Adaptive Prioritized Random Linear Coding and Scheduling for Layered Data Delivery from Multiple Servers

Nikolaos Thomos, *Member, IEEE*, Eymen Kurdoglu, *Student Member, IEEE*, Pascal Frossard, *Senior Member, IEEE*, and Mihaela van der Schaar, *Fellow, IEEE*

Abstract

In this paper, we deal with the problem of jointly determining the optimal coding strategy and the scheduling decisions when receivers obtain layered data from multiple servers. The layered data is encoded by means of Prioritized Random Linear Coding (PRLC) in order to be resilient to channel loss while respecting the unequal levels of importance in the data, and data blocks are transmitted simultaneously in order to reduce decoding delays and improve the delivery performance. We formulate the optimal coding and scheduling decisions problem in our novel framework with the help of Markov Decision Processes (MDP), which are effective tools for modeling adapting streaming systems. Reinforcement learning approaches are then proposed to derive reduced computational complexity solutions to the adaptive coding and scheduling problems. The novel reinforcement learning approaches and the MDP solution are examined in an illustrative example for scalable video transmission. Our methods offer large performance gains over competing methods that deliver the data blocks sequentially. The experimental evaluation also shows that our novel algorithms offer continuous playback and guarantee small quality variations which is not the case for baseline solutions. Finally, our work highlights the advantages of reinforcement learning algorithms to forecast the temporal evolution of data demands and to decide the optimal coding and scheduling decisions.

N. Thomos is with the Computer Science and Electronic Engineering Department, University of Essex, Colchester, United Kingdom (e-mail: nthomos@essex.ac.uk). E. Kurdoglu is with the Polytechnic Institute of NYU, NY, USA, USA (e-mail: eymen@vision.poly.edu). P. Frossard is with the Signal Processing Laboratory 4 (LTS4), Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland (e-mail: pascal.frossard@epfl.ch). M. van der Schaar is with the Networks, Economics, Communication Systems, Informatics and Multimedia Research Lab, UCLA, CA, USA (e-mail: mihaela@ee.ucla.edu). This work has been initiated while the first two authors were at EPFL. It has been supported by the Swiss National Science Foundation, under grants PZ00P2-121906, PZ00P2-137275, and the project "Adaptive Network Coding for Video Communications" funded by Hasler foundation.

Index Terms

Prioritized Random Linear Codes, rateless codes, layered data, MDP, Q-learning, virtual experience.

I. INTRODUCTION

The emergence of new multimedia devices such as laptops or smartphones has motivated the research on efficient data delivery mechanisms that are able to adapt to dynamic network conditions and heterogeneous devices' capabilities. The quality of experience of the different receivers depends on their display size, processing power, network bandwidth, *etc.* In order to accommodate for such a diversity, the data is progressively encoded in several quality layers. This permits to offer a basic quality to receivers with limited capabilities, while other devices can have higher quality of experience. Further, the layered encoding of the data helps data delivery systems to cope with the strict delivery deadlines as data quality can be adapted to meet decoding deadlines when network resources get scarce.

Methods such as Fountain codes (Random Linear Coding [1], Raptor codes [2], LT codes [3]) have been proposed recently for efficient data delivery in error-prone networks. Usually, Fountain codes are applied at servers, and the receivers obtain the requested data from possibly multiple servers. The use of such codes leads to significant performance gains [4], [5] compared to classical channel codes because channel protection does not have to be determined beforehand. If the set of clients is very heterogeneous, the source data can be encoded in a scalable way for adaptive delivery. This can be achieved with the use of embedded (prioritized) codes such as Expanding Fountain (EWF) Codes [6], Prioritized Random Linear Codes (PRLC) [7] and Prioritized Random Linear Network Codes (PRLNC) [8]. These schemes encode the layered data progressively and form classes of packets with unequal importance. However, in order to optimize the decoding quality an optimized coding algorithm along with a proper scheduling mechanism and inter-server coordination are needed for efficient resources allocation.

In this paper, we examine data transmission from multiple servers, which improves the resilience of the communication process [4]. When a receiver is connected to multiple servers, the receiver could reliably receive the data from the other servers even if one of the servers fails. Our communication scenario can be seen as a special case of data delivery in peer-to-peer systems [9], [10]. Here, we focus on the multi-server time-constrained transmission of layered data encoded with PRLC. PRLC is employed to respect the unequal importance of the data layers. We consider a novel framework where the data is encoded in groups of elements (*i.e.*, generations) that have the same decoding deadline, which are transmitted

concurrently to the receiver in order to improve performance and decrease the decoding delay. Our framework is different from common rateless and network coding based systems [11]–[15], where data is sequentially transmitted. In this context, we devise a receiver driven protocol in order to communicate information about the link statistics and the buffers states of the different servers and decide on the data requests. Then, we investigate a new problem of deciding the optimal coding and scheduling policies for layered data transmission such that the quality at the receivers is maximized, while the decoding deadlines are respected. We first propose a data-aware optimization framework based on Markov Decision Processes (MDPs) [16] in order to characterize the performance of our adaptive streaming system. While MDPs offer an elegant way to describe our novel framework, they require high computational complexity and perfect knowledge of the environment in order to describe the state transition probabilities and reward matrices. In order to cope with these limitations, we propose a variant of the Q-learning approach [17] that is inspired from the "virtual experience" concept introduced in [18]. We start with the observation that channel dynamics are independent from the states and actions, which in our formulation represent the buffer content of the receiver and the senders and the requested packets, respectively. Then, we exploit the knowledge acquired by observing a state-action pair by extending it to multiple statistically equivalent state-action pairs. This new algorithm trades off convergence rate of the optimization algorithm with computational complexity. Like Q-learning, our solution does not require precomputing and storing the transition probabilities and reward matrices. It however converges faster than Q-learning, with a higher computational complexity. We present simulation results for an illustrative scalable video transmission example. These results show the advantages of the proposed method compared to myopic methods for coding and scheduling and to the original Q-learning, in terms of video quality and quality fluctuations. Moreover, our scheme offers small quality variations as it efficiently considers the data importance in the coding and scheduling decisions. This is an important advantage in real-time video communication systems. Further, the proposed Q-learning variant improves the data quality with only minimal additional computational cost compared to Q-learning. Overall, the proposed solution permits easy adaptation to changing network conditions, as the optimal coding and scheduling decisions can be continuously updated based on the actual network dynamics.

In summary, the main contributions in this paper are the following:

- we propose the first foresighted decision making scheme in RLC and network coding literature that simultaneously considers data from multiple generations and multiple senders.
- we formulate the optimal coding and scheduling decision problem for layered data and multiple

generations with the help of MDPs.

- we propose a Q-learning variant that converges faster than the original Q-learning algorithm to the optimal coding and scheduling policies.

The paper is organized as follows. In Section II, we provide a brief discussion of Fountain and network coding communication systems that target the delay-constrained delivery of data. Next, we present in Section III the basic principles of the Prioritized Random Linear Coding and describe our receiver-driven communication protocol. In Section IV, we formulate our joint PRLC coding and scheduling optimization problem with the help of Markov Decision Processes. Then, we describe our reduced complexity solution that is based on Q-learning in Section V. In Section VI, we analyze our framework for an illustrative video application with multiple servers and a single receiver and evaluate its performance for transmission of layered video data. Finally, conclusions are drawn in Section VII.

II. Related work

In this section, we briefly overview and position our work with respect to other systems that employ Fountain codes and RLC for delivery of time-constrained data from single and multiple sources.

Streaming of scalable video from multiple servers has been investigated in [4], where data from each Group of Pictures (GOP) and layer is independently protected by means of Raptor codes. A rate allocation algorithm determines the optimal number of Raptor packets to be sent from each GOP and layer. Similar concepts have been presented in [5] for transmission of scalable video in mobile ad hoc networks (MANETS). Video on Demand delivery in wireless mesh networks has been studied in [19] where the video streams are protected with Reed Solomon codes prior to transmission. Then, the streams are transmitted over multiple disjoint paths that are discovered by two heuristics algorithms. The optimal rate per path is decided by a data unaware optimization algorithm. The problem of time-constrained data streaming from a single server has been more extensively studied in the literature. Although the extension from a single server to multiple servers is not trivial, the investigation of single server systems sheds light on the properties that efficient multi-source streaming schemes should have.

A variety of Application Layer Forward Error Correction (AL-FEC) schemes appropriate for transmission of scalable video over broadcast erasure channels have been recently presented in [4], [5], [11]–[13], [20]. In [11], the sliding window concept has been introduced and it has been later combined in [12] with Unequal Error Protection (UEP) Raptor codes for providing enhanced error robustness. This scheme has been further improved in [13], where data replication is used prior to the application of Fountain codes.

This results in stronger protection for the most important layers. The expanding window approach has been proposed in [20] for video multicast as an alternative to the sliding window method.

Unfortunately, digital Fountain codes such as Raptor codes and LT codes may perform poorly in terms of delay [21], [22]. To this aim, systematic Random Linear Codes (RLC) with feedback have been studied in [21] for minimizing the average delay in database replication systems. The feedback messages contain information about the packets that have been delivered to the receivers. These messages are used to optimize the coding decisions such that the decoding delay is small. The decoding delay distribution has been investigated in [22] for RLC systems. Markov chains are utilized in order to find the optimal coding solution for the case of two receivers. However, this problem is computationally intractable for three or more receivers and hence only a heuristic solution is presented. The delay benefits of a RLC based scheme have been examined in [23], where repair packets are generated on-the-fly through RLC according to feedback messages that are periodically received by the servers. This scheme copes efficiently with the packet erasures and is appropriate for real-time sources such as video. From the above studies, the advantages of RLC over Fountain codes in terms of delay become clear.

Although the employment of RLC may result in large delay gains, scheduling of RLC encoded data is a NP-hard problem [21]. Since it is hard to find the optimal coding and scheduling solution, a number of practical solutions in wireless networks that exploit the feedback messages that are available to each network node are proposed in [14], [15], [24]. In [24], the size of the coding generations is adapted by considering hard deadlines imposed by the target applications. It is shown that the generation size should become smaller as the delivery deadline approaches. The joint coding and scheduling problem is studied in [15] for video transmission in wireless networks, where coding operations are performed in binary fields. The problem is modeled as MDP and is solved by a dynamic programming algorithm. The employed XOR-based coding permits instantaneous decoding but with some performance penalty compared to schemes that work in larger finite fields. The approaches in [15], [24] provide some interesting directions towards the development of adaptive data delivery mechanisms. However, their applications are rather limited as the work in [15] is appropriate only for small generations and the work in [24] is data oblivious. Both works are myopic and process sequentially the generations which generally penalizes the performance compared to foresighted strategies. Finally, the work in [25] proposes an interesting framework with concurrent generation delivery. Two major problems are associated with this approach: (a) the decoding computational cost and (b) the coding overhead [26] (a header with the coding coefficients that should be communicated along with the payload of the packets). The decoding computational cost and the overhead increase with the number of generations involved in the encoding operations. This

prohibits the use of the approach in [25] in real-time applications. The system that we propose in this paper does not have such limitations.

Our work departs from the common choice of sequential processing of generations and considers simultaneously coding and scheduling of data from multiple generations. It aims to find the optimal PRLC coding and scheduling decisions so that the data quality is maximized and the playback of streaming data is smooth at receivers. To the best of our knowledge, this strategy has not been investigated for PRLC. Our system is inspired by the recent advances in cross layer optimization for wireless communication systems [18], [27]. In [27], the joint scheduling and rate allocation problem of wireless video from multiple servers is modeled as Multi-User MDP (MUMDP) and then is effectively solved with online reinforcement learning methods [28] where multiple states are simultaneously updated using stochastic sub-gradient methods [29]. The work in [18] makes joint physical layer and system level power management decisions for delay sensitive applications. It uses the concepts of the "postdecision state" [30] in order to decompose the problem in two learning problems: one that factors the known dynamics and another that has the unknown dynamics. Further, it proposes the concept of "virtual experience" in order to accelerate the learning process.

III. Prioritized delivery of PRLC encoded data

A. Network model

We consider a framework where receivers request layered data from multiple servers. The time is slotted and receivers send their data requests to all their parents simultaneously. The servers then encode the data in order to meet the requests of receivers, and deliver coded packets of uniform size to the receivers. The link connecting a server i with the receiver j is denoted by the tuple (i, j) and is characterized by three parameters: the transmission rate f_{ij} expressed in packets per second, the packet loss probability ϵ_{ij} and the transmission delay η_{ij}. We assume that f_{ij} and ϵ_{ij} do not change very frequently, at least not faster than the optimization decisions. This is a valid assumption in slow fading wireless channels [31]. Similar assumptions have been made by other reinforcement based schemes proposed for wireless channels [32], [33]. It is also valid in wired channels, as such channels do not change frequently [31] (chapter 2). In general, our scheme can deal with channels that vary fast as long as accurate channel statistics are available to the receiver. When channel conditions change fast, the convergence is slower, because the optimal policies also change. However, the employment of PRLC coding makes our system more resilient to channel variations as shown in [8], [34].

Due to nodes' limited upload and download capabilities, all the servers are not necessarily available for every receiver. Hence, the set of servers for the receiver j is represented by

$$P_j = \{i : (i,j) \in E\},$$

where E is the set of links, which are in general non-homogeneous in terms of their loss rates. Here, without loss of generality we assume that the set of servers P_j does not change with the time. Our algorithm can deal with changing P_j, however at the cost of slower convergence. The link parameters, as well as the amount of data available at servers, are periodically sent to receivers. Finally, the transmission delays on each link are non-homogeneous and modeled as *i.i.d.* random variables. Thus, the channels do not interfere with each other and no congestion problems occur. To deal with congestion, we could adopt methods similar to [33], [35].

B. Layered data

Since servers and receivers are heterogeneous devices with diverse capabilities, the data is progressively encoded in L quality layers, *i.e.*, a base layer and several enhancement layers. Such an encoding enables receivers with limited capabilities to receive the base layer, while receivers with more resources decode the data in higher quality. The data from the base layer is the most important, followed by the data of the successive enhancement layers that offer incrementally finer levels of quality. A receiver can decode data from an enhancement layer only when it has decoded all the lower layers.

The data is further segmented in generations $G_0, G_1, \ldots, G_k$, which are groups of time-constrained data (*e.g.*, images in video case). We consider that the data of the lth layer of a generation is packetized into α_l packets. Therefore, the first k layers consist of $\beta_k = \sum_{l=1}^{k} \alpha_l$ packets. The distortion reduction associated with the decoding of the data layer l of a generation G_m is denoted as $\delta_{l,m}$, while the cumulative distortion reduction after decoding the first l layers is $\Delta_{l,m} = \sum_{i=1}^{l} \delta_{l,m}$.

Each generation has in total β_L packets and is associated with a decoding delay deadline. The deadline of the nth generation is written as

$$D_n = D_0 + nD_G, \tag{1}$$

where D_G is the duration of the rendering of the information contained in a generation. We focus on the case where D_G is identical for all generations. The parameter D_0 in (1) represents the playback delay and depends on the application's requirements.

C. Prioritized Random Linear Coding

We consider that the servers encode the data with PRLC codes [7] in order to respect the unequal levels of importance in the data layers. The encoding of the data permits to deal with losses in the network; at the same time, it avoids the need for precise coordination among the multiple servers transmitting data to the same receiver. All the PRLC operations are performed in a finite field $\mathbb{F}_q$ of size q. We restrict the coding operations to packets that belong to the same generation. Then, for a generation of β_L packets, an overhead of length β_L symbols is appended to each packet that belongs to that particular generation, in order to convey the coding coefficients to the receiver. Although some bandwidth is spent for transmitting the coding overhead, the use of PRLC has several appealing features. Specifically, PRLC codes simplify scheduling decisions, improve the robustness of the communication and perform close to Maximum Distance Separable (MDS) codes. The coding overhead is not a real limitation in our system as there are techniques to drastically reduce its size [36], [37].

Similarly to the Prioritized Randomized Network Coding method presented in [8], [34] we consider that a packet of class l is a random linear combination of data from the first l video layers. Finally, the type of each packet is described by a tuple (G_m, l), where G_m and l stand for the generation m and the class l, respectively. Upon the arrival of the PRLC packets at the receiver, the packets are sorted according to their generation G_m. These packets are examined in terms of their innovation (*i.e.*, whether they bring innovative information with respect to the information that is already available at the receiver). If they are innovative, they are stored in the decoding matrix $\boldsymbol{W}_m$, otherwise they are dropped. Each row of the decoding matrix $\boldsymbol{W}_m$ corresponds to an innovative packet from generation G_m. This row contains the coding coefficients contained in the header of the received packet. Although the packets do not necessarily arrive in order, we assume that the rows of the $\boldsymbol{W}_m$ matrix are sorted in an ascending class order. Therefore, packets from the first class are placed on the top of the matrix, followed by packets from the second class, *etc*. The matrix $\boldsymbol{W}_m$ is divided into L submatrices $\boldsymbol{F}_{lm}$, where $\boldsymbol{F}_{lm}$ is a $n_{lm} \times \beta_L$ matrix and n_{lm} denotes the number of received packets of class l from generation G_m. Recall that β_l is the number of packets from the first l layers of a generation. This is illustrated in Fig. 1.

Within the matrix $\boldsymbol{W}_m$ one can form submatrices $\boldsymbol{R}_{lm}$ that are the unions of the first l submatrices $\boldsymbol{F}_{1m}, \ldots, \boldsymbol{F}_{lm}$. For these submatrices, it holds that $0 \leq r_{lm} \leq \beta_l$, where $r_{lm} = \mathrm{rank}(\boldsymbol{R}_{lm})$, as the number of innovative packets from class l in generation G_m cannot exceed the cumulative number of source packets from the first l layers. When $r_{lm} = \beta_l$, a receiver can successfully decode the first l layers of the generation G_m by means of Gaussian elimination, for example.

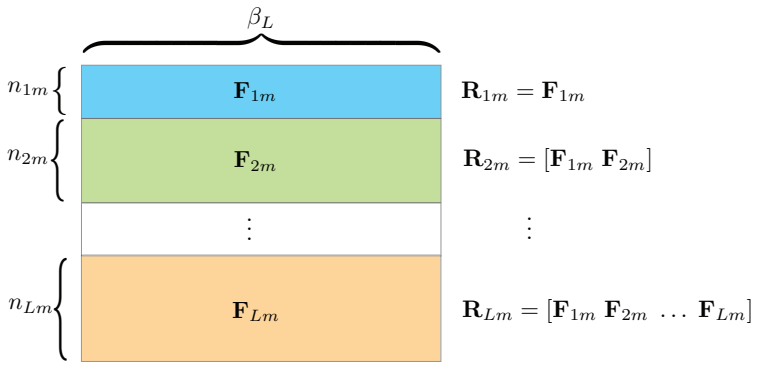


Fig. 1. Decoding matrix $\boldsymbol{W}_m$ of generation G_m constructed at the receivers. It consists of submatrices $\boldsymbol{F}_{im}$ that contain the innovative data from the ith video layer of the generation G_m. Matrices $\boldsymbol{R}_{lm}$ are the unions of the first l submatrices $\boldsymbol{F}_{1m}, \ldots, \boldsymbol{F}_{lm}$, whose number of rows n_{im} denotes the number of innovative packets from the layer i of the generation G_m. Decoding of *only* the first l layers of the generation G_m is possible when $\sum_{i=1}^{l} n_{im} = \beta_l$ and $\sum_{j=1}^{i} n_{jm} \leq \beta_i, \forall i > l$.

The rank vector $\boldsymbol{r}_m = (r_{1m}, \ldots, r_{Lm})$ for the submatrices $\boldsymbol{R}_{lm}$ of the generation G_m, completely defines the buffer state of the receiver. For notational convenience we define hereafter the buffer state of a receiver i as $B_i = \{\boldsymbol{r}_m\}$, *i.e.*, B_i is the collection of the $\boldsymbol{r}_m$ vectors (one per generation G_m) in the buffer of the ith receiver. A similar buffer state variable is used to describe the data available at servers.

D. Receiver-driven delivery optimization

In our receiver-driven system multiple servers simultaneously transmit data from different generations to the receiver. When the receiver estimates that it cannot receive all the data of a particular generation at a given rate by its decoding deadline, it either decides to request lower data rate, hence lower data quality or to skip the particular generation and proceed with the next one. With their decisions, the receivers naturally attempt to obtain high data quality with small quality fluctuations. This happens as the transmission of packets from future generations leads to an increased probability of decoding these generations before their playback delay. Thus, the probability that a receiver has data that cannot be decoded is minimized. Essentially, our scheme works similarly to prefetching algorithms.

The receiver periodically computes the optimal coding and scheduling strategies taking into account the local statistics, *i.e.*, the channel loss rate ϵ_{ij} and the capacity f_{ij} of each link connecting the receiver to the available servers.[1] It also considers information about the available data at each server (*i.e.*, the buffer maps $B_i[n]$ of the server i at time $t = t_n$). This information becomes available to the receivers with the form of messages $\mathcal{I}_i = (B_i[n], \epsilon_{ij}, f_{ij}, \eta_{ij})$ that are transmitted from the servers $i \in P_j$ to the

[1]In wireless channels, this information can be obtained by measuring the channel statistics in real-time.

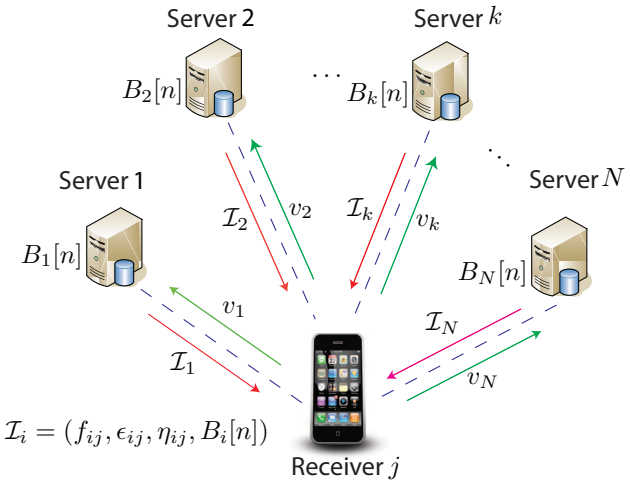


Fig. 2. Data delivery system under consideration. Multiple servers that have layered data simultaneously transmit packets to a single receiver (node j). Each link (i, j) is characterized by the transmission rate f_{ij}, the loss rate ϵ_{ij} and the transmission delay η_{ij}. The receiver j computes the request allocation vectors v_i based on the $\mathcal{I}_j = (B_j[n], \epsilon_{ji}, f_{ji}, \eta_{ji})$

receiver j. The size of these messages depends on the number of considered generations and the number of layers. Hence, when the channel parameters $(\epsilon_{ij}, f_{ij}, \eta_{ij})$ can be represented with one byte, the coding operations are in $\mathbb{F}_{256}$ and in each decision interval decisions are made for two generations and two layers (with less than 256 packets), the length of the messages $\mathcal{I}_i$ is equal to seven bytes. These messages are transmitted once per decision interval.[2]

The receiver j takes into account these messages $\mathcal{I}_i$ to calculate a request allocation vector $v_j = [v_{ij},\ \forall i \in P_j]$, which is sent to all the servers. The request allocation vector for the ith server is written as $v_{ij} = [v_{11}^{ij}, \ldots, v_{1L}^{ij}, \ldots, v_{l1}^{ij}, \ldots, v_{lL}^{ij}, \ldots, v_{m1}^{ij}, \ldots, v_{mL}^{ij}]$, where v_{lk}^{ij} denotes the number of packets of class k in generation G_l that receiver j requests from server i. The request allocation vector v_j is computed at the receiver in order to maximize the cumulative expected distortion reduction in the nth time slot for all the generations that have not expired. The overall data delivery system is illustrated in Fig. 2.

IV. MDP FORMULATION

In this section, we propose a Markov Decision Process (MDP) formulation to find the request allocation vector v_j at receiver j that maximizes the cumulative expected distortion reduction given the messages $\mathcal{I}_i$ that are received from the servers $i \in P_j$. The receiver, also called agent in our formulation, periodically determines the optimal request allocation policy π^* every T time slots. Hence, the first decision is made

[2]The messages $\mathcal{I}_i$ can be transmitted over the same channel or over a separate, more reliable, channel.

at $t = t_0$, while the nth decision is at $t_n = t_0 + nT$. We consider that all receivers then act independently in our system. In the rest of the paper, we drop the index j of the receiver and hence $P = P_j$, $v = v_j$, $\epsilon_{ij} = \epsilon_i$, $f_{ij} = f_i$, $\eta_{ij} = \eta_i$ for easy notation. The actions, states and the rewards of the proposed MDP framework are now presented in detail for one receiver.

A. States

The state of the receiver at time t_n is completely characterized by its own buffer map $B[n]$ and the buffer maps of the servers ($B_k[n],\ \forall k \in P$). Therefore, the state of the receiver is defined as

$$S[n] = \{B_k[n]\ (\forall k \in P),\ B[n],\ t_n\} \tag{2}$$

For infinite length data, the states space is enormous. It becomes smaller when the states space is quantized by considering that time is discrete and that not all the generations are active (*i.e.*, decoding deadlines expire during the transmission).

In order to find which generations should be included in the states definition, at time instant t the generations are relabeled based on their expiration time (urgency). The relabeling is done as $G_m \xrightarrow{t} G^*_{m-\mu(t_n)}$, where $\mu(t_n)$ is the index of the generation with the closest decoding deadline at time t_n,

$$\mu(t_n) = \underset{m:\ t_n < D_m}{\operatorname{argmin}}\ D_m \tag{3}$$

Therefore, the remaining time $\tau(n)$ until the deadline of the most urgent generation $G_{\mu(t_n)}$ is denoted as $\tau(n) = D_{\mu(t_n)} - t_n$. Clearly, $\tau(n)$ satisfies $1 \leq \tau(n) \leq D_G$, with D_G the duration of a generation.

Hence, the state definition of (2) can be re-written as follows:

$$S[n] = \{B_k[n]\ (\forall k \in P),\ B[n],\ \tau[n]\}\,, \tag{4}$$

We now analyze briefly the size of the states space $\mathcal{S}$. We note that for any generation G_m, the number of valid vectors (states) is finite, constant and dependent on the source data, *i.e.*, the number of packets per layer, α_l. Therefore, the overall number of states $|\mathcal{S}|$ is given by

$$|\mathcal{S}| = \sum_{v_1=0}^{\beta_1} \sum_{u_2=0}^{\beta_2 - u_1} \cdots \sum_{u_{L-1}=0}^{\beta_{L-1} - \sum_{k=1}^{L-2} u_k} \left(\beta_L - \sum_{k=1}^{L-1} u_k \right). \tag{5}$$

From (5), we conclude that the states space increases exponentially with the number of packets per layer. The number of generations that might be available in the buffer of the receiver at time t_n is

determined by the capacity of its incoming links and the decoding deadlines of the generations. We note that we only consider information from the generations that have not expired nor been decoded in the states definition.

B. Actions

An action A_k is defined as the request allocation vector (v_k) transmitted from the receiver to the server k. This vector determines, for the server k, the class and the generation of the PRLC encoded packets to be sent. Then, an action A consists of actions $A_k,\ \forall k\ \in\ P$ and is written as

$$A = [A_k,\ \forall k\ \in\ P] \tag{6}$$

The length of the request vector v_k that corresponds to the action A_k is equal to the overall number of layers $L \cdot N_m$, where N_m stands for the number of generations that have not expired and L the number of layers per generation.[3] It is assumed that these vectors arrive at the servers before the next decision interval. Based on v_k, each server k generates packets by PLRC coding and transmits them to the receiver. The lth packet in the vector is scheduled for transmission at $t = t_n + (l-1)f_k^{-1}$, where f_k^{-1} is the time interval between the transmission of two consecutive packets on the link from the server k to the receiver.

We now study briefly the size of the action space $\mathcal{A}$. Now, if $\mathcal{A}_k$ is the action space of the server k, it holds $\mathcal{A} = \times_{k\in P}\mathcal{A}_k$. Given a state $S[n]$, and that the receiver might request packets that belong to M_k different classes from the server k, hence, the size of the action space $|\mathcal{A}|$ for a given state $S[n]$ is

$$|\mathcal{A}| = \prod_{k\in P} \binom{N_k + M_k - 1}{N_k} \tag{7}$$

where N_k is the number of requested packets from server k.

[3]We assume that all the generations have the same number of layers

C. State transition Probability

The transition probability from an arbitrary state-action pair to another arbitrary state is written as

$$\begin{aligned}
&\Pr\left(S[n+1] \mid S[n], A[n]\right) \\
&= \Pr\left(B_k[n+1]\ (\forall k \in P), B[n+1], \tau[n+1] \mid B_k[n]\ (\forall k \in P), B[n], \tau[n], A[n]\right) \\
&= \Pr\left(B_k[n+1]\ (\forall k \in P), B[n+1] \,\middle|\, B_k[n]\ (\forall k \in P), B[n], \tau[n], A[n]\right) \\
&= \Pr\left(B[n+1] \mid \mathcal{I}[n]\right) \times \Pr\left(B_k[n+1]\ (\forall k \in P) \mid \mathcal{I}[n], B[n+1]\right)
\end{aligned} \tag{8}$$

where $\mathcal{I}[n] = \{B_k[n]\ (\forall k \in P), B[n], \tau[n], A[n]\}$ is the complete local information set at t_n. For the sake of simplicity, we have not included the channel parameters f_k, ϵ_k, η_k in (8). The first conditional probability in (8) represents the belief of the receiver for the buffer state while the second is the belief of the receiver for the buffer states of the parents. Specifically, $\Pr(B[n+1] \mid \mathcal{I}[n])$ denotes the probability for the receiver to be in the buffer state $B[n+1]$ at the next decision time. Since the action $A[n]$ is given, the types of the requested packets and their transmission order is known. Using the remaining time information extracted from $\tau[n]$, along with the channel capacity $f_k, \forall k \in P$, the packet loss probabilities $(\epsilon_k, \forall k \in P)$ and the delay distributions $(\eta_k, \forall k \in P)$ in the incoming links, the probability that exactly z_{ml} packets arrive in time (before their corresponding deadlines) for any packet type $(G_m, l) \in A[n]$ can be calculated. Then, for a given z_{ml} value, the problem reduces to a slightly more complex version of the coupon collector problem, as we have different classes of packets. The transition probabilities are calculated using the propositions presented in the Appendix A in [38] in which take into consideration the maximum number of innovative packets per packet type (G_m, l) and the size of the finite field $\mathbb{F}_q$. Regarding the second term in (8), we assume that it is equal to one as the content of the servers is known to the receiver. This is due to the fact that the servers and the receivers know when the data from each generation becomes available to the servers.

D. Rewards

Since the successful decoding of a layer is associated with a reduction in the distortion of the decoded data, the expected reward $J(\cdot)$ at time slot n is expressed as the resulting cumulative distortion reduction for all active generations after the completion of all actions decided at time t_{n-1}. The reward $J(S[n], A[n])$ is a non-negative function of the current state-action pair.

Let $\mathcal{G}[n] = \{G_m : t_n < D_m \leq t_{n+1}\}$ be the set of generations that have deadlines in $(t_n, t_{n+1}]$. If $\mathcal{G}[n]$ is non-empty, $\tau[n]$ is smaller than T and then the distortion reduction might be positive. When $\mathcal{G}[n] \neq \emptyset$,

the expected distortion reduction is $\Delta_{\ell,m} = \sum_{k=1}^{\ell} \delta_{k,m}$ (with $\Delta_{0,m} = 0$), when no layers have been decoded for the generation G_m in the previous decision interval. Then, the reward function becomes

$$J(S[n], A[n]) = \sum_{G_m \in \mathcal{G}[n]} \sum_{\ell=0}^{L} \mathbb{P}_{\ell,m}(S[n], A[n]) \Delta_{\ell,m}, \tag{9}$$

where $\mathbb{P}_{\ell,m}(S[n], A[n])$ denotes the probability that the receiver recovers *only* the first ℓ layers of the generation G_m, that is, $r_{\ell m} = \beta_\ell$ and $r_{lm} < \beta_l, \forall l > \ell$, given $S[n]$ and $A[n]$. This probability can be calculated by considering the subspaces formed by the received packets from different classes. Let us denote by $V_{\ell,m}$ the subspace spanned by the source packets from the first ℓ layers of the generation G_m. Furthermore, let $B^i_{\ell,m}[n]$ denote the subspace spanned by the innovative packets from the class ℓ of the generation G_m that are transmitted by the node i to the receiver.[4] To compute $\mathbb{P}_{\ell,m}(S[n], A[n])$, we need to consider the events in which the received packets span $V_{\ell,m}$, but cannot span $V_{r,m}$, $\forall r > \ell$. Thus,

$$\mathbb{P}_{\ell,m}(S[n], A[n]) = \Pr\left(\bigcup_{i \in P'} \bigcup_{k=1}^{\ell} B^i_{k,m}[n] = V_{\ell,m}, \bigcup_{i \in P'} \bigcup_{k=\ell+1}^{r} B^i_{k,m}[n] \subset V_{r,m}, r \in \{ \ell+1, ..., L \} \right) \tag{10}$$

where P' is the set comprised of the receiver and its parental servers.

In order to calculate the probability $\mathbb{P}_{\ell,m}(S[n], A[n])$, we need to consider the probability that the PRLC coded packets received from the servers are innovative. Non innovative packets (*i.e.*, packets that can be expressed as linear combinations of the existing in the buffer packets) can be received as PRLC selects the coding coefficients uniformly at random. These packets can be discarded as only innovative packets bring benefits to the receiver. Based on those considerations, the probability $\mathbb{P}_{\ell,m}$ can be computed by considering the set of packets that are available to the servers and the receiver (we need to know how many packets we require to decode a layer) and the probability to receive as many packets are required for rendering decodable the packets that belong to class ℓ of generation m. Details are provided in the Appendix A in [38] and in particular in Proposition 3.

In order to avoid myopic behaviors and greedy decisions, the MDP framework considers future rewards in the optimization of the rate allocation vector. Hence, we consider the expected rewards for future

[4]This is a slight abuse of the notation $B^i_{k,m}[n]$ which originally denotes the subspace spanned by the innovative packets from the class ℓ of the generation G_m that are stored at server i.

generations as

$$J_k(S[n], A[n]) = \sum_{G_m \in \mathcal{G}[n']} \sum_{\ell=0}^{L} \mathbb{P}_{\ell,m}(S[n], A[n]) \Delta_{\ell,m}, \tag{11}$$

with $n^{'} = n + k$ and $k > 0$.

The rewards in (11) are discounted by a factor $\gamma \in [0, 1]$ which is known as "discount rate" and controls the contribution of the future rewards (delayed rewards). Hence, the reward of actions $A[n]$ for generations G_{n+k} is weighted by γ^k. If $\gamma = 0$, the decisions are myopic. In contrast, if $\gamma = 1$ the agent becomes farsighted and is interested only in future rewards. This is further discussed in the next section.

E. Optimization Problem

With the above MDP framework, we can now define the problem that is solved for the optimal rate vector selection. We want to maximize the total expected discounted reward (value function), which is the discounted sum of the received rewards starting from state $S[0]$ and following a decision policy π that defines the action $A[n]$ at state $S[n]$, *i.e.*, $A[n] = \pi(S[n])$. Formally, the value function is given by

$$V^{\pi}(S[n]) = \sum_{k=0}^{\infty} \gamma^k J_k(S[n], \pi) \tag{12}$$

The objective of our optimization problem is to maximize the expected distortion reduction:

$$\max_{\pi} V^{\pi}(S[n]) \tag{13}$$

subject to the following conditions

1) $v_{jl}^{i} > 0$, if for server i, we have $r_{jl}^{i} > 0$ (r_{jl}^{i} is the rank of the packets from the jth generation and lth layer at server i)
2) $v_{jl}^{i} = 0$, if $r_{jl} = \beta_l$
3) $\mathcal{G}[n] = \{G_m : t_n < D_m \leq t_{n+1}\}$

The first condition means that the receiver requests packets that belong to a given class only if the server has packets of this class or if it can generate them by PRLC encoding. The second condition implies that, once a receiver decodes the layer l, it will not request packets from the first l layers as it has already β_l innovative packets from these layers. Hence, any further received packet from classes $l^{'} \leq l$ cannot bring innovative information. Finally, the last condition determines the set of active generations.

The above MDP can be solved by the value iteration method [39], which is a form of dynamic programming algorithm. This method updates the values of $V(\cdot)$ function according to the Bellman

equation

$$V_{k+1}(S[n]) = \max_{A[n]} \left\{ \sum_{S[n+1]} \left(\Pr\left(S[n+1] \mid S[n], A[n]\right) J_{k+1}(S[n], A[n]) + \gamma V_k(S[n+1]) \right) \right\}, \quad (14)$$

where $V_k(\cdot)$ is the value function at the kth iteration and $J_{k+1}(S[n], A[n])$ is given in (9) and the transition probability $\Pr(S[n+1]|S[n], A[n])$ is computed as in Section IV-C. This process is repeated until $\left|V_{k+1}(S[n]) - V_k S[n]\right|$, $\forall S[n] \in \mathcal{S}$ is smaller than a threshold value. If the algorithm terminates at the k iteration, the derived policy π is given by

$$\pi(S[n]) = \arg\max_{A[n]} \left\{ \sum_{S[n+1]} \left(\Pr\left(S[n+1] \mid S[n], A[n]\right) J_{k+1}(S[n], A[n]) + \gamma V_k(S[n+1]) \right) \right\} \forall S[n] \in \mathcal{S}. \quad (15)$$

More details can be found in Section 4.4 of [28].

V. Q-Learning solution

MDP offers an interesting framework to formalize the scheduling and coding decision processes in multi-source streaming, but it unfortunately requires real time computation of the transition probabilities and rewards functions as shown in (14). It also necessitates significant storage space, which can be as large as $|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$ for the transition probability function. The cost of computing and storing these functions becomes very large as the states space increases exponentially with the number of packets per data layer. In order to avoid the computation of these functions beforehand, we propose alternative solutions to (13) based on model-free reinforcement learning approaches [28] and more specifically on the Q-learning method [17], which is a TD (Temporal Difference) control algorithm. These methods could further learn the model in an environment and then update it online in another. Although optimality is not guaranteed with the Q-learning method, the resulting solutions are close to the optimal ones as long as the method runs for a sufficiently large number of iterations. We describe below the Q-learning method, and more specifically the virtual experience algorithm that is used in our solution.

A. Q-Learning Algorithm

Similarly to the value iteration algorithm, the Q-learning aims to find the optimal policy $\pi^\star$ that maximizes the expected reward given in (11). It achieves that with no need to precompute the transition probabilities and rewards functions and hence can be applied online. Specifically, the Q-learning algorithm [17] computes a $Q(S[n], A[n])$ function for deciding the $\pi^\star$. This is a function of both the states and

the actions and has size equal to $|\mathcal{S}| \times |\mathcal{A}|$. Q-learning learns the optimal actions to be taken based on experience tuples of the form $(S[n], A[n], J(S[n], A[n]), S[n+1])$.

In more details the Q-learning algorithm works as follows. Initially, all the entries of the $\mathbf{Q}$ function are set to zero. At the nth decision interval (when the algorithm is applied online, the decision interval may coincide with the time t_n of the delivery process), the next state $S[n+1]$ is determined by the current state $S[n]$ and an action $A[n]$ that is selected with probability

$$\frac{e^{Q(S[n],A[n])/\Theta(n)}}{\sum\limits_{A[n+1]\in A} e^{Q(S[n],A[n+1])/\Theta(n)}}, \tag{16}$$

where $\Theta(n)$ is the temperature value at the nth decision interval. For a given action $A[n]$, one can then find the next state $S[n+1]$ and calculate the $J(S[n], A[n])$. The Q-learning algorithm then updates the function value $Q(S[n], A[n])$ as follows

$$Q(S[n], A[n]) \leftarrow (1-\lambda_n)Q(S[n], A[n]) + \lambda_n(J(S[n], A[n]) + \gamma \max_{A[n+1]} Q(S[n+1], A[n+1])), \tag{17}$$

where λ_n is the learning rate at time t_n that is given by

$$\lambda_n = \frac{1}{1 + N_v(S[n], A[n])},$$

where $N_v(S[n], A[n])$ is the number of visits of $(S[n], A[n])$ action-state pair up to the nth iteration (time t_n). For time t_1, the parameter λ_1 is set to one.

The above procedure is repeated $\mathcal{N}$ times. The value of $\mathcal{N}$ depends on the considered application. At time t_n, the policy π ($A[n] = \pi(S[n])$) can be found as

$$\pi(S[n]) = \max_{A[n]} Q(S[n], A[n]),\ \forall S[n] \in \mathcal{S} \tag{18}$$

As $\mathcal{N}$ goes to infinity, the learned $Q(S[n], A[n])$ function approximates the optimal $Q^\star(S[n], A[n])$ and the derived policy is identical to the optimal policy $\pi^\star$. This is true when the transition probability is stationary, when all the state-action pairs are visited infinitely often and when λ_n satisfies the stochastic approximation conditions, *i.e.*, $\sum_{n=0}^{\infty} \lambda_n = \infty$ and $\sum_{n=0}^{\infty} \lambda_n^2 < \infty$ [18]. The accuracy of the approximation depends on λ_n and $\Theta(n)$ that will be discussed later and the number of Q-learning iterations (episodes) $\mathcal{N}$. In practice, the optimal $Q^\star(S[n], A[n])$ can generally be found for $\mathcal{N} \ll \infty$.

A critical parameter for the convergence of the Q-learning algorithm is the temperature value $\Theta(n)$ used in (16). In the beginning, the temperature value $\Theta(n)$ is typically set to a high value in order to permit consistent exploration of the states-actions space. This is called the exploration phase, as all

the pairs $(S[n], A[n])$ have almost equal probability of being selected due to the high value of $\Theta(n)$. With time, the temperature value decreases and the knowledge acquired during the exploration phase is exploited, *i.e.*, the values of the $Q(\cdot)$ function are not equal and the probabilities given by (16) become different. The temperature generally decreases following the model in [40], *i.e.*,

$$\Theta(n) = \Theta_{\min} + \varphi \cdot (\Theta(n-1) - \Theta_{\min}), \tag{19}$$

where n is the index of the decision interval (the index of the current episode), $\Theta(0) = \Theta_{\max}$ is the initial value of the temperature, Θ_{min} is the minimum temperature and φ is a parameter that controls the exploration/exploitation rate. The value of φ is found by experimentation and takes values in the range $(0, 1]$.

B. Q-Learning with Virtual Experience

Although Q-learning eliminates the need to precompute the transition probabilities and rewards matrices that is required by the value iteration algorithm (14), it is characterized by a slow convergence rate, which may limit its application in practical settings. Hence, in order to improve the convergence rate, we propose to use in our problem a variant of the Q-learning algorithm that is hereafter referred to as "Q-learning VE", where VE stands for virtual experience. Our method is inspired by [18] where the concept of virtual experience has been first proposed for improved convergence of the PDS-learning algorithm (a variant of Q-learning).

Differently from the Q-learning algorithm, where, in each decision interval, a single state-action pair $Q(S[n], A[n])$ is updated, the "Q-learning VE" simultaneously updates multiple state-action pairs $Q(S[n], A[n])$ that are statistically equivalent (virtual states-actions). In particular pair $(\bar{S}[n], \bar{A}[n])$ is statistically equivalent to $(S[n], A[n])$ if the transition probability $\Pr(S[n+1] \big| S[n], A[n]) = \Pr(\bar{S}[n+1] \big| \bar{S}[n], \bar{A}[n])$ and the reward $J(\bar{S}[n], \bar{A}[n])$ can be determined from $J(S[n], A[n])$.

In our specific problem, when an action $A[n]$ is selected, a request allocation vector v is sent to the receivers. Since the links connecting the servers with the receiver are lossy, the vector of packets $\bar{v}$ that arrive at the receiver may be different from the request allocation vector v that is determined by the action $A[n]$. The probability the receiver to observe a vector $\bar{v}$ while the request allocation vector was v is equal to

$$\prod_{i=1}^{m} \prod_{j=1}^{L} \prod_{l \in \mathcal{G}_n} \binom{v_{lj}^i}{\bar{v}_{lj}^i} (\epsilon_i)^{v_{lj}^i - \bar{v}_{lj}^i} (1-\epsilon_i)^{\bar{v}_{lj}^i}, \tag{20}$$

where L is the number of data layers per generation, m is the number of receivers, $\mathcal{G}_n$ is the set of active

Algorithm 1 Q-learning VE algorithm

1: Initialize: $\mathbf{Q} = 0$, $\mathbf{N_v} = 0$, $\Theta(0) = \Theta_{\max}$, $n = 1$, and U
2: Observe $S[n]$
3: **repeat**
4: $N_v(S[n], A[n]) = N_v(S[n], A[n]) + 1$
5: $\lambda_n = \frac{1}{1+N_v(S[n],A[n])}$
6: $\Theta(n) = \Theta_{\min} + \varphi \cdot (\Theta(n-1) - \Theta_{\min})$
7: Select action $A[n]$ with probability $\frac{e^{Q(S[n],A[n])/\Theta(n)}}{\sum\limits_{A[n+1]\in A} e^{Q(S[n],A[n+1])/\Theta(n)}}$
8: Sample the noise function (20)
9: Find the next state $S[n+1]$ considering $S[n]$, the action A[n] (step 7) and the noise (step 8)
10: Calculate the discounted reward $J(S[n], A[n])$ as in (11)
11: $Q(S[n], A[n]) \leftarrow (1-\lambda_n)Q(S[n], A[n]) + \lambda_n(J(S[n], A[n]) + \gamma \max\limits_{A[n+1]} Q(S[n+1], A[n+1]))$
12: **if** $\mod(n, U) = 0$ **then**
13: **for** $\bar{S}[n] \in \mathcal{S}$ **do**
14: **for** $\bar{A}[n] \in \mathcal{A}$ **do**
15: **if** $\Pr(S[n+1] \big| S[n], A[n]) = \Pr(\bar{S}[n+1] \big| \bar{S}[n], \bar{A}[n])$, $J(S[n], A[n]) = J(\bar{S}[n], \bar{A}[n])$ and $\bar{S}[n+1] = S[n+1]$ **then**
16: $N_v(\bar{S}[n], \bar{A}[n]) = N_v(\bar{S}[n], \bar{A}[n]) + 1$
17: $\lambda_n = \frac{1}{1+N_v(\bar{S}[n],\bar{A}[n])}$
18: $Q(\bar{S}[n], \bar{A}[n]) \leftarrow (1-\lambda_n)Q(\bar{S}[n], \bar{A}[n]) + \lambda_n(J(\bar{S}[n], \bar{A}[n]) + \gamma \max\limits_{A[n+1]} Q(S[n+1], A[n+1]))$
19: **end if**
20: **end for**
21: **end for**
22: **end if**
23: $S[n] \leftarrow S[n+1]$
24: $n = n + 1$
25: **until** $n \leq \mathcal{N}$
26: $\pi(S[n]) = \max_{A[n]} Q(S[n], A[n]),\ \forall S[n] \in \mathcal{S}$

generations at time t_n and ϵ_i is loss probability on the link connecting the ith sender with the receiver. Therefore, given an action $A[n]$, *i.e.* a request allocation vector v, and (20), we can find the next state $S[n+1]$ and calculate $J(S[n], A[n])$ as in (11).

The convergence is improved with the Q-learning VE, but we should mention that it comes with increased computational complexity as we perform multiple updates of the (17) for all equivalent states-actions pairs. The frequency of the updates U (every U iterations the algorithm updates all statistically equivalent virtual states-actions), hence constitutes a tradeoff between convergence speed and computational complexity. In each decision interval, we update all the $Q(\bar{S}[n], \bar{A}[n])$ that result in the same reward $J(S[n], A[n])$, *i.e.*, the same expected distortion reduction, and lead to the same next state $S[n+1]$

instead of updating only a single state-action pair $Q(S[n], A[n])$ as is done by the Q-learning algorithm. Moreover, we assume that all the request allocation vectors v (that are determined by the $A[n]$) are affected by the same error pattern. This is due to the fact that the channel conditions are independent from the current state $S[n]$ and the action $A[n]$, *i.e.*, the probability of packet loss depends only on the channel and not to the packet class and generation. We should note that in our system the link capacity is always fully exploited and the receivers never send packet replicas. This becomes possible as PRLC belong to the rateless codes family.

The proposed Q-learning-VE algorithm is finally summarized in Algorithm 1.

C. Complexity of the Q-learning methods

In Table I, we report the action selection and the learning update complexity for the Q-learning and Q-learning VE algorithms. In Q-learning only one state-action pair $(S[n], A[n])$ is updated in each decision interval n; both the action selection and learning update complexities are equal to $\mathcal{O}(|\mathcal{A}|)$ [18]. As expected, the Q-learning VE method has higher complexity than Q-learning as all the equivalent virtual state-action pairs are simultaneously updated in each decision interval. Hence, the action selection of the Q-learning VE is $\mathcal{O}(|\mathcal{S}^{'}| \times |\mathcal{A}^{'}|)$ and the update learning complexity is also $\mathcal{O}(|\mathcal{S}^{'}| \times |\mathcal{A}^{'}|)$, where $|\mathcal{S}^{'}| < |\mathcal{S}|$ and $|\mathcal{A}^{'}| < |\mathcal{A}|$. However, we should note that the multiple state-action pairs can be updated in parallel. In such case, the complexity of the Q-learning VE becomes identical to the one of Q-learning.

TABLE I
COMPLEXITY COMPARISON OF THE Q-LEARNING METHODS PER DECISION INTERVAL. IT IS $|\mathcal{S}^{'}| < |\mathcal{S}|$ AND $|\mathcal{A}^{'}| < |\mathcal{A}|$.

	Action selection complexity	Learning update complexity
Q-learning	$\mathcal{O}(\|\mathcal{A}\|)$	$\mathcal{O}(\|\mathcal{A}\|)$
Q-learning VE	$\mathcal{O}(\|\mathcal{S}^{'}\| \times \|\mathcal{A}\|)$	$\mathcal{O}(\|\mathcal{S}^{'}\| \times \|\mathcal{A}\|)$

VI. SIMULATION RESULTS

In this section, the proposed MDP and Q-learning solutions are examined for transmission of layered data from one or multiple servers. First, we describe the settings of a transmission scenario with one server. Then, we evaluate the performance of the optimized policies in terms of quality in video transmission from one server and eventually two servers.

A. Data delivery from one server: setup

We first consider a scenario where the receiver obtains layered data from the server 1 at a rate equal to 1 *packet/time slot*. The nth generation of the layered data stream is available at the server at $t = D_G n = 5n$ and the playback delay D_0 is 10 time slots. Therefore, the decoding deadlines for the nth generation is $D_n = 10 + 5n$. The receiver makes decisions at $t = t_0 = 5$ and then every $T = 5$ time slots such that $\tau[n] = 5$ for all the states $S[n]$.

In this simple scenario, the state of the server's buffer $B_1[n]$ is deterministic for all the time slots since the data generations are only available progressively at the server. As a result, we can remove $\tau[n]$ and $B_1[n]$ from the states definition in (4). In order to further simplify the analysis, we remove the packet transmission delays and set $\eta_1 = 0$. Since the decisions are made at $(t_1, t_2, \dots)$ and since these time instants coincide with the decoding deadlines $(D_0, D_1, \dots)$, at time t_n there is only the decoding matrix $\boldsymbol{W}_n$ (recall that this matrix contains the innovative packets from generation n) in the buffer of the receiver, which should be played out immediately. The matrices that correspond to $\boldsymbol{W}_{n-l},\ l < n$ have been removed as their corresponding deadlines $D_{n-1}, D_{n-2}, \dots$ have already passed, while the $\boldsymbol{W}_{n+1}, \boldsymbol{W}_{n+2}, \dots$ are empty as at time $t + n$ the layered data for generations $n + l,\ l > n$ is not available at servers. Hence, the states definition contains only the current generation, *i.e.*, $\mathcal{G}[n] = \{G_n\}$ and the receiver might only gain a reward (observe a distortion reduction) by decoding G_n at time t_n.

The actions are defined as $A[n] = A_1[n]$, as the layered data is transmitted from only one server in the present case. Since $t_n < D_n$ in every decision time, there are exactly two generations from which packets can be requested, namely G_n and G_{n+1} in order to allow foresighted behavior. These packets become available at the server at t_{n-1} and t_n, respectively. The considered generations depend on the channel capacity and the decoding deadlines. When the channel capacity is larger and the decoding deadlines are more distant, the actions can also involve packets from generations $G_{n+l},\ l > 1$, which results in larger actions space.

In the above settings we further have,

$$\Pr\big(S[n+1] \mid S[n], A[n]\big) = \Pr\big(B[n+1] \mid B[n], A[n]\big) \tag{21}$$

The transition probability from a state $S[n]$ to $S[n+1]$ given an action $A[n]$ (the request allocation vector that is sent from the receiver to the server) can be computed as discussed in Section IV-C, where the states are described by the buffer maps $B[n]$ and $B[n+1]$ of the receiver at the decision intervals n and $n + 1$.

B. Transmission of scalable video data from one server

In this section, we evaluate the policies of the MDP algorithm presented in Section IV-E which we call it hereafter "Model-MDP" (this scheme considers the actual loss rates for deciding the optimal scheduling policies) and the Q-Learning algorithms shown in Section V in an illustrative scenario for transmission of scalable video data. The video data is encoded in two data layers, namely a base and one enhancement layer. The parameters of the source data are reported in Table II. All the generations have constant size, *i.e.*, $D_G = 5$.

TABLE II
PARAMETERS OF A GENERATION WITH TWO DATA LAYERS. THE PARAMETERS α_i, β_i, δ_i STAND FOR THE NUMBER OF PACKETS IN THE iTH LAYER, THE CUMULATIVE NUMBER OF PACKETS IN THE FIRST i LAYERS AND THE DISTORTION REDUCTION AFTER THE SUCCESSFUL DECODING OF THE iTH LAYER, RESPECTIVELY.

	α_i	β_i	δ_i
Base Layer	3	3	11
Enhancement Layer	2	5	9

Using the settings presented in Section VI-A, the states are described by only the receiver buffer, *i.e.*, $S[n] = \boldsymbol{r}_n$. By evaluating the valid rate allocation vectors in (5), we see that $\boldsymbol{r}_n$ can take 18 different values in this simple scenario, which are reported in Table III.

TABLE III
STATES SPACE. A STATE IS DESCRIBED BY A TWO-TUPLE WHICH HAS AS ENTRIES THE NUMBER OF PACKET PER LAYER (WITH INCREASING ORDER) OF THE MOST URGENT GENERATION.

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(1,0)	(1,1)	(1,2)
(1,3)	(1,4)	(2,0)	(2,1)	(2,2)	(2,3)	(3,0)	(3,1)	(3,2)

For the above setting, the actions involve packets from generations G_n and G_{n+1} and hence there are $M_1 = 4$ possible packet types. Each action involves the transmission of $N_1 = fT = 1 \times 5 = 5$ packets, as the channel does not experience delay and the decisions are made every 5 time slots. Under these assumptions, it is trivially shown by evaluating (7) that the action space size is $\binom{5+4-1}{5} = 56$. A few possible actions in this scenario are presented in Table IV.

The first row of Table IV corresponds to an action (and a request allocation vector) where the receiver demands from the server to send three packets from the first class of generation G_n and two packets from the second class of the same generation, while no packets that belong to the generation G_{n+1} are requested from the server.

Before proceeding with the evaluation, we should note that here the policies for the proposed schemes are determined offline. Then, the derived policies are evaluated in the simulations for various error patterns.

TABLE IV

SAMPLES OF ACTIONS $A[n]$ (REQUEST VECTORS AT t_n). (G_n, l) PACKET TYPE CORRESPONDS TO PACKETS OF CLASS l AND GENERATION G_n. EACH ROW CONTAINS THE NUMBER OF PACKETS PER CLASS AND FOR GENERATIONS G_n AND G_{n+1}.

$(G_n, 1)$	$(G_n, 2)$	$(G_{n+1}, 1)$	$(G_{n+1}, 2)$
3	2	0	0
0	0	5	0
0	2	2	1
2	1	1	1

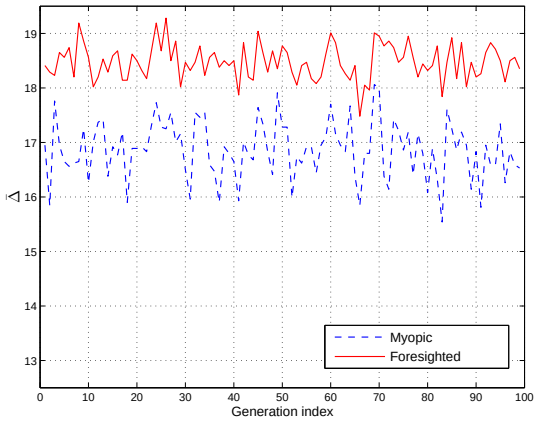


Fig. 3. Average distortion reduction evaluation of the model-MDP algorithm optimized for $\gamma = 0.0$ (myopic) and $\gamma = 0.9$ (foresighted) with respect to the generation index. The scalable video is encoded in two quality layers and the video parameters are listed in Table II. The link connecting the server with the receiver experiences 5% loss rate.

We follow the value iteration method [39] in order to solve the proposed model-MDP. The threshold value is set to 10^{-9}. When this threshold value is reached, the output of the value-iteration algorithm is the policy used in the simulations. This policy is considered as the optimal and is used as a benchmark. Similarly, the Q-learning algorithm outputs the tested policy when the maximum number of iterations $\mathcal{N}$ is reached. In all the experiments, the parameters $\Theta(0)$ and $\Theta_{\min}$ of the Q-learning algorithms are set to 75 and 0.5, respectively. All the presented results are averaged over 100 simulations. Each simulation has duration of 100 sec, namely 100 generations. For the sake of fairness, in all schemes under comparison, the transmitted packets in each link experience the same loss patterns.

1) Foresighted versus Myopic policies: We consider that the scalable video is encoded in two quality layers. The video parameters are reported in Table II. The link connecting the server with the receiver has 5% loss rate. We examine the influence of the discount factor γ for the Model-MDP scheme to the quality (*i.e.*, the achieved average distortion reduction $\overline{\Delta}$). Specifically, the value iteration algorithm is executed for $\gamma = 0.9$ and $\gamma = 0.0$, which correspond to foresighted and myopic policies, respectively. The average cumulative distortion reduction with respect to the generations index is depicted in Fig. 3.

We observe that the foresighted policies offer large performance gains (average gain of approximately 1.5 dB) compared to the myopic policies. Further, we can see that the foresighted policies offer small quality fluctuations compared to the myopic ones which is important for streaming systems. Hence, the joint consideration of multiple generations in the scheduling and coding decisions is certainly advantageous. Therefore, in the following we focus only on foresighted policies.

2) Influence of the number of Q-learning algorithms iterations to the quality: We evaluate the performance of Q-learning and Q-learning VE algorithms in terms of the average expected distortion reduction $\overline{\Delta}$ for various numbers of iterations $\mathcal{N}$. The φ values are found by experimentation for each $\mathcal{N}$.[5] We assume that the frequency of the updates U of the virtual states-actions is 10 (every 10 decision intervals, we apply the batch update, *i.e.*, we perform the steps 12-20 of Algorithm 1, for all equivalent states-actions pairs). The results are presented in Fig. 4. For the sake of completeness, we depict the average value of the distortion reduction achieved by the Model-MDP for $\gamma = 0.0$ and $\gamma = 0.9$. From Fig. 4, we can observe that Q-learning VE requires a relative small number of iterations in order to reach the performance of Model-MDP with $\gamma = 0.9$ (foresighted). Specifically, for $\mathcal{N} = 50000$, the performance difference between Q-learning VE and the Model-MDP with $\gamma = 0.9$ is only 0.06 dB. We can also see that the Q-learning algorithm requires more than 200000 iterations to converge to the performance of the foresighted Model-MDP. Further, we can observe that even with 10000 iterations the performance gains of the Q-learning VE over Myopic policies (Model-MDP with $\gamma = 0.0$) are noticeable and exceed 1 dB. Moreover, Q-learning VE outperforms Q-learning by approximately 0.5 dB. For 30000 iterations, Q-learning VE performs very close to the Model-MDP with $\gamma = 0.9$, which shows that close to optimal performance can be achieved with a relatively small number of iterations. We should emphasize that Q-learning VE algorithm achieves significantly better performance than Q-learning for the same number of iterations with slightly higher computational complexity, as the batch update of all equivalent virtual states-actions is performed every 10 iterations.

3) Influence of the update rate U of the Q-learning VE algorithm to the quality: We examine the influence of the frequency of the updates U of the Q-learning VE algorithm on the average expected distortion reduction. The φ is set to 0.99986. The results are illustrated in Fig. 5. We consider 50000 iterations and then we change the frequency U from 1 (*i.e.*, in every decision interval, we apply the batch update of all the virtual states-actions) up to 100 (*i.e.*, every 100 decision intervals, we perform a batch update of all virtual states-actions). We also depict the performance of the Model-MDP and the original

[5]The same φ values are considered for both Q-learning algorithms. These are not presented for brevity reasons.

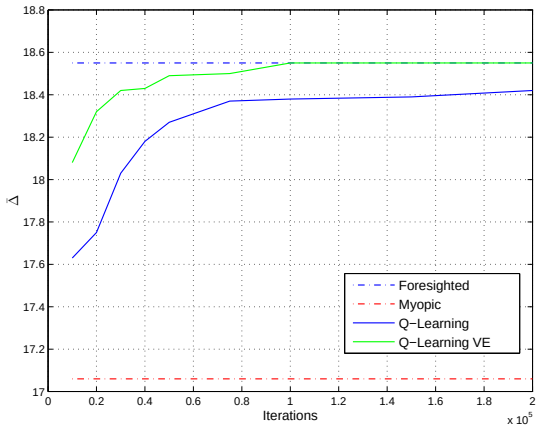


Fig. 4. Average cumulative distortion reduction achieved by the Q-learning and Q-learning VE algorithms, optimized for $\gamma = 0.9$, with respect to the number of iterations. The distortion values of Model-MDP algorithm for $\gamma = 0.0$ (myopic) and $\gamma = 0.9$ (foresighted) are also depicted. Transmission of a video encoded in two quality layers from a single server to a receiver with the parameters in Table II is considered.

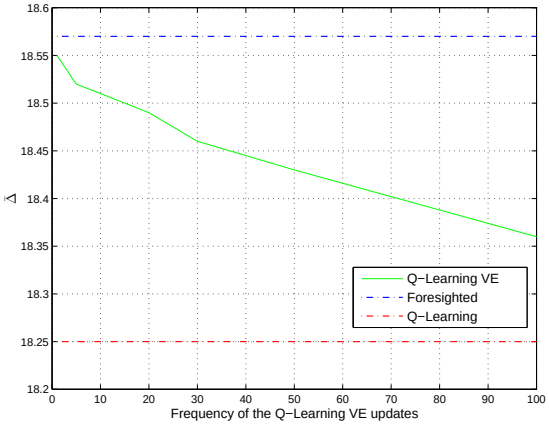


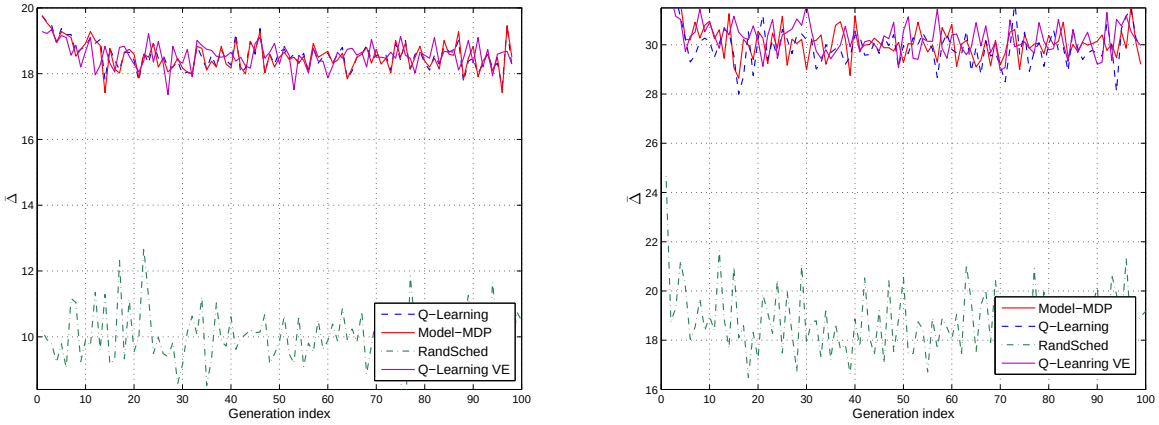
Fig. 5. Average cumulative distortion reduction achieved by the Q-learning VE algorithm, optimized for $\gamma = 0.9$ and $\mathcal{N} = 50000$ iterations, with respect to the frequency of the updates of the virtual actions-states. The distortion values of Model-MDP algorithm for $\gamma = 0.9$ and the Q-learning algorithm are also depicted. Transmission of a video encoded in two quality layers from a single server to a receive with the parameters in Table II is considered.

Q-learning algorithm with $\gamma = 0.9$. We see that, when the update rate U is 1, the performance difference is less than 0.06 dB, which is slightly better than the performance achieved when U equals to 10 and $\mathcal{N}$ is 200000 (Fig. 4). As expected, when the updates are less frequent (larger U), the performance drops. However, it is worth to mention that Q-learning VE outperforms significantly the Q-learning even with an update rate of 100. Such an update rate brings only a minimal additional computational cost compared to the Q-learning. This is particularly valuable for scenarios involving receivers with low computational capabilities.

TABLE V

PARAMETERS OF A GENERATION OF A SCALABLE VIDEO ENCODED IN THREE LAYERS. THE PARAMETERS α_i, β_i, δ_i STAND FOR THE NUMBER OF PACKETS IN THE iTH LAYER, THE CUMULATIVE NUMBER OF PACKETS IN THE FIRST i LAYERS AND THE DISTORTION REDUCTION AFTER THE SUCCESSFUL DECODING OF THE iTH LAYER, RESPECTIVELY.

	α_i	β_i	δ_i
Base Layer	3	3	11
Enhancement Layer 1	2	5	9
Enhancement Layer 2	2	7	12



(a) $D_G = 5$ time slots

(b) $D_G = 7$ time slots

Fig. 6. Average cumulative distortion reduction $\overline{\Delta}$ comparison of model-MDP, Q-learning, Q-learning VE and RandSched policies optimized for $\gamma = 0.9$ and 5% loss rate. The scalable video is encoded in three quality layers and the video parameters are listed in Table V. (a) decision interval is $D_G = 5$ time slots and (b) the decision interval is $D_G = 7$ time slots. When $D_G = 5$, $\overline{\Delta}$ for the video sequence are 18.56, 18.53, 18.54, and 10.01 dB for the model-MDP, Q-learning VE, Q-learning and RandSched algorithms, respectively. When $D_G = 7$, the $\overline{\Delta}$ values become 30.28, 30.18, 29.95, and 18.81 dB.

4) Transmission of three layer video: We further consider the transmission of videos encoded in three quality layers. The parameters of the video are presented in Table V. Similarly to the case with two data layers, the decision interval is set to 5 time slots, which is sufficient for transmitting 5 packets. The nth generation of the layered data stream becomes available at the server at $t = D_G n = 5n$ and the playback delay D_0 is 10 decision intervals. The decoding deadlines for the nth generation is $D_n = 10 + 5n$. In the nth decision interval, the generations G_n and G_{n+1} are considered as urgent. The model-MDP, the Q-learning and the Q-learning VE algorithms are optimized for $\gamma = 0.9$ and 5% loss rate. For the Q-learning VE an update rate U of 10 is chosen. We set the $(\mathcal{N}, \varphi)$ tuple to ($2{\cdot}10^6$,0.999995) and ($2{\cdot}10^5$,0.99996) for the Q-learning and Q-learning VE schemes, respectively. These solutions are compared to a scheme called "RandSched" that implements random scheduling of PRLC packets from the most urgent generations. Specifically, in each decision interval t_n, this scheme randomly selects packets for transmission from the generations G_n and G_{n+1}. The results are given in Fig. 6(a). We remark that the Q-learning VE is able

to achieve the same performance as the Model-MDP with 10 times less iterations than those required by the Q-learning algorithm. Both schemes are not able to decode the third layer. We can also observe that the RandSched scheme performs very poorly. Despite the limited bandwidth, packets that belong to third class are transmitted with RandSched, which reduces the bandwidth available for the second layer, hence its decoding probability. Interestingly, we can further observe in Fig. 6(a) that model-MDP and Q-learning methods behave similarly, although their policies are not necessarily the same. This is due to the fact that the requested packets result in similar distortion reduction.

Finally, we evaluate the same schemes when a larger bandwidth is available. Specifically, we assume that the duration of each time slot is shorter, namely $1/7$ sec, hence the bandwidth is 7 *packets/sec*. The decision interval is set to 7 time slots, which is sufficient for transmitting 7 packets. The decoding deadlines for the nth generation is $D_n = 10+7n$. Again, two generations are marked as urgent. We set the $(\mathcal{N}, \varphi)$ tuple to ($13\cdot10^6$,0.9999986) and ($4\cdot10^5$,0.999983) for the Q-learning and Q-learning VE schemes, respectively. The results are illustrated in Fig. 6(b). We can see that the model-MDP and Q-Learning algorithms are able to take advantage of the additional bandwidth and improve significantly the video quality. We can further observe that both model-MDP and Q-learning VE outperform Q-learning. This performance difference is due to the fact that the states space grows from 18 to 88 and the actions space from 56 to 792. Specifically, as the states and actions space becomes larger, the probability that Q-learning stops in a local minimum increases. Identical performance of model-MDP and Q-Leaning is still possible by properly selecting the φ and $\mathcal{N}$ values. From Fig. 6(b), we can also see that RandSched algorithm is not competitive with the model-MDP and Q-Learning schemes and their performance difference in distortion is approximately 11 dB in this illustrative example.

C. Video Transmission from Multiple Servers

We finally evaluate our model-MDP and Q-Learning algorithms for an illustrative scenario of multi-server scalable video transmission. We consider a scenario with two servers with bandwidths $f_1 = 0.6$ *packets/slot* (3 *packets/sec*) and $f_2 = 0.4$ *packets/slot* (2 *packets/sec*) and that both links have a 5% loss rate. Essentially, the examined scenario and the scenario presented in Section VI-A are identical from the receiver's point of view as the overall capacity is one *packet/sec* and loss rate in both cases is 5%. First, we examine the performance of Q-learning and Q-learning VE algorithms in terms of the average expected distortion $\overline{\Delta}$ for various numbers of iterations $\mathcal{N}$. The discount factor is set to 0.9. The frequency of the updates of the virtual states-actions U is set to 10. The φ value are found by experimentation for each $\mathcal{N}$. We also present the performance of the foresighted and myopic schemes,

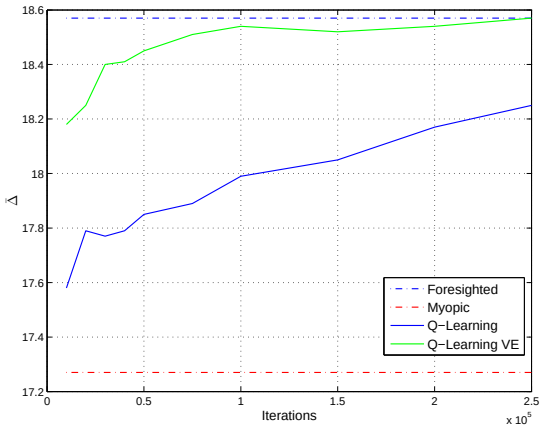


Fig. 7. Average cumulative distortion reduction achieved by the Q-learning and Q-learning VE algorithms, optimized for $\gamma = 0.9$, with respect to the number of iterations. The distortion values of Model-MDP algorithm for $\gamma = 0.0$ (Myopic) and $\gamma = 0.9$ (Foresighted) are also depicted. Transmission of a video encoded in two quality layers from two servers to a single receiver with the video parameters listed in Table II is considered.

i.e., Model-MDP for $\gamma = 0.0$ and $\gamma = 0.9$, respectively. The results are presented in Fig. 7. We see that Q-learning VE performs significantly better than the Q-learning algorithm in terms of average expected distortion. Similarly to Fig. 4, we observe that for 10000 iterations Q-learning VE outperforms Model-MDP with $\gamma = 0.0$ by almost 1 dB. For 30000 iterations, the performance difference between Q-learning VE and Model-MDP with $\gamma = 0.9$ is less than 0.2 dB. For 50000 iterations, the performance gap in terms of distortion between the Q-learning VE and the Model-MDP is small and for 100000 iterations it becomes negligible. Further, we can see that Q-learning can not reach the performance (in terms of $\overline{\Delta}$) achieved by the Q-learning VE even with 250000 iterations, and the performance difference is close to 0.3 dB. We have observed that by increasing $\mathcal{N}$ to 650000 for the Q-learning scheme, it is possible to approach the performance that Q-learning VE achieves with three times less iterations. More details can be found in [38]. By observing Figs. 4 and 7, we can conclude that the increased number of servers affects mainly the performance of the Q-learning algorithm, while the performance of the Q-learning VE stays almost invariant. This is due to the larger states-actions space (the action space increases from 56 to 200, while the action space remains the same with 18 states) that slow down the convergence of the Q-learning algorithm. Q-learning VE does not suffer from that as the batch updates help to exploit the knowledge acquired by visiting an action-state pair to multiple.

We finally consider a scenario with three servers. We set the bandwidths values to $f_1 = 0.4$ *packets/slot* (2 *packets/sec*), $f_2 = 0.4$ *packets/slot* (2 *packets/sec*) and $f_3 = 0.2$ *packets/slot* (1 *packet/sec*). All the links experience a loss rate of 5%. This scenario is identical to the one presented in Section VI-A from the receiver's point of view. We set the update rate of the virtual state-action pairs U to 10 and the discount

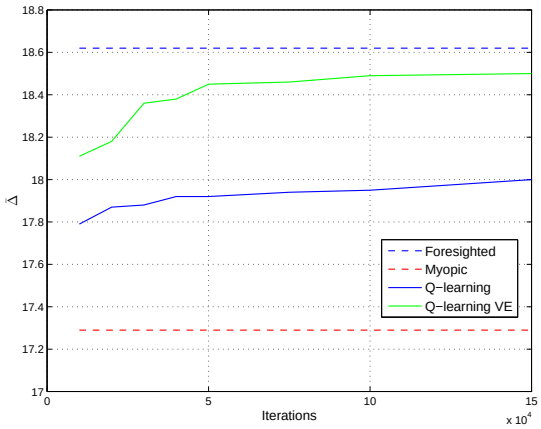


Fig. 8. Average cumulative distortion reduction achieved by the Q-learning and Q-learning VE algorithms, optimized for $\gamma = 0.9$, with respect to the number of iterations. The distortion values of Model-MDP algorithm for $\gamma = 0.0$ (Myopic) and $\gamma = 0.9$ (Foresighted) are also depicted. Transmission of a video encoded in two quality layers from three servers to a single receiver with the video parameters listed in Table II is considered.

factor to 0.9. Like in the comparison illustrated in Fig. 7, the φ values are found by experimentation for each $\mathcal{N}$. The results are shown in Fig. 8. The schemes under comparison are the Q-learning and Q-learning VE. We also depict the performance of the Model-MDP for $\gamma = 0.0$ and $\gamma = 0.9$. From the results, we observe a similar performance to the case of two servers (Fig. 7), *i.e.*, Q-learning VE outperforms significantly Q-learning. This is attributed to the multiple state-action pair updates of the Q-learning VE. The convergence of the Q-learning is slower than in the case of two servers. This is expected as the action space doubles. Despite the increase of the action space, the Q-learning VE method however needs less than 50000 iterations to approach 18.50 dB (in terms of cumulative distortion reduction).

VII. Conclusions

We have investigated the problem of jointly selecting the optimal coding strategy and scheduling decisions when receivers can obtain delay constrained layered data encoded with Prioritized Random Linear Codes from multiple servers. Markov Decision Processes are used for formulating the optimization problem. Then, we propose a Q-learning variant called Q-learning VE that exploits the fact that multiple states-actions are statistically equivalent. This permits to find effective coding and scheduling solutions with less iterations compared to the traditional Q-learning method with only slightly higher computational complexity. In illustrative video transmission examples, first we show the benefits of the foresighted over the myopic decisions. The results confirm the advantages of considering multiple generations concurrently in the scheduling. Then, we see that the Q-learning VE algorithm outperforms significantly the Q-learning method in terms of expected distortion reduction for the same number of iterations. We note that for

the Q-learning VE algorithm a relative small number of iterations are needed for taking most of the performance gains. Further, we observe that the Q-learning algorithm converges fast to that of the MDP in terms of distortion reduction. Interestingly, with the proposed methods continuous playback and small quality variations are favored. This work is a step towards using PRLC methods for real-time video streaming. Our future research will investigate further simplification of the Q-learning approaches and eventually end up with the definition of simple decision rules that approximate the performance of the Q-learning VE algorithm.

REFERENCES

[1] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger, "On Randomized Network Coding," in *41st Allerton Annual Conference on Communication, Control, and Computing*, Monticello, IL, USA, Oct 2003.

[2] A. Shokrollahi, "Raptor codes," *IEEE Trans. Information Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.

[3] M. Luby, "LT codes," in *Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '02)*, Vancouver, Canada, Nov. 2002, pp. 271–280.

[4] J. P. Wagner, J. Chakareski, and P. Frossard, "Streaming of Scalable Video from Multiple Servers using Rateless Codes," in *Int. Conf. on Multimedia and Expo, ICME'06*, Toronto, Canada, Jul. 2006.

[5] T. Schierl, S. Johansen, A. Perkis, and T. Wiegand, "Rateless Scalable Video Coding for Overlay Multisource Streaming in MANETs," *ELSEVIER Journal of Visual Communication and Image Representation*, vol. 19, no. 8, pp. 500–507, Dec. 2008.

[6] D. Sejdinovic, D. Vukobratovic, A. Doufexi, V. Senk, and R. Piechocki, "Expanding Window Fountain Codes for Unequal Error Protection," in *Proc of the 41st Annual Asilomar 2007 Conf. on Signals, Systems and Computers*, Pacific Grove, CA, USA, Nov. 2007.

[7] Y. Lin, B. Liang, and B. Li, "Priority Random Linear Codes in Distributed Storage Systems," *Trans on Parallel and Distributed Systems*, vol. 20, no. 11, pp. 1653–1667, Nov. 2009.

[8] N. Thomos, J. Chakareski, and P. Frossard, "Prioritized Distributed Video Delivery with Randomized Network Coding," *IEEE Trans. Multimedia*, vol. 13, no. 4, pp. 776–787, Aug. 2011.

[9] Y. Ding, Y. Yang, and L. Xiao, "Multisource Video On-Demand Streaming in Wireless Mesh Networks," *IEEE/ACM Trans. on Networking*, vol. 20, no. 6, pp. 1800–1813, Dec. 2012.

[10] M. Wang and B. Li, "R^2: Random Push with Random Network Coding in Live Peer-to-Peer Streaming," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1655–1666, Dec. 2007.

[11] M. C. O. Bogino, M. Sagduyu, P. Cataldi, M. Grangetto, E. Magli, and G. Olmo, "Sliding-Window Digital Fountain Codes for Streaming of Multimedia Contents," in *Proc. IEEE Int. Symp. Circuits and Systems*, New Orleans, LA, USA, May 2007.

[12] P. Cataldi, M. Grangetto, T. Tillo, E. Magli, and G. Olmo, "Sliding-Window Raptor Codes for Efficient Scalable Wireless Video Broadcasting With Unequal Loss Protection," *IEEE Trans. Image Processing*, vol. 19, no. 6, pp. 1491–1503, Jun. 2010.

[13] S. Ahmad, R. Hamzaoui, and M. M. Al-Akaidi, "Unequal Error Protection Using Fountain Codes With Applications to Video Communication," *IEEE Trans. Multimedia*, vol. 13, no. 1, pp. 92–101, Jan. 2011.

[14] H. Seferoglu and A. Markopoulou, "Video-Aware Opportunistic Network Coding over Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 5, pp. 713–728, June 2009.

[15] D. Nguyen, T. Nguyen, and X. Yang, "Joint Network Coding and Scheduling for Media Streaming Over Multiuser Wireless Networks," *Trans Vehicular Technology*, vol. 60, no. 3, pp. 1086–1098, Mar. 2011.

[16] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[17] C. J. C. H. Watkins, *Learning from Delayed Rewards*. Cambridge, UK: PhD thesis, Cambridge University, 1998.

[18] N. Mastronarde and M. V. D. Schaar, "Joint Physical-Layer and System-Level Power Management for Delay-Sensitive Wireless Communications," *IEEE Trans. on Mobile Computing*, vol. 12, no. 4, pp. 694–709, Apr. 2013.

[19] Y. Ding, Y. Yang, and L. Xiao, "Multisource Video On-Demand Streaming in Wireless Mesh Networks," *IEEE/ACM Trans. on Networking*, vol. 20, no. 6, pp. 1800–1813, Dec. 2012.

[20] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, and Z. Xiong, "Scalable Video Multicast Using Expanding Window Fountain Codes," *IEEE Trans. Multimedia*, vol. 11, no. 6, pp. 1094–1104, Jun. 2009.

[21] E. Drinea, L. Keller, and C. Fragouli, "Real-time Delay With Network Coding and Feedback," *ELSEVIER Physical Communication*, vol. 6, pp. 100–113, Mar. 2013.

[22] M. Nistor, D. E. Lucani, T. T. N. Vinhoza, R. C. Costa, and J. Barros, "On the Delay Distribution of Random Linear Network Coding," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 5, pp. 1084–1093, May 2011.

[23] P. U. Tournoux, E. Lochin, J. Lacan, A. Bouabdallah, and V. Roca, "On-the-Fly Erasure Coding for Real-Time Video Applications," *IEEE Trans. Multimedia*, vol. 13, no. 4, pp. 797–812, Apr. 2011.

[24] L. Yang, Y. E. Sagduyu, and J. H. Li, "Adaptive Network Coding for Scheduling Real-time Traffic With Hard Deadlines," in *Proc. of the ACM Int. Symp. on Mobile Ad Hoc Networking and Computing, MobiHoc'12*, Hilton Head Island, SC, USA, June 2012.

[25] M. Halloush and H. Radha, "Network Coding with Multi-Generation Mixing: A Generalized Framework for Practical Network Coding," *IEEE Trans. on Wireless Communications*, vol. 10, no. 2, pp. 466–473, Feb. 2011.

[26] P. A. Chou, Y. Wu, and K. Jain, "Practical Network Coding," in *Proc. 41st Allerton Conf. on Communication Control and Computing*, Monticell, IL, USA, Oct. 2003.

[27] F. Fu and M. V. D. Schaar, "A Systematic Framework for Dynamically Optimizing Multi-User Video Transmission," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 3, pp. 308–320, Apr. 2010.

[28] R. S. Sutton and A. G. Barto, *Reinforcement Learning*. Cambridge, UK: The MIT press, 1998.

[29] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.

[30] N. Salodkar, A. K. A. Bhorkar, and V. S. Borkhar, "An On-line Learning Algorithm for Energy Efficient Delay Constrained Scheduling Over a Fading Channel," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 732–742, Apr. 2008.

[31] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[32] F. Fu and M. V. D. Schaar, "Structure-Aware Stochastic Control for Transmission Scheduling," *IEEE Trans. Vehicular Technology*, vol. 61, no. 9, pp. 3931–3945, Nov. 2012.

[33] H. P. Shiang and M. V. D. Schaar, "Online Learning in Autonomic Multi-Hop Wireless Networks for Transmitting Mission-Critical Applications," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 5, pp. 728–741, Jun. 2010.

[34] E. Kurdoglu, N. Thomos, and P. Frossard, "Scalable Video Dissemination with Prioritized Network Coding," in *Proc. of Workshop on Streaming and Media Communication, StreamComm11 (in conjunction with ICME11)*, Barcelona, Spain, July 2011.

[35] H. P. Shiang and M. V. D. Schaar, "Quality-Centric TCP-Friendly Congestion Control for Multimedia Transmission," *IEEE Trans. Multimedia*, vol. 14, no. 3, pp. 896–909, Jun. 2012.

[36] N. Thomos and P. Frossard, "Toward one Symbol Network Coding Vectors," *IEEE Communications letters*, vol. 16, no. 11, pp. 1860–1863, Nov. 2012.

[37] D. E. Lucani, M. V. Pedersen, J. Heide, and F. H. P. Fitzek, "Fulcrum Network Codes: A Code for Fluid Allocation of Complexity," *available at http://arxiv.org/abs/1404.6620*.

[38] N. Thomos, E. Kurdoglu, P. Frossard, and M. V. D. Schaar, "Adaptive Prioritized Random Linear Coding and Scheduling for Layered Data Delivery from Multiple Servers," *available at http://arxiv.org/abs/1409.8650*.

[39] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[40] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Learning to act Using Real-time Dynamic Programming," *ELSEVIER Artificial Intelligence*, vol. 72, no. 1-2, pp. 81–138, Jan. 1995.