Efficient queue control policies for latency-critical traffic in mobile networks

Mohammed Abdullah, Salah Eddine Elayoubi, Tijani Chahed

Efficient queue control policies for latency-critical traffic in mobile networks

Mohammed ABDULLAH[12], Salah Eddine ELAYOUBI[1], Tijani CHAHED[2]
[1]Université Paris Saclay; CentraleSupelec; L2S, CNRS, Gif-Sur-Yvette, France,
[2]Institut Polytechnique de Paris; Telecom SudParis; Palaiseau, France

***Abstract*—We propose a novel resource allocation framework for latency-critical traffic, namely Ultra Reliable Low Latency Communications (URLLC), in mobile networks which meets stringent latency and reliability requirements while minimizing the allocated resources. The Quality of Service (QoS) requirement is formulated in terms of the probability that the latency exceeds a maximal allowed budget. We develop a discrete-time queuing model for the system, in the case where the URLLC reservation is fully-flexible, and when the reservation is made on a slot basis while URLLC packets arrive in mini-slots. We then exploit this model to propose a control scheme that dynamically updates the amount of resources to be allocated per time slot so as to meet the QoS requirement. We formulate an optimization framework that derives the policy which achieves the QoS target while minimizing resource consumption and propose offline algorithms that converge to the quasi optimal reservation policy. In the case when traffic is unknown, we propose online algorithms based on stochastic bandits to achieve this aim. Numerical experiments validate our model and confirm the efficiency of our algorithms in terms of meeting the delay violation target at minimal cost.**

***Index Terms*—URLLC, latency, discrete-time queue, reliability, 5G Networks.**

I. Introduction

Ultra-Reliable Low Latency Communications (URLLC) service was introduced in 5G [1] to tackle critical services such as autonomous driving, industry 4.0, smart grid, etc. A typical performance target is $1ms$ and $99,999\%$ delay and reliability constraints, respectively [2]. Several features were introduced in the 3GPP standardization to help reach the URLLC low latency constraint. For instance, short Transmission Time Interval (TTI) allows a mini-slot as small as 0.144 ms and resource preemption allows borrowing resources on-the-fly from other services such as enhanced Mobile Broadband (eMBB) [3]. These techniques enable the radio latency (i.e., the time between the packet generation and its decoding by the base station) to be below 0.5 ms. However, the underlying assumption is that resources are always available and latency is only due to the packet alignment, scheduling grant reception, over-the-air transmission and packet decoding.

When resources are scarce or traffic load is high, queuing delay, i.e., the delay before a resource is available for the packet to be scheduled, occurs. When URLLC service is in competition with eMBB service, the problem of queuing is solved by the feature of preemptive scheduling, wherein URLLC packets are served immediately upon arrival by preempting some eMBB resources [3]. However, when URLLC packets compete with other URLLC packets, preemption is not possible and over-reservation of resources may be needed. For instance, when URLLC traffic is periodic, semi-persistent scheduling (SPS) is proposed and resources are pre-reserved for each of the users [4]. However, for sporadic traffic scenarios, SPS is highly inefficient and mastering the queuing delay is still an open problem. Solving this problem requires efficient performance models for resource dimensioning on one hand, and control schemes that dynamically adapt the reserved resources to the system status, on the other hand. These are precisely the objectives of this paper.

Related works

Many works addressing the issue of URLLC resource allocation control focused on the contention-based aspect of such resource allocation (see for instance [5], [6] and the references therein). These works are beyond the scope of our present work as they focus on grant-free scheduling. We instead focus here on works related to grant-based scheduling for URLLC. In [7] for instance, a constrained Markov Decision Process (CMDP) approach has been proposed, slices of different types, including URLLC, are allocated resources dynamically as a function of their demand, QoS requirement and level of priority, the solution is of the form of a randomized policy. The QoS requirement however was modeled as the ratio of allocated resources to demand, and does not explicitly quantify the reliability and delay performances of URLLC in terms for instance of delay violation probability, which we do in our present work. In [8], the authors addressed the issue of allocating efficiently orthogonal channels to URLLC devices without knowing their Channel State Information (CSI). A constrained multi-agent MDP approach has been adopted, wherein the state of a device represents its packet loss rate and the action is the allocation of channels across the devices. A low complexity approximate algorithm has also been devised along with optimality gap analysis. The aim of their work however is to maximize the overall devices throughput while keeping their packet loss rates as low as possible, whereas our aim is to minimize needed resources while achieving required QoS. In [9], the authors aimed to minimize the quantity of reserved resources while satisfying the URLLC QoS by making use of unsupervised learning for the training of a Deep Neural Network (DNN) that expresses the relationship between the solution to their constrained optimization problem and the system parameters. In our work however, we derive a model relating the quantity of resources to be reserved to the

system parameters (user's arrival rates and radio conditions) and make use of it in the optimization and control.

Several works adopted queuing approach for the modeling of URLLC dynamics. [10] proposed an M/M/1 model based on the assumption of Poisson arrivals of packets and an exponential model for the variation of packet sizes due to different radio conditions. [11] relaxed the exponential assumption for the service rate and adopted an M/G/1 model with vacations (to account for the presence of other users), but with two restrictive assumptions. First, the "General" service model is due to different packet sizes and not different radio conditions, and second, packets are supposed to be served by one server in continuous time, while packets in 5G are multiplexed in the spectrum dimension (several servers) and time is slotted. [12] makes use of an M/GI/∞ model in order to study resource allocation for URLLC. [13] considers a M/M/m/K queue to model the system reliability for a worst case scenario where users are assumed to be at the cell edge. The work in [14] derives generic end-to-end latency distribution of any network topology which enables to determine percentiles, and applies it to the cases of exponential and deterministic service distributions. The work in [15] considers a risk-resistant approach (risk is outage) to minimize the latter based on an M/G/1 queuing model. In our work, however, we assume general arrival distribution, not restricted to Poisson and we consider a discrete time system with frequency multiplexing.

As of works dealing with URLLC, eMBB and/or massive machine-type communications (mMTC), the work in [16] investigates the radio resource allocation across these three types. The primary goal was to maximize network utility while ensuring QoS standards are met with an upper bound on the resources that can be reserved. However, for the URLLC slice, they use a link-layer model and effective capacity theory based on large deviation theory to describe the QoS requirements, which differs from our method, where we focuses on packet loss due to delay violation. In [17], the focus is on resource allocation in the downlink to accommodate both eMBB and URLLC packets. Their approach assumes that URLLC packets are either immediately served or discarded, and that they are transmitted by puncturing eMBB packets. The overarching objective is to maximize eMBB slice throughput while guarantying some minimal number of URLLC packets. This differs from our work where the objective is again to serve URLLC packets subject to delay and reliability constraints. To address resource allocation, they introduce two problems. The first one operates on a slot basis to satisfy the throughput requirement of the eMBB slice, while the second one operates on a mini-slot basis to serve as many URLLC packets as possible while minimizing also the loss in the eMBB throughput slice. In [18] the authors aimed to maximize system throughput in a distributed massive MIMO system serving eMBB and URLLC slices. They ensured a minimum data rate as a QoS criterion but did not consider packet loss probability. In their approach, they assume that all users are active and that traffic is known. In contrast, our work addresses uncertainty in user activity.

Other works consider Age of Information (AoI) and not delay and reliability as the performance criteria for time-critical traffic. In [19], the URLLC slice's performance is modeled based on the probability that the AoI exceeds a specific threshold. This is done within a discrete-time framework, considering Last Come First Served (LCFS) scheduling and preemption. A similar approach is taken in [20], where the aim is to maximize the average information freshness for all users. The authors propose online and offline algorithms based on MDP for user selection control. These studies focus on the case of serving a single user at a time and again on the AoI while we consider packet multiplication in the frequency domain and focus on URLLC performance in terms of outage (probability of packets received beyond a certain delay budget), as defined by 3GPP.

Paper contributions

We propose in this paper a new control model which enables to meet URLLC stringent performance target while minimizing the allocated resources. Recall that the QoS requirements for URLLC are specified by bounds on two metrics: one on delay, 1ms, and one on reliability, which is the probability of exceeding the delay bound, and which is set to 10^{-5}. The delay violation probability which we use in our work combines both metrics, as it quantifies the probability of violating the delay bound, which should be equal to the reliability bound. To do so, we first develop a generic mathematical model for the evolution of the number of packets waiting in the queue and show how to derive the delay violation probability for a policy that controls the resource allocation dynamically. We formulate an optimization problem whose objective is to preserve the system resources while achieving the performance target. We then study the structure of the problem and propose adequate algorithms that converge quickly to efficient resource allocation policies.

This paper has original contributions on two fronts: modeling and control. Specifically:

- We propose a discrete-time queuing model on the mini-slot basis where packets belonging to different users may have different radio conditions and occupy different amounts of resources. This model does not make particular assumptions on the arrival process of packets and models the service discipline following the 5G time/frequency scheduling.
- We derive the delay violation probability, defined as the probability that the packets waiting in the queue at some time cannot be completely served within the delay budget.
- We propose two offline algorithms for resource reservation that adapt to the traffic conditions and achieve minimal resource reservation while meeting stringent QoS requirements, with rapid convergence.
- We propose an online algorithm based on a stochastic bandit framework and show how our analytical model can be used for guiding the sequential policy selection process until convergence to the optimal one.
- We apply our resource allocation algorithms to two cases: when resource allocation to URLLC service is

immediate upon decision, at the mini-slot level, as well as to less flexible case where the decision is effective at some reconfiguration epochs, e.g., at the slot level, while URLLC is scheduled on the mini-slot level.

Paper organization

In section II, we model the delay violation probability and resource consumption for both fixed and dynamic resource reservation. Section III describes the policy performance for both immediate and delayed resource allocation settings. Section IV formulates an optimization framework for meeting the violation target while minimizing the cost in terms of resources and proposes online and offline efficient algorithms for finding the optimal policy. Section IV validates our model against simulations and illustrates the optimal policies for resource allocation. Section V eventually concludes the paper.

II. System and Model

In the following, we will use the notation $\mathcal{X}$ for sets, $|\mathcal{X}|$ for the cardinality of sets, $\mathbf{x}$ for line vectors and $\mathbf{x}^T$ their transpose, and $\mathbf{X}$ for matrices, of elements X_{ij}.

A. System model

We consider a 5G cell where resources are organized into Physical Resource Blocks (PRBs) in the frequency and time domains. The time domain has two levels of granularity: the slot level, on the order of 1ms, for the scheduling of eMBB users, and the mini-slot level, with smaller duration denoted by T, to serve URLLC users. Our aim is to devise policies which determine the (minimal) quantity of resources that should be reserved for URLLC traffic so as to meet its QoS requirement. To this end, we investigate two main cases: a first one where URLLC resources are adapted at each mini-slot, in a fully flexible way. The difficulty in this case resides in anticipating bursts of packet arrivals by proactively reserving resources within the set of available ones. The second case, which we present in the next section, focuses on the setting where resource reservation is updated at the border of the slot, and not at each mini-slot, considering that the resources are occupied by eMBB traffic and cannot be preempted until the next slot. As stated above, the QoS requirement is formulated in terms of the probability that the delay exceeds a maximal allowed budget (1ms); this delay violation probability should be kept smaller than 10^{-5}.

B. Traffic model

We start by describing the radio configuration that depends on the sub-carrier spacing and the mini-slot definition. Even if a slot is composed of 14 symbols its length depends on the subcarrier spacing (1 ms for 15 kHz and 0.5 ms for 30 kHz, and so on). However, to accommodate URLLC packets with stringent delay requirements, 5G NR also allows for mini-slot transmissions, providing scalable TTIs. Each mini-slot could consist of 2, 4, or 7 symbols, corresponding to mini-slot transmission times of 70µs, 140µs, or 250µs, respectively when using the 30 KHZ subcarrier spacing. In our work, we assume the use of a 30 kHz subcarrier spacing and mini-slots containing 7 symbols. Thus the maximum number of consecutive slots that a given packet can stay in the buffer is in this case equal to 4 ($\delta = 4$), which corresponds to 1ms delay budget. In each mini-slot, URLLC packets are generated following some stochastic process, and packet arrivals in different mini-slots are independent. Packets are small, and might be of equal or variable sizes. As of the delay budget, a packet may stay for δ mini-slots in the system before its delay budget expires otherwise it is in violation.

In this section, we will not make further assumptions on the arrival process and packet sizes, but we will suppose that, during mini-slot t, the number of resources required for serving new arriving packets is a discrete random variable $a(t)$, defined in some subset $\mathcal{A}$ of $\mathbb{N}$, the set of positive integers[1]. Let z_j be the probability that $a(t) = j$, $j \in \mathcal{A}$. For the ease of notation, we define $z_j = 0$ for $j \notin \mathcal{A}$. We will show in section V how the z_j's are derived in typical 5G scenarios.

C. Delay violation model for a fixed reservation

We start by the simple, yet practical case, where the amount of resources reserved for URLLC is constant and equal to R PRBs. This is done at the mini-slot level. As the queue follows a First Come First Serve (FCFS) discipline, a packet generated in a mini-slot sees other packets generated within the same mini-slot and those generated in previous mini-slots that are still in the queue waiting for service, if any.

In mini-slot t, knowing that there are R reserved resources, the queue length after scheduling, denoted by $B(t)$, is defined as the amount of resource units that will be needed in the future mini-slots ($t' > t$) to serve the backlogged traffic after using all the resources of mini-slot t, and is given by:

$$B(t) = \left(a(t) + B(t-1) - R\right)^+, \tag{1}$$

where $(x)^+ = \max(0, x)$. $a(t)$ is, as stated above, the amount of resources needed for serving new packets arriving in mini-slot t, $B(t-1)$ is the queue length from the previous mini-slot and $a(t) + B(t-1)$ is the total amount of resources needed for serving all the backlogged packets from previous mini-slots plus the new packets arriving at mini-slot t. As there are R resources available in each mini-slot, at most R among the required resources are used, and the remaining packets, if any, are backlogged in the next mini-slot[2].

We define the probability of delay violation V as the probability that the packets that are present in mini-slot t cannot all be served until mini-slot $t + \delta - 1$:

$$V = \lim_{t \to \infty} Pr[B(t) > (\delta - 1)R] \tag{2}$$

[1] $a(t)$ is in PRBs reserved in a mini-slot. For instance, for two arriving packets and when, considering the Modulation and Coding Scheme (MCS) and the packet size, the first packet needs 1 PRB and the second 3 PRBs, $a(t) = 4$.

[2] The term queue length can be considered as an abuse of language, as it refers classically to the number of packets in the queue, while it designates here the PRBs required to serve the packets. We note that this information is known to the scheduler as MCS is decided before the packet is queued.

as packets that are still in the queue after scheduling have been in the system at least during mini-slot t, and some of them will exceed the delay limit if the resources in the next $(\delta-1)$ mini-slots are not sufficient to serve all of them. We will show afterwards that this limit exists.

D. Model for a dynamic resource reservation policy

We now consider the more general case of a controller that dynamically adapts URLLC resources at every mini-slot for achieving a low violation probability, not exceeding a small value v_{max}. We note that the resource adaptation has to be performed without knowing the arrivals in the next mini-slot, but based on the queue length of the current one.

We define a policy by a set of resource reservations for each queue length event. Formally, let $\mathcal{R} \subset \mathbb{N}$ be the set of possible resource reservations for URLLC, and define $\mathbf{r}$ as the vector grouping the elements of $\mathcal{R}$ sorted in ascending order[3]. Under policy $\mathbf{p}$, when the queue length is equal to $b \in \mathcal{B}$, the system reserves an amount of resources for the next mini-slot equal to $p_b \in \mathcal{R}$. The set of all possible policies is $\mathcal{P}$. Formally, we define a policy $\mathbf{p}$ as a vector of $|\mathcal{B}|$ elements, with $p_b \in \mathcal{R}$.

For convenience, we define the matrix $\mathbf{M}(\mathbf{p})$ of zeros and ones, of size $|\mathcal{B}| \times |\mathcal{R}|$, where there is a single "1" for each line b, placed at position (b, j) such that p_b is the jth element of $\mathbf{r}$. Under policy $\mathbf{p}$, we can compute the amount of resource reservation when the queue length is equal to $b \in \mathcal{B}$ by:

$$R(b, \mathbf{p}) = p_b = \mathbf{M}(\mathbf{p})[b+1, :]\mathbf{r}^T \tag{3}$$

where $\mathbf{M}(\mathbf{p})[b+1, :]$ is the $b+1$ row of matrix $\mathbf{M}(\mathbf{p})$.

Note that R was taken as constant in the previous section, now it depends on the queue length b and policy $\mathbf{p}$.

Under policy $\mathbf{p}$, the queue length evolves following:

$$B(t) = \left(a(t) + B(t-1) - R(B(t-1), \mathbf{p})\right)^+, \tag{4}$$

The delay violation probability is:

$$V(\mathbf{p}) = \lim_{t\to\infty} Pr[B(t) > \sum_{i=1}^{\delta-1} R(B(t+i), \mathbf{p})] \tag{5}$$

Equation (5) reduces to equation (2) for R taken as constant, as in the previous section. The average amount of consumed resources, which quantifies the cost of the policy, is:

$$C(\mathbf{p}) = \lim_{t\to\infty} E[R(B(t), \mathbf{p})] \tag{6}$$

This equation represents the expected number of resources C to be reserved within the system, under policy $\mathbf{p}$, considering the distribution of the system's state, i.e., queue length $B(t)$, as it approaches its long-term or limiting behavior.

Our goal is to minimize this cost while ensuring that the delay violation probability remains below a threshold v_{max}. This leads to the following optimization problem:

$$\mathbf{p}^* = \arg\min_{\mathbf{p}\in\mathcal{P}} [C(\mathbf{p})] \quad \text{subject to } V(\mathbf{p}) \leq v_{max} \tag{7}$$

[3]For instance, if $\mathcal{R} = [R_{min}, R_{max}]$, with $R_{min} < R_{max}$ positive integers, $|\mathcal{R}| = R_{max} - R_{min}$ and $\mathbf{r} = (R_{min}, R_{min}+1, ..., R_{max})$. Another practical example is when there are two possible resource allocations for URLLC, i.e., $\mathcal{R} = \{R_{min}, R_{max}\}$, $\mathbf{r} = (R_{min}, R_{max})$ and $|\mathcal{R}| = 2$.

Here too, taking R as constant makes $C = R$.

III. Performance Model

We now turn to evaluating the performance of a given policy, in terms of delay violation probability and cost.

A. Evaluation of the performance of an arbitrary policy

In order to compute the violation probability, we need to determine the distribution of the queue length in steady-state. We observe that eqn. (4) involves three random variables:

- $B(t)$, that is a discrete integer random variable that takes its values in $[0, \infty[$. Let $q_b(t)$ be the probability that $B(t)$ takes the value $b \in [0, \infty[$.
- $B(t-1)$, that has the same limiting distribution as $B(t)$,
- and $a(t)$, that is the amount of resources needed for serving the new packet arrivals. $a(t)$ is independent from $B(t-1)$ and takes its values in some set $[0, a_{max}]$, where a_{max} is a positive integer (that might be infinite).

In steady-state, let q_b be the probability that the queue length is equal to $b \geq 0$. Setting a maximal queue length $B_{max} >> R$ and defining the space of queue lengths $\mathcal{B} = [0, B_{max}] \subset \mathbb{N}$, we write the following system of linear equations:

$$\mathbf{q}(\mathbf{p}) = \mathbf{q}(\mathbf{p})\mathbf{Q}(\mathbf{p}) \tag{8}$$

where $\mathbf{Q}(\mathbf{p})$ is the transition matrix under policy $(\mathbf{p})$ with element Q_{jb}, j and $b \in \mathcal{B}$, designating the transition probability from queue length j at time t to queue length b at time $(t+1)$, and $\mathbf{q}$ is the vector of queue lengths probabilities. The elements of the transition matrix $\mathbf{Q}(\mathbf{p})$ are given by:

$$Q_{jb}(\mathbf{p}) = \begin{cases} z_{b+R(b,\mathbf{p})-j}, & \text{if } b \in]0, B_{max}[\\ \sum_{i\geq b+R(b,\mathbf{p})-j} z_i, & \text{if } b = B_{max} \\ \sum_{i\leq R(b,\mathbf{p})-j} z_i, & \text{if } b = 0 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Recall that $z_j = Pr[a(t) = j], j \in \mathcal{A}$. This set of equations can be solved by adding the normalizing equation:

$$\mathbf{q}(\mathbf{p})\mathbf{q}^T(\mathbf{p}) = 1 \tag{10}$$

We establish the existence of the limit in equation (2), which relies on the properties of the Markov chain characterized by transition matrix $\mathbf{Q}$ as delineated in equation (9). This Markov chain satisfies both the irreducibility and aperiodicity conditions. To show irreducibility, we make use of the following assumptions: i. the number of possible reservations $R(b, \mathbf{p})$ should be smaller than the maximal value of $a(t)$, and ii. the probability vector of $a(t)$, denoted by z, should have no zero values. These assumptions enable the observation that from any state b of the Markov chain, one can either remain at b, move to $b-1$, or transition to $b+1$. This observation directly implies that for any pair of states x and y, it is always feasible to transition from state x to state y within at most $|y-x|$ steps of the Markov chain, thereby establishing irreducibility. As of aperiodicity, for an irreducible Markov chain to be aperiodic, it suffices to have at least one self-loop at one state. For instance, let us consider a scenario

where there are no backlogged packets and the number of resource blocks required for serving the newly arriving packets is less than the reservation ($R(b,\mathbf{p}) \geq a(t)$). In such case, a self-loop exists at each state of the Markov chain, and hence aperiodicity. All in all, we conclude that the Markov chain under consideration fulfills both the irreducibility and aperiodicity conditions, yielding the existence of the limit in equation (2).

These steady-state probabilities $\mathbf{q}(\mathbf{p})$ enable the computation of the distribution of the amount of reserved resources. For this purpose, we define, for all $m \in \mathcal{R}$, $\mathcal{P}^{-1}(m,\mathbf{p})$ as the following subset of $\mathcal{B}$:

$$b \in \mathcal{P}^{-1}(m,\mathbf{p}) \Longleftrightarrow b \in \mathcal{B} \text{ and } R(b,\mathbf{p}) = m.$$

Note that $\mathcal{P}^{-1}(m,\mathbf{p})$ is a set while $R(b,\mathbf{p})$ is an integer, and $\cup_{m\in\mathcal{R}}\mathcal{P}^{-1}(m,\mathbf{p}) = \mathcal{B}$.

The policy cost (6) which quantifies the quantity of resources of a given policy can be computed by:

$$C(\mathbf{p}) = \mathbf{q}(\mathbf{p})\mathbf{M}(\mathbf{p})\mathbf{r}^T \tag{11}$$

From equation (3), which computes the number of PRBs to be reserved for each queue length b under policy $\mathbf{p}$, we have that $\mathbf{M}(\mathbf{p})r^T$ represents the policy $\mathbf{p}$, it is a vector with each entry denoting the number of PRBs to be reserved for each queue length. Multiplying this vector by $\mathbf{q}(\mathbf{p})$, which is the probability distribution of the queue length, yields the *expected* number of reserved resources based on policy $\mathbf{p}$.

Computing the delay violation of (5) is less straightforward as it depends on the amount of resources allocated during the next mini-slots, which is not constant. For this purpose, for every mini-slot $t \geq t_0$, we define the variable $S(t)$ as the cumulative resources the system has reserved for URLLC from time t_0 until t given that the system starts reserving resources at t_0 ($S(t_0) = 0$). We have the recursion:

$$S(t+1) = S(t) + R(B(t),\mathbf{p}).$$

The violation probability of equation (5) is thus:

$$V(\mathbf{p}) = \lim_{t\to\infty} Pr[S(t_0+\delta-1) < B(t)\,|\mathbf{p}] \tag{12}$$

We define the resource/queue state of the system at time t as the couple $(S(t),B(t))$. The transition probability between states (x,j) and (s,b) in one mini-slot is computed by:

$$T((x,j)\to(s,b)) = Q_{j,b}\mathbb{1}_{R(j,\mathbf{p})=s-x} \tag{13}$$

where $\mathbb{1}_{R(j,\mathbf{p})=s-x}$ is the indicator function. We note that $S(t)$ tends to ∞ when t increases. However, we are interested in its evolution on $\delta-1$ mini-slots, starting from $S(t)=0$, to account for the amount of accumulated resources since a queue length is generated. The maximal value of S after $\delta-1$ mini-slots is $S_{max} = (\delta-1)r_{R_{max}}$, and the state space of $S(t)$ is $\mathcal{S} \subset [0, S_{max}] \subset \mathbb{N}$. In matrix form, we define the transition matrix $\mathbf{Y}(\mathbf{p})$ of $(S(t),B(t))$, grouping these transition probabilities as a matrix of dimensions $S_{max}|\mathcal{B}| \times S_{max}|\mathcal{B}|$. By rearranging the vector of states into $|\mathcal{B}|$ blocks, with block b corresponding to the states $((0,b),...(S_{max},b)$, $\mathbf{Y}$ can be defined in blocks, as follows:

$$\mathbf{Y}(\mathbf{p}) = \begin{pmatrix} Q_{00}\mathbf{Y}^{(0)} & Q_{01}\mathbf{Y}^{(0)} & ... & Q_{0|\mathcal{B}|}\mathbf{Y}^{(0)} \\ Q_{10}\mathbf{Y}^{(1)} & Q_{11}\mathbf{Y}^{(1)} & ... & Q_{1|\mathcal{B}|}\mathbf{Y}^{(1)} \\ & & ... & \\ Q_{|\mathcal{B}|0}\mathbf{Y}^{(|\mathcal{B}|)} & Q_{|\mathcal{B}|1}\mathbf{Y}^{(|\mathcal{B}|)} & ... & Q_{|\mathcal{B}||\mathcal{B}|}\mathbf{Y}^{(|\mathcal{B}|)} \end{pmatrix} \tag{14}$$

Note that the components of $\mathbf{Y}$ depend on the policy, but we have omitted it from the notation for convenience. Sub-matrix $\mathbf{Y}^{(b)}(\mathbf{p})$ corresponds to the evolution of the accumulated resources $S(t)$, when the queue length is equal to b:

$$Y^{(b)}_{sx}(\mathbf{p}) = \begin{cases} 1, & \text{if } x = s + R(b,\mathbf{p}) \\ 1, & \text{if } s + R(b,\mathbf{p}) > S_{max} \text{ and } x = S_{max} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

The transition matrix of $(S(t),B(t))$ after $\delta-1$ steps is then computed by $\mathbf{Y}^{\delta-1}(\mathbf{p})$. Our objective is to study the evolution of the resource/queue length starting from t, knowing that $S(t)=0$. Let $\mathbf{y}_0(\mathbf{p})$ be the vector of probabilities of the couple (queue length, accumulated resources) at some reference time t when we start accounting for it. The steady-state probabilities for the queue length being computed as the solution of the set of equations (8), this probability vector is:

$$\mathbf{y}_0(\mathbf{p}) = [\underbrace{q_0(\mathbf{p}),0,...0}_{B=0,S_{max}\text{ terms}},...,\underbrace{q_{|\mathbf{B}|}(\mathbf{p}),0,...0}_{B=B_{max},S_{max}\text{ terms}}].$$

The resource/queue length state probability when the delay budget expires is then:

$$\mathbf{y}(\mathbf{p}) = \mathbf{y}_0(\mathbf{p})\mathbf{Y}^{\delta-1}(\mathbf{p}) \tag{16}$$

The violation probability is computed by:

$$V(\mathbf{p}) = \sum_{b\in\mathcal{B}}\sum_{s\in\mathcal{S};s<b} y_{b|S|+s}, \tag{17}$$

where y_i is the i^{th} element of the vector $\mathbf{y}$ given in equation (16). Vector $\mathbf{y}$ represents the state (cumulative number of reserved resources with cardinality $|S|$, queue length with cardinality $|\mathcal{B}|$) starting from time 0 where the cumulative number of resources is 0 until the delay budget δ expires. The index in $y_{b|S|+s}$ in the above equation represents the probability that the number of cumulative reserved resources is s when the system starts with b backlogged packets (multiplied by $|S|$ to account for all possible values of s). The first summation in the above equation iterates over all initial values of queue length b, while the second summation, constrained by $s<b$, calculates the probability that after $\delta-1$ time steps, the cumulative number of resources is smaller than the queue length in the initial state ($s<b$). This effectively computes the violation probability for each specific queue length b. This process is repeated for all values of b, ultimately yielding the violation probability value for the entire system.

B. Model extension to a delayed resource reservation

So far we considered a fully flexible resource allocation, where the system observes the queue status and reserves an amount of resources for URLLC within the set of available resources $\mathcal{R}$ at each mini-slot. However, in many practical settings, delay may occur before resources are to be effectively reserved, and the most common case is when the system serves jointly URLLC and eMBB traffic, scheduled on different scales: a mini-slot of length T for URLLC and a slot of length τT, with τ a positive integer, for eMBB. Once resources are allocated for eMBB, they are occupied for the whole slot and cannot be preempted for URLLC until the beginning of the next slot. In this context, the amount of resources will be reserved to URLLC depending on the length of the queue at the boundary of the slot, and not of the mini-slot.

A state is then defined by a three-tuple: the current queue length at the boundary of the slot which we denote by B_1 and for which we reserve $R(B_1, \mathbf{p})$ resources, the queue length at the boundary of the mini-slot which we denote by B_2, and an index ℓ, $\ell \in \{0, 1, \ldots, \tau - 1\}$, for the mini-slot we are in now (within one slot).

There are $|\mathcal{B}| \times |\mathcal{B}| \times \tau$ states. At time t (taking mini-slot as time unit), and knowing all the system states at t' where $t - \ell(t) \leq t' < t$, the elements of state $(B_1(t), B_2(t), \ell(t))$ are given as follows:

$$B_2(t) = (B_2(t-1) + a(t) - R(B_1(t-1), \mathbf{p}))^+ \tag{18}$$

$$B_1(t) = \begin{cases} B_2(t) = (B_2(t-1) + a(t) - R(B_1(t-1), \mathbf{p}))^+ \\ \text{if } t = k\tau \text{ for } k \in \mathbf{N} \\ B_1(t') \text{ s.t. } t - \tau < t' < t \text{ and } t' = k\tau \\ \text{otherwise} \end{cases} \tag{19}$$

where $a(t)$ is the quantity of resources needed for newly arriving packets in the mini-slot, and $\ell(t) = t - \tau \lfloor \frac{t}{\tau} \rfloor$ where $\lfloor x \rfloor$ is the greatest integer less than or equal to x.

The violation probability for a given policy $\mathbf{p}$ is now:

$$V(\mathbf{p}) = \lim_{t \to \infty} Pr[B_2(t) > \sum_{i=1}^{\delta - 1} R(B_1(t+i), \mathbf{p})], \tag{20}$$

and the average consumed resources becomes:

$$C(\mathbf{p}) = \lim_{t \to \infty} E[R(B_1(t), \mathbf{p})] \tag{21}$$

In order to calculate the distribution of the queue length in steady-state, we need first to define the transition probability matrix, denoted by $\mathbf{D}(\mathbf{p})$. It is of size $\tau|\mathcal{B}|^2 \times \tau|\mathcal{B}|^2$ and is composed of τ^2 matrices, denoted by $\mathbf{A}_{lk}$, each of size $|\mathcal{B}|^2 \times |\mathcal{B}|^2$. As $\ell(t)$ designates the mini-slot index, only the transition probabilities between states $(\cdot, \cdot, \ell(t))$ and $(\cdot, \cdot, \ell(t+1))$ are non-zeros, and as $\ell(t) \in \{0, \ldots, \tau - 1\}$ we have:

$$\mathbf{D}(\mathbf{p}) = \begin{bmatrix} 0 & \mathbf{A}_{01} & 0 & \cdots & 0 \\ 0 & 0 & \mathbf{A}_{12} & \cdots & \vdots \\ \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \cdots & \vdots & 0 & \mathbf{A}_{(\tau-2)\tau-1} \\ \mathbf{A}_{(\tau-1)0} & 0 & 0 & \cdots & 0 \end{bmatrix}$$

There are three different types of these non-zero matrices:

- $\mathbf{A}_{01}$ contains the transition probabilities of the system when the slot and mini-slot borders coincide,
- $\mathbf{A}_{\ell,\ell+1}$ with $\ell \in \{1, \ldots, \tau - 2\}$, expresses the transition probabilities within the slot,
- and $\mathbf{A}_{(\tau-1)0}$, represents the end of the slot, in which case ℓ is reinitialized to 0.

We show in appendix B how the matrices $\mathbf{A}$ can be expressed in terms of the transition matrix $\mathbf{Q}$ of the non-delayed system described in the previous section.

Let $\mathbf{d}(\mathbf{p})$ be the stationary probability of the queue length, it can be calculated by solving $\mathbf{d}(\mathbf{p}) = \mathbf{d}(\mathbf{p})\mathbf{D}(\mathbf{p})$.

In order to calculate the violation probability, we introduce a new expression for $\mathbf{S}(\mathbf{t})$, the cumulative resources the system has reserved for URLLC until time t, which is:

$$\mathbf{S}(t) = \mathbf{S}(t-1) + R(B_1(t), \mathbf{p}) \tag{22}$$

The violation probability is calculated as in equation (17) and the cost is calculated as in equation (11), where instead of $q(\mathbf{p})$ we use the probability distribution vector of B_1.

IV. Optimal policy derivation

The objective is to minimize resource consumption while respecting a target on the delay. For this purpose, we formulate the following optimization problem:

$$\mathbf{p}^* = \arg\min_{\mathbf{p} \in \mathcal{P}} [C(\mathbf{p})] \tag{23}$$
$$\text{subject to } V(\mathbf{p}) \leq v_{max}$$

The above model evaluates the performance of any policy in $\mathcal{P}$, for cases with immediate or delayed resource reservation. However, the number of possible policies is very large, and there is a need for efficient algorithms for selecting the policy that solves problem (23), other than the exhaustive search.

If no constraints were imposed on the structure of the policy, the cardinality of $\mathcal{P}$ is equal $|\mathcal{R}|^{|\mathcal{B}|}$. One can impose that $p_b \geq p_j$, if $b \geq j$, meaning that a larger queue length leads to a larger resource reservation. This constraint reduces the number of possible policies, but does not change the order of complexity. For instance, for $|B| = 2$, there are $\frac{|\mathcal{R}|(|\mathcal{R}|+1)}{2}$ policies[4], and for $|B| = 3$, the number of policies is on the order $\frac{|\mathcal{R}|^3}{6}$ policies[5].

[4]It is sufficient to observe that if p_0 takes the i^{th} value, p_1 can take any value between the i^{th} and the last one, leading to $|R| - i + 1$ possibilities. The number of policies is then $\sum_{i=1}^{|R|}(|R| - i + 1) = \frac{|\mathcal{R}|(|\mathcal{R}|+1)}{2}$

[5]A similar approach as for $|B| = 2$ is followed, fixing the value for p_0 ($|R|$ possibilities), and enumerating the policies knowing that there are two lines remaining, with a reduced space of possible resource allocations as $p_1 \geq p_0$.

A. An algorithm for policy selection

We now propose a heuristic algorithm for choosing the policy. We first define, on $\mathcal{P}$, the following ordering relation $\preceq$ for policies that will help designing the algorithm:

Definition 1. *Two policies* $\mathbf{p}$ *and* $\mathbf{g}$ *of* $\mathcal{P}$ *verify* $\mathbf{p} \preceq \mathbf{g}$*, if and only if, for all* $b \in \mathcal{B}$*,* $p_b = R(b, \mathbf{p}) \leq g_b = R(b, \mathbf{g})$.

Note that $\mathcal{P}$ is partially ordered with respect to $\preceq$. We have the following:

Lemma 1. *For any two policies* $\mathbf{p}$ *and* $\mathbf{g}$ *of* $\mathcal{P}$*, if* $\mathbf{p} \preceq \mathbf{g}$*, then* $V(\mathbf{p}) \geq V(\mathbf{g})$.

Proof. If we reserve more resources for any amount of packets in the queue, the delay violation is necessarily lower as the queue vanishes more quickly, knowing that the exogenous arrival process is the same. □

Note that we consider also that

$$C(\mathbf{p}) \leq C(\mathbf{g}) \text{ for } \mathbf{p} \preceq \mathbf{g}, \tag{24}$$

as there is systematically a larger amount of resources reserved for each state. We show numerically the validity of our assumption (24) in appendix (C).

We define the policy incrementation $\oplus$ in $\mathcal{P}$ by a constant vector $\mathbf{c}$ of real elements in $\mathbb{R}$ by the following:

Definition 2. *For all* $\mathbf{p} \in \mathcal{P}$ *and* $\mathbf{c} \in \mathbb{R}^{|B|}$*,* $\mathbf{g} = \mathbf{p} \oplus \mathbf{c}$ *is the element of* $\mathcal{P}$ *such that:*

$$g_b = \min[g_{b+1}; \arg\min_{l \in \mathcal{R}} \{|p_b + c_b - l|\}] \tag{25}$$

with the convention $g_{|B|+1} = R_{max}$.

Based on lemma 1, we propose algorithm 1 for problem (23). It starts by a random policy $\mathbf{p}_0$ and a random vector $\mathbf{e}$ of 0's and 1's. The first bloc (lines 4 to 6) corresponds to a policy $\mathbf{p}_0$ with large delay violation that is incremented by $\mathbf{e}$ until reaching an acceptable performance. The second bloc (lines 7 to 9) corresponds to the opposite case where the violation is too low and the policy is decremented by $\mathbf{e}$ until reaching the target. The stopping rule is not straightforward, as the minimal cost is not known a priori. This is why at least N_{min} paths are tested, for constructing some empirical statistics, and the algorithm is stopped when the cost of the policy is very close (e.g., by 1%) to the minimal previously observed cost.

B. Optimal policy on an ordered subset of threshold policies

In the previous section, we proposed algorithm 1 that explores policies with a random structure, determined by a random policy $\mathbf{p}_0$ and a random constant shift $\mathbf{e}$. While policies explored on one path are ordered, policies encountered on different paths are not necessarily comparable. Our objective here is to define a totally ordered subset of policies on which it is possible to perform more efficient search algorithms. We define the following translation function on $\mathcal{P}$:

Algorithm 1 Policy search algorithm

1: Input: the probability distribution z
2: **repeat**
3: $\quad \mathbf{p}_0 \leftarrow$ random from $\mathcal{P}$
4: $\quad \mathbf{e} \leftarrow$ random from $\{0,1\}^{|B|}$
5: $\quad$ **while** $V(\mathbf{p}_0) > v_{max}$ **do**
6: $\quad\quad \mathbf{p}_0 \leftarrow \mathbf{p}_0 \oplus \mathbf{e}$
7: $\quad$ **end while**
8: $\quad$ **while** $V(\mathbf{p}_0) < v_{max}$ **do**
9: $\quad\quad \mathbf{p}_0 \leftarrow \mathbf{p}_0 \oplus -\mathbf{e}$
10: $\quad$ **end while**
11: **until** number of random policy generations $> N_{min}$ and $C(\mathbf{p}_0)$ is very close to the minimum observed cost
12: Output $\mathbf{p}^*$ is the policy with the smallest observed cost.
13: =0

Definition 3. $\forall \mathbf{p} \in \mathcal{P}, tr(\mathbf{p}) \in \mathcal{P}$ *such that:*

$$R(b, tr(\mathbf{p})) = \begin{cases} R(b-1, \mathbf{p}) & \text{if } b > 0 \\ r_{i-1} & \text{if } b = 0,\ R(b, \mathbf{p}) = r_i,\ i > 1 \\ R(b, \mathbf{p}) & \text{otherwise} \end{cases} \tag{26}$$

where r_i *is the* i^{th} *element of* $\mathbf{r}$*, the vector grouping the elements of set* $\mathcal{R}$ *sorted in ascending order.*

Defining $tr^{(n)}(\cdot)$ as the n times function composition of $tr(\cdot)$, we can build a subset $\hat{\mathcal{P}}$ of $\mathcal{P}$, by translation:

Definition 4. *The extended threshold set* $\hat{\mathcal{P}}$ *is defined as the subset of* $\mathcal{P}$*, grouping the maximal allocation policy* $\sup_{\mathcal{P}}(\mathbf{p})$ *and its subsequent one-step translations.*

One can see that starting from $\sup_{\mathcal{P}}(\mathbf{p})$, which is the maximal policy in the set of all policies $\mathcal{P}$, and applying the translation function (following definition 3 of the translation function), we need $|R|-1$ translations so that the first element of this policy is $r_1 = R_{\min}$, and for all the elements of this policy to be equal to $R_{\min}$ we need $B_{\max}$ translations. Adding $\sup_{\mathcal{P}}(\mathbf{p})$ policy to $\hat{\mathcal{P}}$, we get that:

$$|\hat{\mathcal{P}}| = B_{max} + 1 + |\mathcal{R}| - 1 = B_{\max} + |\mathcal{R}|$$

Indeed, two families of policies belong to $\hat{\mathcal{P}}$:

- Moderate policies ranked $i \in [0, B_{max}]$, that start by allocating R_{min}, until a threshold $B(t) = B_{max} - i$, staring from which the allocation increases by one step in $\mathbf{r}$ each time the queue length increases until reaching R_{max}.
- Aggressive policies ranked $i \in [B_{max}+1, B_{max}+|\mathcal{R}|]$, that start, for $B(t) = 0$, by a reservation equal to $r_{i-B_{max}+1}$, and then increases until reaching R_{max}.

To introduce this family of policies, we illustrate a toy example by describing the shape of the policies in the ordered set $\hat{\mathcal{P}}$, when $r = \{R_{min} = R_1, R_2, R_{max} = R_3\}$ and $B_{max} = 4$ in figure 1. The y-axis refers to the amount of resources reserved following a policy $\mathbf{p} \in \hat{\mathcal{P}}$ and the x-axis specifies the possible size of the queue in the system. For each

policy, we give the number of resources reserved at each state (queue length). We know that $|\hat{\mathcal{P}}| = 7$, the first 5 policies are categorized as moderate policies, whereas 6^{th} and 7^{th} are aggressive policies.

Lemma 2. *$\hat{\mathcal{P}}$ is totally ordered with respect to $\preceq$.*

Proof. First we recall that $tr^i(\mathbf{p}) \preceq \mathbf{p}$ for $i \geq 1$. Let $\mathbf{p}_1$ and $\mathbf{p}_2$ be two policies in $\hat{\mathcal{P}}$, following the setup given in definition (4). One can notice that $\mathbf{p}_1$ and $\mathbf{p}_2$ can be expressed as $tr^i(sup_{\mathcal{P}}(\mathbf{p}))$ and $tr^j(sup_{\mathcal{P}}(\mathbf{p}))$ respectively for specific values of i and j. Therefore, if $i \leq j$, we have $\mathbf{p}_1 \preceq \mathbf{p}_2$ and vice versa. □

We are now able to define the distance metric in $\hat{\mathcal{P}}$:

Definition 5. *Let $\mathbf{p}_1 \preceq \mathbf{p}_2$ be two policies in $\hat{\mathcal{P}}$, the distance between them $|\mathbf{p}_2 - \mathbf{p}_1|$ is defined as the number of translations to reach $\mathbf{p}_1$ starting from $\mathbf{p}_2$ (or simply the difference of ranks between $\mathbf{p}_1$ and $\mathbf{p}_2$). In other terms, $|\mathbf{p}_2 - \mathbf{p}_1| = \Delta$ if and only if $tr^{(\Delta)}(\mathbf{p}_2) = \mathbf{p}_1$.*

Based on the above, and restricting ourselves to threshold policies, we define the Dichotomy algorithm 2. We suppose that $V(\sup_{\mathcal{P}}(\mathbf{p})) < v_{max}$, otherwise there is no solution that satisfies the delay requirement in $\mathcal{P}$. The algorithm starts within set $I_1 = [\mathbf{p}_{\min}, \mathbf{p}_{\max}]$, where $\mathbf{p}_{\min}$ allocates R_{min} constantly, and $\mathbf{p}_{\max} = \sup_{\mathcal{P}}(\mathbf{p})$ allocates R_{max} constantly. We then consider the intermediate policy $\mathbf{p}$, obtained by translating $\mathbf{p}_{\max}$ by half the distance with $\mathbf{p}_{\min}$ (Dichotomy). If, for this intermediate policy, the violation probability is larger than the target, $\mathbf{p}_{\min}$ is replaced by it, resulting in $I_2 = [\mathbf{p}, \mathbf{p}_{\max}]$ otherwise, $\mathbf{p}$ replaces $\mathbf{p}_{\max}$ and we get $I_2 = [\mathbf{p}_{\min}, \mathbf{p}]$. This Dichotomy is repeated until the distance between the two extremes is 1.

Algorithm 2 Dichotomy search algorithm

```
Input: the probability distribution z
p_max ← sup_P(p)
p_min ← inf_P(p)
Δ ← ⌊ |p_max − p_min| / 2 ⌋
while Δ > 1 do
    p ← tr^(Δ)(p_max)
    if V(p) > v_max then
        p_min ← p
    else
        p_max ← p
    end if
    Δ ← ⌊ |p_max − p_min| / 2 ⌋
end while
p ← p_max
Output p
=0
```

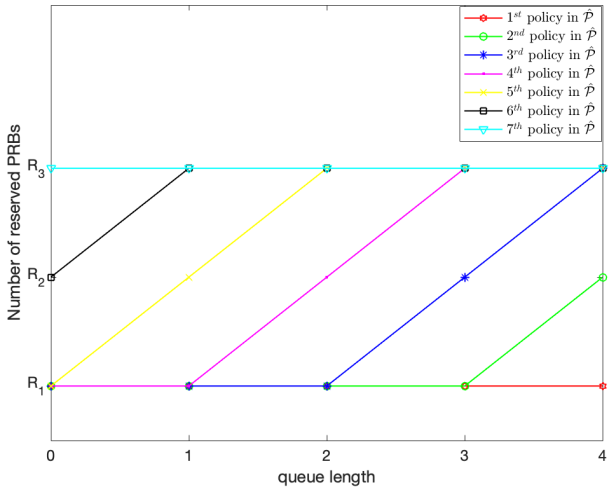

Fig. 1: The policies in the ordered set $\hat{\mathcal{P}}$

C. Dichotomy algorithm convergence

We prove the convergence of the Dichotomy procedure to the solution of (23) in the following theorems.

Theorem 1. *There exists a unique solution for problem* (23) *denoted by* $\mathbf{p}^*$.

Proof. The proof of the existence of a global solution to problem (23) can be inferred from two observations: firstly, $V(\inf_{\mathcal{P}}(\mathbf{p})) > v_{\max}$ and $V(\sup_{\mathcal{P}}(\mathbf{p})) < v_{\max}$, indicating the existence of a feasible policy within the set $\hat{\mathcal{P}}$. Additionally, since the cost function is increasing, there exists a unique solution denoted by $\mathbf{p}^*$. □

Lemma 3. *Let I_k be the interval described in the dichotomy algorithm at iteration k, function $C_k = \max_{p \in I_k} C(p)$ exhibits a monotonically decreasing behavior. Furthermore, within each interval I_k, there exists at least one feasible policy p.*

Proof. The argument behind this lemma is that following the fact that $I_k \supseteq I_{k+1}$ it is obvious that C_k is a decreasing function in k. Moreover, at each iteration of the Dichotomy algorithm, we have that $V(p_1) < v_{\max}$, and thus at least one feasible policy exists in each iteration. □

Theorem 2. *The Dichotomy algorithm converges to the optimal policy $\mathbf{p}^* = \cap_{k=1}^{\infty} I_k$, with each I_k representing the set of policies considered in the k-th iteration of the Dichotomy algorithm.*

Proof. The proof of this theorem unfolds in two steps. Firstly, we demonstrate that the Dichotomy algorithm converges to a single point. Secondly, we prove that this single point is the optimal policy in $\hat{\mathcal{P}}$. For the first step, in order to ensure the convergence of the Dichotomy algorithm to a single policy, we must satisfy the conditions outlined in Theorem 4 of [21], known as the Nested Interval Theorem. This theorem asserts that if $S_1 \supseteq S_2 \supseteq \cdots \supseteq S_k \supseteq \ldots$ is a sequence of nested, closed, bounded, non-empty intervals, then $\cap_{k=1}^{\infty} S_k$ is non-empty. Additionally, if $|S_k| \to 0 (k \to \infty)$, then $\cap_{k=1}^{\infty} S_k$

consists of a single point. Here, S_i can be interpreted as the set of considered policies in iteration i of the Dichotomy algorithm. As previously described in the Dichotomy algorithm, we initiate the process with the interval $I_1 = [p_{\min}, p_{\max}]$. Subsequently, we select I_2 as either $[p_{\min}, p]$ or $[p, p_{\max}]$, where $p = tr^{(\Delta)}(p_{\max})$ and $\Delta = \frac{|p_{\max}-p_{\min}|}{2}$. This naturally establishes a nesting relationship, i.e., $I_1 \supseteq I_2 \supseteq \ldots \supseteq I_K$. Moreover, it is obvious that the length of these intervals, denoted as $|I_k|$, decreases as k increases, following the pattern $|I_k| = \frac{|p_{\max}-p_{\min}|}{2^k} \to 0 (k \to \infty)$. This construction satisfies the conditions specified in the Nested Interval Theorem. For the second and final step, we show in lemma 3 that this single policy is the optimal solution of equation (23).

Thus we proved the two steps needed for the convergence of the Dichotomy algorithm to the unique optimal policy. □

Note 1. *We now assess the complexity of both algorithms, the Policy Search and Dichotomy. At each iteration, we need to calculate the overflow probability and thus solve both equations* (8) *and* (16)*. Solving equation* (8) *has a complexity of* $\mathcal{O}(|\mathcal{B}|^3)$ *and solving equation* (16) *requires* $\mathcal{O}((\delta-1)(S_{\max} \times |\mathcal{B}|)^2)$*. To this we add the complexity of updating the policy, it is equal to* $\mathcal{O}(|\mathcal{B}| \times |\mathcal{R}|)$ *for the Policy Search algorithm and* $\mathcal{O}(|\mathcal{B}|)$ *for the Dichotomy one. In total, for the Policy search algorithm, as we require at least* N *iterations, the complexity is:*

$$\mathcal{O}(N(|\mathcal{B}| \times |\mathcal{R}| + |\mathcal{B}|^3 + (\delta-1)(S_{\max} \times |\mathcal{B}|)^2))$$

The number of iterations in the Dichotomy algorithm depends on how many times we can divide $\mathcal{B} + |\mathcal{R}| - 1$ *by 2, which is* $\log_2(\mathcal{B} + |\mathcal{R}| - 1)$*. Thus, the complexity is:*

$$\begin{aligned}&\mathcal{O}(\log_2(\mathcal{B} + |\mathcal{R}| - 1)(|\mathcal{B}| + |\mathcal{B}|^3 + (\delta-1)(S_{\max} \times |\mathcal{B}|)^2))\\ &= \mathcal{O}(\log_2(\mathcal{B} + |\mathcal{R}| - 1)(|\mathcal{B}|^3 + (\delta-1)(S_{\max} \times |\mathcal{B}|)^2))\end{aligned}$$

D. Online dynamic resource allocation as a unimodal bandit

The above algorithms 1 and 2 are based on the analytical solution of equations (8,17) which are used to evaluate the intermediate policies, that take as input the traffic and system parameters. This corresponds to an offline strategy where an optimal policy is determined for each setting. However, a network operator may rely on online experimentation where a policy is implemented and tested for some time T_e. For $T_e < \infty$, the violation rate observed for a policy $\mathbf{p}$ is a random variable, distributed around $V(\mathbf{p})$.

Based on the above, we define a stochastic bandit problem on $\hat{\mathcal{P}}$ whose arms are the different policies. When an arm $\mathbf{p}$ is selected, we define its reward by $\mu(\mathbf{p}) = -|V(\mathbf{p}) - v_{\max}|$ where $v_{\max}$ is again the target delay violation probability. Our objective is to define an efficient algorithm π, from the subset of sequential algorithms Π, that finds the optimal arm while minimizing the regret. This latter is defined, after T_r rounds, as $G(T_r) = \mu^* - \sum_{n=1}^{T_r} \mu(\mathbf{p}^\pi(n))$, where $\mathbf{p}^\pi(n)$ denotes the arm selected in round n under algorithm π, and μ^* is the maximal reward achieved under the optimal policy $\hat{\mathbf{p}}^* \in \hat{\mathcal{P}}$.

Lemma 4. *If* $V(\inf_{\mathcal{P}}(\mathbf{p})) > v_{\max}$ *and* $V(\sup_{\mathcal{P}}(\mathbf{p})) < v_{\max}$*, then the reward function* $\mu(\mathbf{p})$ *is unimodal in* $\hat{\mathcal{P}}$*.*

Proof. Lemma 1 states that if a policy increases, its violation rate decreases. This means that, starting from $\inf_{\mathcal{P}}(\mathbf{p}))$, when policy $\mathbf{p}$ increases (reserves more resources earlier) and before reaching $\hat{\mathbf{p}}^*$, the delay violation probability decreases and the gap with $V(\hat{\mathbf{p}}^*)$ decreases, meaning that $\mu(\mathbf{p}) = -|V(\mathbf{p}) - V(\hat{\mathbf{p}}^*)|$ increases. For a policy that is larger than $\hat{\mathbf{p}}^*$, the violation further decreases, meaning that the gap starts increasing until reaching $v_{\max} - V(\sup_{\mathcal{P}}(\mathbf{p}))$. $\mu(.)$ is thus unimodal on $\hat{\mathcal{P}}$. □

The problem is then a unimodal bandit [22], and efficient algorithms can be proposed that exploit its structure for minimizing the regret as in [23]. However, as $|\hat{\mathcal{P}}|$ is large, we might approximate it as a continuous unimodal bandit with continuous arms $x \in [0, 1]$, with x defined as the rank of the policy in $\hat{\mathcal{P}}$ divided by $|\hat{\mathcal{P}}|$. For solving such problems, [24] shows that Stochastic Polychotomy (SP) algorithms exhibit regrets and optimization errors with optimal scaling. SP algorithms consist in successively narrowing an interval in $[0, 1]$ while ensuring that the best arm remains in this interval with high probability, and can be seen as an extension of the Dichotomy algorithm in the deterministic case. In particular, the SP_K algorithm uses the IT_K interval trimming subroutine defined in [24], with K sampled arms within the input interval. The idea is that, if a subroutine starts with some interval $I = [\underline{i}, \overline{i}]$, K arms (policies) are sampled with ranks $\underline{i} \leq i_1 \leq ... \leq i_K \leq \overline{i}$, it tests sequentially the sampled arms until collecting sufficient information about the reward of each of the arms. It then removes one of the extremities of the initial interval and outputs $I_1 = [\underline{i}, i_K]$ (respectively $I_2 = [i_1, \overline{i}]$), if it estimates that the optimal arm lies in I_1 (respectively I_2) with a large probability. I is initialized as $[0, |\hat{\mathcal{P}}|$ and narrowed down until finding the optimal arm i^* and its corresponding policy.

V. Numerical experiments

In our work, we assume the use of a 30 kHz subcarrier spacing and mini-slots containing 7 symbols. Thus the maximum number of consecutive slots that a given packet can stay in the buffer is in this case equal to 4 ($\delta = 4$), which corresponds to 1ms delay budget.

A. Integration within the network management architecture

Before evaluating the performance of the proposed control schemes, we show how they integrate within the 5G and Beyond architecture. The following three inputs are exploited by our scheme for deriving the optimal control policy:

1) Channel conditions distribution, i.e. the probability that a user in a given base station uses a given MCS k,
2) Traffic intensity, i.e., the number of new packets arriving per time slot at each base station, and
3) Queue status, i.e., the number of packets waiting in the scheduler queue at each time slot.

The first two inputs serve for computing the z_i's in the transition rates of equation (9), and are to be computed on a large time scale (in the order of tens of seconds). The latter input is updated in real-time, at the scheduler level and serves for computing the resource allocation for the next slot. We discuss in the following how these information can be obtained and where the proposed scheme is ideally implemented, and then how they can be exploited for deriving the optimal policy.

1) MCS and traffic analytics module: Based on the network measurements, the following information are computed, at the gNodeB level:

1) The number of active users U, i.e. the number of connected users that generate packets from time to time.
2) The activity probability f, i.e. the probability for a user to generate a packet during a slot.
3) The probability of using MCS k once a packet is generated, denoted by β_k, with $\sum_{k=1}^{K} \beta_k = 1$, K being the number of available MCS. In the case of a forced common MCS for URLLC, $K = 1$ and $\beta_1 = 1$.

Based on this information, the analytics module is able to provide the z_i's, used for computing the transition matrices in the performance model. Indeed, when a packet uses MCS k, it consumes an amount of PRBs equal to α_k, computed as follows: when an MCS with a spectral efficiency of ζ bit/s/Hz is used, and knowing that the PRB size is h Hz, a packet of size s bits occupies a number of PRBs equal to $\lceil \frac{s}{Th\zeta} \rceil$, where $\lceil x \rceil$ is the largest integer greater than or equal to x and T is the duration of the mini-slot.

The amount of resources consumed by a user u in mini-slot i is thus a random variable with distribution:

$$X_{u,i} = \begin{cases} 0, & \text{with prob. } (1-f) \\ \alpha_k & \text{with prob. } f\beta_k \end{cases} \tag{27}$$

Without loss of generality, we assume that the MCS are sorted in increasing order of spectral efficiency, meaning that $\alpha_1 > ... > \alpha_K$. The total number of resources requested by new packets generated in mini-slot i is then given by:

$$a(i) = \sum_{u=1}^{U} X_{u,i} \tag{28}$$

z_j describes now the probability that new arrivals in a mini-slot require j resources (PRBs), while in the homogeneous case, z_j was equal to the probability of having j new packet arrivals. z_j can now be computed using the multinomial distribution. Let $m(u_0, u_1, ..., u_K)$ be the probability of having, in a given mini-slot, a vector of generated packets $\vec{u} = (u_0, u_1, ..., u_K)$, where u_k is the number of packets with MCS k, and u_0 is the number of users that did not generate any packet. Let $\mathcal{U}$ be the space of all possible vectors $\vec{u}$ such that $\sum_{k=0}^{K} u_k = U$,

$$m(\vec{u}) = \frac{U!}{\prod_{k=0}^{K} u_k} (1-f)^{u_0} f^{U-u_0} \prod_{k=1}^{K} \beta_k^{u_k} \tag{29}$$

The term $\frac{U!}{\prod_{k=0}^{K} u_k}$ calculates the multinomial coefficient, accounting for the number of permutations of active and non-active users. The term $(1-f)^{u_0} f^{U-u_0}$ account for the probability of having u_0 non-active users and $U - u_0$ active users. The term $\prod_{k=1}^{K} \beta_k^{u_k}$ captures the probability that active users consume specific number of resources according to their MCS. Let $\mathcal{U}_j$, $j \in [0, \alpha_1 U]$, be the subset of $\mathcal{U}$ such that $\sum_{k=0}^{K} u_k \alpha_k = j$ ($\alpha_1 U$ corresponds to the limiting case where all users generate packets and use the worst MCS). The probability of consuming j resources is thus computed by:

$$z_j = \sum_{\vec{u} \in \mathcal{U}_j} m(\vec{u}) \tag{30}$$

2) Policy selection module: Based on the per-gNodeB MCS distributions provided by the analytics module, the policy selection module implements the performance evaluation model and outputs the per-gNodeB policy that achieves the target delay violation of 10^{-5} by solving problem (23). The solution is based on one of the algorithms of section IV. The real-time implementation of the policy is left for the scheduler that updates the URLLC resources by observing the instantaneous state of the queue. Note that the policy remains constant until a significant change is detected at the analytics modules within the Non RT or the Near RT RIC (e.g. traffic intensity or MCS distribution) that leads to a policy update.

B. Model validation with respect to simulations

Before using the models for deriving optimal policies, we assess their accuracy with respect to a system simulator as follows. A BS serves a set of URLLC users. At the start of each run, the positions of the users are drawn randomly in the cell, and their path losses are computed accordingly. The path loss for user i at time t is thus expressed by:

$$p_i(t) = \frac{a d_i^b}{GLS_i F_i(t)}, \tag{31}$$

where d_i is the distance to the base station, a and b are the path loss coefficients, G is the antenna gain, L accounts for the equipment imperfections, S_i and $F_i(t)$ are shadowing and fast fading, respectively. At each time step, the simulator computes the Signal-to-Interference-plus-Noise Ratio (SINR), as follows:

$$SINR_i(t) = \frac{P/p_i(t)}{\eta + I_i} \tag{32}$$

where η is the noise, P is the transmission power and I_i is the interference from other base stations. The system then dynamically selects the MCS based on the computed SINR values, and then computes the amount of PRBs necessary for serving the corresponding packets.

The performance model's validation is conducted as follows: a policy is established, associating each queue state with a reserved resource. This policy is then implemented both in the simulator and the model to compute the probability of delay violation and the cost. This process is performed for both common and heterogeneous MCS cases. The former case corresponds to a system where URLLC users do not

implement link adaptation, e.g. in order to avoid additional delays in channel estimation, and use a pre-determined common MCS, while the latter performs link adaptation and chooses a dynamic MCS that fits to the varying radio conditions. We consider a scenario where users have different arrival rates. We plot in Figure 2 the violation probability and the cost as a function of increasing *average* probability of users being active, where each user has a different probability f that is uniformly distributed between 0.1 and 0.4. This is conducted under the policy of order $B_{\max}+3$, where the set of possible amounts of reserved PRBs is $\mathcal{R}=\{0,1,\ldots,20\}$. The figure shows a perfect fit between both curves in both setups, which validates the accuracy of our model.

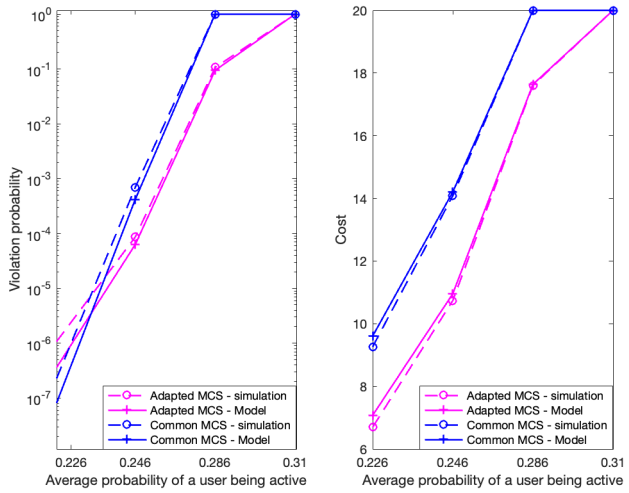

Fig. 2: Delay violation probability and cost ($U=30$, $\delta=4$)

C. Optimal policy illustration

We now investigate the performance of our proposed resource allocation policies. We consider $U=30$ users, each active with a probability $f=0.2$. Unless stated otherwise, we consider for illustration the case of homogeneous MCS.

1) Illustration of control policies on a simple policy set: We start, for illustration purposes, by policies defined on a reduced set of possible resource reservations, $\mathcal{R}$, so that exhaustive search on policies is possible. As a baseline, we consider the fixed resource allocation case, and apply the model of equation (2) with a target violation probability of $V(\mathbf{p})=10^{-5}$; the amount of needed resources, continuously reserved, is equal to $R=8$ resource units.

We illustrate a very simple policy with two reservation levels, R_1 and R_2 resources. In this case, the set of possible reservations is $\mathcal{R}=\{R_1,R_2\}$, a policy $\mathbf{p}$ corresponds to a threshold $\bar{B}$ on the queue length above which the resource reservation switches from R_1 to R_2:

$$p_b=\begin{cases}R_1, & b\leq\bar{B}\\ R_2, & \text{otherwise}\end{cases}$$

We plot in figure 3 the costs, in terms of allocated resources, and violation probabilities for different thresholds, for different sets of policies (we consider different values for R_1, we fix $R_2=8$ resources). If we focus on a given set $\mathcal{R}=\{R_1,R_2\}$, we observe that the violation probability increases and the cost decreases when $\bar{B}$ increases. Second, we observe that there is a jump in the violation probability that corresponds to threshold $\bar{B}=R_1(\delta-1)$, i.e., to a queue that cannot be evacuated with R_1 resources within the delay budget. Another important observation is that, for all the illustrated cases, the target performance can be achieved for some thresholds and that the optimization problem (23) has a solution that corresponds, for most of the cases, to a cost close to 6, i.e., with a gain of 25% with respect to the constant reservation case ($R=8$). Note that this limit cost corresponds to the average number of packets per slot ($Uf=30\times0.2$), so that the system is stable.

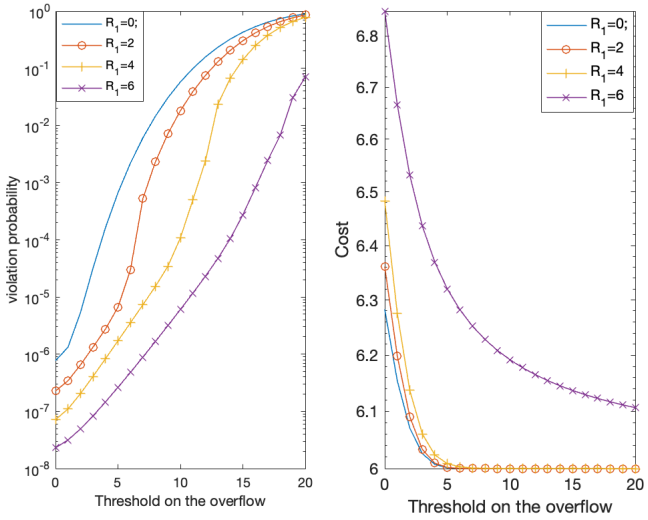

Fig. 3: Average cost and delay violation for the threshold policies on $\mathcal{R}=\{R_1,R_2\}$, R_1 varies while $R_2=8$.

We show in Figure 4, the convergence of the Dichotomy algorithm towards the optimal policy that solves equation (23) within the set $\hat{\mathcal{P}}$. The optimal policy is determined through an exhaustive search, initiated by selecting the policy with the highest rank in $\hat{\mathcal{P}}$ (supremum policy). This initial policy must satisfy the chance constraints. The Dichotomy algorithm adjusts resource reservation by iteratively translating to policies with higher or lower ranks, depending on whether the initial point in the Dichotomy algorithm is lower or higher than the optimal one, until reaching the smallest policy in $\hat{\mathcal{P}}$ that satisfies the chance constraint. Figure 4 illustrates the distance between the policy obtained from the Dichotomy algorithm and the optimal one. This distance represents the number of translations required to transition from one policy to another within $\hat{\mathcal{P}}$. It converges to zero after 8 iterations.

2) Policy search algorithms: We consider a general policy, where resource allocation can take any positive value below $R_{max}=10$. We plot in figure 5 the evolution of the cost

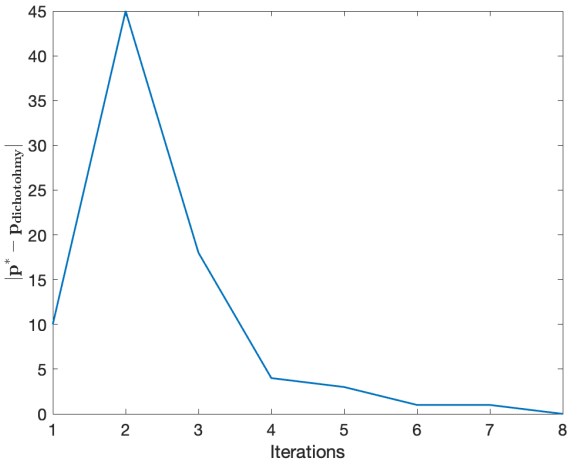

Fig. 4: Convergence of the Dichotomy algorithm to the optimal policy $\mathbf{p}^*$ ($U = 30$, $\delta = 4$).

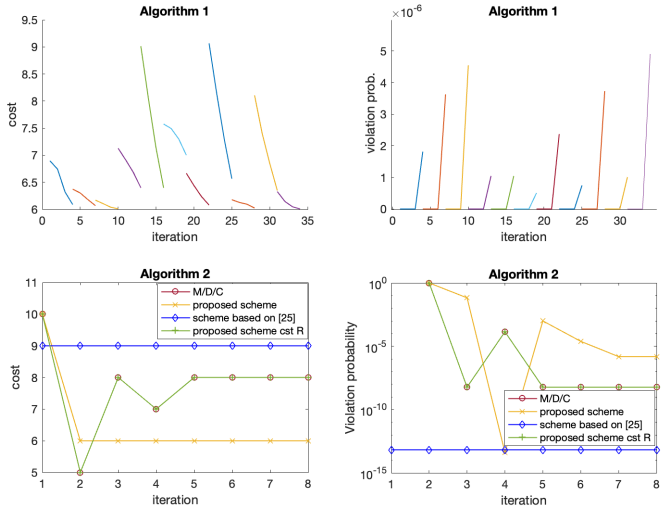

Fig. 5: Cost evolution for the policy search using algorithms 1 and 2, and the schemes based on the state of the art.

and violation probability when following algorithms 1 and 2 for our dynamic policy. For algorithm 2, we also compare the dynamic algorithm with the static one as well as the state of the art algorithm, based on the model of [25], where the performance of URLLC is based on an M/M/1 model, and also we consider modeling the performance by an M/D/c model. In the latter models (M/M/1 and M/D/c), the policy is not dynamic (does not depend on the queue length), but is based on a dimensioning of resources so that the performance is equal to the target. For the M/D/c model we make use of the Dichotomy algorithm for calculating the optimal c, however for the M/M/1 model, the violation probability in [25], for an average service time of $\frac{1}{R}$ packets per slot is given by:

$$\bar{o}(R) = e^{-(R-\bar{a})\delta} \tag{33}$$

where $\bar{a}$ is the average packet arrival rate. This leads to the following reservation for a target delay violation of ϵ:

$$\bar{R} = \frac{1}{\delta} \ln \frac{1}{\epsilon} + \bar{a} \tag{34}$$

Let us first compare algorithms 1 and 2 for the dynamic policy (proposed scheme) with homogeneous MCS setup (each packet consumes 1 resource unit, defined as the amount of PRBs for carrying one packet with the common MCS).

When using algorithm 1, we start by a "large" policy corresponding to a very low violation and reduce it in several iterations as detailed in the algorithm until reaching the target violation. We repeat this process at least $N_{min} = 10$ times, and continue until the violation is within 1% of the minimal observed performance. Each portion of the upper side of figure 5 corresponds to a "trial", starting from a random large policy to the policy that yields the target violation. 34 iterations, each corresponding to a policy evaluation, are needed for this example. Note that if we seek an accuracy of 0.1%, up to 100 iterations are needed. However when using the Dichotomy algorithm 2, the algorithm converges in 8 iterations to obtain the optimal dynamic policy in $\hat{\mathcal{P}}$. We illustrate the associated policies in figure 6. For algorithm 1, we illustrate the initial random policy corresponding to the trial that leads to the lowest cost, along with the final policy. For algorithm 2, we illustrate the initial largest policy ($\sup_{\mathcal{P}}(\mathbf{p})$ with a constant maximal resource allocation) and the final policy, obtained by translation and corresponding to smooth increase of the resource allocation starting from a threshold.

We now turn to the comparison of algorithm 2 for the cases of dynamic versus non-dynamic (proposed scheme with constant reservation R) versus M/D/c versus state of the art policies, as illustrated in figure 5 below. We observe that the dynamic, non-dynamic and M/D/C policies outperform the one based on the model in [25] as they reduce the cost. Indeed, the latter leads to an over-reservation and thus to an unnecessarily low delay violation. We have that both the M/D/c and non-dynamic policies exhibit the same behavior. However, the dynamic policy outperforms both of them as its cost is lower.

We now consider in figure 7 the case of adapted MCS, where the MCS distribution and the corresponding resource consumption distribution is issued from the MCS and traffic analytics module of the simulator. We compare our proposal, based on the developed analytical model, with the one obtained using an M/D/c model. However, the M/D/c model could not be directly applied, as it supposes the existence of c homogeneous servers, corresponding to equal packet sizes, while the adapted MCS case corresponds to heterogeneous packet sizes. We then consider three flavours for M/D/c, with three different approximations of the server capacity: the first considers the distribution's expected value, i.e. deals with the system as if all packets use the average MCS, the second involves the 95-th percentile MCS, and the third supposes that the worst-case MCS is used. Our proposed scheme ensures the target outage, while the other model either over-dimension (95-th percentile and worst case) or violate the target outage

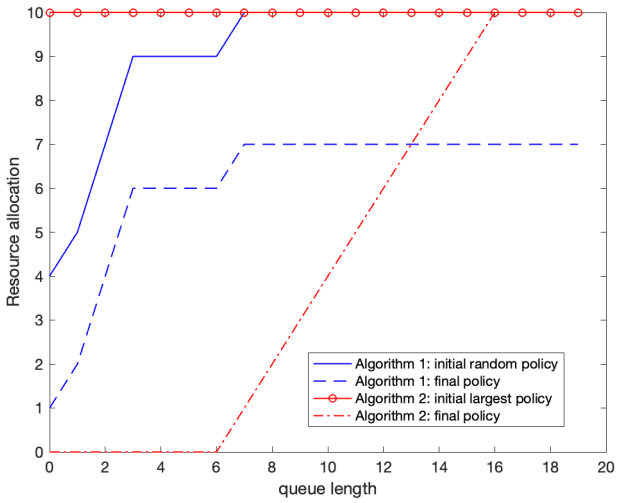

Fig. 6: Evolution of the policy for algorithms 1 and 2.

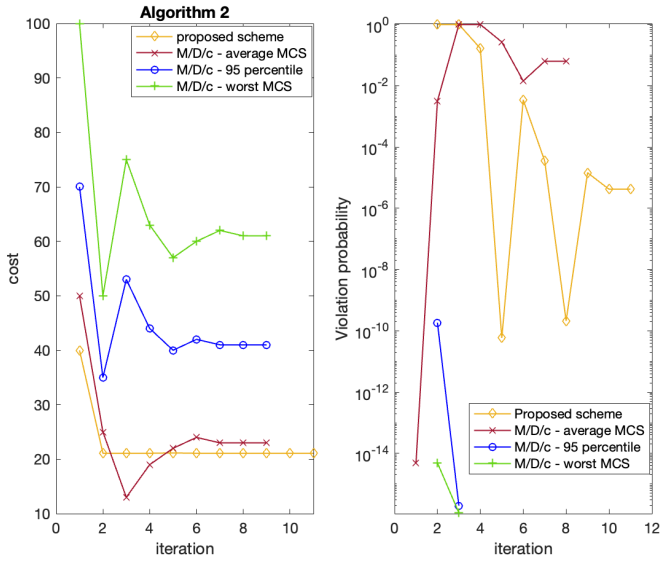

Fig. 7: Cost evolution for algorithm 2 applied to our model and the M/D/c model with heterogeneous MCS setup.

(average MCS model). The latter case (average MCS) may be surprising as, in the same time, the cost is slightly higher than our proposal and the outage is violated. Indeed, the policy with the M/D/c assumption is static (constant resource reservation independent of queue state), while it is dynamic with our proposal, leading to a lower reservation most of the time in our case.

D. Implementing the optimal online controller

The previous analysis showed that a search algorithm on threshold-like policies is efficient. However, this corresponds to the offline strategy where a policy is determined using the analytical model. We now consider the online setting, where the policy optimization problem is modeled as a stochastic bandit, and, when the policies are restricted to the set $\hat{\mathcal{P}}$, stochastic polychotomy can be used for selecting the best arm (policy), as it has been shown in section IV-D. In this algorithm, when an arm is selected to be evaluated, the system is run for some time T_e, long enough for observing the rare event that is the delay violation. If the policy is large, i.e. the violation probability is very low, T_e could be for tens of seconds. Otherwise, the violation probability is high and the policy is quickly adjusted (after a minimal time of $1000TTI = 0.25$ sec).

We allow also for the number of users to fluctuate. In this case, we need to timely update the upper and lower limits of the interval of the Dichotomy algorithm (section IV-D): if the number of users increases, the upper limit of the actual interval may not be large enough to keep the delay violation probability within the 10^{-5} bound, and if the number of users decreases, the lower limit might be too high for the new traffic.

Specifically,

1) if the number of users increases and/or the radio conditions degrade, the system detects it within 1 second approximately (as will be described in the implementation proposal in the next subsection). During this time, if the delay violation probability is smaller than the 10^{-5} bound, the dichotomy algorithm stays as is (the upper limit becomes equal to the current policy). If the delay violation exceeds the bound, the system adjusts the lower limit of the Dichotomy algorithm to the current policy and sets the upper limit to the maximal policy in $\hat{\mathcal{P}}$.
2) if the number of users decreases and /or the radio conditions improve, we look for the new lower limit of the Dichotomy algorithm by decreasing the actual policy, say by k. If the decrease by k results in a high violation probability, this policy becomes the lower limit, otherwise we continue to decrease by k until violating the bound. The upper limit stays as before and dichotomy is re-applied. The value of k might take small values, especially if the operator is risk avert, but this might however take a long time to converge. If the operator wishes to accelerate the convergence, it could take larger k steps, with the risk of violating the delay bound to a large extent. It is hence a trade-off between efficiency versus conservatism, depending on the level of risk aversion of the operator.

In Figure 8, we study the evolution of the violation and cost with the sequential policy trials under dynamic conditions, where the number of users varies over time. Starting with $U = 38$, transitioning to $U = 43$, and subsequently to $U = 33$, this investigation evaluates the efficiency and adaptability of our algorithm in real-world scenarios. We have chosen to implement abrupt changes in the traffic so that the performance of the algorithm in extreme conditions is tested. From $t = 0$ to $t \sim 250$, the system is empty. Subsequently, as the number of users increases, we observe that we test a bunch of policies in a short time due to their resulting high violation probabilities. The algorithm swiftly adapts to these conditions.

At $t \sim 750$, as the number of users increases, so does the violation probability. However, the algorithm quickly adjusts within a few seconds, by changing the upper and lower bounds of the dichotomy interval, and resumes operation. Prior to convergence for the final user count, at $t \sim 1250$, there is a sudden decrease in the number of users. The algorithm continues testing the policy for an extended period before changing. At this point, we observe that the violation probability is zero. Consequently, we initiate a 5-step decrementation ($k = 5$) process for the policy in hand, repeated three times until we reach a policy with a violation probability exceeding 10^{-5} that we test for a short time (until a large violation is detected), enabling the algorithm to establish new upper and lower limits and resume operation until convergence. The algorithm maintains consistent performance despite varying user counts, highlighting its robustness in dynamic environments.

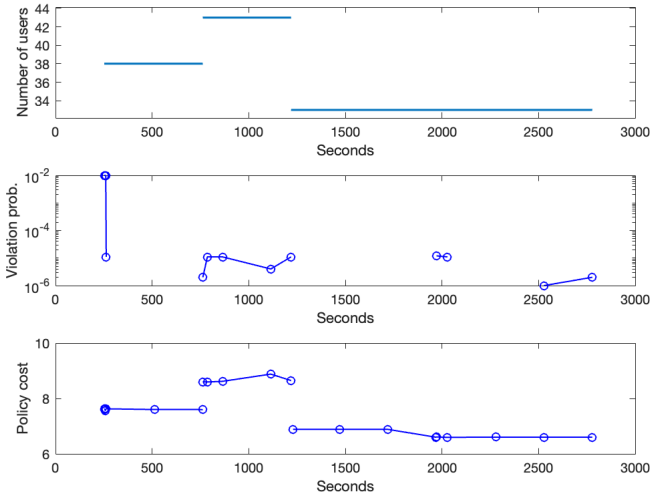

Fig. 8: Online policy optimization navigating through varying numbers of users

E. Illustrating the impact of reservation delay

In this section, we study the impact of reserving the resources at the slot boundaries instead of mini-slot boundaries on both the cost and violation.

We illustrate in figure 9 the impact of increasing τ (the number of mini-slots during which the system cannot be interrupted) on the delay violation. We derive the optimal policies using the Dichotomy algorithm when $\tau = 1$ (case with no delay in the reservation), and then apply these policies without modification to the system when $\tau = 2,\ 3$ and 4. The URLLC delay budget is fixed and equal to $\delta = 4$ mini-slots. The figure shows that the violation increases with the increase of τ, as the policy, derived under the assumption that the system is fully flexible, i.e., at each mini-slot, is not reactive enough when τ increases. Observing that the optimal policy for a fully flexible system does not perform well for a delayed system, we now use algorithm 2 to generate optimal policies

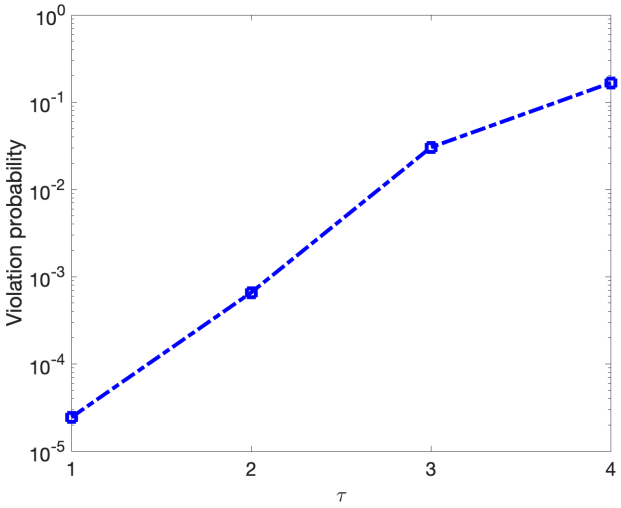

Fig. 9: Violation probability variation with reservation on slot boundary when increasing τ while keeping the same policy

for all values of τ. These policies will result in roughly the same violation probability of 10^{-5}, and so it is reasonable to compare their costs. We plot in figure 10 (left) the cost with respect to τ, one can see that the cost has increased by 22% as τ becomes equal to 4. The rank of these policies in the set $\hat{\mathcal{P}}$ is illustrated in the right plot. Here the optimal policy when $\tau = 1$ is a moderate policy that reserves $R_{min} = 0$ until the queue length exceeds 4, whereas the optimal policy when $\tau = 4$ is an aggressive policy that starts by reserving 5 PRBs even for an empty queue (i.e., $B(t) = 0$). This is due to the fact that these policies account proactively for the packets that will arrive in the next mini-slots within the slot. This has been done in the case where users have different arrival rates that are uniformly distributed between 0.1 and 0.4.

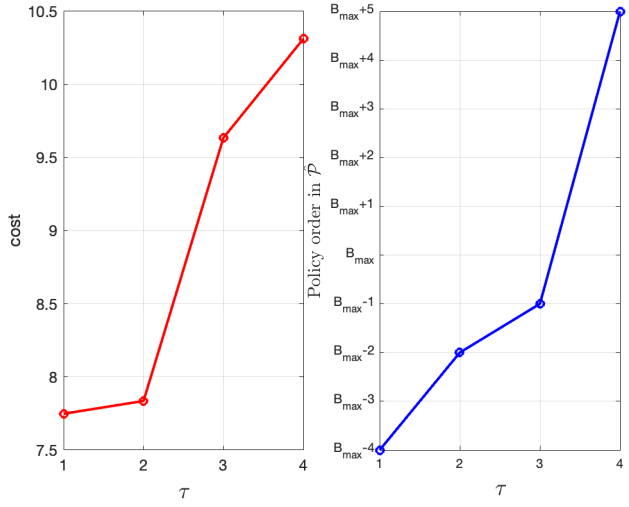

Fig. 10: The evolution of cost and optimal policy order in terms of τ ($U = 30$)

F. Implementation in the 5G and Beyond architecture

We propose an implementation within the Open Radio Access Network (O-RAN) architecture, specifically within the RAN Intelligent Controller (RIC), as illustrated in Figure 11. Two distinct modules are proposed as follows. The MCS and traffic analytics module (rApp), within the Non Real-Time (RT) RIC, operates on a time unit on the order of 1 second. It is responsible for constructing a per-gNodeB MCS distribution and the associated traffic intensity based on direct measurements from the gNodeBs, including the generated packets and their associated Channel Quality Indicator (CQI). It then sends these distributions to the Near RT RIC. The Non RT RIC also detects changes in the number of users in the cell and/or their radio conditions and notifies the Near RT RIC. The latter, which operates within a time range between 10 milliseconds and 1 second, implements two modules. The first is the policy optimization module, implemented as an xApp, that selects the resource allocation policy, and the other one, the QoS monitoring xApp, that observes the packet delays and calculates the violation probability. Moreover, the policy optimization module uses the violation probability and the traffic and radio condition distributions as input and implements the proposed algorithms for selecting the per-gNodeB resource allocation policy. This scheme has a very low time complexity $\mathcal{O}(|\mathcal{B}|)$, as shown in Note 1. These per-gNodeB policies, i.e., the amount of URLLC resources associated with each queue state, are provided to the schedulers located in the O-RAN Distributed Unit (O-DU), which is a logical node hosting Radio Link Control (RLC)/Medium Access Control (MAC). In other words, it is responsible for applying the policies in real-time.

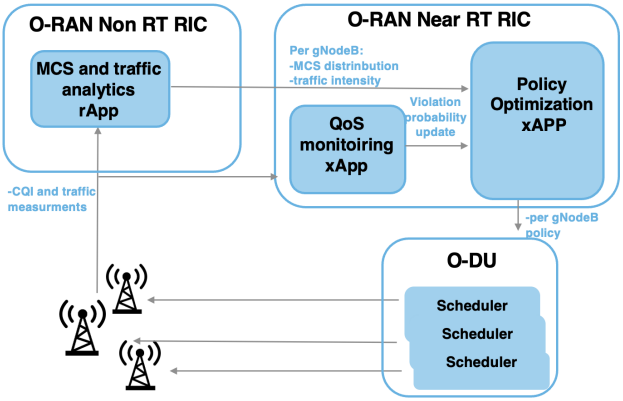

Fig. 11: Architecture proposal.

VI. Conclusion

We focused in this paper on resource allocation for latency-critical traffic in 5G networks. We considered devices that transmit packets with stringent delay and reliability constraints and formulated the equations describing the evolution of the number of packets waiting in the system. We derived the delay violation probability, i.e. the probability that the delay exceeds a maximal threshold and used it in a general resource allocation setting where the system adapts dynamically the reserved URLLC resources to the queue state. We showed that, by exploiting some structure of the policy, we can devise algorithms with quick convergence, especially in online optimization where we formulated the problem as a stochastic unimodal bandit and proposed polychotomy to solve it. Our numerical results show that our policy optimization algorithms allow reaching efficient solutions in practical 5G scenarios.

References

[1] *System Architecture for the 5G System*, 3GPP, TS 23.501, Dec. 2017, version 15.0.0 Release 15.

[2] M. Bennis, M. Debbah, and H. V. Poor, "Ultra reliable and low-latency wireless communication: Tail, risk, and scale," *Proceedings of the IEEE*, vol. 106, no. 10, pp. 1834–1853, 2018.

[3] K. I. Pedersen, G. Pocovi, J. Steiner, and S. R. Khosravirad, "Punctured scheduling for critical low latency data on a shared channel with mobile broadband," in *IEEE VTC-Fall*, 2017.

[4] Z. Li, M. A. Uusitalo, H. Shariatmadari, and B. Singh, "5g urllc: Design challenges and system concepts," in *2018 15th international symposium on wireless communication systems (ISWCS)*. IEEE, 2018, pp. 1–6.

[5] B. Singh, O. Tirkkonen, Z. Li, and M. A. Uusitalo, "Contention-based access for ultra-reliable low latency uplink transmissions," *IEEE Wireless Communications Letters*, vol. 7, no. 2, pp. 182–185, April 2018.

[6] Z. Zhou, R. Ratasuk, N. Mangalvedhe, and A. Ghosh, "Resource allocation for uplink grant-free ultra-reliable and low latency communications," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*. IEEE, 2018, pp. 1–5.

[7] H. Ko, J. Lee, and S. Pack, "Priority-based dynamic resource allocation scheme in network slicing," in *2021 International Conference on Information Networking (ICOIN)*, 2021, pp. 62–64.

[8] N. Ben Khalifa, V. Angilella, M. Assaad, and M. Debbah, "Low-complexity channel allocation scheme for urllc traffic," *IEEE Transactions on Communications*, vol. 69, no. 1, pp. 194–206, 2021.

[9] C. Sun and C. Yang, "Learning to optimize with unsupervised learning: Training deep neural networks for urllc," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2019, pp. 1–7.

[10] A. Chagdali, S. E. Elayoubi, A. M. Masucci, and A. Simonian, "Performance of urllc traffic scheduling policies with redundancy," in *2020 32nd International Teletraffic Congress (ITC 32)*.

[11] H. Jang, J. Kim, W. Yoo, and J.-M. Chung, "Urllc mode optimal resource allocation to support harq in 5g wireless networks," *IEEE Access*, vol. 8, pp. 126 797–126 804, 2020.

[12] A. Anand and G. de Veciana, "Resource allocation and harq optimization for urllc traffic in 5g wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 11, pp. 2411–2421, 2018.

[13] C.-P. Li, J. Jiang, W. Chen, T. Ji, and J. Smee, "5g ultra-reliable and low-latency systems design," in *2017 European Conference on Networks and Communications (EuCNC)*, 2017, pp. 1–5.

[14] P. Schulz, L. Ong, B. Abdullah, M. Simsek, and G. Fettweis, "End-to-end latency distribution in future mobile communication networks," in *WSA 2020; 24th International ITG Workshop on Smart Antennas*, 2020.

[15] B. Shi, F.-C. Zheng, C. She, J. Luo, and A. G. Burr, "Risk-resistant resource allocation for embb and urllc coexistence under m/g/1 queueing model," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 6, pp. 6279–6290, 2022.

[16] S. O. Oladejo and O. E. Falowo, "Latency-aware dynamic resource allocation scheme for multi-tier 5g network: A network slicing-multitenancy scenario," *IEEE Access*, vol. 8, pp. 74 834–74 852, 2020.

[17] A.-M. Mohammed, M. A. Arfaoui, M. Elhattab, C. Assi, and A. Ghrayeb, "Joint resource allocation and phase shift optimization for ris-aided embb/urllc traffic multiplexing," *arXiv preprint arXiv:2108.02346*, 2021.

[18] B. Liu, Z. Zhang, P. Zhu, J. Li, and D. Wang, "Resource allocation in distributed massive mimo systems for slicing embb and urllc services," in *2021 13th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2021, pp. 1–5.

[19] J. Östman, R. Devassy, G. Durisi, and E. Uysal, "Peak-age violation guarantees for the transmission of short packets over fading channels," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 109–114.
[20] Y.-P. Hsu, E. Modiano, and L. Duan, "Scheduling algorithms for minimizing age of information in wireless broadcast networks with random arrivals," *IEEE Transactions on Mobile Computing*, vol. 19, no. 12, pp. 2903–2915, 2019.
[21] H. Lin and X. Yang, "Dichotomy algorithm for solving weighted min-max programming problem with addition-min fuzzy relation inequalities constraint," *Computers & Industrial Engineering*, vol. 146.
[22] U. Herkenrath, "The n-armed bandit with unimodal structure," *Metrika*, vol. 30, no. 1, pp. 195–210, 1983.
[23] R. Combes, S. Magureanu, and A. Proutiere, "Minimal exploration in structured stochastic bandits," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
[24] R. Combes, A. Proutière, and A. Fauquette, "Unimodal bandits with continuous arms: Order-optimal regret without smoothness," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 4, no. 1, pp. 1–28, 2020.
[25] T. Ma, Y. Zhang, F. Wang, D. Wang, and D. Guo, "Slicing resource allocation for embb and urllc in 5g ran," *Wireless Communications and Mobile Computing*, vol. 2020, pp. 1–11, 2020.

VII. Author's Biographies

Mohammed Abdullah is currently pursuing a Ph.D. at Telecom SudParis, Institut Polytechnique de Paris, on optimal resource allocation policies for IoT in 5G and 6G networks. Prior to this, he completed his Master degree in Optimization at Paris Saclay University in 2022. He holds Bachelor and Master degrees in Mathematics from the Lebanese University, which he completed in 2020 and 2021, respectively. His expertise is mainly on mathematics and wireless communications.

Salah Eddine Elayoubi received the M.S. degree in telecommunications from the National Polytechnic Institute of Toulouse, Toulouse, France, in 2001, and the Ph.D. and Habilitation degrees in computer science from the University of Paris VI, Paris, France, in 2004 and 2009, respectively. From 2004 to 2013, he was with Orange Labs in France. Since January 2018, he has been a Full Professor with CentraleSupélec, France. He holds the Sustainable 6G chair funded by Orange. His research interests include modeling, performance evaluation and optimization of mobile networks.

Tijani Chahed received the B.S. and M.S. degrees in electrical and electronics engineering from Bilkent University, Turkey, and the Ph.D. and Habilitation a Diriger des Recherches (HDR) degrees in computer science from the University of Versailles and the University of Paris 6, France, respectively. He is currently a Professor with Telecom SudParis, an engineering school that belongs to both Institut Polytechnique de Paris and Institut Mines-Telecom, France. His research interests include resource allocation and quality of service, notably in wireless networks.

APPENDIX

A. Computing the performance numerically

In this section, we use a tree exploration method to calculate the performance numerically. To calculate the violation probability, we should study the evolution of the cumulative reserved resources in the next $\delta - 1$ mini-slots, whereas this metric is a random variable that depends on the queue length in the upcoming mini-slots.

Algorithm 3 describes a numerical method for calculating this metric. Assume we begin at state b, then define 2 zero matrices V_{old} and V_{new} with sizes $B_{max} \times S_{max}$, then let V_{old} initially hold only one nonzero element at index $(b, 1)$ which is $\mathbf{q}_b$, and passing by block (4-7), we fill out V_{new}, which describes the possible evolutions of the system (queue length and cumulative number of resources for one step (mini-slot)). In other words, the element (j, s) in V_{new} represents the probability of having s resources reserved and being at state j in the next mini-slot, after that in line (8) we update V_{old} to be equal to V_{new} because it describes the current information of the system, and repeating the same methodology for $\delta - 1$ we get V_{new} as the system possible states (j, s) after $\delta - 1$ mini-slots. So we simply add all the elements of the s column of V_{new} for all $s \leq b - 1$. Then we repeat the process for all states b, summing the results to get the delay violation probability.

Algorithm 3 Violation probability calculation

Ensure: $V_{old}, V_{new} \in \mathcal{R}^{B_{max} \times S_{max}}$

1: **for** $b \leq B_{max}$ **do**
2: $V_{old}[b, 1] = \mathbf{q}_b$
3: **for** $t = 1 : \delta - 1$ **do**
4: $V_{new} = 0^{B_{max} \times S_{max}}$
5: **for** all $j \leq B_{max}$ and $s \leq S_{max}$ **do**
6: $V_{new}[:, s + R(j, \mathbf{p})] = V_{new}[:, s + R(j, \mathbf{p})] + V_{old}[j, s]\mathbf{Q}[j, :]^T$
7: **end for**
8: $V_{old} \leftarrow V_{new}$
9: **end for**
10: $V(\mathbf{p}) = V(\mathbf{p}) + \sum_{j \leq B_{max}} \sum_{s \leq b} V_{new}[j, s]$
11: **end for**
=0

B. Transition matrix in the delayed case

We show here how the transition matrix $\mathbf{D}$ for the delayed system can be expressed in terms of the transition matrix $\mathbf{Q}$ of the fully flexible system. Let Q^R be the transition matrix with no delays while reserving R resource blocks with $Q^R_{j,:}$ and $Q^R_{:,j}$ to be the j^{th} raw and the j^{th} column of matrix Q respectively, thus the three different sub-matrices can be written as follows:

- In the first matrix $\mathbf{A}_{(\tau-1)0}$ we will consider only the following indices $(i, j, \ell = \tau - 1) \rightarrow (b, b, \ell = 0)$ for all $i, j, b \in [0, B_{max}]$, then $\mathbf{A}_{(\tau-1)0} =:$

$$\begin{bmatrix} \mathbf{Q}^{R_1}_{:,1} & \overbrace{0 & \cdots & 0}^{B_{max}} & \mathbf{Q}^{R_1}_{:,2} & \cdots & \mathbf{Q}^{R_1}_{:,B_{max}} \\ \mathbf{Q}^{R_2}_{:,1} & 0 & \cdots & 0 & \mathbf{Q}^{R_2}_{:,2} & \cdots & \mathbf{Q}^{R_2}_{:,B_{max}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & & \vdots & \vdots & \vdots & \cdots & \vdots \\ \mathbf{Q}^{R_{B_{max}}}_{:,1} & 0 & \cdots & 0 & \mathbf{Q}^{R_{B_{max}}}_{:,2} & \cdots & \mathbf{Q}^{R_{B_{max}}}_{:,B_{max}} \end{bmatrix}$$

- For a_{01}, only the indices $(j, j, \ell = 0) \rightarrow (j, b, \ell = 1)$ where $j, b \in [0, \ldots, B_{max}]$ are non-zeros, by the fact that at $\ell = 0$, B_1 and B_2 should be equal, then $\mathbf{A}_{0,1} =:$

$$\begin{matrix} \\ \\ B_{max} \\ \\ \\ \\ B_{max} \\ \\ \\ \\ \end{matrix} \begin{bmatrix} \mathbf{Q}^{R_1}_{1,:} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \mathbf{Q}^{R_2}_{2,:} & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \mathbf{Q}^{R_{B_{max}}}_{B_{max},:} \end{bmatrix}$$

- All other matrices, $\mathbf{A}_{\ell,\ell+1}$ with $\ell \in \{1, \ldots, \tau - 2\}$, will have $(k, j, \ell) \rightarrow (k, b, \ell + 1)$ as non-zeros indices for all $k, j, b \in [0, B_{max}]$ then $\mathbf{A}_{\ell,\ell+1} =$

$$\begin{bmatrix} \mathbf{Q}^{R_1} & 0 & 0 & \ldots & 0 \\ 0 & \mathbf{Q}^{R_2} & 0 & \ldots & \vdots \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \ldots & 0 & \mathbf{Q}^{R_{Bmax}} \end{bmatrix}$$

C. Cost comparison of threshold policies

We assumed that minimizing policies (in terms of reservations for each state) would result in cost savings, and we now intend to illustrate this statement. Then we aim to get $C(\mathbf{p}) \leq C(\mathbf{g})$ when using two policies $\mathbf{p} \leq \mathbf{g} \in \hat{\mathcal{P}}$, for which we run multiple simulations, the results of which are shown in figure 12, using descending policies starting with the maximal policy, and we can see that the cost decreases when the policy decreases. We should note here that after a certain number of translations applied to the maximal policy, the cost will be constant as the number of translations is less than B_{max}, and this in fact is due to the dependency of the distribution of the queue length on the policies, resulting in a type of translation of the distribution so that it is concentrated in the larger states, which leads to approximately the same cost. This shows that the cost does not increase as policies decrease, thus equation (24) is valid.

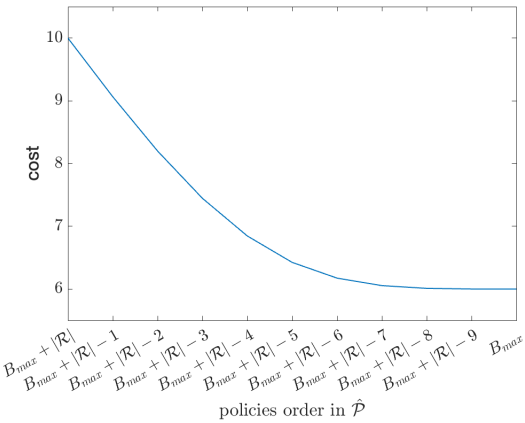

Fig. 12: The variation of cost in the extended threshold set $\hat{\mathcal{P}}$