GNN-based SDN Admission Control in Beyond 5G Networks

Sofiane Messaoudi*, Adlen Ksentini*, Franck Messaoudi†, Christian Bonnet*
*Eurecom Institute, †OpenAirInterface Alliance
* †Sophia Antipolis, France
Email: *name.surname@eurecom.fr, †name.surname@openairinterface.org

***Abstract*—This paper proposes a novel approach to Software-Defined Networking (SDN) Admission Control (AC) based on Graph Neural Networks (GNNs) for Beyond 5G (B5G). AC is a critical function in SDN, as it determines which traffic flow to pass by the network and which should be rejected. GNNs are a type of Neural Networks (NNs) that are able to learn how to make real-time AC decisions by training on pre-existing data, including network topologies and traffic characteristics. The solution we propose is made of two layers: (i) Network Delay Predictor (NetDelP) leveraging on the RouteNet-Fermi GNN model, used to predict the network latency for different topologies and traffic patterns. (ii) Admission Control Agent (AdConAgt) supporting the SDN and used to regulate the traffic flow in the network. The outlined concept is able to manage large-scale and complex topology networks with optimized Key Performance Indicators (KPIs) such as network latency and Packet Loss Rate (PLR). The envisioned approach is evaluated with various network topology scales and classes of traffic. The obtained results outperform the SDN-Shortest Path (SDN-SP) solution by demonstrating the ability of our proposal to guarantee the End-To-End (E2E) latency and prevent link congestion in order to meet the QoS requirements.**

I. Introduction

The emerging Beyond 5G (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 5^{th} Generation (5G) architecture is the support of different use cases and applications such as e-health, autonomous cars, and metaverse [1]. Their coexistence will depend on B5G networks' ability to serve these classes of traffic, having *different needs* in terms of Quality of Service (QoS) that include latency, bandwidth, reliability, and availability. Software-Defined Networking (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 Network Model (NM), and the Optimisation Algorithm (OA) [2]. The former predicts the network Key Performance Indicators (KPIs), such as delay and throughput; it features approaches such as Network Prediction (NP) [3]. The latter explores different configurations until it meets the optimization goals defined in the NM, which include solutions based on Time Sensitive Networking (TSN) standards [4] such as Admission Control (AC) [5], and Traffic Engineering (TE) [6], to name few. Although TSN is an interesting approach, it provides hop-by-hop QoS guarantee; high End-To-End (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. Indeed, Analytic models based on Queuing Theory (QT) [7] 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, Deep Learning (DL) [8] models are proposed. Existing solutions such as [9], [10] use the Neural Network (NN) architectures like Recurrent NNs or Variational AutoEncoders. 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 paper a solution that combines Graph Neural Network (GNN) and AC, named *Graph Neural Network-Admission Control (GNN-AC)*. It leverages the RouteNet-Fermi (RouteNet-F) model [11], [12] as a real-time Latency Prediction (LP) framework for each path, that is to achieve Load Balancing (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 classes of traffic running in competition mode. GNN-AC relies on the new 5G Data Plane (DP) specifications. It uses the Quality Flow Identification (QFI) [13] to map the 5G services to

Transport Network (TN) classes of traffic. 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++ [14]. We compared our solution with the Open Network Operating System (ONOS) SDN controller' default one (i.e., SDN-Shortest Path (SDN-SP)). The rest of the paper is organized as follows: Section II provides background on GNN, RouteNet-F, and reviews related state-of-the-art. Our solution is presented in Section III and evaluated in Section IV. Section V concludes the paper.

II. Background & Related Works

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

GNNs have shown outstanding applications in communication networks [15]. 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 NN 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 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 LP, AC, and QoS guarantee. Indeed, in [16], 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.

[17], 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 machine learning techniques.

[18] 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 NN (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.

III. 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.

A. 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.

In 5G, a User Equipment (UE) traffic has to pass through dedicated bi-directional tunnels or bearers to interact with the outside. These tunnels are created by the Session Management Function (SMF) Packet Data Unit (PDU) session establishment request. The bearer encapsulates UE' Internet Protocol (IP) packets in GPRS Tunneling Protocol (GTP) tunnels between the gNodeB (gNB) and the User Plane Function (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 5G QoS Identifier (5QI) that classifies packets into different classes of traffic, each with specific QoS characteristics that include resource type, priority level, and Packet Delay Budget (PDB). There are approximately two dozen standard 5QI values grouped into two types of resources: Guaranteed Bit Rate (GBR) and Non-Guaranteed Bit Rate (Non-GBR). The QFI value is assigned by the 5G Core Network (CN) based on the UE subscription and the service to run, making it critical when carrying tunnel' traffic over 5G TN or Backhaul.

Figure 1 shows the positioning of GNN-AC on the 5G architecture. The GNN-AC is on top of an SDN controller. The SDN control plane 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 identifier values known as *Differentiated Services Code Point (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. Internet Engineering Task Force (IETF) has proposed a mapping of the 3rd Generation Partnership Project (3GPP) QoS Class Identifier (QCI) and the 5QI to the DSCP, which aligns their marking recommendations, as stated in [19]. 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 Uplink and Downlink directions.

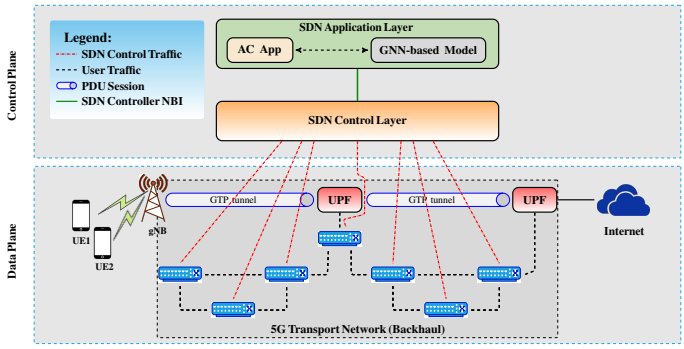

Figure 1: Simplified 5G architecture with SDN perspective

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 1 is triggered as a response. In what follows, we describe the GNN-AC architecture design along with the workflow.

B. GNN-AC Solution Design

GNN-AC framework is made of two main modules; (i) *Network Delay Predictor (NetDelP)*, a network LP module leveraging the RouteNet-F GNN-based model, and (ii) *Admission Control Agent (AdConAgt)*, an AC module supporting SDN.

1) NetDelP: Is the module predicting, *in real-time*, the delay on the entire network topology paths using RouteNet-F GNN-based modelling. 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.

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 Create, Read, Update, and Delete (CRUD) the flow rules.

Figure 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 Open Networking Foundation (ONF) standards [20].

Application Layer: hosts the GNN model (i.e., NetDelP) and the SDN application (i.e., AdConAgt), which communicates its network requirements toward the control plane via the NorthBound Interfacess (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 Application Programming Interface (API). As a client, it monitors the network traffic and elements.

Infrastructure Layer: Or data plane, is a composition of network elements, switches such as Open vSwitch (OVS), which expose their capabilities toward the SDN controller via the SouthBound Interfacess (SBIs) and apply the forwarding rules as computed by the controller.

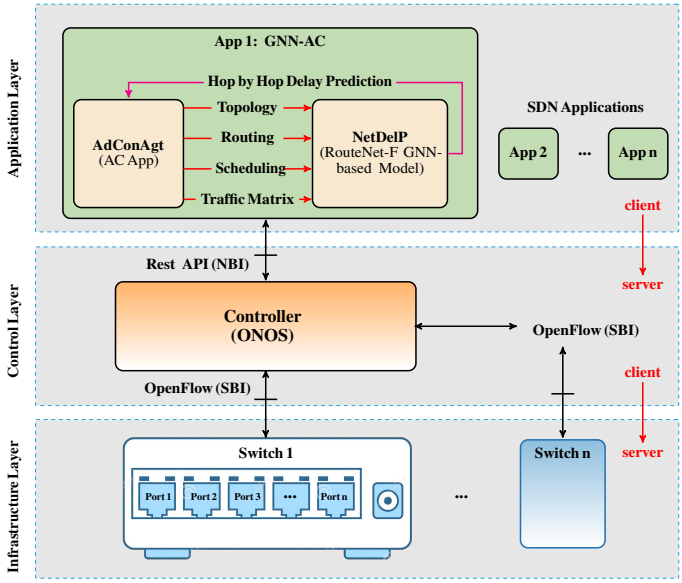

Figure 2: GNN-AC architecture view

C. 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 1, 2, 3, and 4).

$$\xi = \{[\mu], [\tau], \rho\} \quad (1)$$

$$\mu = \{0, 1, 2, ..., 56\} \quad (2)$$

$$\tau = \{output(port_{number}), \quad drop\} \quad (3)$$

$$\rho = \{0, 1, 2, ..., 65535\} \quad (4)$$

D. GNN-AC Workflow

The GNN-AC framework reduces the load imbalances in the network by real-time LP (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**).

Algorithm 1 GNN-AC Workflow

1: *Inputs:* {Paths}, {Hosts}, {Links}, {ξ}, {PPD}, {PDB}
2: *Outputs:* {New Flow Rule (ξ) (Accept/Drop packet)}
3: SDN CTRL receives PACKET-IN message with μ info
4: if (exist path with a ξ satisfying packet DSCP μ) **then**:
5: Apply ξ
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: Get the PDB
12: Get the PPDs using real-time Traffic Matrix TM(t)
13: if $PPDs < PDB$ **then**:
14: Chose the path with the minimum PPD
15: else
16: Reject the packet
17: Install the ξ on the devices
18: Goto 5

IV. Performance Evaluation

A. 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 is balancing the traffic compared with SDN-SP. 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 classes of traffic (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 I summarizes the different parameters, and Figure 3 draws the setup. We used ONOS as an SDN controller, OVS for the OpenFlow switches, and Mininet for the network topology.

Table I: **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

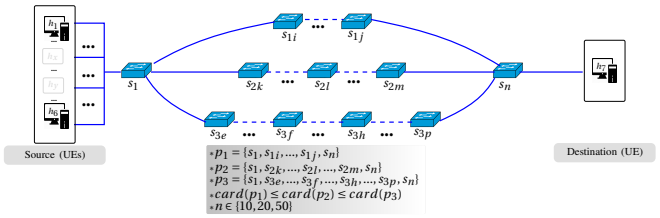

Figure 3: Setup's Network Topology with 3 flavours

We have put the network under realistic conditions (with different topology flavours, paths, and classes of traffic 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.

B. Results

In this section, we focus on both GNN-AC and SDN-SP 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 $20ms$ (see below). For the two first flavours c_1 and c_2, the average latencies is $3\,ms$ and $8\,ms$, respectively (as we will 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).

1) E2E Network Latency: Figure 4 (respectively, Figure 5) shows the min, max, and median E2E latencies (in ms) obtained with GNN-AC (respectively, SDN-SP), for the classes of traffic f_1, f_2, f_3, f_4, f_5, and f_6 (respectively, independently from classes of traffic) 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 classes of traffic 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, 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 transmission and propagation 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.

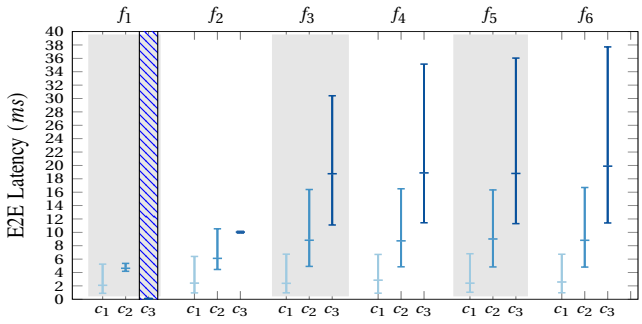

Figure 4: E2E latency obtained with GNN-AC for the aforementioned classes of traffic (f_1 to f_6) under the 3 flavours (c_1 to c_3)

SDN-SP: We remark in Figure 5, that latencies obtained for all the classes of traffic (i.e., S_1 and S_2) are similar to *only* S_2 values in Figure 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 SDN-SP 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 and 5), we demonstrated that our solution GNN-AC respects the 3GPP PDB specifications.

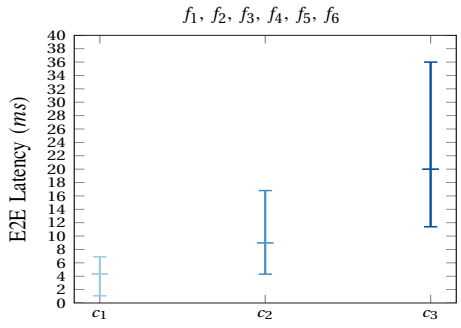

Figure 5: E2E latency obtained with SDN-SP independently from classes of traffic (f_1 to f_6) under the 3 flavours (c_1 to c_3)

2) Packet Loss Rate: Table II compares Packet Loss Rate (PLR) for GNN-AC with SDN-SP in percentage (%) for the six aforementioned classes of traffic 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 SDN-SP, we notice that the PLR is increasing, with respect to flavours, between 14% and 21%, independently from Class of Traffic (CoT). It is relevant to note that SDN-SP 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 SDN-SP 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 SDN-SP based on the shortest path strategy.

Table II: **Packet Loss Rate (in %)**

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

3) QoS Breach: Table III provides a QoS breach comparison in percentage (%) between GNN-AC and SDN-SP. 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 LP mean relative error of RouteNet-F module estimated around 6.24% [21]. For the SDN-SP 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 5 represents the average QoS breach calculation (G_x) for GNN-AC (G_{GNN-AC}) and SDN-SP (G_{SDN-SP}) for all the classes of traffic and all the topology flavours ($g_x(f_i, c_j)$). For GNN-AC, $G_{GNN-AC} = 3.191$, while for SDN-SP, $G_{SDN-SP} = 21.5$ which is 7× bigger than G_{GNN-AC}

$$G_x = \frac{\sum_{i=1}^{6}\sum_{j=1}^{3} g_x(f_i, c_j)}{6 \times 3} \tag{5}$$

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

Table III: **QoS Breach (in %)**

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

V. Conclusion

In this paper, we have explored the use of GNN-based RouteNet-F model for LP and its application for SDN AC in B5G networks. We have investigated the performance of our solution GNN-AC for various topology flavours (i.e., scales) and classes of traffic 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 resource allocation instead of taking a binary decision by accepting or rejecting the packets.

Acknowledgment

This work was partially supported by the European Union's Horizon Europe SNS D ImagineB5G project (Grant No. 101096452).

References

[1] I.-T. N. 2030, "A blueprint of technology, applications and market drivers towards the year 2030 and beyond," 2019. [Online]. Available: https://www.itu.int/en/ITU-T/focusgroups/net2030/Documents/White_Paper.pdf

[2] W. da Silva Coelho, "Modeling and optimization of 5g network design," in *2021 33th International Teletraffic Congress (ITC-33)*, 2021, pp. 1–3.

[3] D. Minovski, N. Ögren, K. Mitra, and C. Åhlund, "Throughput prediction using machine learning in lte and 5g networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 3, pp. 1825–1840, 2023.

[4] Y. Zhang, Q. Xu, M. Li, C. Chen, and X. Guan, "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.

[5] M. O. Ojijo and O. E. Falowo, "A survey on slice admission control strategies and optimization schemes in 5g network," *IEEE Access*, vol. 8, pp. 14 977–14 990, 2020.

[6] S. MESSAOUDI, A. Ksentini, and C. BONNET, "Sdn framework for qos provisioning and latency guarantee in 5g and beyond," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, 2023, pp. 587–592.

[7] G. Shen, Q. Li, W. Shi, Y. Jiang, P. Zhang, L. Gu, and M. Xu, "Modeling and optimization of the data plane in the sdn-based DCN by queuing theory," *J. Netw. Comput. Appl.*, vol. 207, p. 103481, 2022. [Online]. Available: https://doi.org/10.1016/j.jnca.2022.103481

[8] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "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. [Online]. Available: https://doi.org/10.1109/INFOCOM.2018.8485853

[9] S. Xiao, D. He, and Z. 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. [Online]. Available: https://doi.org/10.1145/3229543.3229549

[10] A. Mestres, E. Alarcón, Y. Ji, and A. Cabellos-Aparicio, "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*, P. Casas, M. Mellia, A. Dainotti, and T. Zseby, Eds. ACM, 2018, pp. 46–52. [Online]. Available: https://doi.org/10.1145/3229607.3229613

[11] J. Suárez-Varela *et al.*, "Technical report: Routenet-fermi," 2022. [Online]. Available: https://bnn.upc.edu/download/technical_report_routenet_fermi

[12] ——, "The graph neural networking challenge: a worldwide competition for education in ai/ml for networks," *ACM SIGCOMM Computer Communication Review*, vol. 51, no. 3, pp. 9–16, 2021.

[13] X. Yin, Y. Liu, L. Yan, and D. Li, "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.

[14] A. Varga and R. 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*, S. Molnár, J. R. Heath, O. Dalle, and G. A. Wainer, Eds. ICST/ACM, 2008, p. 60. [Online]. Available: https://doi.org/10.4108/ICST.SIMUTOOLS2008.3027

[15] J. Suárez-Varela, P. Almasan, M. F. Galmés, K. Rusek, F. Geyer, X. Cheng, X. Shi, S. Xiao, F. Scarselli, A. Cabellos-Aparicio, and P. Barlet-Ros, "Graph neural networks for communication networks: Context, use cases and opportunities," *CoRR*, vol. abs/2112.14792, 2021. [Online]. Available: https://arxiv.org/abs/2112.14792

[16] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet: Leveraging graph neural networks for network modeling and optimization in SDN," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2260–2270, 2020. [Online]. Available: https://doi.org/10.1109/JSAC.2020.3000405

[17] Z. Ge, J. Hou, and A. Nayak, "Forecasting sdn end-to-end latency using graph neural network," in *2023 International Conference on Information Networking (ICOIN)*, 2023, pp. 293–298.

[18] D. Heo, S. Lange, H. Kim, and H. Choi, "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. [Online]. Available: https://doi.org/10.23919/APNOMS50412.2020.9236954

[19] J. Henry, T. Szigeti, and L. M. Contreras, "Diffserv to QCI Mapping," IETF, Internet-Draft draft-henry-tsvwg-diffserv-to-qci-04, Apr. 2020, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-henry-tsvwg-diffserv-to-qci/04/

[20] ONF, "Software-Defined Networking: The New Norm for Networks," Apr. 2012.

[21] M. Ferriol-Galmés, J. Paillisse, J. Suárez-Varela, K. Rusek, S. Xiao, X. Shi, X. Cheng, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet-fermi: Network modeling with graph neural networks," *arXiv preprint arXiv:2212.12070*, 2022.