Sorbonne University
Doctoral School of Informatics, Telecommunications and Electronics of Paris
EURECOM

Deterministic network design and optimization to support low-latency communications in 6G

Presented by **Sofiane MESSAOUDI**

Dissertation for Doctor of Philosophy in Information and Communication Engineering

Directed by **Prof. Adlen Ksentini**
Co-Directed by **Prof. Christian Bonnet**

The Jury committee is composed of:

Prof. Pascal Lorenz	(University of Haute Alsace)	Reviewer
HDR. Moez Esghir	(University of Technology of Troyes)	Reviewer
Prof. Mai Trang Nguyen	(University of Paris Nord - Sorbonne)	Examiner
Prof. César Viho	(IRISA - ISTIC/University Rennes 1)	Examiner
Prof. Jérome Harri	(Eurecom)	Jury President
Prof. Adlen Ksentini	(Eurecom)	Thesis Director
Prof. Christian Bonnet	(Eurecom)	Thesis Co-Director

Sophia Antipolis, march 31st, 2025

Sorbonne Université
École Doctorale Informatique, Télécommunications et Électronique de Paris
EURECOM

Conception et optimisation de réseaux déterministes pour garantir les communications à faible latence dans les réseaux 6G

Présenté par **Sofiane MESSAOUDI**

Thèse de doctorat en Sciences de l'Information et de la Communication

Dirigée par **Prof. Adlen Ksentini**
Co-Dirigée par **Prof. Christian Bonnet**

Le jury est composé de :

Prof. Pascal Lorenz	(Université de Haute Alsace)	Rapporteur
MCF. Moez Esghir	(Université de Technologie de Troyes)	Rapporteur
Prof. Mai Trang Nguyen	(Université de Paris Nord - Sorbonne)	Examinateur
Prof. César Viho	(IRISA - ISTIC/Université Rennes 1)	Examinateur
Prof. Jérome Harri	(Eurecom)	Président du Jury
Prof. Adlen Ksentini	(Eurecom)	Directeur de Thèse
Prof. Christian Bonnet	(Eurecom)	Co-Directeur de Thèse

Sophia Antipolis, 31 mars , 2025

Abstract

The rapid advancement of time-sensitive applications, such as holographic communication, the tactile Internet, autonomous systems, and the metaverse, has placed unprecedented demands on network infrastructures. These applications require ultra-low latency, high reliability, and predictable performance to ensure seamless user experiences. Meeting these demands necessitates deterministic network behaviors, where latency, jitter, and reliability are predictable and bounded. However, traditional network architectures struggle to meet these requirements, particularly in dynamic, large-scale environments characterized by fluctuating traffic patterns and resource availability.

The transition from 5^{th} Generation (5G) to 6^{th} Generation (6G) introduces a paradigm shift, emphasizing deterministic networking to guarantee bounded latency, jitter, and reliability. A key enabler of this shift is Software-Defined Networking (SDN), which provides programmability, centralized control, and real-time adaptability. Despite its potential, SDN's capabilities for optimizing the Data Plane (DP) to support time-sensitive applications remain underexplored. This thesis addresses this gap by proposing SDN-based frameworks and solutions to ensure deterministic End-To-End (E2E) latency and Quality of Service (QoS) in next-generation networks.

The primary contributions of this thesis include the development of innovative mechanisms for dynamic queuing, intelligent path selection, predictive traffic management using Graph Neural Network (GNN), advanced congestion control leveraging Low-Latency, Low-Loss, Scalable Throughput (L4S) protocols, and kernel-level packet processing with Extended Berkeley Packet Filtering (eBPF). These contributions collectively enhance the SDN-enabled DP to meet the stringent requirements of Ultra-Reliable Low Latency Communications (URLLC) in 6G networks. These contributions are described as follows:

1. SDN-based Path Selection for Time-Sensitive Applications Using Dynamic Queuing: To address limitations in Transport Network (TN) scalability and adaptability, this contribution introduces the Software-Defined Low Latency (SDLL) framework. SDLL leverages SDN principles to achieve deterministic E2E latency through: (i) dynamic Queue Management (QM) mechanism that prioritize delay-sensitive flows while preventing interference from lower-priority traffic; (ii) Traffic Engineering (TE) algorithms for adaptive path selection and Resource Allocation (RA), ensuring optimal utilization of network resources while avoiding congestion; and (iii) integration of 5G QoS Flow Identifier (QFI) for precise QoS provisioning across TN traffic classes. These mechanisms collectively address the static nature of traditional solutions by introducing flexibility and scalability.

2. SDN-based Admission Control (AC) for Time-Sensitive Applications Leveraging GNNs: This thesis proposes a novel Graph Neural Network-Admission Control (GNN-AC) mechanism , which uses the RouteNet-Fermi (RouteNet-F) model to predict latency across dynamic network topologies and traffic conditions. GNN-AC incorporates a two-layer architecture: (i) the Network Delay Predictor (NetDelP) layer predicts network latency with high accuracy in real-time, enabling proactive traffic regulation; and (ii) the Admission Control Agent (AdConAgt) layer integrates latency predictions with SDN controller to dynamically regulate flow admission and ensure QoS guarantees while avoiding congestion. The proposed solution addresses scalability challenges and traffic variability in large-scale networks.

3. SDN-based Congestion Control for Time-Sensitive Applications Adopting L4S Transport Layer: Traditional congestion control protocols such as Transmission Control Protocol (TCP) struggle to achieve low-latency and high-throughput in Beyond 5G (B5G) environments. This thesis introduces an SDN-based

congestion control framework that integrates L4S mechanisms (i.e., Software-Defined Networking-Low-Latency, Low-Loss, Scalable Throughput (SDN-L4S)). The proposed framework leverages : (i) Explicit Congestion Notification (ECN) for proactive congestion marking; (ii) a dual-queue structure for isolating L4S traffic; and (iii) adaptive transmission rate to ensure that critical, time-sensitive flows maintain ultra-low latency and high throughput. These innovations enable robust congestion avoidance while dynamically adapting to fluctuating traffic loads.

4. SDN for Cloud Edge Continuum (CEC) Computing Nodes Interconnection Utilizing eBPF: To optimize the interconnection of cloud, edge, and far-edge nodes, this thesis introduces an eBPF-based SDN framework, High-Efficiency Layered Infrastructure with eBPF for Optimized SD-WAN (HELIOS). HELIOS includes: (i) eBPF-based packet processing at the edges; and (ii) an SDN-based framework that updates the user-space flow rules in real time, which are used by the eBPF code (i.e., eBPF maps) via Google Remote Procedure Calls (gRPC). By implementing custom packet processing directly in the kernel, eBPF significantly reduces processing overhead compared to user-space solutions. The Software-defined Wide Area Network (SD-WAN) edge nodes are centrally managed by an SDN controller, which enforces QoS policies and enables dynamic traffic steering. This approach ensures efficient resource utilization and low-latency interconnection of cloud, edge, and far-edge nodes within the CEC.

The proposed solutions were rigorously evaluated using both virtual and physical prototypes, primarily focusing on the context of 5G DP, specifically the TN, particularly the backhaul and midhaul. Results demonstrate significant reductions in latency, improvements in resource efficiency, and scalability under dynamic traffic loads. The contributions of this thesis are validated through multiple publications in leading Institute of Electrical and Electronics Engineers (IEEE) conferences. This work lays the foundation for SDN-enabled deterministic, low-latency communication in 6G networks, addressing the critical challenges of dynamic traffic management, AC, congestion control, and edge processing. By bridging the gap between application demands and network capabilities, it lays the foundation for supporting the diverse and stringent requirements of next-generation services.

Résumé

L'avancement rapide des applications sensibles au temps, telles que la communication holographique, l'Internet tactile, les systèmes autonomes et le métavers, a créé des exigences sans précédent pour les infrastructures réseau. Ces applications nécessitent une latence ultra-faible, une grande fiabilité et des performances prévisibles pour garantir une expérience utilisateur sans faille. Répondre à ces exigences nécessite un comportement déterministe du réseau, où la latence, le jitter et la fiabilité sont prévisibles et bornés. Cependant, les architectures réseau traditionnelles peinent à répondre à ces exigences, en particulier dans des environnements dynamiques à grande échelle caractérisés par des modèles de trafic fluctuants et une disponibilité variable des ressources.

La transition de la 5^{th} Generation (5G) vers la 6^{th} Generation (6G) introduit un changement de paradigme, mettant l'accent sur la mise en réseau déterministe pour garantir des latences, des jitters et une fiabilité bornés. Un facteur clé de ce changement est le Software-Defined Networking (SDN), qui offre une programmabilité, un contrôle centralisé et une adaptabilité en temps réel. Malgré son potentiel, les capacités du SDN pour optimiser le Data Plane (DP) afin de soutenir les applications sensibles au temps restent sous-explorées. Cette thèse aborde cette lacune en proposant des cadres et des solutions basés sur le SDN pour garantir une latence End-To-End (E2E) déterministe et une Quality of Service (QoS) dans les réseaux de prochaine génération.

Les principales contributions de cette thèse incluent le développement de mécanismes innovants pour la gestion dynamique des files d'attente, la sélection intelligente de chemins, la gestion prédictive du trafic utilisant des Graph Neural Networks (GNNs), le contrôle avancé de la congestion en s'appuyant sur les protocoles Low-Latency, Low-Loss, Scalable Throughput (L4S), et le traitement des paquets au niveau du noyau avec Extended Berkeley Packet Filtering (eBPF). Ces contributions améliorent collectivement le DP activé par SDN pour répondre aux exigences strictes des Ultra-Reliable Low Latency Communications (URLLC) dans les réseaux 6G. Ces contributions sont décrites comme suit :

1. Sélection de chemin basée sur SDN pour les applications sensibles au temps en utilisant la gestion dynamique des files d'attente : Pour répondre aux limitations de scalabilité et d'adaptabilité du Transport Network (TN), cette contribution introduit le cadre Software-Defined Low Latency (SDLL). SDLL exploite les principes du SDN pour atteindre une latence déterministe E2E grâce à : (i) des mécanismes de Queue Management (QM) dynamiques qui priorisent les flux sensibles à la latence tout en empêchant les interférences provenant de trafic de moindre priorité ; (ii) des algorithmes de Traffic Engineering (TE) pour une sélection adaptative des chemins et de Resource Allocation (RA), assurant une utilisation optimale des ressources du réseau tout en évitant la congestion ; et (iii) l'intégration des QoS Flow Identifier (QFI) de la 5G pour un approvisionnement précis en QoS à travers les classes de trafic TN. Ces mécanismes répondent collectivement à la nature statique des solutions traditionnelles en introduisant flexibilité et scalabilité.

2. Admission Control (AC) basée sur SDN pour les applications sensibles au temps en exploitant les GNNs : Cette thèse propose un nouveau mécanisme Graph Neural Network-Admission Control (GNN-AC), qui utilise le modèle RouteNet-Fermi (RouteNet-F) pour prédire la latence à travers des topologies de réseaux dynamiques et des conditions de trafic variables. Le GNN-AC intègre une architecture à deux couches : (i) la couche Network Delay Predictor (NetDelP) qui prédit la latence du réseau avec une grande précision en temps réel, permettant une régulation proactive du trafic ; et (ii) la couche Admission Control Agent (AdConAgt) qui intègre les prédictions de latence avec le contrôleur SDN pour réguler

dynamiquement l'admission des flux et garantir la QoS tout en évitant la congestion. La solution proposée aborde les défis de scalabilité et de variabilité du trafic dans les réseaux à grande échelle.

3. Contrôle de la congestion basé sur SDN pour les applications sensibles au temps en adoptant la couche de transport L4S : Les protocoles traditionnels de contrôle de la congestion tels que Transmission Control Protocol (TCP) peinent à atteindre une faible latence et un haut débit dans les environnements Beyond 5G (B5G). Cette thèse introduit un cadre de contrôle de la congestion basé sur SDN qui intègre les mécanismes L4S (c'est-à-dire Software-Defined Networking-Low-Latency, Low-Loss, Scalable Throughput (SDN-L4S)). Le cadre proposé repose sur : (i) Explicit Congestion Notification (ECN) pour le marquage proactif de la congestion ; (ii) une structure de file d'attente double pour isoler le trafic L4S ; et (iii) un taux de transmission adaptatif pour garantir que les flux critiques et sensibles au temps maintiennent une latence ultra-faible et un débit élevé. Ces innovations permettent une évitabilité robuste de la congestion tout en s'adaptant dynamiquement aux variations des charges de trafic.

4. SDN pour l'interconnexion des nœuds de calcul Cloud Edge Continuum (CEC) en utilisant eBPF : Afin d'optimiser l'interconnexion des nœuds cloud, edge et far-edge, cette thèse introduit un cadre SDN basé sur eBPF, High-Efficiency Layered Infrastructure with eBPF for Optimized SD-WAN (HELIOS). HELIOS comprend : (i) un traitement des paquets basé sur eBPF aux bords ; et (ii) un cadre SDN qui met à jour en temps réel les règles de flux en espace utilisateur, utilisées par le code eBPF (c'est-à-dire, les tables de correspondance eBPF) via Google Remote Procedure Calls (gRPC). En implémentant un traitement personnalisé des paquets directement dans le noyau, eBPF réduit considérablement la surcharge de traitement par rapport aux solutions en espace utilisateur. Les nœuds de bord Software-defined Wide Area Network (SD-WAN) sont gérés de manière centralisée par un contrôleur SDN, qui applique des politiques de QoS et permet un acheminement dynamique du trafic. Cette approche garantit une utilisation efficace des ressources et une interconnexion à faible latence des nœuds cloud, edge et far-edge au sein du CEC.

Les solutions proposées ont été évaluées de manière rigoureuse à l'aide de prototypes virtuels et physiques, principalement dans le contexte du DP 5G, spécifiquement le TN, notamment le backhaul et le midhaul. Les résultats démontrent des réductions significatives de la latence, des améliorations de l'efficacité des ressources et de la scalabilité sous des charges de trafic dynamiques. Les contributions de cette thèse sont validées par plusieurs publications dans des conférences de premier plan de l'Institute of Electrical and Electronics Engineers (IEEE). Ce travail jette les bases de la communication déterministe à faible latence activée par SDN dans les réseaux 6G, en répondant aux défis critiques de la gestion dynamique du trafic, de l'AC, du contrôle de la congestion et du traitement en périphérie. En comblant le fossé entre les exigences des applications et les capacités des réseaux, il pose les bases du soutien aux exigences diversifiées et strictes des services de prochaine génération.

Remerciements

Avant tout, je tiens à exprimer ma profonde gratitude à mes encadrants, le Professeur Adlen Ksentini et le Professeur Christian Bonnet, pour leurs précieux conseils, leur disponibilité sans faille, et leur dévouement tout au long de cette aventure doctorale. Leur vision claire et perspicace des problématiques de recherche m'a guidé dans mes premiers pas dans le domaine scientifique. Leur soutien indéfectible m'a offert des opportunités inestimables, dont je leur suis profondément reconnaissant. Qu'ils trouvent ici l'expression de ma reconnaissance sincère, de mon profond respect, et de mes remerciements les plus chaleureux.

Je tiens également à remercier le Docteur Franck Messaoudi, mon frère, pour son accompagnement constant, ses conseils avisés, et son soutien indéfectible tout au long de cette expérience. Son aide précieuse a été une véritable source de motivation et de réconfort.

Je souhaite exprimer ma gratitude envers toute l'équipe pédagogique d'EURECOM ainsi que les intervenants professionnels responsables de ma formation, notamment dans le cadre du Plan Individuel de Formation (PIF), pour la qualité de l'enseignement et des échanges qui ont enrichi cette étape de mon parcours académique.

Ma reconnaissance s'étend également aux fonctionnaires d'EURECOM pour leur aide précieuse et leur disponibilité constante, notamment Tom, Sophie et Marie. Je remercie également mes collègues doctorants et chercheurs du laboratoire de recherche d'EURECOM : Karim, Mohamed, Ayoub, Abdelkader, Abdelrahmane, Giulio, et bien d'autres, pour leur soutien moral, leurs conseils précieux, et les moments partagés qui ont égayé mon quotidien.

Je remercie sincèrement les membres du jury d'avoir accepté de juger ce travail et de m'honorer de leur expertise et de leurs critiques constructives, qui contribueront sans nul doute à enrichir mes perspectives futures.

Je tiens à exprimer une reconnaissance particulière à mes parents, mes frères et sœurs, ainsi qu'à toute ma famille, pour leur soutien inconditionnel et leurs encouragements constants. Leur amour et leur confiance ont été ma source d'énergie tout au long de mon parcours académique, de l'école jusqu'à l'université, et jusqu'à la soutenance de cette thèse.

Enfin, je souhaite remercier tous ceux qui, de près ou de loin, ont contribué au succès de cette thèse doctorale. Leur soutien, leur encouragement et leur aide précieuse, notamment durant la rédaction de ce mémoire, ont été d'une importance capitale pour mener à bien ce projet.

À toutes et à tous, un grand merci.
Sofiane

Contents

List of Figures

List of Tables

Acronyms

(e)BPF (extended) Berkeley Packet Filtering.

3GPP 3rd Generation Partnership Project.

4G 4^{th} Generation.

5G 5^{th} Generation.

5G NR 5G New Radio.

5G&B 5G and Beyond.

5QI 5G QoS Identifier.

6G 6^{th} Generation.

AC Admission Control.

ACK ACKnowledgment.

AdConAgt Admission Control Agent.

AI Artificial Intelligence.

AIMD Additive Increase/Multiplicative Decrease.

AMF Access and Mobility Management Function.

AN Access Network.

API Application Programming Interface.

AQM Active Queue Management.

AR Augmented Reality.

AW Average Window.

B5G Beyond 5G.

BGP-LS Border Gateway Protocol-Link State.

CDF Cumulative Distribution Function.

CE Congestion Experienced.

CEC Cloud Edge Continuum.

CN Core Network.

CoDel Controlled Delay.

CoT Class of Traffic.

CP Control Plane.

CPU Central Processing Unit.

CRUD Create, Read, Update, and Delete.

CU Centralized Unit.

CWND Congestion Window.

D-Learn Deep Learning.

D2D device-to-device.

DB Data Base.

DC Delay-Critical.

DL Downlink.

DMA Direct Memory Access.

DNN Deep Neural Network.

DP Data Plane.

DPDK Data Plane Development Kit.

DQN Deep Q-Learning.

DRB Data Radio Bearer.

DSCP Differentiated Services Code Point.

DU Distributed Unit.

DUT Device Under Test.

E-Mail Electronic Mail.

E2E End-To-End.

eBPF Extended Berkeley Packet Filtering.

ECN Explicit Congestion Notification.

ECT ECN-Capable Transport.

ELF Executable and Linkable Format.

eMBB enhanced Mobile Broadband.

eNB evolved Node B.

ETSI European Telecommunications Standards Institute.

FAR Forwarding Action Rule.

FTP File Transfer Protocol.

GBR Guaranteed Bit Rate.

GCC Google Congestion Control.

gNB gNodeB.

GNN Graph Neural Network.

GNN-AC Graph Neural Network-Admission Control.

GRE Generic Routing Encapsulation.

gRPC Google Remote Procedure Calls.
GTP GPRS Tunneling Protocol.

HardIRQ Hardware IRQ.
HARQ Hybrid Automatic Repeat Request.
HELIOS High-Efficiency Layered Infrastructure with eBPF for Optimized SD-WAN.
HP High Priority.
HTB Hierarchical Token Bucket.

ICMP Internet Control Message Protocol.
ID Identifier.
IE Information Element.
IEEE Institute of Electrical and Electronics Engineers.
IETF Internet Engineering Task Force.
IMS IP Multimedia Subsystem.
IoT Internet of Things.
IP Internet Protocol.
IPsec Internet Protocol security.
IPv4 Internet Protocol version 4.
IPv6 Internet Protocol version 6.
IRQ Interrupt Request.
ISP Internet Service Provider.

JIT Just In Time.

KPI Key Performance Indicator.

L4S Low-Latency, Low-Loss, Scalable Throughput.
LAN Local Area Network.
LB Load Balancing.
LBR Load Balancing Rule.
LINX Low-Latency Intelligent Network eXecution using eBPF.
LLVM Low Level Virtual Machine.
LP Low Priority.
LTE Long Term Evolution.
LTE-A Long Term Evolution-Advanced.
LVE Linux Virtual Emulator.

MAC Medium Access Control.
MCPTT Mission Critical Push To Talk.

MDBV Maximum Data Burst Volume.
MEC Multi-Access Edge Computing.
MEF Metro Ethernet Forum.
MIMO Multiple Input Multiple Output.
ML Machine Learning.
mMTC massive Machine Type Communication.
mmWave millimeter wave.
MP Medium Priority.
MP-TCP Multi-Path TCP.
MPLS Multiprotocol Label Switching.
MR Mixed Reality.

NAPI New API.
NBI NorthBound Interface.
NETCONF Network Configuration Protocol.
NetDelP Network Delay Predictor.
NF Network Function.
NFV Network Function Virtualization.
NIC Network Interface Controller.
NM Network Model.
NN Neural Network.
NP Network Prediction.

OA Optimisation Algorithm.
ONF Open Networking Foundation.
ONOS Open Network Operating System.
OSGi Open Services Gateway initiative.
OVS Open vSwitch.
OVSDB Open vSwitch Database Management.

P2P peer-to-peer.
P4 Programming Protocol-independent Packet Processors.
PDB Packet Delay Budget.
PDI Packet Detection Information.
PDR Packet Detection Rule.
PDU Packet Data Unit.
PER Packet Error Rate.
PhD Doctor of Philosophy.

PL Priority Level.
PLMN Public Land Mobile Network.
PLR Packet Loss Rate.
PPD Packet Predicted Delay.
PPO Proximal Policy Optimization.

QCI QoS Class Identifier.
qdisc queueing discipline.
QFI QoS Flow Identifier.
QM Queue Management.
QoS Quality of Service.
QT Queuing Theory.
QUIC Quick UDP Internet Connections.

RA Resource Allocation.
RAN Radio Access Network.
RAT Radio Access Technology.
RED Random Early Detection.
REST REpresentational State Transfer.
RF Radio Frequency.
RFC Requests For Comments.
RL Reinforcement Learning.
RNN Recurrent Neural Network.
RouteNet-F RouteNet-Fermi.
RPC Remote Procedure Call.
RT Resource Type.
RTP Real-time Transport Protocol.
RTT Round Trip Time.
RU Radio Unit.

SBA Service-Based Architecture.
SBI SouthBound Interface.
SCTP Stream Control Transmission Protocol.
SD-WAN Software-defined Wide Area Network.
SDLL Software-Defined Low Latency.
SDN Software-Defined Networking.
SDN-L4S Software-Defined Networking-Low-Latency, Low-Loss, Scalable Throughput.
SDNSP SDN Shortest Path.

SDQ Software-Defined Queuing.
SDQoS Software-Defined QoS.
SFC Service Function Chaining.
SKB Socket Kernel Buffer.
SLA Service-Level Agreement.
SMF Session Management Function.
SNMP Simple Network Management Protocol.
SoftIRQ Software IRQ.
SRv6 Segment Routing IPv6.
STGCN Spatial-Temporal Graph Convolutional Network.
sTTI short Transmission Time Interval.
SUT System Under Test.

tc traffic control.
TCC Traditional Congestion Control.
TCP Transmission Control Protocol.
TE Traffic Engineering.
TFRC TCP Friendly Rate Control.
TI Tactile Interaction.
TN Transport Network.
ToS Type of Service.
TS Technical Specification.
TSN Time Sensitive Networking.

UAV Unmanned Aerial Vehicle.
UDN Ultra-Dense Networks.
UDP User Datagram Protocol.
UE User Equipment.
UHD Ultra High Definition.
UL Uplink.
UPF User Plane Function.
URLLC Ultra-Reliable Low Latency Communications.

V2X Vehicle-to-Everything.
VAE Variational Autoencoder.
VLAN Virtual Local Area Network.
VM Virtual Machine.
VNF Virtual Network Function.

VoIP Voice over Internet Protocol.

VR Virtual Reality.

WAN Wide Area Network.

WebRTC Web Real-Time Communication.

WWW World Wide Web.

XDP eXpress Data Path.

XR Extended Reality.

YAML Yet Another Markup Language.

YANG Yet Another Next Generation.

Chapter 1

General Introduction

1.1 Context

The rise of time-sensitive applications, such as holographic communication, tactile Internet, autonomous systems, and the metaverse, has driven stringent requirements for ultra-low latency, ultra-reliable communication, and high throughput [1]. These applications demand not only high performance but also predictability in network behavior to meet their requirements and ensure a seamless user experience. Traditional network architectures, struggle to meet these demands, particularly in dynamic and large-scale environments where the variability of traffic and resource availability can introduce significant performance challenges.

The transition from 5^{th} Generation (5G) to 6^{th} Generation (6G) introduces a paradigm shift in how networks are designed and managed. With a focus on deterministic networking, where latency, jitter, and reliability are predictable and bounded, 6G aims to overcome the limitations of prior generations. This shift is essential to support the growing diversity of applications, ranging from mission-critical systems like autonomous vehicles to immersive technologies like virtual and Augmented Reality (AR). Deterministic networking ensures that the network can meet application-specific requirements without compromising scalability or efficiency.

A critical enabler for supporting these requirements is Software-Defined Networking (SDN), which offers network programmability, global visibility, and dynamic adaptability. SDN principles can address key challenges in the Data Plane (DP), particularly for supporting time-sensitive applications. By decoupling the Control Plane (CP) from the DP, SDN enables centralized management and real-time decision-making, allowing it to optimize network resources dynamically and ensure predictable performance.

This thesis explores how SDN can optimize the DP to guarantee End-To-End (E2E) latency and Quality of Service (QoS) for diverse application requirements. The proposed solutions leverage state-of-the-art technologies, including dynamic queuing, Machine Learning (ML) models such as Graph Neural Network (GNN) [2], and advanced congestion control protocols like the Internet Engineering Task Force (IETF) Low-Latency, Low-Loss, Scalable Throughput (L4S) [3], which are applied to the 5G backbone (i.e., the DP), and more precisely at the Transport Network (TN) level, including the backhaul. Additionally, Extended Berkeley Packet Filtering (eBPF) [4] capabilities are utilized within the interconnection of Cloud Edge Continuum (CEC) nodes. By integrating these technologies, this thesis aims to bridge the gap between the requirements of time-sensitive applications and the capabilities of current and emerging network infrastructures.

1.2 Motivation

The motivation for this thesis stems from the limitations of existing network frameworks in meeting the stringent requirements of time-sensitive applications [5]. While 5G introduced SDN as a critical component for network monitoring and programmability, its potential to optimize the DP for ultra-low latency has not been fully explored. Key challenges include the inability of static configurations to adapt to varying traffic patterns, leading to inefficiencies in resource utilization and degraded QoS for critical flows; the difficulty of scaling traditional deterministic networking approaches across large and dynamic networks; the lack of predictive traffic management mechanisms to anticipate and address emerging congestion patterns; and the challenge of ensuring QoS and high-performance in the CEC.

This thesis addresses these limitations by proposing SDN-based frameworks for dynamic queuing, GNN-driven prediction, advanced congestion control using L4S, and eBPF-powered edge processing in CEC environments. These contributions collectively enable deterministic and scalable support for time-sensitive applications in the DP.

1.3 Thesis Challenges and Contributions

This thesis contributes to the transition from 5G to 6G by focusing on optimizing current SDN-enabled DP features to enhance TN functionality. These enhancements aim to enable more efficient traffic management in Beyond 5G (B5G) networks, paving the way toward 6G. The primary challenge addressed in this thesis involves achieving deterministic low-latency communications to support time-sensitive applications, with a particular emphasis on TN. The TN, comprising the backhaul, midhaul, and fronthaul links that connect the Radio Access Network (RAN) and Core Network (CN), is a critical component in enabling ultra-low E2E latency.

Existing SDN approaches rely on static queue configurations, which lack real-time and Classes of Traffic (CoTs)-aware adaptation of Resource Allocation (RA). These limitations result in challenges related to scalability, flexibility, and resource efficiency, making them insufficient to meet the stringent requirements of next-generation services. To address these limitations, this thesis explores the potential of dynamic queuing and intelligent path selection mechanisms. By integrating scalable Traffic Engineering (TE) techniques with real-time adaptability, the proposed solutions ensure efficient resource utilization and prioritize traffic flows based on their criticality.

Furthermore, the SDN paradigm is inherently stateless, meaning it does not implement feedback control mechanisms. Transitioning to a stateful SDN requires the development of mechanisms to maintain the current state and context of network devices and flows. This allows for *more granular control and management* of network resources. To enable this, a network model is required to predict Key Performance Indicators (KPIs), such as latency, which serve as inputs to SDN frameworks to take further actions. In this thesis, a GNN-based model is leveraged to predict E2E latency, enhancing the Admission Control (AC) decisions for multiple flows. These methods represent a significant step forward in addressing the limitations of stateless SDN and improving the performance of time-sensitive applications.

Additionally, this thesis tackles key challenges related to DP congestion control in B5G networks. Another critical dimension of this work involves leveraging advanced transport layer protocols, such as L4S, to optimize congestion control mechanisms in SDN. By adopting dual-queue management and Explicit Congestion Notification (ECN)-based signaling, this research ensures seamless support for low-latency and high-throughput requirements. These innovations bridge the gap between current congestion management

practices and the demands of next-generation applications.

Finally, this thesis seeks to enhance the efficiency of CEC edge nodes by enabling kernel-level packet processing, which offers substantial performance improvements over traditional user-space processing approaches. Figure 1.1 summarizes the journey of this Doctor of Philosophy (PhD) work using a mind map.

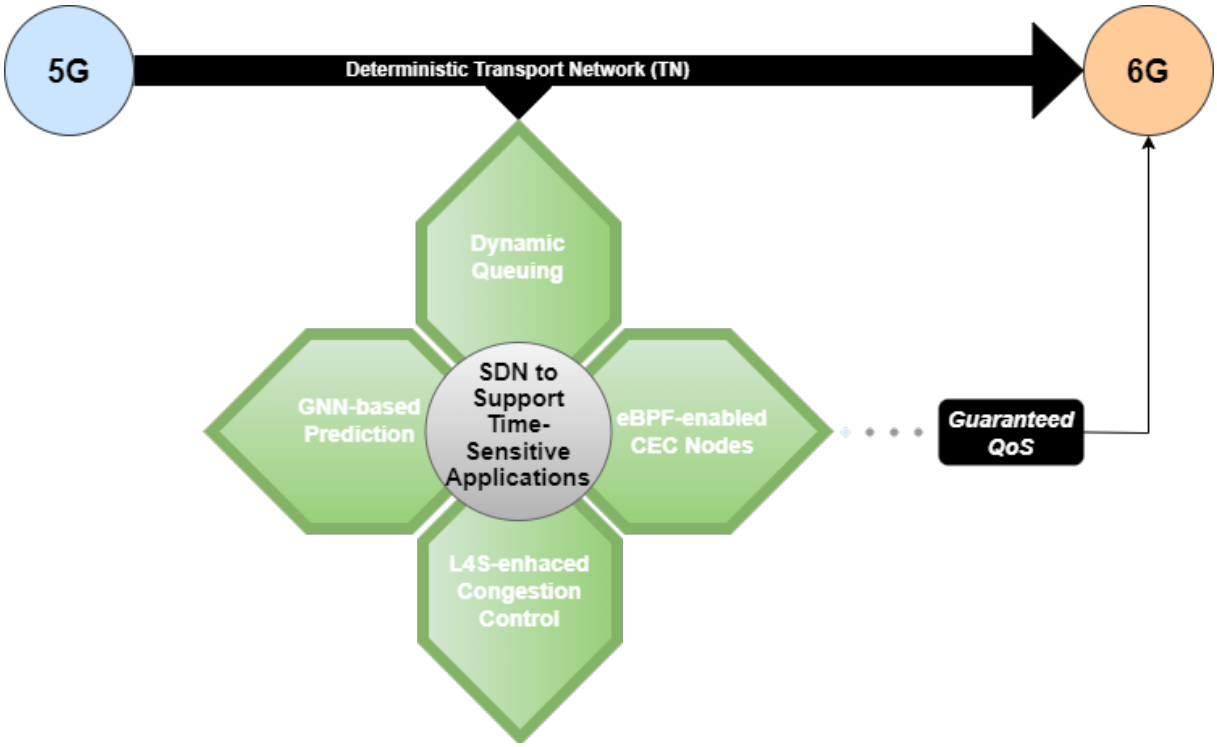


Figure 1.1: Thesis story map

In the following, we outline, in detail, the key challenges tackled in this thesis and the solutions proposed to address them:

1. **SDN-based Path Selection for Time-Sensitive Applications Using Dynamic Queuing:**

   (a) *Challenge description:* Ultra-Reliable Low Latency Communications (URLLC) services in B5G demand an E2E latency guarantee of less than 5 milliseconds for critical applications such as autonomous systems. The TN often acts as a bottleneck, significantly contributing to the E2E delay. The main challenges in the TN include:

   - **Lack of E2E latency visibility:** Many existing solutions focus on hop-by-hop bounded latency without considering overall E2E latency, limiting their effectiveness in multi-path networks or scenarios requiring rerouting during congestion.
   - **Static queue configurations:** Fixed priority-based queuing mechanisms cannot adapt dynamically to changing traffic patterns and load conditions, leading to suboptimal resource utilization.
   - **Scalability limitations:** The challenge of maintaining consistent and deterministic traffic handling grows significantly in large-scale and dynamic network environments, especially under the diverse demands of 6G.

- **Inefficiency in handling diverse service requirements:** Existing methods lack the ability to dynamically prioritize multiple URLLC flows with varying criticality levels, resulting in degraded performance for the most delay-sensitive services during congestion.

(b) *Proposed solutions:* To address these challenges, this thesis proposes the **Software-Defined Low Latency (SDLL)** framework, an innovative solution that leverages the flexibility and programmability of SDN. The SDLL framework includes the following key components:

i. **Dynamic Queue Management (QM):** A real-time queuing mechanism that dynamically adjusts the number of queues based on traffic load and service criticality. This ensures prioritization of delay-sensitive traffic and prevents interference from less critical flows.

ii. **Traffic Engineering (TE) Algorithm:** A scalable TE mechanism for adaptive path selection and RA. This algorithm actively monitors network conditions and reconfigures paths to avoid congestion while ensuring deterministic E2E latency.

iii. **Integration of QoS Flow Identifier (QFI):** SDLL uses 5G's QFI [6] information to map services to traffic classes within the TN. This allows precise QoS provisioning, ensuring that each service is handled according to its specific latency and reliability requirements.

These components collectively address the challenges of scalability, adaptability, and dynamic traffic management, offering an effective solution for achieving deterministic low-latency communication in 6G TN.

(c) *Publications:*

- **SDN Framework for QoS Provisioning and Latency Guarantee in 5G and Beyond** Authors: Sofiane Messaoudi, Adlen Ksentini, Christian Bonnet Published in: *IEEE 20th Consumer Communications & Networking Conference (CCNC) 2023, Las Vegas, USA*

2. **SDN-based AC for Time-Sensitive Applications Leveraging GNNs:**

(a) *Challenge description:* The growing complexity and scale of B5G networks, which support diverse services such as autonomous vehicles, e-health, and the metaverse, pose significant challenges in managing traffic flows with different latency, bandwidth, and reliability requirements. Key difficulties include:

- **Ensuring QoS:** Traffic flows in B5G networks have varying requirements, and a robust AC mechanism is needed to ensure that only flows that can meet these QoS parameters are admitted. This is critical for preventing congestion and maintaining low latency for time-sensitive services.
- **Scalability of AC Mechanisms:** Traditional AC mechanisms, which often rely on static rules or heuristic-based approaches, are insufficient for large-scale and dynamic networks. The fluctuating nature of network conditions, such as varying traffic loads and congestion levels, requires a flexible and adaptive AC strategy capable of real-time optimization.
- **Inaccurate Network Models:** Existing network models, including queuing theory-based and packet-level simulations, are either computationally intensive or overly simplistic, leading to inaccuracies in latency and congestion predictions, especially in large-scale, dynamic environments.

(b) *Proposed solutions:* This thesis proposes a novel SDN-based AC mechanism that leverages GNNs to predict network latency in real-time and optimize AC decisions. This complements the first contribution by enabling predictive traffic management for large-scale, dynamic networks.The proposed solution, called **Graph Neural Network-Admission Control (GNN-AC)**, consists of two main layers:

i. **Network Delay Predictor (NetDelP):** The NetDelP layer employs the RouteNet-Fermi (RouteNet-F) model [7], a GNN-based framework, to predict network latency for various topologies and traffic patterns. This layer provides accurate, real-time latency predictions that are crucial for effective traffic management in B5G networks.

ii. **Admission Control Agent (AdConAgt):** The AdConAgt layer operates within the SDN framework to regulate traffic flows based on the latency predictions generated by the NetDelP layer. By integrating dynamic network state information and GNN-based latency predictions, AdConAgt can make adaptive decisions that guarantee QoS for accepted traffic flows while preventing congestion.

Together, these components optimize the AC process by ensuring efficient allocation of network resources and meeting the latency and reliability requirements of various service types. Additionally, the solution leverages 5G DP specifications and QFI to map 5G services to TN traffic classes, allowing for scalable and resource-aware handling of diverse service requirements.

(c) *Publications:*

- **GNN-based SDN Admission Control in Beyond 5G Networks** Authors: Sofiane Messaoudi, Adlen Ksentini, Franck Messaoudi, Christian Bonnet Published in: *IEEE Global Communications Conference 2023, Kuala Lumpur, Malaysia*

3. **SDN-based Congestion Control for Time-Sensitive Applications Adopting L4S Transport Layer:**

(a) *Challenge description:* The increasing demand for real-time, immersive applications like Virtual Reality (VR), AR, and high-quality video streaming in 5G and Beyond (5G&B) networks imposes stringent requirements for low latency, minimal packet loss, and high throughput. Addressing these requirements entails overcoming several key challenges:

- **Limitations of Traditional Transmission Control Protocol (TCP) Congestion Control:** TCP's slow-start mechanism and aggressive rate reduction in response to congestion result in high E2E latency and inefficient bandwidth usage, making it unsuitable for time-sensitive applications [8].
- **Handling High Traffic Loads:** Immersive applications generate high traffic loads, necessitating mechanisms that can dynamically adapt to varying network conditions while maintaining low latency and high throughput.
- **Congestion Prediction and Prevention:** Accurate detection and proactive avoidance of congestion are essential to minimize packet loss, avoid unnecessary retransmissions, and ensure smooth traffic flow.
- **QM in SDN:** Optimizing queue management at the switch level is critical for fair RA and prioritization of time-sensitive traffic over regular traffic.

(b) *Proposed solutions:* To address these challenges, this thesis introduces **Software-Defined Networking-Low-Latency, Low-Loss, Scalable Throughput (SDN-L4S)**, an SDN-based congestion control framework that integrates L4S techniques into 5G&B networks. The solution includes the following components:

i. **Congestion Control Framework:** The integration of L4S techniques within the SDN architecture enhances congestion control by leveraging ECN for packet marking during congestion. The framework employs a dual-queue mechanism to prioritize L4S traffic over regular traffic, ensuring improved E2E latency and throughput.

ii. **Adaptive QM:** An adaptive QM algorithm dynamically adjusts queue prioritization based on network load and traffic requirements. This ensures that L4S traffic maintains low latency and high reliability even under heavy load conditions.

iii. **Dynamic Transmission Rate Adjustment:** By leveraging ECN feedback, the transmission rate of L4S traffic is dynamically adjusted according to network congestion levels. This reduces packet loss, prevents bottlenecks, and optimizes the utilization of available network resources.

By integrating L4S techniques into the SDN architecture, the proposed solution effectively manages diverse traffic types while guaranteeing ultra-low latency and high reliability for critical applications.

(c) *Publications:*

- **SDN-based L4S Congestion Control in Beyond 5G** Authors: Sofiane Messaoudi, Adlen Ksentini, Franck Messaoudi, Christian Bonnet Published in: *IEEE 25th International Conference on High Performance Switching and Routing (HPSR) 2024, Pisa, Italy*

4. **SDN for CEC Computing Nodes Interconnection Utilizing eBPF:**

(a) *Challenge description:* The evolving landscape of cloud computing towards CEC introduces significant challenges in interconnecting cloud, edge, and far-edge nodes. These challenges are further amplified when considering:

- **Limited Control Over Network Infrastructure:** Cloud and edge nodes are often connected via third-party Internet Service Provider (ISP), and edge resources have limited control over the underlying network. This limitation hinders the enforcement of Service-Level Agreement (SLA) requirements, including low latency, high bandwidth, and minimal packet loss.
- **Programmability and QoS Enforcement:** There is a need for network programmability to allow network orchestrators to specify QoS levels and resiliency for each service, especially given the diverse nature of workloads and interconnection requirements across the CEC.
- **Resiliency and Route Selection Flexibility:** The Software-defined Wide Area Network (SD-WAN) framework must provide route selection flexibility and ensure resiliency across different CEC nodes, which are often connected via heterogeneous and unreliable networks.

(b) *Proposed solutions:* This thesis proposes a novel SD-WAN framework named **High-Efficiency Layered Infrastructure with eBPF for Optimized SD-WAN (HELIOS)**, which leverages eBPF to enable efficient nodes interconnection within a CEC environment. The solution addresses the identified challenges by integrating key SDN principles and offering several innovations:

i. **eBPF-powered CEC Nodes Interconnection:** The solution leverages eBPF in Linux kernel for packet processing within CEC nodes interconnection. By performing custom packet handling directly in the kernel, eBPF ensures low-latency, high-performance processing that surpasses the bottlenecks associated with user-space CEC solutions.

ii. **Centralized Control via SDN Controller:** The eBPF-based CEC nodes interconnection are centrally controlled by an SDN controller (i.e., Open Network Operating System (ONOS)), which provides the programmability (i.e., eBPF maps updates using Google Remote Procedure Calls (gRPC)) and QoS enforcement for managing microservices across the CEC. This centralized approach allows for dynamic network optimization based on SLA requirements and resources availability.

iii. **Traffic Steering, Load Balancing (LB), and QoS Management:** The eBPF-based CEC nodes interconnection allows for flexible traffic steering and LB across different overlay links, enabling optimal RA and ensuring that traffic prioritization is maintained according to application needs. The eBPF framework also facilitates the enforcement of QoS policies to ensure the desired network performance for critical services.

These innovations together provide a robust solution for interconnecting CEC nodes with enhanced performance and programmability.

(c) *Publications:*

- **When SD-WAN meets eBPF** Authors: Sofiane Messaoudi, Franck Messaoudi, Adlen Ksentini, Christian Bonnet Published in: *IEEE International Conference on Communications (ICC) 2025, Montreal, Canada*

1.4 Thesis Structure

The rest of this thesis is organized into six chapters, each focusing on a specific aspect of deterministic network design and optimization.

Chapter 2 provides a comprehensive exploration of the state of the art on several key concepts critical to the research presented in this thesis. These include topics such as the 5G-related DP, URLLC, QFI, and QoS, as well as Time Sensitive Networking (TSN), SDN, and DP programming, including path selection, traffic AC, and congestion management, among others. This chapter establishes the foundation for understanding the current technological landscape and highlights the limitations that this thesis seeks to address.

Chapter 3 delves into SDN-based Path Selection for Time-Sensitive Applications Using Dynamic Queuing. It introduces the SDLL framework, detailing its dynamic QoS mapping and QM mechanisms. These innovations are aimed at achieving low-latency guarantees in the TNs, a critical component for deterministic 5G&B communication systems.

Chapter 4 focuses on SDN-based AC Leveraging GNNs. It presents the GNN-AC solution, which combines ML techniques with SDN principles. The chapter emphasizes the optimization of AC and RA decisions in real-time, ensuring that stringent QoS requirements are met while preventing network congestion.

Chapter 5 addresses SDN-based Congestion Control Adopting L4S Transport Layer. This chapter discusses the integration of L4S congestion control techniques within SDN architectures. It introduces novel algorithms designed to enhance QM and ECN marking. These innovations enable the system to predict and prevent congestion, ensuring high reliability and ultra-low latency for time-sensitive applications.

Chapter 6 explores SDN for CEC Computing Nodes Interconnection Utilizing eBPF. This chapter highlights the application of eBPF in CEC nodes interconnection to enable stateful QoS enforcement for microservices deployed across the CEC environments. The proposed solution leverages eBPF's capabilities for high-performance packet processing and efficient network programmability.

Finally, Chapter 7 concludes the thesis by summarizing the contributions made, evaluating their impact, and identifying potential avenues for future research. This chapter provides a critical reflection on the outcomes of the research and outlines directions for advancing deterministic, low-latency communications

in 6G networks.

By addressing these critical challenges, this thesis establishes a foundation for SDN-enabled deterministic networking to support time-sensitive applications in next-generation networks.

Chapter 2

State of The Art

2.1 Introduction

The rapid evolution of time-sensitive applications, such as autonomous vehicles, requires network architectures capable of meeting stringent latency and reliability demands. These applications impose significant challenges on existing network infrastructures, necessitating advancements in network design, RA, and QoS guarantees. In the context of 5G and URLLC, the integration of SDN is essential to meet the high-performance requirements of these emerging use cases. SDN plays a vital role by decoupling the CP from the DP, enabling centralized, real-time management of network resources. This approach optimizes the flow of time-sensitive data and ensures predictable network behavior. Together, SDN and 5G/URLLC form a comprehensive solution to support the next generation of time-sensitive applications.

This chapter begins by providing an overview of 5G networks, which serve as the foundation for the development of next-generation networks (i.e., 6G), with a particular focus on the DP, URLLC, QFI, and QoS, as these constitute the primary areas of our contributions. Subsequently, the chapter introduces the key technologies and approaches, such as TSN, and SDN. Moreover, it presents a comprehensive review of state-of-the-art solutions and frameworks that address these challenges, by focusing on DP programming including path selection, traffic AC, and congestion management. By analyzing existing methodologies, this chapter establishes a reliable foundation for enabling deterministic and low-latency communication in B5G networks. The insights presented here lay the groundwork for identifying research gaps and proposing innovative approaches in subsequent chapters.

2.2 5G Networks

5G networks have emerged as a transformative foundation for industrial digitalization and advanced communication technologies. They promise ultra-reliable services, high mobility, ultra-fast data rates, and extremely low latency, addressing limitations of previous generations. Unlike earlier technologies, such as 4^{th} Generation (4G) Long Term Evolution-Advanced (LTE-A), which supported a maximum Downlink (DL) data rate of up to 3 Gbps, an Uplink (UL) rate of 1.5 Gbps, and latency between 30–50 ms with connectivity for approximately 600 users per cell, 5G enables a broader range of applications. Applications such as VR, AR, Ultra High Definition (UHD) streaming, video conferencing, and 360° video streaming, which were constrained by 4G capabilities, are now supported through 5G's advanced features and technologies. Key enabling technologies include Massive Multiple Input Multiple Output (MIMO), millimeter wave (mmWave) communication, full-duplex radio, device-to-device (D2D) communication, Ultra-Dense Networks (UDN), multi-Radio Access Technology (RAT) integration, and cognitive radio.

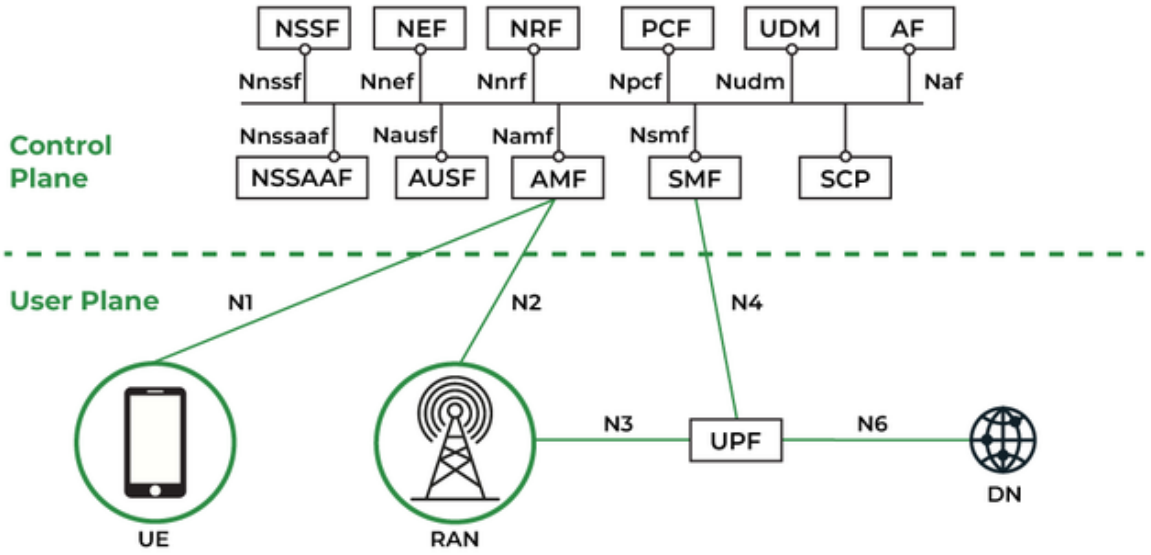


Figure 2.1: 5G networks architecture [11]

These innovations collectively address challenges in data rates, connectivity, and latency [9, 10].

As illustrated in Figure 2.1, the 5G network architecture is split into two planes: the CP, which contains the CN functions except for the User Plane Function (UPF), and the User Plane, which includes the User Equipment (UE), RAN, and DP. All these components can be categorized into three main segments: the RAN, the CN, and the TN. Although the TN is not explicitly shown in Figure 2.1, its role is critical in the DP. For a detailed illustration of the TN architecture, refer to Figure 2.3.

The RAN consists of multiple gNodeBs (gNBs), which replace the evolved Node B (eNB) used in 4G Long Term Evolution (LTE). These gNBs handle all radio-related functions, including providing wireless connectivity to UE and supporting advanced technologies like Massive MIMO and mmWave, thereby enabling high-capacity, low-latency connections.

The CN in 5G adopts a Service-Based Architecture (SBA), dividing Network Functions (NFs) into modular and interoperable entities, as defined by 3rd Generation Partnership Project (3GPP). This modularity enables flexibility and supports diverse use cases through network slicing. Key NFs include the Access and Mobility Management Function (AMF), responsible for UE registration and mobility, and the Session Management Function (SMF), which manages session creation and Internet Protocol (IP) address allocation. The UPF serves as a gateway between the RAN and external networks, handling data forwarding, QoS enforcement, and packet processing. The UPF interacts with the RAN to map QoS flows to Data Radio Bearer (DRB), ensuring differentiated treatment of traffic (See Table 2.1).

The TN interconnects the RAN, CN and Internet, enabling efficient data routing and synchronization across the network. The TN is critical for ensuring low-latency and high-throughput communication, particularly for time-sensitive applications like autonomous vehicles, industrial automation, and telemedicine. Advanced techniques, including SDN and Network Function Virtualization (NFV), enhance the TN's adaptability and scalability, making it a vital component of the 5G architecture.

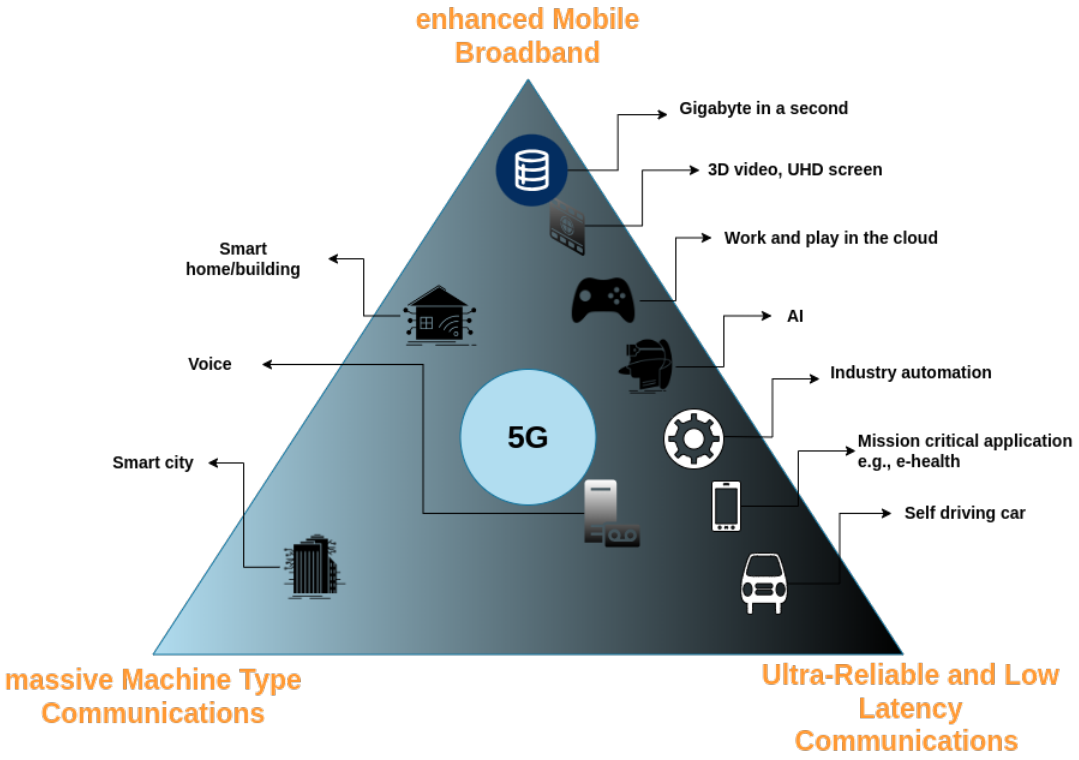


Figure 2.2: Key use cases enabled by 5G

To meet the stringent requirements of 5G, a new radio air interface, known as 5G New Radio (5G NR), has been developed. 5G NR operates across a broad spectrum, from sub-6 GHz frequencies to mmWave bands (i.e., up to 100 GHz), ensuring both flexibility and scalability. The 3GPP has structured its development, as of now, into four phases: release 15, release 16, release 17, and release 18 [12, 13]. Each release has progressively addressed KPIs for 5G including three core use cases: enhanced Mobile Broadband (eMBB), massive Machine Type Communication (mMTC), and URLLC, while introducing innovations to meet growing demands for connectivity, efficiency, and flexibility. which enhanced reliability, latency, and connectivity capabilities.

As depicted in Figure 2.2, these KPIs enable diverse use cases such as high-speed broadband (i.e., eMBB), massive Internet of Things (IoT) connectivity (i.e., mMTC), and mission-critical communications (i.e., URLLC). eMBB ensures consistent user experiences for applications like UHD streaming, AR, and VR. mMTC supports IoT networks with massive device connectivity, and URLLC extends network reliability and latency limits, enabling applications like industrial automation and remote surgery.

Although 5G has brought a lot of innovation, it is in its initial phase of commercialization and need to be improved to meet the next generation of critical applications. The architecture's scalability, flexibility, and advanced features provide a robust platform for the migration towards B5G and 6G networks. The following subsections focuses on critical components of 5G, particularly the DP, URLLC, and QoS mechanisms, which form the basis of our contributions. By addressing these areas, we establish a foundation for achieving deterministic, low-latency communication and exploring innovative solutions for future networks.

2.2.1 5G DP and TN

5G DP: 5G networks rely on a highly flexible DP, which separates user traffic from control functions as depicted in Figure 2.1. A 5G DP refers to the technical framework that defines how user's data is transmitted and managed within a 5G network. It is primarily realized through Packet Data Unit (PDU) sessions and QoS flows, enabling efficient and flexible data communication. A PDU session establishes an E2E connection between the UE and a data network, such as the Internet or private enterprise networks. Each session supports specific data protocols, including IP, Ethernet, or unstructured formats. Multiple PDU sessions can operate simultaneously for a single UE, each tailored for different applications or services. Within each PDU session, data traffic is categorized into QoS flows, where each flow is assigned a QFI and managed based on its latency, reliability, and bandwidth requirements. These QoS flows are mapped to DRBs in the RAN to ensure optimized handling of diverse service needs. Furthermore, 5G DPs integrate seamlessly with network slicing, allowing virtualized and isolated network slices to serve specific applications, industries, or user groups while adhering to unique QoS parameters.

5G TN: The 5G TN is the critical infrastructure that facilitates high-capacity, low-latency data transfer between various components of the 5G architecture, including the RAN, the CN, and external networks such as the Internet or cloud services. The TN in 5G is designed to meet the stringent requirements of emerging use cases like eMBB, URLLC, and mMTC. It employs advanced technologies such as SDN and NFV to ensure flexible and dynamic RA. Additionally, it supports segment routing, packet-based transport protocols, and precise synchronization to guarantee deterministic performance. The 5G TN incorporates optical transport, IP/Multiprotocol Label Switching (MPLS) layers, and Ethernet-based solutions, enabling it to handle the substantial bandwidth demands of 5G applications while maintaining high reliability and scalability. This TN acts as a backbone, connecting distributed elements like edge computing nodes, gNBs, and data centers to provide seamless and efficient connectivity across the 5G ecosystem.

A key feature of the RAN in the 5G architecture is its functional split into three components: Radio Unit (RU), Distributed Unit (DU), and Centralized Unit (CU). This division optimizes performance, scalability, and flexibility.The RU, positioned close to the user, handle Radio Frequency (RF) operations to minimize signal degradation and latency. The DU manages real-time processing tasks such as scheduling and Medium Access Control (MAC)-layer functions, while the CU centralizes non-real-time operations, enabling efficient resource pooling and coordination across multiple sites. This architecture supports advanced features like network slicing and enables operators to adapt deployments to varying traffic demands, geographic constraints, and latency requirements, enhancing overall network efficiency.

As shown in Figure 2.3, the architecture of the 5G TN is illustrated. It is composed of three main links that connect the RAN components (i.e., the RU, DU, and CU), the CN, and the external network:

1. **Fronthaul:** Connects the gNB's RU and DU components.
2. **Midhaul:** Provides connectivity between the DU and CU.
3. **Backhaul:** Links the RAN's CU to the CN and facilitates end-user data transport to external networks, including the Internet and edge services.

2.2.2 URLLC

URLLC is a cornerstone of 5G, enabling latency-sensitive applications with E2E latency below 1 millisecond and reliability above 99.999%. It incorporates advanced features such as Hybrid Automatic Repeat

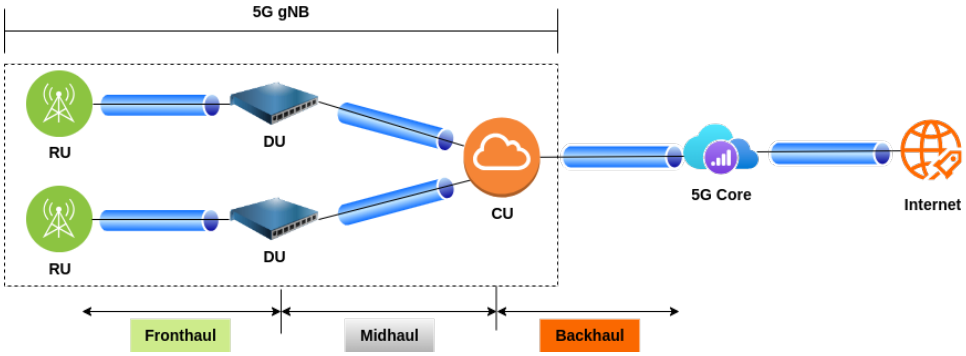


Figure 2.3: 5G TN architecture

Request (HARQ), short Transmission Time Interval (sTTI), and preemptive scheduling.

URLLC underpins a new generation of use cases across vertical industries, including:

- **Intelligent Transportation:** Autonomous driving and Vehicle-to-Everything (V2X) communication.
- **Remote Healthcare:** Remote surgery and telemedicine applications.
- **Industrial Automation:** Cloud robotics, and Industry 4.0 use cases.

In addition, URLLC supports immersive technologies such as Tactile Interaction (TI), AR, VR, and Mixed Reality (MR). These applications demand stringent performance metrics summarized below [14]:

- E2E latency: Maximum of 1 ms.
- Reliability: Less than 10^{-5} packet drop rate.
- Availability: Up to 99.999%.

Deploying URLLC in real-world scenarios necessitates enhancements to existing technologies and the development of new ones. Key enablers such as HARQ, efficient scheduling, and robust network management must work together to meet the low latency and high reliability targets. These stringent requirements represent a significant challenge for 5G&B network design [15]. Looking ahead, the 6G networks are expected to further reduce E2E latency to as low as 0.1 milliseconds, highlighting the ongoing evolution of URLLC technologies [16].

2.2.3 5G QoS Framework and QFI

The 5G QoS framework provides precise traffic differentiation and management by assigning a QFI to each flow. Unlike legacy systems, 5G supports both Guaranteed Bit Rate (GBR) and Non-GBR services, enabling flexible and efficient RA. The QoS model is governed by key parameters, including:

- **5G QoS Identifier (5QI) Value:** A unique Identifier (ID) for each CoT.
- **Resource Type (RT):** Specifies the type of resource as one of the following: GBR, Non-GBR, or Delay-Critical (DC) GBR.
- **Priority Level (PL):** Determines the handling order of packets, where lower values correspond to higher priorities.
- **Packet Delay Budget (PDB):** Represents the maximum allowable time for a packet to traverse the network from Sender to Receiver.

- **Packet Error Rate (PER):** Denotes the acceptable ratio of lost packets to transmitted packets.
- **Maximum Data Burst Volume (MDBV):** Specifies the maximum amount of data that can be sent in a single burst.
- **Average Window (AW):** Specifies the average time window used for certain QoS measurements.

The QFI mapping process, as illustrated in Equation 2.1, integrates these parameters:

$$QFI = f(5\text{QI}, \text{RT}, \text{PL}, \text{PDB}, \text{PER}, \text{MBDV}, \text{AW}) \tag{2.1}$$

To establish a communication link, a UE initiates a session that creates dedicated bi-directional bearers. In the CN, these bearers encapsulate the UE's IP packets in GPRS Tunneling Protocol (GTP) tunnels between the gNB and the UPF. The GTP header includes QoS information, which the gNB and UPF apply to manage the tunnel. This information corresponds to the 5G QFI field, a crucial component of QoS management.

5G QoS flows use the 5QI mechanism, classifying packets into predefined QoS classes tailored to application requirements. Each QoS class is characterized by attributes such as PL, PDB, and PER. Approximately two dozen standardized 5QI values are grouped into three resource types:

1. **GBR:** Ensures a dedicated rate for critical services.
2. **Non-GBR:** Provides flexible RA without guaranteeing specific rates.
3. **DC GBR:** Supports applications with stringent latency and reliability requirements.

In addition, QoS flows are mapped to Differentiated Services Code Point (DSCP) values for consistent QoS treatment across 5G networks and the Internet [17]. This specifies a set of 5QI to DSCP mappings to reconcile the marking recommendations offered by the 3GPP with the recommendations offered by the IETF, respectively. This process involves border switches parsing GTP headers to extract the QFI and mapping it to IP headers for both UL and DL traffic.

This thesis relies on the new 5G DP specifications as a basis, where the 5QI mechanism is used. In fact, this QoS mechanism ensures that each flow meets its unique performance requirements across backhaul and midhaul networks. For instance, traffic classified with higher priority and stricter latency constraints, such as IP Multimedia Subsystem (IMS) signaling, receives preferential treatment compared to buffered streaming.

Table 2.1 illustrates a standardized mapping of 5QI values to QoS characteristics, adapted from European Telecommunications Standards Institute (ETSI) standards [17].

Table 2.1: Standardized 5QI to QoS characteristics mapping

5QI Value	**Resource Type**	**Default Priority Level**	**Packet Delay Budget**	**Packet Error Rate**	**Default Maximum Data Burst Volume (NOTE 2)**	**Default Average Window**	**Example Services**
1	GBR (NOTE 1)	20	100 ms	10^{-2}	N/A	2000 ms	Conversational Voice
2		40	150 ms	10^{-3}	N/A	2000 ms	Conversational Video (Live Streaming)

3		30	50 ms	10^{-3}	N/A	2000 ms	Real-Time Gaming, V2X messages, Electricity distribution - medium voltage, Process automation - monitoring
4		50	300 ms	10^{-6}	N/A	2000 ms	Non-Conversational Video (Buffered Streaming)
65		7	75 ms	10^{-2}	N/A	2000 ms	Mission Critical user plane Push To Talk voice (e.g., Mission Critical Push To Talk (MCPTT))
66		20	100 ms	10^{-2}	N/A	2000 ms	Non-Mission-Critical user plane Push To Talk voice
67		15	100 ms	10^{-3}	N/A	2000 ms	Mission Critical Video user plane
75		25	50 ms	10^{-2}	N/A	2000 ms	V2X messages
5	Non-GBR (NOTE 1)	10	100 ms	10^{-6}	N/A	N/A	IMS Signaling
6		60	300 ms	10^{-6}	N/A	N/A	Video (Buffered Streaming) TCP-based (e.g., World Wide Web (WWW), Electronic Mail (E-Mail), chat, File Transfer Protocol (FTP), peer-to-peer (P2P) file sharing, progressive video, etc.)
7		70	100 ms	10^{-3}	N/A	N/A	Voice, Video (Live Streaming), Interactive Gaming
8		80	300 ms	10^{-6}	N/A	N/A	Video (Buffered Streaming) TCP-based (e.g., WWW, E-MAIL, chat, FTP, P2P file sharing, progressive video, etc.)
9		90					
69		5	60 ms	10^{-6}	N/A	N/A	Mission Critical delay sensitive signaling (e.g., MCPTT signaling)
70		55	200 ms	10^{-6}	N/A	N/A	Mission Critical Data (e.g. example services are the same as QoS Class Identifier (QCI) 6/8/9)
79		65	50 ms	10^{-2}	N/A	N/A	V2X messages
80		68	10 ms	10^{-6}	N/A	N/A	Low Latency eMBB applications, AR

82	DC GBR	19	10 ms (NOTE 4)	10^{-4}	255 bytes	2000 ms	Discrete Automation (see Technical Specification (TS) 22.261 [18])
83		22	10 ms (NOTE 4)	10^{-4}	1354 bytes (NOTE 3)	2000 ms	Discrete Automation (see TS 22.261 [18])
84		24	30 ms (NOTE 6)	10^{-6}	1354 bytes	2000 ms	Intelligent Transport Systems (see TS 22.261 [18])
85		21	5 ms (NOTE 5)	10^{-5}	255 bytes	2000 ms	Electricity Distribution - high voltage (see TS 22.261 [18])

NOTES:

1. A packet which is delayed more than PDB is not counted as lost, thus not included in the PER.
2. It is required that default MDBV is supported by a Public Land Mobile Network (PLMN) supporting the related 5QIs.
3. This MDBV value is set to 1354 bytes to avoid IP fragmentation for the Internet Protocol version 6 (IPv6)-based, Internet Protocol security (IPsec)-protected GTP tunnel to the 5G-Access Network (AN) node.
4. A delay of 1 ms for the delay between a UPF terminating N6 and a 5G-AN should be subtracted from a given PDB to derive the PDB that applies to the radio interface.
5. A delay of 2 ms for the delay between a UPF terminating N6 and a 5G-AN should be subtracted from a given PDB to derive the PDB that applies to the radio interface.
6. A delay of 5 ms for the delay between a UPF terminating N6 and a 5G-AN should be subtracted from a given PDB to derive the PDB that applies to the radio interface.

2.3 TSN

TSN is a set of Institute of Electrical and Electronics Engineers (IEEE) 802.1 standards developed to enable deterministic, low-latency communication over Ethernet networks. The majority of projects define extensions to the IEEE 802.1Q – Bridges and Bridged Networks, which describes Virtual Local Area Network (VLAN) and network switches [19]. These extensions in particular address transmission with very low latency and high availability. TSN aims to extend Ethernet capabilities by incorporating features that ensure predictable, time-bound data delivery, which is crucial for applications requiring real-time communication, such as industrial automation, autonomous vehicles, and professional audio-video systems.

Key features of TSN include:

- **Time synchronization:** TSN relies on precise time synchronization mechanisms, such as IEEE 802.1AS [20], ensuring that all devices in the network share a synchronized clock to guarantee timely delivery of data.
- **Traffic scheduling:** The IEEE 802.1Qbv standard [21] defines time-sensitive traffic scheduling, which ensures that high-priority data flows are sent at specific times, reducing latency and avoiding packet collisions.

- **Traffic shaping:** The IEEE 802.1Qav standard [21] ensures that data traffic is shaped and managed in a way that prioritizes time-sensitive packets, ensuring minimal delay for critical data.
- **Frame preemption:** The IEEE 802.1Qbu standard [22] enables preemption of low-priority traffic to allow for the immediate transmission of higher-priority, time-sensitive data.

These features make TSN particularly well-suited for environments where low latency and reliability are paramount. However, while TSN significantly improves the predictability of network behavior, several limitations affect its scalability and effectiveness in large-scale, dynamic environments. Indeed, TSN provide some level of deterministic behavior but fall short in achieving E2E low latency under varying network loads and conditions.

2.3.1 Challenges and Limitations of TSN

Although TSN provides essential features for deterministic networking, several challenges hinder its deployment in large, complex, or dynamic systems [23]. These limitations include:

- **Scalability:** TSN relies heavily on precise synchronization across all devices in the network. As the size of the network grows, maintaining synchronization becomes increasingly difficult. Large-scale deployments, especially in distributed environments, require extensive coordination to ensure all devices remain time-synchronized, leading to potential overhead and complexity.
- **Configuration Complexity:** Setting up and configuring a TSN network is a complex process. It requires detailed knowledge of time synchronization, traffic scheduling, and network topology. As networks grow, managing and maintaining the configurations becomes increasingly error-prone and challenging.
- **Limited Adaptability:** TSN, by design, is focused on deterministic behavior with fixed schedules for data transmission. In dynamic environments where network traffic patterns or nodes mobility change frequently, TSN networks struggle to adapt in real-time, resulting in inefficiencies and potential delays in data delivery.
- **Interoperability Issues:** Integrating TSN into existing Ethernet-based networks, or with non-TSN devices, can lead to compatibility issues. Since TSN involves specialized hardware and software for time synchronization and traffic management, older devices or non-compliant equipment can interfere with the performance and reliability of the network.
- **Resource Management:** TSN provides mechanisms for traffic prioritization, but it lacks mechanisms for optimizing resource usage across the network. It doesn't provide dynamic RA in response to changing network conditions, which can limit the efficiency of time-sensitive traffic under varying load or resource constraints.

These challenges indicate a clear need for a more flexible and scalable approach to handle the complexities of modern network environments, particularly for time-sensitive applications.

2.4 SDN

It is widely recognized that 6G network architecture will heavily rely on SDN and will support URLLC traffic [24]. An SDN controller provides a global, centralized view of the network, enabling efficient decision-making and communication with network devices. This makes SDN a key enabler for improving QoS provisioning for critical applications.

SDN is an architectural paradigm designed to make networks more agile, programmable, and capable of supporting multi-tenancy [25]. The fundamental principle of SDN is the decoupling of the DP (responsible for packet forwarding) from the CP (responsible for decision-making and policy enforcement). SDN has been extensively promoted by the Open Networking Foundation (ONF) [26], which defines its core principles and architecture.

The SDN architecture is typically organized into three layers, as shown in Figure 2.4:

- **Application Layer**: This layer hosts SDN applications, which specify high-level network requirements and policies. These applications communicate their demands to the CP via NorthBound Interfaces (NBIs), which vary depending on the controller and the application (e.g., Java Application Programming Interface (API), REpresentational State Transfer (REST) API, gRPC, Open Services Gateway initiative (OSGi), Network Configuration Protocol (NETCONF), Yet Another Next Generation (YANG)).
- **Control Layer:** This layer is represented by the SDN controller (e.g., ONOS, POX, Ryu, OpenDaylight, Floodlight). The controller acts as an intermediary between the application and infrastructure layers, translating application-level requirements into low-level commands for network devices. Additionally, it gathers and provides network state information to applications, enabling intelligent decision-making. The SDN controller is central to network orchestration, offering mechanisms for routing, monitoring, scheduling, and QoS management.
- **Infrastructure Layer:** This layer consists of the underlying network elements (e.g., switches, routers) that perform data forwarding. These devices expose their capabilities to the controller via SouthBound Interfaces (SBIs), which are switch- and protocol-specific (e.g., OpenFlow, NETCONF, Border Gateway Protocol-Link State (BGP-LS), Programming Protocol-independent Packet Processors (P4) Runtime, Simple Network Management Protocol (SNMP)). The infrastructure layer executes the commands issued by the controller, ensuring the enforcement of network policies.

By abstracting and centralizing the control of network elements, SDN enhances network programmability, simplifies management, and enables the dynamic allocation of resources to meet the stringent demands of emerging applications.

2.4.1 SD-WAN

SD-WAN leverages SDN principles to optimize Wide Area Network (WAN). It provides centralized management, intelligent path selection, and improved performance for applications with strict QoS requirements. Figure 2.5 depicts the SD-WAN architecture.

SD-WAN is an evolution of traditional WAN technologies, such as MPLS (Requests For Comments (RFC) 3031) [27], IPsec (RFC 4301) [28], and Generic Routing Encapsulation (GRE) (RFC 2784) [29], among others. Unlike WAN, SD-WAN lacks an official standard but is typically built around common principles and protocols. Organizations such as the Metro Ethernet Forum (MEF) [30] provide best practices and frameworks to guide SD-WAN implementations. SD-WAN meets the need for scalable, flexible connectivity across distributed networks by decoupling the CP from the DP. This separation enables centralized network management while routing traffic over WAN links based on application needs and network conditions. Figure 2.5 illustrates the SD-WAN architecture, comprising:

- **Management Plane (MP):** A centralized interface (e.g., dashboard) for managing WAN infrastructure. Administrators monitor network performance, configure traffic policies, and enforce security, communicating requirements to the CP via NBIs.

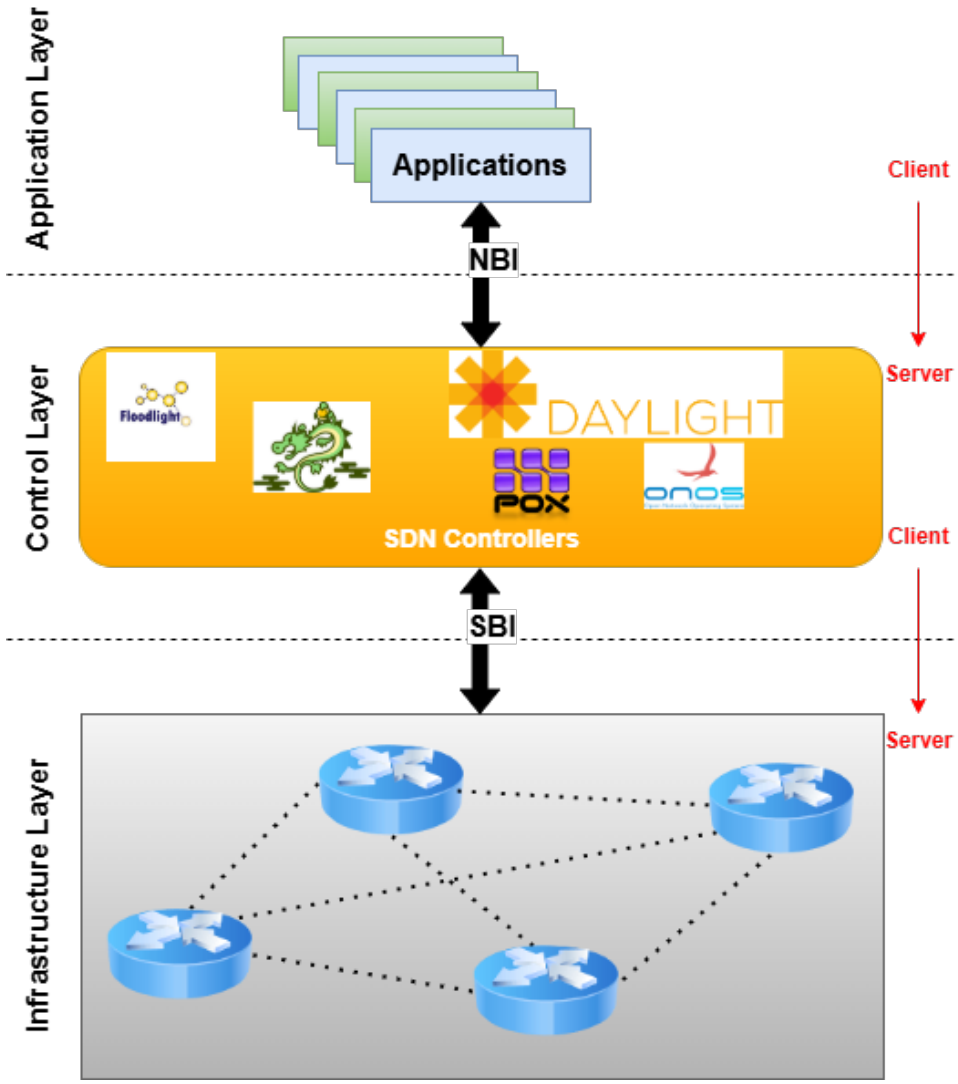


Figure 2.4: SDN architecture view

- **Control Plane (CP):** Often cloud-hosted, it defines and distributes routing and traffic policies to edge devices on the DP via SBIs. This centralization enhances flexibility, allowing real-time updates without site-specific manual configurations.
- **Data Plane (DP):** Comprised of edge devices (routers/appliances) at each site (e.g., branches, data centers), the DP forwards packets, enforces policies, and encrypts traffic over WAN links such as *MPLS*, broadband, and *LTE*.

SD-WAN addresses limitations of traditional WAN, which relied on costly, static MPLS links and manual configurations. Key features of SD-WAN include:

- *Dynamic Path Selection:* Routes traffic based on real-time network conditions (e.g., available bandwidth), optimizing application performance.
- *Application-Aware Traffic Steering:* Prioritizes traffic for critical applications (e.g., Voice over Internet Protocol (VoIP), video), ensuring low-latency paths.

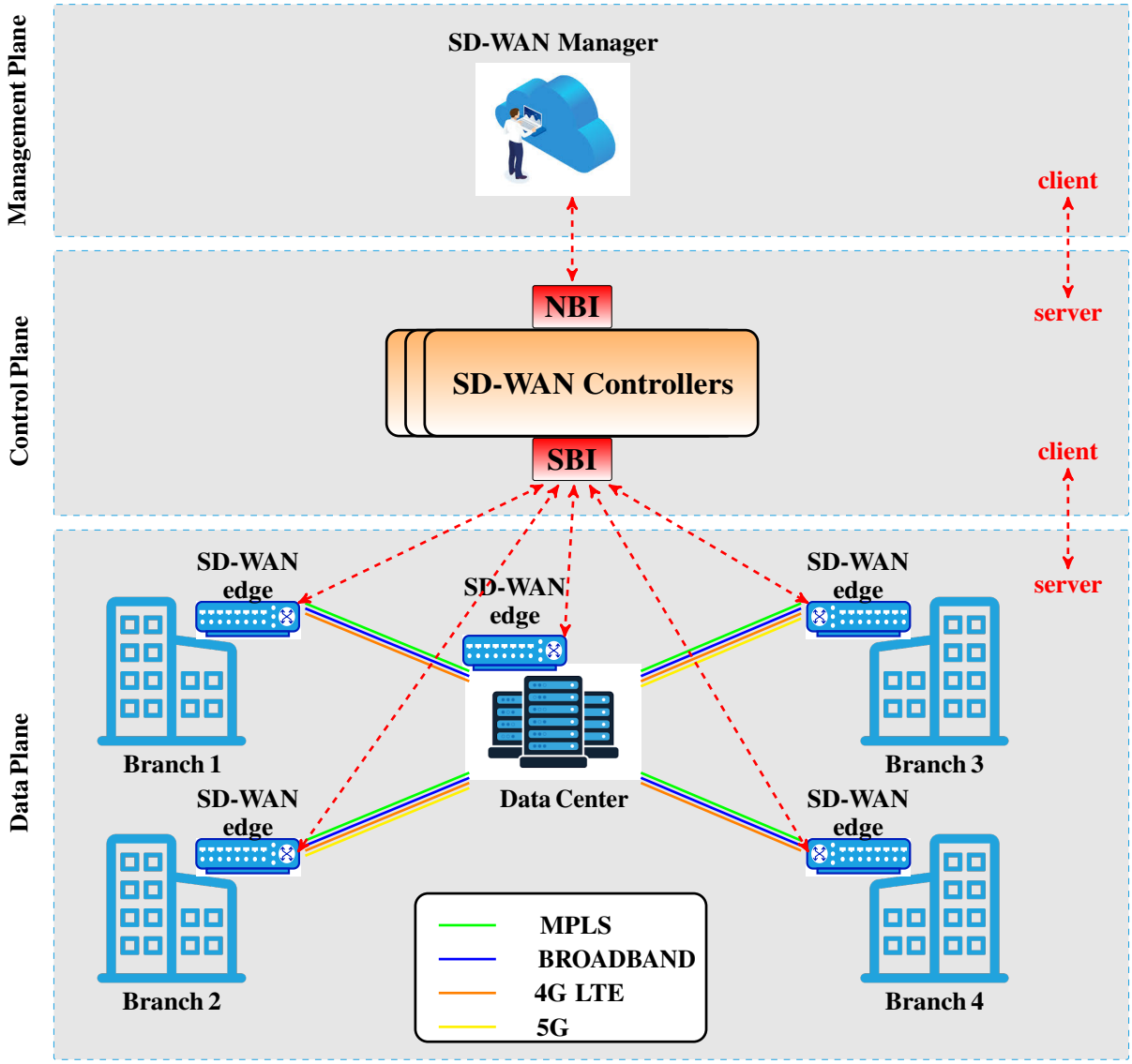


Figure 2.5: SD-WAN architecture view

- *Bandwidth Aggregation:* Combines multiple WAN links (MPLS, broadband, LTE) to increase efficiency and reduce reliance on MPLS circuits.
- *E2E Encryption:* Uses secure tunnels (e.g., IPsec) to protect traffic between sites over public networks.
- *Zero-Touch Provisioning:* Allows remote deployment of edge devices, reducing setup time and cost.

2.4.2 The Role of SDN in Overcoming TSN Limitations

Given the limitations of TSN, particularly in scalability, flexibility, and resource management, SDN offers a promising solution. While TSN ensures deterministic behavior, SDN provides the necessary dynamic control and flexibility that TSN lacks. SDN's ability to decouple CP from the DP allows for centralized, real-time management of network resources, enabling better optimization and adaptation to changing network conditions.

Specifically, SDN addresses the limitations of TSN in the following ways:

- **Scalability:** SDN centralizes network control, enabling real-time adjustments and optimizations across large-scale networks. By dynamically adjusting paths, SDN allows for more efficient handling of time-sensitive traffic as the network expands.
- **Dynamic Adaptability:** Unlike TSN's static configuration, SDN allows for dynamic reconfiguration of network resources based on real-time traffic conditions and network load. This adaptability ensures that time-sensitive traffic can be prioritized without being constrained by fixed schedules.
- **Simplified Configuration:** SDN provides a programmable, centralized CP that simplifies network configuration and management. This centralized approach reduces the complexity of configuring a large-scale TSN network and minimizes the chances of configuration errors.
- **Enhanced Resource Management:** SDN enables more efficient RA by offering detailed visibility into network traffic patterns. Through centralized control, SDN can allocate bandwidth, compute, and storage resources dynamically to optimize the performance of time-sensitive applications.
- **Interoperability:** SDN's flexibility allows it to integrate devices more easily. Through its abstraction layer, SDN can provide a consistent interface for managing different network devices, making it easier to incorporate and communicate with the network infrastructure elements.

By leveraging SDN's centralized control and dynamic capabilities, next-generation networks can address the scalability, adaptability, and resource management challenges faced by TSN. This makes SDN essential, particularly in environments where real-time adjustments and large-scale deployments are necessary.

2.5 SDN-based DP Programming

SDN-based DP programming provides the flexibility needed to meet the demands of modern networks, especially in the era of 5G&B. This section highlights key advancements related to this thesis in path selection, traffic AC, and congestion control, showcasing the state-of-the-art solutions and how SDN enables dynamic and efficient network management.

2.5.1 Path Selection Algorithms

Efficient path selection is fundamental to ensure optimal resource utilization, LB, and guaranteed low-latency communication in modern networks, particularly in the context of B5G and 6G systems with the arrival of time-sensitive applications (e.g., autonomous cars). Traditional approaches, such as those based on graph theory [31], including shortest path algorithms like Dijkstra [32] and Bellman-Ford [33], provide foundational methods for routing traffic. However, these static algorithms are designed for networks with relatively fixed conditions and fail to adapt to the dynamic and heterogeneous nature of traffic in modern networks. As a result, they struggle to meet the stringent latency and reliability requirements of critical 6G applications.

To address these limitations, some solutions have modeled the path selection problem as a multi-constraint optimization problem, solved using heuristic algorithms and exhaustive simulations [34]. While these methods improve upon traditional approaches by considering multiple constraints such as bandwidth and delay, they remain largely theoretical and lack adaptability to real-time traffic and network topology changes.

Recent advancements have introduced more dynamic solutions, particularly in the context of asynchronous Time Sensitive Networking (TSN). TSN-based solutions, as discussed in [35], provide hop-by-hop bounded latency guarantees by prioritizing time-sensitive flows. However, TSN frameworks do not inherently provide E2E latency guarantees, which are crucial for applications requiring deterministic communication across multi-hop networks.

SDN-based approaches, on the other hand, have filled the TSN gap by leveraging real-time TE and link monitoring [36] to dynamically adapt routing decisions. These approaches use graph-theoretic algorithms and/or monitoring systems to route traffic along paths with the smallest number of hops or the highest available bandwidth with leveraging the SDN programmability and flexibility but specifically the E2E view of the network. While these methods improve adaptability compared to traditional solutions, their configurations are often static, failing to account for real-time changes in traffic patterns, network topology, or load distributions. This lack of adaptability can result in suboptimal routing, especially in highly dynamic 6G networks where traffic demands and link states evolve rapidly.

For deterministic and low-latency communication in 6G, novel path selection algorithms must be devised to meet the delay constraints and bandwidth requirements of critical applications. These algorithms should leverage the enhanced DP capabilities introduced by 5G and extended in 6G. Specifically, they should utilize the concept of QFI, which replace traditional bearer-based flows with more granular traffic classifications that reflect the QoS requirements of different services. By associating QFIs with IEEE 802.1Q queues [37], it is possible to ensure that flows are routed according to their specific requirements, providing deterministic QoS guarantees throughout the backhaul, midhaul, and fronthaul segments of the TN. Achieving this would imply for path selection, choosing not only the E2E links but also the E2E queues for each flow.

However, existing solutions often overlook critical factors such as packet priorities, the number of hops, the dynamic nature of traffic patterns and network topology, and more importantly the state and load of individual queues. Indeed, they fail to incorporate mechanisms such as advanced QM and real-time queue monitoring, which are essential for managing congestion and ensuring consistent low-latency performance.

Indeed, path selection algorithms should integrate these overlooked elements into their computation, dynamically balancing packet priorities, queue loads, and network topology changes. By holistically considering these factors, such algorithms can achieve guaranteed E2E low-latency communication, even under high traffic loads. These advancements are particularly critical for enabling deterministic communication in 6G networks, supporting applications such as autonomous vehicles, holographic communications, and industrial automation, where any deviation from latency requirements can have severe consequences.

2.5.2 Traffic AC Mechanisms

Traffic AC ensures that only flows meeting QoS requirements are admitted into the network. Traditional AC methods, relying on static rules or heuristics, often fail to adapt to the dynamic and large-scale nature

of modern networks.

Recent research integrates two main components: the Network Model (NM), which predicts network KPIs like delay and throughput, and the Optimisation Algorithm (OA), which optimizes configurations based on NM inputs [38]. While TSN provides hop-by-hop QoS guarantees, achieving high E2E visibility is critical, especially under congestion or with multiple available paths. The accuracy and fidelity of NM models are therefore pivotal to ensure optimal performance.

Analytic models based on Queuing Theory (QT) [39] are lightweight but rely on unrealistic assumptions like Poisson traffic distribution. Packet-level simulators offer higher accuracy but are computationally intensive for large-scale networks. To balance precision, fidelity, and efficiency, state-of-the-art approaches incorporate Artificial Intelligence (AI)-driven methods. For instance, Deep Learning (D-Learn) models use architectures like Recurrent Neural Network (RNN) and Variational Autoencoder (VAE) [40, 41, 42], but these struggle with generalizing across network topologies due to their limited ability to handle graph-structured data.

This thesis addresses these challenges by leveraging GNN as the NM in an SDN environment. GNN models provide the adaptability, precision, and scalability needed to manage dynamic topologies and traffic patterns. By combining SDN programmability with the graph-based learning of GNNs, this approach delivers efficient and scalable AC with robust QoS guarantees.

2.5.3 Congestion Control Algorithms

Congestion control is essential for ensuring low latency and high throughput, especially in time-sensitive applications like autonomous vehicles, AR, and industrial automation. Traditional algorithms, such as TCP Cubic and TCP Reno [43], perform well in general-purpose networks but fail to meet the stringent latency requirements of 6G applications. These algorithms often rely on reactive mechanisms that respond to congestion only after it has occurred, leading to increased delay and packet loss, which are unacceptable for latency-sensitive traffic.

More advanced approaches, such as delay-based congestion control algorithms (e.g., TCP Vegas [44]), attempt to predict congestion by monitoring Round Trip Time (RTT). However, these methods are highly sensitive to network variability and often misinterpret natural RTT fluctuations as signs of congestion, leading to underutilization of available bandwidth. Similarly, Active Queue Management (AQM) schemes like Random Early Detection (RED) and Controlled Delay (CoDel) [45] aim to manage congestion at the router level by dropping or marking packets before queues overflow. Despite their improvements over passive methods, these approaches struggle with achieving fairness and maintaining low latency under heavy traffic conditions.

Emerging frameworks, such as L4S, offer a promising alternative by combining ECN with dual-queue mechanisms to prioritize time-sensitive traffic [46]. Unlike traditional congestion control schemes, L4S allows flows to react to congestion signals more quickly, reducing queuing delays while maintaining high throughput. However, L4S alone does not fully leverage the potential of network programmability for dynamic traffic management in 6G networks.

In this thesis, we adapt the L4S transport layer for integration within an SDN environment. SDN-based frameworks enhance L4S capabilities by enabling centralized control and real-time adjustments to flow priorities and transmission rates. By combining SDN's global network visibility with L4S's advanced congestion signaling, the proposed solution achieves superior performance in managing congestion,

particularly for time-sensitive applications. This integration ensures that latency requirements are met even under high traffic loads and dynamic network conditions.

2.6 Conclusion

In this chapter, we have presented a comprehensive analysis of the state-of-the-art technologies and methodologies underpinning the evolution of 5G networks and their progression towards 6G. We began by exploring the foundational components of 5G, including the DP, TN, and QoS mechanisms such as the QFI, which collectively play a crucial role in meeting the stringent requirements of time-sensitive applications.

Next, we delved into TSN, examining its definition, challenges, and implications in the context of URLLC. The chapter then transitioned to a discussion of SDN, emphasizing its capabilities and its role in enabling SD-WAN and overcoming TSN limitations.

Finally, we surveyed advanced SDN-based DP programming approaches, including path selection algorithms, traffic AC mechanisms, and congestion control strategies. These methodologies demonstrate the potential to optimize RA and guarantee low-latency communication in dynamic network environments.

The insights gathered in this chapter not only highlight existing challenges but also establish a robust foundation for the development of novel solutions in the subsequent chapters. By addressing research gaps in latency optimization, DP programmability, and traffic management, this thesis aims to contribute to the realization of deterministic and efficient B5G networks capable of supporting next-generation applications.

Chapter 3

SDN-based Path Selection for Time-Sensitive Applications Using Dynamic Queuing

3.1 Introduction

Although 5G is still under deployment, the research community is already establishing the requirements of 6G systems. Ultra low latency is one of the main requirements that should be natively supported, which is not been well addressed in 5G. The expected services and applications [47] in 6G such as holograms and tactile Internet need a highly synchronized communication from the RAN to the network service to guarantee ultra-low E2E latency.

5G and beyond connectivity involves different network segments, i.e., RAN, CN, and TN. The TN includes not only the backhaul links connecting the CN to the RAN but also the fronthaul and midhaul, which result from the functional splits of the RAN that appeared in 5G [48]. Therefore, to guarantee a very low E2E latency, the three parts need to be optimized, i.e., RAN, CN, and TN. For the RAN, works need to be done to reduce the slot duration further and improve the HARQ mechanisms [5]. The reduction of latency at the CN has already been addressed in 5G, where network slicing is used. However, ensuring very low latency at the TN level is still challenging.

Most of the existing approaches to guarantee low latency in TNs are relying on TSN, which was devised by the IEEE. TSN specifications define two standards: The first one is based on synchronous communication (IEEE Std 802.1AS and IEEE 802.1Qbv) [20]. It uses a precise and common time reference shared among all the TSN devices (i.e., routers). It allows the scheduling of the traffic transmission in a deterministic way to maintain a lower bound for the latency. However, it needs a network-wide coordinated time, which hinders the scalability of the network. In addition, using reserved time slots for each flow may lead to poor utilization of network resources. The second one is based on asynchronous communications (IEEE 802.1Qcr) [21]. In contrast to the synchronised version, no common time reference is needed and shared between devices. Each device implements advanced scheduling on a two-level queuing hierarchy to shape the traffic and achieve a bounded latency at each hop.

Although the TSN is an interesting approach, it still has weaknesses in fully sustaining the low latency expectation of 5G&B services. On one hand, TSN provides hop-by-hop bounded latency and not E2E latency guarantee; the latter is highly needed when multiple paths are available. Indeed, high E2E visibility is crucial when the links get congested, and changing the path is necessary to continue ensuring

low latency. On the other hand, TSN adopts a static configuration of the different queues of the switch, which has limitations when several URLLC traffics are competing for the same high-priority queue. Indeed, 5G specifications define more than 10 services as delay critical, with different levels of criticality. For instance, Electricity Distribution is the highest delay critical service (5ms E2E latency), while V2X message and real-time gaming can be considered less critical (50ms E2E latency). Accordingly, the network should be in the capacity to protect the highest DC services, even in the presence of others. This can be achieved only by using an agile QM system, which is impossible in TSN.

To overcome the above-cited limitations, in this chapter, we embrace an SDN-based approach. SDN provides all the flexibility and agility to guarantee E2E low latency even when several DC services are running in parallel. SDN provides the necessary functions, known as Create, Read, Update, and Delete (CRUD), to program and monitor the network elements. It includes dynamic QM, to ensure a real-time E2E control of the different flows running in the network. The proposed solution, namely (SDLL), relies on the new 5G DP specifications. It uses the QFI [6] information to map the 5G services to TN traffic classes (and queue) to fulfill the needed QoS. Furthermore, SDLL adapts to the network load by dynamically updating queues (add or remove) which permits segregating among URLLC traffics and hence protecting the highest priority traffics (strongest DC services). The contributions of this work are manifolds:

- We introduce a SDN-based novel framework for QoS provisioning in 5G&B, featuring a low latency guarantee;
- We propose a TE algorithm that ensures efficient resource management while providing a bounded E2E latency for URLLC services even in a loaded network;
- We present a dynamic QM algorithm to enforce priority among URLLC services.

The rest of the chapter is organized as follows: Section 3.2 introduces the state-of-the-art solutions. Our solution is shown in Section 3.3 and evaluated in Section 3.4. We conclude the chapter in Section 3.5.

3.2 Related Work

In this section, we review different works that have been done in correlation with QoS provisioning and latency guarantee in communication networks.

In [36], authors developed an SDN framework named Software-Defined Queuing (SDQ) for QoS provisioning by selecting the right path and queue according to the network bandwidth management. Indeed, they route the traffic through the path with the smallest available bandwidth weight. However, their configuration was static and not updated compared to our SDLL solution.

[49] proposed a queuing model to analyze an OpenFlow–based SDN network. They also considered a classification of the incoming packets. But, the topology was fixed and included only one switch, which reduced the number of available paths and hence the complexity of the conducted analysis.

[34] delivered a framework that synthesizes paths through the network. To guarantee that flows meet both the bandwidth and E2E timing requirements, they solved a multi-constraint optimization problem using a heuristic algorithm and exhaustive emulations and experiments on hardware switches to demonstrate the techniques and feasibility of their approach based on SDN. Despite that, their solution is theoretical and is not dynamic in term of queue management comparing to ours.

In [50], evaluations were done on the performance of LEARNET through simulation in a 5G asynchronous deterministic backhaul network where incoming flows have characteristics similar to the four critical 5QIs defined in 3GPP Technical Specification [51]. This solution is based on asynchronous TSN which provides hop-by-hop bounded latency and not E2E latency guarantee (the latter is highly needed when multiple paths are available).

All the above solutions do not consider the QM part and their monitoring. Instead, our solution considers several inputs (i.e., packets priority, number of hops, number of queues, and their loads) in the path computation to route traffic, aiming at achieving a guaranteed E2E low latency for critical applications even in a loaded network.

3.3 Software-Defined Low Latency (SDLL)

3.3.1 SDLL Concept and Interaction with 5G DP

SDLL is a SDN-based solution that controls, monitors, and manages a 5G TN (backhaul) composed of a number of programmable switches. SDLL uses the CRUD functions to ensure agility in managing 5G services traffic to guarantee required QoS, and particularly low-latency for DC services (known as URLLC services). Figure 3.1 shows the positioning of SDLL in a 5G DP. To recall, to create a communication link with outside and start sending and receiving traffics, a UE needs to establish a session that creates dedicated bi-directional tunnels or bearers. At the core network side, the bearer encapsulates UE's packets IP in GTP tunnels between the gNB and UPF. The GTP header includes information on the user traffic, like the QoS that the gNB and UPF should apply to the tunnel. This information in 5G corresponds to the 5G QFI field. 5QI is a mechanism whereby packets are classified into various QoS classes. Thus, the QoS can be tailored to specific requirements. Each QoS class has specific QoS characteristics (such as packet delay and packet loss). There are approximately two dozen standard 5QI values, grouped into two types of resources: GBR and non-GBR. QoS characteristics include RT, PL (lower number implies higher priority), PDB, PER, MDBV, etc. The QFI value is assigned by the 5G CN according to the UE subscription and the service to run. Therefore, QFI is a critical value that needs to be taken into account when carrying the tunnel traffic over the 5G TN or backhaul.

As depicted in the Figure 3.1, the SDLL is composed of the SDN controller and the SDN applications that run the TE and QM modules. The SDN CP operates independently of the 5G network infrastructure. The only interaction between SDLL and 5G data plane is the border switches (i.e., the routers connecting the gNB and UPFs to the 5G TN) that need to map the QFI to a DSCP value. In SDLL, we use the mapping proposed in [17], which specifies a set of 3GPP QCI and 5QI to DSCP mappings to reconcile the marking recommendations offered by the 3GPP with the recommendations offered by the IETF. This maintains a consistent QoS treatment between 5G networks and the Internet. It is worth noting that border switches need to parse the GTP header to extract the QFI and modify the IP header (layer 3) with the corresponding DSCP value to route the packet through the 5G TN according to SDLL control. The mapping process is done in the two directions of the traffic, i.e., UL and DL.

A more focused representation of SDLL is shown in Figure 3.2. Here, we detail the different elements interacting SDLL with 5G TN. The solution is represented within the three layers of the SDN. The SDLL application is located on the first layer. It is installed on top of a SDN controller. In this work, it corresponds to a Java-made ONOS controller. The SDLL applications contain two parts, namely *Queue Management (QM)* and *Traffic Engineering (TE)*. The former utilizes ONOS REST API and the Open vSwitch Database Management (OVSDB) Protocol to manage queues at the switches and get their statis-

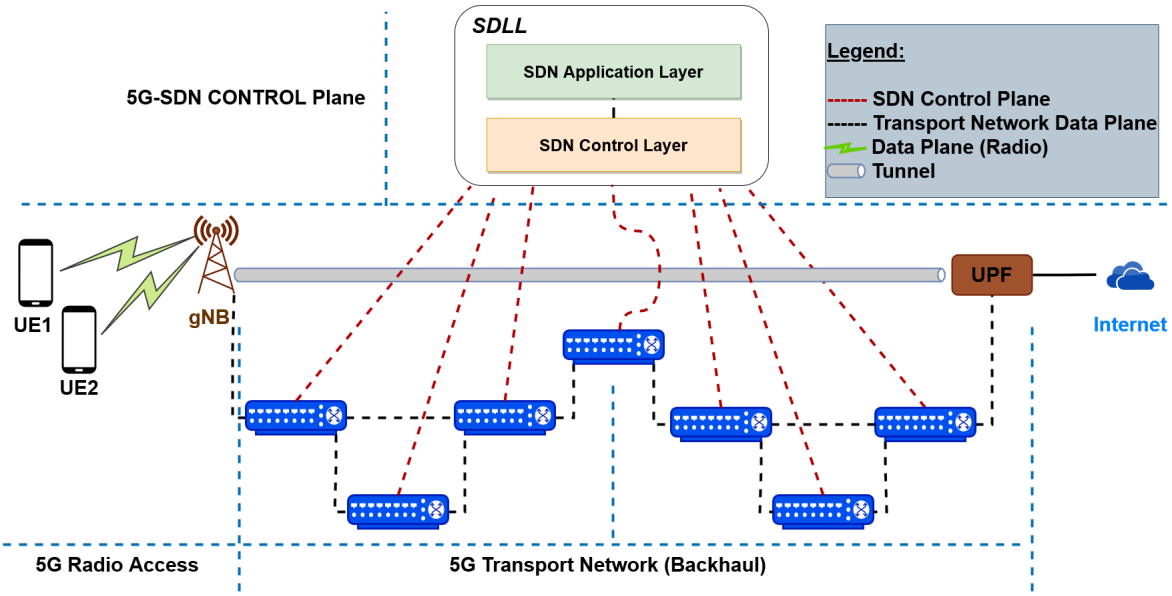


Figure 3.1: The 5G SDNized network infrastructure

tics, while the former decides on the flow rules to communicate to switches via the OpenFlow protocol. Accordingly, each packet entering a switch is routed by designating the right queues along a chosen path. It is worth mentioning that our solution is applicable to Open vSwitch (OVS) but can be extended to any other technology (i.e., switch) with integrated Data Base (DB) and NBI that enable the QM and the flow rules installation.

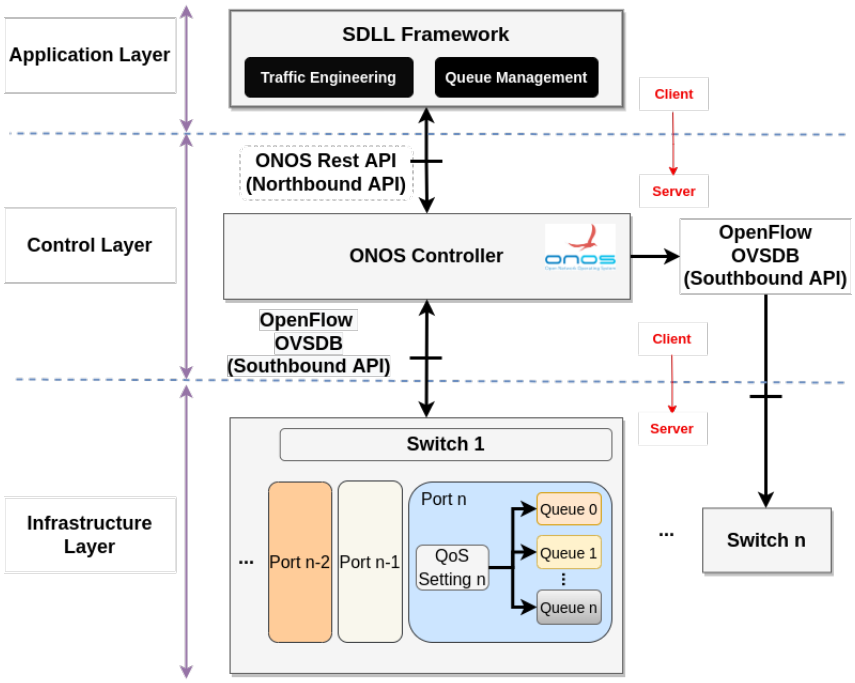


Figure 3.2: The architecture view of the SDLL framework

A flow rule ξ is composed of: *Match set* (μ) to identify a flow; *Action set* (τ) to define the actions executed on each packet of the flow; and *Priority* (ρ) that is used to relatively order rules in the forwarding switch. In our solution the Match set includes the DSCP value of the packets, the Action set is to route the packets of each flow through a set of port numbers and queue IDs, while the priority is the same for all the flow rules. Please refer to the equations 3.1, 3.2, 3.3, and 3.4.

$$\xi = \{[\mu], [\tau], \rho\} \tag{3.1}$$
$$\mu = \{0, 1, 2, ..., 56\} \tag{3.2}$$
$$\tau = \{output(port_{number}, queue_{id})\} \tag{3.3}$$
$$\rho = \{0, 1, 2, ..., 65535\} \tag{3.4}$$

3.3.2 SDLL Details

In SDLL, for every flow newly coming into the network, a PACKET-IN message is generated by the respective access node (i.e., border switches of the SDN forwarding plane or backhaul). As a response, the process shown in Algorithm 3.1 is triggered. In what follows, we describe the two main components of the SDLL framework along with the workflow they use.

3.3.2.1 Traffic Engineering (TE)

This module is responsible for choosing an E2E path for every incoming packet and hence for the 5G flow. It operates as follows: **(1)** Check if there is already an allocated path (i.e., flow rule) for the incoming packet. **(2)** If not, list all the available paths between the source and the target destination. **(3)** Choose the paths according to the packet priority: **-** Shortest path (the path with the minimum number of hops π) for High Priority (HP) packet to satisfy its requirements; **-** The path with the minimum queue number (the sum of the queues number in the ports of a given path) (see Eq. 3.5) for Medium Priority (MP) packet to be sure that the average output rate of the queues in that path is maximum so that it reduce the E2E transmission delay; **-** The path with the minimum average queue load (the sum of the average queues loads in the ports of a given path) (see Eq. 3.7) for Low Priority (LP) packets so that the E2E queuing delay be minimal. It is worth mentioning that border switches need to parse the GTP header to extract the QFI and modify the IP header (layer 3) with the corresponding DSCP value to route the packet through the 5G TN.

$$\psi = min\left\{\sum_{i=1}^{\pi-1} \varrho_i\right\}$$
$$\text{where} \begin{cases} \psi \text{ is the minimum queue number in the list of paths,} \\ \pi \text{ is the number of hops between source and destination,} \\ \varrho_i \text{ is the number of queues in a port } i \end{cases} \tag{3.5}$$

$$\gamma = min\left\{\frac{\sigma_{j,i} \times 100}{\delta}\right\}$$
$$\text{where} \begin{cases} \gamma \text{ is the shortest path hop by hop minimum queue load,} \\ \sigma_{j,i} \text{ is the load of queue } j \text{ in port } i, \\ \delta \text{ is the queue size (100 packet for each)} \end{cases} \tag{3.6}$$

$$\varphi = min\left\{\sum_{i=1}^{\pi-1} \sum_{j=1}^{\varrho_i} \left(\frac{\sigma_{j,i} \times 100}{\delta}\right)\right\}$$
$$\text{where} \begin{cases} \varphi \text{ is the minimum average queue load in all the paths,} \end{cases} \tag{3.7}$$

3.3.2.2 Queue Management (QM)

For each flow, this module enforces the QoS settings received from the SDLL framework on the forwarding devices along the communication path selected for the flow. This module implements the different functions: CRUD queues, specifies the scheduling algorithms, and sets the parameters for these queues. The detailed workflow is described in Algorithm 3.1.

3.3.2.3 SDLL Workflow

The SDLL framework reduces the load imbalances in the network. Also, the chosen path for the specific traffic priority μ ($\{0,1,2,3,4,8,10,12,14\}$ for LP, $\{16,18,20,22,24,26,28,30\}$ for MP and $\{32,34,36,38\}$ for HP) is used as a reference for the QM module to install the QoS settings and the required queues (lines **14**, **22** and **24**). The TE module translates the chosen path into flow rules, which include data such as the output port number and queue ID (line **25**). Furthermore, a threshold is specified for the HP packets by γ with a predicate of (80 %, 50 % and 30 %) of δ. Hence, if γ is exceeded (line **15**), the LP packets are diverted to another φ path. Also, a new queue with higher priority for HP$^+$ packets (μ in $\{40,44,46,48,56\}$) is created in the switch port where the HP packet is located and the other queues in that port will be updated by reducing their maximum rate (lines **17**, **18**). Finally, the flow rules are pushed to the intermediate nodes involved in the communication (lines **25** and **26**).

Algorithm 3.1 SDLL workflow

1: *Inputs:* {Paths} : {Hosts} : {Links} : {Flow Rules} : {Queues statistics}
2: *Outputs:* {New Flow Rule with best path and queue}
3: The controller receives a packet with a μ (packet-in message)
4: if it exists a path with a flow rule for this packet with the DSCP μ **then**:
5: Send the packet via this path and queues
6: else
7: Find all the potential paths to the destination
8: if path list is empty **then**:
9: The destination is unreachable
10: else
11: Sort the list by (π, ψ, φ)
12: Check the packet priority μ
13: if $\mu \geq 32$ **then**:
14: Chose the path with the minimum π
15: if $\gamma > Threshold$ **then**:
16: Diverts ($\mu < 16$) packets to other φ paths
17: Create a new queue and update the others maximum rate
18: Split the ($\mu \geq 32$) packets and diverts the ($\mu \geq 40$) packets to the new queue
19: else
20: Goto 25
21: elseif ($\mu < 32$) and ($\mu \geq 16$):
22: Chose the path with ψ
23: elseif ($\mu < 16$) and ($\mu \geq 0$):
24: Chose the path with φ
25: Install the flow rules on the devices for the path
26: Goto 5

3.4 Performance Evaluation

3.4.1 Technical Details

In order to evaluate the performance of SDLL framework, we consider a network topology composed of (8) virtual switches and (24) hosts, including one destination host (i.e., H24). Figure 3.3 depicts that topology. We use ONOS as an SDN controller, OVS for the SDN switches and Mininet for the network topology. Table 3.1 includes all technical details of the envisioned setup. In the performance evaluation, a maximum of $1,2 \times 10^6$ packet (5×10^4 packet for each host) have been generated for each of the following two scenarios:

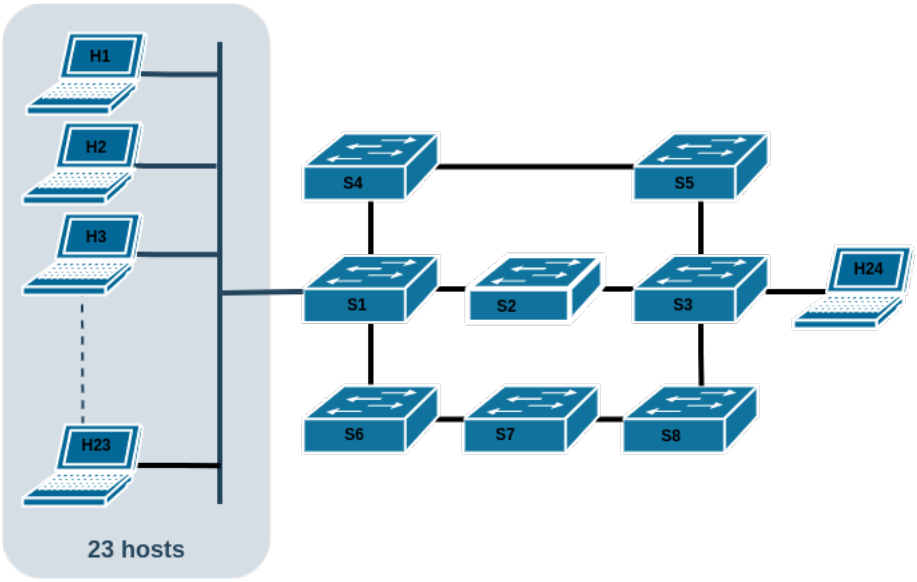


Figure 3.3: Network topology used for the setup

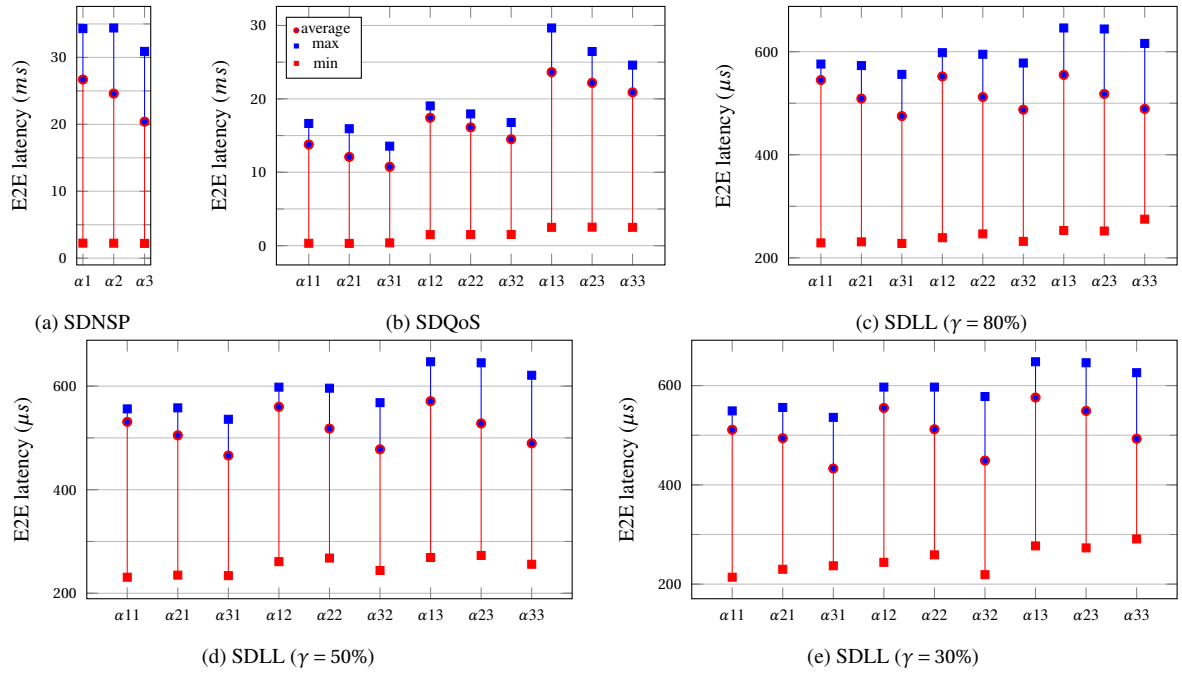
Scenario 1: The 24 hosts are grouped into three sets of 8 hosts. Each group generates the same type of traffic (α_{i1} is HP, α_{i2} is MP, and α_{i3} is LP). In each group, we choose a host as a reference, while the others are seen as interferences (i.e., 3 hosts for traffic interests and 21 hosts for traffic interferences). We varied the number of packets generated by each host ($\alpha_{1j} = 5 \times 10^4$, $\alpha_{2j} = 2.5 \times 10^4$, and $\alpha_{3j} = 5 \times 10^3$; where j represents the type of the traffic as quoted above), and γ from (80%, 50%, and 30%). Comparison is made with the default and static configurations SDN Shortest Path (SDNSP) and Software-Defined QoS (SDQoS). SDNSP is configured with one queue on each port of the network switches while SDQoS includes three queues on each. Also, there is no QM on the two solutions.

Scenario 2: In this scenario, we focus only on HP traffic. Therefore, 12 hosts generate HP packets and the 12 others HP^+ with (5×10^4 packet for each). Only two hosts are considered for traffic interest. The others are seen as interference. We also varied γ from (80%, 50%, and 30%) for HP packets. We compared the results also with SDNSP and SDQoS.

We have put the network under realistic conditions aiming at stressing the switches (especially Switch 1, which is the entry point of the entire hosts) with a large number of packets and ensuring none of the queues are empty.

Table 3.1: Technical details of the setup

Operating System	Ubuntu 20.04.4 LTS
Software and protocols	ONOS 2.7.0, Mininet 2.2.2, OVS 2.13.5, Ping iputils s20190709, OpenFlow 1.6
Topology	Linear
Packets generated	Maximum of $1.2 * 10^6$ packet divided between 24 hosts (5×10^4 packet each)
Generation rate	10,000 packet/second
Packet size	65507 bytes
Number of iterations	100
Priority queues	High, medium and low
Bandwidth	50 Gb/s for each link
Queuing mechanism	Hierarchical Token Bucket (HTB)



(a) SDNSP (b) SDQoS (c) SDLL ($\gamma = 80\%$)

(d) SDLL ($\gamma = 50\%$) (e) SDLL ($\gamma = 30\%$)

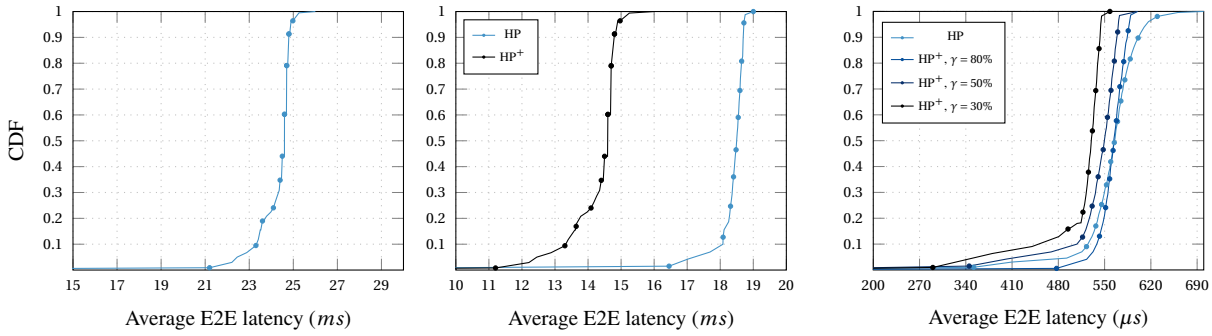
Figure 3.4: Scenario 1 E2E latency results experienced by (HP, MP, LP) packets ($\alpha_{.j}$) and handled by SDNSP and SDQoS static solutions, and SDLL framework with different threshold (γ) and number of packets ($\alpha_{i.}$)

3.4.2 Results

In the performance evaluation, we consider the network latency as the main comparison metric.

Figure 3.4 depicts the obtained results in scenario 1 (in ms and μs) regarding γ, and α_{ij} packets type and number. Indeed, figure 3.4a, 3.4b and (3.4c, 3.4d, 3.4e) shows the (minimum, maximum, and median) E2E latency experienced on scenario 1 by SDNSP, SDQoS and (SDLL, respectively) for HP, MP, and LP (α_{i1} is HP, α_{i2} is MP, and α_{i3} is LP) packets with different γ relative to HP queue load (80 %, 50 % and 30 %) and α_{ij} ($\alpha_{1j} = 5 \times 10^4$, $\alpha_{2j} = 2.5 \times 10^4$,and $\alpha_{3j} = 5 \times 10^3$). It is worth mentioning that SDNSP and SDQoS are traffic type agnostic solutions. We make three main observations here.

- *Low latency is guaranteed by SDLL*: As expected in these figures, the latency is bounded under 649.9 μs for HP, MP, and LP traffic. As mentioned above, for SDNSP and SDQoS, packets of all traffic types traverse the shortest path (i.e., from switch 1 to switch 2 to switch 3 as shown in Figure 3.3) until it gets saturated, so the packets split to other longer paths (such as from switch 1 to switch 4 to switch 5 to switch 3) which increase the latency. The value is less than 34.4 ms, and this represents a factor of almost ×53 in comparison with SDLL value. We also notice a reduction in latency on this experience when $\alpha_{i.}$ is getting low which is related to the underload of the system (queues and links). Furthermore, we can see a difference in the latency between each type of traffic when using SDQoS compared to SDNSP, and this is related to the prioritization of HP and MP over LP packet by setting the HP, MP and LP queues maximum rate to (50%, 30%, and 20%) of the total bandwidth respectively.
- *(Almost) similar results were seen for each priority*: We observe that the average latency of the three types of traffic (i.e., HP, MP and LP) is comparable with a small tendency to prioritize the HP packets over the MP packets and the MP over the LP packets. Indeed, exploiting the same high bandwidth offered through the different switches, hence providing several paths, the SDLL could handle incoming packets at each switch faster and without significant queuing delays. It can also be noticed that when the network is loaded, the latency experienced by packets of the same traffic category does not fluctuate hugely (0.3ms with SDLL and 32.16ms with SDNSP) and remains stable below a target value, which is in line with the core spirit of deterministic networking.
- *Very low latency is guaranteed by reducing the threshold*: Indeed we can see from the pictures that the latency of HP packets is correlated with the threshold value related to hop-by-hop queue load. Thus, a low threshold could be used for strict priority traffic to make a ceil on the queues load. Hence this will reduce the queuing delay and guarantee very low latency for that type of traffic.



(a) SDNSP (b) SDNQoS (c) SDLL

Figure 3.5: Scenario 2 E2E latency results experienced by HP and HP$^+$ packets when using SDNSP, SDQoS and SDLL with $\gamma = (80\%, 50\%, 30\%)$ for HP packets

Figure 3.5 depicts the Cumulative Distribution Function (CDF) of latency (in ms and μs) obtained for HP and HP$^+$ packets as addressed in scenario 2. It shows as in scenario 1, a bounded latency around 650 μs when SDLL is in use, 19 ms for SDQoS and 25 ms for SDNSP with more advantage for HP$^+$ traffic over HP in SDLL related to the threshold value. The results reveal that our solution is well suited for different numbers and types of traffic. Indeed, in SDLL, the entire HP and HP$^+$ traffic are sent over the shortest path, which is from switch 1 to switch 2 to switch 3. When the queues in this path are overloaded, new queues are created on this path, and others are updated as explained previously in the Algorithm 3.1.

Figure 3.6 shows the latency variation during a period of time when using SDLL. We considered a sample of 200 packets sent over 50,000 HP and HP^+ packets for each host during a period of 1.3 seconds. It shows the adaptability of the solution regarding the variation of the latency, which increases when the HP queue is filling up, and when γ is exceeded ($time = 0.24s$), a new queue is created with a high maximum rate, while other queues are updated to reduce the average E2E latency of HP^+ packets.

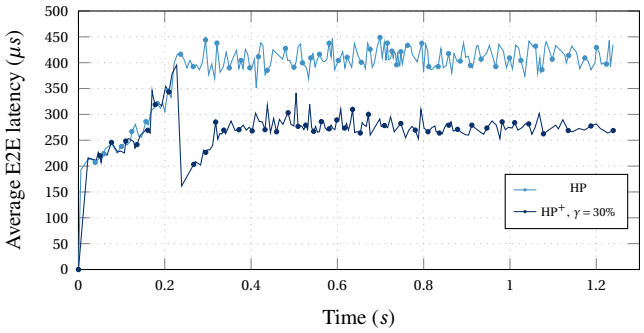


Figure 3.6: SDLL latency variation during a period of time

These results can confirm the efficiency of the TE module for guaranteeing a very low E2E latency bound for different types of traffic. They also show the power and the adaptability of the QM module for the same purpose, which is the main contribution of SDLL.

3.5 Conclusion

In this chapter, we introduced SDLL solution for QoS provisioning and latency guaranty in 5G&B. The main purpose was to explore all the E2E paths in the 5G TN and choose the optimal ones in terms of the number of hops, number of queues, and queues load in a dynamic and seamless way. It allows TE and QM in real time to guarantee low latency according to traffic load and priority. Results show that our solution outperforms two reference solutions. Our future focus will be working on a theoretical way to set the best thresholds and implementing this solution on an open-source hardware target (NetFPGA) [52] to demonstrate its efficiency in a real and scaled environment.

Chapter 4

SDN-based Admission Control (AC) for Time-Sensitive Applications Leveraging GNNs

4.1 Introduction

The emerging B5G market is expected to bring a variety of services, allowing to meet the requirements of a highly mobile and connected society. The key enabler for 5G architecture is the support of different use cases and applications such as e-health, autonomous cars, and metaverse [47]. Their coexistence will depend on B5G networks' ability to serve these CoTs, having *different needs* in terms of QoS that include latency, bandwidth, reliability, and availability. SDN trends may accommodate these needs and provide the missing capabilities.

Being deployed in research laboratories, and industry, SDN is emerging as a promising technology to manage large-scale networks since it has a global view of the network. However, the SDN concept itself is stateless as it does not implement the feedback control command, so it is harder to ensure consistent and predictable QoS, such as maintaining low latency and avoiding traffic congestion. Bring up a stateful SDN involves implementing mechanisms to maintain the current state and context of network devices and flows, allowing for *more granular control and management* of network resources.

Enabling statefulness in SDN is not a new topic. Indeed, the research community has introduced various approaches that combine two main elements: the NM, and the OA [38]. The former predicts the network KPIs, such as delay and throughput; it features approaches such as Network Prediction (NP) [53]. The latter explores different configurations until it meets the optimization goals defined in the NM, which include solutions based on TSN standards [35] such as AC [54]. Although TSN is an interesting approach, it provides hop-by-hop QoS guarantee; high E2E visibility is crucial when links get congested and multiple paths are available. Yet, the NM feeds the OA with inputs in order to seek the optimal configurations. Therefore, the OA accuracy highly depends on the NM *correctness and fidelity*.

Providing accurate and low-cost NMs has already concentrated a lot of consideration in the research community. In fact, analytical models based on QT [39] are largely developed. Even though these models are deterministic solutions, hence low resource consumption, they assume some non-realistic network properties and use probabilistic rules such as Poisson distribution, which are unsuitable in real networks. Another example is packet-level network simulators, which bring accuracy and fidelity. Unfortunately,

such models are time-consuming and need high computational resources in large-scale networks.

To provide a balance between precision, fidelity, and swiftness, D-Learn [40] models are proposed. Existing solutions such as [41], [42] use the Neural Network (NN) architectures like RNNs or VAEs. However, computer networks are fundamentally represented as graphs, and NN cannot learn graph-structured information and generalize over other topologies or routing configurations. Moreover, their accuracy is limited.

To overcome these limitations, we unveil in this chapter a solution that combines GNN and AC, named *GNN-AC*. It leverages the RouteNet-F model [55, 56] as a real-time latency prediction framework for each path, that is to achieve LB and optimize the AC decisions. The solution is inspired by the SDN-based AC that provides flexibility and agility to guarantee QoS of several CoTs running in competition mode. GNN-AC relies on the new 5G DP specifications. It uses the QFI [6] to map the 5G services to TN CoTs. The solution regulates the traffic and guarantees QoS for the accepted packets by predicting for each traffic flow if its QoS can be satisfied, particularly traffic with a low latency requirement. In addition to these contributions, our solution is resource-aware and delay-aware, thanks to the TE module and path listing algorithm that we developed.

We have evaluated the GNN-AC accuracy with a dataset including various topology flavours (i.e., scales) generated by the packet-level simulator OMNeT++ [57]. We compared our solution with the ONOS SDN controller' default one (i.e., SDNSP).

The rest of the chapter is organized as follows: Section 4.2 provides background on GNNs, RouteNet-F, and reviews related state of the art. Our solution is presented in Section 4.3 and evaluated in Section 4.4. Section 4.5 concludes the chapter.

4.2 Background & Related Works

To better understand our contribution, we introduce in what follows the concepts of GNNs, and RouteNet-F model.

GNNs have shown outstanding applications in communication networks [58]. Indeed, Networking systems comprise many components represented as a graph (e.g., topology or routing). GNNs represents a new generation of data-driven models that can accurately learn and reproduce the complex behaviors behind real-world networks. As a result, these models can be applied to a wide variety of networking use cases, such as planning. The main advantage of GNNs over traditional NNs lies in their generalization capabilities when applied to other network topologies and configurations unseen during training.

RouteNet-F is a custom GNN model for network performance prediction. It has a network state description (sample), defined by: a network topology, a routing scheme, a queuing configuration, and a set of traffic flows characterized by some parameters as input, and flow-level performance predictions as output, such as delay. This model implements a custom three-stage message-passing algorithm that represents key elements for the NM mentioned above. RouteNet-F supports a wide variety of features present in real-world networks, such as complex traffic models. Figure 4.2 includes a black-box representation of this model communicating with the AC module on the top of an SDN controller. RouteNet-F is based on two main design principles: **(i)** finding a good representation of the network components and **(ii)** exploiting scale-independent features of networks to encompass larger networks unseen during training.

In what follows, we review different works in correlation with latency prediction, AC, and QoS guarantee. Indeed, in [59], authors introduced RouteNet, a NP model based on GNN that estimates the per-source/destination packet delay distribution and loss without including queuing configuration as input. They simulated some use cases where they leverage the KPI predictions of the model to achieve efficient routing optimization and network planning. However, their solutions did not include the LB mechanism compared to GNN-AC.

[60], addressed the challenge of accurately predicting E2E delay in SDN to improve network performance and user experience. The authors propose a GNN-based model, Spatial-Temporal Graph Convolutional Network (STGCN), which captures spatial and temporal dependencies of traffic data, outperforming traditional ML techniques.

[61] proposed a GNN architecture for Service Function Chaining (SFC). The proposed model consists of an encoder and a decoder, where the first finds the network topology representation, and the latter estimates probabilities of neighborhood nodes to process a Virtual Network Function (VNF). In the experiments, the architecture outperformed the performances of Deep Neural Network (DNN) based model. Despite being different from GNN-AC, their solution satisfies QoS requirements as ours.

All the above proposals did not consider LB part, unlike our solution that takes into account several points (i.e., multiple topology flavours test, guaranteeing low E2E latency, LB, and preventing congestion in order to maintain the QoS.

4.3 GNN-AC Solution

In this section, we describe the high-level view of our proposal how it fits within the 5G architecture, its design, and workflow.

4.3.1 Interaction with 5G

It is generally agreed that B5G architectures will rely on the 5G specifications. In what follows, we summarize the 5G QoS in correlation with GNN-AC and how the latter can be used within the DP.

To recall, in 5G, UE traffic has to pass through dedicated bi-directional tunnels or bearers to interact with the outside. These tunnels are created by the SMF PDU session establishment request. The bearer encapsulates UE' IP packets in GTP tunnels between the gNB and the UPF. The GTP header contains information on the user traffic, such as QoS that the gNB and the UPF should apply to the tunnel. This GTP header information corresponds to the 5G QFI field. QoS is tailored to specific requirements using 5QI that classifies packets into different CoTs, each with specific QoS characteristics that include RT, PL, and PDB. There are approximately two dozen standard 5QI values grouped into two types of resources: GBR and non-GBR. The QFI value is assigned by the 5G CN based on the UE subscription and the service to run, making it critical when carrying tunnel' traffic over 5G TN or backhaul.

Figure 4.1 shows the positioning of GNN-AC on the 5G architecture. The GNN-AC is on top of an SDN controller. The SDN CP is independent of the 5G network infrastructure. The only interaction between GNN-AC and 5G DP is the border switches (i.e., routers connecting the gNB and UPFs to the 5G TN) that needs to map the QFI to IP networks QoS ID values known as *DSCP*. DSCP specifies a mechanism for classifying and managing network traffic and providing QoS on IP networks. DSCP uses 6 bits in the IP header for packet classification purpose. IETF has proposed a mapping of the 3GPP QCI and the 5QI

to the DSCP, which aligns their marking recommendations, as stated in [17]. This mapping is used by the border switches when parsing the GTP header and extracting the QFI to modify the IP header with the corresponding DSCP value. The mapping process is done for both UL and DL directions.

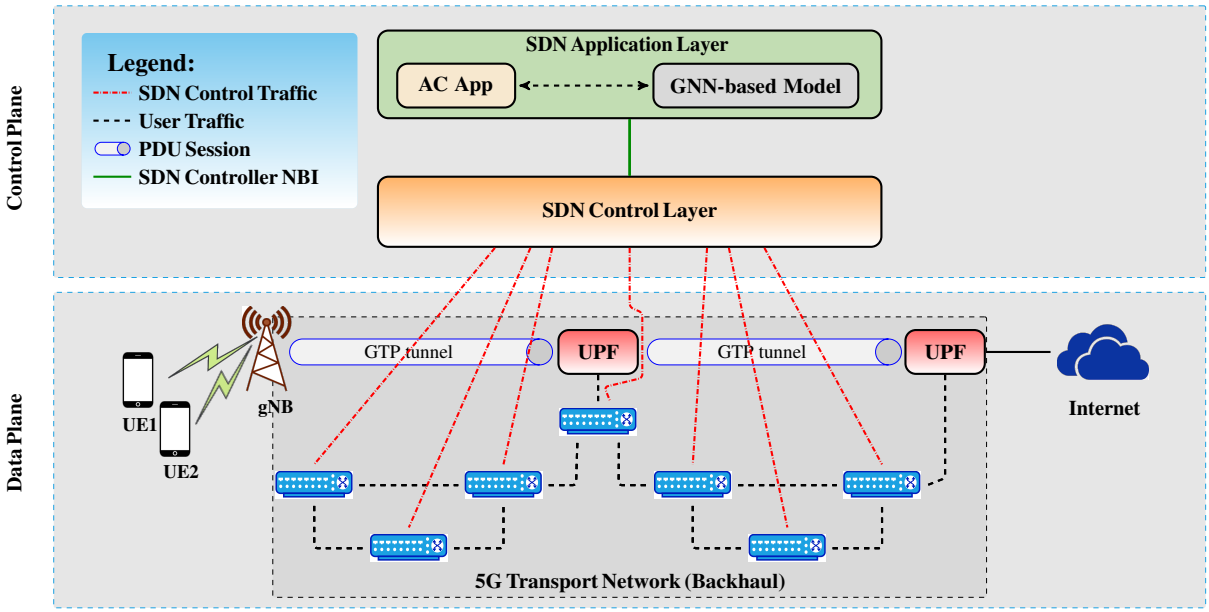


Figure 4.1: Simplified 5G architecture with SDN perspective

In GNN-AC, for every flow newly coming into the network, a PACKET-IN message is generated by the respective access node (i.e., border switches of the SDN forwarding plane or backhaul). The process shown in Algorithm 4.1 is triggered as a response. In what follows, we describe the GNN-AC architecture design along with its workflow.

4.3.2 GNN-AC Solution Design

GNN-AC framework is made of two main modules; (i) *NetDelP*, a network latency prediction module leveraging the RouteNet-F GNN-based model, and (ii) *AdConAgt*, an AC module supporting SDN.

4.3.2.1 NetDelP

Is the module predicting, *in real-time*, the delay on the entire network topology paths using RouteNet-F GNN-based model. It is a data-driven model learning from a dataset with various network topologies and traffic. It has as inputs - A *Graph* schematising the network topology where nodes (edges, respectively) represent switches (links, respectively), - All the available *Paths* between the nodes. - A *Scheduling Policies* such as type and queue size. Finally, - A *Traffic Model* in the form of a matrix including the distribution type and the packet size. As outputs, the NetDelP predicts links' delay.

4.3.2.2 AdConAgt

It is a packet AC guard. Its function is to decide which flow rules to communicate to switches via OpenFlow protocol. These flow rules regulate the traffic in a selection process (i.e., to be forwarded or dropped) to meet QoS agreement. The decision is based on the NetDelP' real-time predicted delay, provided as entry to the AdConAgt. This module interacts with each switch via an SDN controller to

monitor the network traffic and nodes and CRUD the flow rules.

Figure 4.2 details the GNN-AC architecture view represented as a recursive client-server based on SDN controller. The architecture is designed within 3 layers according to SDN ONF standards [26].

Application Layer: Hosts the GNN model (i.e., NetDelP) and the SDN application (i.e., AdConAgt), which communicates its network requirements toward the CP via the NBIs. This application is installed on the top of an SDN controller, for instance, ONOS.

Control Layer: The SDN controller is a central element of this layer. It acts as a server and a client for, respectively, the above and below layers. As a server, it controls network elements by pushing, updating, or deleting flow rules on switches requested by the AdConAgt via REST API. As a client, it monitors the network traffic and elements.

Infrastructure Layer: Or DP, is a composition of network elements, switches such as OVS, which expose their capabilities toward the SDN controller via the SBIs and apply the forwarding rules as computed by the controller.

4.3.3 Mathematical Representation

A flow rule ξ is composed of: *Match set* (μ) to identify a flow; *Action set* (τ) to define the actions executed on each packet of the flow; and *Priority* (ρ) that is used to order rules in the forwarding switch. In our solution, the Match set includes packet DSCP value, the Action set is to *accept and forward* flow's packets through a set of port numbers or to *reject and drop* them, while the priority is the same for all the flow rules (equations 4.1, 4.2, 4.3, and 4.4).

$$\xi = \{[\mu], [\tau], \rho\} \tag{4.1}$$

$$\mu = \{0, 1, 2, ..., 56\} \tag{4.2}$$

$$\tau = \{output(port_{number}), \quad drop\} \tag{4.3}$$

$$\rho = \{0, 1, 2, ..., 65535\} \tag{4.4}$$

4.3.4 GNN-AC Workflow

The GNN-AC framework reduces the load imbalances in the network by real-time latency prediction (line **12**). Also, the specific traffic priority μ is used as a reference for the AC module to get their respective PDB (line **11**) and check if it is exceeded by the Packet Predicted Delays (PPDs) (line **13**). If NO, the AC module translates the chosen path with the minimum PPD into flow rules, which include data such as the output port number (line **14**). Otherwise, the packet will be rejected (line **16**). Finally, the flow rules are pushed to the intermediate nodes involved in the communication (line **17**).

4.4 Performance Evaluation

4.4.1 Setup

We have tested the GNN-AC framework within a linear network topology with three flavours (i.e., scales) (small (c_1), medium (c_2), and large (c_3), respectively), according to the number of used virtual switches (10, 20, and 50, respectively). Each flavour defines 3 different paths (short (p_1), moderate (p_2), and long (p_3)), not necessarily similar between these flavours. We distinguished the paths to show how GNN-AC

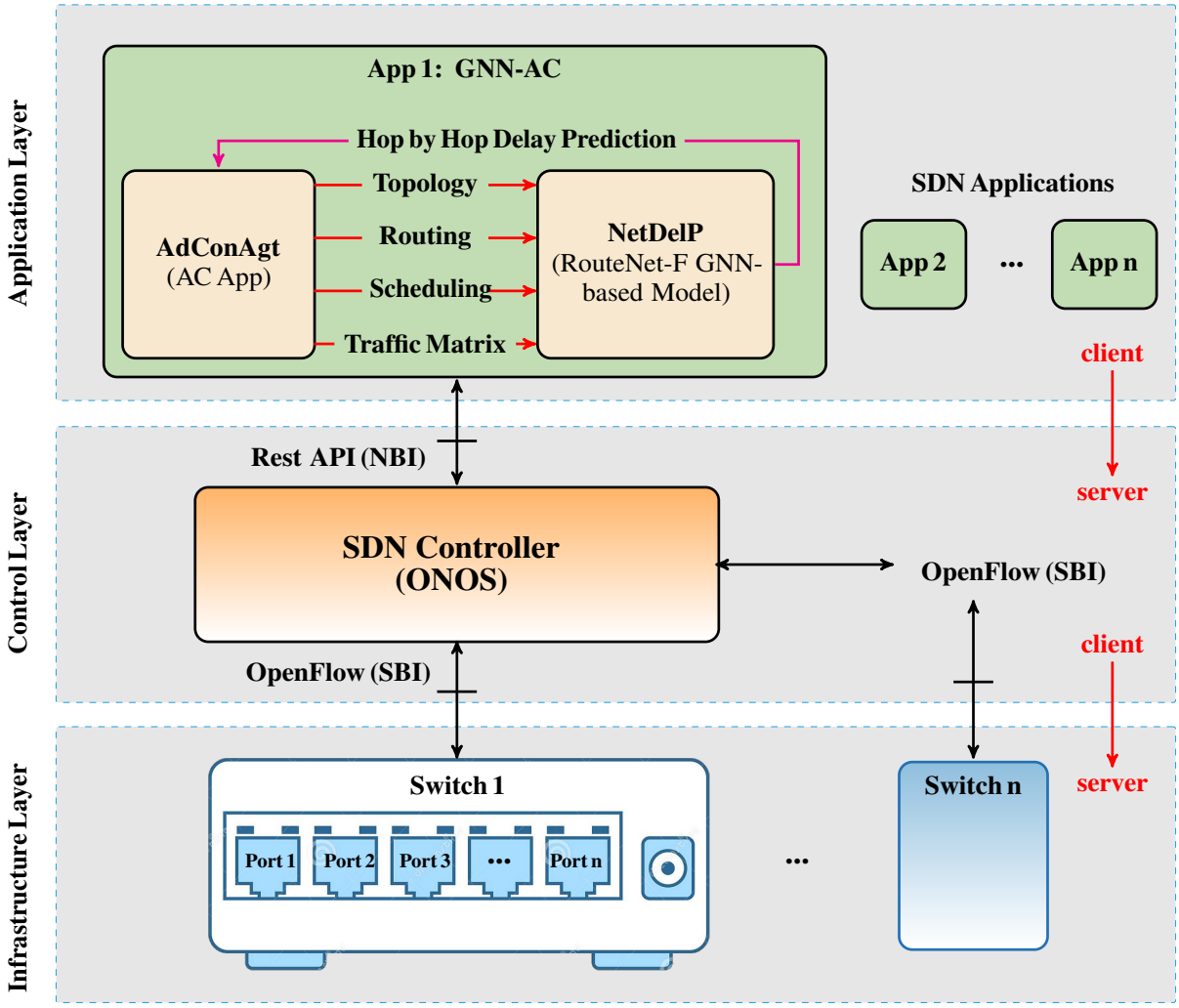


Figure 4.2: GNN-AC architecture view

is balancing the traffic compared with SDNSP. As a source traffic generator, we used 6 Linux Virtual Emulators (LVEs), each of which generates 10,000 Internet Control Message Protocol (ICMP) packets per second (pps) of size 2,000 Bytes (B). Each packet has a specific DSCP (μ) and a respective PDB, which simulate different CoTs (6 flows (f_1 to f_6) in total: f_1 with $5ms$ PDB, f_2 ($10ms$), f_3 ($30ms$), f_4 ($50ms$), f_5 ($60ms$), and finally, f_6 ($75ms$)). We also used an LVE as a destination that receives all the generated traffic in a competition mode. Table 4.1 summarizes the different parameters, and Figure 4.3 draws the setup. We used ONOS as an SDN controller, OVS for the OpenFlow switches, and Mininet for the network topology.

We have put the network under realistic conditions (with different topology flavours, paths, and CoTs from urgent to non-urgent traffic), with a large number of packets and ensuring none of the links is empty, aiming at stressing the different switches. Especially, switches s_1 and s_n, which are shared between the three paths of each flavour.

Algorithm 4.1 GNN-AC workflow

Inputs: {Paths}, {Hosts}, {Links}, {ξ}, {PPD}, {PDB}
Outputs: {New Flow Rule (ξ) (Accept/Drop packet)}
SDN CTRL receives PACKET-IN message with μ info
if (exist path with a ξ satisfying packet DSCP μ) **then**:
    Apply ξ
else
    Find all the potential paths to the destination
    if path list is empty **then**:
        The destination is unreachable
    else
        Get the PDB
        Get the PPDs using real-time Traffic Matrix TM(t)
        if $PPDs < PDB$ **then**:
            Chose the path with the minimum PPD
        else
            Reject the packet
        Install the ξ on the devices
        Goto 5

Table 4.1: Technical details of the setup

Parameter	Value
Operating System	Ubuntu 20.04.4 LTS
Software	ONOS 2.7.0, Mininet 2.2.2, OpnVSwitch 2.13.5, Python 3.9
Protocols	OpenFlow 1.6
Network Topology	Linear
Packet Amount	Max of 6×10^4 packet; (10^4 packet each of the 6 hosts)
Generation rate	10,000 pps
Packet size	2000 Bytes
Number of iterations	100
Bandwidth	200 Mb/s for each link

4.4.2 Results

In this section, we focus on both GNN-AC and SDNSP E2E latency, packet loss, and QoS Breach evaluation. We may distinguish the analysis of the GNN-AC results according to flow types into 2 sets (S_1 and S_2): In S_1, we regroup flows f_1 and f_2, while S_2 contains the rest of flows (i.e., f_3, f_4, f_5, and f_6). In S_1, the $PDB \leq 10ms$ (i.e., $f_1(5ms)$, $f_2(10ms)$), and this requirement is hardly satisfied by the three flavours, especially the third one (i.e., c_3) where the estimated latency is around $20ms$ (as we will see below). For the two first flavours c_1 and c_2, the average latencies is $3\,ms$ and $8\,ms$, respectively (see below). Whilst, in S_2, the $30 \leq PDB \leq 75ms$, and this requirement is relaxed and can be satisfied by all flavours except for the couple (f_3, c_3).

4.4.2.1 E2E Network Latency

Figure 4.4 (respectively, Figure 4.5) shows the minimum, maximum, and median E2E latencies (in ms) obtained with GNN-AC (respectively, SDNSP), for the CoTs f_1, f_2, f_3, f_4, f_5, and f_6 (respectively,

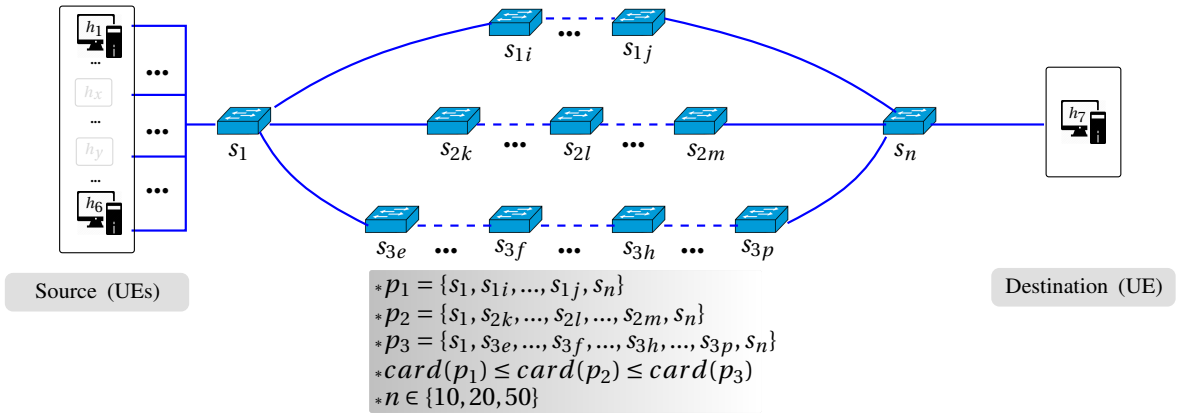


Figure 4.3: Setup's network topology with 3 flavours

independently from CoTs) under the 3 topology flavours (i.e., c_1, c_2, and c_3).

GNN-AC: We first notice that in S_2, we have similar results between the CoTs f_3, f_4, f_5, and f_6. This is for the reason that the S_2 PDB condition is *often satisfied* excluding some packets related to the pair (f_3, c_3). Therefore, GNN-AC is LB the traffic (i.e., f_3, f_4, f_5, and f_6) over the available resources, unlike for S_1 where the f_2 PDB requirement could be satisfied *more* than the f_1. Although we may think that the latency for S_2 is bigger than for S_1 as shown in Figure 4.4, this is due to the latencies representation of the *only accepted* packets under the flows' PDB requirement. We also noted that the latency is increasing with the topology flavour, which is obvious as flavour c_2 (respectively, c_3) includes more switches than flavour c_1 (respectively, c_2 and c_1), which adds additional delays to the E2E latency.

We should mention that for the couple (f_1, c_3), latencies are not represented as c_3 cannot satisfy the f_1's hard PDB condition (i.e., $5ms$), therefore all f_1' packets were rejected.

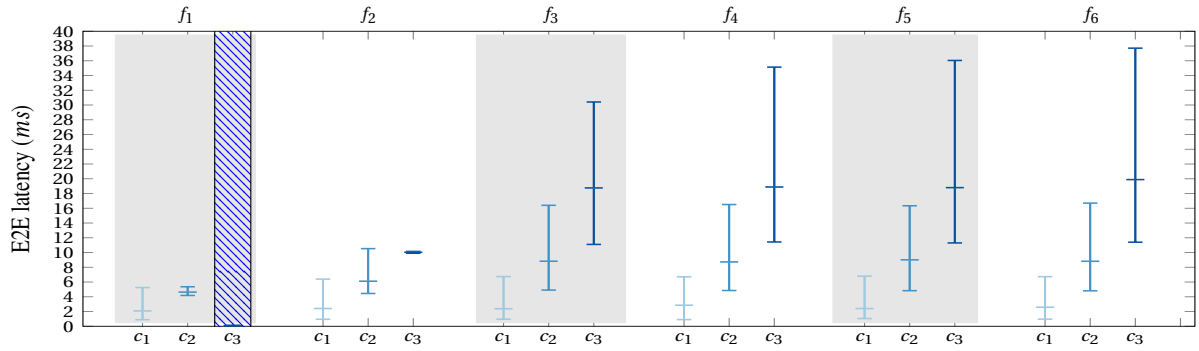


Figure 4.4: E2E latency obtained with GNN-AC for the aforementioned CoTs (f_1 to f_6) under the 3 flavours (c_1 to c_3)

SDN-SP: We remark in Figure 4.5, that latencies obtained for all the CoTs (i.e., S_1 and S_2) are similar to *only* S_2 values in Figure 4.4, on the same way we note that the maximum latency values in S_1 *exceed* the flows respective PDB requirement. Therefore, it does not use the notion of PDB. Indeed, as SDNSP is based on TE shortest path algorithm and not on AC. Hence, all the traffic is treated the same way. The six emulators compete for the shortest path p_1 until it gets saturated. After that, the traffic is forwarded to p_2,

then p_3.

From these results (Figure 4.4 and 4.5), we demonstrated that our solution GNN-AC respects the 3GPP PDB specifications.

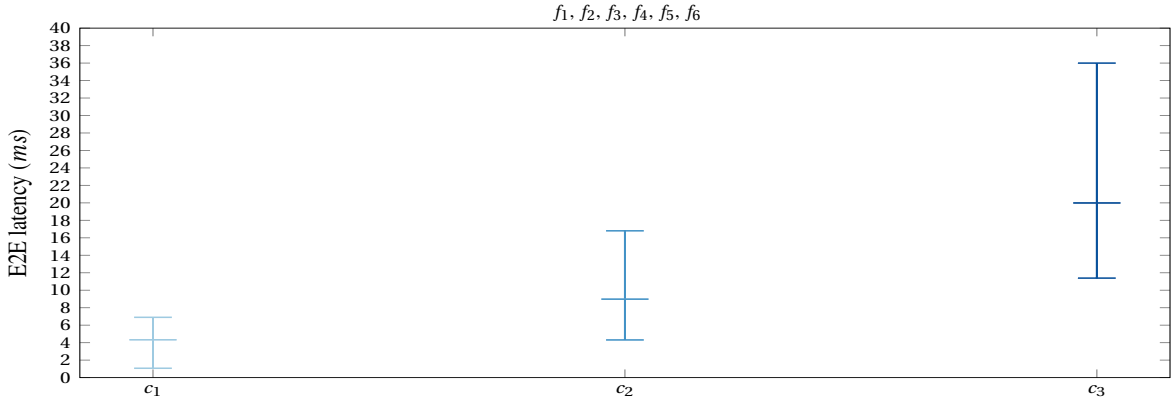


Figure 4.5: E2E latency obtained with SDNSP independently from CoTs (f_1 to f_6) under the 3 flavours (c_1 to c_3)

4.4.2.2 Packet Loss Rate (PLR)

Table 4.2 compares the PLR for GNN-AC with SDNSP in percentage (%) for the six aforementioned CoTs and topology flavours.

GNN-AC: In S_1, we notice that the PLR increases with the number of switches. This is due to the E2E latency increase that yields to the *non-respect of PDB*. Second, we remark that the f_2 PLR is lower than the f_1 PLR as the PDB condition for f_2 is *less strict* than f_1. Last, we can also point the PLR approximating the 100% for flavour c_3 as the estimated latency (i.e., around $20\,ms$) does not satisfy the PDB conditions. In S_2, to our delight, the PLR is smaller than in S_1. In fact, The smallest PDB value in S_2 is $30ms$ used by flow f_3, which is mostly satisfied by the setup even with the flavour c_3 offering approximately $20ms$ delay. Therefore, we reduce considerably the packet loss. The other observation is that the PLR is decreasing, from 13% $\sim$ 12% to 0, for the same flow and the three flavours except for 5% of packets in the pair (f_3, c_3). The reason is that the six emulators are in competition mode, and around 75% (87%, respectively) of traffic flow f_1 (f_2, respectively) is accepted on flavour c_1, which lets less room for flows f_3 to f_6 on flavour c_1. Inversely, PLR which approximates the 100% for flows f_1 and f_2 on flavour c_3 (due to the delay) is *advantageous* for flows f_3 to f_6 on flavour c_3. The same applies on flavour c_2.

SDN-SP: In contrast, in the SDNSP, we notice that the PLR is increasing, with respect to flavours, between 14% and 21%, independently from CoT. It is relevant to note that SDNSP is using shortest path algorithms without any LB approach. We believe that the PLR increase is due to more delay, congestion, and error, potentially introduced by each additional switch.

To sum up, with respect to the SDNSP state of the art, our solution reduces the PLR by more than 4× order of magnitude. In addition, it takes into consideration the CoT and the predicted latency to load-balance the traffic over different existing paths, which is not the case of SDNSP based on the shortest path strategy.

Table 4.2: Packet Loss Rate (PLR) (in %)

CoT	Flavour	GNN-AC	SDN-SP	CoT	Flavour	GNN-AC	SDN-SP
f_1	c_1 c_2 c_3	25 74 100	15 15 19	f_4	c_1 c_2 c_3	12 1 0	14 18 20
f_2	c_1 c_2 c_3	13 31 96	16 17 18	f_5	c_1 c_2 c_3	12 0 0	15 15 18
f_3	c_1 c_2 c_3	13 2 5	15 15 21	f_6	c_1 c_2 c_3	13 1 0	15 16 19

4.4.2.3 QoS Breach

Table 4.3 provides a QoS breach comparison in percentage (%) between GNN-AC and SDNSP. By QoS breach, we mean how many packets from a flow with a given PDB were transmitted, although the PDB condition is not satisfied (i.e., $PDB <$ E2E latency). In case of GNN-AC, we noticed that some values are *not null* for the couples (f_1, c_1), (f_2, c_2), (f_2, c_3) and (f_3, c_3), so we obtain this incorrectness. The reason is due to the latency prediction mean relative error of RouteNet-F model estimated around 6.24% [7]. For the SDNSP case, two more values for couples (f_1, c_2) and (f_1, c_3) violate the QoS agreement. The explanation is due to *non-consideration* of PDB. For both cases, we also remarked that the more PDB increases, the more QoS breach decreases.

Equation 4.5 represents the average QoS breach calculation (G_x) for GNN-AC (G_{GNN-AC}) and SDNSP (G_{SDN-SP}) for all the CoTs and all the topology flavours ($g_x(f_i, c_j)$). For GNN-AC, $G_{GNN-AC} = 3.191$, while for SDNSP, $G_{SDN-SP} = 21.5$ which is 7× bigger than G_{GNN-AC}

$$G_x = \frac{\sum_{i=1}^{6}\sum_{j=1}^{3} g_x(f_i, c_j)}{6 \times 3} \tag{4.5}$$

Overall, GNN-AC solution *significantly* outperforms the SDNSP for the entire flows. Our solution reacts *much* better to CoTs, to meet the QoS of each one.

Table 4.3: QoS breach (in %)

CoT	Flavour	GNN-AC	SDN-SP	CoT	Flavour	GNN-AC	SDN-SP
f_1	c_1 c_2 c_3	4 0 0	40.2 84.7 100	f_4	c_1 c_2 c_3	0 0 0	0 0 0
f_2	c_1 c_2 c_3	0 1.44 50	0 37.4 100	f_5	c_1 c_2 c_3	0 0 0	0 0 0
f_3	c_1 c_2 c_3	0 0 2	0 0 24.7	f_6	c_1 c_2 c_3	0 0 0	0 0 0

4.5 Conclusion

In this chapter, we have explored the use of GNN-based RouteNet-F model for latency prediction and its application for SDN AC in B5G networks. We have investigated the performance of our solution (i.e., GNN-AC) for various topology flavours (i.e., scales) and CoTs and compared the results with the state of the art. The obtained results have demonstrated the potential of GNN models for SDN AC. Indeed, our solution maintains QoS while avoiding congestion in the network. In the future, we will focus on refining the proposed model as well as investigating the use of RA instead of taking a binary decision by accepting or rejecting the packets.

Chapter 5

SDN-based Congestion Control for Time-Sensitive Applications Adopting L4S Transport Layer

5.1 Introduction

The emerging 5G&B networks have tailored the market with new services, including VR, AR, and diverse forms of video streaming (ranging from gaming to conferencing and entertainment). These real-time applications necessitate hard requirements for latency, packet loss, and throughput to ensure seamless, high-volume data streaming for a fully immersive user experience [62]. Transported over TCP, the protocol's congestion control mechanism dynamically adjusts the TCP window size to regulate data transmission rates, impacting throughput. Indeed, upon detecting network congestion, often signaled by packet loss, the TCP congestion control algorithm iteratively reduces its window size until congestion recovery. Besides, TCP employs the *slow start* mechanism to gradually increase transmission rates, which may take time to reach optimal throughput. TCP exhibits notable limitations, including aggressive rate reduction leading to under-utilization of bandwidth and sub-optimal throughput, delayed congestion detection resulting in packet loss and increasing E2E latency, and RA inequity between different TCP streams [63].

To address these limitations, the IETF has introduced a new architecture, currently under specification, known as L4S [64]. This approach leverages the ECN to mark packets [65] and uses a dual-queue framework within switches to distinguish and prioritize L4S traffic. By using the ECN and dual-queue logic, the L4S architecture reduces packet loss, increases throughput, and decreases E2E latency. Indeed, for the packet loss problem, the L4S architecture avoids packet dropping and re-transmission by predicting and preventing congestion (i.e., packet marking). For the high throughput, it adjusts the transmission rate at the Sender derived from congestion probability estimates provided by the Receiver. Besides that, the use of an additional queue sorts the traffic within the two queues (regular and L4S), which reduces the number of packets per queue and leads to reduced queuing delay hence the E2E latency.

E2E latency in packet-switched networks is mostly the *sum of 4 delays* known as processing (t_{proc}), queuing (t_{queue}), transmission (t_{trans}), and propagation (t_{prop}) delays [63]. t_{proc} is the required time to parse packets, verify their checksum, and direct them. t_{queue} is the time a packet spends in a queue before being transmitted onto the link; it is primarily impacted by the queue size and the number of packets in the queue. t_{trans} represents the needed time to push all the packet's bits into the link; determined by packet size and transmission rate ratio. Finally, t_{prop} is the time needed to cross a link between two nodes. t_{prop}

is obtained as a ratio of distance and link speed. While t_{proc}, t_{trans}, and t_{prop} are relatively negligible (order of μs to few ms) compared to t_{queue}, most of efforts to reduce the E2E latency have focused on various solutions including, hardware acceleration [66], parallel processing [67], and computation offloading [68], for t_{proc}. Solutions such as Gigabit Ethernet cards (×10, ×100 $Gbps$), and (extended) Berkeley Packet Filtering ((e)BPF) eXpress Data Path (XDP) [4] address t_{trans}, whilst t_{prop} is reduced via optical fiber with repeaters and amplifiers, Multi-Access Edge Computing (MEC) [69], Multi-Path TCP (MP-TCP) [70], and path optimization algorithms. Improvements to t_{queue} have been made through QoS, queueing discipline (qdisc), and traffic control (tc) implementations [71], and ECN L4S application.

Several studies have explored the potential of L4S and ECN in congestion control. In [3], the authors conducted a comprehensive study on L4S, introducing a congestion control scheduler aimed at enhancing latency and throughput without impacting classical traffic. Despite its innovative features, this solution has yet to be deployed in real-world scenarios. N. Nguyen et al. [72] employed P4 and in-band network telemetry to monitor L4S switch metrics. Their experimentation yielded promising results, showing minimal processing overhead and interesting results, albeit with limitations in deployment flexibility due to reliance on P4 switches. In a separate study, authors in [73] addressed challenges arising from heavy traffic and impairments in data centers, such as TCP-Incast, buffer overflow, and long queues, and proposed an enhanced algorithm leveraging ECN. While effective within data center environments, the applicability of this approach to broader network contexts is limited. Furthermore, [74] introduced a congestion control algorithm aligned with L4S specification, implemented in Web Real-Time Communication (WebRTC), which used ECN for adaptive sending rate adjustments. Comparative evaluations against Google Congestion Control (GCC) baseline demonstrated improved responsiveness. Nevertheless, its notable dependency on manual adjustments and hardware/kernel modifications posed scalability challenges.

Despite the considerable efforts made in these studies, a common limitation to our best knowledge appears to persist. None of these works have fully considered a holistic perspective of the network, which is crucial to accurately identifying bottlenecks. Surprisingly, SDN paradigm was not applied through these works with regard to its significance in orchestrating control across the entire network. In this chapter, we propose to integrate L4S techniques within an *´sdnized'* network. We enhanced the efficiency of QM and ECN marking. This integration allows the server to adjust the sending rate dynamically, guaranteeing ultra-low latency and elevated QoS levels, especially for time-sensitive applications. Overall, this solution provides two key contributions. Firstly, it presents a novel SDN-L4S framework, embedding a new approach to low-latency networking that capitalizes on the strengths of SDN, tailored specifically for the demands of 5G&B in scenarios marked by high network loads. Secondly, it uses an adaptive QM addressing the intricate requirements of diverse delay-critical services; we introduce a dynamic QM algorithm that enforces priority, safeguarding the L4S traffic even with the presence of normal traffic.

The rest of the chapter is organized as follows: Section 5.2 introduces the background. Our solution is shown in Section 5.3 and evaluated in Section 5.4. Section 5.5 concludes the chapter.

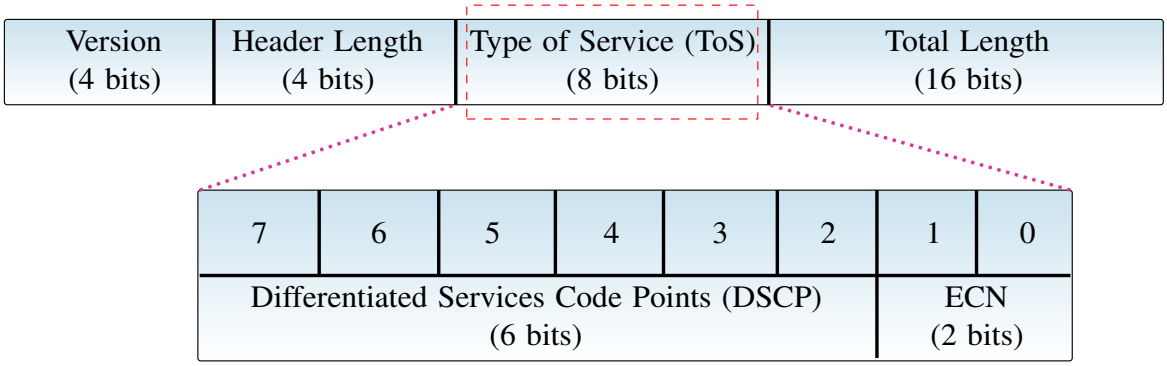
5.2 Background

In TCP/IP networks, congestion is signaled by packet drops, an effective yet performance-degrading mechanism. Thus, we need alternative methods to communicate congestion information to endpoints. In this section, we introduce the concepts of ECN and L4S, which form the core of our solution.

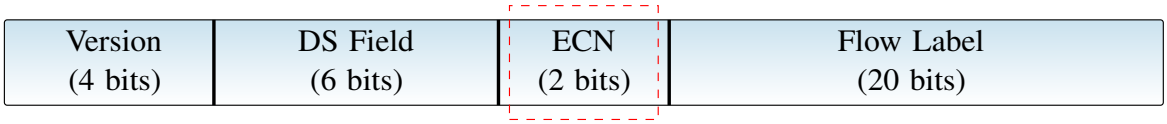
5.2.1 ECN

ECN serves as a mechanism to signal or anticipate network congestion. It facilitates E2E congestion notification between two endpoints (i.e., Sender and Receiver) within TCP/IP-based networks. For optimal E2E congestion control, all the nodes involved in the transmission path including the endpoints *must* be ECN-enabled. The presence of any device along the path that does not support ECN breaks down the E2E ECN functionality.

The ECN congestion notifications aim to reduce packet loss and delay by prompting the Sender to reduce transmission rates without dropping packets. The IETF has introduced the document RFC 3168 [65], which describes how the ECN is added to the IP header. IETF has reserved two bits in the IP header's Type of Service (ToS) field or the DSCP field (see Figure 5.1) to signal congestion more explicitly and proactively. The nodes within the network (routers, switches, access points, and base stations) set the ECN field to 11 (that is, Congestion Experienced (CE)) when detecting increased congestion, providing advanced notice to the Sender and fostering congestion feedback with minimal packet loss. The Sender interprets the bits of the ECN and adjusts its transmission rates or congestion control algorithms accordingly.



(a) Internet Protocol version 4 (IPv4) header, first 4 bytes focusing on ToS



(b) IPv6 header, first 4 bytes

Figure 5.1: ECN bits in the IP headers

5.2.2 L4S

L4S is an innovative technology intended to guarantee high throughput while minimizing delay and packet loss in Internet traffic. It is based on the idea of signaling early on congestion (CE) when the number of queued packets in a network node surpasses a predefined threshold. This approach ensures a consistently low queuing delay, reducing the E2E latency while optimizing link utilization. This technology relies mainly on two main mechanisms, namely *congestion control* and *transport feedback*.

ECN is the central element in the *congestion control* mechanism. Table 5.1 shows the possible values of ECN and their meaning. A setting of 00 or 10 in the ECN field indicates that one or more network nodes in the path do not support L4S capability. On the contrary, the values 01 and 11 denote the presence of

L4S traffic, with all nodes of the network along the path supporting L4S capability, at the difference that 11 is used to express congestion on the path.

Table 5.1: L4S codepoints and meaning

Binary Codepoint	Codepoint Name	Description
00	Not-ECT	Not ECN-capable transport
01	ECT(1)	L4S-capable transport
10	ECT(0)	Not L4S-capable transport
11	CE	Congestion Experienced

Transport feedback represents the transport protocols used to alert the Sender accurately about congestion. Today's most transport protocols support the ECN feedback, we can cite TCP [75], Stream Control Transmission Protocol (SCTP) [76], Quick UDP Internet Connections (QUIC) [77], and Real-time Transport Protocol (RTP) [78]. Unlike traditional ECN, L4S promptly marks packets and smoothens congestion feedback for each marked packet.

Figure 5.2 illustrates the key components of the L4S architecture and their interactions. A server, embodying the L4S Sender, responds to a client by generating streams with a dynamically determined transmission rate $\lambda(t)$ in each round trip. The client serves as the L4S Receiver, consuming the Sender's streams and subsequently providing congestion feedback. Furthermore, the illustration emphasizes the importance of having L4S-capable transport nodes along the path between the L4S Sender and Receiver.

In congestion signaling and rate adaptation, the process unfolds as follows: The Sender signals L4S support using ECN codepoint ECN-Capable Transport (ECT) (i.e., *ECT(1)*). Network nodes then recognize the packet as an L4S packet and, in case of congestion, alter the ECN bits to indicate *CE*. Upon reaching the Receiver, if the ECN bits indicate congestion along the transmission path, the Receiver notifies the Sender. In response to the congestion notification, the Sender reduces the transmission rate.

In protocols like TCP, transmission rate reduction occurs through congestion control mechanisms, such as TCP Friendly Rate Control (TFRC) [79], Additive Increase/Multiplicative Decrease (AIMD) [80] scheme, along with other schemes including slow start [81] and Congestion Window (CWND) [82]. These algorithms adjust the Sender's transmission rate based on received ACKnowledgment (ACK) and congestion indicators, including ECN markings.

5.3 SDN-L4S Solution

In this section, we provide an overview of our proposal, detailing its design and workflow.

5.3.1 Solution Overview

The solution is designed within a distributed architecture involving an L4S Sender, an L4S Receiver, and an SDN controller. It comprises the components shown in Figure 5.3. In the following we describe the components of interest. Note that our solution is applied to 5G&B. For more details on the SDN architecture and its applicability to 5G&B, we encourage readers to refer to the state of the art Chapter 2.

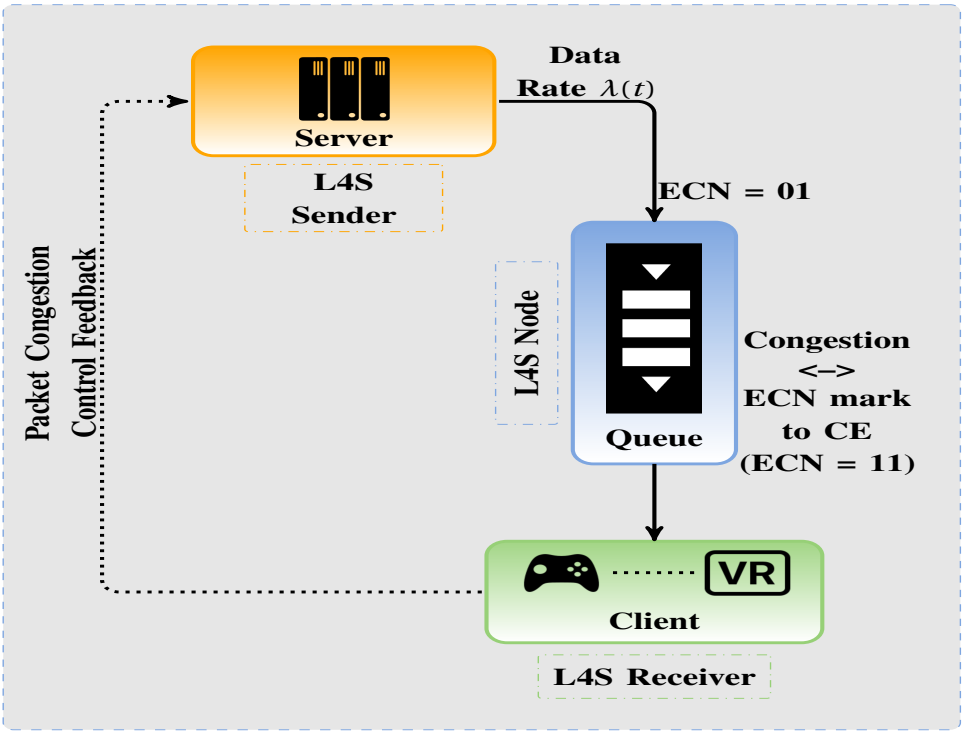


Figure 5.2: L4S mechanism

- *Queue Manager:* It configures an *on-demand* L4S and/or classical queue with a $queue_{id}$ based on traffic type. L4S traffic is directed to L4S queue and regular traffic to classical queue. Each switch port may have at most two queues, with the L4S queue enjoying superior priority and output rate.
- *ECN Marker:* It applies ECN markings to packets based on congestion conditions.
- *Flow Rules Manager & Path Selector:* It manages and processes flow rules, determining the optimal path and queue for traffic based on predefined policies and network conditions.
- *Packet Interceptor:* It intercepts incoming packets, allowing for real-time analysis.
- *Packet Parser:* Manipulates packet headers and payload contents, and reads the field of interest.
- *Rate Calculator:* It dynamically regulates the packet transmission rate $\lambda(t)$. This controls the data flow, preventing rapid bursts and contributing to smoother and more efficient data transfer.
- *Ingress/Egress Listener:* Sends/receives L4S/regular packets and receives congestion feedback's from the L4S Receiver.
- *Congestion Probability Calculator:* Upon getting L4S packets, the L4S Receiver calculates the probability of congestion and notifies the Feedback Agent.
- *Feedback Agent Sender/Receiver:* It sends/receives feedback information to/from the L4S Sender/Receiver via *In-Band Channel* by extending the TCP header and setting the optional field to express the congestion probability.

5.3.2 Notation

In this section, we present a mathematical notation outlining our solution. We define an OpenFlow rule, denoted as ξ (see Equation 5.1), as a composite of essential components [83]:

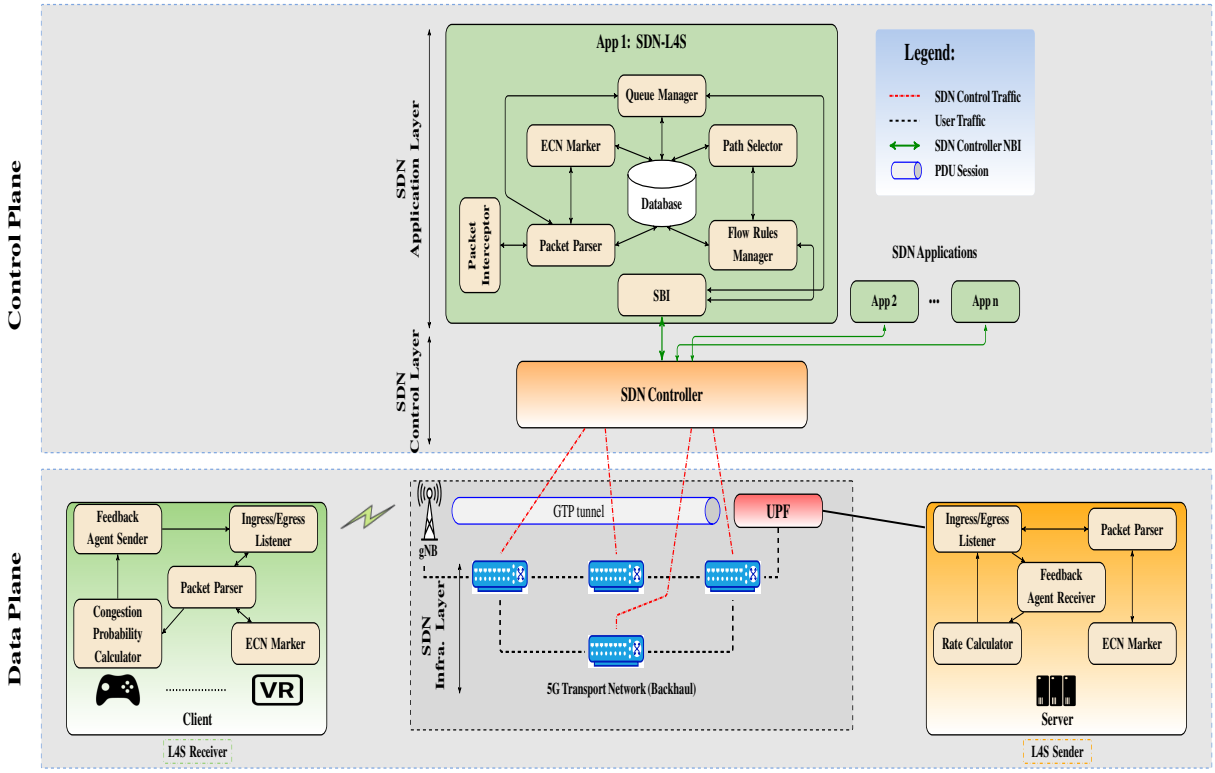


Figure 5.3: SDN-L4S solution applied to 5G&B

(i) The *Match set* (μ) (see Equation 5.2), which characterizes the packet's ToS value. Used to classify flows into two categories: *L4S* traffic, identified by $\mu = 185$, and *regular* (Non-L4S) traffic, where $\mu \neq 185$.

(ii) The *Action set* (α) (see Equation 5.3) specifies the action to be executed for each flow packet. We identify three main actions. The first action directs L4S traffic to the L4S queue, while regular traffic goes throw the classical queue based on the $queue_{id}$ parameter. The second action is used in the event of queue congestion detection (in L4S or classical queue) to mark each packet with the ECN value equal to 11. Congestion is determined when the queue load (γ) exceeds a predefined threshold (q_1) of the queue size (σ). The third action differentiates between congestion and non-congestion events; only regular traffic is subject to dropping in case of congestion; elsewhere, the traffic, whether L4S or not, is accepted and forwarded to the Receiver through a specified set of port numbers.

(iii) *Priority* (δ) (Equation 5.4) that is used to order rules in the forwarding switch.

$$\xi = \{ [\mu], [\alpha], \delta \} \tag{5.1}$$

$$\mu = \{0, 1, 2, ..., 224\} \tag{5.2}$$

$$\alpha = \{ setECN(value),\ output(port_{n^\circ}, queue_{id}),\ \ drop \} \tag{5.3}$$

$$\delta = \{0, 1, 2, ..., 65535\} \tag{5.4}$$

Equation 5.5 defines the new rate of packets departure λ dynamically adjusted by the L4S Sender in response to congestion. This new rate, measured in $Mbps$, is intricately linked to the probability of congestion, denoted as p (see Equation 5.6, in which n is the number of received packets in congestion state (i.e., packets marked with $ECN = 11$) and N is the total number of received L4S packets (i.e., L4S traffic with $\mu = 185$)). The probability p is computed on the L4S Receiver side for each new packet arrival

and used to guide the L4S Sender's rate adaptability using a predefined threshold (q_2). Indeed, in each system, the probability of system occupancy $p \approx \frac{\lambda_a}{\lambda_a+\lambda_d}$, where λ_a is the arrival rate to the system and λ_d is the departure rate [84]. This can be further simplified as $p \approx \frac{1}{1+\lambda_d}$, ultimately leading to the equation 5.5.

$$\lambda \approx \begin{cases} \lambda_0 & \text{if } 0 \leqslant p < q_2 \\ \frac{1}{p} - 1 & \text{if } q_2 \leqslant p < 1 \end{cases} \tag{5.5}$$

$$p = \begin{cases} 0 & \text{if } N = 0 \\ \frac{n}{N} & \text{if } N \neq 0 \end{cases} \tag{5.6}$$

5.3.3 Workflow

The proposed SDN-L4S algorithm aims to prevent congestion in an SDN-managed network by integrating adaptive rate control, QM, and congestion feedback mechanisms. The algorithm takes into account the following inputs: the network topology information (encompassing paths {Paths}, hosts {Hosts}, and link characteristics {Links}), the set of available flow rules $\{\xi\}$, the real-time queue statistics {Stats}, the initial transmission rate λ_0 matching the speed of Network Interface Controller (NIC), and the congestion-related counters (n, N, p) initialized to zero. Outputs from the algorithm include updated flow rules, a probability of congestion (p), and a revised transmission rate λ. The algorithm is partitioned into the following sections.

L4S Sender: The logic behind this component is to dynamically fine-tune responses to the Receiver according to a designated rate λ. This rate undergoes *near real-time* adjustments, where, with each newly received congestion probability (p), the Sender recalibrates the transmission rate (λ) by applying Equation 5.5 (see Algorithm 5.1).

Algorithm 5.1 SDN-L4S Sender algorithm

1: **Inputs:** λ_0, $p_0 = 0$, p, q_2
2: **Outputs:** λ
3: if ($p \geqslant q_2$) **then**:
4: Calculate the new rate: $\lambda = \frac{1}{p} - 1$
5: else
6: Re-initialize the new rate: $\lambda = \lambda_0$
7: Adjust transmission rate to λ
8: Send next packet

SDN Controller: Performs actions on traffic and manages switch queues based on processed packet information (μ) received from PACKET-IN messages. If no suitable path adhering to the specified configuration ξ for the given μ, alternative routes are explored. If no path is found, the destination is labeled as unreachable. Subsequently, the controller manages both L4S and regular traffic. For L4S traffic, it handles the L4S queue, creating one if needed, and monitors its load. If the load exceeds a predefined threshold (q_1), the packet is marked as congested ($ECN = 11$) and forwarded. Meanwhile, the regular traffic is routed through the standard queue, with packets marked as congested and dropped if the queue load exceeds q_1. A new configuration ξ is applied to the switches (see Algorithm 5.2).

Algorithm 5.2 SDN-L4S Controller algorithm

Inputs: {Paths}, {Hosts}, {Links}, {ξ}, {Stats}
Outputs: {ξ}
SDN CTRL receives PACKET-IN message with μ info
if (exists path with a ξ satisfying packet μ) **then**:
    Apply ξ
else
    Find all potential paths to the destination
    if path list is empty **then**:
        The destination is unreachable
    else
        if ($\mu = 185$) **then**:
            Create L4S queue if not exists
            if (load (L4S queue) $\geqslant q_1$) **then**:
                Mark packet as congested: $ECN = 11$
        else
            Choose the path with a regular queue
            if (load(regular queue) $\geqslant q_1$) **then**:
                Mark packet as congested: $ECN = 11$
                Drop the packet
        Install the ξ on switches

L4S Receiver: The main action of this component is to prevent congestion and notify the L4S Sender to reduce its transmission rate. A probability of congestion p for L4S traffic is determined by the ratio of n, representing the received packets flagged as congested ($ECN = 11$), to the total number of L4S packets N or simply set to zero (using equation 5.6). The counters N and n are then updated accordingly (see Algorithm 5.3).

Algorithm 5.3 SDN-L4S Receiver algorithm

Inputs: $n = 0$, $N = 0$
Outputs: p
Handle received packet:
if ($\mu = 185$) **then**:
    $N++$
    if ($ECN = 11$) **then**:
        n++
    Calculate the congestion probability: $p = \frac{n}{N}$
Notify the L4S Sender of the updated value of p

5.4 Performance Evaluation

5.4.1 Setup

Our framework has been tested within a linear topology with 2 OVS switches (v2.13.5). We have used 24 Linux nodes (Mininet v2.2.2) as traffic source generators, where 12 represent L4S Senders and the other 12 for regular Senders. Each Sender generates a TCP stream of size 65,507 Bytes. We have distinguished

the entire traffic within 2 sets according to ToS (μ) (12 flows with $\mu = 185$ and 12 flows with $\mu \neq 185$). We also used a Linux node as an L4S Receiver that receives all the generated traffic in a competition mode. As SDN controller, we used ONOS (v2.7.0) with OpenFlow (v1.6). The tests have been repeated 100 times with a number of packets generated of 24×10^4 packet for each iteration (10^4 packet for each of the 24 hosts). For the bandwidth of the links, it was fixed to $10\,Gbps$.

5.4.2 Results

Now, we present the results of the measurement campaign regarding the E2E latency and PLR impacted by queue load, congestion probability, as well as QM.

5.4.2.1 E2E Network Latency

In this section, we analyze the causes that affect the latency.

Congestion Probability Threshold Impact: Figure 5.4 shows the impact of congestion probability threshold q_2 on the average E2E latency (in ms) for both SDN-L4S and SDN-Traditional Congestion Control (TCC), with the queue load threshold q_1 held constant at 80%. We may notice that for all q_2, SDN-L4S latency is *always* shorter than SDN-TCC latency, attributed to its adaptive transmission rate that reduces the queuing delay. Furthermore, as q_2 increases, the SDN-L4S latency gradually rises, eventually approaching the SDN-TCC latency as q_2 approaches 100%. Conversely, SDN-TCC demonstrates higher and relatively stable latency, attributed to its packet drop mechanism when q_1 is reached, regardless of the value of q_2.

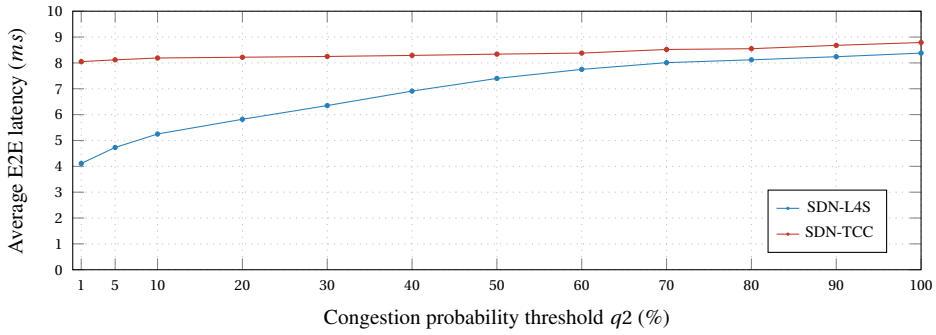


Figure 5.4: E2E average latency versus q_2 ($q_1 = 80\%$)

Queue Load Threshold Impact: Figure 5.5 shows the impact of queue load threshold q_1 on the average E2E latency (in ms) for both SDN-L4S and SDN-TCC, with the congestion probability threshold q_2 fixed at 1%. We remark that SDN-L4S latency is shorter than SDN-TCC once, and this is true for all q_1 values, thanks to the L4S mechanism. Besides that, SDN-L4S latency is approximately $\times 1/2$ that of SDN-TCC. Despite the fact that q_1 impacts both solutions, SDN-L4S responds better to congestion because it is based on the early signaling of congestion (q_2).

Queue Management Impact: Figure 5.6 highlights the impact of Queue Management (QM) on the E2E latency. It displays the minimum, maximum, and median values (in ms) obtained for both SDN-L4S and SDN-TCC with/without QM (i.e., SDN-L4S$_{\text{QM}}$, SDN-TCC$_{\text{QM}}$, ¬SDN-L4S$_{\text{QM}}$, and ¬SDN-TCC$_{\text{QM}}$, respectively). We have fixed in this scenario the couple q_1 and q_2 at 80% and 1%, respectively. Initially,

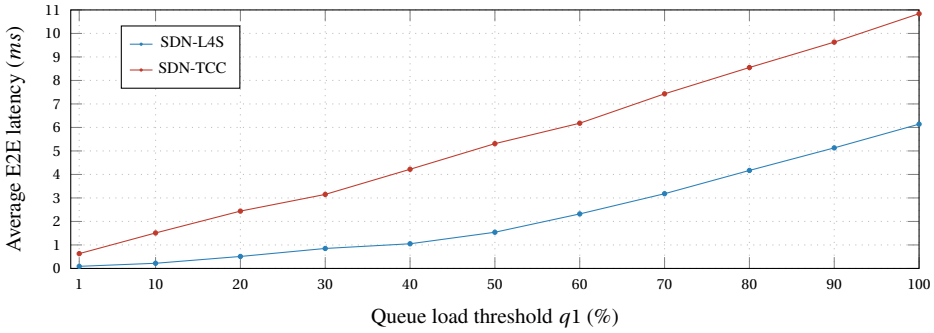


Figure 5.5: E2E latency versus q_1 ($q_2 = 1\%$)

latency values are consistently lower across all scenarios during the early phase of traffic transmission before queues become congested. It is worth noting that when QM is employed, a dedicated queue is created for L4S traffic with higher priority and transmission rate, resulting in notably lower latency for SDN-L4S$_{\text{QM}}$ than other scenarios. Conversely, ¬SDN-L4S$_{\text{QM}}$ presents better latency then SDN-TCC, thanks to the ECN mechanism. Furthermore, SDN-TCC$_{\text{QM}}$ demonstrates smaller values compared to ¬SDN-TCC$_{\text{QM}}$, as it avoids congestion caused by all traffic (i.e., L4S and regular) competing for the same queue.

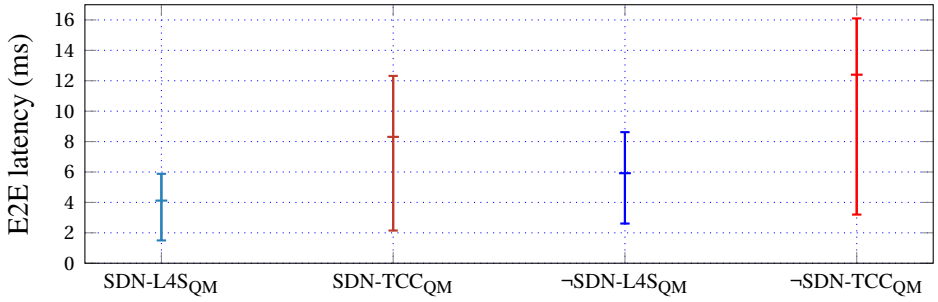


Figure 5.6: QM impact on the E2E latency ($q_1, q_2 = 80\%, 1\%$)

5.4.2.2 Packet Loss Rate (PLR)

Now, we focus on the causes of PLR variability. Table 5.2 presents the PLR obtained for SDN-L4S and SDN-TCC under different combinations of q_1 and q_2 (i.e., SDN-L4S$_{(q_1,\, q_2)}$, SDN-TCC$_{(q_1,\, q_2)}$), while $q_1 \in \{50\%, 80\%\}$ and $q_2 \in \{1\%, 10\%\}$. Please notice that SDN-TCC is not impacted by q_2 according to Section 5.4.2.1, so the PLR. The PLR results demonstrate that SDN-L4S surpasses SDN-TCC and this is true for all q_1 and q_2. When q_1 if fixed, SDN-L4S$_{(x,\, 1)}$ PLR is smaller than SDN-L4S$_{(x,\, 10)}$ as the congestion is detected earlier when $q_2 = 1\%$, so the L4S Sender reduces the transmission rate earlier. Whilst, by fixing q_2, SDN-L4S$_{(50,\, x)}$ PLR is smaller than SDN-L4S$_{(80,\, x)}$, this is obvious since the congestion probability increases slower when q_1 is high (please refer to Algorithm 5.2 lines 13-14 and Algorithm 5.3 lines 6-8). Lastly, SDN-TCC$_{(50,\, x)}$ PLR is higher than SDN-TCC$_{(80,\, x)}$ due to drop mechanism that occurs frequently for $q_1 = 50\%$ (Algorithm 5.2 lines 17-19).

Table 5.2: Packet Loss Rate (PLR) (in %)

$SDN-L4S_{(q_1,q_2)}$	$PLR(\%)$	$SDN-TCC_{(q_1,q_2)}$	$PLR(\%)$
$SDN-L4S_{(50,1)}$	0.16	$SDN-TCC_{(50,x)}$	24.2
$SDN-L4S_{(50,10)}$	0.5		
$SDN-L4S_{(80,1)}$	2.66	$SDN-TCC_{(80,x)}$	18.4
$SDN-L4S_{(80,10)}$	3.7		

5.5 Conclusion

In this chapter, we have designed a congestion control solution inspired by the IETF standards on the L4S, that we applied to *SDNized* network tailored for video streaming applications in B5G. We have demonstrated through the measurement campaign that our solution outperform TCP congestion control mechanisms in term of latency and packet loss. As future work, we aim to explore how to integrate the L4S mechanism within the RAN. Besides that, we would like to investigate resource usage and identify a trade-off between queue loads and congestion probability.

Chapter 6

SDN for CEC Computing Nodes Interconnection Utilizing eBPF

6.1 Introduction

CEC is undoubtedly the next evolution of cloud computing, where workload is executed over the continuum to gain the low-latency capabilities of edge and far-edge computing nodes, while centralized cloud resources handle high-load, and delay-tolerant tasks. The CEC will build on the advance and democratization of edge computing, with nodes positioned close to end users and an increasing array of end-user devices—such as IoT gateways, Unmanned Aerial Vehicle (UAV), and smartphones—capable of running workloads. Meanwhile, the shift toward microservice-based applications will further drive CEC adoption, enabling application components to be distributed across the continuum to meet SLA requirements. However, interconnecting cloud, edge, and far-edge nodes presents challenges, as multiple providers must collaborate to build a cohesive continuum and ensure seamless application deployment per SLA standards. This cooperation is essential, as edge nodes have limited computing capacity and require support from cloud and other edge providers to manage sudden computation demands.

Interconnecting CEC nodes that carry the DP of deployed microservices is essential, requiring (i) programmability to enable network orchestrators to specify QoS levels and resiliency for each service, (ii) QoS enforcement to ensure SLAs—such as low latency, high bandwidth, and minimal packet loss—and (iii) resiliency by enabling route selection flexibility for services. Two primary interconnection solutions are considered: relying on the underlying network (e.g., Segment Routing or MPLS tunnels) or using overlay networks over existing Internet links (e.g., SD-WAN). While the first approach efficiently supports QoS, it assumes that all CEC nodes can control the underlying network to define QoS levels and paths for application microservices, a capability that cloud providers typically possess when connecting their own data centers with MPLS tunnels. However, this approach is limited since cloud and edge providers often rely on network links operated by third-tier ISPs. Additionally, CEC infrastructures often include edge and far-edge computing resources connected via standard network connectivity, with no control over the underlying network.

On the other hand, SD-WAN is a key technology for interconnecting CEC nodes as it meets the three main requirements of programmability [85], QoS, and resiliency. Leveraging SDN principles, SD-WAN simplifies network management by decoupling hardware from control programs and using software and open APIs to abstract infrastructure. It creates overlay networks on top of heterogeneous underlay networks, including those from different ISPs, while maintaining consistent addressing. SD-WAN supports multiple concurrent WAN connections and can interconnect sites in various topologies, such as mesh, to build

a full overlay where SD-WAN edge nodes manage routing for SLA, resiliency, and scalability without relying on underlying network resources. However, most current SD-WAN solutions are proprietary, limiting innovation. This work addresses that gap by proposing an SD-WAN framework that leverages eBPF in the Linux kernel to design SD-WAN edge nodes, enabling overlays between CEC nodes. These nodes are managed by ONOS controller to ensure QoS across links that connect microservices across the continuum.

The rest of this chapter is organized as follows: Section 6.2 presents the eBPF technologies. Section 6.3 details our solution. Section 6.4 describes the methodology and testbed. Section 6.5 presents the performance evaluation and results. Finally, Section 6.6 concludes the chapter.

6.2 Background

To better understand our contribution in this chapter, we introduce the (e)BPF and the XDP paradigms.

6.2.1 (e)BPF

eBPF is a Linux-based Virtual Machine (VM) that runs sandboxed programs in a privileged mode to *safely* and *efficiently* extend the capabilities of the kernel with custom code that can be injected at run-time without requiring changes in the kernel source code or load kernel modules. The eBPF is an *event-driven* program triggered when the kernel or application passes a certain *hook point*. eBPF has some predefined hook points that include system calls, every kernel function, kernel trace points, and network events, to name few. If a predefined hook does not exists for particular need, it is possible to create one at kernel probe or user probe, almost anywhere (see Figure 6.1).

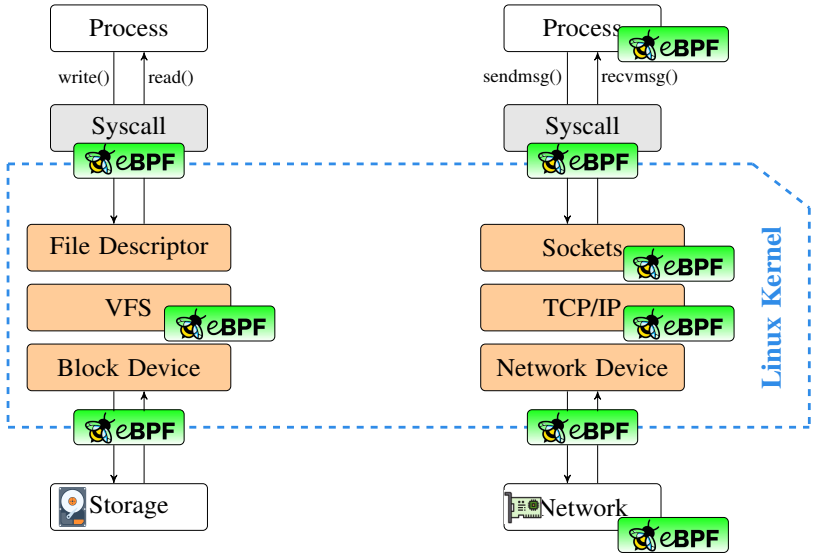


Figure 6.1: eBPF potential hook points

Executing an eBPF program involves several steps. First, the compilation step, where the eBPF source code (in restricted C) is compiled by the Clang/Low Level Virtual Machine (LLVM) toolchain into bytecode stored in an Executable and Linkable Format (ELF) object file. This is followed by a loading step, where 'libbpf' loads the ELF file to a designated *hook point* via system calls. Upon injection, the bytecode undergoes a verification step to ensure safe execution, checking for proper memory access and

termination, enforcing limitations like maximum instruction counts and memory access through *eBPF maps* only, which are set of key-value stores with different access semantics (e.g., array, hash, and queue). Finally, the program goes through the Just In Time (JIT) Compilation step before execution, translating the bytecode to machine-specific instructions for execution as native kernel code.

6.2.2 XDP

Is a network type of eBPF programs, designed for high-performance packet processing. Identified by its hook point within Linux kernel's networking stack, specifically in the reception chain of the *network device driver*, prior to Socket Kernel Buffer (SKB) allocation. The XDP program is triggered *immediately* on the *ingress* path in response to network events, typically upon packet reception. The hook point varies according to how the XDP program is attached. We distinguish the *generic*, *native*, and *offloaded* XDP. Generic XDP is loaded into the kernel as part of the regular network path, making it suitable for testing. Native XDP is loaded within the network card driver, providing better performance but requiring driver support. Offloaded XDP runs directly within (smart) embedded NICs and requires specific device support. Network device drivers may not support XDP hooks, in which case the generic model is utilized. In Linux 4.18 and later, supported drivers include Veth, Virtio, Tap, Tun, Lxgbe, I40e, Mlx5, and MLX4, to name few. XDP has become a popular mechanism for accelerating and offloading packet processing from user-space applications for high-performance networking applications.

6.3 SD-WAN Featuring eBPF Proposal

6.3.1 Holistic Perspective

The HELIOS architecture depicted in Figure 6.2, was inspired by the MEF organization [30]. It is designed to meet the need for scalable and flexible connectivity across distributed networks by decoupling the CP from the DP. The architecture comprises two layer.

Control Layer: This layer hosts the central attraction (i.e., *ONOS SDN controller*). It acts as a server and a client for, respectively, the above and below layers. As a server, it receives requests from the application logic via its NBIs, extracts the flow rules and distribute them among the infrastructure using the gRPC protocol. It aims to dynamically, seamlessly, and simultaneously create, update, and delete eBPF maps within Edge-Gateways. As a client, it monitors the network traffic and devices, gathering the necessary information from the infrastructure (below) layer to push modifications in real-time in the event of misbehavior, while also returning statistics and monitoring data to Verticals.

Infrastructure Layer: (Or DP) comprises the network elements such as end-user devices, Edge-Gateways, MPLS equipments, and Cloud resources. Our focus is on the Edge-Gateways, which serve as endpoints for SD-WAN tunnels, where we implement eBPF accelerations. Traditional SD-WAN transit traffic by user-space, causing overhead and reducing performance; even optimized solutions like Data Plane Development Kit (DPDK) dedicate hardware (Central Processing Unit (CPU) and NICs) exclusively to specific applications. In contrast, eBPF enables resource sharing, improving efficiency. Each Edge-Gateway hosts our Low-Latency Intelligent Network eXecution using eBPF (LINX) prototype (see Section 6.3.2), which maintains eBPF maps that store the flow rules. Each LINX XDP Section Program attaches to one interface of the gateway. We distinguish three types namely; *GRE* interface(s), which facilitate(s) communication between sites over WAN links. *Non-GRE* interface(s) for Local Area Network (LAN) communication, for instance between end-user devices, and the Edge-Gateway. respectively. Beside that a control (*gRPC*) interface is used to listen for SDN controller requests.

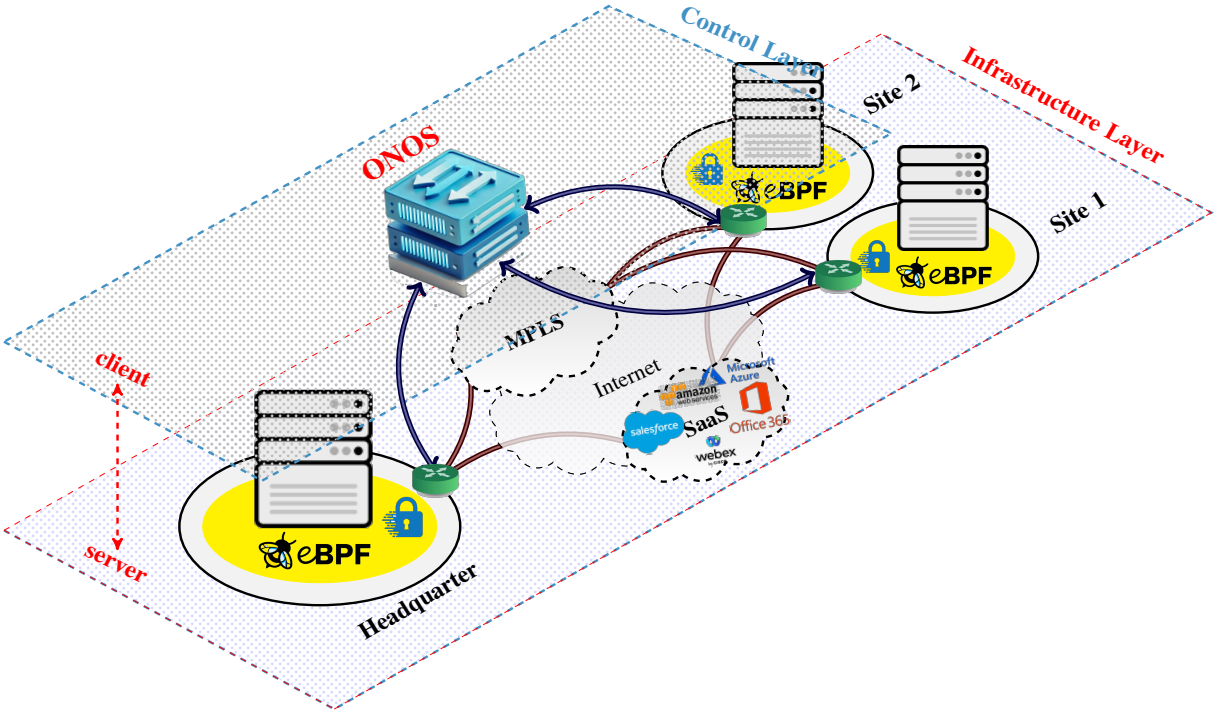


Figure 6.2: HELIOS architecture (global view)

6.3.2 Detailed View

Now, we focus on the eBPF-based DP architecture. Modern SD-WAN supports two deployment approaches: Datacenter-to-Datacenter (DC-to-DC) or End-User-to-Datacenter (End-User-to-DC). Our solution supports both, managing traffic from End-User and/or DCs. To meet these requirements, and in line with the SD-WAN purposes (including, efficiently forwarding and LB traffic across DCs), we have designed the LINX architecture (Figure 6.3) as follows:

Management Layer: Operates at the user space. This layer functions as a core library responsible for configuring both user Data-Path and the edge itself. It includes the following components: (i) - *Yet Another Markup Language (YAML) Config Validator*, which applies configuration settings for specific Data-Path aspects, defining CP and DP interfaces, ports, and supporting eBPF acceleration. (ii) - *gRPC Server* enables interaction with the SDN controller via Remote Procedure Call (RPC), handling control messages on the default port 50051 for creating, updating, and deleting GRE overlays. (iii) - *GRE Overlay Manager* oversees GRE tunnels, it receives packet processing rules from the gRPC server, and coordinates tasks like prioritizing Packet Detection Rules (PDRs) and managing Forwarding Action Rules (FARs) and Load Balancing Rules (LBRs). (iv) - *eBPF Program Manager* manages the lifecycle of eBPF programs within the device driver, enabling real-time configuration changes on the flow rules via on-the-fly generated *eBPF skeleton* APIs.

Data-Path Layer: Responsible for processing End-User/DC traffic, implementing a pipeline where decisions are made about the fate of each packet—whether it is passed, dropped, or redirected. The pipeline is divided into four key components as follow: (1) - *Traffic Parser* is considered as the entry-point to the Data-Path; it parses incoming traffic and matches fields in PDRs. (2) - *Traffic Classifier* categorizes traffic by UL (e.g., GRE) and DL (e.g., NON-GRE) flows, as well as other criteria such as

CoT and protocols, using PDRs and matching Packet Detection Information (PDI). (3) - *Traffic Forwarder*, following classification, it directs traffic to its destination: either to the LAN (i.e., End-User) by removing GRE headers or to the WAN (i.e., DC) with appropriate GRE encapsulation, in alignment with the FARs. Finally, (4) - *Traffic Load Balancer*, ensures traffic adheres to predefined LBRs updates, regarding GRE overlay selection, monitoring and dynamically adjusting flows to optimize network KPIs and enhance resource efficiency.

6.3.3 Control Signaling

In this chapter, we define the following key concepts and terms to better understand our data model in Schema 6.1:

- *Flow Rule*: A set of instructions dictating how traffic is handled within SDN architectures. They are crucial for efficient secure traffic management, as they define specific criteria and attributes (such as source and destination IP addresses, protocols, and ports) against which traffic is evaluated for redirection, modification, or dropping.

- *Match Field*: An attribute within the PDR that is used to identify and categorize incoming packets based on predefined criteria. It plays a crucial role in determining how packets are processed by the network, serving as basis element of a flow rule. In addition to the standard criteria, match field can also include a VLAN tag, an MPLS Label, or any other header field such as ToS.

- *Information Element (IE)*: A structured data field used to convey specific types of information within control gRPC signaling. Each IE encapsulates essential data required for the management and control of packet flows, represented as a tuple (type, length, value), where type denotes the information kind (e.g., IP address or protocol), length specifies the data size, and value contains the actual content.

- *PDR*: A set of rules or criteria used to identify and handle packets in a specific manner within the Data-Path. Each PDR defines conditions (i.e., Match Fields) and actions (ie., FAR and LBR) for packet processing.

- *FAR*: Defines the actions to be taken on packets matching a PDR, such as forwarding, buffering, or duplicating packets.

- *LBR*: A rule designed to distribute network traffic across multiple paths or endpoints efficiently, often for LB, redundancy, or failover purposes.

6.3.4 Data-Path Traffic Processing

The in-kernel packet processing journey begins at the NIC Receiver (Rx) queue where the NIC's Direct Memory Access (DMA) triggers a Hardware IRQ (HardIRQ), invoking the NIC Driver's Interrupt Request (IRQ) handler. This handler then initiates the New API (NAPI) subsystem via a Software IRQ (SoftIRQ), starting packet processing via the driver's registered poll function, which implements the XDP hook for eBPF XDP program. To ensure uninterrupted processing, the NIC driver disables further IRQs. The eBPF XDP program, executed by the XDP hook, marks the entry-point for the created XDP pipline. Starting execution, the XDP program accesses a context object metadata, encapsulated within the optimized *xdp_md struct* [86]. Following data parsing, control may transfer to other XDP programs via *bpf_tail_call* function [87]. After parsing, metadata fields can be read from the context object, which also allows attachment of custom metadata [88]. In this regard, XDP programs access persistent data structures (*eBPF*

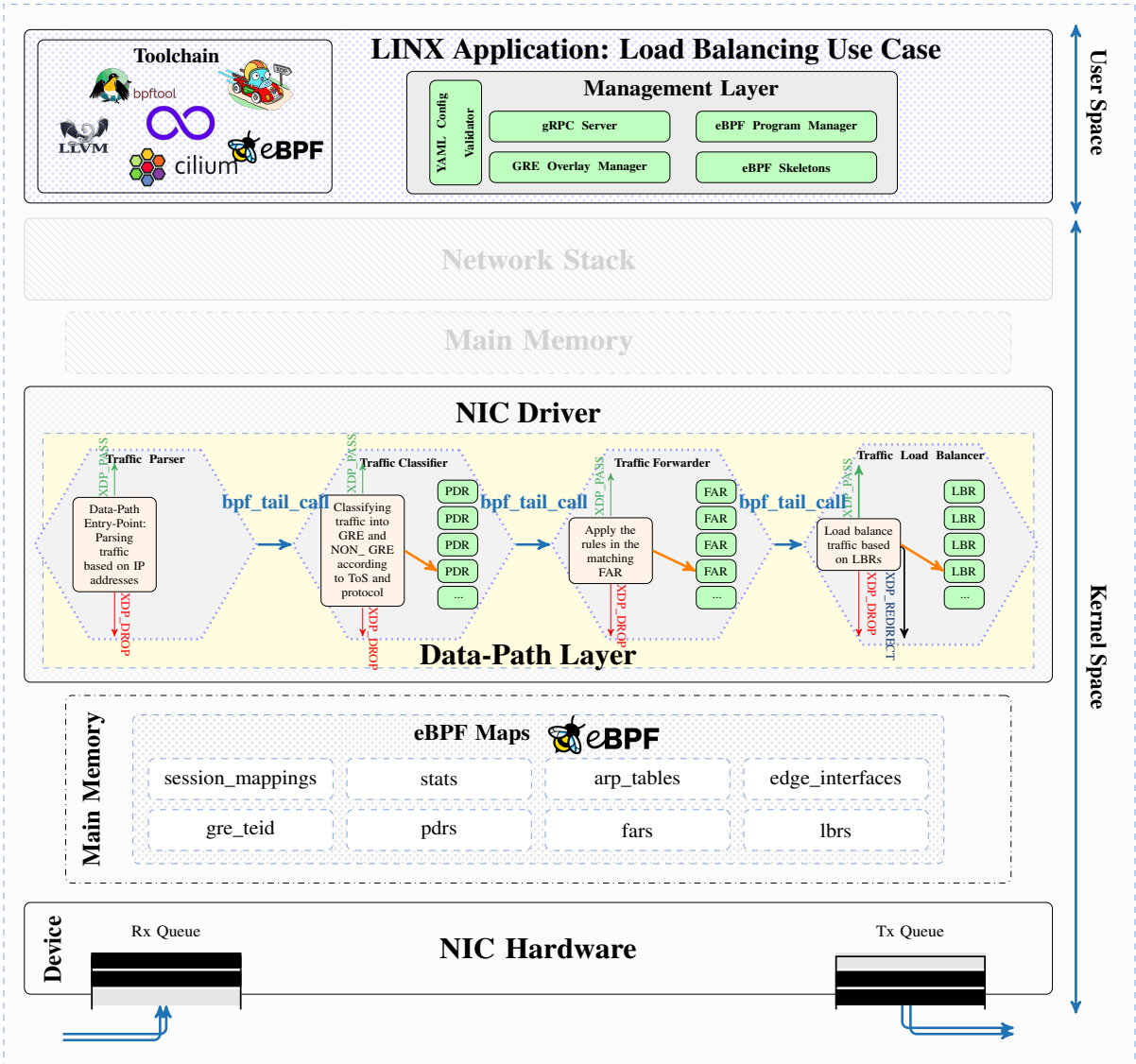


Figure 6.3: LINX architecture details

maps) via functions such as *bpf_map_lookup_elem* or *bpf_map_update_elem* [89, 90]. A final verdict is issued, determining packet handling, such as dropping, re-transmission, kernel processing, or redirection. Once packet processing is completed, the NAPI subsystem is deactivated, and IRQs from the NIC device are re-enabled. This treatment is summarized in Algorithm 6.1.

6.4 METHODOLOGY AND TESTBED

6.4.1 Experimental Setup

We are interested into subjecting our solution to rigorous stress testing to measure its resilience, using RFC 2544-like tests [91]. These tests are designed to evaluate the throughput, latency, and overall performance

Schema 6.1 Flow rules' data-model/grammar

```
request:  operand rules
operand:  create | update | delete
rules:  rule [,rules]
rule:  PDR | FAR | LBR | [IEs]
IEs:  IE [,IEs]
IE: (type, length, value)
length:  u8 | u16 | u32 | u64
value:  CONST (decimal)
typ:  IP address | MAC address | protocol | ToS | action | interface |
interface index
PDR: (IP address, u32, src_ip), (IP address, u32, dst_ip), (protocol, u8,
TCP|User Datagram Protocol (UDP)), (ToS, u8, CONST)
FAR: (action, u8, REDIRECT|PASS|DROP), (interface, u32,
GRE-WAN_1|GRE-WAN_2|..| GRE-WAN_i|NON-GRE), (ifIndex,u32, 1|2|..|i+1)
LBR: FAR, (ToS, u8, CONST)
```

Algorithm 6.1 Data-path packet processing

```
Receive Packet on Rx queue
DMA Triggers HardIRQ
Invoke NIC Driver's IRQ Handler
Initiate NAPI Subsystem (SoftIRQ)
Start Packet Processing via Driver Poll
Trigger XDP pipeline
Packet enters XDP_Parser
bfp_tail_call XDP_Classifier
bfp_tail_call XDP_Forwarder
bfp_tail_call XDP_Load_Balancer
Redirect packet to TX queue
```

of the solution under various conditions. The System Under Test (SUT) consists of a *loopback* between two devices: one hosting a traffic generator such as TRex, and the other, the Device Under Test (DUT), hosting the solution to test, in our case LINX. TRex [92] is an open-source realistic traffic generator powered by DPDK, used to measure the maximum sustainable throughput of a DUT.

6.4.2 Experimental Platform

To evaluate the performance of the HELIOS framework we utilized two identical hosts for the testing environment. One host was configured with TRex traffic generator, while the LINX solution was deployed on the second host. On both devices, we have installed an Ubuntu 22.04 operating system, with kernel version 5.15 to ensure optimal compatibility with eBPF and DPDK. Each device is powered with the 12th Generation of the Intel i7 embedding 12 Cores, clocking at 4.9 GHz, and using 25 MiB of L3 cache memory. Regarding the network interfaces, both hosts were outfitted with Dual-port Intel Ethernet X550T adapters, supporting 10 Gbps on each port, ensuring reliable high-speed data transmission.

6.4.3 Test Cases

We scripted Python test cases to mimic data traffic on both UL and DL scenarios, leveraging TRex APIs [93].

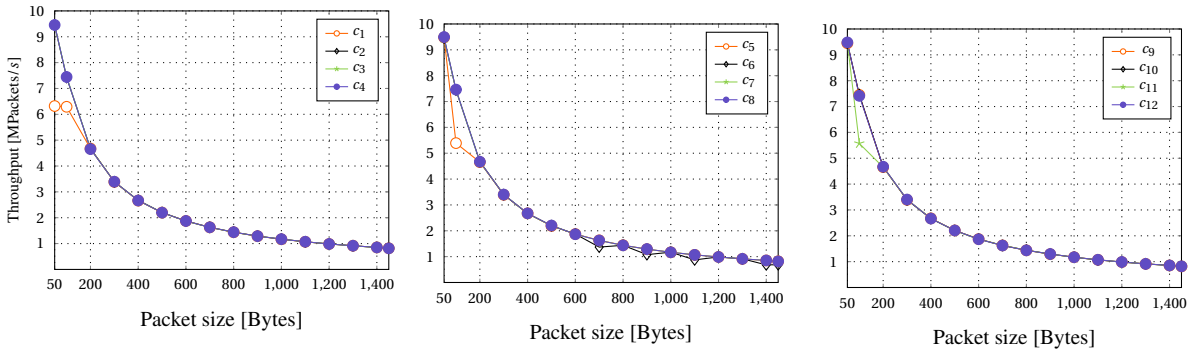
- *UL*: We generate UDP/TCP traffic in TRex, simulating a client sending data via one of the dual-port X550T adapters to the DUT. The LINX gateway, acting as the edge node, processes the traffic, *encapsulates* it in a GRE header, and forwards it to the appropriate WAN interface, as determined by the load balancer based on SDN controller rules. The traffic is looped back over the GRE overlay to TRex on the second port, simulating a remote-region node.
- *DL*: In the DL scenario, the simulated remote-region node, represented by TRex, generates UDP/TCP traffic that is encapsulated in a GRE tunnel and sent to the DUT (i.e., LINX gateway) via the WAN interface. The DUT *decapsulates* the traffic and forwards the resulting packets to the end-user (i.e., TRex) over the LAN through the other port on the Intel X550T adapter, completing the E2E delivery process.
- *Control Signaling*: The ONOS SDN Controller communicates with LINX gateways to dynamically manage and distribute the flow rules. We simulate situations where the SDN needs to update the flow rules within the remote regions, and estimate the delay.

6.5 Performance

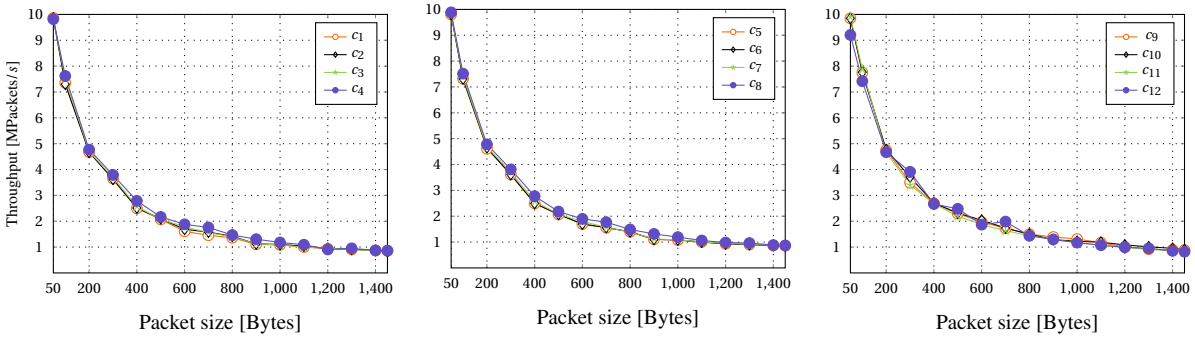
Now, we present the results of our measurement campaign regarding throughput, CPU usage, PLR, and control signaling latency. Across all tests, we analyze the effect of packet injection rate (f_i), packet size (s_p) ($\{100,\ldots,1400\} \cup \{50, 1450\}$), and the number of Rx queues (c_j), where $(i, j, p) \in \{1,2,\ldots,10\} \times \{1,2,\ldots,12\} \times \{1,2,\ldots,16\}$. We used the 'irqbalance' service to distribute incoming traffic evenly across the active Rx Queues, with each Rx Queue pinned to a single CPU core to optimize performance. Each test configuration was repeated 50× to ensure accuracy and consistency.

Throughput: Figure 6.4 shows the *peak* throughput (in MPackets/s) obtained for DL (Figure 6.4a) and UL (Figure 6.4b) scenarios (see Section 6.4.3) function of s_p and c_j. All curves exhibit a similar pattern, decreasing smoothly from 10 MPackets/s and converging toward 1 MPackets/s. This trend resembles an exponential decay function, with throughput gradually declining and approaching an asymptote. The similarity between the UL and DL curves suggests that direction has minimal impact on throughput behavior; however, packet size s_p and, to a lesser extent, the number of Rx Queues c_j, appear to influence the results. Indeed, the more s_p increases, the more throughput decreases, which is intuitive given that throughput is measured in MPackets/s. With a fixed 10 Gbps bandwidth, the maximum achievable throughput is approximately 10 MPackets/s (according to TRex performance) for 50-Byte packets; as packet size grows, the packet rate declines, even though the bandwidth remains fully utilized. The impact of c_j appears minimal with the current SUT, but we anticipate that with a more powerful SUT, this impact will become evident. Transmitting hundreds of millions of packets would more clearly demonstrate the importance of multiple Rx Queues in handling high packet volumes efficiently.

CPU Load: Table 6.1 displays the average CPU usage in percentage as a function of f_i and c_j. The results reveal a trend where, the higher f_i, the greater CPU load will be across all configurations. At higher packet rates, particularly beyond $f4$ (4 MPackets/s), CPU usage nears saturation with a single Rx Queue (i.e., c_1), indicating inefficiency under high packet rates. In contrast, increasing the c_j value *significantly* improves CPU efficiency. For example, in configuration (f_4, c_1), CPU load is 99.23%, reaching 100% in (f_7, c_1) and higher. However, this load drops to 19.19% with only two Rx Queues. With $c_j \geq 6$, CPU load



(a) DL (GRE headers decapsulation)



(b) UL (GRE headers encapsulation)

Figure 6.4: Throughput obtained on the DL and UL per packet size and number of Rx queues

remains below 30% and increases more slowly with further packet injections, showing better efficiency and reduced risk of saturation. Configurations c_8 to c_{12} exhibit the lowest CPU usage, with c_8 at 20.06% and c_{12} at 7.84% for 10 MPackets/s, highlighting the effectiveness of multiple Rx Queues in reducing CPU load even at high packet rates.

Table 6.1: CPU load function of packet rates and Rx queues

Injections [MPackets/s]	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
c_1	7.92	10.79	31.41	99.23	99.33	99.73	100	100	100	100
c_2	2.88	5.56	9.35	19.19	65.15	68.34	98.41	98.87	98.95	99.98
c_3	1.86	3.52	5.74	10.63	15.21	23.22	68.38	76.85	93.11	93.29
c_4	1.82	2.75	5.67	7.47	9.39	12.29	23.45	30.76	83.85	84.49
c_5	1.46	2.48	4.98	4.39	7.78	15.67	24.30	27.51	45.09	45.67
c_6	1.19	2.36	3.44	3.62	6.34	6.35	17.51	18.32	29.56	29.86
c_7	1.12	1.48	2.84	4.68	4.69	5.65	16.14	17.10	24.88	25.17
c_8	1.01	1.84	3.18	3.46	5.38	5.80	15.43	16.88	19.78	20.06
c_9	1.09	1.46	2.45	3.18	3.59	4.07	9.48	10.01	15.98	16.69
c_{10}	1.01	1.28	2.45	3.35	3.64	4.28	8.32	9.62	11.34	11.69
c_{11}	1.20	1.24	2.33	3.16	3.23	5.34	5.98	6.57	8.7	8.9
c_{12}	1.10	1.18	2.12	2.91	3.11	3.27	3.89	4.48	7.28	7.84

Control Signaling Latency: Figure 6.5 depicts the CDF obtained for the control signaling scenario, derived from a dataset of 1,000 values representing the delay ratio achieved in less than $x\,ms$. The results show a low-latency distribution characterized by minimal variation with the following statistical measures: a mean of $0.8155\,ms$, variance of $0.3539\,ms^2$, standard deviation of $0.5949\,ms$, and a coefficient of variation of 0.7295. We find that approximately 50% of CP operations complete within $0.7\,ms$, illustrating that half of all operations complete under $1\,ms$, while the 10^{th} percentile is as low as $0.146\,ms$, indicating that a significant portion of operations are nearly instantaneous. The curve progresses smoothly, with 90% of operations achieving latency below $1.6\,ms$. The upper tail of the CDF reaches $2.6\,ms$, indicating the maximum observed latency, which likely represents rare worst-case delays. Overall, these results indicate reliable low-latency performance ideal for real-time applications requiring immediate rerouting or policy adjustments.

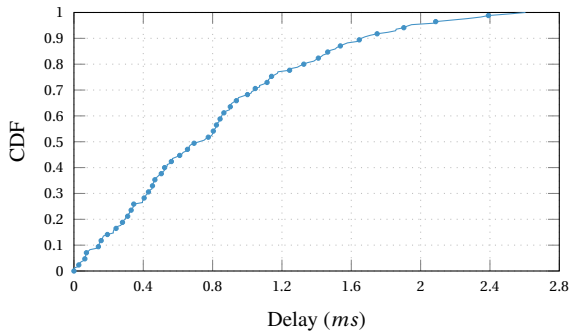


Figure 6.5: Control signaling latency

Packet Loss Rate (PLR): Figure 6.6 presents the PLR (in percentage) plotted against packet size (s_p) and Rx Queue count (c_j) for the UL scenario, where we fully utilize the system's available bandwidth (i.e., 10 Gbps). For each s_p, we have represented 12 PLR average values corresponding to the values of c_j, we thus obtained s_p-dependent c_j-dependent global measures.

We have made several observations from the graph:

(i) - The observed PLR trend resembles an exponential decay function, where the rate of decrease is rapid for smaller packet sizes and gradually levels off for larger packet sizes, approaching an asymptote rather than continuing to decrease linearly.

(ii) - The c_j-dependent PLR values generally converge except for c_1 and c_2. In these cases, the high traffic volume-approximately 10 and 8 MPackets/s, for packet sizes of 50 and 100 Bytes, respectively-overwhelms a single Rx Queue, resulting in increased PLR.

(iii) - With a sufficient number of Rx queues ($c_j \geq 3$), the PLR remains consistently below 0.15% and eventually stabilizes around 0.03%. This value is negligible in comparison to the total number of transmitted packets, indicating highly efficient packet delivery.

(iv) - The PLR is impacted by the packet volume, not size. For instance, with 50-Byte packets, we can generate *in practice* around 10, MPackets/s (based on TRex experience). In contrast, larger packets like 1450 Bytes result in fewer packets per second, reducing the load on each RX queue, since this later can only handle a certain packet rate before overflow.

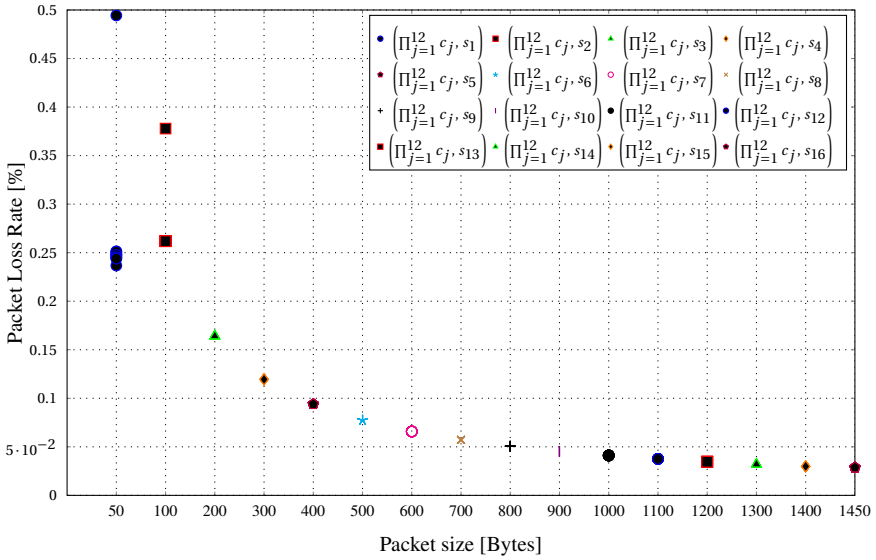


Figure 6.6: PLR per packet size and number of Rx queues

To sum up our findings in this chapter; we first found that eBPF XDP-based solutions are primarily influenced by the volume of injected traffic and the number of Rx Queues, while packet size affects throughput only at the injection phase.

6.6 Conclusion

In this chapter, we introduced the HELIOS architecture, an eBPF-based SD-WAN solution tailored for high-performance applications within CEC networks. We detailed the design of the LINX framework, which implements essential functionalities such as LB and firewalling on Edge Gateways. This solution effectively enables LB, protocol/ToS-aware traffic redirection across multiple GRE overlays, and early-stage filtering and dropping of packets. Our experimental results emphasize the need to reconsider both SD-WAN solutions and other DP components, like the 5G UPF. The findings show significant improvements in throughput, PLR, and resource utilization, underscoring the potential of our solution to enhance network efficiency and performance.

Chapter 7

Conclusions and Perspectives

7.1 Conclusion

In conclusion, this thesis has thoroughly examined the design and optimization of deterministic networks to support ultra-low-latency communications in 6G. The proliferation of time-sensitive applications, such as holographic communication, tactile Internet, autonomous systems, and the metaverse, has introduced unprecedented demands on network infrastructures. These applications necessitate ultra-low latency, high reliability, and predictable performance, ensuring seamless user experiences. Achieving these objectives requires deterministic network behaviors where latency, jitter, and reliability are not only predictable but also bounded. However, traditional network architectures struggle to meet these stringent requirements, particularly in dynamic, large-scale environments characterized by fluctuating traffic patterns and resource availability.

This thesis has addressed these challenges through the development of SDN-based frameworks and solutions that ensure deterministic E2E latency and QoS in next-generation networks. The primary contributions include innovative mechanisms for dynamic queuing and intelligent path selection, predictive traffic management and AC using GNNs, advanced congestion control leveraging L4S protocol, and kernel-level packet processing with eBPF. These contributions collectively enhance the SDN-enabled DP to meet the stringent requirements of URLLC in 6G networks.

The proposed solutions were rigorously evaluated using both virtual prototypes and physical network deployments. In particular, the final contribution, involving eBPF-based CEC interconnection, was tested on a physical platform (see Subsection 6.4.2). The evaluations focused primarily on the TN, including the backhaul and midhaul, and on CEC interconnection nodes. Results demonstrated significant reductions in latency, improvements in resource efficiency, and scalability under dynamic traffic loads. These findings validate the effectiveness of the proposed solutions, as documented through multiple publications in leading IEEE conferences. This work lays a solid foundation for SDN-enabled deterministic, low-latency communication in 6G, addressing the critical challenges of dynamic traffic management, path selection, predictable network KPIs, AC, congestion control, and eBPF-based edge node processing.

7.2 Future Perspectives

The future perspectives of this research are broad and multifaceted, encompassing further improvements to the proposed solutions and exploring several critical areas to enhance the capabilities and applications of deterministic networks in 6G. These perspectives aim to address emerging challenges, extend the applicability of the research, and ensure practical implementation.

7.2.1 Advanced ML Integration

The integration of advanced ML techniques within SDN frameworks holds immense potential for enhancing network management and decision-making processes. This thesis demonstrated the utility of GNNs for predictive traffic management; however, the incorporation of other AI-driven methodologies, such as Reinforcement Learning (RL) and DL, could further elevate the network's adaptability and efficiency.

Future research can also extend ML integration to other contributions in this thesis, such as dynamic queuing and congestion control mechanisms. AI-enhanced systems could predict traffic spikes and reconfigure network resources proactively, minimizing delays and optimizing performance. For example, RL algorithms, particularly Deep Q-Learning (DQN) or Proximal Policy Optimization (PPO) [94], could dynamically optimize RA based on real-time network states. These approaches could enable continuous learning and adaptation, allowing networks to self-adjust in response to fluctuating traffic patterns. DL, on the other hand, could leverage extensive datasets from network telemetry to uncover hidden trends and correlations. Models such as Transformer-based architectures could process multi-modal data, improving congestion prediction accuracy and enabling proactive resource management.

Future work could also explore the convergence of federated learning techniques with SDN-based frameworks. By decentralizing the learning process, federated ML can enhance privacy and scalability, enabling distributed decision-making across edge nodes without centralizing sensitive data. Such advancements would ensure robust, scalable, and intelligent network management tailored to evolving 6G requirements.

7.2.2 Edge Computing and IoT Integration

The proliferation of IoT devices and edge computing architectures offers a promising frontier for future research. This thesis touched upon leveraging eBPF-based CEC interconnection, which could be further enhanced to accommodate diverse IoT scenarios, including smart cities, industrial IoT, and autonomous systems.

Edge computing's ability to perform localized processing can significantly reduce communication latency and improve reliability, especially for URLLC applications. Advanced algorithms for resource orchestration at edge nodes could be developed, considering dynamic workload distribution, energy efficiency, and real-time constraints. For instance, integrating edge AI with SDN could enable real-time anomaly detection, resource prediction, and context-aware QoS provisioning.

Moreover, the interplay between IoT and deterministic networking principles opens avenues for securing E2E communication. Future research could focus on extending the proposed solutions to validate their applicability in IoT ecosystems, ensuring seamless interconnection and reliable operation in large-scale deployments.

7.2.3 Extension to Other Network Segments

While the thesis primarily addressed deterministic networking in the TN, extending these principles to the RAN and CN is critical for achieving a fully deterministic E2E communication framework. Tackling bottlenecks and ensuring consistent QoS across these additional segments is essential for supporting the stringent requirements of time-sensitive applications [95].

In the RAN, SDN-based solutions can significantly enhance traffic prioritization and QoS management, especially for use cases such as eMBB, URLLC, and mMTC. Dynamic spectrum allocation, supported by

SDN, can optimize resource utilization by enforcing real-time traffic prioritization policies for each slice. Advanced SDN-driven schedulers can prioritize traffic based on latency, jitter, and packet loss requirements, ensuring deterministic performance. These improvements can be validated through simulations or RAN testbeds to demonstrate their effectiveness under diverse conditions.

For the CN, deterministic networking can leverage advanced traffic engineering techniques. Segment Routing (e.g., Segment Routing IPv6 (SRv6)) enables granular, QoS-aware route computation, ensuring low-latency paths for HP traffic. SDN controllers can dynamically enforce QoS policies, such as adjusting buffer sizes or reconfiguring VNFs, to optimize traffic flows. These mechanisms allow the network to adapt to evolving application requirements and traffic patterns, particularly for bandwidth-intensive and latency-critical services.

A crucial aspect of extending deterministic networking is synchronization and seamless coordination between the RAN, TN, and CN. Establishing unified SDN control planes across these segments can enable E2E traffic orchestration and consistent QoS enforcement. Algorithms to dynamically mitigate cross-segment bottlenecks can further ensure smooth data flow and reduce latency. Global resource optimization by SDN controllers can maximize network efficiency while meeting stringent E2E requirements.

By building on the SDN-based solutions developed for the TN, this research provides a foundation for designing unified frameworks that address multi-segment challenges and align with the vision of 6G, where ultra-low latency and high reliability are paramount.

7.2.4 Real-World Deployment and Testing

Real-world validation is a critical step toward bridging the gap between theoretical frameworks and practical implementation. Deploying the proposed solutions in live 5G/6G environments and conducting comprehensive field trials would provide invaluable insights into their performance under diverse conditions.

For instance, testing the solutions with heterogeneous traffic types, such as interactive cloud gaming, IoT telemetry, and real-time video streaming, would reveal their adaptability and robustness. Specific emphasis should be placed on evaluating solutions for both UL and DL traffic flows, ensuring balanced performance across diverse scenarios.

Moreover, integrating QFIs into practical deployments could refine QoS provisioning mechanisms. Mapping QFIs to different service classes and analyzing their impact on performance metrics such as latency, jitter, and packet loss would enable a granular understanding of the frameworks' effectiveness in meeting diverse application demands.

7.2.5 Emerging Applications and Use Cases

The rapid evolution of emerging applications, such as digital twins [96], Extended Reality (XR), and Industry 4.0, provides compelling use cases for deterministic networking principles. These applications demand ultra-low latency, high reliability, and scalable solutions, aligning well with the frameworks proposed in this thesis.

Digital twins, for example, require seamless synchronization between physical and virtual entities. Deterministic networking can play a pivotal role in ensuring the timely delivery of updates and maintaining

fidelity between the twin systems. Similarly, XR applications, encompassing VR and AR, could benefit from E2E latency guarantees and adaptive bandwidth allocation, ensuring an immersive user experience.

Industry 4.0, characterized by smart manufacturing and autonomous systems, presents additional challenges, such as managing massive IoT deployments and maintaining strict QoS requirements. Future research could explore the intersection of SDN, TSN, and edge computing in addressing these challenges, focusing on scalability, interoperability, and fault tolerance.

In conclusion, the research presented in this thesis provides a robust foundation for the development of SDN-enabled deterministic networking solutions for 6G. The proposed solutions address critical challenges and pave the way for future advancements in network performance, scalability, and application support. The outlined future perspectives highlight the potential for further innovations, driving the evolution of next-generation networks and enabling the seamless integration of time-sensitive applications.

Bibliography

[1] Shuping Dang et al. “What should 6G be?” In: *Nature Electronics* 3.1 (2020), pp. 20–29.

[2] José Suárez-Varela et al. “Graph neural networks for communication networks: Context, use cases and opportunities”. In: *IEEE network* 37.3 (2022), pp. 146–153.

[3] K De Schepper, M Bagnulo, and G White. *RFC 9330: Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture*. 2023.

[4] Anant Deepak, Richard Huang, and Puneet Mehra. “eBPF/XDP based firewall and packet filtering”. In: *Proc. Linux Plumbers Conf.* 2018, pp. 1–5.

[5] Karim Boutiba et al. “NRflex: Enforcing network slicing in 5G New Radio”. In: *Comput. Commun.* 181 (2022), pp. 284–292.

[6] Xiyang Yin et al. “QoS Flow Mapping Method of Multi-service 5G Communication for Urban Energy Interconnection”. In: *2021 International Conference on Wireless Communications and Smart Grid (ICWCSG)*. 2021, pp. 75–78. DOI: 10.1109/ICWCSG53609.2021.00022.

[7] Miquel Ferriol-Galmés et al. “RouteNet-Fermi: Network Modeling with Graph Neural Networks”. In: *arXiv preprint arXiv:2212.12070* (2022).

[8] Reza Poorzare and Anna Calveras Augé. “Challenges on the way of implementing TCP over 5G networks”. In: *IEEE access* 8 (2020), pp. 176393–176415.

[9] Akhil Gupta and Rakesh Kumar Jha. “A survey of 5G network: Architecture and emerging technologies”. In: *IEEE access* 3 (2015), pp. 1206–1232.

[10] Ali Sufyan et al. “From 5G to beyond 5G: A comprehensive survey of wireless network evolution, challenges, and promising technologies”. In: *Electronics* 12.10 (2023), p. 2200.

[11] Geeksforgeeks. *5G Network Architecture*. 2023. URL: https://www.geeksforgeeks.org/5g-network-architecture/. (accessed: 18.10.2023).

[12] Trung-Kien Le, Umer Salim, and Florian Kaltenberger. “An overview of physical layer design for ultra-reliable low-latency communications in 3GPP releases 15, 16, and 17”. In: *IEEE access* 9 (2020), pp. 433–444.

[13] Xingqin Lin. “An overview of 5G advanced evolution in 3GPP release 18”. In: *IEEE Communications Standards Magazine* 6.3 (2022), pp. 77–83.

[14] Takahito Yoshizawa, Sheeba Backia Mary Baskaran, and Andreas Kunz. “Overview of 5G URLLC system and security aspects in 3GPP”. In: *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*. IEEE. 2019, pp. 1–5.

[15] Jeong Wook Won and Jae Min Ahn. “3GPP URLLC patent analysis”. In: *ICT Express* 7.2 (2021), pp. 221–228.

[16] Md Emdadul Haque et al. “A survey of scheduling in 5g urllc and outlook for emerging 6g systems”. In: *IEEE access* 11 (2023), pp. 34372–34396.

[17] Jerome Henry, Tim Szigeti, and Luis M. Contreras. *Diffserv to QCI Mapping*. Internet-Draft draft-henry-tsvwg-diffserv-to-qci-04. Work in Progress. Internet Engineering Task Force, Apr. 2020. 43 pp. URL: https://datatracker.ietf.org/doc/draft-henry-tsvwg-diffserv-to-qci/04/.

[18] 3GPP. *Service Requirements for Next Generation New Services and Markets*. Technical Specification 22.261. Version v15.2.0. 3rd Generation Partnership Project (3GPP), Sept. 2017. URL: https://www.3gpp.org/DynaReport/22261.htm.

[19] Stream Reservation Protocol SRP. "Bridges and Bridged Networks". In: ().

[20] Luxi Zhao, Paul Pop, and Silviu S. Craciunas. "Worst-Case Latency Analysis for IEEE 802.1Qbv Time Sensitive Networks Using Network Calculus". In: *IEEE Access* 6 (2018), pp. 41803–41815. DOI: 10.1109/ACCESS.2018.2858767. URL: https://doi.org/10.1109/ACCESS.2018.2858767.

[21] Zifan Zhou et al. "Simulating TSN traffic scheduling and shaping for future automotive Ethernet". In: *J. Commun. Networks* 23.1 (2021), pp. 53–62. DOI: 10.23919/JCN.2021.000001. URL: https://doi.org/10.23919/JCN.2021.000001.

[22] Wen-Kang Jia, Gen-Hen Liu, and Yaw-Chung Chen. "Performance evaluation of IEEE 802.1 Qbu: Experimental and simulation results". In: *38th Annual IEEE Conference on Local Computer Networks*. IEEE. 2013, pp. 659–662.

[23] Luis Silva et al. "On the adequacy of SDN and TSN for Industry 4.0". In: *2019 IEEE 22nd international symposium on real-time distributed computing (ISORC)*. IEEE. 2019, pp. 43–51.

[24] Qingyue Long et al. "Software defined 5G and 6G networks: A survey". In: *Mobile networks and applications* 27.5 (2022), pp. 1792–1812.

[25] Yuan Zhang et al. "A survey on software defined networking with multiple controllers". In: *Journal of Network and Computer Applications* 103 (2018), pp. 101–118.

[26] ONF. "Software-Defined Networking: The New Norm for Networks". In: (Apr. 2012).

[27] Eric Rosen, Arun Viswanathan, and Ross Callon. *RFC3031: Multiprotocol label switching architecture*. 2001.

[28] Stephen Kent and Karen Seo. *RFC 4301: Security architecture for the Internet protocol*. 2005.

[29] Dino Farinacci et al. *RFC2784: Generic routing encapsulation (GRE)*. 2000.

[30] How Specifically Do You Expect Standardization and Will Drive SD-WAN Market Growth. "MEF Leads SD-WAN Service Standardization & Certification". In: ().

[31] Mohamed Lahby and Abderrahim Sekkaki. "A Graph Theory Based Network Selection Algorithm in Heterogeneous Wireless Networks". In: *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. 2018, pp. 1–4. DOI: 10.1109/NTMS.2018.8328670.

[32] Eko Kuncoro Adiyanto, Sri Wahjuni, and Hendra Rahmawan. "Modification of Load Calculation in The Dijkstra Algorithm to Achieve High Throughput and Low Latency on 5G Networks". In: *Journal of Applied Engineering and Technological Science (JAETS)* 5.2 (2024), pp. 1182–1198.

[33] Somaya A Aboulrous et al. "Parallel multi-hop routing protocol for 5G backhauling network using HPC platform". In: *2020 15th International Conference on Computer Engineering and Systems (ICCES)*. IEEE. 2020, pp. 1–6.

[34] Rakesh Kumar et al. "End-to-End Network Delay Guarantees for Real-Time Systems Using SDN". In: *2017 IEEE Real-Time Systems Symposium, RTSS 2017, Paris, France, December 5-8, 2017*. IEEE Computer Society, 2017, pp. 231–242.

[35] Yajing Zhang et al. "QoS-Aware Mapping and Scheduling for Virtual Network Functions in Industrial 5G-TSN Network". In: *2021 IEEE Global Communications Conference (GLOBECOM)*. 2021, pp. 1–6. DOI: 10.1109/GLOBECOM46510.2021.9685749.

[36] Aiman Nait Abbou, Tarik Taleb, and JaeSeung Song. "A Software-Defined Queuing Framework for QoS Provisioning in 5G and Beyond Mobile Systems". In: *IEEE Netw.* 35.2 (2021), pp. 168–173.

[37] Patricia Thaler et al. *IEEE 802.1 Q*. 2013.

[38] Wesley da Silva Coelho. "Modeling and optimization of 5G network design". In: *2021 33th International Teletraffic Congress (ITC-33)*. 2021, pp. 1–3.

[39] Gengbiao Shen et al. “Modeling and optimization of the data plane in the SDN-based DCN by queuing theory”. In: *J. Netw. Comput. Appl.* 207 (2022), p. 103481. DOI: 10.1016/j.jnca.2022.103481. URL: https://doi.org/10.1016/j.jnca.2022.103481.

[40] Zhiyuan Xu et al. “Experience-driven Networking: A Deep Reinforcement Learning based Approach”. In: *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*. IEEE, 2018, pp. 1871–1879. DOI: 10.1109/INFOCOM.2018.8485853. URL: https://doi.org/10.1109/INFOCOM.2018.8485853.

[41] Shihan Xiao, Dongdong He, and Zhibo Gong. “Deep-Q: Traffic-driven QoS Inference using Deep Generative Network”. In: *Proceedings of the 2018 Workshop on Network Meets AI & ML, NetAI@SIGCOMM 2018, Budapest, Hungary, August 24, 2018*. ACM, 2018, pp. 67–73. DOI: 10.1145/3229543.3229549. URL: https://doi.org/10.1145/3229543.3229549.

[42] Albert Mestres et al. “Understanding the Modeling of Computer Network Delays using Neural Networks”. In: *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, Big-DAMA@SIGCOMM 2018, Budapest, Hungary, August 20, 2018*. Ed. by Pedro Casas et al. ACM, 2018, pp. 46–52. DOI: 10.1145/3229607.3229613. URL: https://doi.org/10.1145/3229607.3229613.

[43] James Roberts et al. *A comparison of sdn based tcp congestion control with tcp reno and cubic*. Tech. rep. Technical Report, 2016.

[44] Richard J La, Jean Walrand, and Venkatachalam Anantharam. *Issues in TCP vegas*. Citeseer, 1999.

[45] Nicolas Kuhn, Emmanuel Lochin, and Olivier Mehani. “Revisiting old friends: is CoDel really achieving what RED cannot?” In: *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop*. 2014, pp. 3–8.

[46] K De Schepper, M Bagnulo, and G White. *RFC 9330: Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture*. 2023.

[47] ITU-T Network 2030. “A Blueprint of Technology, Applications and Market Drivers Towards the Year 2030 and Beyond”. In: (2019). URL: https://www.itu.int/en/ITU-T/focusgroups/net2030/Documents/White_Paper.pdf.

[48] Adlen Ksentini et al. “Providing Low Latency Guarantees for Slicing-Ready 5G Systems via Two-Level MAC Scheduling”. In: *IEEE Netw.* 32.6 (2018), pp. 116–123. DOI: 10.1109/MNET.2018.1800005. URL: https://doi.org/10.1109/MNET.2018.1800005.

[49] Yuki Goto et al. “Queueing analysis of software defined network with realistic OpenFlow-based switch model”. In: *Comput. Networks* 164 (2019).

[50] Jonathan Prados-Garzon, Tarik Taleb, and Miloud Bagaa. “LEARNET: Reinforcement Learning Based Flow Scheduling for Asynchronous Deterministic Networks”. In: *2020 IEEE International Conference on Communications, ICC 2020, Dublin, Ireland, June 7-11, 2020*. IEEE, 2020, pp. 1–6.

[51] 3GPP TS 23.501. “System Architecture for the 5G System, V16.1.0”. In: (2019).

[52] Noa Zilberman et al. “NetFPGA SUME: Toward 100 Gbps as Research Commodity”. In: *IEEE Micro* 34 (Sept. 2014), pp. 32–41.

[53] Dimitar Minovski et al. “Throughput Prediction Using Machine Learning in LTE and 5G Networks”. In: *IEEE Transactions on Mobile Computing* 22.3 (2023), pp. 1825–1840. DOI: 10.1109/TMC.2021.3099397.

[54] Lisa Maile, Kai-Steffen J. Hielscher, and Reinhard German. “Delay-Guaranteeing Admission Control for Time-Sensitive Networking Using the Credit-Based Shaper”. In: *IEEE Open Journal of the Communications Society* 3 (2022), pp. 1834–1852. DOI: 10.1109/OJCOMS.2022.3212939.

[55] José Suárez-Varela et al. “Technical report: RouteNet-Fermi”. In: (2022). URL: https://bnn.upc.edu/download/technical_report_routenet_fermi.

[56] José Suárez-Varela et al. “The graph neural networking challenge: a worldwide competition for education in AI/ML for networks”. In: *ACM SIGCOMM Computer Communication Review* 51.3 (2021), pp. 9–16.

[57] András Varga and Rudolf Hornig. "An overview of the OMNeT++ simulation environment". In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, SimuTools 2008, Marseille, France, March 3-7, 2008*. Ed. by Sándor Molnár et al. ICST/ACM, 2008, p. 60. DOI: 10.4108/ICST.SIMUTOOLS2008.3027. URL: https://doi.org/10.4108/ICST.SIMUTOOLS2008.3027.

[58] José Suárez-Varela et al. "Graph Neural Networks for Communication Networks: Context, Use Cases and Opportunities". In: *CoRR* abs/2112.14792 (2021). arXiv: 2112.14792. URL: https://arxiv.org/abs/2112.14792.

[59] Krzysztof Rusek et al. "RouteNet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN". In: *IEEE J. Sel. Areas Commun.* 38.10 (2020), pp. 2260–2270. DOI: 10.1109/JSAC.2020.3000405. URL: https://doi.org/10.1109/JSAC.2020.3000405.

[60] Zhun Ge, Jiacheng Hou, and Amiya Nayak. "Forecasting SDN End-to-End Latency Using Graph Neural Network". In: *2023 International Conference on Information Networking (ICOIN)*. 2023, pp. 293–298. DOI: 10.1109/ICOIN56518.2023.10048915.

[61] DongNyeong Heo et al. "Graph Neural Network based Service Function Chaining for Automatic Network Control". In: *21st Asia-Pacific Network Operations and Management Symposium, APNOMS 2020, Daegu, South Korea, September 22-25, 2020*. IEEE, 2020, pp. 7–12. DOI: 10.23919/APNOMS50412.2020.9236954. URL: https://doi.org/10.23919/APNOMS50412.2020.9236954.

[62] Farouk Messaoudi et al. "Performance Analysis of Game Engines on Mobile and Fixed Devices". In: *ACM Trans. Multim. Comput. Commun. Appl.* 13.4 (2017), 57:1–57:28.

[63] James F. Kurose and Keith W. Ross. *Computer networking - a top-down approach featuring the internet*. Addison-Wesley-Longman, 2001. ISBN: 978-0-201-47711-5.

[64] M Bagnulo and G White. "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture". In: (2023).

[65] Sally Floyd, Dr. K. K. Ramakrishnan, and David L. Black. *The Addition of Explicit Congestion Notification (ECN) to IP*. RFC 3168. Sept. 2001.

[66] Pat Bosshart et al. "P4: programming protocol-independent packet processors". In: *Comput. Commun. Rev.* 44.3 (2014), pp. 87–95.

[67] Philip Alexander Levis. "TinyOS: An Open Operating System for Wireless Sensor Networks (Invited Seminar)". In: *7th International Conference on Mobile Data Management (MDM 2006), Nara, Japan, May 9-13, 2006*. IEEE Computer Society, 2006, p. 63.

[68] Farouk Messaoudi, Adlen Ksentini, and Philippe Bertin. "Toward a Mobile Gaming Based-Computation Offloading". In: *2018 IEEE International Conference on Communications, ICC 2018, Kansas City, MO, USA, May 20-24, 2018*. IEEE, 2018, pp. 1–7.

[69] ETSI. "Mobile edge computing (MEC); Framework and Reference Architecture". In: *ETSI GS MEC* 3 (Mar. 2016), p. V1.1.1.

[70] A. Abdelsalam et al. "Linux MP-TCP performance evaluation in a combined terrestrial-satellite access". In: *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*. 2019.

[71] Rami Rosen. *Linux kernel networking: Implementation and theory*. Apress, 2014.

[72] Huu Nghia Nguyen et al. "A Comprehensive P4-based Monitoring Framework for L4S leveraging In-band Network Telemetry". In: *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. 2023, pp. 1–6. DOI: 10.1109/NOMS56928.2023.10154331.

[73] Ezekia Gilliard et al. "Explicit Congestion Notification-Based Congestion Control Algorithm for High-Performing Data Centers". In: *2023 IEEE AFRICON*. 2023, pp. 1–6. DOI: 10.1109/AFRICON55910.2023.10293272.

[74] Jangwoo Son et al. "L4S Congestion Control Algorithm for Interactive Low Latency Applications over 5G". In: *2023 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2023, pp. 1002–1007.

[75] Mirja Kühlewind, Richard Scheffenegger, and Bob Briscoe. *Problem Statement and Requirements for Increased Accuracy in Explicit Congestion Notification (ECN) Feedback*. RFC 7560. Aug. 2015.

[76] Randall R. Stewart. *Stream Control Transmission Protocol*. RFC 4960. Sept. 2007.

[77] Jana Iyengar and Martin Thomson. *QUIC: A UDP-Based Multiplexed and Secure Transport*. RFC 9000. May 2021.

[78] *RTP Control Protocol (RTCP) Feedback for Congestion Control*. RFC 8888. Jan. 2021.

[79] Mark Handley et al. *TCP friendly rate control (TFRC): Protocol specification*. Tech. rep. 2003.

[80] Paul Hurley, Jean-Yves Le Boudec, and Patrick Thiran. *A note on the fairness of additive increase and multiplicative decrease*. Tech. rep. 1998.

[81] Haining Wang et al. "A simple refinement of slow-start of TCP congestion control". In: *Proceedings ISCC 2000. Fifth IEEE Symposium on Computers and Communications*. IEEE. 2000, pp. 98–105.

[82] Hanaa Torkey, Gamal Attiya, and Ahmed Abdel Nabi. "An efficient congestion control protocol for wired/wireless networks". In: *International Journal of Electronics Communication and Computer Engineering* 5.1 (2014), p. 77.

[83] Sofiane MESSAOUDI, Adlen Ksentini, and Christian BONNET. "SDN Framework for QoS provisioning and latency guarantee in 5G and beyond". In: *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*. IEEE. 2023, pp. 587–592.

[84] Koen De Schepper and G White. *RFC 9332: Dual-Queue Coupled Active Queue Management (AQM) for Low Latency, Low Loss, and Scalable Throughput (L4S)*. 2023.

[85] Ayoub Mokhtari and Adlen Ksentini. "SD-WAN for Cloud Edge Computing Continuum interconnection". In: *Proceedings of IEEE Globecom 2024, Cap Town, South Africa*. 2024.

[86] Marcos Augusto M. Vieira et al. "Fast Packet Processing with eBPF and XDP: Concepts, Code, Challenges, and Applications". In: *ACM Comput. Surv.* 53.1 (2021), 16:1–16:36.

[87] eBPF. *bpf_tail_call*. [online] available at: https://docs.ebpf.io/linux/helper-function/bpf_tail_call/ (last checked: Oct. 2024).

[88] Toke Høiland-Jørgensen et al. "The eXpress data path: fast programmable packet processing in the operating system kernel". In: *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies, CoNEXT 2018, Heraklion, Greece, December 04-07, 2018*. Ed. by Xenofontas A. Dimitropoulos et al. ACM, 2018, pp. 54–66.

[89] eBPF. *bpf_map_lookup_elem*. [online] available at: https://docs.ebpf.io/linux/helper-function/bpf_map_lookup_elem/ (last checked: Oct. 2024).

[90] eBPF. *bpf_map_update_elem*. [online] available at: https://docs.ebpf.io/linux/helper-function/bpf_map_update_elem/ (last checked: Oct. 2024).

[91] S. Bradner and J. McQuaid. *Benchmarking Methodology for Network Interconnect Devices*. RFC 7560. 1999.

[92] Cisco. *TRex Stateless API Reference*. [online] available at: https://trex-tgn.cisco.com/ (last checked: Mai 2024). 2024.

[93] Cisco. *TRex Stateless API Reference*. [online] available at: https://trex-tgn.cisco.com/trex/doc/cp_stl_docs/api/index.html (last checked: Mai 2024). 2015.

[94] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[95] Leonardo Bonati et al. "Open, programmable, and virtualized 5G networks: State-of-the-art and the road ahead". In: *Computer Networks* 182 (2020), p. 107516.

[96] Maulshree Singh et al. "Digital twin: Origin to future". In: *Applied System Innovation* 4.2 (2021), p. 36.