POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Learning how to QUIC-kly adapt to Wireless Network Conditions

Supervisors

Prof. Flavio ESPOSITO

Prof. Alessio SACCO

Prof. Guido MARCHETTO

Candidate

Cristiano SERRA

July 2024

Summary

QUIC (Quick UDP Internet Connections) is a transport layer protocol, proposed by Google and used both in Chrome and YouTube. Its aim is to improve overall performance by solving some problems of the standard transport protocol currently used by most websites, i.e., TCP. Amongst the other, those most notable are Head of Line blocking and long handshake times.

Researchers have put their efforts in improving many aspects of transport layer protocols, i.e., power and bandwidth constraints and congestion control in wireless networks. Specifically, research efforts have tried to improve the performances of congestion control algorithm, either with fixed heuristic or with new machine learning algorithms. However, none of these solutions is able to exploit and combine lower-level metrics with transport layer information, thus providing a more comprehensive view of the state of the network.

In this work, we enhance the congestion control mechanism of the QUIC protocol via usage of a Deep Reinforcement Learning model. We design a neural network-based model that aims at obtaining better inference performance by means of cross-layer information. Finally, the model was trained by exploiting a full-stack dataset, which was collected during this thesis.

The proposed approach has been validated in a NextG wireless network architecture. Experimental results shows interesting performance trade-offs between RTT and bandwidth in a variety of conditions, from video streaming to random web browsing, proving the effectiveness of the discussed approach.

The code and the dataset are available with an open-source license.

Acknowledgements

I would like to thanks all the people that helped me throughout this long journey.

To my parents, my brother, and all my relatives: your love and understanding sustained me through the challenges. Your constant support, encouragement, and belief in my abilities have been the foundation upon which I built this journey.

To my close friends Cristian, Denis, Federico, Gianmario, Giovanni, Luca, Mirko, Stefano C, Stefano D, and Yann, thank you for being there through both tough times and silly moments, always bringing lightness when I needed it most.

To all the people in the lab, thank you for all the help, both inside and outside the lab, you made my time in the USA as magical and unforgettable as it could be.

To my university colleagues and friends, Alessandro, Giacomo and Lorenzo, thank you for your support and help, and for the silly moments we shared. I would have definitely skipped portions of some exams if I hadn't studied with you.

And finally, a special thanks to my late grandfather, Mario, whenever you might be.

To everyone, I am deeply grateful for your sacrifices, patience, and the joy you brought into my life. Thank you for always being there for me.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Since the introduction of QUIC in 2012, its adoption has steadily grown, with it being used by approximately 7.9% of websites as of February 2024 [1] and accounting for around 75% of Meta's traffic [2].
QUIC, short for Quick UDP Internet Connections, is a modern transport protocol designed to improve the speed and security of internet communication.
Unlike traditional protocols like TCP, QUIC operates over UDP (User Datagram Protocol), which leads to faster connection setup and lower latency.
QUIC introduces multiplexed streams, allowing multiple application-layer streams to be sent over a single QUIC connection. This enables efficient communication between clients and servers, reducing latency and improving performance.
Additionally, QUIC supports 0-RTT (Zero Round Trip Time) connections. The initial handshake takes one RTT, this is due to the fact that initially, the client has no information about the server. Initially, when the client lacks information about the server, it initiates a handshake process by sending a preliminary client hello (CHLO) message. This CHLO prompts the server to respond with a reject (REJ) message. The REJ message comprises several components: (i) a server configuration that includes the server's long-term Diffie-Hellman public value, (ii) a certificate chain for authenticating the server, (iii) a signature of the server configuration using the private key from the leaf certificate of the chain, and (iv) a source-address token. This token, an authenticated-encryption block, contains the client's publicly visible IP address (as observed at the server) along with a timestamp provided by the server. Subsequently, the client utilizes this token in later handshakes to validate ownership of its IP address. Upon receiving the server configuration, the client proceeds to authenticate it by verifying the certificate chain and signature. Following successful authentication, the client transmits a complete CHLO message, which includes the client's ephemeral Diffie-Hellman public value.
This feature allows clients to send encrypted data to servers during the initial

handshake, further reducing latency and speeding up the connection process.
Once the client has sent a complete CHLO, it obtains initial keys for the connection. This is possible because it can calculate the shared value using the server's long-term Diffie-Hellman public value and its own ephemeral Diffie-Hellman private key. At this stage, the client can begin transmitting application data to the server. If the client aims to achieve zero round-trip time (0-RTT) latency for data transmission, it must start sending data encrypted with its initial keys without waiting for the server's response.
This allows the protocol to have particularly high gains in high RTT networks, in the original QUIC paper [3], a set of tests in different region has been conducted, showing that, while in low average RTT, the gain of QUIC over TCP is negligible (like in South Korea), in other region where the average RTT is higher, like India, the gains are much bigger as shown in Table 1.1.

			% Reduction in Search Latency		% Reduction in Rebuffer Rate	
Country	Mean Min RTT (ms)	Mean TCP Rtx %	Desktop	Mobile	Desktop	Mobile
South Korea	38	1	1.3	1.1	0.0	10.1
USA	50	2	3.4	2.0	4.1	12.9
India	188	8	13.2	5.5	22.1	20.2

Table 1.1: Network characteristics of selected countries and the changes to mean Search Latency and mean Video Rebuffer Rate for users in QUIC.

The most interesting result is that the % Reduction in Search Latency is less pronounced in mobile devices than in Desktop ones, while it's the other way around in the case of Rebuffer Rate.

1.1 Motivation

The original QUIC paper [3], emphasizes several metrics, with specific focus on handshake times, in particular it shows that the main advantage of the 0-RTT mechanism is it's ability to be largely insensitive to RTT, due to the fixed latency cost of 0-RTTs.
A big limitation of QUIC is its relatively modest gains for mobile users compared to desktop users. This is partially due to the fact that mobile applications are often fine-tuned for their environment and partially to the CPU constraint of mobile devices.
The vast majority of connections in the latter devices is via wireless, this means that some work has to be done to address the performances of QUIC in this kind of enviroment.
In order to tackle this underlying problem, different modified versions of QUIC has been proposed by researchers worldwide, i.e., QUIC-go [4], Microsoft QUIC

(MsQUIC) [5], Modified QUIC [6], Quiche [7] and aioquic [8]. QUIC-go is an implentation of QUIC in Go language. MsQUIC is a Microsoft implementation of the QUIC protocol. Modified QUIC proposes a modification to the handshaking mechanism to minimize the time required to update the CWnd. Quiche is an implementation of the QUIC protocol in Rust and C. aioquic is an implementation of quic in Python.
So far, Machine Learning based congestion control algorithm have been proposed but only for the TCP protocol, such as: Remy [9], PCC-Vivace [10], Q-Learning TCP [11], Orca [12] and Aurora [13]. While this works demonstrate the capability and potential of ML in this field, they were not designed and developed for the QUIC protocol or wireless networks in general.
To address this issue some other works has been proposed, for example PBQ-Enhanced QUIC [14] uses a deep RL algorithm based on PPO [15] to adjust the CWnd. However, this last work does not incorporate lower-level metrics in the algorithm.

1.2 Thesis Contribution

To address this issues, we propose a novel congestion control algorithm based on Reinforcement Learning, which allows to use lower level metrics in order to have a deeper understanding of the underlying network conditions.

1.3 Object

The significance of this work lies in its potential to address critical limitations of existing congestion control algorithms in wireless networks. As 5G technology becomes increasingly prevalent, the ability to optimize network performance is crucial. By integrating reinforcement learning and cross-layer information, this research offers a groundbreaking solution that could lead to more efficient, reliable, and adaptable network protocols. Another issue that has been tackled in this work, is the creation of a new full-stack dataset, containing features from different layers.

1.4 Findings

We anticipate that by carrying out this task, the suggested method, which combines cross-layer information with reinforcement learning, will greatly improve QUIC's performance in a 5G wireless setting. It is expected that this approach will offer a more thorough and precise knowledge of network circumstances, resulting in better overall network efficiency and more successful congestion control. In particular, we anticipate lower latency, higher throughput, and more effective management of network congestion when compared to conventional approaches that exclusively depend on transport layer features. By creating the dataset mentioned early, we also expect to provide researchers the ability to easily train and test new algorithms.

1.5 Meanings

The findings of this study can hold significant implications in the study of congestion control algorithms in wireless networks. The successful integration of reinforcement learning techniques, together with the cross-layer approach, highlights the potential of advanced machine learning techniques in addressing complex network challenge

1.6 Perspectives

Based on these findings, we recommend further exploration and adoption of cross-layer, machine learning based approaches in network protocol design. By continuing research into both, more general, and real-world settings, this could pave the way for more resilient, adaptive, and optimized network infrastructure

Chapter 2

Related Work

2.1 Previous Work in Congestion Control

Congestion control is a critical aspect of network management, ensuring that data is transmitted efficiently and reliably across a network. Over the years, several traditional methods have been developed to address congestion in various network environments. However, these methods often face significant challenges when applied to wireless networks.

2.1.1 Overview of Traditional Congestion Control Methods

The primary goal of congestion control is to prevent network congestion, which can lead to packet loss, increased latency, and reduced throughput. Traditional congestion control mechanisms have been primarily implemented at the transport layer, with TCP (Transmission Control Protocol) being one of the most widely used protocols.

TCP Congestion Control Mechanisms

- **TCP Cubic** [16]: TCP Cubic enhances congestion control with a scalable approach using a cubic function to adjust the congestion window dynamically. It balances throughput and fairness in varying network conditions, efficiently utilizing available bandwidth while maintaining stability.

- **TCP Reno** [17]: Building on early congestion control algorithms, TCP Reno introduced the fast recovery mechanism. It improves performance in high-latency environments by quickly recovering from packet losses without waiting for a timeout.

- **TCP NewReno** [18]: An enhancement over TCP Reno, NewReno includes improvements in the fast recovery algorithm, allowing it to recover multiple packet losses within a single window of data.
- **TCP Vegas** [19]: This algorithm focuses on detecting congestion before packet loss occurs by monitoring round-trip times. TCP Vegas adjusts the congestion window more conservatively compared to Tahoe and Reno, aiming to maintain a stable network state.
- **BBR (Bottleneck Bandwidth and Round-trip Propagation Time)** [**20**]: Unlike traditional loss-based congestion control algorithms, BBR is a model-based algorithm that estimates the available bandwidth and round-trip time to determine the optimal sending rate. Developed by Google, BBR aims to maximize throughput while minimizing queuing delay by operating at the estimated bottleneck bandwidth and keeping the network queue as short as possible.

While these traditional congestion control algorithms have been instrumental in managing network traffic, they encounter specific limitations in wireless environments. Traditional algorithms rely on hard-coded rules and predefined thresholds, which means they are not adaptive. These fixed rules can lead to suboptimal performance when network conditions change rapidly, as is often the case in wireless networks.

2.2 Previous Work on Congestion Control Using Machine Learning and Reinforcement Learning

The application of machine learning (ML) and reinforcement learning (RL) to congestion control represents a significant advancement over traditional methods. Unlike conventional algorithms that rely on hard-coded rules and predefined thresholds, ML and RL approaches can learn and adapt to dynamic network conditions, making them particularly suitable for complex and variable environments such as wireless networks.

- **Aurora** [**13**]: Aurora employs a neural network to predict the optimal sending rate based on observed network conditions such as latency and packet loss. It has shown significant improvements in throughput and latency compared to traditional congestion control algorithms like TCP Cubic and BBR.
- **Remy** [**9**]: Remy employs supervised learning to generate rules that optimize performance based on historical network data and predefined objectives.

Remy-generated algorithms have demonstrated better performance in terms of throughput and fairness compared to standard TCP variants.

- **PCC [21]**: PCC optimizes network performance by dynamically adjusting congestion control parameters based on observed network conditions. It aims to maximize throughput and minimize latency through adaptive learning techniques, ensuring efficient utilization of available bandwidth.
- **Orca [12]**: Orca Congestion Control is a scheme designed to enhance the performance of network traffic by adapting to varying network conditions. It leverages a combination of delay-based and loss-based signals to make dynamic adjustments in real-time, ensuring efficient utilization of network resources. By optimizing the balance between throughput and latency, Orca aims to improve the overall quality of experience for end-users, particularly in high-speed and complex network environments.
- **Sage [22]**: Sage Congestion Control is a congestion control algorithm developed to improve network performance by combining predictive modeling with real-time data analysis. It uses machine learning techniques to forecast network congestion and adjust data transmission rates accordingly, aiming to prevent packet loss and minimize latency. By proactively managing congestion, Sage enhances throughput and stability, making it suitable for modern, high-demand network applications.
- **Owl [23]**: Owl is a novel congestion control protocol leveraging reinforcement learning to improve network performance. Owl dynamically adjusts the congestion window based on both end-to-end features and partial network signals, outperforming traditional end-to-end or in-network methods. The protocol is evaluated extensively, demonstrating significant improvements in bandwidth and delay across various network scenarios, and proving effective even with partial network visibility.

These examples highlight the potential of machine learning and reinforcement learning techniques in developing more adaptive and efficient congestion control algorithms. By leveraging data-driven approaches, these algorithms can better handle the dynamic and unpredictable nature of network environments, especially in wireless networks.

2.3 Previous work in congestion control for wireless networks

Congestion control in wireless networks plays a critical role in optimizing data transmission efficiency and network performance. This section reviews significant advancements and methodologies in congestion control tailored specifically for wireless communication environments.

- **Verus** [24]: Verus is a congestion control algorithm specifically designed for wireless networks, notable for its innovative approach to optimizing throughput and managing network resources. Verus employs machine learning techniques to predict available bandwidth in wireless environments. By forecasting network conditions, Verus can dynamically adjust transmission rates to maximize throughput while maintaining stability.

- **Sprout** [25]: Sprout is a congestion control algorithm designed primarily for real-time multimedia applications, especially those operating over wireless networks. Developed by researchers at MIT, Sprout aims to provide low-latency, high-quality video streaming and real-time communication experiences.

Current research in congestion control for wireless networks highlights the evolution towards adaptive methodologies, leveraging machine learning to enhance efficiency and effectively manage dynamic network environments, particularly in real-time multimedia applications.

Chapter 3

Background on QUIC and RL

This section will provide essential background both on QUIC and on Reinforcement Learning. This will be crucial to understand the problem we are trying to solve and the possible solutions.

3.1 Introduction to QUIC

QUIC (Quick UDP Internet Connections) is a transport layer network protocol initially developed by Google in 2012. It was designed to address some of the limitations of the Transmission Control Protocol (TCP), particularly in terms of connection establishment latency and reliability. QUIC has since been standardized by the Internet Engineering Task Force (IETF) as RFC 9000, marking its evolution from an experimental protocol into a widely adopted standard.

3.1.1 Purpose

The primary purpose of QUIC is to enhance the performance of web applications by reducing latency and improving security. QUIC achieves this by combining the features of the traditional transport and security layers into a single protocol, thus optimizing the overall communication process. It leverages User Datagram Protocol (UDP) as its underlying protocol, enabling faster connection setup and data transfer compared to TCP.

3.1.2 Multiplexed Connections

QUIC supports multiple streams within a single connection, preventing head-of-line blocking, a common issue in TCP where a delay in one stream can block all others. This feature enhances the performance of applications that rely on multiple parallel data streams, such as modern web pages that load numerous resources concurrently.

3.1.3 Connection Establishment and Latency

QUIC significantly reduces connection establishment latency through its 0-RTT (Zero Round-Trip Time) and 1-RTT (One Round-Trip Time) handshakes. Unlike TCP, which requires multiple round trips to establish a secure connection, QUIC can establish a secure connection much more quickly, improving page load times and user experience.

3.1.4 Security

QUIC incorporates TLS 1.3 encryption directly into its protocol, ensuring that all connections are secure from the start. This integration reduces the need for separate security handshakes and simplifies the security model.

3.1.5 Congestion Control

As described on RFC 9002, the congestion control algorithms used in QUIC, are the classical TCP ones. However, QUIC supports various congestion control algorithms, such as Cubic and New Reno, and it provides flow control on a per-stream basis, unlike TCP's connection level flow control.

3.1.6 Adoption and Applications

Major web browsers like Google Chrome and Mozilla Firefox, and services like Google Search and YouTube, have adopted QUIC, demonstrating its effectiveness in real-world applications. Additionally, the adoption of HTTP/3, which runs over QUIC, further underscores its significance in the next generation of web protocols.

3.2 Introduction to Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning where an agent learns to make decisions by performing actions in an environment to maximize some notion of cumulative reward. This section provides a comprehensive introduction to RL, covering its definitions, core components, fundamental concepts, and various types of RL algorithms.

3.2.1 Definitions and Basics

Reinforcement learning (RL) is an interdisciplinary area of machine learning and optimal control concerned with how an intelligent agent ought to take actions in a dynamic environment in order to maximize the cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.

Reinforcement learning differs from supervised learning in not needing labelled input/output pairs to be presented, and in not needing sub-optimal actions to be explicitly corrected. Instead the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge) with the goal of maximizing the long term reward, whose feedback might be incomplete or delayed.

3.2.2 Core Components

The core components of a Reinforcement Learning system are:

- **Agent:** The entity that takes actions and learns from the environment.
- **Environment:** The world in which the agent operates and interacts.
- **State:** A description of the world's current situation.
- **Action:** Choices available to the agent at any given state, they affect the state of the environment.
- **Reward Function:** A function that measures the feedback of the last action of the agent in the environment.
- **Policy:** A mapping from state to action that defines the agent's behavior.
- **Value Function:** Value functions specify what is good in the long run. The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state.

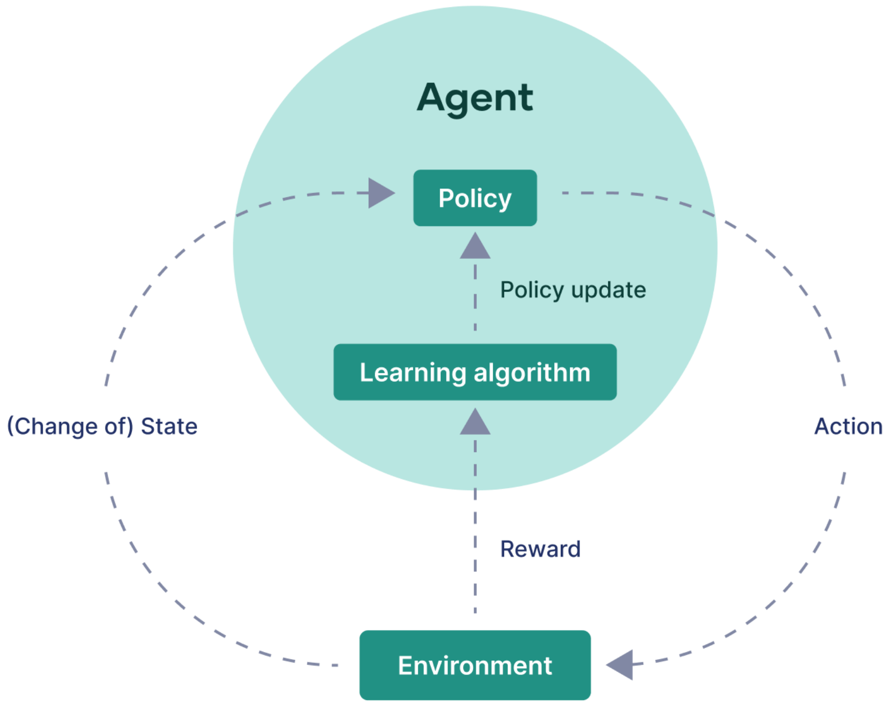

Figure 3.1: General Framework of Reinforcement Learning

3.2.3 Markov Decision Processes (MDP)

The Markov Decision Process (MDP) is a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of the agent. An MDP is defined by a 4-tuple (S, A, P_a, R_a):

- **S:** The set of all possible states called the state space.
- **A:** The set of actions called the action space.
- **$P_a(s, s') = P_r(s_{t+1} = s'|s_t = s, a_t = a)$:** The probability that action a in state s at time t will lead to state s' at time $t + 1$.
- **$R_a(s, s')$:** The immediate reward receive after transitioning from state s to state s', due to action a.

A policy function π is a (potentially probabilistic) mapping from state space (S) to action space (A).
The goal in and MDP is to find a policy that maximizes the expected cumulative reward over time.

3.2.4 Exploration vs Exploitation

One of the key challenges in RL is the balance between exploration and exploitation:

- **Exploration:** The agent tries new actions to discover their effects and improve its knowledge of the environment.
- **Exploitation:** The agent uses its current knowledge to maximize the reward, choosing actions that it believes to be the best based on past experience.

Effective RL strategies must find a balance between these two approaches to ensure that the agent can discover optimal policies while still obtaining high rewards.

3.2.5 Model-Free vs Model-Based

RL algorithms can be broadly categorized into two types based on whether they use a model of the environment:

- **Model-Free:** These algorithms do not use a model of the environment. Instead, they learn policies or value functions directly from experience.
- **Model-Based:** These algorithms use a model of the environment to make decisions. The model predicts the next state and reward given a current state and action.

3.2.6 Deep Reinforcement Learning

Deep Reinforcement Learning combines RL with deep learning, allowing agents to handle high-dimensional state spaces and learn complex policies. Some well known algorithms are:

- **Deep Q Network (DQN):** Deep Q-Networks (DQN) combine Q-learning with deep neural networks to approximate the action-value function, enabling agents to learn policies directly from high-dimensional sensory inputs like images. DQN uses experience replay and a target network to stabilize training, which has achieved impressive results in complex environments such as Atari games.
- **Proximal Policy Optimization (PPO):** Proximal Policy Optimization (PPO) is a policy gradient method that improves training stability and efficiency by using a clipped objective function to limit the policy updates within a trust region. PPO balances the benefits of policy gradient methods with robustness, making it effective for various continuous control tasks and commonly used in complex environments like robotics and video games.

Chapter 4

Dataset Collection with ETTUS B210

In the context of 5G networks, three main components are needed:

- Core Network
- Next Generation NodeB (gNodeB/RAN)
- User Equipment (UE)

4.1 Core Network

The more complex element of the network is the Core Network, this consists of 11 functions. Each one of them represent a distinct and specialized task or operation that contribute to the overall functionality and performance of the network. The components of the network are namely:

- User Plane Function (UPF)
- Data network (DN)
- Core Access and Mobility Management Function (AMF)
- Authentication Server Function (AUSF)
- Session Management Function (SMF)
- Network Slice Selection Function (NSSF)
- Network Exposure Function (NEF)

- NF Repository Function (NRF)
- Policy Control function (PCF)
- Unified Data Management (UDM)
- Application Function (AF)

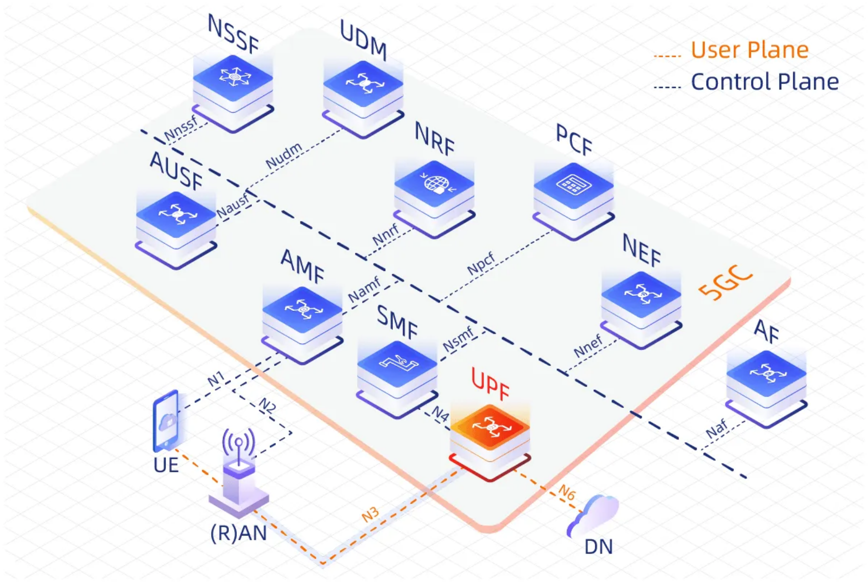

Figure 4.1: Core Network Schematic

4.1.1 User Plane Function

The User Plane Function (UPF) is a crucial component in the architecture of 5G networks. It is responsible for handling the user data plane, which involves the actual transmission and routing of data packets between the user equipment (UE) and external data networks. This function allows the data plane to be shifted closer to the network edge, resulting in faster data rates and shorter latencies.

4.1.2 Data Network (DN)

In the context of 5G, the data network is often referred to as the internet. This network is a complex infrastructure that enables the transmission, processing, and management of data across 5G systems.

4.1.3 Core Access and Mobility Management Function (AMF)

The AMF is the most critical component of the 5g Core Network. It is responsible for managing access and mobility for 5G devices, and it interacts with other network functions such as the UPF (User Plane Function), SMF (Session Management Function), and AUSF (Authentication Server Function). The AMF performs the following functions:

- **Device Registration**
  The AMF is responsible for registering a 5G device with the network and assigning it a unique identifier. This allows the network to keep track of the device and its location.

- **Access Management**
  The AMF performs access management functions such as authentication, authorization, and accounting (AAA) for 5G devices. It verifies the identity of the device and determines whether it is authorized to access the network.

- **Mobility Management**
  The AMF tracks the location of the device and manages handovers between cells and base stations. It ensures that the device maintains connectivity as it moves through different areas of the network.

- **Policy Enforcement**
  The AMF enforces network policies such as Quality of Service (QoS) and charging policies. It ensures that the network resources are allocated appropriately and that the device is charged correctly for the services it uses.

- **Session Management**
  The AMF manages the establishment, modification, and termination of 5G sessions for devices. It coordinates with other network functions such as the Session Management Function (SMF) to ensure that sessions are set up correctly and resources are allocated appropriately.

- **User Plane Function Selection**
  The AMF selects the appropriate UPF based on network policies and the device's location. The UPF is responsible for forwarding user data between the device and the network.

- **Subscriber Data Management**
  The AMF stores and manages subscriber data such as the device's profile, subscription data, and service data. This allows the network to provide personalized services to the device.

- **Security Management**
  The AMF is responsible for ensuring the security of 5G devices and the network. It manages security functions such as key management, authentication, and encryption.

- **Network Slicing**
  The AMF plays a critical role in network slicing, which allows the network to create virtualized network segments with dedicated resources and services for different use cases. The AMF is responsible for managing the access and mobility of devices within each network slice.

- **Network Integration**
  The AMF is responsible for integrating the 5G core network with external networks, such as 4G LTE networks or Wi-Fi networks. It coordinates with other network functions to ensure a seamless handover between different networks.

- **Control Plane Management**
  The AMF manages the control plane of the 5G network, which is responsible for signalling and network management. It ensures that signalling messages are transmitted correctly between network functions and that the network resources are managed efficiently.

- **Fault Management**
  The AMF is responsible for detecting and managing faults within the 5G core network. It monitors the network for anomalies and alerts network operators if a fault is detected.

- **Policy Control**
  The AMF is responsible for enforcing policies related to network resource allocation, Quality of Service (QoS), and charging. It ensures that the policies are applied correctly and that the device is charged appropriately for the services it uses.

- **Location Management**
  The AMF is responsible for tracking the location of 5G devices and managing their mobility. It ensures that devices maintain connectivity as they move through different areas of the network.

- **Network Optimization**
  The AMF plays a key role in optimizing the 5G network for performance and efficiency. It monitors network usage and adapts the network resources to meet the demands of the devices.

4.1.4 Authentication Server Function (AUSF)

The Authentication Server Function (AUSF) in the 5G network is a core component responsible for verifying the identity of users and devices trying to access the network. It enforces security measures to ensure only authorized entities can connect, collaborates with other network functions, supports flexible authentication methods, and manages authentication vectors for secure communication. Overall, it plays a crucial role in maintaining network integrity and security

4.1.5 Session Management Function (SMF)

The Session Management Function (SMF) in the 5G network is central to managing user sessions, ensuring optimal IP address allocation, controlling Quality of Service (QoS), enforcing network policies, coordinating with other functions, dynamically adapting to user and network conditions, managing data bearers, and implementing robust security measures.

4.1.6 Network Slice Selection Function (NSSF)

The Network Slice Selection Function, as the name suggests, manages different network slices for diverse user needs, ensuring efficient resource use, coordinating with other network elements, facilitating customization, handling slice creation and termination, ensuring smooth user experience, enabling flexible network deployment, and optimizing services through policy control.

4.1.7 Network Exposure Function (NEF)

The Network Exposure Function (NEF) in 5G enables third-party applications to access network capabilities and services through APIs, integrating external services securely, managing authentication and authorization, fostering service innovation, supporting monetization, and enhancing agility in service provisioning without pinpointing specific details.

4.1.8 NF Repository Function (NRF)

The Network Repository Function (NRF) in 5G plays a fundamental role in managing information related to network functions (NFs) within the network architecture. Firstly, it acts as a central registry where information about available NFs in the network is stored. This registry serves as a kind of "database" of NFs, allowing other network elements to discover and access these functions when needed. One of the primary functions of the NRF is to facilitate NF discovery by other network elements. This means it provides mechanisms and procedures through which other network elements can obtain detailed information about which NFs are available, what functionalities they offer, and how they can be used. Essentially, it acts as a "catalog" of NFs within the network. In addition to NF discovery, the NRF also supports dynamic updates of NF information. This means that when network conditions change, such as when new NFs are added or when capabilities of existing NFs are modified, the NRF is capable of updating and keeping the information in its registry up to date. Another important function of the NRF is load balancing management. Since it can provide detailed information about available NF instances and their capabilities, it can be used to balance the workload within the network. For example, it can route traffic to less burdened NF instances to ensure efficient resource utilization. Furthermore, the NRF is involved in managing the security of NF information. This includes handling permissions and credentials required to access NFs, as well as protecting sensitive data related to NFs themselves.

4.1.9 Policy Control function (PCF)

The Policy Control Function (PCF) is responsible for enforcing network policies related to user access, data usage, Quality of Service (QoS), and traffic management. It dynamically manages policies based on real-time network conditions and user requirements, ensures appropriate QoS levels, prioritizes traffic, allocates resources efficiently, enables service differentiation, integrates with charging systems, and ensures policy consistency and interoperability across network elements and interfaces.

4.1.10 Unified Data Management (UDM)

The Unified Data Management (UDM) function in 5G networks plays a central role in managing subscriber data, authentication, authorization, service subscriptions, policies, identity management, service orchestration, interoperability, and data analytics. It ensures secure access to the network, manages service entitlements, enforces network policies, supports service orchestration, and facilitates data analytics for network optimization and business intelligence.

4.1.11 Application Function (AF)

The Application Function (AF) in 5G networks manages network applications and services, controls service delivery based on policies and user preferences, oversees session management, enforces network policies, exposes services to external applications, allocates network resources, ensures interoperability, and provides analytics for network optimization.

4.2 Next Generation NodeB (gNodeB/RAN)

The gNodeB, or Next Generation NodeB, represents a significant evolution in radio access technology within 5G networks, introducing several innovative features and capabilities that are distinct from the core network elements. Here are some key aspects highlighting the role and functions of the gNodeB:

- **Massive MIMO Technology**
  One of the hallmark features of the gNodeB is its support for massive multiple-input multiple-output (MIMO) technology. This advancement allows the gNodeB to utilize a large number of antennas for transmitting and receiving signals simultaneously, significantly enhancing spectral efficiency, data rates, and overall network capacity. Massive MIMO enables the gNodeB to serve multiple users concurrently and improve the overall performance of wireless communication.

- **Beamforming Techniques**
  The gNodeB implements advanced beamforming techniques to focus radio signals in specific directions towards UEs, improving signal strength, coverage, and reliability. Beamforming optimizes the use of radio resources and enhances the quality of wireless connections, especially in environments with high user density or challenging propagation conditions.

- **Dynamic Spectrum Sharing (DSS)**
  Another key feature of the gNodeB is its ability to support dynamic spectrum sharing. This technology allows the gNodeB to dynamically allocate spectrum resources based on demand and network conditions, optimizing spectrum utilization and improving overall network efficiency. DSS enables flexible allocation of frequency bands between 4G LTE and 5G NR (New Radio) users, ensuring seamless coexistence and transition between the two technologies.

- **Low Latency Communication**
  The gNodeB is designed to facilitate low-latency communication, critical for applications requiring real-time responsiveness such as autonomous vehicles, industrial automation, and virtual reality. By reducing signal processing delays and optimizing transmission protocols, the gNodeB enables ultra-reliable, low-latency communication (URLLC) for time-sensitive applications, enhancing user experience and enabling new use cases.

- **Network Slicing Support**
  As a key component of network slicing, the gNodeB plays a crucial role in supporting the creation and management of dedicated virtual network slices tailored to specific use cases, industries, or applications. This capability allows

operators to offer customized services with differentiated performance characteristics, QoS levels, and security parameters, optimizing resource allocation and meeting diverse customer requirements.

- **Edge Computing Integration**
  The gNodeB can integrate with edge computing resources at the network edge, enabling distributed processing, data caching, and content delivery closer to end users. This integration enhances application performance, reduces latency, and supports bandwidth-intensive applications by leveraging edge computing capabilities at the gNodeB site.
- **Self-Optimization and Self-Healing**
  The gNodeB incorporates self-optimization and self-healing mechanisms to autonomously monitor, analyze, and optimize its performance. It can dynamically adjust radio parameters, mitigate interference, and adapt to changing network conditions, ensuring optimal operation and resilience against potential failures or disruptions.

4.3 User Equipment (UE)

A User Equipment (UE) in the context of telecommunications refers to the devices used by end-users to access and utilize mobile network services. UEs include smartphones, tablets, laptops with cellular connectivity, IoT devices, and any other device capable of connecting to a mobile network. While there are some overarching similarities between UEs across different generations of mobile networks, such as the ability to make calls, send messages, and access the internet, there are notable differences in UEs designed for 5G compared to previous generations like 4G (LTE), 3G, and 2G. Here's a breakdown of what defines a UE in 5G and how it differs from earlier generations:

- **Data Rates and Throughput**
  5G UEs are designed to deliver significantly higher data rates and throughput compared to their predecessors. This allows for faster download and upload speeds, smoother streaming of high-definition content, and improved performance for bandwidth-intensive applications like video conferencing, online gaming, and content streaming.
- **Latency**
  One of the key improvements in 5G UEs is reduced latency, referring to the time it takes for data to travel between the device and the network. Lower latency in 5G UEs enables real-time communication and applications with minimal delays, making them suitable for applications like augmented reality (AR), virtual reality (VR), autonomous vehicles, and industrial automation.

- **Capacity and Connectivity**
  5G UEs support higher device densities and massive connectivity, allowing for a greater number of devices to be connected simultaneously. This is particularly important for IoT deployments, where a large number of sensors, devices, and machines need to communicate with the network efficiently.

- **Spectrum Utilization**
  5G UEs can operate across a wider range of frequency bands, including higher frequency bands such as mmWave (millimeter wave), mid-band, and sub-6GHz frequencies. This expanded spectrum enables 5G UEs to access more bandwidth, leading to improved network performance and capacity.

- **Advanced Radio Technologies**
  5G UEs leverage advanced radio technologies like massive MIMO (Multiple Input Multiple Output), beamforming, and carrier aggregation. These technologies enhance signal coverage, improve spectral efficiency, and optimize network resource utilization, resulting in better overall user experience and network performance.

- **Network Slicing and Customization**
  5G UEs benefit from network slicing capabilities, allowing network operators to create customized virtual network slices tailored to specific use cases or industries. This enables differentiated services with varying Quality of Service (QoS) levels, security parameters, and performance characteristics, catering to diverse user requirements.

In summary, 5G UEs represent a significant advancement in mobile device technology, offering higher data rates, lower latency, increased capacity, advanced radio technologies, spectrum utilization, and customization options compared to earlier generations. These enhancements pave the way for transformative applications, services, and use cases across various sectors, driving the evolution of connected technologies in the 5G era.

4.4 Data Collection

To collect the data, we used OpenAirInterface (OAI) to simulate both the core network and the gNodeB and UE. OAI is an open-source software platform that allowed us to accurately replicate a real-world network environment. This setup enabled us to design, test, and validate the interactions between the core network, gNodeB, and UE, ensuring we gathered reliable and detailed data for our study. The setup consisted of two separate PCs: one functioning as the UE simulator (a laptop) and the other as the gNodeB simulator (a server), which also simulated the core network. Each PC was equipped with an ETTUS B210, enabling communication between them using 5G capabilities. This configuration allowed us to effectively simulate and analyze the interactions between the UE, gNodeB, and the core network. The data collection part consisted in reading values from the L1 layer (physical layer) and the second layer (MAC layer).

4.4.1 The Hardware

We will now provide a brief overview of the equipment specifications used in this project. This information will be helpful in our later discussion of the limitations encountered during the project's development.

Server Specifications

- **CPU**: Intel(R) Xeon(R) Gold 6312U CPU @ 2.40GHz
- **Cores**: 24
- **Threads**: 48
- **Memory**: 64GB ECC
- **GPU**: Not present

Laptop Specifications

- **CPU**: 12th Gen Intel(R) Core(TM) i7-1260P
- **Cores**: 12
- **Threads**: 16
- **Memory**: 16GB non-ECC
- **GPU**: Integrated, not compatible with cuda

Antennas Specifications

Concerning the radio frontend, an ETTUS USRP B210 has been used. This device provides a fully integrated, single-board, Universal Software Radio Peripheral platform. The B210 supports a RF frequency range from 70 MHz to 6 GHz, has a Spartan6 FPGA, and USB 3.0 connectivity. It uses an Analog Devices RFIC to deliver a cost-effective RF experimentation platform, and can stream up to 56 MHz of instantaneous bandwidth over a high- bandwidth USB 3.0 bus on select USB 3.0 chipsets (with backward compatibly to USB 2.0).

4.4.2 Implementation And Limitations

To collect the data, the Open Air Interface's driver code was modified, in order to allow it to log the data to two different files, one for the Layer 1 metrics and one for the MAC Layer metrics. While convenient, modifying the Open Air Interface driver code introduces overhead, potentially impacting performance. However, since the overhead is minimal, it was an acceptable trade-off. The main issue with this approach was the granularity. Every time the driver was asked to log the different values, it also had to compute them simultaneously. While this typically shouldn't be a problem, inefficiencies in the OAI code led to significant overhead. To mitigate this, logging was limited to once per second, as more frequent logging caused the entire system to crash. This issue could be resolved by either developing more efficient drivers or using a more powerful machine.

The data collection was done using both a Python implementation called aioquic [8] and a Rust based implementation called QUICHE [7], developed by Cloudflare.

The main convenience of the first one is the ability to easily implement and switch between different congestion algorithms, this was useful to collect data from CUBIC [16] and New Reno [18], later used for comparison purposes.

Another key element of the data collection was the type of requests done. While having more complex requests could have lead to a more diverse dataset, given the time and resource constraints, a simple approach, using random requests ranging from 500 Kilo Bytes to 500 Mega Bytes has been chosen. This ensures a good amount of variability of requests, while providing a simple way for the reproduction of the results.

In computers, random numbers are not truly random but are generated by algorithms called pseudorandom number generators (PRNGs). These numbers mimic randomness but are actually produced in a predictable sequence, starting from an initial value known as a seed. While this can be seen as a detrimental feature, it can also be leveraged to create the same sequence of requests every time, ensuring repeatability.

4.4.3 The Dataset

The datasets we created, one with CUBIC and another with New Reno, encompass a comprehensive range of metrics from Layer 1 (Physical Layer), the MAC Layer, and Layer 4 (Transport Layer). To the best of our knowledge, this is the only existing dataset that integrates all these metrics, providing a unique and valuable resource.

These datasets offer a wealth of information for further studies in the field. Researchers can leverage this data to explore various aspects of network performance, congestion control, and protocol efficiency across different layers. Additionally, the combined metrics can facilitate cross-layer analysis, enabling a more holistic understanding of network behavior.

By providing a foundation for subsequent research, these datasets can significantly contribute to advancements in networking and communication technologies. Future studies can build on this work to develop more efficient protocols, optimize network performance, and enhance the overall understanding of how different layers interact within complex network environments.

Chapter 5

Transport Protocol and Algorithmic Design

5.1 RL model

As stated before, in reinforcement learning (RL), a state is defined by a set of different values that describe the environment at a specific point in time. These values encapsulate all the necessary information that an agent needs to make informed decisions about its actions. To provide extensive context, as already said, we provided metrics coming from the physical layer and from the transport layer. The former are able to provide useful information about the real capabilities and limitations of the physical carrier, while the latter is able to provide feedback about real time performances.

5.1.1 State set

In this section we will provide more information about the metrics used in the state, whith a brief description of the meaning of them, and the layer they are coming from.

Name	**Description**	**Layer**
Blacklisted PRBs	Blacklisted Physical Resource Blocks	L1
Total PRBs	Total Physical Resource Blocks	L1
Max IO	Metric relative to the signal strength	L1
Min IO	Metric relative to the signal strength	L1
Avg IO	Metric relative to the signal strength	L1
PRACH IO	Quality of PRACH (Random Access Channel from UE to GNB)	L1
Current QM DL	Current Modulation Order (Number of Bits per Symbol) for DL	L1
Current RI DL	Rank Indicator for DL	L1
Total Bytes TX	Total number of Bytes Transmitted by the GNB since the start of the connection	L1
ULSCH Power	UpLink Shared Channel Power Level	L1
ULSCH Noise Power	UpLink Shared Channel Noise Power Level	L1
Sync Pos	Synchronization Position of UpLink Transmission	L1
Round Trials	Number of round trials for UpLink Transmission	L1
DTX	Discontinuous Transmission, number of times DTX has occured	L1
Current QM UL	Current Modulation Order (Number of Bits per Symbol) for DL	L1
Current RI UL	Number of Independent Data Streams for UL	L1
Total Bytes RX	Total number of Bytes Received by the GNB since the start of the connection	L1
Total Bytes Scheduled	Total number of Bytes scheduled to be sent from the GNB	L1
PH	Power Headroom	MAC
PCMAX	Maximum allowed transmit power constraint	MAC
RSRP	Reference Signal Received Power	MAC
meas	Number of measurement for RSRP	MAC
UL RI	Rank Indicator for UpLink	MAC
TPMI	Transmit Precoding Matrix Indicator	MAC
Dlsch Rounds	Number of rounds of transmission attempts for the DownLink shared channel (DLSCH)	MAC
Dlsch Errors	Number of errors during DownLink transmission	MAC
PUCCH0 DTX	Number of DTX in PUCCH number 0	MAC
BLER DL	Block Error Rate of the DownLink transmission, measures the quality of the transmitted data	MAC
MCS DL	Modulation and Coding Schemes for DownLink	MAC
Dlsch Total Bytes	Total number of bytes transmitted through the DownLink Shared Channel	MAC
Ulsch Rounds	Number of rounds of transmission attempts for the UpLink Shared Channel (ULSCH)	MAC
Ulsch DTX	Number of DTX for the UpLink Shared Channel	MAC
Ulsch Errors	Number of errors during UpLink transmission	MAC
BLER UL	Block Error Rate of the UpLink transmission	MAC
MCS UL	Modulation and Coding Schemes for UpLink	MAC
Ulsch Total Bytes Scheduled	Number of Total Bytes SCheduled for the UpLink Shared Channel	MAC
Ulsch Total Bytes Received	Number Of Total Bytes Received on the UpLink Shared Channel	MAC
Bandwidth Average	The Bandwidth in KBytes averaged over a second	L4
Congestion Window Average	The Congestion Window Value averaged over a second	L4
Smoothed RTT	The Smoothed Round Trip Time	L4
RTT Mean Deviation	Mean Deviation of the Round Trip Time	L4

Table 5.1: State Set

The metrics listed above are used as the input of the Reinforcement Learning algorithm, for representation's sake, some of them, namely Dlsch and Ulsch Rounds are listed as a single entity, while in reality they contain four different values, one for each retransmission round. Specifically, Dlsch and Ulsch Rounds metrics represent the number of attempts made for data transmission and reception at different stages of HARQ (Hybrid Automatic Repeat reQuest) retransmissions. These stages are denoted by 0, 1, 2, and 3, each corresponding to:

- **0**: The first attempt where the data packet is transmitted and ideally received successfully without any need for retransmission.
- **1**: The second attempt, indicating that the initial transmission failed, and the packet had to be retransmitted once.

- **2**: The third attempt, showing that the packet required two retransmissions after the first two attempts failed.
- **3**: The fourth attempt, where the data packet needed three retransmissions before being successfully received.

5.1.2 Action Space

In this section we will provide further information about the chosen action space. The action space in a Reinforcement Learning (RL) algorithm refers to the set of all possible actions that an agent can take in a given environment. It is a crucial component because the agent's goal is to learn a policy that dictates the best action to take in each state to maximize cumulative rewards.

Advantages of a Larger Action Space

- **Increased Flexibility**: With more actions available, the agent has more options to choose from, potentially leading to more optimal solutions and better performance.
- **Finer Control**: A larger action space allows for more granular control over the environment. This can be beneficial in complex scenarios where subtle adjustments are necessary for achieving the best outcomes.
- **Exploration**: More actions provide the agent with a broader range to explore, which can lead to discovering new strategies that might not be evident with a limited set of actions.

Disavantages of a Larger Action Space

- **Increased Complexity**: A larger action space makes the learning process more complex. The agent needs to evaluate and choose from more options, which can slow down the learning process.
- **Longer Training Time**: More actions mean more potential policies to explore, which can significantly increase the time required for the agent to learn an optimal policy.
- **Exploration vs. Exploitation Dilemma**: With a larger action space, the agent might spend more time exploring suboptimal actions before converging on the optimal policy, making it harder to balance exploration and exploitation.
- **Higher Computational Cost**: Evaluating a larger set of actions requires more computational resources, both in terms of memory and processing power, which can be a limitation for resource-constrained applications.

Given these reasons, various solutions have been considered, ranging from a very small, discrete action space to a much more complex continuous action space. Although a continuous action space could offer theoretically infinite granularity for the congestion window value, it would significantly increase training time. After some tuning, a good trade-off between the two was to use a discrete action space composed of 11 different actions, namely:

Action Index	**Action Definition**
0	$f(x) = \frac{x}{3}$
1	$f(x) = \frac{x}{2}$
2	$f(x) = x - 10$
3	$f(x) = x - 7$
4	$f(x) = x - 3$
5	$f(x) = x$
6	$f(x) = x + 3$
7	$f(x) = x + 7$
8	$f(x) = x + 10$
9	$f(x) = x \times 2$
10	$f(x) = x \times 3$

Table 5.2: Actions Space

5.2 Intersection of QUIC and RL

The integration of QUIC (Quick UDP Internet Connections) with Reinforcement Learning (RL) opens up novel opportunities for improving network performance and adaptability. This section explores the challenges in QUIC, the advantages of RL, and how RL can be applied to dynamic congestion control.

5.2.1 Challenges in QUIC

QUIC faces several challenges despite its numerous advantages over traditional transport protocols. Efficiently managing network congestion is critical for maintaining optimal performance, requiring adaptive congestion control mechanisms. Handling packet loss and ensuring reliable delivery can be challenging, especially in high-latency or unstable network environments. QUIC must balance resource allocation among multiple streams within a single connection, ensuring fair distribution and preventing any one stream from monopolizing bandwidth. Additionally, maintaining robust security features without compromising performance is crucial for QUIC's adoption. As QUIC becomes more widely adopted, it must scale efficiently to handle increased traffic and a growing number of connections without degrading performance.

5.2.2 Advantages of RL

Reinforcement Learning (RL) is particularly well-suited for network environments due to its ability to adapt and optimize in dynamic and complex settings. In network environments, conditions such as bandwidth availability, latency, and congestion can change rapidly. RL algorithms can learn and respond to these changes in real-time, optimizing network performance dynamically based on continuous feedback. This adaptability makes RL ideal for managing tasks like congestion control, where quick adjustments are necessary to maintain optimal data flow and minimize packet loss and delays. RL excels at handling multi-dimensional optimization problems, making it effective for balancing various performance metrics like throughput, latency, and fairness across different network streams.

5.2.3 Dynamic Congestion Control

Dynamic congestion control is a critical area where RL can significantly enhance QUIC's performance. Traditional congestion control algorithms are rule-based and may not perform optimally under all network conditions. RL can offer a more flexible and adaptive approach. RL algorithms can adjust congestion control parameters in real-time based on current network conditions, such as latency, packet loss, and throughput. By leveraging past data, RL can predict future network conditions and proactively adjust congestion control strategies to avoid congestion before it occurs. RL can balance multiple objectives, such as minimizing latency, maximizing throughput, and ensuring fairness among streams, to achieve an optimal trade-off. This approach can make congestion control more robust to sudden changes in network conditions, such as spikes in traffic or variations in bandwidth availability.

Chapter 6

Protocol and System Implementation

6.1 Comparison between different models

Since the introduction of Deep Reinforcement Learning in 2013, a huge variety of different algorithms has been proposed through the years. The first algorithm was introduced by Mnih et al. [26], the algorithm in question is called Deep Q-Network and, even if simpler than newer ones, it offers some advantages, some of them crucial to the goal of this project.

6.1.1 Advantages of Deep Q-Network

- **Simplicity**: DQN is relatively straightforward to implement compared to some newer algorithms. It uses a single network architecture to approximate the Q-function, making it easier to understand.
- **Robustness**: DQN has been shown to perform robustly across a variety of tasks and environments, including Atari games and simple control problems. Its simplicity often leads to more stable training and reliable convergence.
- **Experience Replay**: DQN utilizes experience replay, where experiences (state, action, reward, next state) are stored in a replay buffer and sampled randomly during training. This technique helps in breaking correlations between consecutive experiences, improving sample efficiency and enhancing learning stability.
- **Discrete Action Spaces**: DQN is particularly effective for tasks with discrete action spaces, where it can efficiently learn the Q-values for each action without needing modifications to handle continuous actions.

- **Single Network Architecture**: DQN uses a single neural network to approximate the Q-function, reducing the computational load compared to algorithms that require multiple networks (e.g., actor-critic methods which use separate networks for the actor and the critic).

6.1.2 Disadvantages of Deep Q-Network

- **Scalability to Continuous Action Spaces**: DQN is designed for discrete action spaces. It struggles with problems that require continuous actions, making it less suitable for tasks where actions cannot be discretized effectively.
- **Hyperparameter Sensitivity**: DQN requires careful tuning of various hyperparameters (e.g., learning rate, discount factor, replay buffer size). Finding the optimal settings can be challenging and time-consuming.

6.1.3 Advantages and disadvantages of newer algorithms

In this section we will provide some context of newer, more refined algorithms, while taking a look at the main advantages and disadvantages of them over Deep Q-Network.

In recent years, advancements in deep reinforcement learning have led to the development of several sophisticated algorithms that address some of the limitations of Deep Q-Networks (DQN). Notable among these are Deep Deterministic Policy Gradient (DDPG), Asynchronous Advantage Actor-Critic (A3C), Proximal Policy Optimization (PPO), and Soft Actor-Critic (SAC).

DDPG extends the capabilities of DQN to continuous action spaces by using an actor-critic approach, where the actor network outputs deterministic actions and the critic network evaluates these actions. This method allows for effective learning in environments where actions are not discrete.

A3C introduces asynchronous updates, where multiple agents interact with their own environments in parallel, sharing their experiences to stabilize learning. This approach reduces training time and improves the exploration of the state space.

PPO simplifies the policy optimization process by using a clipped objective function to ensure that updates are within a safe range. This method strikes a balance between performance and stability, making it widely adopted for various complex tasks.

SAC incorporates entropy maximization into the learning process, promoting more exploration by encouraging the policy to act more randomly. This results in robust policies that can handle a wide range of environments, particularly those with high uncertainty.

Each of these newer algorithms brings unique strengths to the table, improving upon DQN in aspects like handling continuous actions, improving stability and sample efficiency, and reducing training time. However, they also come with their own set of challenges and complexities, which we will explore in the following sections.

6.1.4 Advantages of newer algorithms

- **Sample Efficiency**: Newer algorithms often enhance sample efficiency, meaning they can achieve comparable or better performance using fewer interactions with the environment. This efficiency is crucial for applications where data collection is costly or time-consuming.

- **Stability**: Many newer algorithms address issues of instability during training, which can arise due to the non-stationarity of the data distribution or the large variance in gradient estimates. Improved stability leads to faster convergence and more reliable learning.

- **Scalability**: As environments and tasks become more complex, scalability becomes essential. Newer algorithms often incorporate techniques that allow them to handle larger state and action spaces, as well as more agents in multi-agent settings.

- **Generalization**: Some newer algorithms focus on improving generalization, meaning they can transfer knowledge learned from one task to another or adapt to changes in the environment more effectively. This adaptability is crucial for real-world applications where conditions may vary.

6.1.5 Disadvantages of newer algorithms

- **Complexity**: Many newer algorithms are more complex both in terms of implementation and theoretical understanding. This complexity can make them harder to implement, debug, and fine-tune for specific applications.

- **Computational Intensity**: Some algorithms require extensive computational resources, such as powerful GPUs or even distributed computing setups, to achieve optimal performance. This can increase costs and limit their practical deployment in resource-constrained environments.

- **Hyperparameter Sensitivity**: Many deep reinforcement learning algorithms involve tuning various hyperparameters (learning rates, exploration rates, network architectures, etc.) to achieve good performance. Finding the optimal set of hyperparameters can be challenging and time-consuming.

6.2 The model

Given the computational limitations, mainly the lack of a GPU, and the implementation limitations of Stable Baselines 3, the Deep Reinforcement Learning library used during the project, a Deep Q-Network has been used.
While this model is simple, as explained in the section above, it offers some crucial advantages over other models in our particular setting.
Particularly, the robustness and the computational efficiency of DQN make it highly suitable for our needs.
Another advantage, offered by the library used, is the ability to easily change the network that is used by the algorithm to approximate the mapping between the state set and the action space. One of the main proposal during this project was the usage of transformers, this was backed by some recent surveys [27] [28] that highlighted the capability and potential of using such architecture inside a Deep Reinforcement Learning environment. In the following we will discuss some of the key advantages of the latter.

6.2.1 Advantages of Transformers in DRL

- **Handling Sequential Data**: Transformers are highly effective at processing sequential data, which is essential in reinforcement learning where states and actions form a sequence over time. They can model long-range dependencies and capture temporal relationships better than traditional RNNs or LSTMs.

- **Attention Mechanism**: The attention mechanism is crucial in transformers for sequence modeling of states. It enables the RL agent to focus selectively on relevant cues in the environment and ignore redundant features, which leads to faster training. This is particularly useful in high-dimensional state spaces where there are a large number of input elements.

6.2.2 Disadvantages of Transformers in DRL

- **Computational Complexity**: Transformers are computationally intensive, particularly in terms of memory and processing power. The self-attention mechanism has a quadratic complexity with respect to the sequence length, making it resource-heavy for long sequences.

- **Large Memory Requirements**: The large number of parameters in transformers requires substantial memory, which can be a bottleneck, especially for longer sequences or higher-dimensional inputs. This can limit their applicability on devices with limited resources.

6.2.3 The final choice

While all the previously discussed ideas may appear sound, significant computational constraints emerged as a challenge. Given the inefficiency and complexity associated with transformers, coupled with the absence of a GPU-equipped server, the final model adopted was a Deep Q-Network (DQN). Section 5.1.2 details the exploration of different action spaces. Various Deep Reinforcement Learning models has been investigated, along with experimentation on different network architectures using the previously collected dataset.

Due to these computational limitations, the final model was simplified to a basic DQN utilizing the standard network provided by Stable Baselines 3. This model consists of a straightforward Multi-Layer Perceptron (MLP) with two layers, each containing 64 neurons.

6.2.4 The Reward Function

In order to train the model, a reward function is needed. This is used by the model to try to maximize the cumulative reward over time.
While a lot of different reward functions has been proposed in previous works, given the aim of the project and the simplicity of the requests, we opted for a standard function called Power. This is defined as:

$$R = \frac{\text{Bandwidth}}{\text{RTT}} \tag{6.1}$$

Some of the key points that led to the choice of the latter are:

- **Throughput Maximization**: By maximizing bandwidth, the algorithm seeks to utilize the network capacity effectively, ensuring that as much data as possible is transmitted in a given time frame.

- **Latency Minimization**: By minimizing RTT, the algorithm aims to reduce delays in the network, leading to quicker data exchanges and a more responsive network experience.

- **Balance Between Throughput and Latency**: The reward function inherently balances the need for high throughput with the need for low latency. A high reward is achieved when the network can transfer data quickly (high bandwidth) with minimal delay (low RTT).

This choice has shown to provide really good efficiency, while leading to good training results.

6.3 The Setup and Implementation

To train the model, we utilized a similar approach as detailed in Section 4.4.2. The network conditions during training closely mirrored those from the data collection phase, acknowledging the inherent variability and unpredictability of these conditions. Consequently, we conducted multiple runs and selected the results from the longest run.

Due to factors beyond our control, the antennas occasionally disconnected from each other. Therefore, among the numerous runs conducted, we selected the longest one to ensure consistent and reliable results.

In the following, we will detail the testbed setup to provide further insight into the network conditions during the training phase.

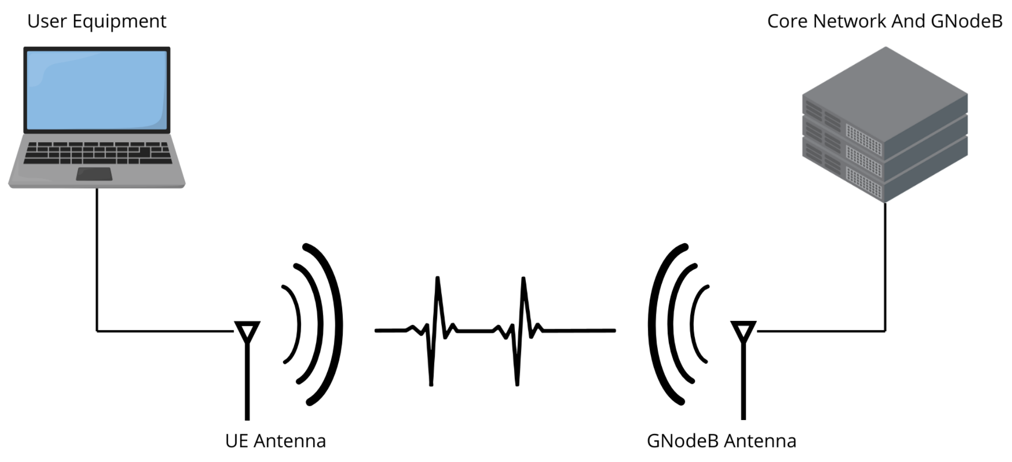

Figure 6.1: The hardware setup

Using the hardware listed in Section 4.4.1, we aimed to replicate the same network conditions as in the previous phase. This approach was taken to ensure fair comparisons of different algorithms during the subsequent evaluation stage.

The primary challenge during the implementation phase was effectively implementing the training process. One of the main constraints of reinforcement learning is the sequential nature of the data, necessitating real-time feedback from the antennas. While reading the data is similar to the data collection phase, the crucial difference is that this feedback must be fed to the model in real time to facilitate learning.

Stable Baselines 3 provides two main functions for training: `learn()` and `train()`. The `learn()` function interacts directly with the environment to learn, while the `train()` function can learn from previously saved experiences, typically stored in a buffer.

Given this framework, we first saved experiences in a buffer provided by Stable Baselines 3, then used the `train()` function to train the model. The main challenge was tuning the hyper-parameters, specifically the buffer size and the number of gradient steps. However, after some experimentation, we identified values that met our needs and overcame the computational constraints.

Chapter 7

Evaluation

7.1 Evaluation Metrics

In this section, we will outline the methodology used to evaluate the algorithm, discuss the rationale behind selecting specific metrics over others, and then present and analyze the results.

7.1.1 Challenges in Congestion Control Algorithms Evaluation

In this section we will list some of the main challenges in the evaluation of congestion control algorithms, this will be crucial to understand the choices made in the evaluation phase.

- **Dynamic Network Conditions**: Network conditions such as bandwidth, latency, and packet loss can vary greatly over time and across different environments. This variability makes it difficult to create a standardized evaluation framework that accurately reflects real-world conditions.
- **Complexity of Metrics**: Congestion control performance depends on multiple metrics like throughput, latency, jitter, and packet loss. Balancing these metrics to achieve a fair evaluation can be challenging, as improvements in one metric may lead to degradation in others.
- **Reproducibility**: Ensuring that evaluation results are reproducible across different studies is difficult due to variations in test environments, traffic patterns, and network configurations.

While some other issues may be found in other settings, we believe that this are the main one faced during this phase.

7.2 Evaluation Results

In this section we will provide and show the results of the training phase, and the testing phase, along with comparisons with other, more traditional, congestion control algorithms, namely Cubic [16] and New Reno [18].

7.2.1 Training Phase

In order to show the training trend, some graphs will be provided, showing the Reward vs Bandwidth and RTT, and than some graphs showing each one the metrics above separately.
Given the definition of our reward function and the varying scales of the metrics, we present a normalized version. To achieve normalization, each vector of values was scaled to the range [0, 1] by dividing each element in the vector by the maximum value of that vector. This process was applied to all three vectors individually.

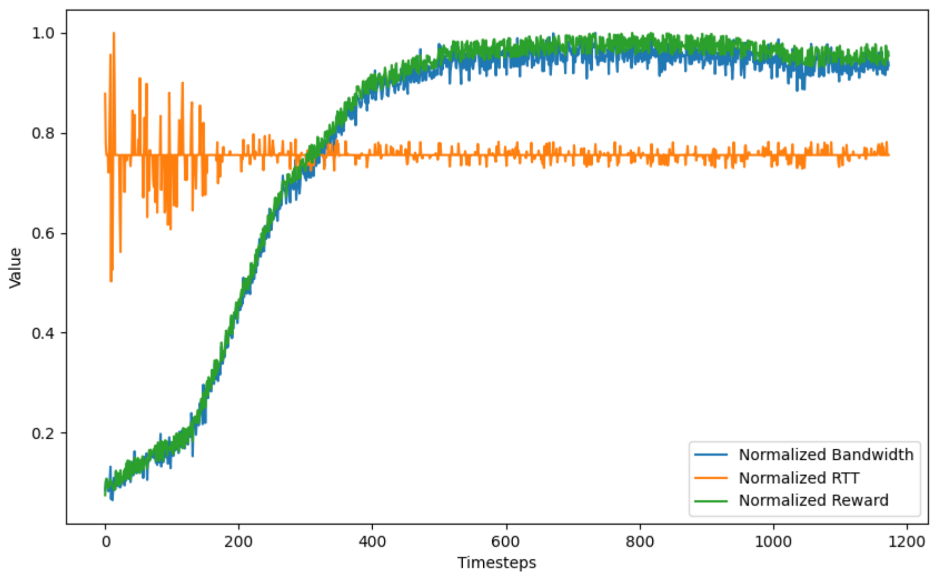

Figure 7.1: Training reward with Bandwidth and RTT

In order to show the true scale of the said metrics, we also present a non-normalized version of the latter, this will provide some context of the scale of things and will be useful later to compare them with other congestion control algorithms.

The next plot illustrates the trend of the reward function during training, showcasing the model's ability to learn how to maximize it. Each point represents a packet acknowledged. Although this inherently includes some time information, the progression is not linear. Initially, due to the low values of the congestion window, response times are generally shorter. As training progresses and the model learns to assign larger values to the congestion window, the granularity of the logging decreases, meaning that in the later stages, the time difference between two points is higher.

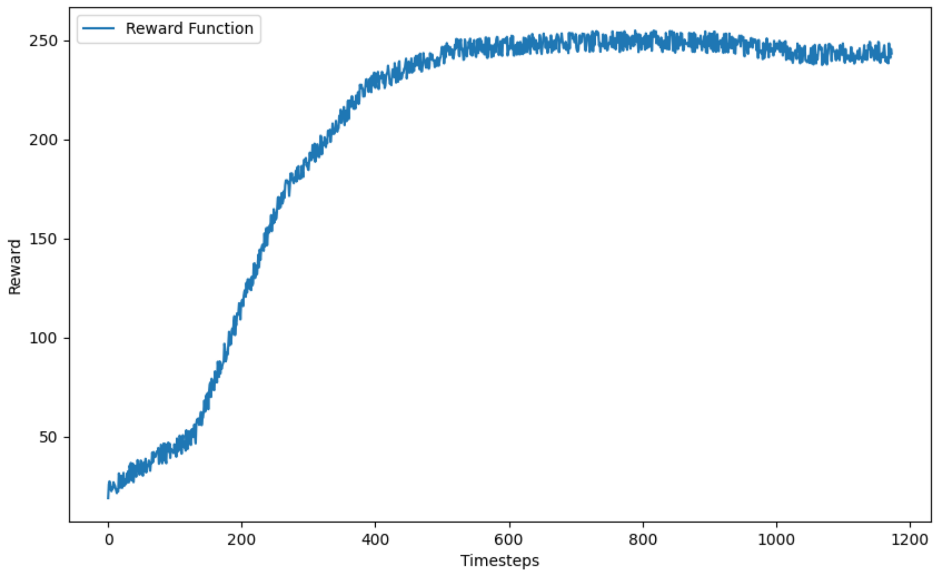

Figure 7.2: Reward Trend during training

The next plots will individually display the values of the bandwidth and the Round Trip Time (RTT). These metrics are essential for understanding the performance and behavior of the network during training.

By examining the bandwidth plot, we can observe how the data transmission rate evolves over time and how efficiently the network is utilized. This helps us understand the model's ability to optimize data flow and maintain high throughput, which is crucial for evaluating network performance.

The RTT plot provides insights into the latency and the responsiveness of the network. It reveals how quickly packets are sent and acknowledged, which is a critical factor in assessing the network's efficiency and the impact of different congestion window settings.

Displaying these metrics individually is important to show the scale of each. By isolating the bandwidth and RTT, we can see the absolute values and variations over time, making it easier to interpret their significance and relationship to the training process.

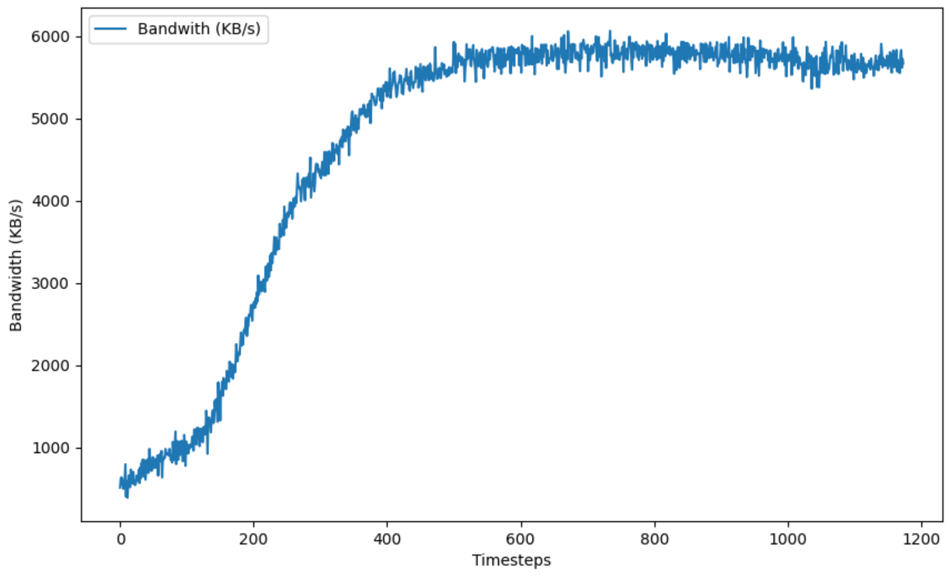

Figure 7.3: Bandwidth Trend during training

By analyzing these metrics separately, we gain a clearer understanding of the model's learning process and its effects on different aspects of network performance, enabling more informed decisions for future improvements.

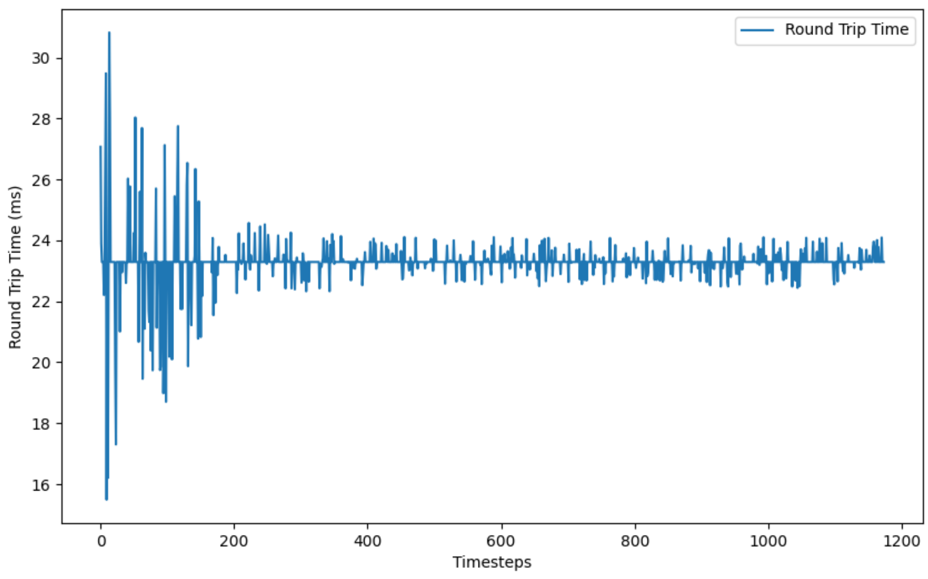

Figure 7.4: Round Trip Time trend during training

7.2.2 Comparison with other algorithms

Given the reasons explained in section 7.1, in order to show the performance differences between different algorithms, Bandwidth and Round Trip Time where chosen as the to go metrics. To compare the algorithms, and to assure fairness between them, several key considerations should be taken into account.

Building upon the discussion of data collection and pseudo-random request generation in section 4.4.2, we evaluated each algorithm using the same overall approach but with a distinct seed for the pseudo-random number generator. This strategy ensures our algorithm is tested against request sequences that differ from the ones encountered during training.

Furthermore, another key attention was given to the physical setup of the antennas. In a 5G environment, RSRP is a key performance indicator used to measure the signal strength of a cell's reference signal as received by a user equipment (UE), such as a smartphone or other connected device. Here are some key points about RSRP:

RSRP

- **Definition**: RSRP is defined as the linear average of the received power of all resource elements that carry cell-specific reference signals within the considered measurement bandwidth.

- **Measurement**: It is typically measured in dBm (decibels relative to one milliwatt). Higher RSRP values indicate stronger signal strength, which usually translates to better network performance and connectivity.
- **Impact on User Experience**: Strong RSRP generally leads to better throughput, lower latency, and fewer dropped connections, enhancing the overall user experience.

Given the logarithmic nature of the dBm measurement unit for RSRP, significant changes in actual power levels appear as relatively small changes in dBm values. However, these values can fluctuate slightly from second to second due to factors such as changes in air conditions or the presence of other 5G devices. Consequently, complete repeatability is rarely achievable, except in some specific controlled environments.

In order to create the most similar conditions, we tried our best to ensure that the Reference Signal Received Power (RSRP) was the same across all the tests.

As previously explained, although exact conditions cannot be guaranteed due to the limitations of the test-bed, this methodology was the best possible approach under the circumstances.

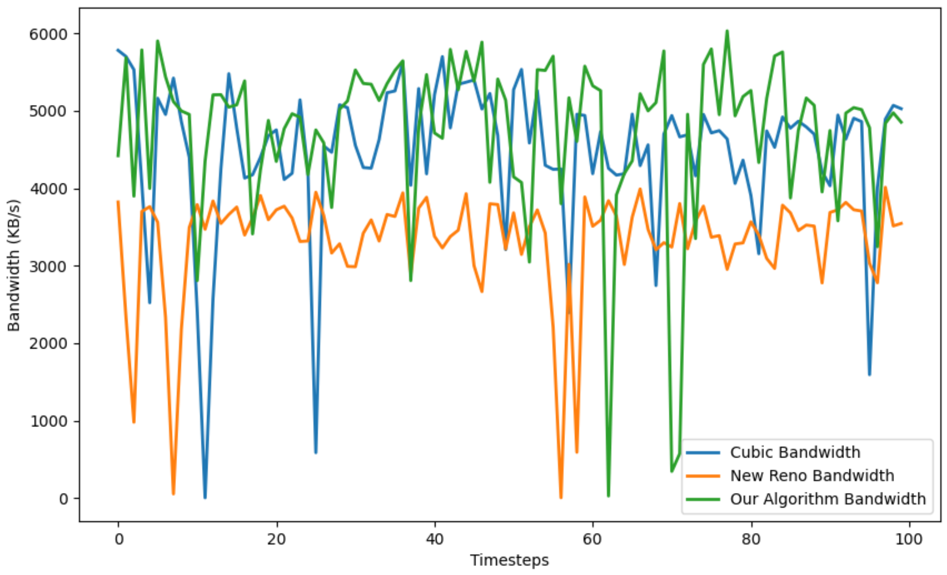

Figure 7.5: Bandwidth Comparison between different algorithms

In this first graph, related to the bandwidth, it is possible to appreciate the bandwidth gain of our algorithm compared to the other ones. As we can see, New Reno has turned out to be the worse amongst the three, this is most likely due to the conservative approach of the algorithm. New Reno increases the congestion window size using the additive increase approach, which is relatively conservative. This means it only increases the congestion window by one segment size per round-trip time (RTT) during the congestion avoidance phase. This slow growth limits the rate at which the sender can transmit more data, particularly in high-bandwidth, high-latency networks.

On the other hand, CUBIC has demonstrated significantly higher efficiency and better channel utilization.

This is due to, mainly, to a couple of reasons:

- **Cubic Growth Function**: CUBIC uses a cubic function to control the growth of the congestion window. This allows for more aggressive window growth in high-bandwidth environments, which helps in quickly ramping up the transmission rate and fully utilizing the available bandwidth.
- **Better Handling of Packet Loss**: CUBIC's algorithm is designed to recover from packet loss more gracefully. It reduces the congestion window more conservatively compared to New Reno, allowing for quicker recovery and less dramatic reductions in data transmission rates. This results in a more stable and efficient utilization of the channel.

Our algorithm, however, has demonstrated the ability to deliver superior results overall, achieving higher bandwidth utilization. Measuring both the average and standard deviation of the bandwidth across the three algorithms provides additional insights. In the following table all the values are in KBytes/s.

Algorithm	**Average**	**Standard Deviation**
Cubic	4267.2	876.2
New Reno	2674.1	1123.0
Our Algorithm	4539.5	950.3

Table 7.1: Bandwidth Average and Standard Deviation Comparison

As noted earlier, our algorithm achieved higher bandwidth utilization. However, as indicated in the table, this came at the expense of a higher standard deviation. This behavior is likely due to the algorithm's capability to enact more pronounced changes in the congestion window value, both during ramp-up and ramp-down phases.

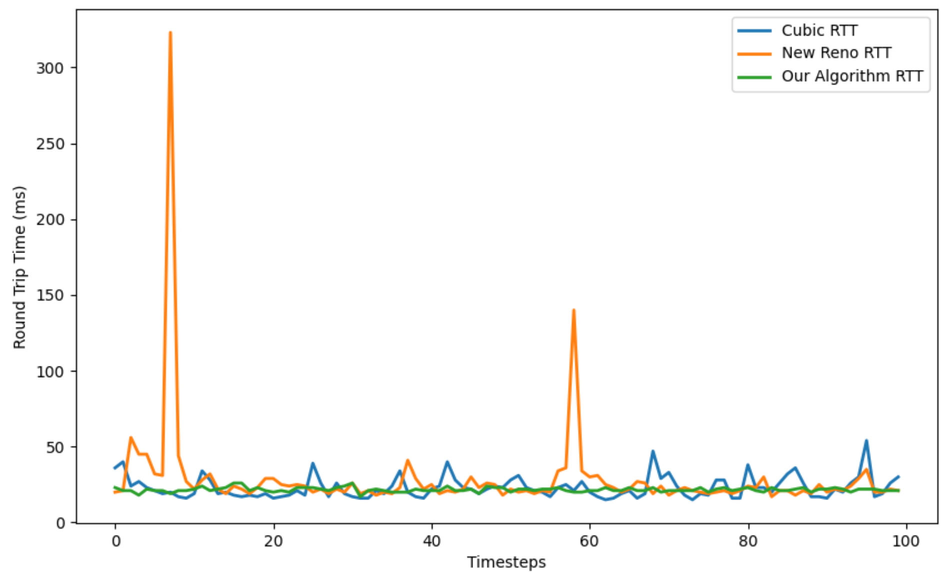

Figure 7.6: Round Trip Time Comparison between different algorithms

In this graph, we observe a similar trend as seen in the bandwidth plot. New Reno exhibits numerous large spikes, indicating its unsuitability for this type of problem. In contrast, CUBIC performs significantly better, demonstrating more stable and consistent behavior.

As done with the bandwidth, we will provide further insight into the behavior of the three algorithms, by presenting both the average and the standard deviation during testing phase. The values will be shown in ms, in order to keep the same scale as done in the plot.

Algorithm	**Average**	**Standard Deviation**
Cubic	27.6	23.2
New Reno	174.9	182.6
Our Algorithm	21.8	1.1

Table 7.2: Round Trip Time Average and Standard Deviation Comparison

The table below further shows the unsuitability of New Reno in this kind of setting. At the same time it shows the surprising stability and performances of our algorithm in the optimization of Round Trip Times. By looking at the standard deviation, we can see how it learned how to control the RTT surprisingly well. While there is no mathematically correct explanation for this behavior, we hypothesize that this is most likely due to the availability of lower level metrics.

Due to the significant variance observed in New Reno, we aim to offer a clearer comparison between our algorithm and CUBIC. Therefore, we will provide an additional plot that focuses solely on comparing our algorithm with CUBIC.

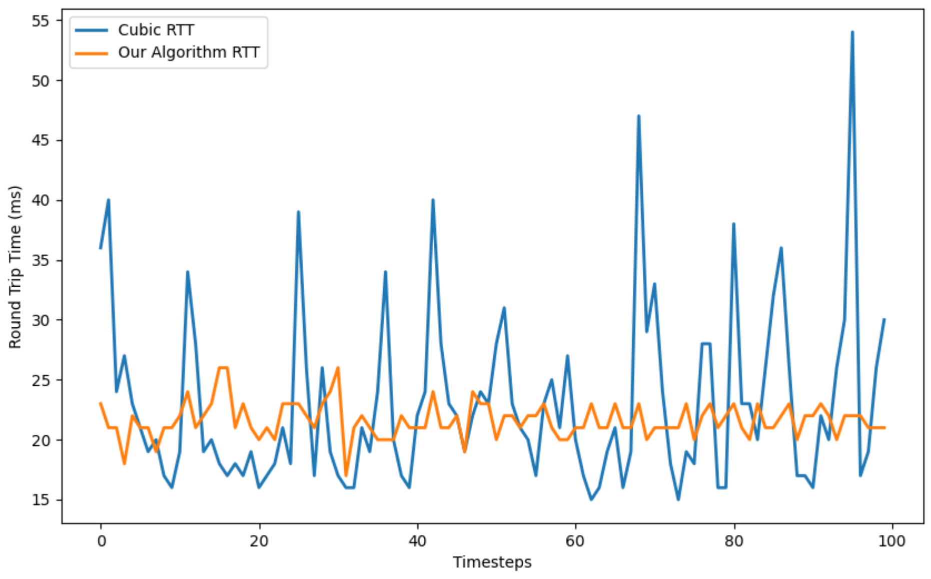

Figure 7.7: Round Trip Time Comparison between our algorithm and CUBIC

This plot allows us to more easily appreciate the differences between the two algorithms. While CUBIC's performance is much more stable compared to New Reno, it is out shined by the stability of our approach. Our algorithm demonstrates superior consistency and efficiency, largely due to its utilization of lower-level metrics. These metrics provide a more granular understanding of the network conditions, enabling our algorithm to make more informed decisions and adapt more effectively. As a result, our approach not only maintains higher stability but also optimizes bandwidth utilization, leading to improved overall performance.

Chapter 8

Conclusion

8.1 Summary of Findings

This thesis presented a novel approach to congestion algorithm for wireless network, by blending machine learning techniques with cross-layer metrics.

The primary goal was to enhance network performance by leveraging the predictive capabilities of machine learning to adapt congestion control algorithms dynamically.

Key finding for our research include:

- **Improved Bandwidth Utilization**: The proposed algorithm demonstrated superior bandwidth utilization compared to traditional congestion control algorithms. This was achieved through more accurate predictions of network conditions, allowing for optimal adjustment of transmission rates.
- **Enhanced Stability**: Despite the dynamic nature of wireless networks, our approach maintained a surprisingly higher level of stability in network performance, compared to traditional congestion control algorithms.
- **Reduced Latency**: Our approach proved effective in reducing latency while maintaining greater stability, resulting in smoother data flow and enhanced performance.

The findings listed above highlight the effectiveness of our approach, showcasing its ability to achieve remarkable stability while delivering enhanced bandwidth utilization and reduced latency. The stability provided by our method ensures consistent performance, even in the face of fluctuating network conditions. These results underscore the potential of integrating machine learning with cross-layer metrics in developing advanced congestion control algorithms for wireless networks.

8.2 Contributions

The primary contributions of this thesis include the development of a novel dataset, encompassing metrics from the physical layer to layer 4, collected in a 5G environment. Additionally, we demonstrated the effectiveness of using Reinforcement Learning combined with cross-layer metrics, tested within a 5G environment, to manage the congestion window. This approach achieved significant improvements over traditional congestion control algorithms.

8.3 Implications of Findings

The implications of our findings extend well beyond the immediate improvements in congestion control. They provide a robust foundation for future research and development in network management and optimization. By demonstrating the successful integration of machine learning with cross-layer metrics, our work opens up several avenues for further exploration and application.

The creation of the dataset described in section 4.4.3 not only supports current research efforts but also lays the groundwork for future innovations in wireless networking. By providing a comprehensive and standardized dataset, we aim to foster collaboration, accelerate advancements, and ultimately enhance the performance and reliability of wireless communication systems.

8.4 Limitations

While the results are quite promising, some further insight should be discussed, in order to understand the main limitations.

As detailed in Section 4.4.2, our methodology for data collection and model training, while straightforward, primarily involves using the same type of requests with varying values. This simplicity may restrict the model's ability to generalize across a wider range of request types and scenarios.

Another significant limitation is the static nature of our test bed. While this configuration enhances repeatability, it presents challenges for generalization, particularly regarding the input values of the reinforcement learning algorithm derived from the physical layer, which tend to be largely homogeneous.

In summary, a primary limitation of our work lies in its ability to generalize to diverse environments and conditions beyond the static setup of our test bed. This constraint raises concerns about the applicability of our findings to real-world scenarios characterized by varying network dynamics and conditions. Addressing these challenges will be crucial for extending the relevance and impact of our research beyond controlled experimental settings.

8.5 Recommendations for Future Research

As discussed in Section 6.2, the utilization of transformers shows considerable promise due to the sequential nature of the data involved. A potential improvement could be the adoption of a more advanced reinforcement learning algorithm, such as Proximal Policy Optimization (PPO) [15], which might offer enhanced performance due to its sophisticated optimization processes.

Additionally, transforming the input features in innovative ways could enhance the model's effectiveness. For instance, applying various transformations or normalizations to the input data could provide the model with a more refined and informative feature set.

Another promising idea is to explore different reward functions. Given the extensive feature space and the amount of available information, designing more complex and robust reward functions could leverage this data more effectively, potentially leading to significant performance improvements. By carefully tuning these reward functions, the model can be guided towards more desirable behaviors, further optimizing network performance.

Bibliography

[1] *Usage statistics of QUIC for websites.* `https://w3techs.com/technologies/details/ce-quic` (cit. on p. 1).

[2] *How Facebook is bringing QUIC to billions.* `https://engineering.fb.com/2020/10/21/networking-traffic/how-facebook-is-bringing-quic-to-billions/` (cit. on p. 1).

[3] Google. «The QUIC Transport Protocol: Design and Internet-Scale Deployment». In: (2017) (cit. on p. 2).

[4] *QUIC-go.* `https://github.com/lucas-clemente/quic-go` (cit. on p. 2).

[5] *MsQUIC.* `https://github.com/microsoft/msquic` (cit. on p. 3).

[6] *Modified QUIC.* `https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/cmu2.12154` (cit. on p. 3).

[7] *Quiche.* `https://github.com/cloudflare/quiche` (cit. on pp. 3, 24).

[8] *aioquic.* `https://github.com/aiortc/aioquic` (cit. on pp. 3, 24).

[9] Keith Winstein and Hari Balakrishnan. «TCP ex machina: computer-generated congestion control». In: *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM* (2013) (cit. on pp. 3, 6).

[10] Dong M., Meng T., Zarchy D., Arslan E., Gilad Y., Godfrey P.B., and Schapira M. «PCC Vivace: Online-Learning Congestion Control». In: *Proceedings of the 15th Usenix Symposium on Networked Systems Design and Implementation (Nsdi'18)* (2018) (cit. on p. 3).

[11] Li W., Zhou F., Chowdhury K.R., and Meleis W. «QTCP: Adaptive Congestion Control with Reinforcement Learning». In: *IEEE Netw. Sci. Eng.* (2019) (cit. on p. 3).

[12] Abbasloo S., Yen C.-Y., and Chao H.J. «Classic Meets Modern: A Pragmatic Learning-Based Congestion Control for the Internet». In: *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication* (2020) (cit. on pp. 3, 7).

[13] Nathan Jay, Noga H. Rotman, P. Brighten Godfrey, Michael Schapira, and Aviv Tamar. «A Deep Reinforcement Learning Perspective on Internet Congestion Control». In: *Proceedings of the 36th International Conference on Machine Learning* (2019) (cit. on pp. 3, 6).

[14] Zhang Z, Li S, Ge Y, Xiong G, Zhang Y, and Xiong K. «PBQ-Enhanced QUIC: QUIC with Deep Reinforcement Learning Congestion Control Mechanism». In: *Entropy* (2023) (cit. on p. 3).

[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. «Proximal Policy Optimization Algorithms». In: (2017) (cit. on pp. 3, 49).

[16] Injong Rhee and Lisong Xu. «CUBIC: A New TCP-Friendly High-Speed TCP Variant». In: () (cit. on pp. 5, 24, 39).

[17] Reno explanation (cit. on p. 5).

[18] Andrei Gurtov, Tom Henderson, Sally Floyd, and Yoshifumi Nishida. «The NewReno Modification to TCP's Fast Recovery Algorithm». In: () (cit. on pp. 6, 24, 39).

[19] Lawrence S. Brakmo and Larry L. Peterson. «TCP Vegas: A Fair and Efficient Congestion Control Algorithm». In: *IEEE Journal on selected areas in communications* (1995) (cit. on p. 6).

[20] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. «BBR: Congestion-Based Congestion Control». In: *ACM Queue, Volume 14, Issue 5* (2016) (cit. on p. 6).

[21] Mo Dong, Qingxi Li, Doron Zarchy, P. Brighten Godfrey, and Michael Schapira. «PCC: Re-architecting Congestion Control for Consistent High Performance». In: *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI '15)* (2015) (cit. on p. 7).

[22] H. Jonathan Chao Chen-Yu Yen Soheil Abbasloo. «Sage: Predictive Congestion Control». In: *Proceedings of the Annual Conference on Computer Communications (INFOCOM).* ACM, 2023. DOI: `10.1145/3603269.3604838`. URL: `https://dl.acm.org/doi/10.1145/3603269.3604838` (cit. on p. 7).

[23] Alessio Sacco, Matteo Flocco, Flavio Esposito, and Guido Marchetto. «Owl: Congestion Control with Partially Invisible Networks via Reinforcement Learning». In: *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications.* IEEE, 2021, pp. 1–9. DOI: `10.1109/INFOCOM42981.2021.9488851`. URL: `https://doi.org/10.1109/INFOCOM42981.2021.9488851` (cit. on p. 7).

[24] Yutaka Sakai, Tero Aalto, Ramesh Chandra, and Michael Steiner. «Adaptive Congestion Control for Unpredictable Cellular Networks». In: *Proceedings of the 20th ACM International Conference on Mobile Computing and Networking.* ACM. 2015, pp. 127–138. DOI: `10.1145/2785956.2787498` (cit. on p. 8).

[25] Joseph Winstein and D. Katabi. «Sprout: Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks». In: *Proceedings of the 12th USENIX Conference on Networked System Design and Implementation (NSDI).* USENIX Association. 2013, pp. 307–320. URL: `https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/winstein` (cit. on p. 8).

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. «Playing Atari with Deep Reinforcement Learning». In: (2013) (cit. on p. 31).

[27] Pranav Agarwal, Aamer Abdul Rahman, Pierre-Luc St-Charles, Simon J.D. Prince, and Samira Ebrahimi Kahou. «Transformers in Reinforcement Learning: A Survey». In: (2023) (cit. on p. 34).

[28] Wenzhe Li, Hao Luo, Zichuan Lin, Chongjie Zhang, Zongqing Lu, and Deheng Ye. «A Survey on Transformers in Reinforcement Learning». In: (2023) (cit. on p. 34).