Optimal Latency-Oriented Coding and Scheduling in Parallel Queuing Systems

Andrea Bedin, Federico Chiariotti, *Member, IEEE,* Stepan Kucera, *Senior Member, IEEE,* and Andrea Zanella, *Senior Member, IEEE*

***Abstract*—The evolution of 5G and Beyond networks has enabled new applications with stringent end-to-end latency requirements, but providing reliable low-latency service with high throughput over public wireless networks is still a significant challenge. One of the possible ways to solve this is to exploit path diversity, encoding the information flow over multiple streams across parallel links. The challenge presented by this approach is the design of joint coding and scheduling algorithms that adapt to the state of links to take full advantage of path diversity. In this paper, we address this problem for a synchronous traffic source that generates data blocks at regular time intervals (e.g., a video with constant frame rate) and needs to deliver each block within a predetermined deadline. We first develop a closed-form performance analysis in the simple case of two parallel servers without any buffering and single-packet blocks, and propose a model for the general problem based on a Markov Decision Process (MDP). We apply policy iteration to obtain the coding and scheduling policy that maximizes the fraction of source blocks delivered within the deadline: our simulations show the drawbacks of different commonly applied heuristic solutions, drawing general design insights on the optimal policy.**

I. Introduction

Over the past few years, the evolution of 5G and Beyond networks has opened new possibilities for interactive applications, such as mobile Virtual Reality (VR) or remote control of industry machinery, which were previously constrained to wired scenarios. Besides a fairly large transmission capacity, these applications have strict latency constraints which may be difficult to meet (e.g., the interactivity requirement for VR or video conferencing) over wireless links because of the volatile nature of the medium, with fluctuating capacity and a relatively high packet error probability. The use of multiple wireless interfaces, often over different technologies, is a way to provide the required Quality of Service (QoS) even when individual links are unreliable. Indeed, encoding data blocks and sending redundant information over multiple paths can protect the transmission from failures and delays on individual paths. A possible example of reliable multipath communication is depicted in Fig. 1, in which a VR user receives a stream of frames from a remote server with strict real-time requirements. The primary 5G link might not be sufficient, particularly in underserved rural areas, and 4G or private WiFi can be used to provide additional reliability. The end-to-end connections between the client and server can be affected by several factors, such as propagation and mobility issues or cross traffic. The objective of this paper is to provide a theoretical model of such a scenario, abstracting each link to a queuing model to find the optimal schedule to reliably transmit the data with bounded latency. The existing Multipath TCP (MPTCP) standard is woefully inadequate for this reason: improper scheduling can cause significant delays due to the head of line blocking problem, and retransmissions can compound the problem, often providing worse QoS than even a single-path flow on the best available path [1]. The practical applications of our work are in real-time solutions that avoid retransmissions, relying on packet-level coding to protect the transmission.

While efficient ways to exploit multiple paths to reliably transmit Ultra-Reliable Low Latency Communications (URLLC) traffic [2] exist, they are limited to applications with very low throughput and very tight delay constraints, while applications with looser real-time constraints, but a far higher data rate, mostly operate on a best-effort basis. In our work, we focus on this type of sources, considering a heavy flow with periodic block arrivals and a tight latency constraint, such as VR streams or sensor data flows generated by, e.g., autonomous vehicles [3] or high-throughput industrial closed-loop control data [4]. Recently, there has been an effort to provide reliable end-to-end service using redundant coding over multiple paths: in particular, the Decoupled Multipath Scheduler (DEMS) MPTCP scheduler [5] introduced the notion of data blocks, which must be delivered as a whole, and exploited packet-level coding across different paths to ensure a faster, more reliable delivery of the data, recovering from failures on any single path, and our previous work [6] introduces a dynamic coding rate adaptation to the available capacity. However, even state-of-the-art protocols still use *ad hoc* heuristic mechanisms to balance the tradeoff between maintaining a high reliability (which would require more redundancy) and avoiding self-queuing delays for future blocks (which inherently limits the possible redundancy that can be sent over the available paths). In this context, the tradeoff is extremely complex: adding too much redundancy on the wrong path can cause congestion (if the capacity of the path is exceeded), while adding too little can reduce reliability and make the transmission less robust to errors or capacity fluctuations. The interaction between queues in a multipath system is even more complex, and optimizing decisions even

Andrea Bedin (corresponding author, andrea.bedin.2@phd.unipd.it) and Andrea Zanella (andrea.zanella@unipd.it) are with the Dept. of Information Engineering, University of Padova, Italy. Federico Chiariotti (fchi@es.aau.dk) is with the Dept. of Electronic Systems, Aalborg University, Denmark. Stepan Kucera is with Nokia Bell Labs, Munich, Germany (stepan.kucera@nokia.com). Andrea Bedin is also with Nokia Bell Labs, Espoo, Finland. This work has received funding from the European Union's EU Framework Programme for Research and Innovation Horizon 2020 under Grant Agreement No 861222.

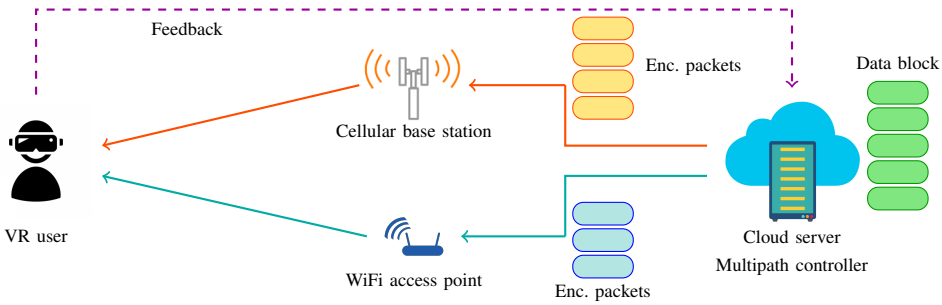


Fig. 1: Schematic of an encoded multipath transmission: the 5 green packets are encoded into 7 and divided between the cellular link (4 yellow packets) and the WiFi access point (3 blue packets). The user will receive the data block as soon as any 5 packets are transmitted successfully.

in a relatively idealized scenario is a formidable problem. To the best of our knowledge, this is the first work in the literature to rigorously model this tradeoff and optimize coding and scheduling jointly, considering a simplified scenario that can provide general insights for more practical future schemes. Existing theoretical models [7] often use static scheduling policies and attempt to derive bounds on the latency [8], but a joint optimization of coding and scheduling has never been attempted.

In this paper, we model a multipath communication scenario as a fork-join Queuing System (QS), i.e., a system in which packets from the same flow are sent over multiple queues and gathered by the receiver. We then cast the queuing model into a Markov Decision Process (MDP) framework to capture the long-term effects of the controller's decisions. We solve the joint coding and scheduling MDP to derive the optimal policy, deciding *how much* redundancy to add to each data block and *how to split* the coded data among the available paths. In other words, the controller has a dual objective: firstly, it needs to determine the necessary amount of redundancy for the data block, and secondly, it needs to act as an optimal scheduler, distributing the encoded packets over a number of parallel queues in order to deliver the data within a fixed deadline with stochastic reliability guarantees, i.e., to maintain below a certain threshold the probability that the data block latency exceeds the deadline. We also consider the implications of packet loss and delayed feedback on the state of the queue, as well as time-varying queues that can model different wireless scenarios. In fact, one of the main potential drawbacks of redundancy is to build-up packet queues at temporarily slower links. This might trigger a "snowball" effect, as schedulers try to react to the increase in the delay due to the queued packets by adding even more redundancy, as observed practically in [6]. Therefore, the optimal policy must properly account for self-inflicted latency, striking a balance between immediate and long-term reward.

The main contribution of our work is a rigorous mathematical model of the parallel communication as a fork-join queuing problem [9], with a full derivation of the reliability and latency violation probability. We find the optimal solution to the problem explicitly in a simple case and by policy iteration in the more general scenario. While policy iteration is computationally heavy, it is provably optimal and analytically tractable, and practical solutions can use more flexible reinforcement learning methods. The optimality gap in our model can give important insights on the design of practical algorithm, and while the model's complexity is limited by the tractability of the equations, its features can be used to simulate a wide array of potential scenarios, including cases with delayed feedback or incorrect channel parameter estimation. Since, to the best of our knowledge, the literature is currently lacking theoretical tools to analyze these scenarios, our analysis can spur further development in coded multipath communications.

The rest of the paper is structured as follows: first, we examine the state of the art on parallel QSs while their practical applications are discussed in Sec. II. We then describe the generic system model for the considered system in Sec. III. Sec. IV defines the MDP formulation of the problem and presents its solution in the most general case. We also present the result for some notable reward functions, such as the overall expected amount of data delivered within the deadline. Then, Sec. V presents the derivation of an analytical policy for a simple example in which it is possible to do so. In Sec. VI, we show our simulation results, including an analysis of the optimal policy and its comparison against some practical heuristics taken from the literature, such as load balancing and max redundancy. Finally, Sec. VII concludes the paper.

II. Related Work

The stochastic characterization of parallel queues with multiple servers [10] has gradually become an active research subject [7], following the development of parallel computing and multipath networking. The problem of scheduling Poisson arrivals from multiple sources on multiple queues, minimizing the response time for each source, can be solved using classical nonlinear optimization [11]. Policies can be found even if the state of the queues is not directly observable, getting a maximum likelihood estimate from the known capacity distributions [12] or employing periodic policies [13].

As mentioned above, our work deals with the *fork-join* queuing model [9], where incoming tasks or blocks of data are divided among several parallel QSs with independent queues [14]. The first work to find the latency bounds for the transmission of a block of data over parallel queues with erasure codes was [8]. Another analysis focused on the combination of redundant and uncoded requests [15], while

a subsequent work analyzed the expected latency of different scheduling policies [16]. However, these works still consider simple static queuing policies and only study the one-step latency, neglecting the potential impact of the redundancy on future blocks scheduled on the same queues. Conversely, we examine the long-term effects of the coding and scheduling policy and determine the optimal policy. To the best of our knowledge, these are novel contributions to the theoretical queuing literature.

The scheduling problem over parallel queues is not just theoretical, but a very real issue for multipath transport protocols such as MPTCP, which can be heavily impacted by the head-of-line blocking problem [1]. The most basic MPTCP scheduler, currently used in the Linux implementation of the protocol, adopts the lowest RTT first (LowRTT) policy: packets are sent in the order in which they are written by the application, on the path with the lowest measured Round-Trip Time (RTT) among those with enough available space in their congestion window. Round robin [17] and loss-based [18] scheduling schemes have also been proposed, but fail when there is a strong imbalance between the subflows. These heuristics are often inefficient [19] and can lead to significant performance losses [20]. In fact, it might be convenient to send data on slower paths in advance, exploiting the difference in the path's RTTs to have packets arriving in the correct order to the receiver. This more complex scheduling requires to model each path's RTT and capacity in order to properly interleave packets among the parallel paths. Schedulers such as the Slide Together Multipath Scheduler (STMS) [21] and Delay Aware Packet Scheduling (DAPS) [22] are designed to do so, and have better performance than simpler heuristics in most situations. The Blocking Estimation (BLEST) [23] scheduler adds the awareness of the possibility of head-of-line blocking to this mechanism, explicitly trying to prevent it.

However, standard MPTCP is not a viable option for real-time traffic, particularly over wireless links [24], makes reliable low-latency service extremely hard to provide. Packet-level coding can help to avoid lengthy retransmissions: if a packet is lost on one path, it can be recovered from redundancy packets received on other paths. Several coding-based MPTCP schedulers have been proposed [25], [26], using different coding schemes. DEMS [5] is a first attempt to exploit parallel paths to deliver messages from block-based applications: in a two path setting, DEMS transmits data on one path starting from the beginning of the block, and on the other path starting from the end. The scheduler foresees an adaptive redundancy mechanism to improve delivery times in variable network conditions. The same concept was the basis of the High-reliability latency-bounded Overlay Protocol (HOP) [27], which gives explicit QoS guarantees by using a greedy policy. These two protocols aim at guaranteeing reliable low-latency communications on a block-by-block basis. For a more thorough survey of the state of the art on scheduling in multipath transport protocols, we refer the reader to [28].

The theoretical studies on fork-join queuing and the practical work on multipath scheduling have a fundamental limit: while the parallel queue scheduling problem requires foresighted strategies to avoid self-queuing delay and balance the tradeoff between immediate and future reliability, the existing literature focuses on one-step heuristics and policies, using *ad hoc* mechanisms to ensure stability. To the best of our knowledge, our work is the first to provide a rigorous model of the long-term tradeoff, providing the optimal policy even with delayed feedback and packet erasures. Moreover, our model combines most of the features studied in the literature, providing a complete approach to redundant multipath communication.

III. System Model

We consider a sender that periodically generates blocks of K packets of g bits every τ_g seconds. Each block has to be delivered within a deadline τ_d from its generation time. Vectors are denoted in bold, Probability Density Functions (PDFs) and Probability Mass Functions (PMFs) are denoted with lower-case letters, whose upper-case versions indicate the respective Cumulative Distribution Functions (CDFs). The assumption that the traffic source is periodic and has packets with constant size is required for tractability, but is also justified by a number of use cases. Recent technical documents from industrial associations [4], [29] and research papers [30], [31] describe the use of periodic messages of constant size in several industrial scenarios: a recent 3GPP specification [4] defines the *deterministic periodic communication* class for closed-loop control systems, specifically targeting reliable, low-latency transmission and distinguishing between wireless channel latency and end-to-end latency. While most of these applications are low-throughput, some cases (such as video transmission or high-dimensional sensing data) can run into the tradeoff studied in this work. Another use case in which Constant Bit Rate (CBR) flows play a crucial role is represented by Vehicle to Everything (V2X) communications and cooperative driving [32], as specified by the 3GPP [29]. Finally, while VR applications are mostly not CBR, some commercial applications use it to provide a smoother service [33].

The sender has M available connections to the receiver with independent bottleneck links, using different technologies. The objective of the controller is to encode the K packets of the generic i-th block into N_i encoded packets, then schedule them for transmission over the links. The proposed model and schemes are agnostic to the specific implementation of the packet-level code, as long as it allows a block to be decoded as soon as any set of K packets is received. Efficient implementations of systematic packet-level codes with this property, such as shortened-and-punctured Reed-Solomon codes [34], are available in the literature.

The schedule is decided once for the whole block, as the transmitter needs to encode the packets and send them. The notion of blocks of data that need to be delivered as a whole is natural to most throughput-intensive applications, so including it in the optimization is more effective than packet-by-packet scheduling, which considers individual packets as the scheduling units. In the following, we will use block generation times as a natural timestep for the optimization, as they correspond to scheduling decisions; however, the

communication is performed over continuous time, so packets and feedback may be delivered at any point in time.

The connections are modeled as parallel single-server QSs, denoted as $Q_1, \dots, Q_M$: this model is extremely common in the networking literature, and has historically been used to approximate end-to-end connections [35], [36], and more recently, in the optimization of Cloud systems [15], [16]. In general, modeling an end-to-end connection as the combination of a stochastic queuing system and a deterministic propagation delay is common when there is a significant bottleneck in the wireless access link [37], [38], which is the main application scenario we envision for multipath wireless communications. We can also consider the state of the queues and channels as Markov Chains (MCs) using the block generation times as a timestep, computing the transition probabilities that correspond to the possible sequences of packet deliveries between subsequent blocks. The transmitter then divides the N_i encoded packets belonging to block i over the M QSs, generating the scheduling vector $\mathbf{s}(i) = (s_1(i), \dots, s_M(i))$, where $s_m(i)$ denotes the number of packets scheduled for transmission on Q_m, with $\sum_{m=1}^{M} s_m(i) = N_i$. Furthermore, the vector $\mathbf{q}(i) = (q_1(i), \dots, q_M(i))$ indicates the number of packets in the M queues just before the generation time of the i-th block of packets. The service times in the QSs are then independent, while the arrival process depends on the scheduler, which takes into account the state of all QSs.

The block is considered to be delivered whenever at least K of the N_i transmitted packets reach the destination within the deadline, across any combination of QSs. We assume that packets sent on each channel m can be randomly and independently dropped with probability ε_m after service. Such events are named *erasures*, and are assumed to occur the m-th QS according to an independent Bernoulli process with parameter ε_m. We conservatively assume that erased packets occupy the servers as any other packet, but are discarded by the receiver. Feedback about each delivered or erased packet is assumed to be received by the sender with a constant delay τ_f. If $\tau_f > 0$, the sender does not know the state of the queues at time t, but only their past state at time $t - \tau_f$. As we will discuss in detail in Sec. IV-B, we can still find the optimal schedule given the available information on the system state, but the delay in the knowledge of the state makes the policy slightly more conservative. The case $\tau_f = 0$ is valid for local networks in which the sender is co-located with the wireless access point or connected directly to it, so that the delay of the return channel (which is assumed to be lightly loaded) can be assumed to be negligible with respect to that of the forward channel. A schematic of the system is depicted in Fig. 2, and the main notation used in the paper is listed in Table I.

Note that the actual size of each batch depends on the controller's decisions; even though we assume that the blocks are generated according to a deterministic process, each QS is a $G/G/1/\ell$ system according to Kendall's notation, where ℓ is the (finite) capacity of the queue. The service time depends, in general, on the queuing state and on the underlying channel state. However, note that the problem is solvable only if the average service rate is higher than the uncoded arrival rate. As multipath communications are usually over different

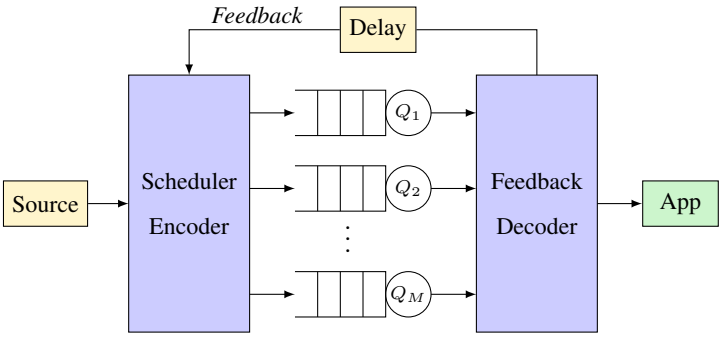


Fig. 2: Basic schematic of a fork-join QS.

technologies or orthogonal frequency bands, we assume there are no direct interactions between the paths, and connections can be modeled as independent QSs. We can consider a general model in which connections are time-varying, and their bandwidth depends on a parameter c_m, which changes according to a MC with state space $\mathcal{C}_m = \{1, \dots, C_m\}$ and transition matrix $\boldsymbol{\theta}_m$ (i.e., there is a finite set of possible values for c_m). A classic example is the binary Gilbert-Elliott model [39]. We denote the combination of $c_m(i)$ and $q_m(i)$ as seen by the i-th block as $\psi_m(i) = (c_m(i), q_m(i))$.

For the generic m-th QS, the service time of the n-th packet transmitted on that QS after the generation of block i is then a random variable $\mathcal{Y}_{c_m(i)}(n)$, which is conditioned on $c_m(i)$. In this section, we do not make any assumptions about the distribution of the service time, so as to maintain full generality. The M QSs are mutually independent, so that $\psi_m(i)$ and $\mathcal{Y}_{c_m(i)}(n)$ are independent between the QSs. We also define the state vector $\boldsymbol{\psi}(i) = (\psi_1(i), ..., \psi_m(i))$, containing the states of the M QSs for block i.

Finally, we define $\mu_m(i) = \frac{1}{\mathbb{E}\left[\mathcal{Y}_{c_m(i)}\right]}$ as the average service rate of Q_m for block i: by the fundamental renewal theorem, the service rate can be computed as the number of packets in a renewal interval, i.e., 1 (as the service of 1 packet corresponds to a renewal) divided by the average duration $\mathbb{E}\left[\mathcal{Y}_{c_m(i)}\right]$. We can extend this to get $\mu(i) = \sum_{m=0}^{M} \mu_m(i)$, the aggregate service rate of all the QSs. We make the conservative approximation (which is exact in the $G/M/1/\ell$ case) that, upon a new block arrival to a QS, the residual service time of the packet under service is distributed as a general service time (i.e., we neglect the time already spent in service). As we show in the Appendix, the actual probability of the packet being delivered before a given time is always lower than the approximated value for light-tailed service time distributions, i.e., distributions with a monotonically non-decreasing hazard rate [40]. The resulting policy will be suboptimal in the actual system, as it will consider a slightly different state, but this only affects the first packet in the queue, and as such, has a limited effect on the system for larger blocks.

IV. MDP Formulation and Solution

We can now model the scheduling process as a finite MDP, whose decision instants correspond to the arrival of new

Symbol	Meaning	Symbol	Meaning	Symbol	Meaning
K	Size of a data block (in packets)	M	Number of QSs	$c_m(i)$	QS state
τ_d	Delivery deadline	τ_g	Inter-block generation time	$\boldsymbol{\Theta}_m$	QS state transition matrix
$q_m(i)$	Queue state	$\psi_m(i)$	Overall state for QS m seen by block i	$\mathcal{Y}_{c_m(i)}(n)$	n-th packet service time in state $c_m(i)$
μ_m	QS average service rate	ℓ	Queue length (in packets)	ε_m	Packet erasure probability
T	Transition matrix	$\mathcal{A}$	Action set	$\mathbf{s}$	Scheduling vector
τ_f	Feedback delay	$\mathcal{D}_m$	Delivered packets before the next block	$y_{\psi_m(i)}(n)$	Overall delay of the i-th packet
ρ	Block delivery probability	δ_m	PMF of the total number of delivered packets	ω_m	PMF of the number of delivered packets
$\mathcal{N}(\mathbf{s})$	Set of delivery vectors leading to decoding	Π	Scheduling policy	$\boldsymbol{\phi}$	Steady-state distribution
$r_{\boldsymbol{\psi}}$	Reward function	λ	Discount factor	$R_{\boldsymbol{\psi}}$	Long-term reward
Γ	CDF of the block delivery delay	$\tilde{\boldsymbol{\psi}}$	Delayed state		

TABLE I: Main notation used in the paper.

blocks. MDPs are defined by a state space, an action space, a matrix of transition probabilities, and a reward function. The state of the MDP contains all the information available to the controller when it makes a decision, while the actions are naturally the possible schedules that can be applied, and the reward is the success probability of current and future blocks. In our case, the state of the system for block i is $\boldsymbol{\psi}(i)$. Consequently, the state space is simply $\Psi = \left(\prod_{m=1}^{M} \mathcal{C}_m\right) \times \{0,\dots,\ell\}^M$. In the following, we first model the case in which feedback is instantaneous, i.e., $\tau_f = 0$, then extend the derivation to the general case.

The action space for the controller is also simple. Since the number of packets on each QS cannot exceed the queue capacity ℓ, the action space is simply $\mathcal{A} = \{0,\dots,\ell\}^M$, and each action is a possible vector $\mathbf{s}(i) \in \mathcal{A}$. It should be noted that any actions that entail transmitting fewer than K packets in total are always outperformed by dropping the block entirely (i.e., $\boldsymbol{s}(i) = \mathbf{0}$, also referred to as a *block drop*), and can then be removed from the action set.

The transition probabilities $T(\boldsymbol{\psi}, \boldsymbol{\psi}', \mathbf{s})$ can be computed from the statistics of the service time. The delay $y(n|\psi_m(i))$ for the delivery of the n-th scheduled packet on QS m is given by:

$$y(n|\psi_m(i)) = \sum_{j=1}^{q_m(i)+n} \mathcal{Y}_{c_m}(j). \tag{1}$$

The PMF of $y(n|\psi_m(i))$, which accounts for the time to serve the residual $q_m(i)$ packets of the previous blocks (if any), plus the time to serve the n packets scheduled on queue m for the current block, can be computed from the known statistics of the service time. In turn, the PMF $\delta_m(x|\tau, \psi_m, s_m)$ of the number of packets $\mathcal{D}_m$ delivered by time τ can be computed as:

$$\delta_m(x|\tau,\psi_m,s_m) = \begin{cases} P\left[y(1|\psi_m(i)) > \tau\right], & x = 0; \\ P[y(x|\psi_m(i)) \le \tau, \\ y(x+1|\psi_m(i)) > \tau], & 0 < x < q_m + s_m; \\ P\left[y(x|\psi_m(i)) \le \tau\right], & x = q_m + s_m, \end{cases} \tag{2}$$

and 0 in all other cases. We can easily derive the CDF of $\mathcal{D}_m$, $\Delta_m(x|\tau,\psi_m,s_m)$, as:

$$\Delta_m(x|\tau,\psi_m,s_m) = \sum_{k=0}^{x} \delta_m(k|\tau,\psi_m,s_m). \tag{3}$$

A. Reward Calculation

The reward function will depend on the probability that the block of data is decoded within its deadline τ_d. In this case, we also have to consider packet erasures. In order to deliver new packets, a QS must first flush the queue, i.e., deliver the $q_m(i)$ packets already in flight (irrespective of whether they are erased, as they do not count for the current block). Let $\omega_m(x|\tau,\varepsilon_m,\psi_m,s_m)$ be the PMF of the number of packets ω_m of the current block useful for the block reconstruction that were received from QS m over the time interval τ. We hence have

$$\omega_m(x|\tau,\varepsilon_m,\psi_m,s_m) = \sum_{r=x}^{s_m} \delta_m(r+q_m|\tau,\psi_m,s_m) \binom{r}{r-x} \varepsilon_m^{r-x}(1-\varepsilon_m)^x, \tag{4}$$

where the rightmost term accounts for the probability that x packets out of r are delivered, while $r-x$ are erased (we remind the reader that packet erasures on link m occur independently with probability ε_m). The CDF $\Omega_m(x|\tau,\varepsilon_m,\psi_m,s_m)$ of the number of packets delivered in time is obtained by simply summing the PMF in (4). Extending the calculation from a single QS to the overall system, the block delivery PMF is given by the convolution of the QSs' PMFs:

$$\rho(\tau,\boldsymbol{\varepsilon}|\boldsymbol{\psi},\mathbf{s}) = \sum_{\mathbf{n}\in\mathcal{N}(\mathbf{s})} \prod_{m=1}^{M} \omega_m(n_m|\tau,\varepsilon_m,\psi_m,s_m). \tag{5}$$

where $\mathcal{N}(\mathbf{s})$ is the set of all possible vectors $\mathbf{n} = (n_1,...,n_M)$ of correctly received packets that result in the block being successfully decoded, i.e.:

$$\mathcal{N}(\boldsymbol{s}) = \Big\{ \mathbf{n} \in \mathcal{A} : n_m \le s_m, \forall m \in \{1,\dots,M\}, \sum_{m=1}^{M} n_m \ge K \Big\}. \tag{6}$$

In this work, we choose the reward function $r_{\boldsymbol{\psi}}(\mathbf{s}) = \rho(\tau,\boldsymbol{\varepsilon}|\boldsymbol{\psi},\mathbf{s})$. We now define a *policy* $\Pi : \Psi \to \mathcal{A}$ mapping states to actions. Let $\mathbf{r}(\Pi)$ be the vector with elements $r_{\boldsymbol{\psi}}(\Pi(\boldsymbol{\psi}))$ associated to the states $\boldsymbol{\psi} \in \Psi$, and let $\mathbf{T}_\Pi$ be the transition probability matrix associated with the policy, whose elements are simply given by $T_\Pi(\boldsymbol{\psi},\boldsymbol{\psi}',\tau_g) = T(\boldsymbol{\psi},\boldsymbol{\psi}'|\Pi(\boldsymbol{\psi}),\tau_g)$. We can define the long-term discounted reward $R_{\boldsymbol{\psi}}(\Pi)$ from state ψ as

$$R_{\boldsymbol{\psi}}(\Pi) = \sum_{j=0}^{\infty} \lambda^j \sum_{\boldsymbol{\psi}'\in\Psi} P[\boldsymbol{\psi}'(j) = \boldsymbol{\psi}'|\boldsymbol{\psi}(0) = \boldsymbol{\psi};\Pi] r_{\psi'}(\Pi), \tag{7}$$

where $\lambda \in [0,1)$ is the discount factor. The probability of being in state $\boldsymbol{\psi}'$ after j steps is an element of the matrix $\mathbf{T}_\Pi$

elevated to the j-th power. The vector $\mathbf{R}(\Pi)$, whose elements are associated to the long-term reward for each state for the given policy, is then such that $\mathbf{R}(\Pi) = (\mathbf{I} - \lambda \mathbf{T}_\Pi)^{-1}\mathbf{r}(\Pi)$. This allows us to compute the total reward starting from the known $\mathbf{r}(\Pi)$ and $\mathbf{T}_\Pi$. Finally, from the transition matrix we can also compute the steady state probability distribution $\boldsymbol{\varphi}$ of the system, and therefore the steady-state total reward $R(\Pi) = \boldsymbol{\varphi}^T \mathbf{R}(\Pi)$, where $(\cdot)^T$ is the transpose operator. Moreover, recalling (5) and defining the vector $\boldsymbol{\rho}(\tau, \Pi, \varepsilon)$ as the vector of values $\rho(\tau, \boldsymbol{\varepsilon}|\boldsymbol{\psi}, \Pi(\boldsymbol{\psi}))$ over the whole state space, the CDF of the block delivery time can be expressed as $\Gamma(\tau) = \boldsymbol{\varphi}^T \boldsymbol{\rho}(\tau, \Pi, \varepsilon)$.

B. Delayed Feedback

We now consider the delayed feedback scenario, in which $\tau_f > 0$. In this case, the state of the MDP is not the real queue size, but the size of the queue as perceived by the sender, i.e., with a delay of τ_f, and the state of the MC driving the service times is delayed by one step. We denote the delayed state as $\tilde{\boldsymbol{\psi}}$. The CDF of the number of delivered packets on a QS, given in (2), needs to be updated to reflect the delay in the feedback:

$$\tilde{D}_m(x|\tau_d, \tau_f, \tilde{\boldsymbol{\psi}}, s_m) = \sum_{c'_m \in \mathcal{C}_m} \Bigg[\theta_m(\tilde{c}_m, c'_m) \cdot \sum_{r=0}^{\tilde{q}_m} \delta_m \left(r|\tau_f, \tilde{\psi}_m, 0 \right) \Delta_m \left(x|\tau_d, (c'_m, \tilde{q}_m - r), s_m \right) \Bigg]. \quad (8)$$

This equation takes into account that packets that have not been acknowledged might already have been delivered, computing the *a posteriori* probability of the real state $\boldsymbol{\psi}$ given the delayed observation $\tilde{\boldsymbol{\psi}}$. The values of $\tilde{\Omega}_m(x|\tau, \varepsilon_m, \tilde{\psi}_m, s_m)$ and $\tilde{\rho}(\tau, \boldsymbol{\varepsilon}|\boldsymbol{\psi}, \mathbf{s})$ can then be calculated with the same method we used in (4) and (5), after simply replacing (8) to (2).

The transition probabilities for each QS also need to be updated as follows:

$$\tilde{T}(\tilde{\boldsymbol{\psi}}, \tilde{\boldsymbol{\psi}}'|\mathbf{s}, \tau_g, \tau_f) = \prod_{m=1}^{M} \sum_{r=0}^{\tilde{q}_m} \Bigg[\theta(c_m, c'_m) \delta_m \left(r|\tau_f, (c_m, 0), \tilde{q}_m \right) \times \delta_m \left(\tilde{q}_m - r + s_m - \tilde{q}'_m | \tau_g - \tau_f, (c'_m, 0), \tilde{q}_m - r + s_m \right) \Bigg]. \quad (9)$$

The equations from Sec. IV-A can be used to find the optimal policy with delayed feedback, replacing $\mathbf{T}_m$, D_m, Ω_m, and ρ with their *a posteriori* delayed versions $\tilde{\mathbf{T}}_m$, $\tilde{D}_m$. $\tilde{\Omega}_m$, and $\tilde{\rho}$.

C. Computation of the Optimal Policy

We can now solve the problem using the classic policy iteration algorithm [41, Ch. 4], which consists of two steps, policy evaluation and policy improvement, which are repeated until convergence. The algorithm is initialized with a policy function Π^0 and a value function v_Π^0, which are both set to all zeros. The iterative steps are then the following:

1) The policy is evaluated using

$$v_\Pi^{n+1}(\psi) = \sum_{\psi' \in \Psi} p(\psi'|\psi, \Pi^n(\psi)) \left(r(\psi, \Pi^n(\psi), \psi') + \lambda v_\Pi^n(\psi') \right), \quad (10)$$

for all ψ, where ψ is the current state, ψ' is the new state, $\mathbf{s}$ is the chosen schedule, and r is the instantaneous reward. The value function is an estimate of the long-term value that can be achieved in a given state using policy Π^n.

2) The policy is improved by choosing the action that maximizes the long-term value:

$$\Pi^{n+1}(\psi) = \arg\max_{\mathbf{s} \in \mathcal{A}} \sum_{\psi' \in \Psi} p(\psi'|\psi, \mathbf{s}) \left(r(\psi, \mathbf{s}, \psi') + \lambda v_\Pi^{n+1}(\psi') \right). \quad (11)$$

Policy iteration is guaranteed to converge to the optimal policy in finite-state MDPs with finite reward, as it improves the policy at each step, gradually converging to the optimum [42]. In the general case, the complexity of policy iteration is exponential in the number of states, making it particularly impractical for realistic problems. However, if the correct pivoting rule is adopted and the discount factor is constant in time, policy iteration reaches convergence after $N_{\text{it}} \leq \frac{|\Psi|^2(|\mathcal{A}|-1)}{1-\lambda} \log\left(\frac{|\Psi|^2}{1-\lambda}\right)$ cycles, and is strongly polynomial in the size $|\Psi|$ of the state space and the size $|\mathcal{A}|$ of the action space [43]. Each iteration uses at most $O(|\mathcal{A}||\Psi|^2)$ operations, making the resulting bound polynomial in the MDP size. Naturally, due to the curse of dimensionality, even a two-path system is extremely complex in practice, as there are thousands of possible states and tens of actions. It is worth remarking that the model can be extended to account for more general assumptions, such as random block size distributions, different service time statistics for the M queues, other coding schemes, and diverse queue lengths. However, such generalizations take a toll in terms of complexity of notation and analysis. We hence preferred simplicity over generality, in an effort to make it easier for the reader to follow the rationale and capture the essence of the proposed study. In any case, the complexity of the policy iteration approach can soon become an obstacle in practical settings, because of the exponential growth of the computational complexity. Reinforcement learning solutions might be a practical alternative to policy iteration if the problem becomes too large, but we leave this analysis to future work.

V. Analytical policy derivation

For very limited cases, it is possible to derive an analytical solution to the problem by computing the expected reward for each policy without recurring to iterative strategies. In particular, we consider a system with unit block size (i.e., $K = 1$) and $M = 2$ QSs with queue capacity $\ell = 1$: if a packet is already in service, no further packets can be queued in the same QS. We consider exponential service times with rates μ_1 and μ_2. Without loss of generality, we also assume $\mu_1 \geq \mu_2$. The QSs have instantaneous feedback and no erasures. In this case, there are only 4 possible states, namely, $\psi \in \{0, 1\}^2$. For the state $(1, 1)$ (i.e., when both queues are occupied), the only meaningful action is to transmit no packets, as anything that is transmitted will be dropped by the QS. Similarly, transmitting packets on the first link in state $(1, 0)$ is meaningless, so the only meaningful actions in that

State	Policy							
	A	B	C	D	E	F	G	H
(0,0)	(1,1)	(1,1)	(1,1)	(1,1)	(1,0)	(1,0)	(0,1)	(0,1)
(1,0)	(0,1)	(0,1)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(0,0)
(0,1)	(1,0)	(0,0)	(1,0)	(0,0)	(1,0)	(0,0)	(1,0)	(1,0)
(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)

TABLE II: Possible strategies.

state are $\mathcal{A}((1,0)) \in \{(0,0),(0,1)\}$, and the same goes for state $(0,1)$, with swapped indices. We can also immediately discard action $(0,0)$ in state $(0,0)$, which corresponds to the system never transmitting anything, so the meaningful actions in state $(0,0)$ are $\{(0,1),(1,0),(1,1)\}$.

With this knowledge, and excluding policies which differ only for actions in unreachable states and policies that only use one channel, we can identify the 8 non-trivial policies listed in Table II. We also note that the results for symmetric policies can be easily computed by swapping the indices and parameters. We have an MDP with the transition probability matrix $\mathbf{T}$ for a generic policy Π and expected instantaneous reward $\rho(q_1,q_2)$ expressed in equations (12) and (13) (see pg. 8) where $p_m = e^{-\mu_m \tau_g}$ and $r_m = e^{-\mu_m \tau_d}$. The expected reward is $\mathbb{E}[\rho|\Pi] = \boldsymbol{\phi}_\Pi \boldsymbol{\rho}_\Pi$, where the steady-state distribution $\boldsymbol{\phi}_\Pi$ is the left eigenvector of matrix $\mathbf{I} - \mathbf{T}_\Pi$ with eigenvalue 1.

As an example, we analyze policy A, which always transmits when possible. Its steady-state probability is $\boldsymbol{\phi}_A = ((1-p_1)(1-p_2), p_1(1-p_2), p_2(1-p_1), p_1 p_2)$. The expected reward is then:

$$\begin{aligned}\mathbb{E}_A[\rho] = \boldsymbol{\phi}_A \boldsymbol{\rho}_A =& 1 - p_1 p_2 - p_2 r_1 (1-p_1) - p_1 r_2 (1-p_2) \\ & - r_1 r_2 (1-p_1)(1-p_2).\end{aligned} \tag{14}$$

In general, finding the optimal policy requires an enumeration over the policy space, but this is computationally light, as the expected reward can be computed in closed form as we did for policy A. In the case where $\mu_1 = \mu_2 = \mu$, and consequently $p_1 = p_2 = p$ and $r_1 = r_2 = r$, the solution is simple, and the optimal policies are only A or E (which is identical to F, as the two paths have the same rate in this case). The expected reward for policy E is given by:

$$\mathbb{E}_E[\rho] = (1-p)(1-r) - \frac{p(1-p)(1-r)}{1+p(1-p)}. \tag{15}$$

Which policy is optimal then depends on the precise values of the parameters τ_g, τ_d, and μ, and we can define a boundary function that tells us the region of the parameter space in which the optimal policy is A, i.e., the values of the parameters for which $\mathbb{E}_A[\rho] \geq \mathbb{E}_E[\rho]$:

$$\tau_g \geq \frac{1}{\mu} \ln\left(\frac{1+\sqrt{4e^{\mu\tau_d}-3}}{2}\right) \Rightarrow \Pi^* = A. \tag{16}$$

Fig. 3 shows the optimal policy as a function of the parameter values in the cases with $\mu_1 = \mu_2 = 1$ and $\tau_g = \tau_d = 1$. In the first case, policies E and G are equivalent, as the two paths are identical. Asymmetrical policies such as E and G, which privilege one of the two paths and use the other as backup, are more convenient in cases in which the two paths have very different rates, while the symmetrical policy is optimal if the two paths are similar.

The set of possible deterministic policies contains $|\mathcal{A}|^{|\mathcal{S}|}$ elements, as a policy is a function $\Pi : \mathcal{S} \to \mathcal{A}$ mapping states to actions. Computing the expected reward for each policy explicitly, as we did for the simple example, has exponential complexity in terms of the state space size $|\mathcal{S}|$, while policy iteration converges to the optimum in polynomial time [43]. Therefore, we will only consider policy iteration for the full-sized problem, as both methods return the optimal policy, but the iterative solution is practically computable in a short time.

VI. Results

In this section, we investigate the performance and behavior of the optimal and heuristic strategies in some specific scenarios. We choose to define $r_\psi(\mathbf{s})$ as the delivery probability itself and the discount factor as $\lambda = 0.99$. With this choice the long term reward is between 0 and 100 and is a linear function of the delivery probabilities. As the discount factor is very close to 1, the reward is close to the long-term probability of success, expressed as a percentage, with only a slight preference for immediate rewards. We derived the optimal policy in each case using policy iteration, which converged in fewer than 10 iterations in all cases.

All the results below are derived analytically by computing the steady-state probabilities of the MDP and applying the strategies in each scenario. We can now analyze the optimal policy for the fork-join system in different conditions.

A. Heuristic Policies

We can now look at some practical policies, which are implemented in real MPTCP schedulers.

The *Constant Coding Rate (CCR)* policy sets a constant amount of redundancy $\beta \in [1,2]$, such that $N = \beta K$ (β is the inverse of the coding rate). The N packets are then split among the QSs proportionally to their rate at the current state. The policy can thus be defined as:

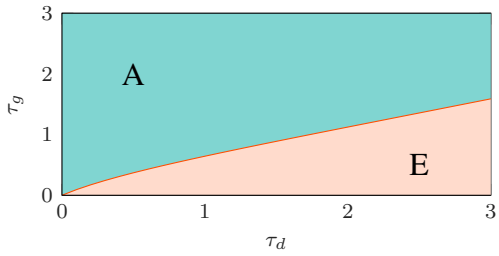
$$s_m(i) = \left\lfloor \frac{\mu_m(i) N}{\mu(i)} + \frac{1}{2} \right\rfloor. \tag{17}$$

The DAPS [22], DEMS [5], and BLEST [23] schedulers all employ different variants of the CCR policy, scheduling packets in different ways but setting a fixed redundancy. The CCR policy used in the simulation is an upper bound to their performance, as it finds the optimal scheduling for the given redundancy level. A particular instance of this policy is that with $\beta = 1$, where the original packets are split between the QSs without any coding. This policy, which we call *Plain Split (PS)*, is almost universally used in legacy schedulers such as STMS [21] or LowRTT.

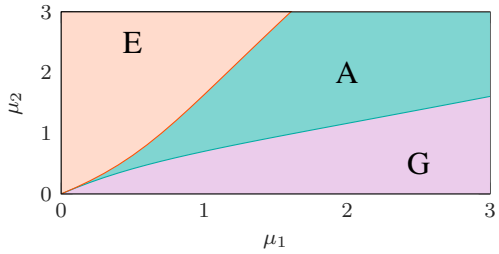
The *greedy* policy aims to achieve a delivery probability of the blocks above a specified threshold. In particular, for each block i the policy $s(i)$ is such that: *(i)*, the schedule is stable, i.e., $s_m(i) \leq \gamma \frac{\tau_i}{\mu_m(i)}$, with $\gamma < 1$, *(ii)*, the schedule should be such that the delivery probability for that block is not less than P_{thr} if allowed by the first constraint, otherwise it should maximize the delivery probability, and *(iii)*, the

$$\mathbf{T}_\Pi = \begin{pmatrix} (1-p_1\Pi_1(0,0))(1-p_2\Pi_2(0,0)) & p_1\Pi_1(0,0)(1-p_2\Pi_2(0,0)) & p_2\Pi_2(0,0)(1-p_1\Pi_1(0,0)) & p_1\Pi_1(0,0)p_2\Pi_2(0,0) \\ (1-p_1)(1-p_2\Pi_2(1,0)) & p_1(1-p_2\Pi_2(1,0)) & p_2\Pi_2(1,0)(1-p_1) & p_1p_2\Pi_2(1,0) \\ (1-p_1\Pi_1(0,1))(1-p_2) & p_2(1-p_1\Pi_1(0,1)) & p_1\Pi_1(0,1)(1-p_2) & p_1\Pi_1(0,1)p_2 \\ (1-p_1)(1-p_2) & p_1(1-p_2) & p_2(1-p_1) & p_1p_2 \end{pmatrix} \tag{12}$$

$$\boldsymbol{\rho}_\Pi = (1-(1-\Pi_1(0,0)(1-r_1))(1-\Pi_2(0,0)(1-r_2)), \Pi_2(1,0)(1-r_2), \Pi_1(0,1)(1-r_1), 0)^T, \tag{13}$$



(a) $\mu_1 = \mu_2 = 1$.



(b) $\tau_d = \tau_g = 1$.

Fig. 3: Optimal action boundary in the $\ell = 1$ example.

Scenario	ℓ	K	τ_g	τ_d	ε	β	γ	P_{thr}
Low load	60	20	20	12	0	1.5	0.8	0.9
Average load	60	20	15	15	0	1.3	0.8	0.9
High load	60	20	12	20	0	1.1	0.8	0.9

TABLE III: Parameters for the analyzed scenarios.

schedule should minimize N while satisfying the first two constraints.

This policy can be computed iteratively adding packets to the QS that has the highest delivery probability at each iteration, while checking the constraints and total delivery probability, and is used in the Latency-controlled End-to-End Aggregation Protocol (LEAP) [6] and HOP [27] schedulers. It is much more computationally efficient than the optimal policy, as it does not require the full solution of the Bellman equation, but it can also lead to reliability collapse if the QSs cannot support the requirements: in that case, indeed, it will progressively increase the redundancy (making the system unstable) until the maximum queue size is reached. On the other hand, the optimal policy can sacrifice some reliability or even drop a block to ensure that future blocks have the best chance to be delivered.

B. Performance Evaluation

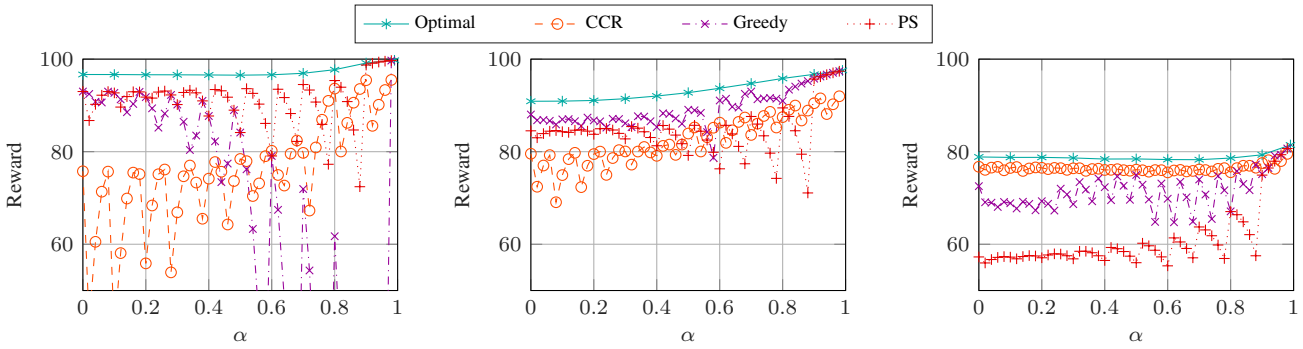
First, we consider how asymmetries in the capacity of the QSs affects the delivery probability. We consider at first a scenario with two QSs, each providing Independent and Identically Distributed (IID) service times with exponential distribution $\mathcal{Y}_m \sim \text{Exp}(\mu_m)$. We assume no delay in the feedback and no erasures. We set $\mu_1 = 1-\alpha$ and $\mu_2 = 1+\alpha$, so that the aggregate rate remains constant when varying the asymmetry parameter α in $[0,1]$[1]. In the following, all latencies and time intervals are normalized to the average packet service time, which is hence set to 1 time unit. We define three scenarios, whose main parameters are listed in Table III:

- The *high load* scenario assumes a offered load (before adding redundancy) of about 83% of the average capacity: the block generation period is short and the system is always at risk of building up queues. In this case, the latency requirement is relatively relaxed.
- The *average load* scenario has an offered load of about 67%: blocks have a longer generation period, but the deadline is tighter.
- The *low load* scenario has an offered load of 50%, leaving the queues mostly empty if no Forward Error Correction (FEC) is added, but the deadline in this case is extremely tight.

We can expect higher coding rates to be highly beneficial in the low load scenario, as in that case the main issue is not queuing delay but the natural variability of the QSs. Conversely, coding can be detrimental in the high load scenario, in which queuing is the most pressing issue, and adding redundant traffic to the already significant load on the system can move it closer to instability.

1) Channel Asymmetry Effects: The results, showing the achievable reward as a function of α, are shown in Fig. 4. In all scenarios we can see that the optimal reward slightly improves when $\alpha \to 1$, i.e., one channel has a much higher capacity than the other. This is due to the well-known queuing theory result that states that a single QS with service rate equal to 2 is better than two parallel ones with service rate 1. We can also observe that the optimal reward varies smoothly with α, whereas the other methods presents significant instability due to the finite granularity of packets. It is interesting to note that the CCR policy performs extremely well, almost at the level of the optimal policy, in the low load scenario, while it is the worst option in the high load scenario: this is because setting a constant level of redundancy can be beneficial if the load is low, but it increases the queuing delay if the load is the main limiting factor, even when adapting the coding rate β to the scenario. As expected, the PS policy shows the opposite pattern, with good performance in the high load scenario. The greedy policy is the closest to the optimum in the average load scenario, as its adaptive nature can balance queue accumulations on the two QSs. However, we can see that the high load scenario has a "snowball effect" if the QSs are asymmetrical: an increase of the queue on the fast QS cannot be compensated by the other, which in turn leads the controller

[1]Note that the two QSs are perfectly balanced for $\alpha = 0$ while the larger α the bigger the capacity of Q_2 over Q_1.



(a) High load scenario. (b) Average load scenario. (c) Low load scenario.

Fig. 4: Reward as a function of the system asymmetry.

to allocate even more redundancy to the fast QS, until the system is limited by the stability constraint. Something similar happens in the low load scenario, as the greedy policy is far from its target of 90% reliability, and will therefore tend to add too much redundancy and cause self-queuing delays.

The CDFs of the delivery time in the three scenarios are depicted in Fig. 5. We can see that the CDFs for the optimal policy depend mostly on the load of the system. Furthermore, the CDFs for the strategies that perform well in each scenario do not have significant differences in shape, which suggests that they are robust to changes in the deadline and reliability threshold. As we discussed above, the load on the system is the most important parameter in determining the efficiency of the heuristic schemes, as well as the optimal performance.

2) Markov-Modulated Channels: We now examine what happens when QSs have variable capacities. In the Markov scenario the service time, as a function of the asymmetry parameter ξ, is distributed as $\mathcal{Y}_{c_m} \sim \text{Exp}(\lambda_{c_m})$, where $\lambda_1 = \frac{1}{1-\xi}$ and $\lambda_2 = \frac{1}{1+\xi}$, thus maintaining unitary average rate. We set the transition matrices to:

$$\mathbf{\Theta}_1 = \mathbf{\Theta}_2 = \begin{bmatrix} 0.95 & 0.05 \\ 0.8 & 0.2 \end{bmatrix}. \tag{18}$$

This matrix has a corresponding steady state probability of $\kappa_1 \simeq 0.94$ for state 1 and $\kappa_2 = 1 - \kappa_1 \simeq 0.06$ for state 2. We set the exponential parameter for QS 1 as before to $\lambda_1 = \frac{1}{1+\xi}$, but in order to maintain the same average capacity we need to set $\lambda_2 = \frac{1-\kappa_1}{1-(1+\xi)\kappa_1}$. Moreover, in order to maintain positive capacities, it must be $\xi \leq \xi_{max} = \frac{1}{\kappa_1} - 1 = 0.0625$. For this reason, we define the normalized state asymmetry parameter as $\bar{\xi} = \frac{\xi}{\xi_{max}}$. All the other parameters for this scenario are the same that we used in the average load scenario.

The performance we obtained in this scenario is shown in Fig. 6: if the asymmetry is small, the optimum is still very close to the value of the average load scenario, while other strategies cannot compensate correctly for the variations in the capacity, with a much sharper decline in performance. We can observe that the optimal reward for $\bar{\xi} = 0.8$ is approximately 85%, which is close to the percentage of blocks with both QSs in state 1, which is 88.6%. This confirms the intuition that for high enough $\bar{\xi}$ the delivery probability depends mostly on the channel state rather than on the statistics of the service time in that state. In fact, if we consider each channel state combination separately, we find out that the success probability given that both QSs are in state 1 is 0.94, and 0.22 when only one of the channel is in state 2 while with both channels in state 2, in time delivery is basically impossible. We also notice that the heuristic methods are not suffering from quantization effects, as they where doing in scenarios 1, 2 and 3. This is because the channels are in the same state with very high probability ($\kappa_1^2 + \kappa_2^2 \approx 0.9$), so that asymmetry between the channel rates is rare. In this case, adding redundancy is not a good policy, as it leads to building up a huge queue when one or both channels are in the low-capacity state, and the CCR policy underperforms for this reason.

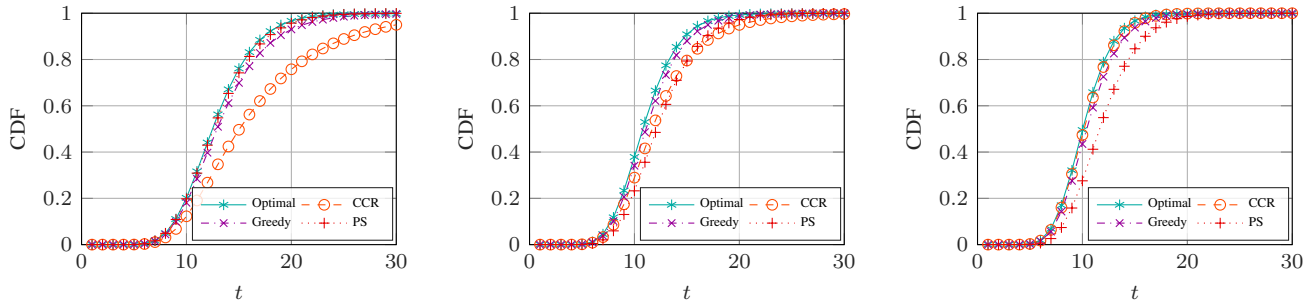
Interestingly, Fig. 6b shows that the PS policy actually has a better latency than the optimal policy if $\tau_d > 20$. This is not a violation of the optimality, as the aim of the optimal design is to maximize the probability at $\tau_d = 15$. However, in contrast to what we observed in the previous scenarios, here the choice of the deadline forces the controller to drop some packets, so that the effectiveness of the policy significantly depends on the choice of the deadline.

We can also see what happens if we vary the sojourn times of the MCs. We fix the steady state probabilities as they were in the Markov scenario and the state asymmetry parameter to $\bar{\xi} = 0.32$, then design the transition matrix to change the sojourn time in each state. In particular, given κ_2 and the transition probability $\Theta_{2,2}$ from state 2 to state 2, we compute the transition matrix as:

$$\mathbf{\Theta}_1 = \mathbf{\Theta}_2 = \begin{bmatrix} \frac{1-2\kappa_2-\Theta_{2,2}\kappa_2}{1-\kappa_2} & 1-\Theta_{2,2} \\ 1 - \frac{1-2\kappa_2-\Theta_{2,2}\kappa_2}{1-\kappa_2} & \Theta_{2,2} \end{bmatrix}. \tag{19}$$

It is easy to verify that this matrix has the properties described above.

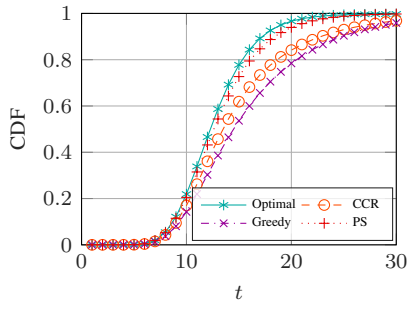
The results for the variable sojourn time scenario are shown in Fig. 7. The reward plot in Fig. 7a shows that a longer sojourn time on each state of the MC affects the optimal and the PS strategies only slightly. The CCR and greedy strategies seem to be impacted more severely, as they tend to send more redundancy during the long periods with lower capacity. In fact, the more packets are sent in this state, the longer it will take to clear out the backlog when the channel goes back to normal. In particular, this effect highlights the problem of the greedy policy, as it optimizes the chances for the current



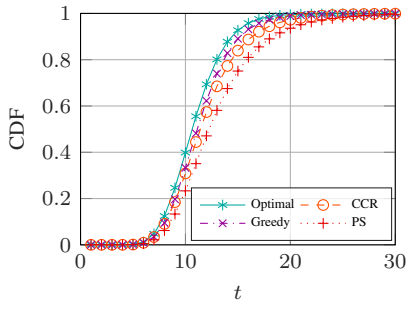
(a) High load, $\alpha = 0$.

(b) Average load, $\alpha = 0$.

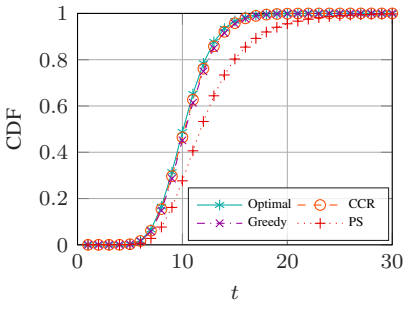
(c) Low load, $\alpha = 0$.



(d) High load, $\alpha = 0.5$.

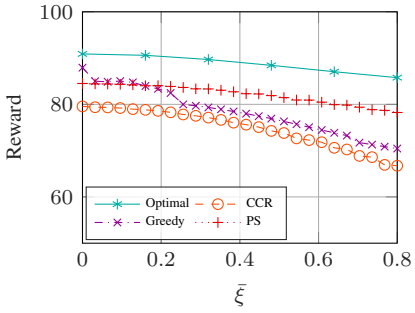


(e) Average load, $\alpha = 0.5$.

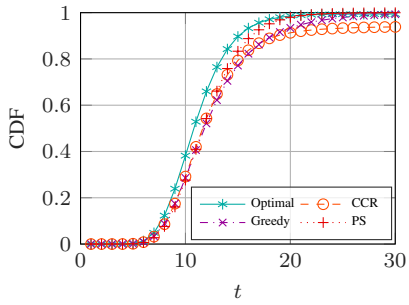


(f) Low load, $\alpha = 0.5$.

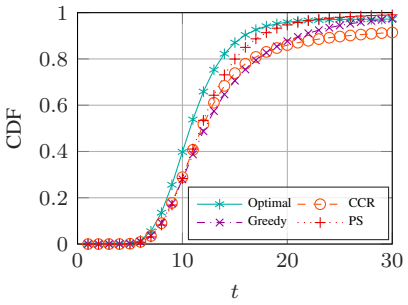
Fig. 5: Delivery time CDFs.



(a) Reward as a function of $\bar{\xi}$.



(b) Latency CDF with $\bar{\xi} = 0.32$.



(c) Latency CDF with $\bar{\xi} = 0.64$.

Fig. 6: Performance in the Markov scenario.

packet without considering the impact on the future. On the other hand, the optimal policy is barely affected by the length of the sojourn times, and it can outperform the PS policy by dropping some blocks during the low-capacity periods.

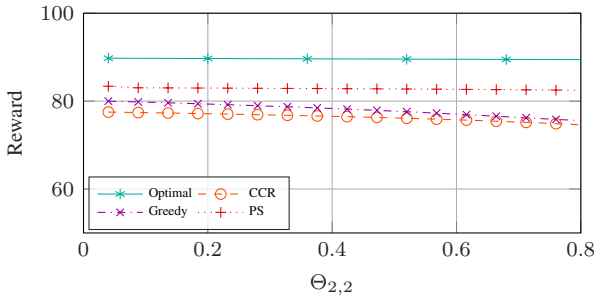
3) Feedback Delay and Error: Finally, we analyze the effects of channel impairments, substituting the channel with a Packet Erasure Channel (PEC) or adding a feedback delay τ_f on both channels. We consider the scenario with average load and add either an error probability or a feedback delay, checking their effect on the reward and, indirectly, on the delivery probability.

Fig. 8a shows that all strategies are affected by channel errors, as it requires redundancy just to recover the dropped packets. We can also notice an example of the "snowball effect" for the greedy policy: if the error probability is large enough, the greedy policy will increase redundancy and make the queue unstable, reducing the reward because of self-inflicted queuing delay and making the greedy policy worse than CCR. Naturally, the PS policy performs worse, as it does not include any redundancy and results in a block decoding failure for every dropped packet. As before, the optimal policy significantly outperforms the others, setting the correct amount of redundancy to balance the protection of the current block with the stability of the queue.

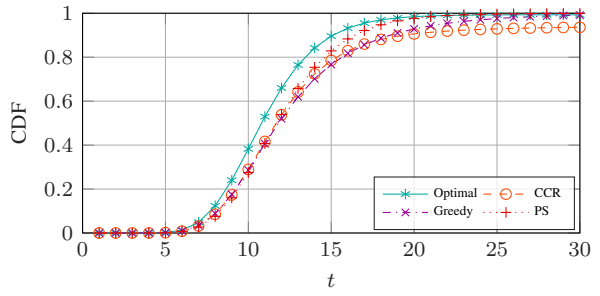
Fig. 8b shows the reward for the different strategies as a function of the feedback delay τ_f. The PS and CCR strategies, which do not rely on feedback, are unaffected by τ_f. Interestingly, the greedy and optimal strategies are also barely affected if they are aware of the delay, i.e., if the strategies are computed with the correct value of τ_f. However, by using the optimal policy for $\tau_f = 0$ in the case with $\tau_f > 0$, performance quickly becomes even worse than PS.

C. Parameter Sensitivity Analysis

We now investigate how much the policies are robust to uncertainty on the knowledge of the QS parameters. Fig. 9 shows what happens if the asymmetry α, the Markov state

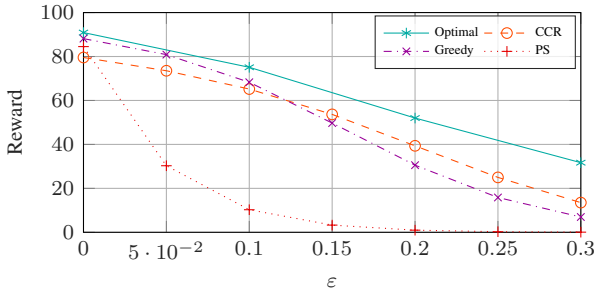


(a) Reward as a function of $\Theta_{2,2}$.

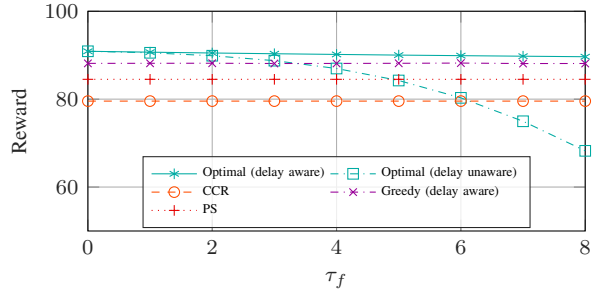


(b) Latency CDF with $\Theta_{2,2} = 0.36$.

Fig. 7: Performance in the Markov scenario with variable sojourn times.



(a) Reward as a function of the channel error probability.



(b) Reward as a function of the feedback delay.

Fig. 8: Performance in an imperfect channel.

asymmetry $\bar{\xi}$, the feedback delay τ_f, or the erasure probability ε are different from the ones used to compute the policy. Fig. 9a clearly shows that errors in estimating α have the strongest effect, and can significantly impact the reward. It is easy to see how inverting the fast and slow channels might wreak havoc on the strategies in cases with high asymmetry, reducing the reliability significantly. The other parameters are less impacting, although the error rate ε can also have a significant impact, as shown in Fig. 9d: this is due to the fact that the error rate alters the required redundancy, as it increases the difference between δ_m and ω_m. The feedback delay also has an effect on the reward, particularly if it is large: a significant mismatch between the expected and real feedback delays can significantly degrade performance, although not as much as ε or α.

D. Optimal Policy Analysis

We can also examine in depth the schedules generated by the optimal policy, looking at which states (i.e., queues length at any scheduling time) are visited more often and how the balance between reliability and low congestion is achieved.

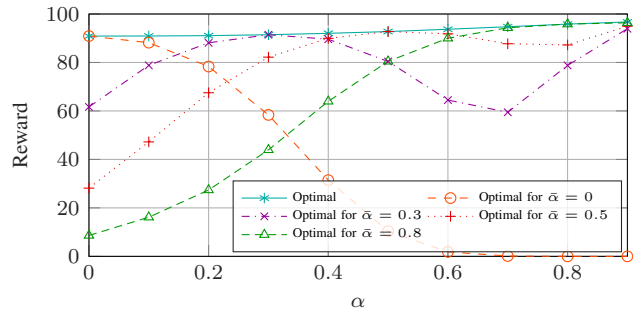
1) Load and Capacity Asymmetry: First, we analyze the strategies for different values of the asymmetry α with static channels with no error or feedback delay.

Fig. 10 shows three heatmaps representing the fraction of packets the first queue $\chi_1 = \frac{s_1}{s_1+s_2}$, the redundancy N/K and the state probability for all states with less than 8 packets in the queue, in the symmetric case ($\alpha = 0$). It is interesting to note that blocks are dropped more often for lower loads, for which the time deadline is looser. This counterintuitive behavior is explained by considering that, if the deadline is tight, dropping a block when the queue is long has marginal effects on the final performance, as that block has a low chance of being delivered anyway. On the other hand, if the deadline is looser, the probability of delivering the block on time is higher. However, these states are very rarely reached in practice, as the right side of the figure shows: while the scenarios with a higher load have a higher probability of reaching longer queues, the scheduling almost always maintains one of the two queues empty, effectively alternating the two QSs by placing more packets on the empty queue and reducing redundancy if the queues start filling up. In fact, the optimal policy is to maintain the queues as empty as possible, as any additional redundancy would hurt future blocks by causing self-queuing delay. In this case, and in most of the more complex ones we analyze below, state $(0, 0)$ has a very high probability, and the state of the queues changes only in unlucky cases.

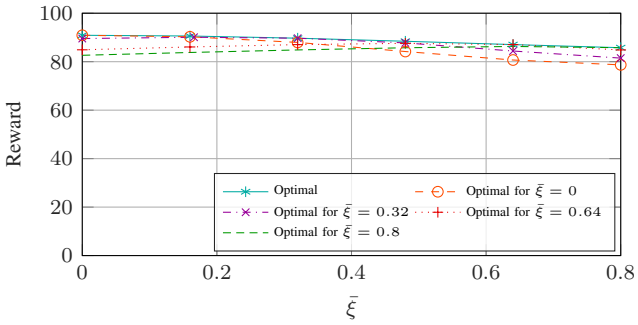
In this simple scenario, we can also give some additional results to explain the choices made by the optimal policy. We consider the highest possible reliability that can theoretically be obtained for the next step, given by schedule $\mathbf{s} = (\infty, \infty)$. This action will clearly penalize all future packet blocks, which will find infinitely many packets in the queues. However, it is optimal if we only consider the next step. The upper bound δ^*_m to the probability of delivering x packets from the current block on path m is then simply given by a shifted Poisson distribution:

$$\delta^*_m(x|\tau, q_m) = \begin{cases} \frac{\Gamma(q_m+1, \mu'_m\tau)}{q_m!}, & x = 0; \\ \frac{(\mu'_m\tau)^{q_m+x} e^{-\mu'_m\tau}}{(q_m+x)!}, & x > 0; \end{cases} \tag{20}$$

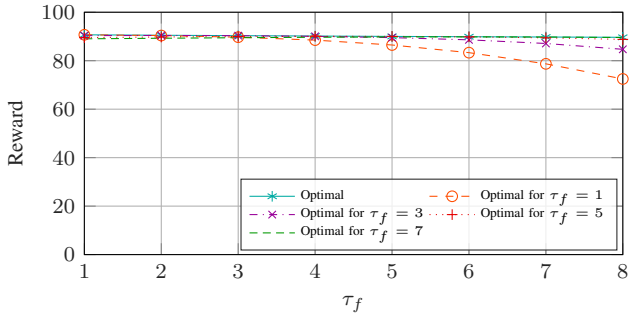
where $\mu'_m = \mu_m(1-\varepsilon_m)$, and $\Gamma(m, x)$ is the upper incomplete gamma function. As a block is on time if K or more of its



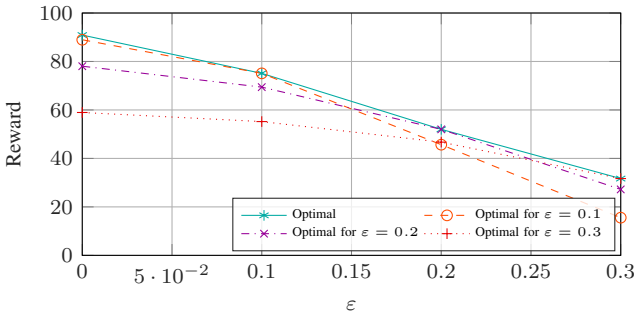
(a) Sensitivity analysis for channel asymmetry.



(b) Sensitivity analysis for state asymmetry.



(c) Sensitivity analysis for delayed feedback.



(d) Sensitivity analysis for erasure probability.

Fig. 9: Sensitivity analysis for several parameters in the average load scenario.

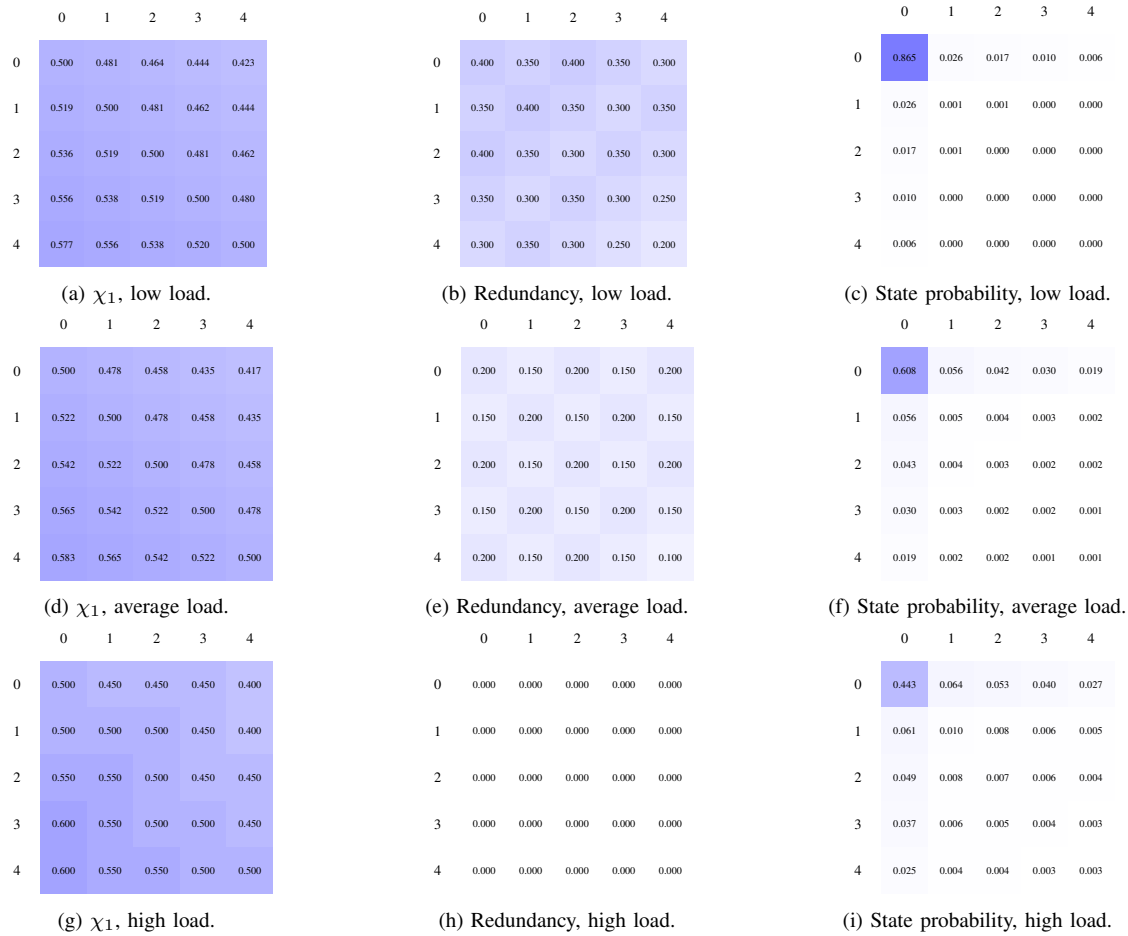
packets arrive by τ, we get the following delivery probability bound:

$$\begin{aligned}\rho^*(\tau,\boldsymbol{\varepsilon}|\mathbf{q}) &= 1-\sum_{x_1=0}^{K-1}\sum_{x_2=0}^{K-1-x_1} d_1^*(x_1|\tau,q_1)d_2^*(x_2|\tau,q_2)\\ &= 1-\frac{\Gamma(q_1+q_2+K,(\mu_1'+\mu_2')\tau)}{(q_1+q_2+K-1)!}\\ &\quad -\sum_{j=1}^{2}\frac{\Gamma(q_j+1,\mu_j'\tau)\Gamma(q_{3-j}+K,\mu_{3-j}'\tau)}{q_j!(q_{3-j}+K-1)!}.\end{aligned} \tag{21}$$

This result can simply be achieved by convolving the two Poisson distributions, considering the cases in which one of the two paths delivers no new packets separately. We define the *normalized reliability* $\eta(\mathbf{s},\mathbf{q})$ of schedule $\mathbf{s}$ as $\eta(\mathbf{s},\mathbf{q}) = \frac{\rho(\tau,\boldsymbol{\varepsilon}|\mathbf{q},\mathbf{s})}{\rho^*(\tau,\boldsymbol{\varepsilon}|\mathbf{q})}$. We can also define the *sustainability* of a schedule $\zeta(\mathbf{s})$ as the probability that the queue on each path will decrease or remain stable in the next time step. On a single path, this probability is ì $\zeta_m(s_m) = 1-\frac{\Gamma(s_m,\mu_m'\tau)}{(s_m-1)!}$. Since the two paths are independent we can define $\zeta(\mathbf{s}) = \zeta_1(s_1)\zeta_2(s_2)$. Naturally, there is a tradeoff between the η and ζ metrics, as schedules with a lower redundancy will tend to be more sustainable, as they require the transmission of fewer packets in time τ to maintain a stable queue, but their reliability will decrease, as fewer errors or late packets can make the block miss the deadline.

The values of $\eta(\mathbf{s},\mathbf{q})$ and $\zeta(\mathbf{s})$ for the optimal policy when $\alpha = 0$ in the average load scenario, shown in Fig. 12, can help us understand the tradeoff between the reliability of the current block and the sustainability of the schedule. Fig. 12a clearly shows that the optimal policy tends to become more conservative as the number of packets in the two queues increases: if the queues are empty, it can add more redundancy without affecting future blocks too heavily. However, as Fig. 10e shows, this does not imply that redundancy is monotonically decreasing. For example, the added redundancy when the queue state is $(1,0)$ is lower than when the state is $(2,0)$. What the normalized reliability shows is that the policy gradually becomes less focused on the current block as the queues increase in size, adding enough redundancy to compensate for the paths' unpredictability but looking more at the future. This is confirmed by Fig. 12b: while normalized reliability uniformly decreases as the queues increase, the sustainability shows an opposite trend. The probability of the queue decreasing or remaining stable tends to grow as the queues become more occupied, taking the system back to a more fruitful state for the next packet. This is not perfectly monotonic, as the optimal policy tends to be more aggressive and concentrate on one path if that path has an empty queue and the other is very full, but the general tradeoff holds, and the choice between immediate and future rewards is highly dependent on the state of the queues. Although this is not shown in the figures, if the queues become extremely big the optimal policy can even decide to drop a block: in this case, it accepts the certainty that the next block will not be delivered, but "resets" the queues by letting them empty out and finding a better state at the next block arrival. This case can be relatively frequent in Markov-modulated channels, when some states do not have enough capacity to sustain even the uncoded traffic.

We next examine the optimal policy in case of asymmetric channels. Fig. 11 shows the heatmaps for the average load scenario and two values of α, namely, 0.2 and 0.8. It is easy to see that, as Q_2 capacity grows, the number of packets on it



(a) χ_1, low load. (b) Redundancy, low load. (c) State probability, low load.

(d) χ_1, average load. (e) Redundancy, average load. (f) State probability, average load.

(g) χ_1, high load. (h) Redundancy, high load. (i) State probability, high load.

Fig. 10: Strategies for $\alpha = 0$.

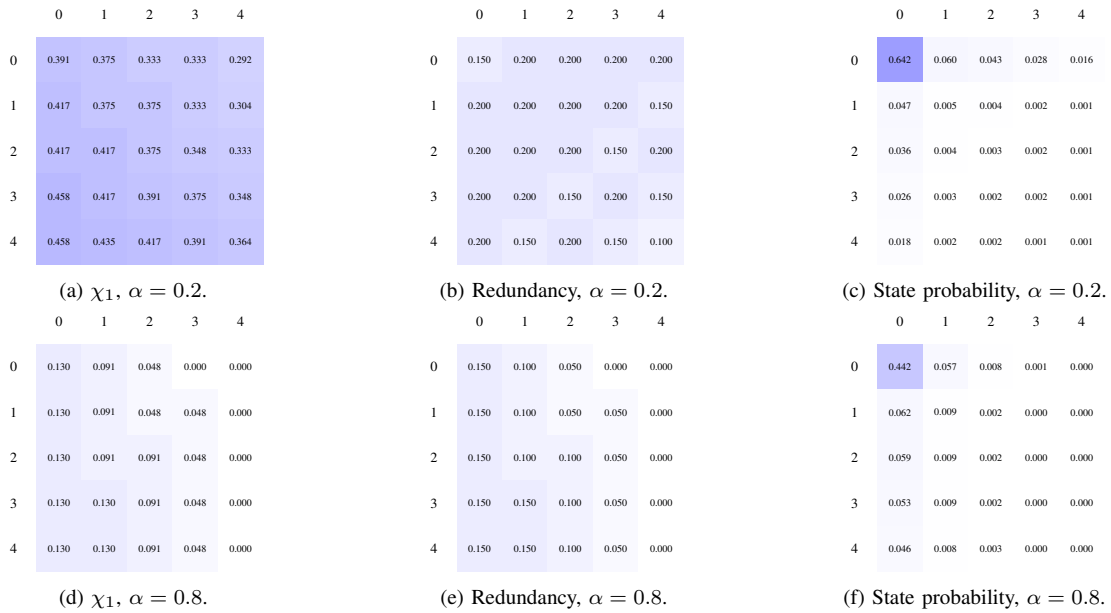
grows correspondingly, although redundancy decreases: if the first QS takes up more and more of the load, and the second one cannot provide additional reliability, it becomes harder to remain in favorable states with short queues, as the heatmaps on the right show. This is similar to a single-path transmission, and reliability is correspondingly lower, as we will see in the sections below.

2) Markov-Modulated Capacity: In this section, we analyze the schedules obtained with the optimal policy in the scenario described in VI-B2, and with the same parameters for the heuristic policy used for the average load scenario in section VI-B1.

Fig. 13 shows the same plots we presented above, for the states $(c_1, c_2) \in \{(1,1),(1,2),(2,2)\}$. We omit the state $(2,1)$ as it is symmetric to $(1,2)$. The probability of the queue state probability is conditional on the channel Markov chain state. Surprisingly, in state $(1,2)$ the threshold for dropping the block and the amount of redundancy only depend on the aggregate number of packets on both queues, whereas the fraction of packets sent to Q_1 is the only asymmetric feature of the policy (i.e.,it does not remain the same if we swap the QSs). Moreover, the policy in state $(1,2)$ is significantly less aggressive than that for static asymmetric channels. This yields a higher probability of empty queues in state $(1,1)$. In other words, the optimal policy involves sacrificing some performance in bad states to ensure higher success probabilities in the more favorable states: this is even clearer in state $(2,2)$, as blocks are almost always dropped.

If we set $\Theta_{2,2} = 0.84$, the sojourn time in state $(2,2)$ significantly increases, but the state is visited much less frequently, so having a bad connection is a rare but long-term event. In this case, blocks are dropped less frequently, and the controller deals with the bad state by putting more packets on the good QS. Indeed, as long as the QS will remain in the bad state 2, dropping blocks would lead to a very low reliability, and it is better to risk filling up the queue than just waiting for the QS to return to a good state. The state probability heatmaps on the right side also show that state $(0,0)$ has a lower probability if there is at least one bad QS.

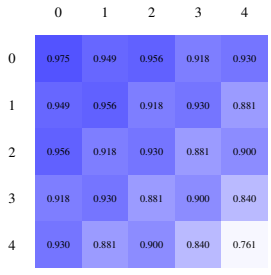
3) Delayed Feedback: Finally, we consider the delayed feedback case in the average load scenario. Fig. 15 shows the policy for different values of the delay τ_f. The delayed feedback does not have a large effect on the policy, but as τ_f increases, the state the scheduler sees corresponds to a gradually older picture of the actual state of the queues. If $\tau_f = 8$, as in Fig. 15d-f, the paths can deliver an average of 8 packets before the next block arrives. This leads to the optimal policy accepting longer queues, as the QSs have more time to empty them.



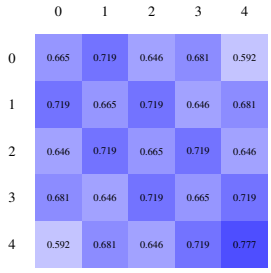
(a) χ_1, $\alpha = 0.2$. (b) Redundancy, $\alpha = 0.2$. (c) State probability, $\alpha = 0.2$.

(d) χ_1, $\alpha = 0.8$. (e) Redundancy, $\alpha = 0.8$. (f) State probability, $\alpha = 0.8$.

Fig. 11: Strategies for varying α and average load.



(a) $\eta(\mathbf{s}, \mathbf{q})$.



(b) $\zeta(\mathbf{s})$.

Fig. 12: Optimal tradeoff between reliability and sustainability with $\alpha = 0$ in the average load scenario.

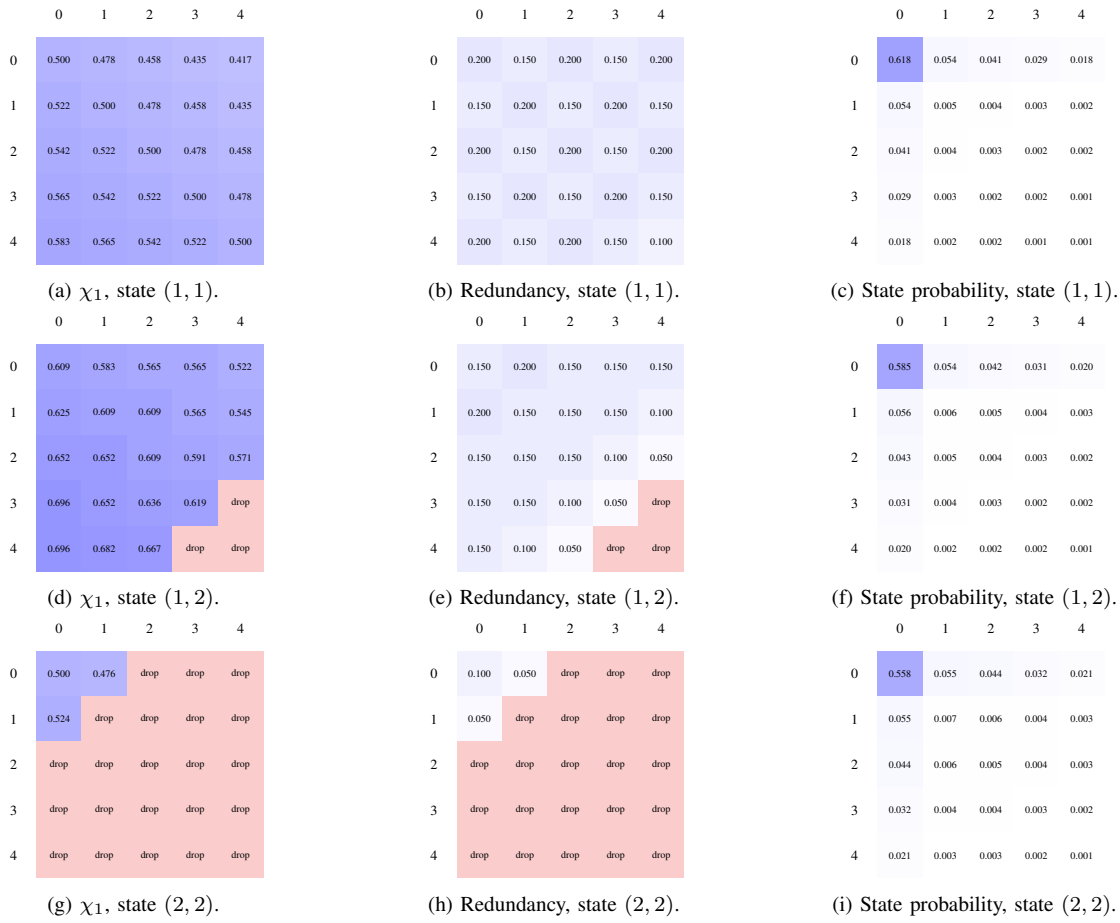
VII. Conclusions and Future Work

In this work, we have presented a model of parallel QSs with batch arrivals and latency constraints, deriving the optimal policy in terms of scheduling and packet-level coding to respect the delay constraint over the long term. The tradeoff between adding redundancy to protect the current block and avoiding self-queuing delay is complex, but we show that the optimal policy is robust to parameter estimation errors and significantly outperforms existing heuristic strategies.

The model we present in this work, and the results we presented, show that the decisions on coding and scheduling (i.e., the amount of redundancy needed to protect a block and the paths on which to send the coded packets) are inextricably tied, and are affected by a number of factors in non-trivial ways. The load on the system and the paths' capacities, the erasure probability of each path, the tightness of the deadline, the asymmetry between the available paths, the feedback delay, and the time-varying nature of paths all need to be considered to achieve good performance, even in a relatively simple scenario. In fact, even the simple example presented in Sec. V results in a non-trivial set of conditions under which redundancy is beneficial.

The results in more complex cases show that the heuristic strategies often adopted in the literature do not have consistent performance, and even relatively small changes in the scenario might require different settings or even an entirely different approach. In practice, this means that the coding and scheduling policy must be adapted to the specific conditions of the system, e.g., by adapting the amount of redundancy to the load and the tightness of the deadline. Furthermore, a greedy approach that only considers the next block also runs into issues, as it can trigger a *snowball effect* by increasing redundancy more and more as self-queuing delay builds, until the system is completely congested. In some cases, even dropping a block entirely might be warranted to reduce congestion and preserve future performance.

Even though the model does not fully represent a real system (e.g., capacity estimation is assumed to be perfect), it is complex enough to showcase these tradeoffs and for an initial evaluation of practical, foresighted scheduling algorithms. Future work on this subject might include the translation of the principles and insights from this model into more realistic systems and protocols, as well as applying the optimal policy directly in simpler systems.



(a) χ_1, state $(1,1)$. (b) Redundancy, state $(1,1)$. (c) State probability, state $(1,1)$.

(d) χ_1, state $(1,2)$. (e) Redundancy, state $(1,2)$. (f) State probability, state $(1,2)$.

(g) χ_1, state $(2,2)$. (h) Redundancy, state $(2,2)$. (i) State probability, state $(2,2)$.

Fig. 13: Strategies for the Markov-modulated channels with $\Theta = 0.32$.

REFERENCES

[1] S. Ferlin, T. Dreibholz, and Ö. Alay, "Multi-path transport over heterogeneous wireless networks: Does it really pay off?" in *Global Comm. Conf. (GLOBECOM)*. IEEE, Dec. 2014, pp. 4807–4813.

[2] J. J. Nielsen, R. Liu, and P. Popovski, "Ultra-reliable low latency communication using interface diversity," *IEEE Trans. Comm.*, vol. 66, no. 3, pp. 1322–1334, Nov. 2017.

[3] F. Chiariotti, A. A. Deshpande, M. Giordani, K. Antonakoglou, T. Mahmoodi, and A. Zanella, "QUIC-EST: A QUIC-enabled scheduling and transmission scheme to maximize VoI with correlated data flows," *IEEE Comm. Mag.*, vol. 59, no. 4, pp. 30–36, May 2021.

[4] 3GPP, "Service requirements for cyber-physical control applications in vertical domains," Dec. 2021, TS 22.104.

[5] Y. E. Guo, A. Nikravesh, Z. M. Mao, F. Qian, and S. Sen, "Accelerating multipath transport through balanced subflow completion," in *23rd Int. Conf. Mobile Computing Net. (MobiCom)*. ACM, Oct. 2017, pp. 141–153.

[6] F. Chiariotti, S. Kucera, A. Zanella, and H. Claussen, "Analysis and design of a latency control protocol for multi-path data delivery with pre-defined QoS guarantees," *IEEE/ACM Trans. Net.*, vol. 27, no. 3, pp. 1165–1178, May 2019.

[7] M. S. Squillante, "Stochastic analysis of multiserver systems," *ACM SIGMETRICS Perf. Eval. Review*, vol. 34, no. 4, pp. 44–51, Mar. 2007.

[8] G. Joshi, E. Soljanin, and G. Wornell, "Queues with redundancy: Latency-cost analysis," *ACM SIGMETRICS Perf. Eval. Review*, vol. 43, no. 2, pp. 54–56, Sep. 2015.

[9] C. Kim and A. K. Agrawala, "Analysis of the fork-join queue," *IEEE Trans. Computers*, vol. 38, no. 2, pp. 250–255, Feb. 1989.

[10] W. K. Grassmann, "Transient and steady state results for two parallel queues," *Omega*, vol. 8, no. 1, pp. 105–112, Jan. 1980.

[11] J. Sethuraman and M. S. Squillante, "Optimal stochastic scheduling in multiclass parallel queues," in *SIGMETRICS Int. Conf. Meas. Modeling Computer Sys.* ACM, May 1999, pp. 93–102.

[12] M. Konovalov and R. Razumchik, "Using inter-arrival times for scheduling in non-observable queues," in *31st European Conf. on Modelling and Simulation (ECMS)*, May 2017, pp. 667–672.

[13] J. Anselmi, B. Gaujal, and T. Nesti, "Control of parallel non-observable queues: asymptotic equivalence and optimality of periodic policies," *Stochastic Sys.*, vol. 5, no. 1, pp. 120–145, 2015.

[14] W. R. KhudaBukhsh, A. Rizk, A. Frömmgen, and H. Koeppl, "Optimizing stochastic scheduling in fork-join queueing models: Bounds and applications," in *Conf. Computer Comm. (INFOCOM)*. IEEE, May 2017.

[15] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, and E. Hyytia, "Reducing latency via redundant requests: Exact analysis," *ACM SIGMETRICS Perf. Eval. Review*, vol. 43, no. 1, pp. 347–360, 2015.

[16] G. Joshi, E. Soljanin, and G. Wornell, "Efficient redundancy techniques for latency reduction in cloud systems," *ACM Trans. Modeling Perf. Eval. Computing Sys. (TOMPECS)*, vol. 2, no. 2, pp. 1–30, Apr. 2017.

[17] K. W. Choi, Y. S. Cho, J. W. Lee, S. M. Cho, J. Choi *et al.*, "Optimal load balancing scheduler for MPTCP-based bandwidth aggregation in heterogeneous wireless environments," *Computer Comm.*, vol. 112, pp. 116–130, Nov. 2017.

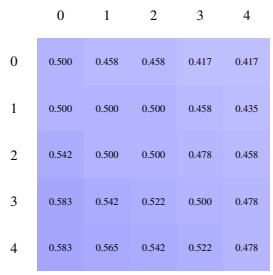
[18] E. Dong, M. Xu, X. Fu, and Y. Cao, "LAMPS: A loss aware scheduler for Multipath TCP over highly lossy networks," in *42nd Conf. Local Computer Net. (LCN)*. IEEE, Oct. 2017.

[19] J. Hwang and J. Yoo, "Packet scheduling for Multipath TCP," in *7th Int. Conf. Ubiq. Future Net. (ICUFN)*. IEEE, 2015, pp. 177–179.

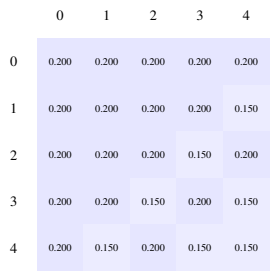
[20] D. Ni, K. Xue, P. Hong, and S. Shen, "Fine-grained forward prediction based dynamic packet scheduling mechanism for multipath TCP in lossy networks," in *23rd Int. Conf. Computer Comm. Net. (ICCCN)*. IEEE, Aug. 2014.

[21] H. Shi, Y. Cui, X. Wang, Y. Hu, M. Dai, F. Wang, and K. Zheng, "STMS: Improving MPTCP throughput under Heterogeneous Networks," in *Ann. Tech. Conf. (ATC)*. USENIX, 2018, pp. 719–730.

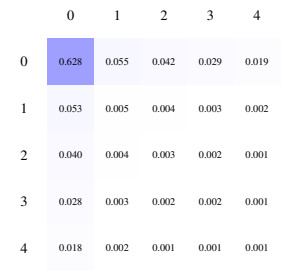
[22] N. Kuhn, E. Lochin, A. Mifdaoui, G. Sarwar, O. Mehani, and R. Boreli, "DAPS: Intelligent delay-aware packet scheduling for multipath transport," in *Int. Conf. Comm. (ICC)*. IEEE, 2014, pp. 1222–1227.



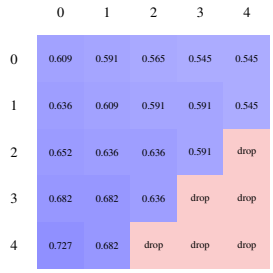
(a) χ_1, state $(1,1)$.



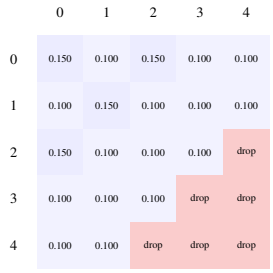
(b) Redundancy, state $(1,1)$.



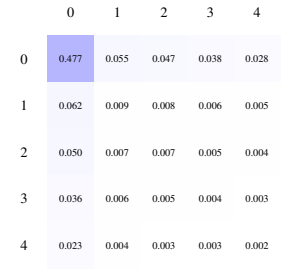
(c) State probability, state $(1,1)$.



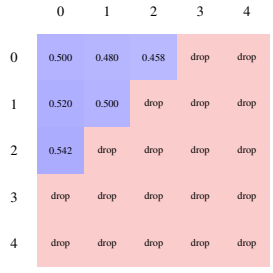
(d) χ_1, state $(1,2)$.



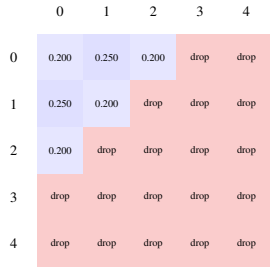
(e) Redundancy, state $(1,2)$.



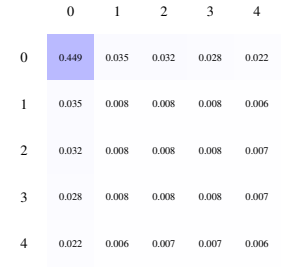
(f) State probability, state $(1,2)$.



(g) χ_1, state $(2,2)$.

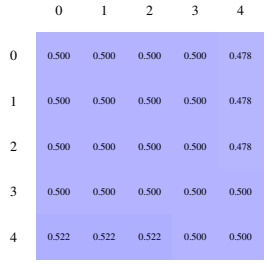


(h) Redundancy, state $(2,2)$.

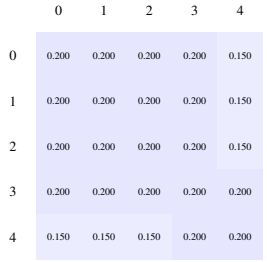


(i) State probability, state $(2,2)$.

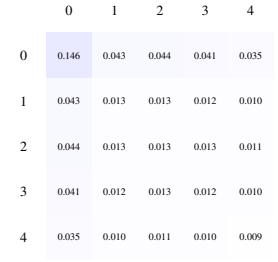
Fig. 14: Strategies for $\Theta_{2,2} = 0.84$.



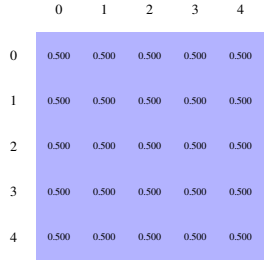
(a) χ_1, $\tau_f = 4$.



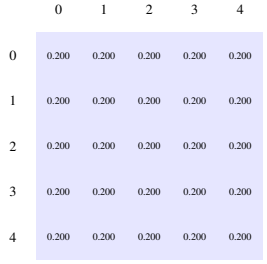
(b) Redundancy, $\tau_f = 4$.



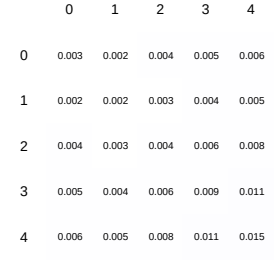
(c) State probability, $\tau_f = 4$.



(d) χ_1, $\tau_f = 8$.



(e) Redundancy, $\tau_f = 8$.



(f) State probability, $\tau_f = 8$.

Fig. 15: Strategies for varying τ_f.

[23] S. Ferlin, O. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *IFIP Net. Conf.* IEEE, May 2016, pp. 431–439.

[24] Y.-C. Chen, Y.-s. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley, "A measurement-based study of Multipath TCP performance over wireless networks," in *SIGCOMM Internet Meas. Conf. (IMC)*. ACM, Oct. 2013, pp. 455–468.

[25] A. Garcia-Saavedra, M. Karzand, and D. J. Leith, "Low delay random

linear coding and scheduling over multiple interfaces," *IEEE Trans. Mobile Computing*, vol. 16, no. 11, pp. 3100–3114, Mar. 2017.
[26] Y. Cui, L. Wang, X. Wang, H. Wang, and Y. Wang, "FMTCP: A fountain code-based Multipath Transmission Control Protocol," *IEEE/ACM Trans. Net.*, vol. 23, no. 2, pp. 465–478, Apr. 2015.
[27] F. Chiariotti, A. Zanella, S. Kucera, K. Fahmi, and H. Claussen, "The HOP protocol: Reliable latency-bounded end-to-end multipath communication," *IEEE/ACM Trans. Net.*, vol. 29, no. 5, pp. 2281–2295, 2021.
[28] M. Polese, F. Chiariotti, E. Bonetto, F. Rigotto, A. Zanella, and M. Zorzi, "A survey on recent advances in transport layer protocols," *IEEE Comm. Surveys Tut.*, vol. 21, no. 4, pp. 3584–3608, Aug. 2019.
[29] 3GPP, "Enhancement of 3GPP support for V2X scenarios," Apr. 2022, TS 22.186.
[30] T. H. Szymanski, "Supporting consumer services in a deterministic industrial internet core network," *IEEE Comm. Mag.*, vol. 54, no. 6, pp. 110–117, 2016.
[31] F.-C. Kuo, "Assessment of LTE wireless accessing for managing traffic flow of IoT services," *Mobile Net. Appl.*, vol. 24, no. 3, pp. 853–863, 2019.
[32] W. Qi, B. Landfeldt, Q. Song, L. Guo, and A. Jamalipour, "Traffic differentiated clustering routing in DSRC and C-V2X hybrid vehicular networks," *IEEE Trans. Vehic. Tech.*, vol. 69, no. 7, pp. 7723–7734, Apr. 2020.
[33] M. Lecci, M. Drago, A. Zanella, and M. Zorzi, "An open framework for analyzing and modeling XR network traffic," *IEEE Access*, vol. 9, pp. 129 782–129 795, Sep. 2021.
[34] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM SIGCOMM Comp. Comm. Review*, vol. 27, no. 2, pp. 24–36, Apr. 1997.
[35] T. Bonald, "Comparison of TCP Reno and TCP Vegas: efficiency and fairness," *Perf. Eval.*, vol. 36, pp. 307–332, Aug. 1999.
[36] G. Vinnicombe, "On the stability of networks operating TCP-like congestion control," *IFAC Proc. Vol.*, vol. 35, no. 1, pp. 217–222, Jan. 2002.
[37] N. Gholipoor, S. Parsaeefard, M. R. Javan, N. Mokari, H. Saeedi, and H. Pishro-Nik, "Cloud-based queuing model for tactile internet in next generation of RAN," in *91st Vehic. Tech. Conf. (VTC2020-Spring)*. IEEE, May 2020.
[38] A. Mathew, M. Srinivasan, and C. S. R. Murthy, "Packet generation schemes and network latency implications in SDN-enabled 5G C-RANs: Queuing model based analysis," in *30th Ann. Int. Symp. on Personal, Indoor and Mobile Radio Comm. (PIMRC)*. IEEE, Sep. 2019.
[39] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Sys. Tech. J.*, vol. 39, no. 5, pp. 1253–1265, Sep. 1960.
[40] R. E. Barlow, A. W. Marshall, and F. Proschan, "Properties of probability distributions with monotone hazard rate," *The Annals of Mathematical Statistics*, pp. 375–389, 1963.
[41] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, Oct. 2018.
[42] R. A. Howard, *Dynamic programming and Markov processes*. John Wiley, 1960.
[43] Y. Ye, "The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate," *Math. Op. Research*, vol. 36, no. 4, pp. 593–603, Nov. 2011.

APPENDIX

We define a random variable X which can take values in $\mathbb{R}^+$, whose PDF and CDF are denoted as $f(x)$ and $F(x)$, respectively. We then define two values y and z, with $y > z > 0$. We define the conditioned probability $\xi(z) = P(X \geq y + z | X \geq z)$. The value of $\xi(0)$ is simply given by $\xi(0) = 1 - F(y)$. We can compute $\xi(z)$ by using Bayes' theorem:

$$\begin{aligned} P(X \geq y + z | X \geq z) &= \frac{P(X \geq z | X \geq y + z) P(X \geq y + z)}{P(X \geq z)} \\ &= \frac{1 - F(y + z)}{1 - F(z)}. \end{aligned} \tag{22}$$

We can then give the condition for which $\xi(z)$ is monotonically decreasing:

$$\frac{f(y + z)}{1 - F(y + z)} - \frac{f(z)}{1 - F(z)} \geq 0. \tag{23}$$

This is a condition on the hazard rate function $\nu(z) = \frac{f(z)}{1 - F(z)}$, often used in reliability applications [40]. If $\nu(z)$ is monotonically non-decreasing, $\xi(z)$ is monotonically non-increasing, i.e., the conditional probability of X being larger than $y + z$ if we already know that it is larger than z decreases as z increases. The hazard rate is often hard to compute, but it is monotonically increasing for normal distributions, Gamma and Weibull distributions with a shape parameter $\alpha > 1$. The exponential distribution has a constant hazard rate, as it is memoryless. A decreasing hazard rate is sometimes used as the definition of heavy-tailedness.