Characterizing and Improving Next-Generation Network Infrastructures and Applications

by

Xumiao Zhang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2024

Doctoral Committee:

Professor Z. Morley Mao, Chair
Associate Professor Mosharaf Chowdhury
Associate Professor Hessam Mahdavifar
Associate Professor Feng Qian, University of Southern California

Xumiao Zhang

xumiao@umich.edu

ORCID iD: 0000-0002-3551-4074

To my family, my love, and my friends.

ACKNOWLEDGEMENTS

Six years ago, I arrived in Michigan with the goal of "mastering my field" in the next few years. Now, six years later, while I may not claim to have mastered it completely, I have experienced it deeply and learned immensely. This journey has been beautiful and enriching, largely because of the wonderful people I have encountered along the way.

Foremost among them is my advisor, Professor Zhuoqing Morley Mao. I am profoundly grateful for the opportunity to work with her. She has guided me through the door to the palace of knowledge and ideas. Whenever I struggled with challenges in my research, she patiently helped me navigate through the fog. As an expert in computer networks, she has taught me the essence of independent research, critical thinking, and project management. She always encouraged me to start low and aim high. Her constructive feedback has been invaluable in tackling one research problem after another. Without her, this dissertation would not have been possible.

Professor Feng Qian has also played a significant role in my research journey. We collaborated on all the projects in this dissertation. More than just a mentor, he has been a friend, offering me excellent guidance through my research projects and advice on surviving the demanding Ph.D. life.

I would also like to express my appreciation to Professor Mosharaf Chowdhury and

Professor Hessam Mahdavifar. Thank you for supporting me and serving on my dissertation committee. Your insightful comments have greatly enhanced the quality of my work.

I am grateful to my collaborators, Professor Zhi-Li Zhang, for the collaboration on many of the projects in this dissertation, and Fan Bai, who I worked with on vehicle research and offered me an internship opportunity at General Motors. I am fortunate to have completed two internships in the Network Research Group at Alibaba, where I received great support from Yunfei Ma, Pan Hu, and Xuan Zeng.

My lab mates have provided tremendous support throughout my journey. I have learned a lot from senior lab mates, including Yihua Guo, Yuru Shao, Shichang Xu, David Ke Hong, Yikai Lin, Xiao Zhu, Shengtuo Hu, and Yulong Cao. I also enjoyed the time spent with and help received from my dear colleagues, Jiachen Sun, Won Park, Jiwon Joung, Can Carlak, Qingzhao Zhang, Shuowei Jin, Ruiyang Zhu, Wenyuan Ma, Xueshen Liu, and Minkyoung Cho.

Thank you, Ann Arbor. I will always remember this lovely place, a peaceful haven where I have lived for six years, enjoying its picturesque four seasons: the blooming springs that bring a burst of color, the vibrant summers filled with outdoor activities, the breathtaking autumns with their tapestry of red, orange, and gold leaves, and the serene winters that blanket the city in a peaceful, snowy embrace.

I also want to thank my roommate Ye Yuan, who played computer games and had fun times with me, and Yibo Wang, who was always kind and willing to join us in dining and shopping. We shared many good memories and supported each other through the difficult COVID-19 times.

To Wenjia He, my love. Plain words cannot fully express the depth of my gratitude and

love for you. You celebrated my successes with unbridled joy and comforted me during my failures with unwavering compassion. Your presence has made this journey not only bearable but also beautiful. Now, we have reached the peak together! I am incredibly happy to have you by my side, and I look forward to a future filled with love, adventure, and countless more cherished memories together.

My sincere thanks to my parents, Hengbin Zhang and Dian Wang, for your endless love and support. Also, my apologies for not being able to stay close to you while pursuing my Ph.D. abroad. Big thanks for all other family members as well.

I truly enjoyed the past six years with everyone mentioned. I wish all of you a great future. Life will go on, as Albus Dumbledore said, "*It does not do to dwell on dreams and forget to live.*"

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

The rapid evolution of network technologies and the increasing demand for fast, flexible, and reliable connectivity have led to the emergence of next-generation network infrastructures, including new mobile networks such as 5G, new network protocols such as QUIC, and even new communication paradigms such as LEO satellite networking. These infrastructures possess the potential to revolutionize a wide range of applications such as connected and autonomous vehicles. However, there is a lack of comprehensive investigation into their unique characteristics for enhancing network applications, as well as the adaptation needed for existing applications to harness their full capabilities. To address this challenge, in this dissertation, we demonstrate that systematic measurements and analyses aimed at unveiling the intricacies of emerging network infrastructures, along with the development and innovation of efficient network applications, hold the key to unlocking the full potential of the next-generation network ecosystem.

For network application innovations, leveraging emerging vehicular connectivity and advanced sensor perception capabilities, we explore cooperative sensing for connected and autonomous vehicles. Specifically, we design an edge-assisted multi-vehicle collaboration framework based on Voronoi diagrams. As for network infrastructure measurements and improvements, we first characterize 5G network performance, power consumption, and

application QoE implications through large-scale real-world experiments. Then, we examine the QUIC transport protocol over high-speed Internet, reveal QUIC's performance issues after comparing it with the traditional TCP protocol stack, and conduct an in-depth root cause analysis. Lastly, to understand LEO satellite networks, we take Starlink as an example and compare it with existing cellular networks in various aspects. We also explore the potential of enabling multipath transport between LEO satellite and cellular networks. Collectively, this dissertation showcases the interconnected impacts of next-generation network infrastructures and applications, and advocates for an organic integration of empirical analysis with practical design.

CHAPTER I

Introduction

The Internet has been undergoing a remarkable transformation in recent years, fueled by the rapid advancement of network hardware [127, 74] and software [235, 121]. This revolution has given rise to next-generation network infrastructures and applications that promise to reshape the way we communicate, work, and live. The widespread adoption of new technologies has been nothing short of astonishing, with the number of connected devices projected to exceed 80 billion by 2025 [15], and global Internet bandwidth reaching a staggering 997 Tbps in 2022 [71].

Some of the most notable progress being made includes the following aspects, which together form the backbone of the next-generation network ecosystem. They strive to not only advance individual capabilities beyond their predecessors but also synergize with others to enhance overall network performance, reliability, and accessibility.

- **New network applications.** Connected and autonomous vehicles (CAVs) are transforming ground transportation systems by significantly improving road safety [19] and traffic efficiency [20]. Equipped with various 2D/3D sensors, CAVs continu-

ously sense the surrounding environment and make appropriate driving decisions. Numerous companies have begun developing CAV models and services [2, 4, 27], and significant investment has been poured into autonomous driving research [81].

- **New mobile networks.** 5G is ushering in a new era of mobile connectivity. With its massive MIMO [101], beamforming [208], network slicing [120], and more support from different layers, 5G aims to deliver significantly higher throughput, lower latency, and increased capacity compared to 3G and 4G. The advent of 5G opens up new possibilities for use cases requiring high-speed and reliable communication, such as mixed reality [230, 229] and remote surgery [157]. It is estimated that 5G will support 1.67 billion subscriptions worldwide by the end of 2023 [1].

- **New network protocols.** QUIC is expected to be a game-changer in improving web applications. It is a multiplexed transport-layer protocol over UDP, originally developed by Google (known as gQUIC [161]) and later adopted by the IETF (known as IETF QUIC [145]) as the transport layer basis of HTTP/3 [96]. Designed to enable reliable and secure connections, QUIC is intended to replace the traditional reliable TCP in web communication. QUIC is already responsible for over 75% of Meta's Internet traffic [32] and its adoption continues to grow fast [70].

- **New communication paradigms.** Beyond terrestrial networks, low-Earth-orbit (LEO) satellite networks, such as SpaceX's Starlink [6], are revolutionizing network connectivity by providing Internet access to remote and underserved areas. With more than 4800 satellites already in orbit and plans to deploy thousands more [5], LEO satellite networks have the potential to connect billions of people who currently

lack reliable Internet access, particularly while on the move.

However, as these next-generation network infrastructures and applications continue to evolve, they bring forth new challenges that need to be addressed to fully realize their potential. Emerging applications may not have achieved optimal performance levels. For example, CAVs suffer from limited perception range and cannot see through occlusions due to technological constraints. Similarly, the unique characteristics brought by new infrastructures are not yet fully understood, potentially causing unexpected performance degradation. For example, blindly running applications over 5G may lead to worse QoE compared to 4G, and QUIC may fail to surpass TCP if not properly utilized. More specifically, several concrete questions can be asked: (1) How can we deal with the limited visibility of CAVs' onboard sensors? (2) How do 5G's performance, power consumption, and QoE implications compare to those of 4G? (3) What is the performance of QUIC over high-speed networks? (4) Can LEO satellite networks offer stable performance and global coverage? Therefore, efficient application designs are essential for maximizing overall performance and harnessing the capabilities of the networks. At the same time, comprehensive investigations into the characteristics of new network infrastructures are indispensable for gaining insights into the adaptation of network applications.

This research is dedicated to addressing these challenges. The overall goal is to conduct systematic measurements to characterize emerging network infrastructures and build efficient network applications. With new generations of networks and network protocols, it is crucial to compare them with existing counterparts or predecessors and develop innovative solutions for better utilization and integration with existing components (*e.g.,* upper-layer applications). With new generations of systems with network capabilities, we

Table 1.1: Summary of dissertation work.

Problem scope	Project
Designing connected and autonomous vehicle cooperative sensing	EMP: Edge-assisted Multi-vehicle Perception
Characterizing power and application performance in 5G networks	A Variegated Look at 5G in the Wild: Performance, Power, and QoE Implications
Examining the QUIC transport protocol over high-speed Internet	QUIC is not Quick Enough over Fast Internet
Understanding network behavior in LEO satellite networking	LEO Satellite vs. Cellular Networks: Exploring the Potential for Synergistic Integration

should explore ways to improve performance with cross-entity communications in mind.

We demonstrate that: **Systematic measurements and analyses aimed at unveiling the intricacies of emerging network infrastructures, along with the development and innovation of efficient network applications, hold the key to unlocking the full potential of the next-generation network ecosystem.** As summarized in Table 1.1, this dissertation explores this problem along four use cases. We develop novel methodologies to address the unique challenges posed by each technology and explore their joint impact on advancing the state of network infrastructures and applications.

1.1 Designing connected and autonomous vehicle cooperative sensing

We identify the limited single-vehicle sensing problems, which cause poor perception performance for occluded and distant objects. To address this issue, we propose multi-vehicle *cooperative sensing* with assistance from nearby vehicles and computational edge nodes to enhance the situational awareness of individual vehicles and avoid road hazards. Faced with the challenge of bandwidth-overwhelming raw sensor data sharing over

wireless links, we develop a principled 3D data partitioning algorithm to divide overlapping areas in data from multiple vehicles. Furthermore, we build and evaluate EMP, an edge-assisted multi-vehicle perception system that enables scalable, adaptive, and efficient sensor data sharing.

1.2 Characterizing power and application performance in 5G networks

We carry out a comprehensive measurement study to reveal the effects of 5G deployment strategies, radio bands, and protocol-specific properties on power consumption and application QoE, compared with 4G. First, we characterize the power consumption of 5G networks, including understanding the built-in radio state machine, measuring power consumption for data transfer, and constructing a 5G power model considering various impacting factors. Then, we examine 5G's power and QoE implications for two mainstream applications, ABR video streaming and web browsing, to identify the new benefits and challenges introduced by 5G. Based on the insights, we propose solutions to improve application performance over 5G.

1.3 Examining the QUIC transport protocol over high-speed Internet

We investigate QUIC's performance over high-speed networks by comparing the UDP+QUIC+HTTP/3 stack with TCP+TLS+HTTP/2. We first examine the two stacks' bulk data transfer performance under QUIC toy applications, file downloaders, and web browsers, and find that QUIC suffers from a reduced data rate. We also compare the QoE

for video streaming and web browsing using both stacks and demonstrate that QUIC's slowness affects not only file transfer but also various web applications. We further determine the root causes of the observed performance gap. Through fine-grained packet trace analysis and profiling experiments in both kernel and user spaces, we find the culprit of performance issues to be excessive receiver-side processing and make recommendations for mitigating the issues.

1.4 Understanding network behavior in LEO satellite networking.

Using Starlink as an example, we look at an emerging communication paradigm, LEO satellite networks. We conduct a large-scale data collection campaign to understand the network performance of LEO satellite networks and their fundamental differences from terrestrial mobile networks. The data collection covers two Starlink services and three cellular carriers in the US, with a variety of network performance statistics recorded. We study the basic performance of the two network types, the impact of different factors on Starlink's performance, and their coverage in different areas. Lastly, we explore the potential of enabling multipath transport between them.

1.5 Thesis Organization

The rest of the dissertation is organized as follows. We first provide background knowledge in Chapter II. Chapter III presents an edge-assisted multi-vehicle collaboration system that enables scalable, adaptive, and efficient sensor data sharing for enhancing the local processing (*e.g.,* perception) of individual vehicles [259]. In Chapter IV, we

describe our comprehensive examination of 5G to understand the effects of 5G deployment strategies, radio bands, and protocol-specific properties on network performance, power usage, and application QoE, in comparison with 4G [185]. In Chapter V, we systematically examine QUIC over high-speed Internet. Specifically, we compare the UDP, QUIC, and HTTP/3 protocol stack with TCP, TLS, and HTTP/2 on different web applications. We also perform an in-depth root cause analysis [258]. Chapter VI describes our measurement study comparing the Starlink LEO satellite network with cellular networks and explores the potential for synergistic integration of the two network types (*e.g.,* using MPTCP) [136]. We discuss related work in Chapter VII and conclude the dissertation in Chapter VIII.

CHAPTER II

Background

In this chapter, we provide more details on the background of next-generation network infrastructures and applications, including connected and autonomous vehicles (§2.1), 5G networks (§2.2), the QUIC transport protocol (§2.3), and LEO satellite networks (§2.4).

2.1 Connected and Autonomous Vehicles

Connected and autonomous vehicles (CAVs) are vehicles equipped with various on-board sensors that allow them to perceive their surrounding environment, including other vehicles, pedestrians, and cyclists. LiDAR (Light Detection and Ranging) [55, 63] is one of the primary sensors used on CAVs. A LiDAR sensor functions by emitting uniform laser pulses at different angles and capturing their reflections from objects. It then calculates the distance to those objects and generates a 3D point cloud consisting of point coordinates relative to the sensor, to represent the surroundings. Compared with cameras, LiDARs offer advantages including a longer range and greater robustness under poor lighting conditions and inclement weather. CAVs are also equipped with wireless communication capabilities

and thus can exchange safety messages with other vehicles or infrastructures to enhance situational awareness. Different software modules, such as perception, prediction, and planning, are installed to further process the collected sensor data and make appropriate driving decisions for safe movement.

2.2 5G Networks

5G is bringing in a new era of mobile connectivity. 5G deployment is rapidly increasing since its roll-out in 2019. With more than 145,000 deployments in 142 countries worldwide [192], and predictions of over 4.6 billion 5G subscriptions by 2028 [3], 5G aims to deliver significantly higher throughput, lower latency, and increased capacity compared to its predecessors, 3G and 4G.

5G is expected to benefit a wide range of new applications in three major areas: enhanced mobile broadband (eMBB), ultra-reliable and low latency communications (URLLC), and massive machine type communications (mMTC). However, given the diverse factors such as frequency bands, deployment schemes, and mobility patterns, there are still many unknowns in terms of network performance, power consumption, and implications on application QoE.

2.3 QUIC Transport Protocol

QUIC is a user-space transport over UDP and comes with enforced encryption. It was initially proposed and developed by Google (**gQUIC**) [161] with the goal of enabling fast, reliable, and secure connections. Earlier, Google reported significant performance

gains compared with TCP [13, 30]. An IETF working group was launched in 2016 to improve the original gQUIC design which fuses the transport, cryptographic handshakes, and upper-layer HTTP. The group teased various functionalities into parts and later standardized the refined version into **IETF QUIC** [145]. As the application layer wrapper of QUIC, HTTP/3 was also adopted as an IETF standard recently [96]. Essentially, HTTP/3 was structured to make the HTTP syntax as well as existing HTTP/2 functionalities compatible with QUIC. Together with the network layer and layers below, UDP, QUIC, and HTTP/3 form a new protocol stack for next-generation network communication, whose current counterpart is the stack of TCP, TLS, and HTTP/2. While QUIC's design brings benefits such as 0/1-RTT fast handshake, stream multiplexing for the removal of head-of-line blocking, and connection migration, there are also potential downsides. For example, QUIC involves heavy processing and copying data between the kernel space and user space.

2.4 LEO Satellite Networks

Low-Earth-Orbit (LEO) satellite networks utilize distinct communication and connectivity technologies that set them apart from traditional cellular networks. Unlike cellular networks which rely on terrestrial base stations, an LEO satellite network operates through a constellation of satellites orbiting the Earth at an altitude of hundreds of miles. A user-side dish connects with a satellite, which in turn, communicates with a ground station. These ground stations relay data to and from the Internet. Several commercial LEO satellite networks have been announced, including SpaceX's StarLink, Amazon's Kuiper, and OneWeb. They aim to improve global connectivity and offer a competitive alternative to

traditional broadband services.

It is important to note that Starlink requires a Line-of-Sight between user dishes and satellites. Obstructions such as tall buildings or trees can disrupt the satellite connections. Consequently, Starlink has better performance in open and remote areas. In contrast, cellular networks excel in densely populated areas where a dense deployment of base stations ensures reliable connectivity. Moreover, due to their differing deployment strategies and thus service availability, the two types of networks can exhibit highly varied performance and coverage characteristics.

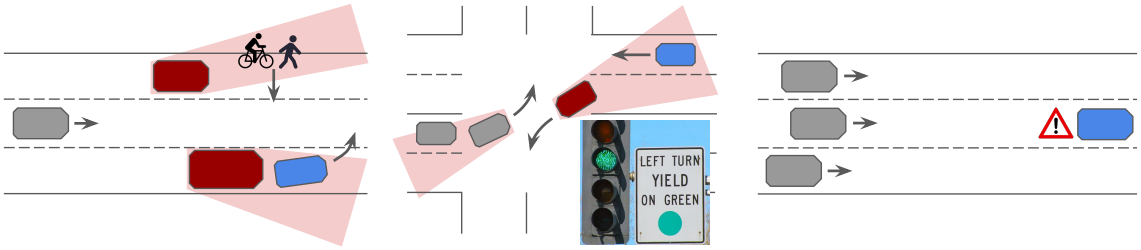
CHAPTER III

EMP: Edge-assisted Multi-vehicle Perception

Connected and Autonomous Vehicles (CAVs) are vehicles equipped with network connecivity and sensing capabilities. They heavily rely on 3D sensors such as LiDARs, radars, and stereo cameras. However, 3D sensors from a single vehicle suffer from two fundamental limitations: vulnerability to occlusion and loss of details on far-away objects. In this chapter, to overcome both limitations, we design, implement, and evaluate EMP, a novel edge-assisted multi-vehicle perception system for CAVs.

3.1 Introduction

Connected and autonomous vehicles (CAVs) are expected to transform the ground transportation systems by significantly improving road safety [19] and traffic efficiency [20]. 3D sensors such as LiDAR, radars, and stereo cameras are extremely important to CAVs as the sensors are their "eyes" that continuously sense the surrounding environment. However, these sensors suffer from two fundamental limitations. First, they are vulnerable to occlusion. Because of the rectilinear propagation of light, these sensors



(a) Blind Spots (b) Unprotected Left Turn (c) Broken Down Vehicle

Figure 3.1: Use cases of multi-CAV sensor data sharing.

cannot perceive objects occluded by non-transparent objects. Second, similar to human eyes, the farther an object is, the fewer details the sensors can capture. Take LiDAR as an example, it emits uniform laser pulses and constructs the environment based on the pulses reflected from objects. Therefore, the density of the pulses and henceforth the perception resolution decrease with increasing distance.

To overcome the above limitations, nearby CAVs can *share* their sensor data so that each vehicle can have a more complete view with a higher resolution compared to the view constructed from its own sensors. We consider three use cases of sensor sharing among nearby vehicles, as shown in Figure 3.1.

- Scenario 1 (blind spots from blocking vehicles): A pedestrian and a cyclist are crossing a street, and a blue vehicle is changing to the center lane. However, the gray vehicle cannot see them due to the occlusions of the two red vehicles. This can be resolved by sharing the sensor data from either the red vehicle with the gray vehicle.

- Scenario 2 (blind spots from turning): A gray vehicle is turning left at an intersection without a protected left-turn signal [39]. Meanwhile, a blue vehicle, which the gray vehicle cannot see, is traveling straight from the opposite direction, causing a

potential collision. This risk can be eliminated if the red vehicle shares its sensor data with the gray vehicle.

- Scenario 3 (distance-induced limited visibility): The blue vehicle breaks down in the middle of a road. Several oncoming gray vehicles cannot detect it from far away due to the low sensor data resolution. By aggregating their observed data, the broken down vehicle is more likely to be detected much sooner, preventing potential accidents.

Sharing Raw Sensor Data as Opposed to Processed Data. Several studies explored vehicles sharing *processed data* such as information of detected objects [170, 103]. We instead advocate sharing *raw sensor data* (when network resources permit) due to several limitations of processed data. First, its limited data granularity cannot support application-specific requirements. In other words, there will be a loss of information during data processing. For example, in Figure 3.1c, if none of the three gray vehicles can detect the blue vehicle, combining their processed data is ineffective, whereas sharing raw sensor data may lead to successful detection. Second, sharing processed data lacks generality, and may not be compatible with diverse CAV applications. In contrast, raw sensor data has a simple, fundamental, and universal data format to flexibly support a wide range of CAV applications. Traditionally, sharing raw sensor data was constrained by limited network resources, but this is being changed by high-speed wireless networks such as 5G [179, 242, 185].

Sharing Raw Sensor Data in a Scalable Manner. There are a limited number of works that do allow vehicles to share raw sensor data [202, 104, 190], but at a very limited scale. They all take a vehicle-to-vehicle (V2V) sharing approach which suffers from

poor scalability. As illustrated in Figure 3.1, oftentimes *multiple* vehicles need to get involved in sensor data sharing, particularly for congested roads. However, when many vehicles need to share their sensor data over V2V, each vehicle has either to perform multicast/broadcast, which suffers from low throughput in particular under mobility [245], or to unicast multiple copies of the data, incurring high delay and bandwidth overheads. Furthermore, a vehicle may not have enough computational resources to process other vehicles' data at line rate.

Differing from all existing works, we develop a system called EMP that *scales up* multi-vehicle sensor data sharing through edge computing [223, 137, 218]. We define *edge* to be computing and storage resources in close proximity to the vehicles, which provides low network latency to each vehicle. In our scheme, nearby vehicles upload their sensor data to the edge which creates a global view by merging individual vehicles' data. The edge can then run customized CAV algorithms and return the corresponding results (*e.g.,* detected vehicles as shown in Figure 3.2). With the edge support, each vehicle's workload and network bandwidth usage can be drastically reduced compared to the V2V scheme. Note that EMP does not fully replace a CAV's local processing, which is instead *enhanced* by EMP. For example, the local object detection results and the results from the edge can be combined to increase the detection coverage and accuracy. When there is a network blackout or the edge is unavailable, vehicles can always fall back to the local mode.

Principled Spatial Partition. The key technical merit of this work is to address the core algorithmic challenge for EMP: how do CAVs efficiently share their raw sensor data? Ideally, CAVs can cooperatively create a disjoint *spatial partition* of the environment, where (1) each CAV uploads only sensor data in its proximity, and (2) the union of all

CAVs' sensor data forms the entire surrounding environment. This strategy strikes a desired balance between bandwidth consumption and data quality: there is no overlap among CAVs' data so no bandwidth is wasted; meanwhile, as mentioned earlier, each CAV's close proximity has the highest sensor data quality. We find that mathematically, such a desired partition can be generated by a Voronoi Diagram [90] where the area that each vertex v (a CAV in our context) belongs to consists of the points whose distances to v are less than or equal to those to any other vertices. This key property nicely satisfies the above "proximity" requirement of EMP.

Adapting to Available Network Resources. While partitioning based on Voronoi diagrams is effective, it does not consider the available network bandwidth. For example, if the available bandwidth of a CAV is low, then it should upload less data. This can be realized by adjusting its uploaded area's boundary in the Voronoi diagram. We develop a robust algorithm that adaptively adjusts the sensor data uploading area of each vehicle (*i.e.,* the boundaries in the Voronoi diagram) in real time according to the estimated bandwidth of each CAV. In this way, vehicles with slow wireless connections can partially "offload" their uploading tasks to their neighboring vehicles.

Adapting to Network Resource Uncertainty. Wireless network conditions are known to be highly fluctuating, in particular under mobility. EMP embraces this through three mechanisms. First, it assigns priorities to each CAV's to-be-uploaded data, to ensure that important portions (*e.g.,* those that cannot be covered by other CAVs' data) are uploaded first, so they are the least vulnerable to the network resource uncertainty. Second, the above scheme naturally provides redundancy for regions that are perceivable from more than one CAV, thus boosting the resilience to the network condition fluctuations.

Third, in order to minimize the bandwidth waste incurred by the above redundancy, the edge employs a lightweight graph-based scheduling algorithm to efficiently detect if the entire environment is fully uploaded in real time.

Implementation and Evaluations. We incorporate the above algorithms into an edge-assisted multi-vehicle perception system developed by us. Our system consists of a full-fledged cooperative sensing pipeline including sensor data uploading, edge-side data merging, 3D object detection, and vehicle-side perception enhancement using the edge-side results. The above components are judiciously pipelined to ensure good runtime performance. We evaluate our system through extensive emulations using photorealistic sensor data and real-world LTE/60GHz network traces, and real-world live tests. Our key results consist of the following:

- EMP can achieve real-time processing at 24 FPS and end-to-end latency of 86 – 102 ms for the full partitioning-uploading-merging-detection pipeline, when 2 to 6 vehicles are involved in sensor sharing. EMP reduces the end-to-end latency by 49% – 65% compared to its V2V sharing counterpart.

- Compared to having all vehicles upload their full frames, EMP's approach to adaptively uploading sensor data in vehicles' proximity incurs a negligible perception accuracy loss (0.1% – 2.2%) when vehicle detection is performed on the merged view. Meanwhile, EMP's approach leads to a significant bandwidth usage reduction of 32% – 58%.

- We conduct case studies under realistic traffic scenarios and show that cooperative sensing powered by EMP can detect dangers (*e.g.,* occluded vehicles in blind spots

and far-away vehicles) earlier by 0.5 to 1.1 seconds, compared to a single vehicle's perception, leading to successful collision avoidance.

- To complement the real system results, we also conduct large-scale simulations involving up to 20 vehicles. The results further showcase the scalability and robustness of EMP under diverse road traffic and wireless network conditions.

Overall, EMP is to our knowledge the first system that enables edge-assisted multi-vehicle perception through raw sensor data sharing. We make two major contributions: (1) From the algorithmic perspective, we develop robust algorithms for scalable, adaptive, and resource-efficient sensor data sharing under potentially fluctuating network conditions. (2) From the system perspective, we incorporate our algorithms into a real system that can provide extended perceptual range and detection of occluded objects for CAVs.

3.2 Motivation

3.2.1 Benefit of Sensor Data Sharing

CAVs rely on various on-board sensors such as LiDARs for the perception of the environment to make driving decisions. However, there are limitations of these on-board sensors: (1) They suffer from occlusion. Because of the rectilinear propagation of light, they cannot "see through" non-transparent objects and can only provide line-of-sight information. (2) The farther, the fewer details they can capture. As LiDAR emits uniformly distributed lights and generates data based on the reflected lights, the data resolution decreases as the object distance from the sensor increases. Figure 3.2 illustrates these limitations with a LiDAR point cloud. There are several blind spots caused by the occlusion

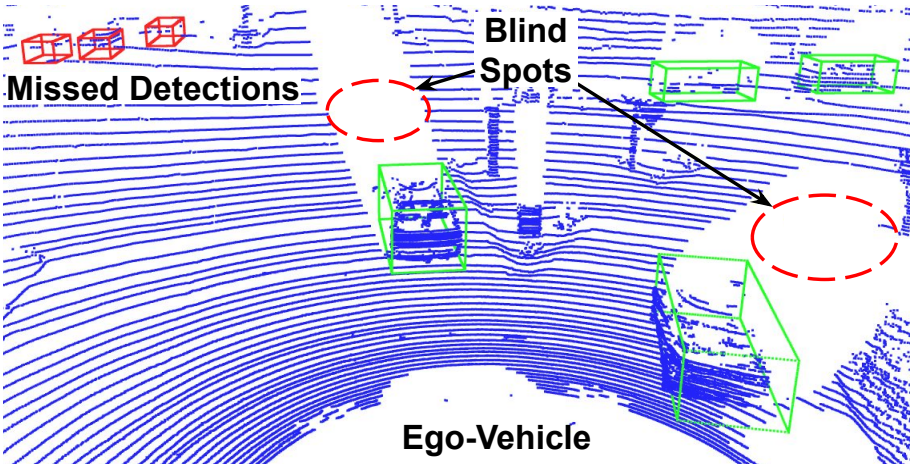


Figure 3.2: A LiDAR point cloud (blue) and detected object bounding boxes (green). The ego-vehicle fails to detect distant objects and loses information in the blind spots (red).

of the vehicles near the ego-vehicle. The ego-vehicle also fails to detect some distant vehicles. All these limitations bring road risks and affect driving efficiency for CAVs.

Different vehicles have views at different locations. It is possible that objects occluded in the views of some vehicles can be easily perceived by some others. Therefore, combining sensor data from vehicles perceiving objects at various perspectives can effectively eliminate occlusions and increase the perception resolution, thus further avoiding potential road hazards, as demonstrated in the examples in Figure 3.1.

As opposed to sharing processed data such as detected objects, we advocate sharing raw sensor data due to several limitations of sharing processed data. First, the information granularity decreases after processing the raw data to a higher layer of data, such as extracted features or detected objects. For example, in Figure 3.1c, a single gray vehicle may not detect the blue vehicle from far away on its own (*e.g.,* due to long distances or poor weather [38]) and merging their detection results will yield nothing, whereas combining the observations from all the vehicles altogether may lead to successful detection.

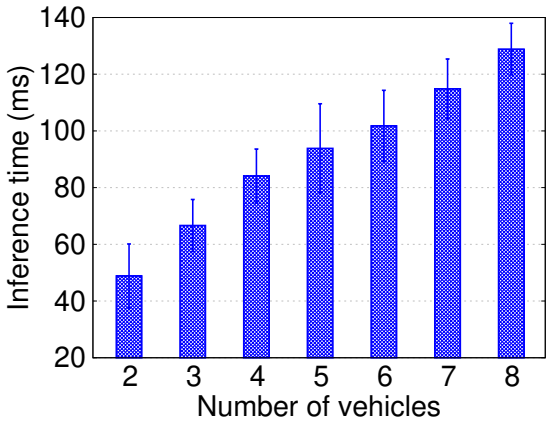


Figure 3.3: Inference time on data from varying numbers of vehicles.

Second, sharing processed data lacks generality. For example, vehicles may have different representations for processed data (*e.g.,* object classes). One detection algorithm may output car, human, *etc.* while another outputs sedan, bus, cyclist, pedestrian, *etc.* Instead, raw sensor data has a simpler data format to flexibly support a wide range of CAV applications. Furthermore, CAV's local processing takes time. A CAV can share its sensor data once the data is captured and some prepossessing is done. Then it starts the local processing while waiting for the results from the edge. Upon receiving the edge's enhanced results, it can adjust the driving decisions accordingly. In contrast, a CAV cannot share the processed data until it completely finishes the local processing.

3.2.2 Need for an Edge-assisted System

In order to avoid such hazards during daily driving, some existing works leverage sensor data sharing to improve the vehicle's visibility [190, 202, 104]. However, these systems only enable vehicle-to-vehicle (V2V) data sharing which suffers from poor scalability from both the network and computation perspective. **Network overhead**: There

are many scenarios (Figure 3.1) involving multiple vehicles for sensor data sharing. When the number of vehicles grows larger than two, each vehicle has to either send more than once or rely on another vehicle to relay its data, which introduces redundancy, additional delay, and higher bandwidth consumption. **Computational overhead**: Processing data shared from other vehicles involves additional overhead, challenging the limited on-board resources. We examine how the sensor data volume affects the inference time of 3D object detection using a state-of-the-art detection framework, PointPillars [160]. As shown in Figure 3.3, the inference time is roughly proportional to the number of vehicles which has a positive correlation with the data volume.

Edge computing services are becoming increasingly popular [26, 22, 34]. Edge nodes usually have more computational resources to process aggregated sensor data compared to on-board hardware which is equipped to process single-vehicle data. Communicating with an edge server also involves lower latency compared to using a remote cloud. Unlike V2V sharing, the vehicles only need to upload their data once to an edge node which can process them together.

3.2.3 Challenges

Building such an edge-assisted system that processes vehicle sensor data in real-time still poses several scalability challenges regarding network and computational resources.

- It is extremely hard for existing wireless techniques to support multiple vehicles simultaneously uploading raw sensor data in real-time. A commercial 64-beam LiDAR collects point clouds ($\sim$2MB[1]) at 5-20Hz [63], which means the data can be

[1]A point cloud contains $\sim$130K points (64 vertical angles and 2083 horizontal angles) consisting of location

generated at up to 300Mbps. How does the system reduce the data size with little impact on perception performance?

- Although an edge node usually has more computational power than individual CAVs, the processing time grows as the data volume increases as demonstrated in Figure 3.3. How can the system ensure real-time operation while providing an extended perception range?

- The available network bandwidths vary across vehicles, leading to different transmission times of data from different vehicles. Plus, wireless networks can fluctuate in particular under high mobility. How does the system adapt to the variability of network resources?

- Vehicles may upload frames at different times. There will be tremendous computational overhead if the edge processes a frame once it is received. In order to process data collected at similar times from different vehicles together, how does the edge determine when it can start processing the current frame and schedule for the next?

3.3 System Design

We propose EMP, an Edge-assisted Multi-vehicle Perception system for efficiently sharing sensor data over wireless networks and improving perception accuracy for CAVs. EMP tackles the above challenges through several design decisions: (1) EMP offloads heavy computation of cooperative perception from vehicles to an edge node (§3.3.1); (2) EMP efficiently partitions point cloud data to reduce network latency (§3.3.2); and (3)

and intensity information (xyz-i, 4 floating-point numbers).

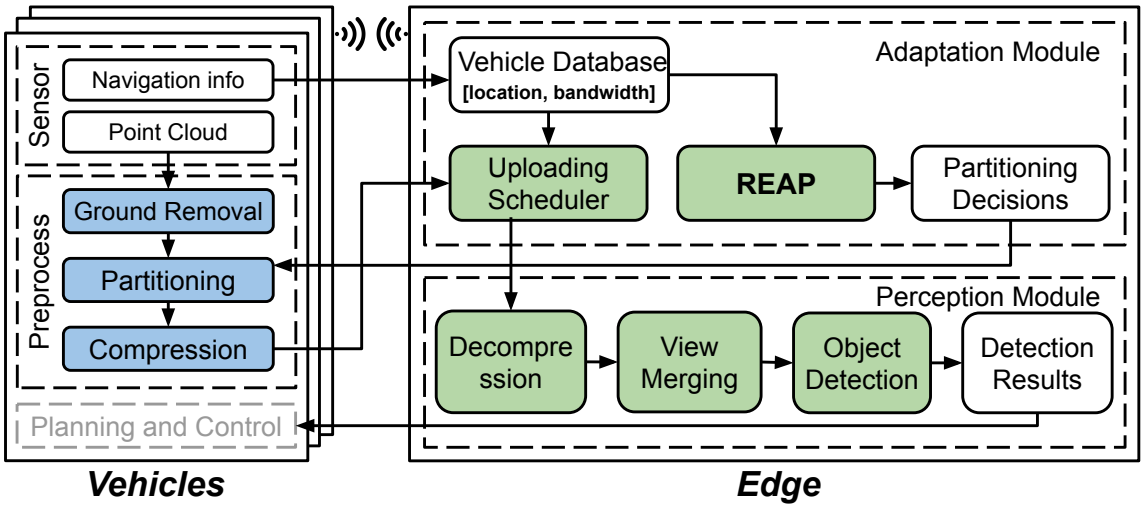


Figure 3.4: System Architecture of EMP.

EMP strategically coordinates the uploads from different vehicles (§3.3.3). EMP also incorporates view merging (§3.3.4), ground removal and system-level optimizations (§3.3.5) for boosting the performance of cooperative perception.

3.3.1 Edge-assisted Perception Architecture

At a high level, EMP offloads the cooperative perception from each vehicle to the edge side so that the edge performs object detection based on the aggregated sensor data and provides improved perception results for better driving decisions. To achieve this, EMP connects each vehicle with the same edge node with network channels in two layers, as shown in Figure 3.4.

Data Plane transmits the sensor data from the vehicles to the edge, and performs perception tasks at the edge. As shown in Figure 3.5, each vehicle preprocesses a single frame of sensor data (the **Preprocessing Module** in Figure 3.4), and uploads the chunks to

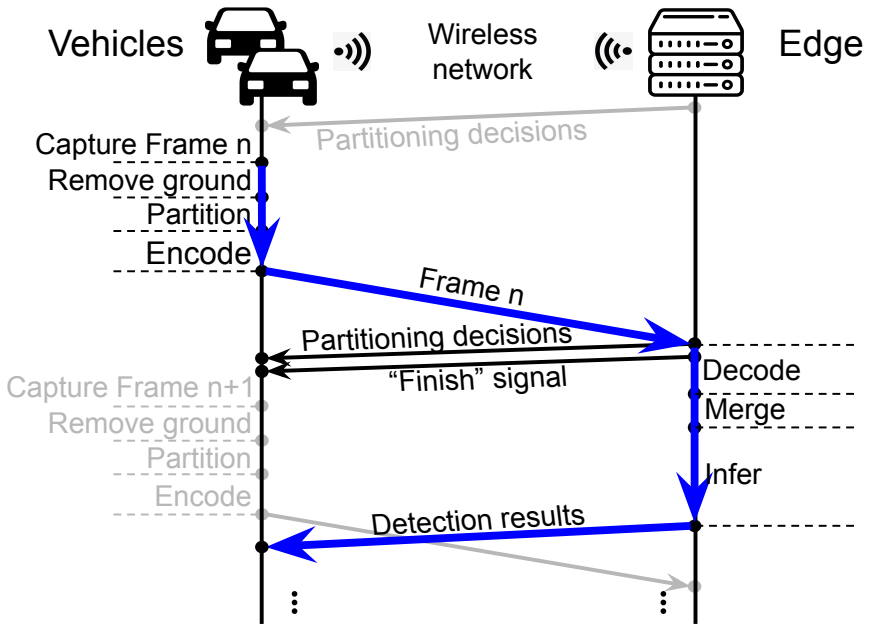


Figure 3.5: EMP Data Plane.

the edge. The partitioning is necessary for point cloud data as the size of a single frame can be large and there may not be enough bandwidth to upload full frames from all vehicles to the edge in time. Uploading chunk by chunk allows the edge to leverage partial point cloud data if available. Upon receiving the point cloud chunks, the edge merges these chunks with point cloud data from other vehicles based on the precise locations of all vehicles after decompression, forming a holistic point cloud as the view of the surrounding area. The edge can thus perform 3D object detection [160] on the holistic point cloud, and finally send the detection results back to each vehicle (the **Perception Module** in Figure 3.4). The results consist of locations, dimensions, headings, and confidence scores of the detected objects. EMP pipelines the vehicle's preprocessing and edge's perception, *i.e.,* a vehicle can start transmitting the next frame of the point cloud before receiving the detection results.

Control Plane optimizes the network transmission of all vehicles according to their locations and network conditions by guiding the point cloud partitioning for vehicles, so that the data to be uploaded by each vehicle is balanced and the edge can construct the holistic point cloud promptly. The partitioning allows each vehicle to upload its surroundings first, with the uploaded area adapted to its available network resources. Before uploading the sensor data, each vehicle sends a control message containing its real-time location to the edge. For each control message received, the edge uses the location, along with other vehicles' locations, to determine the region of the point cloud to upload for each vehicle. The decision region is sent back to the vehicles in the form of multiple line equations representing the region boundaries. Once a new frame of the point cloud is generated, the vehicle partitions the frame following the latest partitioning decision provided by the edge to reduce the data size. All the logic resides in the **Adaptation Module** in Figure 3.4.

Note that when the network connectivity is poor or the edge is unavailable, the vehicles can always run their local processing for basic services. Besides, while we focus on the assistance from a *single* edge in this work, the EMP's design can be flexibly extended to the scenarios of *multiple* edge nodes by introducing sensor data sharing among edge nodes based on vehicle locations and a handover mechanism, which we leave for future work.

3.3.2 Edge-assisted Point Cloud Partitioning

Since a full frame of the point cloud data may not be uploaded in time to the edge, the edge partitions the whole area into non-overlapping regions so that each vehicle only uploads a subset of the points according to the corresponding decision region, to reduce the amount of upload data.

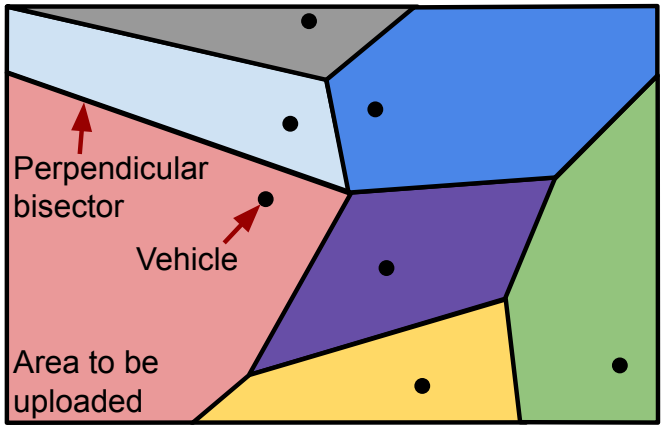


Figure 3.6: Voronoi Diagram.

One intuitive idea of such partitioning is to assign each point in the 2D region (bird-eye view) to the closest vehicle. In this case, the whole region is partitioned into multiple non-overlapping regions close to each vehicle. Since the LiDAR point cloud has more detailed information for the closer region, such partitioning ensures that each region has the finest representation from point clouds of multiple vehicles. Mathematically, such a partition is a Voronoi diagram [90], as shown in Figure 3.6. For each vertex v (a vehicle in our context), there is a corresponding area that consists of the points whose distances to v are less than or equal to those to any other vertices. Figure 3.7 visualizes an example of such a partition on a global region consisting of five vehicles. The letters $A - E$ represent the vehicles and $L_1 - L_7$ represent region boundaries, *i.e.,* the perpendicular bisectors in the Voronoi diagram, between two vehicles. For example, any points in B's region are closer to vehicle B than to any other, so we have $d_1 > d_2$ where d_1 and d_2 are the distances from the point to each vehicle. Note that the boundaries of a region for a vehicle only depend on neighboring vehicle locations, and such boundaries can be derived by finding the perpendicular bisectors between each pair of neighboring vehicles.

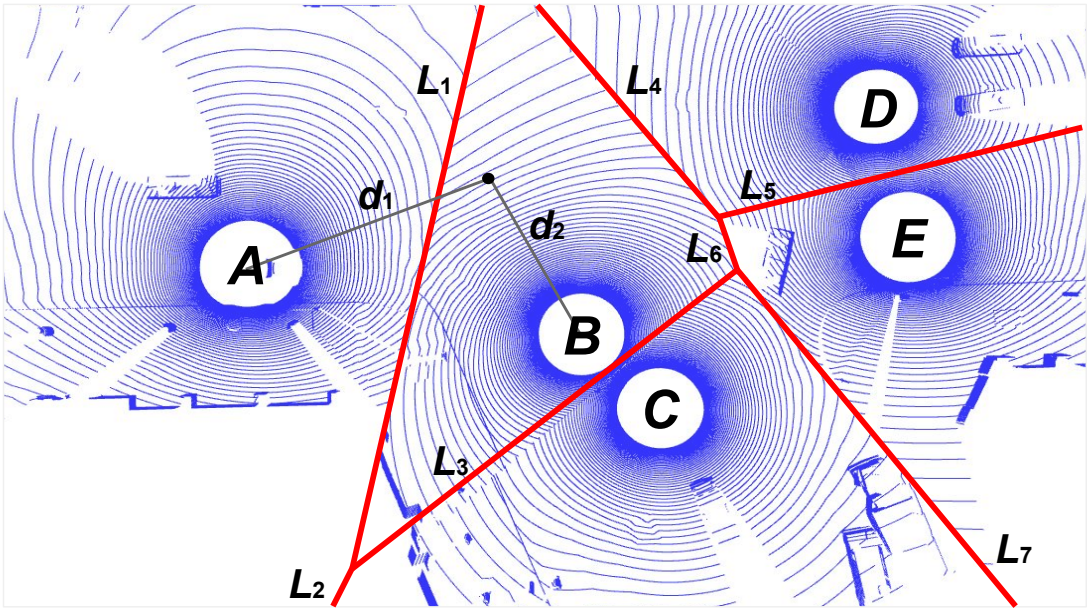


Figure 3.7: Naive partitioning through Voronoi Diagram.

While the partitioning based on Voronoi diagrams is simple, it suffers from two limitations. First, such partitioning depends only on the relative location of the vehicles, without considering vehicles' network conditions. This can still lead to a large network transmission time. For example, in Figure 3.7, when the network bandwidth of vehicle A is much lower than that of vehicle B, A can still take a longer time to upload its share of point cloud data than B to the edge, causing the edge to wait longer before leveraging A's point cloud data. Second, even if the initial partition of the region is proportional to the uplink bandwidth of each vehicle, as the vehicles move, the bandwidth can fluctuate significantly and thus lead to longer transmission times for some vehicles.

To address these limitations, we propose REAP, a Region-based Edge-Assisted Partitioning which is bandwidth-aware and adaptive to the bandwidth fluctuations. REAP decides the partitioning boundaries based on both the vehicle's location and estimated up-

link bandwidth (§3.3.2.1). REAP adapts to the fluctuating bandwidth by assigning multiple small chunks to each vehicle for transmission, taking the degree of bandwidth variation into account while partitioning, and dynamically determining when to finish transmission (§3.3.2.2).

3.3.2.1 Bandwidth-aware Partitioning

The available network resources of different vehicles can very likely vary. To cope with the different wireless uplink bandwidths across vehicles, REAP partitions the global region based on both the vehicle location and the estimated uplink bandwidth. At a high level, REAP achieves this through moving the region boundary between two vehicles towards the vehicle that has lower bandwidth. Such partitioning results in a smaller region to upload for vehicles with low bandwidth and a larger region for those with high bandwidth.

Specifically, REAP uses Power Diagram [89] in Mathematics to determine the precise partitioning boundaries. Recall that a Voronoi diagram draws the perpendicular bisector of the connection between every two neighboring vehicles as the partitioning boundary (Figure 3.7), which means the distance to the boundary from both vehicles are the same. A power diagram is a form of weighted Voronoi diagram, in which each vehicle is assigned a *weight*, and the ratio between the distances of two vehicles to the boundary is positively correlated to the ratio of the corresponding weights of the two vehicles. By adjusting the weights of the vehicles based on their corresponding estimated uplink bandwidth, we can thus make the partitioned region adapt to the vehicle's uplink bandwidth (the bandwidth usage is largely proportional to the uploaded area).

Figure 3.8 visualizes how the weights of vehicle A and B, r_1 and r_2 respectively,

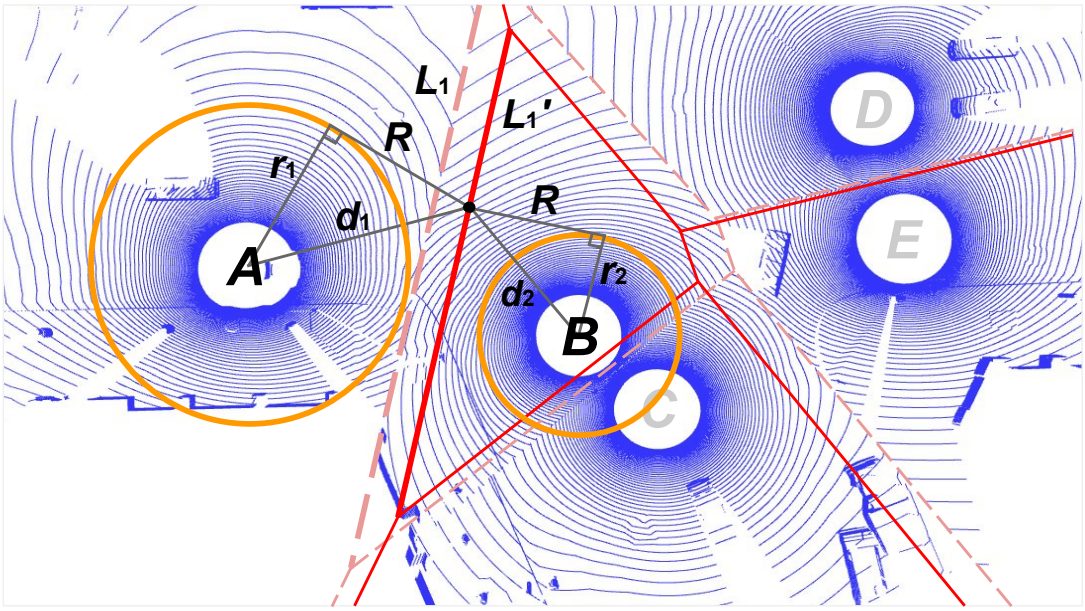


Figure 3.8: BW-aware partitioning through Power Diagram.

determine the boundary between the two vehicles. The boundary here is the radical axis of two circles centered on these two vehicles and the weights are the circle radii. Any point on the radical axis has the same power distance (R) to both circles. That is, $R^2 = d_1^2 - r_1^2 = d_2^2 - r_2^2$. As the estimated uplink bandwidth of A, and thus r_1, increases, the boundary L_1 is moved to L_1'. As a result, vehicle A with a better network condition is scheduled to upload more data. Note that the bandwidth bw (data volume divided by time) and the weight r (distance) in such a diagram intrinsically have different units. In REAP, we interpret the factor between these two values as a configurable parameter k which reflects the sensitivity of the system to the bandwidth differences, so we have $r = k \times bw$.

To estimate the uplink bandwidth of each vehicle, the edge measures the size and transmission time of each point cloud chunk sent from each vehicle, and computes its bandwidth as the exponentially weighted moving average (EWMA) of the ratio between

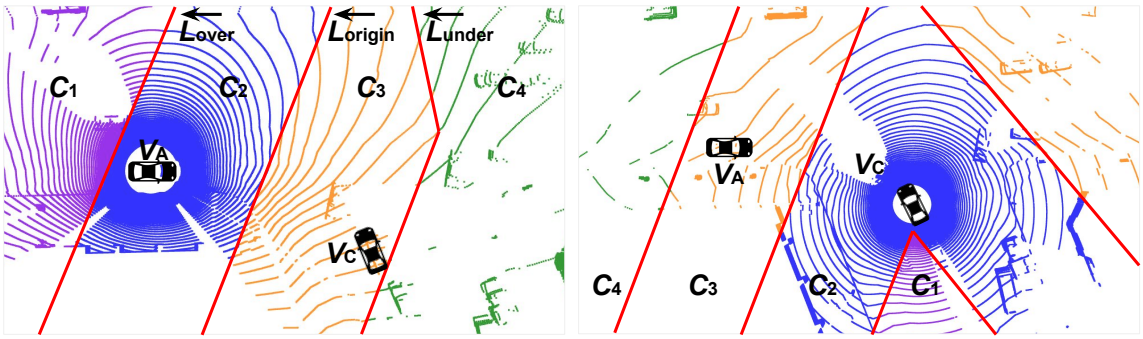


Figure 3.9: Vehicle A's (left) and C's (right) point clouds partitioned into chunks by REAP for adaptation to fluctuating bandwidth. L_{over}, L_{origin}, and L_{under} are region boundaries. C_1-C_4 represents chunk IDs.

size and transmission time.

3.3.2.2 Adaptation to Bandwidth Fluctuation

The bandwidth estimation from the edge can be inaccurate as the network condition is changing rapidly, especially for vehicles under high mobility. REAP addresses this challenge by further partitioning each vehicle's region into multiple chunks, based on two scenarios, bandwidth underestimation and overestimation. Each chunk is assigned an upload priority to ensure that important portions are uploaded first, so they are least vulnerable to the network resource uncertainty.

Specifically, for each pair of neighboring vehicles, besides the boundary (L_{origin}) calculated based on the estimated bandwidths (§3.3.2.1), REAP determines two additional boundaries, L_{over} and L_{under}, by replacing the original estimations bw_A and bw_B (bandwidths of vehicle A and vehicle B) with two new pairs of values: (1) $bw_A \times (1-\alpha)$ and $bw_B \times (1+\alpha)$, to account for the extreme case of overestimation of A's bandwidth and (2) $bw_A \times (1+\alpha)$ and $bw_B \times (1-\alpha)$, to account for the extreme case of underestima-

tion. These boundaries together partition a point cloud into smaller chunks, as illustrated in Figure 3.9[2]. Here α defines the degree of network fluctuation to tolerate and thus is correlated to the actual network characteristics. We adopt an auto-tuning strategy to set α during runtime. More specifically, the edge can adjust the α value using the standard deviation of estimated bandwidth values for different vehicles in the system across a past period of time.

As shown, each vehicle has chunks numbered from 1 to 4. Chunk 1 (C_1) is the area enclosed by L_{over} and is on the side far from neighboring vehicles. C_1 should be uploaded in the highest priority because 1) it is the easiest area to upload as it is derived assuming the bandwidth is overestimated, and 2) it has the least overlapping with other point clouds and other vehicles may not be able to help. Chunk 2 (C_2) is enclosed by L_{origin} and C_1 boundary. If all vehicles upload C_1 and C_2, the entire area is covered without overlapping, which is the best case. Chunk 3 (C_3) is enclosed by L_{under} and C_2 boundary. It further extends towards the neighboring vehicles and is closer to them. C_2 of one vehicle can be replaced by its neighbors' C_3 in reduced quality. Chunk 4 (C_4) is essentially the point cloud excluding the first three areas. It is the least important to the vehicle because this chunk is mostly blocked by its neighbors which can also capture more details. In case one vehicle is suffering from very bad network conditions or just gets disconnected, its neighbors can help by uploading their C_4. From Figure 3.9, we can find that C_2 of vehicle C is partially covered by C_3 of vehicle A (together with C_3s of A's other neighbors). C_1 of vehicle C is partially covered by C_4 of vehicle A.

Each vehicle sequentially uploads from C_1 to C_4. In this way, vehicles first share ar-

[2]The notations (A-E) used in previous figures are kept for consistency and we select vehicle A and vehicle C for better visualization.

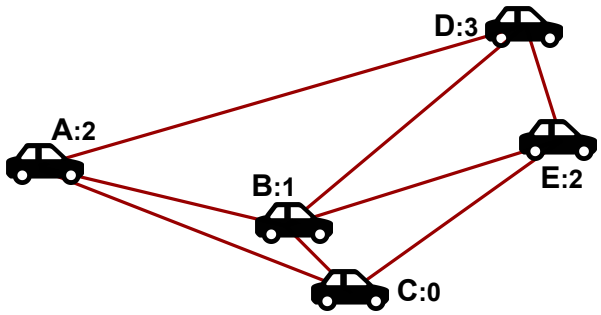


Figure 3.10: Neighboring relationships (*e.g.,* A-D) and chunk uploading progresses (vehicleId: chunkId).

eas that can be better captured and may only upload overlapped areas at the later time of the transmission. In different scenarios as the actual bandwidth differs from the estimated bandwidth, such a mechanism allows the edge to receive enough data to construct a holistic view as soon as possible, reducing the transmission time. In short, the goal of the adaptation in REAP is to achieve that a chunk is always finished by the "best" candidates who can provide the most details while other vehicles can help provide data with fewer details to meet real-time requirements, balancing the trade-off between the level of details in the point cloud and the time to start perception at the edge.

3.3.3 Upload Scheduling

According to REAP, each vehicle uploads its chunks sequentially. However, there will be unnecessary bandwidth waste if vehicles keep uploading the remaining chunks after the edge has received enough data to construct the global view. Besides, vehicles may start uploading their frames at different times but it is meaningless for the edge to process a frame without combining frames from other vehicles. Therefore, the edge needs to schedule when the transmission of a frame should be ended and it can start processing.

We propose a scheduling algorithm based on Delaunay Triangulation [91] for the edge to determine from the received data. For a given set (P) of discrete points, P's Delaunay Triangulation (DT) is a triangulation of P such that no point in P is inside the circumcircle of any triangle in DT. In other words, there will be no points inside the triangle formed by joining any three "neighboring" points. Figure 3.10 shows the vehicle locations and their neighboring relationships. For example, vehicle A, B, and D are neighbors to each other while vehicle E is not A's neighbor. Vehicle A has neighbor B, C, and D. In this way, when calculating a Voronoi diagram (§3.3.2, or Power diagram in §3.3.2.1), only A-B, A-C, and A-D are considered for vehicle A while A-E is not, which reduces the processing overhead compared to deriving perpendicular lines of connections between every two vehicles.

Based on how the chunks of each vehicle are divided, we define three conditions where the edge can determine the transmission of the current frame is finished: (1) C_1 and C_2 of all vehicles have arrived; (2) C_2 of one vehicle has not arrived (e.g., due to limited bandwidth) but the C_3 of all its neighboring vehicles have been delivered; (3) Neither C_1 nor C_2 of one vehicle has arrived but its neighbors finish uploading their C_3s and C_4s. Once any of these conditions is satisfied, the edge broadcasts a "finish" signal to stop all vehicles from uploading the remaining chunks.

As shown in Figure 3.10, the numbers after the vehicle letters (names) represent the largest IDs of chunks received by the edge. According to the conditions defined above, if all the numbers are 2, then the entire area is perfectly covered and the edge can notify the vehicles of the end of the transmission. However, although the REAP algorithm enables vehicles to upload chunk by chunk, from C_1 to C_4, their uploading progress will not be at

the same pace and the conditions are not satisfied by all the vehicles at once. Therefore, in order to check the satisfaction of these conditions, we develop a scheduling algorithm as follows: Every time a new chunk is received, the edge will check whether the sum of a vehicle's largest chunk ID and another vehicle's, is greater than 4. If so, that means these two vehicles locally satisfy the conditions and the ridge between them is removed from the set of ridges to be checked. When no ridges in the diagram are left, the entire area is fully covered and the frame is ready to be processed. Otherwise, the edge keeps waiting for the remaining data and rechecks when the next chunk arrives.

3.3.4 View Merging

After receiving frames from different vehicles, the edge performs 3D object detection on the holistic point cloud. However, generated from the perspective of a vehicle, the point cloud frame origin is the vehicle LiDAR sensor. Thus, in order to merge the data collected by different vehicles, the edge needs to transform the points of each point cloud from their original perspectives to a unified coordinate system. Given a target origin and axis orientations, the relative position (Δx, Δy, Δz) and orientation (α, β, γ) of a vehicle can be derived based on its navigation data (GPS/IMU) by calculating the differences. The edge further generates a translation matrix, $T = [\Delta x, \Delta y, \Delta z]^T$, and three rotation matrices, $R_z(\alpha)$, $R_y(\beta)$, $R_x(\gamma)$. Then, the transformation of a point $P = [X, Y, Z]$ can be calculated as follows: $P_{dst} = R_z R_y R_x \times P + T$. Note that this approach assumes the location data is reasonably accurate, thanks to high-performance localization techniques [50, 64] which can achieve centimeter-level accuracy. Existing point cloud calibration/registration techniques [112, 124, 173, 211] can be applied when the navigation signals are less accu-

rate.

3.3.5 Performance optimizations

We make several optimizations to save bandwidth, reduce end-to-end latency, and improve processing efficiency.

Ground Removal. LiDAR sensors collect a significant amount of data from the ground plane which is less useful than the data of surrounding objects for perception. Therefore, EMP detects and removes the ground points before sharing to save bandwidth. We use an algorithm called Random Sample Consensus (RANSAC) [119] assuming that the ground plane is the plane containing the most points in a point cloud. Specifically, EMP randomly picks several points to construct a plane and counts how many points in the point cloud fall near this plane. It repeats until the plane contains enough points. As the height of the sensor (atop the CAV) is known, we can estimate the approximate location of the ground to effectively reduce ground detection time.

Edge-side Parallelization. As the edge receives multiple point cloud chunks and locations from different vehicles, the processing of incoming data can be done concurrently. EMP takes decoding, merging, and location updating as individual tasks and parallelizes the tasks for different chunks by scheduling a corresponding task once a data chunk is received or decoded, or real-time navigation data is received. In this way, the edge saves a significant amount of time while waiting for new chunks.

Pipelining. To improve the system throughput, *i.e.,* the frame rate that EMP can support, we further pipeline the three parts (vehicle, network, edge) in the system, any of which does not have to wait until the current frame goes through the entire workflow be-

fore processing the next available frame. The vehicle is responsible for ground removal, point cloud partitioning, and decoding. After pushing frame n into the send buffer queue, a vehicle can process frame $n + 1$ once it is available. Meanwhile, the edge is working on a received frame such as frame $n - 1$.

Cloud Mode. Although edge nodes are being increasingly deployed [26, 22, 34], there could be areas where no edge nodes are available. In this case, EMP will fall back to rely on a cloud server for data aggregation and processing to provide seamless support. This may lead to a longer transmission latency but CAVs can still benefit from the cooperative perception. We evaluate the impact of EMP's cloud mode on detecting road hazards in §3.5.4.

3.4 Implementation

We implement EMP [48] in Java and the prototype consists of about 10K lines of code. **Vehicle-side**: Our prototype supports obtaining the incoming sensor data (*e.g.,* point clouds and navigation data) from various sources including both real LiDAR / GPS and recorded traces. The sensor data is provided to the processing pipeline at a configurable and fixed rate (*e.g.,* 10Hz in our experiments). The partitioning module takes as input the coefficients of line equations representing the chunk boundaries received from the edge and then crops out the chunks through linear algebra operations. We modify Draco [47] for LiDAR point cloud compression. The Draco APIs are invoked through JNI. **Edge-side**: The real-time 3D inference is built upon PointPillars [160], a state-of-the-art open-source 3D object detection framework. It is computationally efficient and is adopted by existing industry-level autonomous driving platforms such as Baidu Apollo [29] and Au-

toware [43]. In the original implementation of PointPillars, the code for the entire object detection pipeline is integrated. We thus separate different modules in the pipeline (model loading, model configuration, inference, *etc.*), and make model loading/configuration a one-time operation to enable fast inference. The edge also uses Draco to decompress the uploaded point clouds. For all other components in Figure 3.4, we implement them by ourselves. The vehicle and edge communicate through a custom protocol over TCP. Changing the underlying transport protocol to other protocols such as QUIC [161] is straightforward.

Note, EMP is designed for efficient point cloud data sharing. In this work, we focus on LiDAR point clouds for demonstration while the system is generally compatible with other CAV sensors which capture point cloud data, such as stereo cameras.

3.5 Evaluation

We evaluate the performance and scalability of EMP under different vehicle and network settings, demonstrating its advantage over vehicle-to-vehicle sharing schemes (§3.5.2). We examine EMP's enhancement on perception (§3.5.3). We showcase how EMP improves road safety with driving case studies (§3.5.4) and show the benefit of sharing raw data over sharing processed data. In addition, we present the latency contributed by key EMP system components and the processing throughput improvement brought by system-level optimizations (§3.5.5). A series of large-scale simulations are conducted to further prove the effectiveness of REAP under a wider range of vehicle numbers (§3.5.6).

3.5.1 Experimental Setup and Methodology

Due to a lack of open infrastructure support for multi-vehicle experiments, we adopt trace-driven emulation to evaluate EMP in our local testbed and compare our system with existing work. To emulate vehicle behavior in our experiments, instead of running a LiDAR device to generate data in real time, we replay LiDAR traces from a multi-vehicle LiDAR dataset we collected in advance. Our setup considers both diverse driving scenarios and realistic network conditions to comprehensively evaluate the performance and scalability of EMP. We also conduct live tests to demonstrate that EMP can work well under real networks.

Comparing EMP with Existing Work. We consider two variants of EMP to evaluate our design choices: (1) **EMP-Naïve**: EMP without REAP adaptive partitioning and scheduling, *i.e.,* vehicles upload full point cloud frames to the edge node; (2) **EMP**: EMP with all components enabled. We further compare them with vehicle-to-vehicle sharing schemes: (1) **V2V-Naïve**: each vehicle shares full frames with every other vehicle; (2) **V2V-Pro**: each vehicle shares partial point clouds with other vehicles using REAP partitioning. Note that the partitioning for V2V-Pro is only based on vehicles' relative locations, as bandwidth awareness cannot be achieved without an edge.

Multi-vehicle LiDAR Dataset. All existing LiDAR datasets such as KITTI [123] only contain traces collected by a single vehicle, while the evaluation of multi-vehicle perception requires the traces collected from multiple vehicles which are physically proximate at the same time. To fill this gap, we collect the *first* multi-vehicle LiDAR dataset using DeepGTAV-PreSIL [141], a tool to collect synthetic LiDAR traces simultaneously from multiple vehicles in a video game, GTA V. GTA V contains realistic 3D modeling of city

landscape, vehicles, stationary objects to emulate real-world scenarios. Besides LiDAR data, DeepGTAV-PreSIL also generates object labels for training machine learning models of perception. We extend the tool to enable panoramic ($360°$) LiDAR scans besides the default front-view-only settings. We construct our dataset by randomly driving a car in the game and collect sensor data from multiple nearby cars. Our multi-vehicle LiDAR dataset contains driving scenarios in both densely-populated urban areas and open rural areas, with various numbers of vehicles in the scene.

Network Conditions. The vehicle-to-infrastructure networking conditions [195] are emulated by throttling the bandwidth for individual TCP connections between each vehicle and the edge node and adding 10ms latency using Linux `tc` [8]. To acquire realistic uplink bandwidth of cellular networks for our experiments, we collect LTE uplink traces from driving at different urban and rural locations. We run 10-minute 100Mbps UDP uploads for a number of times over AT&T LTE networks on two smartphones (Pixel 2 and Nexus 6), to saturate the uplink. We run `tcpdump` at the server side to record raw packet traces and calculate the uplink throughput every 100 ms. To emulate high-bandwidth networking used by future vehicular communication [106], using a similar approach, we collect uplink bandwidth traces under 60GHz networks (802.11ad) with a stationary NETGEAR Nighthawk X10 AD7200 WiFi router [61] and a moving 802.11ad-compliant laptop. Note the bandwidth statistics of our LTE and 60GHz network traces are 14.0 ± 3.4Mbps and 267.0 ± 71.4Mbps, respectively. Thus the standard deviation is around 24% and 27% of the mean throughput which is close to the α value (~ 0.3, defined in §3.3.2.2) we observed during emulation.

Trace-driven Emulation and Real-world Test. We deploy the EMP-edge instance

on a server equipped with an Intel Xeon 4110 CPU clocked at 2.10GHz, an NVIDIA RTX 2080 GPU and 96GB of DDR4 RAM. The edge takes up to 390% CPU usage and 5GB memory when running with 6 vehicles. For the trace-driven emulation, We run multiple EMP-vehicle instances on another machine equipped with an Intel Xeon E5-2640 v2 CPU clocked at 2.00GHz to share the computation resources. Each vehicle uses up to 2 cores and 2GB memory, representing the often less computing power of a vehicle compared to an edge. For our real-world tests, we place a laptop (4 cores, 8GB memory) connected to LTE networks via tethering on a vehicle to run an EMP-vehicle instance.

Large-scale Simulation. We perform large-scale simulations to understand the performance of REAP algorithm under scenarios having a large number of vehicles. To mimic real-world driving scenarios, we randomly generate vehicle locations in a fixed area (120m$\times$30m) and ensure the horizontal/vertical distance between any two vehicles is greater than 3m/6m. We simulate the network transfer of each vehicle based on the size of to-be-uploaded data which is measured after applying REAP partitioning and Draco [47] compression algorithms to the point cloud data, with the replay of our real network uplink bandwidth traces.

3.5.2 End-to-end Performance

EMP is able to provide real-time enhanced perception under various processing workload and network conditions. We first compare the end-to-end performance and scalability of EMP with V2V. Note that the bandwidth between vehicles will be constrained when a vehicle simultaneously shares its data to multiple vehicles, which we also emulate using `tc`.

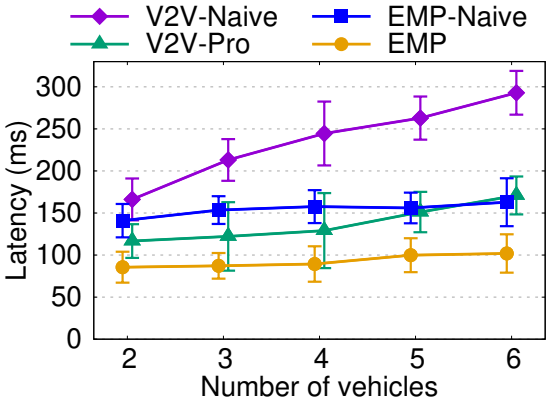


Figure 3.11: End-to-end latency of EMP and V2V systems.

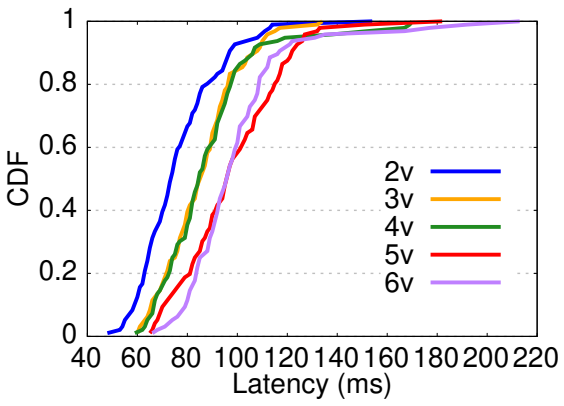


Figure 3.12: Latency distribution of EMP with varied numbers of vehicles.

Figure 3.11 shows the end-to-end latency of the four schemes when the number of vehicles in the system varies from 2 to 6[3]. End-to-end latency here is defined as the duration between when a vehicle starts preparing the collected sensor data and when the edge or a receiver vehicle finishes processing (*e.g.,* decoding, merging) data from all vehicles, which means the data is ready for perception. This is the additional latency introduced by EMP or the V2V counterpart and all the following steps such as perception need to be performed either on the edge or on a CAV. From the figure, we can find that EMP performs the best among all schemes and EMP-Naïve performs slightly worse because vehicles are dealing with full frames, which increases overhead for encoding/decoding and uploading. V2V-Pro benefits from the partitioning algorithm. However, as the number of vehicles increases, the latency of either vehicle-to-vehicle schemes (V2V-Naïve and V2V-Pro) skyrockets while the EMP latency stays around 100ms, saving 49–65%. EMP also outperforms EMP-Naïve by 36%–43%, which highlights the advantage of REAP

[3]Based on the average traffic and vehicle speed in city areas in the US [16, 33], the average vehicle density is 0.06 /m (6 vehicles in a 100m long road).

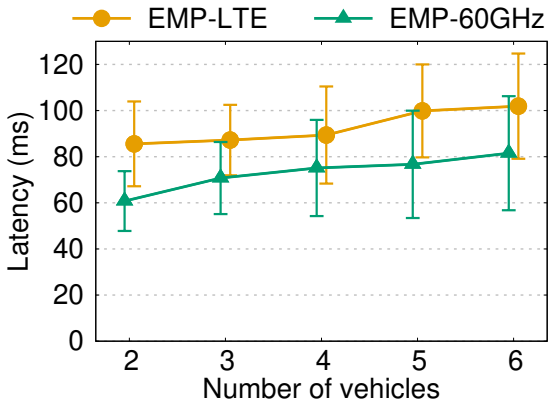


Figure 3.13: Latency of EMP under different wireless networks.

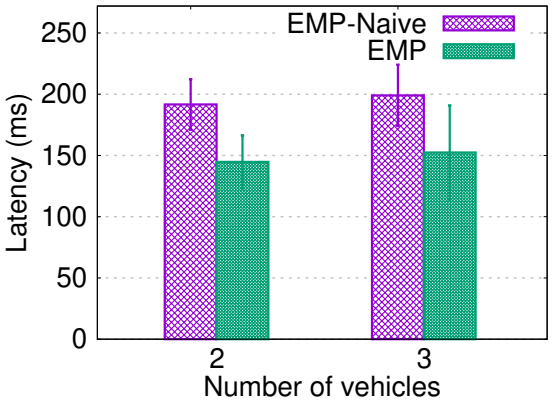


Figure 3.14: Latency of EMP-Naive and EMP in real-world driving tests.

Table 3.1: Size reduction brought by REAP partitioning and the size of shared data per frame (raw point cloud size: ∼2.0MB).

# of vehicles	2	3	4	5	6
REAP size reduction	32.4%	52.4%	58.0%	50.0%	50.3%
Shared data size (KB)	38.8	29.4	29.7	37.1	36.4

partitioning. Next, we show the distribution of EMP latency in Figure 3.12. The vast majority of frames experience less than 100ms latency, the recommended processing delay for autonomous driving [169]. Even in the worst case (*e.g.,* due to a network blackout) a vehicle does not receive the results for some frames, it can still rely on local processing for driving decision making. EMP is designed to enhance CAVs' local processing instead of completely replacing it.

To better understand how EMP saves bandwidth, we also calculate and show the size reduction of REAP partitioning in Table 3.1: the average size of partitioned chunks which are shared to the edge is 49.7% of the original point cloud size across all setups (2 to 6 vehicles). Further with ground removal and point cloud compression applied, each vehicle

only needs to upload 30–38KB for each frame. The data transmission can be finished within ∼23ms over LTE.

EMP is robust under various network conditions. The current CAV communication technologies are mainly DSRC and C-V2X which have limited bandwidth [153]. The emergence of 5G NR and other short-range mmWave networks can provide higher bandwidth and increase the vehicular communication capability [178]. However, there could be severer fluctuations under mobility, so we evaluate EMP under LTE and 60GHz (also used in [202]) networks and plot the results in Figure 3.13. EMP performs better under 60GHz networks as the high bandwidth helps reduce uploading times and the adaptation mechanism still maintains the system robustness under bandwidth variability. Note the bandwidth standard deviation of the LTE and 60GHz traces are 3Mbps and 71Mbps, respectively.

Lastly, we conduct real-world driving tests with EMP. Figure 3.14 shows the end-to-end latency of EMP and EMP-Naïve under 2/3-vehicle scenarios. As the vehicles only need to share the data once to the edge instead of multiple times to different receiver vehicles, the latency does not inflate when increasing the number of vehicles. REAP helps reduce the processing delay by reducing the uploaded data size so that EMP outperforms EMP-Naïve. We notice that the latency under real networks is higher than that measured in the emulation. This is likely because we are using commercial cellular networks (56ms of average RTT as measured) instead of directly communicating between vehicles and a real edge node (¡10ms).

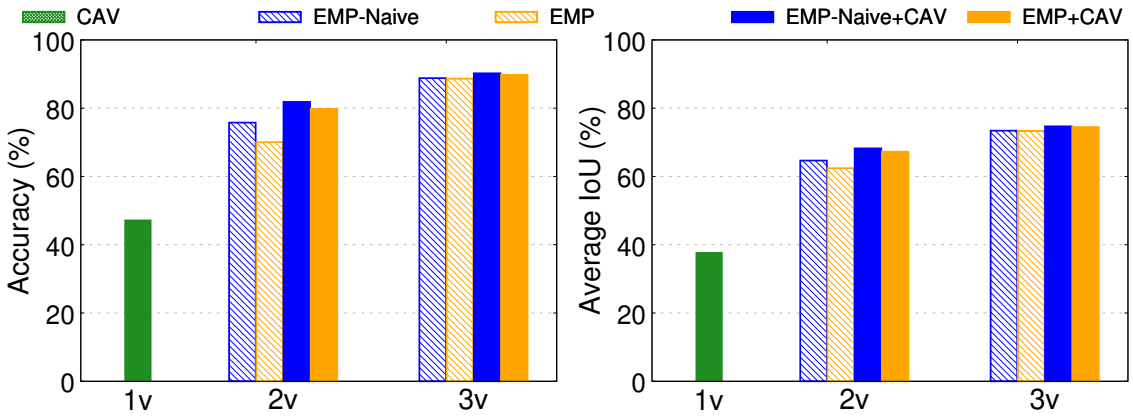


Figure 3.15: Detection Accuracy (left, IoU threshold = 0.5) and Average IoU (right) of single-CAV perception, multi-CAV perception, and combined perception.

3.5.3 Perception Enhancement

EMP can enhance CAVs' local perception while reducing bandwidth consumption to achieve real-time processing. We examine how EMP improves the perception of autonomous driving, by comparing the detection accuracy of single-CAV perception on one vehicle's point clouds (CAV), multi-CAV perception on views merged from 2/3 vehicles' data (EMP), and combined perception (EMP+CAV). The point clouds are merged in two ways: merge from full frames (EMP-Naïve) and merge from partitioned frames using REAP.

To measure the detection accuracy, we calculate the *Intersection over Union* (IoU) between the detection results (locations and dimensions of detected object bounding boxes) and the ground truth, which ranges from 0 to 1. A detection is *true* if its IoU with the label is higher than a threshold (0.5, as widely used in the computer vision community). Then we calculate the ratio of true detections out of all ground truth objects. We also directly calculate the *average IoU* for the detected objects, in order to evaluate in a more

fine-grained manner. When combining results from the edge and the CAV, if two detections match the same object, we record the one with a higher IoU. Besides, the locations of vehicles in the system are known (IoU = 1) as they are reported to the edge together with point cloud data. We focus on the $80m \times 50m$ area in front of vehicles (front view).

Figure 3.15 plots the detection accuracy and average IoU for each setup. We find that EMP perception outperforms CAV perception under both metrics. With EMP, the detection accuracy of combined results from edge's and CAV's is even higher. Comparing the performance of detection on views merged from full frames and those on REAP-generated frames, the accuracy is reduced by 0.1% – 2.2%, for 2-vehicle and 3-vehicle results. This indicates that REAP only introduces a negligible perception accuracy loss while bringing significant bandwidth saving as shown in §3.5.2. Therefore, we can conclude that EMP successfully enhances CAVs' local perceptions.

We also study the impact of ground removal on perception, by running object detection on the point clouds with ground points (original data). Compared with the detection accuracy shown in Figure 3.15 (left), the accuracy differs by -2.58% – 1.26% (not shown in the figure). Hence, the perception will not be affected by ground removal while sometimes it can even be slightly improved, possibly due to the reduction of noise from the ground. We run the same emulation on EMP without ground removal and the results show that EMP creates 27% – 33% less end-to-end latency.

It is worth noting that the dataset and the object detection model have limitations that may negatively affect the results. First, the vehicle sensors are not fully synchronized. As a result, the object locations in the frames of two vehicles can be slightly different and thus the detected object will have a location offset, lowering the IoU. The issue can be

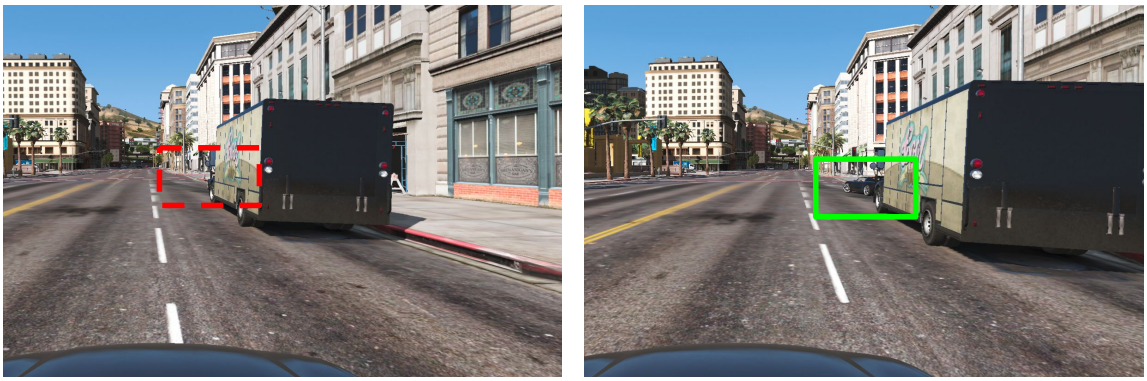
caused by the speed difference between vehicles and the movement of the object itself. To mitigate it, we use data collected by stationary vehicles while other objects can still move. Second, the detection model is trained with single-vehicle data[4] whose patterns are different from merged data, so the model may not perform perfectly on multi-vehicle data. Hence, the multi-vehicle perception is expected to perform better without these issues. Fully solving them can be non-trivial and we leave it for future work.

3.5.4 Case Study: Road Hazards Avoidance

Autonomous driving can benefit from EMP which provides vehicles with more knowledge of road traffic and more time to make decisions. EMP's design of sharing raw sensor data performs better than sharing processed data. To showcase such benefits, we conduct case studies and assess how EMP avoids potential road hazards in different scenarios. We customize vehicle locations in GTA V to construct the three scenarios mentioned earlier in Figure 3.1. Due to the limited performance of existing 3D object detection frameworks on pedestrians and cyclists [123], we simplify the scenarios to only involve vehicles and the benefits can still be shown. We measure in which frame the vehicles can detect the hazards with EMP at the earliest versus in a single-CAV setup.

Blind Spots. In this scenario (Figure 3.16 (a, b)), from the view of the ego-vehicle (the vehicle with a first-person view), a sedan is blocked by a big delivery truck behind it. The sedan is changing to the center lane. As illustrated in the camera images, without EMP, the ego-vehicle cannot detect the sedan until Frame X+8 due to occlusion. However, with EMP, the ego-vehicle can detect the sedan in Frame X, 0.8s earlier than the single-CAV

[4]To our knowledge, no existing work on point cloud-based object detection has investigated training using views merged from multiple point clouds.



(a) Scenario 1 - Frame X (b) Scenario 1 - Frame X+8

(c) Scenario 2 - Frame Y (d) Scenario 2 - Frame Y+12

Figure 3.16: Image data collected by the ego-vehicle in two scenarios where EMP detect road hazards earlier.

setup.

Unprotected Left Turn. As shown in Figure 3.16 (c, d), the ego-vehicle is trying to make a left turn, and has to judge on its own whether there are vehicles going straight from the opposite direction. An SUV is blocking the ego-vehicle view of a sedan behind the SUV. With EMP, the ego-vehicle can detect the sedan in Frame Y instead of Frame Y+12 at which point it may have already started turning. Figure 3.17 visualizes the point cloud of the ego-vehicle (CAV) for Frame Y, the point cloud merged from two full frames (EMP-Naïve), and the point cloud merged from partitioned frames (EMP). The sedan is

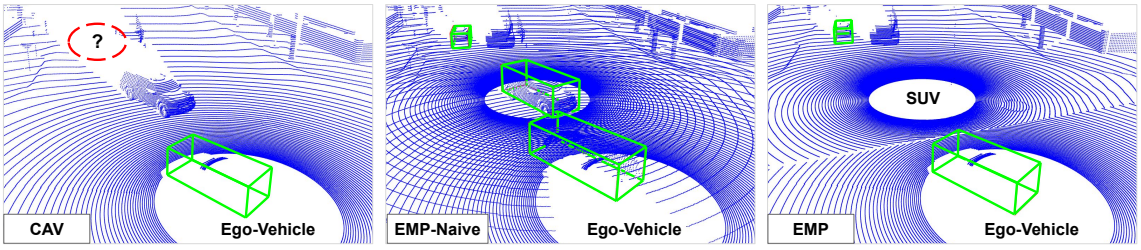


Figure 3.17: LiDAR point clouds in Scenario 2. The occluded sedan can be detected in both EMP setups.

successfully detected in the last two setups.

Table 3.2: Distance when detecting the target vehicle (m).

Scenarios	Initial distance	EMP (EMP-cloud)	CAV
Blind spots	28	26.3 (25.4)	19.8
Unprotected left turn	25	23.5 (22.5)	13.2

We also analyze how EMP can avoid the potential collision in both scenarios. As shown in Table 3.2, the initial distances between the target vehicle and the ego-vehicle are 28m and 25m, respectively. In city areas, vehicles drive on average at 9.2m/s [33] and the braking distance is around 20m correspondingly [12]. For a single CAV, there will be a 0.8s/1.2s delay from frame differences. Together with the earlier experimental data on processing latency, we derive the remaining distance when the ego-vehicle detects the target vehicle. We can learn from the results in Table 3.2 that, without EMP, the CAV only has a distance of less than 20m when it is aware of the occluded vehicle, while with EMP, the CAV system or the drivers have enough time to react. We also evaluate EMP cloud mode and calculate the remaining distance, and it is still above 20m.

Distant Broken Down Vehicle. In this case study (Figure 3.1c), we show that sharing processed data can fail to detect the distant vehicle earlier. Three vehicles are approaching

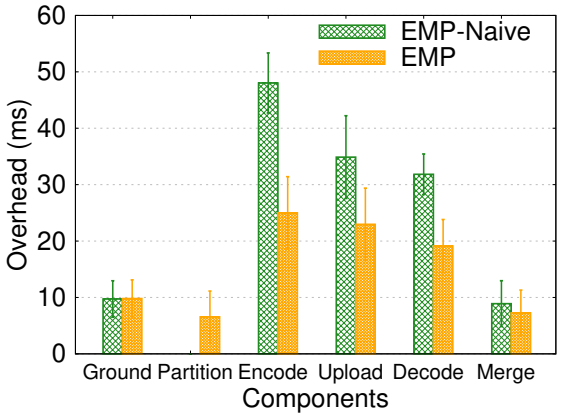


Figure 3.18: Overhead of each system component.

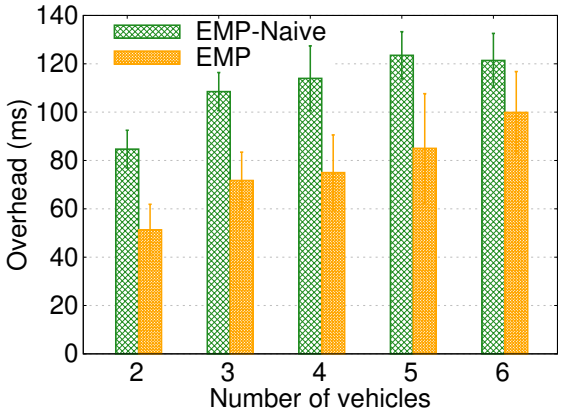


Figure 3.19: Inference overhead on merged frames.

a car broken down in the middle of the road. With EMP, the broken down vehicle is detected in Frame Z, while the earliest frames where each single vehicle (left, middle, and right) can detect the broken down vehicle are Z+10, Z+6, and Z+14, respectively. That means, by combining their processed data on the edge the detection success frame is Frame Z+6. Further taking into account the processing latency, sharing raw sensor data with EMP can detect the hazards 0.5s earlier than sharing processed data. Additionally, we repeat the test over EMP-Naïve in which the vehicles share full frames and the broken down vehicle is detected 7 frames earlier. Without partitioning, all data were uploaded to the edge so the details of the object build up even faster as the three vehicles approach the distant one, showing a trade-off between transmission overhead perception performance.

3.5.5 Overhead Breakdown

We break down the latency of each system component and compare them in EMP and EMP-Naïve (no REAP partitioning and scheduling) to highlight the benefit of our design

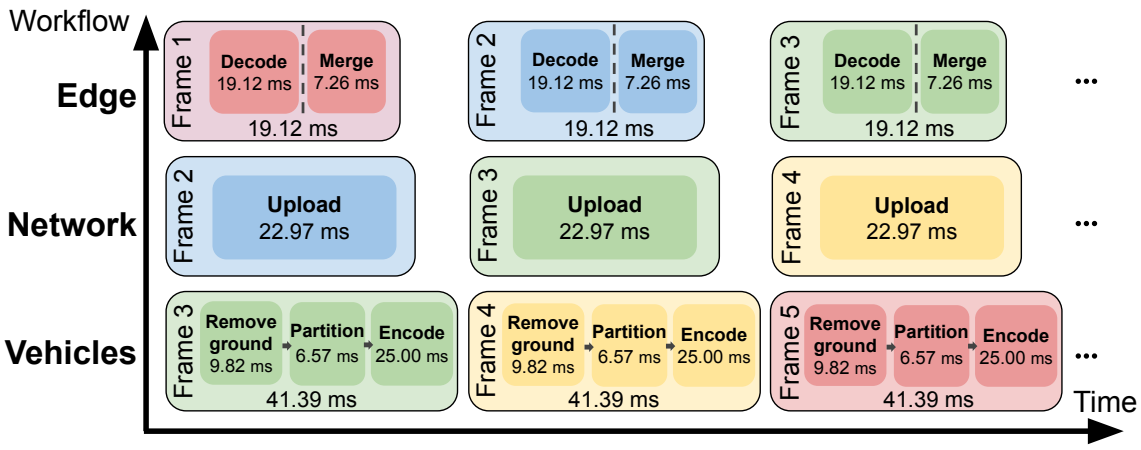


Figure 3.20: Edge-side parallelization and pipelining enable EMP to process at 24 FPS.

decisions. We then show the throughput (FPS) improvement brought by EMP edge-side parallelization and pipelining.

Figure 3.18 presents the overhead of each component. Thanks to the REAP partitioning and scheduling, the uploading can be finished before all the vehicles share their full frames, as soon as the edge receives enough chunks to build the global view. Thus, the encoding, uploading, and decoding times of EMP are significantly reduced, compared to those of EMP-Naïve. There is also a small saving on the merging time. The saving on these components is much more than the additional latency introduced by REAP partitioning (6.57 ms). Besides, we also measure the overhead of inference which is the time to run 3D object detection on merged frames generated from both system schemes (Figure 3.19). Corresponding to the preliminary results discussed in §3.2.2, inference overhead increases as the number of vehicles increases. This is because more vehicles lead to a larger amount of points in the merged global view. EMP saves 21-33 ms for this step.

As mentioned in §3.3.5, we optimize the system workflow to increase throughput of

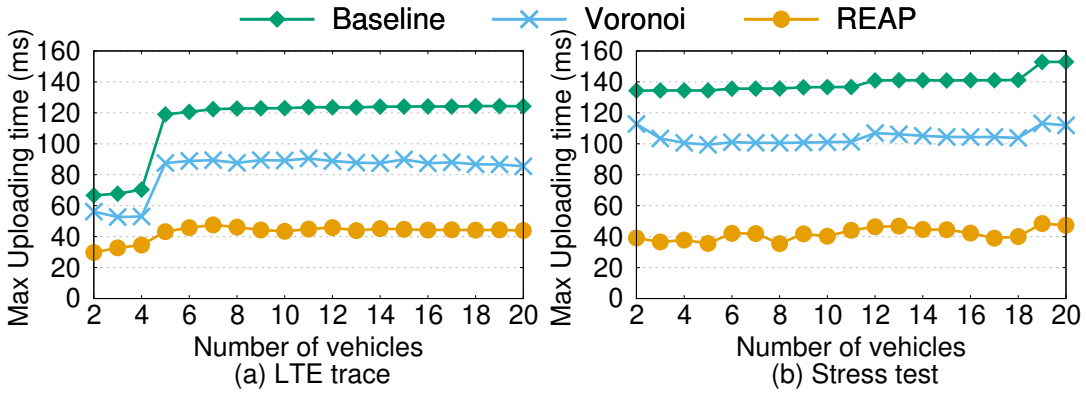


Figure 3.21: Max uploading time of EMP remains stable when there are different numbers of vehicles in the system.

EMP. Figure 3.20 illustrates the pipeline with the average latency of each part. The throughput is determined by the vehicle side processing which takes the longest time (41.39 ms), which means the system can process at 24 frames per second. Note that we directly apply Draco [47] for point cloud compression. More advanced compression approaches [164] can further reduce the vehicle-side latency.

3.5.6 Large-scale Simulation

We next study the scalability of REAP partitioning algorithm on reducing uploaded data size and coping with network fluctuations under a large number of vehicles. We conduct large-scale simulation on the point cloud transmission with LTE uplink network traces and compare the performances of 3 settings: (1) vehicles upload full point cloud frames (Baseline); (2) vehicles partition data based on their locations following the Voronoi diagram (Voronoi); (3) vehicles partition data with REAP (REAP). In detail, we measure maximum frame uploading time (t_{max}) among all vehicles. For REAP, it is the time between when the first byte from any vehicle is sent and when the data from all vehicles

can cover the entire area, which means it is ready for processing based on the conditions defined in §3.3.3. Figure 3.21 (a) plots t_{max} averaged over 1000 runs. By removing redundant data based on vehicle locations, the Voronoi partitioning outperforms the Baseline. Further considering estimated bandwidths of vehicles and dividing the frames into several parts so that vehicles can help each other by opportunistically uploading, REAP provides the best transmission performance. Noticeably, when the number of vehicles increases from 4 to 5, t_{max} increases dramatically for Baseline and Voronoi schemes. The reason for such a sharp increase is that the bandwidths in the fifth randomly selected piece of LTE traces are mostly very low, making the fifth vehicle send the slowest. However, the results of REAP remain stable, thanks to its bandwidth awareness and adaptation. Besides, the standard deviation of t_{max} across different runs becomes better as we enable partitioning, and add bandwidth-aware and adaptation (Baseline: 50.30, Voronoi: 40.07, REAP: 11.68).

To evaluate the system scalability under extreme network conditions, we conduct a stress test. We randomly generate bandwidth profiles for different vehicles following normal distributions. The average bandwidths vary greatly across different vehicles (4-18Mbps) and the standard deviation is 1/4 of the mean. We also impose up to $\pm 40\%$ estimation errors for REAP. As shown in Figure 3.21 (b), the performance of different algorithms aligns with the first simulation results, proving the robustness of REAP.

3.6 Summary

Through edge assistance and adaptive spatial partitioning, EMP makes multi-vehicle perception scalable, robust, and efficient. We believe that EMP can enable or boost a wide range of cooperative sensing applications that require multiple participating vehicles,

in particular given the fast deployment of mobile networks such as 5G that offers high bandwidth and low latency. In addition to ground transportation, the underlying concept of EMP can be potentially generalized to other domains such as cooperative UAVs (drones).

CHAPTER IV

A Variegated Look at 5G in the Wild: Performance, Power, and QoE Implications

This chapter[1] describes an in-depth measurement study of the performance, power consumption, and application quality-of-experience (QoE) of commercial 5G networks in the wild. We examine different 5G carriers, deployment schemes (Non-Standalone, NSA *vs.* Standalone, SA), radio bands (mmWave and sub 6-GHz), protocol configurations (*e.g.,* Radio Resource Control state transitions), mobility patterns (stationary, walking, driving), client devices (*i.e.,* User Equipment), and upper-layer applications (file download, video streaming, and web browsing).

4.1 Introduction

5G New Radio (NR) specifications [83] open a wide spectrum of frequencies. High-band millimeter wave (mmWave) 5G, along with its mid-/low-band sub-6 GHz counter-

[1]Part of this work was carried out in collaboration with Arvind Narayanan, who was a Ph.D. student at the University of Minnesota at the time.

part, make up the current 5G market. We pay close attention to mmWave 5G due to its ultra-high bandwidth which attracts emerging bandwidth-hungry applications. On the other hand, mmWave is very sensitive to factors such as mobility and blockage due to its much shorter wavelength, making the upper-layer network management (*e.g.,* bitrate adaptation of video streaming) more challenging. Despite numerous studies on modeling and simulation of mmWave links [125, 162, 132, 154, 262, 206, 266, 263], the impact of mmWave on commercial 5G performance, power consumption, as well as mobile application Quality-of-Experience (QoE) is largely under-explored.

In addition to its high bandwidth and low latency enabled by physical-layer innovations (*e.g.,* massive MIMO, advanced channel coding, *etc.*), power saving is a top concern to mobile users of 5G. In cellular networks, this is usually achieved by different Radio Resource Control (RRC) states. 5G makes no exception. It is thus important to understand the RRC state machine of commercial 5G networks and its implications. To reduce time to market, most carriers employ the Non-Standalone (NSA) mode for their initial deployment. NSA leverages 5G for data plane operations while reusing the existing 4G infrastructure for control plane operations, making the RRC state machine 4G-like. Very recently, Standalone (SA) 5G deployment has hit the commercial landspace. SA is completely independent of the legacy 4G cellular infrastructure, fully unleashing the potential of 5G. The configurations of key parameters in the state machine lead to important performance and energy trade-offs. They are usually carrier-specific and can be very different between NSA and SA deployment modes.

In order to understand commercial 5G networks' end-to-end performance and power characteristics, as well as their Quality of Experience (QoE) implications on mobile appli-

cations, we conduct a comprehensive yet in-depth measurement study of two commercial 5G networks in the US. As 5G technology evolves, its performance is expected to improve over time. We therefore compare our measurement results with earlier studies to get the initial longitudinal insights on 5G's evolution. We also compare our findings on mmWave with its low-band counterpart. Our study faced a number of challenges:

- 5G-NR supports a wide range of frequency spectrum: low-band, mid-band, and mmWave. All these frequency bands have different performance and signal propagation characteristics. Additionally, 5G can be deployed in either SA or NSA mode, which further has implications on performance [135]. Conducting a measurement study on such a heterogeneous ecosystem is challenging.

- The coverage of different bands and deployment modes is often sporadic. For instance, in the case of mmWave with poor signal propagation characteristics, most of its deployment is outdoor. Surveying the availability of band-specific 5G service requires extensive field experiments.

- Evaluating mobile carriers' end-to-end network performance in the wild is known to be difficult. Many entities can become the performance bottleneck including the Internet, mobile carrier's infrastructure, as well as end devices themselves. Identifying the bottleneck in mmWave 5G is particularly challenging due to its ultra-high bandwidth.

- 5G power measurement is not trivial. The state-of-the-art hardware power monitors often require a stable external power supply, making mobility experiments difficult to perform in the wild. In addition, vendors have been making smartphones more

"closed" by integrating the battery and back cover with the main body. Skilled engineering efforts are required to connect off-the-shelf 5G smartphones to a power monitor. It's not easy to connect any commodity off-the-shelf (COTS) 5G capable smartphones to a power monitor and conduct outdoor experiments.

- To understand the benefits that 5G brings to mobile applications and to identify the new challenges in 5G, we need fair comparisons with the 4G baseline. However, 4G and 5G have very different characteristics, making it difficult to experimentally compare them in a fair, efficient, and representative way.

To address these challenges, we first build a holistic testbed consisting of commercial 5G smartphones, external power monitors, and cloud servers. We further develop a set of software and hardware tools to control the workloads and physical environments, as well as to log important information at different layers in a fine-grained manner. Through carefully designed experiments, we demystify the current 5G performance, power, and QoE implications with special emphasis on mmWave. Our experiments over a 4-month period consumed more than 15 TB of cellular data. The key contributions of our study are summarized as follows.

- We perform a detailed performance examination of 5G over multiple frequency bands including sub-6 GHz and mmWave. We find that both their throughput and latency have experienced noticeable improvements compared to its initial deployment. The end-to-end performance is highly correlated with geographical properties. We quantify such properties and their vastly different impacts on NSA and SA 5G. In particular, we perform experiments over T-Mobile's SA 5G deployed for their low-

band network. This is to our knowledge the first examination of commercial SA 5G performance.

- Through principled probing algorithms, we infer the RRC states and configuration parameters for SA 5G (T-Mobile) and NSA 5G (Verizon and T-Mobile). For NSA 5G that relies on 4G as an anchor, we find that the NR_RRC_CONNECTED to LTE_RRC_IDLE state transition (due to data inactivity on UE) for the carriers considered in our study is 2× more energy efficient than those studied in a previous NSA 5G measurement study [242].

- We take a closer look at the power characteristics of 5G and 4G/LTE. Over downlink (uplink), 5G can be 79% (74%) less energy-efficient than 4G at low throughput but up to 5× (2×) more energy-efficient when the throughput is high. Using a data-driven approach, we build a *first* throughput and signal strength-aware radio power model for different frequency bands of 5G.

- We conduct a *first* evaluation of state-of-the-art adaptive video bitrate adaptation (ABR) algorithms over mmWave 5G, which is the key radio technology for supporting ultra-high definition (UHD) videos and beyond. We find that due to the poor signal propagation characteristics of mmWave 5G, existing ABR mechanisms over mmWave 5G can incur ∼3.7% to 259.5% higher stall time than 4G/LTE. We propose simple yet effective interface selection mechanisms for 5G video streaming. It can yield a 26.9% video stall reduction and a 4.2% improvement in energy efficiency without compromising user-perceived video quality, compared to unmodified streaming algorithms.

Table 4.1: Statistics of the data collected using two commercial 5G carriers: Verizon and T-Mobile.

Dataset Statistics	
5G Network Performance Tests	12,500+
Unique servers tested with	157+
Cumulative time of measurement traces	2,666 minutes+
Power Measurements @ 5000 Hz	2,336 minutes+
Total kilometers walked	148.5 km+
# of real Web Page Load Tests	30,000+
# of 5G smartphones (and models)	7 (3)

- We collect a large dataset consisting of more than 30,000 web page loadings of diverse websites, and use it to compare mmWave 5G vs. 4G page load time and energy consumption. We find that overall 5G improves the page load time at the cost of higher energy consumption compared to 4G. Moreover, this impact is highly web-page-dependent. We build decision tree models that can intelligently select the appropriate network (5G or 4G) for web browsing.

- We have released the functional artifacts (both datasets and tools) of our study [62].

4.2 Measurement Settings & Tools

5G Carriers, 5G Bands and Locations. Since its commercial launch, the 5G ecosystem – which includes service deployments, coverage, 5G-capable devices – is rapidly expanding and evolving. In our measurement study, we select two commercial carriers in the US for our experiments – Verizon and T-Mobile. While both these carriers have deployed 5G services on several bands, in our dataset, we find that Verizon has deployed NSA-based 5G service that provides both mmWave 5G over 28/39 GHz frequency bands

(n261/n260) and low-band 5G (n5) w/ 4G bands by leveraging dynamic spectrum sharing (DSS) technology. In contrast, T-Mobile provides low-band (@ 600MHz or n71)[2] 5G service using *both NSA and SA* modes. The measurement study is conducted in two US cities where both carriers have deployed 5G services. Key statistics of the datasets collected are summarized in Table 4.1.

5G UE and Android Measurement Tool. We use multiple smartphone models of user equipment (UE) with 5G support: Google Pixel 5 (PX5), Samsung Galaxy S20 Ultra 5G (S20U) and Samsung Galaxy S10 5G (S10). These phones have diverse specifications. For instance, compared to PX5, S20U has a superior chipset, 5G modem, increased RAM and CPU frequencies. We make considerable additions to 5G Tracker [183] and build a comprehensive monitoring toolkit with various functions to monitor network traffic, battery status (current and voltage), signal strength *etc.*. Some of these functions require rooting the phones. We use both rooted and non-rooted phones (based on needs) to measure various aspects of 5G performance and power usage under different settings.

Power Monitoring Tool. We use Monsoon Power Monitor [59] to power smartphones and measure the power consumption. For outdoor walking experiments, we use a portable external power source to supply power to the monitor.

4.3 Improvements and New Findings in 5G Network Performance

In this section, we closely examine the end-to-end network performance of commercial 5G networks by conducting several carefully designed experiments in the wild.

[2] T-Mobile also provides NSA-based mid-band (n41) and mmWave 5G (n261/n260) in select areas. However, these services were not considered in this study.

4.3.1 Measurement Methodology

Challenges. There are several known challenges while evaluating end-to-end network performance of mobile carriers in the wild. **[C1]** First, Internet-side congestion can adversely affect network performance. **[C2]** Secondly, we also have no clear visibility into the carrier's network/transport infrastructure and policies enforced by them. **[C3]** Finally, there is significant diversity in end-device (*e.g.,* server or smartphone) specifications and capabilities which can affect network performance.

Methodology. We now describe our carefully designed methodology for evaluating 5G network performance. Ookla's *Speedtest* service [191] is a widely used and state-of-the-art tool for testing Internet connection bandwidth and latency. By default, Speedtest chooses a geographically nearby server with the least round-trip latency to measure downlink/uplink throughput. They also allow users to choose a server from a pool of geographically distributed servers. More importantly, both the 5G carriers studied host servers on Ookla. For instance, Verizon hosts 48 servers while T-Mobile hosts 47 servers. These are mainly located in major metropolitan US cities. We leverage the flexibility of server selection as well as the carrier's presence in Speedtest's pool of server network to evaluate a carrier's network performance by conducting several tests on carrier-hosted servers. Particularly, this strategy helps us reduce the impact of **[C1]** and **[C2]** on our measurement tests.

The default policy of server selection from Speedtest is to choose a server located in the same city as the UE. We also experimentally confirm that using carrier-hosted Speedtest servers (especially if one is available in the UE's city) usually provides best performance over non-carrier based servers. Even when testing using carrier-hosted servers in other states and cities, we believe this strategy helps eliminate most of the Internet side bot-

tleneck as the carrier would usually place Speedtest servers at the edge of the carrier's city-level ingress points. Speedtest service uses TCP protocol for all its tests. Speedtest additionally also allows us to conduct a test in one of the two connection modes: **(i)** using a single connection and **(ii)** using multiple connections that is non-configurable. The number of multiple connections varies from one test to another, and the algorithm is not disclosed on how Speedtest decides the number of connections to establish for a test. To account for this limitation, we also provision VMs with high network-throughput (in different US locations) provided by Microsoft Azure's public cloud service. This allowed us to evaluate the impact of different transport layer protocols and parameters.

Lastly, we take two steps to address **[C3]**. First, to account for UE diversity, we use two 5G smartphones: PX5 and a more powerful S20U (§4.2). Secondly, in addition to the carrier-hosted Speedtest servers, we also use *all* the Speedtest servers located in the local state of the UE. This allows to reduce the impact of geographic distance on network performance, rather allows us to understand the impact of other *potential* server-side factors over 5G network performance. For each unique `<UE-model,carrier,server>` setting, we repeat the test at least 10 times per connection mode. Our dataset contains over 12,500 Speedtest measurements. We develop scripts for Android smartphones to completely automate the process of conducting a test using Ookla's Speedtest service (free version). We report the 95th percentile performance results of all Speedtest sessions repeatedly conducted for a setting. In other words, our approach measures the peak network performance, and should *not* be confused with the user perceived network quality metrics [23]. Focusing on the peak metrics helps us to further reduce the impact of congestion and other Internet-side factors on our performance measurements, and rather helps us understand the impact

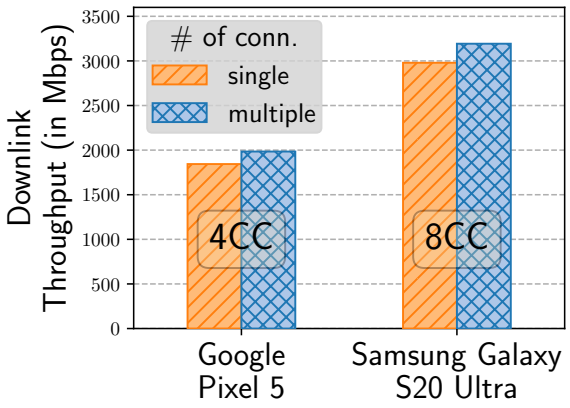


Figure 4.1: Support for improved carrier aggregation schemes in 5G-NR radios boost throughput performance.

of UE-Server distance and radio technology/band over network performance. Having this information is particularly important for application and service providers so that they can better harness 5G. Unless specified otherwise, all mmWave-5G based experiments were conducted outdoors and the UE was held stationary with clear LoS to the 5G tower.

Baseline. To provide the initial longitudinal insights of commercial 5G's network performance in the US, we consider *5Gophers* [180] dataset (reportedly measured in the US as of October 2019) as the baseline for comparing results.

4.3.2 Impact of UE-Specs and Capabilities

Commercial 5G landspace has improved over time along several dimensions. Most notably for this experiment that tries to quantify the impact of UE-specs on network performance, we find that latest high-end smartphones such as S20U are able to improve downlink and uplink throughput by increasing the number of radio channels (often referred to as carrier aggregation) used between the UE and RAN. For example, previous genera-

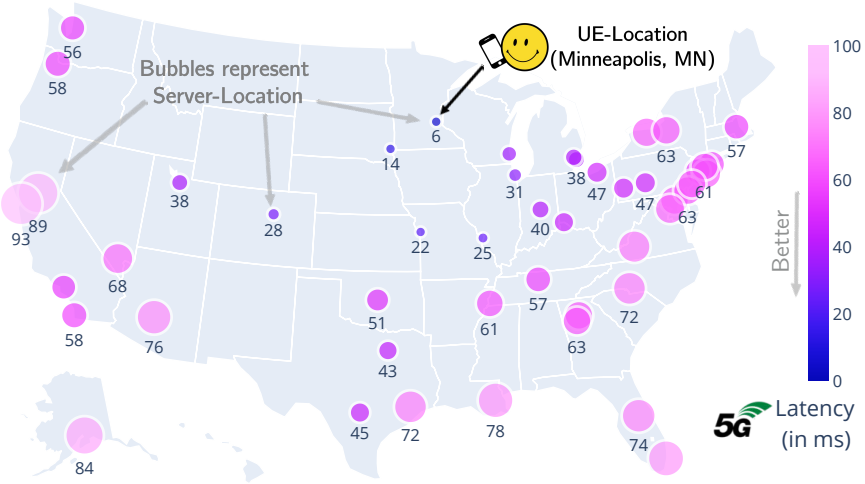
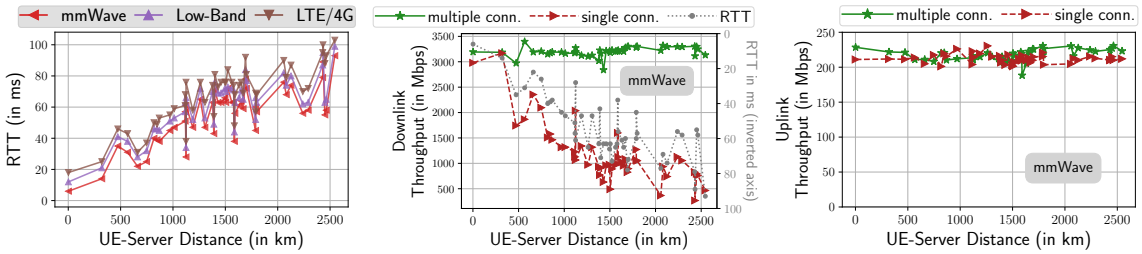


Figure 4.2: Impact of UE-Server distance on RTT.

tion of 5G smartphones (*e.g.,* considered in the baseline with QC X50 modem [25]) as well as the cheaper variants of mmWave 5G phones (*e.g.,* PX5 with QC X52 modem [36]) uses 4×100 MHz or 4CC (component carriers) for downlink data transfers and 1CC for uplink. On the other hand, S20U (with QC X55 modem [37]) supports 8CC over downlink (and 2CC over uplink) resulting in significant improvements in throughput performance. Figure 4.1 compares the downlink and uplink throughput between PX5 and S20U. Clearly, S20U provides 50% to 60% improvements in both uplink and downlink throughput over PX5 and the baseline. Of course, harnessing for such carrier aggregation schemes over mmWave bands also requires support from 5G carriers and their infrastructure. We did not find any significant impact of UE specs over latency.

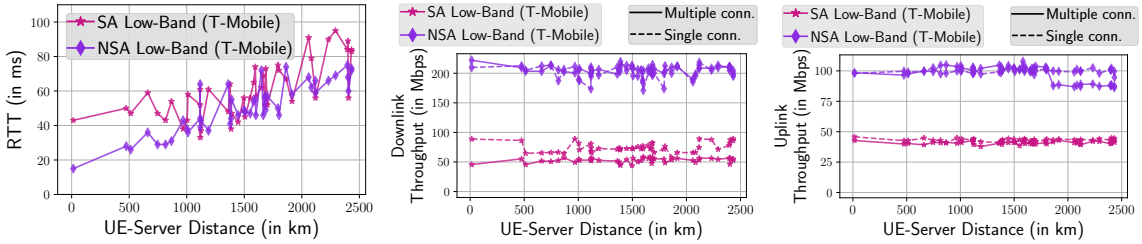
4.3.3 Impact of UE-Server Distance

Latency. By tapping into the 5G carriers' nationwide network of Speedtest servers, we next quantify the impact of UE-Server distance over round-trip time (RTT). Figure 4.2



(a) Latency. (b) Downlink throughput. (c) Uplink throughput.

Figure 4.3: End-to-end performance of Verizon's networks in the wild.



(a) Latency. (b) Downlink throughput. (c) Uplink throughput.

Figure 4.4: End-to-end performance of T-Mobile's networks in the wild.

shows the latency characteristics of Verizon's mmWave 5G service for different server locations on a geographic map. UE's location is fixed as Minneapolis, MN. Clearly, RTT degrades severely as the UE-Server distance increases. The lowest observed RTT is ∼6ms when tested with a server located closest (∼3 km) to the UE. Compared to latency observed back in 2019 [180] (*i.e.,* during early deployment), this is a ∼50% improvement over the baseline performance. RTT gets doubled as the UE-Server distance increases to 320 km. This trend is more clearly visible in Figure 4.3a[3] which further signifies the importance of *edge computing* for latency-sensitive apps and services. Figure 4.3a also compares RTT values of mmWave 5G against that of low-band 5G and 4G/LTE. We find that low-

[3]Figures 4.3 and 4.4 shows servers located in the contiguous US region only.

band 5G suffers an additional delay of ∼6 to 8 ms over mmWave 5G across the entire UE-Server distance range. This is not surprising as mmWave 5G bands (n260/n261) with higher subcarrier spacing and shorter OFDM symbol duration lead to lower latency when compared to low-band 5G [216, 214]. On the other hand, due to flexible frame structure and fine-grained transmission time interval (TTI) in 5G NR, we find both low-band and mmWave 5G exhibit better RTT (6 to 15ms reduction) than LTE. Similar experiments were also conducted over T-Mobile's network (including SA Low-Band 5G) and results are shown in Figure 4.4a. While the earlier trend observed in Verizon's network about the impact of UE-Server distance over RTT also holds true for T-Mobile's network, we do not find any significant difference yet in RTT performance between T-Mobile's SA and NSA deployments of low-band 5G.

Throughput. Figure 4.3b shows the impact of UE-Server distance on Verizon mmWave 5G downlink throughput performance. With multiple TCP connections, the UE is able to achieve an impressive downlink throughput of over 3 Gbps across all the servers in the US. This is a ∼50-60% improvement over the baseline. We attribute this improvement to ramping up of carrier aggregation from 4CC to 8CC which requires improvements in carrier's infrastructure as well as the UE's chipset specifications (see §4.3.2). As pointed out earlier, Speedtest does not allow us to control the number of TCP connections for a test. Using packet dumps, we found that Speedtest would establish anywhere between 15 to 25 TCP connections for the multiple connection test. The packet loss rate was less than 1%. However, with a single TCP connection, we find that the throughput degrades as the UE-Server distance increases (see Figure 4.3b). We suspect this degradation is due to the: **(1)** increase in RTT which is known to affect TCP performance, **(2)** packet loss

(even at the slightest rate). The impact of both coupled with existing TCP mechanisms gets amplified at ultra-high bandwidth levels thus degrading TCP performance. Nonetheless, compared to the baseline, we find there is a significant improvement in the single TCP (1-TCP) connection's performance. 1-TCP connection (with less overhead compared to multiple connections) can also achieve close to 3 Gbps throughput provided the server is much closer to the UE. This again signifies the importance of the edge especially for bandwidth-hungry applications. Uplink throughput (see Figure 4.3c) performance has also improved by a factor of 3× to 4× over the baseline. Both single and multiple connection uplink tests can achieve a throughput of ∼220 Mbps. On the other hand, for T-Mobile which also has SA-based deployments for the low-band 5G, we find that both downlink and uplink performance can achieve only half the performance of what their low-band NSA 5G service can provide (see Figure 4.4b and Figure 4.4c). We believe this to be due to carrier aggregation not yet supported for SA or that the 5G core is not fully mature to provide the benefits envisioned by SA 5G.

4.3.4 A Closer Look at Single-Connection Throughput

To get a better understanding of single-connection's performance with mmWave 5G (known to provide ultra-high bandwidth capacity), we perform controlled experiments using Microsoft Azure's public cloud service. We provision a high-network bandwidth capacity VM (Type: `DS4_v2`) at every region in the US provided by Microsoft Azure. In order to capture packet dumps and have the ability to change kernel parameters, we use *rooted* PX5 to conduct these experiments. Unlike S20U that can achieve a throughput of more than 3 Gbps, PX5 has a maximum observable downlink throughput of ∼2.2 Gbps

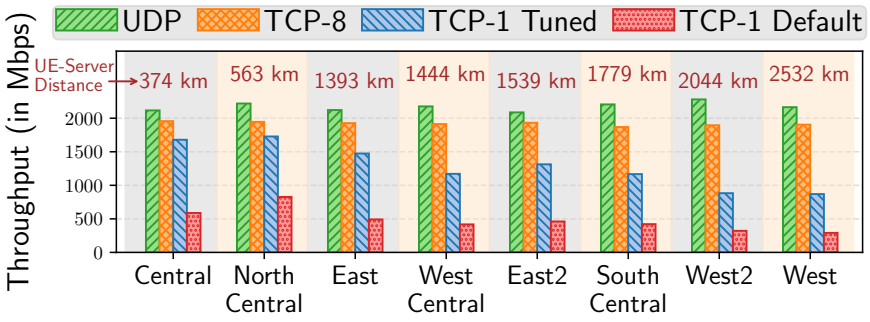


Figure 4.5: Single connection downlink throughput across all US-based Azure regions under different transport layer settings.

(see §4.3.2). For TCP, we use *CUBIC* [56] as the TCP congestion control algorithm. The experimental setup uses UDP performance as baseline. As shown in Figure 4.5, UDP is able to achieve peak observable throughput across all the server locations. We observe a small yet noticeable gap between UDP and 8-TCP performance most likely due to the protocol overhead of TCP. However, with default Linux kernel (`v4.18.0`) parameters for TCP, we find 1-TCP connection's throughput is limited to no more than 500 Mbps for all servers. Upon further investigation, increasing the maximum size of TCP write buffer (`tcp_wmem`) parameter of Linux's TCP kernel significantly improves the UE's downlink throughput using 1-TCP connection by a factor of 2.1× to 3× (denoted as "1-TCP tuned" in Figure 4.5). Theoretically, the sender's TCP buffer size (which is a per socket configuration) must at the least be equal to the bandwidth-delay product (BDP) of the high-throughput flow's capacity. In other words, transport-layer kernel parameters should be carefully tuned to meet the desired application QoE requirements. Nonetheless, even the tuned 1-TCP performance falls short by ∼886 Mbps on average when compared to UDP. Similar to the impact of UE-Server distance observed earlier in Figure 4.3b for the single-connection performance using Ookla's Speedtest service, we make similar observations in

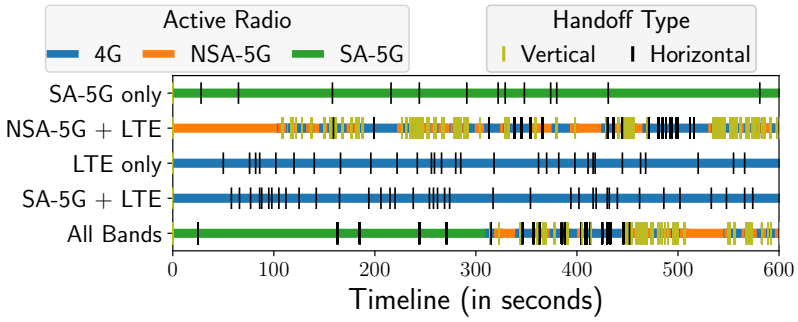


Figure 4.6: Handoff frequency (while driving) across different T-Mobile low-band settings.

performance under controlled experimental settings using Azure servers. In that, we again find that TCP performance (including that of *1-TCP tuned*) exacerbates as the UE-Server distance increases. These observations highlight the inefficacies that exist in current TCP and congestion control mechanisms over mmWave 5G networks.

4.3.5 Handoffs in (Low-Band) NSA & SA 5G

Previous studies on handoffs[4] of NSA mmWave 5G [180] have shown that, compared to 4G/LTE, there are far more frequent handoffs. This is mainly due to the smaller coverage footprint of mmWave towers as well as the fact that NSA 5G still relies on LTE for control plane signaling. In this preliminary study, we focus on comparing T-Mobile's SA 5G with NSA 5G that are commercially deployed. T-Mobile is the only carrier that has deployed both NSA and SA-based 5G for their low-band network. To obtain connectivity to SA 5G (over n71 band), it was critical to use T-Mobile's firmware in S20U. We selected a 10 km driving route which traversed via busy downtown regions and freeways with driving speeds ranging from 0 to 100 kph. Using Samsung's service code (`*#2263#`), we

[4]Handoff here refers to the change in tower or data transmission technology.

selectively enable a set of radio bands to configure the UE in one of the 5 setting: **(i)** enable SA-n71 band only, **(ii)** enable NSA-n71 and LTE bands only, **(iii)** enable LTE bands only, **(iv)** enable SA-n71 and LTE bands only, and, **(v)** enables all bands (default setting). For each configuration, while the UE was handheld by a passenger, we drove the route 2× per direction and monitored the handoff activity. Figure 4.6 shows a representative set of results. There are five horizontal bars, one for each of the 5 band configuration settings. Within each horizontal bar, there are several colored-segments that denoted the active radio (blue for 4G/LTE, orange for NSA-5G, and green for SA-5G). Ticks on these bars indicate the occurrence of a horizontal handoff (*i.e.,* across towers) or a vertical handoff (*i.e.,* across radio technologies). The most important finding here is that SA 5G has far fewer handoffs (*i.e.,* 13 handoffs) compared to other configurations, NSA-5G + LTE (110), LTE (30), SA+LTE (38) and all bands (64). These will have implications not just on control plane signaling and scheduling overheads, but also over network performance. Due to increased coverage of the low-band RF n71 band, both SA and NSA over n71 band experience very few horizontal handoffs (13 to 20). But, in NSA, we found close to 90 vertical handoffs (*e.g.,* 4G to 5G or vice-versa) highlighting the complexities involved in NSA.

Now that we have seen the network performance characteristics of different 5G technologies, next we investigate how such performance characteristics impact power.

4.4 Power characteristics

In this section, we discuss the power characterization of 5G network and compare with the latest 4G results. To better understand the UE's power consumption, we construct power models for different 5G networks with multiple factors including signal strength,

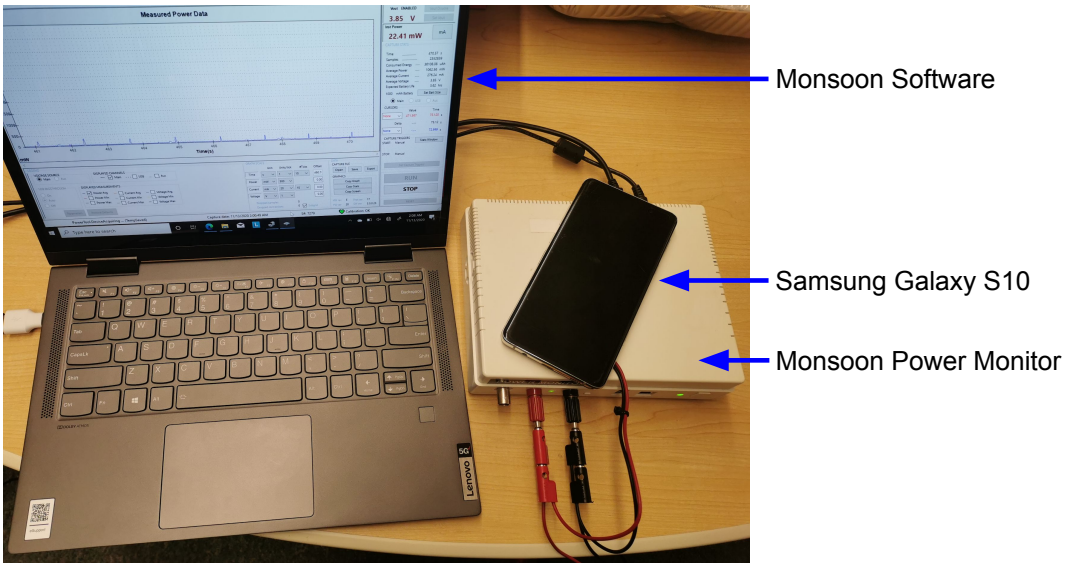


Figure 4.7: Experimental setup: a smartphone is powered by a Monsoon Power Monitor and a laptop running the Monsoon software is connected to the power monitor.

throughput, and frequency bands.

4.4.1 Methodology

RRC state inference. We first derive the built-in radio state machine which was designed for power management of mobile devices, *e.g.,* parameters of RRC states and transitions for 4G [41] and 5G [42]. For the parameter inference, we improve a network-based approach used in existing studies [138, 209] to build our own inference tool, RRC-Probe, in which a server sends UDP packets to a client (UE) at different packet intervals and the UE sends an ACK once a packet is received. The length of RTT depends on the UE's instant RRC state when receiving the packet. Therefore, by measuring the RTT for different packet intervals, we can identify different states and calculate the timers for the transitions between states. Note that this approach does not require root access on smartphones.

Power measurement. We use Monsoon power monitor [59] to measure the UE's power consumption for two purposes: First, we aim to understand power consumption during RRC state transitions. To measure this, the UE is left idle without any data activity for sufficient time (20s in our experiments) thus forcing the UE to be in RRC_IDLE state. A server then sends a packet to the UE which subsequently triggers an RRC_IDLE $\rightarrow$ RRC_CONNECTED transition and switch to 5G. Then, the UE starts its inactivity timer and demotes to RRC_IDLE at the end. In this way, the power monitor can capture the full tail period[5] for RRC_CONNECTED. Second, to study the throughput-power relationship and its implications on energy efficiency, we control the UE's data transfer throughput while measuring its power. To reduce the impact of power consumption due to display screen and brightness, we set the screen at the maximum brightness level and subtract the screen power (which is obtained separately) from the total when presenting the results. Figure 4.7 shows our experimental setup. In this study, power (in W) refers to energy consumed per unit time.

Data Collection Methodology. We conduct both controlled and in-the-wild walking experiments to collect network and power traces at two different cities in the US – Minneapolis, MN and Ann Arbor, MI – using two commercial 5G carriers (Verizon and T-Mobile). For Verizon, we collect data for their NSA-based mmWave 5G as well as their low-band 5G service. For T-Mobile, we focus on their low-band 5G which is deployed in both SA and NSA modes. For all our experiments, we use two models of 5G smartphones: S10 and S20U. For the walking experiments, we fixed a 20-min loop ($\sim$1.6km).

[5]The period after Continuous Reception (*i.e.,* when UE finishes its data transfer) and before demoting to RRC_IDLE in which there are discontinuous reception cycles (DRX) and the UE can reduce power consumption.

Table 4.2: Important 4G/5G RRC parameters using RRC-Probe.

Mobile Service		RRC Parameter (ms)				
Carrier	**Radio type**	**UE-inactivity timer**	**Long DRX cycle**	**IDLE DRX cycle**	**4G promotion delay**	**5G promotion delay**
T-Mobile	SA low-band	10400	40	1250	N/A	341
T-Mobile	NSA low-band	10400 (12120)	320	1200	210	1440
Verizon	NSA mmWave	10500	320	1280	396	1907
Verizon	NSA low-band	10200 (18800)	400	1100	288	N/A
T-Mobile	4G	5000	400	1300	190	N/A
Verizon	4G	10200	300	1280	265	N/A

While low-band 5G connectivity for both carriers was omnipresent, mmWave was rather limited. The loop contains three mmWave 5G towers each fitted with three directional mmWave transceivers. We collect 10 traces for each unique carrier-mode-band setting (*e.g.,* Verizon-NSA-Low Band). The power monitor collects data at 5000Hz while we set the network logging rate at 10Hz. As the traces are collected separately by 5G Tracker tool [183] and Monsoon power monitor, we synchronize them by starting both loggers at the same time and further verify by correlating measurements activities known to cause significant power jump.

4.4.2 RRC Parameters and Power

4.4.2.1 RRC State Machine Parameters

Using RRC-Probe, we infer a list of RRC parameters for 4G and 5G. We summarize the timers of RRC state transition for different networks, carriers, and band configurations in Table 4.2. When the radio is active and there are no incoming/outgoing packets, UE starts the tail timer (*i.e.,* UE-inactivity timer) and stays in RRC_CONNECTED for T_{tail} before demoting to RRC_IDLE. Discontinuous Reception (DRX) is adopted by both 4G and 5G for power saving in which UE periodically wakes up to check paging messages and rests for the remaining time of the cycle. The periods in RRC_CONNECTED and RRC_IDLE

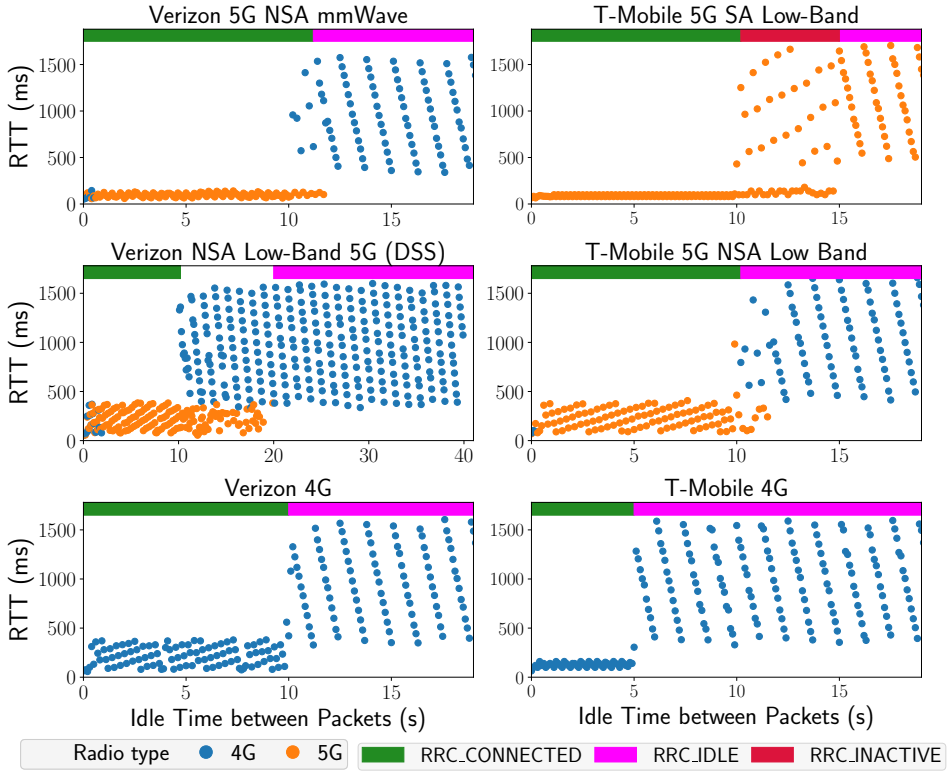


Figure 4.8: Results of inferring different RRC States using RRC-Probe for SA 5G, NSA 5G and 4G/LTE.

are different. T_{long_drx} is the cycle period of Long DRX in RRC_CONNECTED and T_{idle_drx} is the cycle period of DRX in RRC_IDLE. We do not observe and infer Short DRX cycle with RRC-Probe due to its very small cycle period. We also calculate the delay for promotion from RRC_IDLE to 4G and 5G which is T_{4g_pro} and T_{5g_pro} respectively. From the results, we find that the timers of NSA 5G and 4G LTE are very similar. This is because NSA 5G still retains the existing 4G infrastructure for control plane operations while innovating the data plane to enhance the network capacity.

Figure 4.8 illustrates the results of inferring the RRC states for all the configurations. For NSA 5G, the RRC states are basically the same as 4G. However, according to the

5G-NR specifications [83], a new RRC state called RRC_INACTIVE is introduced in SA 5G. We believe this new state can be seen in Figure 4.8 (see top right part representing T-Mobile SA 5G). We find that the UE remains in this state for about 5s (*i.e.,* 10s to 15s of interval) before transitioning to RRC_IDLE. The main purpose of this state (akin to a low-power state) is to provide an efficient mechanism for the UE's radio to sleep (thus saving power) and at the same time enable a quick and lightweight transition back to the RRC_CONNECTED state (thus improving latency by reducing the radio's wake up time). These benefits are largely achieved by reducing the control plane signaling overhead. Besides, we notice that T-Mobile SA 5G has a tail timer of 10s which is similar to that of T-Mobile NSA 5G and Verizon NSA 5G, indicating UE directly enters RRC_IDLE after leaving RRC_CONNECTED. We also confirm the timers using Monsoon power monitor. This is different from the observations in prior work [242] that found the 5G tail is 20s, *i.e.,* 2× of 4G tail (10s), which indicates the 5G module must go through both 5G and 4G tails before entering RRC_IDLE. Careful attention needs to be given in configuring such timers as they impact energy efficiency. Although not shown, for 4G → 5G promotion in NSA 5G, UE will first promote to 4G's CONNECTED state before switching to 5G (*i.e.,* LTE_RRC_IDLE → LTE_RRC_CONNECTED → NR_RRC_CONNECTED). In SA 5G though, the UE will directly reach NR_RRC_CONNECTED. Note, we observe that in NSA, sometimes the packets might arrive over 4G interface (with higher latency) while other times packets might arrive over 5G interface (with lower latency). This can be seen for the NSA low-band 5G setting for both Verizon and T-Mobile carriers. We have therefore also mentioned a second tail-timer for such settings (see timers in brackets in Table 4.2).

Table 4.3: Power during RRC state transitions.

Carrier	Network	Power (mW)	
		Tail	*4G→5G switch*
Verizon	4G	178	N/A
T-Mobile	4G	66	N/A
Verizon	NSA 5G (low-band,DSS)	249	799
Verizon	NSA 5G (mmWave)	1092	1494
T-Mobile	NSA 5G (low-band)	260	699
T-Mobile	SA 5G (low-band)	593	245

4.4.2.2 Power during RRC State Transitions

We next study the impact of 5G on power during RRC state transitions. We calculate the tail power by averaging the power readings during the entire tail period considering both DRX On duration and the rest of the DRX cycle. As shown in Table 4.3, 5G consumes more energy than 4G during the tail period and for mmWave 5G the tail power is especially higher. This is likely because the UE's radio remains active during the tail period in order to wake up periodically for paging and 5G radio consumes more power than 4G (when the throughput is zero, shown later in §4.4.3). Further taking into account the 4G $\rightarrow$ 5G switch which consumes additional power and is very common (see Figure 4.6), 5G is less efficient in terms of state transitions. Therefore, to save power, traffic patterns like periodical data transmission or intermittent waking up should be avoided under 5G. One solution would be forcing the UE to stay in 4G when high throughput is not needed.

4.4.3 Power for Data Transfer

Previous work on 3G/4G power modeling [138] has constructed power models for data transfer by taking into account the device throughput and concluded that higher through-

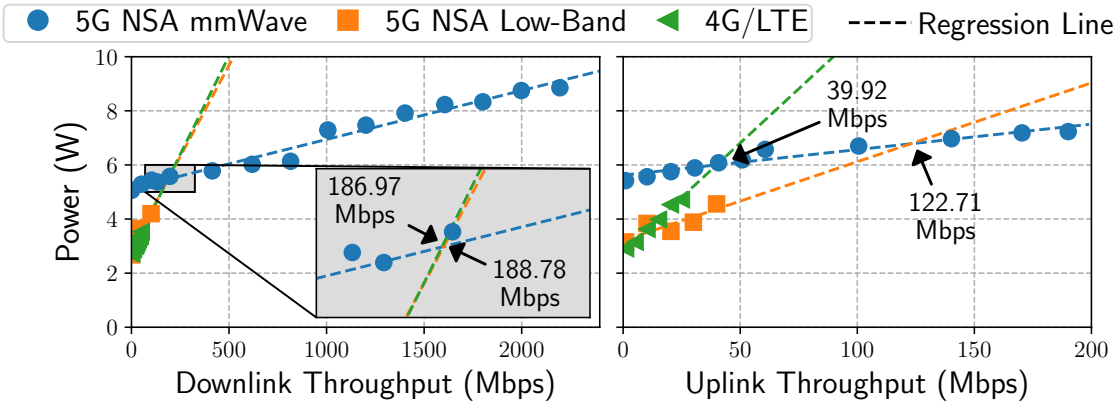


Figure 4.9: Throughput *vs.* power for 4G and 5G (S20U, Minneapolis).

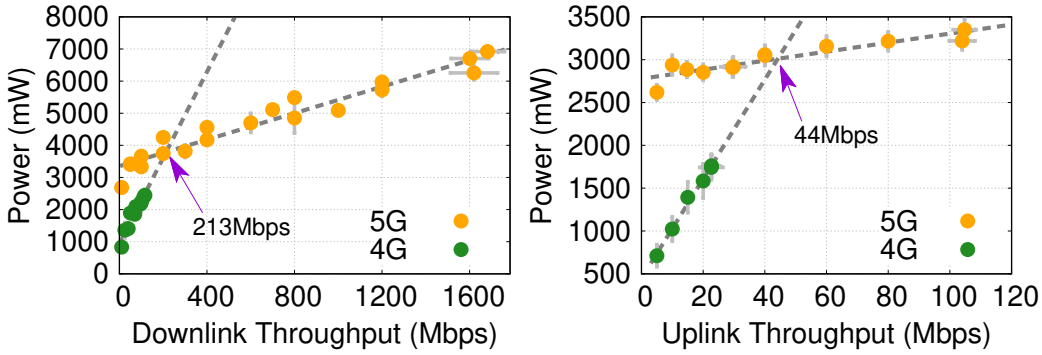


Figure 4.10: Throughput *vs.* power for 4G & mmWave 5G (S10, Ann Arbor).

put leads to higher power consumption. As 5G (especially mmWave) can provide much higher throughput compared to 4G, we study how throughput affects the device power over 5G. With controlled experiments, we measure the device power when transferring data at different download/upload throughput over 4G and 5G. We run UDP data transfer and vary the target throughput using `iPerf`. To reduce the impact of poor signal propagation issues of mmWave 5G, we run the experiments by hand-holding the smartphone at a fixed location with Line-of-Sight (LoS) to a 5G panel.

Figure 4.9 presents the relationship between throughput and power with a comparison

between 4G/LTE and 5G. We also show this relationship across two different bands of 5G: NSA low-band (LB) and NSA mmWave. These experiments were done on S20U over Verizon. We can find that for both 4G and 5G, and for both uplink and downlink directions, the power increases linearly as throughput increases. However, the power for mmWave 5G (uplink and downlink) increases at a slower rate than for the other two radio networks. Although at low throughput levels the power consumption for mmWave 5G is higher, it becomes more efficient when the throughput is high. As seen in Figure 4.9, the crossover point at which mmWave 5G becomes more efficient than 4G and low-band 5G is: **(1)** 187 Mbps and 189 Mbps for downlink; **(2)** 40 Mbps and 123 Mbps for uplink. These results clearly reveal the power-performance relationships (and trade-offs) between not just 4G and 5G but also the different bands within 5G. Note that different UE models may have varied levels of power consumption [105].

We find that the power consumption across different UE models can be different [105]. Similar to Figure 4.9 which reports the throughput-power relationship for mmWave 5G, low-band 5G, and 4G using S20U, we also conduct the same set of experiments using S10 smartphones which have relatively older 5G modems and chipsets. The results are shown in Figure 4.10. For the downlink and uplink transfer, we echo the observations made above that mmWave 5G uses more power than 4G/LTE at low throughput levels, but mmWave becomes more efficient at higher throughput levels. The crossover points between mmWave 5G and 4G/LTE observed using S10 are different from those measured using S20U.

It is also interesting to compare the slopes between low-band 5G and 4G/LTE. We derive the slopes of throughput-power curves across different device models and radio

bands/technologies and list them in Table 4.4. In the downlink direction, the slopes of LB-5G ad 4G/LTE are almost identical. In the uplink direction though, LB 5G is much more efficient than 4G/LTE.

Table 4.4: Slopes of Throughput-Power curves indicating increase in power for every 1 Mbps rise in throughput.

Device	**Network**	**Downlink (mW/Mbps)**	**Uplink (mW/Mbps)**
S10	4G	13.38	57.99
S10	5G (mmWave)	2.06	5.27
S20U	4G	14.55	80.21
S20U	5G (low-band)	13.52	29.15
S20U	5G (mmWave)	1.81	9.42

Next, we calculate the proportion of power consumed by data transfer activity out of total power. On average, data transfer in mmWave 5G consumes 48-76% of the total power consumption for downlink and 46-66% for uplink, while the same for 4G are 21-53% (downlink) and 20-66% (uplink). This is similar to what was also observed earlier by Xu *et al.* [242] (for mid-band 5G). But our results show that the upper bound for 5G downlink is higher by an additional 21% when compared to [242], which is likely due to higher data rates offered by mmWave 5G.

We further calculate the energy efficiency (energy per bit) and plot the results in Figure 4.11 and Figure 4.12 with a log scale, where we can also conclude the higher efficiency when transferring at higher speeds under 5G. 5G can be 79% (74%) less efficient than 4G at a low throughput but up to 5× (2×) more when the throughput is high, for downlink (uplink). In fact, this can also be confirmed from mathematical modeling: Assume the device power is P, energy efficiency is E and the throughput is T, we will have $P = c_1 * T + c_2$

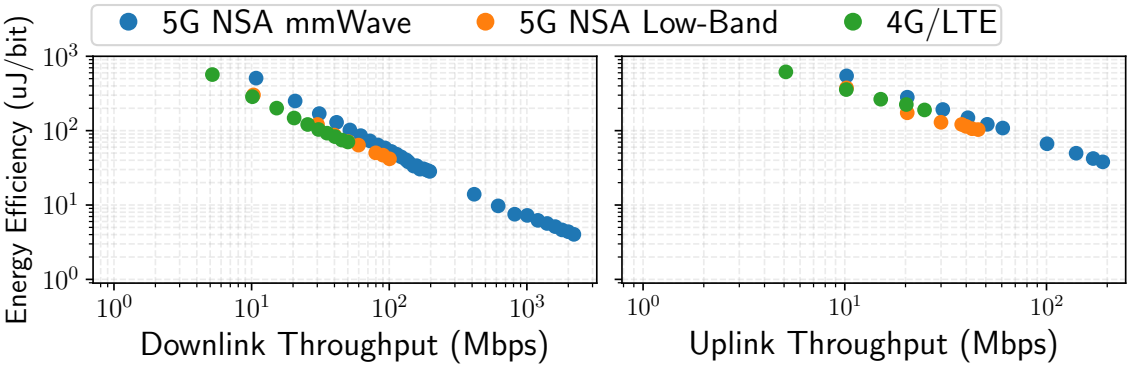


Figure 4.11: Throughput *vs.* energy efficiency for 4G and 5G (S20U, Minneapolis).

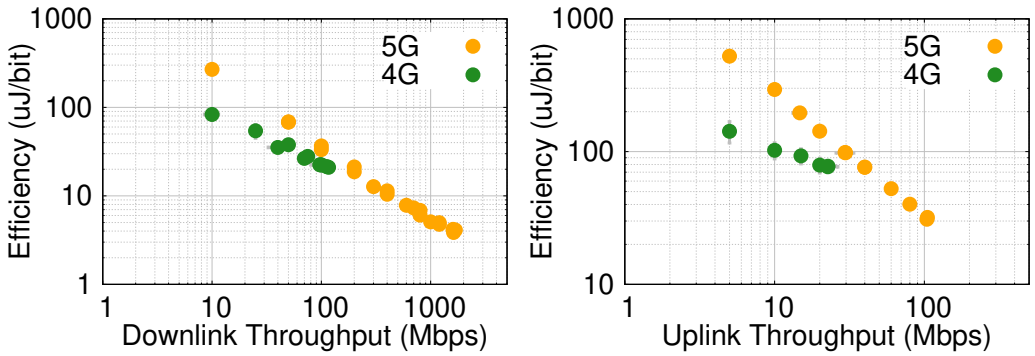


Figure 4.12: Throughput *vs.* energy efficiency for 4G & mmWave 5G (S10, Ann Arbor).

and $E = P/T = c_2/T + c_1$. So we can get $\log E \approx c_3 * \log T + c_4$, by taking logarithm on both sides of the equation. Here c_i is constant.

Downlink *vs.* Uplink. We next compare the downlink transfer with uplink transfer for 4G and 5G (Figure 4.9). Based on the carrier configurations, we conclude that the rate of increase in power consumption for uplink is higher by 2.2× to 5.9× than downlink, which is in consensus to prior work on 3G/4G [138]. Unsurprisingly, UE's radio requires more power for sending data than to receive [108].

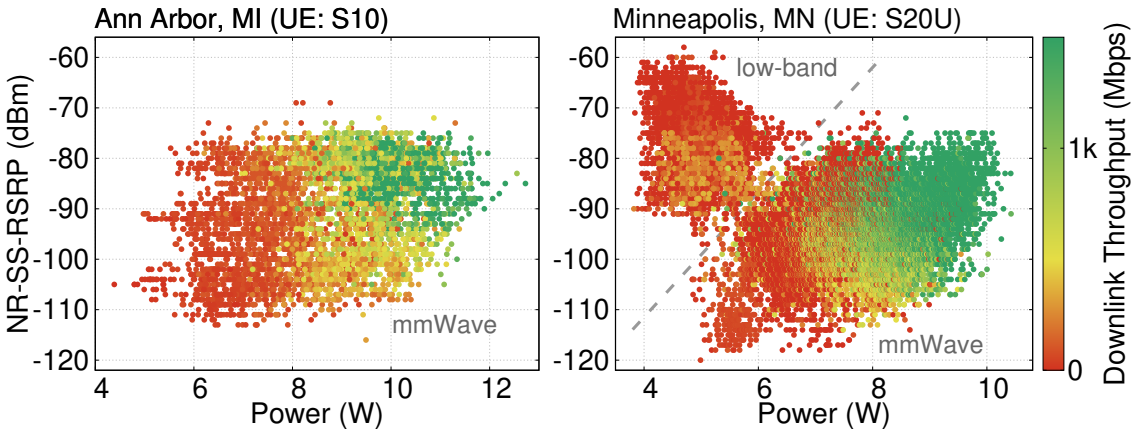


Figure 4.13: Power-RSRP-Throughput relationship.

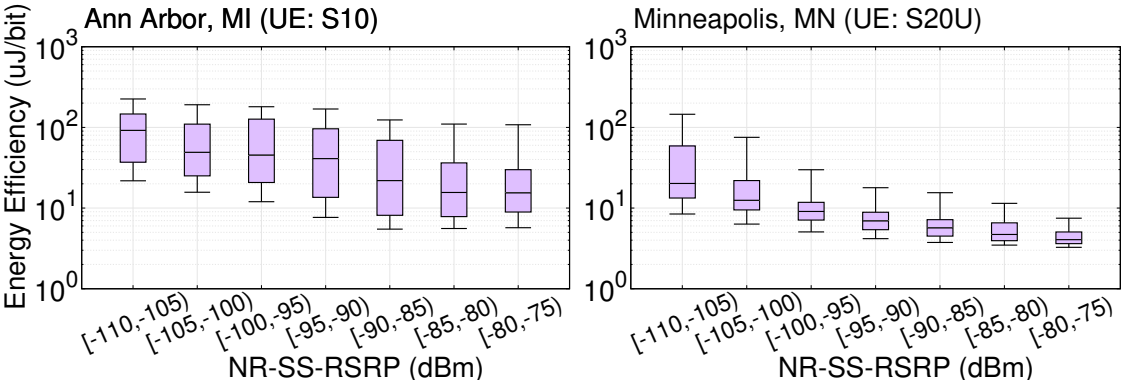


Figure 4.14: Energy efficiency-RSRP relationship (mmWave).

4.4.4 Impact of Signal Strength on Power

In addition to throughput, there are other factors affecting the power consumption during data transfer. For example, poor wireless signal strength can negatively affect the device power saving [219, 114]. Moreover, due to poor signal propagation, mmWave's signal strength are known to fluctuate frequently and wildly due to impact of UE-side factors such as mobility or signal reflection characteristics of the surroundings (*e.g.,* open space vs concrete buildings) [181].

We conduct in-the-wild data transfer experiments to collect network throughput and power traces at two locations with Verizon 5G: **(1)** Ann Arbor, MI: mmWave 5G only, **(2)** Minneapolis, MN: both mmWave and low-band 5G. Figure 4.13 summarizes how power can be affected by both RSRP and throughput. From the results, we find that **(1)** higher throughput leads to higher power consumption; **(2)** Signal strength also affects the power consumption, which aligns well with earlier findings (§4.4.3) and previous work [114]. To better isolate the impact of signal strength and understand how it affects power consumption, we show the energy efficiency for different signal strength (RSRP) levels in Figure 4.14. As NR-SS-RSRP increases, the energy per bit decreases. This indicates that better signal strength leads to improved energy efficiency. Moreover, from the Minneapolis results in Figure 4.13, we can clearly see there are two clusters of data points. By looking at the network status information, we further confirm that the points in the upper-left cluster represent the data collected when the device is connected to low-band 5G while the other points are for mmWave 5G. In Ann Arbor, we only see mmWave 5G in the logs. Hence, we quantitatively observe that the power consumption varies across different 5G bands that the device is actively using.

4.4.5 5G Power Model Construction

Previous studies either only consider downlink/uplink throughput [138] or signal strength [105, 188] when modeling the device power for data transfer. However, neither of the assumptions hold given the high variability of 5G throughput in particular for downlink and the vulnerability of 5G signal to the physical environment. Besides, we have seen different bands can have varied power consumption characteristics, hence, it is also important

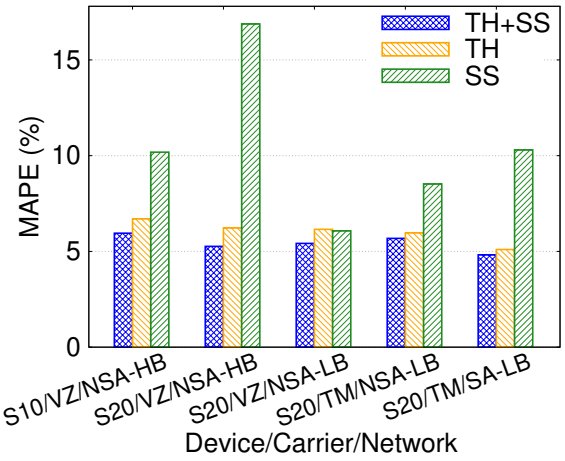


Figure 4.15: Comparing performance of different models.

to take into account the band information. To improve model accuracy, we propose to build a network power model for 5G by considering both signal strength and throughput. Based on the observations in §4.4.3, a linear model can fit well for both uplink and downlink if we solely consider throughput while controlling other factors. However, our preliminary experiments show that linearly regressing with multiple factors such as throughput and signal strength together on our walking dataset leads to even higher errors compared to only considering throughput, indicating that the diverse array of multiple impacting factors may not be accurately fit linearly, we instead turn to machine learning-based data-driven approaches to identify the relationships among features for power modeling. Specifically, we apply the Decision Tree Regression (DTR) algorithm.

Model Evaluation. We construct our models and evaluate using a standard metric for regression performance – *Mean Absolute Percentage Error (MAPE)* to reflect the accuracy of our model in terms of relative errors. As observed in §4.4.4, we construct the power model for different devices (S10, S20U), networks (Verizon, T-Mobile), and radio

technologies (NSA/SA, mmWave/low-bands) separately. Note we build models for each setting as opposed to using such information as a feature in the model. We also generate models using previous approaches for comparison. We plot the performance results for all the models in Figure 4.15, in which **TH+SS** represents our model which takes into account both throughput and signal strength while **TH** and **SS** represent the models generated only considering throughput or signal strength, respectively. Our models always outperform the models generated from both the previous approaches, which indicates that both features play an important role in affecting the device network power consumption. Without considering the throughput information, the errors of **SS** models are found to be huge compared to **TH+SS**, especially for mmWave (high-band, **HB**) which can deliver ultra-high bandwidth. For example, using S20U, Verizon's mmWave 5G service can provide up to 3 Gbps (see in §4.3.3). S10 achieves around 2 Gbps over Verizon mmWave 5G (similar to PX5). This highlights the importance of throughput information for the power model construction, especially for mmWave-based networks. Note that there are performance differences between the models constructed using data from different devices (*e.g.,* between first two models). Not surprisingly, this signifies that different devices have different hardware specs (*e.g.,* chipset lithography) that impact power consumption.

Validation on Real Applications. Finally, we evaluate the accuracy of our power model by running two real-world applications: **(1)** video streaming over YouTube app; **(2)** web browsing over Google Chrome Browser app. For video experiments, we play a video [18] at 2K resolution, in both online mode (over cellular radio) and offline mode (downloaded to SD card). To get the network energy consumption, we subtract the total offline energy which contains energy consumed by decoding and rendering of video from

the total energy measured when running online. Similarly for web experiments, we download the whole website to SD card and open the locally stored homepage (`.html`) file on Chrome to load the website in offline mode and then compare the same when loaded in online mode. We compare the energy consumption estimated by our model with the actual energy consumption measured by Monsoon power monitor. The average relative errors are 3.7% for video streaming and 2.1% for web browsing.

4.4.6 Software Power Monitor Calibration

Although hardware power monitors such as Monsoon [59] provides highly accurate power readings of mobile devices by directly supplying power to them, it will be extremely inconvenient for users to retrieves such information in daily use. In particular, it requires non-trivial hardware engineering efforts on current COTS smartphones (*e.g.,* remove the non-removable back cover and battery). Android exposes battery status such as current (`/sys/class/power_supply/battery/current_now`) and voltage (`/sys/class/power_supply/battery/voltage_now`) which can be used to measure the device power. Thus, besides different impacting factors for power model construction, we also study the accuracy of battery status (current, voltage) readings and whether it can be calibrated and further used to report the device power.

We first benchmark the software-based power monitor with different activities including **(1)** randomly tapping on the screen and opening/closing applications, **(2)** leaving the UE idle with the screen on/off, **(3)** performing UDP download at different speeds, and **(4)** running a video playback. We collect the battery status using both software (API) and hardware (Monsoon) approaches and calculate the average relative errors between the

Table 4.5: Benchmarking results on different test cases.

Test Case	**Relative error = SW / HW**	
	@ 1Hz	*@ 10Hz*
Random activities	84.2%	94.3%
Idle (screen on)	87.9%	93.7%
Idle (screen off)	80.9%	94.9%
UDP DL 50Mbps	87.1%	91.5%
UDP DL 400Mbps	87.4%	89.7%
UDP DL 800Mbps	87.5%	91.3%
UDP DL 1200Mbps	86.8%	91.2%
Video streaming	92.2%	92.9%

Table 4.6: A higher sampling rate incurs more overhead.

Activity	Average Power (mW)
Idle	2014.3
Monitor on (1Hz)	2668.5
Monitor on (10Hz)	3125.7

two approaches. As shown in Table 4.5, the software monitor always underestimates the UE power. A higher sampling rate may help provide better estimation, but this will incur higher energy overhead (Table 4.6).

Next, we use DTR to calibrate the software power values. Figure 4.16 shows the calibration performance (**SW**) together with our **TH+SS** model results. After calibration, the software-based approach can achieve comparable performance. A higher sampling rate (*e.g.,* 10Hz) can even lead to better performance (*i.e.,* lower `MAPE`). However, we argue that a higher sampling rate will incur higher overhead which is less energy-efficient.

To summarize, we empirically characterize several impacting factors such as signal strength and throughput over power consumption by smartphones using 5G services. We propose an ML-based data-driven approach to construct power models for 5G networks.

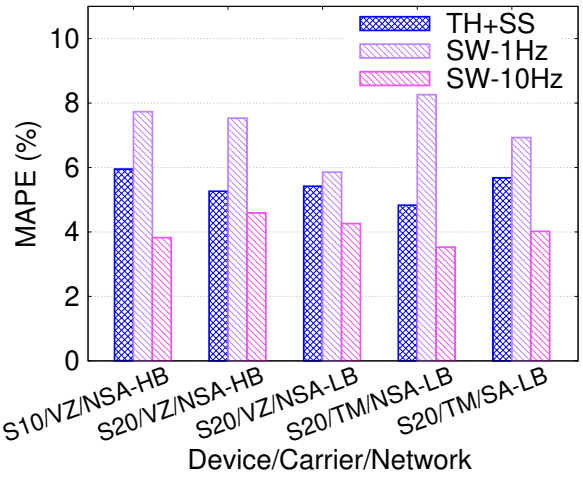


Figure 4.16: Software power monitor calibration.

We demonstrate that our models help increase accuracy in predicting device power consumption. We show that the software power monitor can achieve comparable accuracy after calibration.

Next, we take a closer look at two popular mobile applications, video streaming and web browsing, both combined are expected to cover more than 80% of the mobile traffic share by 2025 [31]. We look at them from the perspective of both application QoE and energy efficiency. We believe the proposed power models can be useful to aid developers in making their application more energy-efficient. For the following sections, we focus on mmWave 5G which are considered key to mainstream 5G and have not been studied before in the context of mobile applications.

4.5 Video Streaming over 5G

Adaptive bitrate (ABR) algorithms are the primary tools used to optimize video quality of experience (QoE). The research community has proposed a plethora of ABR algorithms

for video streaming in recent years [174, 249, 246, 148, 149]. In this section, we demystify 5G's implication on video streaming by conducting the first in-depth investigation of ABR streaming QoE over 5G with mmWave. We aim to answer the following questions:

- What is the performance footprint of the current state-of-the-art ABR algorithms under 5G and how does it compare with 4G?
- What are the major factors that impact ABR streaming performance over 5G?
- What new mechanisms are needed to make future ABR algorithms *5G-aware* and further improve the QoE?

4.5.1 Evaluation Methodology

Our testbed consists of an Apache server hosting the videos and a DASH.js [46] video client. We use trace-driven emulation to ensure that all algorithms experience the same set of network conditions. We use the Lumos5G dataset [181] which contains 121 5G and 175 4G throughput traces, collected at 1-second granularity. We focus on traces collected with mmWave coverage as 5G's high-band frequency range is considered key to support UHD and beyond video streaming [205]. We use a custom 4K video [17] and encode it using `FFmpeg` [24] with `libx264` into 6 tracks (or qualities) with different bitrates. 4K (or even 16K) video streaming usually requires 25-120 Mbps (246-328 Mbps) bandwidth [65, 40, 60, 256] which can be easily met by 5G. Thus, to identify rate adaptation challenges in 5G which has a mean throughput value that is 10$\times$ of 4G, we scale the video bitrate of 5G tracks used to match its throughput range. This ensures avoiding any trivial bitrate selection. We set the bitrate of the top track (*i.e.,* highest video quality) to match the

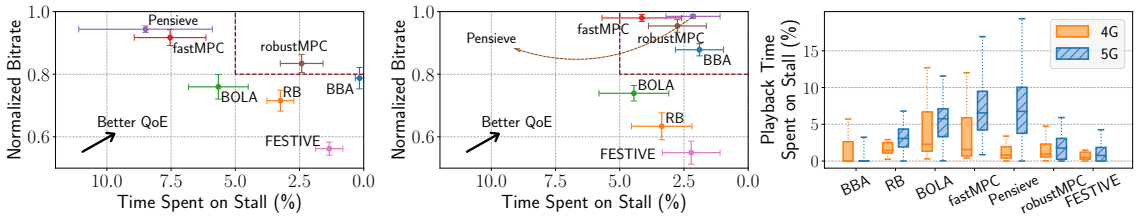
median throughput of 5G/4G network traces. In this study, the maximum bitrate track for 5G is 160 Mbps, and 20 Mbps for 4G. We then decide the bitrates for lower-quality tracks by keeping the encoded bitrate ratio as $\sim$1.5 [198] between two adjacent tracks. Note, our goal here is not to understand whether video streaming is better over 5G or 4G. Rather, we focus on studying whether existing ABR algorithms can work well over mmWave 5G. Using the throughput traces, we use Linux `tc` on the client side and control the instantaneous bandwidth. For showing results, we normalize the video bitrates by the bitrate of the top track.

We study the following 7 state-of-the-art ABR algorithms covering 4 different categories. **(1)** *Buffer-based:* BBA [140] and BOLA [227] make bitrate decisions based on the buffer occupancy. **(2)** *Throughput-based:* simple rate-based (RB) and FESTIVE [148] use information of past chunks to estimate future throughput and decide the bitrate of the next chunk to download. **(3)** *Control theoretic:* FastMPC and RobustMPC [249] make bitrate decisions by solving an optimization problem of the QoE for the next n chunks (*e.g.,* $n = 5$). **(4)** *Machine learning-based:* Pensieve [174] adopts a deep neural-network to learn bitrate decisions that maximize a QoE reward[6].

4.5.2 Performance of Existing ABR Schemes

Overall, we find that multiple ABR algorithms that work well under 4G do not maintain the high performance under 5G. Figure 4.17 summarizes the bitrate and the video stall time for different ABR algorithms. The top-right rectangular region marked using maroon-

[6]We show the results of the Pensieve model trained with real Lumos5G [181] network traces. We also verify that the performance observed by using models trained with synthetic traces (as suggested in their paper [174]) and Lumos5G traces are similar.



(a) Two dimensional QoE for 5G. (b) Two dimensional QoE for 4G. (c) 4G & 5G video stall time.

Figure 4.17: QoE of different ABR algorithms in 4G/5G and a comparison of video stall.

colored dashed lines represents ABR algorithms with better QoE. Here, better QoE refers to ABR algorithms that achieve less than 5% video stall and over 0.8 normalized bitrate across different traces. For 5G, only one algorithm (robustMPC) provides better QoE while for 4G there are 3 more algorithms.

Although most of the ABR algorithms under 5G can achieve similar normalized bitrates as they are in 4G (*i.e.,* similar Y-axis values in Figure 4.17a and Figure 4.17b, with an average drop of only 3.5%), the concerning problem for video streaming over 5G lies in the video stalls. For RB, BOLA, MPC, and Pensieve, we observe a significant increase (58.2% on average) of video stall. Figure 4.17c shows that except for BBA all other ABR algorithms suffer an increase in video stalls when running over 5G. For instance, the mean video stall time for fastMPC and Pensieve has increased by 82.0% and 259.5%, respectively. Pensieve outperforms all other algorithms in 4G but incurs the highest video stall time under 5G setting. Since Pensieve makes bitrate selection to optimize its QoE reward, we also compare its QoE reward with that of fastMPC and robustMPC. Pensieve's QoE reward improvement is also marginal compared to other algorithms (0.66% improvement over fastMPC and 5.93% over robustMPC), which is 3× lower than the results in the original Pensieve paper. A possible explanation is that for 5G networks, a larger dataset

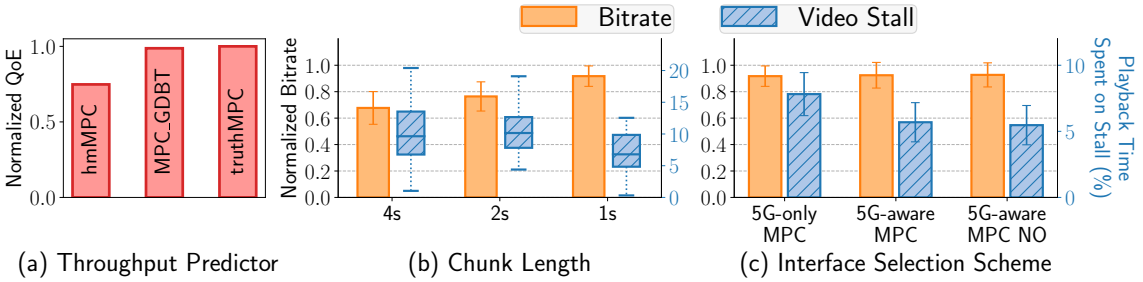


Figure 4.18: QoE impact of: (a) throughput predictors (b) chunk length, and (c) interface selection schemes.

is needed for training the model to learn 5G specific characteristics and make better decisions, which deserves further study. After taking a closer look at the bitrate decisions taken by Pensieve and fastMPC, we find that they sometimes choose the highest bitrate chunk only to regret that it was a wrong decision that is difficult to undo, resulting in a very high stall time. This is not happening in 4G scenarios with the same optimization metric used. Based on this phenomenon, next we dig further to quantitatively understand the challenges involved in running ABR algorithms for video streaming over 5G networks (§4.5.3).

4.5.3 Challenges in ABR Streaming under 5G

Throughput prediction. Many ABR algorithms incorporate network throughput into its decision by leveraging a throughput predictor and their performance heavily depends on prediction accuracy. To study the impact of throughput prediction on 5G video streaming, we fix other parts in an ABR algorithm and plug in different throughput predictors and compare the incurred QoE. Considered as one of the state-of-the-art ABR algorithms, we choose fastMPC as the baseline since it explicitly incorporates a throughput predic-

tor while Fugu [246] and Pensieve use throughput information implicitly. We compare three different throughput predictors: **(1)** *hmMPC*: the original throughput predictor used by fastMPC uses harmonic mean of past throughput values to predict future throughput, **(2)** *MPC_GDBT*: a state-of-the-art mmWave 5G-specific throughput predictor [181] that adopts a ML based approach called Gradient Boosted Decision Tree (GDBT), and **(3)** *truthMPC*: ground-truth throughput trace to represent the optimal online throughput prediction scheme. Since MPC's goal is to maximize its QoE function [249], we use the QoE function as the metric to evaluate the effectiveness of applying the 5G-specific throughput predictor. Figure 4.18a indicates that using the GDBT throughput predictor can achieve 31.98% higher normalized QoE compared to the default harmonic mean predictor. Compared to *truthMPC* though, adopting the GDBT predictor only provides 1.3% less QoE. Therefore, improving throughput prediction accuracy in ABR algorithms can significantly enhance video streaming QoE and provide opportunities to build better 5G-aware throughput-predictors. Since 5G now spans across many different bands and its network performance variation is large (§4.3), building better throughput prediction schemes is not only vital to make ABRs work well over 5G but also to improve our understanding of the 5G ecosystem in general.

Decision making granularity. An ABR algorithm's decisions are coarse-grained in that it has to do chunk selection on chunk boundaries, and once made, such decisions cannot be rolled back. Specifically in our 5G video streaming results, we find that just one or two bad chunk selections can significantly affect QoE of the entire stream. This one chunk download decision indeed quickly drains the playback buffer or even causes 5–10 seconds of rebuffering. One fix is to reduce the video chunk length to support fine-grained selec-

Table 4.7: Energy consumption for different interface selection schemes.

Interface selection scheme	Energy (J)
5G-only MPC	495.0±55.1
5G-aware MPC	474.4±59.1
5G-aware MPC (No Overhead)	475.0±58.9

tions. We study the effect of different chunk lengths (1/2/4s) on 5G video streaming (with fastMPC). Figure 4.18b shows that using 1s chunks provides 21.5% (35.9%) higher bitrate and 33.6% (29.8%) less video stalls compared to 2s (4s) chunks. Therefore, although 2s and 4s chunks are typically suggested for ABR [14], we argue video content providers should consider shorter length chunks (*e.g.,* 1s) so that ABR algorithms can make finer-grained decisions and adapt better to the highly fluctuating 5G network conditions.

4.5.4 Improving 5G ABR Streaming

Based on our observation that 5G consumes more power than 4G when the throughput is low (§4.4) and 5G throughput fluctuates a lot, we propose **5G-aware video streaming**. The idea is switch to 4G when ABR algorithms predict that 5G throughput is low (*i.e.,* ¡4G's average throughput), given that 4G provides relatively stable bandwidth, and switch back to 5G when the video buffer level has reached over some threshold (empirically set to 10s). We also take into account the switching overhead between 4G and 5G (§4.4) and emulate the switching delay using Linux `tc`. Similarly, we use fastMPC as the baseline ABR algorithm. Figure 4.18c depicts that our selection scheme (denoted as *5G-aware MPC*) can reduce video stall time by 26.9% compared to always using 5G interface during the entire video. Compared to the 5G-aware MPC with no overhead version (where we remove the interface switch delay, assuming the UE can instantly switch between 4G and

5G), our realistic interface selection model only incurs 4.0% more stall time. Table 4.7 shows the corresponding energy consumption, measured by feeding the collected video packet traces into our 5G power model (§4.4). As shown, the proposed 5G-aware schemes consumes 4.2% less energy than always using 5G. It's also slightly *"greener"* than the no overhead version by trading a little bit of video quality: downloading higher quality chunks and consuming more energy. Figure 4.18c and Table 4.7 conclude that carefully selecting between 4G and 5G interfaces can both improve adaptive video streaming performance (26.9% fewer stalls) as well as reduce the energy consumption (by 4.2%, comparative to the 4.7% saving achieved in [242]).

4.6 QoE Implications of Web Browsing over mmWave 5G

Previous sections have shown that mmWave 5G is able to provide ultra-high throughput but requires more power to deliver this performance. On the other hand, low-band 5G or LTE uses much less power but delivers lower performance than mmWave. Hence, there is a trade-off between achieving high performance and energy efficiency. To get better insights about this trade-off, in this section we use web browsing as a case study to understand the QoE implications of radio type (*e.g.,* 4G or mmWave 5G) used to load websites in-the-wild.

Data Collection Methodology. Using chrome-har-capturer [44], we build scripts to instrument and load Alexa's top 1500 websites via the Chrome Browser app. For each website, we collect HTTP Archive (*i.e.,* HAR [9]) files as well as capture the packet traces. Since packet capturing requires root permission, we used PX5.We conduct this experiment under stationary conditions in two radio settings: **(i)** mmWave 5G is active, **(ii)** 4G/LTE is

Table 4.8: Factors considered for analyzing their impact on page load time and energy consumption.

Factor	Abbr
# of dynamic/total objs	DNO
Size of dynamic objs / total page size (in bytes)	DSO
# of objects	NO

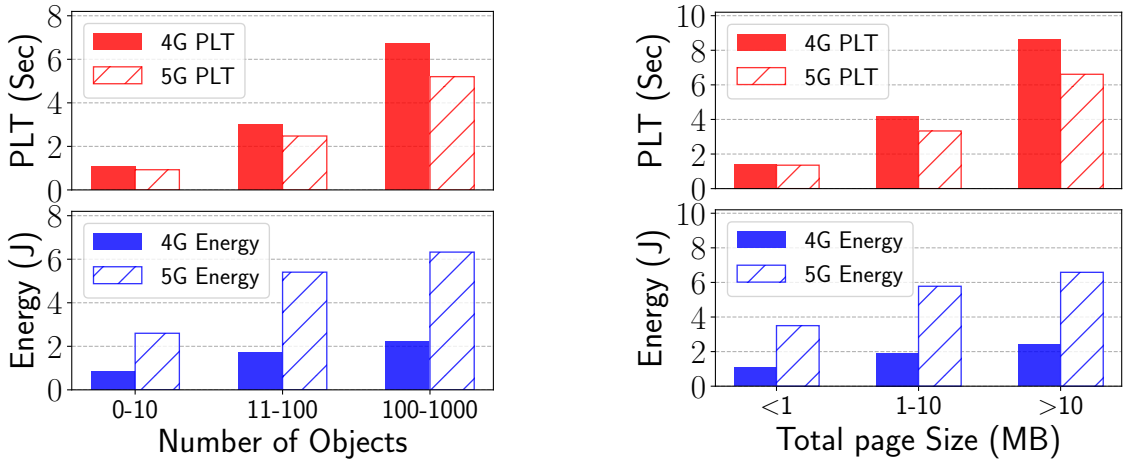
Factor	Abbr
# of images (videos)	NI (NV)
Total Page Size	PS
Avg. Object Size	AOS

active. mmWave-based experiments were conducted with UE having LoS to 5G tower. We repeat the experiment at least 8 times per device per radio type. To eliminate the impact of browser cache, we clear the cache before loading the next website.

The HAR file of each website loading provides us the total page load time (PLT), time to fetch each individual object (*e.g.,* images, `.css` or `.js` files) associated with the website, *etc.* We also extract the per-second throughput trace observed in the packet dumps. This trace is then fed to our power model proposed in §4.4 to estimate the radio's energy consumption for loading the website. All references to 5G in this section refer to Verizon mmWave 5G service.

4.6.1 When does mmWave 5G help?

We list several factors (see Table 4.8 for the entire list) that might potentially affect PLT performance and/or energy utilization. For each radio type, Figure 4.19 compares their empirical impact for a subset of these factors on the two QoE metrics, performance and energy consumption. We have the following key observations: **(i)** As the number of objects contained in a website increases, the PLT performance gap between 4G and 5G increases with 4G being on the poor side. Similar observations are made for other factors such as total page size and number of dynamic objects. **(ii)** On the other hand,



(a) Impact of # of objects.

(b) Impact of total page size.

Figure 4.19: Understanding how different factors affect the page load times under mmWave 5G or 4G setting.

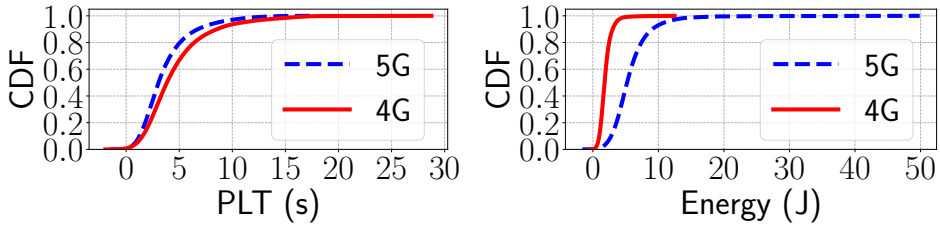


Figure 4.20: CDF for PLT and energy.

the implications of the same factors have an opposite effect when seen under the purview of energy consumption where 4G consumes far less energy than 5G. The CDF plots in Figure 4.20 show these differences more clearly. We find that due to the high throughput offered by mmWave 5G, PLT performance in 5G is always better than 4G. However, as demonstrated earlier in §4.4, when applications are not bandwidth-hungry (*e.g.,* normal web browsing), the energy utilization of 4G is better than that of mmWave 5G.

While the importance of performance and energy utilization can differ based on the us-

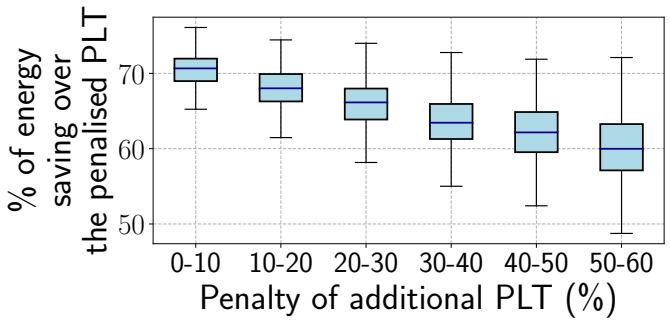


Figure 4.21: 4G's PLT penalty and energy saving over 5G.

age context, we normalize both metrics for fair comparison. Figure 4.21 shows that even a 10% penalty over PLT incurred for choosing 4G over 5G can reduce energy consumption by almost 70%. While such a high level of savings diminish as the PLT penalty grows, the important takeaway here is that the slightest *permissible* penalty in PLT (caused by choosing 4G) leads to high energy savings. Understanding where such a permissible penalty might lie depends on how much additional delay in PLT is permissible such that there is no significant impact on user experience. For example, a PLT of 2s or less remains a widely considered golden standard [196] for web page load times. An average 4G throughput of say 60 Mbps can theoretically load a website with a total page size of $15\ MB$ in $<$2s and might potentially save energy without significantly affecting QoE and/or bounce rate [196] which is the percentage of visitors that leave a page without taking an action.

4.6.2 Interface Selection for Web Browsing

Using all the above insights, we next propose a simple yet insightful model generation algorithm that takes into account all the factors listed in Table 4.8 to decide whether to use the 4G or 5G radio interface for loading a website. To help make this decision, we

come up with a simple linear utility function: QoE $= (\alpha \times EC) + (\beta \times PLT)$ that allows us to tune the weights α and β for the two competing QoE metrics - energy consumption (EC) and page load times (PLT), respectively. To make the generated model insightful, it will be useful to know what factors (from Table 4.8) of a website makes a model choose a particular radio interface over another.

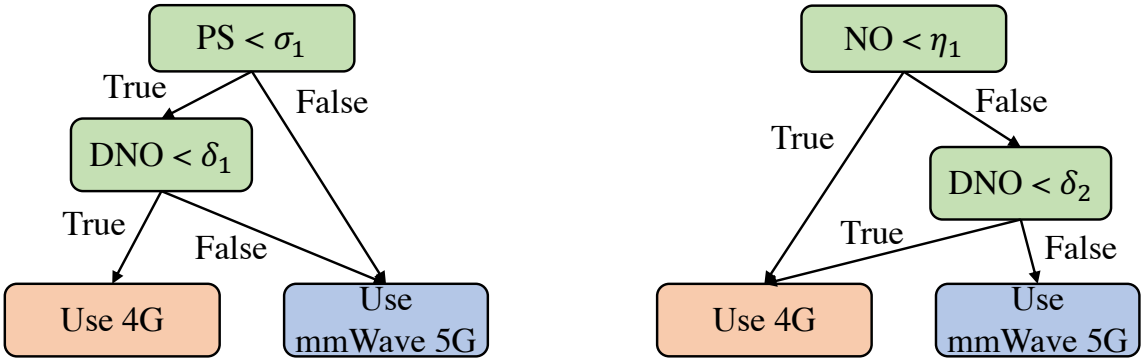
For this case study, we choose *Decision Tree* (DT) learning algorithm for two reasons. First, DT is easy to run as it does not require any massive computational power. Secondly, it provides indices (*e.g.,* Gini index) for each of the features included in the input feature vector making it easily interpretable. Both these benefits can potentially help application/service developers to not only get insights on improving and achieving their designed QoE but also enable them to account for the usage context and quickly build more models for achieving different QoE goals.

Model Setup. We randomly split our dataset using a ratio of 7:3 such that 70% is used for training and validation and the rest is used for testing. With over 30K data points, the time to generate the model was less than a minute on a general-purpose laptop.

Table 4.9: DT's radio interface selection results.

#ID	**Desired QoE**	α	β	**Use 4G**	**Use 5G**
M1	High Performance	0.2	0.8	19	401
M2	Performance Oriented	0.4	0.6	366	54
M3	Balanced	0.5	0.5	387	33
M4	Better Energy Saving	0.6	0.4	405	15
M5	High Energy Saving	0.8	0.2	420	0

Results. Table 4.9 shows the results of different models' radio interface selection results over the 420 websites in the test set. Figure 4.22 shows the bottom-up post-pruned



(a) M1 (σ_1=4.6MB, δ_1=0.2). (b) M4 (η_1=411, δ_2=0.76).

Figure 4.22: High-Performance (M1) vs. Energy-Saving (M4) models.

DT for models M1 and M4. When performance matters (M1), we find that two factors are important in deciding the radio type: **(1)** the total page size in bytes, and **(2)** the proportion of dynamic vs. static objects (*e.g.,* ads vs. logos) In contrast, when energy utilization is preferred (M4), 4G radio can handle more websites while 5G will be the preferred radio when the website has an extremely high number of dynamic objects (¿76%) compared to static objects. By feeding the web packet traces into our constructed power model (§4.4), we find that interface selection help save 15-66% energy while improving the overall QoE. The dynamic 4G/5G switching scheme proposed in [242] brings a 25% saving on energy but does not consider the page load time.

4.7 Summary

Leveraging a custom measurement platform, we have conducted comprehensive measurements of several key aspects of commercial 5G: end-to-end network performance, power characteristics, 4G/5G interaction, and application QoE. Our findings reveal the

state-of-the-art landscape of the 5G ecosystem, in particular the higher protocol stack. We have released our datasets and measurement tools to the research community.

CHAPTER V

QUIC is not Quick Enough over Fast Internet

QUIC is expected to be a game-changer in improving web application performance. In this chapter, we conduct a systematic examination of QUIC's performance over high-speed networks. We find that over fast Internet, the UDP+QUIC+HTTP/3 stack suffers a data rate reduction of up to 45.2% compared to the TCP+TLS+HTTP/2 counterpart. Moreover, the performance gap between QUIC and HTTP/2 grows as the underlying bandwidth increases. We observe this issue on lightweight data transfer clients and major web browsers (Chrome, Edge, Firefox, Opera), on different hosts (desktop, mobile), and over diverse networks (wired broadband, cellular). It affects not only file transfers, but also various applications such as video streaming and web browsing. We conduct root cause analysis through rigorous packet trace analysis and kernel- and user-space profiling, make concrete recommendations for mitigating the observed performance issues.

5.1 Introduction

QUIC is a multiplexed transport-layer protocol over UDP, poised to be a foundational pillar of the next-generation Web infrastructures. It has recently been standardized by the IETF (known as *IETF QUIC* [145]) as the transport foundation of HTTP/3 [96]. Since 2013, QUIC has been commercially deployed by numerous companies including Google, Akamai, Meta, and Cloudflare [11, 161, 52, 32, 70]. As its adoption continues to grow rapidly, QUIC (together with HTTP/3) is standing at the forefront to reshape the performance paradigm of the World Wide Web. There is a plethora of literature on characterizing QUIC performance [150, 197, 213, 239, 175, 250, 122, 224]. They have used various QUIC implementations (customized vs. commercial), compute environments (mobile vs. desktop), and network conditions (wired vs. wireless). Due to such diversity, their findings are understandably a mixture of performance gains, and in some cases, degradations, compared to TCP and earlier generations of HTTP. In addition, a majority of these studies focus on low-throughput use cases.

In this study, we systematically examine an under-explored scenario: running QUIC over high-speed networks. This scenario is becoming increasingly important with the debut of faster networks such as high-speed wired links, WiFi 6/7, and 5G, which often reach more than 500 Mbps and up to 1+ Gbps per connection. Meanwhile, given the ubiquity of HTTP on today's Internet, HTTP (QUIC) is being utilized for bandwidth-intensive applications like ultra-high-resolution videos [193] and VR/AR [248]. This makes understanding QUIC's performance on high-speed networks even more crucial.

QUIC is Slow over Fast Internet. Despite typically being referred to as a transport-layer protocol, QUIC is deeply coupled with upper-layer components, namely TLS and

HTTP. Its user-space nature makes such coupling more complex and extensive. Note that an apple-to-apple comparison should be done on the UDP+QUIC+HTTP/3 protocol stack and TCP+TLS+HTTP/2 stack. For brevity, we refer to the two stacks as **QUIC** and **HTTP/2**. We begin with comparing QUIC and HTTP/2 in a simple environment: file download using a command-line data transfer tool, `cURL` [7], and a Chromium-based client, `quic_client` [79]. For a fair comparison, we keep factors such as the congestion control algorithm, server configuration, and network condition the same. The results show that QUIC and HTTP/2 exhibit similar performance when the network bandwidth is relatively low (below $\sim$600 Mbps), whereas under a higher network bandwidth, QUIC consistently lags behind HTTP/2 by up to 15.7% in terms of throughput. The performance gap becomes more pronounced as the bandwidth increases. Notably, during packet reception, QUIC incurs considerably higher CPU usage than HTTP/2 on state-of-the-art client hosts.

Next, we investigate more realistic scenarios by conducting the same file download experiments on major browsers: Chrome, Edge, Firefox, and Opera. We observe that the performance gap is even larger than that in the `cURL` and `quic_client` experiments: on Chrome, QUIC begins to fall behind when the bandwidth exceeds $\sim$500 Mbps. When the bandwidth reaches 1 Gbps, QUIC becomes 45.2% slower than HTTP/2. On weaker clients such as mobile devices, the gap is even larger.

QUIC's Slowness Impacts Multiple Web Applications. We experimentally demonstrate that QUIC's performance degradation affects not only bulk file transfers but also other applications including video content delivery and web browsing, despite their intermittent traffic patterns. QUIC incurs a video bitrate reduction of up to 9.8% compared

to HTTP/2 when delivering DASH [226] video chunks over high-speed Ethernet and 5G. Again, such QoE degradation only exhibits when the underlying bandwidth is sufficiently high. For example, the impact is hidden over 4G but unleashed over 5G. QUIC's page load time (PLT) is 3.0% longer than HTTP/2's, averaged across 100 representative websites, with a long tail of page load time gaps over 50%.

QUIC's Slowness over Fast Internet is due to Receiver-side Processing. With the above results, we then identify the primary culprit of the QUIC-HTTP/2 performance gap. This is a highly challenging task due to a wide range of factors in the Web ecosystem, the high complexity of QUIC, and various engineering difficulties. We first make two observations by looking into packet traces and performance data: (1) The client running QUIC receives a much higher number of packets compared to those during HTTP/2 downloads; (2) There is a high delay between incoming data packets and their corresponding ACK packets when QUIC receives at a high data rate, suggesting that it takes longer to process QUIC packets. Both observations indicate that the slow performance of QUIC over fast Internet is due to limited receiver-side processing capability. It is important to note that although QUIC's user-space implementation is known to cause performance degradation in general [161] and there have been efforts to optimize UDP/QUIC's sender-side transmission performance [110, 28, 142], we are *the first to identify the receiver side as a more likely performance bottleneck for QUIC over fast Internet*. This is not only because servers are typically more powerful than clients (desktops, laptops, mobile phones), but also attributed to unique challenges in handling data reception per QUIC's design, as detailed next.

The Poor Receiver-side Performance is due to Excessive Data Packets and User-

space ACKs. We conduct deep performance profiling on the user-space Chromium (open-sourced version of Chrome) and the underlying OS networking stack. We identify two main root causes of QUIC's poor receiver-side performance.

- **Issue 1.** When downloading the same file, the in-kernel UDP stack issues much more packet reads (`netif_receive_skb`) than TCP, leading to a significantly higher CPU usage. This is because none of the QUIC implementations we examine uses UDP generic receive offload (GRO) where the link layer module combines multiple received UDP datagrams into a mega datagram before passing it to the transport layer. This is in sharp contrast to the wide deployment of TCP segmentation offload, and recent advocacy of UDP send-side offload (GSO).
- **Issue 2.** In the user space, QUIC incurs a higher overhead when processing received packets and generating responses. This can be attributed to multiple factors: excessive packets passed from the kernel (Issue 1), user-space nature of QUIC ACKs, and lack of certain optimizations such as delayed ACK in QUIC.

Recommendations for Mitigation. We make several recommendations for mitigating the above impact, including deploying UDP GRO on the receiver side, making generic offloading solutions (GSO and GRO) more QUIC-friendly, improving relevant QUIC logic on the receiver side, and using multiple CPU cores to receive data for QUIC. We also discuss some practical challenges of realizing the above recommendations, such as the heterogeneity of today's commodity client hosts (PCs, mobile devices, and embedded devices, with diverse OSes) compared to the servers.

At a high level, we advocate careful examinations of upper-layer protocols over emerging networks, applications, and services. This work instantiates this idea by conducting

Table 5.1: Preliminary file download tests.

Testbed	Download Time (s)		CPU Usage (%)	
	HTTP/2	HTTP/3	HTTP/2	HTTP/3
Desktop, Ethernet	9.32	18.60 (+99%)	77.5	96.9
Pixel 5, low-band 5G	37.11	78.65 (+112%)	121.55	161.77
Pixel 5, mmWave 5G	30.10	63.20 (+110%)	128.43	165.20

a pioneering study on QUIC performance over fast Internet. We make two-fold contributions: the measurement findings and the root cause analysis. We have released all the measurement data and source code associated with this study [82].

5.2 Motivation

QUIC is a user-space transport protocol over UDP, designed to provide fast, reliable, and secure connections. It offers benefits such as 0/1-RTT fast handshake, stream multiplexing to remove head-of-line blocking, and connection migration. However, QUIC also has potential downsides, such as the overhead of processing and copying data between the kernel space and user space.

Downloading data over QUIC can become very slow in particular given the emergence of high-speed Internet. We conduct a preliminary experiment on both desktop and mobile Chrome browsers to download 1 GB files (see §5.3.1 for details). Table 5.1 presents the results averaged over 10 runs. We can find that, the file download time when QUIC is enabled is around double the time with QUIC disabled. The CPU usage is also higher during QUIC download. The performance disparity between QUIC and HTTP/2 is even larger on smartphones. Note that the CPU usage for the desktop is measured from the browser's network service while the measurement refers to the CPU usage of the entire browser pro-

cess for the smartphone. CPU usage exceeding 100% indicates that the browser process was utilizing more than one cores in a multi-core system.

The results raise a couple of questions: When is QUIC data transfer slower than HTTP/2? What are the underlying reasons for the performance gap? Can users benefit from the current deployment of QUIC? To answer these questions, we carry out an in-depth measurement study on QUIC performance over high-speed networks.

5.3 QUIC Transport Performance

In this section, we conduct a series of experiments comparing the performance of QUIC and HTTP/2. We start with introducing our experimental methodologies in §5.3.1. Then, we present file download experiments on lightweight data transfer clients in §5.3.2. Finally, we discuss the results on commercial web browsers in §5.3.3.

5.3.1 Methodology

Various factors within different components in the network can affect the overall performance and potentially become the bottleneck. When comparing the UDP+QUIC+HTTP/3 (QUIC) stack with the TCP+TLS+HTTP/2 (HTTP/2) stack, we carefully set up the following testbed to ensure a fair comparison, that is, the observed performance gaps originate solely from the differences in the protocol themselves.

We deploy a server machine equipped with an Intel Xeon E5-2640 CPU and a client desktop featuring an Intel Core i7-6700 CPU. They are connected through a 1-Gbps Ethernet, only two hops away from each other. This setup avoids several network-related impacts such as network congestion and bandwidth throttling imposed by middleboxes which

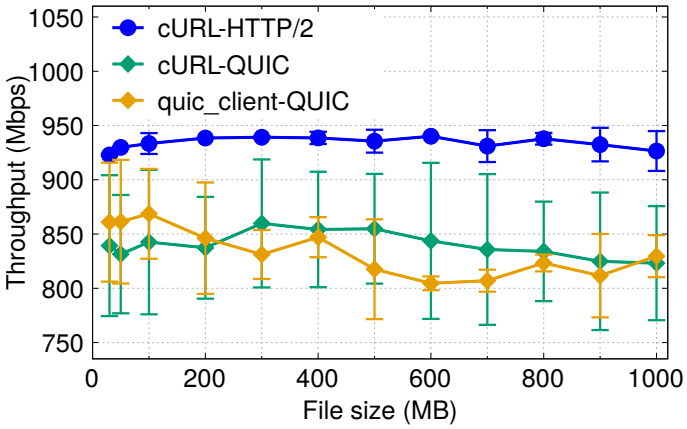


Figure 5.1: Throughput of lightweight clients during file download.

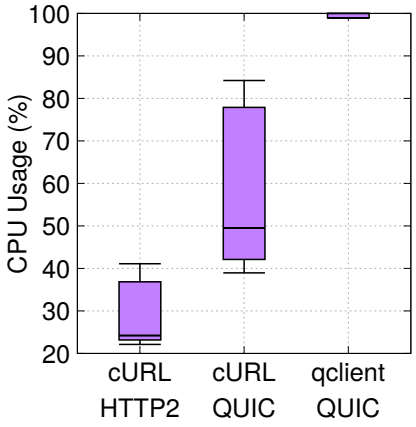


Figure 5.2: CPU usage of lightweight clients.

are often unfriendly to QUIC [161, 111, 247, 134]. Both machines run Ubuntu 18.04. We host an HTTP server using OpenLiteSpeed (v1.7.15) [76] built based on a mainstream QUIC library, LSQUIC [69]. The congestion control algorithm for QUIC is set to CUBIC, which is the default algorithm used for TCP in the OS. We also make sure their initial transport settings stay the same. Furthermore, both the UDP and TCP buffer sizes are adjusted to exceed 10x the link's bandwidth-delay product (BDP) to prevent buffer starvation during experiments. We run `tcpdump` to collect packet traces. We employ Linux `tc` [8] to control available network bandwidth when evaluating QUIC and HTTP/2 under low or changing bandwidth conditions.

5.3.2 File Download on Lightweight Clients

We start our investigation with a simplified setup, using two non-browser download tools, `cURL` [7] and `quic_client` [79]. `cURL` is a command-line data transfer tool that supports both QUIC and HTTP/2. `quic_client` is a standalone QUIC client implemen-

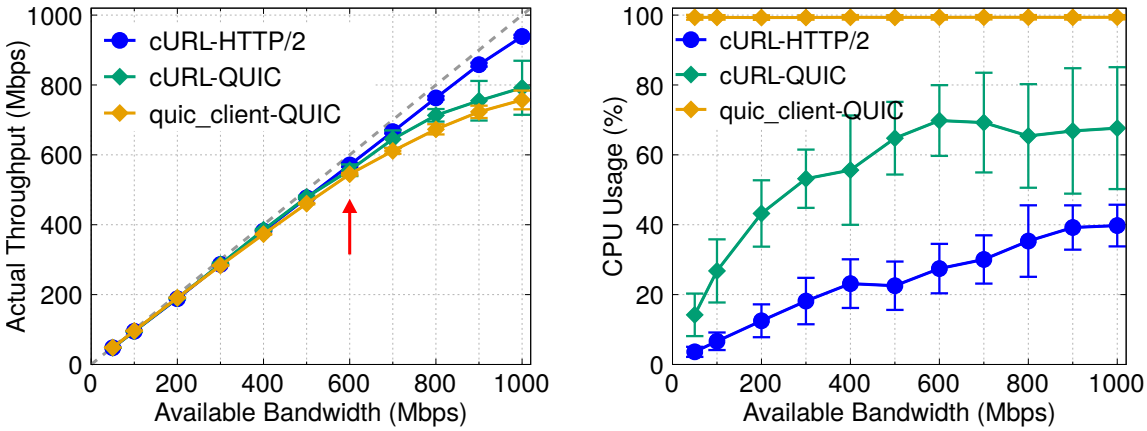


Figure 5.3: Throughput and CPU usage of `cURL` and `quic_client` during file download under limited bandwidth.

tation, built with the same QUIC stack as Chrome/Chromium.

We use the clients to download files of different sizes, ranging from 50 MB to 1 GB, over QUIC and HTTP/2. For each file size, both tools undergo 20 repeated download sessions. Figure 5.1 reports the mean values and standard deviations from the collected traces. The results show that `cURL` running HTTP/2 noticeably outperforms both QUIC clients, well utilizing the 1 Gbps available bandwidth. `quic_client`'s results stand very close to those of `cURL` on QUIC. On average, the throughput of `cURL` running QUIC and that of `quic_client` is 7-16% and 8-12% lower, respectively, compared to `cURL` with HTTP/2. Moreover, both `cURL` on QUIC and `quic_client` display an almost parallel trajectory, which indicate the similar efficiency in their QUIC implementations.

We present in Figure 5.2 the distribution of the client's CPU usage during the download of a 1 GB file. The CPU usage for `cURL` when running QUIC is higher than that of `cURL` on HTTP/2. `quic_client`'s CPU usage is further elevated, nearly maxing out at 100%, while its throughput remains similar to `cURL` on QUIC. Note that, for `quic_client`, we

have deactivated any debug mode for optimal performance (`is_debug=false`). Since it is a simplistic implementation of the QUIC protocol stack, for instance, not designed for handling multiple concurrent connections or non-transfer functionalities such as logging, it just consumes all available CPU resources during the download process without reservation, unlike `cURL` which is engineered for versatility across various scenarios.

We next limit the available network bandwidth from 50 Mbps to 1000 Mbps. As shown in Figure 5.3, when the available bandwidth is low, QUIC and HTTP/2 exhibit similar performance. Both QUIC clients can catch up with the available bandwidth, with `quic_client`'s throughput being slightly lower. However, as the bandwidth provision grows beyond around 600 Mbps, QUIC's actual throughput starts to be bottlenecked and a noticeable throughput disparity between QUIC and HTTP/2 emerges. The CPU usage for `quic_client` is always high and that of `cURL` QUIC hovers around 70%, reemphasizing the computational challenges associated with the protocol. We analyze possible performance inhibitors leading to the high CPU usage later in §5.5.

5.3.3 File Download on Real Browsers

Transitioning from lightweight clients, we look into experiments on real web browsers. This exploration mainly focuses on the well-known Chrome browser.

We repeat the file download tests on Chrome. As shown in Figure 5.4, the performance gap between QUIC and HTTP/2 is even larger than that in our prior lightweight client experiments (§5.3.2). Figure 5.5 plots the CPU usage of the network process ("Utility: Network Service" [80], responsible for network-related tasks.) during the download. It is evident that the Chrome browser running QUIC demands more computational power

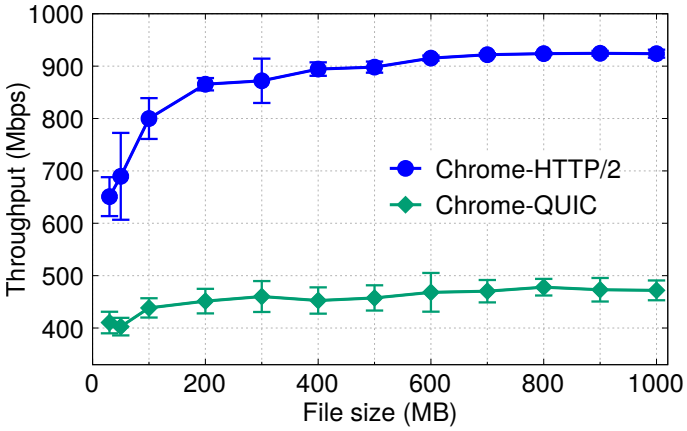


Figure 5.4: Throughput of the Chrome browser during file download.

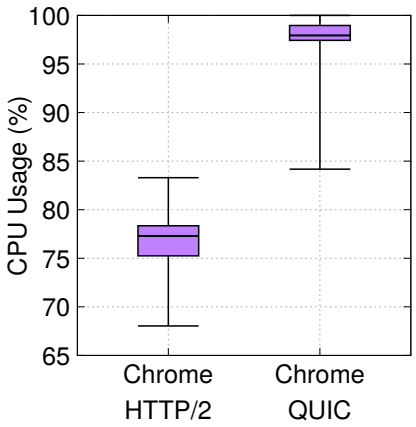


Figure 5.5: CPU usage of the Chrome browser.

than Chrome with HTTP/2. Different from the lightweight `cURL` and `quic_client`, Chrome is a full-fledged web browser, so the CPU saturation issue is exacerbated, leading to even lower QUIC performance. Remarkably, QUIC's average throughput can barely hit 478 Mbps.

The experimental results in controlled bandwidth scenarios are depicted in Figure 5.6. QUIC fails to fully utilize the bandwidth starting earlier at approximately 500 Mbps, compared to the 600 Mbps bottleneck point identified in the lightweight client tests (see Figure 5.3). Chrome with QUIC approaches 100% CPU usage when the throughput is only 200 Mbps. Recall that, with further limited compute resources, the HTTP/2-QUIC performance gap on mobile devices is more pronounced, as shown in Table 5.1.

Additionally, we run experiments of changing the CPU frequency (*i.e.,* CPU clock speed). The Intel Core i7-6700 CPU equipped on the client machine has a base frequency of 3.40 GHz and can be boosted to 4.00 GHz. In Figure 5.7, as we reduce the CPU frequency, Chrome's QUIC download throughput further drops to around 200 Mbps while

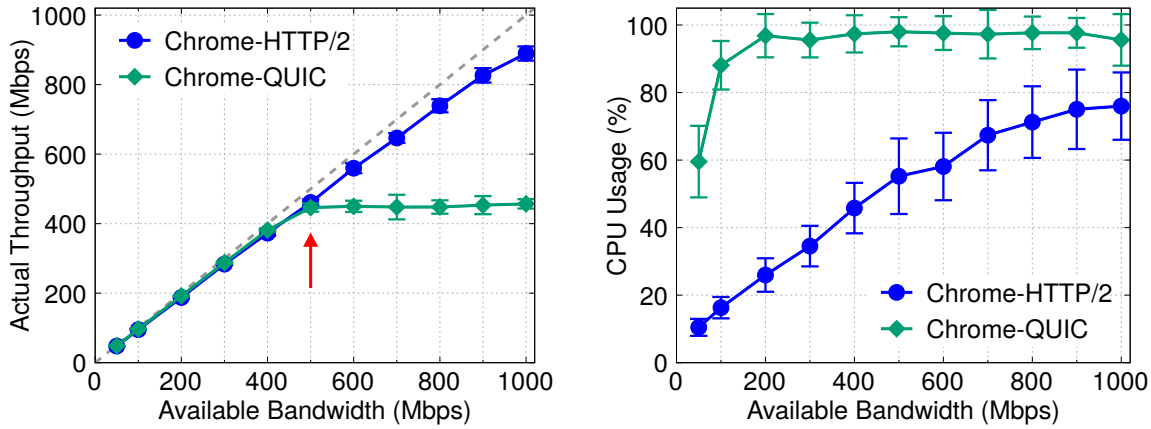


Figure 5.6: Throughput and CPU usage of the Chrome browser during file download under limited bandwidth.

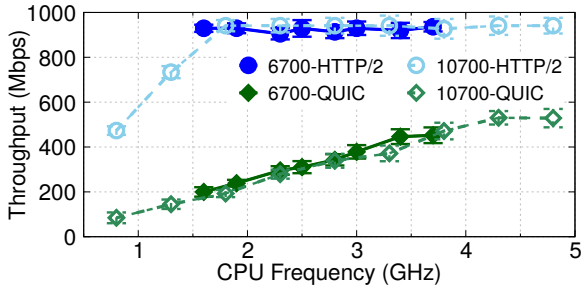


Figure 5.7: Throughput of the Chrome browser at different CPU frequencies.

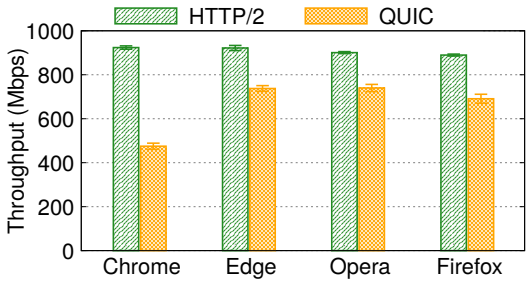


Figure 5.8: Throughput of four different browsers during file download.

the throughput over HTTP/2 still remains above 900 Mbps even at 1.60 GHz. Note, unless otherwise specified, all the other experiments in this work are done with the CPU set to 3.40 GHz. Then, we test on a machine with a more advanced CPU, Intel Core i7-10700 with a 4.80 GHz maximum turbo frequency. The QUIC downlink throughput stays close compared to the 6700 machine at the same frequency while it can reach 530 Mbps at 4.80 GHz. This suggests that increasing CPU computing power can marginally narrow the performance gap between QUIC and HTTP/2.

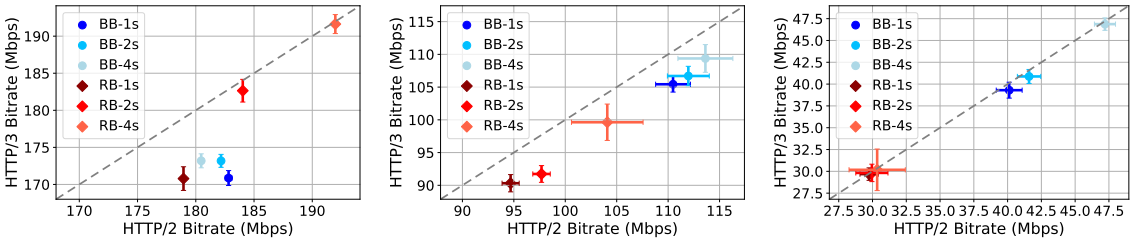
Table 5.2: Browsers' CPU usage (%).

Browser	HTTP/2	HTTP/3
Chrome	77.1±4.7	97.4±4.5
Edge	28.4±3.9	81.1±7.7
Opera	27.9±3.3	84.9±14.5
Firefox	185.8±61.9	213.0±46.5

Comparing Different Browsers. In addition to Google Chrome (v102), we extend our HTTP file download experiments to other QUIC-enabled web browsers: Mozilla Firefox (v105), Microsoft Edge (v106), and Opera (v93). We plot the download throughput statistics for four browsers in Figure 5.8 and list their CPU usage data in Table 5.2. Note that we were unable to isolate the CPU usage of Firefox's network service but we ensure that no other activities running in Firefox. We find that, all the browsers have a worse performance when QUIC is enabled, with increased CPU usage. The variation in QUIC performance across browsers likely comes from differences in their QUIC implementation, networking stack efficiency, and interaction with the underlying OS for packet processing. Therefore, the slow QUIC download issue is prevalent across major commercial browsers. This can significantly affects the end-user experience, especially when downloading bulk data at a high speed.

5.4 Application Study

Our experimental findings have painted a compelling narrative about QUIC's performance not just in bulk file transfers, but also other applications, including video content delivery and web page loading, despite their intermittent traffic patterns. We now delve into these application areas to further showcase the impact of QUIC.



(a) Stream over Ethernet. (b) Stream over 5G. (c) Stream over 4G.

Figure 5.9: Comparing average video chunk bitrate between HTTP/3 (QUIC) and HTTP/2.

5.4.1 Video Streaming

The vast and growing demand for high-quality video content and smooth delivery on the Internet underscores the importance of efficient protocols for adaptive bitrate (ABR) video streaming [88]. Leveraging both QUIC and HTTP/2, we set out to explore their real-world implications on video streaming performance.

We employ `ffmpeg` to encode a custom 4K video with H.264, generating six tracks at different bitrates. 4K video streaming usually requires 35-100 Mbps [65, 40, 185], which can be easily achieved by today's high-speed networking like 5G. In order to challenge the rate adaptation controllers, avoid trivial bitrate selection, and examine future ultra-high resolution videos and extended reality (XR) performance over high-speed connectivity, we scale up the video bitrates with the top track bitrate reaching 200 Mbps, to match the median throughput of 5G network traces [260, 185]. Specifically, the bitrates are 20 Mbps, 40 Mbps, 80 Mbps, 120 Mbps, 160 Mbps, and 200 Mbps. We also encode the video into three different chunk durations, 1s, 2s, and 4s. We set up a dash.js server for ABR video streaming. The server is configured to support two major categories of bitrate adaptation algorithms: Buffer-Based (BB) [140] which selects bitrates with the goal of keeping the

buffer occupancy high, and Rate-Based (RB) [174] which selects the highest bitrate below the bandwidth predicted from experienced throughputs during past chunk downloads. The client machine runs a Chrome browser to fetch and play the video content from the server.

We evaluate ABR video streaming under three types of network conditions. In addition to the 1 Gbps Ethernet link considered in our previous experiments, we run `tc` [8] to emulate 4G and 5G networks using real network traces, randomly selected from the Lumos5G dataset [182]. For each network type, we have two traces each for walking and driving scenarios to incorporate various mobility patterns. We conduct such a trace-driven emulation to ensure QUIC and HTTP/2 experience the same set of network conditions and to provide better reproducibility across different rounds.

We measure video chunk bitrate and CPU usage during the streaming process. Each experimental setup is executed 20 times. As shown in Figure 5.9, the results of streaming ABR videos over QUIC and HTTP/2 suggest that QUIC performs worse than HTTP/2 in Ethernet and 5G scenarios. The bitrate reduction goes up to 9.8%. This is likely due to the bandwidth in these two network settings being high enough to saturate the client CPU. Revisiting our earlier discussions in §5.3, we discover that, the bottleneck bandwidths after which QUIC cannot fully utilize the link capacity for the lightweight clients and Chrome are around 500 Mbps and 600 Mbps, respectively. Taking into account the video playback overhead (*e.g.,* decoding and rendering), this bottleneck point could be further lowered. On the other hand, for the slow 4G networks, shown in Figure 5.9c, the performance difference is not that significant. The HTTP/2 setups have a slightly better overall bitrate.

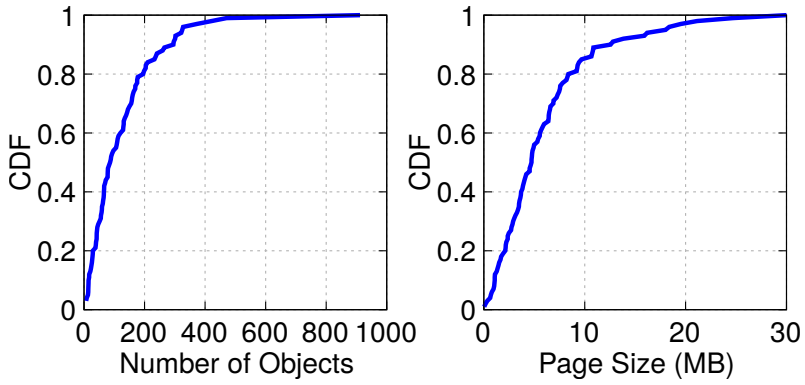


Figure 5.10: Website characterization.

5.4.2 Web Page Loading

Web browsing (*i.e.,* web page loading) plays another crucial role in the Web ecosystem. Unlike bulk file download, loading a web page usually involves transferring multiple small objects that can be either concurrent or sequential depending on object dependencies. We conduct an extensive experiment with Alexa's top 100 websites.

First, we use the original URLs to directly load the remote websites, with QUIC enabled on Chrome. We repeat the tests on each website 20 times. Surprisingly, the page load tests on most websites do not capture any HTTP/3 objects, which means those websites have not enabled QUIC yet. Only 16 websites exhibit HTTP/3 traffic during page loads. The website containing the largest portion of HTTP/3 objects is `www.discord.com` with no HTTP/1.1 objects, 8.5 HTTP/2 objects, and 30.0 HTTP/3 objects on average. It is also noteworthy that, with various third-party links (for example, for tracking or generating dynamic content purposes) visited from JavaScript files embedded in the main page, none of the tested websites are completely loaded over HTTP/3.

Then we download these 100 websites using SiteSucker [72] and host them locally

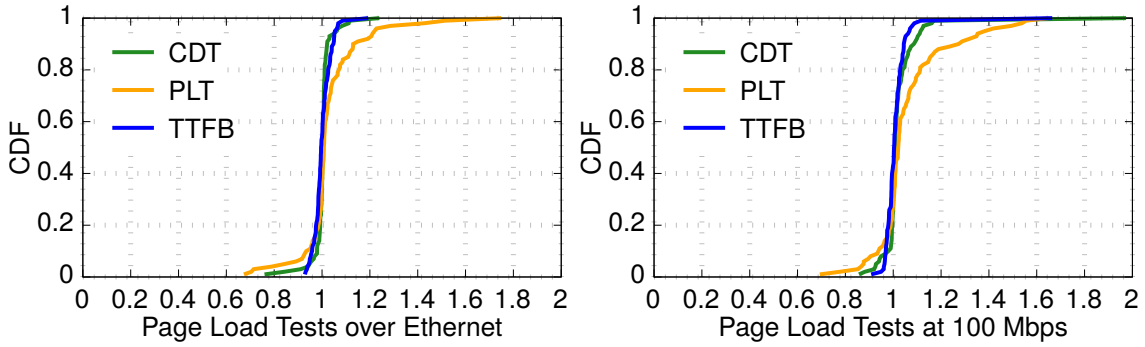


Figure 5.11: Web page loading results (HTTP/3 over HTTP/2).

on our web server. One challenge we encountered is, to our knowledge, there is no tool that can be used to download an entire website with countless external links so that it can be fully hosted locally. There are tools to record web page load and replay locally, *e.g.,* Web Page Replay (WprGo) [67] and Fiddler [68]. However, since they do not really copy the entire website, it is not possible to toggle between different HTTP protocols during the replay. Figure 5.10 shows the website features which include the number of objects and the total page size, illustrating the diversity of our test websites. Using `chrome-har-capturer` [45], we build scripts to collect HTTP Archive (HAR) [10] files and calculate the evaluation metrics. To compare the page load performance of QUIC and HTTP/2, we utilize three major metrics: (1) content download time (CDT) [99], defined as the time to download all content needed to load the website, after which the rendering process can start; (2) page load time (PLT) [236, 186], at which the rendering of all components of the page is finished; and (3) time-to-first-byte (TTFB) [97], which is the delay from sending the request to receiving the first byte of the response.

We repeat the page load tests 20 times for each website over Ethernet. We also use `tc` to throttle bandwidth at 100 Mbps, to examine the performance at limited bandwidth

conditions (*e.g.,* 4G/5G). Note that we do not directly use mobile network traces because a web page load is too fast, making it difficult to ensure consistent network conditions across rounds by replaying traces. Figure 5.11 compares the timers (CDT, PLT, and TTFB) for QUIC and HTTP/2. A data point greater than 1.0 means the corresponding timer is longer in QUIC tests. We can learn from the results that, the performance difference is not as significant as that observed in video streaming tests. On average, QUIC's PLT is 3.0% longer than HTTP/2's. However, there is a long tail indicating that in some cases the gap can be over 50% and up to 74.9%. We also observe the increased CPU usage in QUIC, compared to HTTP/2 page load tests. The PLT increase is not as significant as the bulk download time increase, because web page loading involves both local page rendering, which is not affected by the network protocol selection, and network data transfer.

5.5 Root Cause Analysis

With the QUIC and HTTP/2 results on various applications, we now identify the root cause of the observed performance gap. Unless otherwise noted, for experiments and analysis in this section, we use Chromium (v102) as it is a production-level implementation supporting both HTTP/2, and QUIC and it is not proprietary, thus easy to profile internal activities.

5.5.1 Eliminating Non-contributing Factors

We begin with eliminating several potential factors, most backed up with controlled experiments.

- **Server Software.** We set up another web server, Nginx-quic (v1.14.0) [75], on the

same server machine. We compare the time to download 1 GB file from Nginx and from OpenLiteSpeed (our server setup in §5.3) using Google Chrome. HTTP/2 performs similarly on both web servers while QUIC performs even worse when running on Nginx, being slower by 18%.

- **UDP/TCP Protocols.** We conduct `iPerf` UDP and TCP tests under the same network setup. The results show that both protocols can fully utilize the link bandwidth (1 Gbps), with UDP achieving 958 Mbps and TCP achieving 944 Mbps on average.
- **HTTP syntax.** HTTP/3 [96] serves as the mapping of HTTP for using QUIC as the transport. Adapted from HTTP/2 [93], it has an almost identical syntax structure to HTTP/2 [118, 51].
- **TLS Encryption.** Both QUIC (TLS v1.3) and HTTP/2 (TLS v1.2) employ the `TLS_AES_128_GCM_SHA256` cipher on our web server. We also benchmark different cipher suites and the results do not significantly affect the performance.
- **Parameter Tuning.** We tune QUIC-specific parameters such as enabling/disabling packet pacing and adjusting path MTU discovery [116]. We do not observe noticeable improvements compared to the original performance gap.
- **Client OS.** We repeat the above experiments on Mac OS and Windows for the receiver, and observe similar results.
- **Disk and Memory.** We download files directly to a volatile RAM-based disk using Linux `tmpfs` [58]. We also test with Linux HugePages [92] to avoid frequent memory swaps. Neither approach helps in improving QUIC performance.

5.5.2 Evidence from Packet Trace Analyses

Next, We get insights by analyzing `tcpdump` packet traces.

QUIC Perceives Much More Packets than HTTP/2. We notice that for QUIC, the number of packets received by the OS's UDP stack is an order of magnitude higher than the number of packets received by the TCP stack during HTTP/2 downloads (744K versus 58K on average). We have confirmed this is not caused by retransmissions. While prior tests show that increasing the packet size up to MTU can help [35], all QUIC packets in our experiments are already MTU-sized (1472 bytes, excluding the 8-byte UDP header and the 20-byte IP header in the standard 1500-byte MTU setup). We also verify that the numbers of packets transmitted over the wire are very close between QUIC and HTTP/2. The difference of their transport-layer-perceived packets is because TCP (HTTP/2) uses generic receive offload (GRO), where the link layer module in the OS combines multiple received TCP segments into a large segment of up to 64 KB. However, despite the availability of UDP GRO, it is not used by QUIC, and integrating GRO with QUIC faces challenges as to be discussed in §5.5.3.

QUIC has a much Higher RTT Dominated by Local Processing. We measure the packet round-trip time (RTT), defined as the time between when a data packet is sent out from the server and when the first packet to acknowledge it is received. The RTT consists of the propagation delay spent on the paths and the processing delay spent on the receiver side. Though TCP and QUIC have different ACK mechanisms, the average packet RTT can still reflect how fast packets are transferred and processed, and thus help adjust the sending rate. The average RTT for HTTP/2 download is 1.9ms while QUIC's RTT skyrockets to 16.2ms. Also, both protocols exhibit similar temporal RTT patterns, mostly

stable. Since the `ping` RTT between the two machines is only 0.23ms as measured, the endpoint packet processing takes most of the packet latency.

The above results provide further evidence that the performance bottleneck of QUIC appears to be on the receiver side.

5.5.3 Root Causes via OS/Chromium Profiling

To definitively pinpoint the root cause, we conduct fine-grained profiling in both the OS kernel (OS's networking stack) and the user space (Chromium's networking stack) using Linux `perf` [57].

Excessive Receiver-side Processing in the Kernel. We run 1 GB file downloads on Chromium with QUIC and HTTP/2. Meanwhile, we use `perf` to monitor events in the Linux networking subsystem (`net`) associated with Chromium's network service. For QUIC, we observe a huge number of calls on `netif_receive_skb` which is invoked when a packet is received at the network interface. Specifically, there are 231K calls of this type witnessed during a single QUIC download compared to a mere 15K in an HTTP/2 download. This difference roughly corresponds to the difference in the number of received UDP and TCP packets (§5.5.2).

A standard way to reduce packet processing overhead in the OS is to involve NIC offloading that has been widely used for TCP, including segmentation offload such as TCP Segmentation Offload (TSO) and Generic Segment Offload (GSO) on the sender side and receive offload such as Generic Receive Offload (GRO) on the receiver side[1]. While some existing efforts [28, 35] have shown the effectiveness of UDP sender-side

[1] Another solution, UDP Fragmentation Offload (UFO), uses IP fragmentation. It was deprecated so we do not consider it in this work.

Table 5.3: Download 1 GB file with and without offloading.

Setup	# Sent Packets	# Recv Packets	Time (s)
QUIC (on)	743K	743K	18.60
QUIC (off)	744K	744K	18.82
HTTP/2 (on)	19K	53K	9.36
HTTP/2 (off)	744K	744K	10.84

offloading, our work pioneers in pinpointing the criticality of *receiver-side offloading* for today's commodity QUIC client hosts.

In addition, we note that realizing offloading for QUIC is challenging. First, unlike TCP which uses a byte stream model so its payload can be flexibly (re)packetized, UDP's offloading logic must preserve the packet boundaries. The existing UDP GSO/GRO thus only supports offloading a train of UDP packets with identical lengths specified by the application [110]. This constraint makes directly applying UDP GSO/GRO to QUIC inefficient, due to QUIC's inherent multiplex nature: QUIC frames belonging to different streams vary in size and are multiplexed after encryption. As a result, if a train of UDP datagrams (containing the encrypted frames) has different packet sizes, existing UDP GSO/GRO cannot offload them. Second, blindly aggregating many UDP datagrams and transmitting them in a single burst may cause congestion-related packet losses and fairness issues, particularly over the wide-area Internet [110, 142]. Third, the diverse QUIC variants add complexity to realizing the QUIC offloading logic in NIC hardware. Likely due to the above reasons, although UDP GSO/GRO [21] is available in the newer Linux kernel versions, none of the QUIC implementations have adopted it.

We carry out additional experiments with available offloading mechanisms (TSO, GSO, and GRO) enabled and disabled on both server and client sides. The results in

Table 5.4: A breakdown of packet processing time.

Chromium Networking Stack	**QUIC** (8.5s)	**HTTP/2** (4.1s)
Read UDP/TCP packets from socket	0.248s	0.037s
Process UDP/TCP packets for payload	0.310s	0.084s
Decode QUIC/TLS-encrypted packets	0.660s	0.814s
Parse decrypted QUIC/HTTP2 frames	3.468s	3.182s
Generate QUIC responses (*e.g.,* ACK)	2.972s	–
Others	0.859s	0.001s

Table 5.3 indicate that UDP (QUIC) does not benefit from GRO/GSO. In contrast, TCP shows a more significant reduction in download time, with much fewer packets processed by the OS's TCP stack. The discrepancy in the number of packets sent and received is likely because the server-side offload may have a different power on segmentation compared to client-side receive offload capability on packet reassembly. Note, all other experiments in this study have them turned on.

When profiling kernel-level activities for QUIC, we also observe a more significant proportion of calls to function `do_syscall_64` (17K for QUIC, compared to 4K for HTTP/2) and function `copy_user_enhanced_fast_string` (4K vs. 3K). Such intensive interactions across the user-kernel boundary are resulted from the substantial volume of QUIC packets perceived by the UDP stack. They further increase the processing overhead.

Excessive Receiver-side Processing in the User Space. The high in-kernel packet processing overhead results in high processing overhead in the user space for QUIC. To demonstrate the latter, we profile Chromium's networking stack, specifically, the `Chrome_ChildIOT` thread. Table 5.4 provides a breakdown of the time spent by each packet processing stages. The stack is primarily responsible for (1) reading UDP/TCP

packets from the socket; (2) processing UDP/TCP packets to extract the payload; (3) decoding QUIC/TLS-encrypted packets; and (4) parsing decrypted QUIC/HTTP2 frames. For QUIC, `QuicChromiumPacketReader` is responsible for reading and processing the incoming QUIC packets. Its entry point is `StartReading` and consumes 8.7s out of the total download time of 20.6s on average when downloading a 1 GB file. On the flip side, the HTTP/2 counterpart is `SpdySession` starting from `DoReadLoop`, and spends 4.1s out of 9.4s. QUIC lags behind HTTP/2 at each of the four stages above. Furthermore, we notice that, out of the 8.7s consumed by `QuicChromiumPacketReader`, QUIC spends 3.0s generating responses such as ACKs. In contrast, for HTTP/2, the ACKs are handled by the OS kernel, and they are generated more efficiently and sparsely due to various optimizations such as TCP delayed ACK and receive offload.

5.6 Recommendations for Mitigation

Following the above experiments and analysis, we make several recommendations for mitigating the observed issues.

Adoption of UDP GRO on the Receiver Side. Most importantly, UDP GRO needs to be deployed on the receiver side to reduce the number of packets handled by the UDP stack. This will reduce not only the in-kernel overhead, but also QUIC's processing overhead in the user space. However, given the heterogeneity of today's commodity hosts (PCs, mobile devices, and even embedded devices, with diverse OSes), wide deployment of UDP GRO can be challenging, not to mention supporting it in the NIC hardware.

QUIC-friendly Improvements to the offloading solutions. We advocate that the generic offloading solutions need some QUIC-friendly improvements. First, UDP GSO/-

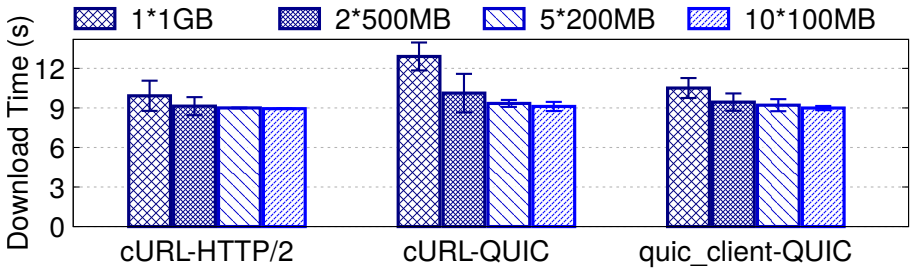


Figure 5.12: Parallel download experiments (instances of `cURL` or `quic_client` download 1 GB of files in total).

GRO needs to support offloading a train of packets with different sizes (§5.5.3). Second, UDP GSO needs proper pacing configurations (*i.e.,* avoid transmitting too many UDP packets in a single burst that may incur network congestion) over the wild Internet. Ideally, the pacing logic should be properly interfaced with QUIC's logic such as congestion control[2].

Optimizing QUIC logic on the Receiver Side. There is also room for improvement in the QUIC logic on the receiver side. Sending delayed QUIC ACKs [144] can help reduce the overhead on generating QUIC responses. Besides, we note that Chromium currently uses `recvmsg` to read individual UDP packets; using `recvmmsg` to read multiple UDP packets in a single system call may help improve the receiver-side performance.

Multi-threaded download. We also notice that Chromium uses a single thread for receiving network data. When fetching large files, using multi-threaded download (each thread running on a separate CPU core) can improve the receive-side performance. Since the tested Chrome browser version does not have built-in support of multi-threaded download, we conduct an experiment where we launch k instances of `cURL` or `quic_client`,

[2]Google has a simple experimental pacing design for UDP GSO, but it is not designed specifically for QUIC and was only tested in data center networks.

each downloading a file of 1 GB$/k$ ($k = 1, 2, 5, 10$). We use the latest finishing time across the k instances to calculate the overall transfer time of 1 GB worth of data. As shown in Figure 5.12, increasing k helps reduce the download time, in particular for QUIC. Nevertheless, similar to parallel TCP connections, this approach may incur fairness issues in network resource allocation. The sender could use existing solutions such as coupled congestion control [204, 238] to bound the aggregated aggressiveness of the k QUIC sessions.

5.7 Summary

QUIC, with its design principles aimed at eliminating head-of-line blocking, introducing fast connection establishments, and integrating transport-layer security, promises a more responsive and secure Web experience. However, our study, along with others, highlights areas where QUIC might not meet expectations. In environments like fast Internet (>500 Mbps in our experiments), QUIC's performance may not always live up to its name ("quick"). Through comprehensive performance profiling, we reveal the root cause to be the pronounced *receiver-side* processing overhead. This overhead manifests in the form of excessive data packets observed at Layer 3 and above, as well as QUIC's distinctive user-space ACKs.

There are notable challenges to grapple with. The absence of certain offloading techniques like UDP GRO, the user-space nature of QUIC, and QUIC's inherent reliance on UDP might complicate its deployment, especially in environments that have been meticulously optimized for TCP. Nevertheless, it is pivotal to note that QUIC is still in its nascent phase, with considerable research, exploration, and development fervently aiming to enhance its performance. The ongoing efforts and collaborations from multiple stakehold-

ers in the Web ecosystem, including OS vendors, QUIC developers, and standardization organizations, will play a crucial role in the evolution of QUIC. As more web services transition to HTTP/3, we can expect a broader adoption of QUIC across the Internet. We hope that our findings can spur more explorations to improve QUIC, and upper-layer protocols in general, boosting their performance for the next generation networks, services, and applications.

CHAPTER VI

LEO Satellite vs. Cellular Networks: Exploring the Potential for Synergistic Integration

Low-Earth-Orbit (LEO) satellite networks, such as Starlink, are transforming global network connectivity by bringing Internet access to remote and underserved areas. However, the current coverage and performance of the LEO satellite network service compared with those of cellular networks are under-explored. In this chapter[1], we present a measurement study of the Starlink LEO satellite network in comparison with cellular networks, aiming to uncover the potential for synergistic integration. Through a large-scale data collection campaign and in-depth analysis, we identify the performance characteristics of two Starlink configurations, evaluate the coverage of the current Starlink deployment compared to major cellular carriers, and investigate the potential benefits of enabling multipath using both LEO satellite and cellular networks.

[1]Part of this work was carried out in collaboration with Bin Hu, who was a Ph.D. student at the University of Southern California at the time.

6.1 Introduction

Since its debut in 2020, SpaceX's revolutionary Starlink, a Low-Earth-Orbit (LEO) satellite network, has gained over 1 million users with its promise of delivering global connectivity [73]. It aims to provide services to remote and inaccessible areas where traditional wired or wireless networks fall short. However, both LEO satellite systems and cellular networks face distinctive challenges that impede their ability to consistently attain peak network performance. The seamless operation of Starlink requires an unobstructed view of the sky. On the other hand, the coverage of 4G/5G cellular networks heavily relies on the extensive deployment of base stations. Although some initial research has previously touched on Starlink's performance [177, 151, 172], there lacks a comprehensive investigation that compares the LEO satellite networks to terrestrial cellular networks. It is also necessary to explore the mobility of Starlink dishes. Additionally, the performance and accessibility of Starlink and cellular networks can often complement each other, and thus enabling multipath connections may bring superior throughput and reliability compared to relying on a single network.

To fill this gap, we conduct a large-scale measurement study to understand the performance characteristics of Starlink satellite networks and compare the performance coverage between Starlink and cellular networks.

First, we collect a unique driving dataset comprising experimental results of both Starlink and cellular networks, including two types of Starlink configurations and three cellular carriers. During our data collection, we conduct a range of experiments on several network performance metrics. Our driving routes that span across five states in the US, consider different geographical areas, densities of infrastructure deployment, and user populations.

Throughout the data collection process, we cover a total distance of over 3,800 km in over one month. The main challenge comes from how to collect data for both network types simultaneously and ensure apple-to-apple fair comparison. We address this by installing two Starlink dishes on the vehicle rooftop and carrying five smartphones set side by side. With our dataset, we aim to answer the following questions:

What is the performance achievable by Starlink networks, in particular under mobility? Previous studies look at Starlink's performance using stationary hardware and none of them considers mobility. For stationary use, there are already numerous ways to access the Internet, while on the move people basically only rely on cellular networks. Therefore, we take a different view to examine Starlink under mobility and compare it with cellular networks. We evaluate key performance metrics of Starlink, such as throughput, latency, and packet loss. Through bulk transfer and ping tests conducted during driving sessions across diverse geographic locations, we evaluated the performance disparities between TCP and UDP, uplink and downlink, and different Starlink configurations. Furthermore, we analyze the impact of different factors such as moving speed and TCP parallelism on network performance.

How does Starlink's performance and coverage compare to that of major cellular carriers? To better understand whether Starlink offers a broader coverage than traditional cellular services. We compare the coverage and performance distribution of Starlink and cellular networks in different areas. We aim to provide insights into the viability of Starlink as a potential alternative to cellular networks for those who require reliable connectivity in the wild.

What is the potential of enabling multipath for Starlink and cellular networks?

The multipath technology was introduced to enhance network performance by utilizing multiple network paths simultaneously. Given the highly variable performance of Starlink and cellular networks, combining both networks may lead to a more satisfactory user experience. Thus, we conduct emulation using our collected network traces to compare the performance of single-path and multipath transports and understand the improvements brought by MPTCP in throughput and reliability.

We summarize our key findings as follows:

- Compared to cellular networks, Starlink suffers from elevated packet loss while the latency stays similar. We find that TCP severely suffers from such a high packet loss of Starlink, leading to only 1/5 of the throughput achieved by UDP over Starlink. TCP parallelism brings more benefits to Starlink than to cellular networks likely due to its effective handling of packet loss. Nonetheless, this finding also calls for better congestion control or Forward Error Correction (FEC) algorithms tailored for such characteristics.

- We compare two types of Starlink configurations, Roam and Mobility, both designed for use outdoors. While we do find better overall performance offered by the more expensive one (Mobility, having $2\times$ higher mean/median throughput), its additional cost cannot be fully justified by current usages. The 75-percentile throughput of Roam, 93 Mbps, can already meet most application requirements in the wild.

- Due to the ultra-high speed operations of LEO satellites, the user's moving speed is negligible and thus poses little impact on Starlink's network performance, resulting in a similar trend across different speeds compared to cellular networks.

- Cellular networks offer better performance in urban areas thanks to the densely deployed base stations, while Starlink wins in suburban and rural areas with fewer obstructions. Since most of the time, the cellular services experiences are either low-band 5G or 4G LTE, the cellular throughput does not often reach very high. Starlink demonstrates better overall performance. However, even after combining cellular and Starlink, there are still areas with low performance (¡50 Mbps), likely due to the combined effect from cellular base station deployment and obstruction to satellite connections.

- In our MPTCP experiments, we first find that, under the default OS configuration, MPTCP using Starlink and cellular networks brings marginal throughput gains compared to single-path transfer due to the high variation of both networks easily filling the buffer. After tuning the OS buffer settings, we see more significant improvements (up to 66% improvement over the better path), benefiting from the complementary characteristics between Starlink and cellular networks. This emphasizes the need for better MPTCP scheduler and congestion control algorithms.

To the best of our knowledge, this work represents the first in-depth investigation that closely examines the performance of Starlink in comparison to cellular networks, particularly under mobility. Through our measurements and analysis, we have firmly supported the case for multi-path transport using LEO satellite networks and cellular networks. We have released all the measurement data and source code associated with this study [78].

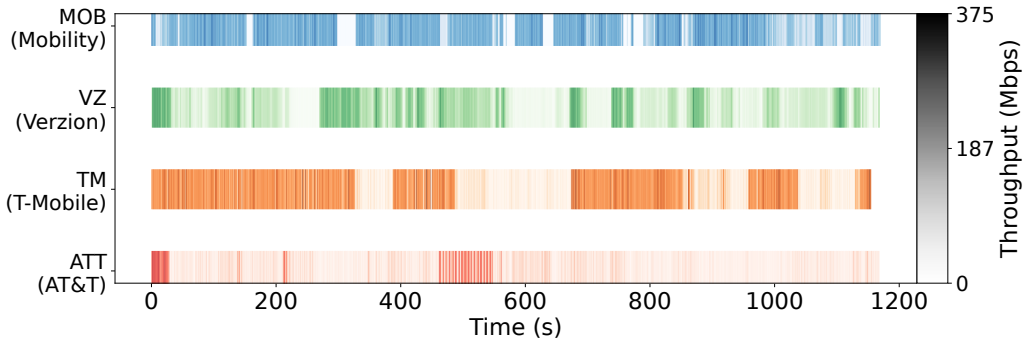


Figure 6.1: Download throughput of different networks.

6.2 Motivation

An LEO satellite network consists of thousands of satellites orbiting the Earth at an altitude of hundreds of miles. These networks operate differently from cellular networks. The user terminal, typically a satellite dish, communicates with an LEO satellite, which is connected to a ground station. The ground station is responsible for relaying data to and from the Internet. They aim to provide services to remote and inaccessible areas where traditional wired or wireless networks fall short.

However, both LEO satellite networks and cellular networks face unique challenges. For example, LEO networks require an unobstructed view of the sky, whereas 4G/5G cellular networks heavily rely on the extensive deployment of base stations. These issues are further pronounced when users are moving. There has been a lack of comprehensive investigation comparing LEO satellite networks to terrestrial cellular networks, and it is necessary to explore the mobility of Starlink dishes. Given the distinct nature of the two types of networks, the performance and accessibility of Starlink and cellular networks may vary, but they may complement each other.

Figure 6.2: Two dishes are mounted on the rooftop and five smartphones are placed side by side in the vehicle.

To validate our hypothesis, we conduct a preliminary experiment using smartphones connected to Starlink routers via Wi-Fi and to cellular networks, in a moving vehicle. We perform `iPerf` data transfer tests to AWS servers and compare the throughput between Starlink and cellular networks. Our results are summarized in Figure 6.1, where darker colors (blue/green/orange/red) indicate periods of higher throughput. As we traversed different areas, we can observe instances where Starlink demonstrated better throughput performance compared to the cellular network, and vice versa. If combining the strengths of both networks, users can enjoy a seamless and stable high-performance experience throughout their journey. This motivates us to conduct further investigations to comprehensively understand the performance and coverage of both network types in real-world scenarios, and to assess the feasibility of multi-path transport for LEO satellite and cellular networks.

6.3 Measurement Methodology

6.3.1 Hardware and Services

We first introduce the hardware setups and network services used.

Starlink offers a variety of plans, including Residential, Business, Roam, and Mobility, among others. In this study, our focus is on the Roam (RM, for short) and Mobility (MOB) plans, specifically designed for portable and in-motion use. The Roam plan offers easy portability compared to a standard Starlink dish and provides Internet access while stationary or on the move. The Mobility plan, on the other hand, receives the highest priority in the network, for instance, during congestion. It is specifically designed to support critical in-motion applications, such as those used by emergency personnel. Besides, Mobility has over $4\times$ the hardware cost and a higher monthly fee than Roam. For a direct comparison between the two plans, we have installed both Starlink dishes on the rooftop of a vehicle. However, we acknowledge the possibility of interference that may affect the results. For experiments involving cellular networks, we have selected three major commercial carriers in the US: AT&T (ATT), T-Mobile (TM), and Verizon (VZ). We utilize five Samsung Galaxy S21 smartphones. Three of these devices are connected to the cellular services, while the remaining two are connected to the Starlink dishes using Wi-Fi.

Figure 6.2 shows our dish placement and smartphone setup.

6.3.2 Software Measurement Tools

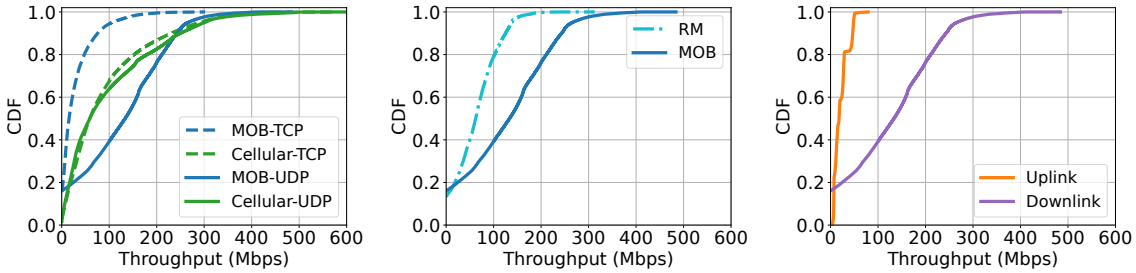
We utilize several software tools for data collection: (1) We use `iPerf` to run TCP/UDP downlink and uplink data transfer tests. (2) To measure latency accurately, we

have developed an Android application that sends ping packets using UDP (`UDP-Ping`), as ICMP ping packets are often blocked by certain servers. (3) To collect information on network type, vehicle speed, GPS location, and signal strength, we employ 5G Tracker [184, 185], a monitoring toolkit for cellular networks. We have made modifications to enable its functionality under both Wi-Fi and cellular connectivity.

6.3.3 Data Collection

We perform extensive drive tests across major cities and interstate freeways (spanning five states) in the US. It encompasses diverse geographical regions, including densely populated urban areas with tall buildings and open rural areas with minimal obstructions. We drive at varying speeds in various areas. However, our driving speed is capped at 100 km/h due to speed limits on different road segments. The driving routes consist of both straight and curved roads, aiming to generalize our results with regard to vehicle steering. We collect data during both daytime and nighttime. Furthermore, our data collection includes not only clear weather conditions but also rainy and snowy conditions, to capture potential performance variations caused by environmental conditions. Note that, despite the breadth of factors considered, not all are explicitly discussed in the following sections. Upon analysis, certain environmental factors such as terrain and the time of day, along with network-related factors such as server locations, are found to have a minimal impact on the network performance.

Our driving trip yields a unique driving dataset, containing 1,239 network tests and 9,083 minutes of traces. Our field trip covers a total travel distance of over 3,800 km.



(a) TCP vs. UDP. (b) Roam vs. Mobility. (c) Uplink vs. Downlink.

Figure 6.3: Throughput performance comparison from different aspects.

6.4 Starlink Basic Performance

In this section, we analyze and compare the performance characteristics of Starlink satellite networks with cellular networks, focusing on two Starlink configurations, Roam and Mobility. We evaluate fundamental performance metrics including throughput, latency, and packet loss, and conduct an in-depth examination of the performance gaps from different aspects. Besides, we also examine the impact of vehicle speed and TCP parallelism on performance.

6.4.1 Throughput, Latency, and Packet Loss

We start with analyzing the key performance metrics of Starlink.

TCP vs. UDP downlink. Figure 6.3a plots the Cumulative Distribution Functions (CDFs) for TCP and UDP downlink throughput of both Starlink (Mobility) and cellular (AT&T, T-Mobile, and Verizon) networks. While the performance disparity between cellular TCP and UDP is minimal, the results consistently reveal that UDP outperforms TCP in satellite networks, with the mean throughput being 128 Mbps and 29 Mbps, respectively. This performance advantage of UDP can be attributed to the significant packet loss

experienced by TCP in satellite networks. To substantiate this proposition, we analyze the `tcpdump` traces collected while running `iPerf` and plot the average TCP packet loss across all networks in Figure 6.5. When using Starlink, there is a much higher occurrence of packet loss in both the uplink and downlink directions, compared to cellular networks. This leads to retransmissions ranging from 0.3% to 1.3%. Such elevated packet loss significantly impacts TCP performance and ultimately decreases Starlink's TCP throughput. Additionally, it is important to note that the UDP performance achieved with the Mobility plan demonstrates a level of throughput that is comparable to that of cellular networks, highlighting its effectiveness in facilitating data transfer.

Roam vs. Mobility. Figure 6.3b compares the network performance between the Starlink Roam and Mobility plans. The Mobility plan exhibits superior performance compared to Roam, likely because Roam's dish lacks the ability to adjust its orientation promptly under high mobility while Mobility is designed for in-motion use with a wider field of view. This may also benefit from the advertised prioritization for Mobility during network congestion. The median/mean throughput for Mobility and Roam are 197/128 Mbps and 93/63 Mbps, respectively. However, such 2x performance improvements are not that significant compared to the over 4x higher cost on the hardware [6], since the network requirements of most applications such as 1080P video streaming can already be met by Roam. Unless for critical applications with demanding requirements, the more cost-friendly Roam plan can effectively serve as a viable alternative to the Mobility plan.

Uplink vs. downlink. Comparing the UDP uplink and downlink transfer of Starlink, we find that the downlink throughput is around 10x higher than the uplink, as depicted

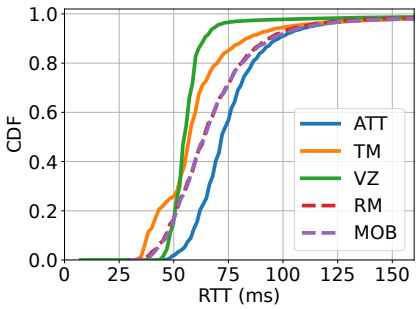


Figure 6.4: UDP Ping Latency.

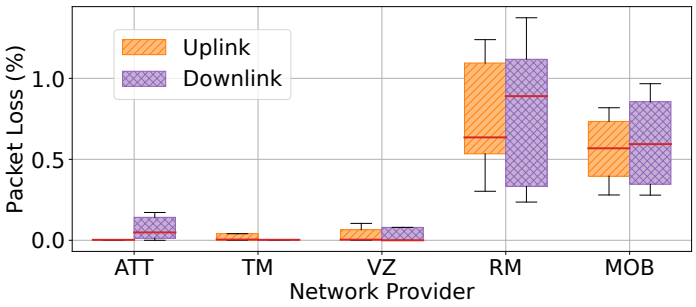


Figure 6.5: Packet loss in TCP transfer.

in Figure 6.3c. This design choice of using FDD for dividing uplink and downlink channels [143] aligns with the inherent characteristics of network traffic, where users typically consume more data in the form of downloads rather than uploads. Additionally, Starlink's satellite dishes are optimized to prioritize downlink speed over uplink speed due to limited power resources and transmitting capacity. It is more energy-efficient to receive a signal than to transmit it [147, 228]. We also learn that, by design, the Starlink network offers higher downlink bandwidth than the uplink [77].

Latency. We utilize our Android application, `UDP-Ping`, to measure latency. We allocate 1024 bytes to each UDP packet and calculate the round-trip time (RTT) for each acknowledged packet. Figure 6.4 illustrates the CDF of latency for five different networks. Overall, the RTTs for all networks primarily fall within the range of 50 to 100ms. Verizon and T-Mobile exhibit the lowest RTT values, while Starlink Roam and Starlink Mobility plans experience comparatively higher latency. It is surprising to see that Starlink's latency is not significantly worse than that of cellular networks. Intuitively, satellite networks should incur significantly higher latency due to the long satellite-ground distance. However, the additional latency introduced by satellite data transmission is only approxi-

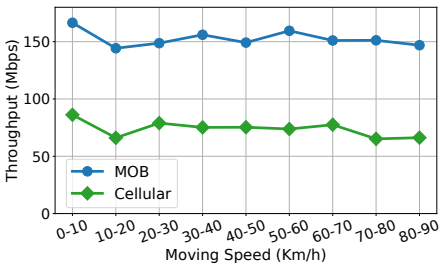


Figure 6.6: Impact of speed.

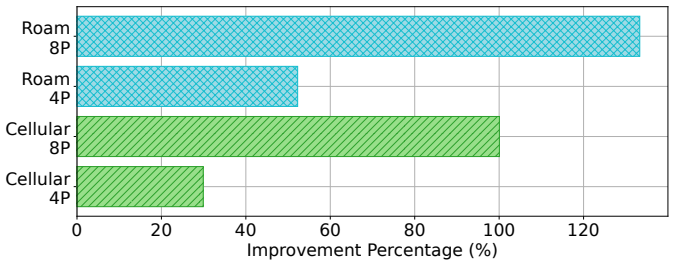


Figure 6.7: Impact of TCP parallelism.

mately 1.8 ms one way, thanks to the "low-earth" orbiting nature. The estimation is derived from the LEO satellites' orbital altitude of around 550 km and the speed of light:

$$\text{Latency} = \left(\frac{\text{Distance}}{\text{Speed of light}}\right) = \left(\frac{550 \text{ km}}{299792 \text{ km/s}}\right) = 1.835\ ms \tag{6.1}$$

This acceptable level of latency indicates that satellite networks can provide reliable performance comparable to cellular networks. Notably, AT&T demonstrates the highest network latency among the tested networks, likely due to its relatively low coverage along our trip [49].

6.4.2 Potential Factors Affecting Throughput

We discuss the impact of two factors, moving speed and TCP parallelism, on the throughput performance of Starlink.

Moving speed. To ensure an unbiased analysis and isolate other factors, we specifically extract data collected in rural areas which offer minimal obstructions. This methodology mitigates the challenges associated with conducting high-speed driving tests in urban environments, where speed limits restrict the full exploration of network performance.

More than 90% of our urban data were collected at speeds below 50 km/h. Moreover, the satellite connections can be negatively impacted by obstructions, while cellular networks may exhibit better performance in urban areas.

Figure 6.6 shows the average throughput grouped by speed. Notably, both satellite (Mobility) and cellular (AT&T, T-Mobile, and Verizon) network throughputs have minimal variation in relation to driving speed. This suggests that the network performance remains largely unaffected by the vehicle's speed during normal driving conditions. Considering Starlink's operation in low earth orbit at an approximate speed of 28,000 km/h, the speed of an object on the ground is negligible and can be considered stationary. For cellular networks, on the other hand, efficient handovers contribute to maintaining consistent throughput.

TCP parallelism. TCP parallelism is a technique that enables parallel TCP connections between senders and receivers to increase throughput. Our experiments compare three schemes: 1, 4, and 8 TCP connections. Figure 6.7 demonstrates the improvement achieved by TCP parallelism on downlink throughput for both satellite (Roam) and cellular networks. "P" denotes parallelism, where '8P' represents 8 parallel connections. Increasing the number of parallel TCP connections enhances throughput in both networks. Starlink achieves a better throughput improvement, over 50% with 4 parallel TCP connections and over 130% improvement with 8 connections. TCP parallelism optimizes bandwidth utilization by distributing data across multiple connections, thereby mitigating the impact of TCP congestion control. It also improves packet loss handling. In case of packet loss in one connection, other connections continue data transmission, minimizing the impact on overall throughput. Given the higher packet loss observed in the Starlink

network (§6.4.1), increasing TCP parallelism enables more efficient handling of packet loss, resulting in improved throughput.

6.5 Coverage Study

This section focuses on the coverage area of Starlink. We discuss the network performance in different geographical regions and the proportion of coverage within each performance level.

6.5.1 Impact of Area Types

Deploying and operating cellular base stations in rural areas incurs much higher costs due to low population density [66], while users in urban areas enjoy more reliable cellular network connections thanks to the dense base station deployment. For Starlink, urban areas with tall buildings can obstruct satellite signal transmission. Thus, the location of the network affects both Starlink and cellular networks.

During data collection, we traversed through different types of areas, recording the latitude and longitude of each data point. Then we compile a list of all cities and towns we passed through, calculate the distances from each data point to these locations, and select the smallest distance. Subsequently, using predetermined thresholds, we categorize the data into three area types: urban, suburban, and rural. The data proportion of the three areas is 29.78%, 34.30%, and 35.91%, respectively.

Figure 6.8 shows the throughput distributions for both Starlink (Mobility) and cellular networks. Here, we highlight the results of UDP downlink since Starlink's downlink throughput is inherently higher and UDP is less affected by packet loss compared to

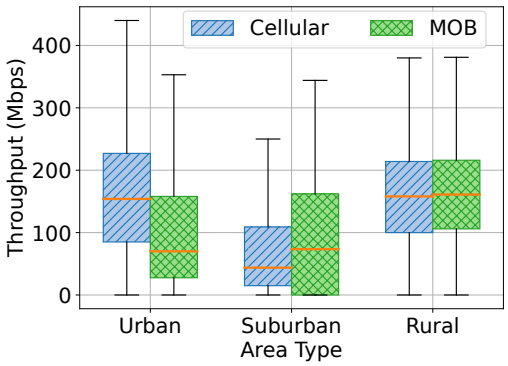


Figure 6.8: Downlink throughput at different area types.

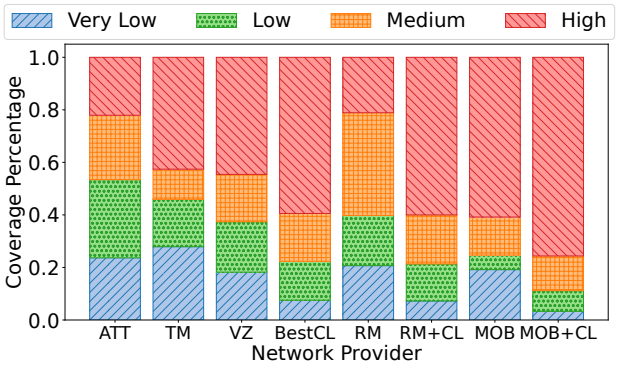


Figure 6.9: Comparison of network performance coverage.

TCP. They also reflect the upper limit of network bandwidth. It can be observed that the throughput of cellular networks decreases when reaching rural areas, while the throughput of Starlink networks increases in rural areas. This is because cellular network base stations are more densely deployed in populated areas, whereas in densely populated areas, the network performance of Starlink can be affected by obstacles such as tall buildings. From our analysis of UDP Downlink, we find that the throughput of Starlink networks is even higher than that of cellular networks in suburban and rural areas. We also find that the throughput of Starlink is distributed similarly in suburban and rural areas. During our driving trip, we found a lot of obstructions only in urban areas. Suburban areas such as towns have much fewer high buildings, leading to similar obstruction conditions to rural areas.

6.5.2 Performance Coverage

To visually represent the coverage of Starlink and cellular networks, we analyze our data and group data points of different network performance based on different perfor-

mance levels. The high-performance regions are characterized by throughput exceeding 100 Mbps, while the medium-performance regions exhibit throughput ranging between 50 and 100 Mbps. The low-performance regions have a throughput between 20 and 50 Mbps. Additionally, we consider a "very-low" performance level, where the throughput is under 20 Mbps, to understand if there are regions with extremely poor coverage for the different technologies. Although our data was collected during certain periods in each region and might not reflect the long-term network performance in specific regions, it still provides insights into the coverage of network performance of Starlink and cellular networks. The results of our analysis are presented in Figure 6.9, showcasing the proportions of different performance regions for the five networks.

We can learn that Starlink Mobility exhibits the best overall performance, with a proportion of high-performance regions at 60.61%. Verizon and T-Mobile closely follow, with proportions of high-performance regions at 44.39% and 42.47%, respectively. Starlink Roam and AT&T, however, demonstrate the poorest performance, with proportions of low and very-low performance regions approximately at 39.88% and 53.45%, respectively. We plot a bar named BestCL which indicates the best performance of all three cellular networks. This is reasonable since many mobile virtual network operators (MVNOs) utilize the services of several mobile carriers and automatically pick the best option for users. We also combine the measurement records of Starlink and cellular as shown as the bars, RM+CL and MOB+CL, in Figure 6.9. Their improvements over a single cellular network are likely due to the significant presence of non-urban areas with minimal obstructions, which currently favor Starlink's performance. Noticeably, combining all cellular networks also leads to results comparable to RM+CL. As mentioned earlier, compared to Roam

which is not designed for mobile use, the Mobility dish has a wider field of view and better positioning capability, resulting in Mobility having the overall best coverage of network performance regions.

We also plot two additional bars representing the best performance if a user has access to both Starlink and cellular networks (and can switch between them with zero effort). From a user experience perspective, this highlights the importance of implementing multipath for Starlink and cellular networks. Due to their inherent differences, Starlink and cellular networks exhibit significant variations in the coverage of high-performance regions. Compared with the original measurement records, these combinations achieve better high-performance network coverage. We are encouraged to explore the multipath feasibility in the next section.

6.6 Multipath Transport

Multipath transport, in particular MPTCP, has shown its success in numerous combinations of networks [107, 168, 215] and various applications [130, 126, 189, 265]. In this section, through realistic emulation, we demonstrate the potential of enabling multipath for Starlink and cellular networks.

Experimental setup. We run MPTCP experiments on two Ubuntu 22.04 hosts, using `MpShell` (a variant of Mahi-mahi [113, 187]). It creates multiple virtual interfaces with controlled network conditions, using packet traces and latency statistics. To this end, we use the UDP downlink throughput traces in our driving dataset and convert them to packet traces for replay on `MpShell`. Different network traces are aligned via timestamps so that they reflect the network conditions experienced by users at the same location and time.

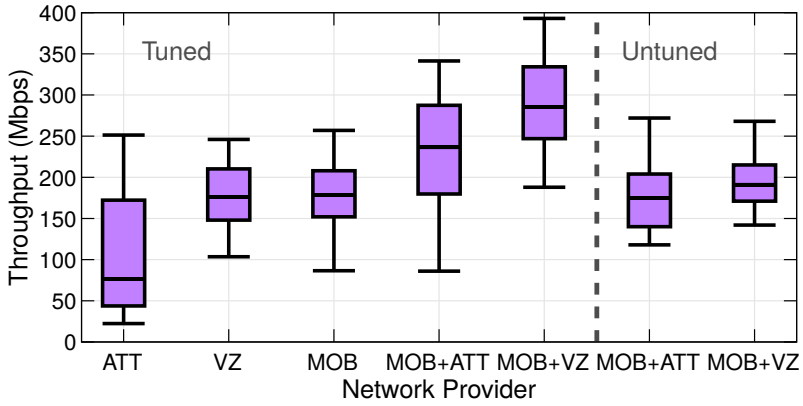
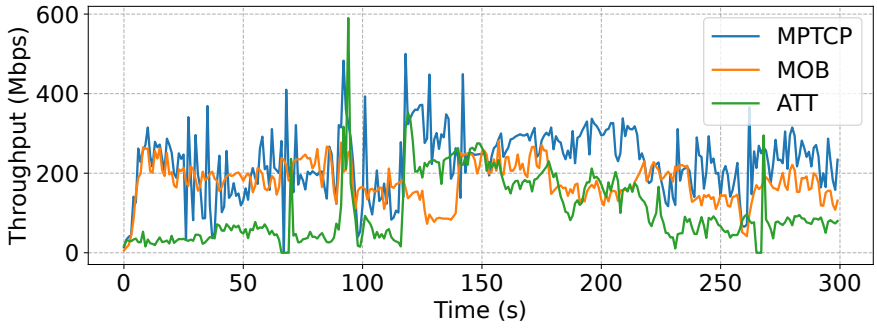


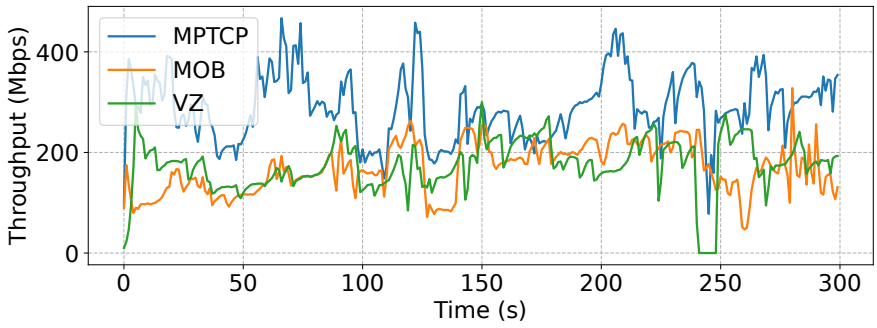
Figure 6.10: Single-path TCP and MPTCP data download performance.

Note that we opt for UDP data instead of TCP data to emulate the available bandwidth at each timestamp and avoid the impact of TCP congestion control. For multipath transport, we run a modified version of `iPerf` [54] that supports MPTCP. It opens MPTCP sockets instead of regular TCP sockets. To compare the performance between MPTCP and each single-path transport, we run two types of experiments in the `MpShell` environment: (1) We start two `iPerf` client instances on the client machine and two server instances on the server side. Each client downloads data using one network interface from the `iPerf` server; (2) We start an `iPerf` client with the MPTCP option enabled and a `iPerf` server. The client downloads data from the server using MPTCP.

Results. Figure 6.10 presents the performance of 5-min download tests. The first three boxes represent the single-path TCP transfer results under AT&T, Verizon, and Mobility networks. The next two show the MPTCP results (Mobility + AT&T, Mobility + Verizon) when concurrently using a Starlink and a cellular service. The benefits of MPTCP are clear; it improves the overall download throughput. On average, the bandwidth utilization of the two tested combinations is 81% and 84%, and the improvement over the better path



(a) Mobility and AT&T.



(b) Mobility and Verizon.

Figure 6.11: Throughput traces for single-path TCP and MPTCP data download.

reaches 30% and 66%, respectively.

We also put another two boxes showing the multipath results before tuning the system buffer. Initially, we notice that, with the default buffer sizes, MPTCP has marginal improvements over single-path transfers. In some cases, the throughput collapses to "0", leading to the failure of the `iPerf` test. Therefore, we increase the buffer size to exceed $10\times$ the link's bandwidth-delay product to accommodate such network fluctuations.

Looking into the throughput progression over time in Figure 6.11, we can learn that MPTCP almost always outperforms either single-path transfer, taking advantage of the

bandwidth of the faster path. For example, in Figure 6.11a, between 0-85s, AT&T experiences severe performance degradation, likely due to weak cellular signal strength. With MPTCP, the throughput is maintained at a much higher level. Also, in Figure 6.11b, when both network conditions are favorable at around 50-90s, MPTCP throughput exceeds 300 Mbps which can never be achieved by either network alone.

The current MPTCP experiments are conducted via emulation and we leave developing a MPTCP scheduler for LEO satellite networks and running real MPTCP experiments as future work. The default MPTCP scheduler implemented in the OS (kernel v5.19) is BLEST [117] which optimizes MPTCP send window occupation to avoid transport-layer head-of-line blocking. We envision that, considering the specific usage scenarios and characteristics of the two network types, further improvements can be made to future MPTCP scheduler design, such as reducing throughput fluctuations.

6.7 Summary

We present a measurement study on the performance of Starlink satellite and cellular networks. Rigorous analyses reveal that Starlink outperforms cellular networks in open areas, but suffers from higher packet loss leading to degraded TCP performance. Their complementary characteristics offer potential improvements to network connectivity but require optimizations. We hope that our findings can spur more research to improve LEO satellite networks.

CHAPTER VII

Related Work

We summarize the related work in several categories below.

7.1 Cooperative Vehicular Sensing

Numerous efforts have been made on cooperative vehicular perception. AVR [202] extends two vehicles' vision by wirelessly sharing stereo camera data between them. The See-Through System [190] streams video data directly from a leader vehicle to a follower to enhance the follower driver's visibility. Cooper [104] improves 3D object detection algorithms for sensor sharing but relies on a single-vehicle dataset and simulates cooperative perception by merging data collected by the same vehicle at different timestamps. Arnold *et al.* [86] instead give attention to perception using stationary infrastructure sensors. Various studies also target vehicle-to-vehicle communications [156, 109, 251, 234, 203, 255]. However, existing wireless techniques struggle to support the sharing of raw sensor data at a high frame rate, particularly as the system scales up. Prior works mostly focus on data exchange between two vehicles (*e.g.,* a leader

and a follower) and do not evaluate vehicle-to-vehicle sharing at scale. EMP employs edge servers to aggregate data from vehicles, which allows each vehicle to only upload the data once instead of sharing multiple times to different peer vehicles. We further evaluate its performance with varying numbers of vehicles in the system. Beyond sharing raw sensor data, feature-level and object-level sharing approaches have been explored to save bandwidth and reduce processing complexity. F-Cooper [103] devises a cooperative perception framework based on features extracted from point cloud data. This solution may not save much bandwidth while sacrificing some original information. FusionEye [170] leverages Bipartite Graph to merge objects detected from image data. Rauch *et al.* [207] discuss sharing locally perceived object data and investigate the temporal and spatial alignment for the shared data. Nonetheless, object-level sharing can fail when there are missed objects in the single-view detection since the missed ones will never appear in the combined data. In our work, to retain important sensor details while reducing bandwidth usage, we enable efficient raw sensor data sharing by carefully partitioning the data and prioritizing different portions to be uploaded.

There are many CAV applications that can make use of on-board sensor data such as 2D images and 3D point clouds and benefit from EMP's sharing framework. For example, connected and autonomous driving systems rely on sensor data for object detection [252, 160, 222], object tracking [87, 217, 257], motion prediction [240], and path planning [165]. Sharing sensor information can provide CAVs and roadside infrastructures with a broader understanding of the surroundings, ultimately leading to a safer driving environment [102, 221]. Research has also looked at the attack and defense aspects of cooperative perception [254, 221, 128].

7.2 5G Performance, Power, and QoE Implications

Xu *et al.* [242] conduct a measurement study of a commercial mid-band 5G service in China. Narayanan *et al.* [179] establish baseline performance for the initial 5G commercial deployments (mmWave and mid-band) in the US. Lumos5G [182] focused on mmWave 5G throughput characterization and proposed machine learning models for throughput prediction. Our work considers both mmWave and low-band 5G with a broader range of 5G smartphone models and server locations. A plethora of studies have extensively explored 3G/4G power characteristics and power modeling [201, 219, 253, 200, 138, 105, 210], but 5G remains under-explored. Xu *et al.* [242] conducted a preliminary measurement study to understand mid-band 5G's power consumption and energy efficiency using a software power monitor. In our work, we utilize both a software-based approach and a hardware power monitor, providing a more thorough characterization of 5G power consumption for mmWave and low-band 5G and comparing it to existing 4G. Besides, we model data transfer power considering factors such as signal strength and throughput. For mobile applications, 4G video delivery has been widely investigated in the literature [244, 231, 163, 264]. Han *et al.* [129] demonstrate streaming volumetric video over mmWave 5G under line-of-sight conditions. Efforts have been made to evaluate different ABR algorithms for HTTP adaptive streaming [174, 84, 167, 246]. Meanwhile, most previous studies focus on understanding and improving web page loading over legacy 3G/4G networks [199, 98, 241]. Narayanan *et al.* [179] experiment with different HTTP versions and encryption configurations using mmWave 5G. Xu *et al.* [242] perform a preliminary study of UHD panoramic video telephony and examine the downloading and rendering performance with various types of websites using mid-band 5G. In our work, we perform a comprehensive exam-

ination of the performance and energy consumption for streaming videos using existing ABR algorithms and loading top websites over 5G. Furthermore, based on the insights, we propose intelligent interface selection schemes to satisfy different QoE goals.

7.3 QUIC Characterization

Since its advent at Google in 2013, QUIC has been extensively researched in the literature. Google presented their experience with QUIC after years of Internet-wide deployment. Carlucci *et al.* [100] examined an early version of QUIC (v21) and showed its superior performance over TCP. Likewise, Megyesi *et al.* [176] compared QUIC with SPDY [237] and HTTP/1.1 and highlighted QUIC's performance improvements. QUIC's rapid evolution has led to efforts investigating the interoperability across QUIC implementations [220, 175, 146]. Longitudinal studies, such as by Kakhki *et al.* [150] and Piraux *et al.* [197], traced QUIC's progression over time. Rüth *et al.* [212] took a first look at QUIC deployment and usage at an early stage. Similarly, QScanner [267] is implemented to analyze early QUIC deployments. There is also a rich tapestry of research diving into the impact of QUIC on various applications including videos streaming [194, 243], web browsing [239, 213], and a mix of different workloads [224, 250], and on various platforms including mobile [122] and satellite communication [155, 115]. None of these works specifically focus on IETF QUIC's performance on major applications over high-speed networks as we do.

With various measurement studies on different aspects of QUIC, some efforts have been made to optimize QUIC from the industry. Through presentations and blogs, companies like Google [142], Cloudflare [28], and Fastly [35] reported their progress on optimiz-

ing QUIC. Some of their findings such as using UDP GSO [21] are relevant to ours. However, all the above works are concerned with the server-side performance. Since downlink traffic dominates the overall server-side traffic, they naturally focus on optimizing data transmission performance. Furthermore, some of their measurement methodologies are not very realistic or not documented in details, making it difficult to reproduce the experiments. For example, Cloudflare [28] made some attempts in accelerating sending data over QUIC including using `sendmmsg` and UDP GSO, but both the client and server run on the same host (a laptop) instead of a production environment. Fastly [35] attributed the high computational cost of QUIC to ACK processing and per-packet sender overhead. However, they also focus on the sender-side optimization and use a simple setting where the CPU's clock is limited at only 400 MHz in order to measure the throughput sustainable with all available computational power. Red Hat [53] also mentioned enabling optional TSO/GSO support in a new release but omitted a performance examination on QUIC. In contrast, we investigate the receiver-side performance bottleneck through rigorous measurements and profiling on multiple dimensions (network traces, OS kernel, QUIC runtime, and higher-layer applications). Besides, we identify new challenges in making offloading solutions harmonize with QUIC.

7.4 Understanding LEO Satellite Networks

Compared to widely deployed mobile networks like 4G/5G [139, 185], LEO satellite networks are relatively new and have not been extensively studied in large-scale commercial deployments. Some works take the first step to evaluate the performance of LEO satellite networks. Michel *et al.* [177] compare Starlink with SatCom and a wired net-

work. Kassem *et al.* [151] analyze Starlink connectivity from a browser-side view. Ma *et al.* [172] present initial measurement results on Starlink's network characteristics. Li *et al.* [166] evaluate the network impact of Starlink's self-driving for its satellites. Our study differs from these, focusing on Starlink's network performance under mobility and its comparison with cellular networks.

Other research has offered insights into future LEO satellite networks and the integration of satellite and terrestrial networks [158, 133, 171]. Some focus on routing [95, 233, 131] and network topology design [261, 225, 94], as well as handovers [85]. L2D2 [232] is a satellite ground station design for low-latency downlink transmission. Tools and platforms are proposed for evaluating LEO satellite networks [85, 159, 152]. In our paper, we explore the potential of combining Starlink and cellular networks for better network performance.

CHAPTER VIII

Conclusion and Future Work

8.1 Conclusion

Recent years have witnessed explosive growth in innovations in network infrastructures and applications. For example, connected and autonomous vehicles aim to improve road safety and driving efficiency. 5G promises to deliver ultra-high throughput, low latency, and large capacity. QUIC is expected to replace the original TCP stack to enhance the Web experience. LEO satellite networks strive to ensure stable connectivity anytime, anywhere. However, these advancements also bring numerous unknowns. Blindly running new network applications on existing networks or running existing applications on top of new network infrastructures may not yield expected performance. This dissertation is dedicated to addressing this challenge. We demonstrate that systematic measurements and analyses aimed at unveiling the intricacies of emerging network infrastructures, along with the development and innovation of efficient network applications, hold the key to unlocking the full potential of the next-generation network ecosystem.

Specifically, we start with investigating and improving the performance of connected

and autonomous vehicles. We identify key limitations in single-vehicle perception capabilities. With the idea of sensor data sharing in mind, we further realize the need for an edge node to process shared sensor data. To address the challenges of system scalability, we leverage Voronoi Diagrams to optimize data sharing, balancing data quality and size. We design and implement EMP, an edge-assisted multi-vehicle collaboration framework for perception enhancement with key components including sensor data partitioning, scheduling, and merging. In order to better evaluate multi-vehicle perception, we collect the first synthetic multi-vehicle LiDAR dataset using a video game containing realistic scenes. Extensive experiments demonstrate that EMP outperforms naïve (no data partitioning) and vehicle-to-vehicle sharing schemes under different vehicle and network settings, with low overhead. A case study of driving scenarios with road hazards showcases EMP benefits in improving road safety.

Next, we study the network performance, power characteristics, and application implications of commercial 5G networks. We examine 5G from several dimensions, including 5G carriers, deployment schemes, frequency bands, mobility patterns, user devices, and upper-layer applications. We compare 5G results with those measured over 4G/LTE. We build various measurement tools including 5G Tracker and RRC-Probe to facilitate data collection. Key findings include but are not limited to: 5G provides satisfactory performance and has gained noticeable improvements since its debut; its performance is impacted by geographical properties, calling for the deployment of edge nodes; During data transfer, 5G's absolute power consumption is higher than 4G but is more power-efficient; downlink transfer is more efficient than uplink; adaptation of ABR algorithms can boost their performance over 5G; and web browsing can benefit from intelligent selection be-

tween 4G and 5G. Overall, our study highlights the challenges posed by the unique characteristics of 5G and demonstrates potential ways for leveraging these features to improve user experience.

Moving forward, we look into the performance of the QUIC transport protocol. Given numerous existing works providing findings of both performance gains and degradations compared to other protocol stacks, we advocate performance examinations in the context of specific services and setups. In our scope, we consider a relatively under-explored yet important scenario, running QUIC over high-speed networks. Our experiments reveal the gap between QUIC's design goal and its actual performance. QUIC is not "quick" enough — its slowness is prevalent across data transfer tools and production browsers, affecting various web applications including file download, video content delivery, and web page loading, though to different extents. We perform rigorous analysis with packet traces and kernel/user space profiling to find the root cause. The results indicate that QUIC incurs higher overhead when processing packets on the receiver side and fails to benefit from existing NIC offloading mechanisms as TCP does. We make several recommendations including the adoption and adaptation of offloading solutions and receiver-side QUIC optimization, to mitigate the performance issues and unleash its potential.

To validate the promise of LEO satellite networks' performance stability and accessibility, we carry out a large-scale inter-state data collection campaign for Starlink satellite networks and cellular networks. The results show that Starlink suffers from elevated packet loss, leading to lower TCP throughput than UDP, while the latency stays similar. A comparison between two Starlink configurations (both designed for outdoor use) indicates that the less expensive one may already meet average applications' requirements in the wild.

User's moving speed is negligible considering LEO satellites' ultra-high speed operations in space. Notably, the two networks exhibit totally different coverage. Cellular networks offer better performance in urban areas, while LEO networks win in suburban and rural areas given their design and deployment nature. This further led us to explore the potential of combining the two types of networks for an always-on experience.

In summary, EMP represents the first framework for multi-vehicle perception with edge assistance. It is expected to enable or boost various cooperative sensing applications involving multiple vehicles. Our findings on commercial 5G reveal the current state of the 5G ecosystem. We have seen follow-ups on understanding the evolution of 5G over the years as well as applications of our proposed algorithms for 5G-specific application adaptation. With the excessive receiver-side processing overhead identified as the major contributor to QUIC's performance issues over fast Internet, and considering the increasing adoption of QUIC, we call for more efforts to explore and refine QUIC, and more broadly, upper-layer protocols. As LEO satellite networks are getting hot, we hope our comparative study of Starlink and cellular networks can inspire further research to improve the functionality and integration of LEO satellite networks with terrestrial networks. We have made our system prototype, datasets, and measurement tools [48, 62, 82, 78] publicly available to support ongoing research in all four areas covered in this dissertation.

8.2 Limitations and Future Work

While this dissertation has provided valuable insights and solutions into characterizing and improving next-generation network infrastructures and applications, a number of directions that are worth exploration remain. We summarize them as follows and encourage

future efforts for further investigation.

- **Cooperative vehicular perception.** Future work can explore improvements in sensor sharing and cooperative perception. EMP's REAP algorithm partitions the sensor data based on vehicle locations and network resources. Following this direction, further accounting for occlusions to adjust the region boundaries may lead to better performance. Besides, vehicle-to-infrastructure based sharing may not always work best. Ideally, a combination of both vehicle-to-infrastructure and vehicle-to-vehicle sensor sharing schemes with dynamic adaptation may benefit more different driving scenarios, environments, weather conditions, and CAV and infrastructure deployment stages. Additionally, object detection algorithms specifically tailored (built and trained) for multi-vehicle sensor data can potentially outperform current general-purpose algorithms made for sensor data collected by single vehicles.

- **5G networks.** We have conducted a comprehensive measurement study on commercial 5G, which provides an overall idea of what 5G currently looks like in terms of network performance, power, and application QoE, compared to 4G/LTE. With the findings revealed, future work can further dig into specific aspects seeking improvements. For example, with our 5G power model, one can build a power consumption prediction module to intelligently wake up or shut down the UE's 5G interface based on instant power saving needs. The application logic can also be adjusted to balance QoE and power consumption. Moreover, taking into account 5G's unique characteristics such as high and fluctuating throughput, video adaptation and web page loading strategies should be redesigned to work better under 5G or even the coexistence of 4G, 5G, and WiFi.

- **QUIC.** As emphasized earlier, it is important to examine a transport protocol "in context". It is hard to generalize the findings obtained from a specific service and setup, for example, high-speed Internet, in our case. Consider the following scenarios: data center networks supporting ultra high bandwidth for large volumes of data traffic versus networks connecting IoT devices, and continuous application traffic versus intermittent traffic. The performance of QUIC, as well as the comparison with other protocols, can be totally different and awaits exploration.

- **LEO satellite networks.** Our research primarily focuses on understanding Starlink's basic network performance compared with cellular networks, their coverage, and potential combination. However, there is still a long way to go for LEO satellite networking research. For example, the handover mechanism in LEO satellite networks is entirely different from terrestrial cellular networks that rely on the deployment of base stations. Also, this will be a black-box study since the operator does not release any design or runtime details. Our initial experiments and analyses have shown that satellite handover exhibits strong periodicity and predictability. Future work can look at how to predict satellite handover events and prepare different layers (*e.g.,* transport, application) for the incoming handover, which potentially leads to temporal disruption. Cross-layer adaptation can improve application performance.

It is crucial to integrate measurement findings with system design to fully harness the potential of next-generation network infrastructures. This dissertation highlights the importance of systematic measurements and analyses in understanding the intricacies of emerging network technologies and applications. By identifying key performance limitations and proposing innovative solutions, we advocate for a balanced approach that

combines empirical analysis with practical design. The lessons learned from this research provide a robust foundation for future advancements, promoting the development of a more resilient, efficient, and intelligent network ecosystem.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] 5G availability: The impact on CSPs and customers - Ericsson. https://www.ericsson.com/en/reports-and-papers/consumerlab/reports/5g-next-wave.

[2] Autopilot - Tesla. https://www.tesla.com/autopilot.

[3] Forecast number of mobile 5G subscriptions worldwide by region from 2019 to 2026. https://www.statista.com/statistics/521598/5g-mobile-subscriptions-worldwide/".

[4] Honda Global - Automated Drive / Advanced Driver Assistance System. https://global.honda/innovation/automated-drive.html.

[5] Jonathan's Space Pages: Enormous ('Mega') Satellite Constellations. https://planet4589.org/space/con/conlist.html.

[6] Starlink. https://www.starlink.com.

[7] cURL - command line tool and library for transferring data with URLs. https://curl.se/, 1998.

[8] Linux Traffic Control (tc). https://man7.org/linux/man-pages/man8/tc.8.html, 2001.

[9] HAR 1.2 Spec. http://www.softwareishard.com/blog/har-12-spec/, 2007.

[10] HTTP Archive (HAR) format. https://w3c.github.io/web-performance/specs/HAR/Overview.html, 2012.

[11] Chromium Blog: Experimenting with QUIC. https://blog.chromium.org/2013/06/experimenting-with-quic.html, 2013.

[12] Vehicle Stopping Distance and Time. https://nacto.org/docs/usdg/vehicle_stopping_distance_and_time_upenn.pdf, 2013.

[13] Chromium Blog: A QUIC update on Google's experimental transport. https://blog.chromium.org/2015/04/a-quic-update-on-googles-experimental.html, 2015.

[14] Optimal Adaptive Streaming Formats MPEG-DASH & HLS Segment Length. https://bitmovin.com/mpeg-dash-hls-segment-length/, 2015.

[15] 152,000 Smart Devices Every Minute In 2025: IDC Outlines The Future of Smart Things. https://www.forbes.com/sites/michaelkanellos/2016/03/03/152000-smart-devices-every-minute-in-2025-idc-outlines-the-future-of-smart-things/, 2016.

[16] Annual Traffic Report - WSDOT. https://www.wsdot.wa.gov/mapsdata/travel/pdf/Annual_Traffic_Report_2016.pdf, 2016.

[17] Real 4K HDR 60fps: LG Jazz HDR UHD (Chromecast Ultra). https://www.youtube.com/watch?v=mkggXE5e2yk, 2016.

[18] Cobra Kai Ep 1 – "Ace Degenerate" – The Karate Kid Saga Continues. https://www.youtube.com/watch?v=_rB36UGoP4Y, 2018.

[19] How autonomous vehicles could save over 350K lives in the US and millions worldwide. https://zdnet.com/article/how-autonomous-vehicles-could-save-over-350k-lives-in-the-us-and-millions-worldwide/, 2018.

[20] Study shows autonomous vehicles can help improve traffic flow. https://phys.org/news/2018-02-autonomous-vehicles-traffic.html, 2018.

[21] UDP GSO. https://lwn.net/Articles/752956/, 2018.

[22] AT&T integrating 5G with Microsoft cloud to enable next-generation solutions on the edge. https://news.microsoft.com/2019/11/26/att-integrating-5g-with-microsoft-cloud-to-enable-next-generation-solutions-on-the-edge/, 2019.

[23] ETSI TR 103 559: Speech and multimedia Transmission Quality (STQ); Best practices for robust network QoS benchmark testing and scoring. https://www.etsi.org/deliver/etsi_tr/103500_103599/103559/01.01.01_60/tr_103559v010101p.pdf, 2019.

[24] FFmpeg Project. http://ffmpeg.org/, 2019.

[25] Snapdragon X50 5G modem-RF system. https://www.qualcomm.com/products/snapdragon-x50-5g-modem, 2019.

[26] Verizon and AWS announce 5G Edge computing partnership. https://techcrunch.com/2019/12/03/verizon-and-aws-announce-5g-edge-computing-partnership/, 2019.

[27] Waymo. https://waymo.com/, 2019.

[28] Accelerating UDP packet transmission for QUIC. https://blog.cloudflare.com/accelerating-udp-packet-transmission-for-quic/, 2020.

[29] Apollo. https://apollo.auto/, 2020.

[30] Chromium Blog: Chrome is deploying HTTP/3 and IETF QUIC. https://blog.chromium.org/2020/10/chrome-is-deploying-http3-and-ietf-quic.html, 2020.

[31] Ericsson Mobility Report. https://www.ericsson.com/49da93/assets/local/mobility-report/documents/2020/june2020-ericsson-mobility-report.pdf, 2020.

[32] How Facebook is bringing QUIC to billions. https://engineering.fb.com/2020/10/21/networking-traffic/how-facebook-is-bringing-quic-to-billions/, 2020.

[33] INRIX Global Traffic Scorecard - Last-Mile Speed. https://inrix.com/scorecard/, 2020.

[34] Microsoft partners with the industry to unlock new 5G scenarios with Azure Edge Zones. https://azure.microsoft.com/en-us/blog/microsoft-partners-with-the-industry-to-unlock-new-5g-scenarios-with-azure-edge-zones/, 2020.

[35] QUIC vs TCP: Which is Better? https://www.fastly.com/blog/measuring-quic-vs-tcp-computational-efficiency, 2020.

[36] Snapdragon 765G 5G Mobile Platform. https://www.qualcomm.com/products/snapdragon-765g-5g-mobile-platform, 2020.

[37] Snapdragon X55 5G modem-RF system. https://www.qualcomm.com/products/snapdragon-x55-5g-modem, 2020.

[38] Snow and Ice Pose a Vexing Obstacle for Self-Driving Cars. https://www.wired.com/story/snow-ice-pose-vexing-obstacle-self-driving-cars/, 2020.

[39] Unprotected Turns - The Right Way To Navigate Complex Intersections. https://www.epermittest.com/drivers-education/unprotected-turns, 2020.

[40] What's the Best Bitrate for the Best Video Quality on YouTube? (1080p, 1440p, 4K). https://www.youtube.com/watch?v=0fz479id_Ic, 2020.

[41] 3GPP TS 36.331: Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification (V16.3.0), 2021.

[42] 3GPP TS 38.331: NR; Radio Resource Control (RRC); Protocol specification (V16.3.1), 2021.

[43] Autoware. https://www.autoware.org/, 2021.

[44] Chrome HAR capturer. https://github.com/cyrus-and/chrome-har-capturer, 2021.

[45] Chrome HAR capturer. https://github.com/cyrus-and/chrome-har-capturer, 2021.

[46] Dash-Industry-Forum, dash.js. https://github.com/Dash-Industry-Forum/dash.js, 2021.

[47] Draco 3D Graphics Compression. https://google.github.io/draco/, 2021.

[48] Emp github repository. https://github.com/Shawnxm/EMP, 2021.

[49] FCC Mobile Broadband Maps. https://tinyurl.com/2ayv5rz6, 2021.

[50] High performance INS for ADAS and autonomous vehicle testing. https://www.oxts.com/products/rt3000/, 2021.

[51] HTTP/2 vs HTTP/3: A comparison. https://ably.com/topic/http-2-vs-http-3, 2021.

[52] HTTP/3 and QUIC: Past, Present, and Future. https://www.akamai.com/blog/performance/http3-and-quic-past-present-and-future, 2021.

[53] Improve UDP performance in RHEL 8.5. https://developers.redhat.com/articles/2021/11/05/improve-udp-performance-rhel-85, 2021.

[54] iPerf GitHub repository - Add MPTCP support with the –multipath flag. https://github.com/esnet/iperf/pull/1166/commits, 2021.

[55] Lidar — Wikipedia. https://en.wikipedia.org/wiki/Lidar, 2021.

[56] Linux CUBIC TCP. https://github.com/torvalds/linux/blob/master/net/ipv4/tcp_cubic.c, 2021.

[57] Linux Perf. https://man7.org/linux/man-pages/man1/perf.1.html, 2021.

[58] Linux Temporary Filesystem (tmpfs). https://man7.org/linux/man-pages/man5/tmpfs.5.html, 2021.

[59] Monsoon power monitor. https://www.msoon.com/LabEquipment/PowerMonitor/, 2021.

[60] Netflix recommended network bandwidth for 4K video. https://help.netflix.com/en/node/306, 2021.

[61] Netgear nighthawk x10 ad7200 smart wifi router (r9000). https://www.netgear.com/home/wifi/routers/ad7200-fastest-router/, 2021.

[62] SIGCOMM21-5G/artifact GitHub repository. https://github.com/SIGCOMM21-5G/artifact, 2021.

[63] Velodyne LiDAR HDL-64E. https://www.velodynelidar.com/hdl-64e.html, 2021.

[64] Verizon Hyper Precise Location. https://thingspace.verizon.com/services/hyper-precise-location/, 2021.

[65] YouTube 4K bitrates enconding. https://support.google.com/youtube/answer/1722171, 2021.

[66] Accelerating Rural Connectivity. https://tinyurl.com/56jk4bx7, 2022.

[67] Catapult - Web Page Replay. https://chromium.googlesource.com/catapult/+/HEAD/web_page_replay_go/, 2022.

[68] Fiddler - Web Debugging Proxy and Troubleshooting Solutions. https://www.telerik.com/fiddler, 2022.

[69] GitHub - litespeedtech/lsquic: LiteSpeed QUIC and HTTP/3 Library. https://github.com/litespeedtech/lsquic, 2022.

[70] HTTP RFCs have evolved: A Cloudflare view of HTTP usage trends. https://blog.cloudflare.com/cloudflare-view-http3-usage/, 2022.

[71] Internet Traffic and Capacity Remain Brisk. https://blog.telegeography.com/internet-traffic-and-capacity-remain-brisk, 2022.

[72] SiteSucker. https://ricks-apps.com/osx/sitesucker/index.html, 2022.

[73] Starlink has hit more than 1 million users despite a drop in download speeds. Here's what you need to know about the service. https://www.businessinsider.com/spacex-starlink-internet-service-elon-musk-all-you-need-know-2021-2, 2022.

[74] Netgear's first Wi-Fi 7 Nighthawk router trades in the batwing look for a tower design. https://www.theverge.com/2023/3/14/23638347/netgear-nighthawk-rs700-wifi-7-router, 2023.

[75] NGINX QUIC. https://quic.nginx.org/, 2023.

[76] OpenLiteSpeed. https://openlitespeed.org/, 2023.

[77] Starlink Specifications. https://www.starlink.com/legal/documents/DOC-1400-28829-70, 2023.

[78] Starlink-vs-Cellular GitHub repository. https://github.com/Starlink-Project/Satellite-vs-Cellular, 2023.

[79] The Chromium Projects. https://www.chromium.org/Home/, 2023.

[80] The Chromium Projects - Network Service. https://chromium.googlesource.com/chromium/src/+/HEAD/services/network/, 2023.

[81] UMich research center receives $15 million grant to study autonomous vehicles. https://www.michigandaily.com/news/umich-research-center-receives-15-million-grant-to-study-autonomous-vehicles/, 2023.

[82] QUIC-not-Quick GitHub repository. https://github.com/Shawnxm/QUIC-not-Quick, 2024.

[83] 3rd Generation Partnership Project. Release 15. https://www.3gpp.org/release-15, April 2019.

[84] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang. Oboe: Auto-tuning video abr algorithms to network conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 44–58, 2018.

[85] A. Al-Hourani. A tractable approach for predicting pass duration in dense satellite networks. *IEEE Communications Letters*, 25(8):2698–2702, 2021.

[86] E. Arnold, M. Dianati, R. de Temple, and S. Fallah. Cooperative perception for 3d object detection in driving scenarios using infrastructure sensors. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):1852–1864, 2020.

[87] A. Asvadi, P. Girao, P. Peixoto, and U. Nunes. 3d object tracking using rgb and lidar data. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1255–1260. IEEE, 2016.

[88] M. Attaran. The impact of 5g on the evolution of intelligent automation and industry digitization. *Journal of ambient intelligence and humanized computing*, 14(5):5977–5993, 2023.

[89] F. Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing*, 16(1):78–96, 1987.

[90] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[91] F. Aurenhammer, R. Klein, and D.-T. Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company, 2013.

[92] A. Basu, J. Gandhi, J. Chang, M. D. Hill, and M. M. Swift. Efficient virtual memory for big memory servers. *ACM SIGARCH Computer Architecture News*, 41(3):237–248, 2013.

[93] M. Belshe, R. Peon, and M. Thomson. Hypertext transfer protocol version 2 (http/2). *RFC 7540, IETF*, 2015.

[94] D. Bhattacherjee and A. Singla. Network topology design at 27,000 km/hour. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, pages 341–354, 2019.

[95] V. Bhosale, A. Saeed, K. Bhardwaj, and A. Gavrilovska. A characterization of route variability in leo satellite networks. In *International Conference on Passive and Active Network Measurement*, pages 313–342. Springer, 2023.

[96] M. Bishop. Http/3. *RFC 9114, IETF*, 2022.

[97] E. Bocchi, L. De Cicco, and D. Rossi. Measuring the quality of experience of web users. *ACM SIGCOMM Computer Communication Review*, 46(4):8–13, 2016.

[98] D. H. Bui, Y. Liu, H. Kim, I. Shin, and F. Zhao. Rethinking energy-performance trade-off in mobile web page loading. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 14–26, 2015.

[99] M. Butkiewicz, H. V. Madhyastha, and V. Sekar. Understanding website complexity: measurements, metrics, and implications. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 313–328, 2011.

[100] G. Carlucci, L. De Cicco, and S. Mascolo. Http over udp: an experimental investigation of quic. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 609–614, 2015.

[101] R. Chataut and R. Akl. Massive mimo systems for 5g and beyond networks—overview, recent trends, challenges, and future research direction. *Sensors*, 20(10):2753, 2020.

[102] H. Chen, B. Liu, X. Zhang, F. Qian, Z. M. Mao, and Y. Feng. A cooperative perception environment for traffic operations and control. *arXiv preprint arXiv:2208.02792*, 2022.

[103] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and S. Fu. F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 88–100, 2019.

[104] Q. Chen, S. Tang, Q. Yang, and S. Fu. Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 514–524. IEEE, 2019.

[105] X. Chen, N. Ding, A. Jindal, Y. C. Hu, M. Gupta, and R. Vannithamby. Smartphone energy drain in the wild: Analysis and implications. *ACM SIGMETRICS Performance Evaluation Review*, 43(1):151–164, 2015.

[106] J. Choi, V. Va, N. Gonzalez-Prelcic, R. Daniels, C. R. Bhat, and R. W. Heath. Millimeter-wave vehicular communication to support massive automotive sensing. *IEEE Communications Magazine*, 54(12):160–167, 2016.

[107] A. Croitoru, D. Niculescu, and C. Raiciu. Towards wifi mobility without fast handover. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 219–234, 2015.

[108] S. Cui, A. J. Goldsmith, and A. Bahai. Energy-efficiency of mimo and cooperative mimo techniques in sensor networks. *IEEE Journal on selected areas in communications*, 22(6):1089–1098, 2004.

[109] T. Das, L. Chen, R. Kundu, A. Bakshi, P. Sinha, K. Srinivasan, G. Bansal, and T. Shimizu. Corecast: Collision resilient broadcasting in vehicular networks. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 217–229, 2018.

[110] W. de Bruijn and E. Dumazet. Optimizing udp for content delivery: Gso, pacing and zerocopy. In *Linux Plumbers Conference*, 2018.

[111] Q. De Coninck, F. Michel, M. Piraux, F. Rochet, T. Given-Wilson, A. Legay, O. Pereira, and O. Bonaventure. Pluginizing quic. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 59–74. 2019.

[112] H. Deng, T. Birdal, and S. Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–205, 2018.

[113] S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan. Wifi, lte, or both? measuring multi-homed wireless internet performance. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 181–194, 2014.

[114] N. Ding, D. Wagner, X. Chen, A. Pathak, Y. C. Hu, and A. Rice. Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. *ACM SIGMETRICS Performance Evaluation Review*, 41(1):29–40, 2013.

[115] S. Endres, J. Deutschmann, K.-S. Hielscher, and R. German. Performance of quic implementations over geostationary satellite links. *arXiv preprint arXiv:2202.08228*, 2022.

[116] G. Fairhurst, T. Jones, M. Tüxen, I. Rüngeler, and T. Völker. Packetization layer path mtu discovery for datagram transports. *RFC 8899, IETF*, 2020.

[117] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli. Blest: Blocking estimation-based mptcp scheduler for heterogeneous networks. In *2016 IFIP networking conference (IFIP networking) and workshops*, pages 431–439. IEEE, 2016.

[118] R. Fielding, M. Nottingham, and J. Reschke. Http semantics. *RFC 9110, IETF*, 2022.

[119] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[120] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina. Network slicing in 5g: Survey and challenges. *IEEE communications magazine*, 55(5):94–100, 2017.

[121] X. Foukas and B. Radunovic. Concordia: Teaching the 5g vran to share compute. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 580–596, 2021.

[122] A. Ganji and M. Shahzad. Characterizing the performance of quic on android and wear os devices. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pages 1–11. IEEE, 2021.

[123] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[124] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5545–5554, 2019.

[125] S. R. Group. A Global Perspective of 5G Network Performance. `https://www.qualcomm.com/media/documents/files/signals-research-group-s-5g-benchmark-study.pdf`, 2019.

[126] Y. E. Guo, A. Nikravesh, Z. M. Mao, F. Qian, and S. Sen. Accelerating multipath transport through balanced subflow completion. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 141–153, 2017.

[127] H. Gupta, M. Curran, J. Longtin, T. Rockwell, K. Zheng, and M. Dasari. Cyclops: an fso-based wireless link for vr headsets. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 601–614, 2022.

[128] R. S. Hallyburton, D. Hunt, S. Luo, and M. Pajic. A multi-agent security testbed for the analysis of attacks and defenses in collaborative sensor fusion. *arXiv preprint arXiv:2401.09387*, 2024.

[129] B. Han, Y. Liu, and F. Qian. Vivo: Visibility-aware mobile volumetric video streaming. In *Proceedings of the 26th annual international conference on mobile computing and networking*, pages 1–13, 2020.

[130] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan. Mp-dash: Adaptive video streaming over preference-aware multipath. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 129–143, 2016.

[131] M. Handley. Using ground relays for low-latency wide-area routing in megaconstellations. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, pages 125–132, 2019.

[132] K. Heimann, J. Tiemann, D. Yolchyan, and C. Wietfeld. Experimental 5g mmwave beam tracking testbed for evaluation of vehicular communications. In *2019 IEEE 2nd 5G World Forum (5GWF)*, pages 382–387, Sep. 2019.

[133] S. Heine, F. Völk, R. T. Schwarz, and A. Knopp. Concept and evaluation of 5g backhauling via starlink. In *39th International Communications Satellite Systems Conference (ICSSC 2022)*, volume 2022, pages 69–75. IET, 2022.

[134] F. Hilal and O. Gasser. Yarrpbox: Detecting middleboxes at internet-scale. *Proceedings of the ACM on Networking*, 1(CoNEXT1):1–23, 2023.

[135] A. Hillbur. 5G deployment options to reduce the complexity. `https://www.ericsson.com/en/blog/2018/11/5g-deployment-options-to-reduce-the-complexity`, 2018.

[136] B. Hu, X. Zhang, Q. Zhang, N. Varyani, Z. M. Mao, F. Qian, and Z.-L. Zhang. Leo satellite vs. cellular networks: Exploring the potential for synergistic integration. In *Companion of the 19th International Conference on emerging Networking EXperiments and Technologies*, pages 45–51, 2023.

[137] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, 11(11):1–16, 2015.

[138] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 225–238, 2012.

[139] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck. An in-depth study of lte: Effect of network protocol and application behavior on performance. *ACM SIGCOMM Computer Communication Review*, 43(4):363–374, 2013.

[140] T.-Y. 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 the 2014 ACM conference on SIGCOMM*, pages 187–198, 2014.

[141] B. Hurl, K. Czarnecki, and S. Waslander. Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2522–2529. IEEE, 2019.

[142] S. Ian. As QUIC as TCP, Optimizing QUIC and HTTP/3 CPU Usage - EPIQ 2020. `https://conferences.sigcomm.org/sigcomm/2020/workshop-epiq.html`, 2020.

[143] P. A. Iannucci and T. E. Humphreys. Fused low-earth-orbit gnss. *IEEE Transactions on Aerospace and Electronic Systems*, 2022.

[144] J. Iyengar, I. Swett, and M. Kühlewind. QUIC Acknowledgement Frequency. *IETF*, 2023. Work in Progress.

[145] J. Iyengar, M. Thomson, et al. Quic: A udp-based multiplexed and secure transport. *RFC 9000, IETF*, 2021.

[146] B. Jaeger, J. Zirngibl, M. Kempf, K. Ploch, and G. Carle. Quic on the highway: Evaluating performance on high-rate links. In *2023 IFIP Networking Conference (IFIP Networking)*, pages 1–9, 2023.

[147] N. Jardak and R. Adam. Practical use of starlink downlink tones for positioning. *Sensors*, 23(6):3234, 2023.

[148] J. Jiang, V. Sekar, and H. Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 97–108, 2012.

[149] S. Jin, R. Zhu, A. Hassan, X. Zhu, X. Zhang, Z. M. Mao, F. Qian, and Z.-L. Zhang. Oasis: Collaborative neural-enhanced mobile video streaming. In *Proceedings of the 15th ACM Multimedia Systems Conference*, pages 45–55, 2024.

[150] A. M. Kakhki, S. Jero, D. Choffnes, C. Nita-Rotaru, and A. Mislove. Taking a long look at quic: an approach for rigorous evaluation of rapidly evolving transport protocols. In *proceedings of the 2017 internet measurement conference*, pages 290–303, 2017.

[151] M. M. Kassem, A. Raman, D. Perino, and N. Sastry. A browser-side view of starlink connectivity. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pages 151–158, 2022.

[152] S. Kassing, D. Bhattacherjee, A. B. Águas, J. E. Saethre, and A. Singla. Exploring the" internet from space" with hypatia. In *Proceedings of the ACM Internet Measurement conference*, pages 214–229, 2020.

[153] J. B. Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, 2011.

[154] W. Kim. Experimental demonstration of mmwave vehicle-to-vehicle communications using ieee 802.11 ad. *Sensors*, 19(9):2057, 2019.

[155] M. Kosek, H. Cech, V. Bajpai, and J. Ott. Exploring proxying quic and http/3 for satellite communication. In *2022 IFIP Networking Conference (IFIP Networking)*, pages 1–9. IEEE, 2022.

[156] S. Kumar, L. Shi, N. Ahmed, S. Gil, D. Katabi, and D. Rus. Carspeak: a content-centric network for autonomous driving. *ACM SIGCOMM Computer Communication Review*, 42(4):259–270, 2012.

[157] A. Lacy, R. Bravo, A. Otero-Piñeiro, R. Pena, F. De Lacy, R. Menchaca, and J. Balibrea. 5g-assisted telementored surgery. *British Journal of Surgery*, 106(12):1576–1579, 2019.

[158] Z. Lai, H. Li, Y. Deng, Q. Wu, J. Liu, Y. Li, J. Li, L. Liu, W. Liu, and J. Wu. {StarryNet}: Empowering researchers to evaluate futuristic integrated space and terrestrial networks. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1309–1324, 2023.

[159] Z. Lai, H. Li, and J. Li. Starperf: Characterizing network performance for emerging mega-constellations. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pages 1–11. IEEE, 2020.

[160] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[161] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, et al. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the conference of the ACM special interest group on data communication*, pages 183–196, 2017.

[162] K. Larsson, B. Halvarsson, D. Singh, R. Chana, J. Manssour, M. Na, C. Choi, and S. Jo. High-speed beam tracking demonstrated using a 28 ghz 5g trial system. In *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, Sep. 2017.

[163] J. Lee, S. Lee, J. Lee, S. D. Sathyanarayana, H. Lim, J. Lee, X. Zhu, S. Ramakrishnan, D. Grunwald, K. Lee, et al. Perceive: deep learning-based cellular uplink prediction using real-time scheduling patterns. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pages 377–390, 2020.

[164] K. Lee, J. Yi, Y. Lee, S. Choi, and Y. M. Kim. Groot: a real-time streaming system of high-fidelity volumetric videos. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[165] J. Li, G. Deng, C. Luo, Q. Lin, Q. Yan, and Z. Ming. A hybrid path planning method in unmanned air/ground vehicle (uav/ugv) cooperative systems. *IEEE Transactions on Vehicular Technology*, 65(12):9585–9596, 2016.

[166] Y. Li, H. Li, W. Liu, L. Liu, W. Zhao, Y. Chen, J. Wu, Q. Wu, J. Liu, Z. Lai, et al. A networking perspective on starlink's self-driving leo mega-constellation. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2023.

[167] M. Licciardello, M. Grüner, and A. Singla. Understanding video streaming algorithms in the wild. In *Passive and Active Measurement: 21st International Conference, PAM 2020, Eugene, Oregon, USA, March 30–31, 2020, Proceedings 21*, pages 298–313. Springer, 2020.

[168] Y.-s. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens. Ecf: An mptcp path scheduler to manage heterogeneous paths. In *Proceedings of the 13th international conference on emerging networking experiments and technologies*, pages 147–159, 2017.

[169] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars. The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 751–766, 2018.

[170] H. Liu, P. Ren, S. Jain, M. Murad, M. Gruteser, and F. Bai. Fusioneye: Perception sharing for connected vehicles and its bandwidth-accuracy trade-offs. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2019.

[171] M. López, S. B. Damsgaard, I. Rodríguez, and P. Mogensen. An empirical analysis of multi-connectivity between 5g terrestrial and leo satellite networks. In *2022 IEEE Globecom Workshops (GC Wkshps)*, pages 1115–1120. IEEE, 2022.

[172] S. Ma, Y. C. Chou, H. Zhao, L. Chen, X. Ma, and J. Liu. Network characteristics of leo satellite constellations: A starlink-based measurement from end users. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2023.

[173] M. Magnusson, A. Lilienthal, and T. Duckett. Scan registration for autonomous mining vehicles using 3d-ndt. *Journal of Field Robotics*, 24(10):803–827, 2007.

[174] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the conference of the ACM special interest group on data communication*, pages 197–210, 2017.

[175] R. Marx, J. Herbots, W. Lamotte, and P. Quax. Same standards, different decisions: A study of quic and http/3 implementation diversity. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, pages 14–20, 2020.

[176] P. Megyesi, Z. Krämer, and S. Molnár. How quick is quic? In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2016.

[177] F. Michel, M. Trevisan, D. Giordano, and O. Bonaventure. A first look at starlink performance. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pages 130–136, 2022.

[178] G. Naik, B. Choudhury, and J.-M. Park. Ieee 802.11 bd & 5g nr v2x: Evolution of radio access technologies for v2x communications. *IEEE Access*, 7:70169–70184, 2019.

[179] A. Narayanan, E. Ramadan, J. Carpenter, Q. Liu, Y. Liu, F. Qian, and Z.-L. Zhang. A first look at commercial 5g performance on smartphones. In *Proceedings of The Web Conference 2020*, pages 894–905, 2020.

[180] A. Narayanan, E. Ramadan, J. Carpenter, Q. Liu, Y. Liu, F. Qian, and Z.-L. Zhang. A first look at commercial 5g performance on smartphones. In *Proceedings of The Web Conference 2020*, WWW '20, pages 894–905, New York, NY, USA, 2020. Association for Computing Machinery.

[181] A. Narayanan, E. Ramadan, R. Mehta, X. Hu, Q. Liu, R. A. Fezeu, U. K. Dayalan, S. Verma, P. Ji, T. Li, et al. Lumos5g: Mapping and predicting commercial mmwave 5g throughput. In *Proceedings of the ACM Internet Measurement Conference*, pages 176–193, 2020.

[182] A. Narayanan, E. Ramadan, R. Mehta, X. Hu, Q. Liu, R. A. Fezeu, U. K. Dayalan, S. Verma, P. Ji, T. Li, et al. Lumos5g: Mapping and predicting commercial mmwave 5g throughput. In *Proceedings of the ACM Internet Measurement Conference*, pages 176–193, 2020.

[183] A. Narayanan, E. Ramadan, J. Quant, P. Ji, F. Qian, and Z.-L. Zhang. 5g tracker – a crowdsourced platform to enable research using commercial 5g services. In *Proceedings of the ACM SIGCOMM 2020 Conference Posters and Demos*, SIGCOMM

Posters and Demos '20, Virtual Event, USA, 2020. Association for Computing Machinery.

[184] A. Narayanan, E. Ramadan, J. Quant, P. Ji, F. Qian, and Z.-L. Zhang. 5g tracker: a crowdsourced platform to enable research using commercial 5g services. In *Proceedings of the SIGCOMM'20 Poster and Demo Sessions*, pages 65–67. 2020.

[185] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin, Z. Yang, Z. M. Mao, et al. A variegated look at 5g in the wild: performance, power, and qoe implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 610–625, 2021.

[186] J. Nejati and A. Balasubramanian. An in-depth study of mobile browser performance. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1305–1315, 2016.

[187] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan. Mahimahi: Accurate record-and-replay for http. In *Usenix annual technical conference*, pages 417–429, 2015.

[188] A. Nika, Y. Zhu, N. Ding, A. Jindal, Y. C. Hu, X. Zhou, B. Y. Zhao, and H. Zheng. Energy and performance of smartphone radio bundling in outdoor environments. In *Proceedings of the 24th International Conference on World Wide Web*, pages 809–819, 2015.

[189] A. Nikravesh, Y. Guo, X. Zhu, F. Qian, and Z. M. Mao. Mp-h2: a client-only multipath solution for http/2. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.

[190] C. Olaverri-Monreal, P. Gomes, R. Fernandes, F. Vieira, and M. Ferreira. The see-through system: A vanet-enabled assistant for overtaking maneuvers. In *2010 IEEE intelligent vehicles symposium*, pages 123–128. IEEE, 2010.

[191] OOKLA. Speedtest® for Android. `https://www.speedtest.net/apps/android`.

[192] OOKLA. OOKLA 5G Map. `https://www.speedtest.net/ookla-5g-map`, January 2021.

[193] T. W. d. P. Paiva, S. Ferlin, A. Brunstrom, O. Alay, and B. Y. L. Kimura. A first look at adaptive video streaming over multipath quic with shared bottleneck detection. In

Proceedings of the 14th Conference on ACM Multimedia Systems, pages 161–172, 2023.

[194] M. Palmer, T. Krüger, B. Chandrasekaran, and A. Feldmann. The quic fix for optimal video streaming. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, pages 43–49, 2018.

[195] A. Papathanassiou and A. Khoryaev. Cellular v2x as the essential enabler of superior global connected transportation services. *IEEE 5G Tech Focus*, 1(2):1–2, 2017.

[196] Pingdom. Does Page Load Time Really Affect Bounce Rate? `https://www.pingdom.com/blog/page-load-time-really-affect-bounce-rate/`, 2018.

[197] M. Piraux, Q. De Coninck, and O. Bonaventure. Observing the evolution of quic implementations. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, pages 8–14, 2018.

[198] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan. Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 99–114, 2018.

[199] F. Qian, S. Sen, and O. Spatscheck. Characterizing resource usage for mobile web browsing. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 218–231, 2014.

[200] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. Profiling resource usage for mobile applications: a cross-layer approach. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 321–334, 2011.

[201] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Characterizing radio resource allocation for 3g networks. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 137–150, 2010.

[202] H. Qiu, F. Ahmad, F. Bai, M. Gruteser, and R. Govindan. Avr: Augmented vehicular reality. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 81–95, 2018.

[203] H. Qiu, P. Huang, N. Asavisanu, X. Liu, K. Psounis, and R. Govindan. Autocast: Scalable infrastructure-less cooperative perception for distributed collaborative driving. *arXiv preprint arXiv:2112.14947*, 2021.

[204] C. Raiciu, M. Handley, and D. Wischik. Coupled congestion control for multipath transport protocols. *RFC 6356, IETF*, 2011.

[205] E. Ramadan, A. Narayanan, U. K. Dayalan, R. A. K. Fezeu, F. Qian, and Z. Zhang. Case for 5g-aware video streaming applications. *ACM SIGCOMM'21 5G-MeMU Workshop*, 2021.

[206] T. S. Rappaport, G. R. MacCartney, M. K. Samimi, and S. Sun. Wideband millimeter-wave propagation measurements and channel models for future wireless communication system design. *IEEE transactions on Communications*, 63(9):3029–3056, 2015.

[207] A. Rauch, F. Klanner, R. Rasshofer, and K. Dietmayer. Car2x-based perception in a high-level fusion architecture for cooperative perception systems. In *2012 IEEE Intelligent Vehicles Symposium*, pages 270–275. IEEE, 2012.

[208] W. Roh, J.-Y. Seol, J. Park, B. Lee, J. Lee, Y. Kim, J. Cho, K. Cheun, and F. Aryanfar. Millimeter-wave beamforming as an enabling technology for 5g cellular communications: Theoretical feasibility and prototype results. *IEEE communications magazine*, 52(2):106–113, 2014.

[209] S. Rosen, H. Luo, Q. A. Chen, Z. M. Mao, J. Hui, A. Drake, and K. Lau. Discovering fine-grained rrc state dynamics and performance impacts in cellular networks. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 177–188, 2014.

[210] S. Rosen, A. Nikravesh, Y. Guo, Z. M. Mao, F. Qian, and S. Sen. Revisiting network energy efficiency of mobile apps: Performance in the wild. In *Proceedings of the 2015 Internet Measurement Conference*, pages 339–345, 2015.

[211] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009.

[212] J. Rüth, I. Poese, C. Dietzel, and O. Hohlfeld. A first look at quic in the wild. In *International Conference on Passive and Active Network Measurement*, pages 255–268. Springer, 2018.

[213] J. Rüth, K. Wolsing, K. Wehrle, and O. Hohlfeld. Perceiving quic: Do users notice or even care? In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, pages 144–150, 2019.

[214] J. Sachs, G. Wikstrom, T. Dudda, R. Baldemair, and K. Kittichokechai. 5g radio network design for ultra-reliable low-latency communication. *IEEE Network*, 32(2):24–31, 2018.

[215] S. K. Saha, S. Aggarwal, R. Pathak, D. Koutsonikolas, and J. Widmer. Musher: An agile multipath-tcp scheduler for dual-band 802.11 ad/ac wireless lans. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.

[216] Samsung. 4G-5G Interworking. `https://images.samsung.com/is/content/samsung/p5/global/business/networks/insights/white-paper/4g-5g-interworking/global-networks-insight-4g-5g-interworking-0.pdf`, 2017.

[217] V. Sarode, X. Li, H. Goforth, Y. Aoki, R. A. Srivatsan, S. Lucey, and H. Choset. Pcrnet: Point cloud registration network using pointnet encoding. *arXiv preprint arXiv:1908.07906*, 2019.

[218] M. Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[219] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. N. Padmanabhan. Bartendr: a practical approach to energy-aware cellular data scheduling. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 85–96, 2010.

[220] M. Seemann and J. Iyengar. Automating quic interoperability testing. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, pages 8–13, 2020.

[221] S. Shi, J. Cui, Z. Jiang, Z. Yan, G. Xing, J. Niu, and Z. Ouyang. Vips: Real-time perception fusion for infrastructure-assisted autonomous driving. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 133–146, 2022.

[222] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[223] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[224] T. Shreedhar, R. Panda, S. Podanev, and V. Bajpai. Evaluating quic performance over web, cloud storage, and video workloads. *IEEE Transactions on Network and Service Management*, 19(2):1366–1381, 2021.

[225] V. Singh, A. Prabhakara, D. Zhang, O. Yağan, and S. Kumar. A community-driven approach to democratize access to satellite ground stations. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2021.

[226] I. Sodagar. The mpeg-dash standard for multimedia streaming over the internet. *IEEE multimedia*, 18(4):62–67, 2011.

[227] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[228] Starlink Hardware. How Much Power Does Starlink Use? `https://www.starlinkhardware.com/how-much-power-does-starlink-use/`, 2023.

[229] X. Su, A. Jiang, J. Cao, W. Zhang, P. Hui, and J. Ye. Enabling continuous object recognition in mobile augmented reality. In *27th International Conference on Intelligent User Interfaces*, pages 42–45, 2022.

[230] S. Sukhmani, M. Sadeghi, M. Erol-Kantarci, and A. El Saddik. Edge caching and computing in 5g for mobile ar/vr and tactile internet. *IEEE MultiMedia*, 26(1):21–30, 2018.

[231] Z. Tan, Y. Li, Q. Li, Z. Zhang, Z. Li, and S. Lu. Supporting mobile vr in lte networks: How close are we? *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(1):1–31, 2018.

[232] D. Vasisht, J. Shenoy, and R. Chandra. L2d2: Low latency distributed downlink for leo satellites. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 151–164, 2021.

[233] R. Wang, M. A. Kishk, and M.-S. Alouini. Reliability analysis of multi-hop routing in multi-tier leo satellite networks. *arXiv preprint arXiv:2303.02286*, 2023.

[234] T.-H. Wang, S. Manivasagam, M. Liang, B. Yang, W. Zeng, and R. Urtasun. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 605–621. Springer, 2020.

[235] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen. In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *Ieee Network*, 33(5):156–165, 2019.

[236] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall. Demystifying page load performance with {WProf}. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 473–485, 2013.

[237] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall. How speedy is {SPDY}? In *11th usenix symposium on networked systems design and implementation (nsdi 14)*, pages 387–399, 2014.

[238] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath {TCP}. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, 2011.

[239] K. Wolsing, J. Rüth, K. Wehrle, and O. Hohlfeld. A performance perspective on web optimized protocol stacks: Tcp+ tls+ http/2 vs. quic. In *Proceedings of the Applied Networking Research Workshop*, pages 1–7, 2019.

[240] P. Wu, S. Chen, and D. N. Metaxas. Motionnet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11385–11395, 2020.

[241] X. Xie, X. Zhang, and S. Zhu. Accelerating mobile web loading using cellular link information. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 427–439, 2017.

[242] D. Xu, A. Zhou, X. Zhang, G. Wang, X. Liu, C. An, Y. Shi, L. Liu, and H. Ma. Understanding operational 5g: A first measurement study on its coverage, performance and energy consumption. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 479–494, 2020.

[243] S. Xu, S. Sen, and Z. M. Mao. Csi: Inferring mobile abr video adaptation behavior under https and quic. In *Proceedings of the Fifteenth European Conference on Computer Systems*, pages 1–16, 2020.

[244] S. Xu, S. Sen, Z. M. Mao, and Y. Jia. Dissecting vod services for cellular: performance, root causes and best practices. In *Proceedings of the 2017 Internet Measurement Conference*, pages 220–234, 2017.

[245] Z. Xu, X. Li, X. Zhao, M. H. Zhang, and Z. Wang. Dsrc versus 4g-lte for connected vehicle applications: A study on field experiments of vehicular communication performance. *Journal of Advanced Transportation*, 2017, 2017.

[246] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 495–511, 2020.

[247] Y. Yan and Z. Yang. When quic's connection migration meets middleboxes a case study on mobile wi-fi hotspot. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2021.

[248] S.-C. Yen, C.-L. Fan, and C.-H. Hsu. Streaming 360° videos to head-mounted virtual reality using dash over quic transport protocol. In *Proceedings of the 24th ACM Workshop on Packet Video*, pages 7–12, 2019.

[249] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 325–338, 2015.

[250] A. Yu and T. A. Benson. Dissecting performance of production quic. In *Proceedings of the Web Conference 2021*, pages 1157–1168, 2021.

[251] H. Zeng, H. Pirayesh, P. K. Sangdeh, and A. Quadri. Vehcom: Delay-guaranteed message broadcast for large-scale vehicular networks. *IEEE Transactions on Wireless Communications*, 20(6):3883–3896, 2021.

[252] Y. Zeng, Y. Hu, S. Liu, J. Ye, Y. Han, X. Li, and N. Sun. Rt3d: Real-time 3-d vehicle detection in lidar point cloud for autonomous driving. *IEEE Robotics and Automation Letters*, 3(4):3434–3440, 2018.

[253] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 105–114, 2010.

[254] Q. Zhang, S. Jin, J. Sun, X. Zhang, R. Zhu, Q. A. Chen, and Z. M. Mao. On data fabrication in collaborative vehicular perception: Attacks and countermeasures. *arXiv preprint arXiv:2309.12955*, 2023.

[255] Q. Zhang, X. Zhang, R. Zhu, F. Bai, M. Naserian, and Z. M. Mao. Robust real-time multi-vehicle collaboration on asynchronous sensors. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–15, 2023.

[256] W. Zhang, F. Qian, B. Han, and P. Hui. Deepvista: 16k panoramic cinema on your mobile device. In *Proceedings of the Web Conference 2021*, WWW '21, page 2232–2244, New York, NY, USA, 2021. Association for Computing Machinery.

[257] W. Zhang, H. Zhou, S. Sun, Z. Wang, J. Shi, and C. C. Loy. Robust multi-modality multi-object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2365–2374, 2019.

[258] X. Zhang, S. Jin, Y. He, A. Hassan, Z. M. Mao, F. Qian, and Z.-L. Zhang. Quic is not quick enough over fast internet. In *Proceedings of the ACM on Web Conference 2024*, pages 2713–2722, 2024.

[259] X. Zhang, A. Zhang, J. Sun, X. Zhu, Y. E. Guo, F. Qian, and Z. M. Mao. Emp: Edge-assisted multi-vehicle perception. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 545–558, 2021.

[260] X. Zhang, X. Zhu, Y. E. Guo, F. Qian, and Z. M. Mao. Poster: characterizing performance and power for mmwave 5g on commodity smartphones. In *Proceedings of the 2019 on Wireless of the Students, by the Students, and for the Students Workshop*, pages 14–14, 2019.

[261] Y. Zhang, Q. Wu, Z. Lai, and H. Li. Enabling low-latency-capable satellite-ground topology for emerging leo satellite networks. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 1329–1338. IEEE, 2022.

[262] H. Zhao, R. Mayzus, S. Sun, M. Samimi, J. K. Schulz, Y. Azar, K. Wang, G. N. Wong, F. Gutierrez Jr, and T. S. Rappaport. 28 ghz millimeter wave cellular communication measurements for reflection and penetration loss in and around buildings in new york city. In *ICC*, pages 5163–5167, 2013.

[263] K. Zhao, J. Helander, D. Sjöberg, S. He, T. Bolin, and Z. Ying. User body effect on phased array in user equipment for the 5g mmwave communication system. *IEEE Antennas and Wireless Propagation Letters*, 16:864–867, 2017.

[264] X. Zhu, S. Sen, and Z. M. Mao. Livelyzer: analyzing the first-mile ingest performance of live video streaming. In *Proceedings of the 12th ACM Multimedia Systems Conference*, pages 36–50, 2021.

[265] X. Zhu, J. Sun, X. Zhang, Y. E. Guo, F. Qian, and Z. M. Mao. Mpbond: efficient network-level collaboration among personal mobile devices. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pages 364–376, 2020.

[266] Y. Zhu, Z. Zhang, Z. Marzi, C. Nelson, U. Madhow, B. Y. Zhao, and H. Zheng. Demystifying 60ghz outdoor picocells. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 5–16, 2014.

[267] J. Zirngibl, P. Buschmann, P. Sattler, B. Jaeger, J. Aulbach, and G. Carle. It's over 9000: analyzing early quic deployments with the standardization on the horizon. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 261–275, 2021.