# A Frame Rate Optimization Framework For Improving Continuity In Video Streaming

Evan Tan and Chun Tung Chou

#### Abstract

This paper aims to reduce the prebuffering requirements, while maintaining continuity, for video streaming. Current approaches do this by making use of adaptive media playout (AMP) to reduce the playout rate. However, this introduces playout distortion to the viewers and increases the viewing latency. We approach this by proposing a frame rate optimization framework that adjusts both the encoder frame generation rate and the decoder playout frame rate. Firstly, we model this problem as the joint adjustment of the encoder frame generation interval and the decoder playout frame interval. This model is used with a discontinuity penalty virtual buffer to track the accumulated difference between the receiving frame interval and the playout frame interval. We then apply Lyapunov optimization to the model to systematically derive a pair of decoupled optimization policies. We show that the occupancy of the discontinuity penalty virtual buffer is correlated to the video discontinuity and that this framework produces a very low playout distortion in addition to a significant reduction in the prebuffering requirements compared to existing approaches. Secondly, we introduced a delay constraint into the framework by using a delay accumulator virtual buffer. Simulation results show that the the delay constrained framework provides a superior tradeoff between the video quality and the delay introduced compared to the existing approach. Finally, we analyzed the impact of delayed feedback between the receiver and the sender on the optimization policies. We show that the delayed feedbacks have a minimal impact on the optimization policies.

#### I. INTRODUCTION

Video continuity is the length of time the video is being played without interruptions. A low video continuity would result in a stop-start video that is known to impact on the viewer perceived video quality [1]. One of the main causes of video discontinuity in streaming video is due to buffer underflow caused by for example network delay. The traditional way to reduce the buffer underflow occurrences is to prebuffer video at the decoder. The prebuffering amount needs to be sufficiently large to maintain video continuity. However, a large prebuffering introduces unwanted initial delays into the system.

To reduce the amount of prebuffering required while maintaining video continuity, current approaches make use of adaptive media playout (AMP) [2], [3], [4]. AMP approaches reduce the *playout frame rate* of the decoder in order to avoid buffer underflows, this is because slowing down the playout frame rate is preferable to halting the playout [5]. The playout frame rate is defined as the rate of frames being removed from the decoder buffer for decoding and playout to the viewer. AMP has been shown to reduce prebuffering while maintaining video continuity [3]. However, it introduces *playout distortion* to the viewers, this is because the video is being played slower than its natural playout frame rate (also known as the video capture frame rate). Furthermore, AMP schemes adjust the playout frame rate independent of the encoder strategy, this potentially introduces more playout distortion than required. While the playout distortion can be reduced by limiting the playout slowdown, it potentially affects the video continuity. Moreover, slowing down the playout frame rate increases the viewing latency and introduces additional delay into streaming system.

In this paper, we aim to reduce the amount of prebuffering required by dealing with the continuity of the video in a reactive manner. We realize this aim by proposing a framework that performs frame rate control at both the encoder and decoder as well as introduce a way to constrain the viewing latency. The key idea is that, if the network bandwidth drops and the number of frames in the decoder buffer is low, the encoder should send more frames to the decoder to prevent buffer underflow from occurring, thus maintaining playout continuity. In order to approach

E. Tan is with the School of Computer Science and Engineering, University of New South Wales, Sydney NSW 2052, Australia (email: evant@cse.unsw.edu.au)

C. T. Chou is with the School of Computer Science and Engineering, University of New South Wales, Sydney NSW 2052, Australia (email: ctchou@cse.unsw.edu.au)

this systematically, we formulate an optimization problem that takes into account the video continuity, video quality and overall playout delay. The contributions of this paper are:

- 1) We propose a frame rate control framework that jointly adjusts the encoder frame generation rate and the playout frame rate. This distinguishes our framework from conventional approaches that do not perform frame rate control and AMP approaches that only adjusts the playout frame rate.
- 2) We derive an optimization formulation of the frame rate control framework using the technique of virtual buffer. We then use Lyapunov optimization [6] on this model to systematically derive the optimization policies. We show that these policies can be decoupled into separate encoder and decoder optimization policies with feedback between the two. This allows for a distributed implementation of the policies. We demonstrate that this framework produces a very low playout distortion in addition to a significant reduction in the prebuffering requirements compared to existing approaches.
- 3) A delay constraint that reduces the accumulated delay from playout slowdowns. We then show that the delay constrained framework provides a superior tradeoff between the video quality and the delay introduced compared to the existing approach.
- 4) An analysis of the impact of delayed feedback between the receiver and the sender. We show that the delayed feedbacks have a minimal impact on the optimization policies.

This paper is organized as follows. Section II reviews the current approaches. Section IV demonstrates how the frame rate control problem can be modelled. Section V demonstrates how a delay constraint can be introduced into the framework. Section VI analyzes the delayed feedback on the optimization policies. Section VII describes the video quality functions that can be used with this framework. Section VIII evaluates the performance of the proposed improved framework. Section IX concludes this paper.

## II. RELATED WORK

Prebuffering video data has been studied in [7], [8], [9]. These techniques are focused on calculating the correct amount of prebuffered data based on a mathematical model to avoid the occurrence of a buffer underflow once the video playout has started. However, these techniques do not take into account of the varying playout rates and the resulting reduction in prebuffering into their models. Furthermore, prebuffering introduces unwanted delay into the system and is known to have an impact on the user perceived quality of the video [10].

Sender rate adaptation techniques such as encoder rate control [11], [12] and scalable rate control[13] achieve video continuity by ensuring the video rate matches the network bandwidth. However, these approaches do not take into account of the network delay variation that might occur (e.g. due to network jitter). Moreover, it has been demonstrated that coupling AMP with sender rate adaptation techniques can further reduce the prebuffering requirements [3].

AMP is another class of techniques that is used to improve video continuity. The main idea behind AMP is to reduce the playout rate of the received media. This reduces the departure rate of the media in the decoder buffer and potentially allows the buffer to be filled up to safer levels.

AMP has been studied extensively for audio applications, [14], [15], [16], [17], [18], [19]. These audio AMP techniques depend on audio scaling techniques such as WSOLA [20], which allows audio to be scaled without changing its pitch. A recent work by Damnjanovic et al has shown that audio and video scaling can be done in real-time [21]. In this paper, we focus on AMP for video.

AMP for video involves scaling the frame intervals to slowdown or speedup the playout rate. The slowdown or speedup is triggered by a threshold set on the buffer occupancy. Once the buffer occupancy drops below the threshold, AMP will slow down the playout rate and vice versa. There have been studies conducted on the dynamic adjustment of this threshold [2], [22], [23], [24]. The adjustment is normally based on the network condition, the poorer the network condition, the higher the threshold. These techniques mainly based the threshold on the buffer occupancy. We instead record the difference between the receiving frame interval and the playout frame interval into a virtual buffer and treat it as a penalty, which is equivalent to soft thresholding.

AMP has also been integrated into the design of packet schedulers [25], [26], [27], [28], [29]. These techniques tend to slowdown the playout rate for important video packets. This ensures that the more important video packets have a higher probability of meeting its deadline and thus avoid being dropped. We do not focus on packet scheduling in this paper and our proposed framework could complement any packet scheduling scheme.

Another aspect of AMP being studied is the smoothness of transition between playout rate adjustment [4], [30], [31]. The goal of these approaches is to ensure that adjustments made to the playout rate is done as smoothly as possible so as to reduce any noticeable effects to the viewers. We do not focus on rate smoothing in the current paper and again any smoothing scheme can be used within the framework.

Steinbach et al [32] and Kalman et al [3] both examined the trade-off between delay and buffer underflow using a two state Markov models. Kalman et al further proposed AMP-Initial, AMP-Robust and AMP-Live. AMP-Initial slows down the playout rate until the buffer reaches a certain target level, this produces a lower perceived initial buffering delay to the viewer. AMP-Robust slowdowns the playout rate if the current buffer occupancy falls below the target buffer level while AMP-Live slowdowns or speedups the playout rate to maintain the target buffer level.

These AMP approaches mainly examine only the effects of adjusting the playout frame rate independently and do not consider any encoder strategy to reduce playout distortion. While our proposed approach aims to examine the effects on video quality and viewing delay based on the adjustment of both the encoder frame generation rate and the playout frame rate. Slowing down the playout frame rate also introduces viewing latency, we will propose a way to constrain this latency in our proposed approach.

## III. FRAME RATE CONTROL FOR VIDEO CONTINUITY



Figure 1. Encoding-decoding flow with encoder frame generation rate i(t), sending frame rate  $\mu(t)$ , receiving frame rate  $\lambda(t)$  and playout frame rate o(t). All rates are in frames per seconds.

Figure 1 shows a typical video transmission scenario. The encoder generates video frames at a rate of i(t) frames per second (fps) at time t into the encoder buffer. The network transport protocol then transmits the video data from the encoder buffer into the network at a rate of  $\mu(t)$  fps. The transmitted video data will be received at the decoder at a rate of  $\lambda(t)$  fps in its buffer. The decoder then proceeds to playout the received video data from the decoder buffer at a rate of o(t) fps. Video data arriving after the playout deadline are assumed to be lost. In the scenario described in fig. 1, there are two mechanisms that can be used to maintain video continuity: reducing playout frame rate o(t) and increasing the encoder frame generation rate i(t).

Our analytical model takes network delay into consideration but does not consider packet losses <sup>1</sup>. In particular, an important factor that affects the performance is the inter-frame delay, which is defined as the difference of network delays for consecutive video frames. We assume that the frames are received into the decoder buffer in display order, so reordering is done before the frames enter the decoder buffer. Thus, inter-frame delays can be affected by network jitter and packet losses. Notice that this model allows us to focus on calculating the amount of frames arriving into and departing from the decoder buffer such that decoder buffer underflow can be avoided, and video continuity can be maintained. We also assume, in this paper, that the encoder is able to generate frames faster than the natural playout frame rate. This is a reasonable assumption since there are existing encoders such as x264 [33] that can encode frames in real-time faster than a typical natural playout frame rate of 30 frames per second (fps).

Slowing down the video playout frame rate o(t) is termed in the literature as AMP. The idea is that the slower video playout allows time for the buffer to fill up to the required level without the need to stop the video for rebuffering.

Another way to maintain video continuity is to increase the *encoder frame generation rate* i(t), which refers to the amount of frames the encoder actually sends out into the network and it is important to point out that the encoder frame generation rate is *not* a temporally scaled frame rate. Note that we purposely chose the term encoder frame generation rate to differentiate it from the commonly used term of encoder frame rate because they represent two *different* concepts. The meaning of encoder frame generation rate is best illustrated by an example: when no frame rate control is used, the encoder frame generation rate i(t) is the natural playout frame rate. Let us

<sup>1</sup>Note that even though we do not consider packet loss explicitly, the simulation results in section VIII show that our optimisation framework works well in the presence of packet loss.

assume that to be 30 fps; when frame rate control is used, the encoder may increase the encoder frame generation rate i(t) to say 60 fps to quickly fill up the decoder buffer to maintain continuity. Note that these 60 frames are still encoded using the same natural playout frame rate of 30 fps, so they form the next 2 seconds of video. This example illustrates that a higher encoder frame generation rate means more than one second of video is generated in one second but the video is always encoded using the same natural playout frame rate.

Increasing i(t) allows potentially more frames to reach the decoder and increase the decoder buffer level. This, in turn, helps to improve the continuity of the video. However, increasing i(t) is likely to cause the video bitrate to increase as more frames are produced per second by the encoder. To ensure that the video bitrate does not exceed the available bandwidth, we introduce additional compression to the video such that the higher the encoder frame generation rate i(t), the higher the compression applied.

For a given encoder generation rate i(t), the higher compression is obtained by the rate controller adjusting the encoder such that the average frame size produced by the encoder is:

average frame size 
$$= r(i(t)) = \frac{ABR(t)}{i(t)}$$
 (1)

where ABR(t) is the available bandwidth<sup>2</sup> at time t. In practice, the average frame size produced by a rate control strategy may not satisfy (1). However, (1) is what most rate control schemes try to meet as they use (1) as part of their fluid flow model to determine the amount of bits to allocate to a frame [11], [12], [34].

It can be shown that when (1) is satisfied by the rate controller, the resulting video bitrate will not exceed the available bandwidth for different values of the encoder frame generation rate i(t). Also, the average frame size decreases when i(t) is increased which would tend to reduce the frame quality. Finally, a higher i(t) also allows more frames to be sent to the decoder, thus it allows us to increase the buffer level and maintain video continuity.

Reducing o(t) and increasing i(t) increases playout distortion and reduces frame quality respectively while improving video continuity. This suggests that an optimal trade-off need to be found. To do this, we model the frame rate control problem as an optimization problem and make use of Lyapunov optimization to obtain policies that help determine the optimal trade-off.

In this paper, we adjust the encoder frame generation rate i(t) and playout frame rate o(t) by adjusting the encoder frame generation interval  $\frac{1}{i(t)}$  and the playout frame interval  $\frac{1}{o(t)}$  respectively. The reason we chose to adjust the intervals is because it allows the optimization problem to be concave and, therefore, easier to solve. This issue will be discussed in more detail in Section VII.

#### IV. DISCONTINUITY PENALTY FOR FRAME RATE OPTIMIZATION

A. Buffering Criteria

$$\lambda(t) \longrightarrow \underbrace{1}_{b(t)} \longrightarrow o(t)$$

Figure 2. Receiver buffer model.

Since the video discontinuity is correlated to decoder buffer underflows, we first study how buffer underflow occurs. To do this, we make use of the receiver model illustrated in fig. 2. We assume that the sender, network and receiver all work in slotted time. At time slot t, the receiver receives  $\lambda(t)$  frames (receiving frame rate) and stores it into the receiver buffer. Simultaneously, o(t) frames are being removed from the buffer and played out to the viewer (playout rate) at time t. Let b(t) be the amount of buffered video frames in the buffer at time t and T be the length of the sequence in time slots, then to avoid an underflow the following condition needs to be met:

$$b(t) \ge \sum_{\tau=t}^{T} o(\tau) - \sum_{\tau=t}^{T} \lambda(\tau) \qquad \forall T \ge t$$
(2)

<sup>2</sup>ideally the encoder should use  $ABR(t + d_s)$ , where  $d_s$  is the sender buffer delay. However, since this is difficult to determine, we use ABR(t) to approximate  $ABR(t + d_s)$ .

Equation (2) intuitively means that, to avoid a buffer underflow, the cumulative buffer drainage for the rest of the sequence playing time should not exceed the current buffer occupancy b(t).

Now we refine the above model to make use of *frame intervals*. This is done to build up a system model using frame intervals. Basically to make use of the frame intervals, we set  $\lambda(t) = \frac{1}{r(t)}$  and  $o(t) = \frac{1}{p(t)}$ . Equation (2) then becomes:

$$b(t) \ge \sum_{\tau=t}^{T} \frac{r(\tau) - p(\tau)}{r(\tau) p(\tau)}$$
(3)

We assume that  $r_{min} \leq r(t) \leq r_{max}$  and  $p_{min} \leq p(t) \leq p_{max}$ , i.e. both r(t) and p(t) are bounded. That would mean that we can approximate (3) as:

$$b(t) r_{min} p_{min} \ge \sum_{\tau=t}^{T} r(\tau) - p(\tau)$$
(4)

Note that the choice of using  $r_{min}$  and  $p_{min}$  results in a more conservative bound. An alternative bound, which is looser, is to replace the left-hand-side of (4) by  $b(t) r_{max} p_{max}$ . We will show in simulation results (section VIII) that these two choices give similar results.

If we divide (4) by T - t, the remaining time slots left, we can estimate the buffer underflow bound *per time slot*. This will result in:

$$\frac{b(t)\,r_{min}\,p_{min}}{T-t} \ge \hat{r} - \hat{p} \tag{5}$$

where  $\hat{r}$  and  $\hat{p}$  are the averages of r(t) and p(t) respectively. Equation (5) provides us with a way to design the optimization policy to avoid buffer underflow in a time slot. Since Lyapunov optimization works on a per time slot basis, we prefer (5) over (4). Essentially, we need to design a policy that produces a *receiving frame interval* r(t) and a *playout frame interval* p(t) in such a way that the above bound is met.

#### B. System Model



Figure 3. System model showing the frame intervals.

We now show how r(t) and p(t) are produced in the complete system model. Fig. 3 illustrates the system model. In a time slot t, the encoder at the sender produces video frames at intervals of f(t) and stores them into the sender buffer. The sender sends the video frames in its buffer at intervals of s(t). Note that f(t) and s(t) are the *encoder* frame generation interval and the sending frame interval respectively. Simultaneously, the receiver receives video frames from the sender at intervals of r(t) and puts them into the decoder buffer. The decoder in the receiver plays out the video frames at intervals of p(t) to the viewer, p(t) is the playout frame interval. The network will also produce a forward delay of  $d_f$  and a backward delay of  $d_b$ .

The goal of our framework is to jointly adjust the encoder frame generation interval f(t) and the playout frame interval p(t) to maintain video continuity. To simplify the model, we assume that f(t) = s(t), while this essentially assumes no delay caused by the sender buffer, delays caused by the sender buffer is simulated in our experiments later on. We also assume that f(t) is bounded within the range  $[f_{min}, f_{max}]$  and r(t) is defined by a network delay variation function F(s(t)) as : r(t) = F(s(t)) = F(f(t)), since f(t) = s(t).

This means that we can represent the delay variation based on the encoder frame generation interval f(t). In this paper, we specify F(f(t)) as:

$$F(f(t)) = e(t) \times f(t) \tag{6}$$

Where e(t) is the frame interval scaling factor due to delay variations from the network. In practice, we estimate e(t) at the receiver by:

$$e(t) = \frac{r(t)}{f(t - d_f)}\tag{7}$$

Where  $d_f$  is the forward delay between the sender and receiver. Note that if there is no delay (i.e.  $d_b = d_f = 0$ ), it will mean that F(f(t)) = r(t).

## C. General Optimization Problem

There are three main objectives that we want to optimize: 1. frame quality, 2. playout distortion and 3. continuity. Frame quality is defined as the perceived visual quality of the video. This will be represented as a frame quality function g(f(t)), where g(f(t)) is an increasing function of f(t). Playout distortion is the perceived distortion when the playout rate deviates from the natural frame rate and will be represented by a function h(p(t)), where h(p(t)) is a convex function of p(t). Both g(f(t)) and h(p(t)) are non-negative functions, i.e.  $g(f(t)) \ge 0$  and  $h(p(t)) \ge 0$ , and are assumed to be uncorrelated. We will suggest a specific form for g(f(t)) and h(p(t)) later in Section VII. Continuity is the length of time the video is played without interruptions due to buffer underflow. We ensure continuity in this framework by ensuring that a *virtual buffer stabilizes*, this concept will be explained further in the next section.

With the system model described in the previous section, we now formulate a general optimization problem:

Maximize: 
$$g(f(t)) - h(p(t))$$
 (8)

Subject to: 
$$U(t)$$
 is stable (9)

$$f_{min} \le f(t) \le f_{max} \tag{10}$$

$$p_{min} \le p(t) \le p_{max} \tag{11}$$

Where U(t) is the virtual buffer representing the *discontinuity penalty*. Since the objective (8) is separable, maximizing (8) can be seen as maximizing the frame quality function g(f(t)) and minimizing the playout distortion function h(p(t)). The constraint (9) is the continuity constraint. Constraints (10) and (11) are the limits set on f(t) and p(t) respectively.

To see how the above general optimization problem is derived, we first replace (9) with a continuity constraint derived from (5):

$$\mathbb{E}\{F(f(t)) - p(t)\} < \beta(t) \tag{12}$$

where  $\beta(t) = \frac{b(t) r_{min} p_{min}}{T-t}$ . Constraint (12) can be satisfied by either decreasing f(t) and/or increasing p(t). However, decreasing f(t) would result in a lower frame quality given by g(f(t)) and increasing p(t) would result in a higher playout distortion given by h(p(t)). The optimization policy would need to handle these tradeoffs. To solve this optimization problem, we make use of the concepts of virtual buffer and Lyapunov optimization.

#### D. Virtual Buffer And Stability

Virtual buffers are a concept introduced by Neely et al [35] to replace certain constraints of an optimization problem. To determine a suitable virtual buffer for our problem, we make an initial virtual buffer design to represent the continuity of the video. The virtual buffer will be updated at every time slot t, and is updated as:

$$U(t+1) = [U(t) - p(t)]^{+} + F(f(t))$$
(13)

where U(0) = 0. The virtual buffer U(t) is lower bounded by 0 (i.e. always non-negative). This virtual buffer can be seen as the discontinuity penalty. If F(f(t)) is higher than p(t), it means that the network throughput is lower

To ensure that U(t) does not keep increasing, we want U(t) to *stabilize*. We will show later that by stabilizing U(t), we can ensure that the video continuity is preserved. The definition of stability used here is  $\mathbb{E}\{U\} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{U(\tau)\} < \infty$ . This intuitively means that the buffer is stable if it does not grow infinitely large over time. We also like the virtual buffer to meet the continuity constraint (12) when it stabilizes. To do that, we extend the initial virtual buffer (13) as:

$$U(t+1) = [U(t) - p(t) - \beta(t)]^{+} + F(f(t))$$
(14)

To see how it works, notice that for the discontinuity penalty U(t) to grow infinitely, the following condition needs to be met:  $F(f(t)) - p(t) > \beta(t)$ . Therefore, in order for U(t) to stabilize the following condition needs to be true:  $F(f(t)) - p(t) \le \beta(t)$ . This is the continuity constraint as defined in (12), so when U(t) stabilizes, the continuity constraint will be met. The discontinuity penalty U(t) is maintained at the receiver in our design.

Thus, with the stability of the discontinuity penalty U(t) as a constraint, we then obtain the general optimization problem presented in Section IV-C. We will demonstrate in Section VIII-C, using simulations, that U(t) is positively correlated with the video discontinuity.

#### E. Lyapunov Optimization Derivation

We show here how we convert the optimization problem presented in Section IV-D into a separate encoder and decoder optimization policies using Lyapunov optimization. We assume that there is no network delay between the sender and receiver (i.e.  $d_f = d_b = 0$ ). This is to simplify the analysis presented here. We will relax this assumption in Section VI.

We define a Lyapunov function L(U(t)) to represent the "energy" of the discontinuity penalty U(t) at time t, this can be any arbitrary non-negative function. We use the following Lyapunov function in this paper:

$$L(U(t)) \triangleq \frac{U^2(t)}{2} \tag{15}$$

We then define the one-step conditional Lyapunov drift  $\Delta(U(t))$  as:

$$\Delta(U(t)) \triangleq \mathbb{E}\{L(U(t+1)) - L(U(t))|U(t)\}$$
(16)

Equation (16) can be understood as the expected change in the energy in one time slot step. The goal of Lyapunov optimization is to show that this energy reduces or stays the same at each slot (i.e. (16) produces a negative or zero drift). This would ensure the stability of the buffer, which in turn enforces the continuity of the video playout. To show that U(t) stabilizes, we need to convert the buffer update equation (14) into a one-step drift (16).

To do that, we square the buffer update equation (14), divide it by two, and take expectations of the result (see Section A in the appendix for details), we will get the following expression:

$$\Delta(U(t)) \le B - U(t)\mathbb{E}\{\beta(t) + p(t) - F(f(t))|U(t)\}$$
(17)

Where B is a constant defined as:

$$B = \frac{1}{2} \left( r_{max}^2 + \left( p_{max} + \frac{T \, r_{min} \, p_{min}}{T - t} \right)^2 \right) \tag{18}$$

From (17), we can use the results in [6] to prove that U(t) stabilizes (see Section D in the appendix). Once U(t) is proven to stabilize, it can be shown using the results in [35] that the continuity constraint (12) is satisfied. However, to optimize the frame quality utility g(f(t)) and the playout distortion h(p(t)), we need to massage the equation more. By subtracting from both sides, the term  $V\mathbb{E}\{g(f(t)) - h(p(t))|U(t)\}$ , which is the expectation of (8) scaled by a positive constant V > 0, and by rearranging the terms. We get:

$$\Delta(U(t)) - V\mathbb{E}\{g(f(t)) - h(p(t))|U(t)\}$$

$$\leq B - \mathbb{E}\{U(t)\beta(t)|U(t)\}$$

$$- \mathbb{E}\{Vg(f(t)) - U(t)F(f(t))|U(t)\}$$

$$- \mathbb{E}\{U(t)p(t) - Vh(p(t))|U(t)\}$$
(19)

The third term on the right hand side of (19) is a function of the encoder frame generation interval f(t) only and represents the sender optimization policy. The last term of (19) is a function of the playout frame interval p(t)and represents the receiver optimization policy.

## To summarize:

1) Encoder optimization policy: From the third term of (19), based on the frame interval scaling factor e(t) (defined in (7)) and discontinuity penalty U(t) feedback from the receiver. The encoder in the sender will calculate F(f(t)) using (6), and it will choose f(t) at each time slot as the solution of the following optimization:

Maximize: 
$$Vg(f(t)) - U(t)F(f(t))$$
  
Subject to:  $f_{min} \le f(t) \le f_{max}$  (20)

2) Decoder optimization policy : From the last term of (19), the decoder in the receiver will observe the current discontinuity penalty U(t) and choose p(t) at each time slot as the solution of the following optimization:

Maximize: 
$$U(t)p(t) - Vh(p(t))$$
  
Subject to:  $p_{min} \le p(t) \le p_{max}$  (21)

Notice that the optimization policies are decoupled into separate optimization subproblems for playout frame and encoder frame generation intervals. Note that the decoder is responsible for updating U(t) using equation (13) and sending the value of U(t) to the encoder. Furthermore, under appropriate conditions, the decoupled problems are convex. This makes the problem easier and more flexible to solve. Given that Lyapunov optimization minimizes the right of (19) instead of the original optimization problem (8) (which is hard to solve), the objective function value realized by Laypunov optimization is sub-optimal but its deviation from the optimal value can be controlled (see Section D in the appendix).

#### V. DELAY CONSTRAINED FRAME RATE OPTIMIZATION

While slowing down the video playout allows us to reduce the occurrences of buffer underflows and preserve the video continuity, it introduces extra viewing latency to the viewers. This becomes an issue when the system has a delay budget, as frequent playout slowdowns might cause the delay budget to be exceeded.

In this section, we examine the problem when there is a constraint imposed on the viewing latency. We focus on constraining the additional playout latency generated by slowing down the playout. More specifically, we want to impose a constraint on how often playout slowdowns occur and how much the playout can be slowed down. This is done by introducing another virtual buffer, called the *delay accumulator*, into the problem to represent the constraint on the accumulated delay due to playout slowdowns. We then use Lyapunov optimization to ensure that this constraint is met.

# A. Delay Constrained Policy Design

The delay constraint can be described as constraining the accumulated playout slowdowns used over the lifetime of the whole video sequence. Specifically, let  $\theta$  be the maximum playout slowdown delay tolerable for the video application and  $p_n$  represent the natural playout interval of the video. Then, the constraint could be written as:

$$\sum_{\tau=0}^{T-1} (p(\tau) - p_n) \le \theta \tag{22}$$

$$\mathbb{E}\{p(t) - p_n\} \le t_d \tag{23}$$

Then, what remains to be done is to design a policy that ensures that constraint (23) is met at each time slot. It can be seen that by adding constraint (23) to the general optimization problem presented in Section IV-D, we obtain a problem that provides video continuity while ensuring that the delay constraint is met.

#### B. Delay Accumulator Virtual Buffer

To apply the delay constraint (23) into the framework using Lyapunov optimization, we again make use of the virtual buffer concept. We introduce another virtual buffer named as the *delay accumulator*. The delay accumulator is a virtual buffer that keeps track of the accumulated delay caused by playout slowdowns. Everytime a playout slowdown is perform, the delay accumulator will increase correspondingly. However, the delay accumulator will reduce when playout speed up is perform. Formally, the buffer dynamics to describe the delay accumulator for each time slot would be:

$$X(t+1) = [X(t) - p_n - t_d]^+ + p(t)$$
(24)

Note that the terms  $p_n$ , p(t) and  $t_d$  are taken from constraint (23) above. Therefore, by showing that the delay accumulator X(t) stabilizes, we can show that the delay constraint (23) can be satisfied [35]. This also means that the delay constraint can be written as: X(t) is stable. We now show how Lyapunov optimization can be used to derive optimization policies to solve the above problem.

#### C. Lyapunov Optimization Derivation

Note that there are two buffers in the problem, the discontinuity penalty U(t) and the delay accumulator X(t). To stabilize both of these simultaneously, we first redefine the Lyapunov function to be:

$$L(U(t), X(t)) \triangleq \frac{U^2(t) + X^2(t)}{2}$$
 (25)

The one step conditional drift also needs to consider both buffers and is defined as:

$$\Delta(U(t), X(t)) \triangleq \mathbb{E}\{L(U(t+1), X(t+1)) - L(U(t), X(t)) | U(t), X(t)\}$$

$$(26)$$

To shorten the formulas, we use  $\Delta$ , U, X, p and f to represent  $\Delta(U(t), X(t))$ , U(t), X(t), p(t) and f(t) respectively. By squaring (14) and (24), taking expectations and dividing by 2, we get (see Section B in the appendix for details):

$$\Delta(U, X) \le B + C - U\mathbb{E}\{p - F(f)|U, X\} - X\mathbb{E}\{p_n + t_d - p|U, X\}$$

$$(27)$$

where B is defined as in (18) and  $C = \frac{1}{2} \left( p_{max}^2 + (p_n + t_d)^2 \right).$ 

It can be proven that (27) results in stability for both the discontinuity U and the delay accumulator X [35] (see Section E of the supplementary materials). Furthermore, it can be proven that the stabilization of X implies that the constraint (23) can be met by using the results from [35].

To optimize the frame quality and the playout distortion, we subtract  $V\mathbb{E}\{g(f(t)) - h(p(t))|U(t)\}$  from both sides of (27) to obtain:

$$\Delta(U, X) - V\mathbb{E}\{g(f) - h(p)|U, X\}$$

$$\leq B + C - X\mathbb{E}\{p_n + t_d|U, X\}$$

$$- \mathbb{E}\{Vg(f) - UF(f)|U, X\}$$

$$- \mathbb{E}\{Up - Vh(p) - Xp|U, X\}$$
(28)

Note that the encoder optimization policy (second last term) remains the same as (20). The last term shows that decoder optimization policy (21) has an additional penalty term -Xp. This will mean that as X increases, the decoder will get penalized more for high playout interval p values. This will encourage the decoder to choose a lower p whenever the accumulated delay in X is high. Lastly, the performance bound for the Lyapunov optimization can be derived (see Section E of the appendix).

## VI. NETWORK DELAY IMPACT

In Sections IV and V. we showed how Lyapunov optimization can derive optimization policies that help ensure the video continuity and enforce a delay constraint. However, the assumption made in those sections for the derivations is that there is no network delay. Specifically, we assumed that the feedback delay from the receiver to the sender is non-existent. In this section, we relax this network delay assumption and analyze the impact of network delay on the optimization policies.

Recall that the network generates a forward delay of  $d_f$  and a backward delay of  $d_b$  from the system model discussed in Section IV-B. With network delay, the discontinuity penalty U(t) updating in the receiver is performed as:

$$U(t+1) = [U(t) - \gamma(t)]^{+} + F(f(t-d_f))$$
(29)

where  $\gamma(t) = p(t) + \beta(t)$  and recall that  $\beta(t) = \frac{b(t) r_{min} p_{min}}{T-t}$ . Note that the difference between the previously presented discontinuity penalty buffer dynamics in (13) and above is that (29) is based on the forward delayed encoder frame generation interval  $f(t - d_f)$ . Furthermore, the sender relies on feedback from the receiver, so it chooses f(t) based on a backward delayed  $U(t - d_b)$ . There are two possible issues that might impact on the optimization policies when delays are present:

- 1) the delayed discontinuity penalty U(t) feedback from the receiver to the sender means the encoder optimization policy will need to make use of  $U(t d_b)$ .
- 2) the current choice of f(t) at the sender would only affect receiver  $d_f$  time slots later.

What we need to do is to derive optimization policies that take the above two issues into account and show that these policies stabilizes the discontinuity penalty U(t).

To begin, note that (29) can be represented recursively as:

$$U(t+1) = [U(t) - \gamma(t)]^{+} + F(f(t-d_{f}))$$

$$U(t) = [U(t-1) - \gamma(t-1)]^{+} + F(f(t-d_{f}-1))$$

$$\vdots$$

$$U(t-d_{b}+1) = [U(t-d_{b}) - \gamma(t-d_{b})]^{+} + F(f(t-d_{f}-d_{b}))$$
(30)

This implies that U(t) can be recursively defined as:

$$U(t) \le \left[ U(t - d_b) - \sum_{\tau = t - d_b}^{t-1} \gamma(\tau) \right]^+ + \sum_{\tau = t - d_b - d_f}^{t-d_f - 1} F(f(\tau))$$
(31)

Issue 2 suggests that we need to predict  $d_f$  time slots in the future. To do that, we change the buffer updating equation (29) into a  $d_f$  slot update:

$$U(t + d_f + 1) \le \left[ U(t) - \sum_{\tau=t}^{t+d_f} \gamma(\tau) \right]^+ + \sum_{\tau=t-d_f}^{t} F(f(\tau))$$
(32)

What (32) means is that the receiver updates the discontinuity penalty U(t) not only based on the known current values  $\gamma(t)$  and  $f(t-d_f)$  but also using the predicted values  $d_f$  slots in the future,  $[\gamma(t+1) \dots \gamma(t+d_f)]$  and  $[f(t-d_f+1)\dots f(t)]$ . Note that  $F(f(\tau))$  in (32) is calculated from  $d_f$  time steps in the past. This is because the current f(t) only affects the discontinuity penalty  $d_f$  time slots later which will be  $U(t + d_f + 1)$ . The  $d_f$  step buffer dynamics can be proved to obtain stability by using the *T-slot Lyapunov drift* [6].

To show how the T-slot Lyapunov drift achieves stability, we first convert the 1-step Lyapunov drift in (16) into a  $d_f$ -step drift:

$$\Delta(U(t)) \triangleq \mathbb{E}\{L(U(t+d_f+1)) - L(U(t))|U(t)\}$$
(33)

We now use the following shortened notations to simplify the equations:  $U \triangleq U(t)$ ,  $U_{d_b} \triangleq U(t - d_b)$ ,  $\gamma \triangleq \sum_{\tau=t-d_b}^{t-1} \gamma(\tau)$ ,  $\gamma_{d_f} \triangleq \sum_{\tau=t}^{t+d_f} \gamma(\tau)$ ,  $F \triangleq \sum_{\tau=t-d_b-d_f}^{t-d_f-1} F(f(\tau))$  and  $F_{d_f} \triangleq \sum_{\tau=t-d_f}^{t} F(f(\tau))$ . Using the same Lyapunov function (15) as in the previous section and drift definition (33). If we square (32),

divide it by two and take expectations (see Section C of the appendix for details), we get:

$$\Delta(U) \le B' - \mathbb{E}\left\{U\gamma_{d_f} \middle| U\right\} + \mathbb{E}\left\{UF_{d_f} \middle| U\right\}$$
(34)

Where:

$$B' = \frac{d_f}{2} \left( r_{max}^2 + \left( p_{max} + \frac{T r_{min} p_{min}}{T - t} \right)^2 \right)$$
(35)

Equation (34) can be shown to achieve stability [36], effectively settling issue 2. However, to deal with issue 1, we need to show how the feedbacked discontinuity penalty  $U(t - d_b)$  affects (34). To do that, we substitute the recursively defined U(t) (31) into (34):

$$\Delta(U) \leq B' - \mathbb{E}\left\{U\gamma_{d_f}\big|U\right\} + \mathbb{E}\left\{\left(\left[U_{d_b} - \gamma\right]^+ + F\right)F_{d_f}\big|U\right\}\right\}$$
(36)

Recall the definitions of F and  $F_{df}$ , and given that F(.) is an increasing function of f(t) (see (6)), this implies that F and  $F_{d_f}$  can be bounded as  $F \leq d_b r_{max}$  and  $F_{d_f} \leq (d_f + 1) r_{max}$  respectively. It then follows that  $FF_{d_f}$ can be bounded as:

$$FF_{d_f} \le d_b(d_f + 1)r_{max}^2 \tag{37}$$

Note that  $[U_{d_b} - \gamma]^+ \leq U_{d_b}$ . Thus, we use (37) in (36), we get:

$$\Delta(U) \leq B'' - U\mathbb{E}\left\{\gamma_{d_f} \middle| U\right\} - \mathbb{E}\left\{-U_{d_b}F_{d_f} \middle| U\right\}$$
(38)

where  $B'' = B' + d_b(d_f + 1) r_{max}^2$  with B' from (35). Equation (38) in that form can be proven to stabilize by using the results from [36] (see Section F in the appendix), thus settling issue 1.

As in Section IV-E, to cater for utility optimization, we subtract from both sides, the term  $V\mathbb{E}\{g(f(t))$  $h(p(t))|U\}$  and rearrange the terms to obtain:

$$\Delta(U) - V\mathbb{E}\{g(f(t)) - h(p(t))|U\}$$

$$\leq B'' - \mathbb{E}\{U\gamma_{d_f} - Vh(p(t))|U\}$$

$$- \mathbb{E}\{Vg(f(t)) - U_{d_b}F_{d_f}|U\}$$
(39)

Using the definitions of  $\gamma_{d_f}$  and  $F_{d_f}$  and that  $\gamma(t) = p(t) + \beta(t)$ . Equation (39) can be rewritten as:

$$\Delta(U) - V\mathbb{E}\left\{g(f(t)) - h(p(t))|U\right\}$$

$$\leq B'' - U\mathbb{E}\left\{\sum_{\tau=t+1}^{t+d_f} p(\tau) + \sum_{\tau=t}^{t+d_f} \beta(\tau) \middle| U\right\}$$

$$+ U_{d_b}\mathbb{E}\left\{\sum_{\tau=t-d_f}^{t-1} F(f(\tau)) \middle| U\right\}$$

$$- \mathbb{E}\left\{Up(t) - Vh(p(t))|U\right\}$$

$$- \mathbb{E}\left\{Vg(f(t)) - U_{d_b}F(f(t))|U\right\}$$
(40)

It can be seen from (40) that the last two terms represent the decoder and encoder optimization policies respectively. The decoder policy is not affected by the network delay while the only change to the encoder policy is to make use of  $U(t - d_b)$  fed back from the decoder. Moreover, since the delay accumulator X(t) is updated locally within the decoder. This would imply that the delay constrained decoder policy (28) would not be affected. Lastly, the performance bound for the Lyapunov optimization can be derived (see Section F of the supplementary materials).

#### VII. VIDEO QUALITY FUNCTIONS

In the previous sections, we showed how using the discontinuity penalty virtual buffer in the system model allows us to express the processes as frame intervals. We then used Lyapunov optimization to derive optimization policies that stabilizes the discontinuity penalty virtual buffer and showed that this helps to maintain the video continuity. We also demonstrated how to add a delay constraint into a framework by using the delay accumulator virtual buffer. By deriving optimization policies that stabilize the delay accumulator virtual buffer, we showed that the delay constraint can be met. Finally, we studied the impact of network delay on the optimization policies and showed how the optimization policies can be derived with network delay consideration.

What is lacking thus far is a discussion on the specific choice of frame quality function g(f(t)) and playout distortion function h(p(t)). In this section, we shall examine the specific forms for g(f(t)) and h(p(t)).

#### A. Frame Quality Function

We first look at an appropriate frame quality function for g(f(t)), g(f(t)) should ideally be concave so that a solution can be easily found for the encoder policy (20). As mentioned before as f(t) decreases, the encoder frame generation rate increases. This means that more compression is needed to meet the available network bandwidth and more compression tends to mean that the frame quality will be reduced. One of the ways to measure frame quality is to measure the peak signal-to-noise-ratio (PSNR) of the video.

An and Nguyen [37] has been shown that PSNR can be represented using a log function of the bitrate. We make use of their result and fit PSNR to the average frame size of the sequence:

$$PSNR(f(t)) = a \log(ABR(t) f(t)) + c$$
(41)

where a and c are the modeling coefficients, and ABR(t) is the current available network bandwidth. Note that we make use of all the available bandwidth to transmit the video frames from the sender. Fig. 4 shows the fitted curve. The video sequence used for fitting is a concatenation of football, city, crew and akiyo sequences in that order. This is done to ensure that the sequence contains several subsequences of different coding complexity. Notice that if we use the encoder frame generation rate i(t) as the input variable instead, (41) becomes:

$$PSNR(i(t)) = a \log\left(\frac{ABR(t)}{i(t)}\right) + c \tag{42}$$

where  $i(t) = \frac{1}{f(t)}$ , the resulting PSNR function would no longer be a concave function of i(t) and would make the optimization problem more difficult to solve. This is the reason why we use the encoder frame generation interval f(t) instead of encoder frame generation rate i(t). Since f(t) is upper bounded by  $f_{max}$ , we fix the maximum of PSNR(f(t)) to  $f_{max}$ . This is done to make the PSNR function differentiable in the interval  $[f_{min}, f_{max}]$  and make the maximization of it easy to calculate. We do this by subtracting  $\frac{a}{f_{max}}$  from the first derivative of (41) (PSNR'(f(t))):

$$PSNR'(f(t)) - \frac{a}{f_{max}} = \frac{a}{f(t)} - \frac{a}{f_{max}}$$
(43)

Integrating (43) will give us the desired g(f(t)):

$$g(f(t)) = a \log(ABR(t) f(t)) + c - \frac{a f(t)}{f_{max}}$$

$$\tag{44}$$

Note that g(f(t)) is concave between the range  $[f_{min}, f_{max}]$  (where  $f_{min} > 0$ ).



Figure 4. PSNR versus average frame size.



Figure 5. Playout distortion function. p(t) is in milliseconds, m = 1 and  $p_n = \frac{1}{30}$  ms.

#### **B.** Playout Distortion Function

In this section, we choose an appropriate playout distortion function for h(p(t)). We modified the version of playout distortion function used in [29]:

$$h(p(t)) = m \cdot (p_n - p(t))^2$$
 (45)

where m is the motion intensity of the sequence, calculated using the technique in [38], and  $p_n$  is the natural playout interval. Fig. 5 shows the playout distortion function h(p(t)). h(p(t)) is convex in the range  $[p_{min}, p_{max}]$ . Eqn. (45) is a combination of the quadratic playout distortion function proposed in [25], [39] and the motion scaling used in [29]. The idea is that the playout distortion increases as the playout rate deviates from the natural playout rate. The playout distortion is also affected by the motion intensity of the sequence. Intuitively, higher motion sequences increases playout distortion more as the change in motion is more perceivable when the playout rate deviates from the natural playout rate.

#### VIII. PERFORMANCE EVALUATION

#### A. Experiment Setup

We made use of ns-2 [40] to simulate a network with time-varying data bandwidths. We implemented our framework into a x264 encoder [33], an open source multi-threaded H.264/AVC encoder. Our implementation in x264 simulates the network transmission using network traces. The decoder buffer evolution is simulated by tracking the arrivals of video frames from the simulated network and the removal of video frames from the buffer for playout. Everytime a frame is encoded, the framework will make a decision on the encoder frame generation rate and the playout frame rate at the simulated decoder. This is to simulate the real-time adjustment of parameters as the video is being encoded for transmission. Network traces obtained from ns-2 are used to simulate the network within the encoder. The ns-2 traces provide the sending and receiving times as well as the loss status of each video packet. Every packet produced by x264 is assigned a sending time, receiving time and a loss status. These infomation is used to simulate the sender and receiver buffers at a packet level. The video packets that arrived at

the receiver buffer are then used to calculate the amount of frames that the decoder buffer contains. A decoder buffer underflow occurs when the frames removed from the decoder buffer exceeds the number of frames in it. Note that our framework could easily be adapted to multiple pre-encoded copies of the video.

For the network simulations, we made use of a dumbell network topology with the bottleneck bandwidth set to 5 Mbits/s. One pair of nodes in the network simulated video streaming using the TFRC transport protocol [41]. To generate background traffic, random webpage sessions were used for the other pairs of nodes. All the random sessions go through the bottleneck link. The average packet delay from the encoder to the decoder is 235 ms, while the feedback delay from the decoder to encoder is 330 ms. These high delay values enable us to show that the our proposed distributed frame rate control algorithm works in presence of delay.

The encoder receives feedback from the decoder and solves the optimization problem to determine the encoder frame generation interval f(t). The x264 encoder makes use of this encoder frame generation interval f(t) by setting the target frame rate of the rate controller to encoder frame generation rate  $\frac{1}{f(t)}$ . The rate controller's target frame rate is used to compute the quantization level for each frame. The lower the target frame rate, the lower the quantization level and, as a result, bigger frame sizes. More details on x264's rate control can be found in [12].

The test sequence used is a concatenation of football, city, crew and akiyo sequences in that order. This is to ensure a mix of high and low motion within the sequence. A 16 minute test sequence is obtained by repeating the concatenated sequence.

The model coefficient a from (44) is found by curve fitting to be 4.91. The constant V is set to 1. In our experiments, we tested the continuity of the video, this is defined here as the amount of time spent playing video, specifically:

Playout Continuity = 
$$1 - \frac{\text{rebuffering time}}{\text{total sequence time}}$$
 (46)

where the rebuffering time is the time needed to refill the buffer to a certain threshold after an underflow event, this is a typical behaviour of a video player [42]. The rebuffering threshold is set to half of the prebuffer amount. The prebuffer amount is varied for each run to determine the performance of each scheme. At the end of each run, we will calculate the playout continuity using (46) for each scheme and make a comparison.

## B. Discontinuity Penalty Lyapunov Optimization Results



Figure 6. Correlation between average discontinuity penalty and continuity.



Figure 8. Prebuffering delay vs playout distortion.



Figure 7. Prebuffering delay vs continuity.



Figure 9. PSNR loss for LOpt.

We next present the results of the Lyapunov optimization framework described in Section IV, labelled as *LOpt* here. We also set up our framework with a less conservative bound than the one described in (5), this is labelled as *LOptM*. This is done by setting  $\beta(t) = \frac{b(t) r_{max} p_{max}}{T-t}$  in (14). We compare our framework with a typical setup of x264 with its target frame rate and its playout rate set to a constant 30 fps, we label this scheme *Norm*. We also compared our framework with the AMP scheme [3]. We implemented a combination of AMP-Initial and AMP-Robust. AMP-Initial slows down the playout rate by a predetermined slowdown factor when the video starts playing until a certain buffer level has been reached. AMP-Robust slows down the playout rate of the video by a predetermined slowdown factor when the buffer level falls below a certain level. In our implementation, AMP-Initial is used in conjunction with AMP-Robust. We empirically chose the smallest slowdown factor of AMP such that it achieves a continuity of 99% or higher in each test case. This effectively simulates a close to optimal adaptive AMP scheme, we label this as *AMP*. We also added in the results of *AMP* with a constant slowdown factor of 25% (used in [3]) as a reference, which we label *AMP25*.

discontinuity penalty U(t) can be used as an indicator of a lack of video continuity.

To examine the performance of each scheme, we compare the continuity of each scheme based on the amount of prebuffering provided. Note that in the results, the prebuffering delay is on a log base 10 scale. Continuity is calculated as in (46).

The continuity results are shown in fig. 7. It can be seen that *LOpt* and *AMP* achieves similar results. This is expected as *AMP* was tuned to achieve a high continuity. The performance disparity between *AMP* and *AMP25* shows that this simulation requires a greater amount of playout slowdown at the decoder in order to reduce the occurrences of buffer underflows. However, both *LOpt* and *AMP* require about a 100 times less prebuffering compared to *Norm* to provide similar continuity. *AMP25* too requires about 7 times less prebuffering than *Norm* but still requires about 50 times more prebuffering than *LOpt* and *AMP*. This suggests that using some form of playout slowdown would reduce the prebuffering requirements of a video application significantly.

We next measure the playout distortion of each scheme using (45) ( $p_n = 1/30$ ), i.e. playout distortion will be produced when the playout interval drops below or goes above 1/30. Note that while (45) will only produce a non-zero value when the playout deviates from the natural playout frame interval, playout interruptions due to buffer underflows are not factored into the playout distortion. Fig. 8 shows the playout distortion results. *Norm* does not have any playout distortion since it has a constant 30 fps playout rate, but suffers from playout discontinuity as discussed earlier. *LOpt* has a very similiar playout distortion characteristic when compared to *AMP25*. In contrast, *AMP* for most cases produces twice the amount of playout distortions compared to *LOpt* and *AMP25*. This is mainly because *AMP* requires a higher slowdown factor to obtain a better video continuity and, as a consequence, this results in a higher playout distortion.

*LOpt*'s comparatively low playout distortion is due to the joint adjustment of both the encoder frame generation rate and playout rate. By increasing the encoder frame generation rate, the receiving frame at the decoder increases. This provides a higher buffer occupancy and reduces the need to slowdown the playout rate, thus reducing the playout distortion. This comes at the expense of frame quality, because increasing encoder frame generation rate will result in a higher amount of compression. To examine *LOpt*'s effect on frame quality, we compare the PSNR of the encoded video before transmission. This is done to eliminate any possible drops in PSNR due to transmission. *Norm, AMP25* and *AMP* have a constant encoding rate of 30 fps, this means all produce an encoded video of the same PSNR. Thus, we only compared *LOpt* with *Norm*. From fig. 9, it is shown that the drop in PSNR is about 0.6 dB for *LOpt*. This is a reasonably small tradeoff in frame quality given the improvements in playout distortion and continuity.

It can be also seen from the graphs that the performance of *LOpt* and *LOptM* are very similiar. This suggests that the bound in (5) is not too conservative. We also tested the schemes on football, see figs. 10, 12, 14 and akiyo, see figs. 11, 13, 15. football and akiyo were chosen because they have the highest and lowest motion content respectively in the four sequences used. The results show a similiar pattern to the concatenated sequence.

We now examine the complexity of each scheme. *Norm* is the least complex scheme while the *AMP25* is marginally more complex than *Norm*. This is because *AMP25* involves some simple logic at the decoder to slowdown the playout once a certain buffer level is reached. *LOpt* is more complex than *AMP25* as it involves more calculations

at both the encoder and decoder end. However, it is not significantly more complex than AMP25 because the optimization policies are concave, so the solution search is very efficient. For example, in our implementation we solve for f(t) and p(t) by using the first derivatives of the encoder policy (20) and decoder policy (21) respectively. AMP is the most complex in our implementation as it requires several runs to determine the most optimal slowdown factor. In summary, the LOpt runs in O(n), where n is the total number of video packets. While AMP runs in  $O(n \times s)$ , where s is the number of slowdown factors to consider. However, it should be noted that complexity-wise, AMP is not representative of the complexity of AMP schemes in practice. Its main purpose in our simulations is to act as the upper bound in the performance of AMP schemes.



Figure 10. Prebuffering delay vs continuity (football).



Figure 12. Prebuffering delay vs playout distortion (football).



Figure 14. PSNR loss for *LOpt* (football).



Figure 11. Prebuffering delay vs continuity (akiyo).



Figure 13. Prebuffering delay vs playout distortion (akiyo).



Figure 15. PSNR loss for LOpt (akiyo).

## C. Delay Constrained Lyapunov Optimization Results

To evaluate the performance of our delay constrained Lyapunov optimization framework, which we call *DLOpt* here. As in the previous section, we also set up out framework with a less conservative bound  $\frac{b(t) r_{max} p_{max}}{T-t}$ , labelled as *DLOptM*. We compare our scheme with AMP-Live [3], labelled as *AMPL*. AMP-Live maintains the buffer level by slowing down or speeding up the playout based on a predetermined scale factor. The scale factor of *AMPL* is set to 40% in our experiments, this value was found to provide the best overall performance for *AMPL*. The playout delay of each scheme is measured as: Total playout delay =  $\sum_{\tau=0}^{T} (p(\tau) - p_n)$ .

Recall that  $p_n$  represents the natural playout frame interval of the sequence. So every playout slowdown will cause p(t) to be larger than  $p_n$ , thus accumulating playout delay. To reduce the total playout delay, the scheme needs to find the proper moment to increase the playout rate.

We compared the schemes by varying the delay constraints from 4.27 seconds to 273.07 seconds with the constraint doubled at each run. At the end of each run, we plot the total playout delay and the performance metric of each scheme, the playout delay in the results are on a log base 10 scale. The performance metric we examined are playout continuity, playout distortion and PSNR.



10

Total Playout Delay (s)

100

1000



Figure 17. Playout delay vs playout distortion.

Figure 18. PSNR loss for DLOpt

36.6 L

We first look at the continuity results with respect to the total playout delay (fig. 16). It can be seen that *DLOpt* achieves maximum continuity regardless of the delay constraint, while *AMPL* requires more than 50 times the amount of total playout delay compared to *DLOpt* to reach maximum continuity. *AMPL* manages the total playout delay constraint by maintaining a certain buffer level [3], thus a tighter delay constraint will result in a lower buffer level which has higher probabilities of buffer underflows. On the other hand, *DLOpt* makes a trade-off between the encoder frame generation rate and playout rate to satisfy the continuity goal and delay constraint.

We next look at the distortion results in fig. 17. Again, *DLOpt* achieves a much lower playout distortion compared to *AMPL*. Note that the lower the total playout delay constraints the lower the playout distortion. This is because a low playout constraint causes *DLOpt* to adjust the playout rate a lot less, thus causing a low playout distortion. *AMPL* has an almost constant playout distortion. This is because, as mentioned before, *AMPL* only tries to maintain a certain buffer level and does not constrain its playout rate adjustment in anyway.

Lastly, we examine the PSNR trade-offs made, see fig. 18. As before, we compared the PSNR of the encoded video of both schemes prior to transmission. It can be seen that *DLOpt* sacrifices about a maximum of 1 dB of PSNR, this PSNR drop is higher than *LOpt* in the previous section. The main cause of this is the additional delay constraint imposed using the virtual buffer (24). This results a lower PSNR due to the need to satisfy the delay constraint, however, this allows the large gains in continuity and distortion.

Again the performances of *DLOpt* and *DLOptM* are mostly similiar. This reinforces the possibility that the bound in (5) is not too conservative..

## IX. CONCLUSIONS

In this paper, we proposed a the frame rate optimization framework. What we achieved with this framework is:

- Performed frame rate control by a joint adjustment of the encoder frame generation interval and the playout frame interval.
- Modelled the frame rate control problem using the encoder frame generation interval, the playout frame interval and the discontinuity penalty virtual buffer. The model is created in such a way that stabilizing the discontinuity penalty virtual buffer allows the video to maintain continuity. We then used Lyapunov optimization on this model to systematically derive the optimization policies. We showed that these policies can be decoupled into separate encoder and decoder optimization policies with feedback between the two. We also showed through experiments that the proposed discontinuity penalty based on the virtual buffer is correlated to the video continuity. Finally, simulation results demonstrates the effectiveness of the discontinuity penalty virtual buffer approach.

- A delay constraint imposed on the accumulated delay from playout slowdowns. We introduced the delay constraint into the framework using the delay accumulator virtual buffer. We showed, using Lyapunov optimization analysis, that by stabilizing the delay accumulator virtual buffer, the delay constraint would be satisfied. Simulation results showed a superior playout continuity and playout distortion performance with a reasonable tradeoff in PSNR.
- An analysis of the impact of delayed feedback from receiver to sender. We derived two different analyses, the first analysis showed very little impact on the optimization polices. The alternate analysis showed that the decoder needed to use an outdated buffer state. Simulation results demonstrated that using the first analysis results in a better performance.

#### REFERENCES

- M. Claypool and J. Tanner, "The effects of jitter on the peceptual quality of video," in *Proceedings of the seventh ACM international conference on Multimedia (Part 2)*. ACM, 1999, pp. 115–118.
- [2] N. Laoutaris and I. Stavrakakis, "Adaptive playout strategies for packet video receivers with finite buffer capacity," in *Communications*, 2001. ICC 2001. IEEE International Conference on, vol. 3. IEEE, 2001, pp. 969–973.
- [3] M. Kalman, E. Steinbach, and B. Girod, "Adaptive media playout for low-delay video streaming over error-prone channels," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 841–851, 2004.
- [4] Y. Su, Y. Yang, M. Lu, and H. Chen, "Smooth control of adaptive media playout for video streaming," *Multimedia, IEEE Transactions* on, vol. 11, no. 7, pp. 1331–1339, 2009.
- [5] E. Steinbach, N. Farber, and B. Girod, "Adaptive playout for low latency video streaming," in *Image Processing*, 2001. Proceedings. 2001 International Conference on, vol. 1. IEEE, 2002, pp. 962–965.
- [6] L. Georgiadis, M. Neely, and L. Tassiulas, Resource allocation and cross layer control in wireless networks. Now Pub, 2006.
- [7] T. Stockhammer, H. Jenkac, and G. Kuhn, "Streaming video over variable bit-rate wireless channels," *Multimedia, IEEE Transactions* on, vol. 6, no. 2, pp. 268–277, 2004.
- [8] B. Steyaert, K. Laevens, D. De Vleeschauwer, and H. Bruneel, "Analysis and design of a playout buffer for VBR streaming video," *Annals of Operations Research*, vol. 162, no. 1, pp. 159–169, 2008.
- [9] G. Liang and B. Liang, "Effect of delay and buffering on jitter-free streaming over random VBR channels," *Multimedia, IEEE Transactions on*, vol. 10, no. 6, pp. 1128–1141, 2008.
- [10] S. Gulliver and G. Ghinea, "The perceptual and attentive impact of delay and jitter in multimedia delivery," *IEEE Transactions on Broadcasting*, vol. 53, no. 2, pp. 449–458, 2007.
- [11] Z. Li, F. Pan, K. Lim, G. Feng, X. Lin, and S. Rahardja, "Adaptive basic unit layer rate control for JVT," JVT-G012-r1, 7th Meeting, Pattaya II, Thailand, Mar, vol. 14, 2003.
- [12] L. Merritt and R. Vanam, "Improved rate control and motion estimation for H. 264 encoder," in *Proceedings of ICIP*, vol. 5, 2007, pp. 309–312.
- [13] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H. 264/AVC standard," IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, no. 9, pp. 1103–1120, 2007.
- [14] S. Moon, J. Kurose, and D. Towsley, "Packet audio playout delay adjustment: performance bounds and algorithms," *Multimedia Systems*, vol. 6, no. 1, pp. 17–28, 1998.
- [15] M. Roccetti, V. Ghini, G. Pau, P. Salomoni, and M. Bonfigli, "Design and experimental evaluation of an adaptive playout delay control mechanism for packetized audio for use over the internet," *Multimedia Tools and Applications*, vol. 14, no. 1, pp. 23–53, 2001.
- [16] A. Markopoulou, F. Tobagi, and M. Karam, "Assessing the quality of voice communications over internet backbones," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 5, pp. 747–760, 2003.
- [17] Y. Liang, N. Farber, and B. Girod, "Adaptive playout scheduling and loss concealment for voice communication over IP networks," *Multimedia, IEEE Transactions on*, vol. 5, no. 4, pp. 532–543, 2003.
- [18] K. Fujimoto, S. Ata, and M. Murata, "Adaptive playout buffer algorithm for enhancing perceived quality of streaming applications," *Telecommunication Systems*, vol. 25, no. 3, pp. 259–271, 2004.
- [19] M. Narbutt, A. Kelly, P. Perry, and L. Murphy, "Adaptive VoIP playout scheduling: assessing user satisfaction," *Internet Computing*, *IEEE*, vol. 9, no. 4, pp. 28–34, 2005.
- [20] W. Verhelst and M. Roelands, "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech," in *ICASSP-93.*, vol. 2, 1993.
- [21] I. Damnjanovic, D. Barry, D. Dorran, and J. Reiss, "A Real-Time Framework for Video Time and Pitch Scale Modification," *Multimedia*, *IEEE Transactions on*, vol. 12, no. 4, pp. 247–256, 2010.
- [22] M. Hassan and M. Krunz, "A playback-adaptive approach for video streaming over wireless networks," in *Global Telecommunications Conference*, 2005. GLOBECOM'05. IEEE, vol. 6. IEEE, 2005, pp. 5–3691.
- [23] S. Deshpande, "Underflow prevention for AV streaming media under varying channel conditions," in *Proceedings of SPIE*, vol. 6507, 2007, p. 65070C.
- [24] J. Seo, V. Leung, and H. Lee, "Optimizing a Playout Buffer with Queueing Performance Metrics for One-Way Streaming Video," in Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE. IEEE, 2008, pp. 1–6.
- [25] M. Kalman, E. Steinbach, and B. Girod, "Rate-distortion optimized video streaming with adaptive playout," in *IEEE ICIP*, vol. 3, 2002, pp. 189–192.
- [26] D. Tao, H. Hoang, and J. Cai, "Optimal frame selection with adaptive playout for delivering stored video under constrained resources," in *Multimedia and Expo*, 2007 IEEE International Conference on. IEEE, 2007, pp. 1798–1801.

- [27] A. Dua and N. Bambos, "Buffer management for wireless media streaming," in *Global Telecommunications Conference*, 2007. GLOBECOM'07. IEEE. IEEE, 2007, pp. 5226–5230.
- [28] S. Deshpande, "High quality video streaming using content-aware adaptive frame scheduling with explicit deadline adjustment," in Proceeding of the 16th ACM international conference on Multimedia. ACM, 2008, pp. 777–780.
- [29] Y. Li, A. Markopoulou, J. Apostolopoulos, and N. Bambos, "Content-Aware Playout and Packet Scheduling for Video Streaming Over Wireless Links," *IEEE Transactions on Multimedia*, vol. 10, no. 5, pp. 885–895, 2008.
- [30] M. Yuang, P. Tien, and S. Liang, "Intelligent video smoother for multimedia communications," *Selected Areas in Communications*, *IEEE Journal on*, vol. 15, no. 2, pp. 136–146, 2002.
- [31] H. Chuang, C. Huang, and T. Chiang, "Content-Aware Adaptive Media Playout Controls for Wireless Video Streaming," IEEE Transactions on Multimedia, vol. 9, no. 6, pp. 1273–1283, 2007.
- [32] E. Steinbach, N. Farber, and B. Girod, "Adaptive playout for low latency video streaming," in *Image Processing*, 2001. Proceedings. 2001 International Conference on, vol. 1. IEEE, 2001, pp. 962–965.
- [33] L. Merritt and R. Vanam, "x264: A high performance H.264/AVC encoder," [online] http://neuron2.net/library/avc/overview\_x264\_v8\_5. pdf.
- [34] H. Lee, T. Chiang, and Y. Zhang, "Scalable rate control for MPEG-4 video," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 10, no. 6, pp. 878–894, 2002.
- [35] M. Neely, "Energy optimal control for time-varying wireless networks," *Information Theory, IEEE Transactions on*, vol. 52, no. 7, pp. 2915–2934, 2006.
- [36] —, "Dynamic power allocation and routing for satellite and wireless networks with time varying channels," 2004.
- [37] C. An and T. Nguyen, "Analysis of utility functions for video," in *Image Processing*, 2007. ICIP 2007. IEEE International Conference on, vol. 5, 16 2007-Oct. 19 2007, pp. 89–92.
- [38] Y. Ou, T. Liu, Z. Zhao, Z. Ma, and Y. Wang, "Modeling the impact of frame rate on perceptual quality of video," *City*, vol. 70, pp. 80–90.
- [39] Y. Li, A. Markopoulou, N. Bambos, and J. Apostolopoulos, "Joint power-playout control for media streaming over wireless links," *Multimedia, IEEE Transactions on*, vol. 8, no. 4, pp. 830–843, 2006.
- [40] "The Network Simulator NS-2," http://www.isi.edu/nsnam/ns/.
- [41] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP friendly rate control (TFRC): Protocol specification," 2003.
- [42] G. Conklin, G. Greenbaum, K. Lillevold, A. Lippman, and Y. Reznik, "Video coding for streaming media delivery on the internet," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 3, pp. 269 –281, Mar. 2001.

#### APPENDIX

# A. Derivation of Discontinuity Penalty Lyapunov Drift Bound

We square the buffer dynamics equation (14), divide it by two and rearrange its terms to get:

$$U(t+1) \leq [U(t) - p(t) - \beta(t)]^{+} + F(f(t))$$
  

$$U(t+1)^{2} \leq U(t)^{2} + (p(t) + \beta(t))^{2} + F(f(t))^{2}$$
  

$$- 2U(t) [p(t) + \beta(t) - F(f(t))]$$
(47)

Dividing the above equation by two and rearranging, we get:

$$\frac{U(t+1)^2}{2} - \frac{U(t)^2}{2} \le \frac{(p(t)+\beta(t))^2 + F(f(t))^2}{2} - U(t)\left[p(t)+\beta(t) - F(f(t))\right]$$
(48)

Using the definition of the Lyapunov function (15) in the above equation yields:

$$L(U(t+1) - L(U(t)) \le \frac{(p(t) + \beta(t))^2 + F(f(t))^2}{2} - U(t) [p(t) + \beta(t) - F(f(t))]$$
(49)

Recall that  $\beta(t)$  is defined as:

$$\beta(t) = \frac{b(t) r_{min} p_{min}}{T - t}$$
(50)

where b(t) is the amount of buffered video frames in the buffer at time t and T is the length of the whole sequence in time slots. Since a time slot corresponds to a frame in our framework, it can be seen that  $b(t) \leq T$ . This is because the maximum amount of video content that can exist in the buffer is the whole sequence. This then implies that:

$$\beta(t) \le \frac{T r_{min} p_{min}}{T - t} \tag{51}$$

Using the above bound in (49), we obtain:

$$L(U(t+1) - L(U(t))) \le \frac{1}{2} \left[ \left( p(t) + \frac{T r_{min} p_{min}}{T - t} \right)^2 + F(f(t))^2 \right] - U(t) \left[ p(t) + \beta(t) - F(f(t)) \right]$$
(52)

Using r(t) = F(f(t)) and  $r(t) \le r_{max}$ , we get  $F(f(t)) \le r_{max}$ . Since  $p(t) \le p_{max}$ , using these two bounds into the above equation yields:

$$L(U(t+1) - L(U(t)) \le \frac{1}{2} \left[ \left( p_{max} + \frac{T r_{min} p_{min}}{T - t} \right)^2 + r_{max}^2 \right] - U(t) \left[ p(t) + \beta(t) - F(f(t)) \right] = B - U(t) \left[ p(t) + \beta(t) - F(f(t)) \right]$$
(53)

where:

$$B = \frac{1}{2} \left( r_{max}^2 + \left( p_{max} + \frac{T r_{min} p_{min}}{T - t} \right)^2 \right)$$
(54)

If we take conditional expectations of (53) with respect to U(t) and use the definition of the one-step conditional Lyapunov drift (16), we get:

$$\Delta(U(t)) \le B - U(t)\mathbb{E}\{\beta(t) + p(t) - F(f(t))|U(t)\}$$
(55)

# B. Derivation of Delay Constrained Lyapunov Drift Bound

We first define a Lyapunov function L(X(t)) based on X(t) as:

$$L(X(t)) \triangleq \frac{X^2(t)}{2} \tag{56}$$

Recall for the discontinuity penalty U(t), its Lyapunov function is:

$$L(U(t)) \triangleq \frac{U^2(t)}{2} \tag{57}$$

Then, it can be seen that (25) can be defined as:

$$L(U(t), X(t)) = \frac{U^{2}(t) + X^{2}(t)}{2}$$
  
=  $\frac{U^{2}(t)}{2} + \frac{X^{2}(t)}{2}$   
=  $L(U(t)) + L(X(t))$  (58)

Now, we square the buffer dynamics equation (24) to obtain:

$$X(t+1) \le [X(t) - p_n - t_d]^+ + p(t)$$
  

$$X(t+1)^2 \le X(t)^2 + p(t)^2 + (p_n + t_d)^2 - 2X(t) [p_n + t_d - p(t)]$$
(59)

Dividing the above equation by two and rearranging yields:

$$\frac{X(t+1)^2}{2} - \frac{X(t)^2}{2} \le \frac{p(t)^2 + (p_n + t_d)^2}{2} - X(t) \left[p_n + t_d - p(t)\right]$$
(60)

Using  $p(t) \leq p_{max}$  and (56) in the above equation yields:

$$L(X(t+1)) - L(X(t)) \le \frac{1}{2} \left( p(t)^2 + (p_n + t_d)^2 \right) - X(t) \left[ p_n + t_d - p(t) \right]$$
  
=  $C - X(t) \left[ p_n + t_d - p(t) \right]$  (61)

where:

$$C = \frac{1}{2} \left( p_{max}^2 + (p_n + t_d)^2 \right)$$
(62)

By adding (61) and (53) together, we get:

$$L(X(t+1)) - L(X(t)) + L(U(t+1) - L(U(t))) \le C - X(t) [p_n + t_d - p(t)] + B - U(t) [p(t) + \beta(t) - F(f(t))]$$
(63)

Using (58) and rearranging the above yields:

$$L(U(t+1), X(t+1)) - L(U(t), X(t)) \le B + C$$
  
- U(t) [p(t) + \beta(t) - F(f(t))]  
- X(t) [p\_n + t\_d - p(t)] (64)

Taking conditional expectations of the above with respects to U(t) and X(t), and using (26) yields:

$$\Delta(U(t), X(t)) \le B + C - U(t) \mathbb{E}\{p(t) - F(f(t)) | U(t), X(t)\} - X(t) \mathbb{E}\{p_n + t_d - p(t) | U(t), X(t)\}$$
(65)

## C. Derivation of Network Delayed Lyapunov Drift Bound

Note that, in this section, we use the same shortened notations defined in Section VI, additionally we also define  $U_{d_f} = U(t + d_f + 1)$ . This means that (32) can be rewritten as:

$$U_{d_f} \le \left[U - \gamma_{d_f}\right]^+ + F_{d_f} \tag{66}$$

By squaring the above equation and dividing it by two, we obtain:

$$\frac{U_{d_f}^2}{2} \le \frac{U^2}{2} + \frac{\gamma_{d_f}^2 + F_{d_f}^2}{2} - U\left[\gamma_{d_f} - F_{d_f}\right]$$
(67)

Using the definition of the Lyapunov function (15) on the above equation and rearranging:

$$L(U_{d_f}) - L(U) \le \frac{\gamma_{d_f}^2 + F_{d_f}^2}{2} - U[\gamma_{d_f} - F_{d_f}]$$
(68)

Using  $\gamma(t) = p(t) + \beta(t)$ ,  $F(f(t)) \leq r_{max}^3$ ,  $p(t) \leq p_{max}$  and (51) on the above equation yields:

$$L\left(U_{d_f}\right) - L(U) \leq \frac{d_f}{2} \left( r_{max}^2 + \left( p_{max} + \frac{T r_{min} p_{min}}{T - t} \right)^2 \right)$$
$$= B' - U \left[ \gamma_{d_f} - F_{d_f} \right]$$
(69)

where:

$$B' = \frac{d_f}{2} \left( r_{max}^2 + \left( p_{max} + \frac{T r_{min} p_{min}}{T - t} \right)^2 \right)$$
(70)

Taking conditional expectations of (69) with respects to U and using (33) yields:

$$\Delta(U) \le B' - \mathbb{E}\left\{U\gamma_{d_f} \middle| U\right\} + \mathbb{E}\left\{UF_{d_f} \middle| U\right\}$$
(71)

## D. Discontinuity Penalty Optimization Stability and Performance Bounds

In this section, we prove that the derived encoder and decoder policies, (20) and (21) respectively, stabilize the discontinuity penalty virtual buffer. This is given by equation (72) in Theorem 1. We also show the performance bound of the Lyapunov optimization.

**Theorem 1.** Let the long term receiving frame interval  $\bar{r} = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{r(t)\}$ . If the network delay variation r(t) is i.i.d<sup>4</sup> over the timeslots, the receiving frame interval is lower bounded by the encoder frame generation interval as  $r(t) \ge f(t)$ , the frame quality function is bounded as  $g(f(t)) \le G_{max}$  and the playout distortion function is bounded as  $h(p(t)) \ge H_{min}$ . Then implementing the optimization policies (20) and (21) in each timeslot stabilizes the discontinuity penalty U(t) (using the stability definition  $\mathbb{E}\{U\} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{U(\tau)\} < \infty$ ) and satisfies the following performance bounds:

$$\limsup_{M \to \infty} \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{U(\tau)\} \le \frac{B + V(G_{max} - H_{min})}{r_{max}}$$
(72)

$$\liminf_{M \to \infty} g(\bar{f}) \ge (g^* - h^*) + H_{min} - \frac{B}{V}$$
(73)

$$\limsup_{M \to \infty} h(\bar{p}) \le \frac{B}{V} - (g^* - h^*) - G_{max}$$
(74)

<sup>3</sup>See Section A.

 $<sup>^{4}</sup>$ With i.i.d processes, the steady state is exactly achieve every timeslot. This allows us to use Lyapunov drift analysis on a per time slot basis. However, Neely has shown that i.i.d processes provides all of the intuition needed to treat general processes. For more details on this, see chapter 4 of [36].

where V is some positive constant (i.e. V > 0),  $g^*$  and  $h^*$  are the specific values of g(.) and h(.) respectively that maximizes the objective (8) subjected to the constraints (9), (10) and (11), B is defined as in (18) and:

$$\overline{f} = \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{f(\tau)\}$$
(75)

$$\overline{p} = \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{p(\tau)\}$$
(76)

*Proof:* We first introduce  $\Lambda$  as a set of receiving frame intervals that stabilizes the discontinuity penalty U(t).  $\Lambda$  is bounded by  $r_{max}$ . Using  $\Lambda$  assumes a complete knowledge of future, but this is only required for the analysis on the performance bounds. With  $\Lambda$ , the optimization problem in Section IV-D can be restated as:

Maximize: g(f(t)) - h(p(t)) (77)

Subject to: 
$$F(f(t)) = r(t)$$
 (78)

$$\bar{r} = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\left\{r(t)\right\}$$
(79)

$$\bar{r} \in \Lambda$$
 (80)

$$f_{min} \le f(t) \le f_{max} \tag{81}$$

$$p_{min} \le p(t) \le p_{max} \tag{82}$$

From the above optimization problem,  $\Lambda$  can be intuitively seen as a range of values that r(t) can take that allows the existence of at least one stationary randomized policy that can stabilize U(t). Constraint (78) is due to r(t) = F(f(t)). Let  $f^*$  and  $p^*$  form the solution that maximizes the objective (77). This means that an optimal stable policy would ensure that the following is met:

$$F(f^*) = \bar{r} \le p^* \tag{83}$$

 $F(f^*) = \bar{r}$  comes from constraint (78), while the inequality  $\bar{r} \le p^*$  is the result of lemma 7 in [35]. Suppose now an  $\epsilon$ -optimal stable policy causes the long term receiving frame interval  $\bar{r}$  to become  $\bar{r}_{\epsilon}$  such that:

$$F(f_{\epsilon}^{*}) = \bar{r}_{\epsilon} \leq p^{*} - \epsilon$$
$$\bar{r}_{\epsilon} \in \Lambda_{\epsilon}$$
$$\bar{r}_{\epsilon} + \epsilon \in \Lambda$$
(84)

where  $\epsilon > 0$  is a positive constant and the receiving frame intervals set  $\Lambda_{\epsilon} \subset \Lambda$ .  $\Lambda_{\epsilon}$  can be seen as  $\Lambda$  that is reduced by  $\epsilon$ . Then, the  $\epsilon$ -optimal policy can be seen as optimizing the following problem:

Maximize: 
$$g(f(t)) - h(p(t))$$
  
Subject to:  $F(f(t)) = r(t)$   
 $r(t) \in \Lambda_{\epsilon}$   
 $f_{min} \leq f(t) \leq f_{max}$   
 $p_{min} \leq p(t) \leq p_{max}$ 
(85)

Note that the long term encoder frame generation interval  $f_{\epsilon}^*$  forms part of the solution that maximizes the above problem. Equations(83) and (84) together implies that:

$$\bar{r}_{\epsilon} \le \bar{r}$$
 (86)

This is because the  $\epsilon$ -optimal policy maximizes the same objective as (77) but with its long term receiving frame interval  $\bar{r}_{\epsilon}$  limited by  $\epsilon$ . Using r(t) = F(f(t)) would mean the above equation becomes:

$$F(f^*_{\epsilon}) \le F(f^*) \tag{87}$$

Using (6) and assuming there exist the long term frame interval scaling factors  $e_{\epsilon}$  and  $e^*$ , such that  $F(f_{\epsilon}^*) = e_{\epsilon}f_{\epsilon}^* = \bar{r}_{\epsilon}$  and  $F(f^*) = e^*f^* = \bar{r}$ . Then, applying these properties to the above equation yields:

$$e_{\epsilon}f_{\epsilon}^* \le e^*f^* \tag{88}$$

Since we made the assumption that  $r(t) \ge f(t)$ , which can be rewritten as  $e(t)f(t) \ge f(t)$ . This implies that  $e_{\epsilon} \ge 1$  and  $e^* \ge 1$ , which means that :

$$f_{\epsilon}^* \le f^* \tag{89}$$

This is because  $f_{\epsilon}^*$  is chosen by the  $\epsilon$ -optimal policy to maximize the objective in (85). Since the frame quality function g(f(t)) in the objective is an increasing function of f(t), this implies that the  $\epsilon$ -optimal policy will choose  $f_{\epsilon}^*$  as large as possible. However, as  $f_{\epsilon}^*$  is upper bounded by  $\bar{r}_{\epsilon}$ , which is in turn limited by  $\epsilon$ , this means that  $f_{\epsilon}^*$  will be at most as large as  $f^*$ . Furthermore,  $f_{\epsilon}^* \to f^*$  as  $\epsilon \to 0$ . This is because [35]:

$$f^* \ge \left(1 - \frac{\epsilon}{r_{max}}\right) f^* + \frac{\epsilon}{r_{max}} f^*_{\epsilon} \ge f^*_{\epsilon}$$
(90)

The middle term of (90) can be seen as an example of a mixed policy that picks  $f^*$  with  $1 - \frac{\epsilon}{r_{max}}$  probability and  $f^*_{\epsilon}$  with  $\frac{\epsilon}{r_{max}}$  probability.

If the long term receiving frame interval is  $\bar{r}_{\epsilon}$ , then there exists a stationary randomized policy that stabilizes U(t) by setting the long term playout interval to  $p^*$  [36]. That is with  $\bar{r}_{\epsilon}$  as the long term average arrival rate into the discontinuity penalty virtual buffer U(t). If the policy chooses the playout interval p(t) over time such that the long term receiving frame interval is  $p^*$ , then U(t) can stabilize since  $p^* > \bar{r}_{\epsilon}$ . Thus, it will not grow infinitely large over time. Under such a policy, the Lyapunov one step drift bound (19) would be calculated using (84) as:

$$\Delta(U(t)) - V\mathbb{E}\{g(f(t)) - h(p(t))|U(t)\}$$

$$\leq B - \mathbb{E}\{U(t)\beta(t)|U(t)\}$$

$$- Vg(f_{\epsilon}^{*}) + U(t)\bar{r}_{\epsilon}$$

$$- U(t)p^{*} + Vh(p^{*})$$

$$\leq B - \mathbb{E}\{U(t)\beta(t)|U(t)\}$$

$$- Vg(f_{\epsilon}^{*}) + U(t)(p^{*} - \epsilon)$$

$$- U(t)p^{*} + Vh(p^{*})$$
(91)

By rearranging the terms:

$$\Delta(U(t)) - V\mathbb{E}\{g(f(t)) - h(p(t)) | U(t)\}$$
  

$$\leq B - V\left(g(f_{\epsilon}^{*}) - h(p^{*})\right) - \epsilon U(t)$$
(92)

$$\leq B - V\left(g(f_{\epsilon}^*) - h^*\right) - \epsilon U(t) \tag{93}$$

By taking expectations, summing over the timeslots  $\tau \in [0 .. M - 1]$  and using the non-negativity of L(U(t)) to drop the term  $\mathbb{E}\{L(U(M - 1))\}$ , we get:

$$-\mathbb{E}\{L(U(0))\} - V \sum_{\tau=0}^{M-1} \mathbb{E}\{g(f(\tau)) - h(p(\tau))\}$$
  
$$\leq MB - \epsilon \sum_{\tau=0}^{M-1} \mathbb{E}\{U(\tau)\} - VM(g(f_{\epsilon}^{*}) - h^{*})$$
(94)

To prove the discontinuity penalty bound (72), we divide (94) by  $M\epsilon$  and rearrange its terms to obtain:

$$\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{U(\tau)\} \leq \frac{B + V(G_{max} - H_{min})}{\epsilon} + \frac{\mathbb{E}\{L(U(0))\}}{M\epsilon}$$
(95)

Taking the limits of (95) as  $M \to \infty$  and setting  $\epsilon = r_{max}$  yields (72). Setting  $\epsilon = r_{max}$  is done to minimize the bound, as a particular choice for  $\epsilon$  will only affect the bound calculation and will not affect the policies in any way.

To prove the utility bounds, note that the concavity of the frame quality function g(f(t)) and the convexity of the playout distortion function h(p(t)) together with Jensen's inequality implies the following:

$$\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{g(f(\tau))\} \le g\left(\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{f(\tau)\}\right)$$
(96)

$$\frac{1}{M}\sum_{\tau=0}^{M-1} \mathbb{E}\{h(p(\tau))\} \ge h\left(\frac{1}{M}\sum_{\tau=0}^{M-1} \mathbb{E}\{p(\tau)\}\right)$$
(97)

If we divide (94) by MV and rearrange it, we obtain:

$$\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{g(f(\tau))\} - \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{h(p(\tau))\} \\ \ge (g(f_{\epsilon}^*) - h^*) - \frac{B}{V} - \frac{\mathbb{E}\{L(U(0))\}}{MV}$$
(98)

To obtain the frame quality bound, we use the fact that  $h(p(t)) \ge H_{min}$  in (98) and rearrange to get:

$$\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{g(f(\tau))\} \\
\geq (g(f_{\epsilon}^{*}) - h^{*}) + H_{min} - \frac{B}{V} - \frac{\mathbb{E}\{L(U(0))\}}{MV}$$
(99)

Using (96) and taking the limits of (99) as  $M \to \infty$  yields:

$$\liminf_{M \to \infty} g(\bar{f}) \ge (g(f_{\epsilon}^*) - h^*) + H_{min} - \frac{B}{V}$$
(100)

Again, the above bound can be maximized by taking the limit as  $\epsilon \to 0$ . This produces the frame quality bound (73).

To obtain the playout distortion bound, we use the fact that  $g(x(t)) \leq G_{max}$  in (98) and rearrange to get:

$$\frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{h(p(\tau))\}$$
(101)

$$\leq \frac{B}{V} - (g(f_{\epsilon}^*) - h^*) - G_{max} + \frac{\mathbb{E}\{L(U(0))\}}{MV}$$
(102)

Using (97) and taking the limits of (101) as  $M \to \infty$  yields:

$$\limsup_{M \to \infty} h(\bar{p}) \le \frac{B}{V} - (g(f_{\epsilon}^*) - h^*) - G_{max}$$
(103)

Taking the limit as  $\epsilon \to 0$  maximizes the above bound and produces the frame quality bound (74).

## E. Delay Constrained Optimization Stability and Performance Bounds

To obtain the performance bound for the delay constrained Lyapunov optimization policies in (28), we develop a policy to enforce a limit on the maximum operating size of the delay accumulator. To determine such a policy, we observe that for the decoder objective in (28) (last term) to be non-negative, we need:

$$U(t)p(t) - Vh(p(t)) - X(t)p(t) \ge 0$$
(104)

$$\Rightarrow U(t)p(t) - X(t)p(t) \ge 0 \tag{105}$$

$$\implies X(t) \le U(t) \tag{106}$$

Therefore, we introduce the following policy:

If  $X(t) \leq \underline{U}$ solve p(t) by maximizing the last term of (28)

else

$$p(t) = p_n + t_d \tag{107}$$

Where  $\underline{U}$  is a positive constant that represents how easily the maximum operating size of X(t) gets enforced,  $\underline{U}$  is set to 100 in our experiments. (107) together with (24) ensures that X(t) does not accumulate anymore in subsequent timeslots. With (107), we introduce the following corollary:

**Corollary 2.** If the network delay variation r(t) is i.i.d over the timeslots, the frame quality function is bounded as  $g(f(t)) \leq G_{max}$  and the playout distortion function is bounded as  $h(p(t)) \geq H_{min}$ . Then implementing the optimization policies from (28) in each timeslot stabilizes the discontinuity penalty U(t) and satisfies the following performance bounds:

$$\limsup_{M \to \infty} \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{U(\tau)\} \le \frac{B + \widetilde{C} + V(G_{max} - H_{min})}{p_{max}}$$
(108)

$$\liminf_{M \to \infty} g(\bar{f}) \ge (g^* - h^*) + H_{min} - \frac{B + C}{V}$$
(109)

$$\limsup_{M \to \infty} h(\bar{p}) \le \frac{B+C}{V} - (g^* - h^*) - G_{max}$$
(110)

Where V > 0,  $g^*$ ,  $r^*$ ,  $\bar{f}$  and  $\bar{p}$  are defined as in theorem 1. B is defined as in (18) and  $\tilde{C}$  is defined using C as:

$$\tilde{C} = C + p_{max}(\underline{U} + p_{max}) \tag{111}$$

Furthermore, implementing limit enforcing policy (107) on the delay accumulator X(t) would result in it deterministically upper bounded for all timeslots t as:

$$X(t) \le \underline{U} + p_{max} \tag{112}$$

*Proof:* Bound (112) is proved by induction. It can be easily seen that (112) is satisfied at time 0. Assume that (112) holds at the current time  $t \ge 0$ , then we need to prove that  $X(t+1) \le \underline{U} + p_{max}$  in the next timeslot t+1. We have two cases:

1.  $X(t) \leq \underline{U}$ : Here  $X(t+1) \leq \underline{U} + p_{max}$ . Since from (24), the maximum delay added to X(t) in one timeslot is  $p_{max}$ .

2.  $X(t) > \underline{U}$ : In this case, the limiting policy (107) will be triggered and X(t) will not increase in time t + 1 resulting in  $X(t+1) \le X(t) \le \underline{U} + p_{max}$ .

This proves (112). To prove (108), (109) and (110), observe that using (112):

$$X(t)\mathbb{E}\{p_n + t_d - p(t)|U(t), X(t)\}$$
  

$$\geq -X(t)\mathbb{E}\{p(t)|U(t), X(t)\}$$
  

$$\geq -(\underline{\mathbf{U}} + p_{max})p_{max}$$
(113)

Using (113) and (111) in (28) will yield:

$$\Delta(U, X) - V\mathbb{E}\{g(f) - h(p)|U, X\}$$

$$\leq B + \tilde{C}$$

$$- \mathbb{E}\{Vg(f) - UF(f)|U, X\}$$

$$- \mathbb{E}\{Up - Vh(p) - Xp|U, X\}$$
(114)

The proof then proceeds exactly as in theorem 1.

## F. Stability and Performance Bounds of Policies with Network Delays

**Corollary 3.** If the network delay variation r(t) is i.i.d over the timeslots, the frame quality function is bounded as  $g(f(t)) \leq G_{max}$  and the playout distortion function is bounded as  $h(p(t)) \geq H_{min}$ . Then implementing the optimization policies from (40) in each timeslot stabilizes the discontinuity penalty U(t) and satisfies the following performance bounds:

$$\limsup_{M \to \infty} \frac{1}{M} \sum_{\tau=0}^{M-1} \mathbb{E}\{U(\tau)\} \le \frac{B''' + V(G_{max} - H_{min})}{p_{max}}$$
(115)

$$\liminf_{M \to \infty} g(\bar{f}) \ge (g^* - h^*) + H_{min} - \frac{B'''}{V}$$
(116)

$$\limsup_{M \to \infty} h(\bar{p}) \le \frac{B'''}{V} - (g^* - h^*) - G_{max}$$
(117)

Where V > 0,  $g^*$ ,  $r^*$ ,  $\bar{f}$  and  $\bar{p}$  are defined as in theorem 1. B''' is defined as:

$$B''' = B' + d_b(d_f + 1) \left( r_{max}^2 + r_{max} p_{max} \right)$$
(118)

with B' from (35).

*Proof:* We first look at (38). Note that with the recursive definition of U(t) (31), (38) can be rewritten as:

$$\Delta(U) \leq B'' - U\mathbb{E}\left\{\gamma_{d_f} \left| U\right\} - \mathbb{E}\left\{-U_{d_b} F_{d_f} \left| U\right\}\right\}$$
  
$$\leq B'' - \mathbb{E}\left\{\left(\gamma_{d_f} \left[U_{d_b} - \gamma\right]^+ + F\gamma_{d_f}\right) \left| U\right\} - \mathbb{E}\left\{-U_{d_b} F_{d_f} \left| U\right\}\right\}$$
(119)

Since  $[U_{d_b} - \gamma]^+ \leq U_{d_b}$ , the above equation becomes:

$$\Delta(U) \leq B'' - \mathbb{E}\left\{\left(\gamma_{d_f} U_{d_b} + F \gamma_{d_f}\right) \left| U\right\} - \mathbb{E}\left\{-U_{d_b} F_{d_f} \right| U\right\}$$
(120)

Note that the term  $F\gamma_{d_f}$  can be upper bounded as:

$$F\gamma_{d_f} \le d_b(d_f + 1)r_{max}p_{max} \tag{121}$$

Using (121) in (120) yields:

$$\Delta(U) \leq B''' - \mathbb{E}\left\{\gamma_{d_f} U_{d_b} - U_{d_b} F_{d_f} \middle| U\right\}$$
(122)

with B''' defined in (118). This then implies that (39) can be bounded as:

$$\Delta(U) - V\mathbb{E}\{g(f(t)) - h(p(t))|U\}$$

$$\leq B'' - \mathbb{E}\{U\gamma_{d_f} - U_{d_b}F_{d_f}|U\}$$

$$- \mathbb{E}\{Vg(f(t)) - Vh(p(t))|U\}$$

$$\leq B''' - \mathbb{E}\{U_{d_b}\gamma_{d_f} - U_{d_b}F_{d_f}|U\}$$

$$- \mathbb{E}\{Vg(f(t)) - Vh(p(t))|U\}$$
(123)

The proof then proceeds as in theorem 1.