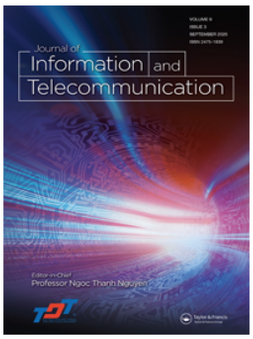


Journal of Information and Telecommunication

ISSN: 2475-1839 (Print) 2475-1847 (Online) Journal homepage: www.tandfonline.com/journals/tjit20

Predictive dynamic multi-flow routing (PD-MFR) algorithm towards sixth generation (6G) software-defined networks

Buse Pehlivan, Volkan Rodoplu, Engincan Tunçay & Dilara Eraslan

To cite this article: Buse Pehlivan, Volkan Rodoplu, Engincan Tunçay & Dilara Eraslan (28 Jul 2025): Predictive dynamic multi-flow routing (PD-MFR) algorithm towards sixth generation (6G) software-defined networks, Journal of Information and Telecommunication, DOI: 10.1080/24751839.2025.2532222

To link to this article: https://doi.org/10.1080/24751839.2025.2532222

Published online: 28 Jul 2025.

JOURNAL OF INFORMATION AND TELECOMMUNICATION
https://doi.org/10.1080/24751839.2025.2532222

Predictive dynamic multi-flow routing (PD-MFR) algorithm towards sixth generation (6G) software-defined networks

Buse Pehlivan [a], Volkan Rodoplu [b], Engincan Tunçay [b] and Dilara Eraslan [b]

[a]Department of Electrical and Electronics Engineering, Graduate School, Yaşar University, Izmir, Turkey;
[b]Department of Electrical and Electronics Engineering, Yaşar University, Izmir, Turkey

ABSTRACT

We develop a dynamic Quality of Service (QoS) routing algorithm based on network traffic prediction for Sixth Generation (6G) SDNs. First, we formulate a mixed integer optimization model that incorporates the key constraints for Ultra-Reliable Low Latency Communication (URLLC), enhanced Mobile Broadband (eMBB), and massive Machine-Type Communication (mMTC) traffic. Second, we develop our Predictive Dynamic Multi-Flow Routing (PD-MFR) algorithm for QoS flows based on this optimization model. In PD-MFR, first, the network forms predictions of the aggregate eMBB traffic flow generation rates and makes reservations for the flows on the upcoming routing window. Second, delay-tolerant mMTC flows are scheduled to be routed to fill up the residual capacities that remain after the eMBB flow reservations. Third, URLLC flows are routed reactively. We demonstrate the performance of our PD-MFR algorithm when Autoregressive Integrated Moving Average (ARIMA) and Multi-Layer Perceptron (MLP) models are used in forecasting the eMBB flow generation rates. We measure the performance of PD-MFR against the benchmark QoS-Shortest Path Algorithm (QoS-SPA) in which all of the QoS flows are routed reactively and show that PD-MFR outperforms QoS-SPA significantly. This work advances the state of the art in QoS routing algorithms based on network traffic prediction geared towards next-generation SDNs.

ARTICLE HISTORY
Received 24 April 2024
Accepted 21 June 2025

KEYWORDS
Quality of Service (QoS); routing; predictive network; Software-Defined Network (SDN)

1. Introduction

Predictive networking constitutes a novel paradigm in which a telecommunication network predicts the future demands that will be experienced by the network and allocates network resources in advance. This contrasts sharply with the traditional reactive networks, in which the network merely reacts to the current demand that the network sees. The evolution of cellular networks standards, including Fifth Generation (5G) systems, is based on the reactive networking paradigm. However, the advent of Artificial Intelligence (AI), in particular, machine learning (ML), has ushered in a new era, in which networks will be able to form accurate predictions of the future demands on the network

CONTACT Buse Pehlivan buse.phlv@gmail.com Department of Electrical and Electronics Engineering, Graduate School, Yaşar University, Agacli Yol, Izmir 35100, Turkey

using ML algorithms, thus unleashing the potential of predictive networks, which are expected to play a major role in the evolution towards Sixth Generation (6G) systems.

Predictive networks present key advantages over reactive ones: First, if the predictions are accurate, the pre-allocation of network resources can significantly reduce the control overhead incurred at distinct layers of the protocol stack. Second, the network can allocate its data plane resources more efficiently, especially among the highly diversified traffic types that have become the norm with the evolution of 5G systems. While network routing optimization among these diverse types of traffic with respect to the Quality of Service (QoS) requirements of each traffic type has been addressed in the past, such optimization has been in reaction to the currently occurring demand on the network.

The key technical challenges in designing a predictive QoS routing algorithm are as follows: (1) One must coordinate the distinct priorities of multiple flow types when predictions of only a subset of these types are available. (2) The prediction accuracy plays a key role in determining QoS routing performance; however, the latter must be designed to be robust to prediction error. (3) The resulting predictive QoS routing algorithm must have practically feasible execution time.

The main contributions of this work to the state of the art are as follows:

- We develop a Predictive Dynamic Multi-Flow Routing (PD-MFR) algorithm that utilizes the prediction of the generation rate of the aggregate eMBB flow between each source-destination pair on the topology in order to make reservations for each such flow. We demonstrate that PD-MFR significantly outperforms the reactive benchmark QoS-Shortest Path Algorithm (QoS-SPA).
- Our PD-MFR algorithm has the potential to significantly impact the evolving 6G core networks by performing joint optimization across eMBB, mMTC and URLLC flow types based on eMBB traffic predictions in order to route these flows by satisfying their QoS requirements.
- Our algorithm is computationally efficient and can be run in real time. Thus, it stands as an important candidate that can be used in next-generation networks that are expected to be based on Artificial Intelligence.

In developing our PD-MFR algorithm, first, we shall present a mixed integer optimization model that incorporates the key constraints for Ultra-Reliable Low Latency Communication (URLLC), enhanced Mobile Broadband (eMBB), and massive Machine-Type Communication (mMTC) traffic, which are the key traffic classes that were introduced in 5G systems.[1] The key novelty in our algorithm is that the aggregate eMBB flow generation rate for each source-destination pair is *predicted* on a topology over the upcoming routing window. Then, route reservations are made for eMBB and delay-tolerant mMTC flows based on these eMBB traffic predictions before the routing window begins. This contrasts sharply with existing QoS routing algorithms that merely react to the QoS flows.

The PD-MFR algorithm demonstrates several key findings that validate its performance in predictive QoS routing:

- PD-MFR outperforms the reactive benchmark QoS-SPA in terms of the fraction of flows delivered and network capacity utilization, ensuring compliance with QoS requirements for eMBB, mMTC, and URLLC traffic types.

- The integration of forecasting models, such as ARIMA and MLP, enables proactive route planning by anticipating traffic demands, even under variable eMBB flow conditions.
- Simulations confirm that the execution time of PD-MFR scales linearly with the number of eMBB and mMTC flows, making it practical for real-world scenarios across different network topologies and traffic scenarios.

In addition to these findings, PD-MFR provides several advantages that highlight its suitability for real-world deployment:

- Proactive resource allocation ensures that routing decisions are pre-planned before the routing window begins, thereby improving flow delivery and reducing latency.
- The hybrid approach combines linear optimization for eMBB and mMTC flows with heuristic methods, such as the Successive Routing Algorithm (SRA) employed for URLLC flows, which leverages Yen's algorithm to meet stringent latency and reliability requirements, ensuring computational efficiency and adaptability.
- Efficient utilization of network resources is achieved by reserving capacity based on predicted traffic demands, balancing the needs of different flow types and improving overall network performance.

While the PD-MFR algorithm offers these advantages, certain limitations are acknowledged. The optimization program includes mixed-integer constraints for ensuring K-path redundancy in URLLC flows, which may introduce computational complexity. However, these constraints are solved offline before the routing window begins, and real-time operations rely solely on efficient heuristic approaches. Additionally, while forecasting accuracy is crucial for resource allocation, PD-MFR's design effectively mitigates potential adverse effects of forecasting errors, ensuring consistent performance even under variable conditions.

The rest of this paper is organized as follows: In Section 2, we compare our work against the state of the art in this area. In Section 3, we present our mathematical framework. In Section 4, we present our results. In Section 5, we present our conclusions.

2. Relationship to the state of the art

In this section, we describe the relationship of our work to the state of the art in three categories: (1) We contrast our approach with the reactive QoS routing and scheduling approaches. (2) We compare our work against the approaches which use predictive networks for QoS-based applications without a routing model. (3) We contrast our work against the approaches which use machine learning and a QoS routing model.

First, we contrast our work against those past articles on routing IoT traffic reactively in order to meet QoS constraints: In Egilmez, Civanlar, et al. (2012), Yen's *K* shortest path algorithm is utilized to find the optimal path for delay-sensitive or loss-sensitive packets. In L. Li et al. (2014), decision making at three layers (application, network, and sensing) is investigated. In Llopis et al. (2016), the authors focus on minimizing end-to-end delay by using QoS routing for SDN-IoT. Reference Awan et al. (2014) focuses on packet queuing for delay-sensitive applications.

In S. Xu et al. (2020), a two-stage optimization program first calculates routing strategies in a reactive fashion for flows without the Ternary Content Access Memory (TCAM) capacity

constraint and then sets paths of the flows under this constraint in the second stage. In Saha et al. (2018), the authors propose two types of routing strategies for flows that are delay-sensitive and loss-sensitive, using a greedy approach with Yen's *K*-shortest path algorithm for Software-Defined Internet of Things networks. The routing algorithm assumes that the IoT applications under consideration are event-driven and unpredictable.

Reference Liu et al. (2016) schedules data traffic by considering distinct packet delays and focuses on video streaming. In Z. Wang and Crowcroft (1996), the QoS routing problem is considered for finding a path for multimedia applications. In Juttner et al. (2001), a Lagrange relaxation-based algorithm is used for delay constrained applications to obtain the least cost route for the real-time delay constrained traffic. In Egilmez, Dane, et al. (2012), dynamic routing is performed over SDN to fulfill QoS requirements for multimedia delivery. The incoming traffic is divided into two categories that are data flows and multimedia flows. While data flows follow the shortest path, multimedia flows are dynamically placed on the paths with guaranteed QoS restrictions. In T.-F. Yu et al. (2015), the incoming video flows are routed dynamically over the shortest path taking into account the constraints within two levels of QoS flows that are the base layer and the enhancement layer, respectively. Reference Alsenwi et al. (2019) presents a risk-sensitive resource allocation approach for 5G URLLC, focussing on the scheduling of eMBB users reactively and aiming to minimize eMBB transmission risk while ensuring URLLC reliability. Reference Bairagi et al. (2020) aims to co-schedule eMBB and uRLLC traffic types reactively and optimize the minimum expected achieved rate for eMBB traffic. In contrast with the given reactive works in this category (which do not make or utilize any traffic predictions), we *predict* the upcoming eMBB flows and develop a joint routing strategy across all of the flow types based on these predictions.

In the second category, we shall address articles that take a predictive approach for QoS-based applications without a routing model: In Atawia, Hassanein, and Noureldin (2017), the authors carry out an optimization study that prioritizes future video quality conditions. In their work, regardless of users and resource allocations, they characterize quality based solely on bandwidth from QoS constraints. References Nakip et al. (2019) and Rodoplu, Nakıp, et al. (2020) use artificial intelligence algorithms such as MLP, LSTM, and ARIMA models to predict the traffic generation patterns of IoT devices. In Khambari et al. (2017), the authors examine a predictive packet dropout technique to maintain the Quality of Experience (QoE) level in bandwidth-limited frames. In Brown and Khan (2015), the authors examine a low complexity predictive resource allocation algorithm for use in Machine-to-Machine (M2M) applications. In Tang et al. (2018), deep learning algorithms are used for traffic load prediction without QoS flow routing.

In Rodoplu, Nakip, et al. (2020), the traffic generation patterns of IoT devices are predicted using machine learning models. Then, the IoT traffic is scheduled at the Medium Access Control layer based on these predictions over multiple channels to maximize the total number of bits delivered under the delay constraints of each application. Reference M. Li et al. (2018) proposes a predictive pre-allocation algorithm to overcome the low-spectrum utilization problem of the non-predictive pre-allocation techniques in the case where allocated resource blocks are not used for uplink scheduling of delay-sensitive traffic. In C. Yao et al. (2016), predictive resource allocation (PRA) is performed to maximize the throughput using excess resources. The resource allocation maximizes the transmission completion time.

In Atawia, Hassanein, Abou-Zeid, et al. (2017), an energy-efficient stochastic predictive resource allocation method to solve the prediction uncertainty problem is presented by modelling the predicted uncertainty rate as a random variable in wireless networks for video delivery in the presence of base stations and mobile users. Reference Wen et al. (2020) performs correlation and causality-based prediction using their complementary strengths together to maximize the prediction accuracy of wireless network traffic. In W.-E. Chen et al. (2019), a convolutional neural network based approach is utilized to classify eMBB, mMTC and URLLC traffic. In Montazerolghaem and Yaghmaee (2020), an SDN framework is presented for load balancing of the network traffic among IoT servers. Predictive time-series analysis using the normalized least mean squares (NLMS) is performed in order to detect the least loaded IoT server. Reference Alsenwi et al. (2021) optimizes 5G resource slicing for URLLC and eMBB traffic types by employing a Deep Reinforcement Learning framework. In contrast with these previous works in the second category which use predictive networks *without* performing QoS routing, we design a predictive QoS routing algorithm in which the aggregate flows between source-destination pairs are routed by utilizing dynamic optimization jointly across the URLLC, eMBB, and mMTC traffic types.

In the third category, we contrast the predictive approach of our PD-MFR algorithm against the past articles that use machine learning for QoS routing: In Bouzidi et al. (2021), the authors introduce an SDN-based Deep Q-Network (DQN) approach that utilizes neural networks to predict traffic congestion in order to improve routing configurations. In W. Sun et al. (2021), an algorithm based on a combination of machine learning algorithms is used for data flow classification. In Eyobu and Edwinah (2023), an LSTM-based algorithm is utilized in order to estimate the route that satisfies QoS. In Awad et al. (2021), a machine learning based algorithm that learns the routing solutions of a constrained heuristic framework is utilized. In order to improve efficiency of routing in IoT networks, Reference Seyfollahi et al. (2023) employs support vector machine and time series prediction methods by forecasting the energy consumption of sensors. In Akçapınar et al. (2022), an ARIMA model is used to forecast the number of eMBB flows. In H. Yao et al. (2019), in order to prevent congestion in SDNs, a deep neural network-based algorithm is employed in order to predict queue utilization. Reference Gunavathie and Umamaheswari (2023) uses a Gated Recurrent Unit (GRU) in order to predict performance metrics such as delay, jitter and bandwidth. Furthermore, it adjusts the link weights based on these predictions in order to find an optimal path through the network reactively. In addition, there have been articles (Al Jameel et al., 2022; Al-Jawad et al., 2021; Casas-Velasco et al., 2021, 2020; Y.-R. Chen et al., 2020; Cong et al., 2021; Dai et al., 2022; Srivastava & Pandey, 2021; P. Sun et al., 2019; X. Wang et al., 2021; C. Xu et al., 2020; C. Yu et al., 2018; Yuan et al., 2021) that use Reinforcement Learning (RL) in order to perform routing on SDNs.

The main differences between the articles in this third category and our work are as follows: (1) Our work makes end-to-end reservations on the edges of the network topology jointly for all of the distinct traffic types based on traffic *predictions* of the aggregate eMBB traffic flows, whereas the articles that appear in this category are not based on such reservations. (Our work also differs in this regard from Reference Njah et al. (2020), which uses both reactive and proactive approaches for flow management but does *not* predict network traffic.) (2) Our PD-MFR algorithm solves a *global* optimization program formulated jointly for all of the traffic types across the network and makes reservations based on traffic predictions *before* the actual routing takes place, whereas the works in this

category use machine learning methods for local optimization (with regard to various aspects of routing) without the aim of achieving global optimality.

3. Mathematical framework

In this section, we present our mathematical framework. First, we develop a mixed integer multi-flow optimization program for the heterogeneous set of traffic flows. Then, we present our Successive Routing Algorithm (SRA) as a heuristic to solve this optimization program. This heuristic serves merely as a conceptually intermediate step towards the development of our Predictive Dynamic Multi-Flow Routing (PD-MFR) algorithm for the case of dynamically generated traffic flows.[2] Second, we present our PD-MFR algorithm in three steps: We give an overview of our algorithm. Then, we describe our algorithm in detail. Finally, we give the pseudocode of our algorithm.

3.1. Multi-Flow QoS optimization program and the successive routing algorithm (SRA)

In this section, we present our formulation of a mixed integer optimization model that incorporates the key constraints for URLLC, eMBB, and mMTC traffic. We note that this formulation does not model the predictions of eMBB traffic, which will be introduced in the next section. We model the network topology as a directed graph G such that each SDN router is a vertex (a.k.a. node) of this graph and a directed edge represents an edge on which flow of information in that direction is possible along that edge. The set of vertices of G is denoted by V, and the set of directed edges on G is denoted by E. In Table 1, we present the mathematical denotations that we use in our program. In this table, the last two lines pertain to URLLC flows: In order to increase the reliability of URLLC flows, we shall send copies of the same flow along distinct paths. In this table, $J(f)$ denotes the index set of these copies, each of which is denoted by f', and $q(f')$ is the particular path selected for that copy. (The context of these denotations shall become clear when we present our optimization program below.)

3.1.1. Statement of the multi-flow QoS optimization program

In this section, we define our multi-flow QoS optimization program and discuss parameters and decision variables. In Table 2, we give the parameters and decision variables

Table 1. Mathematical denotations used in our optimization program.

Symbol	Definition
$\mathcal{F}$	Set of all flows
$\mathcal{F}_{type}$	Set of flows of $type \in \{URLLC, eMBB, mMTC\}$
V	Vertex (node) set
E	Set of directed edges
U	Set of sources
D	Set of destinations
$s(f)$	Source of flow $f \in \mathcal{F}$
$d(f)$	Destination of flow $f \in \mathcal{F}$
N_m^+	Set of outgoing neighbours of vertex $m \in V$
N_m^-	Set of incoming neighbours of vertex $m \in V$
$J(f)$	Index set of the copies of $f \in \mathcal{F}_{URLLC}$

Table 2. Parameters and decision variables of our optimization program.

Symbol	Definition
Parameters:	
c_{mn}	Capacity (in bps) of edge $(m \to n) \in E$
$g^{(f)}$	Generation rate of flow $f \in \mathcal{F}$ at the source $s(f)$
ϵ_{mn}	Probability of error on edge $m \to n$
$P_e^{(f)}$	Maximum allowable probability of error for flow $f \in \mathcal{F}_{URLLC}$
$\delta_{m \to n}$	Delay of edge $m \to n$
$\Delta^{(f)}$	Maximum allowable latency for flow $f \in \mathcal{F}_{URLLC}$
$r_{\min}^{(f)}$	Minimum data rate required for flow $f \in \mathcal{F}_{eMBB}$
Decision Variables:	
$r_{mn}^{(f)}$	Data rate on edge $m \to n$ of flow $f \in \mathcal{F}$, $(m \to n) \in E$
$q(f')$	Path selected for the copy $f' \in J(f)$ for flow $f \in \mathcal{F}_{URLLC}$

of our optimization program. We first discuss the parameters: In this table, the parameter c_{mn}, which denotes the capacity of edge $m \to n$, and $g^{(f)}$, which denotes the rate at which flow f is generated, are common to all of the traffic types. The parameters ϵ_{mn}, $P_e^{(f)}$, $\delta_{m \to n}$ and $\Delta^{(f)}$ pertain to only URLLC flows: As we shall see shortly, each URLLC flow f must satisfy a target probability of error $P_e^{(f)}$ in line with the condition of ultra-reliability, and a target latency constraint $\Delta^{(f)}$ in line with the condition of low latency. The parameters ϵ_{mn} and δ_{mn}, which also appear in this table, are measures of edge reliability and edge latency, respectively. For each eMBB flow f, there is a minimum data rate, denoted by $r_{\min}^{(f)}$, which arises due to the broadband condition on that flow. Finally, we note that in addition to these parameters, the decision variables of our optimization program, which are shown on the last line of this table, are the set of data rates $r_{mn}^{(f)}$ at which information flows on the network topology and $q(f')$, which is the path selected for the copy f' among the set of copies of $f \in \mathcal{F}_{URLLC}$.

We present our entire optimization program below:

Objective Function:

$$\max \sum_{f \in \mathcal{F}} \sum_{n \in N_{s(f)}^+} r_{s(f),n}^{(f)} \tag{1}$$

subject to:

A) Edge Capacity:

$$\sum_{f \in \mathcal{F}} r_{mn}^{(f)} \leq c_{mn}, \quad \forall (m \to n) \in E \tag{2}$$

B) Balance Equations: $\forall f \in \mathcal{F}$

$$\sum_{n \in N_m^+} r_{mn}^{(f)} = \sum_{n \in N_m^-} r_{nm}^{(f)}, \quad \forall m \in V \setminus \{s(f), d(f)\} \tag{3}$$

$$\sum_{n \in N_{s(f)}^+} r_{s(f),n}^{(f)} \leq g^{(f)} \tag{4}$$

$$\sum_{n \in N_{d(f)}^-} r_{n,d(f)}^{(f)} = \sum_{n \in N_{s(f)}^+} r_{s(f),n}^{(f)} \tag{5}$$

C) No flow returns to the source of that flow

$$r^{(f)}_{n,s(f)} = 0, \quad \forall f \in \mathcal{F}, \ \forall n \in N^{-}_{s(f)} \tag{6}$$

D) No flow leaves the destination of that flow

$$r^{(f)}_{d(f),n} = 0, \quad \forall f \in \mathcal{F}, \ \forall n \in N^{+}_{d(f)} \tag{7}$$

E) Constraints that are Specific to Traffic Types:
E.1) Constraints Specific to URLLC
Reliability:

$$\left(1 - \prod_{(m \to n) \in q(f'_j)} (1 - \epsilon_{mn})\right) \leq P^{(f)}_{e}, \quad \forall j \in J(f), 1em \forall f \in \mathcal{F}_{URLLC} \tag{8}$$

Latency:

$$\sum_{(m \to n) \in q(f'_j)} \delta_{m \to n} \leq \Delta^{(f)}, \quad \forall j \in J(f), \forall f \in \mathcal{F}_{URLLC} \tag{9}$$

E.2) Constraint Specific to eMBB
Data Rate:

$$\sum_{n \in N^{+}_{s(f)}} r^{(f)}_{s(f),n} \geq r^{(f)}_{\min}, \quad \forall f \in \mathcal{F}_{eMBB} \tag{10}$$

In (1) above, the objective of our optimization program is to maximize the total data rate from the sources to the destinations across all of the flows on the network. This formulation of the objective function presumes that all of the flows that are offered to the network are of equal value; hence, each such flow has the same weight (which is 1) in the objective function. Note that the first sum in (1) is over all of the flows $\mathcal{F}$, and the second sum is over the outgoing data rates of the sources of each of these flows. This is equivalent to summing over all of the end-to-end flows from the sources to the destinations by virtue of Constraints (6) and (7) of the optimization program, which shall be discussed shortly.

The constraint in (2) ensures that the total data rate on any given edge $m \to n$ summed over all of the flows cannot exceed the capacity of that edge. The constraint in (3) states that for each relay node, the total incoming data rate to that node for flow f is equal to the total outgoing data rate from that node for that flow. The constraint in (4) ensures that the total outgoing data rate for flow f out of the source of that flow cannot exceed the generation rate of that flow. Furthermore, the constraint in (5) states that the total incoming data rate for flow f into the destination of that flow is equal to the total outgoing data rate for flow f out of the source of that flow. This constraint ensures that each flow f reaches its intended destination from its source. The constraints in (6) and (7) prevent any flow f from creating loops by returning to its source or by leaving its destination.

We note that since mMTC flows are delay-tolerant, the most straightforward way of modelling these flows is based on volume (i.e. the total number of bits) rather than the data rate, since these flows can be delayed while still satisfying their QoS requirements. However, we shall model all flows, including mMTC flows, as rate-based rather than a mixture of rate-based and volume-based flows. (The conversion from the volume-based

to the rate-based model for mMTC flows will be described in Section 3.2.1.) Thus, our formulation assumes that rate-based flows for all three traffic types are generated by the sources on the topology and are to be routed towards destinations.

In our optimization program, each URLLC flow, if admitted into the network, is to be sent along K paths from its source to its destination; thus, each URLLC flow is admitted under a K-fold redundancy. This models multipath diversity that is considered essential for reliable URLLC flow routing. Each such URLLC flow must satisfy both end-to-end probability of error and end-to-end latency constraints in our program as stated in constraints (8) and (9), respectively. In contrast, eMBB flows are characterized by the minimum data rate that must be achieved for each such flow as shown in Constraint (10). This accurately models the constraint that QoS flows for video streaming must satisfy end-to-end.

3.1.2. Methodology to solve the optimization program

In order to solve our QoS routing optimization program, first, we have developed a heuristic, which we call the Successive Routing Algorithm (SRA), that solves the program by routing URLLC flows first using Yen's algorithm (Yen, 1971), followed by a solution of the resulting linear optimization program that contains the remaining traffic types (namely, eMBB and mMTC). That is, SRA prioritizes the URLLC flows over the other traffic types since such flows typically involve critical data, e.g. as found in remote surgery. Yen's algorithm returns the redundant paths for each URLLC flow that are ordered with respect to increasing end-to-end latency. Out of those paths, those that do not satisfy the end-to-end probability of error requirement of the URLLC application are eliminated. Only those URLLC flows for which K redundant paths from the source to the destination can be found in this manner are admitted; the remaining URLLC flows are not admitted.

Note that the PD-MFR algorithm's optimization program is inherently mixed-integer due to the reliability and latency constraints in (8) and (9) for URLLC flows, which ensures K-path redundancy. However, since URLLC flows are routed reactively in Stage B using a heuristic approach based on the Successive Routing Algorithm (SRA) and Yen's algorithm which has polynomial computational complexity, the mixed-integer constraints are *not* executed as part of the optimization program in Stage A. This exclusion transforms the problem into a linear optimization program, as the remaining constraints (2)-(7) and (10)–such as flow conservation, capacity limits, and latency for eMBB and mMTC flows–are linear. As a result, the execution time of PD-MFR in Stage A grows linearly with the number of eMBB and mMTC flows, as confirmed by our simulations in Section 4.2.4, making it computationally efficient and scalable for real-world deployment.

3.2. Predictive dynamic multi-flow routing (PD-MFR) algorithm

In order to address the realistic case in which traffic flows are generated dynamically, we have developed our Predictive Dynamic Multi-Flow Routing (PD-MFR) algorithm based on the QoS constraints in the multi-flow optimization program that appears in (1)–(10). The main differences between PD-MFR and the SRA in the previous section are that (1) PD-MFR has been designed specifically for the dynamic case in which new flows are generated and are to be admitted to the network, and (2) PD-MFR is a predictive algorithm, in which the aggregate eMBB flows are predicted for an upcoming routing window.

Our main strategy shall be to run the optimization program in (1)-(7) and (10) before each of the actual eMBB flows is generated. Because this optimization program has non-negligible execution time, we aim to run the optimization program *before* the routing window begins.

Figure 1 shows a single routing window that has been divided into time intervals ('bins') of duration T_{bin}. We let T denote the duration of a routing window. Thus, there are $W \equiv T/T_{bin}$ bins in a single routing window. (We assume that the number of bins in a routing window is fixed throughout this paper. Furthermore, we shall detail the considerations in regard to the choice of the parameters T and T_{bin} in Section 3.2.5.)

Let $\mathcal{B}$ denote the set of bins that fall in the routing window. We assume that the generation rate $g^{(f)}$ of each such flow is constant during a time bin $b \in \mathcal{B}$. (However, $g^{(f)}$ is allowed to change across distinct time bins.) Furthermore, the source $s(f)$ and the destination $d(f)$ of each such flow do not change during the routing window. Finally, we assume that each flow f is generated at the beginning of a time bin, and terminated at the end of the same or some future time bin.

In Figure 2, we present the top-level representation of our PD-MFR algorithm for any given time bin. In this figure, the input of the 'Forecaster' block is past eMBB traffic, and the output of that block is the predicted eMBB traffic; which are the list of past generation rates and predicted generation rates of eMBB flows that fall in that time bin, respectively. Note that the Forecaster block performs the predictions before the routing window begins.

Note that in Figure 2, we divide the PD-MFR algorithm into two stages which are Stage A (that occurs before the routing window begins); and Stage B (that occurs after the routing window begins). The input of the 'Stage A: Route Planning' block is the predicted eMBB traffic. The aim of the Stage A block is to plan the route of each eMBB and mMTC flow for that particular time bin in the upcoming routing window. This block yields reservations for eMBB and mMTC traffic over the entire topology for the upcoming routing window.

The 'Stage B: Routing' block that appears in Figure 2 takes the output of Stage A as input. In addition, the actual eMBB, URLLC and mMTC traffic also enter this block. The aim of this Stage B block is to route the URLLC flows reactively and to send the actual eMBB and mMTC flows using the routes that were reserved for eMBB and mMTC flows in Stage A.

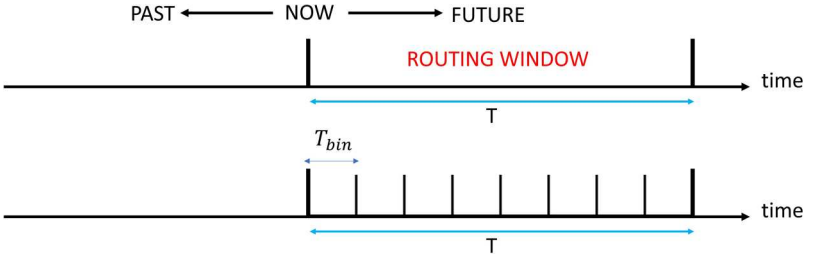


Figure 1. A single routing window over which predictive QoS optimization is performed in our PD-MFR algorithm.

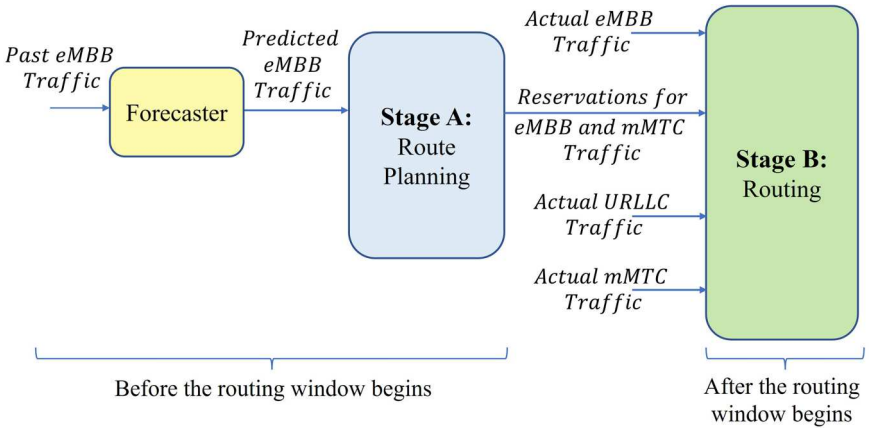


Figure 2. Top-level representation of the PD-MFR algorithm.

3.2.1. Overview

We shall first give an overview of each of the stages of PD-MFR: In Stage A of PD-MFR, for each time bin of the upcoming routing window, the following operations are performed: First, a fixed fraction of the capacity on each edge on the topology is reserved for all of the URLLC flows (for which no predictions are available in our model). Second, the network predicts the generation rate of the aggregate eMBB flow for each source-destination pair over the upcoming routing window. Third, given the set of these predictions, the reservation on the edges (determined by the result of optimization) for each aggregate eMBB flow is made for the predicted amount plus a buffer that it used to handle any overflow (beyond the prediction).

After a fixed fraction of the capacity has been reserved for URLLC flows and a variable fraction of the capacity has been reserved based on predicted eMBB flows as described above, PD-MFR aims to reserve the remaining capacity on the network for mMTC traffic. We assume that there is a large amount of aggregate mMTC volume (in bits) that awaits to be delivered from each source to each destination on the topology. For each source-destination pair, PD-MFR creates a constant bit-rate offered mMTC flow by dividing the mMTC traffic volume (for that source-destination pair) by the length of the upcoming routing window. Then, the optimization program in (1)–(7) is solved for the set of all such mMTC flows on the remaining edge capacities of the topology *before* the routing window begins. The output of the optimization program returns the routes and the rates for mMTC flows on the topology, which are then reserved for mMTC flows.[3]

After the routing window begins, in Stage B of PD-MFR, each URLLC flow that occurs at that time is routed reactively. Then, the edge capacities that have been reserved for eMBB and mMTC flows are utilized to route the actual flows dynamically.

3.2.2. Detailed description of the forecaster

In this section, we provide a detailed description of the forecaster block that appears in Figure 2. In Figure 3, we present the block diagram of the forecaster used in PD-MFR. The main task of this forecaster is to predict the future traffic generation rate of each

eMBB flow in the upcoming routing window. For each aggregate flow $f \in \mathcal{F}_{eMBB}$, the forecaster uses a set of previously observed traffic generation rates as input features. We define $G^{(f)}[b]$ as the aggregate generation rate of an eMBB flow f at time bin b, and $\hat{G}^{(f)}[b]$ as the predicted generation rate of an eMBB flow f at time bin b. In Figure 3, the input to the forecaster for flow f is the collection $\{G^{(f)}[b-l]\}_{l\in\{1,\ldots,L_f\}}$ at time bin b, where l represents the index of the time bin relative to the current time bin b, and L_f denotes the total number of past time bins included as features for predicting the future generation rate of flow f. Based on these inputs, the forecaster predicts future generation rates over a forecast horizon H_f. The output of the forecaster is $\{\hat{G}^{(f)}[b+h]\}_{h\in\{1,\ldots,H_f\}}$, providing the predicted traffic generation rates for flow f for the upcoming H_f steps.

The current set of inputs and outputs of the forecaster is selected to align with the specific operational requirements of the PD-MFR algorithm. The input features, derived from historical traffic generation rates, provide a time-series context necessary for capturing patterns in eMBB flow behaviour. This selection ensures that the forecaster can predict traffic dynamics accurately enough to guide resource reservation in Stage A of PD-MFR. The output, which consists of predicted generation rates over the upcoming routing window, directly supports the proactive planning of routes.

The criterion that a particular forecasting algorithm must satisfy is generating predictions that are sufficiently reliable to enable PD-MFR to meet the QoS requirements for each eMBB flow. While perfect prediction accuracy is not necessary, the forecasts must be reliable enough to support proactive route planning in Stage A. (This is supported by our results in Section 4, where we demonstrate that even with practical forecasting models like MLP or ARIMA, PD-MFR consistently outperforms a reactive benchmark algorithm. This highlights the algorithm's robustness to forecasting inaccuracies while achieving better results in network performance.)

The forecaster operates before the routing window begins to ensure that its computational process does not interfere with real-time operations. Once the routing window starts, the forecaster's role is complete, and the real-time operation of PD-MFR focuses on routing flows based on the actual generation rate of each flow, using the route that is reserved in Stage A for each such flow.

Note that the forecasting model within the forecaster block must be trained offline for each aggregate eMBB flow f using historical data, specifically past generation rates allocated for training. This process involves selecting and tuning forecasting models suitable for capturing traffic patterns. (A detailed explanation of the training methodology is provided in Section 4.1.1.)

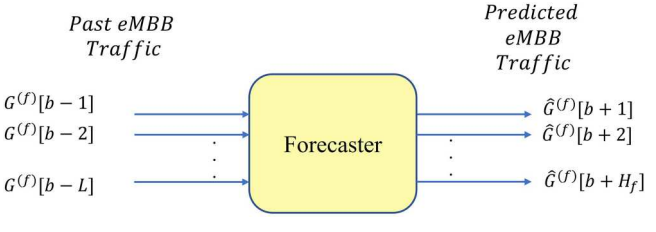


Figure 3. Block diagram of the forecaster used in PD-MFR.

3.2.3. Detailed description of the PD-MFR algorithm

In this section, we give a detailed description of the stages of the PD-MFR algorithm. In preparation for Stage A, the following steps are performed for each time bin b in the upcoming routing window: First, for each edge $m \rightarrow n$, we reserve $(1-\alpha)c_{mn}$ of the capacity c_{mn} of edge $m \rightarrow n$ for each bin on the upcoming routing window for the URLLC flows, where $\alpha \in [0, 1]$. (As a result, the remaining capacity for eMBB and mMTC flows for each bin in the upcoming routing window is αc_{mn}.) The reason that we introduce this parameter α is to make accommodations for the URLLC flows that will be generated in the upcoming routing window, which we assume are not predicted in advance. When the actual flows arrive, URLLC flows shall be given priority over all other flows. By varying the value of α in our simulation, we aim to achieve a trade-off between accommodating URLLC flows and granting the reservations that we have made for the aggregate eMBB and mMTC flows. We shall now focus on any one of the time bins in $\mathcal{B}$. (Identical operations are performed across all of the time bins in $\mathcal{B}$ in parallel for the flows that fall in each such time bin.) Second, we predict the generation rate $g^{(f)}$ of each aggregate flow $f \in \mathcal{F}_{eMBB}$ from every source to every destination on the topology (for the given time bin in question). We let $\hat{g}^{(f)}$ denote the predicted aggregate flow generation rate for such each flow.[4]

Now, for Stage A of PD-MFR, for each time bin of the upcoming routing window, we run the optimization program in (1)–(7) and (10) only for the eMBB flows that are predicted to fall in that time bin. The optimization program outputs the following: (1) the reserved set of routes between source $s(f)$ and destination $d(f)$ of each eMBB flow f that falls in that time bin; and (2) the reserved part of the capacity of each edge on each such reserved route.

Note that the above reservation for an eMBB flow is based merely on a point estimate of the generation rate of the aggregate eMBB flow from each source to each destination (for that time bin). However, the actual aggregate eMBB flow may be smaller or greater than this point estimate. In order to handle the case where the actual flow amount is greater than the point estimate, we shall reserve an extra buffer on each edge that is $\beta \in [0, 1]$ times the remaining capacity of that edge. After the reservation for each eMBB flow (including the extra buffer) is made, we run the optimization program in (1)–(7) only for the mMTC flows (that fall in that time bin) and obtain the reserved routes for the mMTC flows.

The following steps of Stage B are performed for each time bin in the current routing window: First, we run Yen's K-shortest path algorithm for the URLLC flows in the current routing window. This algorithm returns redundant paths that are ordered with respect to increasing end-to-end latency between the source and the destination of each URLLC flow. For each such flow, the path that fails to meet the end-to-end probability of error criterion of that flow is eliminated. For each URLLC flow, if K paths can be found in this manner, the URLLC flow is sent. Second, for each eMBB flow for which we make reservations in Stage A, we check the constraint in (10). For each such flow, if this QoS constraint is satisfied, we send that flow on the reserved route. Otherwise, that flow is not admitted.

3.2.4. Pseudocode of the PD-MFR algorithm

We give the pseudocode of our PD-MFR algorithm in Figures 4–6. Note that for *every* time bin in the upcoming routing window, the *PD-MFR* function, which appears in Figure 4, is

```
1  function PD-MFR (G, F_eMBB, F_mMTC, F_URLLC, cap, g_eMBB, ĝ_eMBB, g_mMTC, g_URLLC)
2      flowList_Stage−A, remCap_Stage−A, reservedEdges_Stage−A = PD-MFR-Stage-A (G,
                                                  [F_eMBB, F_mMTC], α ∗ cap, g_mMTC, ĝ_eMBB)
3      [ ] = PD-MFR-Stage-B (G, F_URLLC, cap, g_eMBB, g_URLLC, g_mMTC, flowList_Stage−A,
                                                  remCap_Stage−A, reservedEdges_Stage−A)
4  end function
```

Figure 4. Pseudocode of the *PD-MFR* function which initiates Stage A and Stage B of *PD-MFR* respectively.

called. This function, in turn, calls the *PD-MFR-Stage-A* and *PD-MFR-Stage-B* functions in Figures 5 and 6, respectively. We note that all of the variables that appear in the pseudocode in Figures 4–6 are internal to the particular time bin for which the function is called.

Recall that before Stage A of PD-MFR (for that particular time bin), we predict the generation rate $g^{(f)}$ of each aggregate flow $f \in \mathcal{F}_{eMBB}$ (that falls in that particular time bin) from every source to every destination on the topology. We now describe the functions that appear in Figures 4–6. (In the description below, we shall omit mentioning passing the topology G as input whenever this applies in order to avoid repetition.)

PD-MFR Function: The inputs to the *PD-MFR* function which appears in Figure 4 (on line 1) are the list of eMBB, mMTC and URLLC flows (that fall in the current time bin), namely, $\mathcal{F}_{eMBB}$, $\mathcal{F}_{mMTC}$ and $\mathcal{F}_{URLLC}$, respectively; the list *cap* of capacities of the edges on the topology; the lists of the actual and the predicted generation rates of the eMBB flows, which are denoted by g_{eMBB} and $\hat{g}_{eMBB}$, respectively; and the list of generation rates of the mMTC and URLLC flows, which are denoted by g_{mMTC} and g_{URLLC}, respectively.

```
5  function PD-MFR-Stage-A (G, F, cap, g_mMTC, ĝ_eMBB)
6      flowList_eMBB, remCap, edges_eMBB = Reserve-eMBB (G, Extract_eMBB(F), cap, ĝ_eMBB)
7      for each flow f in flowList_eMBB do
8          for each edge in edges_eMBB[f] do
9              remCap[edge] = (1 − β) ∗ remCap[edge]
10         end for
11     end for
12     flowList_mMTC, remCap, edges_mMTC = Reserve-mMTC (G, Extract_mMTC(F), remCap, g_mMTC)
13     flowList = [flowList_eMBB, flowList_mMTC]
14     reservedEdges = [edges_eMBB, edges_mMTC]
15 return flowList, remCap, reservedEdges
16 end function
```

Figure 5. Pseudocode of the *PD-MFR-Stage-A* function, which performs route planning for each eMBB and mMTC flow in the upcoming routing window.

17 **function** PD-MFR-Stage-B ($G, \mathcal{F}, cap, g_{eMBB}, g_{URLLC}, g_{mMTC}, flowList_{Stage-A}, remCap_{Stage-A}, reservedEdges_{Stage-A}$)

18 $flowList_{URLLC}, remCap, edges_{URLLC}$ = RouteURLCC ($G, Extract_{URLLC}(\mathcal{F}), ((1-\alpha) * cap + remCap_{Stage-A}), g_{URLLC}$)

19 **for** each flow f in $flowList_{URLLC}$ **do**

20 Send flow f on $edges_{URLLC}$

21 **end for**

22 **for** each flow f in $flowList_{Stage-A}$ **do**

23 **if** Check-QoS(f) **then**

24 Send flow f on $reservedEdges_{Stage-A}$

25 **end if**

26 **end for**

27 **end function**

Figure 6. Pseudocode of the *PD-MFR-Stage-B* function, where reactive routing of URLLC flows is performed, and the actual eMBB and mMTC flows are sent using the routes reserved for each of these flows in Stage A.

On line 2, on which the *PD-MFR-Stage-A* function is called, note that the inputs to the *PD-MFR-Stage-A* function are the concatenated list of flows [$\mathcal{F}_{eMBB}$, $\mathcal{F}_{mMTC}$] that fall in this time bin; α times *cap* (which stands for the collection of αc_{mn} for each edge $m \rightarrow n$ on the topology); $\hat{g}_{eMBB}$ and g_{mMTC}. The *PD-MFR-Stage-A* function returns the list $flowList_{Stage-A}$ of eMBB and mMTC flows for which route reservations are made in this time bin; the list $remCap_{Stage-A}$ of remaining edge capacities; and $reservedEdges_{Stage-A}$, which is a dictionary each element of which is the reserved edge and part of the capacity that is reserved on that edge for each flow *f* over the sets $\mathcal{F}_{eMBB}$ and $\mathcal{F}_{mMTC}$. On line 3, the *PD-MFR-Stage-B* function is called, whose second input $\mathcal{F}_{URLLC}$ is the list of URLLC flows that will be routed in this time bin.

PD-MFR-Stage-A Function: The *PD-MFR-Stage-A* function in Figure 5, which appears on lines 5-16, takes as input the list of all flows, namely $\mathcal{F}$, that fall in the current time bin. On line 6, where the *Reserve-eMBB* function is called, the inputs to this function are the list of eMBB flows that are extracted from $\mathcal{F}$ by using the $Extract_{eMBB}$ function; *cap* that contains the edge capacities that are available for route reservations in Stage A; as well as $\hat{g}_{eMBB}$. The *Reserve-eMBB* function runs the optimization program in (1)–(7) while adding the eMBB data rate constraint in (10) for the list of eMBB flows that fall in the current time bin. The *Reserve-eMBB* function has three outputs: (1) the list of eMBB flows, denoted by $flowList_{eMBB}$, for which reservations shall be made in the upcoming routing window; (2) the list *remCap* each element of which is the remaining capacity of an edge on the topology after the reservations for all of the flows in $\mathcal{F}_{eMBB}$ have been made; and (3) $edges_{eMBB}$, which is a dictionary each element of which is the reserved edge and part of the capacity that is reserved on that edge for each flow *f* in $\mathcal{F}_{eMBB}$.

On lines 7-11, the *remCap* is updated to $(1-\beta)$ times *remCap* in order to reflect the fact that the extra buffer reserved for each eMBB flow is subtracted from the remaining capacity of each edge on which a reservation is being made for that eMBB flow.

On line 12, the *Reserve-mMTC* function is called. The inputs to this function are the list of mMTC flows that are extracted from $\mathcal{F}$ by using the $Extract_{mMTC}$ function; *remCap*; and g_{mMTC}. The *Reserve-mMTC* function runs the optimization program in (1)–(7) for the list of mMTC flows that fall in the current time bin. This function has three outputs: (1) the list of mMTC flows, denoted by $flowList_{mMTC}$, that are planned to be routed in the upcoming routing window; (2) the list *remCap* each element of which is the remaining capacity of an edge over all of the edges on the topology after reservations for all of the flows in $\mathcal{F}_{mMTC}$ have been made; and (3) $edges_{mMTC}$, which is a dictionary each element of which is a reserved edge and the part of the capacity that is reserved on that edge for each flow f in $\mathcal{F}_{mMTC}$. On line 13, $flowList_{eMBB}$ and $flowList_{mMTC}$ are concatenated into a list called *flowList*; and on line 14, $edges_{eMBB}$ and $edges_{mMTC}$ are concatenated into a list called *reservedEdges*.

PD-MFR-Stage-B Function: In the *PD-MFR-Stage-B* function in Figure 6 (line 17), first, the *RouteURLLC* function is called on line 18, whose inputs are set to the list of URLLC flows that are extracted from $\mathcal{F}$ by using the $Extract_{URLLC}$ function; $(1-\alpha)$ times *cap* plus $remCap_{Stage-A}$; and g_{URLLC}. The $remCap_{Stage-A}$ is a list each element of which is the remaining capacity of an edge in Stage A that is not reserved for any of the eMBB or mMTC flows. The reason that we add $remCap_{Stage-A}$ to $(1-\alpha)$ times *cap* in the third input of the *RouteURLLC* function on line 18 is to allow the URLLC flows to use $remCap_{Stage-A}$ in addition to $(1-\alpha)$ times *cap*. The *RouteURLLC* function runs Yen's *K*-shortest path algorithm for each of the URLLC flows in that time bin. Then, this function finds the redundant paths that are ordered with respect to increasing end-to-end latency between the source and destination of each URLLC flow. For each such URLLC flow, every path that cannot satisfy the end-to-end probability of error criterion for that flow is eliminated. Furthermore, for each such flow, if *K* paths can be found, the *RouteURLLC* function returns those paths; otherwise, the *RouteURLLC* function does not return any path for that flow. The *RouteURLLC* function returns the list $flowlist_{URLCC}$ of URLLC flows that can be routed in the current time bin; the list *remCap* each element of which is the remaining capacity of an edge over all of the edges on the topology after the URLLC flows have been routed; and $edges_{URLLC}$, which is a dictionary each element of which is an edge and part of the capacity that is used on that edge for each flow f in $\mathcal{F}_{URLLC}$.

On lines 19–21, each URLLC flow in $flowlist_{URLCC}$ is sent. On lines 22–25, each flow f in $flowList_{Stage-A}$ is sent only if the *Check–QoS* function returns *true*, that is, only if the QoS constraint in (10) is satisfied in reality for each eMBB flow in $flowList_{Stage-A}$ for that flow.

3.2.5. Choice of the key algorithm parameters

We now discuss the relationship between the key parameters of the PD-MFR algorithm and explain how these parameters of the algorithm are set.

To this end, recall that $\mathcal{B}$ denotes the set of time bins that fall in the upcoming routing window. For each $b \in \mathcal{B}$, let $T_{\text{exec}}^{\text{eMBB}}(b)$ denote the time required in Stage A to compute the routes for all of the eMBB flows that fall in bin b (in the upcoming routing window). Let $T_{\text{exec}}^{\text{mMTC}}(b)$ denote the time required in Stage A to compute the routes for all of the mMTC flows that fall in bin b (in the current routing window) on the remaining capacities after

the predicted eMBB flows have been routed. Let $T_{\text{exec}}^{\text{PD}-\text{MFR}}(b)$ denote the total time required to execute Stage A of PD-MFR for time bin b. Then, $T_{\text{exec}}^{\text{PD}-\text{MFR}}(b) = T_{\text{exec}}^{\text{eMBB}}(b) + T_{\text{exec}}^{\text{mMTC}}(b)$. Let $T_{\text{exec}}^{\text{PD}-\text{MFR}}$ denote the total time required to execute Stage A of PD-MFR across all time bins in $\mathcal{B}$ in the current routing window. Recall that since the computations that are performed on each bin are independent, we run PD-MFR in parallel for each bin. Then, $T_{\text{exec}}^{\text{PD}-\text{MFR}} = \max_{b\in\mathcal{B}} T_{\text{exec}}^{\text{PD}-\text{MFR}}(b)$.

Let $T_{\text{pred}}^{\text{eMBB}}$ denote the length of the accurate prediction window for eMBB flows. Recall that T denotes the duration of the routing window. Then, we set T as follows:

$$T = T_{\text{pred}}^{\text{eMBB}} - T_{\text{exec}}^{\text{PD}-\text{MFR}} \tag{11}$$

Note that the optimization program must run for every routing window. The T set by the equation above represents the maximum value that T can have.

We also note that the parameter T_{bin} should be chosen as the resolution at which the generation times of the eMBB flows can be predicted.

4. Results

In this section, first, we present our simulation methodology. Second, we evaluate the performance of PD-MFR against the QoS-Shortest Path Algorithm (QoS-SPA), which serves as a benchmark.

4.1. Simulation methodology

We designed a simulation environment using the AGIS topology from the Internet Topology Zoo (Knight et al., 2011) to evaluate the performance of PD-MFR. This topology, consisting of 25 vertices and 56 edges, represents a mid-sized network and provides a balanced testbed for testing routing algorithms under heterogeneous traffic conditions. Alternative topologies are discussed later in Section 4.2.6.

First, we configured the network to operate with a routing window of $T = 30$ seconds divided into bins of $T_{bin} = 1$ second. This division allows for fine-grained routing decisions to adapt dynamically to traffic variations across the network. The routing window size of 30 seconds was chosen to balance the need for timely routing adjustments with computational efficiency, while the bin size of 1 second provides sufficient granularity to capture traffic dynamics and enable accurate forecasting. Source-destination pairs for all flow types–eMBB, mMTC, and URLLC–were assigned randomly across the topology, mimicking the distributed and unpredictable nature of real-world traffic environments.

Second, eMBB flows were modelled to represent high-throughput, delay-sensitive applications such as video streaming. To capture the predictable nature of eMBB traffic, we utilized a dataset from Heng et al. (2021), which contains 21 million packets of mobile traffic data collected from popular mobile applications. The dataset was divided into a number of segments equal to the number of eMBB flows generated in each simulation, and these segments were randomly assigned to source-destination pairs. The generation rates of these flows ranged from 100 Mbps to 1 Gbps, reflecting typical high-throughput applications. The generation rate of each aggregate eMBB flow, measured in bits per second, was used as the primary input for routing. In addition, the minimum

data rate requirement $r_{\min}^{(f)}$ for each flow $f \in \mathcal{F}_{eMBB}$ was set as $0.5g^{(f)}$, ensuring QoS requirements were met.

Third, we generated mMTC flows synthetically to simulate delay-tolerant IoT applications, such as smart meters and environmental sensors. To ensure efficient use of the network's residual capacity, a large volume of mMTC traffic (in bits) was assumed to be queued for delivery across each source-destination pair. This volume was divided evenly over the routing window, resulting in constant bit-rate mMTC flows that leveraged any unused capacity effectively while respecting their delay-tolerant nature.

Finally, URLLC flows were introduced to reflect critical, ultra-reliable, and low-latency applications such as remote surgery or autonomous vehicle communication. These flows were modelled dynamically, with both their arrival and termination times treated as random variables. This approach captured the sporadic and highly variable nature of URLLC traffic, ensuring that the simulation aligned with real-world requirements. For URLLC flows, stringent QoS requirements were imposed, with a maximum allowable latency of $\Delta^{(f)} = 20$ ms and a probability of error per edge of $\epsilon_{mn} = 10^{-20}$.

All edges in the AGIS topology were assigned a uniform capacity of 1 Gbps, representing a typical backbone network scenario. This uniform capacity ensures that the performance differences observed are due to the routing algorithm rather than varying link capabilities.

By combining these traffic models within the AGIS topology, our simulation setup provides a comprehensive environment for evaluating PD-MFR's ability to handle diverse QoS requirements under realistic conditions.

4.1.1. Forecasting methodology

The main task of the forecasters in PD-MFR is to predict the traffic demand (generation rate) of each aggregate eMBB flow in a particular time bin for the upcoming routing window. These predictions are then used in Stage A of PD-MFR to plan the route of each eMBB flow. By forecasting traffic for the upcoming routing window, the system can proactively allocate resources for optimal flow delivery, ensuring better QoS for both eMBB and other types of traffic (mMTC and URLLC). The inputs of the forecasters are chosen as past generation rates of each eMBB flow. The outputs are the predicted generation rates for each of these eMBB flows in the upcoming routing window. The reason for choosing the current set of inputs and outputs is to align the forecasting process with the specific needs of the PD-MFR algorithm. The selected inputs are critical for accurately predicting the future traffic generation rate of each eMBB flow, which directly impact route planning in Stage A. The outputs, consisting of predicted generation rate of each eMBB flow, enable PD-MFR to proactively allocate and reserve portions of the edge capacities on relevant route for each eMBB flow. In addition, these inputs and outputs ensure that the PD-MFR can anticipate traffic fluctuations between each source-destination pair. The selected forecasting algorithms should satisfy that the predicted generation rates are sufficiently accurate to enable PD-MFR to proactively plan the route of each flow. While perfect prediction accuracy is not required, the key criterion is that the forecasts allow the algorithm to effectively satisfy the QoS requirements for each flow from the time that the flow begins to the time that the flow ends.

We shall test the performance of the ARIMA and the MLP models in predicting the generation rates of eMBB flows. For ARIMA, we perform exhaustive search in order to find the

local-optimal values of the (p, d, q) parameters of this model. We set the search interval for each of these parameters as [0, 10]. We have found that the (p, d, q) value of (8, 1, 1) results in the minimum sMAPE within the search interval. For the MLP model, we performed exhaustive search in order to find the optimal number of hidden layers and number of neurons in each layer. We set the search interval of the number of hidden layers as [1, 5] and the search interval of the number of neurons in each layer as [1, 10]. We have found that the choice of three hidden layers that have (5, 7, 8) neurons respectively results in the minimum sMAPE within the search interval. For the MLP model, in order to minimize sMAPE, we used the *adam* optimizer. Furthermore, we used *ReLU* activation function for the hidden and output layers of MLP model. For both of the machine learning models, we perform 30-step ahead prediction. For forecasting, we utilized both ARIMA and MLP models trained using historical eMBB traffic data. Specifically, the first 70% of the dataset was used for training, and the remaining 30% was used for testing, aligning with our simulation methodology. This separation ensures that the models are evaluated on unseen data.

In the PD-MFR framework, the forecasters operate in real-time to generate predictions for the upcoming routing window. The model uses the most recent historical data to make predictions of the generation rate of each eMBB flow. Inference takes place before the routing window begins.

In addition, we present the values of the simulation parameters in Table 3.

4.1.2. Handling forecasting errors

To clarify the effect of forecasting errors on network behaviour, we note that such errors do not propagate across the network in the PD-MFR framework. If the forecaster overpredicts the demand of an eMBB flow and the reserved capacity in Stage A is insufficient, that flow is not admitted into the routing window in Stage B. Hence, in PD-MFR, an eMBB flow can be admitted only if a route has been successfully reserved in Stage A, and the actual generation rate of that flow in Stage B does not exceed the pre-reserved capacity along that route. This ensures that forecast errors do not lead to over-allocation. Furthermore, for each eMBB flow, route planning is made independently of other flows in Stage A. As a result, any forecasting error affects only that flow and does not influence the admission of other flows. Therefore, forecasting errors do not trigger widespread congestion or disruptions in unrelated parts of the network.

Table 3. Values of the simulation parameters.

Parameter	Value
T	30 s
T_{bin}	1 s
ϵ_{mn}	10^{-20}
c_{mn}	1 Gbps
$p_e^{(f)}$	10^{-5}
K	5
$\delta_{m\to n}$	0.5 ms
$\Delta^{(f)}$	20 ms
$r_{\min}^{(f)}$	0.5 $g^{(f)}$
(p, d, q)	(8, 1, 1)
Number of neurons of the three hidden layers of MLP	(5, 7, 8)

4.1.3. Flow generation methodology

We synthetically generated the URLLC data set as follows: For each such flow, the time bin in which the flow is generated (namely, the generation time of the flow) is chosen from a discrete uniform distribution over the set of time bins over a single routing window. Furthermore, the time bin in which the flow terminates (namely, the termination time of the flow) is chosen from a discrete uniform distribution between the generation time and the end of the routing window.[5]

4.1.4. Reactive benchmark: QoS-SPA

We compare the performance of PD-MFR against that of a reactive benchmark, which we have designed based on the state-of-the-art scheme SWAY that appears in Saha et al. (2018). We call the reactive benchmark the 'QoS-Shortest Path Algorithm' (QoS-SPA). Recall that in general, Yen's *K*-shortest path algorithm computes *K* paths in the order of a non-decreasing cost metric. In SWAY, for each delay-sensitive or loss-sensitive flow, the cost metric in Yen's *K*-shortest path algorithm is set to delay and loss, respectively. Furthermore, for each delay-sensitive or loss-sensitive flow, SWAY iterates through the list of *K* paths (in the order of the non-decreasing cost metric) in order to find the first path that satisfies the QoS constraints of that flow.

QoS-SPA sets the cost metric in Yen's *K*-shortest path algorithm for each flow to latency. If there are flows generated in the same time bin, QoS-SPA prioritizes URLLC flows over eMBB flows and eMBB flows over mMTC flows. The following steps of QoS-SPA are performed in the order of priority of flow *f* for the time bin in which flow *f* is generated: First, QoS-SPA computes *K* shortest paths from the source to the destination of flow *f* using Yen's *K*-Shortest Path Algorithm. (If the number of paths between $s(f)$ and $d(f)$ is less than *K*, then all of the paths between $s(f)$ and $d(f)$ are utilized for flow *f*.) Second, QoS-SPA iterates through the set of paths (in the order of the non-decreasing cost metric) until a path that satisfies the QoS requirements listed in (8)–(10) is found. Third, If no path satisfies the QoS requirements of flow *f*, QoS-SPA does not admit flow *f*. Otherwise, it sends flow *f* over the shortest path found (if such a path exists). If flow *f* is admitted, for all time bins in which flow *f* does not terminate, QoS-SPA continues to send flow *f* over the same path.

4.1.5. Computation platform

Finally, we note that our simulation environment was set up in Python 3.7 on Google Colab by choosing the accelerator as the NVIDIA T4 Graphical Processing Unit (GPU).

4.2. Performance evaluation

First, in Section 4.2.1 we discuss error metrics for forecasting schemes, which are ARIMA and MLP. In Section 4.2.2 we evaluate the performance of our PD-MFR algorithm in the presence of only eMBB and mMTC flows against QoS-SPA, which serves as a reactive benchmark. In Section 4.2.3, we add the URLLC flows and evaluate the performance of PD-MFR. In Section 4.2.4, we examine the execution time of PD-MFR in Stage A. In Section 4.2.5, we evaluate the execution time required to route each URLLC flow in Stage B, where multiple URLLC requests are handled reactively within the same routing

window. Finally, in Section 4.2.6, we discuss the performance of PD-MFR on alternative topologies.

Throughout this section, 'perfect forecasting' shall refer to the ideal situation in which the eMBB traffic forecast equals the actual value at every time instant. We shall report results under perfect forecasting in order to obtain a performance upper bound to forecasting schemes that are used in practice. Recall that $\beta \in [0, 1]$ denotes the coefficient of the buffer size under a practical forecasting scheme that is added to the reserved part of the capacity of each reserved edge on the topology for each flow $f \in \mathcal{F}_{eMBB}$. In contrast, in the case of perfect forecasting, no β buffer shall be allocated to eMBB flows since the forecast and the actual values are equal.

4.2.1. Discussion of error metrics

Before we proceed with the evaluation of network performance, we report the forecasting accuracy of the ARIMA and MLP models using traditional error metrics. These metrics include the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), Symmetric Mean Absolute Percentage Error (sMAPE) and the Pearson correlation coefficient. Table 4 summarizes the results.

We observe that the MLP model outperforms ARIMA across all reported metrics, achieving lower prediction errors. The MLP forecaster achieves significantly lower MSE and RMSE compared to ARIMA and also demonstrates a notably higher correlation coefficient (0.78 vs. 0.32), indicating a stronger linear relationship between predicted and actual values. Overall, MLP provides better alignment with the traffic trends and is more reliable for use within PD-MFR's predictive routing stage. In addition, note that the evaluation of PD-MFR is based not only on the forecasting metrics in Table 4, but also on how these predictions affect the routing performance presented in the following sections.

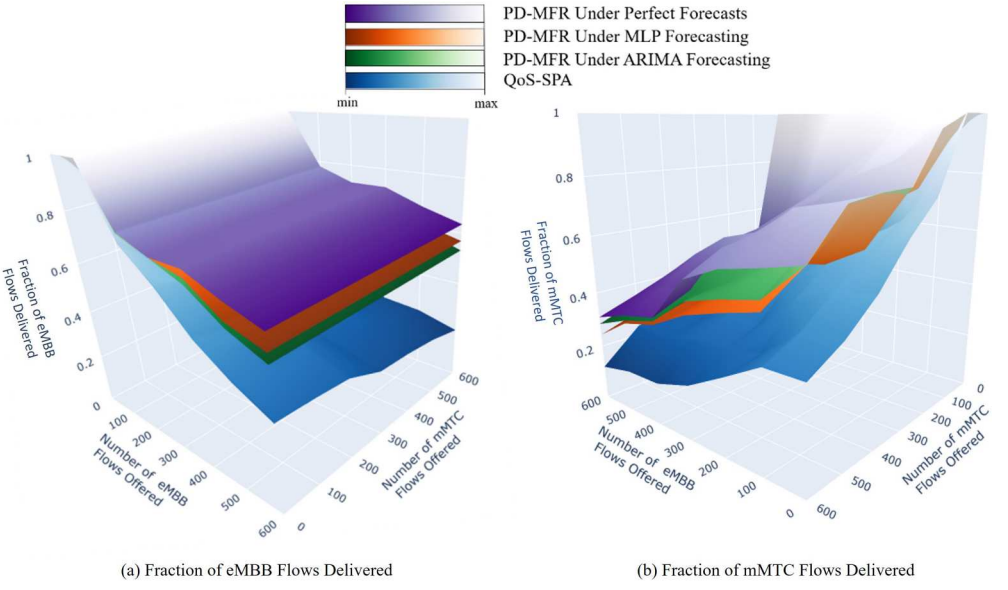
4.2.2. Performance of PD-MFR in the presence of eMBB and mMTC flows

We say that a flow is 'delivered' if and only if it the QoS requirements of that flow can be supported from the time that the flow begins to the time that the flow ends. In this subsection, we aim to observe the effect of forming predictions of eMBB flow generation rates on the fraction of eMBB flows delivered in the presence of only eMBB and mMTC flows. (Consequently, throughout this subsection, we set $\alpha = 1$, since no URLLC flows are generated.) We display our results on the AGIS topology (which has 25 vertices), which we have obtained from the Topology Zoo (Knight et al., 2011).

In Figure 7(a), we display the fraction of eMBB flows delivered as a function of the number of eMBB and mMTC flows offered under $\beta = 0.15$ for ARIMA and MLP forecasting. In Figure 7(a), the *x* axis refers to the number of eMBB flows offered; the *y* axis refers to the number of mMTC flows offered; and the *z* axis refers to the fraction of eMBB flows

Table 4. Performance metrics of the forecasting schemes for PD-MFR.

	ARIMA	MLP
MSE	3731.98	1406.25
RMSE	61.09	37.50
MAPE	401.92%	306.45%
1/2-sMAPE	61.06%	43.08%
Corrrelation Coefficient	0.32	0.78



(a) Fraction of eMBB Flows Delivered

(b) Fraction of mMTC Flows Delivered

Figure 7. Fraction of flows delivered as a function of the number of flows offered under buffer parameter $\beta = 0.15$. In both subfigures, we observe that PD-MFR under perfect forecasting consistently outperforms the reactive benchmark QoS-SPA by a wide margin. Moreover, PD-MFR maintains this advantage even when practical forecasting schemes such as ARIMA and MLP are used.

delivered. In this figure, we compare the performance of PD-MFR under perfect forecasts (the purple surface); PD-MFR under MLP forecasting (orange surface); PD-MFR under ARIMA forecasting (green surface); and QoS-SPA (blue surface). First, we see that PD-MFR under perfect forecasts has the highest fraction of eMBB flows delivered for each (eMBB, mMTC) offered flow pair on the *x*-*y* plane. The reason is that the performance of PD-MFR under perfect forecasts represents the case where the actual value of the generation rate of each eMBB flow is known in advance for each time bin in the routing window; hence, it shows the ultimate performance limit of PD-MFR.

Second, we see in Figure 7(a) that PD-MFR under MLP and ARIMA forecasting outperforms QoS-SPA. The reason is that PD-MFR utilizes predictions of the generation rates of eMBB flows while QoS-SPA sends the flows reactively. Third, we see that PD-MFR under MLP forecasting outperforms PD-MFR under ARIMA forecasting. The reason is that when we use MLP to predict the generation rate of each eMBB flow, the forecasting error is less compared to that of ARIMA. Fourth, as the number of eMBB flows offered increases, the fraction of eMBB flows delivered for PD-MFR under perfect, MLP and ARIMA forecasting remains close to each other while that of QoS-SPA has much lower performance. This shows that PD-MFR outperforms QoS-SPA even when practical forecasting schemes such as ARIMA and MLP are utilized by PD-MFR.

Although the predictions are formed only for eMBB flows, recall that the route of each delay-tolerant mMTC flow is also determined in Stage A of PD-MFR. Now, we aim to observe the effect of forming predictions of the eMBB flow generation rates on the fraction of mMTC flows delivered. To this end, in Figure 7(b), we plot the fraction of mMTC flows delivered on the *z* axis. Furthermore, the *x* axis refers to the number of eMBB

flows offered, and the *y* axis refers to the number of mMTC flows offered. In contrast with Figure 7(a), in this figure, PD-MFR under ARIMA forecasting outperforms PD-MFR under MLP forecasting. The reason is that mMTC flows utilize the remaining capacities on the edges of the topology that are left from the eMBB flows in each time bin. Since PD-MFR under MLP forecasting outperforms PD-MFR under ARIMA forecasting in Figure 7(a) for eMBB flows, on average, the remaining capacity on the topology that is left for mMTC flows to be routed is greater for PD-MFR under ARIMA forecasting than that under MLP forecasting. As a result, in Figure 7(b), a larger fraction of mMTC flows is delivered for PD-MFR under ARIMA forecasting than that under MLP forecasting.

For any given time bin, we define the 'residual capacity' of each edge as the part of the capacity of that edge that is not used by any of the eMBB, mMTC and URLLC flows. We define the 'residual sum capacity' as the sum of the residual capacities of all of the edges on the entire topology *averaged* over the set of time bins $\mathcal{B}$ of the routing window. (We average the sum residual capacities over the time bins in a routing window in order to be able to report a single metric in this regard for that routing window.) Furthermore, recall that β denotes the coefficient of the buffer size for each aggregate eMBB flow under a given practical forecasting scheme (as opposed to perfect forecasting which requires no such buffer). This buffer shall be used in Stage B by each eMBB flow if the predicted generation rate of that flow in Stage A is less than the actual generation rate of that flow in Stage B.

In Figure 8, in order to determine the effect of β on the residual sum capacity, we set β successively to a small and a large value, which are 0.15 and 0.5, respectively. It is crucial to examine the 'residual sum capacity' because this metric provides us information on the *amount* of delivered traffic, which the 'fraction of flows delivered' metric does not

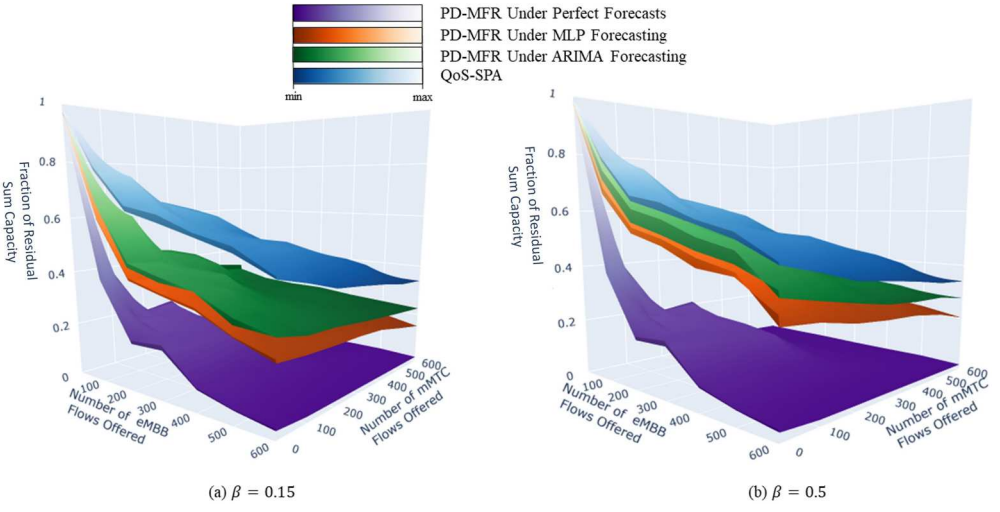


Figure 8. Fraction of residual sum capacity as a function of the number of eMBB and mMTC flows offered. Subfigures (a) and (b) correspond to buffer coefficients $\beta = 0.15$ and $\beta = 0.5$, respectively. In both cases, PD-MFR under MLP and ARIMA forecasting results in lower residual sum capacity compared to QoS-SPA, showing that PD-MFR achieves more efficient utilization of edge capacities, even when practical forecasting models are used and additional buffer reservations are applied.

provide. We define the 'fraction of residual sum capacity' as the residual sum capacity divided by the sum of the capacities of all of the edges on the topology.

In Figure 8, we display the fraction of residual sum capacity for (a) $\beta = 0.15$ and (b) $\beta = 0.5$. First, in Figure 8(a), we see that PD-MFR under perfect forecasts results in a lower fraction of residual sum capacity than that of PD-MFR under MLP and ARIMA forecasting. Second, in Figure 8(a), we see that MLP and ARIMA forecasting result in lower fraction of residual sum capacity than that of QoS-SPA; that is, the former are able to utilize the edge capacities better than the latter. Third, when we compare Figure 8(a,b), we observe that the fraction of residual sum capacity for PF-MFR under MLP and ARIMA forecasting in Figure 8(b) is at least 0.05 greater than each of those that appear in Figure 8(a). The reason is that in Figure 8(b), for each eMBB flow, we reserve a buffer over the capacity of each edge for that flow whose coefficient is $\beta = 0.5$, which is greater than that of $\beta = 0.15$ in Figure 7(a). (Note that the residual capacity under perfect forecasts is invariant across Figure 8(a,b) because β is not a parameter of perfect foreacasting; no buffer is allocated for eMBB flows under perfect forecasting.)

4.2.3. Performance of PD-MFR in the presence of eMBB, mMTC and URLLC flows

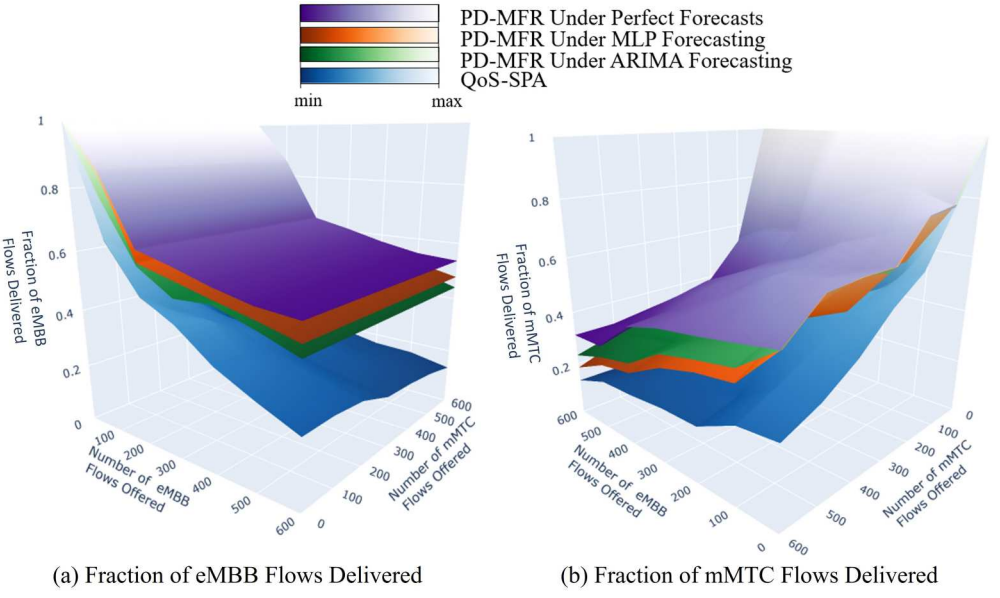
In this subsection, we measure the performance of PD-MFR in the presence of the traffic types eMBB, mMTC, and URLLC in order to determine the effect of URLLC flows on the fraction of eMBB and mMTC flows delivered. In addition, we aim to determine the fraction of URLLC flows delivered in the presence of all of these traffic types.

For each of the subplots in Figure 9, the number of aggregate URLLC flows offered is set to 20. (These flows are assigned at random to distinct source-destination pairs on the topology.) Furthermore, α and β are set to 0.85 and 0.15, respectively. Note that for each time bin in the routing window, $(1 - \alpha)$ times the capacity of each edge over the topology is now reserved for URLLC flows in Stage A of PD-MFR. In Figure 9(a), we display the fraction of eMBB flows delivered as a function of the number of eMBB and mMTC flows offered. We see that the fraction of eMBB flows delivered for PD-MFR under perfect, MLP and ARIMA forecasting in Figure 9(a) is approximately 0.1 lower than each of those that appear in Figure 7(a), respectively. The reason is that only α times the capacity of each edge over the topology is left for all of the eMBB and mMTC flows in the routing window in Figure 9(a) compared to that of Figure 7(a).

In addition, in Figure 9(a), as the number of mMTC flows offered increases, the fraction of eMBB flows delivered remains constant for PD-MFR under perfect forecasts, PD-MFR under MLP forecasting, and PD-MFR under ARIMA forecasting. (This is similar to that observed in Figure 7(a).) The reason is that in PD-MFR, the mMTC flows are not allowed to use the part of the capacity that has been reserved for eMBB flows.

Figure 9(b) displays the fraction of mMTC flows delivered. The fraction of mMTC flows delivered for PD-MFR under perfect, MLP, and ARIMA forecasting in Figure 9(b) is approximately 0.05 lower than each of those that appear in Figure 7(b), respectively. This shows that the trends exhibited by the fractions of eMBB and mMTC flows delivered across distinct forecasting schemes in Figure 7 still hold when URLLC flows are present on the network.

In Figure 10(a), we observe the fraction of URLLC flows delivered as a function of the number of eMBB and mMTC flows offered. First, we see that PD-MFR under perfect forecasts outperforms PD-MFR under MLP and ARIMA forecasting. However, there are points



(a) Fraction of eMBB Flows Delivered (b) Fraction of mMTC Flows Delivered

Figure 9. Fraction of flows delivered as a function of the number of flows offered under $\alpha = 0.85$, $\beta = 0.15$.The number of aggregate URLLC flows is fixed to 20. PD-MFR outperforms QoS-SPA in both subfigures. Under perfect forecasting, PD-MFR delivers the highest fraction of eMBB and mMTC flows. MLP-based forecasting outperforms ARIMA, highlighting the impact of forecasting accuracy.

at which PD-MFR under perfect, MLP and ARIMA forecasting have the same or very close values. Recall that we do not perform predictions for URLLC flows and that URLLC flows are routed reactively in Stage B of PD-MFR.

In this case, since URLLC flows utilize not only the fraction $(1 - \alpha)$ of the edge capacities that have been reserved for these flows but also the remaining capacities over each bin in the routing window at the end of Stage A of PD-MFR, the fraction of URLLC flows delivered decreases as a function of the number of eMBB and mMTC flows offered.

Second, in Figure 10(a), PD-MFR under MLP and ARIMA forecasting outperforms QoS-SPA even though both PD-MFR and QoS-SPA send URLLC flows reactively. The reason is that QoS-SPA routes each of the flows (including eMBB and mMTC types) reactively while PF-MFR only routes only URLLC flows reactively. Since $(1 - \alpha)$ times the capacity of each edge over the topology is reserved for the upcoming URLLC flows in Stage A of PD-MFR, more capacity is left on the edges of the topology to route URLLC flows in PD-MFR than in QoS-SPA.

Finally, in Figure 10, we present the effect of changing α on the fraction of URLLC flows delivered. To this end, in Figure 10(b), we set α to 0.5. Since $(1 - \alpha)c_{mn}$ for each $(m \rightarrow n) \in E$ is reserved for the upcoming URLLC flows for each time bin in the routing window in Stage A of PD-MFR, the fraction of URLLC flows delivered for PD-MFR under perfect, MLP and ARIMA forecasting in Figure 10(b) is greater than each of those that appear in Figure 10(a), respectively.

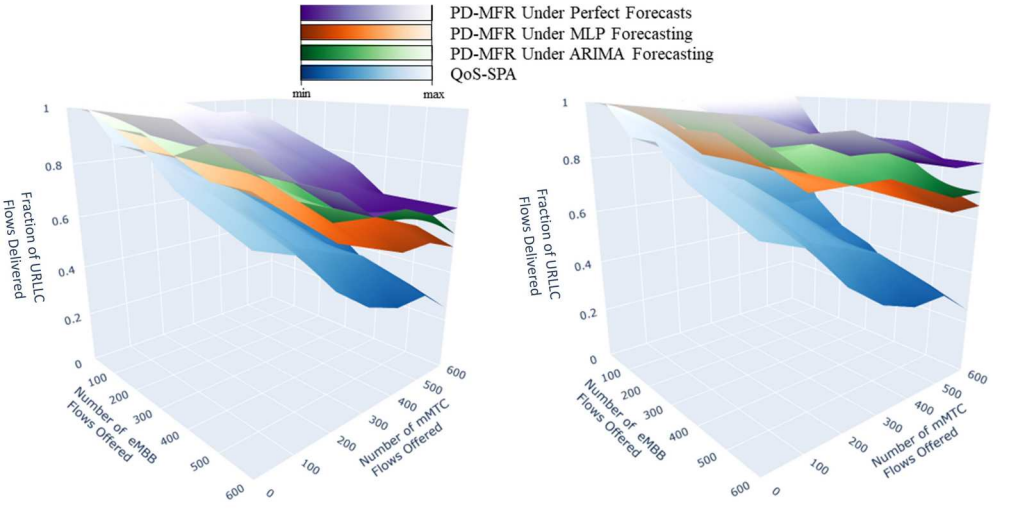


Figure 10. Fraction of URLLC flows delivered as a function of the number of flows offered under $\beta = 0.15$, for two different values of α, where $(1 - \alpha)$ denotes the fraction of edge capacity reserved for URLLC flows in Stage A. PD-MFR outperforms QoS-SPA in terms of the percentage of URLLC flows delivered. Additionally, in plot (b), the percentage of URLLC flows delivered under each forecasting scheme–perfect, MLP, and ARIMA–is consistently higher than the corresponding values in plot (a), highlighting the benefit of reserving more capacity for URLLC traffic under PD-MFR.

4.2.4. Execution time of PD-MFR

In this subsection, we measure the execution time of PD-MFR in Stage A for the AGIS topology. Recall that the routes of eMBB and mMTC flows are determined before the routing window in Stage A begins while URLLC flows are routed reactively after the routing window begins in Stage B. We present the execution time of PD-MFR in Stage A of PD-MFR in Figure 11. (We present the execution time of PD-MFR only for Stage A because in Stage B, only the reactive routing of URLLC flows in the routing window takes place, which has a negligible computation time compared to that of Stage A.) Recall that eMBB flows are given priority over mMTC flows. Therefore, for each time bin $b \in \mathcal{B}$, we run PD-MFR first for the eMBB flows, followed by mMTC flows that fall in that bin. Recall from Section 3.2.5 that in Stage A, the computations across the time bins in the routing window are run in parallel. As a result, the execution time of PD-MFR in Stage A is equal to $\max_{b \in \mathcal{B}} T_{\text{exec}}^{\text{PD}-\text{MFR}}(b)$.

First, in Figure 11, we see that the execution time of PD-MFR in Stage A increases almost linearly as a function of the number of mMTC and the number of eMBB flows offered. The fact that this growth is almost linear implies that PD-MFR is a practically feasible algorithm. We note that the execution time reaches approximately four seconds at most for the fully loaded network (when the number of aggregate flows offered reaches its maximum value). Hence, the duration T of the routing window (which was 30 seconds in our simulations) is reasonable. Moreover, since the execution time displayed in Figure 11 belongs to Stage A, none of the actual eMBB and mMTC flows waits for this execution time to expire in order for each of them to be delivered after the routing window begins in Stage B.

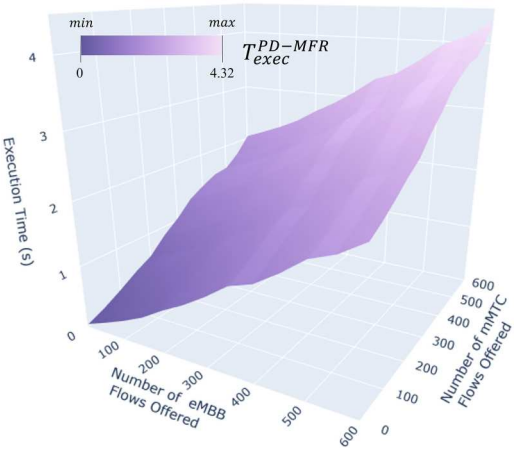


Figure 11. Execution time of Stage A of PD-MFR as a function of the number of offered eMBB and mMTC flows.

4.2.5. Execution time of reactive URLLC routing in stage B of PD-MFR

To further evaluate the computational overhead under URLLC constraints, we measure the execution time of each reactive routing operation performed in Stage B. Although Stage B does not involve solving mixed-integer programs, it still requires selecting feasible paths for newly generated URLLC flows within the current routing window. In Figure 12, the average execution time per URLLC flow over 30 independent trials is presented. In this figure, the *x* axis denotes the 20 individual URLLC flows indexed from *URLLC*0 to *URLLC*19, while the *y* axis shows the average execution time, in milliseconds, required to compute a complete routing decision for each URLLC flow.

Note that the execution time shown in each bar in the figure includes the time to run Yen's *K*-shortest path algorithm to find *K* paths ordered by increasing latency, and to evaluate these paths based on the latency, reliability, and capacity constraints of the

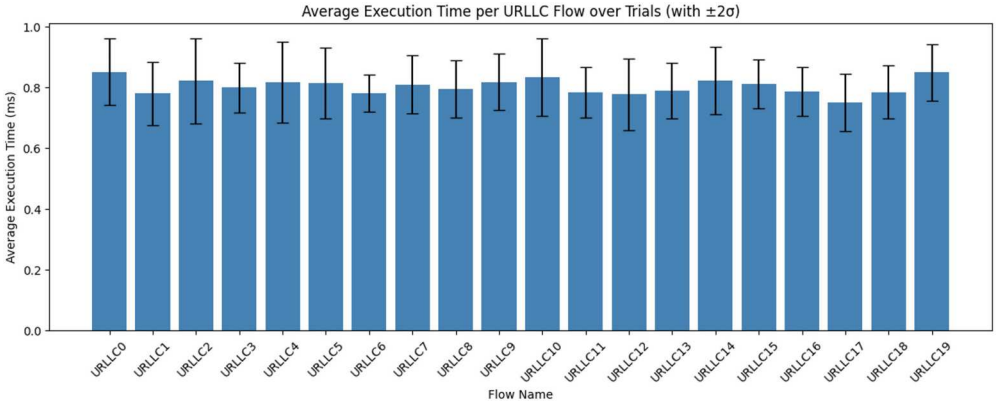


Figure 12. Average execution time per URLLC flow over 30 independent trials. Error bars represent two standard deviations ($\pm 2\sigma$) from the mean, reflecting the consistency of computational overhead for each admission decision. All execution times remain well within the sub-millisecond range, satisfying the real-time requirements of URLLC scenarios.

corresponding URLLC flow. Any path that fails to meet the end-to-end probability of error requirement is discarded. If K such paths can be found, the URLLC flow is admitted and routed. Hence, the values shown in the figure reflect the duration required to complete one reactive routing operation under PD-MFR for each URLLC flow.

All 20 URLLC flows represented in the figure co-exist in the network simultaneously during the routing window. These flows are routed reactively in Stage B, capturing a realistic scenario in which multiple URLLC requests are processed within the same routing window. In each trial, source and destination pairs for URLLC flows are randomly generated across the topology, introducing variability based on flow distance, resource availability, and network congestion. This randomness reflects practical deployment conditions and motivates the averaging over 30 independent runs.

The number of URLLC flows is fixed to 20 in this figure, consistent with the rest of our simulation setup. While URLLC traffic is typically limited in volume compared to eMBB and mMTC flows, using 20 simultaneous URLLC flows represents an extreme-case scenario designed to test the system's ability to meet tight latency and reliability constraints under heavy URLLC load. Moreover, each URLLC flow in our model represents an aggregate demand between a source-destination pair. That is, each URLLC flow reflects the total demand between a source-destination pair, rather than individual user-level transmissions. (Recall that this modelling approach is consistent with how traffic is modelled across all of our simulation scenarios.)

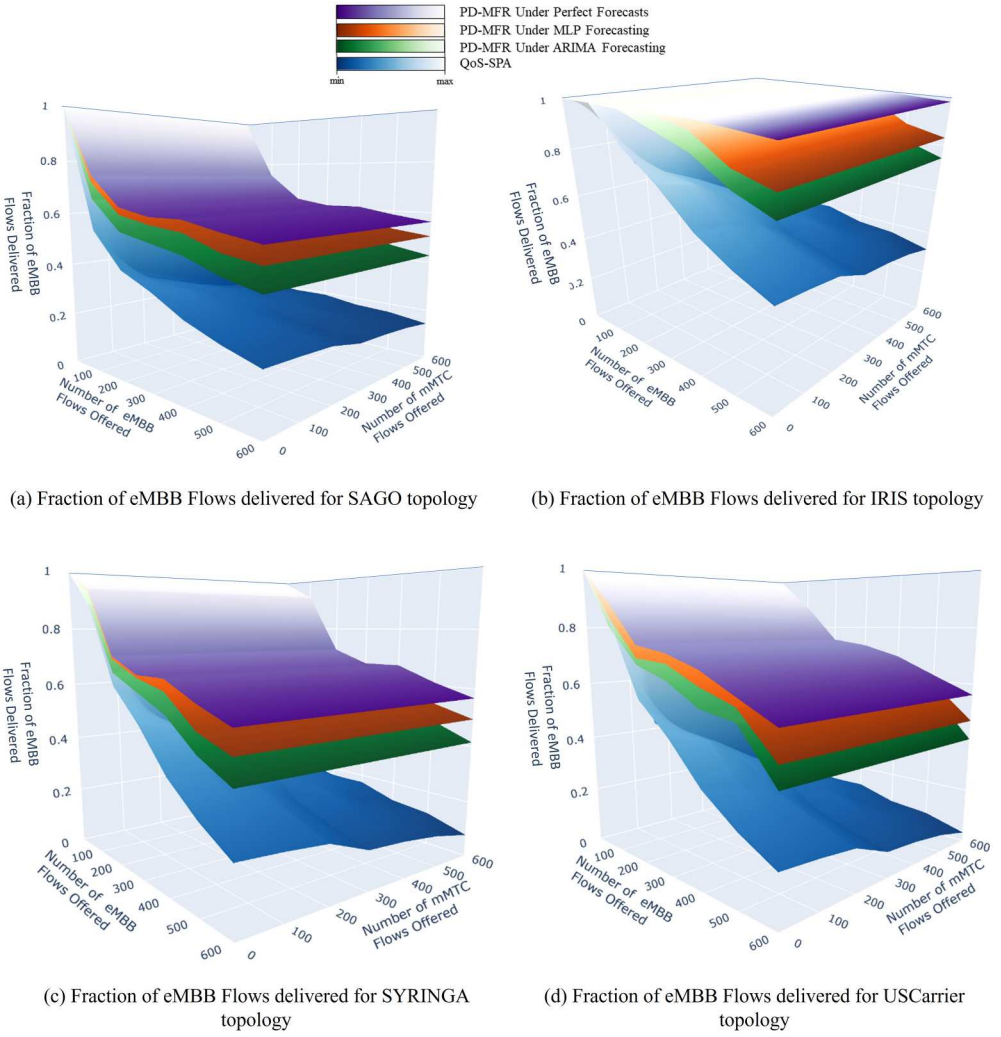
In this figure, we observe that the execution time remains consistently within sub-millisecond bounds for all URLLC flows, with low variability as shown by the error bars (±2 standard deviations). These results confirm that the reactive operations in Stage B remain computationally lightweight even under extreme flow densities, and can satisfy the stringent real-time requirements of URLLC applications. Hence, while Stage A handles the majority of the computational effort, the per-flow overhead in Stage B is demonstrably minimal.

4.2.6. Performance of PD-MFR on alternative topologies

In order to determine whether the results in the previous section generalize to small, medium, and large alternative topologies, we have examined the performance of PD-MFR on four more distinct topologies, which are SAGO, IRIS, SYRINGA and USCarrier (in the Topology Zoo) whose number of vertices is 18, 51, 74 and 158, respectively. We have examined the fractions of eMBB flows delivered as a function of the number of eMBB and mMTC flows offered on these topologies for PD-MFR under perfect forecasts; PD-MFR under MLP forecasting; PD-MFR under ARIMA forecasting; as well as QoS-SPA.

As shown in Figure 13, we have found that similar to that of the AGIS topology, PD-MFR under perfect forecasting significantly outperforms the reactive benchmark QoS-SPA in terms of the fraction of eMBB flows delivered for SAGO, IRIS, SYRINGA and USCarrier topologies. In addition, we have found that similar to that of the AGIS topology, PD-MFR under MLP and ARIMA forecasting also outperforms QoS-SPA significantly on these alternative topologies in terms of eMBB flows delivered. Similar to that of the AGIS topology, PD-MFR under MLP forecasting outperforms PD-MFR under ARIMA forecasting by a small amount in the number of eMBB flows delivered on these topologies.

In summary, we have found that the main conclusions reached in our examination of the results on the AGIS topology generalize to alternative topologies.



(a) Fraction of eMBB Flows delivered for SAGO topology

(b) Fraction of eMBB Flows delivered for IRIS topology

(c) Fraction of eMBB Flows delivered for SYRINGA topology

(d) Fraction of eMBB Flows delivered for USCarrier topology

Figure 13. Fraction of eMBB flows delivered as a function of the number of flows offered under $\alpha = 0.85$, $\beta = 0.15$.

5. Conclusion

We have presented a novel algorithm, called Predictive Dynamic Multi-Flow Routing (PD-MFR), for predictive QoS routing in software-defined networks (SDNs). Our algorithm contrasts sharply with the QoS routing algorithms for SDNs in the literature that do not form predictions and merely react to the currently generated flows. We have compared the performance of PD-MFR against the reactive benchmark QoS-SPA. Our results demonstrate that PD-MFR outperforms QoS-SPA significantly. Furthermore, we showed that the execution time of PD-MFR increases approximately linearly with respect to the number of eMBB and mMTC flows offered.

While PD-MFR relies on mixed-integer optimization models, these are solved prior to the start of each routing window, ensuring that real-time operations are unaffected.

This design mitigates scalability challenges and makes the algorithm suitable for a wide range of network sizes and traffic scenarios. In addition, the accuracy of the traffic forecasts impacts resource allocation, but our results show that even with imperfect predictions, the algorithm maintains its ability to satisfy QoS requirements effectively.

In our future work, we plan to develop an algorithm that adapts both the fraction reserved for URLLC traffic as well as the buffer size for eMBB flows to the traffic statistics. Furthermore, we plan to adapt the length of the QoS routing window to varying forecasting accuracy over time.

Notes

1. Since the new traffic types that arise at the intersection of the three basic ones for 5G are still in the proposal stage for 6G (Jiang et al., 2021), this paper utilizes the 5G traffic types. The framework of this paper, however, is general and can potentially be applied to new traffic types as 6G standards continue to emerge.
2. Since flows are required to be dynamically generated in real-world scenarios, in our simulations, we shall focus exclusively on the performance of our PD-MFR algorithm, which encompasses the SRA as a building block and successfully addresses dynamically generated flows.
3. It is expected that not all of the mMTC traffic volume that awaits at a source will be scheduled for routing over the upcoming routing window, Any amount that is left over is assumed to be used to create the offered traffic for the next routing window.
4. Note that $g^{(f)}$ and $\hat{g}^{(f)}$ denote the generation rate and the predicted generation rate of flow f for a particular time bin. The time index has been dropped in this subsection for convenience because we focus on a single time bin for the entire subsection
5. We do not consider the case in which the termination time is in the future routing windows. All realistic cases can be reduced to our formulation by setting the termination time to the end of the current routing window if the actual termination time exceeds the end of the current routing window. The same flow can be generated as a new flow at the beginning of the next routing window in order to examine the effects of that flow on that routing window. This process can be continued until the routing window that contains the actual termination time of that flow.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was funded by the European Union's Horizon 2020 Research and Innovation Program under the Marie Skłodowska-Curie grant agreement No. 846077, entitled "Quality of Service for the Internet of Things in Smart Cities via Predictive Networks".

ORCID

Buse Pehlivan http://orcid.org/0000-0001-6782-5291
Volkan Rodoplu http://orcid.org/0000-0002-9055-4159
Engincan Tunçay http://orcid.org/0009-0002-7563-7834
Dilara Eraslan http://orcid.org/0009-0002-0860-4522

References

Akçapınar, A., Gürer, Ö., & Rodoplu, V. (2022). ARIMA-based traffic forecasting for quality of service (QoS) flow routing in sixth generation (6G) networks. In *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1–5). IEEE.

Al-Jawad, A., Comşa, I.-S., Shah, P., Gemikonakli, O., & Trestian, R. (2021). An innovative reinforcement learning-based framework for quality of service provisioning over multimedia-based SDN environments. *IEEE Transactions on Broadcasting*, *67*(4), 851–867. https://doi.org/10.1109/TBC.2021.3099728

Al Jameel, M., Kanakis, T., Turner, S., Al-Sherbaz, A., & Bhaya, W. S. (2022). A reinforcement learning-based routing for real-time multimedia traffic transmission over software-defined networking. *Electronics*, *11*(15), 2441. https://doi.org/10.3390/electronics11152441

Alsenwi, M., Tran, N. H., Bennis, M., Bairagi, A. K., & Hong, C. S. (2019). eMBB-URLLC resource slicing: A risk-sensitive approach. *IEEE Communications Letters*, *23*(4), 740–743. https://doi.org/10.1109/COML.4234

Alsenwi, M., Tran, N. H., Bennis, M., Pandey, S. R., Bairagi, A. K., & Hong, C. S. (2021). Intelligent resource slicing for eMBB and URLLC coexistence in 5G and beyond: A deep reinforcement learning based approach. *IEEE Transactions on Wireless Communications*, *20*(7), 4585–4600. https://doi.org/10.1109/TWC.2021.3060514

Atawia, R., Hassanein, H. S., Abou-Zeid, H., & Noureldin, A. (2017). Robust content delivery and uncertainty tracking in predictive wireless networks. *IEEE Transactions on Wireless Communications*, *16*(4), 2327–2339. https://doi.org/10.1109/TWC.2017.2662685

Atawia, R., Hassanein, H. S., & Noureldin, A. (2017). Optimal and robust QoS-aware predictive adaptive video streaming for future wireless networks. In *GLOBECOM 2017 IEEE Global Communications Conference* (pp. 1–6). IEEE.

Awad, M. K., Ahmed, M. H. H., Almutairi, A. F., & Ahmad, I. (2021). Machine learning-based multipath routing for software defined networks. *Journal of Network and Systems Management*, *29*(2), 1–30. https://doi.org/10.1007/s10922-020-09583-4

Awan, I., Younas, M., & Naveed, W. (2014). Modelling QoS in IoT applications. In *2014 17th International Conference on Network-Based Information Systems* (pp. 99–105). IEEE.

Bairagi, A. K., Munir, M. S., Alsenwi, M., Tran, N. H., Alshamrani, S. S., Masud, M., Han, Z., & Hong, C. S. (2020). Coexistence mechanism between eMBB and uRLLC in 5G wireless networks. *IEEE Transactions on Communications*, *69*(3), 1736–1749. https://doi.org/10.1109/TCOMM.2020.3040307

Bouzidi, E. H., Outtagarts, A., Langar, R., & Boutaba, R. (2021). Deep Q-network and traffic prediction based routing optimization in software defined networks. *Journal of Network and Computer Applications*, *192*, 103181. https://doi.org/10.1016/j.jnca.2021.103181

Brown, J., & Khan, J. Y. (2015). A predictive resource allocation algorithm in the LTE uplink for event based M2M applications. *IEEE Transactions on Mobile Computing*, *14*(12), 2433–2446. https://doi.org/10.1109/TMC.2015.2398447

Casas-Velasco, D. M., Rendon, O. M. C., & Fonseca, N. L. S. (2020). Intelligent routing based on reinforcement learning for software-defined networking. *IEEE Transactions on Network and Service Management*, *18*(1), 870–881. https://doi.org/10.1109/TNSM.4275028

Casas-Velasco, D. M., Rendon, O. M. C., & Fonseca, N. L. (2021). DRSIR: A deep reinforcement learning approach for routing in software-defined networking. *IEEE Transactions on Network and Service Management*, *19*(4), 4807–4820. https://doi.org/10.1109/TNSM.2021.3132491

Chen, W.-E., Fan, X.-Y., & Chen, L.-X. (2019). A CNN-based packet classification of eMBB, mMTC and URLLC applications for 5G. In *2019 International Conference on Intelligent Computing and Its Emerging Applications (ICEA)* (pp. 140–145). IEEE.

Chen, Y.-R., Rezapour, A., Tzeng, W.-G., & Tsai, S.-C. (2020). Rl-routing: An SDN routing algorithm based on deep reinforcement learning. *IEEE Transactions on Network Science and Engineering*, *7*(4), 3185–3199. https://doi.org/10.1109/TNSE.6488902

Cong, P., Zhang, Y., Liu, Z., Baker, T., Tawfik, H., Wang, W., Xu, K., Li, R., & Li, F. (2021). A deep reinforcement learning-based multi-optimality routing scheme for dynamic IoT networks. *Computer Networks*, *192*, Article 108057. https://doi.org/10.1016/j.comnet.2021.108057

Dai, B., Cao, Y., Wu, Z., & Xu, Y. (2022). IQoR-LSE: An intelligent QoS on-demand routing algorithm with link state estimation. *IEEE Systems Journal*, *16*(4), 5821–5830. https://doi.org/10.1109/JSYST.2022.3149990

Egilmez, H. E., Civanlar, S., & Tekalp, A. M. (2012). An optimization framework for QoS-enabled adaptive video streaming over openflow networks. *IEEE Transactions on Multimedia*, *15*(3), 710–715. https://doi.org/10.1109/TMM.2012.2232645

Egilmez, H. E., Dane, S. T., Bagci, K. T., & Tekalp, A. M. (2012). OpenQoS: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks. In *Proceedings of the 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference* (pp. 1–8). IEEE.

Eyobu, O. S., & Edwinah, K. (2023). A deep learning-based routing approach for wireless mesh backbone networks. *IEEE Access*, *11*, 49509–49518. https://doi.org/10.1109/ACCESS.2023.3277431

Gunavathie, M., & Umamaheswari, S. (2023). Traffic-aware optimal routing in software defined networks by predicting traffic using neural network. *Expert Systems with Applications*, *239*, 122415. https://doi.org/10.1016/j.eswa.2023.122415

Heng, Y., Chandrasekhar, V., & Andrews, J. G. (2021). Utmobilenettraffic2021: A labeled public network traffic dataset. *IEEE Networking Letters*, *3*(3), 156–160. https://doi.org/10.1109/LNET.2021.3098455

Jiang, W., Han, B., Habibi, M. A., & Schotten, H. D. (2021). The road towards 6G: A comprehensive survey. *IEEE Open Journal of the Communications Society*, *2*, 334–366. https://doi.org/10.1109/OJCOMS

Juttner, A., Szviatovski, B., Mécs, I., & Rajkó, Z. (2001). Lagrange relaxation based method for the QoS routing problem. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)* (Vol. 2, pp. 859–868). IEEE.

Khambari, N., Ghita, B., & Sun, L. (2017). QoE-driven video enhancements in wireless networks through predictive packet drops. In *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (pp. 355–361). IEEE.

Knight, S., Nguyen, H. X., Falkner, N., Bowden, R., & Roughan, M. (2011). The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, *29*(9), 1765–1775. https://doi.org/10.1109/JSAC.2011.111002

Li, M., Guan, X., Hua, C., Chen, C., & Lyu, L. (2018). Predictive pre-allocation for low-latency uplink access in industrial wireless networks. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications* (pp. 306–314). IEEE.

Li, L., Li, S., & Zhao, S. (2014). QoS-aware scheduling of services-oriented internet of things. *IEEE Transactions on Industrial Informatics*, *10*(2), 1497–1505. https://doi.org/10.1109/TII.2014.2306782

Liu, Y., Niu, D., & Li, B. (2016). Delay-optimized video traffic routing in software-defined interdatacenter networks. *IEEE Transactions on Multimedia*, *18*(5), 865–878. https://doi.org/10.1109/TMM.2016.2538718

Llopis, J. M., Pieczerak, J., & Janaszka, T. (2016). Minimizing latency of critical traffic through SDN. In *2016 IEEE International Conference on Networking, Architecture and Storage (NAS)* (pp. 1–6). IEEE.

Montazerolghaem, A., & Yaghmaee, M. H. (2020). Load-balanced and QoS-aware software-defined internet of things. *IEEE Internet of Things Journal*, *7*(4), 3323–3337. https://doi.org/10.1109/JIoT.6488907

Nakip, M., Rodoplu, V., Güzeliş, C., & Eliiyi, D. T. (2019). Joint forecasting-scheduling for the internet of things. In *2019 IEEE Global Conference on Internet of Things (GCIoT)* (pp. 1–7). IEEE.

Njah, Y., Pham, C., & Cheriet, M. (2020). Service and resource aware flow management scheme for an SDN-based smart digital campus environment. *IEEE Access*, *8*, 119635–119653. https://doi.org/10.1109/Access.6287639

Rodoplu, V., Nakip, M., Qorbanian, R., & Eliiyi, D. T. (2020). Multi-channel joint forecasting-scheduling for the internet of things. *IEEE Access*, *8*, 217324–217354. https://doi.org/10.1109/Access.6287639

Rodoplu, V., Nakıp, M., Eliiyi, D. T., & Güzeliş, C. (2020). A multiscale algorithm for joint forecasting–scheduling to solve the massive access problem of IoT. *IEEE Internet of Things Journal*, *7*(9), 8572–8589. https://doi.org/10.1109/JIoT.6488907

Saha, N., Bera, S., & Misra, S. (2018). Sway: Traffic-aware QoS routing in software-defined IoT. *IEEE Transactions on Emerging Topics in Computing*, *9*(1), 390–401. https://doi.org/10.1109/TETC.6245516

Seyfollahi, A., Taami, T., & Ghaffari, A. (2023). Towards developing a machine learning-metaheuristic-enhanced energy-sensitive routing framework for the internet of things. *Microprocessors and Microsystems*, *96*, Article 104747. https://doi.org/10.1016/j.micpro.2022.104747

Srivastava, V., & Pandey, R. S. (2021). Machine intelligence approach: To solve load balancing problem with high quality of service performance for multi-controller based software defined network. *Sustainable Computing: Informatics and Systems*, *30*, Article 100511.

Sun, P., Hu, Y., Lan, J., Tian, L., & Chen, M. (2019). Tide: Time-relevant deep reinforcement learning for routing optimization. *Future Generation Computer Systems*, *99*, 401–409. https://doi.org/10.1016/j.future.2019.04.014

Sun, W., Wang, Z., & Zhang, G. (2021). A QoS-guaranteed intelligent routing mechanism in software-defined networks. *Computer Networks*, *185*, 107709. https://doi.org/10.1016/j.comnet.2020.107709

Tang, F., Fadlullah, Z. M., Mao, B., & Kato, N. (2018). An intelligent traffic load prediction-based adaptive channel assignment algorithm in SDN-IoT: A deep learning approach. *IEEE Internet of Things Journal*, *5*(6), 5141–5154. https://doi.org/10.1109/JIOT.2018.2838574

Wang, Z., & Crowcroft, J. (1996). Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, *14*(7), 1228–1234. https://doi.org/10.1109/49.536364

Wang, X., Hu, J., Lin, H., Garg, S., Kaddoum, G., Piran, M. J., & Hossain, M. S. (2021). QoS and privacy-aware routing for 5G-enabled industrial internet of things: A federated reinforcement learning approach. *IEEE Transactions on Industrial Informatics*, *18*(6), 4189–4197. https://doi.org/10.1109/TII.2021.3124848

Wen, J., Sheng, M., Li, J., & Huang, K. (2020). Assisting intelligent wireless networks with traffic prediction: Exploring and exploiting predictive causality in wireless traffic. *IEEE Communications Magazine*, *58*(6), 26–31. https://doi.org/10.1109/MCOM.35

Xu, S., Wang, X., Yang, G., Ren, J., & Wang, S. (2020). Routing optimization for cloud services in SDN-based internet of things with TCAM capacity constraint. *Journal of Communications and Networks*, *22*(2), 145–158. https://doi.org/10.1109/JCN.5449605

Xu, C., Zhuang, W., & Zhang, H. (2020). A deep-reinforcement learning approach for SDN routing optimization. In *Proceedings of the 4th International Conference on Computer Science and Application Engineering* (pp. 1–5). ACM.

Yao, C., Guo, J., & Yang, C. (2016). Achieving high throughput with predictive resource allocation. In *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)* (pp. 768–772). IEEE.

Yao, H., Yuan, X., Zhang, P., Wang, J., Jiang, C., & Guizani, M. (2019). Machine learning aided load balance routing scheme considering queue utilization. *IEEE Transactions on Vehicular Technology*, *68*(8), 7987–7999. https://doi.org/10.1109/TVT.25

Yen, J. Y. (1971). Finding the k shortest loopless paths in a network. *Management Science*, *17*(11), 712–716. https://doi.org/10.1287/mnsc.17.11.712

Yu, C., Lan, J., Guo, Z., & Hu, Y. (2018). DROM: Optimizing the routing in software-defined networks with deep reinforcement learning. *IEEE Access*, *6*, 64533–64539. https://doi.org/10.1109/ACCESS.2018.2877686

Yu, T.-F., Wang, K., & Hsu, Y.-H. (2015). Adaptive routing for video streaming with QoS support over SDN networks. In *2015 International Conference on Information Networking (ICOIN)* (pp. 318–323). IEEE.

Yuan, X., Yao, H., Wang, J., Mai, T., & Guizani, M. (2021). Artificial intelligence empowered QoS-oriented network association for next-generation mobile networks. *IEEE Transactions on Cognitive Communications and Networking*, *7*(3), 856–870. https://doi.org/10.1109/TCCN.2021.3065463