Author/s:
Roy, D;Rao, AS;Alpcan, T;Das, G;Palaniswami, M

Title:
Achieving QoS for bursty uRLLC applications over passive optical networks

Date:
2022-05-01

Citation:
Roy, D., Rao, A. S., Alpcan, T., Das, G. & Palaniswami, M. (2022). Achieving QoS for bursty uRLLC applications over passive optical networks. Journal of Optical Communications and Networking, 14 (5), pp.411-425. https://doi.org/10.1364/JOCN.446872.

Persistent Link:
https://hdl.handle.net/11343/312999

Achieving QoS for Bursty uRLLC Applications over Passive Optical Networks

DIBBENDU ROY[1,2,*], ARAVINDA S. RAO[1], TANSU ALPCAN[1], GOUTAM DAS[2], AND MARIMUTHU PALANISWAMI[1]

[1] Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville, VIC 3010, Australia
[2] G.S Sanyal School of Telecommunication, Indian Institute of Technology (IIT) Kharagpur, India
[*] Corresponding author: dibbendur@student.unimelbedu.au

Compiled May 13, 2022

Emerging real-time applications such as those classified under ultra-reliable low latency (uRLLC) generate bursty traffic and have strict Quality of Service (QoS) requirements. Passive Optical Network (PON) is a popular access network technology, which is envisioned to handle such applications at the access segment of the network. However, the existing standards cannot handle strict QoS constraints for such applications. The available solutions rely on instantaneous heuristic decisions and maintain QoS constraints (mostly bandwidth) in an average sense. Existing proposals in generic networks with optimal strategies are computationally complex and are therefore not suitable for uRLLC applications. This paper presents a novel computationally-efficient, far-sighted bandwidth allocation policy design for facilitating bursty uRLLC traffic in a PON framework while satisfying strict QoS (age of information/delay and bandwidth) requirements. To this purpose, first we design a delay-tracking mechanism which allows us to model the resource allocation problem from a control-theoretic viewpoint as a Model Predictive Control (MPC). MPC helps in taking far-sighted decisions regarding resource allocations and captures the time-varying dynamics of the network. We provide computationally efficient polynomial-time solutions and show its implementation in the PON framework. Compared to existing approaches, MPC can improve delay violations by 15% and 45% at loads of 0.8 and 0.9 respectively for delay-constrained applications of 1ms and 4ms. Our approach is also robust to varying traffic arrivals.

http://dx.doi.org/10.1364/ao.XX.XXXXXX

1. INTRODUCTION

Modern networking applications such as Augmented Reality/Virtual Reality (AR/VR), haptic communication and autonomous vehicles [1–4] are typical use cases for real-time IoT. International Telecommunication Union broadly categorize mission critical applications under ultra-reliable low latency (uRLLC) which require strict delay constraints to be satisfied with high reliability of 99.99%. These applications demand real-time service with stringent delay constraints in the order of sub milliseconds along with high bandwidth requirements. This is also seen in industrial applications, such as the Smart Grid (3-20 ms) and Smart Factory (0.25-10 ms) settings [5–7]. It has already been identified that the traffic pattern [8–10] for these applications do not resemble those of simple voice/video and data services, currently catered by existing networking standards [11, 12]. These applications are event driven and exhibit bursty traffic situations for which existing resource allocation standards are inefficient. This paper aims at designing a novel mechanism to obtain optimal resource allocation policies for bursty, delay-constrained and bandwidth-demanding applications over access networks.

Among the existing access networking technologies, Passive Optical Networks (PONs) have been identified as a promising solution for the high speed access network owing to its high bandwidth and low energy consumption [13–18]. A PON (see Fig. 1) typically consists of ONUs (connected to users). The fibers from ONUs are aggregated at a passive power splitter called remote node. A feeder fiber with high operating bandwidth connects the remote node to a central office called Optical Line Terminal (OLT). To facilitate emerging uRLLC applications

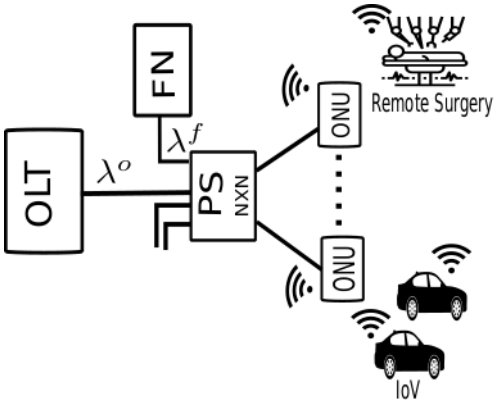


Fig. 1. Optical Distribution Network to facilitate applications like remote surgery and internet of vehicles (IoVs). Fog Node (FN) and OLT use different wavelengths for communicating with ONUs. FN is plugged to any one of many available ports of an $N \times N$ passive power splitter (PS).

in a PON based framework, a fog/edge node can be plugged at the remote node (to reduce round-trip delay compared to plugging at OLT) in an overlay fashion [13, 17, 19] as shown in Fig. 1. In such a framework, similar to the OLT, fog node has the important task of allocating bandwidth to the ONUs so that the strict Quality of Service (QoS) requirements of uRLLC applications can be met. Important QoS parameters may include delay, age of information (AoI)[20], bandwidth/throughput and jitter. Evidently, the allocation decisions at a given instant affects future decisions. To design optimal policies satisfying strict QoS constraints, the following requirements must be taken into consideration:

- The designed policy must be able to handle the bursty traffic nature of modern networking applications
- These policies must be far-sighted i.e., designed with a purview of the future and overall system dynamics.
- For real-time implementations, the algorithm to find such policies must be computationally tractable.

In PON, the existing standards [11, 12] do not have suitable mechanisms to deal with the QoS constraints under bursty traffic conditions. The existing proposals which are employed by the OLT for managing QoS (bandwidth) requirements [11, 12, 21], may be applied at a fog node. These proposals, similar to the available standards, are based on broad traffic classification and employ heuristics based on the currently available information and only consider delays in average sense. Thus, the existing policies cannot handle bursty traffic with strict QoS requirements and are short-sighted.

In past decade, several methods have been proposed to maximize throughput in a network. Under simplifying assumptions, they come up with policies which are provably optimal [22–24]. However such policies perform poorly when strict delay constraints are to be maintained [24]. As pointed out in [24], in order to maintain strict delay constraints and design far-sighted policies, one should consider solving Markov Decision Process (MDP) based models with large state spaces, which are computationally intractable. Although the authors show a reduction of their formulated problem to an LP, the reduction is based on assumptions of relaxation of binary decisions to randomized policies and average capacity constraints. In practice, such linear relaxations may not be optimal. Further, the authors demonstrate that optimality is lost while considering peak capacity constraints. Hence, it is evident that, the resource allocation problem under QoS constraints leads to complex formulations and computationally intractable policies. Due to the high complexity of computing optimal allocation policies, most resource allocation rules are based on heuristic decisions inherently depend on currently available information.

In summary, *to the best of our knowledge, the existing standards cannot handle strict QoS constraints for applications generating bursty traffic. The available proposals are short-sighted heuristics and maintain QoS constraints (mostly bandwidth) in average sense. The proposals with optimal strategies are computationally complex and are not suitable for uRLLC applications*. Thus, it is important to model and design allocation policies under strict delay constraints which are far-sighted, easy to compute and are provably optimal. A detailed literature review is provided in Section B identifying the drawbacks of existing proposals.

To meet the stringent delay constraints of real-time Internet and uRLLC applications in a polling based wire-line system (like PON), we develop a generic delay-tracking mechanism by employing virtual queues. This allows us to pose the resource allocation problem from a control theoretic perspective as a Model Predictive Control (MPC) problem. In contrast to MDP formulations, the MPC problem does not have stochastic state-action relations, leading to reduced state-action space. Also, wire-line systems are subject to peak capacity constraints which presents certain constraints while obtaining policies. MPC helps in taking far-sighted decisions regarding resource allocation rather than short-sighted ones (decisions based only on currently available information) and captures the time-varying dynamics of the network. The MPC problem turns out to be an Integer Linear Program (ILP). We prove that the ILP for the short-sighted scenario can be solved by an equivalent max-flow problem [25] which has several computationally efficient implementations. For the generic MPC problem, we show that linear program (LP) relaxation of the ILP yields optimal solutions. This is a significant reduction in complexity which is desired for providing delay-constrained services as the problem can be solved in polynomial time. We employ the developed technique in a PON framework as shown in Fig.1. Although we apply the methodology in a PON framework, the solution strategies would be applicable for any wire-line polling based network system.

A. Contributions of the paper

The main contributions of the paper can be summarized as:

- We present a delay-tracking mechanism with help of virtual queues enabling us to formulate the problem of resource allocation, subject to delay and bandwidth constraints as an MPC.
- We show that the short-sighted scenario of the problem can be solved by an equivalent max-flow representation while the generic far-sighted problem can be solved by an equivalent LP. Thus, efficient implementation of the MPC is guaranteed with low complexity.
- We show how our developed model can be implemented over an overlay PON.
- Compared to existing proposals, our results show significant improvement in reducing in percentage of delay violations by approximately 15% and increasing the throughput

at the cost of average delay performance. Our approach is robust to varying traffic arrivals.

B. Relevant Works and Literature Gaps

We briefly mention the relevant literature in context of resource allocation. As discussed in Section 1, we classify the available literature in three categories - available standards for PON 1)available standards for PON 2) specific to PON and emerging applications 3) general networking. The Giga-bit PON (GPON) [11] and Ethernet PON (EPON) [12] are the two available PON standards. Although these differ majorly in terms of specifications (like frame structures and data rates), the philosophy of dynamic bandwidth allocation in both are similar. In both GPON and EPON, OLT allocates bandwidth using a polling mechanism where the allocation decisions are sent in the downstream. The ONUs send the status of their buffers and QoS requirements along with their data transmissions in the upstream. In GPON, ONUs are equipped with virtual buffers called transmission containers (T-CONTs) for different kinds of services. However, for strict delay requirements (real-time), only constant bit-rate (CBR) traffic can be accommodated. Same is the case for EPON, where traffic is classified into three broad categories: Expedited Forwarding (EF), Assured Forwarding (AF) and Best Effort (BE). Each ONU has three virtual buffers classifying its traffic in either of the three. EF deals with CBR (similar to GPON) while AF deals with assured bandwidth guarantees without strict delay guarantees.

The existing proposals in PON for real-time Internet and Emerging applications [17, 26, 27] mostly deal with network architecture and present mere feasible allocation propositions based on available standards. The authors of [16, 17] propose a decentralized protocol which employs an out of band (OOB) wavelength for offloading fog data. In [17], each ONU uses the out of band wavelength for offloading its fog node upstream only during its turn for OLT upstream. Hence, the amount of bandwidth allocated for fog upstream is restricted by the duration of OLT upstream transmission. The major drawback of such a scheme is that the fog node upstream scheduling does not depend on the fog node-traffic, instead it depends on the OLT traffic. A more justified treatment was provided by the authors of [26], where they employ slicing of the available bandwidth and reserve a portion of the bandwidth for fog services. The amount to be reserved could be decided by means of prediction or learning mechanisms. Also, the fog services (mentioned as migratory services by the authors) are prioritized over the normal best effort services.

In [27], like most prevalent bandwidth allocation schemes, the authors propose a heuristic DBA algorithm based on reservation along with some decentralization (OLT does not resolve collision) which requires contention resolution among ONUs (ONUs compete to send REPORT message and whichever ONU gets to send, its uRLLC data is cleared). A significant amount of time is wasted in the process specifically at high loads. As mentioned by the authors themselves, the scheme they employed suffers from significant delays at high loads which is why such decentralization efforts have mostly failed to gain attention in optical networks (time wastage is equivalent to bandwidth wastage and since optical networks operate at very high bandwidth, this is detrimental). Further, introducing contention implies sacrificing reliability of transmission, which is strictly opposite to what uRLLC stands for. In the simulations, the authors considered that only three percent of their traffic is uRLLC and have considered poisson traffic which is generally not the case as pointed out in [8–10].

In networking literature, several works discuss the problem of designing routing, scheduling and allocation policies while maximizing throughput. The authors of [24] provide a comprehensive idea of the available works in the area and suggest that under unconstrained scenarios certain policies have been proved to be throughput optimal. They also point out that most of the existing research has been focused towards maximizing throughput while ensuring delay constraints require formulating the problem as an MDP with full network state information. The authors have considered a generic networking scenario with routing and power allocation policies formulated as a Constrained Markov Decision Process (CMDP) where the average power is constrained by an upper bound. This constraint can be equivalently modeled as an average capacity constraint. The authors relax the problem with help of randomization and probabilistic policies and show that the resulting optimization can be solved by a LP. Further, the authors point out that on invoking peak link capacity constraints, the solution becomes sub-optimal due to additional coupling in the variables introduced by the constraint. Also, they do not consider capacity constraints on flows (or classes in our model) which requires further investigation of their model. In general, on invoking desirable capacity constraints, the problem remains computationally intractable and not suitable for emerging applications.

In summary, the existing standard protocols, new proposals and networking literature suffer from the following shortcomings:

- Existing standards based on broad classification of the traffic will not be suitable for bursty traffic with strict delay constraints. A generic model for satisfying different delay and bandwidth requirements is currently absent.
- All existing proposals take short-sighted heuristic decisions and have considered delay constraints in an average sense. Evidently, such proposals fail to guarantee strict constraints and are non-optimal.
- To satisfy strict delay constraints, a complex MDP based model is usually developed. Optimal policies could be obtained on relaxing the model with randomized policies and under average capacity constraints. Under peak capacity constraints, the problem remains computationally intractable and unsuitable for uRLLC applications.

The rest of the paper is organized as follows. We describe the considered system model in Section 2 which contains details of our proposed delay-tracking mechanism followed by a brief description of control theoretic tool - MPC. Section 3 poses the resource allocation problem as an MPC followed by its solution in Section 4. The details of how the obtained solution can be employed in a PON framework is discussed in Section 5. Finally, we present our simulation results in Section 6 and conclude the paper with insightful remarks and future directions in Section 7.

2. PON SYSTEM MODEL

We consider a PON with operating bandwidth of B bits per second. IoT services have different QoS requirements which are broadly classified as certain classes of service (CoS) [1, 2]. Each class of service corresponds to a different delay and bandwidth requirement. The bandwidth and delay requirement for each class $c \in C$ (C denotes the set of classes) is denoted by (b_c, d_c)

where b_c is in bps and d_c is in seconds (see Table 1). The bandwidth requirement is to be maintained for a class of service over some time. We assume that the delays due to wireless access and processing of tasks at fog node can be estimated and accounted for while considering the delay constraint. An important QoS parameter in case of IoT devices is the age of information (AoI) [20]. It is defined as the time elapsed since the most recent received status from an IoT device. For a polling based scheme, AoI is equivalent to the duration of a polling cycle.

Table 1. Description of Notations

Notation	Description
N	Set of ONUs
C	Set of classes
B	Bandwidth of PON
b_c	Bandwidth requirement for class c in bps
d_c	Delay requirement for class c in secs
$A^c(t)$	Number of arrivals of class c in a slot t (Fig. 3)
T_s	Duration of a time slot
P_s	Packet size in bits
$\Lambda = \lfloor \frac{BT_s}{P_s} \rfloor$	Maximum number of packets that can be cleared in T_s
Λ^c	Maximum number of packets that has been agreed upon to be cleared for class c over horizon H
$Q_i^c(t)$	i^{th} Queue for tracking delay of class c and time slot t
x_i^c	Allocation corresponding to $Q_i^c(t)$ obtained by solving Eq. (10)
K^c	Number of virtual queues for tracking delay constraint d_c
H	Considered horizon for MPC
rtt_i^f	Round trip from fog to ONU_i

We consider plugging fog node at the remote node [19]. It is attached to the available free ports of the passive power splitter as shown in Fig. 1.The fog node communicates with the ONUs (equipped with separate transceivers for fog and OLT communication) at a wavelength different from that of the OLTs, thereby avoiding collisions among them. The bandwidth allocation for each ONU is decided by the fog node. The fog node, similar to the OLT, employs a standard polling based scheme. The details of how the fog node implements the allocation protocol will be discussed in Section 5. By means of the polling mechanism, the ONUs inform their demands for each class to the fog node. Once this information is available at the fog node, it decides on how these demands are to be satisfied such that the overall throughput is maximized without violating the QoS constraints.

Since we have strict delay constraints, it is evident that delay for packets need to be tracked. However, tracking delay for each packet requires large storage requirements which might not be available at facilities such as a fog node. By considering a slotted time system, the delay experienced by packets in a slot can be summarized. We present a detailed description of our delay-tracking mechanism which would aid in formulating the allocation problem.

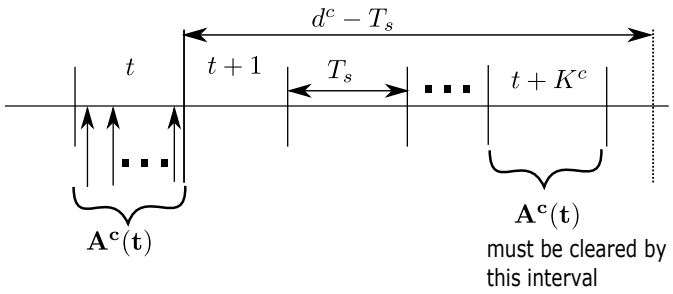


Fig. 2. Finding K^c as the maximum number of slots by which arrivals for a class must be cleared.

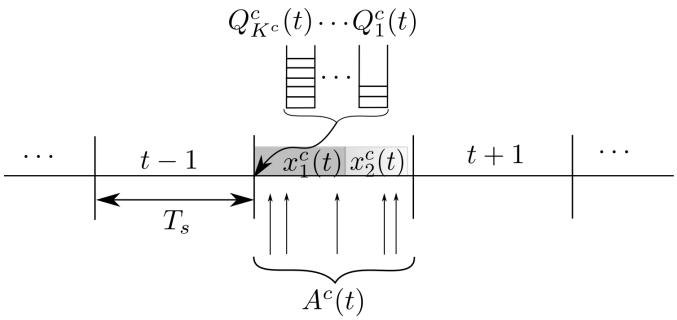


Fig. 3. Mechanism for tracking delay using virtual queues

A. Delay-tracking Model using Virtual Queues

To develop the model, we consider a generalized class c with requirements (d_c, b_c). The model can be easily extended for multiple classes by varying c and introducing consequent variables for the same. We consider that time is slotted with slot duration of T_s (see Fig. 2) and employ a virtual queuing scheme to track the delay requirements. We observe the system at a slot boundary. The slots are indexed in time as $t-1, t, t+1, \ldots$ and so on. The number of arrivals in each slot is denoted by $A^c(t)$ as shown in Fig. 2. Due to the delay constraint d_c, the packets arriving for class c at slot t, must be cleared within some slots. As shown in Fig. 3, the number of slots until which the fresh arrivals in a slot must be cleared in order to satisfy the delay requirements is (see Fig. 2):

$$K^c = \lfloor \frac{d_c - T_s}{T_s} \rfloor \tag{1}$$

Since the dynamic bandwidth allocation of PON operates over a polling based scheme, the arrivals are buffered. We denote the fresh arrivals for class c ($A^c(t)$ for slot t in Fig. 2) to be stored at queue $Q^c_{K^c}(t+1)$ at start of slot $t+1$ (similarly, $A^c(t-1)$ is stored in $Q^c_{K^c}(t)$). Depending on the designed allocation policy, some of the buffered packets are cleared in every slot. Let $x^c_{K^c}(t)$ denote the number of packets cleared from the buffer $Q^c_{K^c}(t)$ at slot t. Then, the remaining number i.e. $Q^c_{K^c}(t) - x^c_{K^c}(t)$ must be cleared within $K^c - 1$ slots, else those packets would violate the delay constraint. To capture this effect, we transfer these packets to a virtual queue denoted by $Q^c_{K^c-1}(t+1)$. The time index is also captured in the notation since this occurs at the immediate next slot (see Fig. 3). Hence, we have $Q^c_{K^c-1}(t+1) = Q^c_{K^c}(t) - x^c_{K^c}(t)$. Continuing in this manner, we can design virtual queues

based on the slots within which they must be cleared without violating the delay constraint.

Thus, we have K^c virtual queues $Q_1^c, \ldots, Q_{K^c}^c$ with Q_1 holding the packets which must be cleared in the immediate slot, whereas $Q_{K^c}^c$ contains the fresh arrivals which can be cleared by K^c slots. At the start of a slot say t, we denote the state of the i^{th} queue or the number of packets that can be cleared within i slots from the slot boundary by $Q_i^c(t)$.

$$Q_{i-1}^c(t+1) = Q_i^c(t) - x_i^c(t) \;\; \forall \;\; 1 < i \leq K^c \tag{2a}$$

$$Q_{K^c}^c(t+1) = A^c(t) \tag{2b}$$

where Eq. (2a) indicates that the packets that remain uncleared at the end of $t+1^{th}$ slot from the i^{th} queue have encountered a delay of T_s and hence should be cleared within $i-1$ slots. Eq. (2b) indicates that fresh arrivals in a slot can be cleared within K^c slots. Although in the model we consider the arrivals $A^c(t)$ to be known, the same may have to be estimated in reality. The effect of estimation errors are shown to be negligible by means of a sensitivity analysis in our results. We present the basics of MPC leading to the formulated problem.

B. Model Predictive Control (MPC)

MPC is used for controlling dynamic systems while satisfying a set of constraints. As discussed in Section 1, an MPC looks ahead in time and obtains efficient far-sighted policies.

An MPC problem is of the form:

$$\min J = \sum_{t=0}^{N-1} l(\mathbf{s}(t), \mathbf{a}(t)) \tag{3a}$$

$$\text{s.t.} \quad \mathbf{s}(t+1) = \mathbf{A}\,\mathbf{a}(t) + \mathbf{B}\,\mathbf{s}(t) \;\; \forall t \tag{3b}$$

$$\mathbf{s}(\mathbf{0}) = \mathbf{z},\ \mathbf{a}(t) \in \mathbb{A} \text{ and } \mathbf{s}(t) \in \mathbb{S} \tag{3c}$$

where $\mathbf{s}(t)$ are *state variables* which are controlled by *action variables* $\mathbf{a}(t)$. The function $l(\cdot)$ is a loss function which is to be minimized over the action variables $\mathbf{a}(t)$. The time index t are the discrete time intervals over which the system is observed. $\mathbf{s}(\mathbf{0}) = \mathbf{z}$ represents the initial state at the start of observations. For our setup, the initial states are obtained as the new arrivals are buffered up in their respective queues following Eq. (2a). N is the *horizon* or future time steps until which the system is being observed. Ideally the horizon should be infinite, however, for computational tractability, a finite horizon is usually implemented. The matrices $\mathbf{A}$ and $\mathbf{B}$ describe the state equation which essentially shows how the future state depends on the current states and actions. $\mathbb{A}$ denotes the set of feasible actions and $\mathbb{S}$ denotes the set of feasible states. In a typical MPC problem, all the associated sets are assumed to be convex and the loss function to be a convex function for ease of implementation. The optimization is solved at each time step to obtain the actions which generate new states by following the state equation.

3. PROBLEM FORMULATION

As discussed in Section 1, optimal far-sighted policy design for delay-constrained services require MDP based formulations involving stochastic state-action transition matrices which become computationally intractable for large state and action spaces. The aforementioned delay-tracking mechanism shows that the queue states and corresponding allocations can be easily cast into linear deterministic state equations 2b and 2a. This allows us to cast the resource allocation problem as a MPC discussed in B. We define the states, actions/policies, time step, horizon and all specifics of an MPC problem to fit our system model.

We use the following notation scheme: the subscript denotes the index of the queue under consideration while the superscript denotes the index of the corresponding class. In this section, we still proceed with single class case as the multi-class can be simply presented as an extension of the same. However, to emphasize the notion of classes, the superscript c has not been omitted.

A. Formulating Resource Allocation as MPC

State Variables: We consider the queue status as the state variable $Q_i^c(t)$. The state equations for the same have already been presented in Eq. (2).

Actions Variables: The allocations are the action variables $x_i^c(t)$. It is important to note that the allocations are in terms of packets or bits and hence are non-negative integers $x_i^c(t) \in \mathbb{Z}^+$. This restriction prohibits our problem to be a conventional convex optimization problem and hence is an integer linear program.

Time step: We represent each time step by T_s. Since in PON, the polling scheme requires at least a round trip time, we consider T_s to be at least equal to the maximum round trip time from the fog node.

$$T_s \geq \max\{rtt_i^f \;\; \forall i\} \tag{4}$$

Horizon: The horizon is the number of future time steps considered for performing the optimization. We denote the horizon by H. The case for $H = 0$ is the short-sighted version of the problem. Increasing the horizon implies looking farther ahead in the future at the cost of computational complexity.

Objective Function: In order to optimize throughput in PON, the sum throughput is maximized. The sum throughput is given by:

$$\max_{x_i^c(t)} \sum_{t=0}^{H} \sum_{i=1}^{K^c} x_i^c(t) \tag{5}$$

It is important to note that our delay-tracking mechanism allows us to guarantee delay constraints. It is evident that the first queue ($Q_1^c(0)$) for the first time slot (indexed by 0), must be cleared unless the number of packets in the queue exceeds the maximum number of packets that can be cleared within a slot. If Λ (refer Table 1 denotes the maximum number of packets that can be cleared within T_s, then

$$x_1^c(0) = \min\{Q_1^c(0), \Lambda\} \;\forall\; t \tag{6}$$

This ensures that the packets which are most critical and are about to violate delay constraints are cleared, thereby avoiding delay violation. Although we may fix the allocation $x_1^c(0)$ from the queue $Q_1^c(0)$, the same cannot be done for the future allocations. This is because the future allocations depend on future queue states that evolve using the state equations. Thus, the sum throughput is computed for all i and c except for $x_1^c(0)$. Effectively, the objective is:

$$\max_{x_i^c(t)} \sum_{t=0}^{H} \sum_{i=1}^{K^c} x_i^c(t) - x_1^c(0) \tag{7}$$

This objective function indicates that the MPC would try and maximize the throughput while minimizing the allocation for $x_1^c(0)$. This implies that other queues would be emptied as much

as possible so that the first queue is left with minimum number of packets.

Constraints: Since in MPC, we look ahead in time, the state evolving equations described in Eq. (2b) come into play. In addition, the maximum number of packets that can be cleared in a slot is bounded by Λ for any slot t. Thus, $\forall t$, we have:

$$\sum_{i=1}^{K^c} x_i^c(t) \leq \Lambda \ \ \forall\, t \tag{8}$$

For a given class c, the corresponding bandwidth constraint b_c should not be violated over time. Here, b_c denotes the peak bandwidth constraint as agreed upon with the service providers. Let us the number of packets that the service provider agrees to serve over a horizon H by Λ^c. This can be calculated as $\Lambda^c = b_c \times H \times T_s$. Although, we imposed a peak bandwidth constraint, one can also impose a minimum bandwidth constraint and an average one as well. As long as the corresponding Λ^c is an integer, the propositions discussed in this paper (refer Section 4) henceforth holds. Hence, the following constraint should be implemented.

$$\sum_{t=0}^{H}\sum_{i=1}^{K^c} x_i^c(t) \leq \Lambda^c \tag{9}$$

B. Far-Sighted Constrained Resource Allocation using MPC

In case of multiple classes, the objective Eq. (7) is summed over all $c \in C$. The state equations Eq. (2) and constraints Eq. (9) are enforced for each class c. However, for constraint Eq. (8), we would have a sum over all $c \in C$, since the total allocation over all classes in a slot cannot exceed the available bandwidth. The multi-class MPC problem can then be stated as:

$$\max_{x_i^c(t)} \ \sum_{c\in C}\sum_{t=0}^{H}\sum_{i=1}^{K^c} x_i^c(t) - x_1^c(0) \tag{10a}$$

$$\text{s.t. } \ Q_{i-1}^c(t+1) = Q_i^c(t) - x_i^c(t)\ \forall c, i\ 1 < i \leq K^c \tag{10b}$$

$$Q_{K^c}(t+1) = A^c(t),\ \forall t, c \tag{10c}$$

$$x_i^c(t) \leq Q_i^c(t),\ \forall t, c \tag{10d}$$

$$\sum_{c\in C}\sum_{i=1}^{K^c} x_i^c(t) \leq \Lambda, \forall\, t \tag{10e}$$

$$\sum_{t=0}^{H}\sum_{i=1}^{K^c} x_i^c(t) \leq \Lambda^c,\ \forall\, c \tag{10f}$$

$$x_1^c(0) = \min\{\Lambda, Q_1^c(0)\},\ \forall\, c \tag{10g}$$

$$x_i^c(t) \in \mathbb{N} \cup 0\ \forall t, c, i,\ 1 < i \leq K^c \tag{10h}$$

We now present an efficient solution to the MPC problem which would be beneficial for real-time systems.

4. METHODOLOGY AND THEORETICAL ANALYSIS

The solution to the MPC problem Eq. (10) requires solving an ILP due to integral restrictions of the variables $x_c^i(t)$. To this purpose, we show two important aspects of the problem. First, we identify that the short-sighted version of the problem with $H = 0$ can be cast as a max-flow problem [25]. This is useful for implementations in PON without any prediction. Next, we show that the ILP can be solved by its equivalent linear program that is obtained by relaxing the integral constraints on the variables. Thus, for solving Eq. (10), we have the following two propositions:

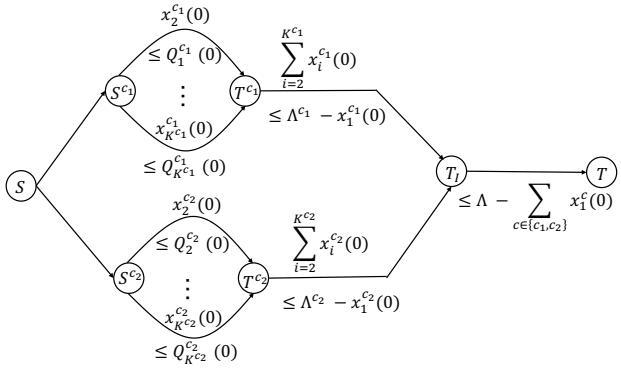


Fig. 4. Shows the max-flow equivalent of the optimization

Proposition 1. *The optimization in* Eq. (10) *can be cast as a max-flow problem for* $H = 0$ *and can be solved in* $\mathcal{O}\Big((2|C|+3)\sum_c K^c\Big)$. *This is the optimal strategy for bandwidth allocation without prediction.*

Proof. The short-sighted version of Eq. (10) is obtained for $H = 0$. Fig. 4 shows the proof for two classes c_1 and c_2. The idea can be easily extended for multi-class scenarios.

$$\max \sum_{c\in\{c_1,c_2\}}\sum_{i=2}^{K^c} x_i^j(0) \tag{11a}$$

$$\text{s.t. } \ x_i^j(0) \leq Q_i^c(0)\ \ \forall i, c \tag{11b}$$

$$\sum_{i=2}^{K^c} x_i^c(0) \leq \Lambda^c - x_1^c(0)\ \ \forall c \tag{11c}$$

$$\sum_{c\in\{c_1,c_2\}}\sum_{i=2}^{K^c} x_i^c(t) \leq \Lambda - \sum_{c\in\{c_1,c_2\}} x_1^c(0) \tag{11d}$$

$$x_1^c(0) = \min\{\Lambda, Q_1^c(0)\},\ \forall\, c \tag{11e}$$

$$x_i^c(0) \in \mathbb{N} \cup \{0\} \tag{11f}$$

Consider a graph G with a source node S and two terminal nodes T_I and T respectively as shown in Fig. 4. Clearly, the constraints in Eq. (11) can be seen as the capacities of the links and the maximum flow from the source S to terminal T solves the required optimization. It is known that the complexity of solving max-flow problem for a graph with V vertices and E edges is $\mathcal{O}(VE)$ [28]. The graph consists of 3 fixed vertices (S, T_I and T) for any number of classes. For each class, there are two vertices and $K^c - 1$ edges. Hence, the complexity of solving the short-sighted case is $\mathcal{O}\Big((2\times 2+3)\sum_c K^c\Big)$. In case of multi-class, we will have $|C|$ such graphs corresponding to each class. Thus the multi-class short-sighted allocation can be solved in $\mathcal{O}\Big((2|C|+3)\sum_c K^c\Big)$ ■

Proposition 2. *The MPC problem* Eq. (10) *can be solved by its equivalent linear program obtained by relaxing the integral constraints.*

Proof. To show this, we use an established result in the field of integer linear programming which provides a sufficient condition for an ILP to yield same solution as its linear program version [29]. A linear program of the form $\max\{\mathbf{cx}|\mathbf{Ax} \leq \mathbf{b}\}$ has integral solutions if the matrix $\mathbf{A}$ satisfies a special property called total unimodularity (TU), provided that $\mathbf{b}$ is integral. We

show that the constraint matrix formed in our problem is TU. Since all bounds on our variables $(Q_i^c(t), \Lambda, \Lambda^c)$ are integers, the corresponding LP yields integral solutions. The details of the proof and relevant results required for the same provided in Appendix of the paper. ■

Generic appeal of the solution: We briefly mention some salient features of our solution technique. It is evident that the MPC problem of Eq. (10) can be cast into an LP equivalent if the constraints are formed by adding the flow variables. Thus, any constraint of the form $\sum x_c^i(t) \lessgtr \alpha$ where α is a known constant may be realized. The sum can be carried over any combination of the variables c, i or t. Additionally one might be interested in considering allocations for each user separately instead of the aggregated demand considered here. The corresponding optimizer would then store each user request separately and introduce more granularity in defining the variables x_i^c as $x_{i,u}^c$ for an user u. Even then, the optimization can be performed by considering constraints for each user. Hence, the applied solution technique is quite powerful and allows efficient implementation for a large class of bandwidth and delay-constrained problems.

5. IMPLEMENTATION IN PON FRAMEWORK

We discuss the details of implementing the obtained allocation policies by solving Eq. (10).

A. Dynamic Bandwidth Allocation at fog node

In order to implement the resource allocation scheme obtained from MPC at a fog Node in PON, the FN should be facilitated with an access protocol. As mentioned here, the fog node can implement a polling based scheme similar to the OLT. In this regard, the specifics of such a protocol are to be decided. We consider that the fog node can poll the ONUs with help of control messages similar to GATE and REPORT as per Multi Point Control Protocol (MPCP) [12]. Each ONU sends a REPORT message containing information regarding its queue status for each class of service. The fog node collects this information from all ONUs and obtains the resource allocation scheme by solving Eq. (10). Evidently, $d_c > \max\{rtt_i^f, \forall i\}$, due to the fact that the dynamic bandwidth allocation is a polling based protocol and polling the ONUs require at least a round-trip. It is to be noted that the solution of Eq. (10) provides the aggregated (combined for all ONUs) number of packets/bits to be cleared for each class. Thus, the obtained solution is to be distributed among the ONUs. Since the requests for each ONU is known, a max-min fair allocation (also known as excess distribution) [30] of the aggregated amount is granted to each ONU for each class. The information regarding the granted amount is sent via a GATE message. The specific details of the cycle time, slot duration considered for optimization and horizon for optimization are discussed below:

B. Constraint on Cycle Time

A cycle is a duration of time in which all ONUs are served once as shown in Fig. 5. After the data transmission, each ONU generates and appends a REPORT message which contains information regarding the current buffer status for each class. It is evident that the cycle time is indeed dependent on the amount of data that the ONUs are allowed to clear on receiving the GATE message. Usually heuristic allocation strategies implement a limit on the amount of granted bandwidth for each ONU (otherwise a malicious ONU might occupy the entire bandwidth) which effectively limits the cycle time. Let us denote the maximum allowable grant for each ONU by T_{mg} in seconds. Then, the maximum amount of time within which all ONUs are served once, is given by $T_m = NT_{mg}$ (neglecting the GATE, REPORT and guard durations). It is easy to see that given a delay constraint say d_c, it is not possible for cycle time T_m to be arbitrarily large. To estimate the limit on the cycle time. we observe the worst possible delay that can be encountered by a packet. A packet that arrives just after the data has been transmitted has to wait for at least a cycle to get reported. Then, if required (depending on the delay constraint) this packet can be cleared in the subsequent cycle. This implies that the maximum allowable cycle time cannot be more than $\frac{d_c}{2}$. Otherwise, two consecutive cycles might be more than $\frac{d_c}{2}$ which would lead to a delay violation for the worst packet as described here (see Fig. 5). Hence,

$$T_m \leq \frac{d_c}{2} \tag{12}$$

C. Optimal Cycle time and choice of Time slot duration

The optimization of Eq. (10), is solved at end of each time slot. As evident from Eq. (4), the minimum duration for time slot can be $\max_i\{rtt_i^f\}$ while the maximum will be that of the maximum cycle time that is $\frac{d_c}{2}$ as we just explained. It is important to note that while defining the cycle we had ignored the guard durations and the control messages. To minimize wastage of bandwidth, it is evident that the number of these messages and guard durations must be minimal. Thus, a good choice for time slot duration should be the maximum cycle time i.e., $\frac{d_c}{2}$.

$$T_s = \min_c\{\frac{d_c}{2}\} \tag{13}$$

6. SIMULATION RESULTS AND DISCUSSION

We compare the performance of our (MPC-based) protocol with 1) existing standards and 2) the two protocols discussed in Section B: (i) OOB protocol [17] and (ii) priority protocol [26]. We compare the performance of these protocols using four metrics: (i) percentage of violation/drop, (ii) throughput, (iii) average delay, and (iv) jitter. The decentralized out of band bandwidth allocation which employs separate wavelengths is referred as "OOB". The one in [26] is referred as "Priority based" and our protocol is referred as "MPC". The simulation parameters have been tabulated in Table 2. Note that the priority based allocation scheme can also be conceived as a heuristic based on current information or a short-sighted allocation scheme. We have simulated a self-similar traffic with 16 traffic sources attached to each ONU in OMNet++ [31]. The self-similar traffic has been simulated with Hurst parameter of 0.8 [32] (highly bursty [9, 10]) to generate background best-effort traffic. Our choice of delay and bandwidth requirements are $(d_1, b_1) = (1\text{ ms}, 100\text{ Mbps})$ and $(d_2, b_2) = (4\text{ ms}, 100\text{ Mbps})$ respectively (our model is generic and can handle any delay and bandwidth requirement unless physically infeasible). Owing to the least delay constraint, the time slot was chosen to be 0.5 ms. This gives $K^1 = 1$ and $K^2 = 7$. We choose a finite horizon $H = 10$. To generate the traffic corresponding to delay-constrained applications, a Hurst parameter of 0.2 for both traffic types was considered owing to their less bursty nature compared to best-effort traffic [10]. To solve the optimization, we used MATLAB2020a and integrated OMNet++ to work seamlessly with MATLAB. All experiments were carried out at least 20 times by varying seeds of the random number

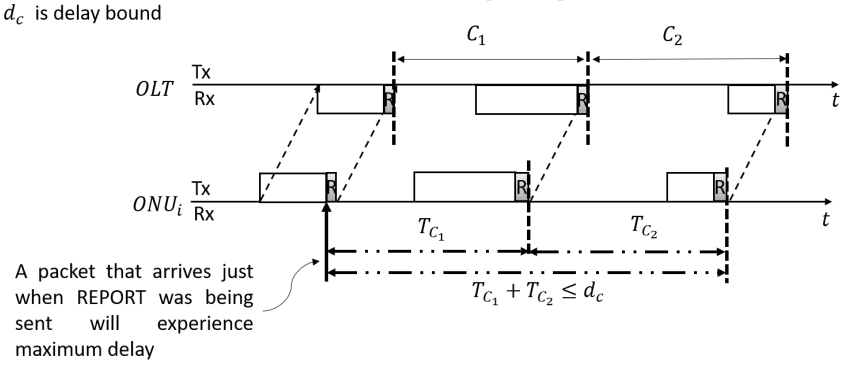


Fig. 5. Illustration of Constraint from dynamic bandwidth allocation protocol

generators. The resulting variations produced the confidence intervals indicated by the shaded areas in the figures. To observe the effect of changing best-effort loads, we considered cases with 25% and 50% of total traffic to be consisting of best-effort traffic. Thus, best-effort served as a background traffic. The rest of the load was generated equally from both the considered classes. Packets were considered to be uniformly distributed between 64 bytes and 1522 bytes.

Table 2. Simulation Parameters

Parameter	Value
Traffic Generator	Self Similar using Pareto [32]
Hurst Parameters	0.8 (Best-effort) and 0.2 (class1 and class2) [9, 10, 32]
Buffer Size for each ONU	10 Mb
Link rate at ONU side	100 Mbps
PON link rate	1 Gbps
Number of ONUs	16
Traffic Sources per ONU	16
Maximum Cycle Time	0.5 ms
Guard Duration	5 μs [32, 33]
Distance between OLT and fog node	15 km
Distance between ONUs and fog node	1-5 km [17]
$(d_1, b1)$	$(1ms, 100Mbps)$[3]
$(d_2, b2)$	$(4ms, 100Mbps)$[3]
Confidence Interval	95%
Packet Distribution	Uniform between 64 bytes and 1522 bytes

A. Comparison based on Percentage of Dropped Packets

Here, we present two types of comparisons. First, we present the comparison based on existing standards as discussed in Section 1. Then, we compare our proposal with some heuristics which were proposed for accommodating new applications.

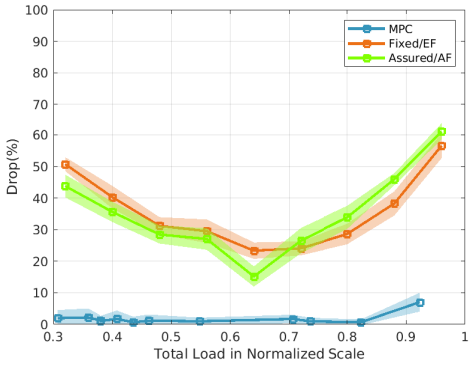
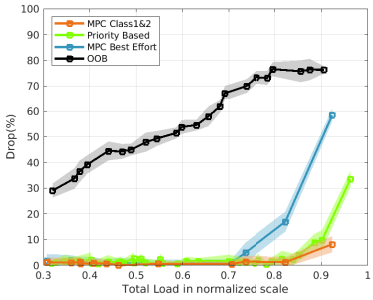


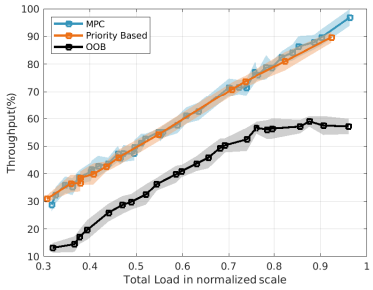
Fig. 6. Drop percentage comparison for delay constraint of 1ms with Fixed Allocation and Assured Allocation policies. The shaded areas show 95% confidence interval.

A.1. Comparison with existing standards

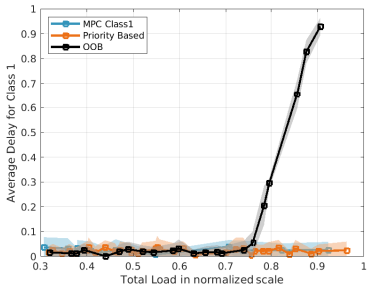
Fig. 6 shows the performance of existing standards which do not cater to satisfying strict delay constraints for bursty traffic conditions. The drop percentage refers to those packets which exceed the delay constraint of 1ms. Evidently, MPC based proposal specifically does not allow drops unless the load crosses a threshold where satisfying the delay constraint becomes impossible due to bandwidth restrictions. On the other hand, Fixed allocations which are primarily designed for constant-bit rate applications, allocate a fixed portion of bandwdith (TDM) for each ONU. Due to the bursty nature of arrivals, this scheme is highly inefficient (specifically at low loads, as it unnecessarily reserves bandwidth which delays subsequently arriving bursts). Assured scheme shows better performance as compared to Fixed allocations only at lower loads as it gets opportunity to dynamically schedule the bursty packet arrivals. However, Assured strategy only guarantees a certain bandwidth and does not aim at maintaining delay constraints. As the load increases, the two strategies show similar performance since, the fixed and assured allocations reach their respective limits.



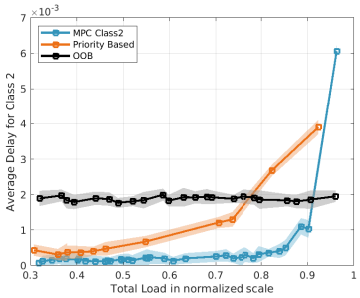
(a) Drop percentage for all classes



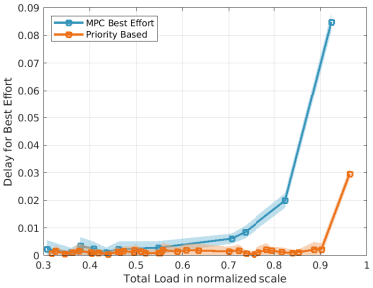
(b) Throughput%



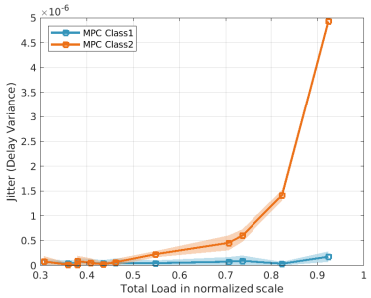
(c) Average Delay for class 1



(d) Average Delay for class 2

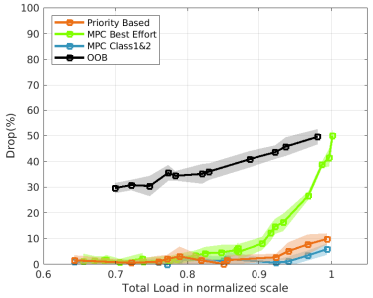


(e) Average Best Effort Delay

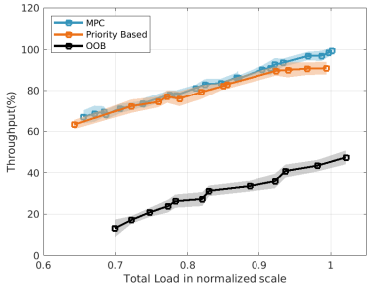


(f) Jitter Performance for classes 1 & 2

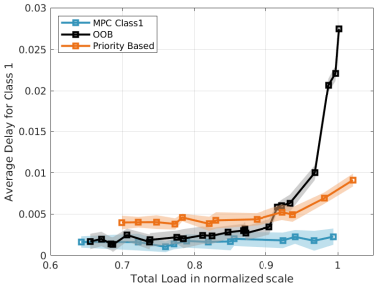
Fig. 7. Figure shows performance in terms of Drop, Throughput, Average Delay and Jitter with Best-effort Load of 25% of the maximum load. Drop and Delay of Classes are maintained at cost of Best-effort delay



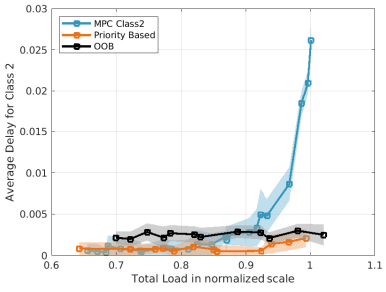
(a) Drop percentage for all classes



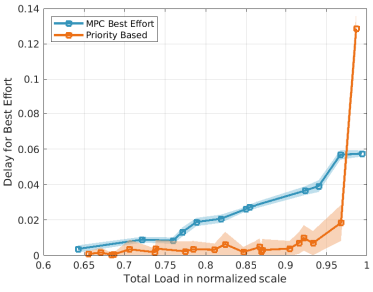
(b) Throughput%



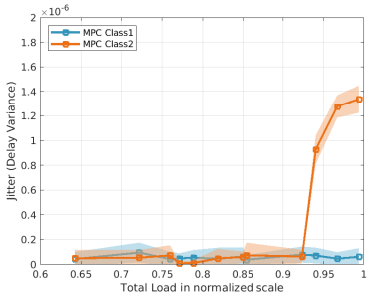
(c) Average Delay for class 1



(d) Average Delay for class 2



(e) Average Best Effort Delay



(f) Jitter Performance for classes 1 & 2

Fig. 8. Figure shows performance in terms of Drop, Throughput, Average Delay and Jitter with Best-effort Load of 50% of the maximum load. Drop and Delay of Classes are maintained at cost of Best-effort delay. With the increase in best effort load, the average delay performance of class 2 suffers (our allocation is not delay optimal).

A.2. Comparison with existing heuristics

To evaluate the performance of our proposal, we find the number of dropped packets with increase in aggregated loads of class 1 and class 2 services. We show the effect of increasing best-effort traffic as well. It is conceivable that the MPC based framework adheres to satisfying the strict QoS requirements first while best-effort traffic is not tackled by the concerned optimization. Best-effort traffic allocations are performed only if there is sufficient bandwidth after the optimal allocations obtained from Eq. (10) are scheduled. Both OOB and priority based methods do not aim at guaranteeing the delay and bandwidth constraints and hence fail drastically when stringent real-time delay requirements are to be satisfied (as shown in Fig. 7a). On the other hand, the MPC based allocation strategy operates well within the feasible delay and bandwidth regions, unless the requests exceed the available bandwidth. Thus, as shown in Fig. 7a, MPC based allocations show almost zero drop. The best-effort drop (full queue size condition) is also plotted in the figure. We observe that the drop percentage for best-effort traffic is quite low because of the low percentage of best-effort traffic generated. With increase in class 1 and class 2 traffic, the percentage of drop for best-effort traffic increases.

We can also observe the effect on increasing percentage of best-effort traffic. As shown in Fig. 8a, the best-effort drop increases significantly and this shows the trade-off for using MPC based allocations. MPC allocations try and ensure that the delay of class 1 and class 2 types of service do not get violated at the expense of dropping best-effort traffic.

B. Comparison based on Throughput

Througput is calculated as the percentage of time the link remains busy for clearing packets with respect to the total simulation time. Since MPC maximizes throughput as its objective, we observe its throughput performance when compared with the existing proposals. As expected, in Fig. 7b and Fig. 8b, the MPC having the optimal throughput shows the best performance. As pointed out earlier, a major drawback of OOB is that allocation for fog requests depend on the corresponding OLT allocation window. Hence, although there are two wavelenghts in use, utilization is drastically affected, as evident from Figs. 7b and 8b. On the contrary, the priority based scheme performs fairly well since it utilizes its available bandwidth with either the OLT requests or the prioritized fog requests. When the best-effort load increases, we start to observe that full throughput is achievable at lower loads for class 1 and class 2 services. This shows that even when the priority classes do not have traffic, the protocol does not suffer from light-load penalty, which is a common problem in priority based allocation [32].

C. Comparison based on Average Delay

As discussed, we compare the average delay performance for the mentioned protocols. The average delay for both classes is drastically less compared to those for OOB and Priority based protocols. It is important to note that the objective of our MPC does not minimize average delay and was rather throughput based. However, the average delay performance for the most stringent class can be seen to outperform other counterparts as shown in Fig. 7c and 8c. The same cannot be said for the other classes as can be seen for some loads for the second class of service as shown in Fig. 7d and 7d. Also as evident from Fig. 7e and 8e, the best effort service suffers in terms of average delay when compared to that of a Priority based slicing protocol. In OOB protocol, the allocations depend on amount of best-effort traffic. With more best-effort traffic, the high priority Fog traffic (class1 in this case) gets more opportunity to be cleared and hence a better performance is achieved compared to the case when best-effort load is less. It is important to note that the average delay performance is mostly by design and can be manipulated by adequate prioritization and handling of allocation decisions. When allocating class 1 and class 2 traffic from same virtual queue, we first allocate those of class 1 first and then allocate class 2. By changing this allocation strategy, we may obtain a desired average delay performance.

We do not show the best effort performance for OOB since it is similar to that of class 2 because OOB does not handle classes separately but have differentiation between services for fog and OLT. We treated the traffic for class 1 as that intended for fog and all other classes get mapped to that for OLT.

D. Jitter performance

Jitter can have many definitions. We use variance of delay as the measure for jitter. The jitter performance shows a peculiar behavior. The jitter performance for other protocols were drastically poor and could not be presented on the same scale (almost 10-100 times more). As shown in Fig. 7f, at low best-effort loads, the MPC based allocations show drastic increase in jitter for class 2 while class 1 experiences almost an uniform jitter performance. This might be due to the fact that at low loads, the buffer does not remain occupied at all times and hence delays mostly depend on the random arrival times. On the other hand, at higher best-effort loads, the buffers remain filled with more probability and hence jitter performance improves as is evident from Fig. 8f.

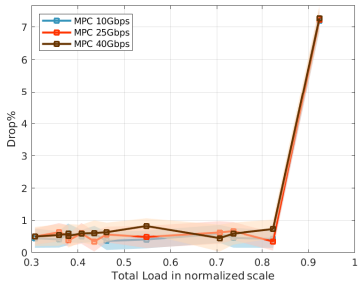
E. Comparison with Different Line Rates

We compare how our proposal evaluates with varying line rates of PON. To keep up with the increasing bandwidth demands, standardization groups like ITU-T and IEEE had launched PON with higher line rates. ITU-T had standardized 10 Gbps G-PON in G.987 while the same was done by IEEE for EPON in IEEE 802.3av. Moving forward, options of 40 Gbps in G.989, 25 and 50 Gbps options were explored in G.9804. In all these standards, the basic frame structures, DBA schemes for reporting and granting remain unaltered and hence are compatible with our proposal. From Fig. 9a, 9b, 9c, we observe that there is no such significant effect on increasing the line speeds in the drop, delay and jitter performance and hence the proposed approach scales for next-generation PON systems as well.

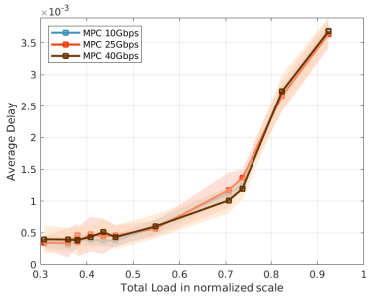
F. Prediction Performance

Although the MPC problem could be solved with $H = 0$, for larger horizon, traffic prediction is desired. We used an LSTM (long short term memory) [34, 35] for predicting multiple time steps ahead for performing our MPC based resource allocation. We used an LSTM with 200 units. Since prediction over larger time steps can lead to more errors, we chose $H = 4$ and $H = 10$, for comparison purposes.

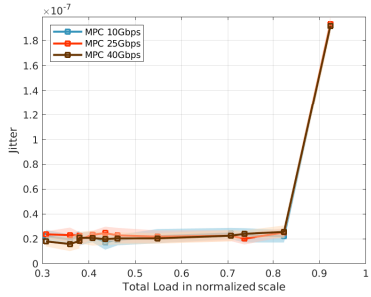
Initially, we trained an LSTM model with recorded traffic data from OMNET++ for different classes and used them for predicting the traffic arrivals for next 4 time steps and 10 time steps respectively. The root mean square error of predicted vs actual traffic was 1.9365 and 6.5803 respectively. Then, during the simulation run, we updated and retrained the network using newly available data of previous time steps. Fig. 10 shows the efficiency of learning. The drop or delay violation figures are



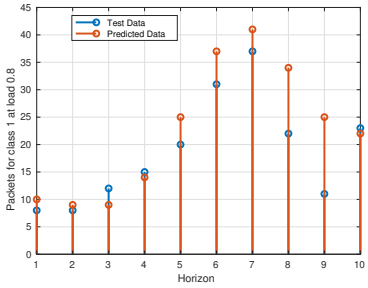
(a) Comparison of drop or violation % for 10,25 and 40 Gbps



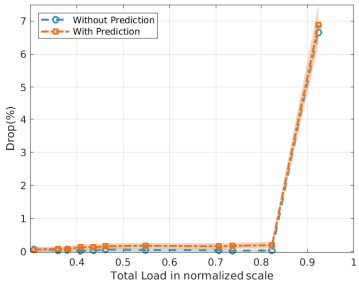
(b) Comparison of average delay for 10,25 and 40 Gbps



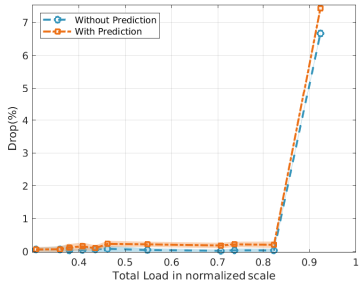
(c) Comparison of jitter for 10,25 and 40 Gbps



(a) Learning with LSTM at load of 0.8 for class 1



(b) Drop percentage for class 1 with and without learning with $H = 4$



(c) Drop percentage for class 1 with and without learning with $H = 10$

Fig. 10. Efficacy of Proposed approach with respect to learning with LSTM

recorded for both settings and are presented in Fig. 10b and Fig. 10b. The drop performance degrades by merely 3.03% for $H = 4$ while for $H = 10$, the drop percentage increased to 11.13%. Thus, as horizon increases the prediction errors creep into the resource allocation decisions. Hence, one should design accordingly keeping the trade-off between learning and performance degradation in hindsight.

G. Complexity Analysis

We ran the corresponding LP in MATLAB with number of variables equal to product of number of classes and number of virtual queues. For the two classes, we had the number of virtual queues to be 1 and 7 respectively which makes the total to be 8. Thus at each time step, we solve a 16 variable linear program. Due to the structure of our formulated optimization, simplex was able to solve the problem in all cycles. If interior-point method is used instead, the worst-case complexity is $\mathcal{O}\left(n^{3.5}L\right)$, where L is the number of bits required to encode the variables (depends on the compiler used). As mentioned in [36], good implementations of simplex and interior-point methods exhibit similar performance for routine applications.

7. CONCLUSION

In this paper, we addressed the problem of resource allocation for emerging applications with bursty traffic patterns, requiring delay tolerance in order of milliseconds and high bandwidth requirements. We introduced a mechanism that aids in tracking delay and subsequently formed an MPC based model for obtaining optimal allocation policies. It is important that the optimal allocation policy is determined with low computational complexity. To this purpose, we showed that the MPC problem could be implemented by solving a linear program. As an implementation exercise, we chose PON as the networking framework where an edge/fog node is installed to provide real-time Internet services. Our MPC based allocations outperform the existing ones with respect to average delay, throughput and delay violation metrics. Further, the obtained allocations scale well when the number of subscribers are increased. Interesting extensions would be to look for further reductions in computational complexity. We also investigated use of a learning technique for estimating the traffic arrivals. We observed that there is a trade-off between the horizon and performance degradation. A future extension could be to characterize this effect to aid automated system design.

PROOF OF TOTAL UNIMODULARITY (PROPOSITION 2)

Definition 1. *A matrix is totally unimodular (TU) if the determinant of any of its square sub-matrices is either* $0, 1,$ *or* -1.

Lemma 1*: If* $\mathbf{A}$ *is TU and* $\mathbf{b}$ *is integral,* $\max\{\mathbf{cx}|\mathbf{Ax} \leq \mathbf{b}\}$ *has integral optima.*[37, 38]

Evidently, to use this result, we need some characterization of such matrices which would allow us to identify a TU matrix. The following results will be useful in our efforts to prove our proposition.

Lemma 2*: If* $\mathbf{A}$ *is TU, the matrix obtained by removing or adding unit columns (column of an identity matrix) or unit rows (row of an identity matrix) is also TU.* [37]

Lemma 3 (Ghouila-Houri)*: A matrix* $\mathbf{A}$ *is TU if and only if for each subset R of rows of A, there exists two disjoint sets* R_1 *and* R_2 *such that* $\sum_{i \in R_1} a_{ij} - \sum_{i \in R_2} a_{ij} \in \{0, -1, 1\}$. *In other words for every subset of rows of* $\mathbf{A}$, *we can find two disjoint sets such that the difference of the sum of rows would result in a vector with entries* $0, 1$ *or* -1. [37, 38]

We construct the matrix **A** from the constraints of our problem and use the aforementioned results to prove the proposition. We consider the single class case. The arguments can be extended to multi-class as well. The constraint set for class c is given by the equations:

$$Q^c_{i-1}(t+1) = Q^c_i(t) - x^c_i(t)\ \forall c, i\ 1 < i \leq K^c \quad \textbf{(14a)}$$
$$Q_{K^c}(t+1) = A^c(t),\ \forall t, c \quad \textbf{(14b)}$$
$$x^c_i(t) \leq Q^c_i(t),\ \forall t, c \quad \textbf{(14c)}$$
$$\sum_{i=1}^{K^c} x^c_i(t) \leq \Lambda, \forall\, t \quad \textbf{(14d)}$$
$$\sum_{t=0}^{H}\sum_{i=1}^{K^c} x^c_i(t) \leq \Lambda^c,\ \forall\, c \quad \textbf{(14e)}$$

It is important to note that the bound $Q^c_i(t)$ is not a constant but involves optimization variables from previous time slots. Hence we should use Eq. (14a),Eq. (14b) recursively and obtain variable free bounds as follows:

$$x^c_i(H) \leq Q^c_i(H) = Q^c_{i+1}(H-1) - x^c_{i+1}(H-1)$$
$$\implies x^c_i(H) + x^c_{i+1}(H-1) \leq Q^c_{i+1}(H-1) \quad \textbf{(15)}$$

Continuing in this way, we have two cases corresponding to $H < K^c$ and $K^c \leq H$:

Case1: $H < K^c$

$$\sum_{j=i}^{H+i} x^c_j(H-j+i) \leq Q^c_{H+i}(0),\ H+i < K^c$$
$$\sum_{j=i}^{K^c} x^c_j(H-j+i) \leq Q^c_{K^c}(H-K^c+i),\ H+i \geq K^c$$

Case2: $K^c \leq H$

$$\sum_{j=i}^{K^c} x^c_j(H-j+i) \leq Q^c_{K^c}(H-K^c+i)$$
$$= A^c(H-K^c+i-1)$$

It is easy to see that in either case, we either end up with a bound which is dependent on initial queue states (corresponding to slot 0) or the arrivals in each slot which govern the subsequent queue states. Now we are ready to construct the matrix. For purpose of illustration, we will use the case of $K^c = 3$ and $H = 3$. However, our arguments will be general and can be extended for any K^c and H. For sake of simplicity, we omit the superscript c.

$$\mathbf{A} = \begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0\\
0&1&0&1&0&0&0&0&0&0&0&0\\
0&0&1&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&1&0&1&0&1&0&0&0&0&0\\
0&0&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&1&0&1&0&1&0&0\\
0&0&0&0&0&0&0&0&1&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&1\\
1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&1&1&1
\end{pmatrix}$$

The first 12 rows of the matrix correspond to the constraints Eq. (14c). The 13^{th} row corresponds to constraint Eq. (14e) while the last 4 rows realize constraints Eq. (14e) for each slot/horizon. The columns are arranged in order of queues followed by slot index i.e., $\{x_1(0), x_2(0), x_3(0), x_1(1), \ldots, x_3(3)\}$. The variables corresponding to initial time slot 0 and the $K^{c\,th}$ queues do not require recursive expansion and hence correspondingly have unit rows. For proving **A** is TU, we need not consider these rows, since adding them do not affect the TU property of the matrix (due to Lemma 2). Hence, we are left with the following matrix.

$$\mathbf{A}' = \begin{pmatrix}
0&1&0&1&0&0&0&0&0&0&0&0\\
0&0&1&0&1&0&0&0&0&0&0&0\\
0&0&1&0&1&0&1&0&0&0&0&0\\
0&0&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&1&0&1&0&1&0&0\\
0&0&0&0&0&0&0&0&1&0&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&1&1&1
\end{pmatrix}$$

We use Result 3 to prove $\mathbf{A}'$ is TU. To do so, we first inspect the matrix and present some observations. The rows corresponding to Eq. (14c) (first 6 rows) have the property that for any two rows, either the relative position of 1's and 0's do not match (commonly termed as orthogonal vectors) or if the positions match, the two rows differ by 1's occurring at regular intervals of 0's. For example: if we have the first two rows, they have 1's

and 0's in alternate positions. If we have the second and third rows, the positions of 1's and 0's match but they differ by a 1 appearing after a 0 following the pattern 101. When $K^c > 3$, the pattern introduces more number zeros between 1's. Further, the row with all 1's can be realized by adding the last four rows since, for each of the last four rows have their 1's placed at different positions (they are also mutually orthogonal).

Suppose we choose a subset of rows from the set of rows. The rows can either correspond to Eq. (14c) (first 6 rows), the all 1's (corresponding Eq. (14e)) or that corresponding to Eq. (14d) (last 4 rows). While considering the rows corresponding to Eq. (14c), the idea is to construct R_1 and R_2 so that the sum of rows in R_1 have maximum number of 1's in the resulting vector. Clearly, this can be done by selecting all orthogonal rows and the rows with greater number of 1's when the relative positions match. The rows having same relative positions with lesser number of 1's compared to others are placed in R_2. Due to this construction, it is evident that the difference of sum of rows in R_1 and R_2 will only result in entries with either 0 or 1. Next, if we have the row with all 1's in our choice of subset, we place it in R_2 so that the difference will always result in entries with 0, 1 or -1. If our choice of subset have rows corresponding to Eq. (14d) but do not contain the row with all 1's, we place it in R_2 else, we place the row with all 1's in R_2 and the rows corresponding to Eq. (14d) in R_1. Since these rows are mutually orthogonal and their sum can only result in consecutive 1's, the mentioned assignment will always leave the difference of the rows between R_1 and R_2 with entries in $\{0, 1, -1\}$.

For example: From the set of rows of $\mathbf{A}'$ let $R = \{2,3,4,5,6,7,8\}$ be a subset of rows. By our mentioned method, we first place rows $\{3,5,6\}$ in R_1 and $\{2,4\}$ in R_2. Note that the difference of sum of rows will only lead to entries 0 or 1. Now we have the row of all 1's and row 8 which has 1's as the first three entries. We place the row 7 in R_2 and row 8 in R_1. Thus, $R_1 = \{3,5,6,8\}$, $R_2 = \{2,4,7\}$. The difference of the sum is $[0,0,0,-1,-1,-1,0,-1,0,0,0,-1]$.

For multi-class scenario, more number of variables are introduced in the matrix for each time slot. As an example we can consider the scenario for two classes, wherein the number of variables for each slot becomes $K^1 + K^2$. In this case, the constraint Eq. (14e) has to be implemented for each class and hence instead of a single row of all 1's, we have a two rows one of which follows the pattern K^1 1's followed by K^2 0's while the other has K^1 0's followed by K^2 1's. We can apply all our previous arguments except for the case when one of these two rows is in the chosen set of rows along with one or more rows corresponding to Eq. (14d). In this case, there is a possibility that the sum of difference between the rows gives 2. However, this can occur only for one particular class since we have only one row corresponding to Eq. (14e). To circumvent this issue, we can first construct the two row sets for each class separately corresponding to constraints Eq. (14c). Earlier it was ensured that the sum of difference of these two sets would have entries 0 or 1. The class which might encounter the aforementioned problem can be readjusted by simply swapping its corresponding row sets so that the sum of difference of those would have entries 0 or -1 instead of 0 or 1. Now, the previously mentioned assignment steps for rows corresponding to Eq. (14d) and Eq. (14e) can be followed to yield the sum of difference in $\{0, 1, -1\}$.

FUNDING

This research was supported in part by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (project DP190102828).

DISCLOSURES

The authors declare no conflicts of interest.

REFERENCES

1. M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6g networks: Use cases and technologies," IEEE Commun. Mag. **58**, 55–61 (2020).
2. M. Z. Chowdhury, M. Shahjalal, S. Ahmed, and Y. M. Jang, "6g wireless communication systems: Applications, requirements, technologies, challenges, and research directions," arXiv preprint arXiv:1909.11315 (2019).
3. F. Fitzek, F. Granelli, and P. Seeling, *Computing in Communication Networks: From Theory to Practice* (Academic Press, 2020).
4. B. V. Philip, T. Alpcan, J. Jin, and M. Palaniswami, "Distributed real-time iot for autonomous vehicles," IEEE Transactions on Ind. Informatics **15**, 1131–1140 (2018).
5. P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel *et al.*, "Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture," IEEE Commun. Mag. **55**, 70–78 (2017).
6. L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," IEEE Transactions on industrial informatics **10**, 2233–2243 (2014).
7. E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," IEEE Transactions on Ind. Informatics **14**, 4724–4734 (2018).
8. K. Apicharttrisorn, B. Balasubramanian, J. Chen, R. Sivaraj, Y.-Z. Tsai, R. Jana, S. Krishnamurthy, T. Tran, and Y. Zhou, "Characterization of multi-user augmented reality over cellular networks," in *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON),* (IEEE, 2020), pp. 1–9.
9. S. Mondal, L. Ruan, M. Maier, D. Larrabeiti, G. Das, and E. Wong, "Enabling remote human-to-machine applications with ai-enhanced servers over access networks," IEEE Open J. Commun. Soc. **1**, 889–899 (2020).
10. A. M. Alba and W. Kellerer, "Large-and small-scale modeling of user traffic in 5g networks," in *2019 15th International Conference on Network and Service Management (CNSM),* (IEEE, 2019), pp. 1–5.
11. R. ITU-T, "G. 984.3: Gigabit-capable passive optical networks (gpon): Transmission convergence layer specification," (2008).
12. IEEE, "IEEE standard for ethernet," IEEE Std 802.3-2015 (Revision IEEE Std 802.3-2012) pp. 1–4017 (2016).
13. B. P. Rimal, D. P. Van, and M. Maier, "Cloudlet enhanced fiber-wireless access networks for mobile-edge computing," IEEE Transactions on Wirel. Commun. **16**, 3601–3618 (2017).
14. B. P. Rimal, M. Maier, and M. Satyanarayanan, "Experimental testbed for edge computing in fiber-wireless broadband access networks," IEEE Commun. Mag. **56**, 160–167 (2018).
15. A. Helmy, N. Krishna, and A. Nayak, "On the feasibility of service composition in a long-reach pon backhaul," in *2018 International Conference on Optical Network Design and Modeling (ONDM),* (IEEE, 2018), pp. 41–46.
16. A. Helmy and A. Nayak, "Toward parallel edge computing in long-reach pons," IEEE/OSA J. Opt. Commun. Netw. **10**, 736–748 (2018).
17. A. H. Helmy and A. Nayak, "Integrating fog with long-reach pons from a dynamic bandwidth allocation perspective," J. Light. Technol. **36**, 5276–5284 (2018).
18. J. Neaime and A. R. Dhaini, "Resource management in cloud and tactile-capable next-generation optical access networks," J. Opt. Commun. Netw. **10**, 902–914 (2018).
19. D. Roy, S. Dutta, A. Datta, and G. Das, "A cost effective architecture and throughput efficient dynamic bandwidth allocation protocol for fog

computing over epon," IEEE Transactions on Green Commun. Netw. (2020).
20. B. Zhou and W. Saad, "Minimizing age of information in the internet of things with non-uniform status packet sizes," in *ICC 2019-2019 IEEE International Conference on Communications (ICC),* (IEEE, 2019), pp. 1–6.
21. C. M. Assi, Y. Ye, S. Dixit, and M. A. Ali, "Dynamic bandwidth allocation for quality-of-service over ethernet pons," IEEE J. on selected Areas Commun. **21**, 1467–1477 (2003).
22. X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," IEEE J. on Sel. areas Commun. **24**, 1452–1463 (2006).
23. L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," IEEE/ACM Transactions on Netw. **19**, 841–854 (2010).
24. R. Singh and P. Kumar, "Throughput optimal decentralized scheduling of multihop networks with end-to-end deadline constraints: Unreliable links," IEEE Transactions on Autom. Control. **64**, 127–142 (2018).
25. C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity* (Courier Corporation, 1998).
26. J. Ou, J. Li, L. Yi, and J. Chen, "Resource allocation in passive optical network based mobile backhaul for user mobility and fog computing," in *Asia Communications and Photonics Conference,* (Optical Society of America, 2017), pp. M3B–1.
27. A. Wei, T. Ye, G. He, and J. He, "A hybrid dba algorithm for epon-based mobile front-haul networks," in *ICC 2020-2020 IEEE International Conference on Communications (ICC),* (IEEE, 2020), pp. 1–6.
28. J. B. Orlin, "Max flows in O(NM) time, or better," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing,* (2013), pp. 765–774.
29. D. P. Bertsekas, "Nonlinear programming," J. Oper. Res. Soc. **48**, 334–334 (1997).
30. J.-Y. Le Boudec, "Rate adaptation, congestion control and fairness: A tutorial," Web page, Novemb. **4** (2005).
31. A. Varga, "The omnet++ discrete event simulation system," in *In ESM'01,* (2001).
32. G. Kramer, B. Mukherjee, and G. Pesavento, "Interleaved polling with adaptive cycle time (IPACT): A dynamic bandwidth distribution scheme in an optical access network," Photonic Netw. Commun. **4**, 89–107 (2002).
33. F.-T. An, Y.-L. Hsueh, K. S. Kim, I. M. White, and L. G. Kazovsky, "A new dynamic bandwidth allocation protocol with quality of service in ethernet-based passive optical networks," arXiv preprint arXiv:1404.2413 (2014).
34. "Multi step ahead forecasting with lstm - ," https://in.mathworks.com/matlabcentral/answers/504175-multi-step-ahead-forecasting-with-lstm. (Accessed on 01/29/2022).
35. "Time series forecasting using deep learning - matlab & simulink - mathworks india," https://in.mathworks.com/help/deeplearning/ug/time-series-forecasting-using-deep-learning.html. (Accessed on 01/29/2022).
36. "Linear programming - wikipedia," https://en.wikipedia.org/wiki/Linear_programming. (Accessed on 01/30/2022).
37. A. Schrijver, *Theory of linear and integer programming* (John Wiley & Sons, 1998).
38. Wikipedia, "Unimodular matrix — Wikipedia, the free encyclopedia," http://en.wikipedia.org/w/index.php?title=Unimodular%20matrix&oldid=977444555 (2021). [Online; accessed 02-February-2021].