^{t'}_{est}$ is the AP estimate of the number of clients that competed in A-BFT round t'. Next, we find an average of $N^{t'}_{est}$ over the previous T rounds to compute the expected number of competing clients in the current A-BFT round t, as follows:

$$\boxed{N^{t}_{expected} = \frac{N^{t-1}_{est} + N^{t-2}_{est} + ... + N^{t-T}_{est}}{T}} \tag{3.9}$$

The averaging over the past T rounds is to dampen any oscillations. In our simulations, we observed smooth performance with rapid adaptation to system dynamics with $T = 5$. Thus, we set T to min(t,5) to also account for initialization.

JPOC's Mini-Slot Adaptation Algorithm: A JPOC AP continuously adapts the number of mini-slots (M) according to the system dynamics. It takes the following steps to determine the optimal number of mini-slots in the current A-

BFT round t: For a given number of good-beams (K) and desired client failure probability (P_0), it first uses Eq. (3.8) to determine the number of clients that participated in the previous T rounds of A-BFT contention. It then uses Eq. (3.9) to estimate the expected number of competing clients in round t and plugs that value into Eq. (3.7) to determine the optimal number of mini-slots that should be used in the current A-BFT round t. The selected value of M is then announced to all clients during the AP sector sweep (i.e., BTI). For the initial value of M (e.g., when an AP is initially turned on), we use a default value of 64 mini-slots. We also set the minimum value of M to 36.

3.2.3 Protocol Overhead

JPOC introduces three types of overhead that are added to each SSW frame transmitted by the AP during its sector sweep: (i) the desired maximum received power at the AP (RxP^{AP}_{max} from Eq. (3.2)), which even for $\gamma = 4$ and AP noise power of -90 dBm would at most need 8 extra signaling bits, (ii) ($G^{AP}_T(j) + P^{AP}_T - G^{AP}_R$), which assuming a 32-antenna array and 20 dBm transmission power would at most need 6 extra bits, and (iii) the number of mini-slots (M), which would be an extra 6 bits for up to 128 mini-slots. This minor increase in control bits substantially improves the system fairness and reduces the overall overhead as we show next[7].

[7]Eq. (3.5) only needs the value of ($RxP^{AP}_{max} + G^{AP}_T(j) + P^{AP}_T - G^{AP}_R$) to find P^{CLT}_T. Thus, we can combine (i) and (ii) to further reduce the signaling.

3.3 Performance Evaluation

In this section, we evaluate the performance of JPOC through both experiments and simulations. We first use our experimental setup from Section 3.1 to evaluate the power control aspect of JPOC with COTS devices. However, even the jailbreak framework does not allow us to change the A-BFT aspect of 802.11 ad. Thus, we next use simulations with a simplified channel model to characterize the accuracy of JPOC in determining the number of competing clients and adjusting the number of mini-slots, accordingly. Next, we use a comprehensive simulator with a standard compatible channel model and in the presence of system dynamics to characterize the full range of system performance in terms of fairness and protocol overhead.

3.3.1 Impact of Power Control (PC)

Experimental Setup. We use the same equipment and experimental setup of Section 3.1. In particular, we modified the default firmware on the sniffer router using the Nexmon framework. We then obtained the corresponding SNR dumps and used them to derive the results of this section.

Impact of PC on the Number of Good-Beams. We first study how client-side PC impacts its number of beams with detectable SNR at the AP (i.e., the number of the client's good-beams). We consider the indoor setup of Fig. 3.1(c), which provides a rich variety of channel conditions, and take multiple samples in each grid-cell with different client orientations. For a given client location and orientation, we obtain the SNR dump of all the client's beams for three different

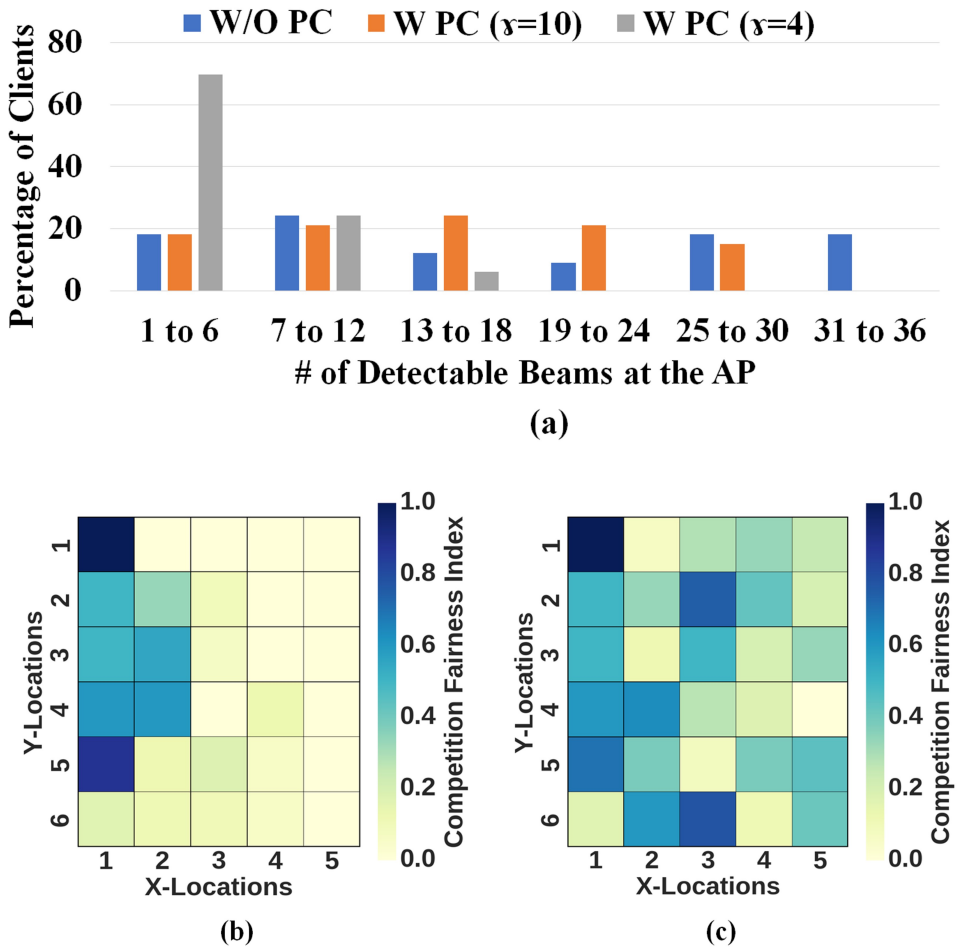


Figure 3.5: (a): Distribution of the number of good-beams with (W) and without (W/O) PC. With PC and $\gamma = 4$, close to 70% of clients would have 1-6 good-beams; (b) and (c): Competition fairness index for γ equal to 10 and 4, respectively. PC drastically improves fairness, particularly with a smaller γ. Note that even with PC, there are client locations that still cannot compete fairly in presence of the near-client. JPOC improves the performance of such clients through an appropriate selection of the number of mini-slots, so that these clients can also send their SSW frames.

power control (PC) mechanisms: (i) without (W/O) PC: each clients uses the same maximum transmission power; (ii) with (W) PC and $\gamma = 10$, and (iii) with PC and $\gamma = 4$. Recall from Section 3.2.1 that γ denotes the desirable SNR at the AP, i.e., each client adjusts its transmission power such that the resulting SNR of the best beam at the AP is equal to γ. Fig. 3.5(a) shows the distribution of good-beams across all experiments. We divide the number of good-beams (x-axis) into six brackets (from [1 to 6] to [31 to 36]), and plot the percentage of clients whose number of good-beams falls into each bracket. We observe that without PC (i.e., the default 802.11 ad implementation), clients are almost equally distributed across all the brackets. Further, clients could have as high as 31-36 or as low as 1-6 good-beams. When PC is employed, the number of good-beams would depend on the selected γ. As γ is reduced, the number of good-beams reduces, and the similarity in the number of good-beams across clients increases. For example, when γ is 10, most clients would have 1 to 24 good-beams, and close to 45% of clients would have 7-18 good-beams. Further, no client would have 31 or more good-beams. As γ is further reduced to 4, more than 90% of clients would have 1-12 good-beams, and no client would have 18 or more good-beams (in fact, our dataset shows that no client would have more than 14 good-beams). *Our results show that PC has the potential to increase the contention fairness among clients. The similarity in the number of good-beams increases across clients as PC is employed. Further, these beams would have a much closer range of SNR values at the AP.*

Impact of PC on Fairness. We next study the effect of PC in alleviating unfairness among competing clients. We consider the indoor environment discussed

in Fig. 3.1(c). Further, we consider the same extreme scenario: a near-client in (x,y) cell location (1,1) and the location of the *other* client in every other cell. We leverage the PC mechanism at all the locations and use two different γ values: 10 and 4. Next, we use the same setup of Fig. 3.1(c) to obtain the clients' beams SNR values and then derive the competition fairness index ($\frac{S^{other(j)}}{S^{near}}$) between the near-client and the *other* client at every other location j. Recall from Section 3.1 that S^{near} (i.e., near-client's success rate) is the fraction of near-client's beams with 3 dB or more SNR than the other client's beams. Fig. 3.5(b) and (c) depict the corresponding fairness index values for γ equal to 10 and 4, respectively. Note that we are considering a very extreme scenario, with the near client in a line-of-sight channel condition and very close to the AP. As a result, a majority of the near-client's beams have a high SNR. However, our results show that compared to the fairness heatmap of Fig. 3.1(c) that did not use PC, leveraging PC substantially improves the fairness. This is because with PC, different clients would *have more similar number of good-beams*, and with *closer SNR values at the AP*. The fairness further improves for a smaller γ, which is due an even more similarity between the number of good-beams, as we showed in Fig. 3.5(a).

Impact of PC on Fairness with higher beams. We further extend our study to understand the effect of PC in alleviating unfairness among competing clients with 64 beams. We consider the similar indoor environment as discussed in Fig. 3.1(c) with a near client as Asus-RoG with 64 beams, and the same extreme scenario in (x,y) cell location (1,1) and the location of the other Asus-RoG client with 64beams in every other cell. We obtain the clients' beams (64 beams) SNR values

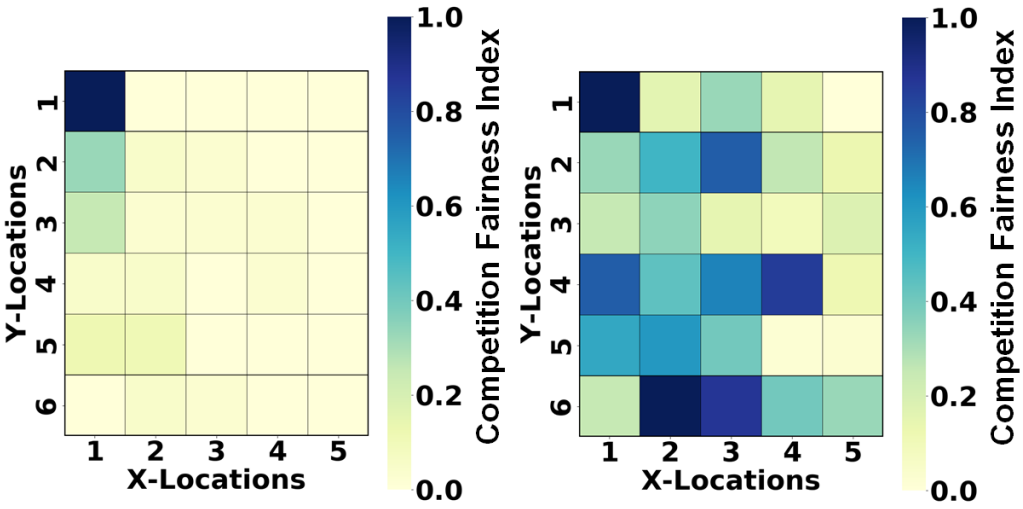


Figure 3.6: (a):Competition fairness index without Power Control for clients with 64beams. (b): Competition fairness index for γ equal to 4. PC drastically improves fairness, Irrespective of the number of beams.

and then derive the competition index ($\frac{S^{other(j)}}{S^{near}}$) without power control between the near-client and the other client at every other location j. Near clients dominate the far clients resulting in an unfair competing environment. Fig. 3.6(a) depicts the corresponding fairness index without power control. Similarly we derive the competition index with PC and a gamma equal to 4. Fig. 3.6(b) depicts the fairness Index values for gamma equal to 4. Further, our results show that compared to the fairness heatmap of Fig. 3.6(a) that did not use PC, leveraging PC substantially improves the fairness irrespective of the number of beams.

3.3.2 Comparison Against 802.11 ad/ay

Simulation Setup. We use a comprehensive mmWave simulator to evaluate JPOC's performance with standard-compatible channel models. We consider an indoor deployment with clients randomly and uniformly deployed in a 25 *m* radius cell and the AP deployed at the center of the cell. We create channels between every client and the AP according to the standardized channel model [72]. We set the center frequency to 60 GHz, and noise figure to 7 dB. All of our devices use the same channel for communication with 2 GHz of bandwidth. Each device has access to a 32-antenna (8x4) phased array with a single Tx-Rx RF chain, and uses 36 beams to cover 120 of Azimuth and 120 of elevation. We consider uplink traffic and assume that clients are fully backlogged with UDP traffic. We set the duration of a beacon interval to 100 msec and run each simulation realization for 10 seconds. We use a probabilistic model to create mmWave blockages [72]. When blockage happens, a client's data transmission gets lost and the client needs to compete in the next A-BFT round to obtain a new path to the AP. The blockage occurrence probability is a function of blocker density (e.g., number of humans) and is a variable in our simulator. Finally, we use a time-fair TDMA MAC scheduler at the AP. The scheduler equally divides the data transmission time interval between all of the associated clients and informs the clients about their schedules at the beginning of every ATI interval.

802.11 ad/ay Implementation. In addition to JPOC, we implement 802.11 in our simulator. Our implementation of 802.11's A-BFT design is according to the standard protocol (see chapter 2). Specifically, it does not perform any uplink

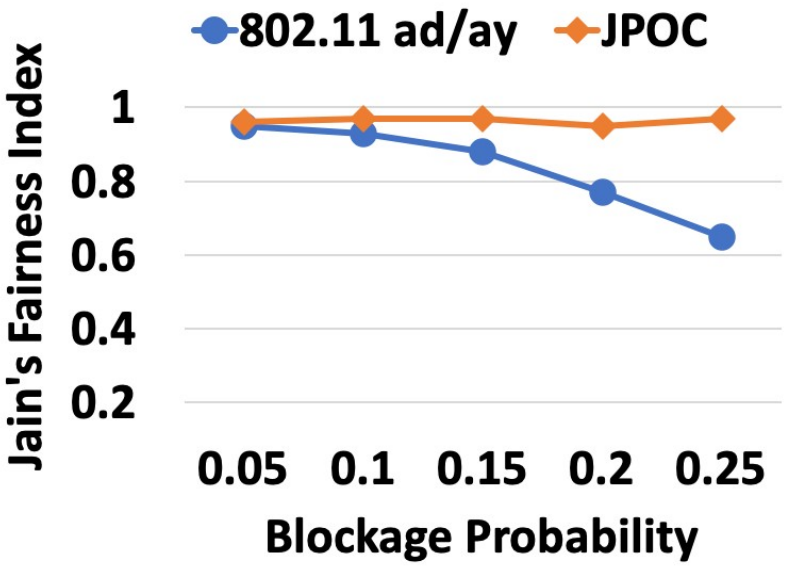


Figure 3.7: (a): Fairness comparison between JPOC and 802.11 ad/ay; (b): Overhead of 802.11 to JPOC with 10% targeted failure rate.

PC. Moreover, its A-BFT interval is composed of a few slots. Each slot is itself composed of a few mini-slots. The standard does not specify how a device should adapt the number of slots/mini-slots according to the traffic load. It only specifies the maximum allowed number of slots (e.g., 8 in ad [13]), and leaves the specific implementation to the chip manufacturer. For example, our 802.11 ad router does not change the number of slots according to the network traffic.

Fairness Definition. We compare Jain's fairness index across the different schemes. Let x_c denote the total time allocated to client c in a simulation realization. Then, Jain's fairness index is defined as $(\sum_{c=1}^{N} x_c)^2/(N\sum_{c=1}^{N} x_c^2)$. When the fairness value is closer to 1, it means that the distribution of air-time among clients is equal, whereas when the fairness index is less than one, it means that the air-

time distribution among clients is imbalanced. Note that as we discussed in our simulation setup, our AP uses a static TDMA schedule during the data transmission interval and divides the time equally among its associated clients. Thus, with a fair competition we expect the Jain's fairness index to be close to 1.

Fairness Comparison. Fig. 3.7(a) shows the Jain's fairness index values across two schemes: (i) 802.11 ad/ay, and (ii) JPOC. In our implementation of 802.11, we set the number of slots to 8, and the number of mini-slots to 36. Thus, a single client can fully perform its sector sweep in a single slot. The simulation corresponds to 16 clients and shows the fairness index as a function of blockage occurrence probability. We observe that JPOC maintains a high fairness among the clients irrespective of the blockage probability, whereas 802.11's performance drastically drops with a higher blockage probability. This is because JPOC (i) uses a PC mechanism that reduces the disparity in the number of good-beams across all clients, and (ii) adapts the number of mini-slots (M) according to its estimate of the number of competing clients.

Protocol Overhead. It is possible to improve 802.11's performance by using a higher number of slots. In this section, we examine the minimum required number of slots (and mini-slots) to achieve a desired client failure rate. Note that the 802.11 ad/ay standard do not specify how an AP should adapt its number of slots/mini-slots according to the traffic load. We set the targeted client failure rate to 10%, and assume a 20% blockage occurrence probability. Next, we conduct simulations to find the minimum number of 802.11 slots that meet the failure rate. Each of our 802.11 slots is composed of 36 mini-slots to accommodate all client sector sweep

frames. Fig. 3.7(b) shows the ratio of the number of 802.11 A-BFT mini-slots to JPOC as a function of the number of clients. We observe that the required number of mini-slots increases by more than 9x.

Chapter 4

Forward and Backward Data Transmission (FBDT) Across Multi RAT

4.1 Motivation

We conduct preliminary experiments to demonstrate the poor performance of MPTCP in the presence of link quality variations and motivate the design of FBDT. The experiment setup is composed of two Netgear Nighthawk X10 routers, which are connected to a server (Fig. 4.1(a)). The router supports both sub-6 GHz WiFi and 802.11 ad (mmWave WiFi or WiGig) technologies. We set one router as a WiFi BS and the other router as a WiGig BS. The two BSs are connected to the server using a 10G LAN SFP+ interface and at the other end the server is equipped with a 10G high speed Ethernet card. We use an Acer Travelmate laptop as a client to connect to both the WiFi and WiGig BSs. In all our experiments, the client remains static. Both the server and client run Ubuntu-22.04 with kernel version-5.18.0-rc7+. MPTCP-V1 is part of the upstream kernel and it is enabled on both the server and client. To study the performance of MPTCP in a controlled environment with configurable loss and delay profiles, we setup Traffic Controller (TC) between the

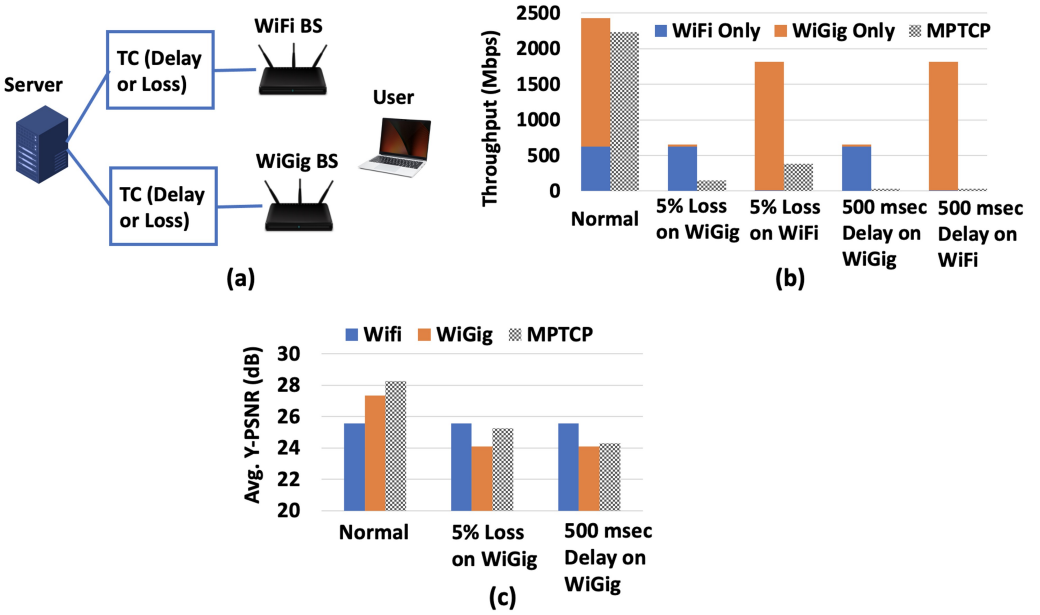


Figure 4.1: (a): We conduct experiments leveraging two Netgear Nighthawk X10 routers, a Dell Server and an Acer TravelMate laptop; (b): Throughput comparison of MPTCP's default scheduler (BLEST) and individual SPTCP throughput under ideal, lossy, and delay induced scenarios. Similar drop in performance was observed with minRTT scheduler.

BSs and the server. TC can be configured to induce controlled delay or loss on the traffic between the server and BSs. We use iPerf3 over MPTCP to generate the TCP traffic and log the throughput results for every 1sec. Each experiment lasts for 5 minutes, is repeated two times, and we take the average of the measurements to derive the average throughput values.

Throughput. Fig. 4.1(b) depicts the impact of packet loss or delay on the performance of each individual RAT as well as when the two RATs are used simultaneously by MPTCP. We conduct five sets of experiments: when there is no loss/delay introduced by TC (Normal), when TC introduces 5% packet loss on

either WiGig or WiFi links, and when TC increases the delay between packets by 500 msec on either WiGig or WiFi links. When no delay/loss is introduced (i.e., Normal), the standalone TCP rates of WiFi and WiGig RATs are about 630 and 1800 Mbps, respectively. We also observe that MPTCP achieves about 2200 Mbps, which is about 10% less than the summation of throughput across the individual RATs. Introducing packet loss or delay drastically reduces the throughout of the affected RAT in a standalone manner as well as when the two RATs are used by MPTCP. For example, introducing a 5% packet loss to the WiGig RAT reduces its standalone TCP rate to about 30 Mbps and the MPTCP rate to 147 Mbps. Similarly, introducing a 500 msec delay to the WiGig RAT reduces the individual TCP rate and even further reduces MPTCP throughput to 30 Mbps. Note that in either of these two scenarios, a standalone WiFi RAT achieves about 630 Mbps throughput, which shows there is a lot of room for improvement for Multi-RAT transport layer protocol design.

4.2 FBDT Design

In this section, we provide details of FBDT design. First, we describe the high level idea of our design and discuss how it can eliminate HoL blocking and provide a throughput that is close to the summation of throughput across individual RATs. Next, we discuss different components of the design, including scheduler, ACKs, sequence numbers, and congestion control, among others. Finally, we summarize how FBDT replaces different MPTCP components. For ease of discussion, we focus on two paths (RATs). We provide details on how FBDT extends to support

N paths (RATs) as well as a detailed pseudo-code of FBDT in the Appendix.

4.2.1 High Level Description

We propose Forward and Backward Data Transmission (FBDT) to eliminate HoL blocking and Out-Of-Order packet arrival issues when streaming data over multiple SPTCP(s). FBDT supports a *common meta-socket, send and receive buffer* to all the applications to send data across multiple SPTCPs. FBDT assigns sequence numbers to the data segments to be transmitted and starts transmitting data in small in-ordered batches - referred to as *transmission window*. For ease of discussion, we assume sequence numbers of packets transmitted from the transmission window are in ascending order (from right to left).

FBDT sends two sets of ordered packets from the transmission window - one with ascending and another with descending sequence numbers. Packets with ascending sequence numbers are sent as forward data transmission over one SPTCP (and the associated RAT) and packets with descending order are sent as backward data transmission over another SPTCP. For example, consider six packets to be transmitted over two SPTCPs with heterogeneous delays as shown in Fig. 4.2. FBDT starts by sending Packet1 and moving towards the end of the transmission window over SPTCP1 (*forward data transmission*). In parallel, FBDT starts sending Packet6 and moves towards the beginning of the transmission window over SPTCP2 (*backward data transmission*). FBDT relies on the fact that SPTCP delivers reliable and in-order packets to the receiver.

Eliminating HoL Blocking. FBDT eliminates HoL and Out-Of-Order packet

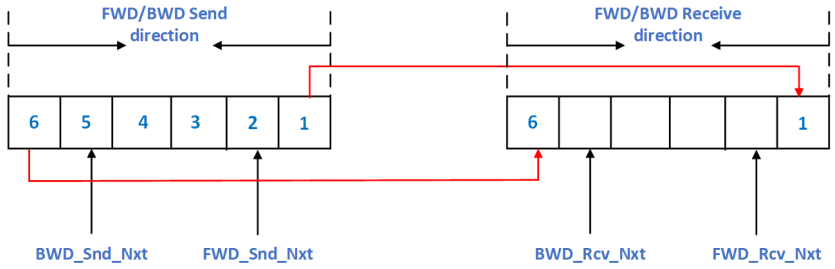


Figure 4.2: Here, transmission window size is six. FBDT serves packets from both forward and backward directions. The pointers showing the next forward and backward packets to be served are also illustrated.

arrivals by delivering an in-ordered small set of packets - referred as transmission window- to the application through meta sockets. For example, consider the setup depicted in Fig. 4.2, in which there are 6 packets to be transmitted from the transmissiom window. FBDT starts by sending Packet1, Packet2, ... from the forward data transmission over SPTCP1 and Packet6, Packet5, ... from the backward data transmission over SPTCP2. Suppose that Packet2 is delayed or lost over SPTCP1 due to the uncertainty of the wireless network. The backward data transmission would continue serving packets from backward including Packet2 over SPTCP2 since SPTCP1 was unable to deliver Packet2. As a result, FBDT will never encounter HoL blocking or Out-Of-Order packets at the meta-level since it will receive the packets either from the forward or backward directions.

Eliminating Re-Transmission and Re-Transmission Timeout. Unacknowledged packets in the FBDT transmission window are transmitted by either of the two SPTCPs or both in case of a delayed SPTCP transmission. FBDT will discard duplicate packets, which can happen due to delayed SPTCP transmissions. For

example, consider there are six packets in the FBDT transmission window as show in Fig. 4.2. Suppose that packet2's transmission is being handled by SPTCP1 but it is delayed. Meanwhile if backward transmission successfully transmits Packet2 over SPTCP2, then the receiver would receive Packet2. Now, suppose packet2 is also finally delivered to the receiver by SPTCP1- forming a duplicate packet. When this happens, the receiver will discard the duplicate packet based on the packets' sequence numbers. This eliminates the need for re-transmission and re-transmission timeout since the best effort solution adopted by FBDT ensures packet transmission on other SPTCP(s).

FBDT always Achieves the Optimal Rate. One of the key issues faced by traditional MPTCP architectures is to decide on how to optimally split the traffic across SPTCPs? Additionally, we strive for an architecture that can get the *summation* of individual RAT throughput values (in isolation) across all channel conditions. FBDT ensures optimal traffic splitting across RATs and as we will show later through experimental evaluation, achieves the desired *additive* throughput aggregation in all channel conditions. Recall that FBDT transmits data from both forward and backward directions and the two pointers move inwards (after successful ACK reception) at their own rates, meeting at a certain point. The rate at which the pointers move towards each other depends on the rate at which each of the individual SPTCPs is able to complete their transmission successfully. As a result, this mechanism automatically splits the traffic optimally between SPTCP(s) based on their individual throughput values. Using the TCP throughput equation in [73], we can theoretically derive the number of packets served from the forward

direction as:

$$N_{fwd} = N(\frac{W_{fwd}RTT_{bwd}}{W_{fwd}RTT_{bwd} + W_{bwd}RTT_{fwd}}) \tag{4.1}$$

Here RTT_{fwd} and W_{fwd} denote the forward RAT RTT and forward path transmission window size, respectively. N denotes the total number of packets to be served. The number of packets to be served from the backward direction can be derived by swapping fwd subscripts with bwd and vice versa.

4.2.2 Detailed Architecture

We now discuss the details of different components of FBDT.

FBDT Transmission Window. To transmit in both forward and backward directions, FBDT buffers data in a transmit buffer before transmitting across SPTCP(s). We refer to this buffer as FBDT transmission window. The size of this buffer can be as low as $(1 + \lceil \max(\frac{R_{bwd}}{R_{fwd}}, \frac{R_{fwd}}{R_{bwd}}) \rceil)$ bytes. For example, suppose the average forwrad and backward throughput values (R_{fwd} and R_{bwd}) are 2Gbps and 400 Mbps, respectively. Then the minimum size of transmit window should be 6 bytes. In our experiments, we fix the size of this window to 64 KB. FBDT sender and receiver will agree on the transmission window size before transmission and can periodically adapt that to accommodate any dynamics in RATs' data rates.

FBDT Sequence Numbers. To ensure in-order data delivery, FBDT assigns unique Sequence numbers to the data segments to be sent by SPTCPs. These unique sequence numbers are also used by the receiver to discard duplicate packets as they are delivered to the receive meta socket.

FBDT Acknowledgements. ACK(s) at the FBDT level are necessary for both forward and backward transmission pointers to advance inwards in the FBDT transmission window. In SPTCP, when an ACK is sent by the receiver, it also includes information about the next set of packets that are expected to be sent by the sender. MPTCP also uses ACKs at its level. In particular, MPTCP piggy backs its ACKs on the option fields of the SPTCP ACKs. The information includes the expected next set of packets (Data Segment Sequence numbers or DSSs) MPTCP expects to receive. We use the same idea of piggybacking information on SPTCP ACKs. However, as FDBT leverages two (forward and backward) data transmission paths, our piggyback data specifies the next set of packets (sequence numbers) the receiver expects to receive on both the forward and backward directions. This idea can significantly boost performance when there is system dynamics. For example, consider a dual radio WiFi + WiGig setup. Suppose there is an outage on the uplink transmission of WiGig, which blocks its TCP ACKs. In this case, downlink WiGig data packets would still be acknowledged through FBDT ACKs transmitted by WiFi. As a result, there would be no need for redundant transmission of WiGig data packets over WiFi.

In-Order Delivery Across FBDT Transmission Window. Forward and backward send windows move inward as they receive FBDT ACKs. The two send windows will meet at a certain point based on the throughput of the two directions. As they reach this point, it is possible that one of the SPTCP send windows has completed successfully and ready to move on to next set of data, while the other sliding window is still waiting for ACKs of the data transmitted.

When this situation happens, the early completed sliding send window would wait for at most RTT of the other transmission end, and then would proceed to transmitting unacknowledged packets in the other sliding window. This could be redundant information transmission but is required to ensure a high overall system throughput. As packets are checked based on their sequence numbers at the receiver, they will be dropped if a duplication occurs at the receiver. For example, assume a transmission window size of ten with ten packets for transmission ordered from one to ten. Forward and backward transmission start by sending from one and ten, respectively, and proceed to moving inwards. Let us assume forward sliding window has successfully completed sending packet three and backward has transmitted packets five and four and is waiting for their ACK(s). The forward sliding window will wait for an RTT_{bwd} before transmitting packets 4 and 5 over its SPTCP. After transmitting this redundant data, the forward or backward will not wait for the ACK(s) any longer since the data should be reached either by forward or backward, which ensures in-order delivery as well. As a result, the data in the transmission window is updated with the next set of ten packets to be transmitted from the meta-socket buffer.

RAT to Direction Mapping. The mapping of RATs to forward or backward directions impacts the performance. In FBDT, the forward direction is mapped to the more reliable RAT (e.g., WiFi) and the backward direction is mapped to the less reliable RAT (e.g., WiGig). Suppose the reverse mapping and packets 1 to 6 in the transmit window size. If WiGig is assigned to the forward direction but is blocked, packet 1 would take a long time until it's reached at the receiver. This also

blocks delivery of packets 6, 5, and 4 (which are sent on WiFi) to the receive meta socket and application. Assigning the more reliable RAT to the forward direction removes this type of blocking. FBDT can also dynamically adapt this assignment based on historical RATs' performance.

FBDT Scheduler. FBDT schedules segments from its transmit window to the SPTCPs from the forward and backward directions by maintaining Send_Window$_{fwd}$ (with W_{fwd} size) and Send_Window$_{bwd}$ (with W_{bwd} size) sliding windows, respectively. For each of the two (forward and backward) directions, the size of these send windows are determined similar to how SPTCP calculates them, i.e., *send_window* = Min(*cwnd*, *rwnd*). Additionally, FBDT maintains separate Snd_Una[1] and Snd_Nxt pointers for both forward and backward sliding windows (see Fig. 4.3). FBDT schedules packets to SPTCPs depending on the available space in *cwnd* and moves Snd_Una and Snd_Nxt appropriately in their FBDT transmission windows. FBDT sender will move the Snd_Una pointers of both forward and backward when it receives the corresponding acknowledgments at FBDT level. Additionally, FBDT receiver will acknowledge both forward and backward with the expected Fwd_Snd_Nxt and Bwd_Snd_Nxt, respectively. Finally, FBDT moves Snd_Nxt pointer of forward and backward as it schedules packet to SPTCP for transmission. The two sliding windows will meet at a point depending on the throughput of forward and backward SPTCPs. FBDT decouples the two congestion control algorithms (CCAs) and lets each CCA to decide on its transmission based on the congestion and reliability of its underlying network.

[1]Una stands for Un-acknowledged.

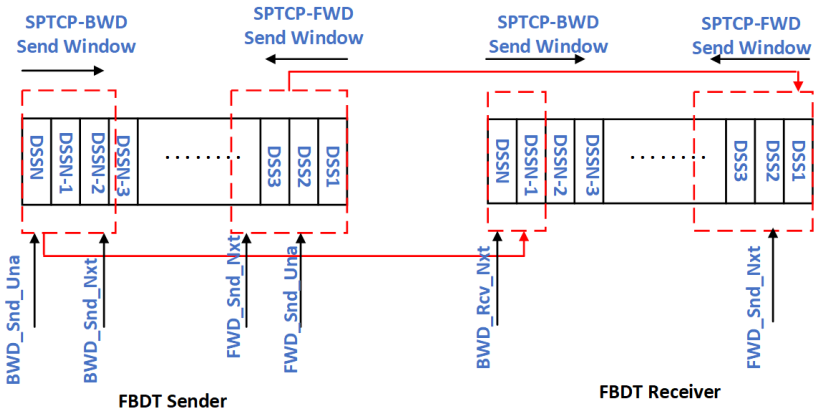


Figure 4.3: FBDT Scheduler.

FBDT completes serving its transmission window when Fwd_Send_Nxt and Bwd_Send_Nxt are equal or cross over each other. At this point, FBDT will wait for either of the Snd_Una to meet their Send_Nxt before redundant packet transmission begins. Suppose backward transmission was able to successfully complete earlier and both Fwd_Send_Nxt and Bwd_Send_Nxt has crossed over each other. Then backward transmission of FBDT will wait for RTT_{fwd} and start transmitting redundant unacknowledged packets in the forward sliding window. FBDT will not wait for the ACKs of redundant packets since the packets are scheduled in both of the SPTCPs, which will likely deliver them in-order to the receiver. FBDT will move on to load the transmission window with the next set of data and proceeds with their transmission.

The rate at which Send_Window_{fwd} and Send_Window_{bwd} move towards each other depends on the rate (throughput) of each SPTCP. As FBDT adds up the throughput of each individual SPTCP, we can approximate its total throughput

leveraging SPTCP throughput formula [73] as:

$$Throughput_{FBDT} = \gamma \times (\frac{W_{fwd}}{RTT_{fwd})} + \frac{W_{bwd}}{RTT_{bwd})}) \tag{4.2}$$

Here, γ is a scalar factor that accounts for overhead. In practice, we have observed that FBDT throughput is very close to the summation of individual RAT data rates, irrespective of the channel conditions. In other words, γ is very close to one.

4.2.3 FBDT Design for N RAT(s).

FBDT design for two SPTCPs can be extended to N number of SPTCPs. Suppose we have N number of SPTCPs. Let $FBDT_{SPTCP} = \{SPTCP_1, SPTCP_2, \ldots, SPTCP_N\}$, where $FBDT_{SPTCP}$ is the set of SPTCPs - sorted based on reliability.

The most reliable SPTCP will serve as forward transmission SPTCP and the other SPTCPs will serve as backward transmission as shown in Fig.4.4. FBDT considers lower sequence numbers in the transmission window as high priority and expects it to be delivered to the receiver through the more reliable SPTCP network. FBDT uses backward SPTCP network to transmit lower priority packets (higher sequence numbers). Suppose that the backward SPTCP network can't deliver the packets due to its network connection unreliability. Then the forward network will eventually transmit the backward data. Though reliable network connection are comparatively slower, using most reliable network connection as forward transmission guarantees reliable and faster packet delivery (for a detailed example of this phenomenon, please refer to "RAT to Direction Mapping" paragraph in

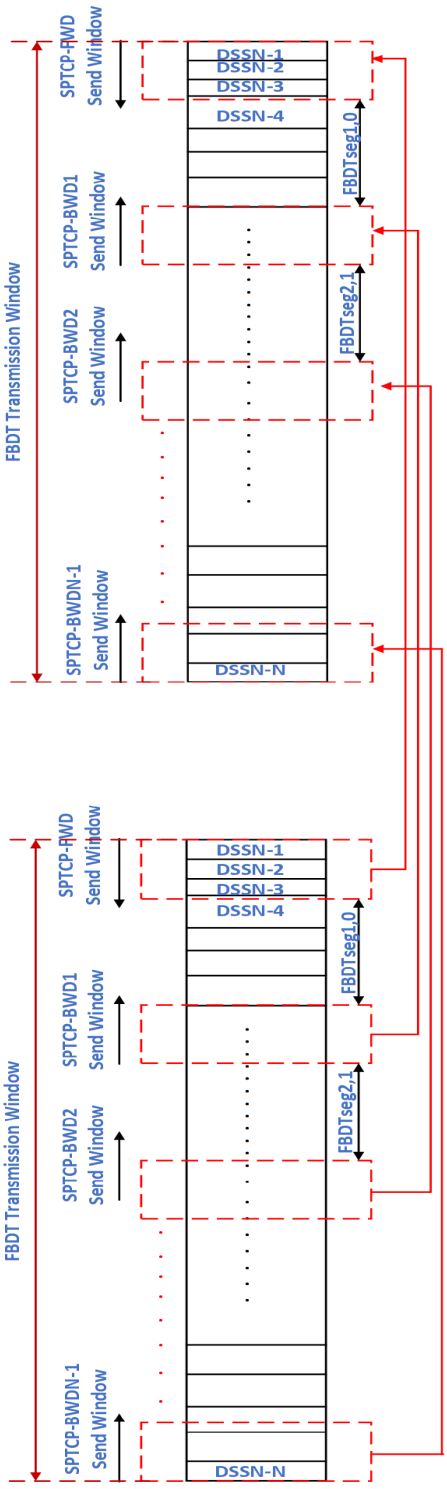


Figure 4.4: FBDT scheduler with N number of SPTCPs. There is one SPTCP-FWD and $N-1$ SPTCP-BWD.

Section 3.2). The overall throughput of FBDT can be approximated as:

$$Throughput_{FBDT} = \gamma \times (\frac{W_{fwd}}{RTT_{fwd}} + \sum_{i=1}^{i=N} \frac{W_{bwd_i}}{RTT_{bwd_i}}) \tag{4.3}$$

To avoid under-utilizing backward SPTCPs, we place them from the forward SPTCP such that there is a sufficient gap between SPTCPs. Each SPTCP uses *send_window* = Min(*cwnd*, *rwnd*) to send data and *cwnd* will grow/shrink based on the underlying network congestion. In FBDT, the backward and forward transmission send windows move towards each other. To avoid crossing over each other quickly, the backward SPTCPs has to be placed far enough so that they have sufficient packets to fill the *cwnd* of the SPTCP. To achieve the highest possible FBDT throughput, backward transmission segment size and offset are vital. Each backward SPTCP is placed at $FBDT_{offset}$ such that it has sufficient packets to fill its SPTCP. The best way to estimate the segment size and offset of backward SPTCP is based on estimated throughput. To begin with, FBDT considers no packet loss and shadows RTT values based on the corresponding SPTCP RTT values. Based on the throughput estimation, the segment size and offset can then be set by:

$$SegSize_{j,j-1} = N(\frac{Throughput_j}{\sum_{k=1}^{k=N-1} Throughput_k}) \tag{4.4}$$

$$SegOffset_{j,j-1} = \sum_{K=1}^{K=j-1} SegSize_{K,K-1} \tag{4.5}$$

Here, j and $j-1$ denote two adjacent backward SPTCPs and N is the total

number of bytes in the transmission window.

Though FBDT places backward SPTCP segments at the appropriate offset but due to network throughput variations, some of the SPTCPs will complete their assigned segments earlier than other SPTCPs and tend to cross over into other SPTCP segments. To handle this, when an SPTCP completes its assigned segments then FBDT will recompute its offset and segment sizes in the transmission window. Once the new SPTCP's alignment in the transmission window is determined, FBDT sender sends a message to the receiver with the new alignment before resuming transmission. The entire functionality of FBDT is elucidated through a pseudocode provided in Algorithm- 1.

Algorithm 1 FBDT Pseudo-Code

1: $SPTCP \in SPTCP1, SPTCP2, ...SPTCPn$
2: $TotalBytes \leftarrow 0$
3: **function** FBDTSENDDATA(SPTCP)
4: **if** $TotalBytes = TransWindowSize$ **then**
5: **return**
6: **end if**
//Check if all the seg have data to transmit
7: **if** CheckAlignment==False **then**
8: ALIGNFBDTSENDWIN($FwdSPTCP$)
9: **end if**
10: $BytesSent \leftarrow$ SENDSPTCP$(SPTCP, W_{SPTCP})$
11: $TotalBytes \leftarrow TotalBytes + BytesSent$
12: $SPTCP \leftarrow NextSPTCP$
13: FBDTSENDDATA(SPTCP)
14: **end function**

15: **function** ALIGNFBDTSENDWIN($SPTCP$)
16: **if** SPTCP=NULL **then**
17: **return**
18: **end if**
19: **if** SPTCP=FwdSPTCP **then**
//Arrange UnAck pkt contiguously
20: REALIGNTRANSWINBUF(TransWin)
21: $C \leftarrow \dfrac{N}{TotalThroughput}$
22: $SegSize[SPTCP] \leftarrow C \times \dfrac{W_{fwd}}{RTT_{fwd}}$
23: $SegOffset[SPTCP] \leftarrow 0$
24: **else**
25: $SegSize[SPTCP] \leftarrow C \times \dfrac{W_{bwd}}{RTT_{bwd}}$
26: $SegOffset[SPTCP] \leftarrow PrevSegOffset$
27: **end if**
28: $PrevSegOffset \leftarrow SegSize[SPTCP] + PrevSegOffset$
29: ALIGNFBDTSENDWIN($NextSPTCP$)
30: **end function**

FBDT Congestion Control. FBDT supports both coupled and decoupled Congestion Control Algorithms (CCAs). Out of the box design supports decoupled CCA where it lets each of the two SPTCPs' congestion control algorithms work to the fullest extent without FBDT's interference. However, FBDT can be easily extended to coupled congestion control by controlling the amount of packets scheduled for each of the two SPTCP(s). Coupled CCA is sometimes preferred over decoupled CCA since it is shown to better maintain fairness over bottleneck links **coupled1**, **coupled2**, [39].

IETF study on MPTCP suggests three goals for a practical coupled CCA [39]: (i) Improve Throughput, i.e., higher performance than a single path TCP, (ii) Do no Harm, i.e., not take up more capacity from any of the resources shared by its different paths than if it were a single flow using only one of these paths, and (iii) Balanced Congestion: move as much traffic as possible off its most congested paths, subject to meeting the first two goals. We next show that we can achieve these goals by optimizing the scheduler and therefore, by regulating the amount of packet scheduled to the individual SPTCP(s) without the need for SPTCP modifications[2].

Goal 1 (Improve Throughput). Since FBDT uses forward and backward data transmission it gives equal chance to both the SPTCPs to transmit to the fullest extent possible. If any of the two SPTCPs performs poorly, the other SPTCP will transmit the data without the need to wait for the other SPTCP to complete. For example, consider FBDT has assigned SPTCP1 and SPTCP2 to transmit from forward and backward, respectively. If SPTCP1 is unable to transmit due to the

[2]We leave a comprehensive design of *fair* coupled CCA for FBDT as part of our future work.

underlying network uncertainty, SPTCP2 will transmit all the packets from the backward, compensating for SPTCP1's poor performance. In the worst case, if one SPTCP is completely blocked, FBDT will achieve a throughput that is equal to the other SPTCP's throughput. As a result, FBDT will *always provide additive throughput gains.*

Goal 2 (Do no Harm). FBDT can be extended to ensure fairness towards single path clients without modifying the SPTCP protocol. To achieve fairness, we let the *cwnd* of the SPTCP(s) untouched, and introduce scaling factors α and β to scale up the forward and backward send window sizes, respectively. By controlling α and β we can regulate the amount of traffic scheduled for each SPTCP.

Goal 3 (Balanced Congestion). FBDT by design optimally divides the traffic across SPTCPs. For example, if Send_Window$_{fwd}$ is congested, by design FBDT will keep on transmitting from the backward direction. Thus, this aspect of CCA is naturally handled by FBDT.

4.2.4 FBDT: Beyond MPTCP

MPTCP-IETF standards propose different components for the protocol such as schedulers, CCAs, re-transmission and Out-Of-Order queues at MPTCP level on top of similar components at the SPTCP level. FBDT breaks this design by relying on strategic scheduling and the conventional SPTCP for reliable transmission, congestion control and retransmission. To summarize, FBDT eliminates retransmission protocols by sending redundant packet over the other subflows, eliminates additional congestion control by controlling the amount of packets scheduled to

each SPTCP and eliminates re-arranging OOO packets through a forward and backward data transmission scheme. We believe that MPTCP has overlooked conventional SPTCP's congestion control and reliability mechanisms, and is lavish in proposing additional retransmission, congestion control and re-arranging OOO packets at the MPTCP level. This comes with an additional cost, adds complexity, and reduces the overall system throughput.

4.3 Implementation

Initially, We implemented FBDT as a daemon in the user space in Linux on both the client and server. Client connects to the server using two different BSD SPTCP sockets - one for WiFi and another for WiGig. FBDT maintains a transmission window using char array - whose size is configurable. The FBDT daemon on the server sends packets through FBDT transmission window. Every segment in the transmission window -set to Maximum Segment Size (MSS)- is numbered with an FBDT sequence number. As mentioned in the design section, we use the more reliable RAT - WiFi- as forward transmission and the less reliable RAT - WiGig- as backward transmission. FBDT transmit window on the server maintains FWD pointer and BWD pointer for both forward and backward transmissions. Both pointers move inward as cumulative ACKs are received from the client. In our implementation, the client sends a cumulative ACK on each transmission from the user space with both forward and backward ACKs. FBDT receive window on the client also maintains a FWD and BWD pointer, and they move inward as they receive packets from the server. When both pointers meet at some point within the

transmission window, the transmission window is reloaded with a new set of data to be transmitted.

Further, We extended the implementation of FBDT for multi-RAT communication, configuring it as a daemon in the user space in Linux for both the client and server. Client connects to the server using two different BSD SPTCP sockets - one for WiFi and another for WiGig. FBDT maintains a transmission window using char array - whose size is configurable based on the class of application. The FBDT daemon on the server sends packets through FBDT transmission window. Every segment in the transmission window -set to Maximum Segment Size (MSS)- is numbered with an FBDT sequence number. As mentioned in the design section, we use the most reliable RAT - WiFi- as forward transmission and the less reliable RAT(s) - WiGig, ethernet- as backward transmission. Based on the throughput Equation-4.5, FBDT transmit window on the server maintains FWD pointer and BWD1, BWD2, BWD3 etc., pointers for forward and all the backward transmissions. Forward and all backward pointers move inward as cumulative ACKs are received from the client. In our implementation, the client sends a cumulative ACK on each transmission from the user space with both forward and all the backward ACKs. FBDT receive window on the client also maintains a FWD and one pointer for each BWD transmission, and they move inward as they receive packets from the server. When any of the two pointers meet at some point within the transmission window, then the FWD and BWD pointers are re-estimated as per Equation-4.5. FBDT resumes its transmission with the new pointers moving inwards. Once all pointers (FWD and BWD) meet at a point inwards, the transmission window is

reloaded with a new set of data to be transmitted.

Migrating FBDT to the Linux Kernel. We have migrated FBDT to the latest Linux Kernel-5.18-rc7+ and replaced the existing MPTCP protocol. This includes replacing the default MPTCP scheduler as well as the protocol with FBDT. Additionally, we modified the (i) protocol.c file of mptcp in the kernel, (ii) mptcp_sendmsg with FBDT transmission algorithm, and (iii) mptcp_send_ack and subflow ack with FBDT ACK(s). These are piggy bagged on TCP ACK(s). As per FBDT design, we eliminated MPTCP retransmission logic, Out-Of-Order buffer logic, and retransmission timeout from the MPTCP code, since it is no longer needed by FBDT. We plan to publicly release our software on GitHub prior to thesis defense so that other researchers in the community can reproduce our results and expand on our software.

4.4 Evaluation

In this section, we evaluate the performance of FBDT through experiments. We first conduct experiments to quantify the throughput that can be achieved with different MPTCP protocols and under different channel and client mobility conditions. We mostly consider a dual WiFi+WiGig setup but we also study FBDT performance with three RATs. Finally, we study the viewport quality under different MPTCP protocols and tile ordering and coding schemes. In all of our experiments, we only consider a single client. We leave performance evaluation in multi-client or multi-BS (e.g., multiple WiGig and WiFi BSs) scenarios as part of our future work.

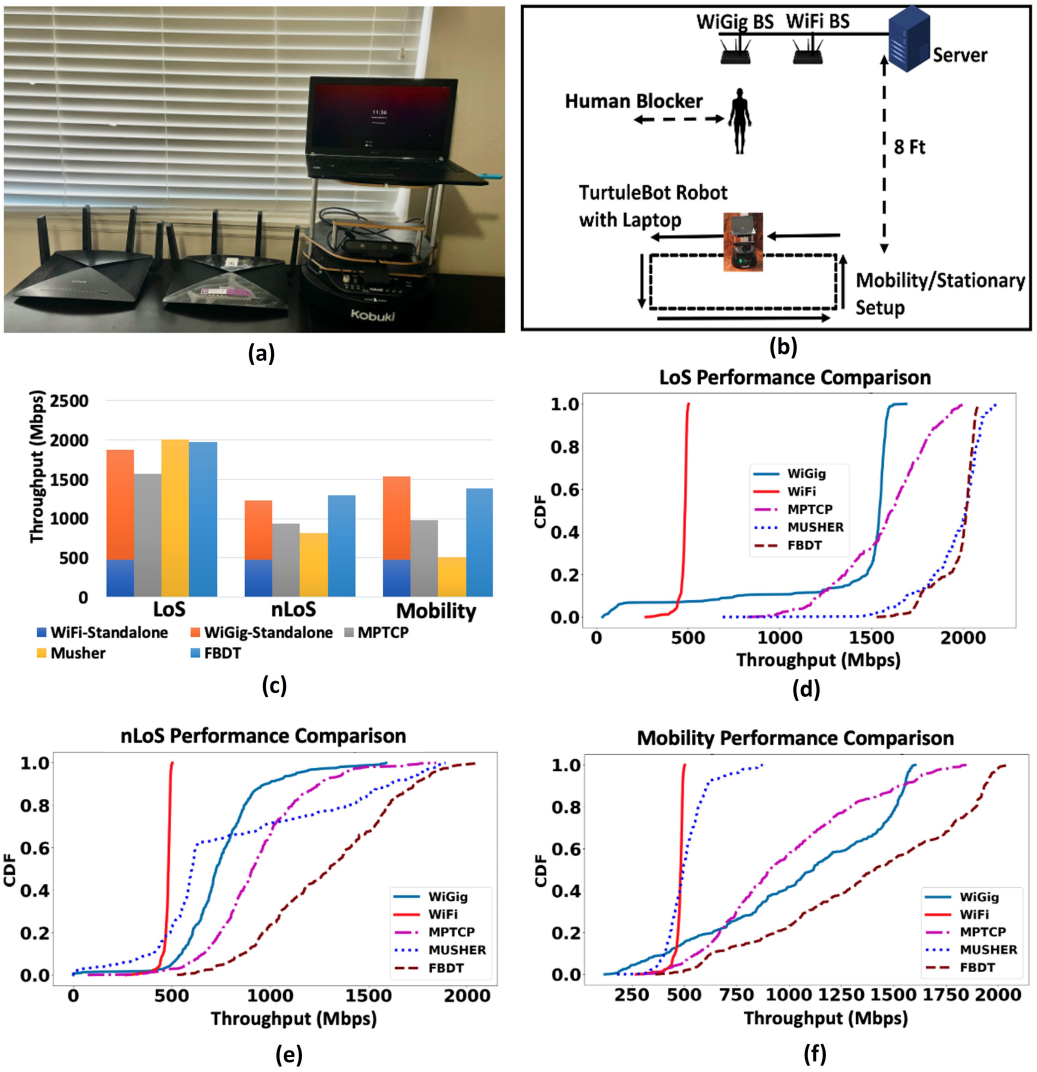


Figure 4.5: (a): Our hardware setup. TravelMate laptop is placed on top of TurtleBot robot for controlled mobility, (b): Network deployment layout. A human stands in front of the WiGig BS to create blockage, (c): Average throughput results across stanalone WiFi, standalone WiGig, MPTCP (BLEST), MuSher, and FBDT in LoS, nLoS, and mobility, (d): LoS CDFs, (e): nLoS CDFs, and (f): Mobility CDFs. The spread in measured throughput across all schemes (except for standalone WiFi) increases as channel condition changes from LoS to nLoS/mobility. MuSher has a low throughput spread under mobility, which is because the scheme primarily resorts to using only WiFi for communication.

4.4.1 Experimental Setup

Network Deployment. Fig. 4.5(a) depicts some of our hardware. Our network is composed of two Netgear Nighthawk X10 routers. These routers support both WiFi and WiGig. We set one router as WiFi only and the other router as WiGig only to emulate a non-colocated BS scenario. The routers are connected to a Dell server with connections discussed in Section 4.1. Our client device is an Acer TravelMate laptop that has both WiFi and WiGig cards. This laptop is placed on top of a TurtleBot robot. The robot can be programmed to stay stationary or mobile with a configurable speed and mobility pattern. These devices are deployed in an indoor office environment depicted in Fig. 4.5(b).

Channel Conditions and Mobility Patterns. We consider three different scenarios: (i) LoS: In this setup, the client (TravelMate laptop) is placed at a fixed location about 8 feet from the WiGig BS. The client has a LoS channel to both BSs and remains fixed at its location throughout the experiment, (ii) nLoS: In this setup, the client remains fixed at the location similar to the LoS experiment but a human blocker stands about 3 feet in front of the WiGig BS throughout the experiment, (iii) Mobility: In this setup, the client device (on top of robot) moves in a rectangular pattern of 6ft by 2ft, as depicted in Fig. 4.5(b). There is no human blocker between the robot and the BSs. In this setup, the client initially faces directly the BSs in a LoS channel condition. But then the robot turns 90° followed by two other 90° turns. In these positions, the client channel becomes nLoS because the body of the laptop blocks the LoS path. As a result, in this setup the client frequently switches between LoS and nLoS channel conditions.

Traffic Generation. We use iPerf to generate downlink TCP traffic from the network to the client. This communication lasts for 5 minutes. We repeat each experiment two times and plot the average and CDF curves (with throughput sampled across one second intervals).

Implemented Solutions. We experimented with the following protocols: (i) FBDT: our proposed architecture, (ii) MuSher [54], which is the state-of-the-art MPTCP scheduler designed for dual radio WiGig+WiFi setups. We used the software code that was publicly released by MuSher authors to implement it in our hardware. We have been able to successfully replicate their results; (iii) MPTCP: As we mentioned in Section 2.3, minRTT and BLEST are two of the default MPTCP schedulers implemented in Linux Kernel. We present only the results achieved under BLEST as in all of our experiments BLEST outperformed minRTT. Finally, in addition to the above MPTCP protocols, we also conduct SPTCP experiments when only one RAT is used for communication (e.g., when only WiFi or WiGig BS is turned on).

4.4.2 Throughput Statistics

Throughput in Dual Radio WiFi+WiGig Setup. Fig. 4.5(c) depicts the average throughput results across all protocols, channels, and mobility conditions. Recall that in LoS/nLoS experiments, the client remains stationary. Under standalone LoS configuration (leftmost bar), WiFi and WiGig achieve an average throughput of 450 and 1400 Mbps, respectively. In standalone mode, we use SPTCP and only a single RAT to measure throughput. MPTCP (with BLEST) aims to use the fastest

subflow (i.e., WiGig) most and directs an estimated amount of bytes on the slower subflow with a combined throughput of 1550 Mbps. FBDT and MuSher utilize both subflows to the fullest extent achieving a combined throughput of 1970 Mbps. This is 20% higher than MPTCP, and 40% and 4.5x better than standalone WiGig and WiFi, respectively. The CDF of the throughput variations in the LoS condition is plotted in Fig. 4.5(d). The throughput of MPTCP scheduler varies from 1000 Mbps to 1800 Mpbs, whereas FBDT gives a throughput of 1970 Mbps 80% of the time.

The second group of bars in Fig. 4.5(c) show the throughput value in stationary nLoS condition. First, we observe no change in WiFi standalone throughput, whereas WiGig drops by almost 50%. This is because sub-6GHz communication is mostly immune to human blockage, whereas WiGig beam switching to a reflective nLoS path significantly drops the signal SNR and throughput. We also observe that MPTCP gets a combined average throughput of 930Mbps, which is lower than its LoS throughput and is due to drop in WiGig performance. MuSher gets 814 Mbps throughput, which is even lower than baseline MPTCP. On the other hand, FBDT provides additive throughput gains with an average of 1300 Mbps. In fact, FBDT throughput is slightly higher than simple summation of average WiFi and WiGig standalone rates. The slight difference (higher or lower) than pure summation of individual RAT data rates can be attributed to the following: (i) each time we run an experiment, we naturally gets some variation in throughput values. We try to minimize this by running two experiments for each measurement but one would need to repeat experiments more to minimize this impact; (ii) FBDT has several

mechanisms such as piggybacked ACKs, duplicate transmissions, etc. built in to the protocol, which can add or remove from the overall system overhead depending on the experiment. The corresponding nLoS CDF curves are plotted in Fig. 4.5(e). We observe a much higher increase in throughput spread across all schemes except for standalone WiFi. This is because the WiGig RAT routinely switches its beam in nLoS channels creating data rate fluctuations.

The last throughput bars in Fig. 4.5(c) show throughput values under client mobility. We observe no major change in WiFi throughput, whereas WiGig throughout is higher than its rate in stationary nLoS condition. This is because under mobility the client switches between LoS and nLoS channels. In LoS channels, WiGig benefits from much higher throughput values, which results in a higher spread as shown in the corresponding CDF throughput curve (Fig. 4.5(f)) and a higher average throughput (Fig. 4.5(c)). On the other hand, MPTCP is not able to benefit effectively from increase in the average WiGig rate as its throughput remains close to its nLoS throughput. This is because the BLEST scheduler is too conservative in its estimation of WiGig blockage, which stops MPTCP from fully benefiting from WiGig when mobile client switches to LoS. We also observe that MuSher gets a performance that is close to half of baseline MPTCP, while FBDT gets a throughput that is slightly less than pure summation of standalone WiFi and WiGig data rates. There are several reasons for MuSher's poor performance. First, our baseline MPTCP is the newest implementation of MPTCP in the Linux kernel and is more up to the date than the baseline compared against in MuSher [54]. Second, our mobility pattern is much more complex than the mobility pattern

studied in MuSher, in which the client remains in LoS and moves towards, away or left-right in front of the BS. Our mobility pattern involves frequent switching between LoS and nLoS as expected in real-world environments. Third, we have observed that MuSher is unable to timely estimate the bandwidth of different RATs, which makes the protocol sub-optimally split the traffic between RATs with much more emphasis on WiFi to a degree that it achieves even a lower performance than MPTCP.

The second group of bars in Fig. 4.5(c) show the throughput value in stationary nLoS condition. First, we observe no change in WiFi standalone throughput, whereas WiGig drops by almost 50%. This is because sub-6GHz communication is mostly immune to human blockage, whereas WiGig beam switching to a reflective nLoS path significantly drops the signal SNR and throughput. We also observe that MPTCP gets a combined average throughput of 930Mbps, which is lower than its LoS throughput and is due to drop in WiGig performance. MuSher gets 814 Mbps throughput, which is even lower than baseline MPTCP. On the other hand, FBDT provides additive throughput gains with an average of 1300 Mbps. In fact, FBDT throughput is slightly higher than simple summation of average WiFi and WiGig standalone rates. The slight difference (higher or lower) than pure summation of individual RAT data rates can be attributed to the following: (i) each time we run an experiment, we naturally gets some variation in throughput values. We try to minimize this by running two experiments for each measurement but one would need to repeat experiments more to minimize this impact; (ii) FBDT has several mechanisms such as piggybacked ACKs, duplicate transmissions, etc. built in to

the protocol, which can add or remove from the overall system overhead depending on the experiment. The corresponding nLoS CDF curves are plotted in Fig. 4.5(e). We observe a much higher increase in throughput spread across all schemes except for standalone WiFi. This is because the WiGig RAT routinely switches its beam in nLoS channels creating data rate fluctuations.

The last throughput bars in Fig. 4.5(c) show throughput values under client mobility. We observe no major change in WiFi throughput, whereas WiGig throughout is higher than its rate in stationary nLoS condition. This is because under mobility the client switches between LoS and nLoS channels. In LoS channels, WiGig benefits from much higher throughput values, which results in a higher spread as shown in the corresponding CDF throughput curve (Fig. 4.5(f)) and a higher average throughput (Fig. 4.5(c)). On the other hand, MPTCP is not able to benefit effectively from increase in the average WiGig rate as its throughput remains close to its nLoS throughput. This is because the BLEST scheduler is too conservative in its estimation of WiGig blockage, which stops MPTCP from fully benefiting from WiGig when mobile client switches to LoS. We also observe that MuSher gets a performance that is close to half of baseline MPTCP, while FBDT gets a throughput that is slightly less than pure summation of standalone WiFi and WiGig data rates. There are several reasons for MuSher's poor performance. First, our baseline MPTCP is the newest implementation of MPTCP in the Linux kernel and is more up to the date than the baseline compared against in MuSher [54]. Second, our mobility pattern is much more complex than the mobility pattern studied in MuSher, in which the client remains in LoS and moves towards, away

or left-right in front of the BS. Our mobility pattern involves frequent switching between LoS and nLoS as expected in real-world environments. Third, we have observed that MuSher is unable to timely estimate the bandwidth of different RATs, which makes the protocol sub-optimally split the traffic between RATs with much more emphasis on WiFi to a degree that it achieves even a lower performance than MPTCP.

We use the Traffic Controller (TC) setup that we discussed in Section 4.1 to perform the experiments. Fig. 4.1(b) shows the standalone RAT data rates as well as when they are simultaneously used by FBDT. We show the results under normal (static, no blockage) and when we introduce 5% loss to WiGig. We observe that FBDT is very close to the summation of standalone throughput values. We have conducted other experiments with delay/loss introduced to Ethernet or WiFi. In all, we have seen the same performance.

4.4.3 FBDT: Enhancing Application traffic QoS

FBDT: Evaluating Mobile Traffic Class QoE FBDT offers optimal performance across various classes of applications, regardless of their traffic characteristics, including short and bursty, large file downloads, and delay-sensitive data. For Interactive and Background classes of traffic, FBDT dynamically adjusts the transmission window size to ensure optimal performance. Despite the lower bandwidth and looser delay requirements of these classes, FBDT efficiently adapts its transmission window size for smaller and bursty traffic. This is done by adjusting the window size based on the data to be transmitted and utilizing both radios additively,

while minimizing transmission of redundant data. For larger data transfers, FBDT divides the data into fixed transmission sizes and optimally transfers data using both radios.

In the case of Interactive and Background classes, FBDT achieves an optimal transmission rate, as detailed in Section 3.2. For Streaming and Conversational classes, FBDT provides applications with the flexibility to prioritize packets for more reliable forward transmission or less reliable backward transmission. Furthermore, applications can duplicate packets within the FBDT transmission window to achieve the desired low-latency reliability in wireless networks.

On-demand streaming applications have the content available in advance at the sender (server) and transmit video streams to a client upon request. These applications are generally less delay-sensitive. FBDT continuously transmits streaming data in fixed chunks known as transmission windows. The rate at which the client receives the data streams depends on the throughput of the underlying wireless network.

Conversational traffic, which is highly delay-sensitive and involves limited data within a given time, benefits from FBDT's optimal performance based on the organization of data within the transmission window. More details about Conversational traffic are provided in [74].

Evaluating QoS for Interactive and Background class of Traffic under Mobility.

Setup. We assessed FBDT performance with Interactive and Background class traffic using web pages and file downloads. For file downloads, we developed

server-side scripts to transmit files of known sizes (ranging from 100KB to 1GB) to the client. On the client side, we automated a test script to download these files and recorded the time taken under various configurations. For web pages, we obtained objects from well-known sites and hosted a local web server, as detailed in Section 4.1. We selected four highly popular sites for evaluation based on their diverse number of page objects and sizes. Using automated client-side scripts, we repeatedly downloaded these web page objects with one-second intervals between downloads. These scripts logged the time taken to download objects from the webpages under different configurations: WiFi only, WiGig only, MPTCP, and FBDT.

Evaluating File-download QoS under Mobility. Fig. 4.6(a) shows file download time for WiFi only, WiGig only, MPTCP and FBDT different configuration. Though WiFi performs better than WiGig for smaller file sizes up to 100K but under performs for larger file sizes (>100K) by 2X as compared to WiGig or MPTCP. This disparity arises from WiGig waking up from sleep mode due to the large volume of data transfers. FBDT performs better than up to 30% higher than WiGig or MPTCP and up to 3X better than WiFi. FBDT Utilizes WiGig and WiFi additively for better performance while minimizing transmitting redundant data.

Traffic Split Ratio Under Mobility. Fig. 4.6(b) denotes the histogram of FBDT traffic split across WiFi and WiGig during file download. The x-axis shows the percentage of traffic over WiGig in brackets of 12 percent intervals. The y-axis shows the percentage of the event happening. We observe that 38% of the times about 72-84% of traffic passes through WiGig (the highest bar). We also observe

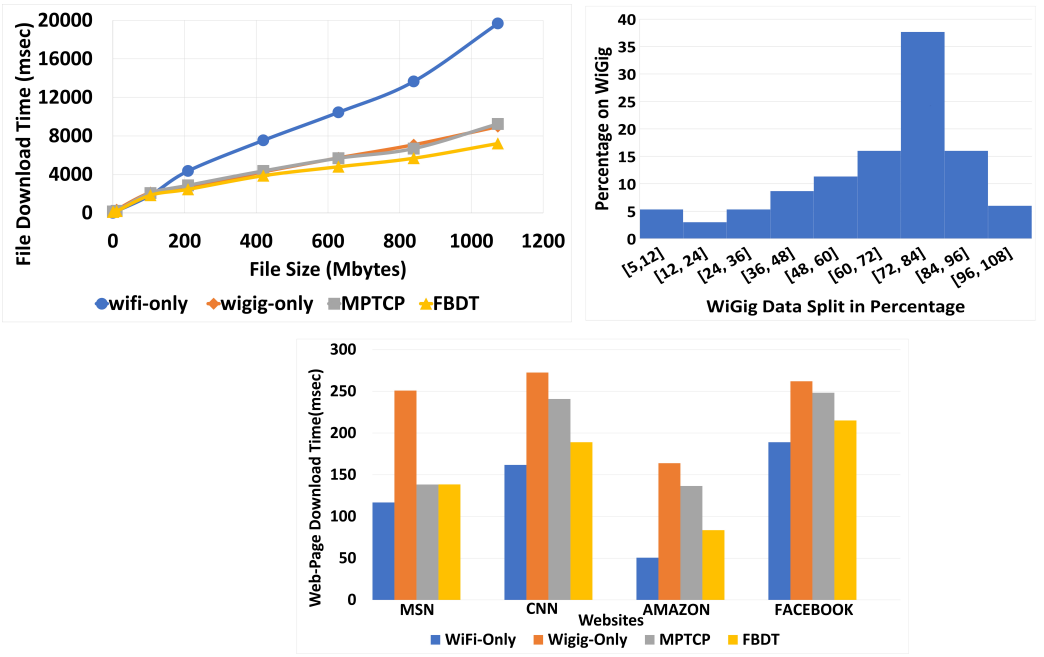


Figure 4.6: (a): File Download Time Comparison between FBDT, MPTCP and standalone RAT(s) (b):Histogram of traffic share passed through WiGig under mobility and when using FBDT. The results show that FBDT uses WiGig to pass majority of the traffic,. (c) Web-page Download time comparison between FBDT, MPTCP and standalone RAT(s)

that WiGig is used quite effectively by FBDT under mobility as majority of the traffic is passed through WiGig. This is because FBDT by design always optimally splits the traffic across RATs. It is therefore very quick at leveraging WiGig when channel condition turns LoS and WiGig rates drastically increase.

Evaluation of Web-page download QoE under Mobility. In Figure 4.6(a), the download times of web pages from various websites are presented under different mobility scenarios: WiFi only, WiGig only, MPTCP, and FBDT configurations. The page download time encompasses the duration required to download all the objects associated with the respective webpage. A typical webpage consists of multiple

object requests/downloads from the server, such as images, scripts, and fonts, each varying in number and size, as detailed in Table 4.1. Webpage downloads often exhibit intermittent bursts of file downloads. Despite WiGig's higher throughput compared to WiFi, WiFi outperforms WiGig in terms of webpage downloads.

To optimize power consumption, mobile clients configure wireless radios to transition from active to sleep or idle states. In our setup, the wakeup time for WiGig is longer than that for WiFi. The intermittent and bursty nature of webpage downloads prompts wireless radios to enter sleep or idle states, enhancing WiFi's webpage download performance compared to WiGig. FBDT demonstrates a 1.2 to 2X improvement over WiGig and MPTCP but lags 15% behind the WiFi performance. Although FBDT predominantly schedules data over WiFi, it also attempts to utilize WiGig when feasible. However, if the data scheduled over WiFi is successfully acknowledged within the transmission window while the scheduled WiGig data is pending acknowledgment, FBDT waits for the WiGig data transmission to avoid redundant data transmission over WiFi. This approach, while reducing redundancy, results in slightly lower performance compared to WiFi.

Website	**No of Objects**	**Total size**	**Average ± SD**
msn	33	5.8M	179K ± 389K
cnn	277	21M	75K ± 231K
amazon	325	8.9M	27K ± 121K
facebook	162	20M	122K ± 249K

Table 4.1: Website Object Statistics.

Evaluating Conversational and Streaming class of Traffic QoE.

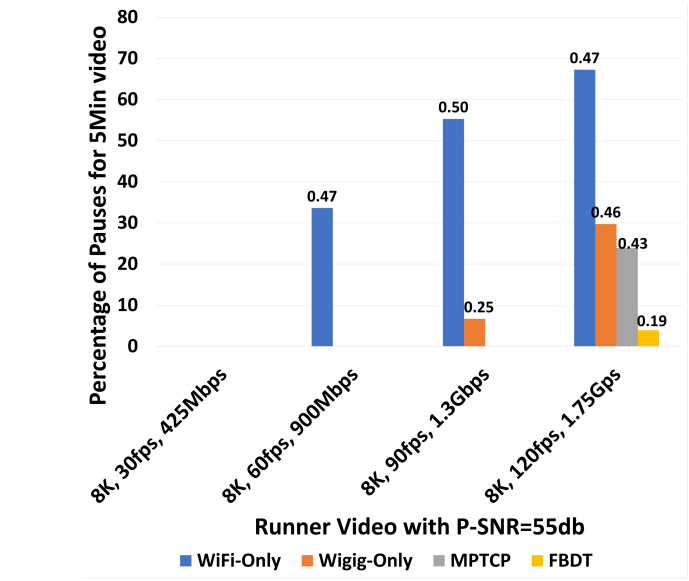
Setup. We examined Conversational and Streaming traffic categories through Progressive Downloads and Multi-RAT Live Video Streaming. For progressive video download assessment, we streamed a 5-minute segment of a compressed 8K UHD video with varying bit rate over time, from the server to the client, as outlined in Section 4.1. The client received these video streams via either a single RAT or dual RAT (WiFi and WiGig) and stored the data in a circular buffer. At one-second intervals, the client checked the cumulative bytes received and recorded pauses if the received data fell short of the required amount for that one second compressed interval (known as Group of Pictures, or GOP) of the video content. This process was automated for 110 trials. We repeated this experiment with various recorded UHD video streams from the server, using the configuration depicted in Fig. 4.1(a). The client received these video streams using WiFi, WiGig, MPTCP, and FBDT configurations.

For Conversational Multi-RAT video streaming, we consider videos that are encoded as Groups of Pictures (GoPs) representing one second worth of video data. Each compressed GOP comprises multiple video frames captured at a given temporal rate. A video set uses 30 frames per GoP universally. In Multi-RAT video streaming, each video is spatially broken up into small sectors or tiles that can be independently compressed across the duration of a GOP. In our study, the tiles are encoded at a given quantization parameter (QP) independently, so a frame can include tiles with many different QPs. The tiles at the same spatial location are jointly encoded across a GoP. We considered all the 15 videos from the publicly available 8K UHD video dataset [25] to choose quality points (QPs), i.e.,

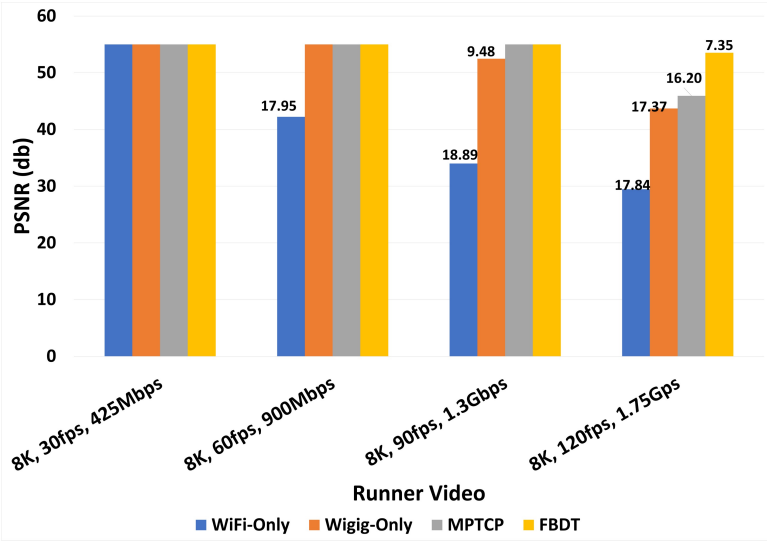
transmission rates, for each of the tiles of the video based on how likely they are to be observed. we consider FIX, an algorithm which performs the same transmission rate selection i,e the video tiles are given equal priority without any tile ordering and transmitted using FBDT and MPTCP transport layer.

Evaluation of Progressive Download under Mobility. In Figure 4.7(a), we present the pauses encountered by the client (Travelmate) across various video and connectivity configurations. For 8K video at 30fps and 425Mbps, no pauses occurred in any configuration. However, when the video quality increased to 8K at 60fps and 900Mbps, WiFi alone led to about 33% of pauses with varying dispersion levels measured using standard deviation. In contrast, WiGig-only, MPTCP, and FBDT configurations experienced no pauses. When the server streamed 8K video at 90fps and 1.3Gbps, WiFi-only and WiGig-only setups led to pauses for the client, with pause percentages at 55% and 7% respectively, each with different dispersion levels. Notably, FBDT and MPTCP showed no pauses. For 8K video at 120fps and 1.75Gbps, WiFi-only, WiGig-only, and MPTCP setups caused pauses for the client, with pause percentages at 67%, 30%, and 24% respectively, all with high dispersion. FBDT, however, resulted in only 4% of pauses with low dispersion.

Furthermore, in Figure 4.7(b), we illustrate the Quality of Experience (QoE) measured as the average PSNR for the client (Travelmate) under different video and connectivity configurations. For 8K video at 30fps and 425Mbps, there was no QoE degradation with average PSNR of 55dB. When the video quality increased to 8K at 60fps and 900Mbps, WiFi alone achieved an average P-SNR of 42dB with varying dispersion levels. In contrast, WiGig-only, MPTCP, and FBDT configurations



(a)



(b)

Figure 4.7: (a): Number of Pauses in percentage for Progressive video downloads. (b): Quality of User Experience (QoE) comparison measured using Average P-SNR

experienced no QoE degradation. When the server streamed 8K video at 90fps and 1.3Gbps, WiFi-only and WiGig-only setups led to QoE degradation for the client, with average P-SNR values of 34dB and 52dB respectively, each with different dispersion levels. Interestingly, FBDT and MPTCP configurations resulted in no QoE degradation. For 8K video at 120fps and 1.75Gbps, WiFi-only, WiGig-only, and MPTCP setups caused QoE degradation for the client, with average P-SNR values of 30dB, 44dB, and 46dB respectively, all with high dispersion. In contrast, FBDT resulted in an average P-SNR of 53.5dB with low dispersion.

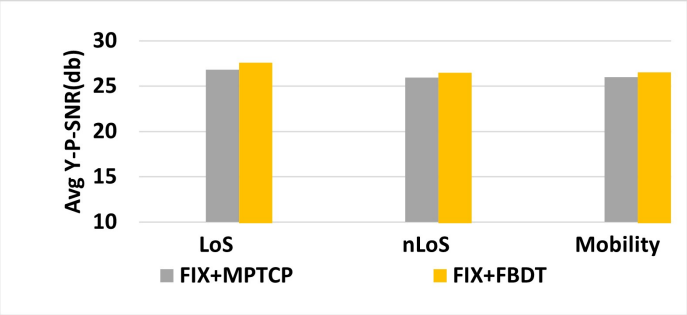


Figure 4.8: Viewport quality measured as PSNR across all video frames.

Evaluation of Conversational Mult-RAT video under Mobility. Fig. 4.8 depicts the mean PSNR gain across all videos frames for three channel conditions: LoS, nLoS, and mobility. The PSNR gain is calculated as the difference in PSNR across two schemes: FIX on top of FBDT and MPTCP. We observe that even in the LoS scenario where the throughput gap between MPTCP and FBDT is low, there is still more than 0.8 dB gain in PSNR. Whereas under nLoS and mobility, we observe a gain of 0.5dB gain in PSNR. Further, we show that by reordering video tiles and transmitting over FBDT can give PSNR gains of upto 13db under

mobility conditions [74].

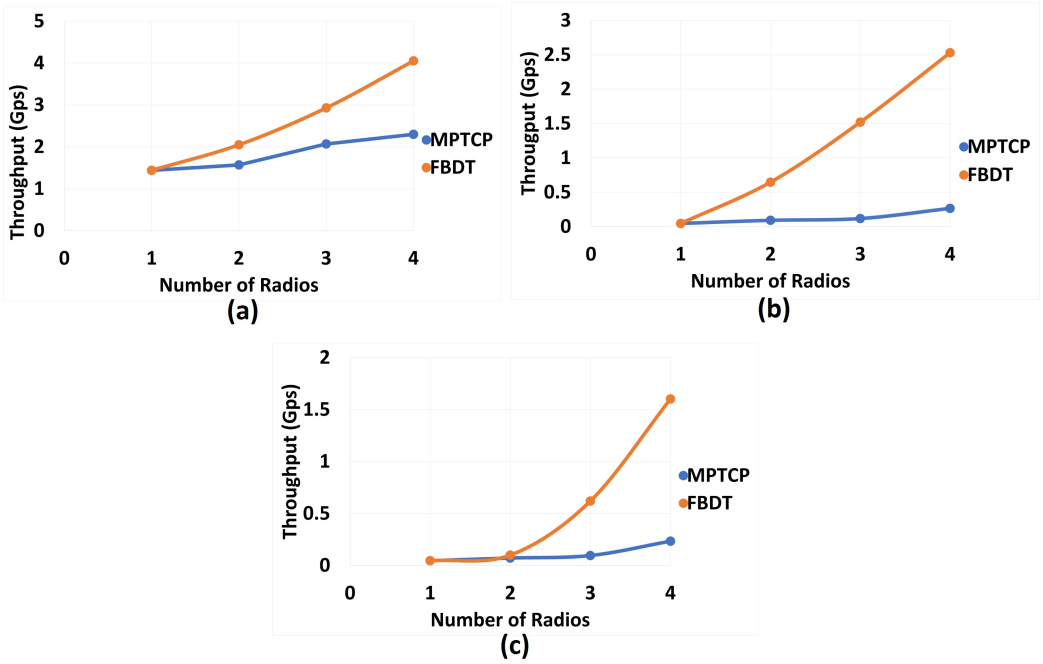


Figure 4.9: MultiRAT Throughput Comparision between FBDT and MPTCP (a): Without Loss (b): TC introduced 5% loss on single Radio (WiGig) (c): TC introduced 5% loss on two radios (WiGig and Ethernet)

Evaluation with Four RATs. Unlike many other protocols (eg., MuSher, which only supports dual RAT setups), FBDT supports any number of RATs. FBDT operation with N RATs are provided in Section 3.2. We now investigate FBDT's performance in a four RAT scenario. We choose two more Ethernet connections as our third and fourth radios instead of adding an additional WiFi or wireless RAT because we have the capability to fully control the wired medium (delay, loss) unlike the wireless channel. We use the Traffic Controller (TC) setup that we discussed in Section 4.1 to perform these experiments. Fig. 4.9(a) shows

the performance comparison of FBDT against MPTCP as the default scheduler. This evaluation was carried out with four RATs operating under normal conditions (static, no blockage) . The throughput of FBDT grows linearly with the number of RATs, and gives up to 2X better performance as compared to MPTCP. Further, we introduce 5% loss to WiGig, and observe that FBDT is very close to the summation of standalone throughput values as shown in Fig. 4.9(b). We have conducted other experiments by introducing 5% loss on one of the Ethernet. Fig. 4.9(c) shows the performance of FBDT growing linearly with the available BW of the underlying wireless network, while MPTCP collapses as we add more loss to the underlying wireless network. In all the three scenarios with four radio variations as shown in Fig. 4.9, we have seen the same performance, where MPTCP starts to collapse as we add radios and introduce losses. The MPTCP scheduler (minRTT/BLEST) is unable to schedule packets while eliminating HoL issues.

Chapter 5

Related Work

5.1 Fair Intial Access Design for mmWave Wireless

In IEEE 802.11 ad and ay standards [13], [14], IA (and beam search) is done in the beginning of every beacon interval. In particular, initially an AP sequentially sends sector sweep frames across its sectors, while all the clients record the signal strength of the received beams. In the next phase, each client randomly chooses a beam training slot and performs sector sweep in that slot. Several research works have proposed alternative methods that find better beams and/or reduce the beam search overhead. These works can be broadly divided into three classes: (i) *exhaustive sweeping* [15]–[17]: narrow spatial beams are used to scan all the directions exhaustively; (ii) *hierarchical sweeping* [18]–[20]: hierarchical codebooks are used to sweep all the directions; and (iii) *random sweeping* [21]–[25]: several random beamforming vectors are used to find the directions. A key missing piece in all these works (including the 802.11 ad/ay standard) is fairness in IA. In particular, IA provides an opportunity for all the clients to train their beams and most existing protocols rely on contention between clients as they sweep their

beams. However, mmWave systems suffer from the near-far problem, in which the beams of a client that is near to the AP would have a much higher power at the AP than the beams of a competing far-client. This power imbalance can create a significant IA unfairness among competing clients, which can delay or even deny far-clients from being admitted to the network. To the best of our knowledge, this is the first paper that addresses the IA fairness problem in mmWave networks. In particular, we identify *received power imbalance* and *poor contention protocols* as the key reasons behind poor IA fairness in multi-client mmWave networks. We then propose a Joint Power cOntrol and Contention adaptation protocol ("JPOC") to address the issue. Note that IA fairness is different from throughput fairness commonly studied in networking problems [26]–[28]. In particular, JPOC provides an opportunity for all clients to train their beams, so that the AP has a complete information of all the clients that need resources. The amount of resources given to each client (and hence the resulting client throughput) can be then determined by the AP (see Chapter 2), and is not addressed by JPOC. As a result, JPOC does not interfere with the desired throughput-fairness metric that the AP is aiming for.

Other works have proposed protocols to better address mmWave's mobility and blockages. Existing mmWave standards respond to these events by re-trigerring the sector sweep procedure and finding a new set of beams. Recent works have proposed several solutions to optimize the standard beam adaptation methods, e.g., (i) out-of-band beam search methods [29], [30] exploit the channel information from a co-existing low-frequency radio to speed up the beam adaptation; (ii) environmental sensing solutions [31], [32] sense the reflective environment and

leverage the sensed information to facilitate beam adaptation, and (iii) pro-active beam adaptation [33]–[36] uses model-driven methods to adjust the beams before blockages happen. These solutions alleviate the impact of mobility and blockages, and ensure a smoother data communication. However, they cannot necessary find the optimal Tx-Rx beams in presence of mobility or blockages. As a result, a client may continue to contend for beam training slots in continuously recurring beam search intervals. Thus, system dynamics (e.g., mobility, blockages) can exacerbate the IA contention unfairness problem as clients frequently compete to re-train their beams. JPOC's design is complementary to all the beam adaptation related work and enhances their fairness performance in presence of system dynamics by employing PC and contention time adaptation.

5.2 Forward and Backward Data Transmission (FBDT) Across Multi RAT

MPTCP Evaluation. Several papers [46]–[50] have studied MPTCP performance but they consider scenarios that consist of Internet paths or wireless setups with only sub-6 GHz RATs. Other works have studied MPTCP performance in networks that use mmWave RATs, e.g., [51], [52] studied dual WLANs with 802.11 ac+ad and show that MPTCP can get a lower performance than using WiGig only, [53] explores MPTCP in 5G+LTE through simulations, and MuSher [54] explores dual WiFi 802.11 ac+ad through implementation. FBDT is implemented in Linux kernel, supports any number of RATs, and significantly outperforms MuSher when client frequently switches between LoS and nLoS channels.

MPTCP Schedulers. In addition to the schedulers discussed in Chapter 2,

several schedulers have been proposed in the community, including schedulers that try to: (ii) address the challenges associated with heterogeneous paths [56], [58]–[60], (ii) leverage the differences in subflow RTTs [56], [58], [60]–[63], (iii) improve MPTCP performance for special use cases [64]–[66], and (iv) require modifications to the application [67]. FBDT can address multi-RAT scenarios with vastly different characteristics across RATs, has superior performance in static/mobile and LoS/nLoS scenarios, and does not require explicit information from the lower layers or modifications to the application.

Application class of traffic. Many studies [75], [76] have explored the performance of Interactive, Background, Conversational and streaming classes of traffic over WiFi and LTE. Other research work has proposed improving the performance of Interactive and Background classes of traffic classes over MPTCP schedulers (DAPS, ECF, OTIAS, LRF) as compared to SPTCP [77]–[79]. There is an extensive survey literature on the limitations of wireless streaming over MPTCP [80]. To overcome them, there have been proposed application oriented solutions for Streaming class of traffic (progressive downloads) over multipath TCP [81]–[84]. FBDT proposes a common transport layer solution to overcome limitations posed by MPTCP for application class of traffic.

Chapter 6

Conclusion

This thesis makes two key contributions to enable required QoS for application class of traffic by leveraging multiple RATs. First, we propose a joint power control and contention adaptation protocol (coined JPOC) that addresses unfairness problem. JPOC uses an open-loop and client-side power control mechanism that reduces the beam power imbalance among competing clients. Comprehensive evaluation through a mixture of experiments and simulations show that compared to existing 802.11 ad/ay standards, JPOC substantially reduces the contention overhead and increases the IA fairness.

Next, we introduced a new multi-RAT transport layer protocol named “FBDT" to address the underlying causes of MPTCP’s poor performance. We implemented our protocols on COTS hardware and conducted numerous experiments to evaluate the system performance in practice. We showed that FBDT provides a 2.5x gain against state-of-the-art MPTCP protocol when a mobile client routinely switches between LoS and nLoS conditions. We also showed that as compared to MPTCP, FBDT effectively aggregates traffic with more than two RATs and gets very close to

the summation of individual RAT data rates irrespective of the channel conditions. We showed that FBDT gains 3X, 10X and 9dBm gains for background, streaming and conversational traffic class respectively.

Bibliography

[1] C. Gustafson and F. Tufvesson, "Characterization of 60 GHz shadowing by human bodies and simple phantoms," in *Proceedings of European Conference on Antennas and Propagation*, 2012.

[2] M. Gapeyenko, A. Samuylov, M. Gerasimenko, *et al.*, "Spatially-consistent human body blockage modeling: a state generation procedure," in *IEEE Transactions on Mobile Computing*, 2020.

[3] 3GPP, "Universal mobile telecommunications system (umts); quality of service (qos) concept and architecture," in *3GPP TS 23.107 version 5.4.0 Release 5*, 2003.

[4] R. Siddavaatam, I. Woungang, G. H. Carvalho, and A. Anpalagan, "Mobile cloud storage over 5G: A mechanism design approach," *IEEE Systems Journal*, 2019.

[5] *Metaverse*, https://en.wikipedia.org/wiki/Metaverse.

[6] 3GPP, "Study on localized mobile metaverse services," in *5G Release 19*, 2022.

[7] J. Chakareski, M. Khan, and M. Yuksel, "Towards enabling next generation societal virtual reality applications for virtual human teleportation," *IEEE-SP-Mag*, Sep. 2022.

[8] *Gsma white paper on cloud AR/VR*, https://www.gsma.com/futurenetworks/wiki/cloud-ar-vr-whitepaper/.

[9] *Internet connection speed recommendations*, https://help.netflix.com/en/node/306.

[10] *Verizon 4G LTE speeds vs. your home network*, https://www.verizon.com/articles/4g-lte-speeds-vs-your-home-network/.

[11] T. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of ACM SIGCOMM*, 2014.

[12] A. Seam, A. Poll, R. Wright, J. Mueller, and F. Hoodbhoy, "Enabling mobile augmented and virtual reality with 5G networks," in *AT&T white paper*, 2017.

[13] T. Nitsche, C. Cordeiro, A. B. Flores, E. W. Knightly, E. Perahia, and J. C. Widmer, "IEEE 802.11ad: directional 60 GHz communication for multi-gigabit-per-second WiFi," in *IEEE Communications Magazine*, 2014.

[14] Y. Ghasempour, C. R. D. Silva, C. Cordeiro, and E. W. Knightly, "IEEE 802.11ay: next-generation 60 GHz communcation for 100 Gbps Wi-Fi," in *IEEE Communications Magazine*, 2017.

[15] J. Lee, G. T. Gil, and Y. H. Lee, "Exploiting spatial sparsity for estimating channels of hybrid MIMO systems in millimeter wave communications," in *Proceedings of IEEE GLOBECOM*, 2014.

[16] S. Payami, M. Shariat, M. Ghoraishi, and M. Dianati, "Effective RF codebook design and channel estimation for millimeter wave communication systems," in *Proceedings of IEEE ICC*, 2015.

[17] D. Zhu, J. Choi, and R. W. Heath, "Auxiliary beam pair enabled AoD and AoA estimation in closed-loop large-scale millimeter-wave MIMO system," in *IEEE Transactions on Wireless Communications*, 2017.

[18] J. Wang, Z. Lan, C. Pyo, *et al.*, "Beam codebook based beamforming protocol for multi-Gbps millimeter-wave WPAN systems," in *IEEE Journal on Selected Areas in Communications*, 2009.

[19] S. Hur, T. Kim, D. J. Love, J. V. Krogmeier, T. A. Thomas, and A. Ghosh, "Millimeter wave beamforming for wireless backhaul and access in small cell networks," in *IEEE Transactions on Communications*, 2013.

[20] A. Alkhateeb, O. E. Ayach, G. Leus, and R. W. Heath, "Channel estimation and hybrid precoding for millimeter wave cellular systems," in *IEEE Journal of Selected Topics in Signal Processing*, 2014.

[21] B. Gao, Z. Xiao, L. Su, Z. Chen, D. Jin, and L. Zeng, "Multi-device multi-path beamforming training for 60-GHz millimeter-wave communications," in *Proceedings of IEEE ICC*, 2015.

[22] A. Alkhateeb, G. Leusz, and R. W. Heath, "Compressed sensing based multi-user millimeter wave systems: How many measurements are needed?" In *Proceedings of IEEE ICASSP*, 2015.

[23] R. Mendez-Rial, C. Rusu, A. Alkhateeb, N. Gozalez-Prelcic, and R. W. Heath, "Hybrid MIMO architectures for millimeter wave communications: phase shifters or switches?" In *IEEE Access*, 2015.

[24] Z. Marzi, D. Ramasamy, and U. Madhow, "Compressive channel estimation and tracking for large arrays in mmwave picocells," in *IEEE Journal of Selected Topics in Signal Processing*, 2016.

[25] O. Abari, H. Hassanieh, M. Rodreguez, and D. Katabi, "Millimeter wave communications: From point-to-point links to agile network connections," in *Proceedings of ACM HOTNETS*, 2016.

[26] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," in *IEEE/ACM Transactions on Networking*, 2000.

[27] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multi-resource allocation: Fairness–efficiency tradeoffs in a unifying framework," in *IEEE/ACM Transactions on Networking*, 2013.

[28] E. Aryafar and A. Keshavarz-Haddad, "Distributed alpha-fair throughput aggregation in multi-RAT wireless networks," in *Proceedings of IEEE WiOpt*, 2020.

[29] S. Sur, I. Pefkianakis, X. Zhang, and K.-H. Kim, "WiFi assisted 60 GHz wireless networks," in *Proceedings of ACM MOBICOM*, 2017.

[30] T. Nitsche, A. B. Flores, E. W. Knightly, and J. Widmer, "Steering with eyes closed: Mm-wave beam steering without in-band measurement," in *Proceedings of IEEE INFOCOM*, 2015.

[31] T. Wei, A. Zhou, and X. Zhang, "Facilitating robust 60 GHz network deployment by sensing ambient reflectors," in *Proceedings of USENIX NSDI*, 2017.

[32] A. Zhou, X. Zhang, and H. Ma, "Beam-forecast: Facilitating mobile 60 GHz networks via model-driven beam steering," in *Proceedings of IEEE INFOCOM*, 2017.

[33] S. Sur, X. Zhang, P. Ramanathan, and R. Chandra, "Beamspy: Enabling robust 60 GHz links under blockage," in *Proceedings of USENIX NSDI*, 2016.

[34] A. Zhou, L. Wu, S. Xu, H. Ma, T. We, and X. Zhang, "Following the shadow: Agile 3-D beam-steering for 60 GHz wireless networks," in *Proceedings of IEEE INFOCOM*, 2018.

[35] M. Rakesh, Z. Marzi, Y. Zhu, U. Madhow, and H. Zheng, "Noncoherent mmwave path tracking," in *Proceedings of ACM HOTMOBILE*, 2017.

[36] D. Steinmetzer, D. Wegemer, M. Schulz, J. Widmer, and M. Hollik, "Compressive millimeter-wave sector selection in off-the-shelf IEEE 802.11ad devices," in *Proceedings of ACM CONEXT*, 2017.

[37] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch, "TCP extensions for multipath operation with multiple addresses," in *Internet Engineering Task Force (IETF)*, 2020.

[38] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath TCP development," in *Internet Engineering Task Force (IETF)*, 2011.

[39] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," in *Internet Engineering Task Force (IETF)*, 2011.

[40] M. Gapeyenko, A. Samuylov, M. Gerasimenko, *et al.*, "Spatially-consistent human body blockage modeling: A state generation procedure," in *IEEE Transactions on Mobile Computing*, 2020.

[41] C. G. Ruiz, A. Pascual-Iserte, and O. Munoz, "Analysis of blocking in mmwave cellular systems: Application to relay positioning," in *IEEE Transactions on Communications*, 2021.

[42] M. K. Haider and E. W. Knightly, "Mobility resilience and overhead constrained adaptation in directional 60 GHz WLANs: Protocol design and system implementation," in *Proceedings of ACM MOBIHOC*, 2016.

[43] T. Nitsche, A. B. Flores, E. W. Knightly, and J. Widmer, "Steering with eyes closed: Mm-wave beam steering without in-band measurement," in *Proceedings of IEEE INFOCOM*, 2015.

[44] S. Sur, X. Zhang, P. Ramanathan, and R. Chandra, "BeamSpy: Enabling robust 60 GHz links under blockage," in *Proceedings of USENIX NSDI*, 2016.

[45] A. Zhou, X. Zhang, and H. Ma, "Beam-forecast: Facilitating mobile 60 GHz networks via model-driven beam steering," in *Proceedings of IEEE INFOCOM*, 2017.

[46] Y. Chen, R. J. G. Y. Lim, E. M. Nahum, R. Khalili, and D. Towsley, "A measurement-based study of multipath TCP performance over wireless networks," in *Proceedings of ACM IMC*, 2013.

[47] Q. Coninck, M. Baerts, B. Hesmans, and O. Bonaventure, "A first analysis of multipath TCP on smartphones," in *Proceedings of Passive and Active Measurement Conference*, 2016.

[48] S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan, "WiFi, LTE, or both? measuring multi-homed wireless internet performance," in *Proceedings of ACM IMC*, 2016.

[49] A. Nikravesh, Y. Guo, F. Qian, Z. Mao, and S. Sen, "An in-depth understanding of multipath TCP on mobile devices: Measurement and system design," in *Proceedings of ACM MOBICOM*, 2016.

[50] Y. Lim, Y. Chen, E. M. Nahum, D. Towsley, and K. Lee, "Cross-layer path management in multi-path transport protocol for mobile devices," in *Proceedings of IEEE INFOCOM*, 2014.

[51] K. Nguyen, M. Kibria, K. Ishiz, and F. Kojima, "Feasibility study of providing backward compatibility with MPTCP to WiGig/IEEE 802.11ad," in *Proceedings of IEEE VTC*, 2017.

[52] S. Sur, I. Pefkianakis, X. Zhang, and K. Kim, "Wifi-assisted 60 GHz wireless networks," in *Proceedings of ACM MOBICOM*, 2017.

[53] M. Polese, R. Jana, and M. Zorzi, "TCP in 5G mmwavenetworks: Link level retransmissions and MP-TCP," in *Proceedings of IEEE Workshop on 5G New Radio Technologies*, 2017.

[54] S. K. Saha, S. Aggarwal, R. Pathak, D. Koutsonikolas, and J. Widmer, "Musher: An agile multipath-tcp scheduler for dualband 802.11 ad/ac wireless LANs," in *IEEE Transactions on Networking*, 2022.

[55] C. Raiciu, C. Paasch, S. Barre, *et al.*, "How hard can it be? designing and implementing a deployable multipath tcp," in *Proceedings of USENIX NSDI*, 2012.

[56] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli, "BLEST: blocking estimation-based MPTCP scheduler for heterogeneous networks," in *Proceedings of IEEE 2016 IFIP Networking Conference (IFIP Networking)*, 2016.

[57] H. Cech, "Analyzing and realizing multipath TCP schedulers in linux," in *Technical Report, Technical University of Munich*, 2020.

[58] N. Kuhn, E. Lochin, A. Mifdaoui, G. Sarwar, O. Mehani, and R. Boreli, "DAPS: intelligent delayaware packet scheduling for multipath transport," in *Proceedings of IEEE ICC*, 2014.

[59] T. Shreedhar, N. Mohan, S. K. Kaul, and J. Kangasharju, "QAware: a cross-layer approach to MPTCP scheduling," in *Proceedings of IFIP Networking*, 2018.

[60] Y. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: an MPTCP path scheduler to manage heterogeneous paths," in *Proceedings of ACM SIGMETRICS*, 2017.

[61] S. Baidya and R. Prakash, "Improving the performance of multipath TCP over heterogeneous paths using slow path adaptation," in *Proceedings of IEEE ICC*, 2014.

[62] J. Hwang and J. Yoo, "Packet scheduling for multipath TCP," in *Proceedings of IEEE ICUFN*, 2015.

[63] D. Ni, K. Xue, P. Hong, H. Zhang, and H. Lu, "OCPS: offset compensation based packet scheduling mechanism for multipath TCP," in *Proceedings of IEEE ICC*, 2015.

[64] X. Corbillon, R. Aparicio-Pardo, N. Kuhn, G. Texier, and G. Simon, "Cross-layer scheduler for video streaming over MPTCP," in *Proceedings of ACM MMSYS*, 2017.

[65] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, "MPDASH: adaptive video streaming over preference-aware multipath," in *Proceedings of ACM CONEXT*, 2016.

[66] A. Nikravesh, Y. Guo, X. Zhu, F. Qian, and Z. Mao, "MP-H2: a client-only multipath solution for HTTP/2," in *Proceedings of ACM MOBICOM*, 2019.

[67] Y. Guo, A. Nikravesh, Z. Mao, F. Qian, and S. Sen, "Accelerating multipath transport through balanced subflow completion," in *Proceedings of ACM MOBICOM*, 2017.

[68] "Nexmon: The c-based firmware patching framework," Available at: https://nexmon.org.

[69] A. Kochut, A. Vasan, A. U. Shankar, and A. Agrawala, "Sniffing out the correct physical layer capture model in 802.11b," in *Proceedings of IEEE ICNP*, 2004.

[70] "Uplink power control in lte," https://www.qualcomm.com/videos/uplink-power-control-lte.

[71] 3. T. 36.213, "E-UTRA physical layer procedure," 2020.

[72] 3. T. 38.901, "Study on channel model for frequencies from 0.5 to 100 GHz," 2019.

[73] J. Kurose and K. Ross, "Computer networking: A top-down approach," in *Pearson; 7th edition*, 2016.

[74] S. Srinivasan, S. Shippey, E. Aryafar, and J. Chakareski, "FBDT: Forward and backward data transmission across rats for high quality mobile 360-degree video VR streaming," in *Proceedings of the 14th Conference on ACM Multimedia Systems*, 2023.

[75] Q. De Coninck, M. Baerts, B. Hesmans, and O. Bonaventure, "Observing real smartphone applications over multipath tcp," *IEEE Communications Magazine*, 2016.

[76] V. Adarsh, P. Schmitt, and E. Belding, "Mptcp performance over heterogenous subpaths," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, 2019.

[77] Y.-s. Lim, Y.-C. Chen, E. M. Nahum, D. Towsley, and R. J. Gibbens, "Improving energy efficiency of mptcp for mobile devices," *arXiv preprint arXiv:1406.4463*, 2014.

[78] A. Nikravesh, Y. Guo, F. Qian, Z. M. Mao, and S. Sen, "An in-depth understanding of multipath tcp on mobile devices: Measurement and system design," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, 2016.

[79] P. Hurtig, K.-J. Grinnemo, A. Brunstrom, S. Ferlin, Ö. Alay, and N. Kuhn, "Low-latency scheduling in mptcp," *IEEE/ACM Transactions on Networking*, 2018.

[80] S. Afzal, V. Testoni, C. E. Rothenberg, P. Kolan, and I. Bouazizi, "A holistic survey of multipath wireless video streaming," *Journal of Network and Computer Applications*, 2023.

[81] J. Wu, C. Yuen, B. Cheng, Y. Yang, M. Wang, and J. Chen, "Bandwidth-efficient multipath transport protocol for quality-guaranteed real-time video over heterogeneous wireless networks," *IEEE Transactions on Communications*,

[82] T. Hiraoka and J. Funasaka, "A progressive download method using multiple tcp flows on multiple paths," in *2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, 2015.

[83] A. Elgabli and V. Aggarwal, "Smartstreamer: Preference-aware multipath video streaming over mptcp," *IEEE Transactions on Vehicular Technology*, 2019.

[84] J. Funasaka, "Evaluation on progressive download methods based on timer-driven requesting schemes on multiple paths with shared links," in *2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW)*, 2020.