Optimal Packet Scheduling Algorithms for Token-Bucket based Rate Control

Neerav B Mehta
Image Formation and Processing Group
Beckman Institute
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA

Abhay Karandikar
Information Networks Laboratory
Department of Electrical Engineering
Indian Institute of Technology, Bombay
Mumbai 400 076, India

***Abstract*— In this paper, we consider a scenario in which the source has been offered QoS guarantees subject to token-bucket regulation. The rate of the source should be controlled such that it conforms to the token-bucket regulation, and also the distortion obtained is the minimum. We have developed an optimal scheduling algorithm for offline (like pre-recorded video) sources with convex distortion function and which can not tolerate any delay. This optimal offline algorithm has been extended for the real-time online source by predicting the number of packets that the source may send in future. The performance of the online scheduler is not substantially degraded as compared to that of the optimal offline scheduler. A sub-optimal offline algorithm has also been developed to reduce the computational complexity and it is shown to perform very well. We later consider the case where the source can toerate a fixed amount of delay and derive optimal offline algorithm for such traffic source.**

***Index Terms*— Rate control, token-bucket regulator, rate-distortion, QoS**

I. Introduction

In next generation Internet offering Quality of Service (QoS), the traffic flow is required to declare its traffic characteristics and the QoS attributes. The traffic charateristics are usually specified in terms of traffic descriptors. Commonly used Traffic Descriptors are Linearly Bounded Arrival Process (LBAP) [1]. An LBAP constraint bounds the maximum number of bits that a source may transmit in a given interval t by a linear function of t. The source employs traffic shapers and the network polices the traffic using Traffic Regulators to ensure that the source adheres to the advertised traffic descriptions. A simple Traffic Regulator for an LBAP descriptor is the Token Bucket Regulator which has two parameters - the token generation rate r and the size of the token bucket B.

The selection of appropriate values of traffic descriptors that characterize a traffic source well, can be very difficult [2]. Moreover, the network may be offering only a discrete set of combinations of the token bucket parameters from which the source would be required to choose. Thus the source may not always obtain its exact requirements from the network. In this paper, we consider a scenario in which the source has been offered QoS guarantees subject to certain parameters of token-bucket regulation. The source is required to schedule the size of its packets to adapt to these constraints. We argue that of all the packet length schedules that honor the imposed traffic constraint, one that minimizes the distortion may be the most appropriate one to choose. In our earlier work [3], an optimal algorithm that solves this problem has been developed under the condition that the scheduler has complete knowledge of the number and arrival times of all the packets. But the algorithm is optimal only for min-max distortion cost. The work presented in this paper generalizes our work in [3] in the sense that we have considered a general cost structure (and not a specific cost assumed in [3]). The treatment of the problem in this paper is completely different from that of [3]. Moreover, we have also extended our framework to the traffic with delay tolerance.

The primary contribution of this paper is to develop an optimal offline scheduling algorithm which minimizes a *general* distortion cost while conforming the traffic to the token-bucket regulation. The only requirement on the distortion function is that it should be convex in the output number of packets. This criterion is satisfied by most distortion functions including mean square loss, proportional mean square loss, exponential, etc.

We have also shown how this optimal offline scheduler can be used in conjunction with a predictor to implement an online scheduler in which the scheduler does not have knowledge about the number of packets that may arrive in future. The online scheduler can be very useful to control the rate of real-time traffic. To demonstrate how an online scheduler can be implemented, we have used 3 types of predictors: (i) Minimum Mean Square Error (MMSE) predictor, (ii) Stationary predictor and (iii) Normalized Least Mean Square (NLMS) predictor. Most of the online schedulers are computationally very expensive and that is primarily due to the complexity of the optimal offline algorithm.

To reduce the processing delay, a real-time system has to be computationally inexpensive. The optimal solution of the stationary predictor in conjunction with the optimal offline scheduler reduces to a very simple analytical expression which depends only on the number of input packets at the current time instant. Hence it is very easy to implement this online scheduler. To implement other online schedulers, we have developed a heuristic offline scheduling algorithm. This algorithm, when used in conjunction with MMSE and NLMS predictors, reduces the compuational complexity while at the same time, the degradation in performance is not significant.

Abhay Karandikar is the corresponding author. His email address is karandi@ee.iitb.ac.in and Fax- +91-22-25723707

A. Related Work

In the literature, many rate control algorithms have been proposed for multimedia sources. Some of the representative work can be found in [4], [5], [6], [7], [8]. Most of these rate control algorithms adapt the output rate of a video source such that the distortion is minimized. In all the schemes presented in the above papers, the rate of video is controlled at the encoding stage according to the available bandwidth.

In [5], the authors have considered the problem of optimizing distortion for constrained VBR streaming in ATM networks. The authors have proposed a complex Viterbi algorithm for quantizer selection at encoder based on apriori knowledge of distortions associated with each choice.

For unconstrained streaming, the problem of joint optimization of average bandwidth consumption and distortion or that of rate-distortion optimized streaming of video has been dealt with in many works. In [9], the authors have considered a lossy best-effort network and presented transmission schemes for rate distortion optimized streaming of packetized media in such networks. The authors have used an MPEG-type data dependency model of frames for modeling distortion. However only the mean or expected values of rate and distortion were considered in their analysis. In [10], a similar problem pertaining to streaming of scalable media consisting of independent frames in best effort networks has been dealt with. In [11], the authors have considered the problem of delay constrained transmission of layer-encoded multimedia presentation in a network with limited constant bandwidth. They have used a MINMAX measure of distortion for evaluating the presentation quality.

Our work differs from all these papers. We consider a network model which is based on the QoS framework and seek a packet length scheduling policy under given constraints on both rate and burstiness. Unlike some of the aforementioned work, our distortion model does not take into account temporal dependencies that may be present in the data units sent in different time-intervals. Such dependencies, which may be represented by an acyclic graph are characteristic of video coding schemes and can result in error propagation over finite durations. This simplification, applicable under reasonable assumptions, allows us to present a distortion model that is simple to analyze and at the same time not tied to any specific coding scheme. Our formulation thus is not specific to any video source and can be applied to an LBAP traffic source that exhibits properties of loss and delay tolerance.

In Section II, we introduce the model and the notations used throughout this paper. We also formulate the problem in Section II. The optimal offline algorithm is presented in Section III. In Section IV, we extend the optimal offline algorithm to the online case and compare the two using simulations. In Section V, we develop a heuristic offline scheduling algorithm which is computationally inexpensive and then extend it to the online case using prediction. In Section VI, we extend our formulation to the case where the source can tolerate a fixed amount of delay. Section VII outlines the future work and concludes the paper.

II. Problem Formulation

Let $\vec{t} = (t_1, \ldots, t_N)$ denote the times at which the source emits the packets and they arrive as inputs to the token-bucket regulator as shown in Fig. 1. Let $\vec{x} = (x_1, \ldots, x_N)$ denote the number of packets that the source sends at time $\vec{t}$. We assume that the application is real-time and the packets are considered useless if they are delayed by the token-bucket regulator. The token-bucket regulator outputs the packets instantaneously. Let $\vec{y} = (y_1, \ldots, y_N)$ denote the number of packets that are output by the token-bucket regulator at time $\vec{t}$. Hence at time t_i, $x_i - y_i$ number of packets are dropped. Let $\vec{R} = (R_1, \ldots, R_N)$ denote the number of tokens refilled at time $\vec{t}$[1]. B_{max} is the maximum number of tokens that the bucket can hold. We assume that the tokens are refilled just when the packets arrive and hence, they do not add to the tokens in the bucket before the packets leave. Let $\vec{B} = (B_1, \ldots, B_N)$ denote the number of tokens remaining in the bucket just before the packets arrive. The vector $\vec{y}$ conforms to the token-bucket regulation.

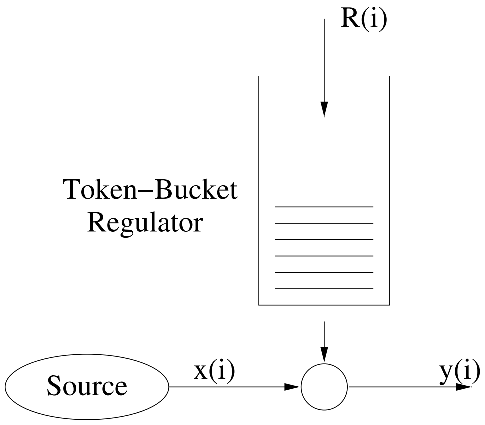

Fig. 1. The token-bucket regulator.

Definition 1: A schedule $\vec{y} = (y_1, \ldots, y_N)$ of output number of packets is *conformant* to the token-bucket regulation if:

$$y_i \leq \min(x_i, B_i + R_i) \;\; \forall\, i \,\epsilon\, [1, N] \tag{1}$$

where the number of remaining tokens in the bucket is updated as:

$$B_{i+1} = \min(B_{max}, B_i + R_i - y_i) \;\; \forall\, i \,\epsilon\, [1, N-1] \tag{2}$$

We assume that the rate-distortion curve of the source is not known. Hence it is reasonable to assume that the distortion at the time instant t_i does not depend on any other time instant. Let $C(x_i, y_i, i)$ denote the distortion cost when only y_i packets of the total x_i input packets are output at time instant t_i. The distortion cost function is assumed to satisfy the following:

1) $C(x_i, y_i, i) \geq 0 \;\; \forall\, i \,\epsilon\, [1, N]$
2) $C(x_i, y_i, i)$ is monotonically decreasing in $y_i \,\epsilon\, [0, x_i]$ with $C(x_i, x_i, i) = 0 \;\; \forall\, i \,\epsilon\, [1, N]$.

[1] Note that we have considered a more general token-bucket regulator. The standard IETF token-bucket regulator is a special case realized by setting inter-arrival times to be constant and $R_i = R \;\; \forall\, i \,\epsilon\, [1, N]$.

3) $C(x_i, y_i, i)$ is convex in $y_i \,\epsilon\, [0, x_i]$.

Assumptions 1 and 2 are obvious. Assumption 3 is valid for most distortion cost measures such as mean square loss, proportional mean square loss, exponential, etc.

Definition 2: A schedule $\vec{y^*}$ of output number of packets is *optimal* if it is conformant to the token-bucket regulation and there is no schedule $\vec{y}$ for which the distortion cost is less than that for the schedule $\vec{y^*}$, given the token-bucket parameters.

We seek an answer to the following question: How should the schedule $\vec{y}$ be chosen such that it is optimal?

We first consider the problem of designing an optimal "offline" algorithm, assuming that the vector $\vec{x}$ is known. We then extend it to the online algorithm by predicting the number of packets that may arrive in future.

III. Optimal Offline Scheduling

In this section, an optimal offline scheduling algorithm is formulated. Suppose $\vec{y}$ is the optimal schedule. It is proved in the appendix (Lemma 1) that to achieve optimality, $y_j \geq \min(x_j, \max(0, R_j - B_{max})) \;\forall\; j \,\epsilon\, [1, N]$, with $y_j \leq x_j$. Hence, the sequence of time indices $[1, N]$ can be divided into two alternating sub-sequences: (i) sub-sequence of type 1 for which $y_j = x_j \leq \max(0, R_j - B_{max})$, and (ii) sub-sequence of type 2 for which $\max(0, R_j - B_{max}) \leq y_j \leq x_j$.

For eg. suppose $\vec{x} = (3, 9, 5, 1, 10, 6, 2, 3, 8, 9)$. Assume that $R_i = 7 \;\forall\; i \,\epsilon\, [1, 10]$ and $B_{max} = 2$. Hence the vector $\vec{x}$ can be divided into sub-sequences: (i) (3), (ii) (9), (iii) $(5, 1)$, (iv) $(10, 6)$, (v) $(2, 3)$ and (vi) $(8, 9)$. The sub-sequences (i), (iii) and (v) are the sub-sequence of type 1 and the sub-sequences (ii), (iv) and (vi) are the sub-sequence of type 2. There is no distortion obtained in the sub-sequence of type 1. Hence the aim is to minimize the distortion in the sub-sequence of type 2.

Consider an instance of sub-sequence of type 2 of length M, starting at index i and ending at index $i + M - 1$ as shown in Fig. 2. It is proved in the appendix (Lemma 2) that for optimality, $y_i \geq \min(x_i, R_i)$ and $y_{i+M-1} \geq \min(x_{i+M-1}, R_{i+M-1})$ with the constraint that the schedule $\vec{y}$ is conformant to the token-bucket regulation. Hence the minimum allotment required in sub-sequence 2 is:

$y_i = \min(x_i, R_i)$,
$y_{i+M-1} = \min(x_{i+M-1}, R_{i+M-1})$, and
$y_j = \max(0, R_j - B_{max}) \;\forall\; j \,\epsilon\, [i + 1, i + M - 2]$.

After the above step, if the allocation is not optimal, then it means that it should be possible to reduce the distortion cost by adding atleast one more packet at some time instant, say t_k.

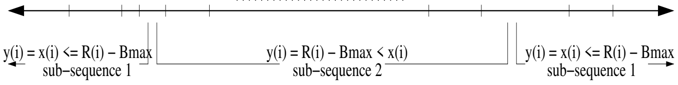

Fig. 2. An instance of sub-sequence 2 of length M.

Assume that the distortion cost function $C(x_j, y_j, j)$ is strictly convex in $y_j \;\; \forall j \,\epsilon\, [1, N]$, and the cost reduced by adding a packet is positive. We will later argue that the algorithm remains optimal even if $C(x_j, y_j, j)$ is convex in $y_j \;\; \forall j \,\epsilon\, [1, N]$ and the cost reduced by adding one more packet is non-negative.

Definition 3: Marginal cost associated with the addition of a packet is defined as the reduction in the distortion cost obtained when that packet is added to the present output schedule.

Since the distortion cost function $C(x_j, y_j, j)$ is assumed to be strictly convex in $y_j \;\; \forall\; j \,\epsilon\, [1, N]$, the marginal cost associated with the addition of a new packet decreases as more packets are added to the schedule at that time instant. Hence at any time instant, the first packet has the highest marginal cost and the last packet has the least marginal cost. Quite clearly, for any sequence of time indices $[i, i + M - 1]$, it is possible to add packets to the schedule such that the marginal cost associated with each additional packet is non-increasing.

It is proved in the appendix (Lemma 3) that a conformant schedule $\vec{y}$ is optimal only if there does not exist a conformant schedule $\vec{\sigma}$ which can be derived from $\vec{y}$ by adding a packet at any time instant.

From above, we can argue that an optimal schedule can be obtained if the packets are added to the schedule such that the marginal cost associated with each additional packet is non-increasing, and the packets are added till no more packet can be added to the schedule while conforming it to the token-bucket regulation. It thus follows that an optimal allocation should proceed as follows.

At each iteration, we calculate the marginal cost ΔC_j associated with a new packet at each time instant t_j. Suppose k is the time instant at which the reduction in the distortion cost is the maximum[2]. Increase y_k by 1, i.e. make $y_k = y_k + 1$.

Next we calculate the tokens remaining in the bucket at each time instant between i and $i + M - 1$ using Eq. 2. Adding a packet at time index k reduces the number of tokens remaining in the bucket by one for the time indices $[k + 1, p]$, where p is such that $B_p + R_p - y_p > B_{max}$. As a result, this may lead to a violation of the token-bucket regulation at some time index $l \,\epsilon\, [k, p - 1]$, i.e. $y_l > B_l + R_l$. If this happens, the packet that was added at the time index k has to be removed, and also the time index k has to be eliminated from future consideration since no packet can be added there now. If we add a packet at any time instant $j \,\epsilon\, [k, l]$, it will again lead to the violation of the token-bucket regulation at l, since the number of remaining tokens in the bucket at time index l will again be decremented by 1. Hence, remove all the time indices from k to l from future consideration. No more packet can be added at these time indices as well. The algorithm would end when all the time indices are eliminated from future consideration. The algorithm has been summarized in Algorithm 1.

Till now, we have assumed that the distortion cost function is strictly convex in $y_j \;\; \forall\; j \,\epsilon\, [1, N]$ and the reduction in cost

[2]Here we have assumed that there is only one time instant at which the reduction in cost is the maximum. It is proved in the appendix (Lemma 4) that if there are more than one time instants at which the reduction in cost obtained is the maximum, then we can select any one of these time instants and add a packet to the output at that time instant. This does not affect the operation of the algorithm.

on adding any packet to the schedule is positive. It is easy to observe that the algorithm remains optimal even if the distortion cost function is convex in $y_j \ \ \forall\, j \,\epsilon\, [1, N]$ and the reduction in cost on adding any packet at any time instant is non-negative.

Algorithm 1 Optimal Offline Algorithm

Inputs: $\vec{\mathbf{x}} = \mathbf{x_1^N}, \vec{\mathbf{R}} = \mathbf{R_1^N}, B_{max}$ ($\mathbf{a_p^q}$ denotes the vector (a_p, $a_{p+1}, \ldots, a_q$))
Output: $\vec{\mathbf{y}} = \mathbf{y_1^N}$

1: Divide $\vec{\mathbf{x}}$ into two alternating sub-sequences such that in one sub-sequence, $x_j \leq \max(0, R_j - B_{max})$ and in the other, $x_j > \max(0, R_j - B_{max})$.
2: Allot $y_j = x_j \ \ \forall\, j \,\epsilon$ sub-sequence 1.
3: Consider $\mathbf{x_i^{i+M-1}}$, an instance of sub-sequence 2 of length M. Allot $y_i = \min(x_i, R_i)$, $y_{i+M-1} = \min(x_{i+M-1}, R_{i+M-1})$ and $\mathbf{y_{i+1}^{i+M-2}} = \max(0, \mathbf{R_{i+1}^{i+M-2}} - B_{max})$.
4: Let $\mathbf{\Delta C_i^{i+M-1}} = 0$.
5: Compute $\Delta C_j = C(x_j, y_j, j) - C(x_j, y_j + 1, j)\ \forall\, j \,\epsilon\, [i, i + M - 1]$ for which $\Delta C_j \neq -1$.
6: Suppose $\Delta C_k = \max\{\mathbf{\Delta C_i^{i+M-1}}\}$. If there are more than one indices for which ΔC is the maximum, then select the minimum of these indices. Let this index be k.
7: Let $y_k = y_k + 1$.
8: Compute $\mathbf{B_i^{i+M-1}}$ starting with $B_i = B_{max}$ and using the equation $B_{j+1} = \min\{B_{max}, B_j + R_j - y_j\}$.
9: If there is any $l \,\epsilon\, [k, i + M - 1]$ for which $y_l > \min(x_l, R_l + B_l)$, make $y_k = y_k - 1$. Make $\mathbf{\Delta C_k^l} = -1$ for all such l.
10: Go to step 11 if $\Delta C_j = -1 \ \ \forall\, j \,\epsilon\, [i, i + M - 1]$. Otherwise, go to step 5.
11: Repeat the above algorithm from step 4 for all the instances of sub-sequence 2.

Tha above offline algorithm can be used only when we have complete information about the number of packets that the source sends and the times at which these packets are sent. However, this algorithm is optimal and will be useful for pre-recorded sources only. In the next section, we develop online algorithms which can be very widely used, even if the scheduler does not have information about the packets that may arrive in future. These online scheduling algorithms are evaluated based on the optimal offline scheduling algorithm.

IV. Online Scheduling

In this section, we proceed by first formulating the model used for online scheduling (Section IV-A). Based on the model, three online algorithms are developed. The optimal offline algorithm and the online algorithms are compared using simulations (Section IV-C). The simulation model used is explained in Section IV-B.

A. Online Scheduling Model

The model used to develop online algorithms is shown in Fig. 3.

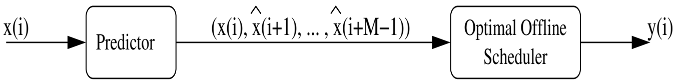

Fig. 3. Model of the online scheduler

The predictor predicts the number of packets that may arrive in the next $M - 1$ time instants. For convenience, we have assumed that the tokens arrive at fixed time intervals. Based on the number of input packets at the current time instant and the predicted number of input packets in the next $M - 1$ time instants, the optimal offline scheduler is used to compute the number of packets that should be output at the current time instant. Instead of separate predictor and optimal offline scheduler, we could have also used a joint predictor-scheduler system which minimizes the expected cost over M time instants. But this model would have been highly dependent on the distortion function used. Hence we have opted for the model shown in Fig. 3.

The performance of the online scheduler depends on the type of predictor used. We have analyzed the system using three types of predictors: (i) MMSE predictor, (ii) stationary predictor, and (iii) NLMS predictor.

1) MMSE predictor: We assume that the source is wide-sense stationary. The number of packets in the next $M - 1$ time instants is estimated as a linear combination of the number of input packets at the current time instant and the number of input packets in the previous K time instants, i.e.,

$$\hat{\mathbf{X}} = \mathbf{C}\mathbf{X} \tag{3}$$

where $\hat{\mathbf{X}}$ is an $M - 1$ dimensional vector of the predicted number of input packets in the next $M - 1$ time instants, $\mathbf{X}$ is a $K + 1$ dimensional vector of the number of input packets at the current time instant and in the previous K time instants and $\mathbf{C}$ is an $(M - 1) \times (K + 1)$ matrix of coefficients. The coefficient matrix $\mathbf{C}$ is chosen such that the mean square error between the actual value and the predicted value is minimized over the whole input sequence. We assume that the matrix $\mathbf{C}$ is already known.

2) Stationary predictor: We assume that the source is not very bursty. The predicted number of input packets in the next $M - 1$ time instants is the same as the number of input packets at the current time instant, i.e.,

$$\hat{x}_{i+1} = \ldots = \hat{x}_{i+M-2} = x_i \tag{4}$$

3) NLMS predictor: This is an adaptive predictor. The predicted number of input packets in the next $M - 1$ time instants is given by

$$\hat{\mathbf{X}} = \mathbf{C^T}\mathbf{X} \tag{5}$$

where $\hat{\mathbf{X}}$ is an $M - 1$ dimensional vector of the predicted number of input packets in the next $M - 1$ time instants, $\mathbf{X}$ is a $K + 1$ dimensional vector of the number of input packets at the current time instant and in the previous K time instants and $\mathbf{C}$ is a $(K + 1) \times (M - 1)$ matrix of coefficients. The coefficient matrix $\mathbf{C}$ is updated as

$$\mathbf{C} = \mathbf{C} + \mu \frac{\mathbf{E}\mathbf{X^T}}{\mathbf{X^T}\mathbf{X}} \tag{6}$$

where $\mathbf{E}$ is $K+1$ dimensional error vector and μ is a constant. The initial value of $\mathbf{C}$ can be chosen randomly. The error decreases to zero with increasing number of iterations.

B. Simulation Model

In deriving the optimal offline scheduling algorithm, we have assumed that the packets are output by the token-bucket regulator instantaneously and the remaining packets are dropped since they are useless if they arrive delayed. This assumption is valid for multimedia streams. Hence in our simulations, we have used video streams.

1) Video Stream Properties: In the simulations, pre-encoded MPEG-4 video streams from [12] are used. In all the traces used, the number of video objects is one. The width of the display is 176 pels and the height is 144 pels. Pel depth is set to 8 bits per pel. The video object layer frame rate is 25 frames/sec. The GoP pattern is IBBPBBPBBPBB. The frame inter-arrival times are 40 ms. We consider the time between two Group of Pictures GoP, i.e., 480 ms as a basic time unit. The tokens are refilled at equal time intervals of 480 ms. Each frame is divided into packets of 48 bytes each. The number 48 is so chosen since it is equal to the size of payload in ATM cells. It is assumed that the token-bucket regulator has no information about the GoP other than its size in terms of number of packets. More importantly, it does not know the rate-distortion curve of the video source. Let x_i denote the number of packets that the source emits in the i^{th} GoP. Let $\vec{y}$ be the token-bucket regulated output stream.

In this paper, we have used 10 video sequences. The size of each video sequence is 5000 GoPs. The mean number of packets in GoPs and its variance for each video sequence are shown in Table I. In simulations, all the elements of the vector $\vec{R}$ are chosen to be equal. Each element of the vector $\vec{R}$ is chosen to be approximately equal to the mean number of packets that the source emits per GoP. The value of B_{max} is chosen to be about one-third of R. The exact values of R and B_{max} chosen for each video sequence are shown in Table I.

2) Distortion Cost Function: In most literatures concerning distortion in video, mean square error (MSE) or peak signal to noise ratio (PSNR) which is related to MSE are taken as accepted distortion cost functions. We know that the exact distortion function for video is very difficult to determine and it depends on human perception. Instead of MSE or PSNR, we have opted for proportional mean square loss as the distortion cost function because:

1) We have assumed that the scheduler does not know the rate-distortion function of the video source. Hence the cost $C(x_i, y_i, i)$ is taken to be independent of any time instant other than t_i. This in effect means that the scheduler assumes that there is no error concealment at the receiver's end. Suppose $y_i = 0$ for some i. If there is no error concealment, what we receive on the screen is a blank. The distortion cost associated with any blank should ideally be the same, irrespective of the value of the actual number of packets in that GoP, x_i. Hence, when $y_i = 0$, the distortion function should be independent of x_i for any i. Proportional mean square loss function obeys this property.
2) We know that normally in MPEG-4 coding, video is divided into basic and enhancement layers. It has been observed that whenever the basic layer has large size, the enhancement layers tend to have large sizes. Now assume that there are two enhancement layers. Intuitively one would feel that if instead of dropping equal number of packets at all time instants, the jitter in the distortion would be less if one enhancement layer is dropped at each time instant. This is in accordance with the proportional mean square loss cost function definition.

Because of above reasons, we have decided to take the proportional mean square loss as the distortion function in our simulations.

$$\therefore C(x_i, y_i, i) = \left(1 - \frac{y_i}{x_i}\right)^2 \tag{7}$$

We have assumed that the distortion cost function is the same for all i. We would like to emphasize that our formulation is general and the simulation could be done with any distortion cost function. The choice of proportional mean square loss as the distortion cost function is only for illustrative purpose.

C. Results and Comparisons

The optimal offline scheduling algorithm and the online scheduling algorithms have been simulated using the simulation model described earlier. The parameters used for different types of predictors are given below.

1) MMSE Predictor: The number of input packets that may arrive in the next $M-1$ time instants are to be predicted based on the number of input packets at the current time instant and the number of input packets in the previous K time instants. From our simulations, we have noted that the prediction error does not change much if we increase K beyond 11. We have also noticed that the average prediction error increases as M is increased. Hence it is advisable to use as small number of predicted samples as possible in the online algorithm. But the optimal offline algorithm works the best if the full input sequence is presented to it. Hence, as an engineering compromise, we have selected $M = 3$[3]. The simulated results for this system for the 10 video traces are shown in Table II. We notice that the distortion cost in all the traces comes to be about 10% more than the optimal offline cost.

2) Stationary Predictor: The number of input packets in the next 2 time instants is assumed to be equal to the number of input packets at the current time instant. Since the distortion cost function is assumed to be the same for every time instant, the closed form solution for the number of output packets at the current instant can be easily derived. From the optimal offline scheduling algorithm, it can be argued that if the number of tokens remaining in the bucket at the current instant is B, then to minimize the distortion,

$$y_i = \min(x_i, \lceil R + \frac{B}{3} \rceil) \tag{8}$$

[3]Note that these values of parameters are not necessarily the best. In this paper, it is only shown how prediction can be used in online scheduling algorithm. Better predictors will definitely improve the online scheduler.

TABLE I
TOKEN-BUCKET PARAMETERS USED FOR 10 VIDEO TRACES.

Video Trace	Mean Packets per GoP	Variance	R (Packets)	B_{max} (Packets)
Aladdin	562.7	1.76×10^7	560	200
Alpin Ski	1.09×10^3	7.49×10^7	1000	300
Die Hard	885.7	1.97×10^5	880	250
Jurassic Park 1	996.5	2.13×10^5	1000	300
Mr. Bean	779.5	8.46×10^4	780	250
Robin Hood	1.13×10^3	1.51×10^5	1200	400
Silence of the Lambs	777.6	3.30×10^5	770	250
Soccer match	1.39×10^3	2.03×10^5	1400	500
Star Trek	389.4	4.85×10^4	380	130
Star Wars	345.7	1.88×10^4	350	120

TABLE II
RESULTS OF SCHEDULING ALGORITHMS ON 10 VIDEO TRACES.

Video Trace	Optimal Offline Algorithm	Online Algorithms using Optimal Offline Algorithm and			Heuristic Offline Algorithm	Online Algorithms using Heuristic Offline Algorithm and	
		MMSE Predictor	Stationary Predictor	NLMS Predictor		MMSE Predictor	NLMS Predictor
Aladdin	0.0293	0.0318	0.0318	0.0325	0.0306	0.0319	0.0324
Alpin Ski	0.0357	0.0374	0.0370	0.0378	0.0363	0.0375	0.0378
Die Hard	0.0393	0.0406	0.0407	0.0415	0.0401	0.0407	0.0414
Jurassic Park 1	0.0360	0.0370	0.0370	0.0377	0.0366	0.0371	0.0377
Mr. Bean	0.0234	0.0248	0.0249	0.0253	0.0245	0.0248	0.0257
Robin Hood	0.0147	0.0162	0.0164	0.0171	0.0158	0.0163	0.0171
Silence of the Lambs	0.0544	0.0553	0.0553	0.0560	0.0550	0.0545	0.0561
Soccer match	0.0191	0.0210	0.0211	0.0220	0.0205	0.0211	0.0221
Star Trek	0.0412	0.0429	0.0429	0.0438	0.0422	0.0427	0.0438
Star Wars	0.0242	0.0261	0.0261	0.0268	0.0254	0.0261	0.0268

The results for this scheduler are presented in Table II. We observe that in all the video traces used, the distortion cost obtained by using this type of predictor is almost the same as that obtained using the MMSE predictor. From Eq. 8, we find that there is no need to predict the number of packets that may arrive in future. Hence this algorithm is computationally very inexpensive and can be easily incorporated in online schedulers.

3) NLMS Predictor: Using the number of input packets at the current time instant and the number of input packets in the previous 11 time instants, the number of packets that may arrive in the next 2 time instants are predicted. The prediction is based on the NLMS algorithm. The value of μ is chosen to be 1 since it was found to give good stability to the system along with a rapid convergence rate. The results obtained using this method are presented in Table II. We note that the NLMS predictor gives a much higher distortion cost than both the MMSE predictor and the stationary predictor. This is because we have not optimized the value of μ for the best performance. The distortion cost will be reduced if an optimum value of μ is found and used.

From the comparisons, we can argue that the online scheduler using stationary predictor performs almost as well as the online scheduler using the MMSE predictor and better than the online scheduler using the NLMS predictor and also it is computationally very inexpensive as compared to the others. For an online scheduler to be implementable in real-time, it has to be computationally inexpensive. Another source of computational complexity of the online scheduler is the optimal offline scheduler block. Hence in the next section, we have developed a sub-optimal, heuristic offline scheduler which is computationally inexpensive as compared to the optimal offline scheduler. This can be used in conjunction with either MMSE predictor or NLMS predictor to implement an online scheduler.

V. HEURISTIC OFFLINE AND ONLINE SCHEDULING

In this section, we replace the optimal offline scheduler in Fig. 3 with a sub-optimal, heuristic offline scheduler. This heuristic offline scheduler is specific to the distortion cost function given in Eq. 7. We then compare how these online schedulers which use MMSE or NLMS predictors and heuristic offline algorithm fare as compared to the online schedulers which use optimal offline algorithm.

A. Heuristic Offline Scheduling

We assume that the scheduler has the complete information about all the packets that the source sends. From the optimal offline scheduling algorithm, we know that for optimality, $y_i = x_i$ if $x_i \leq \min(x_i, \max(0, R_i + B_i - B_{max})) \quad \forall\ i\ \epsilon\ [1, N]$. Now consider the case where $x_i > R_i + B_i - B_{max}$. Suppose the current time index is j and k is the next higher time index at which $x_k \leq \max(0, R_k - B_{max})$. Suppose the number of tokens remaining in the bucket at the current time instant is B_j. Under this condition, it is proved in the appendix that for the distortion cost function given in Eq. 7, y_j can be approximated by

$$y_j = \min(\max(0, R_j + B_j - B_{max}, f), x_j, B_j + R_j) \tag{9}$$

where

$$f = \lceil x_j \frac{\sum_{i=j}^{k-1} x_i(x_i - x_j) + x_j(B_j + \sum_{i=j}^{k-1} R_i)}{\sum_{i=j}^{k-1} x_i^2} \rceil \tag{10}$$

We note that this is a closed form expression. Computation of y_j does not require any iterative algorithm. Hence the computation of y_j is quite straight-forward as compared to the optimal offline scheduling algorithm. For comparison, the results obtained on the 10 video traces using the heuristic offline algorithm are tabulated in Table II. We note that the distortion costs obtained using the heuristic offline algorithm is about 5% more than that obtained using the optimal offline algorithm.

B. Heuristic Online Scheduling

In this section, the heuristic offline algorithm developed above is used in conjunction with MMSE and NLMS predictors to develop an online scheduler, which is computationally inexepnsive. There is no need to use the heuristic offline algorithm with the stationary predictor since the stationary predictor in conjunction with the optimal offline algorithm gives a very simple closed form expression for y_j and hence, is quite easily implementable in real-time.

1) MMSE Predictor: MMSE predictor is used in conjunction with the heuristic offline scheduler to implement an online scheduler. Same parameters of the MMSE predictors that were used before are used here, i.e. $M = 3$ and $K = 11$. The distortion costs obtained using this online scheduler for 10 video traces are tabulated in Table II. We note that in the worst case, the costs obtained using this system of equations are greater than those obtained using the MMSE predictor and optimal offline scheduler by only 0.001. This amounts to a maximum of 6% increase in cost for the video traces used in the simulations. But the computational complexity reduced by this online scheduler is tremendous. Hence this will be useful in implementing the online scheduler where the processor speed and the algorithmic complexity play a major role.

2) NLMS Predictor: NLMS predictor is used in conjunction with the heuristic offline scheduler to implement an online scheduler. Same parameters of the NLMS predictor that were used before are used here, i.e. $M = 3$, $K = 11$ and $\mu = 1$. The distortion costs obtained using this online scheduler for 10 video traces are tabulated in Table II. We note that the distortion cost has not increased much from that obtained using NLMS algorithm with optimal offline scheduler, but the gain due to reduction in computational complexity is tremendous. Hence using the heuristic offline scheduler instead of the optimal offline scheduler to implement an online scheduler is very beneficial.

VI. Generalized Optimal Offline Scheduling Algorithm

In Section III, an optimal offline scheduling algorithm has ben developed. It is assumed that the packets are useless if they are delayed by the token-bucket regulator. In this section, we generalize the optimal offline algorithm by assuming that the packets can be delayed by M time indices without any distortion cost. For example. the input packets arriving at time t_i can be output by the token-bucket regulator at any time index from t_i to t_{i+M} in any order without any change in distortion cost.

As mentioned in Section II, $\vec{t} = (t_1, \ldots, t_N)$ denotes the times at which the packets arrive at the token-bucket regulator. $\vec{x} = (x_1, \ldots, x_N)$ denotes the number of packets that the source sends at time $\vec{t}$. A maximum delay of M time units is permitted. Let y_i^j denote the number of packets, of the total x_i input packets, that are output by the token-bucket regulator at time index t_{i+j}. Hence, $j\ \epsilon\ [0, M]$. Let $\vec{y} = (y_1, \ldots, y_N)$ denote the total number of packets, of the $\vec{x} = (x_1, \ldots, x_N)$ input packets, that are output by the token-bucket regulator. Hence,

$$y_i = \sum_{j=0}^{M} y_i^j \tag{11}$$

Let $\vec{y'} = (y'_1, \ldots, y'_N)$ denote the total number of packets that are output by the token-bucket regulator at time $\vec{t}$. y'_i and y_i are not the same. y'_i may consist of some packets of y_{i-1} also. More specifically,

$$y'_i = \sum_{j=0}^{M} y_{i-j}^j \tag{12}$$

Clearly, $y_{i-j}^j = 0$ if $i \leq j$. The aim is to find the optimal schedule $\{y_i^j\}\ \forall\ i\ \epsilon\ [1, N]\ \forall\ j\ [0, M]$ such that the total distortion cost is the minimum. Assumptions 1, 2 and 3 stated in Section II still apply.

Since every packet can be delayed by a maximum of M time indices, it is obvious that any conformant output schedule can be rearranged such that no packet which arrives at time index t_j is output after a packet which arrives at time index t_{j-1}. Hence, if we know $\vec{y} = (y_1, \ldots, y_N)$, the output schedule $\{y_i^j\}\ \forall\ i\ \epsilon\ [1, N]\ \forall\ j\ [0, M]$ can be easily determined from the token-bucket constraints, i.e., Eq.1 and Eq.2.

The generalized optimal offline scheduling algorithm and its proof proceed along the lines of the optimal offline scheduling algorithm with no delay. Since we have to determine $\vec{y} = (y_1, \ldots, y_N)$ only, we consider the number of output packets against the number of input packets. We may note that in the optimal offline scheduling algorithm with no delay, we had considered the number of output packets against the time index. In this case, we do not divide the sequence of input packets into sub-sequences but start with calculation of the marginal costs.

The rest of the algorithm remains the same with the exception that

$$y_j \leq \min(x_j, B_j + \sum_{k=0}^{M} R_{j+k}) \quad (13)$$

Since y_j can be greater than $B_j + R_j$, B_{j+1} can be negative. This means that more tokens are utilized and hence, some packets have to be delayed. Once $\vec{y} = (y_1, \ldots, y_N)$ is known, the output schedule $\{y_i^j\}\ \forall\, i\, \epsilon\, [1, N]\ \forall\, j\ [0, M]$ can be easily calculated using Eq.1, Eq.2 and the fact that no packet which arrived at time index t_j is output after a packet that arrived at time index t_{j+1}. Algorithm 2 summarizes the generalized optimal offline scheduling algorithm.

Algorithm 2 Generalized Optimal Offline Algorithm

Inputs: $\vec{\mathbf{x}} = \mathbf{x_1^N}, \vec{\mathbf{R}} = \mathbf{R_1^{N+M}}, B_{max}$ ($\mathbf{a_p^q}$ denotes the vector $(a_p, a_{p+1}, \ldots, a_q)$)
Output: $\{y_i^j\}\ \forall\, i\, \epsilon\, [1, N]\ \forall\, j\ [0, M]$

1: Let $\mathbf{\Delta C_1^N} = 0$.
2: Compute $\Delta C_j = C(x_j, y_j, j) - C(x_j, y_j + 1, j)\ \forall\, j\, \epsilon\, [1, N]$ for which $\Delta C_j \neq -1$.
3: Suppose $\Delta C_k = \max\{\mathbf{\Delta C_i^{i+M-1}}\}$. If there are more than one indices for which ΔC is the maximum, then select the minimum of these indices. Let this index be k.
4: Let $y_k = y_k + 1$.
5: Compute $\mathbf{B_1^N}$ starting with $B_1 = B_{max}$ and using the equation $B_{j+1} = \min\{B_{max}, B_j + R_j - y_j\}$.
6: If there is any $l\ \epsilon\ [k, N]$ for which $y_l > \min\left(x_l, B_l + \sum_{k=0}^{M} R_{l+k}\right)$, make $y_k = y_k - 1$. Make $\mathbf{\Delta C_k^l} = -1$ for all such l.
7: Go to step 8 if $\Delta C_j = -1\ \ \forall\, j\, \epsilon\, [i, i + M - 1]$. Otherwise, go to step 2.
8: Using $\mathbf{y_1^N}$, Eq. 1, Eq. 2 and the fact that no packet which arrived at time index t_j is output after a packet which arrived at time index t_{j+1}, determine the optimal output schedule $\{y_i^j\}\ \forall\, i\, \epsilon\, [1, N]\ \forall\, j\ [0, M]$

VII. Conclusions and Future Work

In this paper, we have considered a scenario where a source has been offered QoS guarantees subject to its adherence to the token-bucket regulation. We argue that of all the packet schedules that honour the imposed constraint, one that minimizes the distortion cost should be chosen for transmission. Accordingly we have developed an optimal scheduling algorithm which minimizes the distortion cost given the token-bucket parameters and the number and arrival times of all the packets that the source sends. This algorithm is optimal for any traffic for which the distortion cost function is convex in the output number of packets. We call this optimal offline algorithm. We have extended the optimal offline scheduling algorithm to the online case and have shown that the performance of the online scheduler does not degrade significantly as compared to that of the optimal offline scheduler. We have also shown how heuristic online schedulers, which reduce the computational complexity, can be implemented. Finally, an optimal offline algorithm has been developed in which the packets can be delayed within a certain bound. This work can be extended in future along the following lines:

1) An offline algorithm must be developed which is optimal for any shape of the distortion cost function.
2) Better prediction of the packets that may arrive in future will definitely lead to an improvement in the online scheduling algorithm.

We are currently investigating the above problems.

Appendix

Lemma 1: A necessary condition for optimality is

$$\begin{aligned} y_j &= x_j \text{ if } x_j \leq \gamma_j \\ &\geq \gamma_j \text{ if } x_j > \gamma_j \end{aligned} \quad (14)$$

where, $\gamma_j = \max(0, R_j - B_{max})$.

Proof. We will prove the result by contradiction. Let $\vec{y}$ be an optimal schedule such that $y_l = x_l - k,\ k > 0$ for some l at which $x_l \leq \gamma_l$. Distortion introduced in this schedule is

$$\begin{aligned} C(\vec{x}, \vec{y}) &= \sum_{j=1}^{l-1} C(x_j, y_j, j) + C(x_l, y_l, l) + \\ &\quad \sum_{j=l+1}^{N} C(x_j, y_j, j) \end{aligned} \quad (15)$$

Consider the schedule $\vec{\sigma} = (\sigma_1, \ldots, \sigma_N)$ such that

$$\begin{aligned} \sigma_j &= x_j \text{ for } j = l \\ &= y_j \text{ for } j \neq l \end{aligned} \quad (16)$$

It is easy to verify that the schedule $\vec{\sigma}$ is conformant. Distortion introduced in this schedule is

$$\begin{aligned} C(\vec{x}, \vec{\sigma}) &= \sum_{j=1}^{l-1} C(x_j, y_j, j) + C(x_l, x_l, l) \\ &\quad + \sum_{j=l+1}^{N} C(x_j, y_j, j) \\ &= C(\vec{x}, \vec{y}) + C(x_l, x_l, l) - C(x_l, y_l, l) \\ &= C(\vec{x}, \vec{y}) - C(x_l, x_l - k, l) \\ &< C(\vec{x}, \vec{y}) \end{aligned} \quad (17)$$

where Inequality 17 follows from assumption 2. This contradicts the optimality of the schedule $\vec{y}$. For the schedule $\vec{\sigma}$ to be conformant, $y_l \leq x_l$. Hence y_l can not be increased further and as a result, $y_l = x_l$. This proves the first part of the lemma.

To prove the second part of the lemma, assume that $\vec{y}$ is an optimal schedule such that $y_l = \gamma_l - k,\ k > 0$ for some l at which $x_l > \gamma_l$. Distortion introduced in this schedule is

$$\begin{aligned} C(\vec{x}, \vec{y}) &= \sum_{j=1}^{l-1} C(x_j, y_j, j) + C(x_l, y_l, l) + \\ &\quad \sum_{j=l+1}^{N} C(x_j, y_j, j) \end{aligned} \quad (18)$$

Consider the schedule $\vec{\sigma}$ such that

$$\begin{aligned}\sigma_j &= \gamma_j \text{ for } j = l \\ &= y_j \text{ for } j \neq l \end{aligned} \tag{19}$$

It is easy to verify that the schedule $\vec{\sigma}$ is conformant. Distortion introduced in this schedule is

$$\begin{aligned} C(\vec{x},\vec{\sigma}) &= \sum_{j=1}^{l-1} C(x_j, y_j, j) + C(x_l, \gamma_l, l) \\ &\quad + \sum_{j=l+1}^{N} C(x_j, y_j, j) \\ &= C(\vec{x},\vec{y}) + C(x_l,\gamma_l,l) - C(x_l,y_l,l) \\ &= C(\vec{x},\vec{y}) + C(x_l,\gamma_l,l) - C(x_l,\gamma_l - k,l) \\ &< C(\vec{x},\vec{y}) \end{aligned} \tag{20}$$

where Inequality 20 follows from assumption 2. This contradicts the optimality of the schedule $\vec{y}$. The distortion will decrease further if y_l is increased beyond γ_l with the constraint that the new schedule $\vec{y}$ is conformant to the token-bucket regulation. This proves the second part of the lemma. Hence Lemma 1 is proved.

Lemma 2: For a particular l, suppose $x_l \leq \gamma_l$. The schedule $\vec{y}$ is optimal only if

$$\begin{aligned} y_{l-1} &= x_{l-1} \text{ if } x_{l-1} \leq R_{l-1} \\ &\geq R_{l-1} \text{ if } x_{l-1} > R_{l-1} \end{aligned} \tag{21}$$

and

$$\begin{aligned} y_{l+1} &= x_{l+1} \text{ if } x_{l+1} \leq R_{l+1} \\ &\geq R_{l+1} \text{ if } x_{l+1} > R_{l+1} \end{aligned} \tag{22}$$

Proof. From Lemma 1, we note that $y_l = x_l$. Consider an optimal schedule $\vec{y}$ such that $y_{l-1} = x_{l-1} - k,\ k > 0$, given that $x_l \leq \gamma_l$ and $x_{l-1} \leq R_{l-1}$. Distortion introduced in this schedule is

$$\begin{aligned} C(\vec{x},\vec{y}) &= \sum_{j=1}^{l-2} C(x_j, y_j, j) + C(x_{l-1}, y_{l-1}, l-1) \\ &\quad + \sum_{j=l+1}^{N} C(x_j, y_j, j) \end{aligned} \tag{23}$$

Consider the schedule $\vec{\sigma}$ such that

$$\begin{aligned} \sigma_j &= x_j \text{ for } j = l-1 \\ &= y_j \text{ for } j \neq l-1 \end{aligned} \tag{24}$$

It can be easily verified that the schedule is conformant. Distortion introduced in this schedule is

$$\begin{aligned} C(\vec{x},\vec{\sigma}) &= \sum_{j=1}^{l-2} C(x_j, y_j, j) + C(x_{l-1}, x_{l-1}, l-1) \\ &\quad + \sum_{j=l+1}^{N} C(x_j, y_j, j) \\ &= C(\vec{x},\vec{y}) + C(x_{l-1}, x_{l-1}, l-1) \\ &\quad - C(x_{l-1}, y_{l-1}, l-1) \\ &= C(\vec{x},\vec{y}) - C(x_{l-1}, x_{l-1} - k, l-1) \\ &< C(\vec{x},\vec{y}) \end{aligned} \tag{25}$$

where Inequality 25 follows from assumption 2. This contradicts the optimality of the schedule $\vec{y}$ and proves the first part of the lemma. Proceeding like this for every possibility given in the lemma, we can prove the lemma.

Lemma 3: The conformant schedule $\vec{y}$ is optimal only if there does not exist a conformant schedule $\vec{\sigma}$ which can be obtained from $\vec{y}$ by adding a packet at any time instant.

Proof. Lemma 3 is obvious since if $\vec{\sigma}$ were obtained from $\vec{y}$ by adding a packet, the distortion cost of $\vec{\sigma}$ would be less than that of $\vec{y}$. Hence $\vec{y}$ is no longer optimal. Therefore the lemma is proved by contradiction.

Lemma 4: If there are more than one time indices at which the reduction in the distortion cost is the maximum, the algorithm remains optimal even if the packet is added to any of these time indices.

Proof. The algorithm will not be optimal if the eventual distortion cost obtained by adding a packet at any of these time indices is different from others. Consider two time indices m and n at which the reduction in the distortion cost is the maximum. Consider two events:
(i) Adding a packet at time index m.
(ii) Adding a packet at time index n.

The algorithm will not be optimal only if event (i) followed by event (ii) is possible, but event (ii) followed by event (i) is not possible, or vice versa. Without loss of generality, assume that event (i) followed by event(ii) is possible and event (ii) followed by event (i) is not possible. This implies that a packet can be added at time index n. Now if one more packet is added at time index m, it leads to the violation of the token-bucket regulation at some time index l. If the first packet had been added at time index m and the second at time index n, it would have again led to the violation of the token-bucket regulation, since the new schedules obtained by adding the packets are the same. The order in which the packets are added does not matter. This proves the lemma using contradiction. Hence the algorithm is optimal if the packet is added to any of the time indices at which the reduction in the distortion cost is the maximum.

Approximation for the Heuristic Offline Scheduling

Suppose the current time index is j and $x_j \leq \max(0, R_j + B_j - B_{max})$, where B_j is the number of tokens remaining in the bucket at time index j. From Lemma 1, we know that for optimality, $y_j = x_j$. Now consider $x_j > \max(0, R_j + B_j - B_{max})$. Suppose k is the next higher time index at which $x_k \leq \max(0, R_k - B_{max})$. Hence from Lemma 1, we get $y_k = x_k$ for optimality. Now we have to find the output schedule from the time index j to time index $k-1$.

From the functioning of the optimal offline scheduler, we notice that it tries to equalize $C(x_j, y_j, j) - C(x_j, y_j + 1, j) \ \forall j$. If the packets are considered fluid, then this simplifies to:

$$\frac{\partial C(x_j, y_j, j)}{\partial y_j} = \text{constant} \ \forall j \tag{26}$$

The maximum number of packets that can be transmitted between time indices j and $k-1$ is $B_j + \sum_{i=j}^{k-1} R_i$. Hence,

$$\sum_{i=j}^{k-1} y_i \leq B_j + \sum_{i=j}^{k-1} R_i \tag{27}$$

Assume that the maximum number of packets are able to pass through the token-bucket regulator. Hence,

$$\sum_{i=j}^{k-1} y_i = B_j + \sum_{i=j}^{k-1} R_i \tag{28}$$

For $C(x_j, y_j, j) = (1 - \frac{y_j}{x_j})^2$ and using Eq. 26 and 28, we get

$$y_j = x_j \frac{\sum_{i=j}^{k-1} x_i(x_i - x_j) + x_j(B_j + \sum_{i=j}^{k-1} R_i)}{\sum_{i=j}^{k-1} x_i^2} \tag{29}$$

But y_j has to be an integer. Hence,

$$\begin{aligned} y_j &= \lceil x_j \frac{\sum_{i=j}^{k-1} x_i(x_i - x_j) + x_j(B_j + \sum_{i=j}^{k-1} R_i)}{\sum_{i=j}^{k-1} x_i^2} \rceil \\ &= f \end{aligned} \tag{30}$$

From Lemma 1, we know that $y_j \geq \max(0, R_j + B_j - B_{max})$. Hence,

$$y_j = \max(0, R_j + B_j - B_{max}, f) \tag{31}$$

But, $y_j \leq \min(x_j, B_j + R_j)$. Hence,

$$y_j = \min\left(\max(0, R_j + B_j - B_{max}, f), x_j, B_j + R_j\right) \tag{32}$$

This is the heuristic offline scheduler.

References

[1] Zheng Wang. *Internet QoS: Architectures and Mechanisms for Quality of Service*. Morgan Kaufmann Publishers, 2001.

[2] S. Keshav. *An Engineering Approach to Computer Networking*. Addison Wesley, 2001.

[3] Premal Shah and Abhay Karandikar. Optimal Packet Length Scheduling for Regulated Media Streaming. *IEEE Communication Letters*, 7(8):409–411, August 2003.

[4] M. Hamdi, J. W. Roberts and P. Rolin. Rate Control for VBR Video Coders in Broad-Band Networks. *IEEE J. Select. Areas Commun.*, 15:1040–1051, August 1997.

[5] C.-Y. Hsu, A. Ortega and A. R. Reibman. Joint Selection of Source and Channel Rate for VBR Video Transmission under ATM Policing Constraints. *IEEE J. Select. Areas Commun.*, 15:1016–1028, August 1997.

[6] A. Ortega, K. Ramchandran and M. Vitterli. Optimal Trellis-Based Buffered Compression and Fast Approximation. *IEEE Trans. Image Processing*, 3:26–40, January 1994.

[7] Po-Yuen Cheng, Jin Li and C.-C. Jay Kuo. Rate Control for an Embedded Wavelet Video Coder. *IEEE Trans. Circuits Syst. Video Technol.*, 7(4):696–702, August 1997.

[8] Alfio Lombardo and Giovanni Schembra. Performance Evaluation of an Adaptive-Rate MPEG Encoder Matching IntServ Traffic Constraints. *IEEE/ACM Trans. Networking*, 11(1):47–65, February 2003.

[9] P. A. Chou and Z. Miao. Rate-distortion Optimized Streaming of Packetized Media. Technical Report MSR-TR-2001-35, Microsoft Research, Redmond, WA, USA, February 2001.

[10] Z. Miao and A. Ortega. Optimal Scheduling for Streaming of Scalable Media. In *Proc. Asilomar Conference on Signals, Systems, and Computers*, CA, November 2000.

[11] D.A. Turner and K.W. Ross. Optimal Streaming of a Synchronized Multimedia Presentation with Layered Objects. In *IEEE International Conference on Multimedia and Expo*, New York, July 2000.

[12] The video trace files. http://www-tkn.ee.tu-berlin.de/research/trace/trace.html, May 2003.