TR 07-013

A Network-Aware Approach for Video and Metadata Streaming

Aravindan Raghuveer, Ewa Kusmierek, and David Du

April 30, 2007

A Network-Aware Approach for Video and Metadata Streaming

Aravindan Raghuveer, Ewa Kusmierek, David H.C. Du
Department of Computer Science and Engineering
University of Minnesota,
{aravind,kusmiere, du}@cs.umn.edu

***Abstract*— Providing Quality of Service (QoS) for Internet-based video streaming applications requires the server and/or client to be network-aware and adaptive. We present a dynamic rate and quality adaptation algorithm where the server varies its sending rate (without varying the quality level) to adapt to the network and client conditions and only as a last resort, does quality adaptation. We place the adaptation logic at the client since it has better knowledge about both the demand (buffer conditions, variable bit-rate requirements) and supply (network conditions). Our approach is unique because the server's sending rate is calculated based on the client's varying demand (consumption rate) and the network status. Also, we do not model the network as a black-box but instead augment endpoint observations with a feedback from the network to represent its status more precisely. To make an informed adaptation decision, the client requires sizes of all frames in the variable bit rate video. But the overhead involved in sending this metadata is significant. So we propose a lossy compression technique to reduce the amount of control information and consequently the overhead. We also present a scheduling algorithm, DART(Dynamic Scheduling Algorithm for Reduced Trace delivery), to deliver the compressed control information to the client. This algorithm can be used to deliver any form of metadata (like subtitles, alerts etc), especially in applications like IP-TV. Simulations show that the proposed techniques can significantly improve user perceived QoS when compared to other popular adaptation methods.**

***Index Terms*— Rate adaptation, QoS, underflow curve, trace compression, metadata delivery, IP-TV**

I. Introduction

Internet based video streaming applications like IP-TV, video-conferencing, video classrooms etc. are gaining increased popularity and are being widely deployed [1], [2]. In these applications, the server transfers the requested audio-visual content to the client as a continuous stream and the client consumes the data as it arrives. Such multimedia applications have stringent timing requirements for data consumption. The client can use a *media unit* only if it is delivered *on time*. This requirement imposes bandwidth, delay and loss demands on the underlying network that is responsible for delivering the data. But in a best effort network like the Internet, network parameters such as delay and bandwidth vary unpredictably providing no guarantee on timely delivery. This mismatch forms the fundamental challenge in streaming video over the Internet.

A widely accepted technique to counter this challenge is network-aware demand adaptation. Network aware applications sense the varying resource availability and accordingly adapt their demands. [3] proposes a framework for network-aware applications to provide end-to-end Quality of Service. Network aware adaptation mechanisms operate in two distinct steps. $a)$ Evaluate the status of the network. $b)$ Based on this evaluation, perform adaptation, that maximizes the user perceived quality. One of the key contributions of this paper is such an adaptation technique for video that is not only network-aware but is also conscious of the quality at the client.

Many methods have been proposed to track the rapidly changing network conditions. A popular approach is to model the network as a *black box* and use end-point observations like packet loss ratio, arrival rate etc. to decide the network status. We argue that with more information about the network, finer adaptation decisions can be made to maximize the perceived quality. We employ a simple ECN-like mechanism [4], at the core routers, to gather backlog information and hence do not impose considerable control overhead.

The adaptation steps should maximize the user QoS for the given network conditions. This means that frequent changes in quality should be avoided and playback at the client should be continuous. The proposed adaptation algorithm strives to ensure both. The problem is further complicated for variable bit-rate (VBR) video where the bandwidth demand of the client varies over time. Hence the proposed strategy uses information about the network, current buffer occupancy at the client and future requirement of the video to make fine-grained adaptation decisions.

To estimate the bandwidth requirement, our scheme requires video-specific metadata; in particular sizes of all frames in the video. This requirement poses two challenges: 1) The amount of metadata required for typically accessed video streams, like movies, is high and hence causes considerable overhead. 2) Delivering the control data to the client poses unique challenges like time-sensitivity, reduced startup delay and controlled bandwidth overhead.

To address the first problem, we propose a lossy compression scheme to approximate the underflow curve. (The underflow curve is a cumulative function of the frame sizes of a video.) Our simulations show that we can reduce the control overhead by upto 94% without significantly affecting the adaptation decisions. To tackle the second problem, we propose a scheduling algorithm called DART. In this scheme, we deliver a minimal amount of control data ahead of playback and multiplex the remaining control data along the media stream.

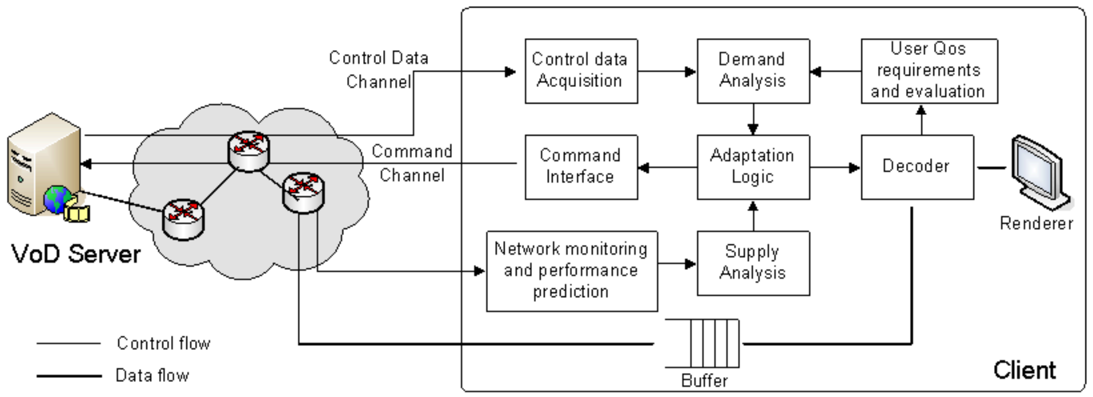

Fig. 1. Unified architecture for video and metadata delivery

This paper makes three primary contributions.

- We show that it is possible to dynamically adapt the bandwidth demand to the network conditions without always compromising on video quality. (Section III)
- We present a method to compress the control information and hence reduce the overhead involved in its delivery. (Section IV-A)
- We present a scheduling algorithm, DART (Dynamic Scheduling Algorithm For Reduced Trace delivery), to deliver the metadata to the client in a bandwidth-efficient and network-aware fashion. (Section IV-B)

Figure 1 shows the proposed unified framework to tackle the issues discussed in the above paragraphs.. The core of this framework is the adaptation logic which uses the *demand analysis unit* (User QoS) and the *supply analysis unit* (Network QoS) to make adaptation decisions. The *demand analysis unit* uses the underflow curve supplied by the *control data acquisition unit* and feedback on user perceived quality to estimate the amount of data needed in the future. The *supply analysis unit* uses the network feedback and performance prediction to estimate the amount of data that will be received in the future. A mismatch between the demand and the supply triggers an adaptation step. All adaptation decisions are conveyed to the server, as commands, through the *command channel*.

The rest of the paper is organized as follows. In the next section, we discuss relevant principles of network aware rate adaptation. In Section III, we present a scheme for dynamic rate selection followed by a complete rate and quality adaptation algorithm. Section IV-A describes our approximation scheme to compress the underflow curve. DART, an algorithm for delivering the reduced trace to the client, is described in Section IV-B. Section V presents related work and Section VI concludes the paper.

II. Principles of Network Aware Rate Adaptation

The 3Rate technique [5] shows that with limited additional knowledge on network conditions, the sending rate of the video can be controlled to avoid frequent quality changes and hence improving perceived quality. We use this finding as the underlying premise in this work. However in [5], only three rates are used and these rates are chosen in a static fashion without knowledge of the network conditions. In Section III, we point out the necessity of network-aware selection of the sending rate and present a technique to choose it dynamically.

A. Network Status Evaluation

To adapt to the varying network conditions, the client should have fairly accurate information about the network status. Current network status can be evaluated using two observations: *available bandwidth and network backlog*. Available bandwidth is determined through end-point observations while network backlog status is obtained by a feedback mechanism.

a) **Available Bandwidth :** The arrival rate, R_a, is periodically measured, at the client, using the *packet train* technique [6]. The results are used to qualitatively estimate the available bandwidth, B_a, as shown below:

$$\text{If } R_a < R_s \text{ then } B_a < R_s$$
$$\text{If } R_a \geq R_s \text{ then } B_a \geq R_s$$

where R_s is the transmission rate of the server.

b) **Network Backlog:** Multilevel ECN (MECN) [7], an active queue management strategy, is used to provide feedback about the backlog status at the core routers. MECN capable routers define three queue length thresholds: min_th, mid_th and max_th. They mark the beginning of incipient, moderate and severe congestion. During incipient and moderate congestion the outgoing packets are marked with ’10’ and ’11’, respectively. with certain probability. The ECN bits of a packet show the state of the most congested router in the path.

B. The 3Rate Adaptation Scheme

Due to varying network conditions, the arrival rate at the client fluctuates compared to a constant sending rate. The client’s buffer is used to cushion these fluctuations. If the client experiences a low arrival rate for considerable amount of time, starvation may occur and the client is forced to switch to a lower quality level with lower bandwidth requirements. The 3Rate scheme tries to postpone, and possibly avoid, such a quality drop by increasing the server’s sending rate hoping that the arrival rate would consequently increase. However, the rate increase is done only when there is no congestion in the network and quality drop is requested when there is congestion.

TABLE I
THE 3RATE ADAPTATION SCHEME

Case	$R_a < R_s$	ECN Value	Buffer after rtt=k frames	Adaptation Decision
1	false	00	$w_l \leq k \leq w_h$	inc. quality
2	true	00	$w_l \leq k \leq w_h$	none
3	true	00	$k < w_l$	inc. rate r$\longrightarrow r_h$
4	true	10	$w_l \leq k \leq w_h$	none
5	true	10	$k < w_l$	dec. quality
6	true	11	-	dec. quality
7	false	-	$k > w_h$	dec. rate r$\longrightarrow r_l$

For each quality level there are three sending rates: the high rate r_h, the default rate r, and the low rate r_l. The client can request from the server a temporary switch to either r_h or r_l in an attempt to increase or decrease the arrival rate. The network status and the client buffer occupancy are periodically evaluated, and either rate or quality adaptation is done as shown in Table I. Two buffer thresholds, low watermark w_l and high watermark w_h, are defined to predict the buffer underflow and overflow, respectively. *Buffer after rtt* is the predicted buffer occupancy (in number of frames) after one rtt (round trip time). It is calculated by predicting the amount of data (in bytes) that would be received in the next rtt and mapping it onto the frames that follow in the play-out order. So it is assumed that sizes of all video frames are known beforehand. To summarize, [5] shows that with additional information about network status, the server's sending rate can be varied to maximize perceived quality. In the next section, we first argue why choosing a-priori values for r r_h or r_l is not optimal and then provide tehniques to choose them dynamically.

III. DYNAMIC RATE ADAPTATION

The 3rate algorithm statically defines three sending rates r_h, r_l, r. The client requests the server to send at a higher rate r_h or switch to a lower rate r_l based on its buffer occupancy and network conditions. In Section III-F we show by simulation results that switching to the statically chosen sending rates r_h and r_l may not be optimal. To explain these observations theoretically, consider the following cases.

Rate increase is requested when buffer underflow is predicted. But if the pre-determined value of r_h is not high enough, the consequent increase in the arrival rate at the client may not be sufficient to avoid underflow. On the other hand, when r_h is higher than what is required to avoid underflow, the client is being unaccountably aggressive in acquiring bandwidth. Symmetric cases can be found for the rate decrease case. Therefore, choosing sending rates without taking the network conditions and the client buffer occupancy into consideration may lead to sub-optimal results.

In the next two sections we present algorithms that select rate dynamically for both rate increase and decrease. For now we still assume that the client knows the entire underflow curve before playback starts. In Section IV we will present techniques to relax this assumption.

A. Algorithm for Dynamic Rate Increase

A rate increase is done when the client faces a buffer underflow and there is no incipient congestion in the network. The proposed algorithm dynamically calculates the new sending rate in two steps. We first find a target arrival rate at the client, which will avoid the buffer underflow. Then we find transmission rate that will result in the targeted arrival rate. The new sending rate is a function of the targeted arrival rate and the responsiveness of the network and is computed as:

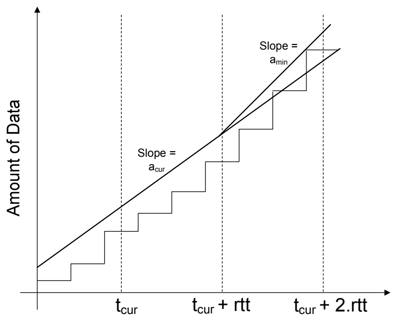

Fig. 2. Selecting minimum required arrival rate,a_{min}

$$r_{new} = (1 + S_f)\beta a_{min} \tag{1}$$

where: β is the *network response index* and a_{min} is the minimum arrival rate required to avoid buffer underflow. S_f is the safety factor ($0 \leq S_f < 1$) to make sure that we do slightly better than the minimum. Network response index is calculated as r_{cur}/a_{cur}, where r_{cur} is the current transmission rate and a_{cur} is the corresponding arrival rate. Figure 2 shows an example of selecting a_{min}. r_{new} represents the minimum transmission rate required for the arrival rate to reach value a_{min}. The calculations are done at the granularity of a single frame playback time in a discrete time model adopted. Specifically, the minimum arrival rate required is computed as:

$$a_{min} = \max_{t_1 < t \leq t_2} \frac{\sum_{i=1}^{t} f_i - \sum_{i=1}^{t_1} f_i}{t - t_1} \tag{2}$$

where: $t_1 = t_{cur} + rtt$, $t_2 = t_{cur} + 2rtt$, t_{cur} is the time at which the rate increase step is initiated and f_i represents the size of the frame that is played out at time i. Let $\hat{t}$ be the value of t for which the maximum is reached in Equation 2. The length of the time interval for which the rate increase has to be applied must be no shorter than $\hat{t} - t_1$. Requesting rate increase for longer time intervals will make the scheme more aggressive.

The fundamental aim of the rate increase is to bridge the gap between the observed and required arrival rate. When the network is not responsive, the sending rate should be greater than a_{min} to boost the arrival rate to a_{min}. β quantifies this degree of sluggishness in the network. The higher the difference between the sending and arrival rate, the less responsive the network is to any rate changes and higher the value of β will be. The requested rate is always the minimum needed and so the degree of aggression is under control.

To address a similar problem in a different setting, TCP-Vegas [8] tracks the relation between observed arrival rate and expected arrival rate to control the window size. Its fundamental contribution, over other previous flavors being, that TCP-Vegas can proactively adapt the window size before

Algorithm 1 convertToFrames($data$)

```
1: counter=0; frames=0
2: while  counter < data do
3:     counter = counter + $f_{t_{cur}+i}$
4:     frames = frames + 1
5:     i = i + 1/fps
6: end while
7: return frames
```

congestion (packet drops) actually sets in. This also avoids to a certain extent frequent rate oscillations of the sender. D-Rate possesses both these characterstics which are very essential for maintaining consistent video quality in streaming video. Hence, the philosophy beind the proposed D-Rate scheme is similar to TCP-Vegas.

B. Algorithm for Dynamic Rate Decrease

A buffer overflow condition typically arises when network congestion is clearing up and one or more rate increases were requested in the past. By transmitting at a higher rate, we possibly send more data into the network than what the client buffer can handle. This data reaches the client when the network backlog clears up causing buffer overflow. To handle this situation, the rate decrease step is used. The client keeps track of the additional amount of data sent ΔD as compared to the default transmission schedule with sending rate r. The new sending rate is calculated as:

$$r_{new} = r - \frac{\Delta D}{\Delta t_{dec}} \tag{3}$$

and Δt_{dec} is the time for which rate decrease is requested. The maximum value of Δt_{dec} is bounded by the predicted time to overflow. This can be calculated as follows:

$$max\Delta t : b(t_{cur}) + \left(r.(\frac{rtt}{2} + \Delta t) - \Delta D \right) - \sum_{i=t_{cur}}^{t_{cur}+\Delta t} f_i \leq B \tag{4}$$

where $b(t_{cur})$ is the buffer occupancy (in bytes) at time t_{cur} when the request is made. f_i is the size of the frame that is played out at time i and B is the size of the client buffer.

C. Algorithm for Quality Increase

After sufficient "*no congestion*" history has been built, quality of the video stream can be increased. One technique to increase quality is to request the server to shift to the higher quality version in a single atomic step [9]. We refer to this technique as the *single step* method. We instead increase the quality of the video in two phases. In the first phase, *bandwidth probing*, we conduct rate increase experiments to check if there is enough available bandwidth to support the higher quality video. In the next phase, we increase the quality if enough bandwidth was found to be available. Doing quality increase in two phases provides two advantages over the *single step* method. Firstly, consider a case wherein the available bandwidth is not sufficient to support the higher quality stream. When the *single step* method is used, it results

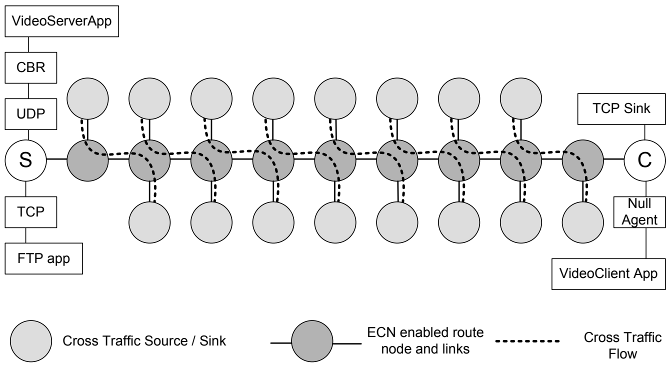

Fig. 3. ns2 Simulation Model

in congestion and consequently results in quality drop. Our method, on the other hand, tests for bandwidth availability with rate increase steps (without varying the quality). So when congestion is imminent, the rate is dropped and quality increase is suspended. The client is unaware of a failed quality increase experiment. So the perceived QoS is not affected. Next, [10] shows that by directly shifting to the higher quality video, the bandwidth demand of the client increases by a large amount instantaneously. This is unfair to competing TCP applications that increase their bandwidth demands additively. So to be fair to competing TCP applications, we increase our bandwidth demand additively and then smoothly shift to the higher quality video.

Bandwidth probing: In this phase, the sending rate of the video is increased by a small step (ΔR), for a duration Δt, and the result on the network is observed. While there is no indication of congestion, we repeat the rate increase and network observation steps until the sending rate reaches a value greater than or equal to the bandwidth of the higher quality video (R_{new}). However if congestion is indicated, we immediately suspend the experiment and remove the extra data from the network with a rate decrease step.

The rate increase steps, may cause overflow of the client buffer because the input rate to the client is increased but the consumption rate remains the same. So before initiating the *bandwidth probing* phase, we check if the client has extra buffer to support the experiment. Let R_{cur} be the sending rate of the video before bandwidth probing is started and let n_{exp} be the total number of rate increase experiments. ΔD represents the extra amount of data pushed into the network due to the bandwidth probing phase.

$$\begin{aligned} n_{exp} &= \frac{R_{new} - R_{cur}}{\Delta R} \\ \Delta D &= \sum_{i=1}^{n_{exp}} (R_{cur} + i.\Delta R).\Delta t \end{aligned} \tag{5}$$

D. Dynamic Rate, Quality Adaptation

In this section we present a complete adaptation algorithm that uses the dynamic rate selection schemes described in previous sections. We first interpret the resource availability metrics and then proceed to describe the adaptation algorithm itself.

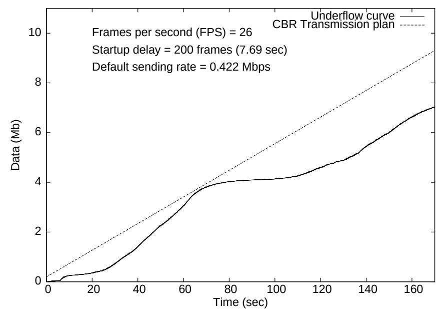

Fig. 4. Underflow curve and transmission plan for the video - Star Wars

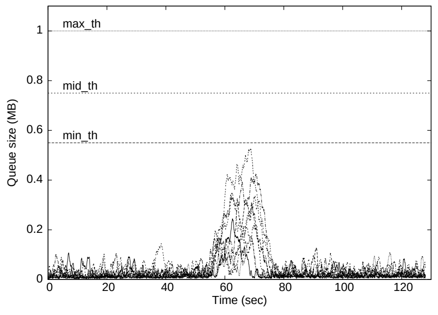

Fig. 5. Scenario-1 (without adaptation): Average queue sizes

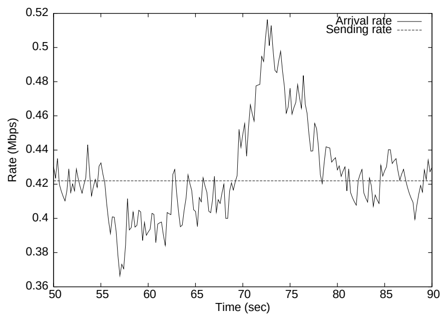

Fig. 6. Scenario-1 (without adaptation): Arrival rate

1) Network Resource Availability Metrics :

a) Network Backlog: The ECN value in each incoming packet is used to interpret the network backlog. If a packet with ECN value is ’10’ is received, it means that the first threshold is reached in at least one router and hence, rate should not be increased further. A value of ’11’ prompts a quality drop.

b) Arrival Rate: The current arrival rate is calculated every Δt seconds as: $r_a = (\sum p_i/\Delta t)$ where p_i is the packet size. The predicted arrival rate for the next Δt is calculated using the exponential smoothing technique. The parameter μ controls the weight given to history.

$$\hat{r_a}(t+1) = \mu\hat{r_a}(t) + (1-\mu)r_a \tag{6}$$

2) Client Metric: Buffer Occupancy: The buffer occupancy is predicted 2rtt into the future based on the arrival rate. The buffer occupancy after Δt seconds from the current time t_{cur} is calculated by adding to the current buffer occupancy, the difference between number of frames expected to be received and the number of frames that will be consumed in Δt:

$$\begin{aligned} buf(t_{cur}+\Delta t) &= buf(t_{cur}) + \xi(\hat{r_a} * \Delta t) - fps * \Delta t \\ \text{and} \quad \xi(d) &= \{\max\ n : \sum_{i=k+1}^{k+n} f_i <= d\} \end{aligned} \tag{7}$$

where: $0 \le \Delta t \le 2rtt$, k is the highest frame index in the buffer and fps is the consumption rate. The function $\xi()$ uses the underflow curve (cumulative function of frame sizes) to translate the predicted amount of data that will arrive to number of frames. Algorithm-1 shows the pseudocode of the function $\xi()$. We predict 2rtt into the future, because rtt is the minimum time required for the rate change request to take effect.

3) The Adaptation Algorithm (D-Rate): Periodically, the client measures the round-trip time (rtt) and arrival rate (a_{cur}). Based on this, the buffer occupancy after 2rtt is predicted. When buffer underflow is predicted and there is no congestion, the client requests rate change to r_{new} (dynamic counterpart of r_h in 3Rate) as in Section III-A. For a buffer overflow case, the client calculates r_{new} (dynamic counterpart of r_l in 3Rate) as in Section III-B. After sufficient *no congestion* history has been built, quality increase is done as described in section III-C. Other cases are handled as in Table I. When quality drop has to be done, the level to which quality is dropped depends on the severity of congestion indicated by the ECN value. When the client requests a change in the sending rate, the value of r_{cur} is correspondingly updated.

To summarize, our scheme takes the network and client conditions into consideration before requesting a rate adjustment. In the rate increase case, if the network is very responsive, i.e., $\beta \to 1$, then we requested rate $r_{new} \to a_{min}$. On the other hand if the network is sluggish, we request a higher rate as β is high. The rate decrease is a compensation step to remove the extra data pumped into the network and return the client to its default schedule.

E. TCP Friendliness of D-Rate

TCP Friendliness and maintaining constant video quality are two conflicting, yet important, demands that Internet-based video streaming applications should address. We highlight in the following paragraphs some salient features of D-Rate that make it TCP-friendly. However we point out that there is scope for improvement of the D-Rate scheme on the TCP friendliness aspects (eg. a slow start phase needs to be added).

1. Controlled Rate Increase Steps: A rogue rate increase step in D-Rate can lead to unfair consumption of bandwidth and starving competing TCP applications. But DRate addresses this concern by controlling the rate increase in three fronts: magnitude, duration and frequency. First of all, the maximum rate that can be requested by a rate increase step is bounded by a system defined parameter R_{max}. R_{max} can be adjusted based on the history/knowledge of the downstream link characterstics or equation-based TCP friendliness techniques [11]. Next we always request a rate increase for a bounded duration, set in our experiments to 1 second. Finally, the effect of a rate increase on the network is carefully monitored before requesting the next one. This ensures that the frequency of rate increase is also under control.

2. Pre-emptive Quality drop: The rate increase step is employed only when we have no signs of congestion in the network. When there are signs of congestion, as observed over a small moving window, D-Rate requests for a quality drop (to reduce its throughput by atleast 0.5) in anticipation of a congestion. This pre-emptive quality drop step strikes a good balance between being TCP friendly and at the same time maintaining smooth quality at the client.

F. Evaluation and Results

In this section, we present simulation results to illustrate the salient features of the D-Rate scheme. We also compare

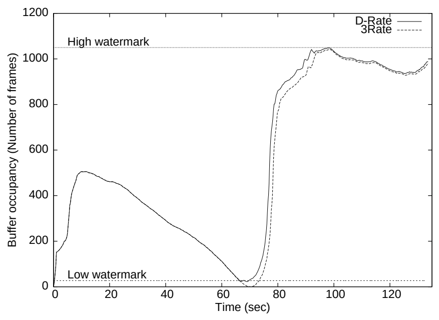

Fig. 7. Scenario-1: Buffer Occupancy for D-Rate and 3Rate schemes

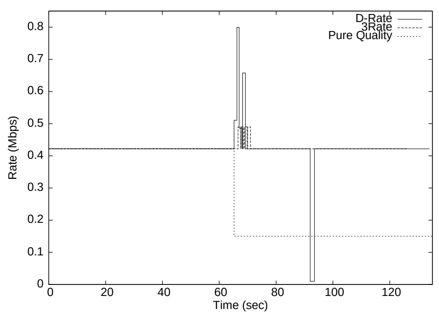

Fig. 8. Scenario-1: Sending Rate for 3Rate, D-Rate and pure quality schemes

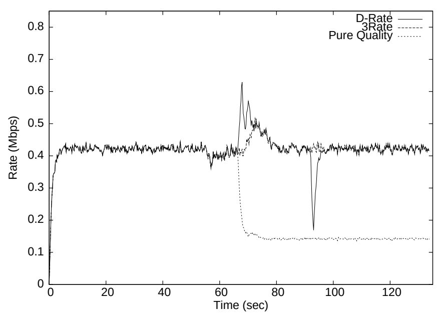

Fig. 9. Scenario-1: Arrival Rate for 3Rate, D-Rate and pure quality schemes

the performance of the D-Rate scheme with that of the 3Rate and the pure quality adaptation schemes under various traffic and input video trace conditions. The *pure quality adaptation* scheme is a generalization of all those schemes [12] [13] [14] that treat rate synonymously with quality. In this genre of techniques, adaptation (to network and buffer conditions) is done only by altering quality. Therefore case 3 in Table I is handled by a quality drop and case 7 (buffer overflow) will not occur for a feasible transmission schedule. Other cases are handled exactly same as described in Table I.

1) Simulation testbed: We use ns2 [15] to simulate our architecture shown in Figure 1. Figure 3 shows the precise simulation model that we use in our experiments. The simulation model consists of three distinct components:

- *Video server and Client:* The VoD server sends data at a specified rate over UDP. The client based on the consumption demand and network status, sends adaptation commands to the server through a reliable TCP connection.
- *MECN capable routers:* We simulate the network cloud, shown in Figure 1, with 10 intervening nodes. These nodes (routers) mark outbound packets using the MECN scheme as described in Section II.
- *Cross traffic generators:* We generate UDP cross traffic in order to explicitly control the bandwidth available to the video stream. A combination of CBR and poisson traffic generators within the network cloud generate cross traffic on intervening routers. The video stream interacts and competes with the CBR cross traffic at each hop as shown in Figure 3. We vary the rate of the cross traffic to vary the backlog at each router. This indirectly controls the arrival rate and the congestion level in the network.

2) Scenario-1: In the first scenario, we demonstrate the network awareness and dynamic rate selection properties of the D-Rate scheme. The video trace for MPEG-4 encoded *Star Wars* [16] is used in this experiment. The scheme described in [17] is used to determine the default sending rate of the server. The underflow curve and transmission plan for this trace is shown in Figure 4. We increase the rate of the cross traffic between 55 and 70 sec so that the arrival rate at each hop is greater than the link capacity. This increase builds up backlog at the intermediate routers and is shown in Figure 5. It can be observed that the queue lengths do not exceed the minimum threshold min_th and hence there is no risk of congestion. The influence of the backlog on the arrival rate is shown in Figure 6. It can be observed that the arrival rate is less than the sending rate during the period 57-70s.

A dip in the arrival rate causes underflow to be predicted according to Equation 7. Since there is no indication of congestion, both the 3Rate and the D-Rate schemes request rate increase. However the pure quality scheme resorts to a quality drop. Figure 7 shows the client buffer occupancy for the 3Rate and the D-Rate schemes. It can be seen that buffer underflow is avoided in the D-Rate scheme while it happens in the 3Rate case. This is because, in the 3Rate case, the statically chosen *high rate*, r_h, is not high enough to avoid underflow for the given network conditions. The pure quality adaptation avoids buffer underflow (not shown in figure) because the bandwidth demand of the client is significantly reduced by dropping the quality level. Figure 8 shows the sending rate for each scheme and Figure 9 shows the corresponding arrival rate. The arrival rate for the D-Rate case is significantly higher than the 3-Rate, in the interval 70-80s, due to higher sending rate selection. In the pure quality case, the drop in the sending rate (and consequently arrival rate) at 70s, corresponds to the quality drop requested.

The D-Rate algorithm initially requests a higher rate and consequently the arrival rate increases reducing the risk of underflow. (The safety factor S_f used was 0.1) The subsequent rate increases are of lesser magnitude demonstrating the adaptivity and fairness of the scheme. The four rate increase steps (of varying magnitude) cause extra data to be pushed into the network. When the backlog clears, the extra data reaches the client causing risk of overflow. So a rate decrease step is required at 92s. The sending rate is dropped to 0.01 Mbps for 1.5 seconds to remove the extra data in the buffer. Summarizing the results of this experiment: with the D-Rate scheme, the client does not experience a quality drop (pure quality scheme) or a discontinuity in playback (3Rate scheme). This provides better user perceived quality for the same client and network conditions.

3) Scenario-2: In the next set of experiments, we analyse the TCP friendliness of the D-Rate scheme during the rate increase and quality decrease steps. We attach a TCP-Vegas source (a ftp application) to the video server node and a TCP sink to the video client. This setup ensures that the TCP flow and video stream share intervening bottleneck links. We increase the bandwidth of links by 0.422 Mbps to ensure that the TCP flow gets a fair share of bandwidth during no-congestion periods. First we repeat the experiment in Scenario-

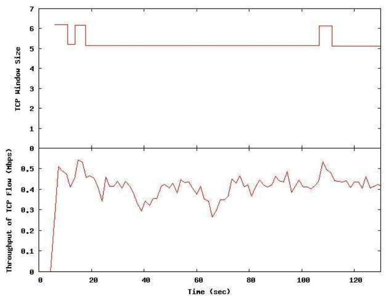

Fig. 10. Scenario-2: TCP Flow with rate increase step of D-Rate

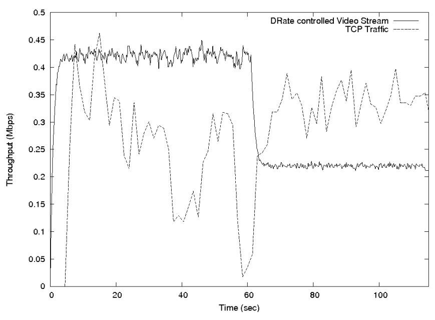

Fig. 11. Scenario-2: TCP Flow with quality drop step of D-Rate

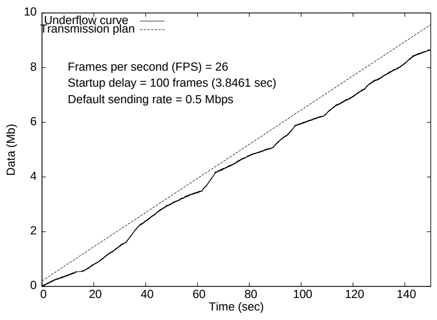

Fig. 12. Underflow Curve and Transmission plan for the video:*The Firm*

1 with the new setup and observe the throughput of the TCP flow. We observe that the buffer occupancy, arrival rate and sending rate charactersitcs of the video stream are very similar what was observed in Scenario-1. Figure 10 shows the throughput and window size of the TCP flow. We see that although the TCP throughput drops by a small amount during the rate increase step (62-69 sec), it regains it pretty quickly soon after. Also since D-Rate requests for a rate increase only when there is no evidence of congestion in the network, it does not cause a reduction in TCP's window size (as seen in the upper part of Figure 10). In the next experiment, we increase the rate of the cross traffic so that the ECN packets are marked 10. When faced with a possible underflow and signs of congestion in the network, D-Rate drops the quality by 0.5 predicting impending congestion (see Figure 11). The TCP sender also throttles its throughput rapidly and regains equally quick. D-Rate tries to smoothen out the rate changes so as to not affect the quality of the video stream adversely. After the quality drop, suffient *no congestion* history needs to be established before a quality increase can be requested. Also the D-Rate scheme is seen to be a bit sluggish in reducing its throughput when compared to the TCP flow. The sluggishness is accounted to the value of the low watermark level w_l which was set to 10 frames. With higher values of w_l, D-Rate can be more reactive in adjusting its rate in such situations.

4) Scenario-3: In this experiment, we further illustrate the behavior of the D-Rate adaptation algorithm with higher levels of backlog at the core routers. We simulate network and client conditions corresponding to cases 3, 4, 6 in Table I and study the behavior of the D-Rate scheme. We use the video trace of *The Firm* in this experiment. The underflow curve and transmission plan of the video is shown in Figure 12.

This experiment was carried out in three phases. In the first phase (time period 0 - 75s), the rate of the cross traffic was adjusted to keep the backlog below the minimum threshold *min_th*. At around 65s, buffer underflow is predicted and rate increase is done.(refer Figure 13) It was observed that even without any (rate) adaptation, buffer underflow does not occur. But the rate increase action taken by the D-Rate algorithm is warranted because no information is available on how long the network backlog may last.

In the second phase (time period 75 - 95s), the rate of the cross traffic is increased further to push the backlog above *min_th* but below the middle threshold *mid_th*. As a result the ECN bits of packets leaving the routers are marked '10'. The client understands this as a sign of of incipient congestion. As the buffer occupancy is healthy (shown in Figure 14), the client does not perform any adaptation.(refer Figure 13) Quality would have been dropped if buffer underflow/overflow was predicted.

In the third phase (time period 110 - 117s), the backlog is pushed higher above *mid_th* but below the maximum threshold *max_th*. This is done by increasing the rate of the cross traffic higher than in phase-2. Consequently, the ECN bits of outgoing packets are marked with '11'. The client immediately requests quality drop on receiving a packet with ECN bits marked '11'. However there is a small delay in the time at which server responds to the quality change request. This is because, the quality can be changed only at frame boundaries. The ECN values and sending rate changes are shown in the same graph (Figure 13.) to illustrate the correlation between the two quantities.

The 3Rate adaptation scheme behaves exactly same as D-Rate in phases 2 and 3. However in the rate increase step of phase-1, the rate r_h is set to 1.4 Mbps. Rate increase is requested twice as in the D-Rate case. But the extra aggressiveness (total amount of extra data pushed into the network) is unaccounted for because D-Rate rate selection algorithm shows that an aggregate lower rate is sufficient to avoid underflow. The pure quality scheme, on the other hand, drops quality twice (Phase-1 and Phase-3).

5) Scenario-4: In the following experiment, we demonstrate the quality increase algorithm in the D-Rate scheme (refer Section III-C) using the trace of the video, *The Firm*. The default quality level has an average bandwidth requirement 0.5 Mbps. The next higher level requires 0.75 Mbps and the highest quality level has an average bandwidth requirement of 1.0 Mbps. The bandwidth of the server-client path is set to a value greater than bandwidth requirement of the highest quality level.

The rate of the cross traffic is chosen such that the bandwidth available to the video stream is lesser than bandwidth requirement of the highest quality level. The available bandwidth is however sufficient to service the previous quality level.

$$B(q_{max} - 1) < B_a < B(q_{max})$$

where B_a is the available bandwidth and $B(q)$ is a function that provides the bandwidth required for a quality level q. *Quality increase experiments* are started twice (at 40s and 80s) after building sufficient *no congestion* history. The increment

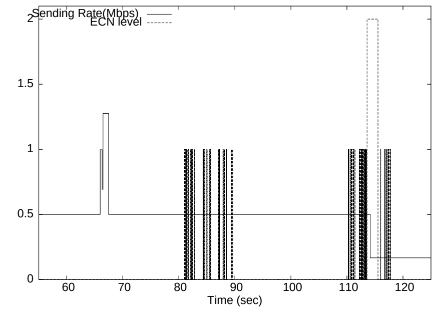

Fig. 13. Scenario-3: Sending Rate and ECN (D-Rate)

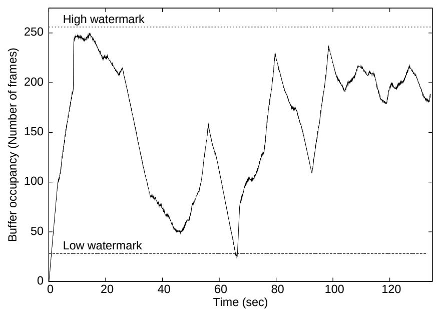

Fig. 14. Scenario-3: Buffer occupancy (D-Rate)

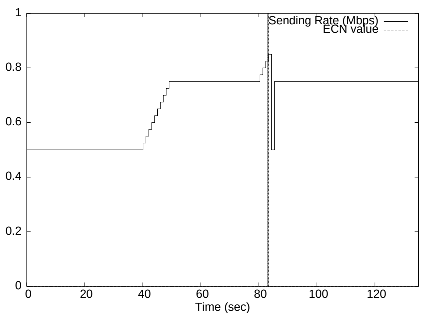

Fig. 15. Scenario-4b: Quality Increase Experiment

amount (ΔR) used was 0.025Mbps. Figure 15 shows the ECN values and sending rates for this experiment. The second quality increase step is suspended at the *bandwidth probing* phase because the *min_th* threshold is reached in atleast one of the router queues. However, since the second quality change has not yet been requested, the user does not perceive failure of the second quality increase experiment. Therefore the perceived QoS is not affected when a quality increase experiment fails. To summarize, we see that over a wide range of network conditions, D-Rate scheme can provide better perceived quality by leveraging on knowledge of the network, video and client characterstics.

IV. Control Data Compression and Delivery

We saw in the Section III-D.2 that the adaptation algorithm requires the underflow curve. This requirement presents two challenges. First, the overhead incurred due to sending this control information can be substantial. For example, the control overhead (amount of control information/amount of media data) for *Jurassic Park* (24MB, 89997 frames, 8 bytes of control data per frame) is 3%. This means that on the average, metadata consumes 3% of the bandwidth. For small frames (common in low quality video) metadata consumes up to 16% when seen on a per-frame basis. Section IV-A presents a technique to reduce this overhead.

The second challenge is posed by the fact that the control information, like media data, is time-sensitive. Sending the entire control information before start of playback increases the startup delay, which is undesirable. In Section IV-B, we describe a scheduling algorithm (DART) for interleaved delivery of the control information and the media data.

A. Control Data Compression

The cumulative function of the frame sizes with respect to their play-out times is known as the underflow curve. We approximate the underflow curve with a *piece-wise linear* function. The portion of the underflow curve for each piece is represented by the pair (N_p, D_p) where N_p is the number of frames and D_p is the amount of data contained in the portion p. A linear approximation is chosen, over a more accurate polynomial function because the curve fitting process is less computationally expensive. The server chooses a maximum tolerable per-frame error Δe and recursively splits the underflow curve into smaller portions until all portions satisfy the condition $|\hat{f} - f_i| \leq \Delta e$, where $\hat{f} = \frac{D_p}{N_p}$. is the estimated frame size.

Algorithm 2 convertToFramesWithReducedTrace$(data)$

```
1:  counter=0
2:  frames=0
3:  update(p,C,n);
4:  while counter < data do
5:      $\hat{f(p)} = \frac{D_p - C}{N_p - n}$
6:      counter = counter + $\hat{f(p)}$
7:      frames = frame + 1
8:      i += 1/fps
9:      update(p,C,n)
10: end while
11: return (frames)
```

1) Client Algorithm for reduced trace: The adaptation algorithm executed by a client converts the amount of data predicted to be received into the number of frames. (Function $\xi()$ in Equation 7 and pseudocode $convert\_to\_frames()$ in Algorithm-1) The number of frames $n_p(d)$ for d bytes of data belonging to portion p is calculated as:

$$n_p(d) = \lfloor d/\hat{f} \rfloor \tag{8}$$

As more data is received in portion p, the estimated frame size can be recomputed for higher accuracy as:

$$\hat{f} = \frac{D_p - C}{N_p - n} \tag{9}$$

where C and n are the amount of data and the number of frames already received in portion p, respectively. A modified version of the $convertToFrames$ algorithm is shown in Algorithm-2. The function $update(p, C, n)$ updates the total amount of data, C, and total number of frames, n, received in portion p. We refer to the algorithms in Algorithms 1 and 2 as the *prediction algorithms.*

2) Evaluation and Results: The underflow curve of *Star Wars* is approximated into 140 linear regions, thus reducing the control overhead by 93.4%. The experiment described in scenario-1 of Section III-F is repeated with the client using the reduced trace. Figure 16 shows the sending rate chosen by the adaptation algorithm with the full trace and compressed trace. The rate increase steps in the reduced trace case is seen to push more amount of extra data (0.81 Mb) into the network when compared to the full trace case (0.62 Mb). So the rate

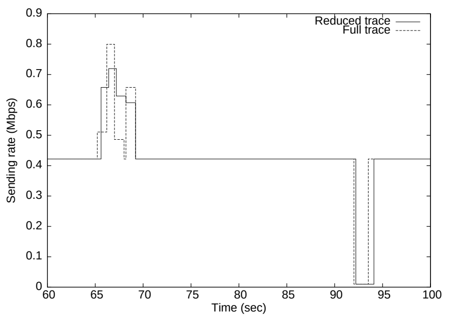

Fig. 16. Scenario-1 with reduced trace: Sending Rate (D-Rate)

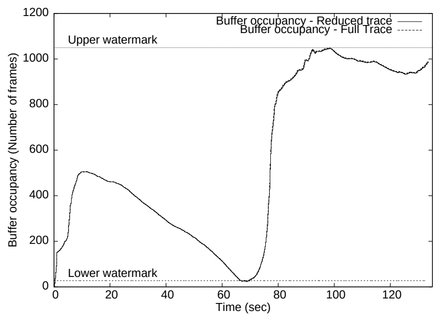

Fig. 17. Scenario-1 with reduced trace: Buffer Occupancy (D-Rate)

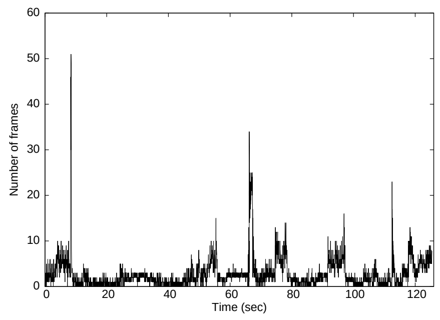

Fig. 18. Scenario-3 with reduced trace: Absolute error in the output of the *prediction algorithm* (Algorithm-2) for reduced trace of *The Firm*

decrease is applied for a longer duration (1.9 sec). Figure 17 shows the corresponding buffer occupancy for each case.

We repeat the experiment described in scenario-3 of Section III-F with a reduced trace of *The Firm*. The compressed trace contains 730 (N_p, D_p) pairs and this reduces the control overhead by 61%.

The *prediction algorithm* uses the underflow curve and the measured arrival rate to estimate the buffer occupancy after 2rtt.(Refer to Algorithms 1 and 2) This estimate drives the adaptation process as described in Section III-D . Figure 18 shows the effect of the reduced trace on the accuracy of the prediction algorithm. The y-axis shows the absolute difference between the outputs of the prediction algorithm while using the complete trace and the approximated trace. The adaptation algorithm is most vulnerable when there is a significant approximation error, in the output of the prediction algorithm, coupled with an adaptation step. We examine this condition in the current experiment. Figure 19 shows the sending rates chosen by the D-Rate scheme for the full trace and reduced trace cases. We see that a higher sending rate is chosen during the rate increase step while using the compressed trace. This leads to overall higher buffer occupancy as shown in Figure 20. The error in the chosen sending rate is introduced due to the error in the output of the prediction algorithm.

Thus, with these experiments, we conclude that it is possible to reduce the amount of control information significantly without adversely affecting the performance of the adaptation algorithm. However, the chosen per-frame error Δe play a vital role in the overall error in the adaptation decisions.

B. Delivery of Control Data

The *prediction algorithm* (shown in Algorithm-2) uses the approximated underflow curve (set of (N_p, D_p) pairs) to convert a predicted amount of data into corresponding number of frames. This prediction drives the decisions made by the adaptation algorithm as discussed in Section III-D.

A (N_p, D_p) pair is required at the client at time, t_{req}, when the *prediction algorithm* predicts to receive one or more frames in portion p. This time $t_{req}(p)$ is called the *trace requirement deadline* of portion p. The fundamental aim of the trace delivery algorithm is to deliver all (N_p, D_p) pairs before their respective *trace requirement deadlines*.

$$\forall p\ , 1 \leq p \leq n : T(p) < t_{req}(p) \qquad (10)$$

where n is the total number of portions and $T(p)$ is the time at which (N_p, D_p) is delivered to the client.

Sending the entire control data before playback will solve the problem of timely delivery but increases the startup delay. For example, sending the control information (2.8799 Mb) for the movie *Jurassic Park*, before playback, on a 300Kbps connection increases the startup delay by 9.5997 seconds. This delay when compounded with the buffering delay associated with the actual video, degrades the user perceived quality. So the delivery scheme should not cause significant additional startup delay.

We list out below essential properties of the delivery algorithm:

i) Deliver control data of all portions before their respective trace requirement deadlines: The primary aim of the delivery algorithm is to ensure that control information of a portion is present at the client before its trace requirement deadline as shown in Equation 10.

ii) Reduce the startup delay due to control data delivery: The example shown in the beginning of this section clearly shows that additional startup delay due to control data delivery should be reduced.

iii) Dynamically adapt to the network conditions: The client predicts the amount of data that it would receive in 2rtt. This predicted amount of data is called the *look-ahead window* and is a function of rtt and arrival rate. Hence the trace requirement deadlines, of all portions, are indirectly dependent on the network status (arrival rate, rtt).

iv) Controlled bandwidth usage: The amount of bandwidth required to deliver the metadata should not be very high.

C. **DART** - **D***ynamic scheduling* **A***lgorithm for* **R***educed* **T***race delivery*

The fundamental principle behind our approach is to send a small portion of the control data ahead of playback such that it does not significantly increase the startup delay and deliver the rest along with the video. So we need a *scheduling* algorithm that will plan interleaved delivery of the control data and the media data. We assume that a very small percentage of the server's outgoing bandwidth is available to stream the control information. We call this bandwidth as the *Control Bandwidth.* and denote it by $R_{control}$. $R_{control}$ is dynamically varied to suit the network and client conditions.

After delivering the startup video frames, the server uses the entire bandwidth for a very small duration to deliver control

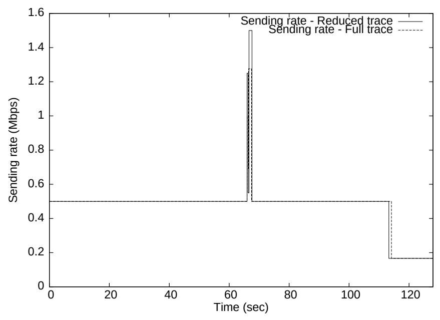

Fig. 19. Scenario-3 with reduced trace: Sending Rate (D-Rate)

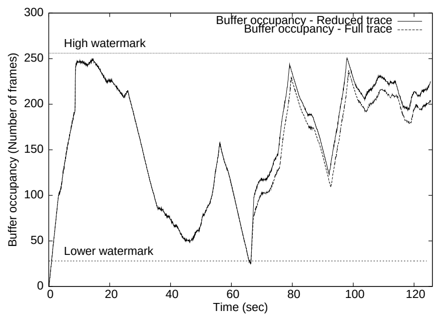

Fig. 20. Scenario-3 with reduced trace: Buffer Occupancy (D-Rate)

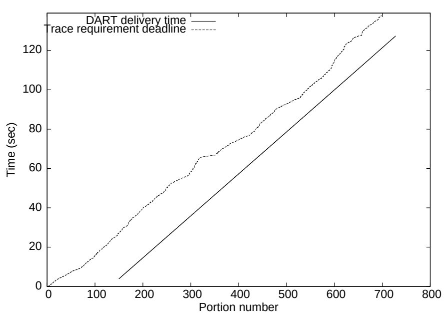

Fig. 21. Scenario-3 with DART and reduced trace ($R_{control}$ = 0.59 Kbps)

data. The remaining control data is sent, in a network-aware fashion, as described in the following paragraphs. The server keeps sending the control data at rate $R_{control}$. The client periodically sends a *status report* to the server. A *status report* consists of the following three fields :

- Current size of the look-ahead window ($2.rtt.a_{cur}$).
- Portion number p_{cur} whose control information is currently being used by the adaptation algorithm.
- Portion number p_{latest} whose control information was received last.

The client also sends a *status report* to the server, if the look-ahead window changes by more than 20% since the last report. The server, on receipt of a *status report,* checks for the following conditions and alters the transmission rate of the control stream ($R_{control}$) accordingly, if necessary:

Condition-1: If the look-ahead window has changed by more than 20% since last report, then the transmission rate of the control data is increased/decreased proportional to the increase/decrease in the size of the look-ahead window. The rate control step is required to ensure that the amount of control data delivered to the client should be commensurate with the amount of media data delivered.

Condition-2: If the difference between p_{cur} and p_{latest} is less than Δp portions then, the transmission rate of the control data is increased by the deficit amount, for Δt seconds. (Values of the parameters Δp and Δt control the aggressiveness of the scheme.) This step ensures a cushion (for control data) to counter immediate effects of the rate increase step (causing increase in arrival rate) and rapid changes in network conditions.

D. Evaluation and results

In this section, we present ns2 simulation results to illustrate the behavior of the DART scheme. The experiments described in Section IV-A.2 are repeated with DART being used to deliver the reduced trace to the client.

The first experiment deals with delivering the compressed trace of the video *Star Wars*. The approximated trace of this video has 140 portions (2240 bytes of control data). So it is seen that the entire control information can be delivered to the client before start of playback with neglible increase in startup delay. However such early delivery is not possible in all cases, as demonstrated in the next experiment.

In the second experiment, the reduced trace of *The Firm* is delivered using DART. The total amount of control information is 11680 bytes (730 portions). The startup delay is increased by 1.2% (0.03716 seconds) and video bandwidth (0.5 Mbps) is used to deliver the control information. In this time period, 145 control points are delivered to the client and and the remaining 585 control points are delivered synchronously with the video. The simulation results prove that DART possesses the essential properties as listed before

We, first, set the base value of $R_{control}$ as 0.59 Kbps. Next, we perform the same experiment with a smaller value of $R_{control} = 0.53$ Kbps. This is to demonstrate how DART increases the sending rate of the control stream, temporarily, to maintain a cushion in the amount of control data present at the client. Both these experiments illustrate the network adaptive property of DART.

a) $R_{control}$ = 0.59 Kbps: Figure 21 shows the t_{req} for each portion in the trace and also time at which control information about each portion is delivered. We can see that the (N_p, D_p) pair of all portions is delivered on time. Figure 22 shows the bandwidth usage for the control data stream. The value of $R_{control}$ increases from 0.59 Kbps to 0.594 Kbps due to increase in the size of the look-ahead window. It settles at 0.594 Kbps because the look-ahead window does not change by a amount greater than 20%. There is an quick increase in $R_{control}$ between 60s and 80s due to the rate increase step (Refer Figure 19). The quality drop between 100s and 120s causes the consumed bandwidth to drop to a smaller value. Both the rate increase and the quality drop steps affect the arrival rate which in turn affects the look-ahead window. Large variations in the look-ahead window are captured by the value of the control bandwidth illustrating the adaptivity to network conditions.

b) $R_{control}$ = 0.53 Kbps: In this experiment, the value of $R_{control}$ is chosen lower than the previous case. At time t = 97s, the difference between p_{cur} (portion whose control information is currently being used by the adaptation algorithm) and p_{latest} (portion whose control information was received last) falls below Δp which is chosen as 10.(Refer Figure 23) So a temporary rate increase is employed to maintain the occupancy of the control buffer above the minimum threshold. Δt was chosen 5 seconds and Figure 24 shows the rate increase step.

V. Related Work

In this section, we present a overview of relevant strategies for network-aware demand adaptation and metadata delivery.

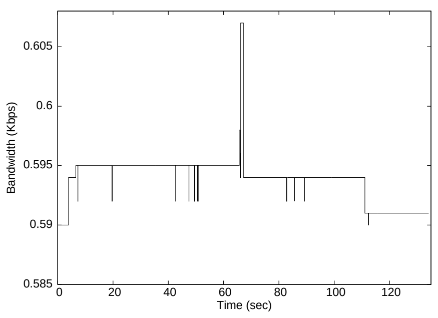

Fig. 22. Scenario-3 with DART and reduced trace: Time (vs) Bandwidth consumed for control stream ($R_{control}$ = 0.59 Kbps)

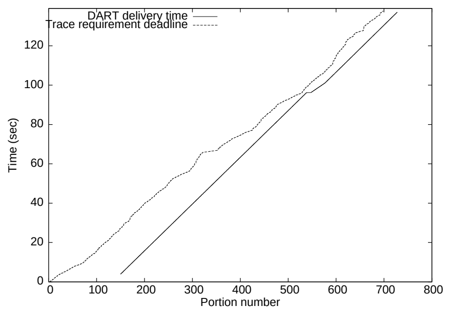

Fig. 23. Scenario-3 with DART and reduced trace, $R_{control}$ = 0.53 Kbps

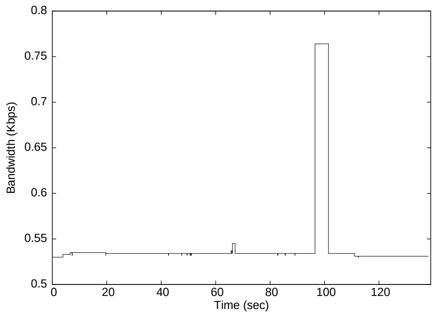

Fig. 24. Scenario-3 with DART and reduced trace: Time (vs) Bandwidth consumed for control stream ($R_{control}$ = 0.53 Kbps)

A. Network aware adaptation techniques

In a comprehensive study on video streaming, [18] identifies *application-layer QoS control* or network-aware adaptation as one of the six key areas. As explained in Section I, any network-aware adaptation strategy involves two steps. The first step evaluates the status of the network and decides if a rate change is required. The second step actually carries out the rate change at the server. Network evaluation and rate estimation techniques can be classified into three broad categories as in [18], [19]: receiver driven, transcoder driven and sender driven. This classification is based on based on where the adaptation happens. Rate change of the video is affected by either scalable video coding [20], transcoding [21] or a combination of both [22].

1) Receiver driven: The RLM scheme [9] is a receiver driven technique wherein the client modifies its bandwidth demands based on network status. The server transmits a base layer and multiple enhancement layers of a video over unique multicast channels. The base layer represents the minimum quality required to view the video. Clients subscribe to the base layer multicast and depending on the available bandwidth subscribe to one or more enhancement layers. When a client experiences congestion, it drops a layer and when spare capacity is available, adds a layer. RLM uses a *join experiment* similar to our *bandwidth probing* phase to decide when to increase the video quality. However the disadvantage of RLM is that the adaptation step of adding a layer makes it unfair to competing TCP connections. This is because by adding a layer, a receiver increases its bandwidth demands by a fairly large amount when compared to a peer TCP connection which increases its bandwidth demands incrementally. This issue is addressed in ThinStreams [10]. A video layer is split into several fixed bandwidth thinstreams. So the bandwidth overhead of a join experiment is considerably lower. Also in ThinStreams, instead of using loss as a measure of network overload, the difference between expected and measured throughput is used to sense the network load. We use a similar approach in determining the *network response index*, β.

2) Transcoder Driven: In transcoder based schemes, gateways are placed at strategic points in the network to vary the quality based on the network status in each region. [23] presents a adaptation framework with strategically placed *smart relays* in the network. [24] presents another approach of adaptive quality adjustment by dynamically transcoding a pre-encoded version.

3) Sender Driven: The Loss delay based adjustment algorithm(LDA) [13] is a sender driven scheme, wherein the sender uses RTCP feedback reports from the receiver to learn network status and consequently adapt the outgoing bandwidth for that client. The receiver also estimates the bottleneck bandwidth using the packet pair approach and includes it in its RTCP report. Like LDA, the D-Rate scheme also uses the packet pair approach.

In the LDA scheme, a sending rate change is fed back into the encoder thereby varying the quality. So, when there is no congestion in the network, the quality that user perceives is periodically stepped up and when there is congestion it is brought down to a comparatively very low level (due to multiplicative decrease). Hence the perceived quality fluctuates (sawtooth waveform) which is not desirable. On the other hand, the D-Rate scheme tries to avoid a quality drop by manipulating the sending rate and only as the worst case reduces quality. Even a quality increase is done only after building a considerable *no congestion state* history leading to a more stable perceived quality. In LDA, RED (Random Early Detection) is used at the routers to provide a early indication of congestion by dropping packets. In other words, network congestion is detected by a perceived loss in quality caused due to lost packets. In the D-Rate scheme, we use a more fine granular approach by adopting MECN to track network backlog. Since the sender is aware of the queue buildup, the adaptation steps start earlier than in the LDA scheme. Therefore the sender has a better chance to avoid lost packets even if it calls for reducing the quality at a earlier stage.

[25] proposes an adaptation technique that performs rate adaptation independently of quality adaptation. Quality adjustment is done over a longer time scale based on estimation of available bandwidth. Sending rate adaptation, on the other hand, is done as a congestion control mechanism over shorter durations. D-Rate shares its underlying principle of decoupling quality and sending-rate with these methods.

In the Frame Rate Adaptation scheme [12], measures the available bandwidth and its buffer occupancy to decide the the list of frames that the server will transmit. Frames are excluded on event of packet loss and/or imminent buffer underflow. Our adaptation scheme gathers more information about the network and manipulates the sending rate (without changing the quality) and only as the final step alters the quality.

B. Control data reduction and delivery schemes

[26] presents a method to approximate the underflow curve. The authors use MCBA, a bandwidth smoothing algorithm, to find the smallest set of piece-wise linear segments that fit between the underflow curve and the same curve shifted upward by Δe, the allowable error. Thus, $(\hat{f} - f_i) \leq \Delta e$. These linear segments form an approximation of the underflow curve. The choice of a lower and upper bound on the frame size estimate is a result of the fact that this approximation is used specifically by bandwidth smoothing algorithms. We position the end points of the piecewise linear segments on the underflow curve itself. This offers the advantage that for the same distance between upper and lower bound equal to Δe our method reduces the per-frame error by half, i.e., $|\hat{f} - f_i| \leq \frac{\Delta e}{2}$.

In [27], the authors present a technique to synchronously deliver MPEG-7 metadata with the video. RTP is used to deliver video, metadata and to synchronize them. The proposed technique DART can be used to deliver any form of time-sensitive metadata sychronously to the client. DART has the additional advantages of being low on control overhead, network-bandwidth sensitive and low startup overhead.

VI. Conclusions

In this paper, we present a network-aware video streaming technique that adapts sending rate of the video to match time-varing network characterstics, client data requirements and buffer occupancy. We also present a technique to efficiently deliver metadata, synchronized with the media stream. Simulations show that the proposed techniques can efficiently adapt to changes in the network to provide better perceived QoS than schemes that consider sending rate synonymously with quality. One key area of future work is improving the TCP-friendliness aspect of D-Rate.

References

[1] B. Girod, J. Chakareski, M. Kalman, Y. Liang, E. Setton, and R. Zhang, "Ad- vances in network-adaptive video streaming," 2002. [Online]. Available: citeseer.ist.psu.edu/girod02advances.html

[2] W.H.Ma and D. Du, "Design a progressive video caching policy for video proxy servers," *IEEE Transactions on Multimedia*, vol. 6, no. 4, pp. 599–610, 2004.

[3] J. Bolliger and T. Gross, "A framework-based approach to the development of network-aware applications," *Software Engineering*, vol. 24, no. 5, pp. 376–390, 1998.

[4] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification," 1998. [Online]. Available: citeseer.ist.psu.edu/ramakrishnan99proposal.html

[5] E. Kusmierek and D. H. Du, "Streaming video delivery over Internet with adaptive End-to-End QoS," *Journal of Systems and Software, Special issue on Adaptive Multimedia Computing*, July 2004.

[6] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *INFOCOM*, 2001.

[7] A. Durresi, M. Sridharan, C. Liu, M. Goyal, and R. Jain, "Traffic management using MECN," 2001.

[8] L. S. Brakmo and L. L. Peterson, "TCP vegas: End to end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, 1995. [Online]. Available: citeseer.ist.psu.edu/brakmo95tcp.html

[9] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*. ACM Press, 1996, pp. 117–130.

[10] L. Wu, R. Sharma, and B. Smith, "Thin streams: An architecture for multicasting layered video." in *Proceedings of NOSSDAV*, 1997.

[11] J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control," *IEEE Network*, vol. 15, no. 3, pp. 28–37, 2001. [Online]. Available: citeseer.ist.psu.edu/widmer01survey.html

[12] C.Fung and S.Liew, "End-to-end frame-rate adaptive streaming of video data," 1999.

[13] D. Sisalem and H. Schulzrinne, "The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme," in *Proceedings of NOSSDAV*, Cambridge, UK., 1998. [Online]. Available: citeseer.ist.psu.edu/sisalem98lossdelay.html

[14] S. Jacobs and A. Eleftheriadis, "Streaming video using dynamic rate shaping and TCP flow control," *Journal of Visual Communication and Image Representation*, vol. 9, no. 3, pp. 211–222, 1998. [Online]. Available: citeseer.ist.psu.edu/jacobs98streaming.html

[15] "Network simulator (ns2), Univ. of California, Berkeley," 1997.

[16] F. H. Fitzek, M. Zorzi, P. Seeling, and M. Reisslein, "Video and audio trace files of pre-encoded video content for network performance measurements," Universita di Ferrara, Arizona State University, Tech. Rep.

[17] J. McManus and K. Ross, "Video-on-demand over ATM: Constant-rate transmission and transport," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 1087–1098, 1996.

[18] D. Wu, Y. Hou, W. Zhu, Y. Zhang, and J. Peha, "Streaming video over the internet: Approaches and directions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 1, pp. 1–20, Feb. 2001. [Online]. Available: citeseer.ist.psu.edu/wu01streaming.html

[19] X. Wang and H. Schulzrinne, "Comparison of adaptive internet multimedia applications." [Online]. Available: citeseer.ist.psu.edu/331270.html

[20] P. de Cuetos and K. W. Ross, "Adaptive rate control for streaming stored fine-grained scalable video," in *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*. New York, NY, USA: ACM Press, 2002, pp. 3–12.

[21] P. Assuncao and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 8, no. 8, pp. 923–967, 1998.

[22] A. Raghuveer, N. Kang, and D. H. Du, "Techniques for efficient streaming of layered video in heteregenous client environments," in *GLOBECOM*, 2005.

[23] Y. Wu, A. Vetro, H. Sun, and S.-Y. Kung, "Intelligent multi-hop video communications," *Lecture Notes in Computer Science*, 2001.

[24] Z. Lei and N. D. Georganas, "Rate adaptation transcoding for video streaming over wireless channels," in *Proceedings of International Conference on Multimedia and Expo*, 2003.

[25] R. Rejaie, M. Handley, and D. Estrin, "Quality adaptation for congestion controlled video playback over the internet," in *SIGCOMM*, 1999, pp. 189–200. [Online]. Available: citeseer.ist.psu.edu/rejaie99quality.html

[26] W.-C. Feng, C. C. Lam, and M. Liu, "Movie approximation technique for the implementation of fast bandwidth-smoothing algorithms," in *Proc. SPIE Vol. 3310, p. 96-110, Multimedia Computing and Networking*, Dec. 1997, pp. 96–110.

[27] J. Lim and M. Kim, "Synchronized delivery of MPEG-7 Metadata with video contents over RTP," in *International Packet Video Workshop*, 2003.