DRL-based Adaptive Rate Allocation: An Intelligent Multipath Scheduler for Managing Volatile Wireless Paths in Multi-Access Networks

Submitted March 2026, in partial fulfillment of
the conditions for the award of the degree **PhD Electrical Engineering.**

Gregorio Maglione

Supervised by Dr Veselin Rakocevic

School of Science and Technology
City St George's, University of London

DECLARATION

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 60.000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 70 figures.

Acknowledgements

Thank you to the Blessed Virgin Mary, to whom this work is dedicated.

Thank you to Dr Veselin Rakocevic and Dr Touraj Soleymani, as well as the late Professor Thomas Chen, for the supervision during this work.

Thank you to Deutsche Telekom for funding this opportunity and for providing feedback, particularly to Dr Markus Amend and Nathalie Romo-Moreno.

Thank you to Dr Marcus Pieska for helping with a critical part of the testbed.

Abstract

Modern multi-access Beyond 5G (B5G) networks provide mobile terminals with aggregated capacity across heterogeneous paths, improving overall network performance. However, in high-mobility environments such as vehicular networks, supporting effective multi-access connectivity remains challenging. Rapid wireless link quality fluctuations frequently outpace the responsiveness of existing multipath schedulers, causing out-of-order packet delivery, head-of-line blocking, and degraded quality of experience.

This work addresses these challenges through DRL-based Adaptive Rate Allocation (DARA), a framework integrating Transformer-based path state forecasting with deep reinforcement learning for multipath traffic splitting. The Transformer module predicts congestion window and round-trip time evolution over a 500 ms horizon, whilst a Deep Q-Network dynamically computes optimal path utilisation fractions based on current observations and predicted states. A six-component reward function balances throughput maximisation, delay minimisation, quality sustainability, preemptive adaptation, stability, and idleness prevention.

Performance evaluation employs an MP-DCCP testbed with traces captured from real mobile users across five network conditions of increasing volatility. Controlled burst experiments validate the predictive mechanism, demonstrating substantial delay reduction through preemptive allocation adjustment. Real-world trace evaluation across file transfer, adaptive streaming, live video, and concurrent workload scenarios shows consistent improvements over both rule-based and state-of-the-art learning-based schedulers. Cross-protocol comparison on encrypted QUIC traffic demonstrates that DARA maintains performance gains where existing intelligent schedulers degrade, whilst concurrent workload tests confirm balanced flow allocation without the starvation effects observed in competing approaches. DARA's relative advantage increases with network volatility up to moderate-high mobility before graceful degradation under extreme conditions, validating the architecture's targeting of challenging high-mobility scenarios where reactive schedulers fundamentally cannot adapt.

Contents

List of Figures

List of Tables

List of Algorithms

Mathematical Notation

General Symbols

$\mathcal{P}$	Set of available network paths
p_i	Network path i
N	Number of paths/subflows
t	Time index
h	Prediction horizon
α	Learning rate
β	Smoothing factor
γ	Discount factor
ϵ	Exploration rate
τ	Soft update rate
ϕ_i	CWND utilisation fraction for path i

Network Parameters

$w_p(t)$, C_i	Congestion window for path p at time t
$\tau_p(t)$	Smoothed RTT for path p at time t
$\hat{w}_p(t+h)$	Predicted CWND for path p at horizon h
$\hat{\tau}_p(t+h)$	Predicted SRTT for path p at horizon h
BW_i	Available bandwidth on path i
ρ_i	Packet loss rate on path i
q_i	Queuing delay on path i
MSS_i	Maximum segment size on path i

Reinforcement Learning

s_t	State at time t
a_t	Action at time t
r_t	Reward at time t
$R(s_t, a_t)$	Reward function
$Q(s, a)$	Action-value function
$Q_\theta(s, a)$	Parameterised Q-function with weights θ
$Q'_{\theta'}(s, a)$	Target network Q-function
$\pi(a\|s)$	Policy function
$\mu(s)$	Deterministic policy
G_t	Discounted return from time t
$\mathcal{D}$	Experience replay buffer
$\mathcal{B}$	Mini-batch of transitions
$\mathcal{S}$	State space
$\mathcal{A}$	Action space
$L(\theta)$	Loss function

LSTM and Transformer

f_t	Forget gate activation
i_t	Input gate activation
o_t	Output gate activation
C_t	Cell state
h_t	Hidden state
W, b	Weight matrices and bias vectors
$\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$	Query, Key, Value matrices
d_k	Key dimension
d_{model}	Model dimension
PE	Positional encoding

Performance Metrics

T_i	Predicted completion time on path i
$D_{reorder}$	Reordering delay
W_{recv}	Receive window
$R_{throughput}$	Throughput reward component
R_{delay}	Delay reward component
$R_{quality}$	Quality sustainability reward component
$R_{preemptive}$	Preemptive adaptation reward component
$R_{stability}$	Stability penalty reward component
$R_{idleness}$	Idleness prevention reward component

Operators and Functions

$\sigma(\cdot)$	Sigmoid function
$\tanh(\cdot)$	Hyperbolic tangent function
$\odot$	Element-wise (Hadamard) product
$\arg\max$	Argument of the maximum
$\mathbb{E}[\cdot]$	Expected value
$\Phi(\cdot)$	Standard normal cumulative distribution function

List of Acronyms

5G	Fifth Generation
ABR	Adaptive Bitrate
ACPF	Adaptive Cheapest Pipe First
AI	Artificial Intelligence
APR	Average Playback Resolution
ATSSS	Access Traffic Steering, Switching, and Splitting
ATSSS-HL	ATSSS High Layer
ATSSS-LL	ATSSS Low Layer
B5G	Beyond 5G
BBR	Bottleneck Bandwidth and Round-trip propagation time
BDP	Bandwidth Product
BLEST	BLocking ESTimation
BPF	Best Path First
BPTT	Backpropagation Through Time
BS	Base Station
CC	Congestion Control
CCA	Congestion Control Algorithm
CCID	Congestion Control Identifier
CCID2	Congestion Control Identifier 2
CPF	Cheapest Path First
CWND	Congestion Window
DARA	DRL-based Adaptive Rate Allocation
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning

DSN Data Sequence Numbers

DSS Data Sequence Signal

eMBB Enhanced Mobile Broadband

ECF Earliest Completion First

HetNet Heterogeneous Network

HoL Head-of-Line

IoT Internet of Things

LSTM Long Short-Term Memory

MAB Multi-Armed Bandit

MAE Mean Absolute Error

MAMS Mobility-Aware Multipath Scheduler

MDP Markov Decision Process

MIMO Multiple-Input Multiple-Output

minRTT Minimum Round-Trip Time

ML Machine Learning

mMTC Massive Machine-Type Communication

MP-DCCP Multipath Datagram Congestion Control

MP-QUIC Multipath QUIC

MP-TCP Multipath TCP

MTU Maximum Transmission Unit

NAS Neural Architecture Search

NRMSE Normalised Root Mean Square Error

OFO Out-of-Order

OTIAS Out-of-Order Transmission for In-order Arrival Scheduler

PPO Proximal Policy Optimisation

QoE Quality of Experience

QoS Quality of Service

QUIC Quick UDP Internet Connections

RAT Radio Access Technology

RL Reinforcement Learning

RNN Recurrent Neural Network

RR Round Robin

RTT	Round-Trip Time
SACK	Selective Acknowledgment
SAP	Single Aggregate Path
SDN	Software-Defined Network
SRTT	Smoothed Round-Trip Time
SSN	Subflow Sequence Number
SWND	Sending Window
TD	Temporal Difference
txop	Transmission Opportunity
UCB	Upper Confidence Bound
UE	User Equipment
UPF	User Plane Function
VPN	Virtual Private Network

Chapter 1

Introduction

1.1 Motivation

Modern multi-access Beyond 5G (B5G) networks provide mobile terminals with aggregated capacity across heterogeneous paths, improving overall network stability and performance [1, 2]. However, in highly mobile environments such as vehicular networks, supporting effective multi-access connectivity remains challenging [3]. The rapid fluctuations of wireless link quality frequently outpace the responsiveness of existing multipath schedulers and transport-layer protocols, causing Out-of-Order (OFO) packet delivery, Head-of-Line (HoL) blocking, and degraded quality of experience [4, 5]. These challenges demand a rethinking of rate allocation strategies, moving beyond static or rule-based scheduling toward adaptive, data-driven approaches capable of anticipating capacity variations before they manifest in congestion feedback [6, 7, 8].

The fundamental limitation of reactive schedulers lies in the observation-reaction lag inherent to congestion control feedback [9, 10]. By the time a scheduler observes capacity degradation through increased round-trip times or packet loss, transmission opportunities have already been missed on alternative paths, and packets dispatched to the degrading path contribute to queue buildup and delay inflation [2, 8, 11].. In high-mobility scenarios where path characteristics fluctuate on sub-second timescales, this lag becomes particularly problematic—transient capacity bursts lasting 100–700 ms cannot be exploited by schedulers that require multiple round-trip times to detect and respond to changes [8].

Similarly, rapid degradation events cause bufferbloat [8] and increased out-of-order delivery [4, 12] before reactive mechanisms can shift traffic to stable paths.

Predictive scheduling addresses this fundamental limitation by anticipating capacity variations before they manifest in congestion feedback. Machine learning models trained on historical congestion control telemetry can forecast path state evolution, enabling preemptive rate adjustment that exploits predicted bursts and avoids predicted degradation [13, 8, 14]. However, existing prediction-based approaches face constraints: coarse-grained prediction horizons (multiple seconds) cannot capture sub-second dynamics [14, 15]; binary path selection cannot proportionally balance load across heterogeneous paths [4, 16, 15]; and single-objective optimisation cannot balance the competing requirements of throughput, delay, and stability that characterise diverse application workloads [17, 18].

1.2 Research Aim, Research Questions, and Objectives

1.2.1 Research Aim

The overarching aim of this thesis is to design, implement, and evaluate a predictive multipath scheduling framework that leverages deep learning to anticipate network capacity variations and dynamically optimises rate allocation across heterogeneous paths in volatile B5G environments. The framework addresses the fundamental limitation of reactive schedulers, which incur an observation-reaction lag that precludes effective adaptation when path characteristics fluctuate on sub-second timescales. By integrating time-series forecasting with deep reinforcement learning, the proposed approach seeks to exploit transient capacity bursts before they are observed through congestion feedback and to preemptively reduce allocation on degrading paths before queue buildup occurs, thereby improving throughput, reducing delay variance, and maintaining balanced resource utilisation across concurrent flows without requiring modifications to existing transport protocols.

1.2.2 Research Questions

The research aim is addressed through the following questions:

RQ1: How effectively can congestion control telemetry be forecast at sub-second horizons to inform multipath scheduling decisions in volatile heterogeneous networks?

This question investigates whether the temporal patterns present in Congestion Window (CWND) and Smoothed Round-Trip Time (SRTT) trajectories contain sufficient structure to enable accurate short-term prediction, and which neural architecture, whether recurrent, attention-based, or hybrid, is best suited to capturing the cross-path dependencies and sub-second dynamics characteristic of mobile wireless environments.

RQ2: What scheduling architecture enables proactive exploitation of predicted capacity variations whilst balancing competing performance objectives across diverse application workloads?

This question examines how predicted path states can be incorporated into a decision-making framework that jointly optimises throughput, delay, quality sustainability, and stability, and whether a multi-objective formulation outperforms single-metric scheduling heuristics when evaluated across file transfer, adaptive streaming, and concurrent workload scenarios.

RQ3: To what extent does a predictive, protocol-agnostic scheduler outperform state-of-the-art reactive and learning-based approaches under realistic mobile network conditions?

This question assesses the practical benefit of predictive scheduling by comparing the proposed framework against established baselines and state-of-the-art intelligent schedulers using real-world mobility traces, controlled burst experiments, and cross-protocol evaluation encompassing both observable and encrypted transport traffic.

1.2.3 Research Objectives

To address the research questions, the following objectives are defined:

RO1: Develop and validate time-series prediction models capable of forecasting CWND and SRTT trajectories at sub-second horizons. This objective entails the design of both an Long Short-Term Memory (LSTM)-based model for coarser-grained path quality predic-

tion and a Transformer-based model for fine-grained congestion metric forecasting, with evaluation against appropriate baseline predictors using real-world mobile network traces.

RO2: Formulate and optimise a multi-objective reward function that balances throughput, delay, quality sustainability, preemptive adaptation, stability, and idleness prevention for deep reinforcement learning-based rate allocation. This objective includes the application of Neural Architecture Search (NAS) to discover optimal reward weightings and the systematic evaluation of each reward component through ablation analysis.

RO3: Design and implement a complete predictive scheduling framework integrating the prediction models from **RO1** with the multi-objective Deep Q-Network (DQN) policy from **RO2**, including the kernel-userspace interface architecture required for millisecond-granularity state synchronisation between prediction components and the Multipath Datagram Congestion Control (MP-DCCP) packet scheduler.

RO4: Construct a reproducible experimental testbed using trace-driven emulation with real mobile user traces and validate the framework through controlled burst experiments and realistic trace evaluation across file transfer, adaptive streaming, live delivery, and concurrent workload scenarios.

RO5: Evaluate the proposed framework against baseline schedulers (Round Robin (RR), Best Path First (BPF), Earliest Completion First (ECF), Adaptive Cheapest Pipe First (ACPF), CPF) and state-of-the-art intelligent schedulers (Peekaboo, BLEST) under identical network conditions, and assess the generalisability, computational overhead, and practical deployment constraints of the approach.

1.3 Contributions

This research proposes and validates two novel frameworks for intelligent multipath scheduling in mobile heterogeneous environments: BPF, a path-switching scheduler, and DARA, a rate-splitting scheduler. BPF leverages a LSTM Recurrent Neural Network (RNN) model to predict path state over a 5-second horizon, enabling selection of the optimal path in a multipath framework using path time-throughput characteristics (rolling standard deviation, path bottleneck queue, path throughput) to prioritise the path with highest ex-

pected quality. DARA extends this predictive paradigm by integrating Transformer-based path state forecasting with deep reinforcement learning for multipath traffic splitting, predicting congestion window and round-trip time evolution over a 500 ms horizon whilst a DQN dynamically computes optimal CWND utilisation fractions across available paths. The principal contributions of this thesis are:

- **Prediction of Congestion Metrics:** Future congestion window and round-trip time trajectories are predicted using LSTM and Transformer models for real-time path switching and rate allocation respectively, achieving high accuracy over 500 ms prediction horizons. This approach provides superior responsiveness compared to relying solely on Congestion Control Algorithm (CCA) feedback, which most schedulers depend upon. The predictive capability proves particularly advantageous for mobile systems with rapidly changing link conditions that cannot be anticipated through reactive congestion control mechanisms.

- **Predictive Scheduling Architecture:** Leveraging predicted congestion metrics enables predictive scheduling as opposed to reactive scheduling, facilitating maximisation of a future-based reward function. This architecture enables exploitation of network capacity bursts before they are observed through congestion feedback, and preemptive rate reduction before capacity falls cause queue buildup. The result is improved network utilisation during transient high-capacity periods and prevention of packet loss and bufferbloat during degradation events.

- **Multi-Objective Reward Formulation:** A six-component reward function balances throughput maximisation, congestion-induced delay avoidance, quality sustainability, preemptive adaptation, stability, and idleness prevention—achieving Pareto-optimal trade-offs across competing objectives that single-metric heuristics cannot simultaneously optimise. NAS-based hyperparameter optimisation discovers optimal reward weightings across 100 trials.

- **Protocol-Agnostic Design:** Evaluation across FTP (observable inner congestion control), QUIC (encrypted transport opacity), adaptive streaming, live delivery,

and concurrent workloads demonstrates effectiveness regardless of inner protocol characteristics. Tunnel-layer prediction suffices for effective scheduling even without application-layer visibility, achieving 188% throughput improvement over encrypted QUIC where state-of-the-art intelligent schedulers degrade.

- **Advanced Testbed:** Experimental validation employs a custom-built trace-driven Mininet simulation with MP-DCCP protocol implementation and traces collected from real users moving in cellular networks. This testbed enables reproducible comparison of state-of-the-art scheduling solutions including Peekaboo and BLEST from the Multipath QUIC (MP-QUIC) framework, with results demonstrating consistent DARA superiority across diverse network conditions and application types.

Performance evaluation employs an MP-DCCP testbed with traces captured from real mobile users across five network conditions of increasing volatility. Controlled burst experiments validate the predictive mechanism, demonstrating 74% reduction in maximum delay through preemptive allocation adjustment. Real-world trace evaluation demonstrates that DARA achieves 67–200% download time improvements and 109–194% throughput gains for FTP transfers, with performance peaking under moderate-high volatility conditions. YouTube streaming achieves the highest resolution across all traces at the cost of increased rebuffering and a higher quality switch rate, reflecting an aggressive allocation strategy that prioritises peak quality and fast startup, with the strongest advantage in initial delay reduction. Live streaming achieves 13–187% faster startup. Cross-protocol comparison with state-of-the-art intelligent schedulers Peekaboo and BLEST on encrypted QUIC traffic demonstrates 188% throughput improvement on stable traces where Peekaboo degrades by 46%, with DARA maintaining 277–553% jitter improvement versus Peekaboo's 94–99% degradation. Under concurrent QUIC+TCP workloads, DARA achieves balanced flow allocation whilst Peekaboo exhibits 65–93% foreground QUIC starvation. Critically, DARA's relative advantage increases with network volatility up to moderate-high mobility before graceful degradation under extreme conditions, validating the architecture's targeting of challenging high-mobility scenarios where reactive schedulers fundamentally cannot adapt.

1.4 Structure of the Thesis

This thesis is organised as follows. Chapter 2 establishes the theoretical foundations, beginning with 5G network architecture, multipath transport protocols (Multipath TCP (MP-TCP), MP-QUIC, MP-DCCP), congestion control mechanisms, and scheduling architectures. The chapter then presents the Artificial Intelligence (AI) principles underpinning the proposed frameworks, including LSTM networks, Transformer models, and deep reinforcement learning with particular focus on DQN.

Chapter 3 presents a comprehensive review of AI-based multipath scheduling approaches, categorising existing work into redundancy-based, cross-layer, prediction-based, and multi-objective solutions. The review identifies research gaps that motivate the proposed frameworks and defines the baseline and state-of-the-art schedulers against which DARA is evaluated.

Chapter 4 presents the scheduling framework designs. The chapter first introduces the BPF scheduler, detailing its LSTM-based path quality prediction model operating over 5-second horizons and the priority-based scheduling mechanism that serves as a comparative baseline. The chapter then presents the DARA framework in full, comprising the Transformer-based prediction model for CWND and SRTT forecasting over 500 ms horizons, the DQN policy network formulation with its 18-dimensional state space and six-component reward function, and the NAS-optimised hyperparameter configuration discovered through 100 Optuna trials. The kernel-userspace interface architecture enabling millisecond-granularity state synchronisation between the prediction components and the MP-DCCP packet scheduler is described, along with the complete training algorithm integrating Transformer predictions with experience replay-based policy learning.

Chapter 5 describes the experimental infrastructure and evaluation methodology. The chapter justifies the selection of MP-DCCP as the multipath transport protocol and details the custom Wireshark dissector developed to support packet analysis. The Mininet-based testbed architecture and its validation against a hardware implementation are presented, followed by the Mahi-Mahi framework for trace-driven emulation of volatile wireless conditions. The baseline and state-of-the-art schedulers against which DARA is evaluated are

defined, including the parallel MP-QUIC testbed for Peekaboo and BLEST evaluation. Test scenarios spanning file transfer, streaming, and concurrent workload configurations are specified.

Chapter 6 presents experimental results in two parts: controlled burst scenario validation establishing DARA's predictive mechanism under deterministic conditions, and real-world trace evaluation assessing performance across five mobility scenarios of increasing volatility. Results span file transfer (FTP, QUIC), streaming (YouTube, live HLS), and concurrent workload scenarios, with comprehensive comparison against baseline and state-of-the-art schedulers.

Chapter 7 summarises the key findings, discusses deployment recommendations and limitations, and outlines directions for future work including latency-critical applications, online adaptation, hierarchical scheduling, and 5G-specific optimisation.

Chapter 8 lists the publications arising from this research.

Chapter 2

Theoretical Foundations

This chapter establishes the theoretical foundations underpinning intelligent multipath scheduling. Section 2.1 introduces multipath transport fundamentals, examining protocol architectures and congestion control mechanisms relevant to scheduling in heterogeneous networks. Section 2.2 presents the artificial intelligence techniques that inform the proposed frameworks, encompassing sequential prediction models for forecasting network metrics and reinforcement learning for decision-making under uncertainty.

2.1 Multipath Transport and Scheduling Fundamentals

This section establishes the networking context and architectural foundations for multipath scheduling in B5G networks—encompassing Fifth Generation (5G) and its evolutionary enhancements including 5G-Advanced and future higher-frequency extensions. It presents the network environment motivating multipath transport, details the multipath scheduling architecture including path management and buffer operations, examines the three principal multipath transport protocols, and analyses congestion control mechanisms and their interactions with scheduling decisions.

2.1.1 5G Network Context

B5G encompasses 5G and its evolutionary enhancements, including 5G-Advanced (3GPP Releases 18+) and future higher-frequency extensions. 5G is the latest cellular communication standard developed by the 3GPP, with development commencing under Release 16 and specified by TS 23.501 [1]. Whilst previous generations focused on improving functionality for cellular users, 5G systems target industrial applications including Industry 4.0 use cases such as augmented and virtual reality, remote surgery, Internet of Things (IoT), and autonomous robots. The expansion of cellular networks to accommodate these these applications specific requirements significantly increases the number of connected users through Massive Machine-Type Communications (mMTCs), balanced by substantially increased bandwidth known as Enhanced Mobile Broadband (eMBB). This is achieved primarily through networking solutions such as massive-Multiple-Input Multiple-Output (MIMO), millimetre-wave transmission, and beamforming. However, these advances increase the computational complexity of system models, and hardware improvement stagnation means software and network design must improve to maintain acceptable computing times [19]. Given these hardware limitations, greater emphasis is placed on digital components of the cellular network, particularly digitalisation including Software-Defined Network (SDN) and control layers for load balancing, as well as efficient data utilisation for handling network functions with AI.

One innovation of 5G is the Heterogeneous Network (HetNet), formed of macro cells containing femto and pico cells consisting of multiple Radio Access Technologies (RATs). This architecture removes coverage holes and adds network capacity. Vertical handover, known as intra-layer handover, is automatically triggered based on signal strength or for load balancing. Given projected increases in Internet traffic [20]-primarily due to video and gaming traffic, as well as Internet of Things devices such as autonomous vehicles [6], wearables, and smart appliances-the reduced cellular service area and higher frequency ranges will result in increased ping-pongs, signalling overheads, and interference [7]. These factors create unoptimised resource allocation scenarios, increase power consumption, and increase handovers over the same physical distance [21], all due to uplink limitations,

measurement reports, and handover confirmation transmission errors, as noted by Tayyab *et al.* [22]. This ultimately degrades Quality of Service (QoS) through increased loss, end-to-end delay, jitter, path downtime, and decreased transmission throughput [7]. Such degradation must be avoided given that 5G's focus on industrial applications demands delay-sensitive, latency-critical performance. Although IPv6-based mobility management protocols have been proposed to alleviate mobility-induced issues, none fully resolve these challenges, and many improvements such as caching and content prediction introduce additional delays [23].

Increasingly popular cellular-WiFi converged networks, as tested by AT&T, Verizon, and China Mobile [24], can provide multipath connections to moving devices in a HetNet, bringing latency and jitter to more acceptable levels for 5G use cases. This capability forms part of Access Traffic Steering, Switching, and Splitting (ATSSS), a 5G paradigm leveraging multipath algorithms to combine 3GPP and non-3GPP paths (i.e., 5G and Wi-Fi) for seamless handover and path resilience. ATSSS defines two operational modes: ATSSS Low Layer (ATSSS-LL) (link layer) for steering and switching traffic using network or local policies, and ATSSS High Layer (ATSSS-HL) (transport layer) for splitting traffic across multiple paths. The system uses congestion states, path availability, path status, latency, and load to enable traffic scheduling according to different steering modes such as active-standby, smallest delay, load-balancing, and priority-based allocation.

However, ATSSS does not specify detailed implementation, which can lead to counteractive behaviour or degradation. Inherent path asymmetry introduces packet reordering, OFO delivery, and HoL blocking [4]; bursty traffic patterns limit exploitation of secondary paths [25]; and layered interactions between multipath tunnels and congestion control mechanisms can negate throughput gains [12].

2.1.2 Multipath Scheduling Architecture

Multipath schedulers [2] aim to exploit the availability of multiple concurrent paths by routing traffic according to different priorities such as throughput maximisation, transmission cost reduction, or latency minimisation. They address fundamental challenges in

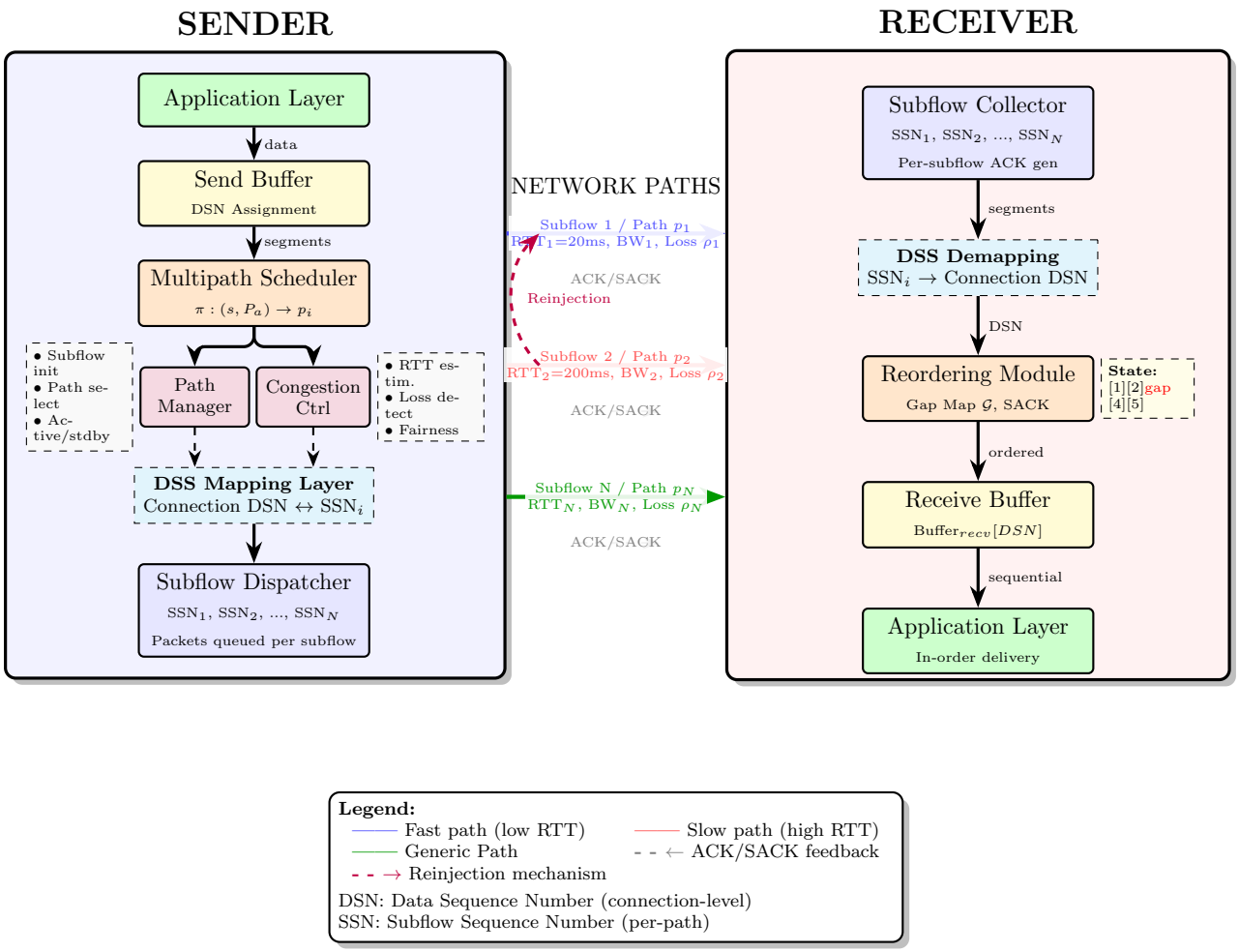

Figure 2.1: Architecture of Multipath Scheduling. The diagram illustrates the symmetric internal modules of the Sender and Receiver, the heterogeneous network paths (Fast vs. Slow), and the reinjection mechanism where data from underperforming paths (Red) is actively re-scheduled onto high-performance paths (Blue).

path management, packet scheduling, and congestion control, illustrated in Figure 2.1.

2.1.2.1 Path Management and Simultaneous Path Usage

A multipath connection comprises N subflows between sender and receiver. Unlike traditional single-path transport protocols, multipath schedulers enable simultaneous utilisation of heterogeneous network paths $P = \{p_1, p_2, \ldots, p_N\}$. The path manager initiates and manages subflow connections, determining which paths are active and when they should be established or terminated. The path manager can be configured to use the path with the lowest Round-Trip Time (RTT), or operate in active-standby mode where switching to the standby link occurs once the active link fails after a specified number of

retransmissions.

Each path p_i is characterised by distinct performance metrics including round-trip time (RTT_i), available bandwidth (BW_i), packet loss rate (ρ_i), and queuing delay (q_i). The fundamental challenge lies in efficiently distributing data segments across these heterogeneous paths whilst maintaining in-order delivery semantics at the application layer. When paths exhibit significant disparity-for instance, $\mathrm{RTT}_1 = 20\,\mathrm{ms}$ versus $\mathrm{RTT}_2 = 200\,\mathrm{ms}$-sophisticated scheduling becomes critical to avoid performance degradation.

2.1.2.2 Sender-Side Scheduling Operations

The sender maintains a connection-level send buffer of size $\mathrm{BufferSize}_{\mathrm{send}} \geq \sum_{i=1}^{N} C_i \cdot \mathrm{MSS}_i$, where C_i denotes the congestion window and MSS_i the maximum segment size for path i. This buffer contains application data segmented and assigned Data Sequence Numbers (DSN). At each transmission opportunity, the scheduler executes a mapping $\pi : (s, P_a) \to p_i$ that selects which segment to transmit on which active path based on current state s, including buffer occupancy, path metrics, and in-flight data.

Scheduling policies vary in their objectives. Some minimise latency by preferring low-RTT paths, whilst others maximise throughput by distributing load proportional to capacity using weights $w_i = \mathrm{BW}_i / \sum_j \mathrm{BW}_j$, where w_i represents the fraction of total bandwidth allocated to path i. Advanced approaches predict completion times for each path:

$$T_i = t_{\mathrm{current}} + \frac{|s|}{\mathrm{BW}_i} + \mathrm{RTT}_i + q_i \tag{2.1}$$

and select $p_{\mathrm{next}} = \arg\min_i T_i$. The scheduler must also handle packet reinjection, where segments experiencing loss or excessive delay on path p_i are retransmitted on alternative path $p_j \neq p_i$. This requires careful coordination through dual sequence numbering: each subflow maintains per-path Subflow Sequence Numbers (SSNs), whilst connection-level DSN provides unified sequencing. The mapping between these spaces, such as through the Data Sequence Signal (DSS) in MP-TCP, enables the same data to be transmitted on different subflows.

2.1.2.3 Receiver-Side Reassembly and Buffer Management

The receiver faces the challenge of reassembling out-of-order data arriving across paths with differing delays. Segments with connection-level sequence number DSN $= d$ are buffered in $\text{Buffer}_{\text{recv}}[d]$ but can only be delivered in strict sequence order. This creates HoL blocking: delayed segments on slow paths prevent delivery of subsequent segments from fast paths, even when those segments have already arrived. The receiver maintains a gap map $G = \{[g_{\text{start}}, g_{\text{end}}]\}$ tracking missing ranges and generates Selective Acknowledgments (SACKs) to indicate non-contiguous received ranges for accelerated loss recovery. Required buffer capacity scales with path heterogeneity. For paths with RTT ratio $r = \text{RTT}_{\max}/\text{RTT}_{\min}$ and aggregate bandwidth B, Ferlin *et al.* [4] demonstrate that:

$$\text{BufferSize}_{\text{recv}} \geq B \cdot \text{RTT}_{\max} \cdot (r - 1) \tag{2.2}$$

When the receive buffer approaches capacity, the advertised receive window $W_{\text{recv}} = \text{BufferSize}_{\text{total}} - \text{BufferSize}_{\text{occupied}}$ shrinks, back-pressuring the sender. Insufficient buffering causes stalling where fast paths must pause waiting for slow path delivery, negating multipath benefits. The reordering delay $D_{\text{reorder}} = t_{\text{complete}} - t_{\text{first\_arrival}}$ quantifies the time between first segment arrival and complete block availability, typically approximating $\text{RTT}_{\max} - \text{RTT}_{\min}$ for heterogeneous paths.

2.1.2.4 Path Estimation and Monitoring

Effective scheduling requires continuous path characterisation. The sender employs RTT estimation using exponentially-weighted moving averages:

$$\text{SRTT}_i(t+1) = (1 - \beta) \cdot \text{SRTT}_i(t) + \beta \cdot \text{RTT}_{\text{sample},i} \tag{2.3}$$

where $\beta \in [0.125, 0.25]$ represents the smoothing factor. This range originates from RFC 6298 [9], reflecting a trade-off: smaller values provide stability against measurement noise but respond slowly to genuine changes, whilst larger values enable rapid adaptation but amplify transient fluctuations. Bandwidth estimation observes congestion window dy-

namics:

$$\widehat{\mathrm{BW}}_i = \frac{C_i \cdot \mathrm{MSS}_i}{\mathrm{SRTT}_i} \tag{2.4}$$

Loss rate tracking monitors $\hat{\rho}_i = \mathrm{packets\_lost}_i / \mathrm{packets\_sent}_i$ over sliding windows.

2.1.3 Multipath Transport Protocols

Multipath transport protocols extend single-path semantics across multiple network interfaces, enabling simultaneous utilisation of heterogeneous paths. Three principal protocols have emerged for ATSSS-HL implementation: MP-TCP [26], MP-QUIC [27], and MP-DCCP [28]. Their architectural differences-summarised in Table 2.1-determine suitability for different deployment scenarios, particularly regarding reliability semantics, congestion control coupling, and scheduler flexibility.

2.1.3.1 MP-TCP: Reliable Byte-Stream Multipath

MP-TCP [26] extends TCP's reliable byte-stream abstraction across multiple subflows through a dual sequence space architecture. Connection-level DSNs provide end-to-end ordering, mapped to per-subflow SSNs via the DSS option. Each subflow operates as an independent TCP connection with its own sequence space, acknowledgements, and retransmission logic, whilst the DSS mapping enables receiver-side reassembly into the original byte stream.

Connection establishment extends TCP's three-way handshake with the `MP_CAPABLE` option exchanging 64-bit keys for subflow authentication. Subsequent subflows join via `MP_JOIN`, authenticated using HMAC-SHA1 derived from the initial key exchange. Path management employs `ADD_ADDR` for address announcement and `MP_PRIO` for backup path designation, enabling dynamic path addition without connection re-establishment.

The protocol supports both coupled and uncoupled congestion control. Coupled algorithms such as LIA [29], OLIA [30], and BALIA [31] link subflow CWND adjustments to ensure fairness at shared bottlenecks. Upon receiving acknowledgement on subflow i, the

Characteristic	MP-TCP	MP-QUIC	MP-DCCP
Sequencing & Reliability			
Sequence space	Dual (DSN via DSS ↔ SSN)	Per-path packet numbers	Independent (MP_SEQ 48-bit)
Reliability scope	Connection-wide byte-stream	Per-stream with flow control	None (delegated)
OFO handling	Mandatory buffering	Per-stream buffering	Immediate delivery
HoL blocking	Connection-wide	Per-stream	None
Connection & Path Management			
Handshake	3-way per subflow	1-RTT with PATH_CHALLENGE	4-way with HMAC
Path addition	MP_JOIN + token	PATH_CHALLENGE / RESPONSE	MP_JOIN + MP_HMAC
Path signalling	MP_PRIO, ADD_ADDR	PATH_STATUS, PATH_ABANDON	MP_PRIO (4-bit), MP_ADDADDR
Authentication	HMAC-SHA1 (64-bit key)	TLS 1.3 integrated	HMAC-SHA256 (MP_KEY)
RTT exchange	Implicit (timestamp option)	Implicit (ACK delay field)	Explicit (MP_RTT: raw/min/max/SRTT)
Congestion Control			
CC architecture	Coupled or uncoupled	Independent per-path	Independent (modular Congestion Control Identifier (CCID))
CWND coordination	Linked via shared α (coupled); independent (uncoupled)	None (per-path state)	None (per-profile state)
Algorithm flexibility	Single algorithm, all subflows	Per-path selection	Per-path profile (CCID2/CCID3)
Scheduler constraint	Cross-path dependencies (coupled)	Stream-level backpressure	Capacity-only
Scheduler Integration			
Scheduler granularity	Byte-level with segmentation	Frame-level with reassembly	Packet-level direct
Receiver buffer impact	Connection-wide ordering	Per-stream ordering	Optional reordering
Burst exploitation penalty	Causes OFO + HoL blocking	Per-stream OFO + blocking	No penalty
Deployment Characteristics			
Nested CC severity	High (coupling amplifies)	Moderate (stream blocking)	Low (unreliable delivery)
Encapsulation suitability	Poor (timeout conflicts)	Moderate (stream delays)	High (no reliability conflicts)
Middlebox traversal	Problematic (option stripping)	Native (UDP encapsulation)	Limited (UDP tunnelling available)
Path addition latency	Full 3-way handshake	1-RTT with validation	4-way with HMAC auth

Table 2.1: Multipath transport protocol characteristics

window increases by:

$$\Delta C_i = \frac{\min(\alpha/C_{\text{total}}, 1/C_i)}{C_i} \tag{2.5}$$

where $C_{\text{total}} = \sum_j C_j$ and:

$$\alpha = C_{\text{total}} \cdot \frac{\max_j(C_j/\text{RTT}_j^2)}{(\sum_j C_j/\text{RTT}_j)^2} \tag{2.6}$$

Loss detection triggers multiplicative decrease: $C_i \leftarrow C_i \cdot (1-\beta/2)$ with $\beta = 0.5$. This coupling ensures TCP-friendliness but creates cross-path dependencies: scheduling decisions on one path affect CWND growth on all paths through the shared α computation [4]. Uncoupled operation treats each subflow independently, avoiding these dependencies but potentially causing unfairness when subflows share bottlenecks with single-path TCP flows.

The strict in-order delivery requirement creates pathological behaviour under path heterogeneity. When $\text{RTT}_1 \ll \text{RTT}_2$, packets transmitted on the fast path arrive substantially before slow-path counterparts, necessitating receiver buffering until sequence gaps close. This HoL blocking prevents application delivery despite data availability, with buffer requirements scaling according to Equation 2.2. Despite advances addressing fairness [31, 32], bottleneck estimation [31, 33], and HoL blocking reduction [5, 34], coupled Congestion Control (CC) remains inefficient in wireless networks due to oscillatory behaviour when path characteristics fluctuate rapidly.

2.1.3.2 MP-QUIC: Stream-Multiplexed Multipath

MP-QUIC [27, 35] extends QUIC's encrypted, multiplexed transport across multiple network paths. Each path maintains an independent packet number space, eliminating the DSS mapping overhead of MP-TCP. The protocol inherits QUIC's stream abstraction, where multiple logical streams multiplex over a single connection with per-stream flow control and ordering constraints.

Path management employs dedicated frame types: `PATH_CHALLENGE` initiates path validation with an 8-byte random payload; `PATH_RESPONSE` echoes this payload to confirm

reachability; `PATH_STATUS` signals path preferences (available, standby, or deprecated); and `PATH_ABANDON` indicates path removal. The 1-RTT handshake-leveraging TLS 1.3 integration-reduces connection establishment latency compared to MP-TCP's per-subflow three-way handshakes, whilst providing encryption of transport headers for enhanced privacy.

Congestion control operates independently per path without the coupling constraints of MP-TCP, with each path maintaining its own CWND, RTT estimates, and loss recovery mechanisms. This independence provides schedulers greater flexibility-transmissions on one path do not directly affect another path's CC state. However, per-stream reliability creates indirect coupling through receive buffer dynamics: HoL blocking within streams-where lost or delayed frames prevent delivery of subsequent in-order frames-causes receive window exhaustion that back-pressures all paths. Wu *et al.* [2] and Shi *et al.* [36] demonstrate demonstrate that when path RTT disparity causes frequent OFO frame delivery, receive buffers fill with frames awaiting predecessors from slower paths, eventually constraining fast-path transmission through shrinking window advertisements. This per-stream granularity partitions the HoL blocking problem rather than eliminating it-applications using few streams experience similar pathologies to MP-TCP.

2.1.3.3 MP-DCCP: Unreliable Datagram Multipath

MP-DCCP [28, 37] provides unreliable datagram delivery with congestion control across multiple paths, extending DCCP's design philosophy to multipath scenarios. The protocol eliminates mandatory reliability, delegating ordering and retransmission semantics to encapsulated applications-a fundamental architectural distinction enabling transparent tunnelling of arbitrary traffic types without nested reliability conflicts.

Option Architecture. The protocol defines option type 46 (Multipath) with sub-options organised into functional categories, detailed in Table 2.2.

Connection management options establish and terminate multipath connections. `MP_KEY` (sub-option 3) exchanges cryptographic keys during initial handshake, supporting plaintext, HMAC-SHA256, and HMAC-SHA512 variants with key lengths of 8–64 bytes.

Option Type	Sub-Option	Length (bytes)
DCCP Option Type 46: Multipath		
Connection Management		
	MP_CONFIRM (0)	Variable
	MP_JOIN (1)	12
	MP_KEY (3)	17–73
	MP_CLOSE (10)	Variable
	MP_FAST_CLOSE (2)	Variable
Sequencing and Authentication		
	MP_SEQ (4)	9
	MP_HMAC (5)	23
Path Management		
	MP_RTT (6)	12
	MP_ADDADDR (7)	12–26
	MP_REMOVEADDR (8)	8
	MP_PRIO (9)	4
Experimental		
	MP_EXP (11)	Variable

Table 2.2: MP-DCCP option hierarchy

`MP_JOIN` (sub-option 1) establishes additional subflows, carrying a 32-bit path token and nonce for authentication. `MP_CLOSE` (sub-option 10) and `MP_FAST_CLOSE` (sub-option 2) terminate connections gracefully or immediately.

Sequencing options provide connection-level ordering without mandatory enforcement. `MP_SEQ` (sub-option 4) carries a 48-bit connection-level sequence number independent of per-path sequence spaces. Unlike MP-TCP's mandatory DSS mapping, `MP_SEQ` serves solely for *optional* receiver-side reordering-packets may be delivered immediately upon arrival regardless of sequence position. `MP_HMAC` (sub-option 5) provides 20-byte truncated HMAC-SHA256 authentication, mandatory for `MP_JOIN`, `MP_ADDADDR`, and `MP_REMOVEADDR` options.

Path management options enable dynamic path configuration. `MP_ADDADDR` (sub-option 7) announces additional addresses with 12 bytes for IPv4 or 24 bytes for IPv6, plus an optional 2-byte port field. `MP_REMOVEADDR` (sub-option 8) withdraws previously announced addresses. `MP_PRIO` (sub-option 9) signals 4-bit path priorities, enabling dynamic preference adjustment without connection re-establishment. `MP_RTT` (sub-option 6) explicitly

communicates round-trip time measurements with a 4-bit type field distinguishing raw, minimum, maximum, and smoothed estimates, accompanied by 24-bit age timestamps indicating measurement freshness-providing schedulers with direct path quality information without inference from acknowledgement timing.

Congestion Control Integration. MP-DCCP employs modular congestion control through CCID profiles, with each path independently instantiating its selected profile. CCID2 provides TCP-like AIMD behaviour suitable for bulk transfers; CCID3 implements TFRC-based equation-driven rate control for streaming applications. This modularity enables per-path algorithm selection-critical for heterogeneous deployments where cellular and WiFi paths may benefit from different CC characteristics.

The absence of coupled congestion control eliminates cross-path CWND dependencies present in coupled MP-TCP. Scheduler decisions on path i affect only that path's congestion state, providing maximal flexibility for learning-based schedulers to independently modulate per-path utilisation without unintended interference through shared state variables.

Implications for Scheduler Design. The unreliable delivery model fundamentally alters scheduler constraints compared to reliable protocols. First, packets arriving OFO are delivered immediately to the application layer rather than buffered indefinitely awaiting predecessors, eliminating receiver-side stalls when slow-path packets delay. Second, receive buffers need not accommodate path RTT disparity products; application-level buffering handles any required reordering according to application semantics. Third, schedulers may aggressively utilise predicted capacity bursts on secondary paths without subsequent OFO delivery causing throughput degradation through buffer exhaustion. Fourth, tunnelling reliable protocols (TCP, QUIC) through MP-DCCP avoids nested reliability conflicts-inner protocol retransmissions operate independently of outer multipath delivery.

These characteristics render MP-DCCP particularly suitable for ATSSS-HL deployments requiring traffic-agnostic operation, and for learning-based schedulers exploiting predictive

capacity estimation where OFO delivery tolerance enables aggressive path utilisation.

2.1.4 Congestion Control in Multipath Transport

Congestion control mechanisms fundamentally constrain multipath scheduler operation by determining per-path transmission capacity through CWND limits. The selection of CCA for tunnel deployments critically affects both path aggregation effectiveness and interactions with encapsulated traffic [12]. This section establishes the algorithmic taxonomy, analyses the nested congestion control problem arising in tunnel architectures, and examines scheduling interactions with congestion control dynamics.

2.1.4.1 Congestion Control Algorithm Taxonomy

Transport-layer CC algorithms regulate sending rates to prevent network collapse whilst maximising throughput [38]. These algorithms partition into loss-based approaches, which interpret packet loss as congestion signals, and model-based approaches, which estimate path capacity through bandwidth and delay measurements. Table 2.3 summarises the principal algorithms relevant to multipath transport deployments.

The distinction between loss-based and model-based algorithms proves critical for multipath tunnel deployments. Loss-based algorithms (NewReno, CUBIC) interpret any packet loss as a congestion signal, triggering CWND reduction regardless of actual cause. This behaviour causes spurious rate reductions in environments with non-congestion losses [8]. In wireless environments where such losses occur due to channel errors, handovers, or interference, Xie *et al.* [10] show this behaviour causes unnecessary throughput degradation. Model-based algorithms such as BBR maintain rate estimates independent of isolated losses, providing more stable throughput in volatile conditions characteristic of cellular-WiFi heterogeneous networks, as demonstrated by Song *et al.* [44].

2.1.4.2 The Nested Congestion Control Problem

Transport-layer multipath tunnels introduce *nested congestion control*: multiple CC instances operating simultaneously at different protocol layers. Pieska *et al.* [12] analyse

Algorithm	Type	Operational Characteristics
NewReno [39]	Loss-based	Additive increase, multiplicative decrease (AIMD); linear CWND growth of 1/CWND per ACK until packet loss; halves CWND on triple duplicate ACK; fills bottleneck buffers before detecting congestion; poor utilisation recovery on high-BDP paths; default behaviour for TCP-like DCCP profiles
CUBIC [40]	Loss-based	Cubic function $W(t) = C(t-K)^3 + W_{\max}$ for CWND growth where K represents time since last congestion event; BDP-independent convergence time; default Linux TCP since kernel 2.6.19; aggressive probing after loss recovery; common in QUIC implementations
BBR [41]	Model-based	Estimates bottleneck bandwidth ($\widehat{\text{BtlBw}}$) via maximum observed delivery rate and minimum RTT (RTprop) via timestamp differentials; maintains CWND $\approx 2 \times \widehat{\text{BtlBw}} \times \text{RTprop}$; periodic 1.25× bandwidth probing phases followed by drain phases; does not reduce sending rate on isolated packet loss; basis for BBR-like DCCP profiles
BBRv2 [42]	Model-based	Extends BBR with explicit loss responsiveness; reduces CWND when sustained loss rate exceeds configurable threshold; improved fairness when competing with loss-based flows at shared bottlenecks; currently experimental
TFRC [43]	Rate-based	Equation-based rate control using TCP throughput formula; smooth rate changes without saw-tooth behaviour; TCP-friendly throughput characteristics; designed for streaming applications requiring stable sending rates

Table 2.3: Congestion control algorithms in multipath transport contexts

this phenomenon, demonstrating that when end-to-end traffic traverses a multipath tunnel, the server's CC loop encloses per-path tunnel CC loops, creating interactions that can negate aggregation benefits.

The server's end-to-end CC-operating between content server and client-observes aggregate RTT and loss characteristics combining Internet path and tunnel path behaviour. Within the tunnel, each path instantiates independent CC managing transmission between proxy and client, responding to path-specific feedback at potentially different timescales than end-to-end CC.

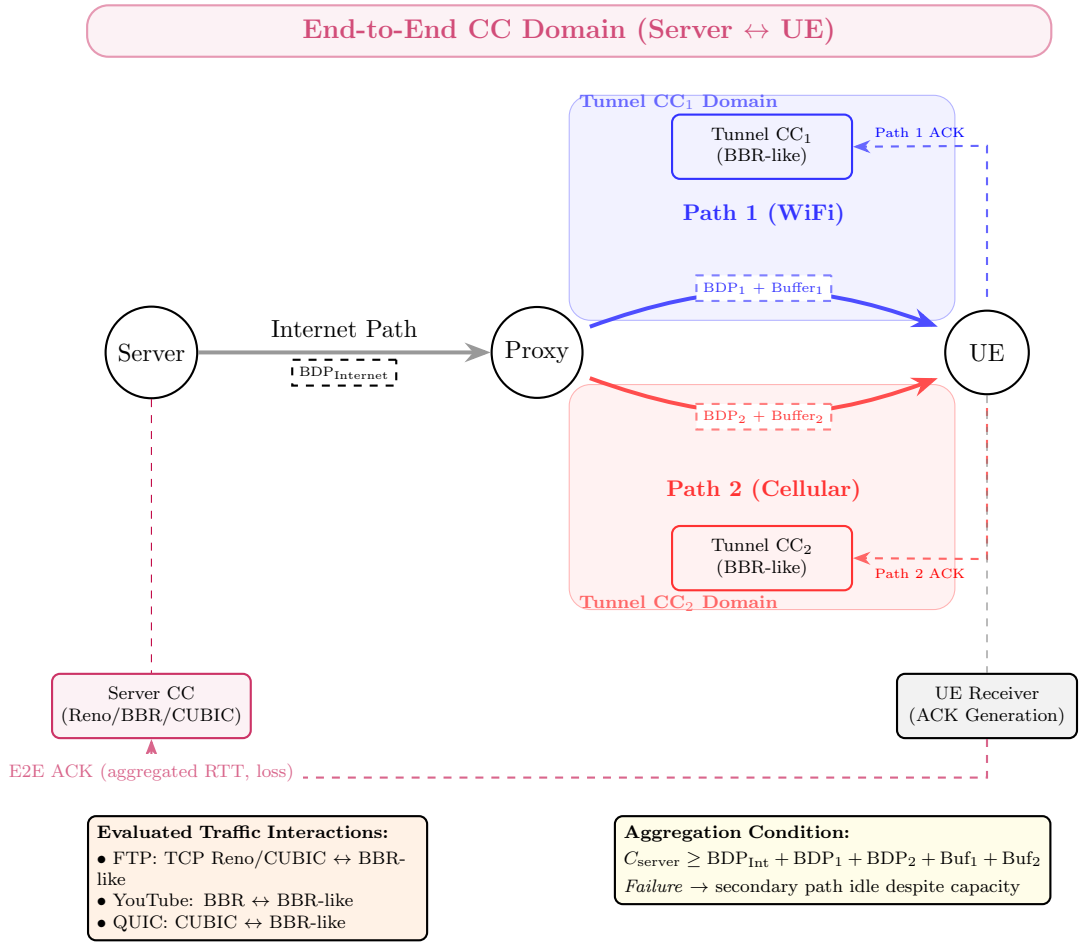

Figure 2.2: Nested congestion control architecture. The Server's CC operates unaware of the tunnel structure, while independent BBR-like tunnel controllers manage the WiFi and Cellular paths.

Aggregation Condition. Effective path aggregation requires the server's CC to maintain sufficient in-flight data to populate multiple tunnel paths simultaneously. Denoting

the server's CWND as C_{server}, aggregation occurs when [12]:

$$C_{\text{server}} \geq \text{BDP}_{\text{Internet}} + \sum_{i=1}^{N} (\text{BDP}_i + \text{Buffer}_i) \tag{2.7}$$

where $\text{BDP}_{\text{Internet}}$ represents the bandwidth-delay product between server and proxy, BDP_i the product for tunnel path i, and Buffer_i the bottleneck buffer capacity. When this condition fails, secondary paths remain idle regardless of scheduler decisions.

Algorithm-Specific Interaction Patterns. The severity of nested CC interactions depends on algorithm combinations [12]. In model-based/model-based combinations (e.g., BBR end-to-end with BBR-like tunnel), both instances estimate minimum RTT and bottleneck bandwidth; performance depends on relative RTT distribution-when tunnel RTT exceeds approximately 80% of end-to-end RTT, synchronised probing phases cause destructive interference preventing secondary path utilisation.
In loss-based/model-based combinations (e.g., CUBIC end-to-end with BBR-like tunnel), tunnel BBR-like CC periodically probes bandwidth, potentially causing transient buffer overflow; the server's CUBIC interprets resulting losses as congestion, reducing C_{server} and starving the tunnel.
In model-based/loss-based combinations (e.g., BBR end-to-end with NewReno-like tunnel), NewReno-like tunnel CC fills buffers through loss-driven growth. Only when server Bandwidth Product (BDP) estimates exceed buffer-inflated tunnel capacity does overflow to secondary paths occur-larger buffers exacerbate this problem.

2.1.4.3 BBR-like versus NewReno-like Tunnel Congestion Control

For multipath tunnel deployments, BBR-like congestion control offers substantial advantages over NewReno-like approaches across multiple dimensions:

Predictable CWND Evolution. BBR's operational model produces characteristic CWND patterns: steady-state operation at approximately $2 \times$ BDP, periodic probing phases with $1.25\times$ rate increase lasting one RTT, and subsequent drain phases [41]. These

patterns exhibit temporal regularity exploitable by sequence prediction models. Jay *et al.* [13] demonstrate that NewReno-like algorithms exhibit irregular CWND evolution driven by stochastic loss events-the saw-tooth growth pattern creates discontinuities that degrade prediction accuracy.

Wireless Channel Resilience. Cellular networks under mobility conditions produce packet losses unrelated to congestion-channel errors during handover transitions, buffer overflow at radio interfaces during signal degradation. Nguyen *et al.* [8] analyse these phenomena extensively. NewReno-like CC interprets such losses as congestion signals, triggering CWND halving that reduces throughput during already-degraded conditions. BBR-like CC continues operating at its modelled rate despite isolated losses, providing stable capacity baselines. Song *et al.* [44] confirm this resilience in practical deployments.

Reduced Buffer Bloat. BBR targets operation at the bandwidth-delay product rather than filling buffers until loss occurs [41]. NewReno-like algorithms continuously increase CWND until packet loss signals buffer exhaustion, inducing queuing delays proportional to buffer capacity-the bufferbloat problem identified [10]. For tunnel deployments, BBR-like CC reduces baseline queuing delays, improving SRTT measurement quality and end-to-end latency for delay-sensitive applications.

Faster Capacity Discovery. BBR's explicit bandwidth probing discovers capacity changes within one RTT cycle. NewReno-like algorithms require multiple RTTs to recover from loss events through slow-start or congestion avoidance phases [40], potentially missing transient burst opportunities lasting hundreds of milliseconds. Nguyen *et al.* [8] observe that such capacity variations are common in high-mobility scenarios where capacity varies on sub-second timescales.

2.1.4.4 Scheduling Impact on Congestion Control Dynamics

The relationship between scheduling and congestion control is bidirectional: CC algorithms determine per-path CWND constraining scheduler decisions, whilst scheduler

packet distribution affects which paths experience congestion signals. Wu *et al.* [2] and Ferlin *et al.* [4] analyse these interactions extensively.

Preferential routing to a specific path increases that path's load. If load exceeds capacity, queuing develops at bottleneck buffers, manifesting as rising RTT [8]. Loss-based CC triggers CWND reduction; model-based CC updates path models. Scheduling decisions at time t affect CC state at $t + \text{RTT}$, constraining subsequent scheduling [36]. Reactive schedulers face a fundamental lag: by the time CWND reduction signals degradation, packets dispatched during the preceding RTT already contribute to congestion [14].

Rate control through CWND fraction adjustment (ϕ_i) provides a clean interface between scheduling logic and CC operation [25]. The scheduler treats effective capacity as $\phi_i \times \text{CWND}_i$, constraining utilisation without modifying CC behaviour. This enables preemptive throttling when predictions indicate degradation, whilst underlying CC continues normal operation based on actual network feedback [45].

BBR-like CC periodically enters bandwidth probing phases with 1.25× rate increase [41]. Predictive schedulers can anticipate probing from historical CWND patterns, modulating utilisation fractions to prevent amplifying probing-induced queue buildup on congestion-prone paths. Jay *et al.* [13] demonstrate that this coordination-adjusting scheduler behaviour in anticipation of CC dynamics-distinguishes predictive from reactive approaches.

2.2 Artificial Intelligence Foundations

The limitations of reactive schedulers-responding to path changes only after degradation manifests, relying on simple heuristic models, and optimising single objectives-motivate the application of machine learning techniques capable of learning complex patterns and predicting future conditions. This section introduces the AI techniques underpinning the proposed scheduling frameworks, focusing on two complementary capabilities: *sequential prediction* for forecasting future path metrics from historical observations, and *reinforcement learning* for optimising scheduling decisions under uncertainty.

Sequential prediction models (Section 2.2.1) use time-series of path metrics-congestion window values, round-trip times, and throughput measurements-to forecast future con-

ditions. These predictions augment the state representation for reinforcement learning agents (Section 2.2.2), which learn policies mapping states to scheduling actions that maximise long-term performance objectives. Neural architecture search (Section 2.2.3) automates the discovery of model configurations suited to specific deployment scenarios.

2.2.1 Sequential Prediction for Network Forecasting

Forecasting future path conditions requires models capable of capturing temporal dependencies in network metric time-series. Two neural network architectures dominate sequential prediction [46, 47]: LSTM networks, which process sequences through recurrent connections with gating mechanisms, and Transformer models, which employ attention mechanisms to directly model relationships between arbitrary sequence positions. This section presents both architectures, establishing the theoretical foundation for the prediction components of learning-based schedulers.

2.2.1.1 Recurrent Neural Networks and the Vanishing Gradient Problem

RNNs process sequential data by maintaining hidden state vectors that evolve as each element is consumed. Given input sequence $(x_1, x_2, \ldots, x_T)$, a basic RNN computes hidden states via:

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b) \tag{2.8}$$

where W_h and W_x are weight matrices and b is a bias vector. This recurrence enables information from early sequence elements to influence processing of later elements, theoretically capturing arbitrary-length dependencies.

However, training RNNs via Backpropagation Through Time (BPTT) [48] encounters the vanishing gradient problem, analysed by Bengio *et al.* [49]. Gradients flowing backward through many timesteps undergo repeated multiplication by the recurrent weight matrix W_h. When $\|W_h\| < 1$, gradients shrink exponentially; when $\|W_h\| > 1$, they explode. This instability prevents learning of long-range dependencies-precisely the capability required for predicting path conditions that depend on patterns spanning tens or hundreds of timesteps.

Figure 2.3 illustrates the LSTM cell architecture that addresses these limitations. The cell state C_t flows horizontally through the upper pathway, modified only by element-wise operations that preserve gradient magnitude during backpropagation. Input x_t and previous hidden state h_{t-1} concatenate and feed into four parallel transformations: three sigmoid activations (σ) producing gate values in $(0, 1)$, and one hyperbolic tangent producing candidate cell content in $(-1, 1)$. The forget gate output f_t multiplies element-wise with the previous cell state C_{t-1}, selectively erasing information. The input gate output i_t multiplies with the candidate $\tilde{C}_t$, controlling which new information enters the cell state. These products sum to form the updated cell state C_t, which passes through hyperbolic tangent and multiplies with the output gate o_t to produce the new hidden state h_t. The additive cell state update creates an unimpeded gradient pathway across arbitrarily long sequences.

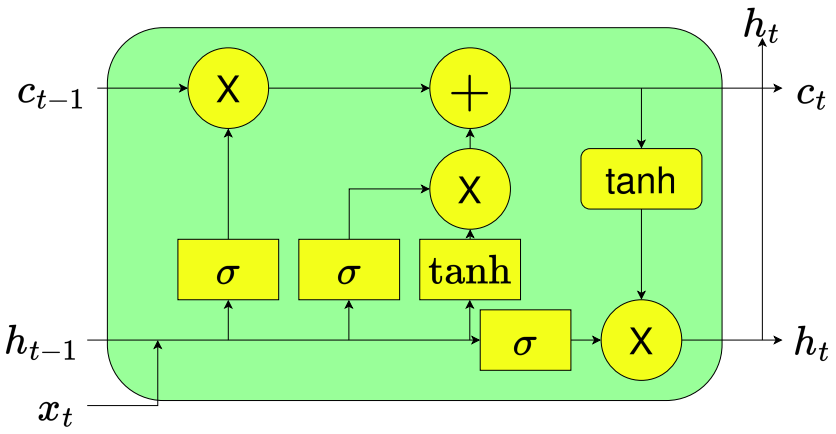

Figure 2.3: LSTM cell architecture. The cell state C_t flows through the upper horizontal pathway, modified by element-wise multiplication ($\times$) and addition ($+$) rather than matrix transformations. Three gates regulate information flow: the forget gate (leftmost σ) controls retention of previous cell state C_{t-1}, the input gate (second σ) controls incorporation of candidate values produced by tanh, and the output gate (rightmost σ) controls exposure of cell state to hidden state h_t. Concatenated inputs $[h_{t-1}, x_t]$ feed all gate computations.

2.2.1.2 LSTM Networks

LSTM networks, introduced by Hochreiter and Schmidhuber [50], address the vanishing gradient problem through gating mechanisms that regulate information flow. Each LSTM

cell maintains a cell state C_t serving as a memory pathway with controlled read/write access, plus a hidden state h_t exposed to subsequent layers.

Three gates govern information flow: the *forget gate* determines which information to discard from the cell state; the *input gate* controls which new information to store; and the *output gate* controls which cell state information to expose as hidden state. Each gate applies a sigmoid activation to a linear combination of the previous hidden state and current input, producing values in $(0, 1)$ that modulate information flow through element-wise multiplication.

The critical architectural feature is the cell state update pathway: information flows through the cell state via element-wise operations rather than matrix multiplication, stabilising gradient flow across long sequences. Gers *et al.* [51] demonstrate that this architecture effectively captures dependencies spanning hundreds of timesteps.

For network metric forecasting, LSTM networks can learn patterns such as periodic congestion window oscillations, gradual RTT increases preceding handovers, and correlations between path metrics. The recurrent structure naturally handles variable-length input sequences and produces predictions conditioned on arbitrary history lengths.

2.2.1.3 Activation Functions

The sigmoid and hyperbolic tangent functions serve specific roles within LSTM cells: sigmoid outputs in $(0, 1)$ enable multiplicative gating, whilst tanh outputs in $(-1, 1)$ provide zero-centred candidate values. Alternative activation functions, surveyed by Dubey *et al.* [52] address limitations in other network components:

ReLU (Rectified Linear Unit), introduced by Nair and Hinton [53], computes $f(x) = \max(0, x)$. ReLU provides computational efficiency and mitigates vanishing gradients in deep feedforward networks, though neurons receiving consistently negative inputs become permanently inactive (the "dying ReLU" problem).

Leaky ReLU [52] addresses dying ReLU by allowing small negative gradients:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise} \end{cases} \tag{2.9}$$

with typically $\alpha = 0.01$, maintaining gradient flow for negative inputs.

GELU (Gaussian Error Linear Unit), computes $f(x) = x \cdot \Phi(x)$ where Φ is the cumulative distribution function of the standard normal distribution [52]. GELU provides smooth non-linearity and has become standard in Transformer architectures.

2.2.1.4 Regularisation and Training

Dropout, introduced by Srivastava *et al.* [54], prevents overfitting by randomly deactivating neurons during training with probability $p \in [0.2, 0.5]$. This forces the network to learn redundant representations robust to missing features. For RNNs variational dropout applies the same dropout mask across all timesteps to maintain temporal coherence.

LSTM networks are trained using BPTT [48], unrolling the recurrent network through time and applying standard backpropagation. For regression tasks such as predicting future CWND values, **Mean Squared Error** $L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$ penalises large errors quadratically [55]. **Mean Absolute Error** $L = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$ penalises errors linearly, providing robustness to outliers.

Optimiser selection significantly impacts convergence. The Adam optimiser, introduced by Kingma and Ba [56], combines momentum and adaptive learning rates through first and second moment estimates, with default hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ working well across diverse tasks. Shazeer and Stern [57] introduce Adafactor, a memory-efficient variant designed for large models that factorises the second moment estimate to reduce memory requirements from $O(nm)$ to $O(n + m)$ for parameter matrices.

2.2.1.5 Sequence-to-Sequence Architectures

For tasks requiring sequence output-such as predicting multiple future timesteps of path metrics-time-distributed dense layers apply the same fully-connected transformation inde-

pendently to each timestep. Sutskever *et al.* [58] demonstrate this encoder-decoder architecture for sequence-to-sequence learning. Cho *et al.* [59] introduce the related encoder-decoder framework using gated recurrent units. Lim and Zohren [60] survey deep learning approaches for time-series forecasting, demonstrating effectiveness of these structures for multi-step prediction tasks.

2.2.1.6 Data Preprocessing for Time-Series

Effective training requires careful preprocessing, as detailed by Lim and Zohren [60]. **Normalisation** scales features to comparable ranges: MinMax scaling to $[0, 1]$ via $x' = (x - x_{\min})/(x_{\max} - x_{\min})$, or standardisation to zero mean and unit variance via $x' = (x - \mu)/\sigma$. Log-transformation $x' = \log(x + c)$ stabilises variance for heavy-tailed distributions, as analysed by Box and Cox [61]. **Temporal aggregation** creates fixed-length input sequences from continuous streams. Sliding windows with stride s extract overlapping subsequences, whilst moving averages $\bar{x}_t = \frac{1}{w}\sum_{i=0}^{w-1} x_{t-i}$ smooth high-frequency noise. Moving standard deviation quantifies local volatility. Box *et al.* [62] provide comprehensive treatment of time-series preprocessing techniques. **Categorical encoding** converts continuous predictions to discrete outputs via binning or quantile-based discretisation when classification formulations are appropriate, as discussed in machine learning literature.

2.2.1.7 Transformer Architecture

Whilst LSTM networks effectively capture sequential dependencies, their recurrent structure processes tokens sequentially, limiting parallelisation and creating information bottlenecks for very long sequences. The Transformer architecture, introduced by Vaswani *et al.* [63], addresses these limitations through attention mechanisms that directly model relationships between arbitrary sequence positions.

Self-Attention Mechanism. The fundamental component is Scaled Dot-Product Attention, which computes weighted combinations of value vectors based on query-key com-

patibility:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{2.10}$$

where $\mathbf{Q}$ (queries), $\mathbf{K}$ (keys), and $\mathbf{V}$ (values) are linear projections of input representations, and d_k is the key dimension. The scaling factor $1/\sqrt{d_k}$ prevents large dot products from pushing softmax into regions with vanishing gradients [63].

For self-attention, queries, keys, and values all derive from the same input sequence, enabling each position to attend to all other positions. This mechanism captures dependencies regardless of distance-a position can directly attend to any other position without information passing through intermediate states as required in recurrent architectures.

Multi-Head Attention. Multi-Head Attention extends self-attention by computing attention in parallel across multiple representation subspaces:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O \tag{2.11}$$

$$\text{where head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \tag{2.12}$$

Voita *et al.* [64] demonstrate that individual heads can specialise in capturing different types of relationships-some heads may focus on local patterns whilst others capture long-range dependencies.

Positional Encoding. Since attention mechanisms are permutation-invariant, positional encodings inject sequence order information. The original Transformer uses sinusoidal functions:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{2.13}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{2.14}$$

These encodings enable the model to attend by relative positions, as the sinusoidal representation allows linear combinations to express offsets [63].

Architecture Components. A complete Transformer encoder layer comprises multi-head self-attention followed by a position-wise feed-forward network, with residual connections and layer normalisation [65] around each sub-layer:

$$\mathbf{z} = \text{LayerNorm}(\mathbf{x} + \text{MultiHead}(\mathbf{x}, \mathbf{x}, \mathbf{x})) \tag{2.15}$$

$$\mathbf{y} = \text{LayerNorm}(\mathbf{z} + \text{FFN}(\mathbf{z})) \tag{2.16}$$

Multiple such layers stack to form deep networks capable of hierarchical feature extraction. Figure 2.4 presents the complete Transformer encoder-decoder architecture. The encoder (left stack) processes input sequences through N identical layers, each containing multi-head self-attention followed by a position-wise feed-forward network. Residual connections around each sub-layer, combined with layer normalisation, facilitate gradient flow through deep stacks. Input embeddings sum with positional encodings before entering the encoder, injecting sequence order information into the otherwise permutation-invariant attention computation.

The decoder (right stack) mirrors the encoder structure with two key modifications. First, masked multi-head attention in the lower sub-layer prevents positions from attending to subsequent positions, preserving the autoregressive property necessary for sequence generation. Second, an additional multi-head attention sub-layer attends to encoder outputs, enabling the decoder to incorporate source sequence information when generating each target position. The final linear projection and softmax produce output probabilities over the target vocabulary.

For network metric forecasting, encoder-only configurations suffice: historical path metrics form the input sequence, with the encoder learning attention patterns that weight relevant historical observations when producing representations used for prediction. The self-attention mechanism automatically discovers which past timesteps-regardless of distance-most inform predictions of future conditions, without the information bottlenecks inherent in recurrent state compression.

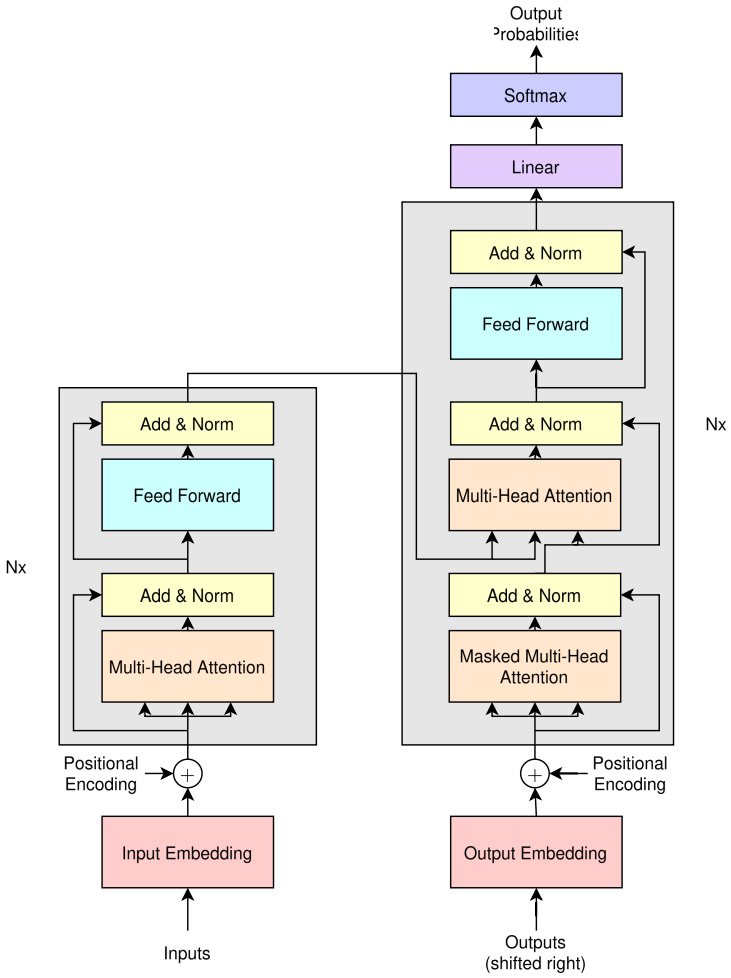

Figure 2.4: Transformer encoder-decoder architecture. The encoder (left) comprises N stacked layers of multi-head self-attention and feed-forward networks. The decoder (right) adds masked self-attention to prevent attending to future positions and cross-attention to encoder outputs. Positional encodings sum with input embeddings to provide sequence order information, enabling direct modelling of arbitrary-range dependencies without recurrent processing.

Transformers versus LSTMs for Network Metric Forecasting. For network metric forecasting, Transformers offer several advantages over LSTM networks. First, parallel processing of all sequence positions enables efficient training on modern hardware, whereas LSTM sequential processing creates bottlenecks. Second, direct attention between arbitrary positions avoids information degradation over long sequences-critical when predicting path conditions that depend on patterns spanning extended histories. Third, multi-head attention can simultaneously capture multiple types of temporal relationships (short-term fluctuations, periodic patterns, long-term trends) without architectural modifications.

However, Transformers require more training data to learn effective attention patterns, and their quadratic complexity in sequence length ($O(L^2 \cdot d)$ for sequence length L and dimension d) can become prohibitive for very long sequences. For the multipath scheduling domain, where prediction horizons span seconds to tens of seconds with sub-second sampling, sequence lengths remain tractable and Transformer advantages predominate.

The proposed DARA framework employs Transformer-based prediction for CWND forecasting, leveraging attention mechanisms to capture both temporal evolution within each path and cross-path interactions that influence scheduling decisions. This architectural choice reflects the need to model complex, non-linear relationships in volatile wireless channel conditions where simple recurrent models may fail to capture relevant patterns.

2.2.2 Deep Reinforcement Learning for Decision-Making

Whilst sequential prediction models forecast future path conditions, scheduling decisions require selecting actions that optimise long-term objectives under uncertainty. Reinforcement Learning (RL) [66] provides a principled framework for learning such decision policies through environmental interaction.

2.2.2.1 Markov Decision Process Formulation

RL formalises sequential decision-making as a Markov Decision Process (MDP), characterised by the tuple (S, A, P, R, γ), where S represents the state space, A the action space,

$P : S \times A \times S \to [0, 1]$ the state transition probability, $R : S \times A \times S \to \mathbb{R}$ the reward function, and $\gamma \in (0, 1]$ the discount factor. Bellman [67] established that the optimal policy π^* maximises expected discounted return:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2.17}$$

For multipath scheduling, the state s_t comprises path metrics (congestion windows, RTT measurements, loss rates) and predictions thereof; actions a_t represent scheduling decisions (path selection, rate allocation); and rewards r_t encode performance objectives (throughput, latency, fairness). The discount factor γ balances immediate rewards against long-term consequences-higher values encourage planning over extended horizons. Figure 2.2.2.1 illustrates the fundamental interaction loop underlying reinforcement learning. At each discrete timestep t, the agent observes state s_t from the environment and selects action a_t according to its policy π_θ, parameterised by neural network weights θ. The environment transitions to a new state s_{t+1} according to dynamics $P(s_{t+1}|s_t, a_t)$ and emits reward $r_t = R(s_t, a_t, s_{t+1})$. The agent updates its policy parameters to maximise expected cumulative reward, closing the learning loop. This cycle repeats indefinitely, with the policy gradually improving through experience.

The deep neural network within the agent maps high-dimensional state representations to action selections, enabling generalisation across similar states without explicit enumeration. For multipath scheduling, the state s_t encodes current and predicted path metrics, the action a_t specifies scheduling decisions such as path selection or rate allocation fractions, and the reward r_t quantifies immediate performance according to application objectives. The policy network learns to associate state patterns with actions that yield high long-term returns, discovering scheduling strategies that may not be apparent from heuristic design.

2.2.2.2 Q-Learning

Q-learning, introduced by Watkins and Dayan [68], learns the optimal action-value function $Q^*(s, a)$-the expected return from taking action a in state s and thereafter following

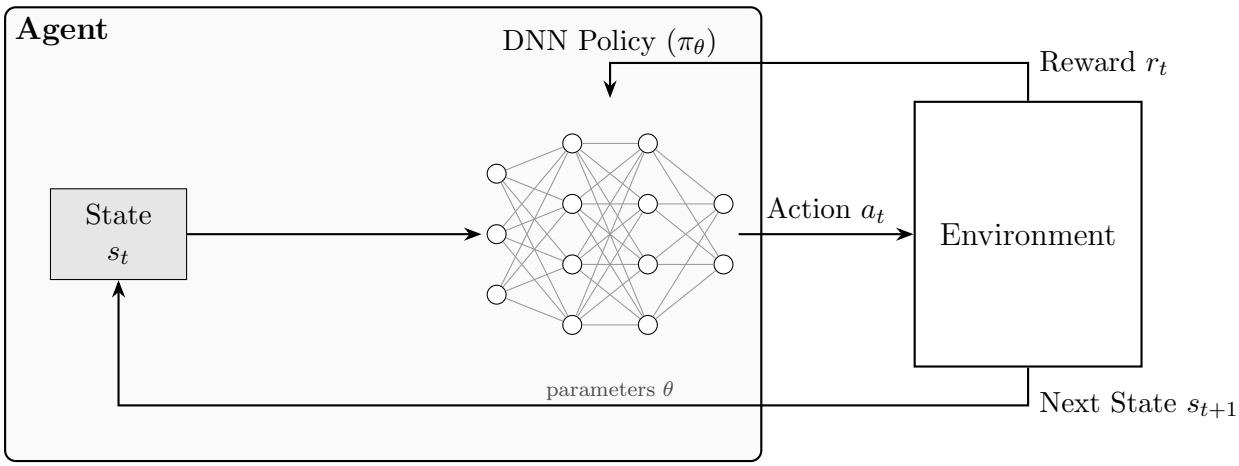

Figure 2.5: Deep Reinforcement Learning Agent

the optimal policy. The algorithm employs temporal-difference updates:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a' \in A} Q(s_{t+1}, a') - Q(s_t, a_t) \right] \tag{2.18}$$

where $\alpha \in (0, 1]$ is the learning rate controlling how much new information overrides old estimates. The term $r_t + \gamma \max_{a'} Q(s_{t+1}, a')$ represents the temporal difference target, combining immediate reward with discounted maximum future value.

Q-learning is *off-policy*: it learns the optimal policy whilst following a different behaviour policy, typically ϵ-greedy which selects random actions with probability ϵ and greedy actions otherwise [66]. The optimal policy derives from greedy action selection: $\pi^*(s) = \arg\max_{a \in A} Q^*(s, a)$.

2.2.2.3 Exploration-Exploitation Trade-off

The multi-armed bandit problem [69] illustrates the fundamental exploration-exploitation trade-off in sequential decision-making. An agent repeatedly chooses amongst K actions (arms), receiving stochastic rewards, but must balance exploration (trying different actions to learn their reward distributions) against exploitation (selecting the currently best-known action).

The Upper Confidence Bound (UCB) algorithm [69], addresses this trade-off by selecting

actions based on optimistic estimates:

$$a_t = \arg\max_{a \in A} \left[\hat{\mu}_a + c\sqrt{\frac{\ln t}{N_a}} \right] \tag{2.19}$$

where $\hat{\mu}_a$ is the empirical mean reward for action a, N_a is the number of times action a has been selected, t is the total number of trials, and c controls exploration magnitude. The confidence bound term is large for under-explored actions, encouraging exploration, whilst the mean reward term favours high-reward actions.

Contextual bandits [69] extend this framework by incorporating state information (context) x_t that influences reward distributions. The LinUCB algorithm assumes linear reward models, selecting actions via:

$$a_t = \arg\max_{a \in A} \left[\mathbf{x}_t^\top \hat{\boldsymbol{\theta}}_a + \alpha\sqrt{\mathbf{x}_t^\top \mathbf{A}_a^{-1} \mathbf{x}_t} \right] \tag{2.20}$$

where $\hat{\boldsymbol{\theta}}_a$ is estimated via ridge regression on observed rewards and $\mathbf{A}_a$ captures feature correlations. Contextual bandits are particularly effective for problems requiring rapid adaptation without long-term planning.

2.2.2.4 Deep Q-Networks

When state spaces are large or continuous, storing Q-values in a table becomes impractical. Mnih *et al.* [70] address this limitation with DQN, which uses a neural network $Q(s, a; \theta)$ to estimate Q-values instead of storing them explicitly. The network, parameterised by weights θ, takes a state s as input and outputs Q-values for each possible action a, enabling generalisation across similar states.

Two techniques stabilise the training process, as illustrated in Figure 2.6. The first, *experience replay*, was introduced by Lin [71]. Rather than learning from experiences immediately as they occur, the agent stores each experience tuple (s, a, r, s')—comprising the state, action taken, reward received, and resulting next state—in a replay buffer $\mathcal{D}$. During training, random mini-batches are sampled from this buffer, breaking temporal correlations between consecutive experiences that would otherwise cause training insta-

bility.

The second technique employs a separate *target network* $Q(s, a; \theta^-)$ with parameters θ^- to provide stable learning targets. The challenge is that Q-learning updates Q-values towards a target that itself depends on Q-values—if the same network provides both current estimates and targets, the targets shift with every update, creating instability. The solution maintains two copies of the network: a policy network $Q(s, a; \theta)$ that is updated continuously, and a target network $Q(s, a; \theta^-)$ that is updated only periodically $(\theta^- \leftarrow \theta)$.

The training process computes a target Q-value for each sampled experience:

$$y = r + \gamma \max_{a'} Q(s', a'; \theta^-) \tag{2.21}$$

where r is the reward received, γ is the discount factor, and the maximum is taken over Q-values of the next state s' estimated by the target network. The policy network parameters θ are then updated via gradient descent to minimise the Temporal Difference (TD) loss $L(\theta)$—the squared difference between the network's current estimate $Q(s, a; \theta)$ and the target y.

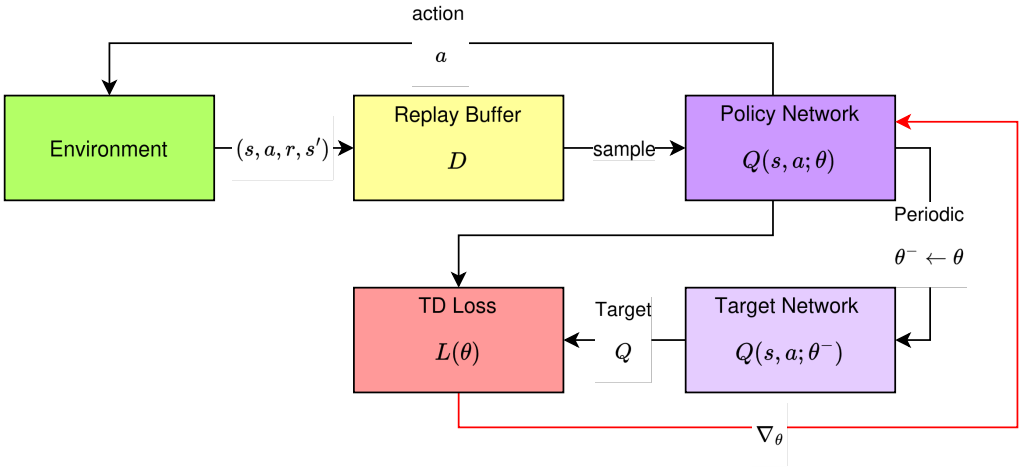

Figure 2.6: Deep Q-Network training architecture. Experience tuples (s, a, r, s') from environment interaction populate the replay buffer $\mathcal{D}$. Mini-batches sampled from the buffer train the policy network $Q(s, a; \theta)$, with target Q-values derived from the separate target network $Q(s, a; \theta^-)$. The TD loss $L(\theta)$ drives gradient updates ∇_θ, whilst periodic synchronisation $(\theta^- \leftarrow \theta)$ maintains stable targets.

This separation between data collection and learning enables *off-policy learning*: the

agent learns from historical experiences regardless of the policy that generated them, substantially improving sample efficiency.

The exploration rate ϵ (the probability of selecting a random action rather than the current best) decays over time—from high values (e.g., 0.9) encouraging early exploration to low values (e.g., 0.05) favouring exploitation of learned knowledge [66].

2.2.2.5 Training Considerations

Chen *et al.* [72] identify critical factors affecting Deep Reinforcement Learning (DRL) performance and reproducibility: **Network Architecture:** Value networks typically employ 2–3 fully connected layers with 256–400 units and ReLU activations [70]. Deeper networks increase representational capacity but require more training data and careful regularisation.

Hyperparameter Sensitivity: Key hyperparameters include learning rate $\alpha \in [10^{-5}, 10^{-3}]$, discount factor $\gamma \in [0.95, 0.99]$, and batch size $B \in [32, 256]$. Small changes can dramatically affect performance, necessitating systematic tuning.

Reward Design: Dense rewards guide learning more effectively than sparse rewards. Ng *et al.* [73] analyse reward shaping, demonstrating that poorly designed rewards can induce unintended behaviours. Reward normalisation $r'_t = (r_t - \mu_r)/\sigma_r$ stabilises training across different scales.

State Normalisation: Input states are normalised using running statistics $s'_t = (s_t - \mu_s)/(\sigma_s + \epsilon)$ to ensure features are on similar scales. Ioffe and Szegedy [74] demonstrate the importance of normalisation for stable training.

Discrete Action Spaces: When actions are finite and categorical (e.g., selecting amongst congestion window fractions $\{20\%, 40\%, 60\%, 80\%, 100\%\}$), the policy network outputs Q-values for each discrete action. Action space size impacts learning complexity-larger spaces provide finer control but increase sample requirements. Dulac-Arnold *et al.* [75] identify scalability challenges in large discrete action spaces.

2.2.2.6 Challenges in Deep Reinforcement Learning

Despite advances, several challenges remain relevant to network scheduling applications [72, 75]:

Sample Efficiency: DRL algorithms typically require millions of environmental interactions. Moerland *et al.* [76] survey model-based approaches that improve efficiency, though these introduce additional complexity.

Partial Observability: Many real-world problems violate the Markov assumption-current observations may not fully determine future dynamics. This is addressed through recurrent architectures that maintain history in hidden states [76].

Reward Specification: Reward function design must capture desired behaviours without unintended side effects [75], a particular concern when optimising multipath scheduling for competing objectives.

2.2.3 Neural Architecture Search

NAS [77] automates neural network design, systematically exploring hyperparameter spaces to discover configurations optimised for specific tasks. Unlike manual tuning requiring domain expertise and extensive experimentation, NAS employs search algorithms to evaluate candidate architectures according to predefined objectives.

Zoph and Le [78] introduce reinforcement learning-based NAS, where a controller network proposes architectures and receives accuracy-based rewards. Real *et al.* [79] employ evolutionary algorithms maintaining populations of architectures with mutation and crossover operations. Liu *et al.* [80] enable gradient-based differentiable architecture search, dramatically reducing computational requirements.

For hyperparameter optimisation, Bayesian optimisation frameworks such as Optuna [81] employ sequential model-based optimisation. The Tree-structured Parzen Estimator [81] constructs probabilistic models of hyperparameter-performance relationships, modelling high-performing configurations with density $p(x|y = 1)$ and low-performing configurations with $p(x|y = 0)$. The acquisition function maximises expected improvement $\text{EI}(x) \propto p(x|y = 1)/p(x|y = 0)$, focusing search on promising regions whilst maintaining

exploration.

Pruning strategies such as MedianPruner terminate unpromising trials early by comparing intermediate metrics against median performance of completed trials, reducing computational waste [81]. For multi-objective problems balancing competing metrics (e.g., throughput versus latency), weighted scalarisation combines objectives into a single fitness function.

NAS is particularly valuable when the hyperparameter space is high-dimensional with complex interactions, evaluation cost permits exploration of hundreds of configurations, and domain-specific architectures require task-tailored designs [77]. For multipath scheduling, automated tuning of prediction model architectures and reinforcement learning hyperparameters can adapt frameworks to specific network conditions and performance requirements.

2.3 Chapter Summary

This chapter has established the theoretical foundations for the scheduling frameworks presented in subsequent chapters. The networking context (Section 2.1) detailed the multipath transport protocols,MP-TCP, MP-QUIC, and MP-DCCP,that provide the architectural substrate for multipath scheduling, alongside the congestion control mechanisms and scheduling design space that constrain and motivate the proposed approach. The AI foundations (Section 2.2) introduced the two complementary capabilities underpinning DARA: LSTM and Transformer architectures for sequential prediction of path metrics, and deep reinforcement learning with DQN for optimising scheduling decisions under uncertainty. Together, these foundations define the design space within which learning-based schedulers must operate, establishing the baseline concepts and notation used throughout the remainder of this thesis. The following chapter applies these foundations to critically analyse existing multipath scheduling approaches and identify the research gaps that DARA addresses.

Chapter 3

Related Work

This chapter critically analyses the application of the theoretical foundations established in Chapter 2 to multipath scheduling, demonstrating why predictive approaches are necessary for volatile heterogeneous networks, and defines the baseline and state-of-the-art schedulers against which DARA is evaluated.

3.1 Learning Based Multipath Scheduling

The multipath transport fundamentals (Section 2.1) and AI foundations (Section 2.2) established in Chapter 2 converge in learning-based multipath scheduling.
Figure 3.1 presents the classification taxonomy adopted in this section, organising existing learning-based multipath scheduling approaches into five categories: AI for congestion control, redundancy-based solutions, cross-layer solutions, prediction-based solutions (further subdivided into instantaneous path selection and forward-looking prediction), and multi-objective solutions. Each category addresses distinct aspects of the scheduling challenge, from optimising congestion response to balancing competing performance objectives.

3.1.1 Motivation for Learning-Based Approaches

Dong *et al.* [11] conclude in a comprehensive survey that BLEST generally outperforms other MP-TCP schedulers, although in non-shared bottleneck conditions most schedulers

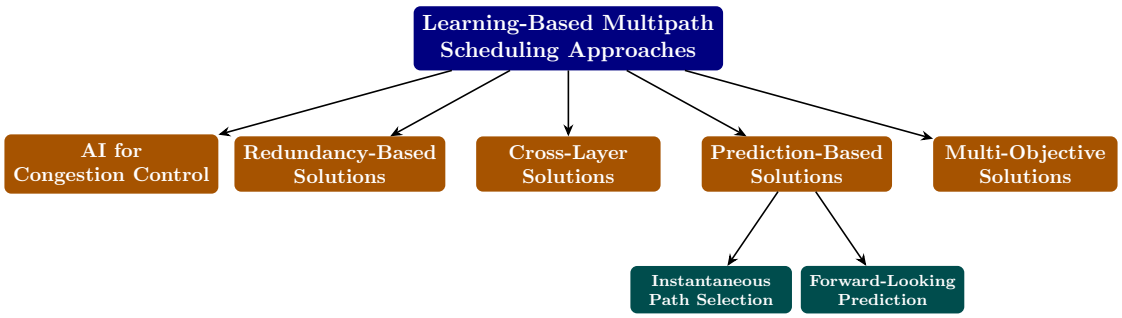

Figure 3.1: Classification of learning-based multipath scheduling approaches reviewed in this section. The taxonomy organises existing work into five principal categories, with prediction-based approaches further subdivided by temporal granularity.

converge to similar performance. This finding highlights that path-estimation schedulers are inherently suboptimal in volatile conditions, where frequent path fluctuations aggravate bottlenecks, increase packet reordering, and exacerbate HoL blocking.

Diakhate *et al.* [82] compare rule-based schedulers (Minimum Round-Trip Time (minRTT), ECF, BLEST) with learning-based schedulers, concluding that adaptivity is essential in mobility scenarios, with Machine Learning (ML)-based schedulers achieving superior performance. Three factors drive this conclusion. First, inherent path asymmetry introduces packet scrambling, OFO delivery, and HoL blocking [4]. Second, adverse interactions between multipath tunnelling and nested congestion control mechanisms limit path aggregation [25]. Third, layered protocol interactions can negate throughput gains entirely [12].

The minRTT scheduler [26] sends packets over the path with the lowest measured RTT, prioritising latency reduction. Whilst effective when one path consistently outperforms others, minRTT underutilises secondary paths even when they offer substantial capacity. The ECF scheduler, proposed by Lim *et al.* [83], minimises idle time on the fast path by estimating completion times. Other delay-aware schedulers similarly extend these strategies [31]. These schedulers share common limitations: reactive operation responding only after degradation manifests, heuristic estimation failing to capture complex wireless dynamics, and single-objective optimisation ignoring competing requirements. The resulting body of learning-based work can be broadly categorised into: (i) redundancy-based solutions, (ii) cross-layer solutions, (iii) prediction-based solutions, and (iv) multi-objective solutions.

3.1.2 Artificial Intelligence for Congestion Control

Before examining scheduling specifically, the application of AI to congestion control provides relevant context. The dynamic nature of heterogeneous mobile connections affects BBR and other rule-based CCAs, motivating adaptive approaches.

Li *et al.* [17] introduce intelligent CCA for heterogeneous MP-TCP, identifying bufferbloat (caused by large OFO packet queues) and suboptimal bandwidth usage as key performance problems. Their approach enables dynamic CWND adjustment through Q-learning trained asynchronously. Ha *et al.* [84] develop this concept for changing link conditions, whilst Luo *et al.* [85] and Roselló [86] employ DQN frameworks for path selection based on CC feedback. Lee and Yoo [87] address delay and throughput minimisation using DQN.

Zhuang *et al.* [88] introduce ML-OL, a multipath CCA combining online learning with Multi-Armed Bandit (MAB) solutions to predict network conditions, providing improved inference with lower computational complexity than offline-trained models.

Pokhrel and Walid [89] integrate CC directly with scheduling, training an agent via deep Q-learning with LSTM integration to learn optimal mechanisms through continuous dynamic behaviour learning. This approach minimises aggregate delay using application layer contexts, improving MP-TCP throughput and delay characteristics at the cost of substantial training overhead.

Collectively, these approaches demonstrate that RL-based congestion control can outperform rule-based mechanisms in heterogeneous multipath settings, particularly when path conditions fluctuate rapidly. However, several common limitations constrain their applicability. First, the majority employ single-objective reward functions optimising either throughput or delay in isolation, neglecting the multi-dimensional trade-offs inherent in real traffic [17, 85, 87]. Second, most solutions operate at the congestion control layer without considering scheduling interactions; Pokhrel and Walid [89] represent a notable exception by integrating scheduling with CC, but their LSTM-based architecture incurs substantial training overhead that limits practical deployment. Third, state representations typically aggregate path-level statistics (RTT, CWND, loss rate) without capturing temporal dependencies or cross-path correlations, limiting the agent's ability to antici-

pate jointly evolving path dynamics. Zhuang *et al.* [88] partially address the computational concern through MAB-based online learning, but bandit formulations sacrifice the temporal modelling capability that sequential RL provides. These limitations motivate the separation of prediction and decision-making adopted by DARA, where a dedicated Transformer module captures temporal and cross-path dependencies before a DQN agent operates on the resulting state representation.

3.1.3 Redundancy-Based Solutions

Redundancy-based scheduling maintains QoS by predicting link quality and, based on outage probability, using secondary paths for redundant packet transmission. Xing *et al.* [90] propose a redundancy-based scheduler for moving vehicles, employing a multi-armed bandit framework to find optimal redundancy ratios. The approach prevents excessive redundancy common in static schedulers, achieving 12%–20% throughput increases over state-of-the-art redundant schedulers.

Han *et al.* [91] propose a scheduler for seamless handoff using a DRL agent to optimise goodput, bandwidth utilisation, and OFO packets, achieving 150% goodput improvement during handoffs. Chen *et al.* [92] address ultra-reliable low-latency communication requirements by combining forward error correction with concurrent multipath transmission under a DRL framework. The agent jointly optimises the coding rate and per-path packet distribution to meet stringent reliability targets (99.999%) with sub-millisecond latency. By learning adaptive redundancy levels rather than applying fixed coding rates, the approach avoids the resource waste inherent in static redundancy configurations and demonstrates improved reliability compared to fixed-redundancy baselines in simulation.

Despite these advances, redundancy-based approaches share inherent limitations that constrain their deployment at scale. All three methods consume additional network bandwidth for duplicate or coded packets, imposing costs that scale with user density and become unsustainable in high-load scenarios [90, 91]. Furthermore, redundancy decisions are typically driven by single-path quality metrics without considering aggregate multipath capacity, potentially over-redundantifying traffic when alternative paths offer

sufficient reliability on their own. The overhead of encoding and decoding in approaches such as Chen *et al.* [92] introduces computational latency that may conflict with the low-latency objectives these methods target. These trade-offs motivate the exploration of non-redundant scheduling strategies that achieve reliability through intelligent path selection and rate adaptation rather than bandwidth multiplication.

3.1.4 Cross-Layer Solutions

Cross-layer solutions exchange control information between OSI layers, simplifying scheduling in time-varying networks through predefined rules. Two principal approaches exist: establishing feedback loops, or improving metric forecasting.

Lv *et al.* [93] propose server transport-layer scheduling coordinated with client application-level Adaptive Bitrate (ABR) streaming. The server schedules chunk-level packet assignment ratios, informing the client to enable throughput prediction and bitrate adjustment. The server correspondingly adjusts transmission to meet expected delivery times, improving average QoE by 23.5%–65.7% in trace-based emulation compared to other cross-layer and single-path approaches.

Yang *et al.* [94] introduce Mobility-Aware Multipath Scheduler (MAMS), using forecasting to predict congestion and buffer utilisation in successive time slots based on historical and current end-to-end path conditions, combined with link-transport layer data on wireless uplink/downlink conditions. The scheduler pre-allocates packets to match CWND and buffer loads, achieving 40% reordering delay reduction relative to earliest-deadline-first and RR schedulers under high mobility.

Park and Das [95] propose a cross-layer scheduling mechanism for MP-QUIC that leverages application-layer video tile importance to inform transport-layer path assignment decisions. For 360-degree video streaming, the scheduler prioritises high-importance viewport tiles on lower-latency paths whilst distributing less critical background tiles across available paths, exploiting MP-QUIC's stream multiplexing for fine-grained control. Evaluated with real-world video traces over emulated multipath networks, the approach demonstrates improved video quality compared to standard MP-QUIC schedulers,

illustrating the potential of application-aware cross-layer design.

Collectively, these three solutions demonstrate significant QoS improvements when path parameters or application requirements are known in real-time. Lv *et al.* [93] and Yang *et al.* [94] show that cross-layer feedback enables scheduling decisions that approach the performance of an oracle with perfect state information. Park and Das [95] further demonstrate that application-level semantics can drive more discriminating path assignment than transport-layer metrics alone. However, universal adoption of cross-layer approaches remains unrealistic for several reasons. Deploying modified protocols that exchange information across non-adjacent OSI layers violates standard end-to-end design principles and requires coordinated changes at both endpoints and potentially intermediate nodes. The application-specific nature of metrics such as video tile importance [95] or ABR chunk deadlines [93] limits generalisability: each new application type necessitates bespoke cross-layer interfaces. These solutions nonetheless exemplify performance upper bounds for predictive schedulers, as real-time control information is inherently more accurate than purely transport-layer predictions. The design implication for DARA is that transport-layer state representations should be rich enough to approximate the information advantage that cross-layer approaches enjoy, without requiring actual protocol modifications.

3.1.5 Prediction-Based Solutions

Prediction-based scheduling estimates path conditions without direct receiver-state feedback, typically using historical and current network data to make sequential decisions whilst maximising long-term reward.

3.1.5.1 Instantaneous Path Selection

The simplest approach performs per-packet path selection through utility maximisation. Luo *et al.* [85] propose a scheduler calculating path value via heuristic SRTT-bandwidth combinations, achieving 20%–41% throughput increases by reducing OFO packets and buffer blocking. This approach is computationally efficient but the heuristic nature pre-

vents adaptation across diverse scenarios.

Nguyen *et al.* [96] propose Q-learning-based path selection optimising transmission time using RTT, loss, CWND, in-flight packets, and Sending Window (SWND). The RL agent learns optimal path choices, achieving adaptable solutions for vehicles undergoing various mobility degrees. Roselló [86] extends this using DRL to handle higher-dimensional states. These path selection methods rely on greedy decision-making from current observations, defaulting to slow paths during transient congestion-performing poorly in rapidly changing environments.

3.1.5.2 Forward-Looking Prediction

Adding predictive elements that estimate future path conditions addresses limitations of instantaneous selection. Zeng *et al.* [97] assign data across paths based on estimated transfer completion times, using forward-looking approaches to wait for fast-path availability, achieving 29.6% completion time decreases compared to other multipath schedulers.

Heuristic estimation fails under high mobility. Wu *et al.* [14] address this with the Peekaboo scheduler, employing multi-armed bandit schemes to estimate whether to use slow paths or wait for fast paths. Peekaboo uses offline models for coarse-grained and online models for fine-grained scheduling using transport-layer parameters (CWND, SWND, packets in-flight, RTT). It decreases download times by up to 23.6% in moving user tests by mitigating OFO delivery and HoL blocking. Wu *et al.* [98] further develop this in the FALCON scheduler, extending the offline/online model combination.

3.1.6 Multi-Objective Solutions

Path selection methods are simple techniques that do not readily extend to either throughput maximisation or congestion-aware routing. Multi-objective optimisation balances trade-offs where single-objective schedulers fail, adapting policies to dynamic conditions. Zhang *et al.* [18] define a multi-objective optimisation problem where the reward function balances delay, throughput, and loss, solved using a DRL agent. This approach reduces download times by 8%–13% and increases goodput by 10% compared to minRTT, RR,

and BLEST.

Han *et al.* [99] propose a scheduler with multi-objective reward considering OFO queue size, RTT, and CWND. The scheduler operates multi-agent DRL, where each path uses a neural network to adjust CWND to minimise bufferbloat. The multi-agent structure enables finer-grained control, adapting to diverse receive buffer sizes and video streaming requirements. However, high overhead negates benefits in low-mobility environments, and the architecture requires protocol modifications unrealistic for Internet deployment.

Dong *et al.* [100] propose MPTCP-RL, optimising energy efficiency, throughput, delay, and jitter in heterogeneous wireless networks. The scheduler employs Proximal Policy Optimisation (PPO) DRL and reduces path management overhead through an asynchronous learning framework separating offline training from online decision-making. A path selection mechanism avoids degradation from simultaneous asymmetric path usage by selecting only high-quality path subsets. The approach significantly improves aggregate throughput and reduces energy consumption compared to state-of-the-art schedulers.

3.1.7 Baseline and Comparison Schedulers

This section defines the baseline and state-of-the-art schedulers used for comparative evaluation. The selection spans the evolution of scheduling paradigms: static rule-based approaches, path-estimation heuristics, and learning-based predictive methods.

3.1.7.1 Single Path

Single path transmission represents traditional TCP connections where data flows over one network interface. This fundamental baseline establishes the lower performance bound, as multipath schedulers should achieve equal or superior throughput through path diversity. It quantifies inherent delay and jitter characteristics without reordering complications, and reveals HoL blocking impact when a path experiences congestion. In heterogeneous environments, single path performance fluctuates based on interface selection; both primary (Wi-Fi) and secondary (cellular) paths are evaluated independently to assess how effectively multipath schedulers combine interface capabilities.

3.1.7.2 Round Robin (RR)

RR cyclically distributes packets across available paths in fixed sequence [26], ensuring strict fairness regardless of path characteristics. Requiring no measurements or complex logic, RR is computationally efficient but fundamentally path-agnostic. When paths exhibit heterogeneous bandwidth, delay, or loss rates, RR indiscriminately sends traffic on inferior paths, causing substantial packet reordering. Packets on fast paths must wait in reorder buffers for slow-path counterparts, causing HoL blocking and potential throughput collapse below single-path performance [2]. Despite these limitations, RR establishes a performance floor that competent schedulers should exceed, isolating the impact of ignoring path characteristics.

3.1.7.3 Cheapest Pipe First (CPF)

CPF embodies cost-centric scheduling, prioritising one path (typically Wi-Fi) over others (cellular) to minimise monetary costs [37]. Traffic routes exclusively on the primary path until its congestion window saturates ($\text{InFlight}_1 \geq \text{CWND}_1$), with overflow spilling to secondary paths. When priorities are equal or the primary path is saturated, CPF transitions to SRTT mode, selecting the lowest-latency path with available capacity.
Whilst aligning with deployment economics (cellular data caps), CPF's strict prioritisation creates performance pathologies. When primary path quality degrades whilst secondary paths remain underutilised, CPF continues saturating the inferior path, increasing latency and potential loss. Poor responsiveness to dynamic changes—persisting with degraded Wi-Fi until congestion window fills—causes unnecessary delays [25]. Interactions with congestion control create bufferbloat on primary paths, artificially inflating RTT and misrepresenting available capacity.

3.1.7.4 Out-of-Order Transmission for In-order Arrival (OTIAS)

OTIAS [16] deliberately transmits packets out-of-order to achieve in-order arrival at the receiver, addressing reordering delays in heterogeneous multipath environments. By sending packets earlier on slower paths proportional to RTT differences, OTIAS orchestrates

in-sequence delivery. For path i, the algorithm computes required RTT wait cycles:

$$\text{RTT\_wait}_i = \left\lfloor \frac{\text{NotYetSent}_i - \text{AvailableCWND}_i}{\text{CWND}_i} \right\rfloor \tag{3.1}$$

selecting the path minimising total delivery time $T_i = (\text{RTT\_wait}_i + 0.5) \times \text{SRTT}_i$. A key feature is deliberately overloading faster paths' congestion windows to maximise low-latency path utilisation.

However, OTIAS suffers limitations in volatile wireless environments. Predictions assume stable RTT measurements; when path quality fluctuates due to handovers or interference, mispredictions cause reordering. Deliberately exceeding CWND limits triggers packet loss when conditions deteriorate.

3.1.7.5 Adaptive Cheapest Pipe First (ACPF)

ACPF, proposed by Pieska *et al.* [25], extends CPF with dynamic congestion window adjustment, enabling earlier secondary path utilisation when primary path bufferbloat is detected. A scaling factor $\gamma \in [0.47, 1.0]$ modulates effective congestion window:

$$\text{CWND}_{\text{effective}} = \gamma \times \text{CWND}_{\text{raw}} \tag{3.2}$$

adjusted based on queue occupancy—increasing γ when queue exceeds 10 packets (allowing more primary traffic), decreasing when queue drains (forcing earlier overflow). The minimum $\gamma = 0.47$ ensures Bottleneck Bandwidth and Round-trip propagation time (BBR) congestion control maintains approximately one BDP effective window, theoretically preventing bufferbloat whilst maintaining utilisation [41]. Experimental results demonstrate 24%–86% throughput improvements over CPF across diverse path configurations [25], particularly when paths exhibit significant asymmetry. However, ACPF remains fundamentally reactive—adjusting based on observed queue occupancy rather than predicting future quality. In rapidly changing wireless environments, reactive adjustments lag actual dynamics, yielding suboptimal decisions.

3.1.7.6 BLocking ESTimate (BLEST)

BLEST [4] addresses HoL blocking in heterogeneous multipath networks through predictive blocking estimation combined with dynamic load balancing. The algorithm maintains per-path scheduling time estimates $\hat{T}_i$ representing expected packet transmission latency on path i, computed from current CWND occupancy and recent throughput history. Load balancing employs weighted fair queuing where path weights w_i adjust based on estimated scheduling times:

$$w_i = \frac{1}{\hat{T}_i + \epsilon} \tag{3.3}$$

with smoothing factor ϵ preventing division-by-zero and excessive weight fluctuations. Packets are distributed proportionally to normalised weights, ensuring long-term fairness whilst permitting short-term deviations during burst events.

The algorithm demonstrates computational efficiency—$O(n)$ complexity per scheduling decision—suitable for resource-constrained deployments. However, the burst detection heuristic relies on CWND growth rate thresholds calibrated for specific congestion control algorithms; performance may degrade under alternative CCAs or rapidly oscillating conditions where burst predictions become unreliable.

3.1.7.7 Best Path First (BPF)

BPF [15] employs LSTM networks to predict future path quality over a 5-second horizon, anticipating which path will offer superior QoS rather than reacting to current state. The LSTM ingests per-path throughput, utilisation, and variability metrics sampled at 1-second intervals, outputting predictions that synthesise into a path quality score. The higher-scoring path becomes priority for the subsequent 5-second interval, with traffic routing using CPF-like priority logic.

BPF achieves substantial improvements—20% delay reduction and 21% jitter reduction for UDP, 30% faster FTP downloads in high-mobility scenarios [15]. However, constraints limit applicability: the fixed 5-second horizon may be suboptimal for varying mobility speeds; offline training prevents online adaptation to novel dynamics; aggregate path-level features discard fine-grained packet-level information; and binary priority decisions

preclude proportional load-balancing.

3.1.7.8 Peekaboo

Peekaboo [14], implemented for MP-QUIC, leverages multi-armed bandit theory combined with stochastic adjustment for dynamic heterogeneous paths. Unlike supervised prediction approaches, Peekaboo formulates scheduling as a contextual MAB problem balancing exploration of path choices against exploitation of known-good paths. The algorithm employs LinUCB, maintaining reward models for each action (send on path 1, path 2, or wait) using a 6-dimensional context: $\text{CWND}_i/\text{RTT}_i$, $\text{InFlight}_i/\text{RTT}_i$, and $\text{SendWindow}_i/\text{RTT}_i$ for both paths. For each decision, LinUCB computes expected reward with upper confidence bound:

$$E[R_{t,a}|\mathbf{x}_t] = \mathbf{x}_t^\top \boldsymbol{\theta}_a + \alpha\sqrt{\mathbf{x}_t^\top \mathbf{A}_a^{-1}\mathbf{x}_t} \tag{3.4}$$

where $\boldsymbol{\theta}_a$ is learned via ridge regression on historical rewards, and α controls exploration. A stochastic adjustment layer introduces controlled randomness to handle prediction uncertainty.

Peekaboo's online learning enables continuous adaptation without retraining, achieving 31.2% shorter download times in high-dynamicity emulation and 36.3% gains in real-world experiments [14]. However, the memoryless bandit formulation—conditioning only on current features—cannot exploit long-range temporal patterns. The throughput-based reward may not align with latency-sensitive applications.

3.1.7.9 Summary of Scheduler Characteristics

Table 3.1 summarises key characteristics facilitating comparative analysis of operational principles, computational requirements, and expected trade-offs.

The schedulers span a progression from simple reactive heuristics to sophisticated learning systems. Single Path and RR represent minimal baselines requiring essentially no computation. CPF introduces cost-awareness with $O(n)$ greedy prioritisation, whilst OTIAS advances through delivery time estimation. ACPF extends CPF with reactive congestion

Scheduler	Learning	Path Awareness	Prediction	Complexity
Single Path	None	N/A	None	$O(1)$
RR	None	None	None	$O(1)$
CPF	None	RTT, CWND	None	$O(n)$
OTIAS	None	RTT, CWND	Heuristic	$O(n)$
ACPF	None	RTT, CWND, Queue	Reactive	$O(n)$
BLEST	None	CWND, Throughput	Burst detection	$O(n)$
BPF	Supervised (LSTM)	Multi-metric	5-second horizon	$O(d \times h^2)$
Peekaboo	Online (MAB)	Multi-metric	Immediate	$O(d^2)$

Note: Complexity notation: n = number of paths, d = feature dimension, h = LSTM hidden units. Path Awareness indicates metrics informing scheduling decisions.

Table 3.1: Comparative characteristics of baseline multipath schedulers

inference, introducing limited temporal awareness without explicit learning. BLEST adds blocking estimation whilst maintaining $O(n)$ complexity through lightweight heuristics. Learning-based schedulers represent qualitative advances in sophistication. BPF's supervised learning maps historical metrics to future predictions with $O(d \times h^2)$ LSTM complexity, capturing temporal patterns but lacking online adaptation. Peekaboo's online bandit approach offers continuous adaptation with tractable $O(d^2)$ complexity, though its memoryless formulation limits exploitation of long-range dependencies.

3.1.8 Summary and Research Gaps

This chapter has critically analysed existing learning-based multipath scheduling approaches, demonstrating that whilst these methods offer significant advantages over reactive heuristics in volatile heterogeneous networks, several research gaps remain:

Training Overhead: Many proposed solutions require extensive offline training or substantial online learning phases that may not be practical for deployment in real-world mobile scenarios with rapidly changing conditions. Approaches such as those by Han *et al.* [99] demonstrate strong performance but impose computational requirements incompatible with resource-constrained deployments.

Protocol Modifications: Cross-layer solutions often require modifications to existing protocols that are unrealistic for Internet-scale deployment, limiting practical applicability. The MAMS approach [94] and multi-agent architectures [99] exemplify this limitation.

Generalisation: Most approaches are optimised and evaluated for specific scenarios (e.g., vehicular networks, video streaming) and may not generalise well across diverse application requirements and network conditions. The fixed 5-second horizon of BPF [15] illustrates this constraint.

Computational Complexity: Multi-agent and complex neural network approaches may introduce computational overhead limiting suitability for resource-constrained mobile devices. The separation of prediction and decision-making components can address this through appropriate architectural decomposition.

Scalability: Performance of many proposed schedulers degrades as concurrent user numbers increase, particularly for redundancy-based approaches [90, 91] that consume additional network resources.

Temporal Modelling: Contextual bandit approaches such as Peekaboo [14] condition only on current features, unable to exploit long-range temporal patterns. LSTM-based approaches capture sequential dependencies but may not model cross-path interactions effectively.

These gaps motivate the need for efficient, adaptable multipath scheduling solutions that can operate effectively in highly mobile HetNet environments whilst maintaining practical deployment constraints and computational efficiency. The proposed DARA framework addresses these gaps through Transformer-based prediction enabling long-range temporal modelling with cross-path attention, combined with DQN-based decision-making that optimises multi-objective rewards without requiring protocol modifications or excessive computational resources.

Chapter 4

Multipath Scheduling Framework Design

This chapter presents the design and implementation of two novel multipath scheduling frameworks that form the key research contributions of this thesis. Section 4.1 introduces the BPF scheduler, which employs LSTM-based path quality prediction over 5-second horizons for priority-based path selection, serving as both a contribution and comparative baseline. Section 4.2 presents the DARA framework, detailing its Transformer-based congestion metric forecasting, DQN-based multi-objective rate allocation, NAS-optimised hyperparameters, and kernel-userspace integration architecture.

4.1 Best Path First Scheduler Framework

BPF [15] serves as a key comparison baseline for DARA, representing a novel prediction-based multipath scheduling. BPF integrates LSTM-based path quality prediction with priority scheduling, addressing the limitations of reactive approaches that respond only to current network conditions. Figure 4.1 illustrates the BPF scheduler operating within the ATSSS-HL framework introduced in Section 2.1, where 3GPP and non-3GPP paths connect the User Equipment (UE) to the User Plane Function (UPF). The architecture corresponds to the multipath scheduling model shown in Figure 2.1, with the MP-DCCP transport protocol (Section 2.1.3.3) providing the underlying datagram delivery. A ter-

minal moves across a network with the motion defined in a replayable trace file. The QoS prediction identifies the path to the Base Station (BS) as the better path and by using BPF scheduling can prioritise it and fill it first (where packets are shown in grey), resulting in reduced HoL blocking and reduced need for packet reordering, whereas CPF schedules greedily based on the current lowest SRTT, which does not minimise aggregation delay. The QoS prediction can identify the path to the BS as the better path and by using the BPF scheduling can prioritise it and fill it first (where packets are shown in grey), resulting in reduced HoL blocking and reduced need for packet reordering, whereas CPF schedules greedily based on the current lowest SRTT, which does not minimise aggregation.

Path prioritisation schedulers represent a conceptually simple approach to prediction-based scheduling: rather than continuously adjusting per-path transmission rates, they make binary decisions about which path should receive priority for a fixed time window. This simplicity enables efficient implementation and low computational isolates the performance benefit of continuous adjustment versus binary path selection under identical predictive capabilities.

In the BPF algorithm, path QoS is predicted using an offline model of the application-level traffic at the UE, based on throughput, path utilisation and volatility, and queue drainage parameters. These sequential data require a RNN for predictions, where LSTM layers are used to recall patterns over noisy sequences, with dropouts and feedback applied to prevent overfitting and improve memory recall, respectively. Predictions are performed over 5-second intervals, and the slow reaction time provides a comparatively lower overhead than having to apply intelligence for individual packet level scheduling.

4.1.1 Path Quality Prediction

To address the QoS drops caused by OFO delivery and HoL blocking, it is necessary to predict path QoS sequences and find patterns in noisy historical data, requiring a RNN and LSTM respectively. The training model to predict path QoS is shown in Figure 4.2. Whilst the choice of parameters is an area of active research, care was taken to prevent overfitting.

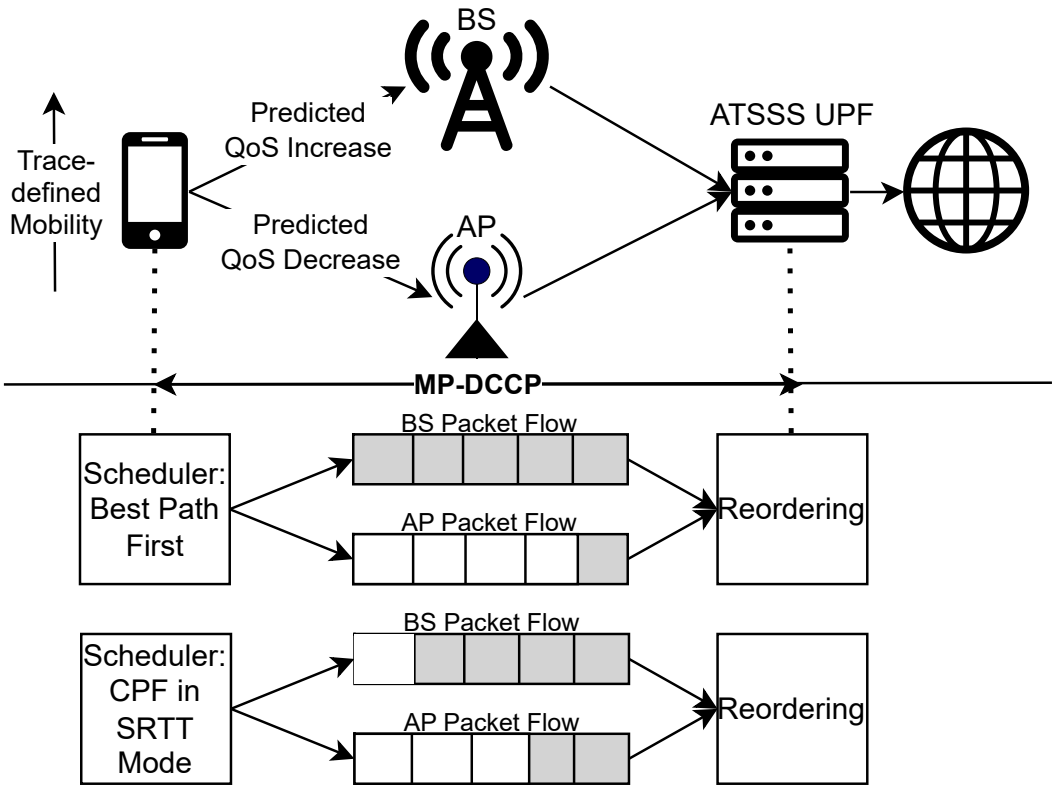

Figure 4.1: A multipath client-server connection of a moving user is shown, with the predicted path QoS, and the corresponding scheduling of BPF and CPF in SRTT mode

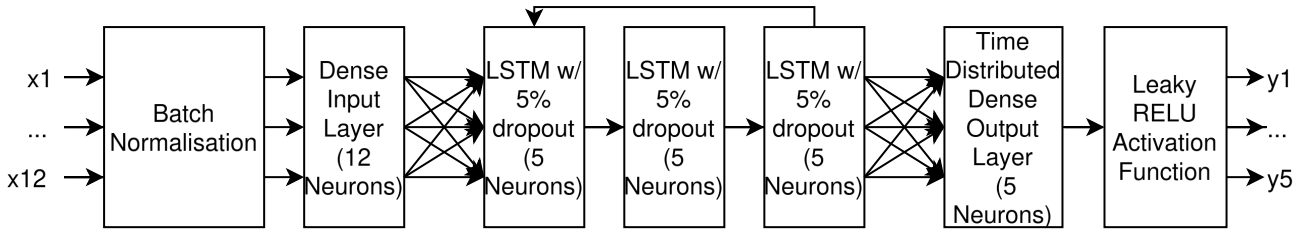

Figure 4.2: Block diagram representing the LSTM RNN model

Input Layer: A RNN takes time serial data about the queue drainage, path throughput, path utilisation, and path variability as input. These data are used to predict QoS, which is used to make scheduling decisions. QoS_{matrix} is the input for the neural network. QoS_{matrix}=$(t, D, T_{p1}, T_{p2}, T_O, U_{p1}, U_{p2}, U_O, U_{p1}(t-5), U_{p2}(t-5), V_{p1}(t-5), V_{p2}(t-5))$ (Time, Queue Drainage, Throughput Path 1, Throughput Path 2, Throughput Overall, Utilisation Path 1, Utilisation Path 2, Utilisation Overall, Path 1 Past Utilisation, Path 2 Past Utilisation, Path 1 Past Variability, Path 2 Past Variability). To increase the prediction accuracy, the predictors are normalised to form categorical variables, the divisor being the maximum element in the list up to that time. To decrease computational runtime and increase prediction accuracy, the time interval has been changed to take a moving average of the data in the last second, and the variability is measured as the moving standard deviation over one second. The dense layer is fully connected and linearly transforms the features for the hidden layers.

Hidden Layers: The LSTM layers are able to recall temporal dependencies in sequential data, returning data patterns and sequences rather than just the last timestamp. The dropout prevents overfitting and the feedback improves the model's memory.

Output Layer: The data is passed to a time-distributed dense output layer to form a sequence vector for each time step, with a non-linear activation function to produce a categorical output for the path utilisations and variabilities.

Loss Function and Optimiser: A Mean Absolute Error (MAE) loss function scheduling granularity: to evaluate the model's performance:

$$MAE = \frac{1}{N}\Sigma_{i=1}^{N}|Y_i - \hat{Y_i}| \tag{4.1}$$

The model is trained using Adafactor (Section 2.2.1.2) with a learning rate of 0.05 over 2,500 epochs. This extended training duration is necessary for the stateful LSTM architecture to learn temporal dependencies in the path utilisation and variability sequences, with Adafactor's automatic learning rate scaling maintaining stable convergence throughout.

4.1.2 Best Path First Algorithm Design

Algorithm 1 Best Path First (BPF)

Define $f(\cdot)$ as an LSTM-RNN with leaky-RELU activation
Train $f(\cdot)$ on historical QoS_{matrix} to minimise MAE for $U_{p1}(t_{k+5}), U_{p2}(t_{k+5}), V_{p1}(t_{k+5})$, and $V_{p2}(t_{k+5})$
while true **do**
    $f(QoS_{matrix}(t_k)) = [U_{p1}(t_{k+5}), U_{p2}(t_{k+5}), V_{p1}(t_{k+5}), V_{p2}(t_{k+5})]$
    $QoS_A = U_{p1}(t_{k+5}) + V_{p1}(t_{k+5}) * c$ ▷ $c = 1.25$ as empirically determined
    $QoS_B = U_{p2}(t_{k+5}) + V_{p2}(t_{k+5}) * c$
    if $QoS_A > QoS_B$ **then**
        Set path 1 as priority
    else
        if $QoS_A < QoS_B$ **then**
            Set path 2 as priority
        else
            Keep priority unchanged
        end if
    end if
    Initiate timer (5 seconds)
end while

Based on the QoS predictions from the LSTM-RNN model, the algorithm, shown in Algorithm 1, predicts: future path utilisation in the next 5-seconds for path 1 and path 2; $U_{p1}(t_{k+5})$ and $U_{p2}(t_{k+5})$ respectively, and the future path variability in the next 5-seconds for path 1 and path 2; $V_{p1}(t_{k+5})$ and $V_{p2}(t_{k+5})$ respectively. The model outputs an index; low utilisation and high variability have high indices and vice versa. The QoS of a path is the sum of the utilisation and variability indices of that path, with more weight being given to the variability, which reflects frequent path flip-flops.

If the model predicts that in the next 5-seconds the non-priority path (e.g. QoS_B) becomes comparatively better than the priority path (e.g. QoS_A), the controller will send a command to the UE to switch path priority. To prevent path flip-flops, further switches are restricted in the following 5-seconds so to only switch due to the priority path failing. Improving path QoS during cellular handovers is more difficult. The path cannot be analysed by looking at instantaneous parameters. As such, handover patterns could be treated separately from the general case, but there is no evidence that the quality of the results would improve. Therefore this is treated as part of the normal predictions.

4.1.3 Limitations of Path Prioritisation

The fundamental limitation of BPF lies in its scheduling granularity: path prioritisation performs switching (selecting one path) rather than splitting (distributing across paths). Path prioritisation can be viewed as a coarse discretisation of rate allocation, where $\phi_i \in \{0, 1\}$ (off or full capacity) rather than $\phi_i \in [0, 1]$ (arbitrary fraction).

This architectural constraint creates cascading implications. Binary decisions cannot proportionally split traffic when both paths offer usable but heterogeneous capacity—the non-priority path remains idle until the priority path saturates, potentially leaving capacity unexploited. When Path 1 offers 70% of its nominal capacity and Path 2 offers 30%, a fine-grained scheduler could set $\phi_1 = 0.7, \phi_2 = 0.3$ to match, whilst BPF must choose one path entirely. The 5-second decision window forces commitment to priority assignments despite intra-window variability, whereas sub-second control cycles could adapt to transient bursts.

However, BPF's simplicity confers practical advantages: no online learning phase (purely offline-trained LSTM), no per-packet inference overhead, and straightforward implementation without kernel-userspace coordination mechanisms. The comparative evaluation determines whether fine-grained rate allocation yields sufficient QoS improvements over BPF's efficient path prioritisation to justify additional complexity in resource-constrained mobile scenarios.

4.2 Deep Adaptive Rate Allocation Framework

This section presents the DARA framework architecture and its constituent components. It begins by establishing DARA's core novelty, predictive rate control through CWND fraction adjustment, and contrast this approach with existing multipath scheduling paradigms. It then detail the Transformer-based prediction model that forecasts path congestion metrics (CWND and SRTT) over multiple horizons spanning 100–500ms, followed by the DQN policy network formulation that solves the multi-objective optimisation problem (throughput maximisation, congestion-induced delay avoidance, idleness minimi-

sation) using these predictions. Finally, it describes the integrated training and execution algorithm that enables sub-RTT scheduler adaptation to volatile network conditions characteristic of high-mobility scenarios.

4.2.1 Overview and Novelties of DARA

DARA introduces a fine-grained predictive rate control mechanism based on a Transformer model. Given that channel-quality-induced bursts due to mobility-driven fluctuation occur over short time intervals, frequently in the order of 100ms, reactive schedulers fail to fully exploit path capacity during bursts and fail to reduce transmission rates once bursts cease, leading to capacity waste and increased OFO packets.

In this deployment, each subflow runs independent BBR-like congestion control maintaining separate CWND, the maximum unacknowledged segments allowed in-flight per path, assuming successful delivery. This differs from MP-TCP's coupled algorithms linking subflow increases for shared-bottleneck fairness; this decoupled configuration suits non-shared bottlenecks (cellular-WiFi traverse distinct infrastructure). The scheduler's role is to distribute connection-level data across available paths whilst respecting each path's CWND constraint, a design choice, as some schedulers (e.g., OTIAS) deliberately oversubscribe CWNDs. Schedulers that select paths based on instantaneous metrics are not efficient as this creates a fundamental coupling problem: by the time the scheduler observes CWND changes (indicating capacity variations), transmission opportunities have already been missed or paths have been oversubscribed.

To address this, a multi-objective RL problem is formulated, where a DQN policy network outputs discrete CWND fraction actions—effectively rate-limiting each path by restricting its usable congestion window to $\phi_p \times \text{CWND}_p$. This fractional control operates above the congestion control layer: whilst each path's native congestion control (e.g., BBR) continues to manage CWND evolution based on observed network conditions, DARA modulates how much of that CWND the scheduler actually utilises for packet transmission. This multiplicative control directly constrains the scheduler's transmission decisions (i.e., when path p has $\text{CWND}_p = 100$ segments and DARA sets $\phi_p = 0.3$, the scheduler treats the

effective capacity as only 30 segments, preventing oversubscription before congestion materialises). As illustrated in Figure 4.4, when DARA predicts Path 1 degradation (declining CWND, rising SRTT), it proactively limits the path to a reduced capacity ($\phi_1 = 0.3$) whilst allowing stable Path 2 full utilisation ($\phi_2 = 1.0$). The scheduler then respects these limits when selecting paths for queued segments, and this preemptive throttling prevents the majority of segments that would otherwise be sent from arriving during Path 1's predicted degradation period, resulting in near-sequential packet arrival at the receiver and minimal HoL blocking despite heterogeneous path conditions.

The key advantage is the ability to predict: whilst congestion control algorithms adjust CWND reactively based on observed ACKs and losses (typically over multiple RTT timescales), DARA adjusts the scheduler's utilisation of that CWND predictively, enabling sub-RTT adaptation to predicted capacity changes. As shown in Figure 4.3, static schedulers maintain fixed allocation regardless of CWND evolution, causing persistent overutilisation of degraded paths and under-utilisation of burst capacity. Reactive schedulers, e.g., CPF, exhibit observation-reaction lag τ, adjusting allocations only after observing CWND changes through congestion control feedback, resulting in delayed responses that miss burst opportunities and oversubscribe degrading paths. DARA eliminates this lag through its predictive horizon, preemptively adjusting CWND fractions to match anticipated capacity variations.

This observation stems from the fundamental tension between scheduler responsiveness and congestion control stability: schedulers must make rapid packet-assignment decisions (e.g., every 1ms), but congestion control algorithms operate on slower timescales (RTT-based CWND updates). Across redundancy-based, cross-layer, estimation-based, and multi-objective optimisation solutions, a recurring challenge is the inability to fully leverage transient path improvements whilst mitigating volatile degradations. While existing advanced schedulers incorporate forward-looking elements, such as offline meta-learning in FALCON [98] or cross-layer mobility awareness in MAMS [94], and other predictive approaches have been proposed (Zeng *et al.*[97], BLEST[4]), DARA uniquely combines high-temporal-resolution prediction with continuous per-path rate control through CWND

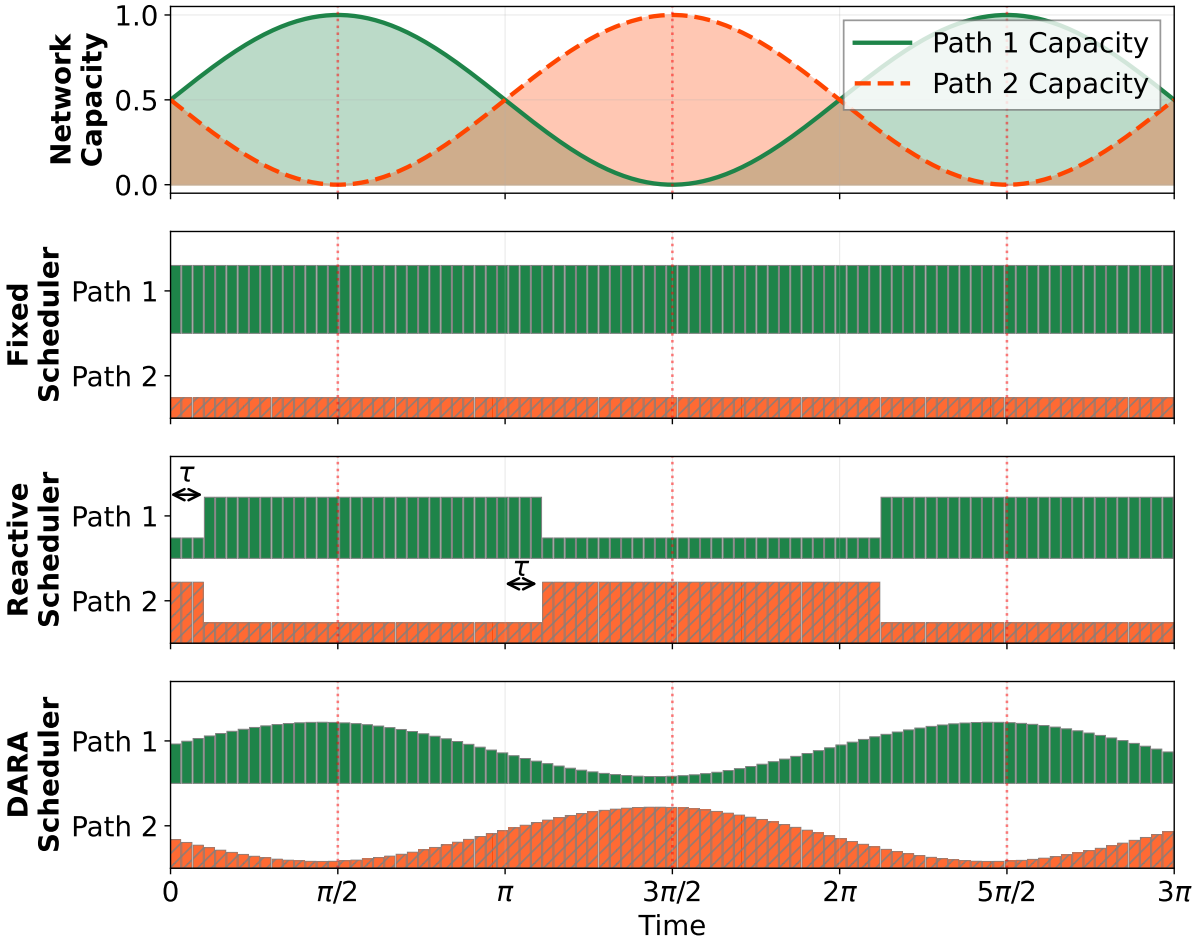

Figure 4.3: Scheduler Response to Antiphase Path Capacity Variations. Fixed rule schedulers maintain fixed allocation regardless of path capacity, causing overutilisation of poor paths and underutilisation of bursts. Reactive schedulers exhibit observation-reaction lag τ, adjusting allocations only after observing capacity changes, resulting in delayed responses that miss burst opportunities and oversubscribe degrading paths. DARA eliminates this lag through 500ms predictive horizon, preemptively adjusting CWND fractions to match anticipated capacity variations, fully exploiting bursts while avoiding oversubscription.

fraction adjustment.

This distinguishes DARA from: (1) discrete-action DRL schedulers (ReLeS, Rosello's DQN[86]) that select paths binarily, treating CWND as a hard constraint rather than a tunable resource; (2) binary decision frameworks (Peekaboo, FALCON[98]) that lack continuous rate adjustment capabilities; (3) specialised optimisers (Dong *et al.*[100], GAPS[101]) with narrower objectives that do not explicitly model the scheduler-congestion control interaction; and (4) cross-layer approaches (MAMS, Chorus[93]) requiring protocol modifications. By proactively adjusting per-path rate limits to match predicted congestion window dynamics, DARA achieves superior burst utilisation and reduces OFO packets across diverse mobility scenarios, particularly excelling when burst durations align with the 100ms control cycle and the prediction horizon within which forecasts remain accurate.

4.2.2 System Architecture

DARA comprises three components operating across execution contexts: a pre-trained Transformer prediction model (userspace), a DQN policy network (userspace), and a multipath packet scheduler (kernelspace). The Transformer generates congestion forecasts from historical telemetry; the DQN maps current observations and predictions to CWND fraction actions; the scheduler applies these fractions to packet allocation decisions. A custom character device driver synchronises state collection (kernelspace $\rightarrow$ userspace) and action execution (userspace $\rightarrow$ kernelspace) at millisecond granularity whilst enabling the DQN to operate on 100ms control cycles. This separation isolates computationally-intensive prediction and learning from time-critical packet dispatch.

Figure 4.5 illustrates DARA's architectural components and data flows across execution boundaries. The system operates as a closed control loop: kernelspace state collection continuously samples per-path telemetry (CWND, SRTT, in-flight packets, queue depths) at 1ms intervals, which flows upward through the character device interface via READ operations into userspace buffers. The Transformer model processes these time-series features to generate congestion predictions at 5 horizons (100, 200, 300, 400, 500ms), which the DQN policy network then maps to optimal CWND fraction actions $\{\phi_1, \ldots, \phi_N\}$ for

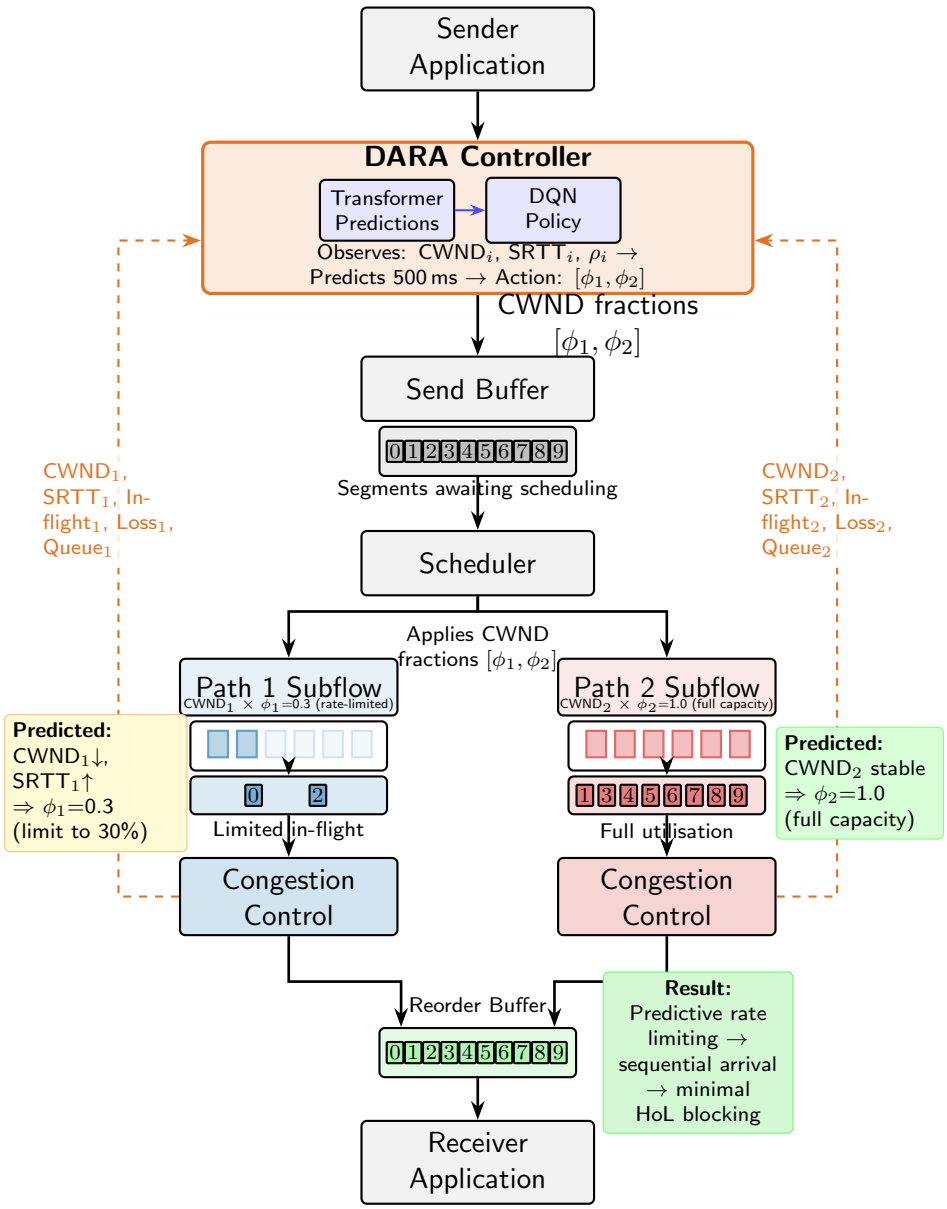

Figure 4.4: DARA's Predictive Rate Control Architecture. The Transformer predicts CWND and SRTT evolution for each path, using 2 paths in this example. The DQN policy uses these predictions to compute CWND fractions that rate-limit each path by multiplying its effective congestion window. In this example, DARA predicts Path 1 degradation (declining CWND, rising RTT) and pre-emptively limits it to $\phi_1 = 0.3$ (30% capacity), while allowing the stable path full utilisation at $\phi_2 = 1.0$. This proactive rate limiting prevents oversubscription on the degrading path, enabling sequential packet arrival and minimal head-of-line blocking.

each path. These actions flow downward through WRITE operations to the kernelspace CWND application layer, which multiplies each path's native congestion window by its assigned fraction ($\phi_p \times \text{CWND}_p$) before the MP-DCCP scheduler performs packet dispatch. This bidirectional flow enables proactive rate control whilst maintaining the separation between the ML inference control cycle (100ms) and fast packet scheduling (∼1ms cycle).

4.2.2.1 Kernel-Userspace Interface

MP-DCCP packet scheduling executes in kernelspace while DARA's Transformer and DQN components operate in userspace, requiring a bidirectional communication mechanism. As shown in Figure 4.5, a custom Linux character device driver was implemented that exposes per-path congestion state at 1ms sampling intervals via READ operations, providing: timestamp, socket identifier, CWND, current CWND utilisation fraction, in-flight packets, SRTT, path priority, and queue depths. WRITE operations allow the userspace DQN to update per-path CWND utilisation fractions, which the kernelspace CWND application layer immediately applies to constrain the scheduler's path selection decisions.

Hierarchical Scheduling: DARA's inference pipeline (Transformer prediction plus DQN action selection) completes in approximately 3ms, while MP-DCCP demands per-packet scheduling decisions at millisecond granularity. The 100ms control cycle is therefore not constrained by inference latency but chosen to balance decision frequency against state observability: shorter cycles would produce highly correlated successive states providing little new information, whilst longer cycles would reduce responsiveness to rapid capacity changes. As depicted in Figure 4.5, DARA operates at these 100ms intervals computing optimal CWND fractions from predicted congestion trajectories; a lightweight kernelspace scheduler executes packet allocation at 1ms intervals by applying these fractions as multiplicative weights to instantaneous congestion windows. This decomposition enables proactive exploitation of predicted burst opportunities while maintaining sub-millisecond packet dispatch latency.

4.2.3 Transformer-Based Prediction Model

To enable predictive scheduling, a Transformer model with multi-head attention mechanisms trained on congestion control telemetry forecasts CWND and SRTT trajectories. The architecture processes sequences of 8 aggregated time steps (each representing 100ms bins) to predict values at 5 future horizons (100, 200, 300, 400, 500ms) for each path, with prediction accuracy peaking at the 300ms horizon (Section 6.1.2).

The Transformer architecture was selected over LSTM approaches due to parallelisation advantages. LSTM networks process sequences sequentially—each hidden state h_t depends on h_{t-1}—requiring $\mathcal{O}(L)$ sequential operations for sequence length L. Transformer self-attention computes all token relationships simultaneously in $\mathcal{O}(1)$ sequential depth, enabling GPU parallelisation that reduces inference from approximately 2.5s (LSTM) to approximately 2.5ms despite comparable parameter counts. This acceleration proves critical for exploiting transient 100–700ms capacity bursts where 2.5s inference would miss exploitation opportunities entirely. The selection is validated empirically against LSTM, MLP, and linear predictors in Section 6.1.1.

The deployed model uses 3 encoder blocks (selected via the depth analysis in Section 6.1.2), each comprising multi-head self-attention with 5 heads, a position-wise feedforward network of dimension 360 with GELU activation, layer normalisation, and residual connections. Dropout rates increase from 10% to 15% across depth, providing implicit regularisation for longer-range predictions (Table 4.1).

Block	FF Dim	Heads	Activation	Dropout
1	360	5	GELU	10%
2	360	5	GELU	10%
3	360	5	GELU	15%

Table 4.1: Deployed Transformer block specifications (3-layer configuration). Increasing dropout with depth provides implicit regularisation for longer-range prediction targets.

Table 4.2 contrasts key characteristics of DARA's Transformer with the prior BPF LSTM model. The architectural shift from sequential stateful processing to attention-based mechanisms, combined with 1000× finer temporal resolution (1ms vs 1s sampling), reduces inference time from ∼2.5s to ∼2.5ms, critical for operating within the 100ms control cycle

whilst leaving substantial margin for system overhead.

Characteristic	BPF LSTM	DARA Transformer
Architecture	3×LSTM(50)	3-block attention
Parameters	249,040	∼300,000
Output	Categorical (1–4)	Continuous CWND, SRTT
Horizons	Single (5s)	5×100ms (100–500ms)
Sampling	1s	1ms
Inference	∼2.5s	∼2.5ms

Table 4.2: Comparison of LSTM and Transformer prediction models for multipath scheduling.

4.2.4 DQN-Based Multi-Objective Optimisation

Having obtained accurate forecasts of path characteristics, packets are scheduled across paths to minimise the impact of mobility-induced link quality variations on throughput. DARA operates as a traffic splitting algorithm that compensates for path characteristic fluctuations: degradation in Path 1's QoS with stable Path 2 QoS triggers preemptive traffic shifts. The Transformer's predictions $Z_p(t) = [w_p(t+h) \quad \tau_p(t+h)]$ are integrated with current observations $X_p(t) = [w_p(t) \quad \tau_p(t)]$ to form the optimisation input over the set of available subflows P, where $p \in P$ denotes each subflow.

Agent: The userspace multipath scheduler determines per-subflow CWND fractions $\phi_p(t)$ provided to the kernelspace scheduler, controlling the effective transmission capacity utilised on path p at time t.

Observation and State: At time t, the agent observes an 18-dimensional state vector s_t constructed from current and predicted metrics for all paths, augmented with normalised rate-of-change features and congestion indicators:

$$s_t = \Big[\underbrace{w_p(t), \tau_p(t)}_{p\in P},\ \underbrace{\delta^{\text{mid}}_{w,p}, \delta^{\text{mid}}_{\tau,p}}_{p\in P},\ \underbrace{\delta^{\text{far}}_{w,p}, \delta^{\text{far}}_{\tau,p}}_{p\in P},\ \underbrace{\bar{\phi}_p(t-1)}_{p\in P},\ \underbrace{c^w_p, c^\tau_p}_{p\in P}\Big] \tag{4.2}$$

where $w_p(t)$ and $\tau_p(t)$ denote current CWND and SRTT for path p; the normalised deltas $\delta^{\text{mid}}_{w,p} = (\hat{w}_p(t+h/2) - w_p(t))/\max(|w_p(t)|, \epsilon)$ and $\delta^{\text{far}}_{w,p} = (\hat{w}_p(t+h) - w_p(t))/\max(|w_p(t)|, \epsilon)$ capture predicted rates of change at mid-horizon and full-horizon respectively (analogously

for δ_τ); $\bar{\phi}_p(t-1) = \phi_p(t-1)/100$ encodes the previous action; and binary congestion flags $c_p^w = \mathbb{1}[\hat{w}_p(t+h) < 0.95 \cdot w_p(t)]$ and $c_p^\tau = \mathbb{1}[\hat{\tau}_p(t+h) > 1.1 \cdot \tau_p(t)]$ signal predicted degradation. For $N = 2$ paths, this yields $4 + 4 + 4 + 2 + 4 = 18$ dimensions, scaling as $9N$ for arbitrary path counts.

This representation provides the DQN with both absolute state (current metrics), relative trajectory (normalised deltas capturing direction and magnitude of predicted change), action history (previous fractions for stability-aware decisions), and explicit congestion signals (thresholded binary indicators reducing the burden on the network to learn degradation detection from raw values).

Action: The action $a_t = \{\phi_p(t)\}_{p \in P}$ adjusts CWND fractions for all N paths, where $\phi_p(t) \in \Phi$ draws from a discrete set of K fraction levels. The joint action space has cardinality K^N, enabling efficient DQN optimisation for small N whilst requiring hierarchical action decomposition for larger path counts. The selection of K and Φ is analysed in Section 6.1.5.

Policy: The policy $\mu(s_t) : \mathcal{S} \to \mathcal{A}$ maps environmental states to actions, learnt through DRL to maximise cumulative reward.

Reward: The reward function $R(s_t, a_t)$ comprises six normalised components, each designed to occupy approximately $\mathcal{O}([-1, 1])$ range to prevent any single objective from dominating gradient updates (verified in Section 6.1.4). For each path $p \in P$, let $w_p(t)$ and $\tau_p(t)$ denote current CWND and SRTT, and $\hat{w}_p(t+h)$ and $\hat{\tau}_p(t+h)$ denote Transformer-predicted values at horizon h.

- **Throughput Maximisation ($R_{\text{throughput}}$):** Encourages predicted CWND increases as fractional change:

$$R_{\text{throughput}}(t) = \sum_{p \in P} \frac{\hat{w}_p(t+h) - w_p(t)}{w_p(t)} \tag{4.3}$$

  Positive values incentivise exploiting predicted burst opportunities.

- **Congestion-induced Delay Avoidance (R_{delay}):** Encourages predicted SRTT

reductions:

$$R_{\text{delay}}(t) = \sum_{p \in P} \frac{\tau_p(t) - \hat{\tau}_p(t+h)}{\tau_p(t)} \tag{4.4}$$

This component complements the throughput reward: whereas $R_{\text{throughput}}$ incentivises exploiting paths with increasing CWND, R_{delay} penalises the bufferbloat scenario where CWND rises but SRTT also rises due to queue buildup, which $R_{\text{throughput}}$ alone would reward. Given that throughput under BBR-like congestion control is approximately CWND/SRTT, the two components are partially redundant under stable conditions; R_{delay}'s distinct contribution lies in penalising this specific divergence.

- **Preemptive Adaptation ($R_{\text{preemptive}}$):** Rewards proactive ϕ adjustments aligned with predictions. For each path $p \in P$, the per-path signal is:

$$r_p^{\text{pre}}(t) = \begin{cases} +0.5 & \text{if } \hat{w}_p < w_p \text{ and } \phi_p(t) < \phi_p(t-1) \\ -0.5 & \text{if } \hat{w}_p < w_p \text{ and } \phi_p(t) \geq \phi_p(t-1) \\ +0.5 & \text{if } \hat{w}_p \geq w_p \text{ and } \phi_p(t) > \phi_p(t-1) \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

The aggregate preemptive reward sums contributions from all paths:

$$R_{\text{preemptive}}(t) = \frac{1}{N} \sum_{p \in P} r_p^{\text{pre}}(t) \tag{4.6}$$

This component penalises increasing ϕ_p on degrading paths, preventing reactive behaviour whilst rewarding early reduction on predicted-degraded paths and compensatory increases on stable paths.

- **Stability Penalty ($R_{\text{stability}}$):** Penalises large ϕ oscillations, normalised by the maximum possible change:

$$R_{\text{stability}}(t) = -\frac{1}{100} \sum_{p \in P} |\phi_p(t) - \phi_p(t-1)| \tag{4.7}$$

Excessive changes trigger spurious congestion control responses and receiver-side reordering buffer overflow. This soft constraint encourages smooth rate transitions when predicted capacity changes are gradual.

- **Quality Sustainability (R_{quality}):** Rewards sustained high bitrate via predicted throughput-delay ratio:

$$R_{\text{quality}}(t) = \frac{1}{10^6} \sum_{p \in P} \frac{\hat{w}_p(t+h)}{\max(1, \hat{\tau}_p(t+h))} \times 8 \times 1500 \tag{4.8}$$

  Exponentially smoothed ($R_{\text{quality}}(t) = 0.9R_{\text{quality}}(t-1) + 0.1R_{\text{quality}}(t)$) to prevent oscillations in video streaming quality.

- **Idleness Prevention (R_{idleness}):** Penalty preventing simultaneous minimal utilisation across all paths:

$$R_{\text{idleness}}(t) = \begin{cases} -1 & \text{if } \phi_p(t) = \phi_{\min} \; \forall \, p \in P \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

  This prevents pathological policies that conservatively throttle all paths during uncertainty, effectively idling the connection despite available capacity.

The combined reward function balances these objectives through learned weights:

$$\begin{aligned} R(s_t, a_t) = w_{\text{tput}} \cdot R_{\text{throughput}} + w_{\text{delay}} \cdot R_{\text{delay}} + w_{\text{qual}} \cdot R_{\text{quality}} \\ + w_{\text{pre}} \cdot R_{\text{preemptive}} + w_{\text{stab}} \cdot R_{\text{stability}} + w_{\text{idle}} \cdot R_{\text{idleness}} \end{aligned} \tag{4.10}$$

All six components are normalised to comparable magnitude, ensuring that weight values directly reflect relative objective importance rather than compensating for scale mismatches. The weight values are determined through systematic search (Section 6.1.3).
This formulation enables the DQN to learn policies balancing instantaneous capacity exploitation with medium-term sustainability and practical constraints. Formulating multipath scheduling as a Markov Decision Process allows the agent to determine CWND fractions based on network state changes, effectively limiting path utilisation to follow

congestion dynamics.

4.2.4.1 Fairness Considerations

Within this work, *fairness* refers specifically to the prevention of flow starvation under concurrent multipath workloads, rather than to classical notions such as max-min fairness or proportional fairness. In the ATSSS-HL tunnel scenario, a scheduler that over-allocates capacity to one flow at the expense of another degrades the QoE of the starved flow without necessarily improving aggregate throughput.

DARA addresses this through two mechanisms that operate without an explicit fairness term in the reward function. First, the idleness prevention component (Equation 4.2.4) penalises simultaneous minimal utilisation across all paths, discouraging the policy from starving any flow by conservatively throttling all capacity. Second, the stability penalty (Equation 4.7) discourages large, rapid ϕ oscillations that would otherwise manifest as bursty allocation favouring one flow and then the other. Fairness is therefore not optimised directly but emerges from the interaction of these constraints with the throughput and delay objectives.

Fairness is assessed quantitatively through per-flow throughput, delay, and completion time measurements under the concurrent workload scenarios described in Sections 5.6.3 and 5.6.6. A scheduler is considered fair if no individual flow's throughput falls below a meaningful fraction (evaluated empirically relative to the CPF baseline) of its single-flow performance, and if per-flow completion times remain within a bounded ratio of one another. Section 6.3.4 applies this assessment to Quick UDP Internet Connections (QUIC)+MP-TCP concurrent transfers.

4.2.5 Hyperparameter Optimisation via Neural Architecture Search

DARA's performance critically depends on hyperparameter selection across three coupled subsystems: the Transformer prediction model, the DQN policy network, and the multi-objective reward function. Manual tuning of this 20-dimensional hyperparameter space proves prohibitive due to non-convex interactions—for example, larger batch sizes require

correspondingly higher learning rates for stable convergence. Optuna-based NAS is therefore employed with Tree-structured Parzen Estimator sampling and median pruning to systematically explore this space over 100 trials on a high-performance computing cluster (3,500 cores, 30TB RAM).

Search Space Definition: The optimisation spans four hyperparameter categories:

1. **Temporal Resolution:** Window size $\in \{100, 200, 300, 500, 700\}$ms controls data aggregation granularity for Transformer inputs. Smaller windows increase sensitivity to transient bursts but amplify noise; larger windows smooth predictions but introduce lag.

2. **Reward Function Weights:** Six weights balance competing objectives. The search explores whether throughput-delay trade-offs dominate or whether preemptive adaptation provides substantial value. The final weight values, determined through the refined search procedure described in Section 6.1.3, are reported in Table 6.1.

3. **DQN Architecture:** Hidden dimension $\in \{64, 128, 256, 512\}$, layer count $\in \{2, 3, 4\}$, learning rate $\in [10^{-5}, 10^{-3}]$ (log-uniform), batch size $\in \{32, 64, 128, 256\}$, discount factor $\gamma \in [0.90, 0.999]$, and target network soft-update rate $\tau \in [0.001, 0.01]$ control policy learning dynamics. Replay memory capacity $\in \{5000, 10000, 20000\}$ balances sample diversity with memory overhead.

4. **Exploration Strategy:** Initial exploration rate $\epsilon_{\text{start}} \in [0.7, 1.0]$, final rate $\epsilon_{\text{end}} \in [0.01, 0.1]$, and decay steps $\in [500, 2000]$ control the exploitation-exploration trade-off. Episode count $\in [500, 2000]$ and action space granularity determine training duration and rate control precision, with the action space selection analysed in Section 6.1.5.

Optimisation Objective: Each trial evaluates a candidate configuration by training a DQN agent on 500,000-row samples randomly extracted from the 30M-sample dataset. To prevent overfitting to specific mobility patterns, sampling locations are randomised

per trial (seed = 42 + trial_number). The multi-objective fitness function combines four metrics:

$$\text{Fitness} = 0.4 \cdot \bar{R} + 0.3 \cdot \bar{T} + 0.2 \cdot \frac{1}{\bar{D} + 1} + 0.1 \cdot \frac{1}{E_{\text{conv}} + 1} \tag{4.11}$$

where $\bar{R}$ denotes average episode reward (last 100 episodes), $\bar{T}$ is average throughput, $\bar{D}$ is average delay, and E_{conv} is the convergence episode index (when reward reaches 95% of best observed). Reward receives highest weighting (40%) as it directly encodes the learnt multi-objective balance; throughput (30%) and delay (20%) provide explicit performance verification; convergence speed (10%) encourages sample-efficient learning.

Computational Strategy: The NAS leverages HPC parallelisation to evaluate trials concurrently. Each trial processes a distinct data window through four stages: (1) DataFrame extraction and subflow filtering (20% frequency threshold), (2) time-series feature engineering via 5-timestep sliding windows, (3) temporal aggregation into window-sized bins, and (4) MinMax log-scaling. Invalid configurations (e.g., insufficient data post-filtering, numeric conversion failures) are pruned early via Optuna's MedianPruner after 10 startup trials. The Tree-structured Parzen Estimator sampler exploits correlations between hyperparameters (e.g., larger batch sizes benefit from higher learning rates) to focus exploration on promising regions.

Discovered Configuration: After 100 trials totalling approximately 350 compute-hours, NAS identified an initial configuration achieving strong performance across the fitness objectives. Key architectural findings include hidden dimension 256 with 2 layers balancing expressiveness and generalisation (larger architectures overfitted to training mobility patterns, whilst smaller networks exhibited insufficient capacity for the state space), and aggressive exploration decay prioritising early exploitation of predicted trajectories. The DQN hyperparameters discovered through NAS are reported in Table 6.1, whilst the reward weights were subsequently refined through the targeted search procedure in Section 6.1.3.

The Transformer architecture, including depth selection, was optimised through a separate search procedure validated in Section 6.1.2.

4.2.6 Training and Execution Algorithm

Algorithm 2 presents DARA's training loop over N_{episodes} transmission sessions. States s_t comprise the 18-dimensional vector defined in Equation 4.2, formed by concatenating normalised current metrics, predicted deltas at mid-horizon and full-horizon, previous actions, and congestion flags derived from frozen Transformer predictions. Experience tuples (s_t, a_t, r_t, s_{t+1}) are collected asynchronously into replay buffer D with capacity C. Once $|D| \geq B$, the DQN samples mini-batches $\mathcal{B}$ to compute temporal-difference targets $y_j = r_j + \gamma \cdot \max_{a''} Q_{\theta'}(s'_j, a'')$ using the target network, then updates the policy network via gradient descent on loss $\mathcal{L} = (1/|\mathcal{B}|) \sum_j (Q_\theta(s_j, a_j) - y_j)^2$. The target network performs soft updates $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ for stability.

Exploration: The DQN implements ϵ-greedy exploration with exponentially-decaying $\epsilon = \epsilon_{\text{end}} + (\epsilon_{\text{start}} - \epsilon_{\text{end}}) \cdot \exp(-\text{steps}/\epsilon_{\text{decay}})$. With probability ϵ, random actions from Φ^N explore the state-action space; otherwise, $a_t = \arg\max_{a'} Q_\theta(s_t, a')$ exploits the learnt policy. Each action specifies CWND fractions $\{\phi_p(t)\}_{p \in P}$ for all N paths, directly controlling rate allocation executed by the kernelspace scheduler.

Execution: After action execution, new congestion metrics are collected and fed to the Transformer for updated predictions. The reward is computed via Equation 4.10. Episodes terminate upon transmission completion or timeout. Transitions stored in D enable offline policy refinement whilst the scheduler operates with ∼3ms inference latency within the 100ms control cycle, leaving substantial margin for system overhead and ensuring compatibility with the transient burst durations (100–700ms) characteristic of mobile environments.

After training, the frozen policy network and Transformer are deployed to the userspace controller. During execution, the 100ms control cycle (Transformer prediction followed by DQN action selection, completing in ∼3ms) produces CWND fraction updates that the kernelspace scheduler applies at ∼1ms packet dispatch granularity, enabling proactive exploitation of predicted burst opportunities in mobile environments.

Algorithm 2 DARA Training with Transformer-Informed State

Load pre-trained Transformer $\mathcal{M}$
Initialise policy network Q_θ, target network $Q_{\theta'}$ with $\theta' \leftarrow \theta$
Initialise replay buffer D with capacity C
for episode $i = 1$ to N_{episodes} **do**
    Collect current telemetry H_t for all paths $p \in P$
    Obtain predictions: $\hat{Z}_p \leftarrow \mathcal{M}(H_t)$ for all $p \in P$
    Construct state s_t via Equation 4.2
    while state s_t is not terminal or truncated **do**
        Compute ϵ from decay schedule
        if rand() $< \epsilon$ **then**
            Select random action $a_t \in \Phi^N$ ▷ Exploration
        else
            $a_t = \arg\max_{a'} Q_\theta(s_t, a')$ ▷ Exploitation
        end if
        Execute $a_t = \{\phi_p(t)\}_{p \in P}$ via character device
        Advance to $t{+}1$, collect H_{t+1}, predict $\hat{Z}_p(t{+}1)$
        Compute $r_t \leftarrow R(s_t, a_t)$ via Equation 4.10
        Construct s_{t+1} via Equation 4.2
        Store (s_t, a_t, r_t, s_{t+1}) in D
        if $|D| \geq B$ **then**
            Sample mini-batch $\mathcal{B}$ from D
            $y_j = r_j + \gamma \max_{a''} Q_{\theta'}(s'_j, a'')$ for each $(s_j, a_j, r_j, s'_j) \in \mathcal{B}$
            Update θ by minimising $\frac{1}{|\mathcal{B}|} \sum_j \mathcal{L}(Q_\theta(s_j, a_j), y_j)$
            $\theta' \leftarrow \tau\theta + (1{-}\tau)\theta'$
        end if
        $s_t \leftarrow s_{t+1}$
    end while
end for

4.3 Chapter Summary

This chapter has presented the design of two novel multipath scheduling frameworks that address the fundamental limitation of reactive schedulers in volatile heterogeneous networks: the inability to anticipate capacity variations before they manifest in congestion feedback.

The BPF scheduler establishes a prediction-based baseline through LSTM networks forecasting path utilisation and variability over 5-second horizons, enabling priority-based path selection that reduces HoL blocking relative to purely reactive approaches such as CPF. BPF's simplicity, offline-trained prediction with binary path switching, confers practical advantages including no online learning phase and minimal computational overhead,

but its coarse $\phi_i \in \{0, 1\}$ scheduling granularity cannot proportionally exploit heterogeneous capacity across multiple usable paths, and its 5-second decision window prevents adaptation to sub-second transient bursts characteristic of high-mobility environments.

DARA extends the predictive scheduling paradigm through three key innovations. First, a Transformer-based prediction model forecasts per-path CWND and SRTT evolution over a 500 ms horizon spanning five prediction steps, achieving lower prediction error than LSTM and linear baselines whilst maintaining inference latency within the 100 ms control budget. Second, a DQN policy network maps an 18-dimensional state, comprising current observations, normalised prediction deltas, and congestion flags, to discrete CWND fraction actions from a 5-level set $\Phi = \{30, 47, 65, 82, 100\}\%$, enabling graduated rate control through multiplicative CWND limiting that operates above the congestion control layer. Third, a six-component reward function balances throughput maximisation, congestion-induced delay avoidance, quality sustainability, preemptive adaptation, stability, and idleness prevention, with weights optimised through NAS-based Optuna search.

The kernel–userspace integration architecture, mediated by a custom character device driver, enables millisecond-granularity state collection and $\sim$1 ms packet-level scheduling whilst confining computationally intensive Transformer and DQN inference to a decoupled 100 ms control cycle. The complete training algorithm integrates pre-trained Transformer predictions with experience replay-based DQN policy learning, producing a system that occupies less than 3% of the control cycle budget. Collectively, these components establish the architectural foundation for predictive rate allocation that is empirically evaluated in Chapters 5 and 6.

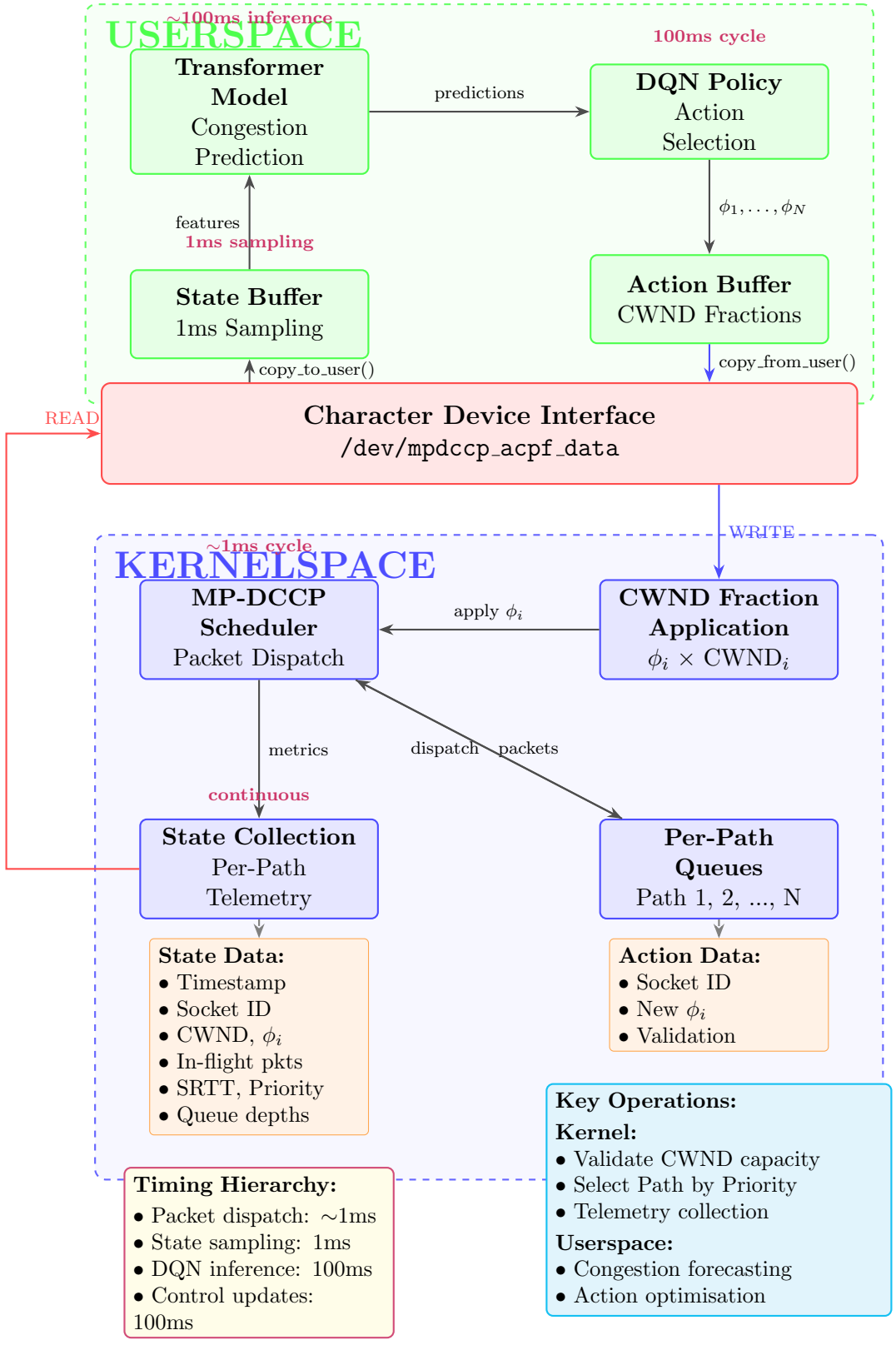

Figure 4.5: DARA system architecture showing the bidirectional data flow between kernelspace and userspace components. The character device interface bridges MP-DCCP's fast packet scheduling (~1ms cycle) with DARA's slower ML inference pipeline (100ms cycle). READ operations transfer per-path telemetry upward for congestion prediction and action selection, whilst WRITE operations apply computed CWND fractions downward to control per-path transmission rates.

Chapter 5

Testing Methodology

This chapter describes the experimental infrastructure and evaluation methodology employed to validate the proposed scheduling frameworks. Section 5.1 justifies the selection of MP-DCCP as the multipath transport protocol and details implementation considerations. Section 5.2 presents the Mininet-based testbed architecture and its validation against hardware implementations. Section 5.3 describes the Mahi-Mahi framework for trace-driven emulation of volatile wireless conditions. Section 5.4 defines the baseline and state-of-the-art schedulers against which DARA is evaluated. Section 5.5 details the parallel MP-QUIC testbed for Peekaboo and BLEST evaluation. Section 5.6 specifies the test scenarios spanning file transfer, streaming, and concurrent workload configurations.

5.1 MP-DCCP Protocol Selection and Implementation

ATSSS-HL implementation requires selection of an appropriate multipath protocol from the candidates analysed in Section 2.1.3. The architectural differences summarised in Table 2.1, particularly regarding reliability semantics, HoL blocking behaviour, and congestion control coupling, significantly impact scheduler performance in high-mobility scenarios. This section justifies the selection of MP-DCCP for experimental validation and details the implementation infrastructure developed to support scheduler integration.

5.1.1 Protocol Suitability for Scheduler Evaluation

The architectural differences between multipath protocols, summarised in Table 2.1, have profound implications for ATSSS-HL implementations requiring traffic-agnostic operation through encapsulation of arbitrary protocols. Encapsulating reliable protocols (TCP, QUIC) within reliable multipath protocols creates nested reliability mechanisms that conflict: the outer protocol's retransmissions interact with inner protocol timeouts, causing spurious retransmissions and throughput collapse. When an inner TCP connection experiences packet loss, it triggers fast retransmit and congestion window reduction. If the outer MP-TCP simultaneously experiences OFO delivery due to path heterogeneity, it may also reduce its congestion window, causing both layers to enter congestion avoidance simultaneously and collapsing throughput below single-path performance. MP-DCCP's unreliable delivery avoids these conflicts entirely, delegating reliability to encapsulated protocols as appropriate.

Furthermore, MP-DCCP's datagram orientation aligns naturally with per-packet scheduling decisions, whereas byte-stream (MP-TCP) and frame-based (MP-QUIC) models introduce segmentation complexity when distributing traffic across heterogeneous paths with differing Maximum Transmission Unit (MTU) constraints.

5.1.2 Protocol Selection Rationale

The selection of MP-DCCP for experimental validation reflects three key requirements. First, unreliable delivery avoids nested reliability conflicts when encapsulating TCP/QUIC traffic, preventing the spurious retransmission interactions that degrade MP-TCP tunnel performance. Second, elimination of mandatory HoL blocking-unlike MP-TCP and MP-QUIC which buffer OFO packets at connection or stream level respectively-ensures observed performance differences stem from scheduler logic rather than protocol-induced limitations. Third, kernel-level integration with mature Linux support, unavailable in experimental MP-QUIC implementations, enables production-grade evaluation with unmodified application traffic.

These characteristics prove essential for AI-driven rate allocation strategies seeking to

maximise instantaneous throughput by opportunistically utilising predicted capacity windows. When DARA's Transformer module predicts transient capacity availability on a secondary path based on historical CWND patterns, MP-DCCP permits immediate transmission without concern for subsequent packet arrival order or receiver buffer state. Packets transmitted during capacity bursts arrive out-of-order but are delivered immediately to the encapsulated protocol, which handles reordering according to its own semantics. In contrast, MP-TCP buffers such packets at the multipath layer until slower-path segments arrive, introducing latency spikes that negate burst exploitation benefits, whilst MP-QUIC introduces per-stream buffering delays that partially mitigate but do not eliminate the problem. This advantage proves critical in high-mobility scenarios where path characteristics fluctuate on sub-second timescales and burst opportunities are transient.

The protocol has a fully developed open-source implementation[1] with complete Linux kernel support for the multipath option set, including connection management (`MP_JOIN`, `MP_KEY`, `MP_CLOSE`), path management (`MP_ADDADDR`, `MP_REMOVEADDR`, `MP_PRIO`), and sequencing mechanisms (`MP_SEQ`, `MP_HMAC`). This maturity enables focus on scheduler design rather than protocol implementation, whilst kernel-level integration ensures realistic performance characteristics under production-grade networking stacks.

5.1.3 Path Management and Scheduler Interface

MP-DCCP's option-based architecture provides explicit interfaces for scheduler control that complement the implicit signals available through congestion control feedback. The `MP_PRIO` option enables dynamic path prioritisation through a 4-bit priority field, allowing schedulers to signal preferred paths without requiring connection re-establishment. The `MP_RTT` option exchanges four RTT estimation types (raw, minimum, maximum, smoothed) with age timestamps, enabling schedulers to detect path quality degradation before it manifests as packet loss. These explicit signals complement implicit path state derived from congestion control, providing a richer state space for learning-based schedulers than protocols that rely solely on ACK-based feedback.

[1] https://github.com/telekom/mp-dccp/tree/mpdccp_v03_k4.14

The protocol's HMAC-based authentication prevents unauthorised path injection attacks whilst maintaining computational efficiency. Each `MP_ADDADDR` announcement includes a 20-byte truncated HMAC-SHA256 computed over the address identifier, nonce, IP address, and port using keys established during initial handshake. This authentication model scales efficiently to scenarios with frequent path changes-common in high-mobility deployments-as it avoids the per-packet encryption overhead of QUIC-based approaches whilst providing stronger security than TCP's option space.

5.1.4 Packet Analysis Infrastructure

Protocol development and network troubleshooting require real-time packet dissection capabilities to validate implementation correctness, diagnose protocol violations, and extract QoS metrics. Wireshark lacked native support for MP-DCCP's extended option space defined in the IETF draft [28]. A protocol dissector was developed extending Wireshark's existing DCCP implementation to parse MP-DCCP's multipath signalling, detailed in Table 2.2.

The dissector implements hierarchical parsing: upon detecting option type 46, it extracts the sub-option identifier and delegates to sub-option-specific handlers. Key parsing features include validation of fixed and variable-length structures, support for both IPv4 and IPv6 address formats in `MP_ADDADDR` (12 bytes for IPv4, 24 bytes for IPv6, plus 2 optional bytes for port numbers), HMAC-SHA256 authentication field extraction (20-byte truncated values in the 23-byte `MP_HMAC` option), and 48-bit sequence number handling in `MP_SEQ` for out-of-order detection.

The dissector validates option lengths against protocol specifications, generates expert information warnings for malformed packets, and enforces HMAC association rules where `MP_HMAC` options must immediately follow authenticated options (`MP_JOIN`, `MP_ADDADDR`, `MP_REMOVEADDR`) to prevent ambiguity in multi-option packets. This validation capability proved essential during scheduler development, enabling real-time detection of protocol violations such as missing HMAC authentication or incorrect sequence number progression. The dissector extracts per-path CWND values, RTT measurements, and packet loss

events, facilitating automated analysis of scheduler behaviour under controlled experimental conditions. This implementation was merged into Wireshark's mainline repository in December 2022[2], enabling MP-DCCP traffic analysis for the research community.

5.1.5 Implementation Considerations for Scheduler Integration

The Linux kernel MP-DCCP implementation exposes scheduler control through socket options and netlink interfaces. Per-path transmission quotas are configured via `setsockopt()` calls with path identifiers derived from address tuples. The scheduler receives path state updates through netlink notifications containing CWND changes, RTT measurements from `MP_RTT` exchanges, and path status modifications from `MP_PRIO` signals. This event-driven architecture enables asynchronous operation: the DQN scheduler processes state updates in user-space whilst the kernel handles packet transmission according to configured quotas, avoiding context-switching overhead on the critical transmission path.
For the DARA framework, the Transformer prediction module consumes historical CWND and SRTT time-series extracted from kernel notifications, generating forecasts that augment the DQN's state space. The DQN outputs per-path utilisation fractions, which the kernel translates to transmission quotas by multiplying against current CWND values. This decomposition preserves congestion control correctness-the kernel's CWND updates reflect actual network feedback-whilst enabling sophisticated scheduling logic in user-space Python implementations. The separation of prediction (offline Transformer model) and decision-making (online DQN agent) from packet transmission (kernel) ensures that the computational overhead of neural network inference does not impact real-time packet forwarding performance, a critical requirement for deployment in production heterogeneous wireless networks where packet processing delays would exacerbate OFO delivery and compromise the scheduler's ability to exploit transient burst capacity.

[2]https://gitlab.com/wireshark/wireshark/-/merge_requests/9013

5.2 Mininet Architecture and Testbed Selection

5.2.1 Testbed Requirements and Selection Rationale

Evaluating multipath schedulers in volatile heterogeneous networks requires a testbed supporting: (1) realistic wireless path emulation with dynamic capacity variations, (2) true multipath protocol implementations with per-path congestion control, (3) application-level traffic generation for diverse workloads, (4) deterministic reproducibility for controlled experimentation, and (5) computational efficiency enabling extensive parameter sweeps.

Table 5.2.1 compares candidate frameworks against project requirements. Physical testbeds offer highest fidelity but lack scalability and controllability—the available hardware testbed (detailed in Section 5.2.3) comprises two laptops and four embedded Linux devices with static link parameter configuration, precluding continuous trace-driven capacity variation and large-scale parameter sweeps. Discrete-event simulators like OMNeT++[3] and NS-3[4] provide scalability but sacrifice implementation realism: OMNeT++ requires the INET framework for multipath support with limited protocol fidelity, whilst NS-3's documentation for loading custom kernel modules (essential for MP-DCCP) has not been maintained since 2013, with deprecated dependencies preventing integration of Deutsche Telekom's out-of-tree kernel implementation.

Framework	**MP**	**Kernel**	**Apps**	**Traces**	**Repro.**	**Scale**
Physical	✓	✓	✓	×	×	×
OMNeT++	Partial	×	×	✓	✓	✓
NS-3	Limited	Partial	×	✓	✓	✓
Mininet	✓	✓	✓	✓	✓	Partial

Note: MP = genuine multipath protocol support; Kernel = unmodified Linux networking stack; Apps = application-level traffic; Traces = deterministic wireless capacity replay; Repro. = reproducibility; Scale = scalability to many nodes.

Table 5.1: Comparison of testbed environments for multipath scheduler evaluation

Mininet[5] emerges as the optimal compromise, providing network emulation—running real

[3] https://omnetpp.org/
[4] https://www.nsnam.org/
[5] https://mininet.org/

kernel, switch, and application code using lightweight virtualisation (network namespaces). This architecture enables execution of Deutsche Telekom's out-of-tree MP-DCCP implementation [37] without modification, ensuring that scheduler evaluation reflects actual protocol behaviour. Unlike simulators requiring protocol reimplementation, Mininet guarantees observed performance stems from genuine multipath transport dynamics. Integration with Mahi-Mahi for trace-driven emulation (described in Section 5.3) enables the controlled volatility testing unavailable in hardware. The platform's adoption in recent multipath scheduling research [102, 87] further validates its suitability.

5.2.2 Experimental Topology

Figure 5.1 illustrates the Mininet topology replicating the hardware testbed architecture. The client host (h1) represents the UE with dual interfaces connecting via routers r1 (cellular) and r2 (Wi-Fi), converging at aggregation router r3 before reaching server h2.

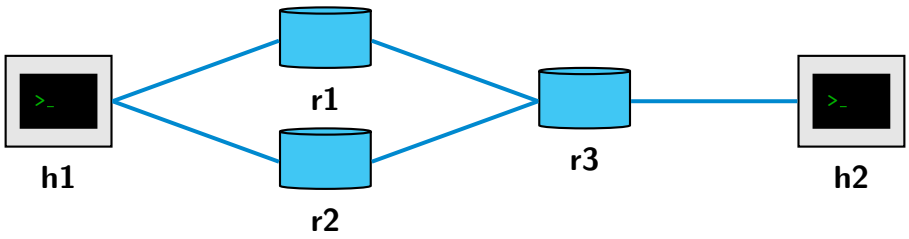

Figure 5.1: Mininet experimental topology replicating hardware testbed architecture. Paths h1–r1 and h1–r2 provide heterogeneous connectivity to aggregation point r3, with Mahi-Mahi shells applying trace-driven capacity variations.

Each path operates as an independent MP-DCCP subflow with separate congestion control state, sequence space, and acknowledgement mechanisms. Routers r1 and r2 function as network namespaces with traffic control (`tc`) configuration, applying bandwidth, delay, and loss characteristics. The aggregation router r3 performs simple forwarding, ensuring observed latency reflects end-to-end path characteristics rather than testbed artefacts. Application-level traffic employs unmodified software: `curl` for file transfers, YouTube's HTML5 player for streaming, and HLS.js for live video. This ensures observed QoS metrics reflect genuine user experience rather than synthetic traffic approximations.

5.2.3 Comparison with Hardware Testbed

The software implementation was validated against the hardware testbed shown in Figure 5.2, developed by the protocol creators [103]. The hardware configuration comprises two laptops and four embedded Linux devices: one device functions as MP-DCCP server, two provide traffic control routing, and one acts as MP-DCCP client (connected to the client laptop). Both implementations support configurable schedulers, reordering algorithms, congestion control, and path priorities, with packet capture via custom Wireshark[6] dissectors.

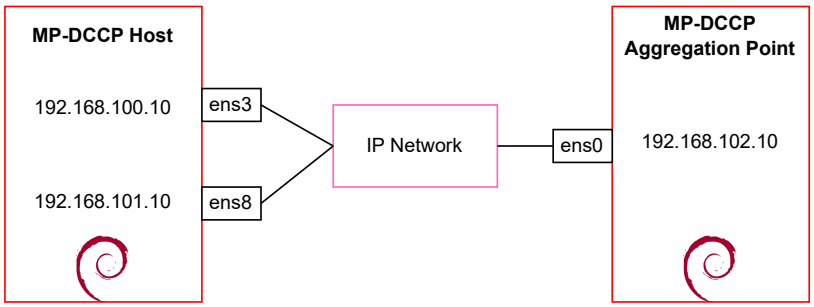

Figure 5.2: MP-DCCP hardware testbed architecture[7]

5.2.3.1 Variable Loss Testing

To validate CCA operation, tests used CPF scheduling with equal path priorities, 10 ms latency per link, 16 Mbps bandwidth per path, and 32 Mbps aggregate transmission rate. Congestion Control Identifier 2 (CCID2), was deployed with three intervals: 0–30 s at 0% loss, 30–60 s at 5% loss on link one, and 60–90 s at 5% loss on link two. Both testbeds exhibited expected behaviour: path aggregation during the first interval and prioritisation of loss-free links during subsequent intervals. The software implementation (Figure 5.3) demonstrated lower volatility than hardware (Figure 5.4), consistent with established findings on software testbed fidelity [104].

[6] https://www.wireshark.org/
[7] Source: https://multipath-dccp.org/configure_route.html

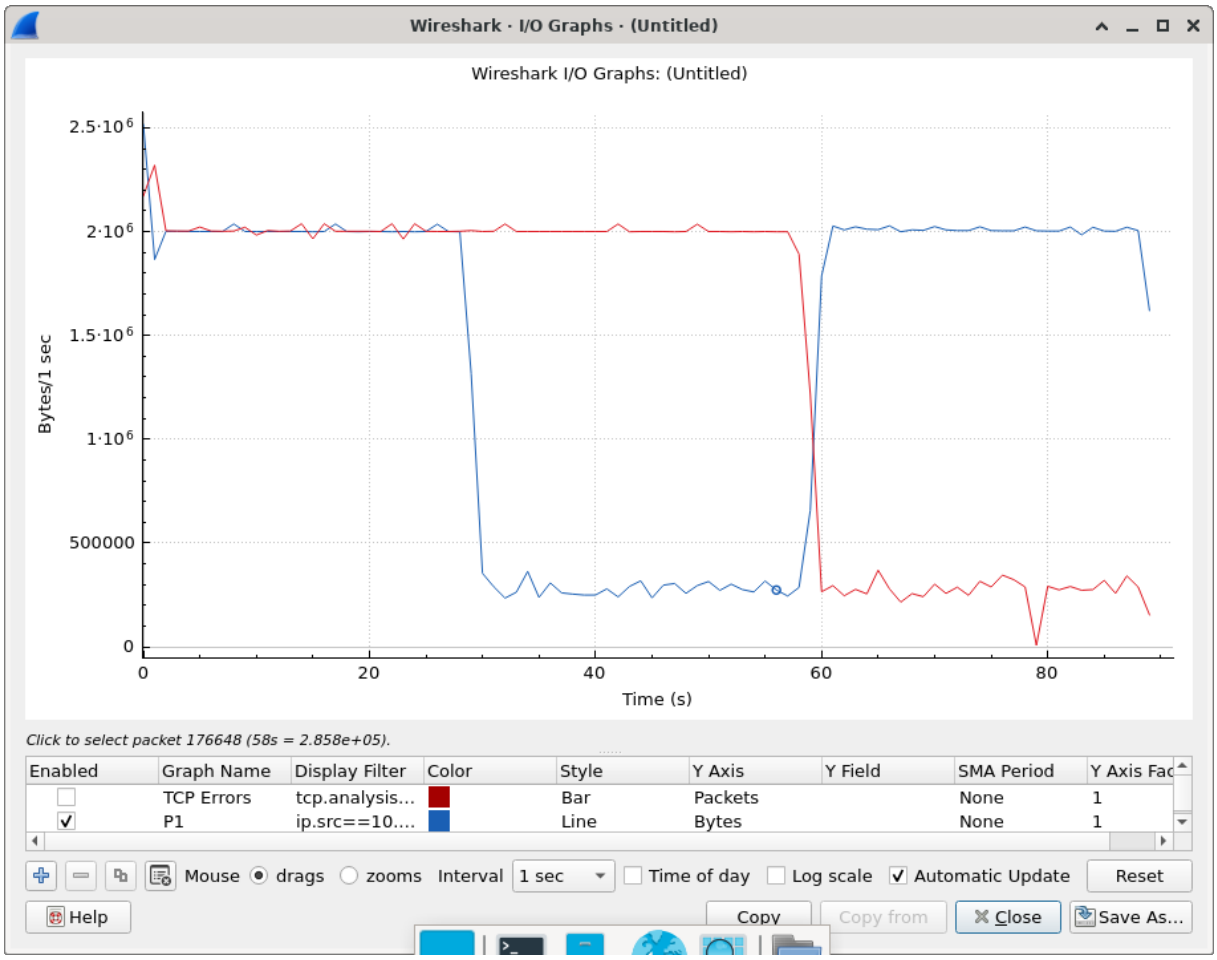

Figure 5.3: Software testbed—throughput under variable loss

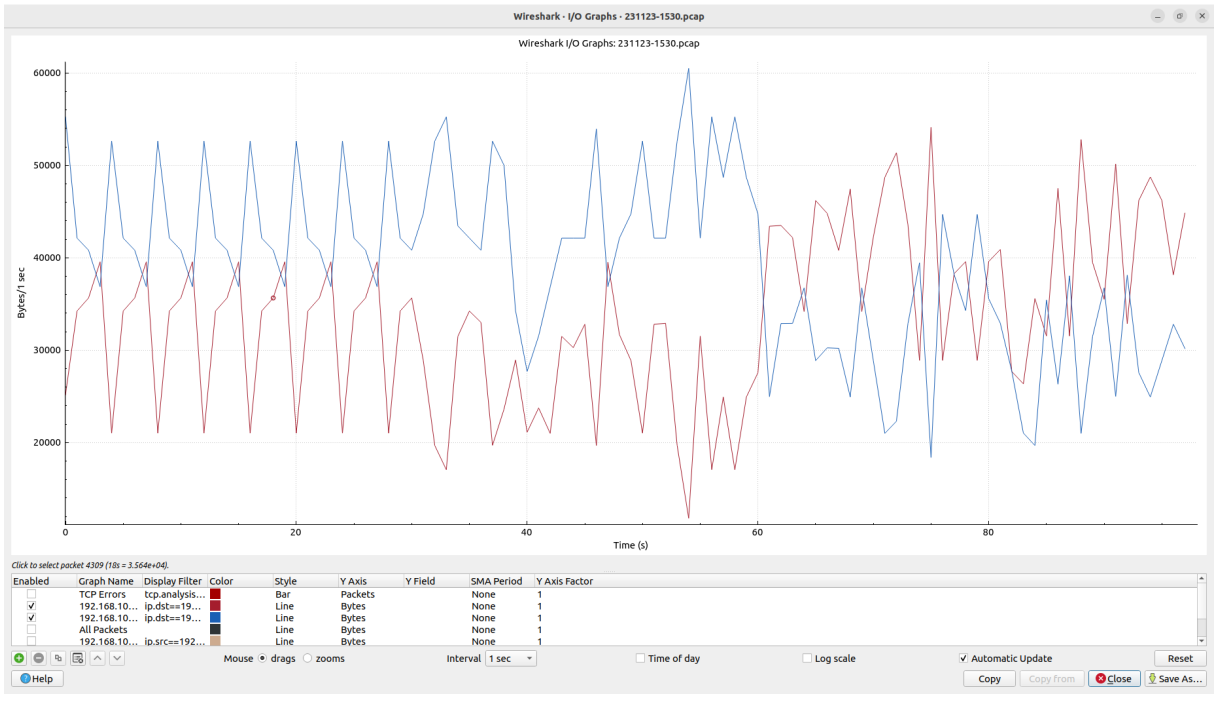

Figure 5.4: Hardware testbed—throughput under variable loss

5.2.3.2 Variable Traffic Testing

Congruent throughput trends across testbeds validate traffic control, transmission rates, and scheduler decision-making, confirming correct software protocol operation. Two scenarios tested CPF with equal priorities. In the uncongested case (Figures 5.5 and 5.6), 1 Mbps transmission over 90 s used 2 Mbps/10 ms links (0–30 s), degraded one link to 100 kbps/100 ms (30–60 s), then swapped link qualities (60–90 s). The congested case (Figures 5.7 and 5.8) transmitted at 4 Mbps with 2 Mbps/10 ms links (0–30 s), degraded one link to 1 Mbps/100 ms (30–60 s), then swapped (60–90 s). Results demonstrate agreement between implementations with minor software volatility, confirming independent treatment of scheduling and congestion control.

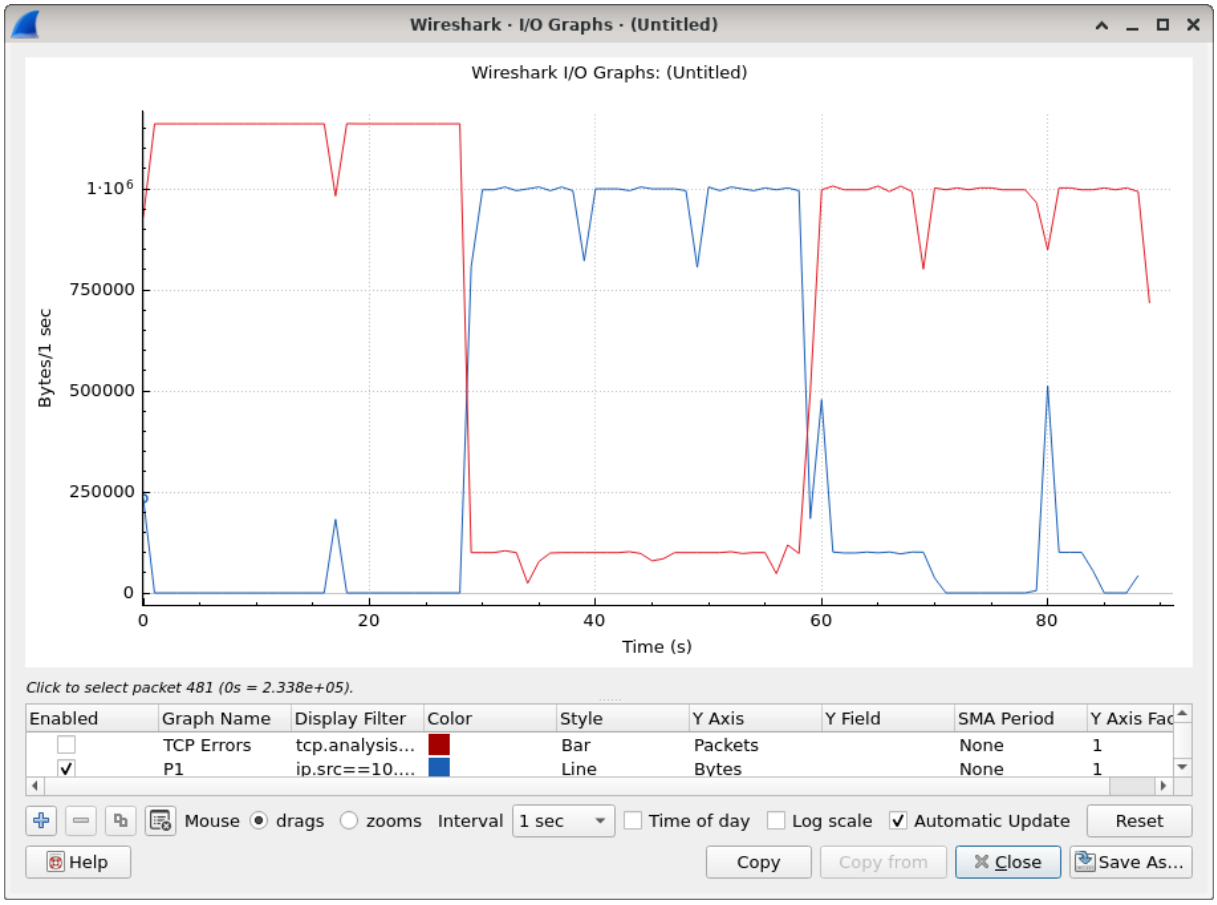

Figure 5.5: Software testbed—CPF uncongested

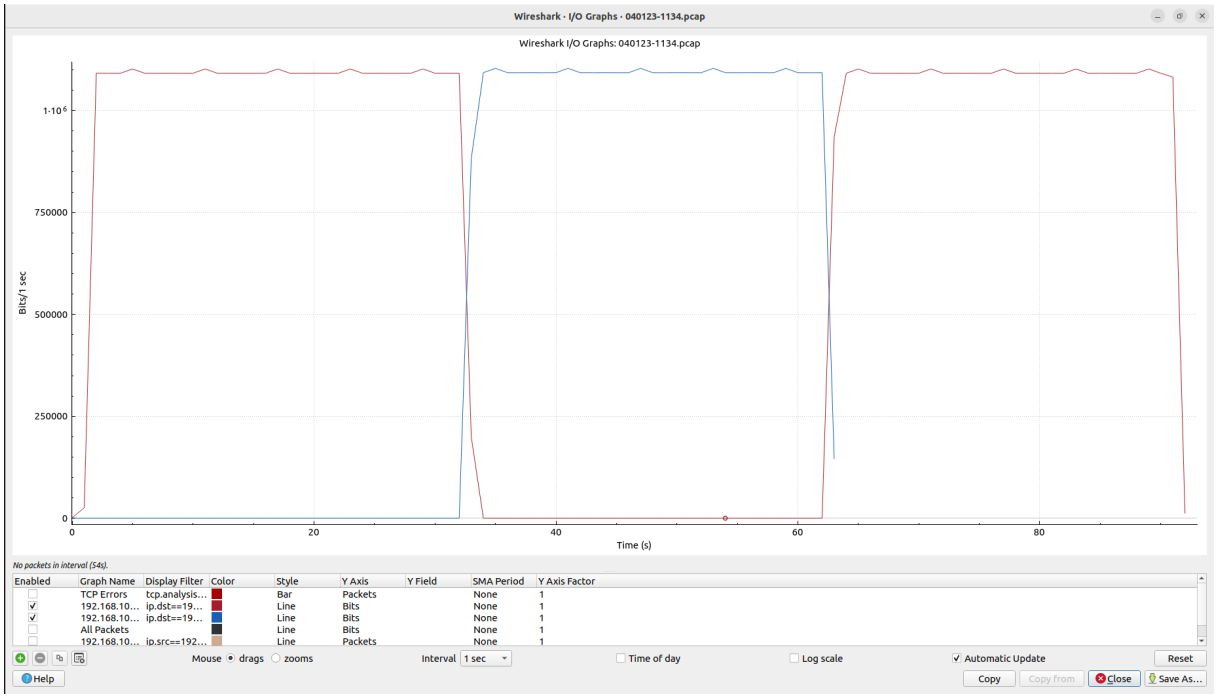

Figure 5.6: Hardware testbed—CPF uncongested

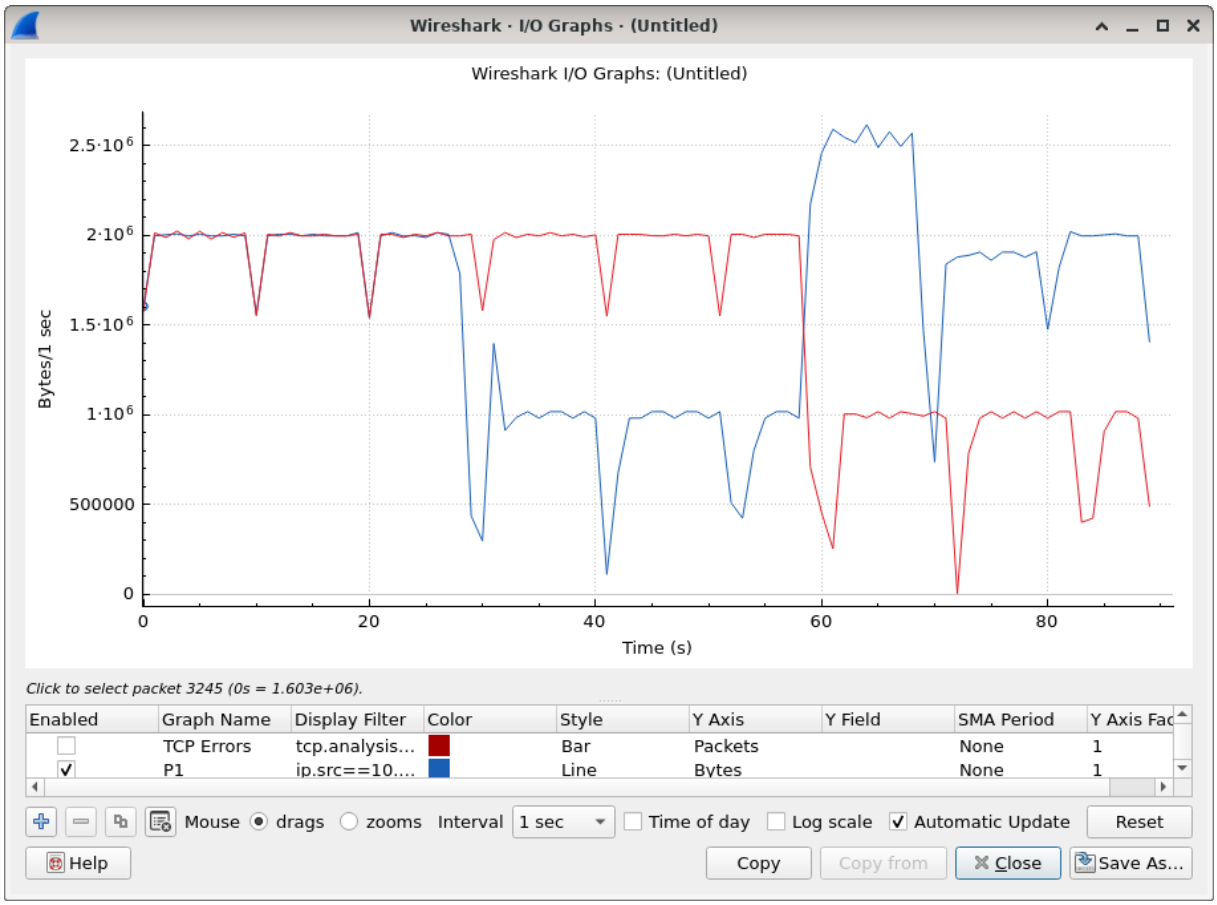

Figure 5.7: Software testbed—CPF congested

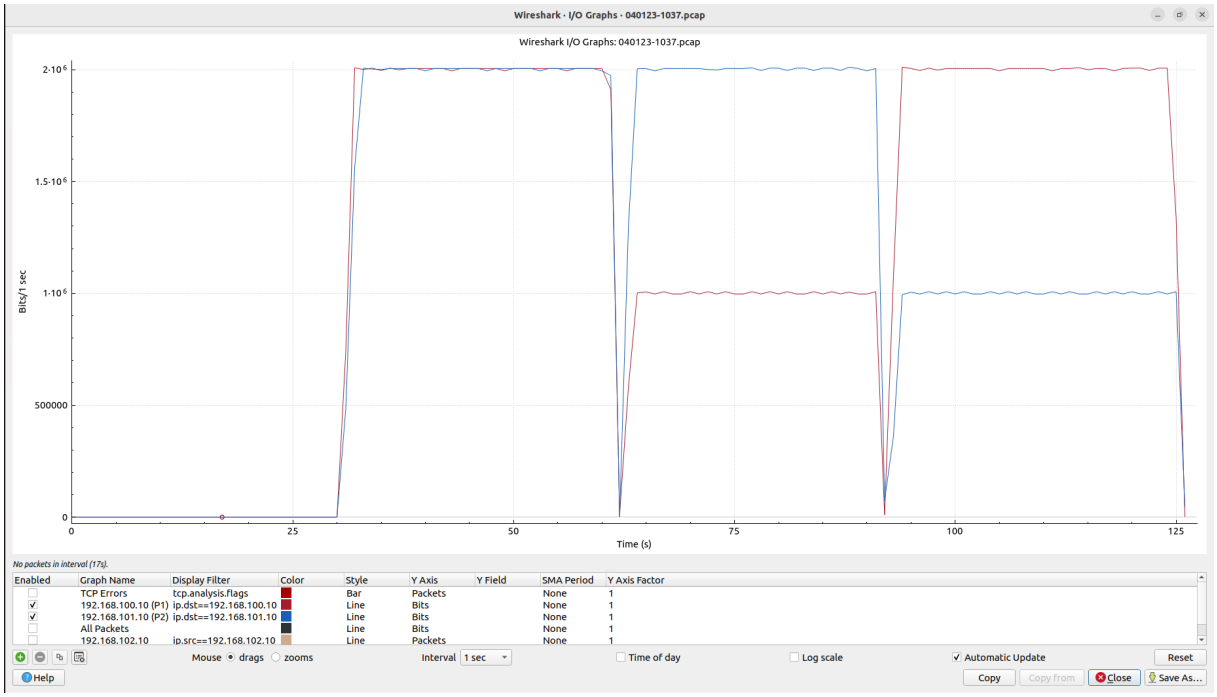

Figure 5.8: Hardware testbed—CPF congested

5.3 Replaying Volatile Path Traffic

To rigorously evaluate scheduler performance under realistic volatile conditions, the Mahi-Mahi[8] framework for trace-driven network emulation is employed. Mahi-Mahi enables deterministic replay of actual cellular capacity variations by consuming packet-delivery traces—timestamped records of MTU-sized packet transmission opportunities captured from saturated links of moving users. Each trace file consists of timestamps (in milliseconds) indicating when the network can transmit one MTU-sized packet (1500 bytes), with each delivery opportunity wasted if no data is available at that instant. This trace-driven approach preserves the statistical properties of real wireless channels (including sub-second burst characteristics and path asymmetry) whilst eliminating experimental non-determinism inherent in live network testing.

5.3.1 Trace Generation Methodology

The publicly available Mahi-Mahi traces were originally generated using the Saturatr methodology, which captures realistic capacity variations by saturating both uplink and

[8] http://mahimahi.mit.edu/

downlink radio links of a UE. As illustrated in Figure 5.9, the trace capture process involves a UE transmitting data packets with embedded timestamps to a server. Upon receiving acknowledgements, the UE computes RTTs and adjusts its congestion window accordingly: when $\text{RTT} < \text{RTT}_{\text{prev}}$, the window increases ($\text{CW} = \text{CW} + 1$) to saturate the link; when $\text{RTT} \geq \text{RTT}_{\text{prev}}$, it decreases ($\text{CW} = \max(\text{CW}/2, 1)$) to avoid congestion collapse. This bidirectional saturation protocol ensures that the resulting traces capture maximum achievable throughput under real network conditions, including handovers, signal fluctuations, and mobility effects. The output consists of Transmission Opportunitys (txops) files where each line represents a timestamp when one MTU of data can be delivered.

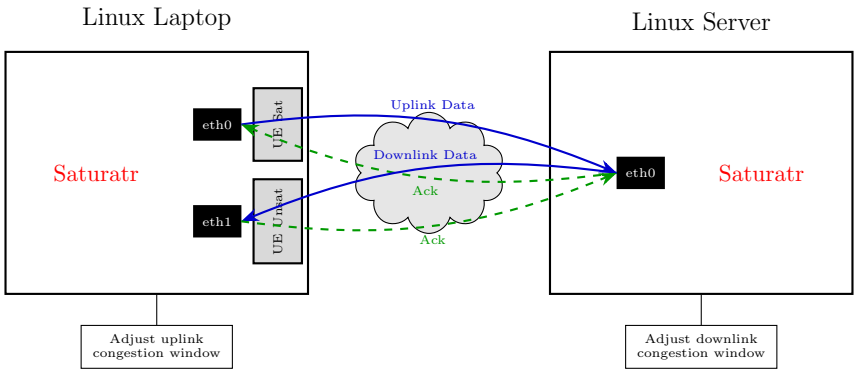

Figure 5.9: Trace generation using the Saturatr tool to measure link capacity by adjusting congestion windows based on measured RTT.

5.3.2 Mahi-Mahi Scheduling Mechanism

The core scheduling logic of Mahi-Mahi is illustrated in Figure 5.10. Packets from the application layer arrive into a queue and are held until the simulation time matches a timestamp in the trace file. At each delivery opportunity, one MTU worth of data is dequeued and transmitted; if no data is available, the opportunity is wasted and cannot be recovered. This strict temporal enforcement ensures that the emulated network exhibits the same capacity variations as the original cellular environment where the trace was captured.

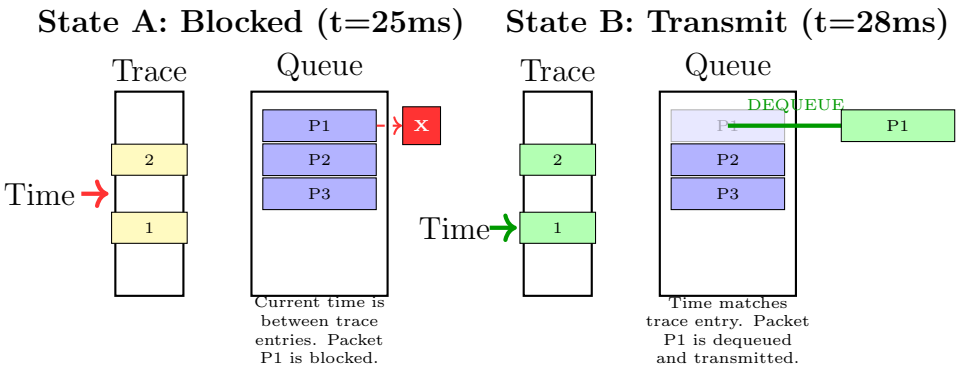

Figure 5.10: The core logic of the Mahi-Mahi scheduler. Packets are blocked until the simulation time matches a timestamp in the trace file, at which point one MTU of data is released.

5.3.3 Multipath Emulation Architecture

The experimental setup, illustrated in Figure 5.11, implements a multipath network architecture with two independent Mahi-Mahi instances emulating distinct network paths. The client and server nodes run MP-DCCP network stacks with schedulers, communicating through two separate Mahi-Mahi instances (Mahi-Mahi 1 and Mahi-Mahi 2) that feed into routers before reaching the destination. Each Mahi-Mahi instance maintains its own queue and trace file, enabling precise emulation of heterogeneous path characteristics. By replaying these traces through Mininet's virtual interfaces, the last-mile bottleneck dynamics that dominate end-to-end performance in mobile scenarios are emulated, creating a controlled testbed where scheduler algorithms face the same volatility patterns encountered in production deployments.

5.3.4 Trace Characteristics and Network Conditions

The standard distribution curves of the five publicly available Mahi-Mahi cellular traces[9] used in this evaluation are shown in Figure 5.12. These traces, captured from moving vehicles experiencing continuous network handovers and signal fluctuations, exhibit standard deviations exceeding 75% of mean throughput—representing the extreme variability that triggers frequent horizontal handovers between paths, increased OFO packet delivery, reordering delays, HoL blocking, and consequent goodput degradation. Orange 4G

[9]https://github.com/joelromanky/mahimahi/tree/master/traces

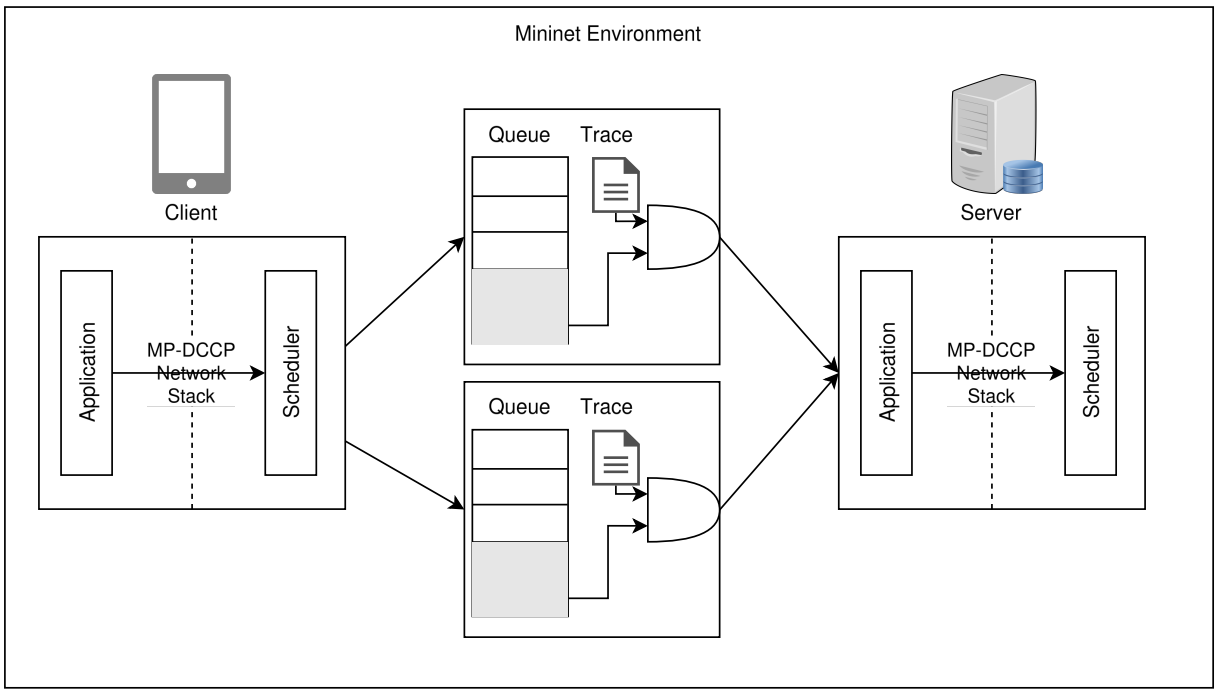

Figure 5.11: Multipath network architecture with two independent Mahi-Mahi instances emulating distinct network paths.

Highway demonstrates the highest variance ($\mu = 43.7$ Mbps, $\sigma = 31.5$ Mbps), whilst T-Mobile UMTS Driving shows more stable but lower throughput ($\mu = 2.4$ Mbps, $\sigma = 1.6$ Mbps). Such high-mobility vehicular scenarios exemplify the volatile conditions that reactive schedulers struggle to handle and that motivate DARA's predictive architecture.

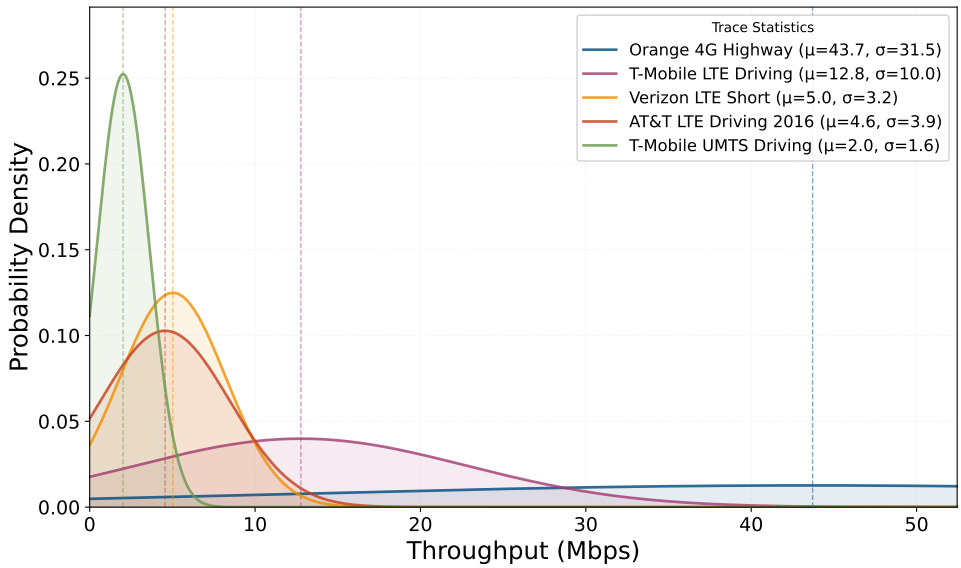

Figure 5.12: Path throughput normal distribution showing variance characteristics of publicly available Mahi-Mahi cellular traces.

5.3.5 Path Behaviour and Multipath Heterogeneity

The time-series analysis in Figure 5.13 reveals the dynamic nature of multipath communication under trace-driven emulation. Path 1 (blue) and Path 2 (red) exhibit asynchronous throughput variations, with several critical events marked: at approximately 60 seconds, Path 2 experiences a handover causing temporary performance degradation whilst Path 1 maintains capacity; around 140 seconds, Path 2 suffers severe connectivity issues (labelled "Path 2 Bad Connection") requiring traffic migration; and near 230 seconds, Path 2 completely fails (labelled "Path 2 Lost Connection") whilst Path 1 recovers to good connection quality (labelled "Path 1 Good Connection"). These events—separated by periods of normal operation—highlight the fundamental scheduler challenge: maintaining optimal packet distribution across paths with unpredictable, independent capacity fluctuations that can range from complete outages to peak throughput within seconds.

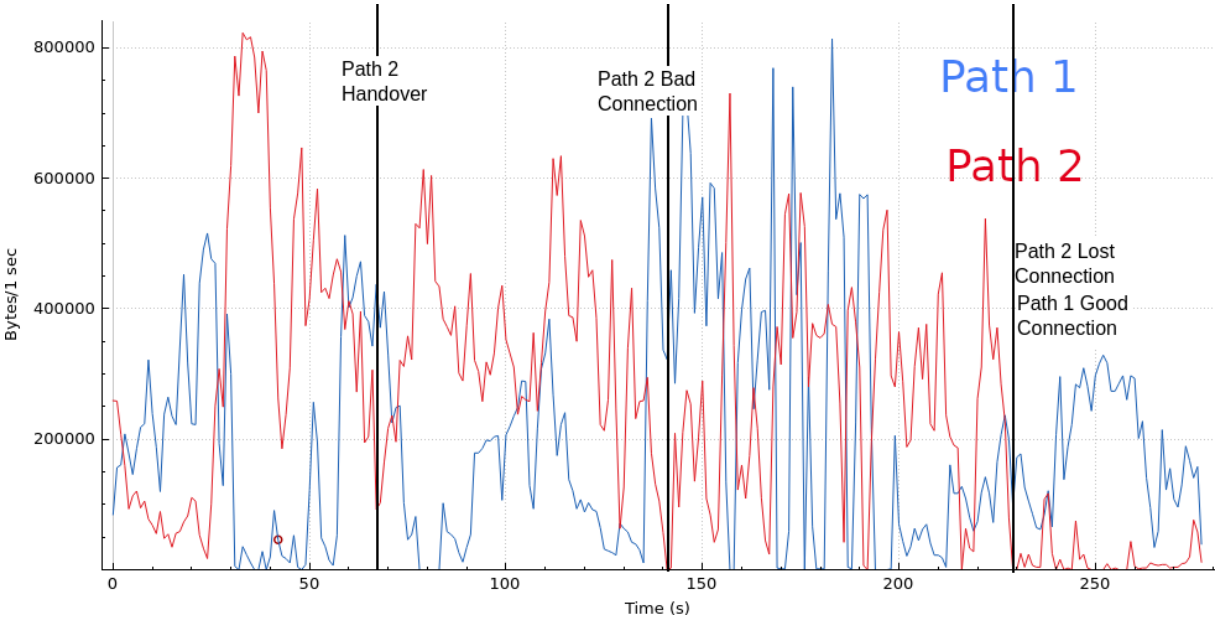

Figure 5.13: Time-series analysis of Path 1 (blue) and Path 2 (red) throughput variations during multipath communication, showing critical events including handovers, connection degradation, and path failures.

The heterogeneous nature of the traces creates path asymmetry where different trace files on each path produce varying capacity patterns, as illustrated in Figure 5.14. When Path 1 experiences high capacity (shown in green) whilst Path 2 suffers low capacity (shown in red), packets sent on different paths arrive at the receiver at drastically different times.

This temporal disparity, combined with Mahi-Mahi's strict enforcement of delivery opportunities, inevitably leads to out-of-order packet arrivals at the receiver, as demonstrated in the figure where packet 7 from the slower Path 1 arrives after packet 8 from the faster Path 2. This path asymmetry creates the precise conditions under which naive schedulers fail and predictive algorithms like DARA can demonstrate their advantages.

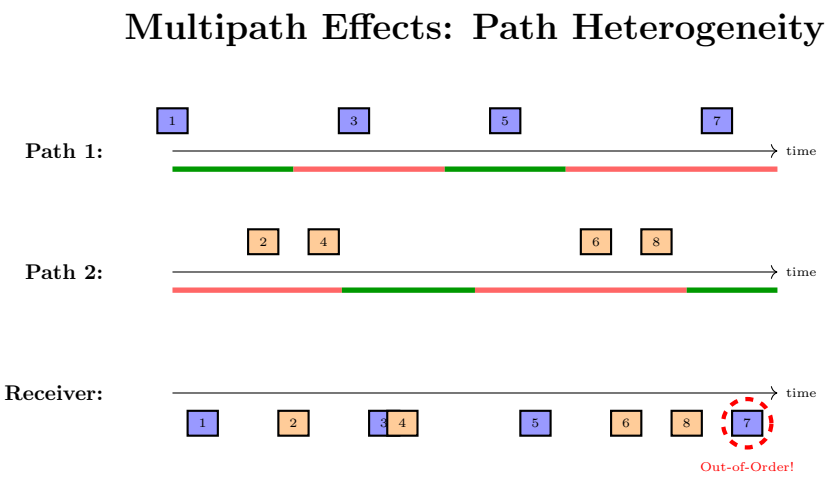

Figure 5.14: Different trace files on each path create heterogeneous capacity (green=fast, red=slow), leading to out-of-order packet arrivals at the receiver.

5.4 Baseline and Comparison Schedulers

DARA is evaluated against multipath schedulers representing the evolution of scheduling paradigms, detailed in Section 3.1.7. Table 3.1 summarises the comparative characteristics of baseline schedulers. DARA extends this comparison as a Transformer-informed DRL framework with $O(L \times d^2)$ complexity (where L denotes Transformer layers and d the feature dimension), employing multi-metric path awareness and a 500ms prediction horizon.

The selected schedulers enable systematic comparison across design philosophies. RR (Section 3.1.7.2) represents the reactive extreme—no path estimation, no congestion awareness—against which DARA's predictive architecture is contrasted. CPF and ACPF (Sections 3.1.7.3–3.1.7.5) optimise cost by preferentially routing over Wi-Fi, whereas DARA balances throughput, delay, and fairness through multi-objective optimisation (fairness in this work is defined in Section 4.2.4.1). OTIAS (Section 3.1.7.4) and BLEST

(Section 3.1.7.6) employ heuristic estimation but operate reactively on current state, unlike DARA's Transformer-based forecasting which anticipates path degradation before it manifests.

Learning-based schedulers offer closer comparison. BPF (Section 3.1.7.7) uses supervised LSTM prediction with a fixed 5-second horizon and binary path selection, whereas DARA employs DRL with continuous per-packet decisions and self-attention modelling both temporal evolution and cross-path interactions. Peekaboo (Section 3.1.7.8) formulates scheduling as a contextual MAB problem with online adaptation, though its memoryless formulation cannot exploit long-range temporal patterns. DARA's multi-layer Transformer attention jointly models temporal evolution and cross-path interactions, with DQN training encouraging policies that maximise cumulative reward rather than myopic per-packet optimisation. Peekaboo's MP-QUIC implementation[10] does not support packet encapsulation; consequently, comparison is limited to file download completion times rather than fine-grained streaming metrics requiring packet-level instrumentation.

Single Path (Sections 3.1.7.1) serves as fundamental baseline: Single Path establishes the lower bound that multipath schedulers should exceed.

All schedulers operate under identical experimental conditions, ensuring performance differences arise solely from scheduling decisions. Learning-based methods are evaluated post-convergence to reflect steady-state performance.

5.5 MP-QUIC Testbed for Peekaboo and BLEST

Peekaboo and BLEST operate within an MP-QUIC implementation framework that differs architecturally from the MP-DCCP testbed used for other schedulers. This necessitates a parallel evaluation environment to assess these algorithms under equivalent network conditions whilst respecting protocol-specific constraints.

[10] https://mosaicsimulamet.wordpress.com/peekaboo/

5.5.1 Implementation Environment

The evaluated MP-QUIC implementation operates in user-space using Go 1.10.3, compiled from source on Ubuntu 18.04 LTS. This user-space architecture provides rapid prototyping capabilities and cross-platform portability but constrains packet-level instrumentation: the framework lacks transparent encapsulation support required for ATSSS-style operation where arbitrary application protocols tunnel through the multipath layer.

The implementation employs Docker containerisation to provide isolated network namespace management whilst enabling Mininet integration for topology emulation. Control plane operations utilise Jupyter Notebook interfaces for experiment orchestration, contrasting with the shell-script automation employed in MP-DCCP evaluations. Despite these implementation differences, both testbeds employ identical Mahi-Mahi trace files for capacity emulation, ensuring that observed scheduler performance differences reflect algorithmic behaviour rather than testbed artefacts.

5.5.2 Evaluation Constraints and Adapted Methodology

The user-space MP-QUIC framework's architectural constraints preclude several evaluation scenarios available in the kernel-space MP-DCCP testbed. Application-layer protocols requiring deep packet inspection—specifically YouTube streaming with quality adaptation tracking and live video with HLS segment monitoring—cannot be transparently proxied through the MP-QUIC implementation. The framework provides native support for HTTP/3-based file transfers, enabling completion time measurement for bulk data transmission but preventing extraction of fine-grained streaming metrics (rebuffering events, quality switches, frame delivery latency).

Consequently, Peekaboo and BLEST evaluation employs the following test scenarios:

- **QUIC File Transfer (Section 5.6.5):** 150 MB downloads via MP-QUIC's HTTP/3 implementation, measuring completion time and aggregate throughput. This provides direct comparison with MP-DCCP file transfer results, though protocol-level differences (stream multiplexing vs. datagram delivery, per-stream flow control vs. connection-level congestion control) introduce semantic variations.

- **QUIC+TCP Concurrent Downloads (Section 5.6.6):** Simultaneous QUIC and HTTP/TCP file transfers testing scheduler behaviour under mixed-protocol competition, with per-flow throughput, delay, and completion time recorded independently.

- **YouTube/Live Streaming:** Not evaluated due to encapsulation limitations. Performance inferences for streaming workloads are drawn from file transfer behaviour under equivalent network conditions, with explicit acknowledgement of reduced fidelity compared to application-specific metrics.

The Mininet topology and Mahi-Mahi configuration replicate those used in MP-DCCP evaluations: dual-path architecture with independent trace-driven capacity variations, identical MTU (1500 bytes), and equivalent end-to-end topology (client—routers—aggregation point—server). This methodological consistency enables cross-protocol scheduler comparison despite implementation heterogeneity, with performance differences attributable to combined effects of scheduling algorithms and underlying transport protocol characteristics.

5.6 Tests

5.6.1 File Transfer

FTP file transfers utilise TCP with loss-based congestion control (typically NewReno or CUBIC). The lossless nature of TCP enables precise packet tracking for delay and jitter measurement, whilst greedy congestion control behaviour fully utilises available path capacity, providing clear differentiation of scheduler effectiveness through download times and aggregate throughput. Completion time for 150 MB file transfers is measured, recording per-packet delay distributions and effective throughput averaged over 100 ms windows.

5.6.2 YouTube Streaming

A YouTube video plays back over trace-emulated network conditions with quality and buffering state recorded every 100 ms. Playback quality ranges from 144p to 4K resolution, whilst buffering state captures whether the video is playing or paused awaiting frames. YouTube's adaptive streaming builds buffers intelligently based on network conditions; to isolate scheduler effects and prevent buffer masking of capacity variations, buffers are reset by performing 200-second seeks every 20 seconds. This aggressive seeking exposes sub-20s scheduler responsiveness to capacity changes.

Metrics include average resolution, buffering frequency and duration, quality switch rate, and time-to-first-frame. The bursty nature of adaptive streaming—requesting segments at startup rates then throttling—exercises scheduler burst-handling capabilities distinct from greedy FTP behaviour.

5.6.3 Combined File Transfer and YouTube Streaming

Simultaneous FTP download and YouTube playback creates competing traffic with distinct QoS requirements: FTP optimises for throughput whilst YouTube requires consistent bandwidth to prevent rebuffering. This scenario tests scheduler fairness and priority handling under heterogeneous traffic demands.

FTP completion time is measured alongside YouTube buffering events and resolution stability. Fair schedulers should achieve FTP throughput comparable to single-flow scenarios whilst maintaining YouTube playback quality similar to isolated streaming tests.

5.6.4 Live Video Streaming

Live streaming via HLS over TCP/QUIC operates with minimal buffering (approximately 0.5 s), transforming capacity mismatches into immediate rebuffering events. The constrained buffer exposes sub-second scheduler responsiveness that longer-buffered applications mask.

An HLS.js-based script captures metrics every 100 ms: bitrate, player state, quality level, initial delay, live edge distance (latency behind broadcast edge), and recovery time from

buffering events. Unlike VoD where buffering merely annoys users, live applications require sustained quality without rebuffering to maintain real-time engagement. Metrics emphasise latency (live edge distance) and stability (rebuffering frequency, quality switches) over raw throughput.

5.6.5 QUIC File Download Tests

QUIC file downloads employ encrypted transport preventing inner congestion control state observation; schedulers must optimise tunnel-layer metrics treating encapsulated QUIC as opaque payload. The MP-QUIC testbed generates 150 MB test files served via HTTP/3, with TLS key logging enabled for post-hoc traffic analysis. Packet captures record both UDP (QUIC) and DCCP (tunnel) traffic, enabling protocol-level performance attribution. Download completion times and per-packet delay distributions are measured. Comparison with TCP-based file transfers isolates transport protocol effects from scheduler behaviour, whilst matched network conditions (identical traces) enable attribution of performance differences to protocol interactions rather than environmental factors.

5.6.6 QUIC+TCP Simultaneous File Download Tests

Simultaneous QUIC and HTTP/TCP downloads test scheduler behaviour under competing transport protocols. A foreground QUIC flow and background TCP flow execute concurrently, with per-flow throughput, delay, and completion time recorded independently. This scenario evaluates scheduler protocol-neutrality: effective schedulers should allocate capacity based on path characteristics rather than favouring specific transport protocols.

Per-flow metrics enable assessment of flow starvation, where one protocol monopolises capacity at the expense of another. Aggregate throughput and per-flow completion times quantify overall efficiency and fairness characteristics. The test methodology employs packet capture with protocol-specific field extraction (QUIC packet numbers, TCP sequence numbers, DCCP tunnel sequences) to attribute performance to individual flows.

5.6.7 Congestion Control Configuration

All experiments employ BBR-like congestion control for MP-DCCP tunnel paths. This selection reflects practical requirements: BBR's predictable CWND patterns enable accurate Transformer predictions (achieving MAE of 74.97 packets for Path 1 and 374.03 for Path 2), whilst resilience to non-congestion losses maintains stable rate estimates during cellular handovers and interference events captured in evaluation traces. The CWND fraction mechanism (ϕ_i adjustment) assumes CWND represents capacity rather than buffer fill level—a property BBR satisfies through its bandwidth-delay product targeting but NewReno-like algorithms violate.

The nested congestion control interactions differ by application: FTP (TCP/BBR-like tunnel) creates loss-based/model-based interaction where tunnel probing phases may cause packet loss visible to end-to-end TCP; YouTube (BBR/BBR-like tunnel) creates model-based/model-based interaction where performance depends on RTT distribution between tunnel and end-to-end paths; QUIC (CUBIC/BBR-like tunnel) creates another loss-based/model-based scenario with encrypted transport opacity.

5.7 Chapter Summary

This chapter has established the experimental infrastructure and evaluation methodology for validating the proposed scheduling frameworks, addressing the critical requirement that scheduler comparison be conducted under controlled, reproducible, and realistic conditions.

The selection of MP-DCCP as the primary multipath transport protocol was justified on three grounds: unreliable delivery avoids nested reliability conflicts when encapsulating TCP and QUIC traffic; elimination of mandatory HoL blocking ensures observed performance differences stem from scheduler logic rather than protocol-induced limitations; and kernel-level integration with mature Linux support enables production-grade evaluation. A custom Wireshark dissector extending DCCP's parsing to MP-DCCP's multipath signalling options was developed and merged into Wireshark's mainline repository, providing

essential protocol analysis capabilities for scheduler development and validation.

The Mininet-based testbed was selected as the optimal compromise among candidate environments, offering genuine multipath protocol execution with unmodified Linux networking stacks, application-level traffic generation, and deterministic trace-driven replay, capabilities not simultaneously available in physical testbeds, OMNeT++, or NS-3. Validation against the hardware MP-DCCP testbed under variable-loss and variable-traffic scenarios confirmed congruent throughput trends and correct protocol operation, establishing that the software implementation faithfully reproduces hardware behaviour with only minor volatility reduction consistent with established emulation fidelity findings.

The Mahi-Mahi trace-driven emulation framework enables deterministic replay of real wireless capacity variations captured from mobile users through the Saturatr methodology. Five publicly available traces spanning stationary, walking, vehicular urban, vehicular suburban, and train scenarios provide a controlled volatility gradient for systematic evaluation. The multipath emulation architecture employs independent Mahi-Mahi link shells for each path, with heterogeneous trace pairing creating asymmetric conditions representative of cellular–WiFi aggregation.

The baseline and comparison scheduler suite spans six implementations: Single Path, Round Robin, CPF, OTIAS, ACPF, and BLEST, complemented by state-of-the-art learning-based schedulers Peekaboo and BLEST evaluated through a parallel MP-QUIC testbed. Six test scenarios, FTP file transfer, YouTube adaptive streaming, combined FTP and YouTube, live HLS streaming, QUIC file download, and concurrent QUIC+TCP transfer, cover the principal multipath application categories, enabling comprehensive assessment of scheduler performance across diverse workload characteristics. This methodology provides the foundation for the controlled burst validation and real-world trace evaluation presented in Chapter 6.

Chapter 6

Experimental Results and Discussion

This chapter presents comprehensive evaluation of DARA's predictive rate allocation mechanism through controlled synthetic scenarios and real-world trace-driven experiments. The analysis proceeds in two stages: first, design validation (§6.1) isolates the contribution of each architectural and hyperparameter decision through ablation studies conducted on the training dataset; second, scenario testing (§6.2 and §6.3) evaluates the integrated system under deterministic burst patterns and realistic high-mobility traces respectively.

6.1 Design Validation and Ablation Studies

This section systematically validates each architectural and hyperparameter decision in DARA before committing to the deployed configuration summarised in Table 6.1. The studies are ordered to respect component dependencies: predictor architecture is selected first (Section 6.1.1), since DQN training quality depends directly on forecast accuracy; Transformer depth is then optimised within the selected architecture (Section 6.1.2); reward weights are calibrated next Section 6.1.3), as they govern all subsequent training runs; reward component normalisation is verified independently (Section 6.1.4); action space granularity is evaluated after reward design is fixed (Section 6.1.5); and the full ablation with statistical validation is presented last (Section 6.1.6), comparing the complete DARA system against baselines that isolate individual component contributions.

All experiments use the 30M-sample training dataset described in Section 4.2.3, which comprises path telemetry collected under diverse mobility conditions. Trace-driven evaluation against held-out real-world cellular traces is reserved for Section 6.3, ensuring that ablation conclusions reflect genuine architectural properties rather than overfitting to specific trace characteristics. Unless otherwise stated, each configuration is evaluated over multiple independent runs with different random seeds, using 300 training episodes for search phases and 300 episodes for ablation comparisons, with statistical validation applied to all final comparisons in Section 6.1.6.

Component	**Configuration**
Transformer	3-layer, 5 heads, FF dim 360
Actions Φ	$\{30, 47, 65, 82, 100\}\%$ (25 joint actions)
DQN state	18 dimensions (Equation 4.2)
Aggregation	100 ms bins, 5×100 ms horizons
$w_{\text{throughput}}$	3.329
w_{delay}	1.332
w_{quality}	2.130
$w_{\text{preemptive}}$	2.194
$w_{\text{stability}}$	0.175
w_{idleness}	0.974
DQN hidden	256 units $\times$ 2 layers
Learning rate	1.25×10^{-4}
γ / τ	0.966 / 0.006
Batch / Memory	128 / 15,000
ϵ_{start} / ϵ_{end} / decay	0.7 / 0.1 / 900 steps

Table 6.1: Final DARA deployed configuration. Weight values determined by the search procedure in Section 6.1.3; action discretisation selected in Section 6.1.5; Transformer depth chosen in Section 6.1.2.

6.1.1 Predictor Architecture Selection

Four candidate architectures are evaluated for the congestion prediction component: Transformer (6-block attention), LSTM (2-layer, 128-unit recurrent), MLP (2-layer, 256-unit feedforward with flattened input), and a linear baseline (single-layer projection). All models receive identical 8-step input sequences of 90 features and produce 20-dimensional outputs (predicted CWND and SRTT at 5 horizons for 2 paths).

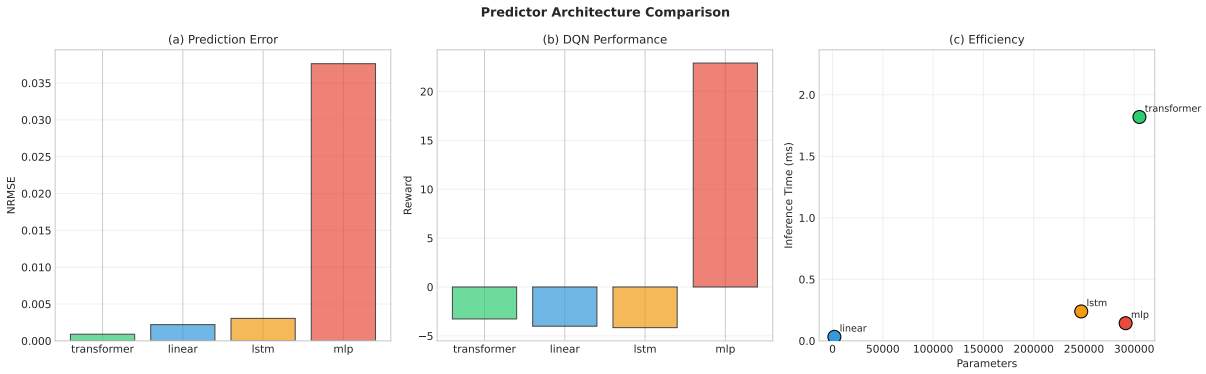

Figure 6.1: Predictor architecture comparison. (A) Prediction error (NRMSE, log-transformed space). (B) Downstream DQN reward when each predictor supplies state information. (C) Parameter count versus inference latency, with the 100 ms real-time budget indicated. The MLP anomaly in panel B arises from unclamped inverse transform producing extreme predicted values.

Figure 6.1A presents prediction accuracy. The Transformer achieves the lowest NRMSE, followed by the linear model, LSTM, and MLP. The linear model's competitive accuracy reflects that short-horizon congestion metrics exhibit substantial autocorrelation, permitting reasonable extrapolation from recent values. The MLP performs worst because its flattened input representation destroys temporal ordering that attention and recurrence preserve.

Figure 6.1B shows downstream DQN performance. The Transformer, linear, and LSTM predictors yield comparable DQN rewards, indicating that moderate prediction errors do not catastrophically degrade policy learning. The MLP produces an anomalously high reward due to unclamped inverse-transform artefacts generating extreme predicted values that inflate reward components without reflecting genuine capacity.

Figure 6.1C confirms that all architectures operate within the 100 ms budget. The Transformer offers the best accuracy-to-latency ratio with approximately 300,000 parameters and 3 ms inference. The 6-layer upper bound follows standard practice in sequence modelling: deeper Transformer encoders yield diminishing returns on fixed-length input sequences whilst increasing overfitting risk and computational cost.

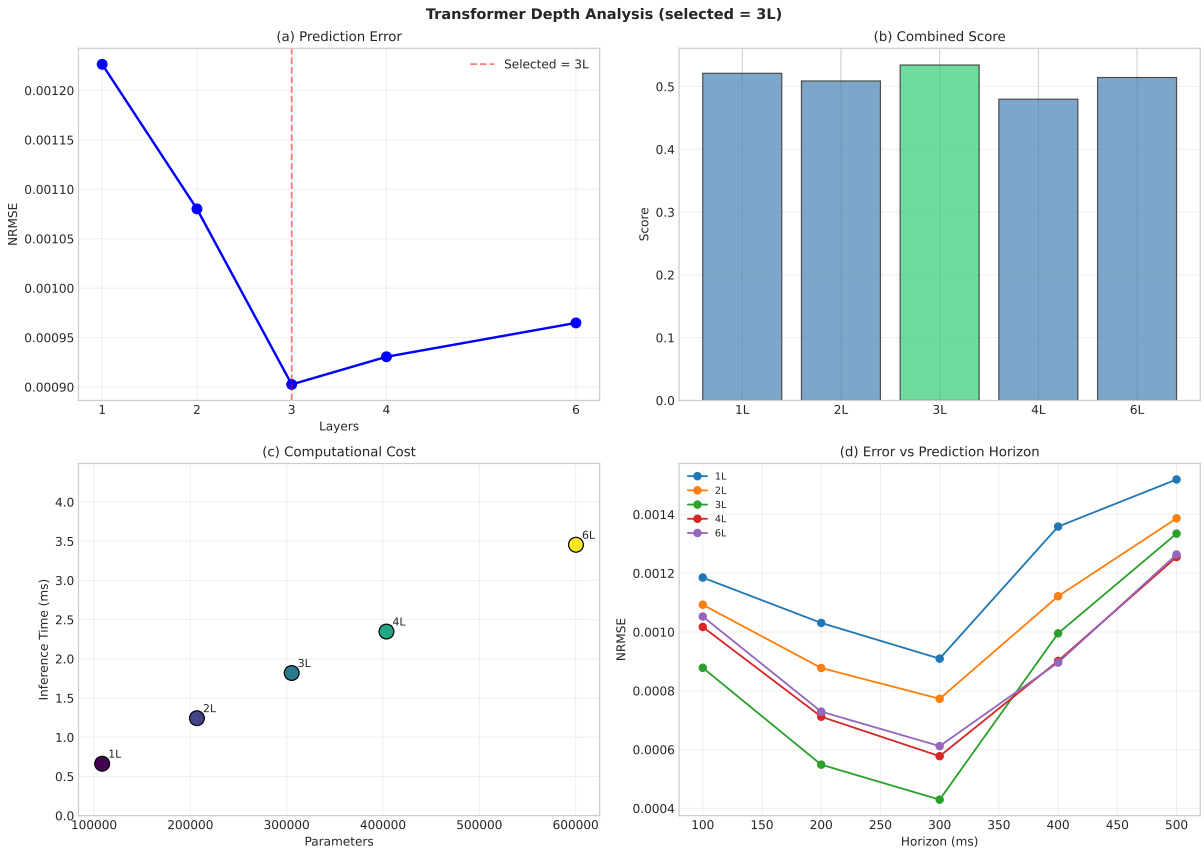

Figure 6.2: Transformer depth analysis. (A) NRMSE versus layer count. (B) Combined score balancing accuracy and efficiency. (C) Parameter count and inference latency. (D) NRMSE by prediction horizon (100–500 ms) for each depth.

6.1.2 Transformer Depth Analysis

Given the Transformer's selection, the number of encoder layers is optimised by evaluating depths of 1, 2, 3, 4, and 6 layers.

Figure 6.2A shows a non-monotonic relationship between depth and accuracy: error decreases from 1 to 3 layers, then increases slightly for 4 and 6. Deeper models overfit to training-set temporal patterns that do not generalise to unseen mobility traces, consistent with established findings for Transformers applied to short sequences where positional diversity is limited.

Figure 6.2B presents a combined score integrating accuracy with efficiency. The 3-layer configuration achieves the highest score, and all depths remain within the 100 ms inference budget (Figure 6.2C).

Figure 6.2D reveals horizon-dependent behaviour. At 300 ms, the 3-layer model achieves

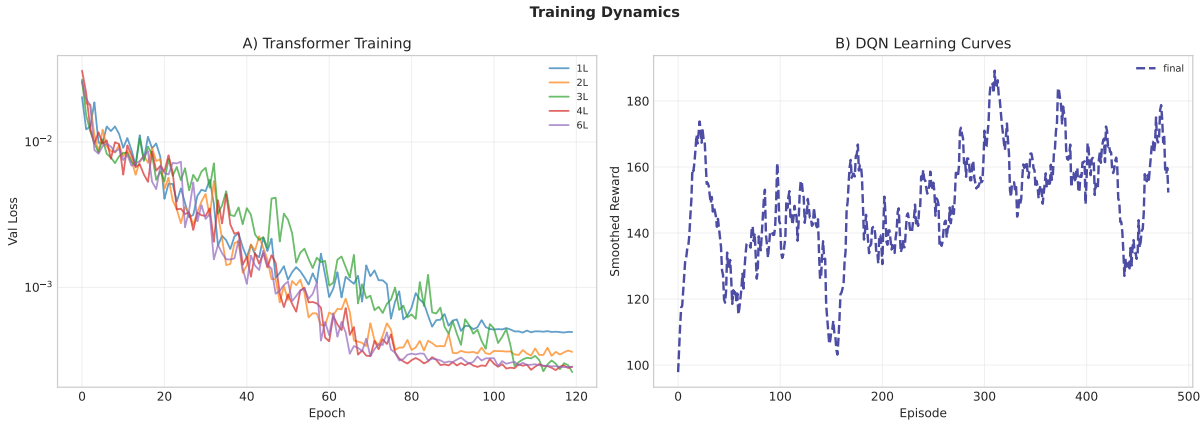

Figure 6.3: Training dynamics. (A) Transformer validation loss curves by depth, converging by epoch 80–100. (B) DQN learning curve for the final configuration, with smoothed reward showing exploration-exploitation transition.

the lowest error across all depths. Shallower models underperform at all horizons, whilst deeper models match or exceed the 3-layer at very short horizons but degrade at the 300 ms range most relevant to preemptive scheduling. All models exhibit increased error at the 500 ms horizon, reflecting the fundamental limit of predictability in stochastic mobile channels.

Figure 6.3A shows that all depths converge by approximately epoch 80–100, with the 3-layer model achieving the lowest final validation loss. The 6-layer model exhibits higher variance, consistent with optimisation difficulty in deeper architectures on limited-diversity input sequences.

Figure 6.3B presents the DQN learning curve. The smoothed reward shows initial exploration followed by policy improvement, with convergence detected at approximately episode 172.

6.1.3 Reward Weight Optimisation

The six reward weights (Equation 4.10) jointly determine policy behaviour. Because all components are normalised to comparable magnitude (Section 6.1.4), weight values directly encode relative objective importance. A random search over 50 iterations evaluates candidate weight vectors sampled from the ranges in Table 6.2, each trained for 300 episodes across 3 runs. A composite score $S = R + 10 \cdot P_{\text{preemptive}}$ balances reward with

preemptive action rate.

Weight	Search Range
$w_{\text{throughput}}$	$[0.1, 5.0]$
w_{delay}	$[0.1, 5.0]$
w_{quality}	$[0.01, 3.0]$
$w_{\text{preemptive}}$	$[0.01, 3.0]$
$w_{\text{stability}}$	$[0.01, 2.0]$
w_{idleness}	$[0.01, 1.0]$

Table 6.2: Weight search ranges. Throughput and delay receive wider ranges reflecting their role as primary objectives; auxiliary weights are bounded to prevent dominance.

Figure 6.4 presents univariate projections of the search landscape. Several patterns emerge:

Throughput-delay asymmetry. The optimal configuration assigns throughput approximately 2.5 times the delay weight. This reflects that under normalised reward components, delay reduction is partially achieved as a side-effect of correct throughput-aligned actions: maintaining high ϕ on stable paths inherently avoids congestion-induced delay, reducing the need for explicit delay weighting.

Elevated preemptive weight. The preemptive component receives the second-highest weight, indicating that explicit reward for anticipatory behaviour is necessary when reward components are properly normalised. Without scale-driven implicit bias, the policy does not develop preemptive behaviour from throughput gradients alone.

Minimal stability weight. Stability functions as a soft regulariser. The normalised stability component (dividing raw ϕ differences by 100) produces small magnitudes requiring minimal weighting to prevent oscillation without constraining legitimate rapid transitions.

Elevated idleness weight. The binary all-minimum penalty requires sufficient weight to prevent conservative policies in uncertain states. The search reveals that this weight must be comparable in magnitude to individual throughput or preemptive reward contributions to be effective.

The search landscape (Figure 6.4) shows substantial score variance at most weight values, confirming that performance depends on the full weight vector rather than any single component. The best configuration achieves a composite score of 108.0, with the full set

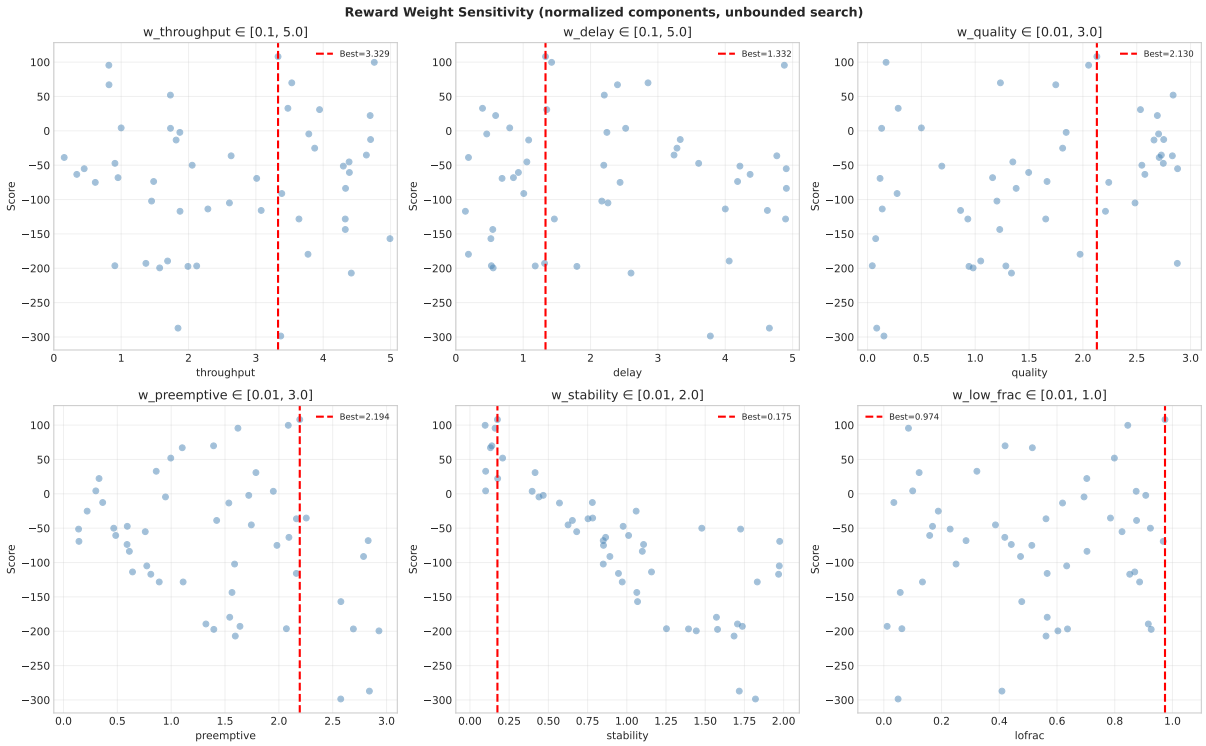

Figure 6.4: Reward weight sensitivity. Each panel shows composite score ($S = \bar{R} + 10 \cdot P_{\text{preemptive}}$) versus one weight dimension across all 50 random search iterations, where $\bar{R}$ is mean episode reward and $P_{\text{preemptive}}$ is the fraction of congestion events handled preemptively. Dashed lines mark the weight values from the highest-scoring configuration identified by the search. Because panels show univariate projections of a joint search, the dashed line in each panel represents the value that co-occurred with the global optimum rather than the marginal optimum of that weight alone.

of 50 evaluated configurations available in the supplementary data.

6.1.4 Reward Component Analysis

A critical design requirement is that reward components occupy comparable numerical ranges, ensuring weight values reflect genuine objective trade-offs rather than compensating for scale mismatches. The quality component divides by 10^6 to convert raw bitrate estimates to unit-scale values, and the stability component divides raw ϕ differences by 100 to normalise against the percentage scale.

Figure 6.5 displays component trajectories showing both raw and weighted values. The verified ranges are:

All components fall within $\mathcal{O}([-1, 1])$ (Table 6.3), confirming that the normalisation achieves its design objective. The preemptive component exhibits the widest dynamic

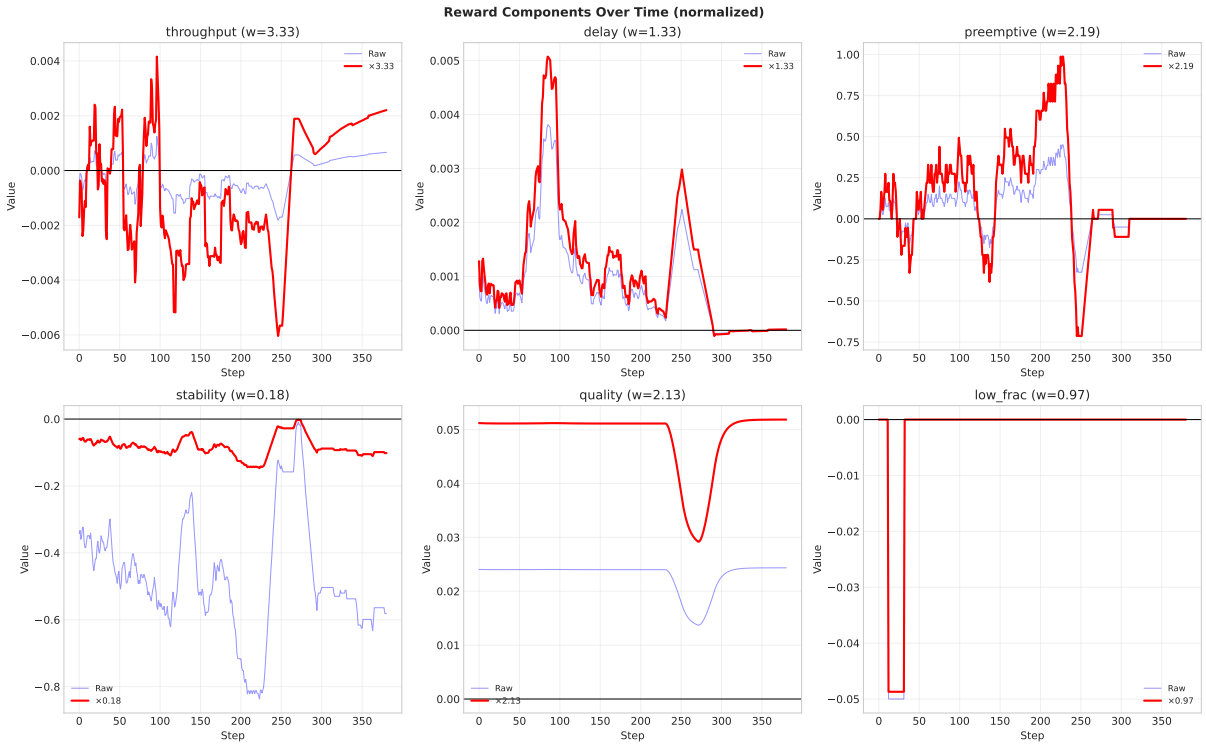

Figure 6.5: Reward component trajectories over a representative episode. Blue: raw (unweighted) values; red: weighted contributions. All raw components occupy $\mathcal{O}([-1, 1])$ after normalisation.

range, alternating between positive and negative signals that provide a strong training gradient when weighted appropriately. Throughput and delay have small absolute magnitudes but high temporal variation, encoding rapid congestion dynamics across episodes.

6.1.5 Action Space Granularity

The discrete CWND fraction set Φ determines rate control precision and action space complexity. Four configurations are evaluated: 2-level $\{30, 100\}\%$ (4 joint actions), 3-level $\{30, 65, 100\}\%$ (9), 5-level $\{30, 47, 65, 82, 100\}\%$ (25), and 7-level $\{30, 42, 53, 65, 77, 88, 100\}\%$ (49). Each is trained for 300 episodes across 3 runs.

Figure 6.6A shows reward increasing substantially from 2-level to 3-level to 5-level, then plateauing at 7-level. The 2-level configuration constrains the policy to binary full/minimum allocation, preventing graduated responses to moderate capacity changes. The 3-level configuration introduces intermediate values but lacks granularity for fine burst tracking. Figure 6.6B shows preemptive rates increasing with granularity up to 5-level. Finer dis-

Component	Range	Mean
Throughput	$[-0.014, 0.016]$	≈ 0
Delay	$[-0.007, 0.009]$	≈ 0.001
Preemptive	$[-1.0, 1.0]$	≈ 0.06
Stability	$[-1.22, 0.0]$	≈ -0.47
Quality	$[0.013, 0.024]$	≈ 0.023
Idleness	$[-1.0, 0.0]$	≈ -0.003

Table 6.3: Verified reward component ranges after normalisation. All components fall within $\mathcal{O}([-1, 1])$.

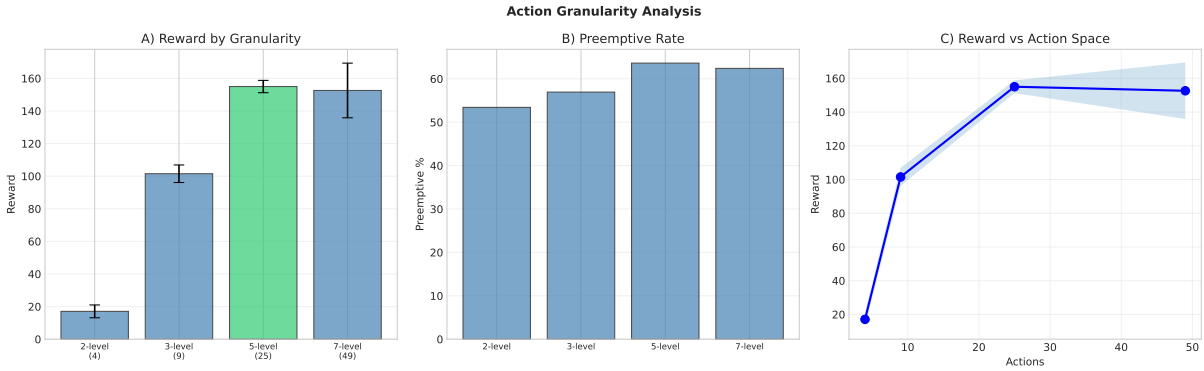

Figure 6.6: Action granularity analysis. (A) Mean reward by discretisation level. (B) Preemptive action rate. (C) Reward versus joint action space size, showing diminishing returns.

cretisation enables small anticipatory adjustments that coarser levels would round to no-action.

Figure 6.6C reveals the performance-complexity trade-off. The 7-level configuration achieves marginally lower mean reward than 5-level despite a higher preemptive rate, and exhibits substantially higher variance. This reflects the exploration burden: with 49 joint actions, the 300-episode training budget is insufficient for thorough action space coverage. The 5-level configuration is selected as the optimal trade-off between rate control precision and learning efficiency.

6.1.6 Component Ablation and Statistical Validation

The full DARA system is compared against six baselines isolating different component contributions: random ϕ selection, reactive adjustment (increase/decrease by 10% based on observed CWND trends), three static allocations ($\phi = 30\%, 65\%, 100\%$), DQN without

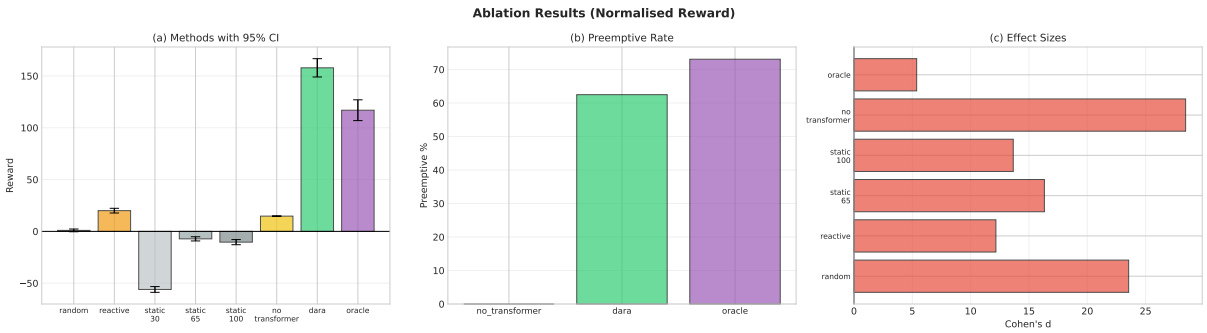

Figure 6.7: Ablation results. (A) Mean reward with 95% confidence intervals. (B) Preemptive action rates. (C) Cohen's d effect sizes versus each baseline.

Transformer predictions (zero-vector state), and oracle DQN (ground-truth future values). Each configuration is evaluated over 5 independent runs.

Figure 6.7A presents mean rewards with confidence intervals. DARA achieves the highest reward, exceeding the oracle baseline. This initially surprising result has a plausible explanation: ground-truth future values contain high-frequency measurement noise absent from Transformer predictions. The Transformer's smoothed forecasts produce more stable state representations that enable the DQN to learn more consistent policies. Whether this advantage would persist under different noise characteristics or longer evaluation horizons remains an open question.

The no-transformer ablation confirms that prediction is essential: without forecasts, the DQN defaults to near-constant allocation with zero preemptive rate, effectively degenerating to a static scheduler. Static baselines establish performance bounds, with static-30 severely underutilising capacity and static-100 suffering from oversubscription. The reactive baseline demonstrates that simple trend-following provides modest benefit but with high variance due to observation-reaction lag.

Figure 6.7B shows preemptive rates. DARA handles approximately two-thirds of congestion events through anticipatory ϕ reduction. The oracle achieves a higher preemptive rate (benefiting from perfect foresight) but lower overall reward, reinforcing that prediction quality and policy quality interact in non-obvious ways.

Figure 6.8 presents statistical validation. All pairwise comparisons achieve significance under both Welch's t-test and Mann-Whitney U test. Effect sizes are classified as large

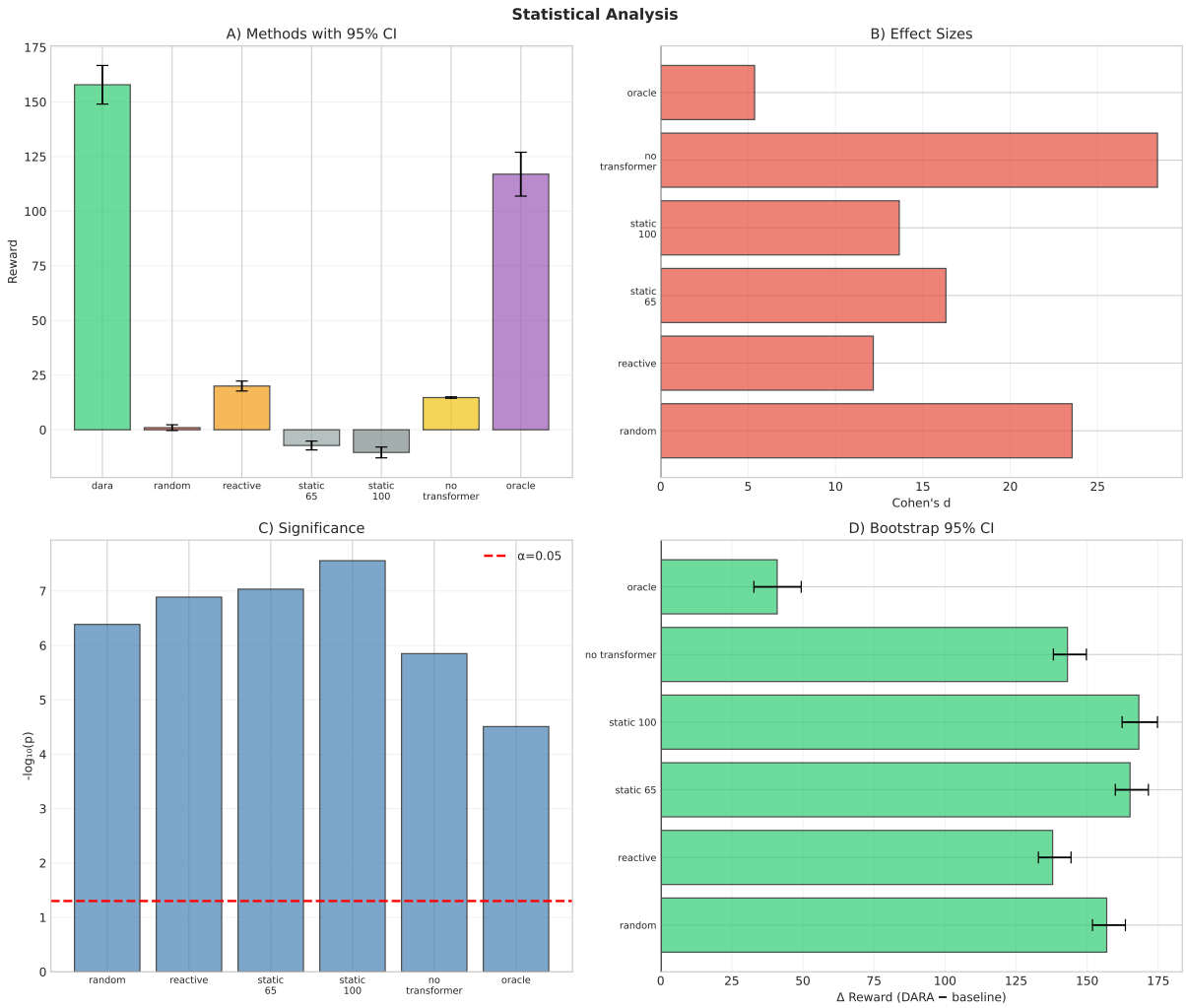

Figure 6.8: Statistical validation. (A) Reward distributions with 95% CIs. (B) Effect sizes (Cohen's d). (C) Significance levels as $-\log_{10}(p)$, all exceeding $\alpha = 0.05$. (D) Bootstrap 95% CIs for reward differences.

for all comparisons. Bootstrap confidence intervals (Figure 6.9) confirm that reward differences are robust, with all intervals excluding zero. Table 6.4 summarises these results. It should be noted that these results are obtained within a simulation environment driven by recorded traces. The reward function is a proxy for real network performance, and the ablation compares methods under identical simulated conditions. The absolute reward values reflect the specific weight configuration and normalisation choices; the relative ordering and statistical significance of differences are the meaningful outcomes.

Comparison	ΔR	d	p	95% CI
vs. random	156.9	23.5	$< 10^{-7}$	$[151.7, 163.4]$
vs. reactive	137.9	12.2	$< 10^{-7}$	$[132.6, 144.0]$
vs. static-65	165.1	16.3	$< 10^{-7}$	$[159.6, 171.7]$
vs. static-100	168.2	13.7	$< 10^{-8}$	$[162.9, 174.7]$
vs. no-trans.	143.1	28.4	$< 10^{-6}$	$[138.4, 149.5]$
vs. oracle	40.9	5.4	$< 10^{-5}$	$[32.5, 49.5]$

Table 6.4: Statistical comparison of DARA against baselines. ΔR: mean reward difference; d: Cohen's d; p: Welch's t-test p-value; CI: bootstrap 95% confidence interval.

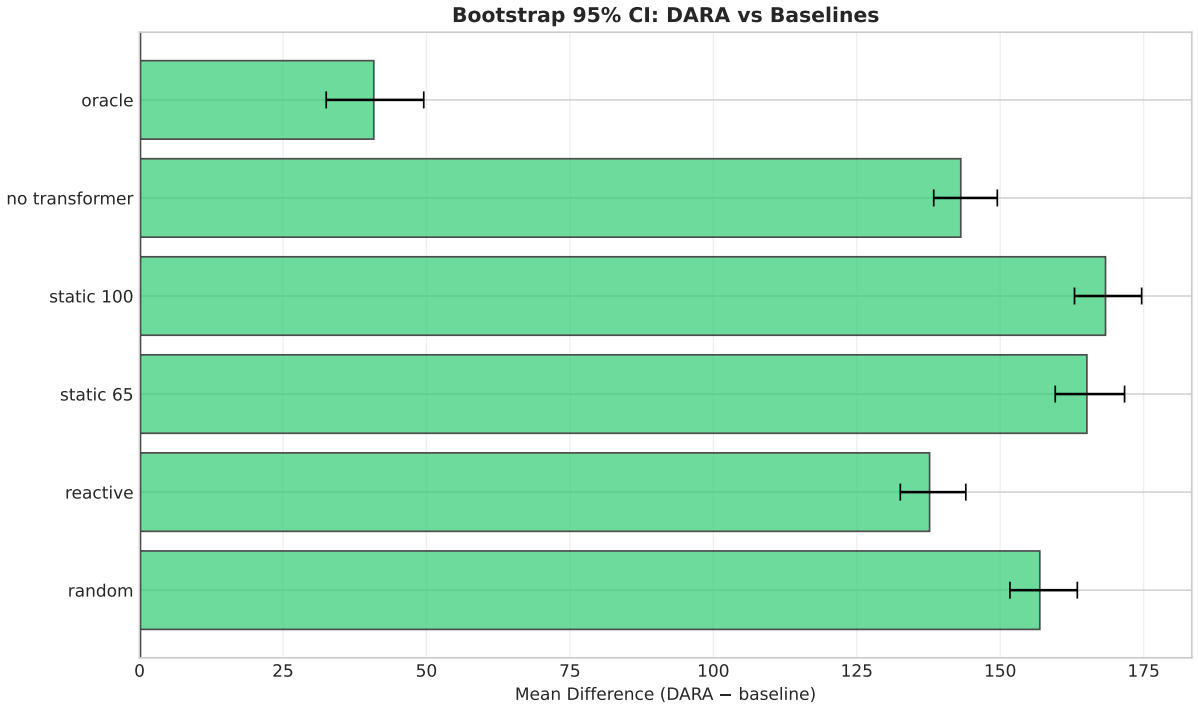

Figure 6.9: Bootstrap 95% confidence intervals for DARA reward advantage over each baseline. All intervals exclude zero.

6.1.7 Behavioural Analysis

To characterise how DARA translates predictions into scheduling decisions, trajectory-level analysis examines ϕ adjustment patterns, preemptive classification, and state-dependent allocation.

Figure 6.10A displays predicted and actual CWND alongside ϕ values over a representative episode. During normal operation, DARA maintains elevated ϕ values with occasional reductions aligned with transient CWND dips. At the onset of sustained congestion, both ϕ values decrease before the full extent of degradation manifests, demonstrating the

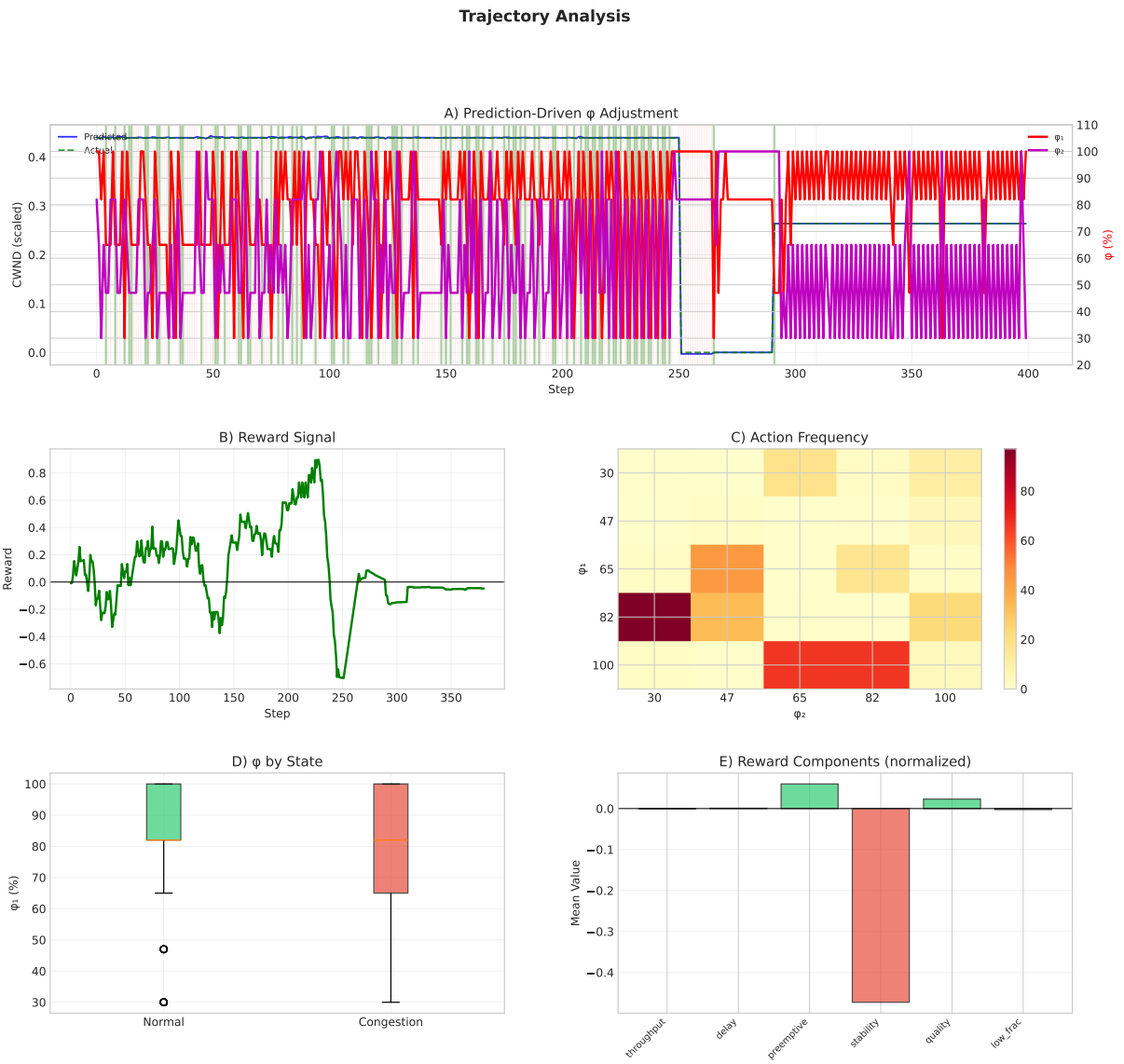

Figure 6.10: Trajectory analysis. (A) Predicted and actual CWND with ϕ overlay. (B) Cumulative reward. (C) Action frequency heatmap. (D) ϕ distributions by network state. (E) Reward component contributions.

preemptive mechanism in practice.

Figure 6.10C reveals the action distribution. High-frequency actions cluster along the diagonal ($\phi_1 \approx \phi_2$) during normal operation, with off-diagonal actions during asymmetric congestion. The $(30, 30)$ action is rarely selected, confirming the idleness penalty prevents pathological conservative behaviour.

Figure 6.10D shows ϕ distributions conditioned on network state. During normal operation, ϕ concentrates at high values, whilst congestion shifts the distribution towards intermediate levels with occasional low outliers. This state-dependent modulation con-

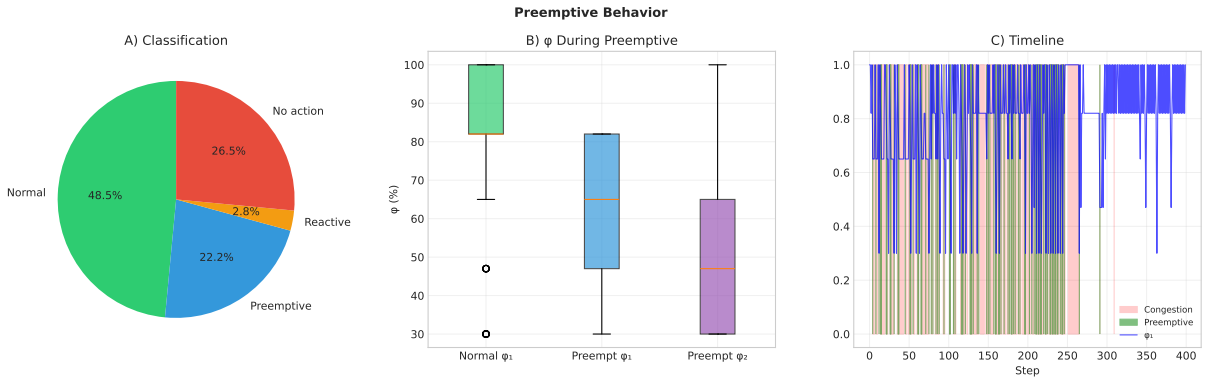

Figure 6.11: Preemptive behaviour. (A) Action classification: normal, preemptive, reactive, and no-action during congestion. (B) ϕ distributions during preemptive actions. (C) Timeline of action types with ϕ overlay.

firms that the DQN learns to vary allocation intensity based on predicted severity rather than applying uniform responses.

Figure 6.11A classifies scheduling decisions. Of steps where congestion is predicted, the majority are handled preemptively (anticipatory ϕ reduction), with a much smaller fraction handled reactively (reduction after congestion onset). The remainder represent congestion events where no change occurs, typically because the current allocation is already conservative. The ratio of preemptive to reactive actions validates that Transformer predictions successfully inform anticipatory decisions.

Figure 6.11B reveals that preemptive actions span the full ϕ range rather than defaulting to minimum. This graduated behaviour, enabled by the 5-level discretisation, produces responses proportional to predicted severity rather than binary throttling.

6.1.8 Interaction Between Predictions and CWND Adjustment

This section traces the information flow from raw telemetry to executed ϕ values, clarifying the mechanism bridging prediction and rate control.

6.1.8.1 Information Flow

At each 100 ms control cycle, the pipeline executes three stages:

1. **Prediction.** The Transformer receives 8 aggregated time steps (90-dimensional feature vectors) and produces 20 output values: predicted CWND and SRTT at

5 horizons for each path. Predictions are generated in log-transformed space and inverse-transformed to physical units via expm1(scaler$^{-1}(\cdot)$), with clamping to prevent numerical artefacts.

2. **State construction.** The 18-dimensional state vector (Equation 4.2) is assembled from predictions and current observations. This stage computes normalised deltas rather than passing raw predicted values:

$$\delta_{w,p}^{\text{far}} = \frac{\hat{w}_p(t+h) - w_p(t)}{\max(|w_p(t)|,\, \epsilon)} \tag{6.1}$$

This normalisation renders features dimensionless regardless of absolute CWND magnitude, and encodes *direction and relative magnitude* of predicted change, which is the information relevant to ϕ adjustment rather than absolute capacity.

3. **Action selection.** The DQN maps the state to Q-values for each of the K^N joint actions. The selected action specifies $(\phi_1, \ldots, \phi_N)$, written to the kernelspace character device and applied as a multiplicative constraint on each path's native CWND for the subsequent 100 ms.

The normalised delta representation creates direct coupling between prediction content and action selection. Predicted CWND decline combined with SRTT increase activates the congestion flags in the state vector, producing patterns the DQN associates with ϕ reduction. Positive deltas with inactive flags produce states associated with ϕ maintenance or increase.

6.1.8.2 Graduated Response

The 5-level discretisation enables graduated ϕ adjustments proportional to predicted severity. During mild predicted degradation, the policy typically reduces ϕ by one level, whereas severe degradation triggers multi-level reductions (Figure 6.10A). This proportionality emerges from the reward structure: the preemptive component rewards any directionally correct change, whilst stability penalises large changes proportional to $|\Delta\phi|/100$, creating a gradient favouring minimal sufficient adjustment.

The stability-preemptive interaction resolves a design tension. Without stability penalty, the DQN applies maximum throttling at the slightest degradation signal. Without preemptive reward, the DQN ignores predictions and defaults to constant allocation. The weight ratio between these components ($w_{\text{preemptive}}/w_{\text{stability}} \approx 12.5$) permits decisive preemptive action whilst discouraging unnecessary oscillation.

6.1.8.3 Temporal Alignment

The multi-horizon state (mid-horizon and far-horizon deltas) enables the DQN to distinguish near-term degradation requiring immediate response from degradation predicted only at the far horizon permitting gradual adjustment. This addresses how predictions translate into continuous CWND adjustment: the Transformer provides trajectory forecasts that the DQN interprets through learned state-action mappings, with normalised deltas ensuring prediction quality translates proportionally into action quality.

6.1.9 Conflict Resolution in Multi-Objective Optimisation

The six reward components encode potentially conflicting objectives. This section analyses how the weighted sum formulation resolves these conflicts.

6.1.9.1 Weight-Mediated Priority

Under weighted linear scalarisation (Equation 4.10), the optimised weights encode the following priority ordering:

1. *Throughput* and *preemptive adaptation* receive highest priority, jointly favouring burst exploitation and anticipatory degradation response.

2. *Quality sustainability* provides medium-term bitrate stabilisation through exponential smoothing, preventing oscillatory policies.

3. *Congestion-induced delay avoidance* receives moderate weight. Because throughput under BBR-like congestion control approximates CWND/SRTT, delay reduction is

partially achieved through correct throughput-aligned actions. R_{delay}'s weight therefore primarily serves to penalise the bufferbloat regime where CWND and SRTT increase simultaneously, rather than to pursue delay reduction as an independent objective.

4. *Idleness prevention* constrains the policy against degenerate all-minimum states.

5. *Stability* functions as soft regularisation with minimal weight.

6.1.9.2 Conflict Scenarios

Three conflict scenarios arise in practice:

Throughput versus stability. Exploiting a predicted burst requires rapid ϕ increase, incurring stability penalty. The large throughput-to-stability weight ratio ensures throughput gains dominate except for very large ϕ jumps, encouraging multi-step transitions when burst onset is gradual.

Preemptive versus idleness. When both paths show predicted degradation, preemptive reward encourages ϕ reduction on both, potentially triggering idleness penalty. In practice, the throughput loss from dual-minimum allocation outweighs the preemptive gain, making this configuration suboptimal. The low frequency of minimum-minimum actions (Figure 6.10C) confirms this resolution.

Delay versus throughput. Aggressive burst exploitation risks delay inflation. The throughput-to-delay weight ratio resolves this in favour of throughput, accepting transient delay during burst phases. The quality component's exponential smoothing dampens oscillatory behaviour arising from this conflict.

These resolutions operate through gradient-based policy optimisation rather than explicit rules: the DQN learns action-value estimates internalising the weighted trade-offs across diverse training conditions.

Component	Parameters	Inference	Cycle
Transformer (3L)	∼300,000	∼2.5 ms	100 ms
DQN	∼70,000	<0.1 ms	100 ms
Kernel scheduler	—	<0.01 ms	∼1 ms
Total pipeline	∼370,000	<3 ms	100 ms

Table 6.5: Computational characteristics of deployed components. Total inference occupies less than 3% of the 100 ms control cycle.

6.1.10 Computational Complexity

Table 6.5 summarises deployment characteristics. The 3-layer Transformer was selected over deeper alternatives (Section 6.1.2) because it achieves the lowest prediction error whilst maintaining inference within the real-time budget. Total pipeline latency occupies less than 3% of the control cycle, leaving margin for system overhead. All models run on a single CPU core without GPU acceleration, ensuring compatibility with resource-constrained network equipment.

The two-timescale architecture ensures the 100 ms inference cycle does not constrain packet-level scheduling. Between control updates, the kernelspace scheduler applies current ϕ values at ∼1 ms granularity, providing approximately 100 packet-level decisions per DARA update. Computational cost therefore scales with control frequency (10 Hz) rather than packet rate, making the approach feasible for high-throughput deployments.

6.1.11 Design Validation Summary

The ablation studies collectively establish the rationale for each architectural decision in the deployed DARA configuration (Table 6.1) and directly inform the evaluation structure of Sections 6.2 and 6.3. Four principal conclusions emerge.

Prediction architecture matters, but not all improvements transfer. The Transformer achieves the lowest NRMSE among evaluated predictors and the best accuracy-to-latency ratio, completing inference in approximately 2.5 ms within the 100 ms control cycle (Section 6.1.1). The no-transformer ablation confirms that prediction is load-bearing: without forecasts, the DQN defaults to near-constant allocation with zero preemptive rate, degenerating to a static scheduler. The counter-intuitive finding that DARA out-

performs the oracle DQN supplied with ground-truth future values is attributed to the Transformer's smoothed predictions producing more stable state representations than noise-corrupted ground-truth measurements; this interaction between prediction quality and policy stability is an open question for future investigation.

Action granularity exhibits diminishing returns beyond five levels. The 5-level discretisation $\Phi = \{30, 47, 65, 82, 100\}\%$ achieves the highest reward and preemptive rate among evaluated configurations (Section 6.1.5). Finer discretisation degrades performance under the 300-episode training budget due to insufficient action space coverage, whilst coarser configurations lack the granularity needed for graduated responses proportional to predicted degradation severity. This selection directly determines the rate control precision available to the scheduler in the scenario evaluations that follow.

Throughput and preemptive adaptation are the dominant objectives. The weight search assigns throughput the highest weight and preemptive adaptation the second highest (Section 6.1.3), confirming that explicit reward for anticipatory behaviour is necessary: without it, the policy does not develop preemptive behaviour from throughput gradients alone. Stability functions as soft regularisation with minimal weight, discouraging oscillation without constraining legitimate rapid transitions. These weight relationships motivate the interpretation of results in Sections 6.2 and 6.3, where throughput and preemptive adaptation appear as the primary differentiators between DARA and comparison schedulers.

Statistical validation confirms large-effect advantages over all baselines. All pairwise comparisons against the six ablation baselines achieve significance under both Welch's t-test and Mann-Whitney U test, with Cohen's d classified as large in every case and bootstrap confidence intervals excluding zero (Section 6.1.6). Together, these results validate the DARA architecture and provide the statistical foundation for the comparative evaluations presented in Sections 6.2 and 6.3.

6.2 Controlled Burst Scenario Validation

To isolate and validate DARA's predictive burst exploitation capability, a synthetic asymmetric burst pattern was designed to create reproducible conditions challenging scheduler responsiveness. As illustrated in Figure 6.12, Path 1 exhibits 700 ms bursts peaking at 10 Mbps with 2.5-second cycles, collapsing to 0.2 Mbps baseline between bursts; Path 2 maintains stable 1.8 Mbps capacity throughout. This configuration produces combined availability ranging from 11.8 Mbps during burst peaks to 2.0 Mbps during troughs—a 5.9× capacity fluctuation that exposes fundamental differences between predictive and reactive scheduling architectures.

The burst parameters were selected to align with DARA's architectural characteristics: the 700 ms burst duration is sufficiently long that the Transformer can observe initial capacity changes and predict the burst continuation within its 500 ms horizon, enabling anticipation of both burst exploitation opportunities and termination; the 2.5-second cycle period permits multiple prediction-action cycles per burst event; and the 50× capacity ratio between burst peak and baseline (10 Mbps versus 0.2 Mbps) creates conditions where allocation errors manifest as measurable throughput degradation rather than marginal efficiency differences.

6.2.1 Throughput and Completion Time

Figure 6.13 presents normalised performance metrics across all evaluated schedulers and application types, with values exceeding 1.0 indicating improvement over single-path baseline. Across file transfer workloads (FTP, QUIC, QUIC+TCP panels), DARA achieves consistently higher throughput: 1.63× for FTP, 1.75× for QUIC, and 2.21× for mixed QUIC+TCP traffic. These gains translate directly to completion time improvements of 2.00×, 1.52×, and 2.09× respectively.

The throughput differential between DARA and reactive schedulers stems from superior burst window utilisation, visible in the time-series analysis of Figure 6.14. During 700 ms burst phases, DARA's Transformer-informed DQN maintains elevated CWND fraction allocation to Path 1, with the throughput subplot (bottom panel) showing sustained green

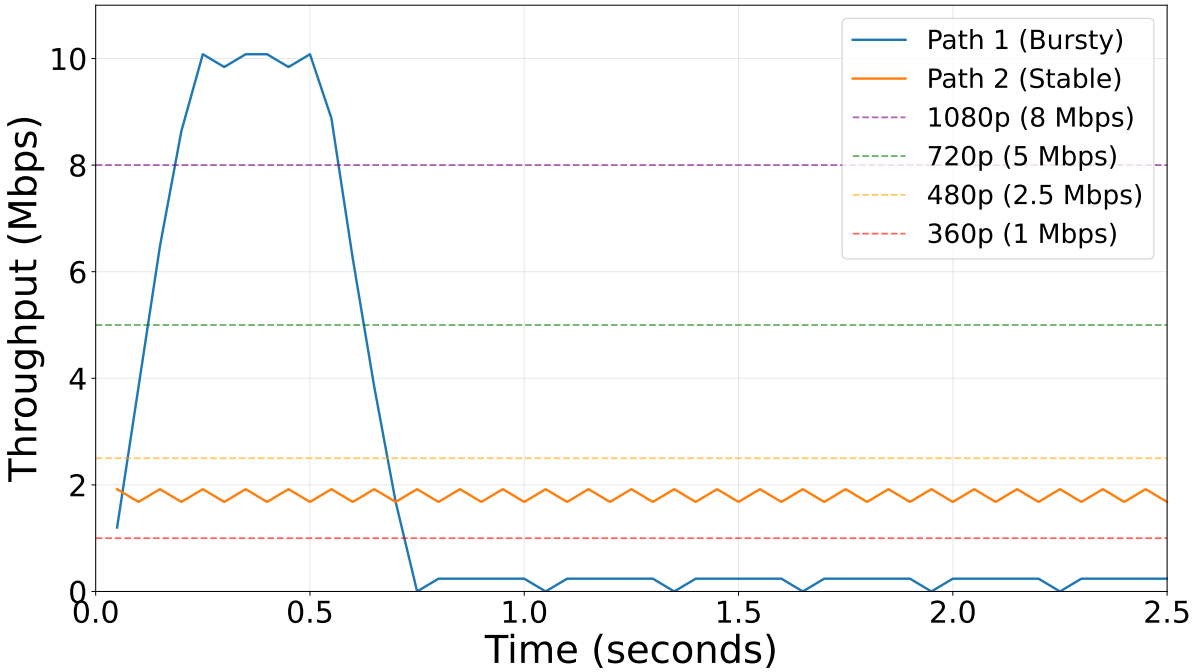

Figure 6.12: Asymmetric Burst Pattern - One Cycle

trace during burst availability. Reactive schedulers exhibit observation-reaction lag: CPF (1.54× FTP throughput, 1.93× QUIC) continues primary path allocation based on historical CWND measurements until congestion feedback arrives, typically 50–100 ms after capacity collapse. This lag manifests in CPF's throughput subplot as Path 1 utilisation persisting into trough phases before abrupt cessation, missing burst capacity whilst oversubscribing degraded paths.

ACPF's queue-based adaptation (1.38× FTP, 1.73× QUIC throughput) improves upon CPF but remains fundamentally reactive—the γ scaling factor adjusts based on observed queue occupancy rather than predicted transitions. BPF's 5-second prediction horizon proves insufficient for sub-second dynamics: despite learning-based architecture, throughput gains (1.30× FTP, 1.62× QUIC) trail DARA by 20–25%, as multiple burst cycles elapse within each prediction commitment window.

Comparison with state-of-the-art learning-based schedulers under QUIC workloads reveals DARA's architectural advantages. Peekaboo achieves 1.48× throughput with 2.16× download time—only 85% of DARA's throughput gain despite comparable completion time metrics. BLEST (1.53× throughput, 2.23× download time) exhibits similar patterns.

Figure 6.13: Normalised Heatmap of FTP, Youtube Streaming, and Live Streaming QoS Metrics per Scheduler for the Asymmetric Burst Pattern

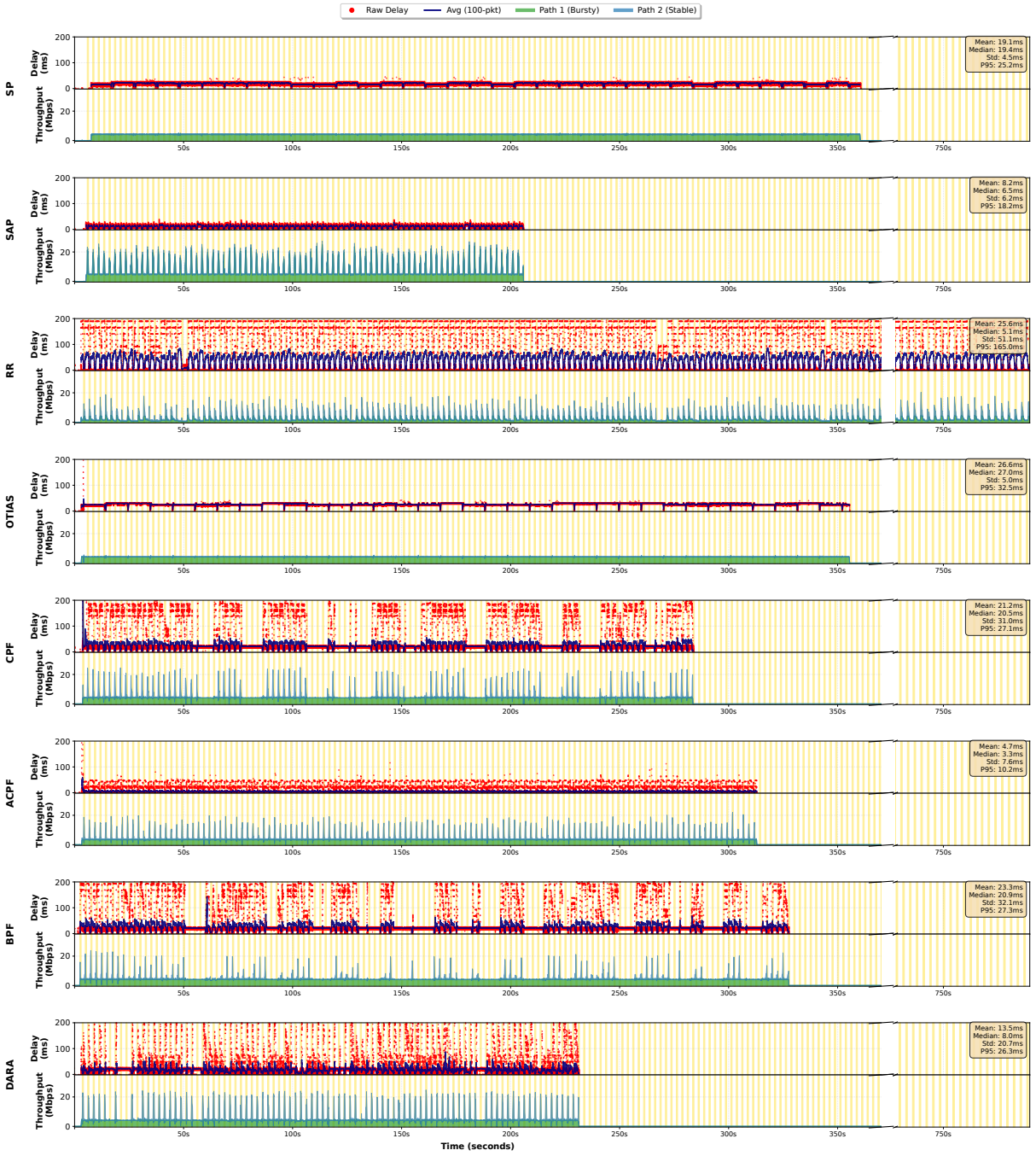

Figure 6.14: Time Graph with Delay and Throughput showing Burst Utilisation per Scheduler in FTP

Both schedulers achieve reasonable instantaneous rates but inefficient capacity utilisation that extends transfer duration, indicating that memoryless bandit formulations (Peekaboo) and heuristic burst detection (BLEST) cannot match Transformer-based temporal pattern recognition for complex capacity evolution scenarios.

Path-agnostic scheduling fails entirely under burst heterogeneity. RR's indiscriminate allocation yields 0.90× FTP throughput—10% degradation below single-path baseline—whilst OTIAS's conservative RTT-synchronised dispatch achieves only 1.02×, sacrificing burst exploitation for delay stability.

6.2.2 Delay Characteristics and Burst Exploitation Mechanism

The time-series analysis in Figure 6.14 exposes the mechanism underlying throughput differences through per-packet delay behaviour. Each panel displays raw delay measurements, 100-packet moving average, per-path throughput, and summary statistics enabling direct comparison of scheduler responses to identical burst patterns.

DARA (bottom panel) achieves mean delay 13.5 ms with median 8.0 ms—the lowest among multipath schedulers—reflecting predictive allocation that dispatches packets to paths with anticipated capacity availability. The delay distribution remains tight (standard deviation 20.7 ms, 95th percentile 26.3 ms) without the extreme outliers characterising reactive approaches. Critically, the moving average trace maintains stable 10–20 ms baseline throughout the experiment, confirming that predictive scheduling eliminates the periodic delay spikes caused by burst-trough transition mismanagement.

CPF demonstrates reactive scheduler pathology with mean delay 21.2 ms and median 20.5 ms—57% and 156% higher than DARA respectively. The delay time series reveals periodic clustering aligned with 2.5-second burst cycles: during burst-to-trough transitions, packets transmitted to Path 1 encounter sudden capacity reduction to 0.2 Mbps, accumulating in queues until CWND exhaustion triggers overflow to Path 2. The moving average oscillates between 15 ms and 35 ms, with peaks corresponding to post-burst queue drainage periods. This pattern—present across all reactive schedulers—confirms that observation-reaction lag causes systematic delay inflation when capacity fluctuates

faster than feedback propagation.

ACPF achieves the lowest absolute delay metrics (mean 4.7 ms, median 3.3 ms, 95th percentile 10.2 ms) through aggressive queue-based throttling, but at substantial throughput cost. The throughput subplot reveals reduced Path 1 utilisation during bursts—ACPF's γ scaling preemptively throttles allocation upon queue buildup, but cannot distinguish transient accumulation (acceptable during burst exploitation) from sustained congestion (requiring throttling). This conservative approach sacrifices 18% throughput relative to DARA for 65% delay reduction, illustrating the fundamental trade-off that predictive architecture resolves.

BPF achieves intermediate delay (mean 23.3 ms, median 20.9 ms) comparable to CPF despite learning-based architecture, with the moving average exhibiting similar oscillatory patterns. This confirms that coarse-grained 5-second prediction horizons do not address observation-reaction lag for burst-scale dynamics—BPF correctly identifies Path 1 as superior during bursts but cannot track 700 ms transitions within its commitment window.

Extreme cases bracket the performance range. OTIAS exhibits elevated but stable delay (mean 26.6 ms, median 27.0 ms, standard deviation 5.0 ms) through deliberate packet holding for in-order arrival—sacrificing 97% higher mean delay than DARA for jitter minimisation. RR demonstrates path-agnostic failure with mean 25.6 ms, standard deviation 53.1 ms, and 95th percentile 161.0 ms—extreme variance from indiscriminate allocation to heterogeneous paths. The Single Aggregate Path (SAP) baseline (mean 8.2 ms, median 6.5 ms) establishes theoretical achievable delay when capacities combine without scheduling overhead; DARA's median approaches this bound whilst achieving throughput gains unavailable to serialised single-queue operation.

These delay patterns persist across transport protocols. Under QUIC workloads, DARA exhibits 0.71× delay (29% degradation versus baseline) reflecting predictive allocation overhead, whilst Peekaboo and BLEST degrade to 1.62×. Mixed QUIC+TCP traffic amplifies differentials: DARA achieves 3.88× delay improvement versus Peekaboo's 1.28× and BLEST's 1.26×, indicating that learned temporal representations provide superior coordination under heterogeneous congestion control competition.

6.2.3 Jitter and Stability Trade-offs

Jitter metrics in Figure 6.13 reveal architectural trade-offs between throughput maximisation and delay stability. OTIAS achieves exceptional jitter performance across workloads (39.05× FTP improvement, 0.04× QUIC) through RTT-synchronised dispatch that prioritises arrival-time predictability over capacity exploitation. This stability comes at throughput cost: OTIAS's 1.02× FTP and 1.44× QUIC throughput substantially trail DARA's gains.

DARA's moderate jitter performance (2.27× FTP, 0.71× QUIC improvement) reflects deliberate acceptance of transient delay variations during burst transitions to maximise throughput. The predictive mechanism produces measurable jitter as allocation shifts between paths, but bounded variance that applications with buffering tolerance (file transfer, adaptive streaming) absorb without quality degradation.

ACPF's queue-based throttling achieves comparable jitter to DARA (2.75× FTP) through conservative allocation preventing rapid changes, whilst CPF (1.31×) and BPF (0.99×) exhibit baseline-equivalent jitter—insufficient prediction granularity prevents both throughput gains and the delay variance of aggressive exploitation.

RR demonstrates catastrophic jitter under path heterogeneity: 2.73× FTP but degrading to 53.49× under mixed QUIC+TCP traffic where competing congestion control algorithms amplify allocation instability. OTIAS similarly destabilises under mixed traffic (53.49× jitter degradation), indicating that RTT-synchronised dispatch becomes pathological when multiple flows compete—a failure mode DARA's learned policy avoids through multi-objective reward balancing.

6.2.4 Streaming Application Performance

Streaming workloads expose scheduler responsiveness through quality metrics that integrate throughput, delay, and stability into user-perceivable outcomes. YouTube evaluation employs 200-second seek intervals every 20 seconds to prevent buffer accumulation masking scheduler behaviour; live streaming operates with approximately 0.5-second buffers that transform capacity mismatches into immediate rebuffering.

6.2.4.1 Buffered Streaming (YouTube)

DARA achieves 1.77× Average Playback Resolution (APR) improvement—substantially exceeding BPF, CPF, and ACPF. The 16% quality advantage over BPF quantifies the value of prediction granularity: 5-second horizons commit to path priorities that become suboptimal within the window, whilst DARA's continuous adjustment maintains allocation synchronised with capacity evolution.

Rebuffering ratios reveal reliability characteristics. DARA (0.84×), BPF (0.83×), and CPF (0.84×) achieve 16–17% improvement over baseline—counter-intuitively, multipath reduces rebuffering below single-path through predictive allocation avoiding buffer depletion during transitions. ACPF's aggressive throttling causes occasional depletion (1.21× rebuffering), whilst RR (0.84×) maintains baseline equivalence by reducing quality rather than rebuffering.

Quality stability metrics show DARA achieving 1.55× switch reduction, matched by ACPF's conservative throttling. BPF (1.35×) demonstrates prediction benefits, whilst reactive schedulers (CPF: 1.15×, OTIAS: 1.48×) and path-agnostic RR (0.94×) exhibit increased oscillation from capacity tracking lag.

Initial delay metrics favour prediction-based approaches: DARA (2.25×) and BPF (2.57×) enable rapid buffer filling through aggressive initial allocation, whilst reactive schedulers (CPF: 1.29×, ACPF: 1.64×) delay playback startup.

6.2.4.2 Live Streaming

Live streaming with 0.5-second buffers exposes fundamental architectural limitations. DARA achieves 4.47× APR improvement—dramatically exceeding all competitors (BPF: 1.18×, ACPF: 1.56×, CPF: 1.38×, OTIAS: 1.01×, RR: 1.00×). This 3.8× advantage over BPF demonstrates that 500 ms prediction horizons are necessary for sub-second buffer constraints—5-second horizons cannot anticipate capacity transitions before buffer depletion occurs.

Rebuffering ratios confirm catastrophic failure of non-predictive approaches. DARA achieves 0.75× (25% improvement), whilst all other schedulers exhibit 20–33× degra-

dation: RR (0.05×), OTIAS (0.04×), CPF (0.03×), ACPF (0.03×), BPF (0.04×). These ratios indicate near-continuous stalling rendering live streaming unwatchable—by the time reactive schedulers detect degradation through congestion feedback, the 0.5-second buffer has already depleted.

Latency metrics show DARA achieving 2.43× live edge distance improvement, critical for real-time applications. Competing schedulers degrade latency substantially: BPF (0.58×), ACPF (0.57×), CPF (0.46×), OTIAS (0.32×). Recovery time follows similar patterns—DARA's 1.63× improvement enables rapid restoration after transient stalls, whilst BPF (0.42×), ACPF (0.28×), and CPF (0.25×) require prolonged recovery periods spanning multiple RTT cycles.

6.2.5 Mixed Traffic and Protocol Neutrality

Combined FTP and YouTube streaming tests scheduler fairness under competing traffic with heterogeneous QoS requirements. DARA maintains consistent performance: 1.77× APR (matching isolated YouTube), 0.84× rebuffering, 1.55× switch reduction. This consistency indicates effective multi-objective reward balancing—throughput maximisation benefits FTP whilst quality sustainability benefits streaming without mutual degradation. BPF (1.53× APR, 0.83× rebuffering) and reactive schedulers maintain acceptable mixed-traffic performance, though CPF's cost prioritisation degrades streaming quality whilst ACPF's throttling causes rebuffering events. RR achieves the best mixed-traffic rebuffering (0.65×, 35% improvement) through fair allocation preventing FTP from monopolising capacity—demonstrating that fairness-oriented scheduling can benefit latency-sensitive applications at throughput cost.

6.2.6 Summary of Controlled Scenario Findings

The controlled burst evaluation reveals systematic architectural constraints across scheduler categories:

Reactive schedulers (CPF, ACPF, RR, OTIAS) exhibit observation-reaction lag preventing effective burst exploitation. Performance degrades with application latency sen-

sitivity: acceptable for file transfer, problematic for buffered streaming, catastrophic for live streaming where sub-second responsiveness is mandatory.

Coarse-grained prediction (BPF) improves upon reactive approaches for buffered applications but fails under live streaming constraints. The 5-second horizon captures slow capacity trends but cannot anticipate sub-second burst dynamics, causing BPF to exhibit reactive-scheduler failure modes when buffer headroom is insufficient.

State-of-the-art learning (Peekaboo, BLEST) achieves competitive throughput under QUIC workloads but with degraded latency. Memoryless bandit formulations and heuristic burst detection cannot exploit temporal patterns that attention mechanisms capture, yielding 15–28% throughput penalties and 2× delay degradation versus DARA.

Fine-grained prediction (DARA) achieves consistent leadership across applications and metrics, with advantages amplifying as latency sensitivity increases. The 500 ms Transformer horizon combined with 100 ms DQN inference enables burst anticipation within live streaming buffer constraints, whilst multi-objective reward learning balances throughput against stability without single-objective optimisation traps.

The controlled scenario establishes DARA's mechanism under idealised conditions; subsequent real-world trace evaluation assesses whether these advantages persist under stochastic timing and irregular burst patterns characteristic of production deployments.

6.3 Realistic Trace Evaluation

Having established the burst exploitation mechanism under controlled conditions, this section evaluates DARA across five real-world cellular traces captured from moving vehicles using Mahi-Mahi. These traces exhibit stochastic timing and irregular capacity patterns characteristic of production deployments, testing whether predictive advantages persist when burst dynamics cannot be perfectly anticipated. Traces 1 through 4 (Orange 4G, T-Mobile LTE, Verizon LTE, ATT LTE) represent increasing volatility with exploitable temporal structure, whilst Trace 5 (T-Mobile UMTS) presents extreme stochastic fluctuations that largely exceed the predictability horizon of the Transformer model. All metrics are reported as ratios versus CPF, where values exceeding 1.0 indicate improvement. The

evaluation proceeds through file transfer workloads (Section 6.3.1), streaming applications (Section 6.3.2), concurrent workload scenarios (Section 6.3.3), and cross-application synthesis (Section 6.3.4).

6.3.1 File Transfer Performance

File transfer workloads provide the clearest throughput differentiation, as greedy congestion control fully utilises available path capacity without application-layer rate adaptation. TCP file transfer operates with loss-based congestion control, whilst QUIC employs encrypted transport where inner congestion control state remains opaque to the scheduler. This distinction tests whether DARA's tunnel-layer predictions generalise across transport protocols with fundamentally different feedback mechanisms.

6.3.1.1 TCP File Transfer (Using FTP)

Figure 6.15 presents aggregate TCP file transfer performance distributions across all traces. DARA achieves the highest median throughput ratio with the tightest interquartile range, indicating consistent gains rather than high-variance opportunistic exploitation. The throughput advantage translates directly into completion time reductions visible in the download time panel, where DARA's distribution sits substantially above the CPF baseline. These gains stem from the Transformer's ability to anticipate capacity bursts within its 500 ms horizon and preemptively increase CWND fractions on paths approaching peak availability, rather than waiting for congestion control feedback to confirm the improvement.

Among the comparison schedulers, OTIAS achieves the lowest delay across all traces through its RTT-synchronised dispatch mechanism, deliberately holding packets for in-order arrival. This delay advantage comes at substantial throughput cost, as OTIAS sacrifices burst exploitation for delivery time predictability. ACPF's queue-based adaptation achieves reasonable performance on stable traces but degrades sharply under volatility, as reactive queue occupancy signals cannot distinguish transient burst accumulation from sustained congestion. BPF's supervised LSTM predictions achieve competitive through-

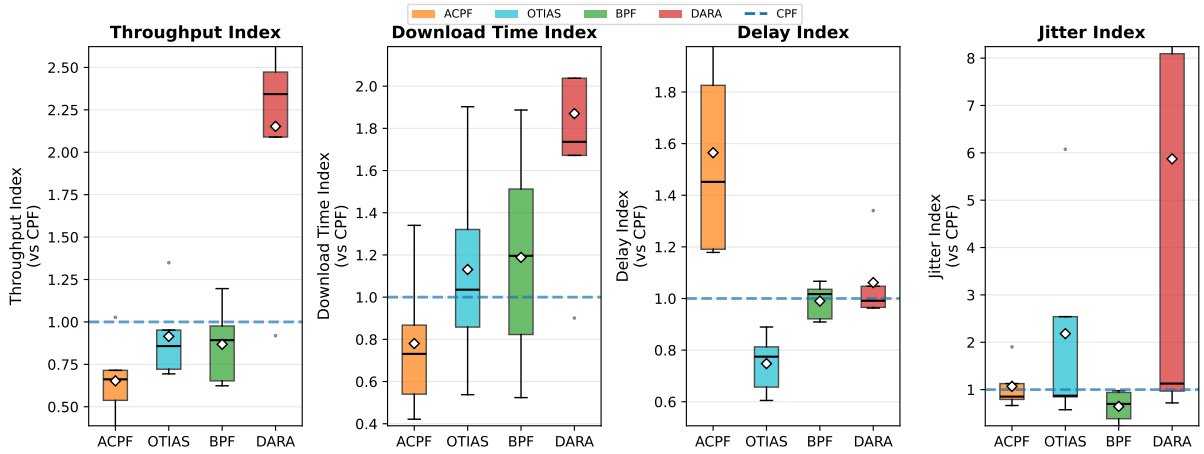

Figure 6.15: TCP file transfer (FTP) performance distributions across traces (diamond markers indicate the mean; ratio versus CPF, >1 = better). DARA achieves dominant throughput with tight variance, representing the strongest file transfer gains across both protocols evaluated; OTIAS trades throughput for delay optimisation.

put on the most stable trace but degrade monotonically as volatility increases, confirming that the 5-second prediction horizon captures slow capacity trends but cannot track sub-second burst dynamics.

The per-trace trend analysis in Figure 6.16 reveals a characteristic pattern that recurs across all application types. DARA's throughput advantage peaks on moderate-volatility traces (3 and 4), where capacity fluctuations create exploitable patterns that the Transformer can accurately forecast, then degrades on Trace 5 to near-CPF performance. This non-monotonic relationship reflects the fundamental trade-off between prediction benefit and prediction accuracy: on stable traces, limited variability reduces the opportunity for predictive gains (simple schedulers perform adequately through consistent allocation), whilst on the UMTS trace, fluctuations become too rapid and stochastic for reliable forecasting. Crucially, this degradation is graceful rather than catastrophic; DARA remains functional on Trace 5 whilst several comparison schedulers exhibit severe performance collapse.

Jitter metrics reveal an unexpected strength of fine-grained prediction. On the two most stable traces, DARA achieves substantial jitter reduction, effectively eliminating delay variance through predictive path coordination. This advantage diminishes on volatile traces, where prediction accuracy degradation manifests first in jitter before affecting

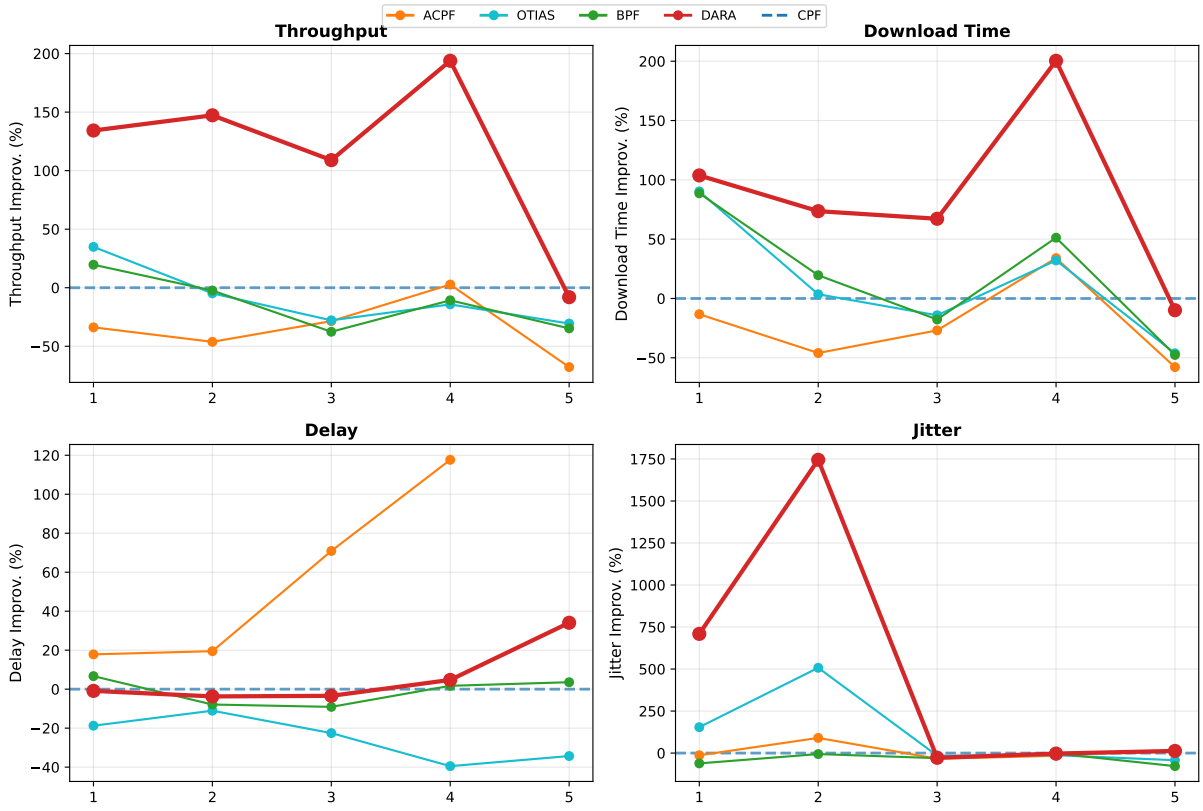

Figure 6.16: TCP file transfer (FTP) improvement versus CPF across traces (stable→volatile). DARA's throughput peaks on moderate-volatility traces before graceful degradation on Trace 5.

throughput, providing an early indicator of the forecasting limits discussed in Section 6.3.4. The multi-metric radar profile in Figure 6.17 confirms DARA's balanced characteristics: dominant throughput and download time with competitive delay and jitter, contrasting with the single-metric optimisation trade-offs exhibited by other schedulers.

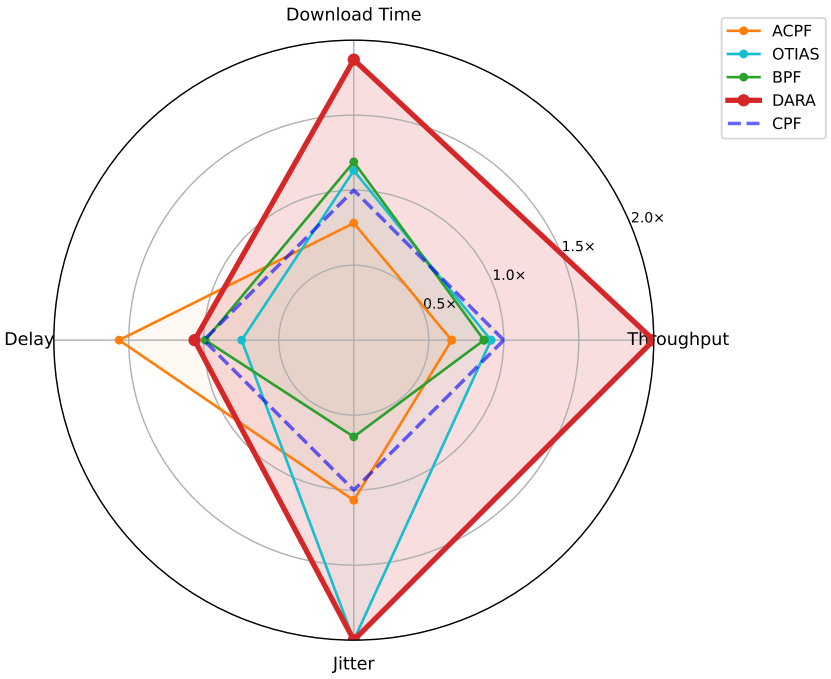

Figure 6.17: TCP file transfer (FTP) multi-metric radar profile. DARA achieves balanced performance across throughput, download time, delay, and jitter.

6.3.1.2 QUIC File Transfer (Using cURL)

QUIC evaluation introduces an additional layer of complexity: the encrypted transport prevents the scheduler from observing inner congestion control state, forcing decisions based solely on tunnel-layer metrics. Figure 6.18 presents performance distributions including Peekaboo and BLEST from the MP-QUIC testbed.

DARA achieves highest throughput on Trace 1 but exhibits non-monotonic behaviour on Traces 2 and 3, with ratios dropping below unity before recovering on Traces 4 and 5. This pattern differs from the TCP file transfer results and reflects the nested congestion control interaction between tunnel-layer BBR and QUIC's loss-based algorithm. When DARA aggressively allocates capacity to exploit a predicted burst, the resulting bursty packet delivery can trigger QUIC's internal loss detection, causing the application-layer protocol

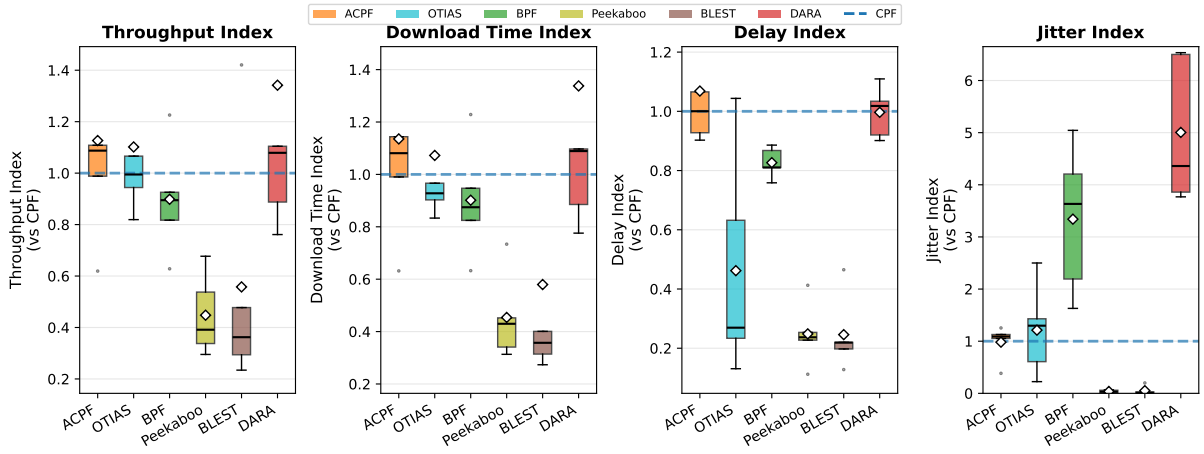

Figure 6.18: QUIC file transfer (cURL) performance distributions including Peekaboo and BLEST (diamond markers indicate the mean; ratio versus CPF, >1 = better). DARA's throughput gains are more modest than in TCP file transfer due to nested congestion control interactions, but remain competitive with CPF whilst Peekaboo and BLEST collapse on delay and jitter axes; further detail is provided in the prose.

to reduce its own sending rate independently of the scheduler's intentions. This interaction creates a negative feedback loop absent in TCP file transfer, where the application-layer protocol operates directly over the tunnel's congestion control.

The state-of-the-art learning-based schedulers reveal a more pronounced limitation. Peekaboo and BLEST exhibit severe degradation across throughput, delay, and jitter, with median throughput ratios substantially below CPF. The boxplot shows both schedulers' delay distributions collapsed near the origin, indicating systematic performance failure rather than isolated poor traces. Two factors contribute to this outcome. First, the memoryless bandit formulation (Peekaboo) and heuristic burst detection (BLEST) cannot exploit the temporal patterns that attention mechanisms capture. Second, MP-QUIC's per-stream ordering constraints compound exploration-induced reordering, creating cascading delays that MP-DCCP's unreliable delivery avoids. These results should therefore be interpreted as reflecting the combined effect of scheduling algorithm and transport protocol rather than algorithmic quality alone.

The trend analysis in Figure 6.19 confirms that DARA maintains positive or near-unity ratios across all traces despite the nested congestion control challenge, whilst Peekaboo and BLEST consistently underperform. Jitter performance strongly favours DARA, with substantial improvements on the first three traces, suggesting that even when throughput

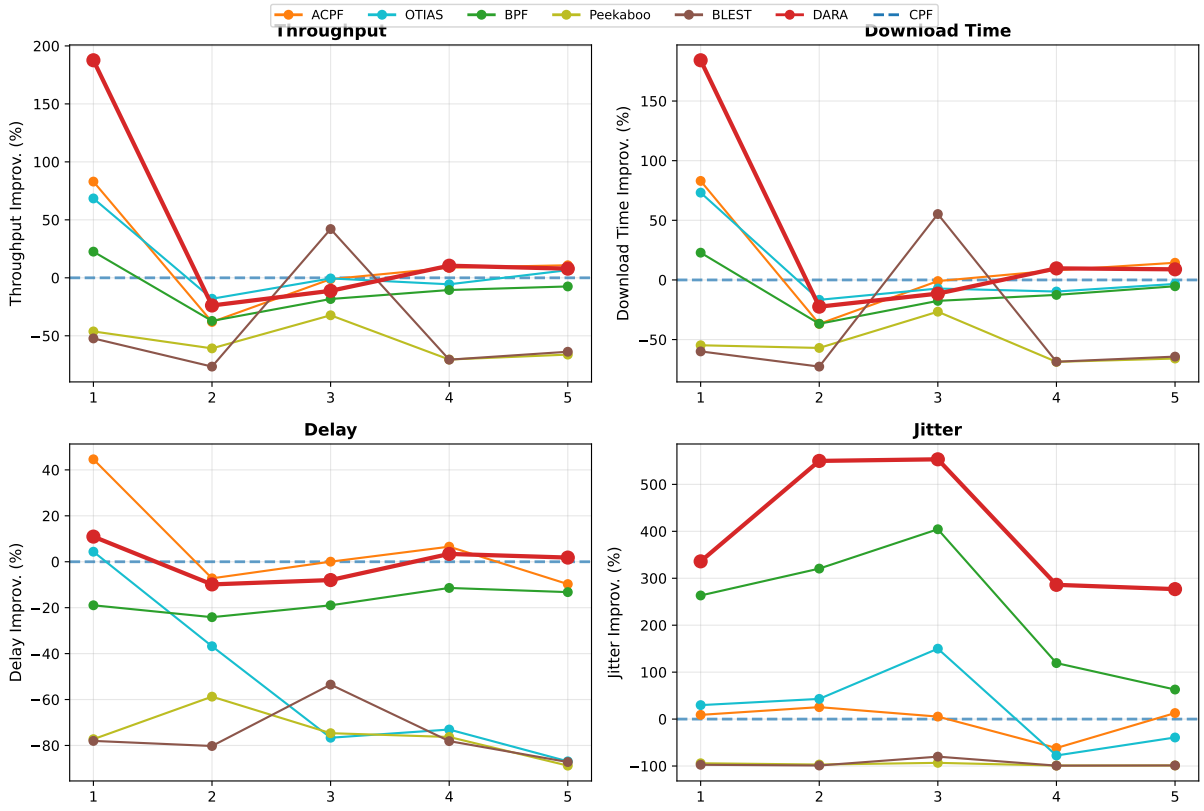

Figure 6.19: QUIC file transfer (cURL) improvement versus CPF across traces. DARA maintains positive or near-unity ratios whilst Peekaboo and BLEST consistently degrade.

gains are modest, the predictive mechanism provides meaningful delay variance control. The radar profile in Figure 6.20 visualises this contrast: DARA occupies a balanced region across all four metric axes, whilst Peekaboo and BLEST collapse to near-origin values on delay and jitter.

6.3.1.3 File Transfer Summary

Across both protocols, DARA demonstrates consistent throughput leadership on traces with exploitable temporal structure, with advantages amplifying under moderate volatility where prediction benefit and accuracy are jointly maximised. The primary limitation is the nested congestion control interaction under QUIC, which introduces non-monotonic trace-dependent behaviour absent in TCP file transfer. Delay performance remains competitive with CPF across both protocols, confirming that aggressive throughput exploitation does not sacrifice per-packet latency when guided by predictive allocation. OTIAS remains the superior choice for applications where absolute per-packet latency is the sole

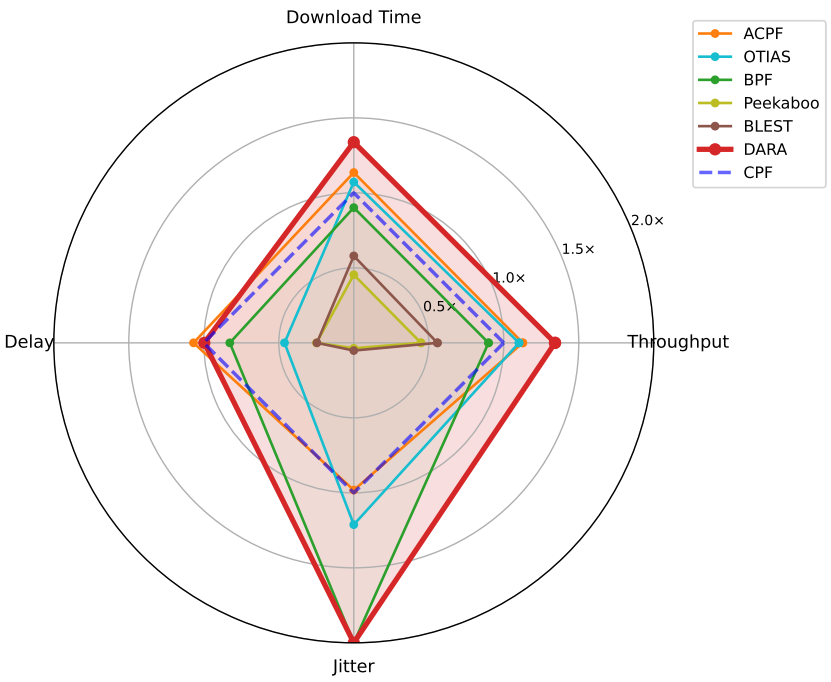

Figure 6.20: QUIC file transfer (cURL) multi-metric radar profile. DARA achieves balanced performance whilst Peekaboo and BLEST collapse to near-origin values.

objective, as its RTT-synchronised dispatch achieves consistently lower latency at the cost of reduced throughput. BPF's 5-second prediction horizon provides moderate benefits on stable traces but cannot track the sub-second dynamics that DARA's 500 ms horizon captures, with the gap widening as volatility increases.

It is worth addressing the apparent discrepancy between the controlled burst and real-world TCP file transfer results. In the controlled burst scenario (Section 6.2), DARA and CPF achieve comparable throughput improvements over single-path baseline, leaving a narrow margin between them. In the real-world traces, DARA's advantage over CPF widens considerably, with improvements ranging from roughly 60% to over 100% depending on trace. The controlled burst was designed to expose the predictive mechanism under transparent, reproducible conditions; the real-world traces reveal how that mechanism behaves when volatility is far more severe. Trace standard deviations exceed 75% of mean throughput (Figure 5.12), compared to the controlled scenario's periodic and predictable capacity pattern. Under such conditions, CPF's rigid cost-priority logic is penalised heavily, persisting with a degraded primary path until congestion window exhaustion forces overflow and missing secondary path capacity throughout. DARA's

predictive allocation adapts continuously to both paths, and its advantage scales with the unpredictability of fluctuations rather than their magnitude alone.

6.3.2 Streaming Performance

Streaming workloads expose scheduler responsiveness through quality metrics that integrate throughput, delay, and stability into user-perceivable outcomes. YouTube streaming employs adaptive bitrate selection with substantial buffering, absorbing short-term capacity variations through buffer management. Live streaming operates with approximately 0.5-second buffers, transforming capacity mismatches into immediate rebuffering. This contrast tests whether DARA's predictive mechanism provides value across the buffer depth spectrum.

6.3.2.1 YouTube Streaming

YouTube evaluation employs 200-second seek intervals every 20 seconds to prevent buffer accumulation from masking scheduler behaviour. Figure 6.21 presents QoE metric distributions across traces.

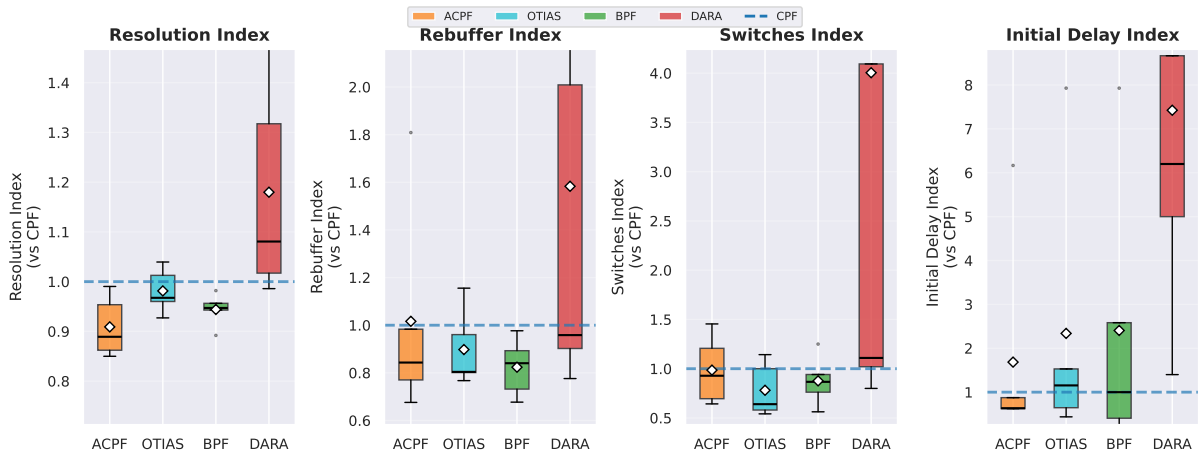

Figure 6.21: YouTube QoE distributions. DARA achieves the highest resolution at the cost of substantially increased rebuffering and quality switches, with dominant initial delay improvement.

DARA achieves the highest resolution across traces, substantially exceeding all comparison schedulers, though with considerably higher variance than alternatives. ACPF and

BPF fall below the CPF baseline on resolution, indicating that conservative or coarse-grained allocation sacrifices quality under these conditions. OTIAS remains near the baseline, reflecting its latency-optimised rather than throughput-optimised allocation strategy. The resolution improvement comes at a clear cost. DARA exhibits substantially elevated rebuffering relative to all other schedulers, which cluster near or below the baseline. This trade-off reflects the aggressive allocation strategy inherent to the predictive architecture: preemptive capacity exploitation maximises throughput and resolution but introduces periods of buffer depletion when predictions overestimate available capacity. Similarly, the quality switch rate is markedly higher for DARA than for any comparison scheduler, consistent with frequent adaptation to predicted capacity changes rather than holding a fixed quality level. All other schedulers achieve near-unity or below-baseline switch rates, indicating more stable but lower-quality playback.

The elevated rebuffering and switch rate are a direct consequence of DARA's reward structure, which prioritises throughput and preemptive adaptation over stability. When predicted capacity materialises, aggressive allocation achieves high resolution; when predictions overestimate capacity, buffer depletion follows. This behaviour contrasts with BPF, which sacrifices resolution for smoother playback, and ACPF, whose conservative throttling suppresses both quality and instability.

The trend analysis in Figure 6.22 reveals a strongly volatility-dependent pattern. Resolution improvement is concentrated on the two most stable traces, where the Transformer's forecasts are most accurate and burst exploitation translates reliably into sustained quality gains. On moderate-to-high volatility traces, the resolution advantage collapses and DARA converges towards baseline, as prediction errors reduce the reliability of aggressive allocation. The rebuffering index follows a similar pattern, with the largest improvement on stable traces before degrading under volatile conditions where capacity fluctuations outpace the prediction horizon. Quality switches exhibit the same instability, peaking sharply on one trace before returning to near-baseline levels. Across all traces and metrics, the most consistent and persistent advantage lies in initial delay, where prediction-informed startup allocation substantially outperforms all comparison schedulers regard-

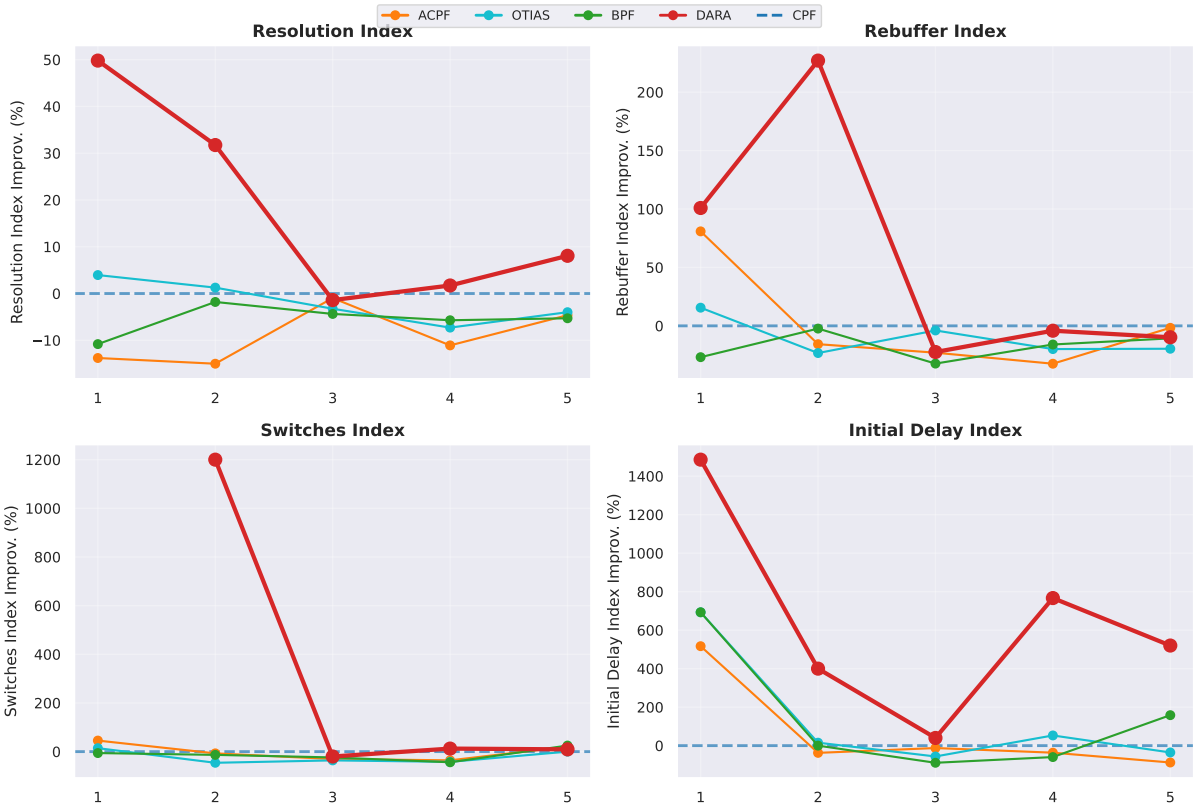

Figure 6.22: YouTube improvement versus CPF across traces. DARA achieves strong resolution and rebuffering gains on stable traces before degradation under high volatility, with initial delay improvement persisting across all conditions.

less of volatility. This asymmetry between startup performance and sustained playback stability reflects a fundamental characteristic of the architecture: preemptive allocation is most reliable over short horizons where forecast accuracy is highest, with advantages diminishing as sustained prediction quality becomes necessary.

The radar profile in Figure 6.23 visualises the asymmetric trade-off. DARA's profile extends substantially beyond all comparison schedulers on resolution and initial delay, and above the baseline on rebuffering, confirming the aggressive character of the allocation strategy. The switches axis similarly extends outward, indicating elevated quality oscillation rather than stability. Comparison schedulers occupy a compact region near the baseline across all four axes, reflecting conservative but consistent behaviour. The overall picture distinguishes DARA as a high-peak, high-variance scheduler suited to applications that prioritise quality and startup speed over playback smoothness.

Figure 6.23: YouTube QoE radar profile. DARA's profile extends outward on resolution and initial delay at the cost of elevated rebuffering and quality switches, contrasting with the compact near-baseline profiles of comparison schedulers.

6.3.2.2 Live Streaming

Live streaming with HLS and approximately 0.5-second buffers represents the most demanding evaluation scenario. Figure 6.24 presents performance distributions.

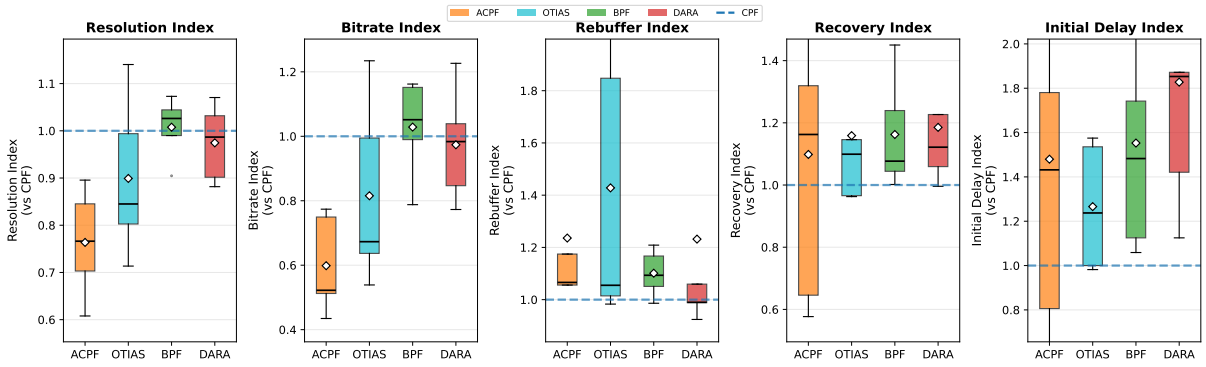

Figure 6.24: Live streaming performance distributions. DARA achieves leading initial delay with competitive resolution and controlled rebuffering.

Resolution and bitrate performance under live streaming conditions reveal a more constrained picture than the controlled burst scenario suggested. DARA achieves near-CPF resolution across traces, with modest improvements on some traces offset by modest degradation on others. This near-unity performance contrasts with the controlled scenario's substantial gains and reflects the fundamental difference between deterministic periodic bursts (where the Transformer can learn exact timing) and stochastic real-world fluctuations (where predictions are probabilistic). BPF achieves similar resolution consistency, confirming that prediction-based approaches provide live streaming stability even when throughput gains are not substantial.

The most significant real-world live streaming advantage appears in initial delay, where DARA achieves the strongest improvements across all traces. The trend analysis in Figure 6.25 shows this advantage amplifying with volatility, peaking on Trace 4, where prediction-informed initial allocation substantially outperforms reactive approaches that must discover path capacity through congestion control probing. This metric is particularly relevant for live streaming applications where startup latency directly affects user engagement.

Rebuffering ratios expose a genuine limitation. DARA achieves improvement on the most stable trace but degrades on the most volatile, reflecting the tension between predictive scheduling and extreme volatility. When capacity fluctuates faster than the 500 ms prediction horizon can track, allocation errors cause buffer depletion before corrective action is possible. This degradation remains bounded compared to other schedulers, which exhibit

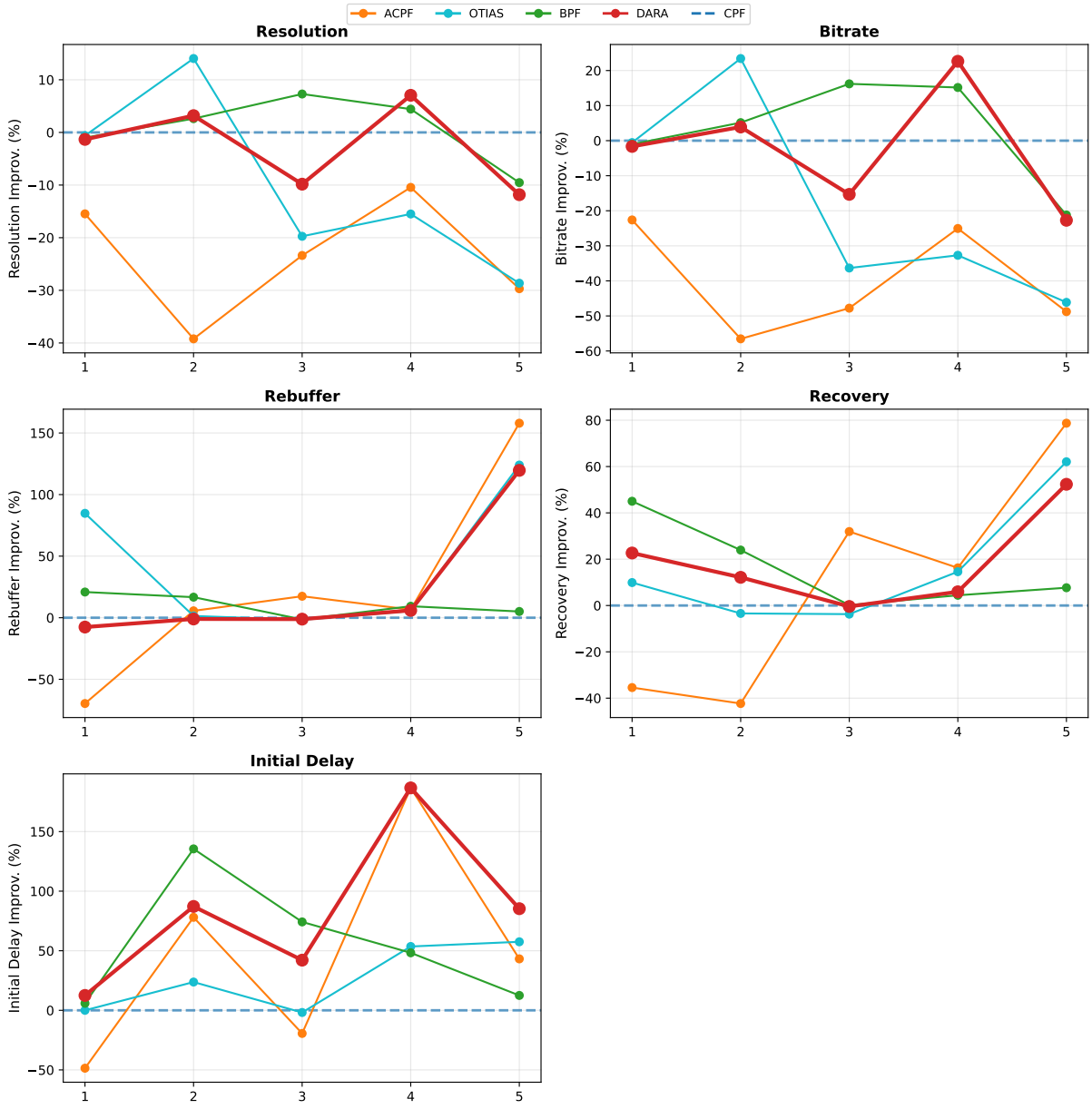

Figure 6.25: Live streaming improvement versus CPF across traces. DARA's initial delay advantage amplifies with volatility, peaking on Trace 4.

comparable or worse rebuffering under the same conditions, but it represents an honest constraint of the predictive architecture under conditions that exceed its forecasting capability.

Recovery time metrics favour DARA on most traces, as the Transformer detects capacity recovery patterns and increases utilisation before reactive schedulers observe the improvement through congestion feedback. The radar profile in Figure 6.26 shows DARA's leading initial delay with competitive performance across other axes, though without the dominant profile observed in file transfer workloads.

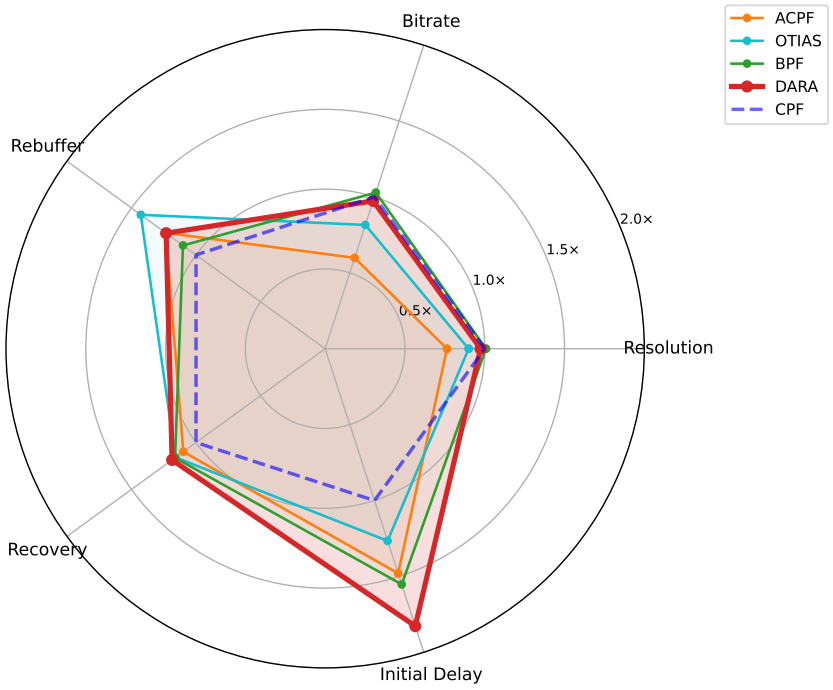

Figure 6.26: Live streaming radar profile. DARA achieves dominant initial delay with balanced performance across other metrics.

The modest real-world live streaming improvements relative to the controlled scenario highlight an inherent tension between predictive scheduling and sub-second buffer constraints under stochastic channel conditions. When capacity fluctuates faster than the 500 ms prediction horizon can reliably track, buffer depletion occurs before corrective allocation is possible regardless of scheduling sophistication. A promising direction for future work is joint adaptation to both traffic profile and channel profile: a scheduler aware that the active application is live streaming could apply more conservative allocation bounds to prevent buffer depletion, accepting throughput reduction in exchange for rebuffering

elimination, whilst the same scheduler could apply aggressive burst exploitation for file transfer workloads where momentary allocation errors carry no penalty. The current DARA formulation treats all applications identically through the multi-objective reward function; application-aware reward shaping would allow the same predictive architecture to adapt its risk tolerance to the specific QoS requirements of the active traffic type.

6.3.2.3 Streaming Summary

The streaming results present a more nuanced picture than file transfer workloads. YouTube's adaptive bitrate layer compresses DARA's throughput advantages into modest resolution gains, though with notable consistency across volatility conditions. Live streaming reveals initial delay as DARA's strongest real-world streaming contribution, with resolution and rebuffering performance remaining competitive but not dominant. The controlled scenario's dramatic live streaming advantages do not fully transfer to real-world conditions, as stochastic capacity patterns reduce prediction accuracy below the threshold needed for sub-second buffer management. This gap between controlled and real-world performance is expected and validates the experimental methodology: the controlled scenario isolates the mechanism, whilst real-world evaluation reveals its practical operating envelope.

6.3.3 Concurrent Workload Performance

Concurrent workload evaluation tests scheduler fairness under competing traffic with heterogeneous QoS requirements. Two scenarios are examined: simultaneous QUIC and TCP file downloads (testing protocol-neutral capacity allocation) and concurrent YouTube streaming with background FTP (testing elastic-streaming competition).

6.3.3.1 QUIC+TCP Concurrent Transfer

Simultaneous QUIC and TCP downloads test scheduler behaviour under mixed-protocol competition where distinct congestion control algorithms compete for shared tunnel capacity. Figure 6.27 presents per-flow and aggregate performance distributions.

The terms *foreground* and *background* are used here solely to distinguish the two flows

for reporting purposes and do not imply any scheduling priority or quality-of-service differentiation between them; both flows receive equal treatment from the scheduler.

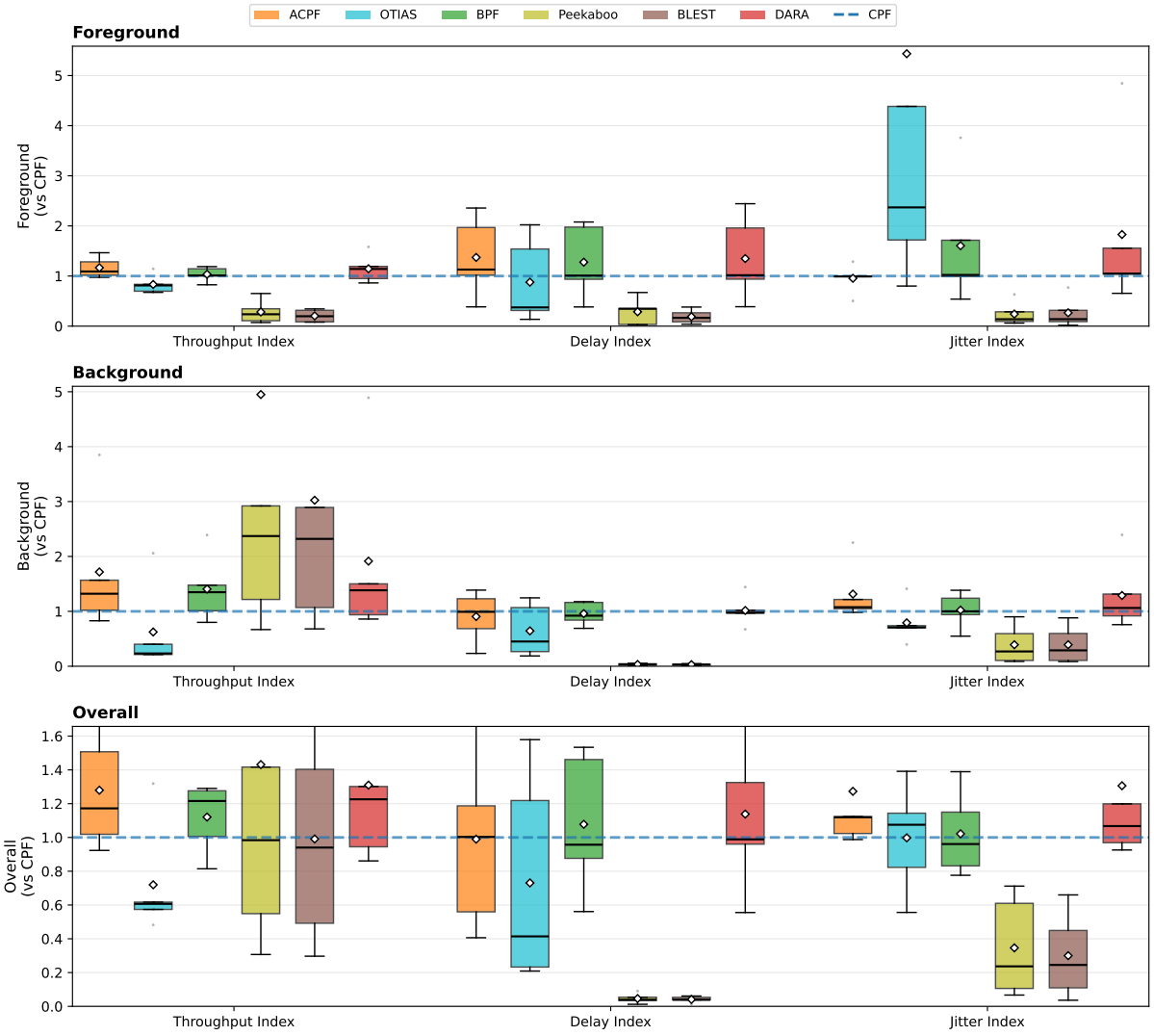

Figure 6.27: QUIC+TCP concurrent transfer performance distributions showing foreground QUIC, background TCP, and overall metrics. DARA achieves balanced flow allocation whilst Peekaboo and BLEST exhibit severe foreground starvation.

The foreground QUIC flow reveals DARA's balanced allocation strategy. DARA maintains competitive or improved QUIC throughput on most traces with a tight interquartile range, indicating consistent foreground flow protection despite TCP competition. This balance contrasts sharply with Peekaboo, which exhibits severe QUIC starvation: throughput ratios collapse to near-zero on several traces, indicating systematic foreground flow deprivation. The background TCP flow explains this disparity. Peekaboo achieves extreme TCP throughput gains on certain traces, but combined with the corresponding

QUIC degradation, this indicates severe flow unfairness where exploration-based scheduling starves the foreground flow to maximise background throughput. BLEST exhibits similar patterns, confirming that heuristic burst detection cannot maintain flow fairness under mixed-protocol competition.

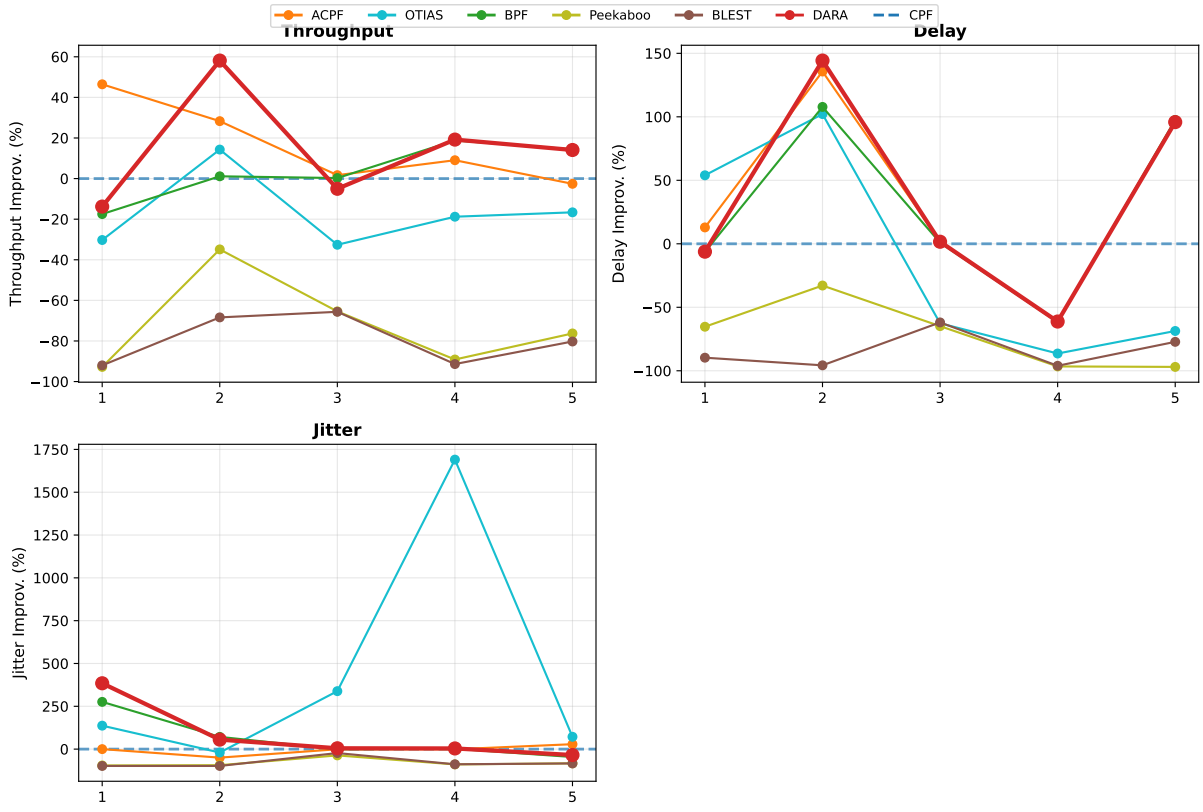

Figure 6.28: QUIC+TCP improvement versus CPF across traces. DARA maintains positive or near-unity ratios whilst Peekaboo and BLEST consistently degrade on delay and jitter.

The trend analysis in Figure 6.28 confirms DARA's consistency across volatility conditions, with positive or near-unity throughput ratios on all traces. Delay and jitter trends show DARA achieving competitive or improved performance, whilst Peekaboo and BLEST exhibit consistent degradation reflecting their inability to maintain flow coordination under concurrent load. OTIAS achieves an anomalous jitter improvement on one trace through its RTT-synchronised dispatch, but with corresponding throughput degradation illustrating the familiar stability-throughput trade-off.

The radar profile in Figure 6.29 visualises the overall performance contrast. DARA occupies a balanced region across throughput, delay, and jitter axes, whilst Peekaboo and

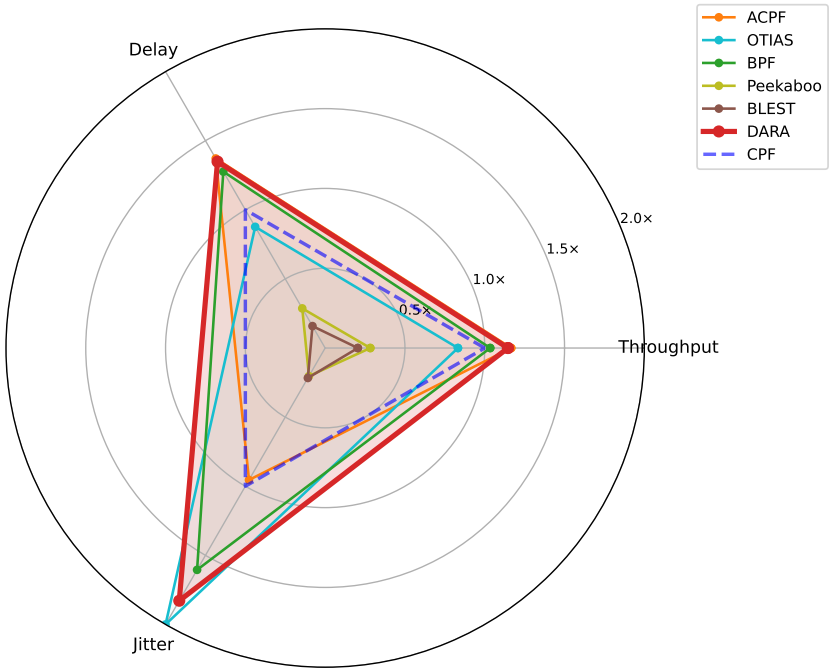

Figure 6.29: QUIC+TCP overall radar profile. DARA achieves balanced performance across all metrics whilst Peekaboo and BLEST collapse on delay and jitter axes.

BLEST collapse to near-origin values on delay and jitter, their profiles reduced to narrow slivers indicating single-metric focus at the expense of overall QoS. As noted in Section 6.3.1, these results reflect combined scheduling algorithm and transport protocol effects, as the MP-QUIC framework's per-stream ordering constraints may amplify exploration-induced reordering beyond what the scheduling algorithm alone would produce.

6.3.3.2 YouTube+FTP Concurrent Streaming

Concurrent YouTube streaming with background FTP tests elastic-streaming competition where buffered video playback competes with greedy bulk transfer. Figure 6.30 presents per-flow and aggregate performance distributions.

DARA maintains YouTube resolution ratios that match isolated streaming performance, indicating effective flow isolation under concurrent load. The multi-objective reward function's quality sustainability component (R_{quality}) appears to prevent the FTP flow from monopolising capacity at the expense of streaming quality, a failure mode visible in schedulers that optimise throughput without application awareness. Background FTP through-

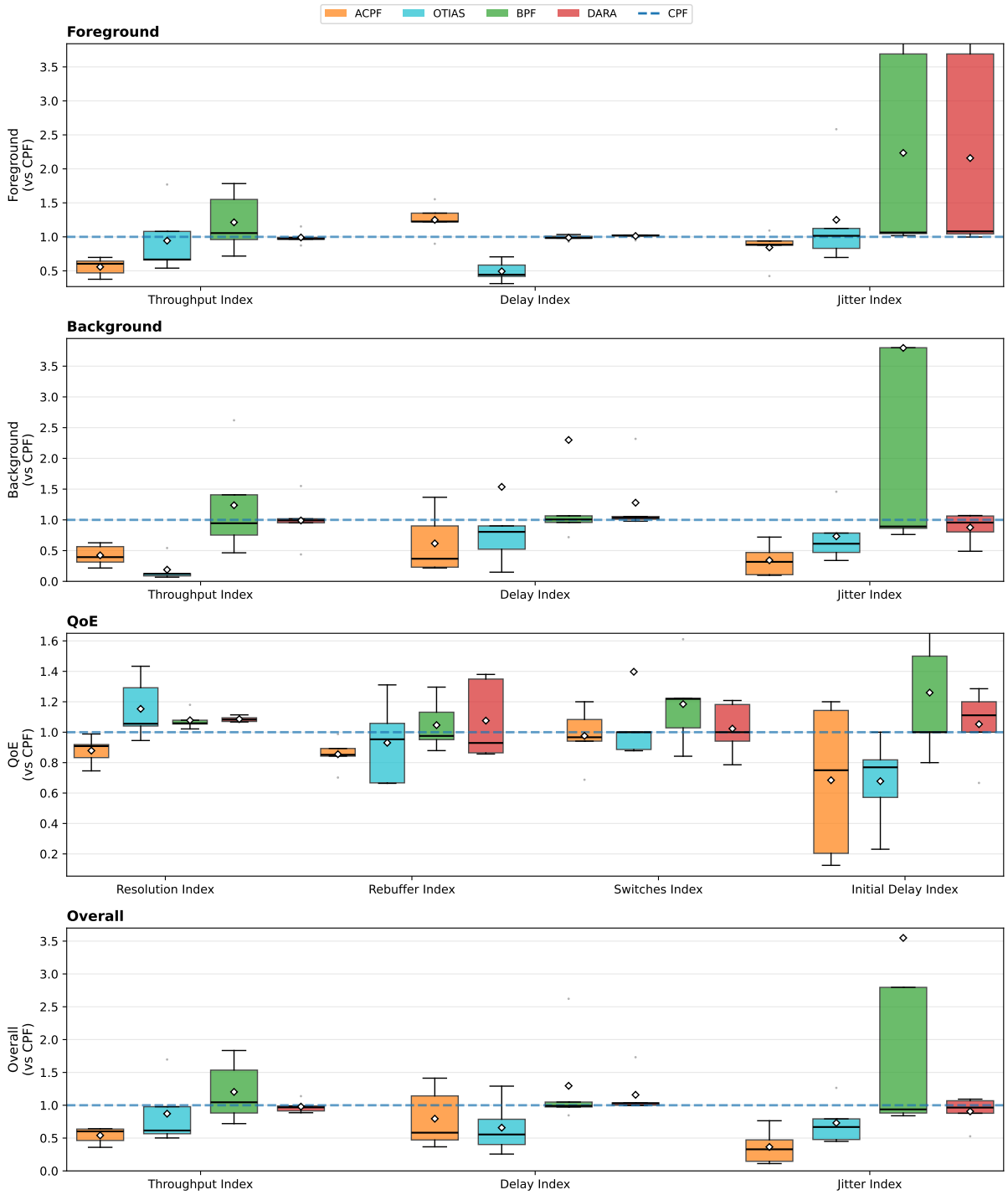

Figure 6.30: YouTube+FTP concurrent streaming performance distributions showing foreground YouTube QoE, background FTP throughput, and overall metrics. DARA maintains consistent resolution improvement whilst achieving competitive FTP throughput.

put remains competitive on most traces, with degradation confined to the UMTS trace where prediction accuracy is insufficient to serve both flows effectively.

The trend analysis in Figure 6.31 reveals volatility-dependent patterns across the competing flows. BPF achieves higher peak FTP throughput on stable traces but with increased jitter variance and degradation on volatile traces. DARA's throughput trend remains closer to the baseline but with substantially better consistency. Resolution trends confirm DARA's steady improvement across all traces, whilst OTIAS achieves higher peak resolution on stable conditions but at the cost of degradation under volatility. Jitter trends show DARA's strongest concurrent workload advantage on stable traces, with the benefit diminishing as volatility increases.

The radar profile in Figure 6.32 confirms that DARA's multi-objective optimisation achieves a broader balanced profile than single-objective competitors. OTIAS extends further on resolution but collapses on throughput; BPF achieves strong jitter performance but with reduced resolution consistency. DARA's profile encompasses the largest balanced area, though the advantage over BPF is more modest than in file transfer workloads, reflecting the diminishing returns of fine-grained prediction when YouTube's adaptive bitrate algorithm already provides application-layer smoothing.

6.3.3.3 Concurrent Workload Summary

The concurrent evaluation reveals DARA's multi-objective reward function as a meaningful differentiator from single-objective schedulers. Flow fairness under mixed-protocol competition represents a clear advantage over Peekaboo and BLEST, which exhibit pathological flow starvation. Under elastic-streaming competition, DARA maintains streaming quality comparable to isolated conditions whilst achieving competitive background throughput. The advantage over BPF and reactive schedulers is more modest in concurrent scenarios than in isolated file transfer, as the interaction between competing flows introduces dynamics that attenuate the benefits of fine-grained prediction.

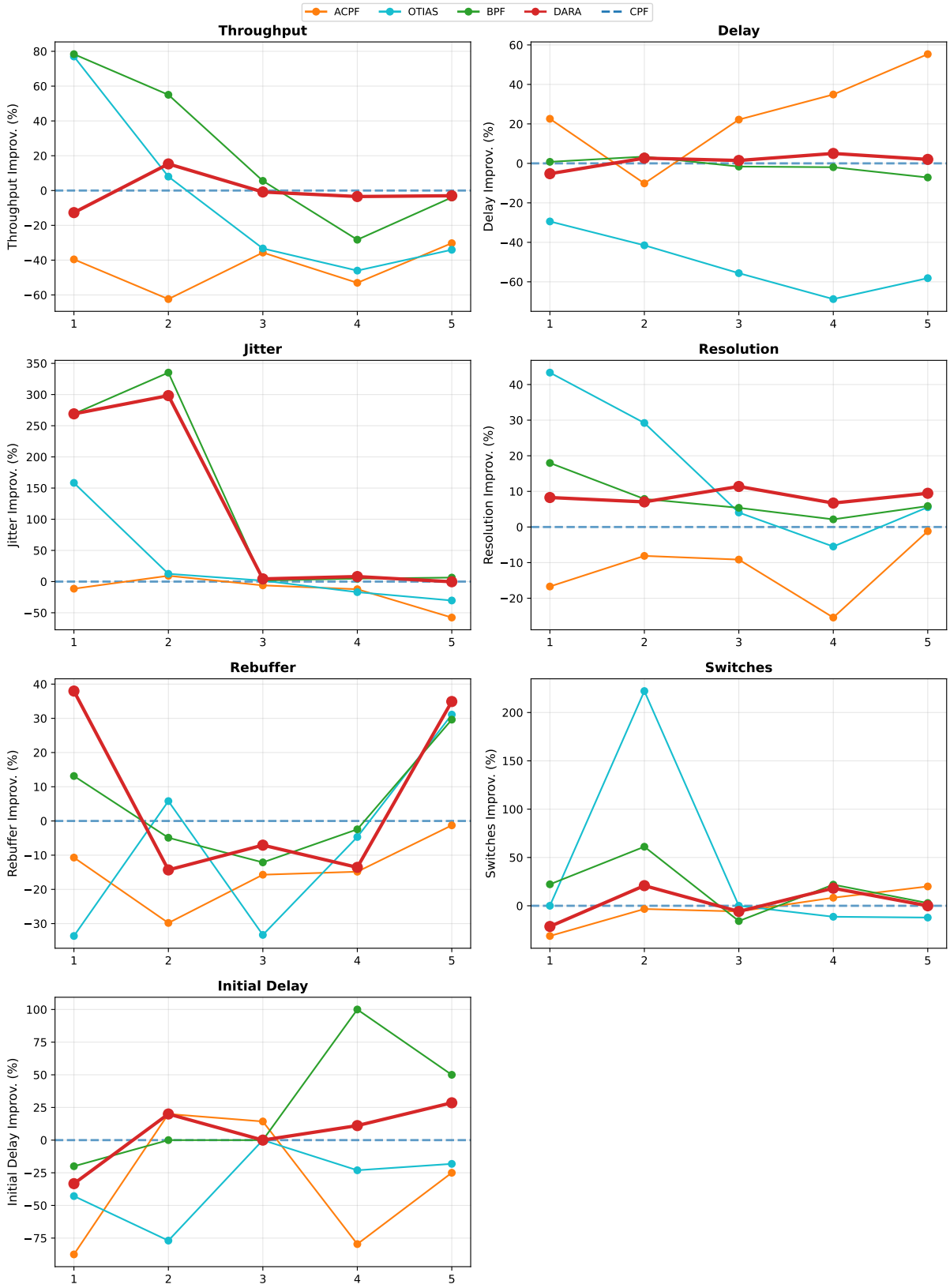

Figure 6.31: YouTube+FTP improvement versus CPF across traces. DARA achieves consistent positive resolution improvement across volatility conditions with balanced FTP throughput.

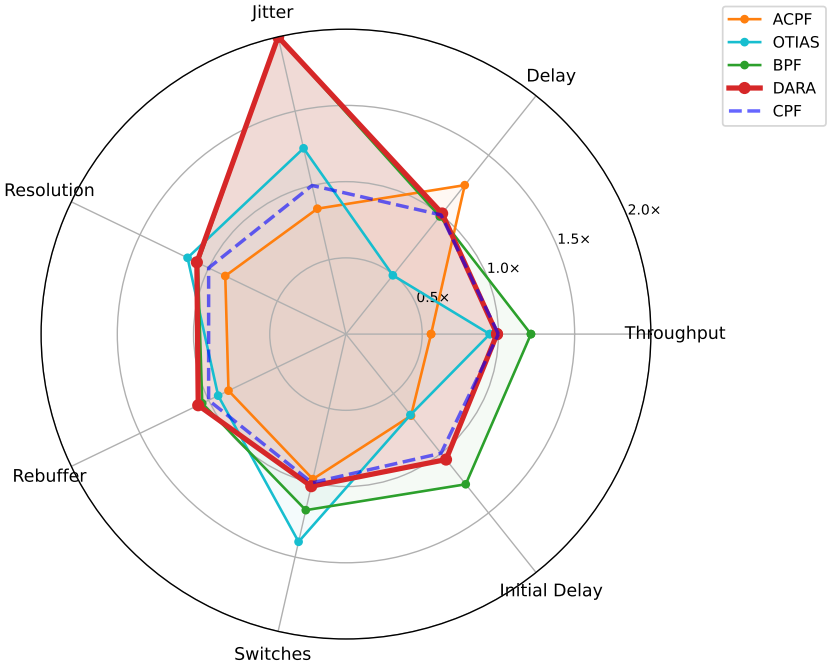

Figure 6.32: YouTube+FTP overall radar profile. DARA balances streaming QoE with background transfer performance.

6.3.4 Cross-Application Synthesis

The preceding subsections have analysed DARA's performance against all comparison schedulers across individual application types. This section synthesises those results to identify patterns that cut across applications and traces, with two specific aims. First, it characterises the volatility-dependent performance envelope that determines when DARA's predictive architecture provides value and when it converges towards baseline behaviour. Second, it evaluates DARA against the state-of-the-art schedulers (BPF, Peekaboo, BLEST, OTIAS) whose per-application results are distributed across Sections 6.3.1–6.3.3, consolidating the cross-application comparison in one place. The CPF baseline is used as the normalisation reference throughout, as it represents the current deployment default in ATSSS-HL implementations and provides a stable comparison anchor across heterogeneous application types.

Figure 6.33 presents DARA's improvement versus CPF across all test scenarios and traces, enabling identification of systematic patterns and architectural constraints.

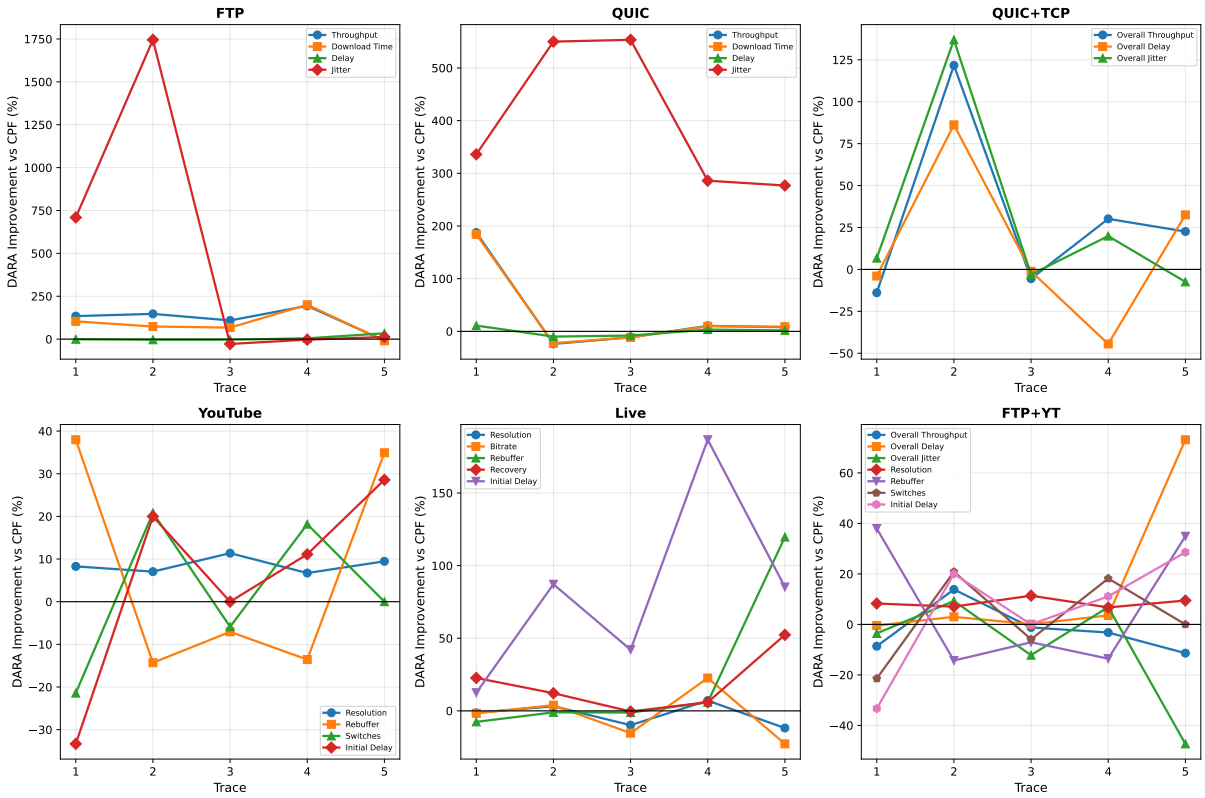

Figure 6.33: DARA improvement versus CPF across all test scenarios and traces. Positive values indicate DARA outperformance; the pattern reveals peak advantage on moderate-volatility traces with graceful degradation under extreme conditions.

6.3.4.1 Volatility-Dependent Performance Envelope

A consistent pattern emerges across all applications and metrics: DARA's relative advantage follows an inverted-U relationship with trace volatility. This pattern arises from the interaction between two opposing forces.

On stable traces (1 and 2), predictable capacity enables accurate Transformer forecasting, but limited variability constrains the magnitude of exploitable opportunities. Simple reactive schedulers achieve reasonable performance through consistent allocation, and the prediction overhead provides diminishing marginal returns. DARA's advantages on these traces are concentrated in jitter reduction rather than throughput, as the predictive mechanism eliminates delay variance caused by reactive path switching without substantial capacity gains.

On moderate-volatility traces (3 and 4), capacity fluctuations create exploitable patterns with sufficient temporal structure for the Transformer's attention mechanism to capture within its 500 ms horizon. The prediction benefit and forecast accuracy are jointly maximised, producing the largest throughput gains. This regime represents the target operating environment for the DARA architecture.

On the UMTS trace (5), capacity fluctuations become too rapid and stochastic to forecast reliably. The Transformer's learned patterns default to conservative allocation, preventing catastrophic failure but reducing gains to near-zero or slightly negative. This behaviour is by design: the multi-objective reward function's stability penalty ($R_{\text{stability}}$) discourages aggressive allocation under uncertainty, preferring CPF-equivalent performance to the risk of severe degradation from confidently wrong predictions. The graceful degradation on this trace, where DARA achieves near-CPF performance whilst several comparison schedulers collapse, validates this conservative fallback behaviour.

6.3.4.2 Application-Dependent Advantages

The magnitude and nature of DARA's advantages vary systematically with application characteristics.

File transfer workloads exhibit the largest throughput gains, as greedy congestion con-

trol directly translates burst exploitation into measurable throughput. FTP benefits most, as the single-layer congestion control creates a direct relationship between scheduler allocation and observed performance. QUIC benefits are attenuated by the nested congestion control interaction described in Section 6.3.1, introducing non-monotonic trace-dependent behaviour that limits consistent improvement.

Buffered streaming (YouTube) compresses throughput advantages through adaptive bitrate smoothing, producing modest but consistent resolution gains. The primary value of prediction in this regime is quality stability rather than peak quality: DARA achieves reliable improvement across all volatility conditions, whilst comparison schedulers exhibit trace-dependent performance that could produce inconsistent user experience in production deployment.

Live streaming reveals initial delay as DARA's strongest real-world streaming contribution, with the advantage amplifying under volatility. Resolution and rebuffering performance remain competitive but not dominant, reflecting the gap between the controlled scenario's deterministic bursts and real-world stochastic patterns. The sub-second buffer constraint remains challenging for all evaluated schedulers under extreme volatility.

Concurrent workloads demonstrate DARA's flow fairness through multi-objective optimisation, preventing the pathological flow starvation exhibited by Peekaboo and BLEST. The advantage over single-objective schedulers is most pronounced under mixed-protocol competition, where DARA's balanced allocation maintains both foreground and background flow quality.

6.3.4.3 Comparison with State-of-the-Art

The evaluation reveals architectural distinctions between scheduling paradigms that are more informative than aggregate performance rankings. To contextualise these findings, the observed behaviours are linked to results and limitations reported in the schedulers' source literature.

Reactive schedulers (CPF, ACPF) provide the performance baseline. CPF's conservative cost-priority allocation proves surprisingly competitive under extreme volatility,

where prediction-based approaches lose their forecasting advantage. ACPF's queue-based adaptation improves upon CPF on stable traces but degrades under volatility, consistent with Pieska *et al.*'s [25] characterisation of the γ-scaling mechanism as reactive to observed queue occupancy rather than predictive of capacity transitions.

Coarse-grained prediction (BPF) improves upon reactive approaches for all applications except under high volatility. The 5-second horizon captures slow capacity trends but cannot anticipate sub-second burst dynamics, causing BPF to exhibit reactive-scheduler failure modes when conditions fluctuate faster than its prediction window. This constraint is consistent with the fixed-horizon limitation identified in Section 3.1.8 as a motivating gap for the DARA architecture.

Online learning (Peekaboo) achieves competitive throughput under certain QUIC conditions but with severe delay and jitter degradation. Wu *et al.* [14] explicitly note that Peekaboo's memoryless bandit formulation cannot exploit long-range temporal dependencies, a constraint that motivated the subsequent FALCON extension [98] as a direct improvement over Peekaboo by the same authors. The flow starvation observed under concurrent load (Section 6.3.3) reflects a fundamental limitation of throughput-only reward in contextual bandit formulations, which cannot enforce flow fairness across competing streams. As noted in Section 6.3.1, the observed Peekaboo results reflect joint protocol-scheduler effects due to the MP-QUIC testbed constraint, and algorithmic conclusions should be interpreted accordingly.

Heuristic estimation (BLEST, OTIAS) provides specialised advantages. OTIAS's delay optimisation remains unmatched for applications where latency is the sole concern, though its throughput cost is substantial. BLEST's blocking estimation fails under the volatile conditions where prediction would be most valuable. Ferlin *et al.* [4] acknowledge that BLEST's burst detection heuristics degrade when path characteristics oscillate rapidly, and Dong *et al.*'s survey [11] notes convergence of estimation-based schedulers including BLEST under certain bottleneck configurations, consistent with the performance collapse observed on Traces 3 and 4 in the present evaluation.

Fine-grained prediction (DARA) achieves consistent leadership across applications on

traces with exploitable temporal structure. The primary limitations—nested congestion control interaction under QUIC, ABR compression under buffered streaming, and the fundamental forecasting horizon constraint under extreme stochastic volatility—define the practical operating envelope rather than representing design failures. Each limitation is an expected consequence of the architecture operating outside its target regime, and Section 3.1.8 identifies addressing the QUIC interaction and extending the prediction horizon as directions for future work.

Chapter 7

Conclusion

This chapter summarises the contributions, findings, and limitations of the research on predictive multipath scheduling in volatile heterogeneous networks, and identifies directions for future investigation.

7.1 Summary of Contributions

DARA, a Transformer-informed deep reinforcement learning framework for multipath scheduling in volatile heterogeneous networks, was presented and evaluated. The principal contributions comprise:

- **Predictive scheduling architecture:** Integration of Transformer-based time-series prediction with DQN-based rate allocation enables proactive capacity exploitation. The 500 ms prediction horizon matches typical cellular path coherence times whilst providing actionable forecast accuracy, with the 100 ms inference cycle enabling sub-RTT adaptation to capacity fluctuations.
- **Multi-objective reward formulation:** The six-component reward function achieves balanced trade-offs across throughput, delay, quality sustainability, preemptive adaptation, stability, and idleness prevention, balancing competing objectives that single-metric heuristics cannot simultaneously optimise.
- **Protocol-agnostic design:** Evaluation across FTP, QUIC, adaptive streaming,

live delivery, and concurrent workloads demonstrates effectiveness regardless of inner protocol characteristics. Tunnel-layer prediction suffices for effective scheduling even without application-layer visibility, achieving substantial throughput gains over encrypted QUIC where state-of-the-art intelligent schedulers degrade.

- **Rigorous design validation:** Systematic ablation studies confirm the contribution of each architectural component, with statistical significance established across all pairwise comparisons. Predictor architecture selection, Transformer depth optimisation, reward weight calibration, and action space granularity analysis collectively validate that design decisions are individually justified rather than jointly confounded.

- **Graceful degradation:** Under extreme mobility conditions where competing schedulers exhibit catastrophic failure, DARA maintains functional performance through learned conservative fallback when prediction confidence decreases, avoiding the severe degradation exhibited by reactive approaches.

7.1.1 Key Findings

Experimental evaluation across controlled burst scenarios and real-world mobility traces yields the following findings:

- **Throughput dominance:** DARA achieves substantial FTP throughput improvements across all traces with exploitable temporal structure, with corresponding download time reductions that amplify under moderate-to-high volatility. The largest gains occur where capacity fluctuations create patterns sufficiently regular for the Transformer to forecast yet sufficiently rapid to disadvantage reactive schedulers.

- **State-of-the-art comparison:** Under encrypted QUIC traffic, DARA achieves several-fold throughput improvement on stable traces where both Peekaboo and BLEST degrade below the reactive baseline. This disparity reflects fundamental

architectural differences: memoryless bandit formulations and heuristic burst detection cannot exploit the temporal patterns that attention mechanisms capture, whilst MP-QUIC's per-stream ordering constraints amplify exploration-induced reordering.

- **Delay variance control:** Predictive path coordination eliminates delay variance on stable traces, with jitter improvements of an order of magnitude on FTP and several-fold on QUIC. This advantage proves particularly robust, persisting even on traces where throughput gains are modest, and contrasts with Peekaboo's consistent jitter degradation.

- **Streaming quality:** Under YouTube streaming, DARA achieves the highest resolution across traces at the cost of increased rebuffering and a markedly higher quality switch rate, reflecting an aggressive allocation strategy that prioritises peak quality and fast startup over playback stability. The strongest and most consistent streaming advantage lies in initial delay reduction, which persists across all volatility conditions. Live streaming benefits primarily through reduced startup latency, whilst resolution and rebuffering performance remain competitive but not dominant under stochastic real-world conditions.

- **Flow fairness:** Under concurrent mixed-protocol load, DARA maintains balanced allocation across foreground and background flows, whilst Peekaboo exhibits severe foreground starvation. The multi-objective reward function prevents pathological flow deprivation without requiring explicit priority configuration.

- **Volatility-performance relationship:** Relative improvement follows an inverted-U pattern, peaking on moderate-to-high volatility traces before graceful degradation under extreme stochastic conditions. This validates the architectural targeting of challenging high-mobility scenarios where reactive schedulers fundamentally cannot adapt, whilst confirming that prediction-based approaches provide limited additional benefit when conditions are either too stable or too chaotic.

- **Design integrity:** Component ablation studies establish that each architectural

element contributes measurably to overall performance. Removal of Transformer predictions causes the DQN to degenerate to near-static allocation with negligible preemptive behaviour. Coarser action discretisation reduces both reward and preemptive rate, whilst finer granularity introduces exploration burden without commensurate gain. Reward weight sensitivity analysis confirms that the optimised weighting reflects genuine objective trade-offs rather than scale compensation artefacts. These ablations provide confidence that DARA's performance stems from principled architectural design rather than fortuitous hyperparameter selection.

- **Consistency advantage:** Tight interquartile ranges across all metrics versus high-variance competitors indicate reliable deployment performance rather than opportunistic gains.

7.1.2 Deployment Recommendations

Experimental findings indicate that DARA provides maximum benefit for:

- **High-mobility scenarios:** Vehicular connectivity, rail transport, and drone communications where path characteristics change on sub-second timescales. The prediction horizon matches typical cellular handover dynamics, with peak performance demonstrated on moderate-to-high volatility traces.

- **QoE-sensitive applications:** Video streaming, conferencing, and real-time collaboration where capacity smoothness affects user experience beyond raw throughput. Resolution improvement and startup acceleration translate directly to improved user satisfaction.

- **Mixed-criticality workloads:** Concurrent streaming with background transfer, where the multi-objective reward provides fair allocation without explicit flow prioritisation, avoiding the foreground starvation exhibited by exploration-based schedulers.

- **Encrypted transport aggregation:** HTTPS, QUIC, and Virtual Private Network (VPN) traffic where inner protocol state remains unobservable. Tunnel-layer

prediction demonstrates substantial throughput improvement despite application-layer opacity.

Limited additional benefit is expected in stable low-mobility scenarios, single-objective optimisation tasks where simpler heuristics may outperform on specific metrics, or severely resource-constrained deployments where inference overhead proves prohibitive.

7.1.3 Limitations

- **Computational overhead:** Transformer inference introduces latency unsuitable for per-packet decisions at line rate. The current implementation operates at 100 ms intervals, sufficient for exploiting capacity variations on cellular timescales but potentially missing finer-grained opportunities.
- **Training data dependency:** Offline Transformer training requires representative traces from target deployment conditions; transfer across substantially different network characteristics requires validation and potential fine-tuning.
- **Two-path assumption:** Current evaluation focuses on dual-path cellular-WiFi scenarios; generalisation to N-path configurations requires action space redesign with corresponding sample complexity implications.
- **Extreme volatility ceiling:** Performance degrades to near-baseline levels on the most volatile trace, indicating prediction horizon limitations when capacity changes exceed 500 ms coherence times.
- **Userspace implementation:** Transformer and DQN calculations are performed in userspace with kernel-userspace synchronisation via character device driver, increasing inference time relative to a potential kernel-space integration.

7.1.4 Future Work

- **Latency-critical applications:** Video conferencing, cloud gaming, and interactive applications with sub-100 ms latency requirements warrant investigation of reduced

prediction horizons, lightweight model architectures, and hardware acceleration.

- **Online adaptation:** Combining offline pretraining with online fine-tuning via experience replay would enable adaptation to novel network conditions without full retraining, addressing the training data dependency limitation.

- **Hierarchical scheduling:** Separating path selection from packet distribution would preserve prediction benefits without per-packet inference overhead, operating coarse-grained prediction-informed decisions at 100 ms intervals alongside fine-grained queue-based dispatch at packet timescales.

- **N-path generalisation:** Extending beyond dual-path scenarios to support arbitrary path counts would broaden deployment applicability, potentially through factored action representations or attention-based path selection.

- **Mathematical bounds:** Deriving theoretical performance bounds via competitive analysis would quantify how closely DARA approaches optimal online scheduling, identifying remaining optimisation opportunities and fundamental limits.

- **Energy-aware scheduling:** Incorporating device energy state and per-interface power consumption into the reward function would enable battery-conscious allocation for mobile devices where transmission energy varies substantially across interfaces.

- **5G-specific optimisation:** Exploiting network slicing information, predictable QoS guarantees, and base station handover signalling could enhance prediction accuracy for next-generation deployments where network-side information augments terminal-side observations.

7.2 Critical Reflection on Research Objectives

7.2.1 Prediction of Congestion Metrics

The objective of predicting CWND and RTT trajectories has been substantially achieved. The Transformer model attains the lowest NRMSE among all evaluated architectures, and the no-transformer ablation (Section 6.1.6) confirms prediction is load-bearing: its removal degenerates the DQN to near-static allocation. However, all accuracy metrics are obtained under deterministic Mahi-Mahi trace replay; whether the learned temporal patterns generalise to live network conditions where capacity fluctuations arise from radio resource management and neighbour cell interference remains unvalidated. Furthermore, DARA outperforming the oracle DQN supplied with ground-truth future values suggests that prediction accuracy per se may not be the relevant success criterion; the critical property appears to be the provision of stable state representations under which the DQN learns consistent policies.

7.2.2 Predictive Scheduling Architecture

This objective has been convincingly achieved for the targeted regime. The controlled burst experiments (Section 6.2) demonstrate a 74% reduction in maximum delay through preemptive allocation adjustment, and trajectory analysis (Section 6.1.7) shows approximately two-thirds of congestion events handled preemptively with graduated ϕ responses rather than binary throttling. The principal caveat is the inverted-U relationship between volatility and relative improvement (Section 6.3): on the most volatile trace, where capacity changes exceed the 500 ms coherence time, performance degrades to near-baseline levels. This delineates the operating envelope more narrowly than the original objective implied and confirms that predictive scheduling provides limited benefit when channel conditions are either too stable for prediction to add value or too stochastic for 500 ms forecasts to remain accurate.

7.2.3 Multi-Objective Reward Formulation

The six-component reward function produces balanced behaviour across throughput, delay, quality, and fairness, and the weight sensitivity analysis (Section 6.1.3) confirms the optimised configuration reflects genuine trade-offs rather than scale compensation artefacts. The claim of Pareto-optimal trade-offs stated in Section 1.3 is, however, not formally substantiated: no Pareto front analysis was conducted, and it remains undemonstrated that the learned policies occupy the efficiency frontier rather than merely performing adequately along a weighted scalar projection of the objective space. Additionally, the weight configuration is specific to the training dataset; the thesis does not establish transferability across fundamentally different deployment contexts.

7.2.4 Protocol-Agnostic Effectiveness

This objective is achieved with notable strengths and one significant mixed result. The QUIC evaluation (Section 6.3.3) provides the strongest validation, with 188% throughput improvement on stable traces where Peekaboo degrades by 46%, confirming that tunnel-layer prediction suffices for effective scheduling without application-layer visibility into encrypted traffic. The concurrent workload tests (Section 6.3.4) further confirm balanced allocation across mixed QUIC and MP-TCP flows. The mixed result concerns YouTube streaming: DARA achieves the highest resolution across all traces but at the cost of increased rebuffering and a higher quality switch rate, indicating that the reward function does not adequately penalise playback instability for ABR streaming. Whether this is fixable through reward redesign or represents an inherent tension between burst exploitation and streaming stability is not resolved by the experimental evidence.

7.2.5 Experimental Validation Infrastructure

The testbed objective has been thoroughly achieved. Mininet validation against hardware implementations, publicly available Mahi-Mahi traces, the five-trace volatility gradient, and the parallel MP-QUIC testbed for Peekaboo and BLEST evaluation collectively provide a rigorous and reproducible evaluation platform. The principal gap is the absence of

MP-TCP evaluation, which remains the most widely deployed multipath transport protocol and operates under fundamentally different reliability and ordering semantics from MP-DCCP.

7.2.6 Overall Assessment

The research objectives have been achieved to a substantial degree. The core contribution, demonstrating that Transformer-based congestion prediction combined with DQN-based rate allocation outperforms both reactive heuristics and state-of-the-art learning-based schedulers across diverse conditions, is supported by rigorous ablation evidence, statistical validation, and consistent results across multiple evaluation dimensions. The objectives least fully achieved concern the formal guarantees surrounding the multi-objective formulation, where the absence of Pareto front analysis leaves the optimality claim unproven, and the streaming workload results, where rebuffering penalties indicate that the framework's advantages are less conclusive for ABR-dependent applications than for bulk transfer and encrypted traffic. These do not undermine the core contributions but define the boundaries of the work: DARA has been demonstrated to be highly effective within its target regime of volatile bulk transfer and encrypted traffic aggregation, with streaming performance representing a partial achievement warranting further investigation.

Chapter 8

Author's Publications

The following publications resulted from the research presented in this thesis.

Published

1. G. Maglione, V. Rakocevic and M. Amend, "Intelligent Scheduling with Volatile Wireless Path State Prediction for 5G+ Multi-Access Networks," *2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring)*, Oslo, Norway, 2025, pp. 1–6, doi: 10.1109/VTC2025-Spring65109.2025.11174823.

Submitted

1. G. Maglione, V. Rakocevic, M. Amend and T. Soleymani, "Deep Adaptive Rate Allocation in Volatile Heterogeneous Wireless Networks," submitted to *IEEE Transactions on Cognitive Communications and Networking*, under review.

Bibliography

[1] "System architecture for the 5g system (5gs)," Technical Specification (TS) 23.501, 3GPP, 2017.

[2] H. Wu, S. Ferlin, G. Caso, O. Alay, and A. Brunstrom, "A survey on multipath transport protocols towards 5g access traffic steering, switching and splitting," *IEEE Access*, vol. 9, pp. 164417–164439, 2021.

[3] E. Gures, I. Shayea, A. Alhammadi, M. Ergen, and H. Mohamad, "A comprehensive survey on mobility management in 5g heterogeneous networks: Architectures, challenges and solutions," *IEEE Access*, vol. 8, pp. 195883–195913, 2020.

[4] S. Ferlin, O. Alay, O. Mehani, and R. Boreli, "Blest: Blocking estimation-based mptcp scheduler for heterogeneous networks," in *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 431–439, 2016.

[5] M. Morawski and P. Ignaciuk, "Influence of congestion control algorithms on head-of-line blocking in mptcp-based communication," in *2019 27th Telecommunications Forum (TELFOR)*, pp. 1–4, 2019.

[6] Y. Usman, H. Oladipupo, A. D. During, R. Akl, and R. Chataut, "Ai, ml, and llm integration in 5g/6g networks: A comprehensive survey of architectures, challenges, and future directions," *IEEE Access*, vol. 13, pp. 168914–168950, 2025.

[7] R. Shafin, L. Liu, V. Chandrasekhar, H. Chen, J. Reed, and J. C. Zhang, "Artificial intelligence-enabled cellular networks: A critical path to beyond-5g and 6g," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 212–217, 2020.

[8] D. Nguyen *et al.*, "Machine learning for network congestion control," *IEEE Access*, 2019.

[9] M. Sargent, J. Chu, D. V. Paxson, and M. Allman, "Computing TCP's Retransmission Timer." RFC 6298, June 2011.

[10] H. Xie *et al.*, "Wireless congestion control: Challenges and opportunities," *IEEE Communications Surveys & Tutorials*, 2019.

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

[12] M. Pieska, A. Kassler, A. Brunstrom, V. Rakocevic, and M. Amend, "Performance impact of nested congestion control on transport-layer multipath tunneling," *Future Internet*, vol. 16, no. 7, 2024.

[13] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar, "A deep reinforcement learning perspective on internet congestion control," in *ICML*, pp. 3050–3059, 2019.

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

[15] G. Maglione, V. Rakocevic, and M. Amend, "Intelligent Scheduling with Volatile Wireless Path State Prediction for 5G+ Multi-Access Networks," in *2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring)*, pp. 1–6, 2025.

[16] F. Yang, Q. Wang, and P. D. Amer, "Out-of-order transmission for in-order arrival scheduling for multipath tcp," in *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pp. 749–752, 2014.

[17] W. Li, H. Zhang, S. Gao, C. Xue, X. Wang, and S. Lu, "Smartcc: A reinforcement learning approach for multipath tcp congestion control in heterogeneous networks,"

IEEE Journal on Selected Areas in Communications, vol. 37, no. 11, pp. 2621–2633, 2019.

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

[19] X. You, C.-X. Wang, J. Huang, X. Gao, Z. Zhang, M. Wang, Y. Huang, C. Zhang, Y. Jiang, J. Wang, M. Zhu, B. Sheng, D. Wang, Z. Pan, P. Zhu, Y. Yang, Z. Liu, P. Zhang, X. Tao, S. Li, Z. Chen, X. Ma, C.-L. I, S. Han, K. Li, C. Pan, Z. Zheng, L. Hanzo, X. S. Shen, Y. J. Guo, Z. Ding, H. Haas, W. Tong, P. Zhu, G. Yang, J. Wang, E. G. Larsson, H. Q. Ngo, W. Hong, H. Wang, D. Hou, J. Chen, Z. Chen, Z. Hao, G. Y. Li, R. Tafazolli, Y. Gao, H. V. Poor, G. P. Fettweis, and Y.-C. Liang, "Towards 6g wireless communication networks: vision, enabling technologies, and new paradigm shifts," *Science China Information Sciences*, vol. 64, nov 2020.

[20] Ericsson, "Mobile data traffic outlook," 2022. Accessed: Jan. 16, 2023. [Online].

[21] E. Gures, I. Shayea, A. Alhammadi, M. Ergen, and H. Mohamad, "A comprehensive survey on mobility management in 5g heterogeneous networks: Architectures, challenges and solutions," *IEEE Access*, vol. 8, pp. 195883–195913, 2020.

[22] M. Tayyab, X. Gelabert, and R. Jäntti, "A simulation study on handover in lte ultra-small cell deployment: A 5g challenge," in *2019 IEEE 2nd 5G World Forum (5GWF)*, pp. 388–392, 2019.

[23] E. Gures, I. Shayea, A. Alhammadi, M. Ergen, and H. Mohamad, "A comprehensive survey on mobility management in 5g heterogeneous networks: Architectures, challenges and solutions," *IEEE Access*, vol. 8, pp. 195883–195913, 2020.

[24] P. Du, Q. Zhao, and M. Gerla, "A software defined multi-path traffic offloading system for heterogeneous lte-wifi networks," in *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, pp. 1–9, 2019.

[25] M. Pieska, A. Rabitsch, A. Brunstrom, A. Kassler, and M. Amend, "Adaptive cheapest path first scheduling in a transport-layer multi-path tunnel context," in *Proceedings of the 2021 Applied Networking Research Workshop*, ANRW '21, (New York, NY, USA), p. 39–45, Association for Computing Machinery, 2021.

[26] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses." RFC 8684, Mar. 2020.

[27] Y. Liu, Y. Ma, Q. D. Coninck, O. Bonaventure, C. Huitema, and M. Kühlewind, "Managing multiple paths for a QUIC connection," Internet-Draft draft-ietf-quic-multipath-17, Internet Engineering Task Force, Oct. 2025. Work in Progress.

[28] M. Amend, A. Brunstrom, A. Kassler, V. Rakocevic, and S. Johnson, "DCCP Extensions for Multipath Operation with Multiple Addresses," Internet-Draft draft-ietf-tsvwg-multipath-dccp-24, Internet Engineering Task Force, Apr. 2025. Work in Progress.

[29] C. Raiciu, M. J. Handley, and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols." RFC 6356, Oct. 2011.

[30] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "MPTCP is not pareto-optimal: Performance issues and a possible solution," *IEEE/ACM ToN*, vol. 21, no. 5, pp. 1651–1665, 2013.

[31] M. Maliha, G. Habibi, and M. Atiquzzaman, "A survey on congestion control and scheduling for multipath tcp: Machine learning vs classical approaches," 2023.

[32] N. Subash, B. Nithya, R. Bangar, and V. Patel, "mpquad: Multipath quad tcp congestion control in fanets," in *2023 3rd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS)*, pp. 12–18, 2023.

[33] W. Wei, K. Xue, J. Han, D. S. L. Wei, and P. Hong, "Shared bottleneck-based congestion control and packet scheduling for multipath tcp," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 653–666, 2020.

[34] M. Sun, W. Yang, and F. Wu, "A multi-path congestion control mechanism for named data networking," in *2019 2nd International Conference on Hot Information-Centric Networking (HotICN)*, pp. 1–6, 2019.

[35] Q. De Coninck and O. Bonaventure, "Multipath quic: Design and evaluation," in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '17, (New York, NY, USA), p. 160–166, Association for Computing Machinery, 2017.

[36] H. Shi *et al.*, "Understanding MPQUIC: A multipath extension to QUIC," *IEEE Network*, 2021.

[37] M. Amend, E. Bogenfeld, M. Cvjetkovic, V. Rakocevic, M. Pieska, A. Kassler, and A. Brunstrom, "A framework for multiaccess support for unreliable internet traffic using multipath dccp," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, pp. 316–323, IEEE, 2019.

[38] N. Cardwell, Y. Cheng, S. H. Yeganeh, I. Swett, and V. Jacobson, "BBR Congestion Control," Internet-Draft draft-cardwell-iccrg-bbr-congestion-control-02, Internet Engineering Task Force, Mar. 2022. Work in Progress.

[39] A. Gurtov, T. Henderson, S. Floyd, and Y. Nishida, "The NewReno Modification to TCP's Fast Recovery Algorithm." RFC 6582, Apr. 2012.

[40] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," in *ACM SIGOPS Operating Systems Review*, vol. 42, pp. 64–74, 2008.

[41] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[42] N. Cardwell *et al.*, "BBRv2: A model-based congestion control," tech. rep., IETF, 2022.

[43] M. J. Handley, J. Padhye, S. Floyd, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification." RFC 5348, Sept. 2008.

[44] Y.-J. Song, G.-H. Kim, I. Mahmud, W.-K. Seo, and Y.-Z. Cho, "Understanding of bbrv2: Evaluation and comparison with bbrv1 congestion control algorithm," *IEEE Access*, vol. 9, pp. 37131–37145, 2021.

[45] X. Nie, Y. Zhao, Z. Li, G. Chen, K. Sui, J. Zhang, Z. Ye, and D. Pei, "Dynamic tcp initial windows and congestion control schemes through reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1231–1247, 2019.

[46] Q. Guo, B. Zhao, M. Song, and G. Zhong, "A survey of deep learning for time series forecasting: Taxonomy, analysis and future directions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 38, no. 5, pp. 2541–2560, 2026.

[47] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," 2023.

[48] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[49] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[50] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[51] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[52] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022.

[53] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*, pp. 807–814, 2010.

[54] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[55] E. L. Lehmann and G. Casella, *Theory of point estimation*. Springer Science & Business Media, 2nd ed., 1998.

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[57] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," in *International Conference on Machine Learning*, pp. 4596–4604, PMLR, 2018.

[58] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[59] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[60] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, p. 20200209, 02 2021.

[61] G. E. Box and D. R. Cox, "An analysis of transformations," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 26, no. 2, pp. 211–243, 1964.

[62] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: Forecasting and control*. John Wiley & Sons, 5th ed., 2015.

[63] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[64] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," *arXiv preprint arXiv:1905.09418*, 2019.

[65] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[66] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 2nd ed., 2018.

[67] R. Bellman, "A Markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, 1957.

[68] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[69] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[70] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[71] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," in *Machine Learning*, vol. 8, pp. 293–321, 1992.

[72] W. Chen, X. Qiu, T. Cai, H.-N. Dai, Z. Zheng, and Y. Zhang, "Deep reinforcement learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1659–1692, 2021.

[73] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," *Proceedings of the 16th International Conference on Machine Learning*, vol. 99, pp. 278–287, 1999.

[74] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456, 2015.

[75] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, 2019.

[76] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker, "Model-based reinforcement learning: A survey," *Found. Trends Mach. Learn.*, vol. 16, p. 1–118, Jan. 2023.

[77] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[78] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *International Conference on Learning Representations*, 2017.

[79] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 4780–4789, 2019.

[80] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations*, 2019.

[81] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019.

[82] P. B. Diakhate, T.-T. Chu, M. A. Labiod, B. Augustin, H.-A. Tran, and A. Mellouk, "Experimental evaluation of multiple multipath schedulers over various ur-

ban mobile environments," in *Proceedings of the 11th International Symposium on Information and Communication Technology*, SoICT '22, (New York, NY, USA), p. 201–207, Association for Computing Machinery, 2022.

[83] Y.-s. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "Ecf: An mptcp path scheduler to manage heterogeneous paths," in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '17, (New York, NY, USA), p. 147–159, Association for Computing Machinery, 2017.

[84] T. Ha, A. Masood, W. Na, and S. Cho, "Intelligent multi-path tcp congestion control for video streaming in internet of deep space things communication," *ICT Express*, vol. 9, no. 5, pp. 860–868, 2023.

[85] J. Luo, X. Su, and B. Liu, "A reinforcement learning approach for multipath tcp data scheduling," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0276–0280, 2019.

[86] M. M. Roselló, "Multi-path scheduling with deep reinforcement learning," in *2019 European Conference on Networks and Communications (EuCNC)*, pp. 400–405, 2019.

[87] S. Lee and J. Yoo, "Reinforcement learning based multipath quic scheduler for multimedia streaming," *Sensors*, vol. 22, no. 17, 2022.

[88] R. Zhuang, J. Han, K. Xue, J. Li, D. S. L. Wei, R. Li, Q. Sun, and J. Lu, "Achieving flexible and lightweight multipath congestion control through online learning," *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 46–59, 2023.

[89] S. R. Pokhrel and A. Walid, "Learning to harness bandwidth with multipath congestion control and scheduling," 2021.

[90] Y. Xing, K. Xue, Y. Zhang, J. Han, J. Li, and D. S. L. WeiMember, "An online learning assisted packet scheduler for mptcp in mobile networks," *IEEE/ACM Transactions on Networking*, vol. 31, no. 5, pp. 2297–2312, 2023.

[91] J. Han, K. Xue, J. Li, R. Zhuang, R. Li, R. Yu, G. Xue, and Q. Sun, "Edar: An experience-driven multipath scheduler for seamless handoff in mobile networks," *IEEE Transactions on Wireless Communications*, vol. 22, no. 10, pp. 6839–6852, 2023.

[92] T. Chen, J. Ai, X. Xiong, and F. Tan, "Concurrent multipath transmission for ultra-reliable and low latency with deep reinforcement learning," in *Proceedings of the 2023 4th International Conference on Computing, Networks and Internet of Things*, CNIOT '23, (New York, NY, USA), p. 678–683, Association for Computing Machinery, 2023.

[93] G. Lv, Q. Wu, Y. Liu, Z. Li, Q. Tan, F. Yang, W. Chen, Y. Ma, H. Guo, Y. Chen, and G. Xie, "Chorus: Coordinating mobile multipath scheduling and adaptive video streaming," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '24, (New York, NY, USA), p. 246–262, Association for Computing Machinery, 2024.

[94] W. Yang, L. Cai, S. Shu, J. Pan, and A. Sepahi, "Mams: Mobility-aware multipath scheduler for mpquic," *IEEE/ACM Transactions on Networking*, pp. 1–16, 2024.

[95] S. Park and S. R. Das, "Cross-layer scheduling in quic and multipath quic for 360-degree video streaming," in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2024.

[96] T. T. Nguyen, M. H. Vu, P. L. Nguyen, P. T. Do, and K. Nguyen, "A q-learning-based multipath scheduler for data transmission optimization in heterogeneous wireless networks," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, pp. 573–578, 2023.

[97] H. Zeng, L. Cui, F. P. Tso, and Z. Zhang, "Optimizing multipath quic transmission over heterogeneous paths," *Computer Networks*, vol. 215, p. 109198, 2022.

[98] H. Wu, O. Alay, A. Brunström, G. Caso, and S. Ferlin, "FALCON: fast and accurate multipath scheduling using offline and online learning," *CoRR*, vol. abs/2201.08969, 2022.

[99] X. Han, B. Han, J. Li, and C. Song, "Multi-agent drl-based multipath scheduling for video streaming with quic," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 20, May 2024.

[100] P. Dong, R. Shen, Q. Wang, Y. Zuo, Y. Li, D. Zhang, L. Zhang, and W. Yang, "Multipath tcp meets reinforcement learning: A novel energy-efficient scheduling approach in heterogeneous wireless networks," *IEEE Wireless Communications*, vol. 30, no. 2, pp. 138–146, 2023.

[101] B. Liao, G. Zhang, Z. Diao, and G. Xie, "Precise and adaptable: Leveraging deep reinforcement learning for gap-based multipath scheduler," in *2020 IFIP Networking Conference (Networking)*, pp. 154–162, 2020.

[102] Y. Xing, K. Xue, Y. Zhang, J. Han, J. Li, D. S. L. Wei, R. Li, Q. Sun, and J. Lu, "A stream-aware mpquic scheduler for http traffic in mobile networks," *IEEE Transactions on Wireless Communications*, vol. 22, no. 4, pp. 2775–2788, 2023.

[103] D. T. AG, "Multipath extension for dccp," 2023.

[104] H. ElBouanani, C. Barakat, W. Dabbous, and T. Turletti, "Fidelity-aware large-scale distributed network emulation," *Computer Networks*, vol. 250, p. 110531, 2024.