UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Engineering Science and Technology
Department of Electrical Engineering

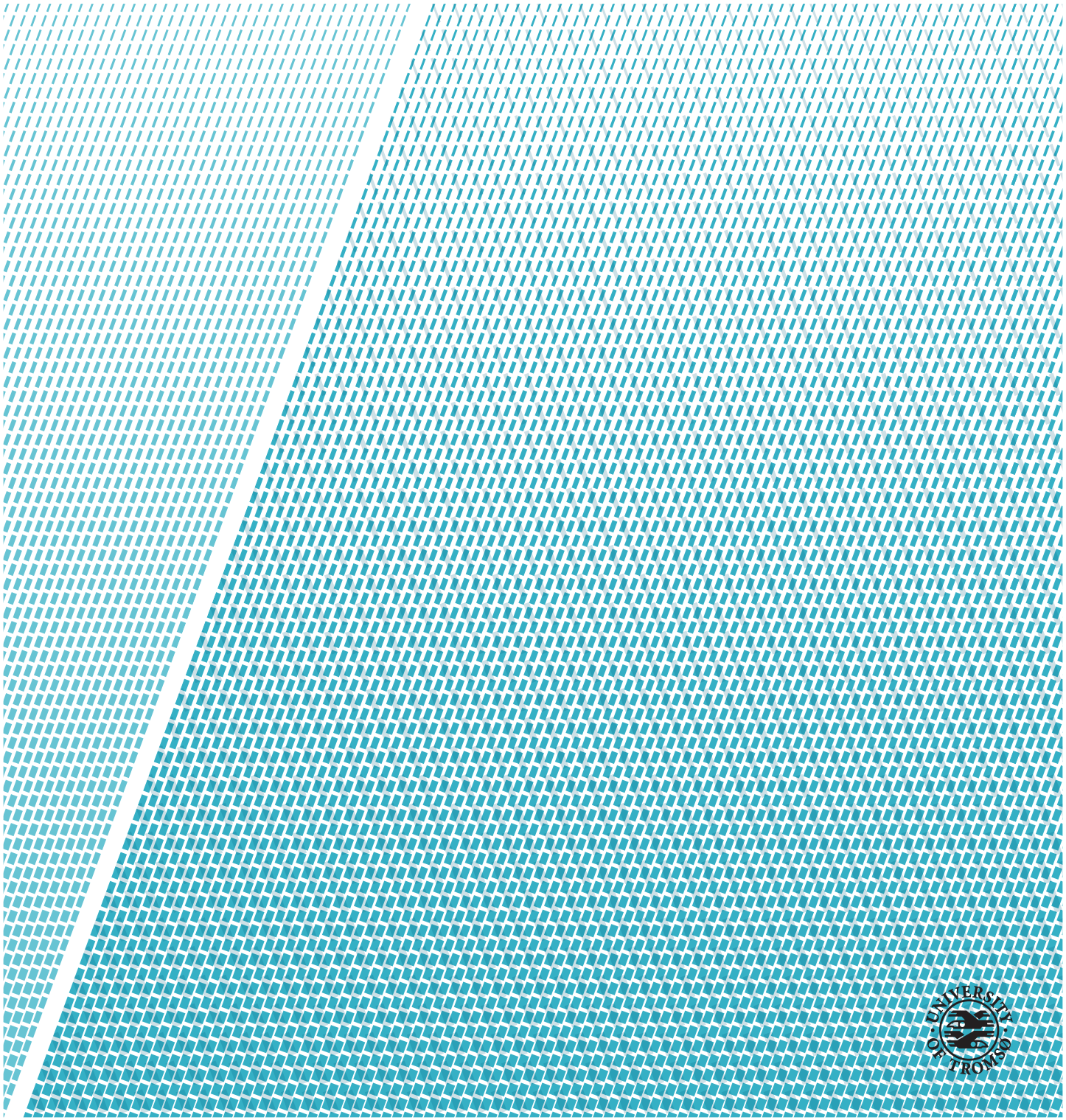
# A proxy for reliable 5G (and beyond) mmWave communications.

## Contributions to multi-path scheduling for a reliability focused mmWave proxy.

—

Dzifa Korbla Wisdom Tofah

Master thesis in Electrical Engineering ELE-3900... May 16, 2022



# Abstract

Reliable, consistent and very high data rate mobile communication will become especially important for future services such as, among other things, future emergency communication needs. MmWave technology provides the needed capacity, however lacks the reliability due to the abrupt capacity changes any one path experiences. Intelligently making use of varying numbers of available mmWave paths, efficiently scheduling data across the paths, perhaps even through multi-operator agreements; and balancing mobile power consumption with path costs and the need for reliable consistent quality will be critical to attaining this aim.

In this thesis, the multipath scheduling problem in a mmWave proxy when the paths have dynamically changing path characteristics is considered. To address this problem, a hybrid scheduler is proposed, the performance of which is compared with the Round Robin scheduler, Random scheduler and the Highest Capacity First scheduler. Forward error correction is explored as a means of enhancing the scheduling.

**Keywords:Multipath Scheduling, mmWave Proxy, Forward Error Correction, beyond 5G.**

# Acknowledgements

I would first like to thank God Almighty for giving me the ability to carry out this project. Special thanks to my supervisors Dr. Tu Dac Ho and Dr. David Hayes, whose doors were always open whenever I needed guidance or had a question about my research or writing. They consistently allowed this thesis to be my own work, but steered me in the right direction whenever necessary.

I would also like to thank the staff of the department of Electrical Engineering and SimulaMet for their immense contribution to the successful completion of this project. I thank my family and friends for accepting nothing less than excellence from me.

Finally, I would like to thank my ever supportive wife Esther Ama Adiku for her inspiration and motivation throughout writing this thesis.

# Contents

# List of Figures

# 1 Introduction

The first phase of the Fifth Generation of Mobile Communications(5G) which is defined in the 3GPP(3rd Generation Partnership Project) Release 15 is designed to support diverse services with different data traffic profiles (e.g., high throughput, low latency and massive connections) and models (e.g., IP data traffic, non-IP data traffic, short data bursts and high throughput data transmissions). 5G characteristics offer the flexibility needed to support services in sectors such as automotive and other transport (trains, maritime communications), logistics, IoT, discrete automation, electricity distribution, public safety, health and wellness, smart cities, media, and entertainment [1].The second phase of 5G is defined in the 3GPP Release 16 and among other things provides new features for URLLC (Ultra-Reliable Low Latency Communication) and Industrial IoT, including Time Sensitive Communication (TSC), enhanced Location Services, and support for Non-Public Networks (NPNs)[2]. More 5G system enhancements will follow in Release 17.

Two deployment options are available for 5G architecture: the Non-Stand Alone(NSA) and the Stand Alone(SA). The NSA architecture features the deployment of the 5G Radio Access Network(AN) and the New Radio(NR) interface in conjunction with the existing LTE and EPC infrastructure Core Network (respectively 4G Radio and 4G Core) as shown in Fig.1.1. This configuration, only supports 4G services, but enjoying the capacities offered by the 5G New Radio (lower latency, etc.). The NSA architecture can be seen as a temporary step towards “full 5G” deployment, where the 5G Access Network connects to the 4G Core Network. However, the SA configuration shown in Fig.1.2, sup-

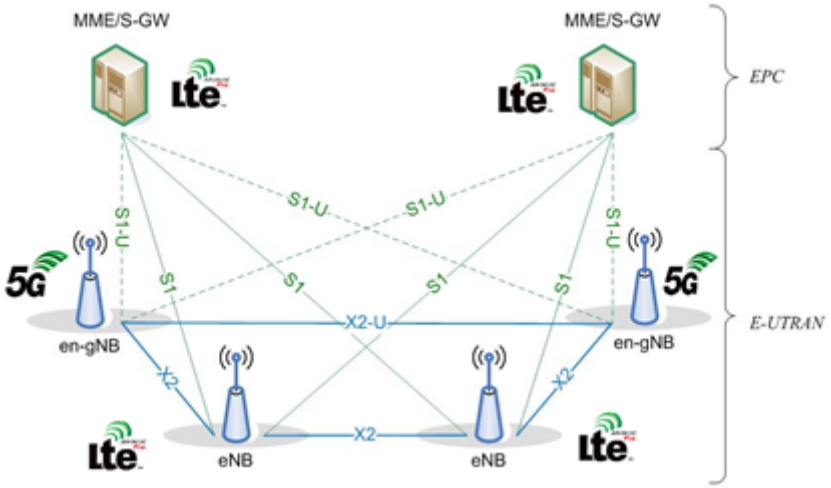



**Figure 1.1:** The Non-Standalone Architecture of 5G Phase 1 [1]

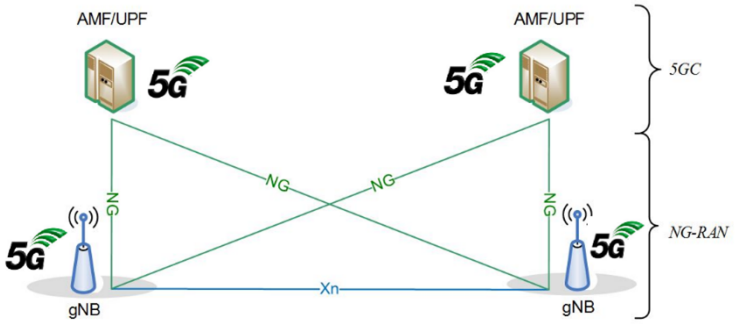



**Figure 1.2:** The Standalone Architecture of 5G Phase 1 [1]

ports the full set of 5G Phase 1 services [1].5G has is deployed commercially throughout the world both at sub-6 GHz and at mmWave frequencies[2] by various operators and choosing the right 5G architecture involves many deciding factors. Among these are desired service offerings, level of investment, and spectrum availability [3].

Previous generations of wireless networks have had various challenges in terms of data throughput, connectivity, and latency. While the 4G LTE-A system ensures the downlink(DL) and uplink(UL) data rate of up to 3Gb/s and 1.5Gb/s respectively, with the connectivity of 600 users per cell approximately and latency of around 30-50 milliseconds [4], 5G is expected to provide significant progress in all aspects of performance, including a 1000-fold growth in system capacity, an enhanced connectivity to at least 100 billion devices, 10 Gbps maximum and 100 Mbps average individual user experience, prolonged battery

life with 1000-fold lower energy consumption per bit, a 90% reduction in network energy usage, support to 500 km/h mobility for high speed users (e.g. high speed trains), a 3-fold increase in spectrum efficiency, perception of 99.99% availability, 100% coverage, and latency from 1 to 10 milliseconds [5].

The evolution beyond 5G is expected to introduce a range of new services dependent on higher capacity, peak throughput of about a terabit per second (Tbps) and low latency, while leveraging the benefits of IoT and big data.

New lightweight devices or wearables will emerge relying on distributed computing, intelligent computing surfaces and storage enabled via edge cloud. Some of these emerging services include:Holographic Teleportation, Extended Reality, Biosensors, Tactile Services such as remote surgery, Internet of Everything, Internet of Skills, Autonomous Services, Enhanced Vehicular Communications, Unmanned Aerial Vehicle Services. B5G is expected to enhance NR and millimeter Wave (mmWave) exploiting a wider range of frequencies from sub-6 GHz to 300 GHz [6].

Due to the ever-increasing demand for higher data rate, wireless communications operating in a mmWave frequency band is one of the promising solutions to alleviate the current resource bottleneck for future communication systems.

With the deployment of mmWave communication networks, the current spectrum bottleneck in conventional microwave LTE systems could be solved with a higher provided bandwidth [5]. mmWave radio will also play an increasing part in future 6G networks and beyond. On the electromagnetic spectrum, mmWaves are located between 30–300 GHz. The wavelength of mmWaves based on the relation $\lambda = c/f$ is $10mm$ and $1mm$ for $30Ghz$ and $300GHz$ respectively. Where f is the frequency in Hz and c is the speed of light($3x10^8$ $ms^{-1}$) [7]. This frequency range falls between the Microwave region and Terahertz region, as depicted in Figure 1.3 The shorter wavelength of the mmWave makes it less penetrating with smaller coverage as compared to microwave. This causes fluctuations in transmission capacity. The fluctuations are more severe when the frequency of switching between LOS and NLOS is high [8].

The application of mmWave communications, however, is accompanied by several technical challenges in the 5G cellular networks. The main issues introduced by communications at such high frequencies are the sensitivity to blockage and the high bandwidth fluctuations due to Line of Sight (LOS) to Non Line of Sight (NLOS) transitions and vice versa [10]. More specifically, mmWave signals are susceptible to blockages such as buildings and human

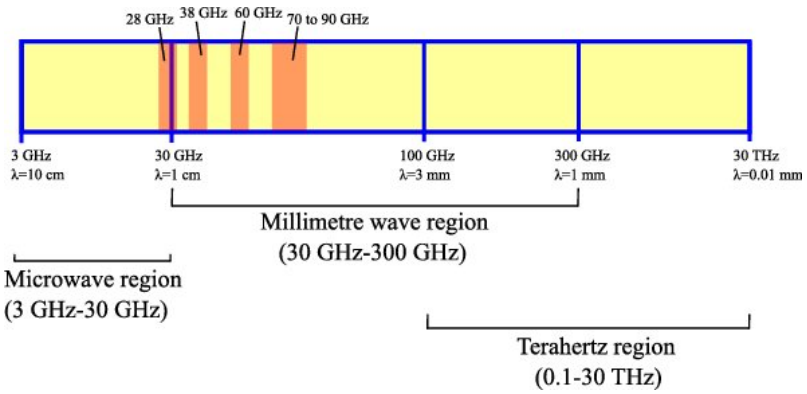



**Figure 1.3:** Section of mmWave on Electromagnetic Spectrum [9]

bodies. Furthermore, at mmWave bands, The attenuation caused by precipitation can not be neglected as rain droplets can absorb mmWave signals whose wavelengths (1 mm to 10 mm) is comparable with the size of a raindrop(a few millimeters)[11]. Figure 1.4 shows the attenuation due to rain at various frequencies.

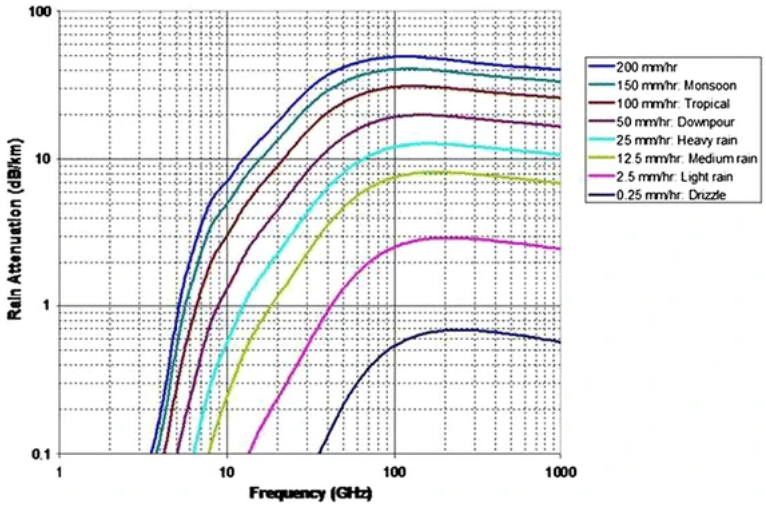



**Figure 1.4:** Rain attenuation at mmWave frequencies [12]

The identified use cases for 5G and beyond networks demand stable, consistent bit rates and minimal delays. Thus, ensuring stable, reliable communication for these very demanding applications over mmWave links is a challenging problem; one that will escalate as coming mobile systems (6G/7G) rely more on mmWave links at even higher frequencies than today [13].

A proxy is a middlebox which plays an intermediary role between a client and a server. One of the reasons for deploying a proxy on the link between

a client and a server is to address performance issues in scenarios where the characteristics of the link has adverse effects [14]. This type of proxy is a Performance Enhancing Proxy(PEP). PEPs are used to address the issue of wireless link characteristics impacting transport protocols. Existing proxies of this nature function on a single radio channel on 4G networks [13]. Aside PEPs, multipath transport protocols are also in existence to mitigate the drawbacks posed by wireless link characteristics. Access Traffic Steering, Switching, and Splitting (ATSSS) in the 3GPP System Architecture for the 5G system provides support in the 5G core for transport layer multi-connectivity between 3GPP and non-3GPP networks. In this regard, the authors in [15] explored multi-connectivity between LTE/5G and WLAN. Plans are in place to expand ATSSS to utilize multiple 3GPP networks for multi-connectivity.

The scenario envisaged for this proxy is similar to what is depicted in Figure 1.5 where a mobile device is equipped with a radio with multiple mmWave connectivity support, so the mobile device can simultaneously connect to multiple base stations (BSs). Considering the LoS path to a BS may be temporarily blocked due to movement of the mobile device or objects around it, the goal is to dynamically select the minimum number of mmWave links necessary to provide the required QoS, for a given application and UE. A proposal is made by [13] for the use of a splitting multipath proxy that separates the multipath mmWave domain from the Internet, with two key roles:

1. for a given application and UE, select the minimum number of mmWave links necessary to provide the required QoS.

2. schedule data packets for transmission across the different selected links.

The work presented by Hayes et al. so far focuses on the multipath management problem, with multipath scheduling slated for future work. This thesis focuses on the scheduling problem.

A mathematical model that tries to capture the LoS/NLoS dynamics of a set of links and the resulting, aggregate bit rates that allow to drain an application flow's buffer in the proxy. The goal is for the proxy to be able to predict, over short time horizons (say, a couple of seconds), the state of the buffer with reasonable accuracy. The efficiency of these models for real-time control in terms of accuracy (i.e. how accurately the models predict) and performance are evaluated. Through event-based simulations, the effectiveness of the proposed mechanism are shown.

The objectives of this thesis are as follows:

- Investigate multipath scheduling mechanisms which can be adopted for

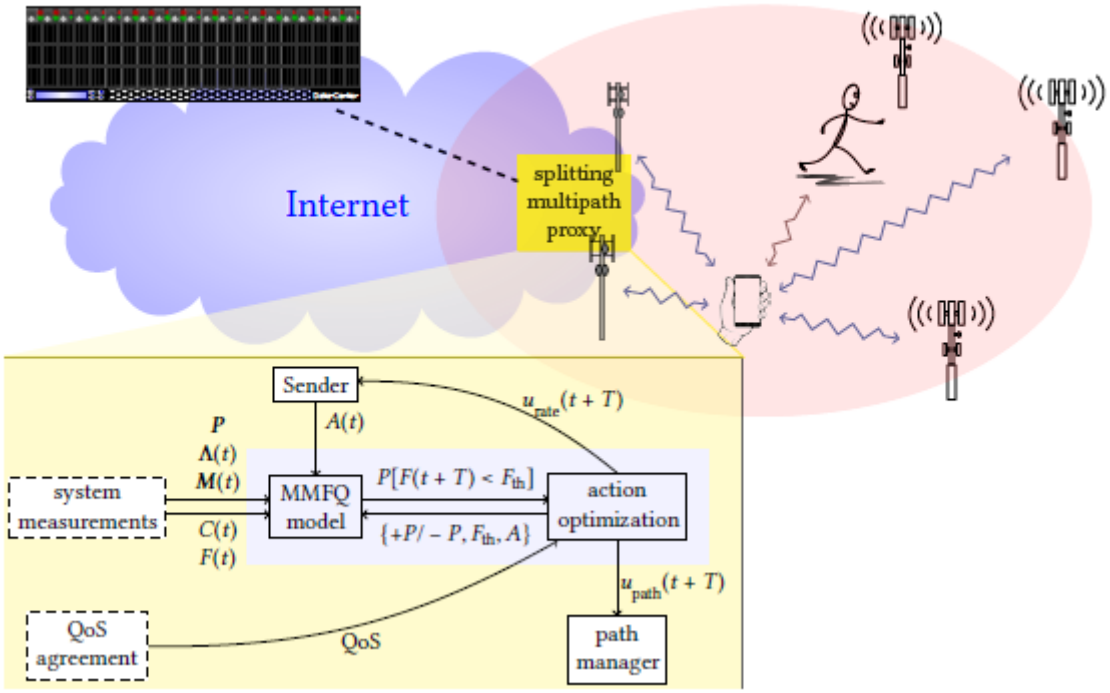



**Figure 1.5:** System architecture overview of mmWave scenario with five base stations connected to the multipath proxy. [13]

5G to enhance mmWave communication.

- Investigate the challenges associated with path scheduling in a multipath proxy.
- Propose dynamic multi-path scheduling algorithms to optimally schedule data packets for transmission across different selected links.
- Evaluate the proposed algorithms by means of simulations to test their efficacy in scheduling packets across a set of predicted paths as compared to other methods.

Chapter two will explore existing works and literature on scheduling. Chapter three will deal with the evaluation of proposed algorithms which are suitable for the purpose of the proxy. Results and analysis with be presented in Chapter four. Final conclusions will be presented in Chapter 5.

# 2

# Literature Review

Proxies for transmission over multiple paths in wireless networks is a widely researched area. Multiple path transmission is a means of mitigating the adverse effects of channel characteristics variation on the network performance and also for capacity expansion. The authors in [8] show that the Transmission Control Protocol (TCP) performance over mmWave is seriously affected by sudden transitions between LOS and NLOS states. A number of proxies, such as the one proposed in [10] seek to alleviate the shortfalls of mmWave communication.

However, these proxies focus on TCP and mostly seek to utilize the full network capacity [16, 17, 18, 19, 20]. This project is not TCP-centric and seeks instead to maintain reliable, consistent rates. Consequently, no existing literature to the best of our knowledge covers the scope of this project. This work aims at making it possible for users of 5G and beyond networks to be able to reliably and consistently communicate over mmWave networks; with devices that are capable of connecting to multiple mmWave networks simultaneously. We seek to efficiently dispatch packets of data across multiple mmWave channels based on factors such as channel capacity and availability.

To begin the subject of scheduling packets across mmWave channels, which is the focus of this project, the topic of queuing will be first addressed.

## 2.1 Queuing Theory

Queues enable systems to provide service efficiently. Queuing theory embodies the full gamut of models covering all perceivable systems which incorporate characteristics of a queue [21].

Modeling and analyzing queues, especially in communication systems, enable us to derive the optimum potential of system resources, which are most often than not limited. The fact that mmWaves are susceptible to fluctuations in capacity depending on whether they are in LoS or not makes it inherently important to be able to model and analyze the flow of data. Knowledge of this helps in deciding the appropriate scheduling scheme to adopt.

### 2.1.1 Challenges in a Queuing System

The goal of analyzing queuing systems is to get a fair idea of how the underlying processes interact and to devise appropriate management decisions. The problems associated with queuing systems can be classified into three categories, namely:

1. Behavioral Problems
2. Statistical Problems
3. Decision Problems

#### Behavioral Problems

The study of behavioral problems of queuing systems is intended to understand how they behave under various conditions. Research on behavioral problems forms the principal basis for the results in queuing theory. The analysis uses mathematical models for the probability relationships among various elements of the underlying process.

#### Statistical Problems

Under statistical problems, we include the analysis of empirical data in order to identify the correct mathematical model, and validation methods to determine whether the proposed model is appropriate. In the course of modeling, we make several assumptions regarding the basic elements of the model. Naturally, there should be a mechanism by which these assumptions could be verified. Starting

with testing the goodness of fit for the arrival and service distributions, one would need to estimate the parameters of the model and/or test hypotheses concerning the parameters or behavior of the system. Other important questions where statistical procedures play a part are in the determination of the inherent dependencies among elements, and dependence of the system on time.

### Decision Problems

Decisions problems include all problems that are inherent in the operation of queuing systems. Some such problems are statistical in nature. Others are related to the design, control, and the measurement of effectiveness of the systems. Figure 2.1 depicts a typical single-server queue system. Packets arrive, wait for transmission, are transmitted, and then leave the system. In modeling

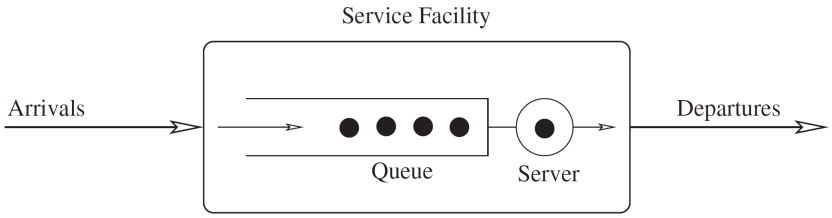



**Figure 2.1:** Single-server queue [22]

and analyzing the queue system, some assumptions below can be made:

- A server is free if it is not serving any customer
- If a server is free, an arriving customer can be scheduled on it.
- If a server is busy, an arriving customer would have to wait to be scheduled on a free path.
- When a server becomes available, a customer in the queue departs based on the scheduling policy in use.
- Packets remain in the queue until they are transmitted.
- A dropped customer is one which arrives when the queue is full.

For a more detailed analysis, events such as how a packet arrives and what server transmits a packet need to be discussed. For a quantitative evaluation of a queuing system, the key characteristics of interest are [23]:

- Arrival pattern
- Service pattern of servers
- Number of servers
- System capacity
- Queue discipline
- Number of service stages

### 2.1.2 Arrival Pattern

In usual queuing situations, the process of arrivals is stochastic, and it is thus necessary to know the probability distribution describing the times between successive customer arrivals. A common arrival process is the Poisson process. Another factor is the manner in which the pattern changes with time. An arrival pattern that does not change with time is a stationary arrival pattern. If it is not time dependent, we refer to it as a non-stationary. The number of arrivals per unit time (the arrival rate) and the time between successive arrivals (the inter-arrival time) characterizes the arrival process [22, 23, 21].

### 2.1.3 Service Patterns

The description of the sequence of customer service times employs a probability distribution, since service times are usually stochastic. Service may also be single or batch. A state-dependent service refers to the situation in which service depends on the number waiting. Service, like arrivals, can be stationary or non-stationary with respect to time. A queuing system can be both non-stationary and state-dependent [22, 23, 21].

### 2.1.4 Number of Servers

The number of servers is an important characteristic of a queuing system and represents a fundamental tradeoff – adding servers incurs extra cost to the system, but can substantially reduce delays. Thus, the choice of the number of servers is often a critical decision [22, 23, 21].

### 2.1.5 System Capacity

In some systems, there is a physical limitation to the amount of space for customers to wait, so that when the line reaches a certain length, nothing enters until space becomes available. These are referred to as finite queuing situations; that is, there is a finite limit to the maximum system size [22, 23, 21].

### 2.1.6 Stages of Service

A queuing system could have only a single stage of service, or it could have several stages.In some multistage queuing processes, recycling or feedback may occur. For example, a telecommunications network may process messages through a randomly selected sequence of nodes, with the possibility that some messages will require rerouting through the same stage [22, 23, 21].

### 2.1.7 Queue Discipline

Queue discipline refers to the manner in which customers in the queue receive service when a queue has formed. This is also sometimes the scheduling policy, scheduling algorithm, or scheduling discipline. Usually, we assume the time it takes to select a customer and move the customer into service is zero. The departure of a customer and the start of service of a waiting customer is instantaneous. A common discipline is first come, first served (FCFS). However, there are many other disciplines [22, 23, 21]. As a way of generalizing the description of queuing systems, Kendall in 1953 introduced a notation to aid in this endeavor. It is popularly known as the Kendall's Notation

### 2.1.8 Kendall's Notation

A series of symbols and slashes A/B/X/Y/Z represent the description of a queuing process. $A$ indicates the inter-arrival time distribution $B$ indicates the service time distribution $X$ indicates the number of servers $Y$ indicates the system capacity $Z$ indicates the queue scheduling discipline.

Regarding A and B in the notation, some possible distributions are:

- $M$ for Markovian
- $E$ for Erlang
- $C$ for Coxian of order k

- $G$ for General
- $D$ for Deterministic or Constant

Queuing System Notation $A/B/X/Y/Z$

| Characteristic | Symbol | Explanation |
|---|---|---|
| | $M$ | Exponential |
| Interarrival-time | $D$ | Deterministic |
| distribution $(A)$ | $E_k$ | Erlang type $k(k = 1, 2, \ldots)$ |
| Service-time | $H_k$ | Mixture of $k$ exponentials |
| distribution $(B)$ | $PH$ | Phase type |
| | $G$ | General |
| Parallel servers $(X)$ | $1, 2, \ldots, \infty$ | |
| System capacity $(Y)$ | $1, 2, \ldots, \infty$ | |
| Queue discipline $(Z)$ | FCFS | First come, first served |
| | LCFS | Last come, first served |
| | RSS | Random selection for service |
| | PR | Priority |
| | GD | General discipline |

For a general arrival process, the notation GI is sometimes used to replace G to indicate that, although the inter-arrival time distribution may be completely general, successive arrivals are independent of each other. The number of servers (C) is often taken to be 1. These first three parameters are always provided. Thus, for example, the M/M/1 queue means that the arrival process and service process are both Markovian (although we usually say that the arrival process is Poisson and the service time distribution is exponential) and there is a single server. The letters that specify the system capacity, customer population and scheduling discipline may be omitted with the understanding that the default values for capacity and population size is infinite and the default for the scheduling discipline is FCFS. Thus the M/M/1 queue has an unlimited amount of space in which to hold waiting customers, an infinite population from which customers are drawn and applies a FCFS scheduling policy.

In analyzing a queuing system, the aim is usually to obtain certain system properties such as:

- the waiting time for a customer
- the number of customers in the system
- the length of a busy or idle period

These values of interest are the measures of effectiveness.

### 2.1.9 Little's Law

Little's law provides a relationship between three fundamental quantities: The average rate $\lambda$ that customers arrive to a system, the average time $W$ that a customer spends in the system, and the average number $L$ of customers in the system in Figure 2.2 is:

$$L = \lambda W.$$

Given two of the three quantities, one can infer the third. Little's law is inde-

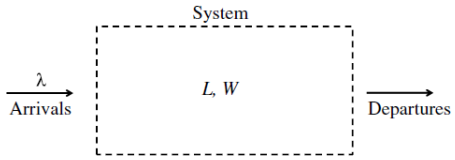



**Figure 2.2:** Little's Law [23]

pendent of :

- specific assumptions regarding the arrival distribution A(t)
- specific assumptions regarding the service time distribution B(t)
- the number of servers
- the particular queuing discipline.

The choice of a scheduling algorithm for the purpose of the proxy is vital for the delivery of the desired quality of service parameters. The design of scheduling algorithms for mmWave networks is challenging due to a high degree of variation in the link characteristics.

## 2.2 Scheduling Algorithms

Scheduling algorithms fit into two broad categories: work-conserving and non-work-conserving [24]. The difference between them is that while a work-conserving algorithm does not idle when a packet is waiting to be transmitted, a

non-work-conserving algorithm is occasionally idle. A non-conserving scheduling algorithm is sometimes idle though there is a packet backlog because the arrival of a higher-priority packet might be in anticipation. Scheduling algorithms such as Processor Sharing (GPS), Weighted Fair Queuing (WFQ) [25], Virtual Clock (VC), Weighted Round-Robin (WRR), Self-Clocked Fair Queuing (SCFQ), and Deficit Round-Robin (DRR) are classified under work-conserving scheduling algorithms. Some examples of non-work-conserving algorithms include Hierarchical Round-Robin (HRR) [26], Stop-and-Go Queuing (SGQ), and Jitter-Earliest-Due-Date (Jitter- EDD).

Although non-work-conserving schedulers usually have higher average packet delays relative to work-conserving schedulers, in situations where time jitter is more important than delay, non-work-conserving schedulers are useful [24]. Schedulers can also be classified based on timestamps. Such schedulers consider the timestamp of arriving packets before they are processed. Once stamping of arriving packets is finishes, they are then split into the various queues. Sorting of head-of-line packets are is based on the packet with the lowest timestamp. Unlike time stamped schedulers, round-robin schedulers do not use timestamps, making them more implementation friendly. The tradeoff however with round-robin relative to timestamped schedulers is poor QoS provision. Another class of schedulers is sorted-priority based scheduler. The transmission priority of a packet is dependent on the priority of the session it belongs to. WFQ is a typical example of a scheduler that belongs to this class.

It is desirable for a scheduling algorithm to possess the following features [24]:

1. Efficient link utilization: there is an expectation for the algorithm to efficiently utilize the channel. Thus, the scheduler must not assign a packet to a link which currently in a bad state (NLoS) because the transmission might go to waste.

2. Throughput: the algorithm must provide guaranteed short-term throughput for error-free sessions and guaranteed long-term throughput for all sessions.

3. Delay bound: the algorithm must be able to provide delay bound guarantees for individual sessions in order to support delay-sensitive applications.

4. Implementation complexity: A low-complexity algorithm is a necessity in high-speed networks in which scheduling decisions have to be made very rapidly.

5. Energy consumption: consideration must be given to the MS battery life in the process of selecting an algorithm.

## 2.3 The State-of-the-Art Multipath Schedulers

The most common and basic multipath schedulers are elemental schedulers [27]. These schedulers are popularly used for comparisons and analysis in research articles. The scheduling decision is based on network characteristics including RTT, windows size, delay, losses, etc. Some of these elemental schedulers include minRTT [19], BLEST (BLocking time ESTimation) scheduler [28]. BLEST uses head-of-line (HoL) blocking to minimise, and decide whether to schedule a packet on a certain subflow. Another is the earliest completion first (ECF) which takes into account the round trip time (RTT), congestion window sizes, sending buffer, and seeks to optimise performance when asymmetric multipaths are available [29].

[30] proposes three multipath schedulers: large window space (LWS) that schedules packets based on the most recent window size, high sending rate (HSR) that prioritizes the path with the earliest goodput and low time space (LTS) that takes the mean ratio between the RTT and the window size to select the best path. In [31], a blocking-time based scheduler is proposed. It calculates the path's blocking delay and then selects the appropriate sub-flow to schedule the packet. The shortest transfer time first (STTF) scheduler is proposed in [32] to address the shortcomings associated with asymmetry, which has been identified as one of the causes of poor scheduling performance. Round Robin schedules the packet on each interface one by one irrespective of the network conditions [33]. RR however does not use any characteristics of the paths in the scheduling decision, it leads to poor performance when the underlying network paths are heterogeneous [17].With the Low Latency scheduler, when two or more paths are available, it schedules the packets to the path with minimum delay [34]. The Random scheduler randomly distributes the packets among available paths. Peekaboo is based on reinforcement learning, it calculates the reward value for each path by formulating a multi-armed bandit problem and utilizing in-flight bytes, congestion window size and RTT for each path. The packets are schedule to maximize the overall reward [17].

In [35] the authors introduce the concept of probability in the scheduler combined with Forward Error Correction (FEC).

## 2.4 Error Correction Scheme

In information theory and coding theory, error detection and correction or error control are techniques that enable reliable delivery of digital data over unreliable communication channels. Many communication channels are subject to channel noise, resulting in the introduction of errors during transmission from the source to a receiver. Error detection techniques allow detecting such errors, while error correction enables reconstruction of the original data in many cases [36]. Figure 2.3 shows the hierarchy of the various error correction schemes.

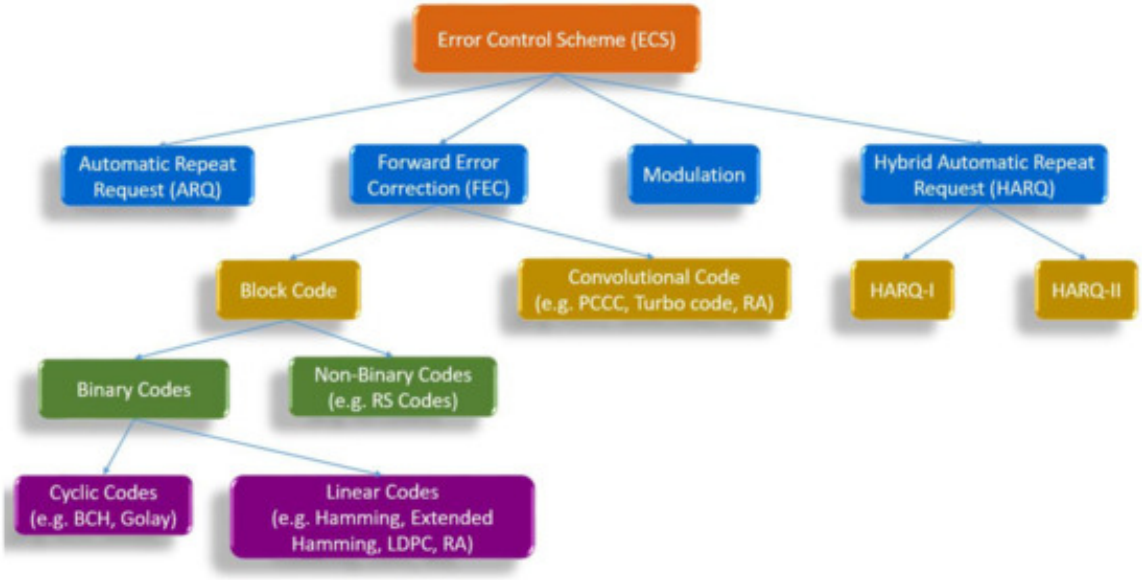



**Figure 2.3:** Error Correction Schemes [37]

### 2.4.1 Forward Error Correction (FEC)

The use of error-correcting codes in networking is sometimes referred to as forward error correction (FEC) because the correction of errors is handled in advance by sending extra information rather than waiting for errors to happen and dealing with them later by retransmission. FEC is commonly used in wireless networks such as 802.11. 802.11 networks apply forward error correction (FEC) to the transmitted packets so that some number of errors can be corrected by the receiver [38].

Forward error correction or channel coding is a classical approach that is used to enhance the performance of a communication link. FEC is used in

communication when retransmission is relatively very costly and delay sensitive. In FEC, Error-Correcting Codes (ECCs) add redundancy to the transmitted packet at the transmitter side that allows for the detection and correction of a certain amount of error at the receiver side. The error can be detected and corrected at the receiver. Hence, it eliminated the need of retransmission that is the key advantage of FEC. The introduction of FEC allows a system to operate at a significantly lower Signal to Noise Ratio (SNR) than an uncoded system to achieve a certain BER. The difference in SNR is called coding gain due to FEC and the coding gain depends on the type of ECC and the decoding algorithm and complexity. The complexity of the algorithm determines the decoding power consumption. However, the lower requirement of transmission power at the transmitter comes with extra costs: energy consumption due to encoding at the transmitter, decoding at the receiver, and the transmission of extra bits introduced by FEC due to redundancy [37]. Therefore, the use of FEC is only justified if extra power consumption introduced due to encoding, decoding, and transmission of extra bits is lower than the power saving due to the use of FEC.

There are different types of FEC available in the literature with varying complexity and performance. ECCs is classified into two types:

1. block codes

2. convolutional codes.

In block codes, the message to be transmitted is divided into smaller blocks of pre-defined length. Subsequently, these blocks are encoded into codewords using an encoder. Commonly, a block code is represented by the triple $(n, k, t)$, where $n$ is the length of code word bits, $k$ is the length of information bits in the code word, and $t$ is the error correction capability in terms of number of bits that can be corrected [39]. To make fair comparison between coded and uncoded system, there is the need to consider energy consumption due to Radio Frequency (RF) transmission of extra bits in coded system, encoding and decoding cost during result comparison. The energy consumption due to encoding at the transmitter, usually insignificant in many cases. Consequently, the main factors that should be considered are the decoding cost at the receiver (which depends on the type of decoding algorithm) and the transmission of extra bits due to redundancy that is introduced by FEC. Therefore, the comparison result between the coded and uncoded system should consider power per decoded bit, as well as the decoding cost of decoding algorithm [40].

A binary block code is linear if and only if the modulo-2 sum of the two codes produces a new codeword. Hamming codes [14] and extended Hamming

codes are categorized into linear binary block codes. Low-Density Parity Check (LDPC) [41, 42] and Repeat Accumulate (RA) [43, 44] are also linear block codes and these codes are considered as the most powerful codes exist in current literature. Another special class of the linear block code is cyclic code [45] where a circular shift of a code word results in a new codeword. Bose-Chaudhuri-Hocquenghem (BCH) [46, 27] and Reed-Solomon (RS) [47, 48, 49] are categorized in to binary and non-binary cyclic codes, respectively.

Convolutional code is another category of FEC, and it differs from block codes in many ways. First, the entire data stream is encoded into a single code word in convolutional code. Secondly, the encoder output $n$ at any given time not only depends on $k$ inputs but also $L$ previous input blocks i.e., constraint length in an $(n, k, L)$ convolution code. However, the definition of the code rate of convolutional code is the same as in block codes. There are various types of convolutional code and two common examples of convolutional codes are Parallel-Concatenated Convolutional Code (PCCC) [50], turbo code[51], and RA code. The encoding and decoding complexity depends on the type of code and the decoding algorithm used.

The encoding complexity of block code is very complex, with larger values of information bits when the code is not linear. The encoding complexity of block code significantly reduces with the use of linear block codes. The cyclic codes have special algebraic properties that make encoding easy and decoding implementation using components, such as exclusive-OR gates, switches, and shift registers. The implementation of encoder of convolution code is very simple and can be performed using shift registers. Normally, the encoding operation is much simpler than the decoding operation. Thus, the encoder consumes substantially less power than the decoder does, and the power that is consumed by the encoder can be ignored during the system evaluation. There are several types of decoding algorithms, and there is a tradeoff between coding gain and the complexity of decoding. The decoding algorithms can be categorized depending on different factors: codes, decoding operations, input from the channels, etc.

The decoding of short block codes, such as Hamming codes, can be performed by various methods:

1. syndrome decoding [52]

2. maximum likelihood (ML) decoding [53] to the nearest code word or Viterbi algorithm [54, 55]

3. maximum a posteriori (MAP) decoding [56] with the BCJR algorithm [57].

Decoding algorithms can be categorized according to decoding operations into iterative decoding and non-iterative decoding. Syndrome decoding and ML decoding using the nearest code word for short block codes, algebraic decoding used in RS and BCH codes, and Viterbi decoding and sequential decoding for convolutional codes lie in non-iterative decoding category. MAP decoding with the BCJR algorithm that is used in Turbo codes and the sum-product algorithm (SPA) or belief propagation (BP) that is used in LDPCs are categorized in the iterative decoding. The iterative decoding was further classified depending on the input to the decoding algorithms. If the decoder received the hard decision from the channel and the hard decision information used during the decoding iteration, then the decoder is referred to as hard-decision decoder. Whereas, if the decoder received the soft decision from the channel and the soft decision information used during the decoding iteration, then the decoder is referred to as soft-decision decoder. Soft decision decoder has better performance (i.e., coding gain) than the respective hard decision decoder. However, the soft decision decoder has higher complexity than the respective hard decision decoder. The FEC scheme involves PHY layer of OSI model.

FEC is a method commonly used to handle losses in real-time communication. FEC techniques enable a receiver to correct errors/losses without further interaction with the sender (Figure 2.4). An $(n, k)$ block erasure code converts $k$ source data into a group of $n$ coded data, such that any $k$ of the n encoded data can be used to reconstruct the original source data. Usually, the first $k$ data in each group are identical to the original $k$ source data; the remaining $(n - k)$ data are referred to as parity data) [58].

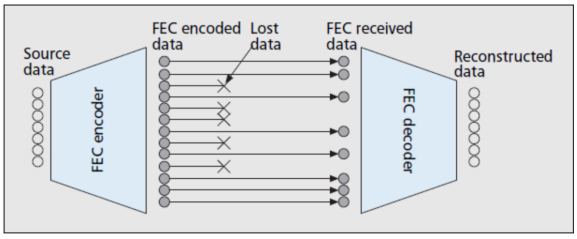



**Figure 2.4:** Forward Error Correction [58]

Usually, FEC codes are able to correct both errors and erasures in a block of $n$ symbols. In coding theory, an error is defined as a corrupted symbol in an unknown position, while an erasure is a corrupted symbol in a known position. In the case of streamed media packets, loss detection is performed based on the sequence numbers in Real-time Transport Protocol (RTP) packets (i.e., erasure codes). FEC can be done at many levels from byte level up to packet level. In byte-level FEC, a symbol is a byte; while in packet-level FEC, a symbol is a packet. Byte-level FEC is implemented at the physical layer of almost all

wireless networks [59, 60]. Packet-level FEC consists of producing $h$ redundant packets from $k$ original ones. An FEC packet is generally based on erasure coding [61], and its usefulness is due to:

- A single parity packet can be used to correct different single-packet losses in a group of packets [62].

- Byte-level FEC is unable to recover a completely lost or delayed packet.

When using byte-level FEC, a corrupted packet is already detected and discarded at the link layer with cyclic redundancy check (CRC), or at the transport layer with CHECKSUM, and so will not be available at the application level. Even though most existing wireless access networks use integrated physical layer adaptive coding and modulation schemes (e.g., IEEE 802.16a uses variable-rate RS/convolutional coding (CC) schemes and variable-modulation scheme), packet-level FEC protocols are usually required at the application level [58]. Wireless communication experiences both:

1. Short-term fast fading and white Gaussian noise, which is addressed by the integrated physical layer coding.

2. Long-term slow fading (e.g., when entering a tunnel), which is addressed by packet-level FEC encoding.

These two levels of FEC encoding are fully complementary, with each level addressing a different problem. However, there is a need for additional packet-level FEC protection to increase the reliability of multimedia communications in a wireless context.

# 3

# Methodology

This is a research thesis, and the focal point is a projection of future scenarios where a mobile device needs reliable, consistent very high data rate communication. Although this type of mmWave mobile network is part of 5*G*, it is more a work in progress in its deployment at the moment, so this will be tested via simulations. Real deployed networks that can do what is intended may be a few years away.

## 3.1 Research Questions

This work seeks to answer the following question:

- What effect does the packet scheduler have on the performance of the proposed multi-path mmWave proxy?
- Does forward error correction significantly improve performance of the scheduler?

These questions are answered by means of an event-based Simulator.

## 3.2 Components of the Simulator

This simulator is based on the work done by David et al. in [13]. The programming language of choice was Julia. This simulation is considering a scenario where a mobile real-time interactive application such as immersive 3D video, UHD augmented reality and other delay-sensitive applications needs to communicate at a constant rate of $2Gbps$ with a very low delay. A sender seeks to send at, $2Gbps$ and there are up to 8 available mmWave paths with varying capacities and NLoS/LoS rates.

The event-based simulator models packet transmissions. transmissions. The simulator model chooses a scenario of a mobile device in a city landscape surrounded by buildings by varying the path loss according to the standard UMi - Street Canyon model. This is a more challenging landscape for mmWave channels, but a likely scenario for dense mmWave deployment, providing a wide selection of possible paths to different base stations. The capacity of each path is calculated using the Shannon-Hartley theorem, with a base signal-to-noise ratio (SNR) discounted by the path loss model. The channel model is primed by calculating the SNR that will yield a target channel rate of $2Gbps$ during LoS operation at a distance of $60m$ from the base station. The LoS/NLoS state of each channel is updated independently according to a two-state Markov model.

### 3.2.1 Julia Programming Language

Julia combines expertise from the diverse fields of computer science and computational science to create a new approach to numerical computing [63]. Modern language design and compiler techniques make it possible to mostly eliminate the performance trade-off and provide a single environment productive enough for prototyping and efficient enough for deploying performance-intensive applications. Julia shows that one can achieve machine performance without sacrificing human convenience. The Julia programming language is a flexible dynamic language, appropriate for scientific and numerical computing, with performance comparable to traditional statically-typed languages.

Because Julia's compiler is different from the interpreters used for languages like Python or R, it is easy to write code that's nearly as fast as C [64]. Julia features optional typing, multiple dispatch, and good performance, achieved using type inference and just-in-time (JIT) compilation, implemented using LLVM. It is multi-paradigm, combining features of imperative, functional, and object-oriented programming. Julia provides ease and expressiveness for high-level numerical computing, in the same way as languages such as R, MATLAB, and Python, but also supports general programming. To achieve this, Julia

builds upon the lineage of mathematical programming languages, but also borrows much from popular dynamic languages, including Lisp, Perl, Python, Lua, and Ruby [65].

### 3.2.2 Simulator Structure

The event-based simulator tests the efficacy of various predictive multipath mmWave proxy mechanisms proposed in [13] coupled with scheduling mechanisms proposed in this work. The simulator uses the Event Simulation module, which is an event-based discrete event simulation engine. The purpose of this work is to further the objective of achieving reliable, consistent communication. The path management aspect of the proxy has been done as presented in [13]. The work done in this thesis is based on this. The diagram in Figure 3.1 gives a high level overview of the simulator.

The work done on the path management treated the proxy queue as a SimRource, however to suit the purpose of this work, the proxy queue is defined as a SimQueue.The simulation engine using two abstract reservoirs i.e., SimResource and SimQueue. SimResource is for holding numeric values (like amount of liquid). It stores current quantity of matter and its allowed lo (minimum quantity of resource) and hi (maximum quantity of resource) amounts. Servers can get matter from the resource with optional maximum number of requests pending for fulfillment. SimQueue holds arbitrary objects. It allows objects to be waiting in a queue with optional maximum queue size. Servers can get objects from the queue with optional maximum number of requests pending for fulfillment. Requests for both resource types are either treated as FIFO or LIFO based on configuration.

For a more thorough study and analysis, it is beneficial to have data on properties such as the sequence number, arrival time, packet size and designated path of each packet at various stages in the simulation. Thus, SimQueue was chosen as the preferred resource type for the proxy queue in this work due to its object-oriented nature.

### 3.2.3 Simulation Flow

A packet arriving in the proxy queue from the sender (source) goes through various stages in the system before being acknowledged as received at the receiver (application). The simulation runs for 70 with a single queue which has a maximum capacity of 10000 packets.

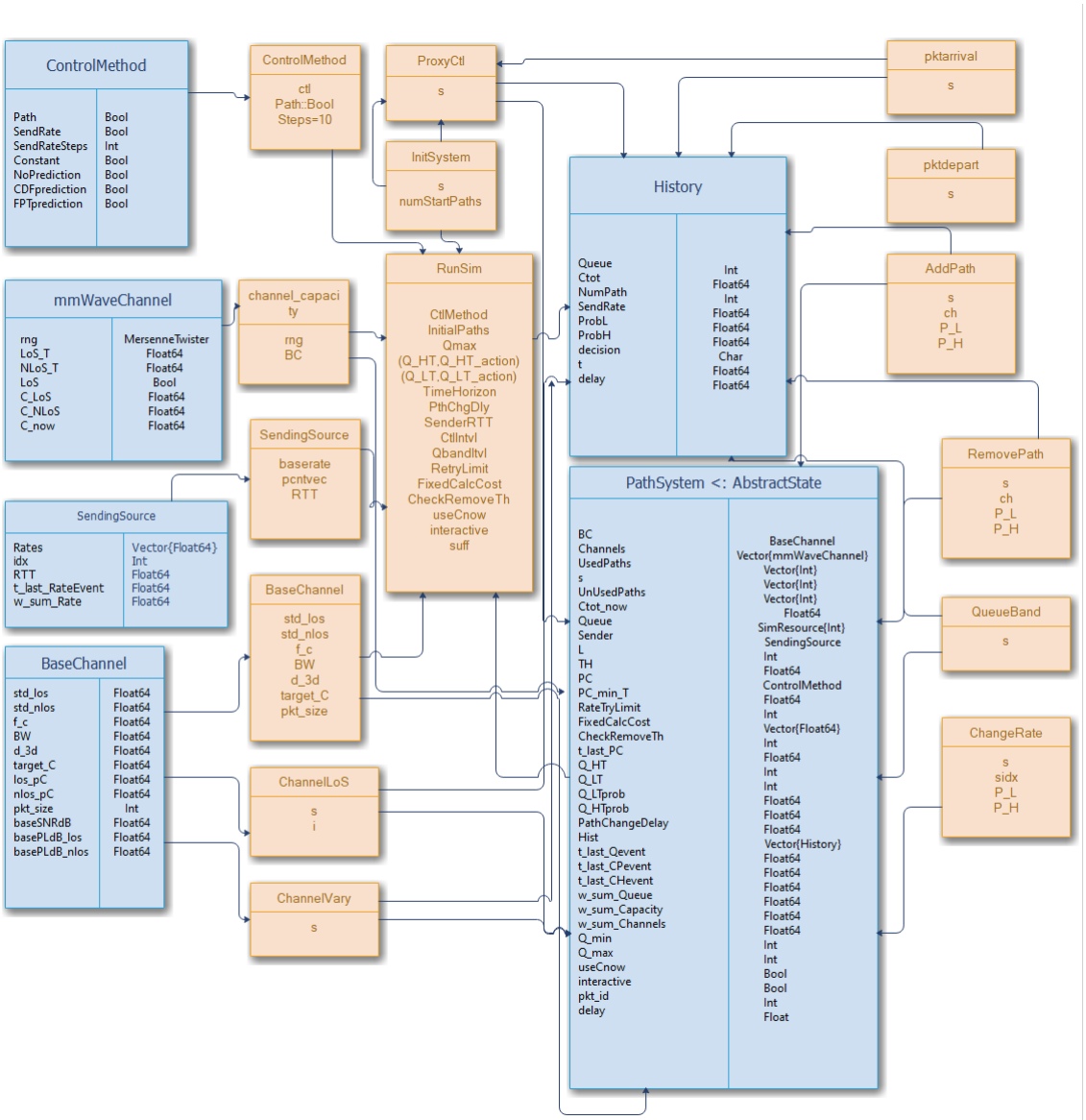



**Figure 3.1:** Simulator Overview

### Sending Source

It is assumed that the sender is sending packets at a constant rate, i.e., $2Gbps$.

### Packet Arrival

When a packet arrives, it is timestamped and placed in the packet queue; A packet arriving in the proxy queue has four parameters:

1. Sequence number
2. Arrival time
3. Packet size
4. Route

This makes it possible to have unique packets which can be tracked at every point in the simulation. A packet arriving when the queue is full is lost. For the sake of simplicity in the simulation, sequence numbers are assigned to packets only when they enter the proxy queue. The lost packets, i.e., packets arriving to a full queue, are tracked by a counter.

### Packet Departure

The single proxy queue is served by up to 8 paths based on the output from the path management. Packets are taken off the queue in a FIFO manner. A packet can only be taken off the queue if there is an available path, i.e., the path is not currently transmitting a packet. The path manager determines how many paths are available to serve the queue based on their status.

### Packet Delivery

Packets reach the receiver in a different sequence depending on the path used for transmission. When a packet arrives at the destination, two delay values are recorded:

1. Time taken for the packet to reach the receiver (application) irrespective of the sequence.
2. Time taken for the packet to reach the receiver (application) in the desired sequence.

Since packets arrive in varying order, a buffer is created at the receiver to hold out of sequence packets. Thus, a packet transmitted on a slow path could cause head of line blocking, subsequently leading to packets being delayed.

### 3.2.4 Simulation scenarios

The capacity of the various paths vary based on their LOS/NLOS status. The following scenarios are considered for the path management:

1. Fixed paths: This is the simplest scenario in terms of control, since there is no dynamic path selection. It is assumed that the chosen paths sustain an average channel capacity higher than the target rate of $2Gbps$.

2. Reactive control: If an arriving packet causes the proxy queue to cross a threshold, a path with the highest available capacity is added. If an arriving packet finds the proxy queue empty, remove the lowest-capacity path from the set of used paths.

3. Distribution based Predictive control : based on the proxy queue distribution (i.e., the probability that the queue will be less than a particular threshold over the time horizon.

4. First Passage Time (FPT) based Predictive control: based on the probability of the queue crossing particular thresholds within the time horizon, i.e., the probability that the first passage time is within the time horizon.

For each of these scenarios, a different scheduling mechanism is deployed and the behavior of the system is studied. For repeatable results, simulations are ran 100 times, each time with a different scenario by changing the random number generator seed. The simulations are ran with different seeds in order to have results for varying scenario. The results are validated if different scenarios produced the desired result.

## 3.3 Scheduling Algorithms

The scheduler determines how to distribute data onto the available paths. The data packets from the sender reside in the proxy queue, and the scheduler assigns each packet to a different path based on a particular scheduling mechanism. One of the significant challenges for designing a multipath scheduler for this mmWave proxy is to deploy a mechanism, which deals with the heterogeneous characteristics of the paths. When the paths are constantly changing, especially in terms of delay and loss, sent packets will arrive to the destination out of order, leading to head of line (HoL) blocking, ultimately reducing the performance.

In this work, four scheduling mechanisms have been considered. The ideal

simple Round Robin, a Random scheduler, Highest Capacity First scheduler, and a proposed scheduler. Subsequently, Forward Error Correction is implemented to investigate the impact it could have on the scheduling. A common baseline for multipath scheduler evaluation is the Round-Robin scheduler algorithm, which cyclically transmits packets over each path, as long as there is space in the proxy queue [17]. Since RR does not use any characteristics of the paths in the scheduling decision, it leads to poor performance. The performance of the Random scheduler is also poor similar to RR for the same reason, however with RR there is a "fair" distribution of packets across the various paths. The eratic nature of the Random scheduler leads to an unpredictable path utilization. Because the HCF scheduler opts for the path with the highest capacity, HoL blocking can be reduced. In terms of channel utilization, RR does a better job compared to HCF however, the priority is to achieve better delay and not necessarily better path utilization.

Thus, to better judge the performance of the proposed scheduler, these three mechanisms (RR, Random, and HCF) can serve as good baselines.

### 3.3.1 Round Robin Scheduler

This is a simple scheduler which selects paths cyclically. Utilization of the paths is fairly uniform. The Round Robin implementation is depicted in Algorithm 1. This scheduler is chosen as a baseline because it does nothing special apart from cyclically utilizing the available paths. Selection of the next path is based on the last path used. If a selected path is busy, the next path is chosen and if only one path is available (especially in the case of the fixed paths) the scheduler selects it for the next packet transmission.

### 3.3.2 Random Scheduler

The Random scheduler as shown in Algorithm 2 is also relatively simple as with the Round Robin scheduler, however the selection of paths is random in nature. Path utilization is therefore not uniform, unlike Round Robin. The path chosen previously does not influence the path chosen. Selection is completely randomized and just like in the case of RR, when only a single path is available, it becomes the chosen path.

### 3.3.3 Highest Capacity First Scheduler

The Highest Capacity First scheduler (Algorithm 3) is selected to improve the possibility of achieving lower packet delay. Path utilization is non-uniform just

**Algorithm 1:** Round Robin Scheduling

```
for i ← 1 to Number of Paths do
    if Queuesize > 0 then
        for i ← 1 to NumberofPaths do
            route = mod1(Lastroute + 1, NumberofPaths);
            if route is available then
                Chosenroute ← route;
                Lastroute ← route;
                Set no path flag off;
                break
            else
                Lastroute ++;
                Set no path flag on;
            end
            if no path flag is off then
                Take a packet off the queue;
                Update route field of the packet;
                Register packet depart;
                Set path to unvailable;
            end
        end
    end
end
```

**Algorithm 2:** Random Scheduling

```
for i ← 1 to Number of Paths do
    if Queuesize > 0 then
        Select a random path;
        if path is available then
            Chosenpath ← randompath;
            Take a packet off the queue;
            Update route field of the packet;
            Register packet depart;
            Set chosen path to unvailable;
        end
    end
end
```

like the Random scheduler because the capacity of the various paths is reliant on the LOS/NLOS state. By choosing the path with the highest capacity among the available paths first, fewer packets are expected to be delivered to the

receiver out of sequence. This invariably would reduce HoL blocking and, by extension, the packet delay.

**Algorithm 3:** Highest Capacity First Scheduling

```
for i ← 1 to Number of Paths do
    if Queuesize > 0 then
        for i ← 1 to NumberofPaths do
            Compute capacity for each path;
            if route is available then
                Chosen path ← pathwithhighestcapacitybreakend
                if no path flag is off then
                    Take a packet off the queue;
                    Update route field of the packet;
                    Register packet depart;
                    Set path to unvailable;
                end
            end
        end
    end
```

### 3.3.4 Proposed Hybrid Scheduler

This mechanism is a hybrid of the highest capacity first scheduler and the BLEST scheduler. The estimated time of arrival is calculated. In situations where a packet has been sent on a relatively slower path, the algorithm considers the estimated time of arrival (ETA) of the current packet. If the ETA of the current packet is significantly shorter than that of the packet in transit, it might be reasonable to send the packet on a slightly slower path to reduce the jitter and head of line delay. If no path is found such that HoL blocking can be avoided, the highest capacity path is chosen.

Thus, when total channel capacity is high enough, the proposed hybrid scheduler works like the Highest Capacity First scheduler. The mechanism for the proposed hybrid scheduler is depicted in Algorithm 4.

### 3.3.5 Random Scheduler with Forward Error Correction

The simulation is set up in a similar fashion as before. However, with the implementation of FEC, a buffer of size $N+M$ is created before the proxy queue. Thus, the packets are added to the proxy queue in blocks of $N + M$.

**Algorithm 4:** Proposed Hybrid Scheduling

```
Compute capacity of all available;
for i ← 1 to Number of Paths do
    if Queuesize > 0 then
        if ETA of previous packet < ETA of next packet then
            if no path flag is off then
                Select highest capacity path;
                Take a packet off the queue;
                Update route field of the packet;
                Register packet depart;
                Set path to unvailable;
            end
        else
            for i ← 1 to Number of Paths do
                if ETA of previous packet < ETA of next packet then
                    Take a packet off the queue;
                    Update route field of the packet;
                    Register packet depart;
                    Set path to unvailable;
                    break;
                else
                    Send on highest capacity path anyway;
                end
            end
        end
    end
end
```

For practical reasons, we will only consider block codes where the transmission of $N$ input packets is complemented by that of $M$ redundancy packets. If at least $N$ packets out of $N + M$ are received correctly, then all $N$ input packets can be retrieved. If fewer than $N$ packets out of $M$ are received, we cannot gain advantage from the redundancy, but we can at least retrieve the fraction of the initial $N$ packets which made their way to the receiver.

In each case, the amount of additional overhead is M/(N + M). The question that we set up to solve is whether the gain of a reduced error rate is worth the pain of this additional overhead, as well as the cost of implementing the redundancy. To test the efficacy of the FEC, it will be implemented with the random scheduler, which does not have any clear advantage on its own.

# 4

# Results and Discussion

The figures in this chapter show the simulation results of the four scheduling mechanisms described in chapter 3. For each scheduler there is a graph of the number of paths being used, the available capacity of the used paths, the queue size (sampled every 200 ms, but plotted as a band of max to min achieved in that period), and delay plotted also as a band of max to min. For these plots, the same random number generator seed was used for consistent results for comparison. Thus the same scenario is used when comparing the various methods. However, with the CDF plots of the delay, 100 different scenarios (each with a different seed) were simulated in order to have repeatable results.

## 4.1 Round Robin (RR)

Results for the comparison of the CDF plot for the delay of packets arriving in sequence versus the packets arriving out of sequence is presented in Figure 1 in the Appendix. In the fixed-paths scenario (Figure 4.1), It can be seen that the choice of 3 paths does yield an aggregate average capacity of $3.33Gbps$, way higher than the $2Gbps$ sent by the source. However, there is a non-negligible chance of one or more paths being in NLoS, and the combined capacity being less than $2Gbps$. As a result, despite the high average capacity, congestion occurs and, worse, bursty packet losses — since the aggregate capacity sometimes falls below the $2Gbps$ target for fairly long periods. As a result, very high delay values (up to $350ms$) are recorded. The average queue size over

the simulation time was 1106 packets. Due to the congestion (queueing delay) there is a significant gap between the packet delay for out of sequence and in sequence delivery (Figure 1).

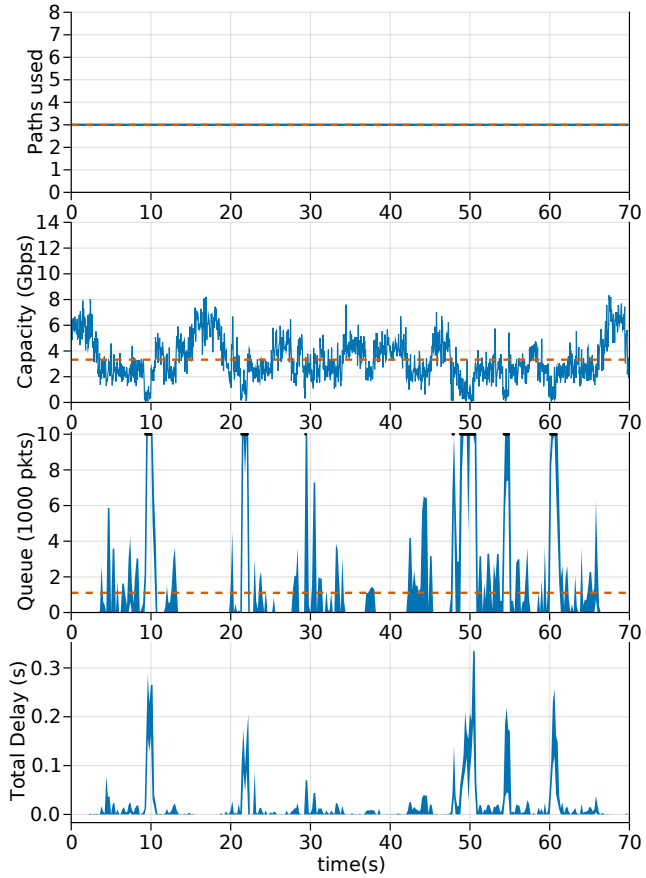



**Figure 4.1:** Simulation results using RR and 3 fixed paths.

Having a path-control policy is integral to improving the situation. The result of a lack of this is illustrated, in Figure 4.2. Here, RR is used with a simple queue threshold based scheme (no prediction). This results in an average number of paths of 1.73. One path is often enough, if it is in LoS, and the resulting aggregate average capacity of 3.06$Gbps$ is slightly lower than in the fixed-paths scenario. Even though the average queue is slightly shorter, and the losses are much less bursty than for a fixed number of paths, there remains extensive queueing delays for much of the time. The abrupt changes in capacity due to LoS/NLoS and shadow fading combined with the time for path changes to take effect cannot be mitigated by a purely reactive control. The average queue size of about 685 packets correlates with a relatively lower peak delay (about 150$ms$) as compare the scenario with fixed number of paths. A CDF plot

(Figure 2 of packet delay for out of sequence and in sequence delay shows a relatively similar delay as compared to the fixed paths case.

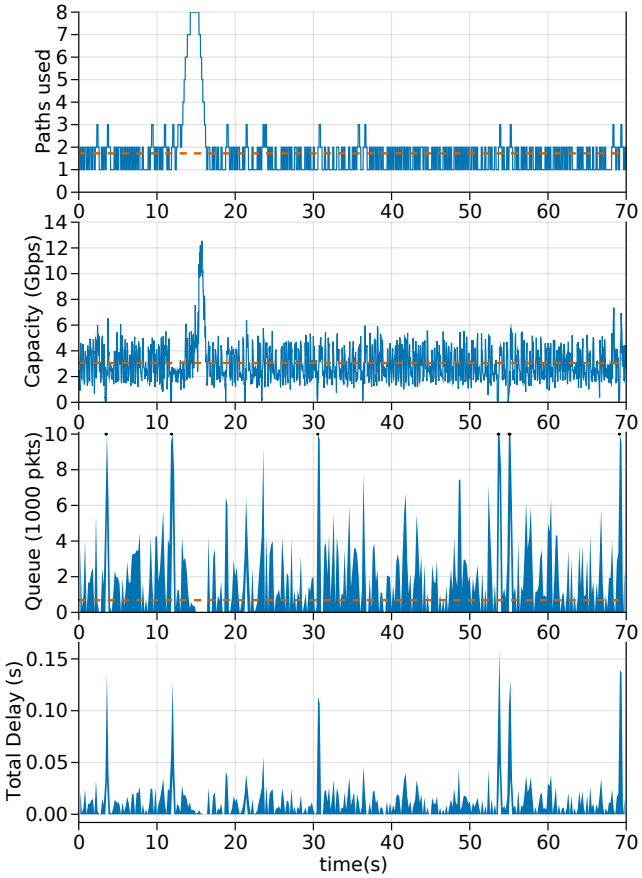



**Figure 4.2:** Simulation results using RR and no prediction of paths.

The results with a predictive CDF based controller are shown in Figure 4.3. By probabilistically predicting the queue distribution over a short time in the future, rather than just reacting to it, this controller is able to keep a very short queue and avoid losses altogether. The controller adds an extra path if the queue CDF over the next 200 ms is predicted to have more than a 1% chance of being over the 500 packet threshold, and removes a path if the queue CDF has a more than 99% chance of being below 250 packets. As a result of the predictive control, an average of 3.70 paths were used and an average capacity of 5.98$Gbps$ (about 3 times the send rate). The high capacity meant a smaller average queue size (76 packets) compared with the fixed paths and no prediction scenarios. It can be seen in Figure 3 that there was about a 90% chance of having delay values below 1$ms$ (Figure 3) as compared to 100$ms$ (Figure 1) and 10$ms$ (Figure 2) in the fixed paths and the no predictions case respectively.

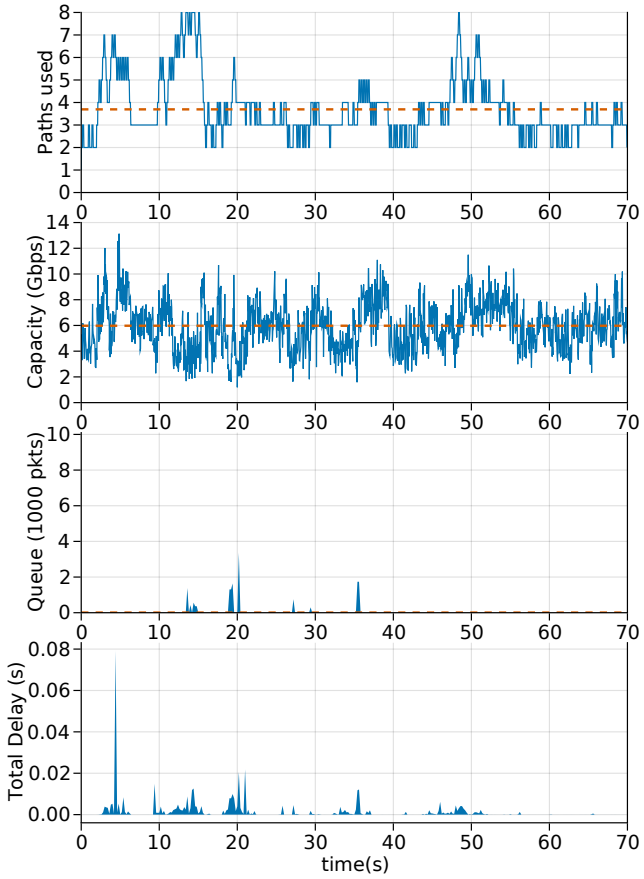



**Figure 4.3:** Simulation results using RR and CDF prediction of paths.

The predictive FPT based controller also manages to maintain a reliable, consistent capacity (Figure 4.4) similar to the CDF based controller. The average number of paths was 3.72, with an average path capacity of 6.05$Gbps$, and an average queue size of 19 packets. The choice of which one to use in practice would depend on which best represents the particular QoS agreement/requirement of the application. E.g. Does an application require a limited queue distribution? then CDF. Or does an application have a strict queue limit? then FPT [13].

## 4.2 Random Scheduler

The simulation results produced by the Random scheduler were similar to RR (Figures 5, 6, 7, 8, 9, 10, 11, and 12) in the Appendix.

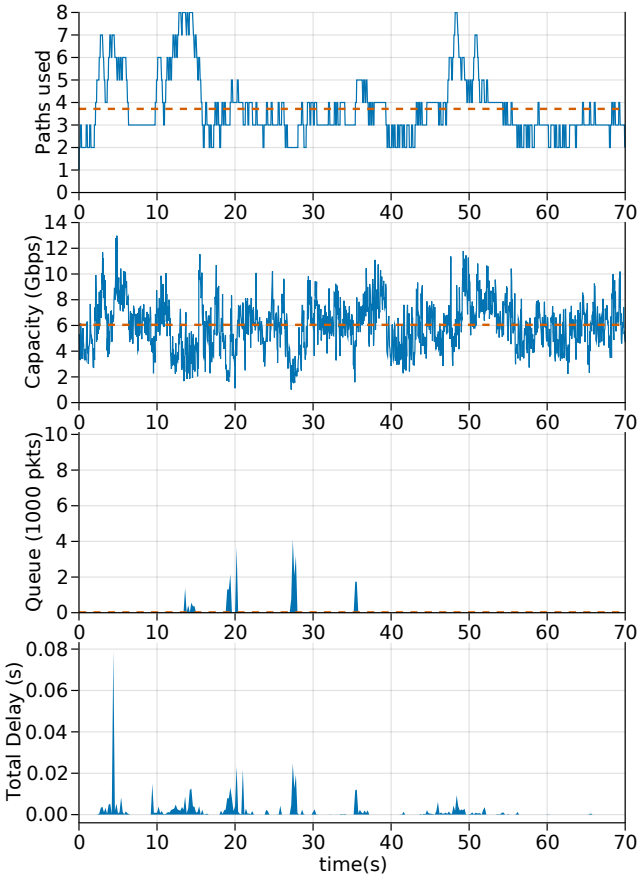



**Figure 4.4:** Simulation results using RR and FPT prediction of paths.

## 4.3 Highest Capacity First (HCF)

From the results seen in RR, Random, and HCF (Figure 4.5, the fixed path scenario produces very similar results in terms of losses and queue build up. The main difference can be seen in the path predictive methods (CDF and FPT). The effects of the scheduling mechanisms can be seen better in the CDF plots, which shines more light on the delay distribution. CDF plots are shown in the appendix (Figures 13, 14, 15, and 16.

## 4.4 Proposed Hybrid Scheduler

In the case of the fixed paths, the proposed scheduler has performance similar to the RR, Random, and HCF for reasons explained in the previous section. Using three fixed paths (Figure 4.9), for an average capacity of 3.33$Gbps$ the

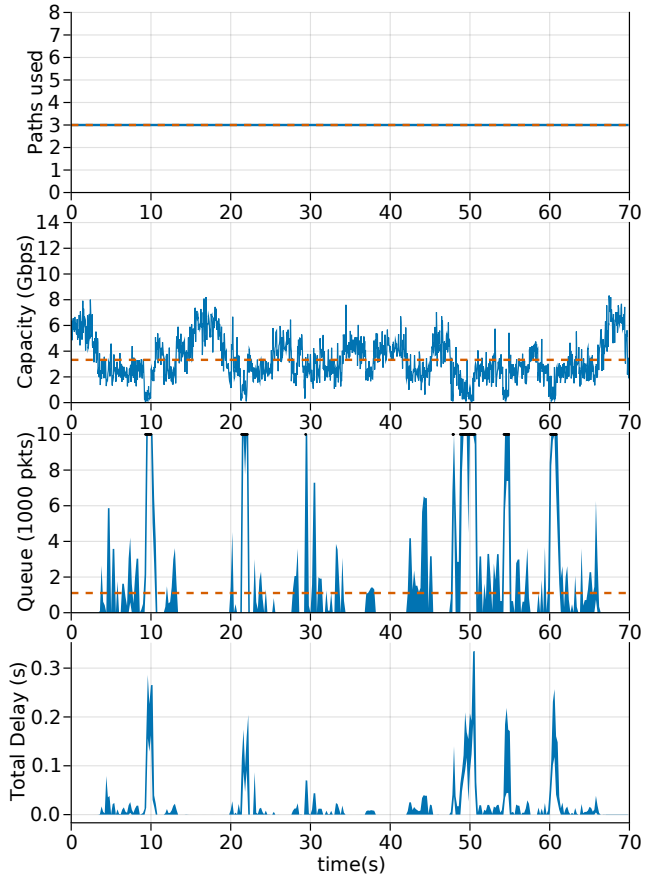



**Figure 4.5:** Simulation results using HCF Scheduling and fixed number of paths.

average queue size was 1114 packets.

An average of 1.6 paths were used with no prediction, resulting in an average capacity of 2.94$Gbps$ and an average queue size of 679 packets. This is comparable to 1.59 paths, 2.95$Gps$ average capacity and average queue size of 723 packets using HCF (Figure 4.6). Peak delay is recorded at about 150$ms$ as compared to 160$ms$ with HCF and fixed paths.

Using the CDF predictive method (Figure 4.11), about 38$ms$ peak delay, while a peak value of about, 20$ms$ is attained with HCF. The average number of paths, average capacity, and average queue size were 3.72, 6.01$Gbps$, and 7 packets respectively with HCF using the CDF predictive method (Figure 4.7).

Using the FPT predictive method (Figure 4.12), about 23$ms$ peak delay, while a peak value of about, 24$ms$ is attained with HCF. The average number of paths, average capacity, and average queue size were 3.73, 6.08$Gbps$, and 18 packets

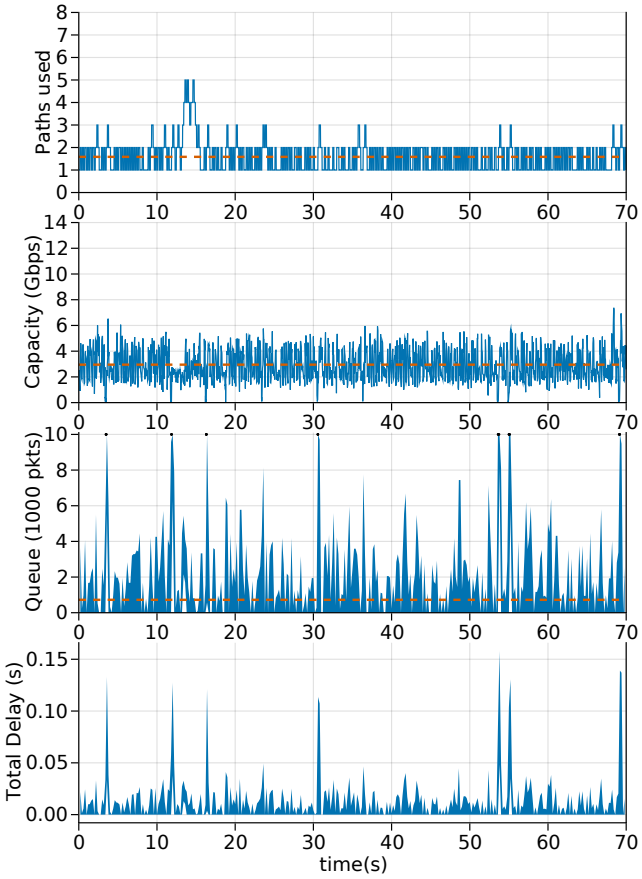



**Figure 4.6:** Simulation results using HCF Scheduling and no prediction of paths.

respectively with HCF using the FPT predictive method (Figure 4.8) while the proposed hybrid scheduler had an average of 17 packets.

CDF plots for the comparison between out of sequence and in sequence are shown in the appendix (Figures 17, 18, 19, and 20).

## 4.5 Comparison of the Four Scheduling Mechanisms

The plot in Figure 4.13 shows a comparison among the RR, Random, HCF and the proposed hybrid scheduler considering the delay for packets arriving in sequence. Packet loss is at about 8%. From Figure 4.13 delay values beyond the $80th$ percentile are queuing delays. Thus, the choice of scheduler has no effect on them. Up to the $60th$ percentile, HCF and the proposed hybrid scheduler have the same performance. These are situations where there is

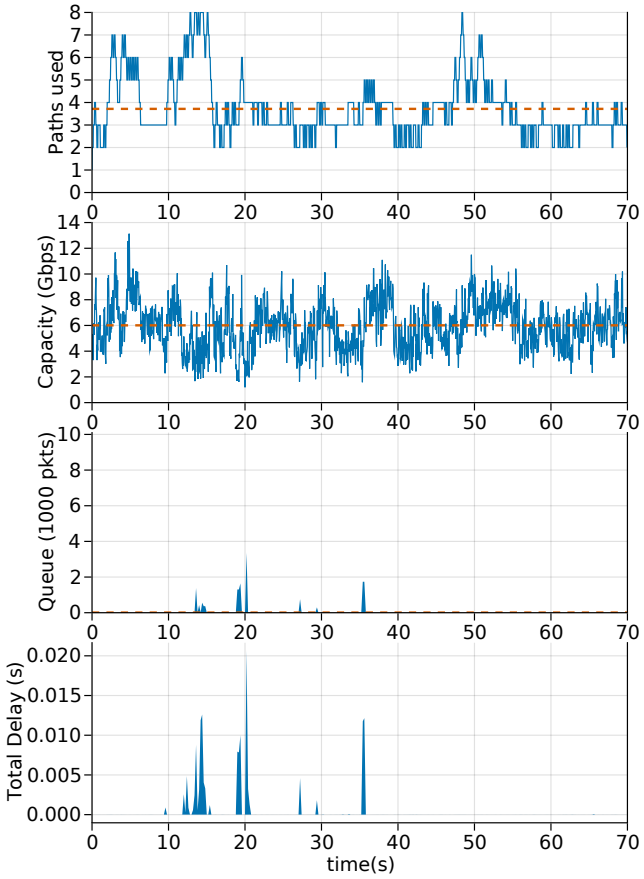



**Figure 4.7:** Simulation results using HCF Scheduling and CDF prediction of paths.

enough capacity and so the hybrid scheduler works like the HCF. Between the $60th$ and $70th$ percentiles, the differences between the HCF and Hybrid scheduler can be seen. The Random scheduler outperforms the RR scheduler slightly (below the $40th$ percentile).

In the no prediction case, as shown in Figure 4.14, there are no significant gains when the Hybrid scheduler is compared to HCF. Packet loss is at about 2%. The difference between the various schedulers can be seen between the $40th$ and $70th$.

For the two predictive methods, CDF and FPT shown in Figures 4.15 and 4.16 respectively, packet loss is below 1%. There is a clear distinction in the pair of HCF and Hybrid as compared to RR and the Random scheduler. Queuing delay is quite insignificant here due to high capacity availability.

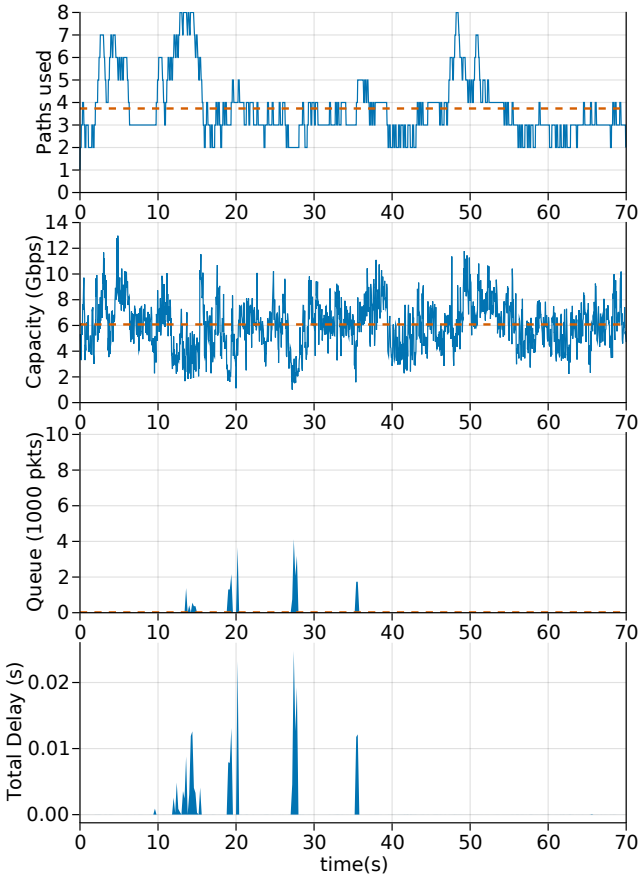



**Figure 4.8:** Simulation results using HCF Scheduling and FPT prediction of paths.

## 4.6 Random Scheduling with Forward Error Correction

From Figure 4.17 and 4.18, FEC implemented with the Random scheduler provide quite consistent results. Sending of packets in blocks instead of individually accounts for the Random scheduler (without FEC) having better performance for below the 60th percentile. Packets incur extra delay before entering the proxy queue.

Figure 4.19 and 4.20 show the performance of FEC with different N and M values. N = 9, M = 1 seem to give better results in most situations. Results for the other N and M values are shown in the Appendix (Figures 21, 22, 23, and 24)

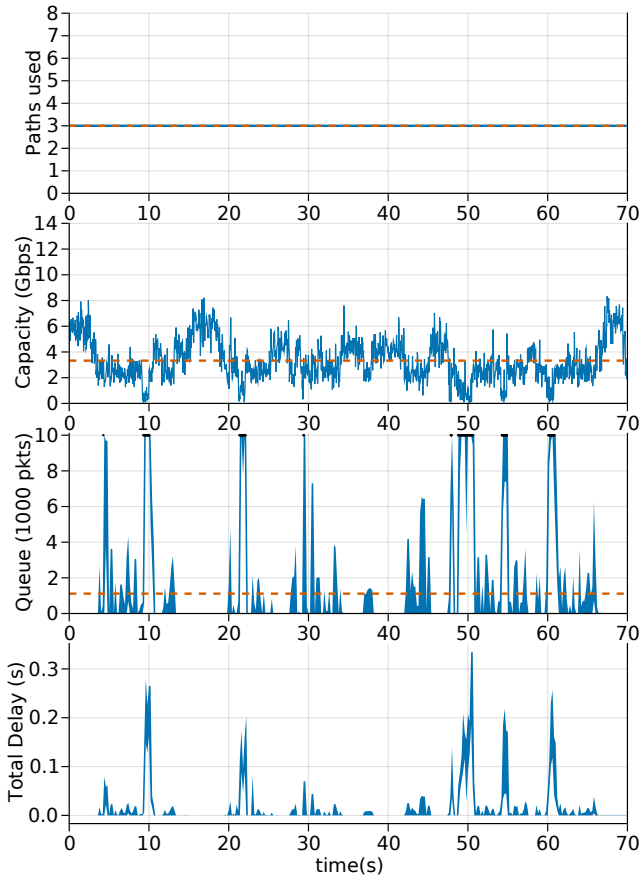



**Figure 4.9:** Simulation results using Proposed Hybrid Scheduling and fixed number of paths.

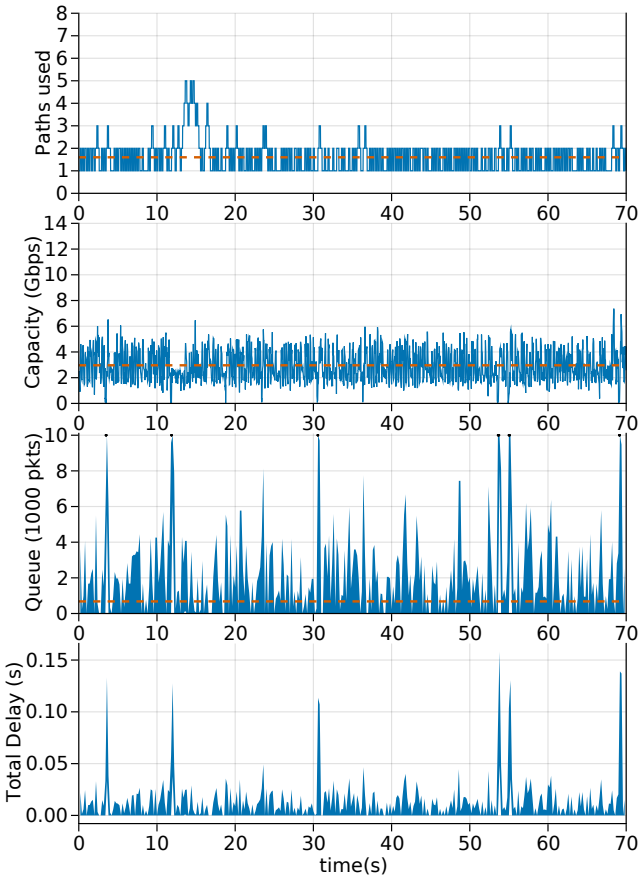



**Figure 4.10:** Simulation results using Proposed Hybrid Scheduling and no prediction of paths.

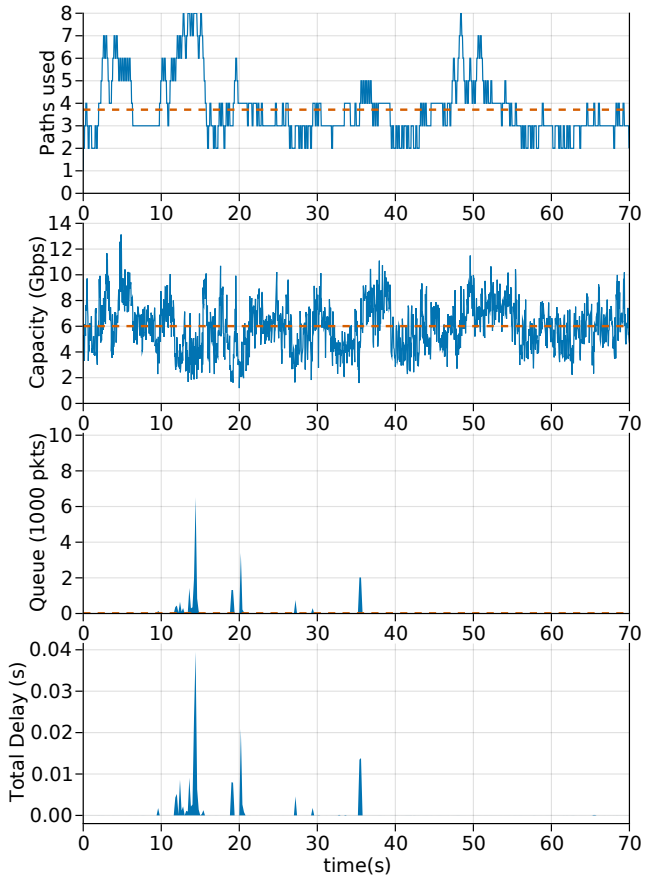



**Figure 4.11:** Simulation results using Proposed Hybrid Scheduling and CDF prediction of paths.

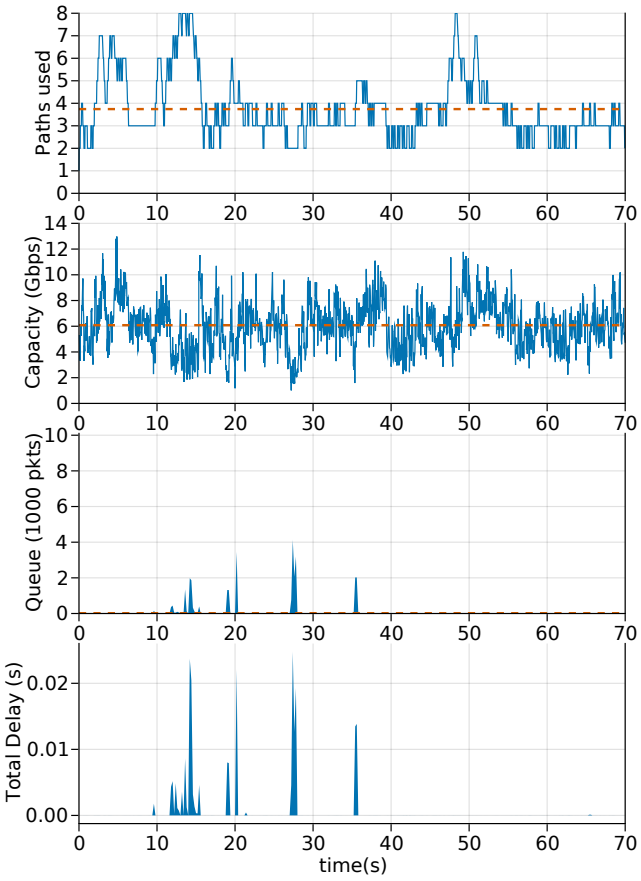



**Figure 4.12:** Simulation results using Proposed Hybrid Scheduling and FPT prediction of paths.

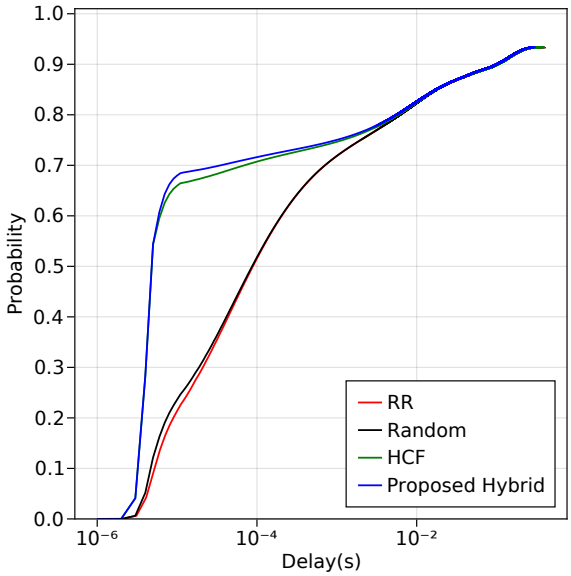



**Figure 4.13:** CDF plot of comparison of the various scheduling mechanisms with fixed paths.

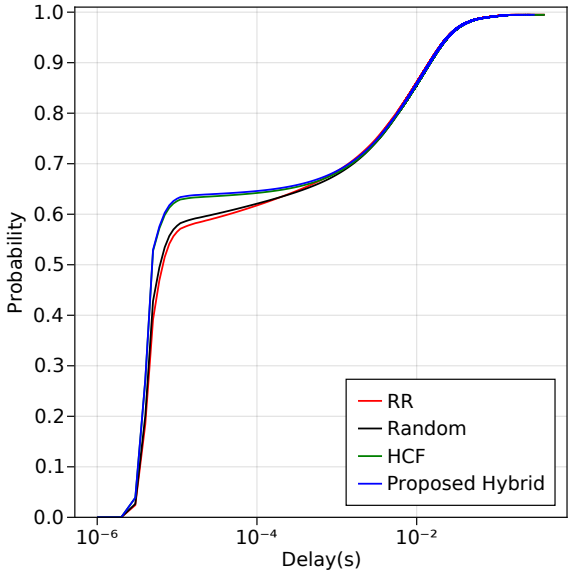



**Figure 4.14:** CDF plot of comparison of the various scheduling mechanisms with no prediction.

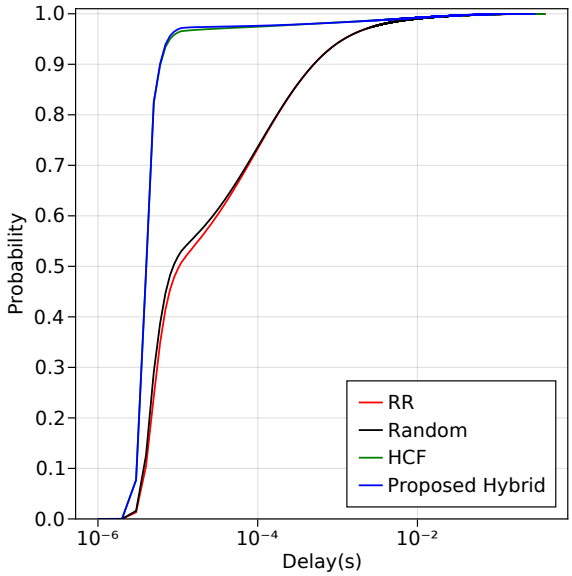



**Figure 4.15:** CDF plot of comparison of the various scheduling mechanisms with CDF prediction of paths.

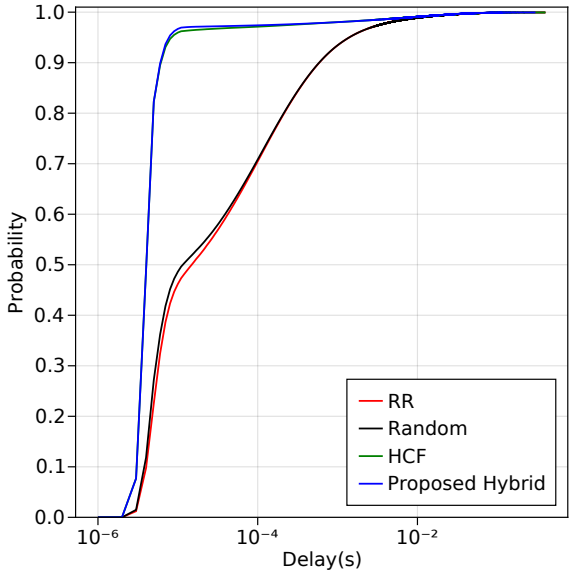



**Figure 4.16:** CDF plot of comparison of the various scheduling mechanisms with FPT prediction of paths.

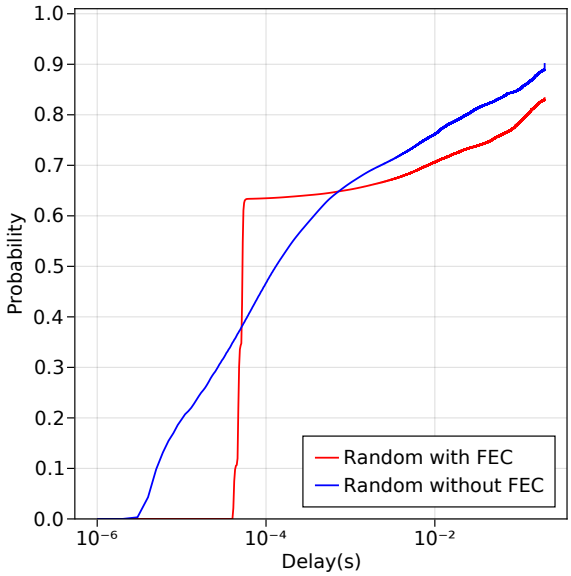



**Figure 4.17:** CDF plot of the Random Scheduler with and without FEC and fixed paths.N=9, M=1

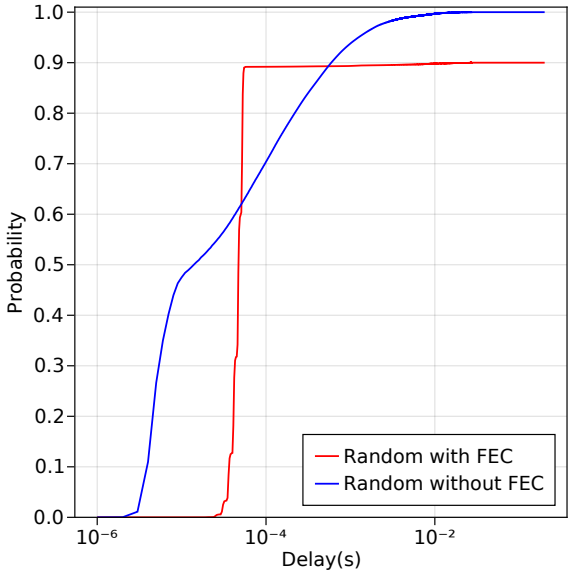



**Figure 4.18:** CDF plot of the Random Scheduler with and without FEC and with CDF path prediction.N=9, M=1

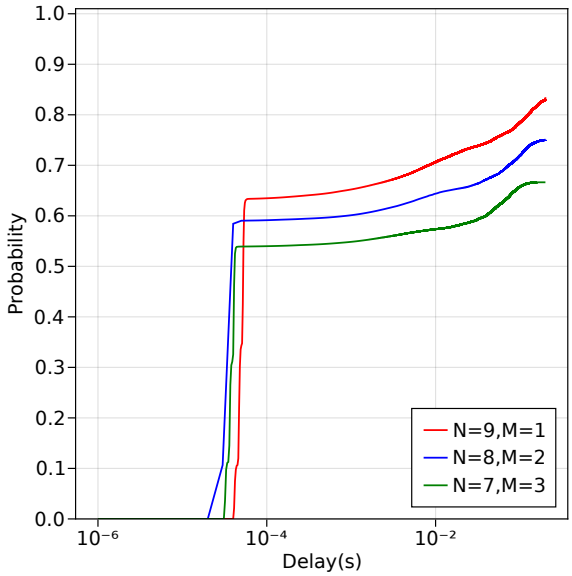



**Figure 4.19:** Comparison of (N=9, M=1), (N=8, M=2) and (N=7, M=3) using fixed paths

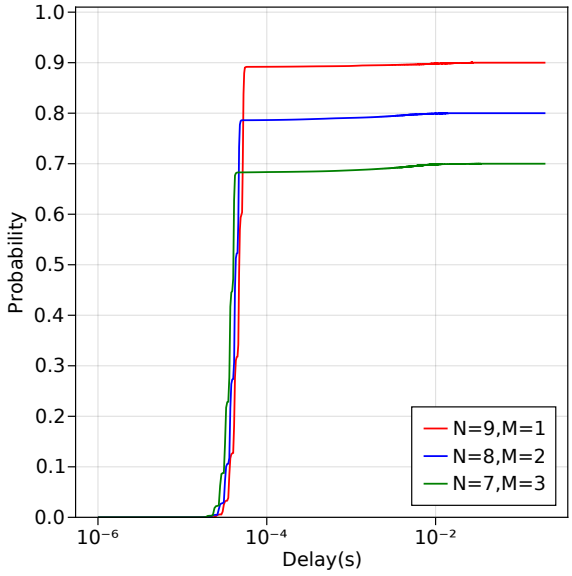



**Figure 4.20:** Comparison of (N=9, M=1), (N=8, M=2) and (N=7, M=3) with CDF prediction

# 5

# Conclusion

Reliable, consistent and very high data rate mobile communication will become especially important for future services such as, among other things, future emergency communication needs. MmWave technology provides the needed capacity, however lacks the reliability due to the abrupt capacity changes any one path experiences. Intelligently making use of varying numbers of available mmWave paths, perhaps even through multi-operator agreements; and balancing mobile power consumption with path costs and the need for reliable consistent quality will be critical to attaining this aim.

In this thesis, we consider the multipath scheduling problem in a mmWave proxy when the paths have dynamically changing path characteristics. To address this problem, we propose a hybrid scheduler, the performance of which was compared with the Round Robin scheduler, Random scheduler and the Highest Capacity First scheduler. The proposed scheduler consistently offers superior or similar performance to the other three schedulers.

Subsequently, forward error correction using block codes is implemented with the random scheduler. The results show that there is a trade-off to be made when using FEC. In some cases, the extra overhead produced a more consistent delay distribution, while in others the extra overhead overshadowed any gains made. In the simulation, choosing $N = 9$ and $M = 1$ gave the best compromise between overhead and delay performance. The choice of the size of redundancy (M) is, therefore, crucial to the performance of FEC. A combination of efficient scheduling and an appropriate error correction mechanism can contribute to

improved performance.

As future work, it would be interesting to explore how Automatic Repeat Request (ARQ) and Forward Error Correction could be used to deliver reliable, and consistent communication via this proxy.

# Bibliography


[1] 3GPP, "Release 15." `https://www.3gpp.org/release-15`.

[2] A. Ghosh, A. Maeder, M. Baker, and D. Chandramouli, "5g evolution: A view on 5g cellular technology beyond 3gpp release 15," *IEEE access*, vol. 7, pp. 127639–127651, 2019.

[3] S. Teral, "5g best choice architecture," *IHS Markit*, 2019.

[4] A. Dogra, R. K. Jha, and S. Jain, "A survey on beyond 5g network with the advent of 6g: Architecture and emerging technologies," *IEEE Access*, vol. 9, pp. 67512–67547, 2020.

[5] M. A. Habibi, M. Nasimi, B. Han, and H. D. Schotten, "A comprehensive survey of ran architectures toward 5g mobile communication system," *IEEE Access*, vol. 7, pp. 70371–70421, 2019.

[6] K. Samdanis and T. Taleb, "The road beyond 5g: A vision and insight of the key technologies," *IEEE Network*, vol. 34, no. 2, pp. 135–141, 2020.

[7] D. M. Sheen, D. L. McMakin, and T. E. Hall, "Detection of explosives by millimeter-wave imaging," in *Counterterrorist Detection Techniques of Explosives*, pp. 237–277, Elsevier, 2007.

[8] Y. Ren, W. Yang, X. Zhou, H. Chen, and B. Liu, "A survey on tcp over mmwave," *Computer Communications*, 2021.

[9] N. P. Lawrence, B. W.-H. Ng, H. J. Hansen, and D. Abbott, "5g terrestrial networks: Mobility and coverage—solution in three dimensions," *IEEE Access*, vol. 5, pp. 8064–8093, 2017.

[10] M. Polese, M. Mezzavilla, M. Zhang, J. Zhu, S. Rangan, S. Panwar, and M. Zorzi, "milliproxy: A tcp proxy architecture for 5g mmwave cellular systems," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pp. 951–957, IEEE, 2017.

[11] S. Mohebi, F. Michelinakis, A. Elmokashfi, O. Grøndalen, K. Mahmood, and A. Zanella, "Sectors, beams and environmental impact on commercial 5g mmwave cell coverage: an empirical study," *arXiv preprint arXiv:2104.06188*, 2021.

[12] Y. Niu, Y. Li, D. Jin, L. Su, and A. V. Vasilakos, "A survey of millimeter wave communications (mmwave) for 5g: opportunities and challenges," *Wireless networks*, vol. 21, no. 8, pp. 2657–2676, 2015.

[13] D. A. Hayes, D. Ros, Ö. Alay, and P. Teymoori, "Reliable consistent multipath mmwave communication," 2021.

[14] R. Zullo, A. Pescapé, K. Edeline, and B. Donnet, "Hic sunt proxies: Unveiling proxy phenomena in mobile networks," in *2019 Network Traffic Measurement and Analysis Conference (TMA)*, pp. 227–232, IEEE, 2019.

[15] M. Polese, R. Jana, and M. Zorzi, "Tcp in 5g mmwave networks: Link level retransmissions and mp-tcp," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 343–348, IEEE, 2017.

[16] R. Gowd, S. Thenappan, and M. GiriPrasad, "A traffic delay and bandwidth based multipath scheduling approach for optimal routing in underwater optical network," *Wireless Personal Communications*, vol. 116, no. 2, pp. 1009–1023, 2021.

[17] H. Wu, Ö. Alay, A. Brunstrom, S. Ferlin, and G. Caso, "Peekaboo: Learning-based multipath scheduling for dynamic heterogeneous environments," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2295–2310, 2020.

[18] P. Dong, J. Xie, W. Tang, N. Xiong, H. Zhong, and A. V. Vasilakos, "Performance evaluation of multipath tcp scheduling algorithms," *IEEE Access*, vol. 7, pp. 29818–29825, 2019.

[19] H. Zhang, W. Li, S. Gao, X. Wang, and B. Ye, "Reles: A neural adaptive multipath scheduler based on deep reinforcement learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1648–1656, IEEE, 2019.

[20] H. Wu, G. Caso, S. Ferlin, Ö. Alay, and A. Brunstrom, "Multipath scheduling for 5g networks: Evaluation and outlook," *IEEE Communications Magazine*, vol. 59, no. 4, pp. 44–50, 2021.

[21] U. N. Bhat, *An introduction to queueing theory: modeling and analysis in applications*, vol. 36. Springer, 2008.

[22] W. J. Stewart, *Probability, Markov chains, queues, and simulation*. Princeton university press, 2009.

[23] J. F. Shortle, J. M. Thompson, D. Gross, and C. M. Harris, *Fundamentals of queueing theory*, vol. 399. John Wiley & Sons, 2018.

[24] H. Fattah and C. Leung, "An overview of scheduling algorithms in wireless multimedia networks," *IEEE wireless communications*, vol. 9, no. 5, pp. 76–83, 2002.

[25] H. Safa, H. Artail, M. Karam, R. Soudah, and S. Khayat, "New scheduling architecture for ieee 802.16 wireless metropolitan area network," in *2007 IEEE/ACS International Conference on Computer Systems and Applications*, pp. 203–210, IEEE, 2007.

[26] D. Serpanos and T. Wolf, *Architecture of network systems*. Elsevier, 2011.

[27] T. Kasami, "Weight distributions of bose-chaudhuri-hocquenghem codes," *Coordinated Science Laboratory Report no. R-317*, 1966.

[28] 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*, pp. 431–439, IEEE, 2016.

[29] 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*, pp. 147–159, 2017.

[30] B. Y. Kimura, D. C. Lima, and A. A. Loureiro, "Alternative scheduling decisions for multipath tcp," *IEEE Communications Letters*, vol. 21, no. 11, pp. 2412–2415, 2017.

[31] C. Ling, W. Tang, P. Dong, W. Yang, X. Lou, and H. Zhou, "Blocking time-based mptcp scheduler for heterogeneous networks," in *International conference on cloud computing and security*, pp. 364–375, Springer, 2018.

[32] P. Hurtig, K.-J. Grinnemo, A. Brunstrom, S. Ferlin, Ö. Alay, and N. Kuhn, "Low-latency scheduling in mptcp," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 302–315, 2018.

[33] P. S. Kumar, N. Fatima, and P. Saxena, "Performance analysis of multipath transport layer schedulers under 5g/b5g hybrid networks," in *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pp. 658–666, IEEE, 2022.

[34] Y. Xing, K. Xue, Y. Zhang, J. Han, J. Li, J. Liu, and R. Li, "A low-latency mptcp scheduler for live video streaming in mobile networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7230–7242, 2021.

[35] F. Michel, Q. De Coninck, and O. Bonaventure, "Adding forward erasure correction to quic," *arXiv preprint arXiv:1809.04822*, 2018.

[36] F. Chang, K. Onohara, and T. Mizuochi, "Forward error correction for 100 g transport networks," *IEEE Communications Magazine*, vol. 48, no. 3, pp. S48–S55, 2010.

[37] R. Kadel, N. Islam, K. Ahmed, and S. J. Halder, "Opportunities and challenges for error correction scheme for wireless body area network—a survey," *Journal of Sensor and Actuator Networks*, vol. 8, no. 1, p. 1, 2018.

[38] L. L. Peterson and B. S. Davie, *Computer networks: a systems approach*. Elsevier, 2007.

[39] J. G. Proakis and M. Salehi, *Digital communications*, vol. 4. McGraw-hill New York, 2001.

[40] R. Puri and K. Ramchandran, "Multiple description source coding using forward error correction codes," in *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers (Cat. No. CH37020)*, vol. 1, pp. 342–346, IEEE, 1999.

[41] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE transactions on information theory*, vol. 47, no. 2, pp. 619–637, 2001.

[42] S. Ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE transactions on communications*, vol. 52, no. 4, pp. 670–678, 2004.

[43] A. Abbasfar, D. Divsalar, and K. Yao, "Accumulate-repeat-accumulate codes," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 692–702, 2007.

[44] H. Jin, A. Khandekar, R. McEliece, *et al.*, "Irregular repeat-accumulate

codes," in *Proc. 2nd Int. Symp. Turbo codes and related topics*, pp. 1–8, Citeseer, 2000.

[45] G. M. Dillard, M. Reuter, J. Zeiddler, and B. Zeidler, "Cyclic code shift keying: a low probability of intercept communication technique," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 3, pp. 786–798, 2003.

[46] R. Chien, "Cyclic decoding procedures for bose-chaudhuri-hocquenghem codes," *IEEE Transactions on information theory*, vol. 10, no. 4, pp. 357–363, 1964.

[47] S. B. Wicker and V. K. Bhargava, *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.

[48] R. J. McEliece and D. V. Sarwate, "On sharing secrets and reed-solomon codes," *Communications of the ACM*, vol. 24, no. 9, pp. 583–584, 1981.

[49] U. Demir and O. Aktas, "Raptor versus reed solomon forward error correction codes," in *2006 International Symposium on Computer Networks*, pp. 264–269, IEEE, 2006.

[50] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Transactions on Communications*, vol. 44, no. 5, pp. 591–600, 1996.

[51] B. Sklar, "A primer on turbo code concepts," *IEEE communications Magazine*, vol. 35, no. 12, pp. 94–102, 1997.

[52] J. Stern, "A new identification scheme based on syndrome decoding," in *Annual International Cryptology Conference*, pp. 13–21, Springer, 1993.

[53] I. Sason and S. Shamai, "Performance analysis of linear codes under maximum-likelihood decoding: A tutorial," 2006.

[54] H.-L. Lou, "Implementing the viterbi algorithm," *IEEE Signal processing magazine*, vol. 12, no. 5, pp. 42–52, 1995.

[55] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[56] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," *European Transactions on Telecommunications*, vol. 8, no. 2, pp. 119–125, 1997.

[57] V. Franz and J. B. Anderson, "Concatenated decoding with a reduced-search bcjr algorithm," *IEEE Journal on selected Areas in Communications*, vol. 16, no. 2, pp. 186–195, 1998.

[58] A. Nafaa, T. Taleb, and L. Murphy, "Forward error correction strategies for media streaming over wireless networks," *IEEE Communications Magazine*, vol. 46, no. 1, pp. 72–79, 2008.

[59] A. Basalamah and T. Sato, "A comparison of packet-level and byte-level reliable fec multicast protocols for wlans," in *IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*, pp. 4702–4707, IEEE, 2007.

[60] T.-W. Lee, S.-H. Chan, Q. Zhang, W.-W. Zhu, and Y.-Q. Zhang, "Optimal allocation of packet-level and byte-level fec in video multicasting over wired and wireless networks," in *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No. 01CH37270)*, vol. 3, pp. 1994–1998, IEEE, 2001.

[61] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *International Workshop on Peer-to-Peer Systems*, pp. 328–337, Springer, 2002.

[62] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," *IEEE network*, vol. 12, no. 5, pp. 40–48, 1998.

[63] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017.

[64] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing," *arXiv preprint arXiv:1209.5145*, 2012.

[65] J. Bezanson, S. Karpinski, V. Shah, A. Edelman, *et al.*, "Julia language documentation," *The Julia Manual. http://docs. julialang. org/en/release-0.2/manual*, pp. 1–261, 2014.

# Appendix

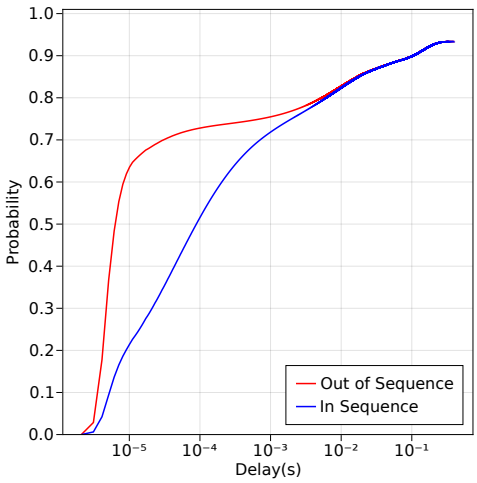



**Figure 1:** CDF plot of packet delay using RR and 3 constant paths.

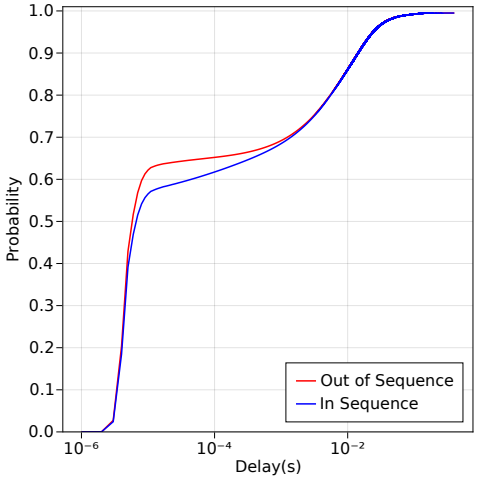



**Figure 2:** CDF plot of packet delay using RR and no prediction of paths.

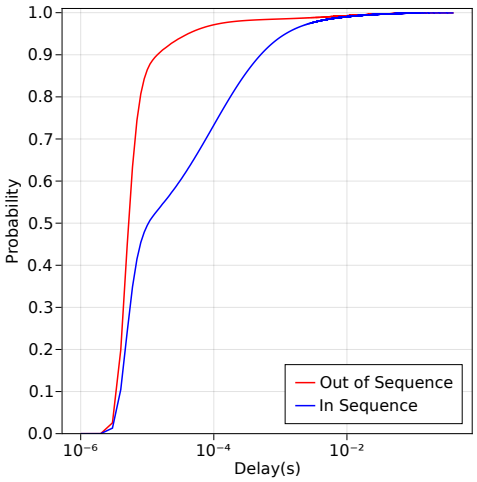



**Figure 3:** CDF plot of packet delay using RR and CDF prediction.

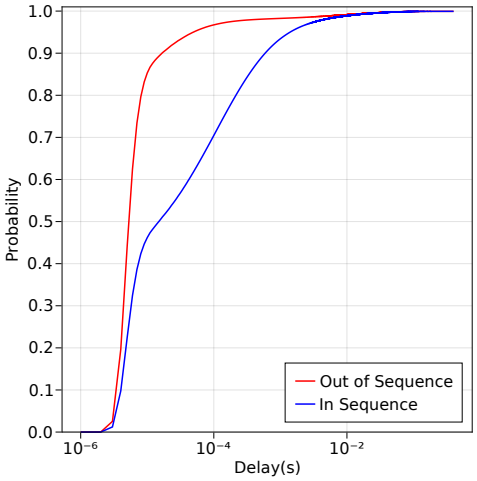



**Figure 4:** CDF plot of packet delay using RR and FPT prediction.

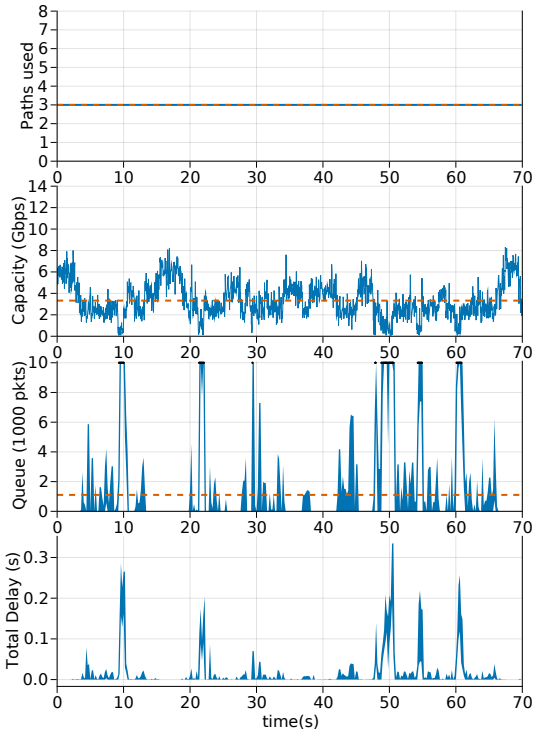



**Figure 5:** Simulation results using Random Scheduling and fixed number of paths.

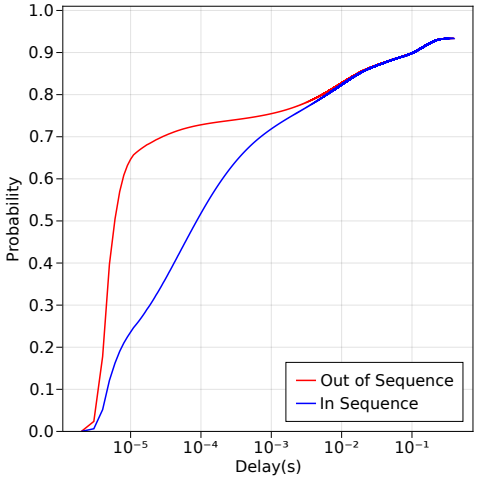



**Figure 6:** CDF plot of packet delay using Random Scheduling and 3 fixed paths.

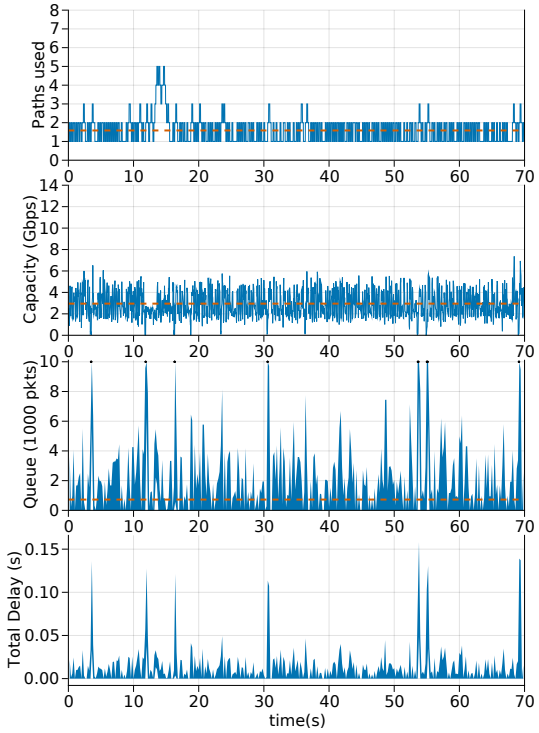



**Figure 7:** Simulation results using Random Scheduling and no prediction of paths.

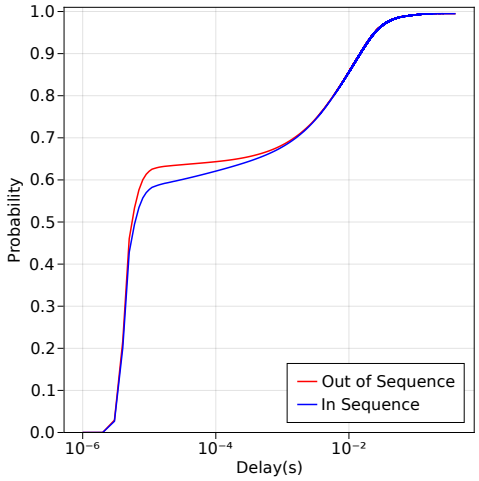



**Figure 8:** CDF plot of packet delay using Random Scheduling and no prediction of paths.

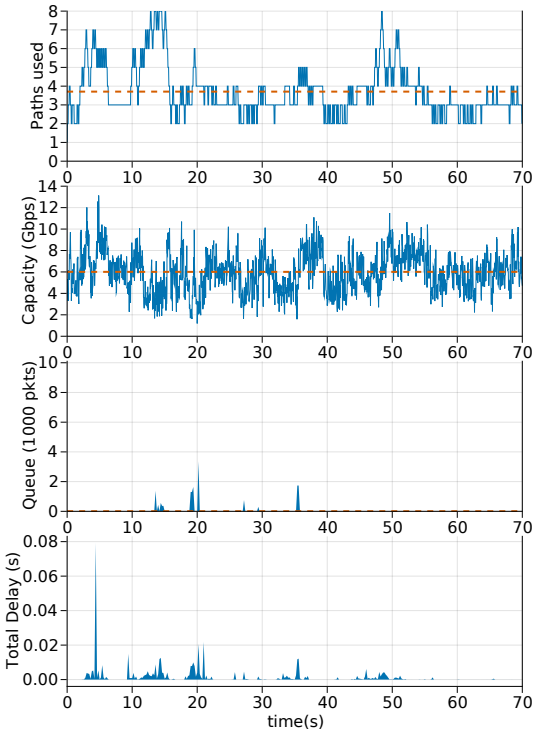



**Figure 9:** Simulation results using Random Scheduling and CDF prediction of paths.

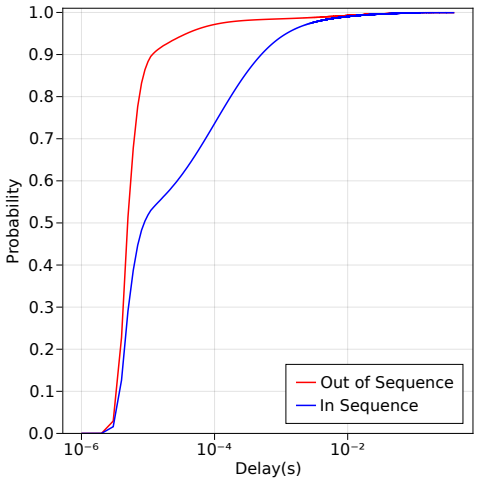



**Figure 10:** CDF plot of packet delay using Random Scheduling and CDF prediction.

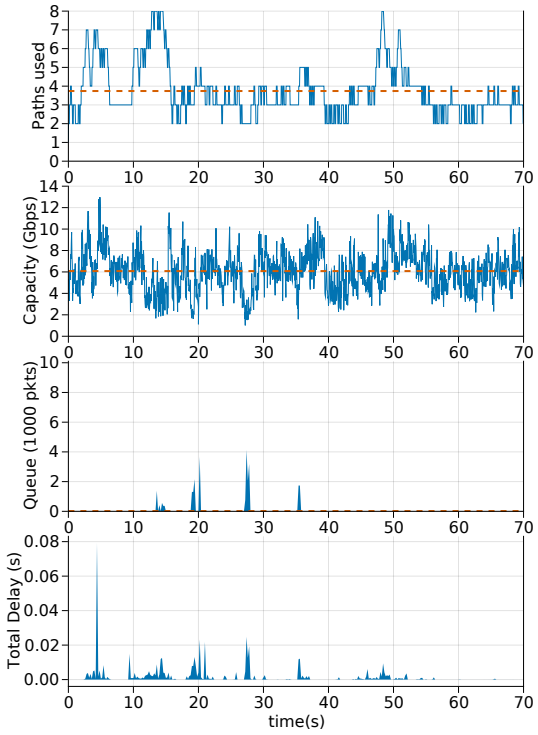



**Figure 11:** Simulation results using Random Scheduling and FPT prediction of paths.

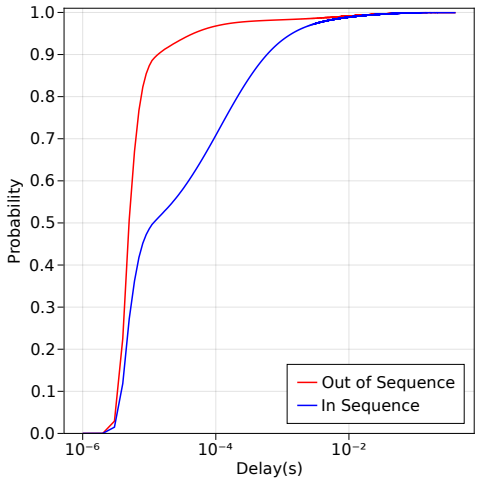



**Figure 12:** CDF plot of packet delay using Random Scheduling and FPT prediction.

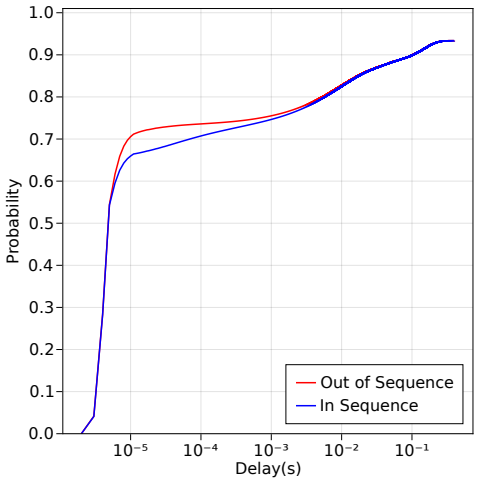



**Figure 13:** CDF plot of packet delay using Highest Capacity First Scheduling and 3 fixed paths.

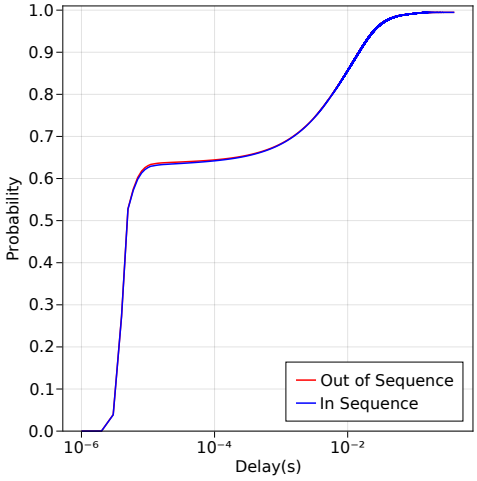



**Figure 14:** CDF plot of packet delay using Highest Capacity First Scheduling and no prediction of paths.

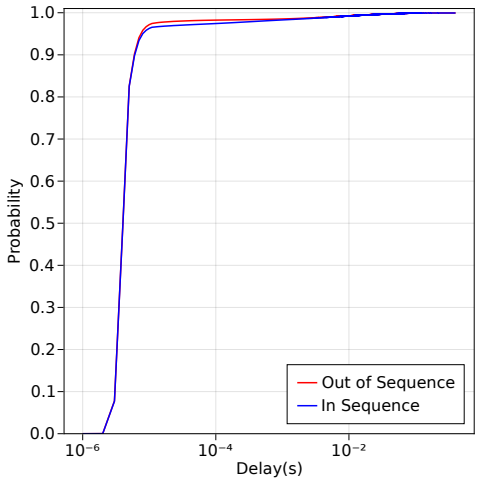



**Figure 15:** CDF plot of packet delay using Highest Capacity First Scheduling and CDF prediction.

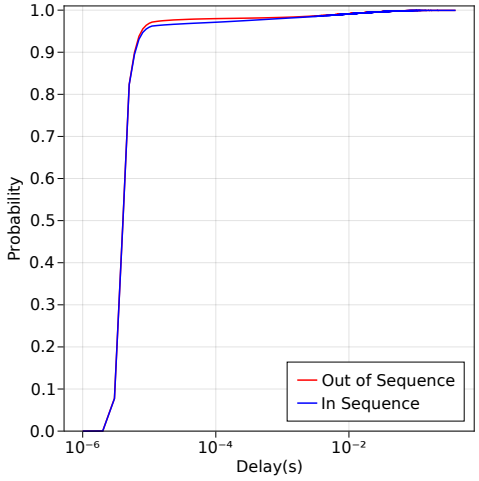



**Figure 16:** CDF plot of packet delay using Highest Capacity First Scheduling and FPT prediction.

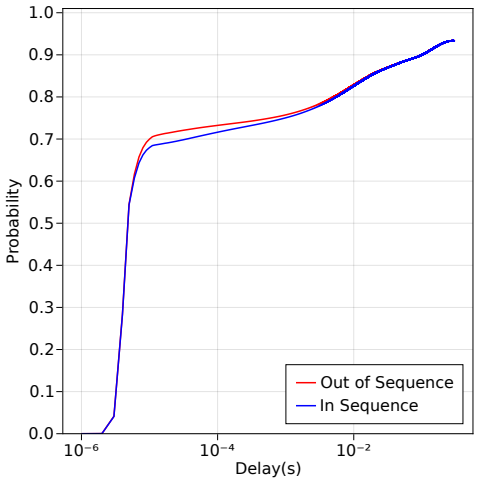



**Figure 17:** CDF plot of packet delay using Proposed Hybrid Scheduling and 3 fixed paths.

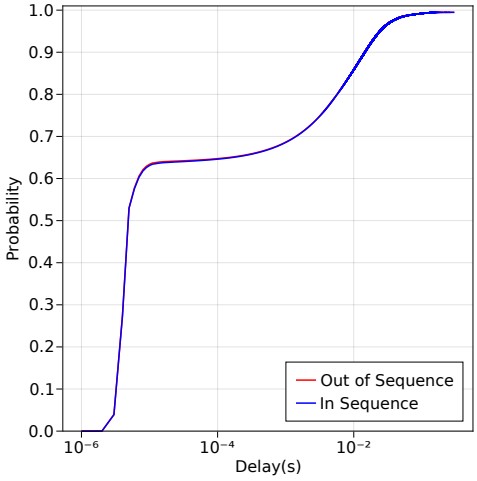



**Figure 18:** CDF plot of packet delay using Proposed Hybrid Scheduling Scheduling and no prediction of paths.

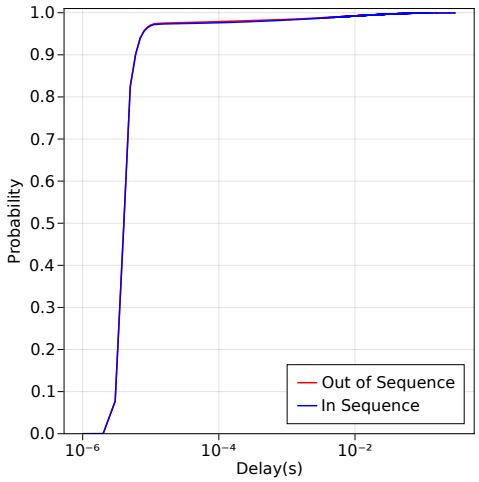



**Figure 19:** CDF plot of packet delay using Proposed Hybrid Scheduling and CDF prediction.

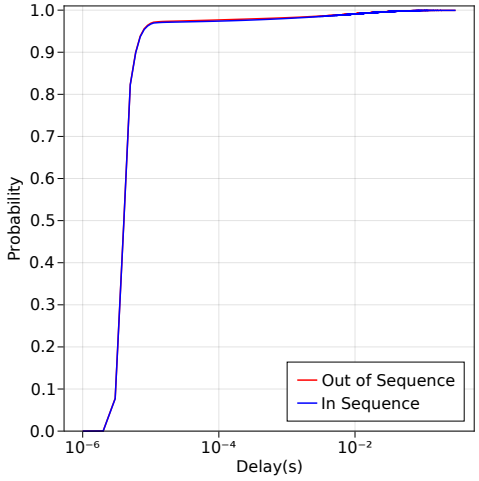



**Figure 20:** CDF plot of packet delay using Proposed Hybrid Scheduling and FPT prediction.

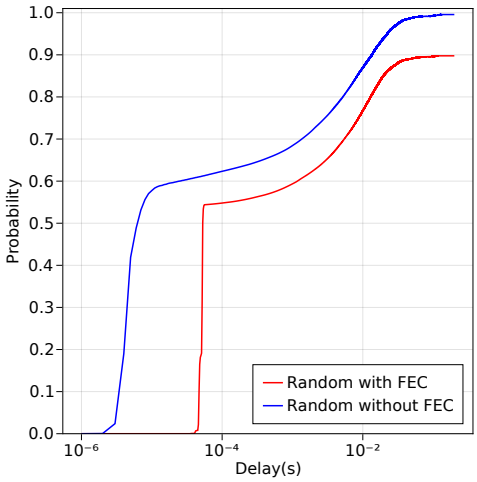



**Figure 21:** CDF plot of the Random Scheduler with and without FEC and with no path prediction.N=9, M=1

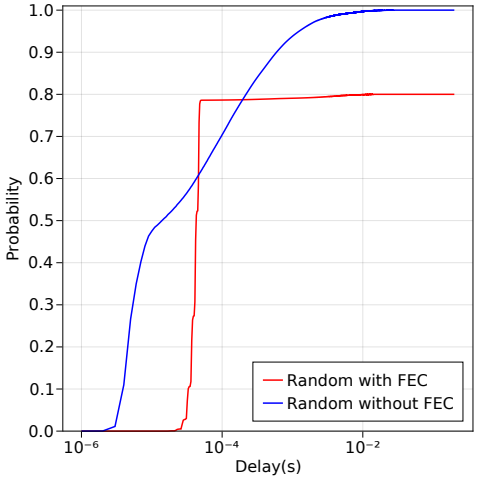



**Figure 22:** CDF plot of the Random Scheduler with and without FEC and with no path prediction.N=8, M=2

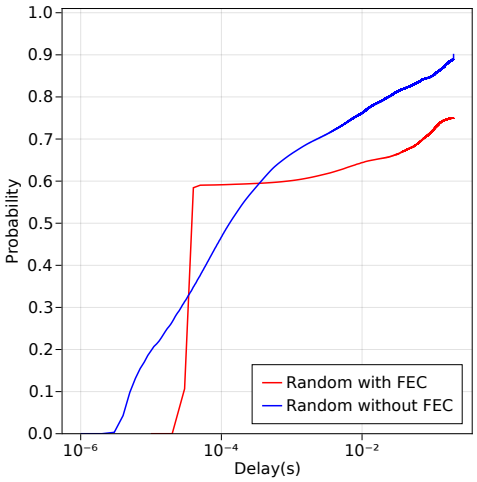



**Figure 23:** CDF plot of the Random Scheduler with and without FEC and with no path prediction.N=8, M=2

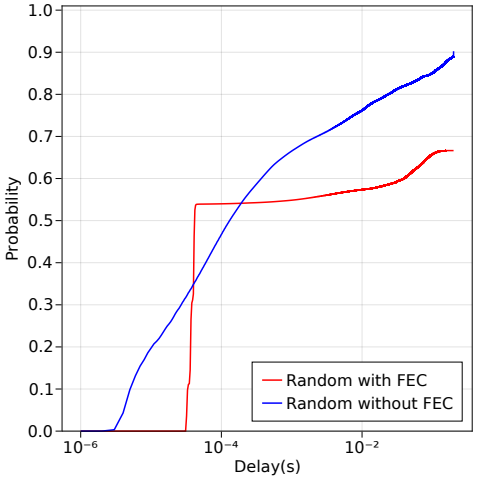



**Figure 24:** CDF plot of the Random Scheduler with and without FEC and with no path prediction.N=7, M=3