Selective Frame Discard for Interactive Video

Kameswari Chebrolu, Ramesh R. Rao
Department of Electrical and Computer Engineering
University of California at San Diego
Email: chebrolu@cwc.ucsd.edu, rrao@ucsd.edu

***Abstract*— A mobile terminal equipped with multiple interfaces can achieve a much higher bandwidth by aggregating the bandwidth offered by the individual networks. This helps support demanding applications like interactive video. Often, in spite of bandwidth aggregation, the available bandwidth may be too small to avoid frame loss altogether. Under these circumstances, it may be necessary to selectively discard frames to minimize the effect of their loss on the overall video quality. In this paper, we consider different frame discard algorithms and study their performance in the presence of multiple interfaces. We show through trace driven simulations that attempting to transmit every frame results in severe performance degradation. In particular we show that our proposed algorithm *MC-Drop* outperforms other algorithms in terms of suitably defined metrics that capture overall video quality.**

I. Introduction

In recent years, explosive growth of the Internet has been a major driving force in the deployment of a variety of wireless technologies. Examples include GPRS, EDGE, CDMA2000, HDR, UMTS, etc. Efforts are now underway to integrate these networks with the Internet to support a rich variety of data applications. Some of the applications we can expect to see would be bandwidth intensive applications like large file transfers, video conferencing etc. These applications often require high bandwidth; interactive video also has stringent delay requirements.

So far the focus has been on using only a single wireless interface at a time and several research challenges related to such use have been explored. Given the scarcity of bandwidth in the wireless domain, it is often the case that no single interface can support these bandwidth intensive applications. Even though the 3G radio access technology UMTS is said to provide up to 2Mbps, this is under ideal conditions, where the channel is dedicated to a single user and the user has excellent radio conditions. In practice, the user is likely to get only a few hundred kbps. A medium quality 128kbps Variable Bit Rate (VBR) *interactive video*, due to huge difference between peak to mean rates (> 15) and stringent QoS (one way delay less than 200ms), often needs more than 400kbps capacity to achieve adequate quality of reception.

When coverage areas of the different wireless technologies overlap, there is no need to restrict oneself to a single interface. The simultaneous use of multiple interfaces permits bandwidth aggregation, thereby allowing support for demanding applications that need high bandwidths. In this paper, we focus our attention on aggregating bandwidth for *interactive video* with strict QoS requirements (one way latency $< 200ms$).

The use of multiple paths, one corresponding to each wireless interface in use, is not without problems. The varying characteristics of the different paths results in packet reordering, which in turn causes excess delay. Stored video applications normally employ smoothing buffers and can tolerate large delays. However, for interactive applications, if care is not taken to minimize delay introduced by reordering, it is very difficult to meet the QoS requirements.

In our previous work [1], [2] we have proposed a network-layer architecture based on an extension to Mobile IP [3], as well as a scheduling algorithm Earliest Delivery Path First (EDPF) that distributes packets onto the multiple interfaces with the objective of minimizing reordering. In that work [1], [2], we have evaluated the performance of bandwidth aggregation for interactive applications when adequate bandwidth is reserved on all the interfaces that satisfies application QoS.

In this paper, we consider the case where adequate bandwidth that satisfies user QoS cannot be reserved on the interfaces. Considerable research has gone into coming up with feasible transmission schedules when network bandwidth (or client buffering capacity) is constrained [4]. In our set up, no such feasible transmission schedule that avoids frame loss exists – available bandwidth is too small. One is then left with two choices. The first choice is to choose an available low quality, low rate video - which may or may not satisfy user's QoS depending on the reserved bandwidth. The second choice is to choose a better quality high rate video but drop frames selectively to minimize their impact on the overall quality of the video. Our overall approach falls under the second choice of selective frame discard.

Rate adaptation [5] at the video server based on fluctuating bandwidth is a well researched area. However, schemes in this domain do not quite fit in our setup, as we have no resource fluctuation. In our setup, once certain bandwidth is reserved, we assume that this bandwidth is provided for the duration of the session by the wireless infrastructure[1]. Normally, the video is encoded at a few standard target rates (64, 128, or 256 kbps etc). The client then has the flexibility of choosing the one that best fits its bandwidth requirement and sticks with the same rate for the entire session.

Frame discard based on time stamps, or priority information of the frames has also been considered in prior work [6], [7]. Most schemes in this domain either time stamp the frames,

[1]Rate adaptation is useful during handoffs and change in radio conditions. However between these events, the available bandwidth is assumed fixed. And this bandwidth is assumed to be too small to avoid frame loss.

whereby intermediate routers drop frames that are unable to meet their deadlines or drop low priority frames in the event of network congestion (increased queue sizes). In [8], the authors propose a selective frame discard algorithm at the video server for stored video applications. The algorithm attempts to discard frames by minimizing a QoS based cost function. However the scheme assumes full knowledge of the video characteristics (frame sizes) and does not fit in our setup since we are dealing with interactive video - frames are generated on the fly.

Our Mobile-IP based network-layer architecture that enables multiple interfaces was designed with the objective of introducing minimal changes to the infrastructure for ease of deployment. Among past work that has considered frame discard algorithms, most schemes either do not fit in with our architecture (they need time stamping, clock synchronization, additional functionality at intermediate routers etc) or do not perform well in our particular setup of multiple interfaces (as will be shown later). Unlike single interface use, when using multiple interfaces, it is more difficult to meet deadlines of frames due to the packet reordering. So, we propose a new frame discard policy - Min Cost Drop (*MC-Drop*) which is in line with the goals of our architecture – we have the Home Agent (HA, an intermediate entity in Mobile IP) discard select frames of the video stream, in addition to the EDPF scheduling onto the multiple paths. In this scheme, we require modifications only at the HA and the end client, and we assume no support from any other part of the network, or the server. In *MC-Drop*, when a frame arrives at the HA: (1) It is determined whether it should be forwarded (*MC-Drop*). (2) If so onto what interface (EDPF). We rely on a crucial aspect of video stream - Group of Pattern (GOP) correlation to decide whether to send or drop a frame. The decision to drop a frame is based on the impact the frame drop has on meeting future frame deadlines and hence on overall quality of the video. Frames with high priority are less likely to be dropped.

We evaluate the performance of *MC-Drop* through simulations carried out using MPEG-4 video and Internet path delay traces. We observe that employing no form of frame discard results in degraded performance. When reserved bandwidth is small, our frame discard algorithm *MC-Drop* outperforms by a large margin other considered approaches.

The rest of the paper is organized as follows. In section II, we describe briefly the architecture and the scheduling algorithm EDPF. In section III, we present the different frame discard algorithms. Section IV provides the experimental methodology and the results are discussed in section V. We finally conclude in section VI.

II. System Architecture

The architecture needs to support multiple communication paths, one corresponding to each interface in use. In this section, we explain briefly the important components that make up our network layer architecture. Our choice for considering a network layer approach as opposed to transport/application

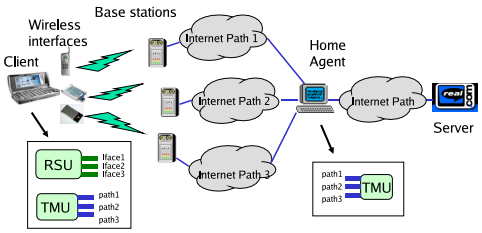

Fig. 1. Architecture to support multiple communication paths

layer solution was to introduce minimal changes in the existing infrastructure thus providing application transparency.

Mobile IP [3] is an IETF standard for mobility support in IP networks. One of the features of Mobile IP is *Simultaneous Binding*, where a mobile node can register more than one care-of address (each corresponding to an interface) at the Home Agent (HA). In this case, the HA duplicates the datagrams and tunnels them to each care-of address. Unlike blind duplication, what we desire when performing bandwidth aggregation is intelligent processing and scheduling of incoming packets. Our architecture extends Mobile IP to support this feature. Fig. 1 shows the details of our architecture.

The client (equivalently, the MH) is connected to the Internet via multiple Radio Access Networks (RANs). A *Radio Access Network Selection Unit (RSU)* is located at the MH and is responsible for selecting suitable set of RANs based on cost and services offered. A *Traffic Management Unit (TMU)* is located in the MH and the HA. When the server streams data to the HA, the TMU processes the incoming packets and tunnels them to the appropriate care-of address of the MH.

In the context of the overall architecture presented above, a crucial aspect that dictates the performance of the application is the scheduling algorithm (EDPF) residing on the TMU at the HA. This algorithm EDPF, splits the traffic onto the different paths with the objective of minimizing reordering and the delay associated with it. The overall idea behind EDPF is to (1) take into consideration the overall path characteristics between the HA and the MH – delay, as well as the wireless bandwidth, and (2) schedule packets on the path which will deliver the packet at the earliest to the MH. The network between the HA and the MH can be simplified as shown in Fig. 2. Each path l (between the HA and the MH) can be associated with three quantities: (1) D_l, the one-way wireline delay associated with the path (between the HA and Base Station - BS), (2) B_l, the reserved bandwidth at the BS, and (3) a variable A_l, which is the time the wireless channel becomes available for the next transmission at the BS. If we denote by a_i, the arrival instance of the i^{th} packet (at the HA) and by L_i, the size of the packet, this packet when scheduled on path l would arrive (estimated) at the MH at d_i^l.

$$d_i^l = MAX(a_i + D_l, A_l) + L_i/B_l \quad (1)$$

The first component computes the time at which transmission can begin at the BS, and the second component computes the packet transmission time. EDPF schedules the packet on the

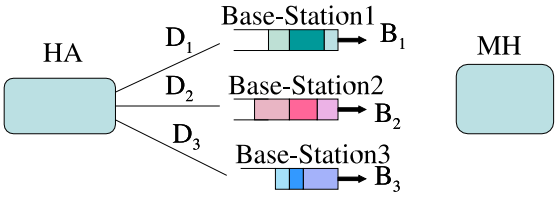

Fig. 2. A Simplified View of the Network between HA and MH

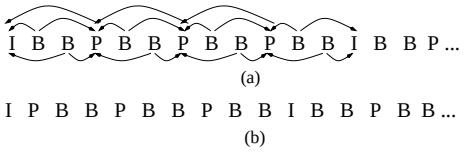

Fig. 3. Dependency structure of MPEG for GOP of size 12: (a) Playback; (b) Transmission Order

path p where, $p = \{l : d_i^l \leq d_i^m, 1 \leq m \leq N\}$, N being the number of interfaces. That is, the path with the earliest delivery time. EDPF then updates A_p to d_i^p. EDPF tracks the queues at each of the base-stations through the A_l variable. By tracking the queues at the base-stations and taking it into account while scheduling packets, EDPF ensures that it uses all the available path bandwidths, while achieving minimal packet reordering. Further details along with some useful properties of this algorithm can be found in [1], [2]. The explanation so far focused on down-link transmission where the MH acts as a sink. The same algorithms can be used for the up-link case where the MH acts as a server.

In this paper, we propose a frame-discard policy to be implemented at the HA in addition to the EDPF algorithm to improve performance in the event of constrained network bandwidth. That is, prior to applying EDPF, the HA first determines if the packet is to be dropped, according to the frame-discard policy in place. We next present various frame discard algorithms that make this decision.

III. Selective Frame Discard Algorithms

The decision to drop or send a frame depends heavily on the structure of the video sequence. In this paper, our focus is on the MPEG standard – the same ideas can be extended to other standards with inter-frame dependencies.

The MPEG standard encodes the information of a scene into multiple Group of Pictures (GOP) consisting of three different types of frames - *I*, *P*, and *B*. *I* frames are coded autonomously, while *P* frames are coded in reference to the most recent *I* or *P* frame. *B* frames are coded using the closest previous and future *I* or *P* frames. Fig 3 illustrates the dependency structure for a GOP of size 12. Because of these interdependencies, the loss of an *I* frame results in the loss of the entire GOP and has a much worse impact on the video quality than the loss of say a *B* frame that has no dependent frames. This frame priority information must be taken into account when dropping frames to minimize the effect of losses on the quality of the video. Note that the transmission order of frames differs from the playback order (Fig 3). *I* or *P* frames sent earlier are displayed later than *B* frames. This is so as to help decode dependent frames. This gives *I/P* frames more leeway with respect to delay than *B* frames.

When network bandwidth is constrained, a naive approach that attempts to transmit every frame results in high loss rate due to missed playback deadlines. This policy, which we term *NO-Drop*, not only wastes scarce bandwidth by transmitting frames whose deadline cannot be met but prevents future frames from meeting their deadlines. This essentially captures the performance (video quality) one can expect to see in the absence of any frame discard policy.

Selective frame discard attempts to effectively utilize the constrained bandwidth by discarding frames whose loss has minimal effect on the overall video quality. An optimal scheme that discards frames to maximize video quality needs perfect knowledge of the frame size of the entire video sequence (not possible for interactive video) and has a complexity of $O(N2^N)$, where N is the number of video frames [8]. This makes this scheme computationally prohibitive and impractical to implement. However, it is possible to achieve comparable results with several low complexity $O(N)$ algorithms (O(1) each step). We discuss two such algorithms here.

A straight forward approach is to drop the frames that are unlikely to meet their playback deadline (similar in spirit to that proposed in [6]). Any future dependents of these dropped frames are also discarded when they arrive at the HA. We term this policy Deadline Drop (*DL-Drop*). This basic idea can be easily integrated into our scheduler EDPF without the need for time stamping and other functionalities. The scheduling algorithm EDPF, estimates the delivery time of a packet to determine the Internet path to use. This estimate can be used to determine if the packet (and hence the frame the packet is a part of[2]) is going to meet its deadline. While *DL-Drop* can achieve much better performance than *NO-Drop*, this approach does not use the priority information of the frame type. If an *I* frame misses its deadline, the entire GOP is dropped. However, had the preceding *B* frame (that was sent) dropped to ease network resources, the *I* frame and its dependents could have been saved. We now present our algorithm Min-Cost Drop (*MC-Drop*) that uses this priority information.

When a packet arrives at the HA, a decision to send or drop the packet has to be made. If sending the packet results in a future high priority frame missing its deadline, we would ideally like to drop the packet. However, it is not possible to know the future frame sizes and hence their deadlines in advance since we are dealing with interactive video - packets come one at a time at the HA. An important observation that helps overcome this drawback is that most video streams have a high degree of correlation (> 0.8) of frame sizes across GOPs, for a lag of 1-2 GOPs [9]. This observation, permits us to estimate future frame sizes in the next 1-2 GOPs based on observed frame (packet) sizes in the current/previous GOP. In

[2]The video server breaks the large frames into smaller sized packets before transmitting them onto the Internet.

MC-Drop, we maintain a window size of k GOPs. Each time a packet arrives, we estimate the delivery time of all the frames in the window based on the scheduling algorithm EDPF. The present packet is dropped only if by dropping it, it is possible to meet the deadline of a future higher priority frame in the window. In addition, we drop all packets of the frames that miss their deadline and their dependents. A window size of 1-2 GOPs is more than adequate for our purposes because any high priority frame is normally preceded by enough low priority *B* frames, that dropping these *B* frames will help meet its deadlines. Note that in *MC-Drop*, it is possible for incorrect estimation to cause some high priority frames to be dropped, but this situation is no worse than *DL-Drop*. Some low priority frames may be unnecessarily dropped in *MC-Drop*, but in general the advantages outweigh the disadvantages as will be shown in Section V.

Fundamental to both *DL-Drop* and *MC-Drop*, is the ability to estimate the delivery time of a packet by EDPF. This depends on the estimation of the one-way delay experienced by a packet (on each of the different paths) as it traverses the Internet from the HA to MH. This delay has two components: (a) Wireline delay - delay experienced by the packet between HA and Base Station (BS) serving the MH, and (b) Wireless delay - delay experienced by the packet between BS and MH (queuing and transmission delay). Both the delay components can be estimated and can be expected to be stable[3]. This is due to the following reasons. First, we assume that once bandwidth (wireless) is reserved on an interface, it is guaranteed for the entire duration of the session. This is a reasonable assumption since present wireless networks do provide QoS support (in the form of bandwidth reservation). It is in general not possible to support real-time applications without some form of QoS support from underlying infrastructure. So, delay variation if any is in the wired part and not on the wireless part. However, the wire-line link speeds are quite high and the mean wire-line delay and its variation about the mean are normally very small (few ms). Even if large ($\simeq$ 10ms), this variation is usually masked by the backlog at the base-station queue serving the mobile – we assume the wireless hop to be the bottleneck link. We did observe this behavior in our experiments [1].

As such, EDPF does not require time synchronization between the HA and the MH for this estimation, only the relative one-way delays between the multiple paths are required, and not the absolute values. However, in the DL-Drop/MC-Drop algorithms the one-way delay estimate needs to be correlated with the playback deadline at the MH. This too does not require strict time synchronization between the HA and the MH, and can be done as follows. Initially, the HA can record the expected delay of packets sent to the MH. The MH can then report to the HA as to how much ahead of (or after) its deadline the packet arrived. The HA can thus correlate the recorded delay estimate for the packet to how much ahead/after its deadline the MH perceived it to be. Since the one-way delay is expected to remain stable, this correlation can be used for future packets as well. The above process can be repeated with multiple packets to improve the estimate.

[3]We have addressed this issue in depth in [1].

IV. Experimental Methodology

We now present our methodology for evaluating the frame-discard policies presented above. Our overall approach is one of trace-based simulation. The components that make up the network topology are as shown in Fig 1.

To simulate video server behavior, we have used frame size traces of two video clips (class room lectures) representative of interactive applications from [9]. The video clip is encoded into a base layer (*I/P* frames) with a set target rate of 128Kbps and an enhancement layer (*B* frames). The server packetizes the video frames into 1000 byte packets and passes them on to the HA. The HA implements the EDPF scheduling algorithm along with the frame discard policy. The Base Stations have a link capacity equal to the reserved bandwidth. We do not simulate cross traffic as the channel is considered dedicated. The wired part of the network has high bandwidth (10Mbps) – the wireless links are assumed to be the bottlenecks. The wire-line delay experienced by the packets was introduced from traces collected by measuring round-trip times on different Internet paths [4]. The mean delay experienced by packets between the server and the HA is 15ms and that between the HA and the BSs is 22ms, making the mean one-way *wireline* delay 37ms . Note that this delay does not include queuing and transmission delay on the wireless segment.

As packets arrive at the client, they are placed in a buffer. The video display begins after a fixed delay, which we term *Startup Latency*. Once the display begins, the client displays frames consecutively every frame period (1/30 sec). If at one of these epochs, the client's buffer does not have the completed frame, the frame is considered lost. We set the *Startup Latency* at 200ms in our experiments i.e. the client begins display 200ms after the server transmits the very first packet. This is to ensure that all frames that are displayed have one way latency (total delay between server and MH) less than 200ms as required by interactive applications.

To measure the quality of video reception, we used the following performance metrics - (1) Peak Signal to Noise Ratio PSNR (in dB) of the received video sequence. For frames that could not be displayed, PSNR is assumed to be zero; (2) Glitch Duration G_d - length of consecutive frames that could not be displayed; (3) Glitch Cost G_c that captures the effect a frame loss has on the perceptual quality of video as was proposed in [8]. Every undisplayed frame *i* is associated with a cost c_i. If frame *i* belongs to a sequence of consecutive undisplayed frames, $c_i = l_i$, if frame *i* is the l_i^{th} consecutive undisplayed frame. Otherwise $c_i = 1 + 1/\sqrt{d_i}$, where d_i is its distance from the previous undisplayed frame. $G_c = \sum_1^N c_i$. This metric captures two important aspects of playback discontinuity – cost due to consecutive discard and that due to spacing between discarded frames, both of which

[4]We collected traces between hosts located at the following universities: UCSD, UCB, Duke, CMU.

are important measure of perceived quality; (4) Total number of Frames that could not be displayed - F_{loss}.

In addition to *NO-Drop*, *DL-Drop* and *MC-Drop*, we consider two other design alternatives. In the first policy, *ENH-Drop*, we only send the base layer (*I/P* frames) and drop the enhancement layer (*B* frames) totally as the reserved bandwidth is too small to accommodate both [5]. In the second policy, *LR-NoDrop*, we consider same video clip encoded at a lower target rate. This gives us a measure of the improvement in performance when employing selective frame discard on a high rate, better quality video as opposed to settling for a low rate, low quality video.

V. Experimental Results

In this section, we present the performance of the different algorithms under different scenarios. Table I lists the characteristics of the two video clips considered encoded at a rate of 128 (medium quality) and 64 kbps (low quality). The medium quality video is temporal scalable encoded - it has a base and enhancement layer. The base layer target rate (not aggregate) is 128kbps. The low quality is single layer encoded (non-scalable). The aggregate target rate is 64kbps. Refer to [10] for further details. While our focus is on understanding the performance of these algorithms when using multiple interfaces, the ideas apply for single interface use as well. We present results for multiple interfaces and explain in passing the results we obtained for single interface use.

Title (Lecture)	Dur. (min)	Rate(kbps)		PSNR(dB)		GOP Corr. lag=2
		avg	max	avg	stdev	
Medium Quality: Base Layer Target rate 128kbps						
Reisslein	30	222	1361	27.1	2.1	0.90
Gupta	30	205	1255	28.1	2.2	0.82
Low Quality: Base+Enhancement Layer Target rate 64kbps						
Reisslein	30	72	946	23.5	1.7	0.86
Gupta	30	68	1878	23.9	1.6	0.72

TABLE I

Characteristics of MPEG4 video traces

Table II shows the performance of the different frame discard algorithms when the number of interfaces considered is two. The overall reserved bandwidth is 240kbps, split among the two interfaces in the ratio 2:1. As can be seen from the table, sending frames without considering the underlying network resources (*NO-Drop*) results in severe performance degradation. The performance is even worse than just sending the base layer and dropping the enhancement layer (*ENH-Drop*). While *DL-Drop* achieves significant performance improvement over *NO-Drop*, the occasional *I* frame drops result in high glitch cost G_c. This situation is avoided mostly by *MC-Drop* and as can be seen, it performs better than *DL-Drop* for all the metrics considered. Since the bandwidth reserved is adequate to get across all frames, the low quality 64kbps encoding *LR-NoDrop* experiences no glitches. However, the PSNR is about 3dB less than *MC-Drop*. The trade off between occasional glitches in high quality video vs no glitch low quality video on overall perceptual quality of video playback is difficult to capture with the metrics considered and needs further study. The performance trend of the different algorithms when using single interface is similar to the case of multiple interfaces except that the avg PSNR is slightly higher and G_c lower. For example, G_c for *DL-Drop* and *MC-Drop* for Reisslein lecture are 8606.8 and 6320.2 respectively.

[5]In temporal encoding, a video is encoded into many sub-streams (scalable extensions). It is common place to employ a hierarchical filter to select required number of sub-streams that satisfy a given QoS.

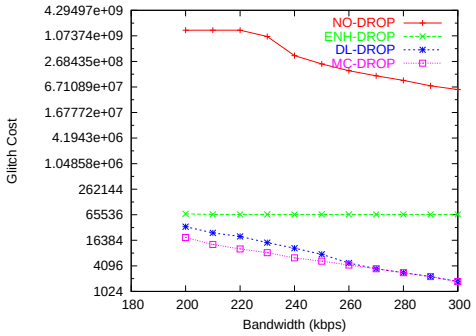

Fig. 4. Variation of Glitch Cost with Bandwidth

Algorithm	Loss (%)	G_c (10^3)	G_d (frames)		PSNR (dB)	
			avg	max	avg	stdev
Reisslein: avg PSNR = 27.16						
NO-Drop	64.7	368023	155	26024	9.8	13.2
ENH-Drop	66.7	66.7	2	2	25.5	2.87
LR-NoDrop	0	0	0	0	23.5	1.71
DL-Drop	7.38	10.658	1.7	12	26.1	5.73
MC-Drop	6.6	6.339	1.4	12	26.4	4.92
Gupta: avg PSNR = 28.15						
NO-Drop	35.3	50394	115	8198	18.9	13.9
ENH-Drop	66.7	66.7	2	2	26.8	3.15
LR-NoDrop	0	0	0	0	23.9	1.63
DL-Drop	4.32	4.87	1.7	12	27.5	4.81
MC-Drop	4.15	3.87	1.5	12	27.6	4.56

TABLE II

Multiple Interfaces, Bandwidth=240kbps, Split=2:1

Fig 4 presents the impact of different reserved bandwidths on the video quality for Reisslein lecture as captured by G_c. The bandwidth is varied from 200kbps to 300kbps. Note that the y axis is in log scale. Both *DL-Drop* and *MC-Drop* outperform *NO-Drop* by a large margin. G_c stays constant for *ENH-Drop* as 200kbps is enough to get the base layer across without any losses. As the bandwidth increases, the difference between *DL-Drop* and *MC-Drop* becomes small - 15,247 at 200kbps to 503 at 260kbps and 0 at 270kbps. This is because as bandwidth increases, the high priority frames *I/P* have less trouble meeting their deadline than *B* frames because they have more delay leeway as explained earlier in sec III. So, when the reserved bandwidths is small, *MC-Drop* can bring about significant benefits. For the case of single interface, the performance trend is similar except that the difference in G_c between the algorithms is smaller. For example, the difference

Algorithm	1:1	3:1	5:1	1:1:1	3:2:1	5:3:1
Glitch Cost						
DL-Drop	6941	8953	14644	11735	13496	18427
MC-Drop	6667	7629	8439	7609	9113	10295
PSNR (dB)						
DL-Drop	26.39	26.19	25.64	25.86	25.56	25.35
MC-Drop	26.41	26.27	26.04	26.11	25.84	25.89

TABLE III

PSNR AND GLITCH COST FOR VARIOUS SPLITS, BANDWIDTH = 240KBPS

between *DL-Drop* and *MC-Drop* is 12,753 at 200kbps, 211 at 260kbps and 0 at 270kbps.

In order to capture the sensitivity of the system to the number of interfaces and asymmetry across them, we computed G_c and PSNR for different bandwidth splits across the different interfaces. Table III shows the performance of *DL-Drop* and *MC-Drop* for Reisslein lecture when the aggregate bandwidth is set at 240kbps [6]. G_c of *NO-Drop* is too high (order of 10^8) for all the cases. PSNR is too low - around 9dB. For *ENH-Drop*, G_c is same as before 66,723. The higher the number of interfaces and asymmetry, the more the reordering and hence delay and missed deadlines. As can be seen in Table III, G_c which captures frame loss increases as the number of interfaces and asymmetry increases. However, PSNR does not follow the same trend in some cases. For *MC-Drop*, it mostly decreases but not always (e.g 3:2:1 vs 5:3:1) but the drop is normally very minor. This is mainly to do with inter-frame dependencies. It may not always be possible to display a *B* frame even if it arrives on time if its reference *I/P* frames do not arrive before its display time (the *I/P* frames may arrive before their respective display times) [7]. But as can be seen *MC-Drop* consistently performs better than *DL-Drop* for all the cases considered.

[6] A 3:1 split on two interfaces corresponds to a bandwidth of 180kbps and 60kbps.

[7] This situation can be avoided by dropping the *B* frame at the HA if it is estimated that its reference frames will not arrive before its display time. We have not implemented this feature yet.

VI. CONCLUSIONS

In this paper, we have considered the problem of frame discard for interactive video over multiple interfaces when network bandwidth is limited. We have evaluated the performance of different frame discard mechanisms using video and delay traces. We observe that our algorithm *MC-Drop* outperforms by large margin a policy that discards no frames (*NO-Drop*). When reserved bandwidths are small, it also performs much better than *DL-Drop*.

REFERENCES

[1] K. Chebrolu and R. Rao, "Bandwidth Aggregation for Real-Time Applications in Heterogeneous Wireless Networks," submitted for publication.

[2] K. Chebrolu and R. Rao, "Communication using Multiple Interfaces," in *IEEE WCNC*, 2002.

[3] C. E. Perkins, "Mobile IP," *IEEE Communication Magazine*, vol. 35, no. 5, pp. 84–99, May 1997.

[4] W. C. Feng and J. Rexford, "A Comparison of Bandwidth Smoothing Techniques for the Transmission of Prerecorded Compressed Video," in *IEEE INFOCOM*, 1997.

[5] D. Wu, Y. T. Hou, and Y.-Q. Zhang, "Transporting Real-Time Video over the Internet: Challenges and Approaches," *Proceedings of the IEEE*, vol. 88, no. 12, pp. 1855–1874, December 2000.

[6] A. Gurtov and R. Ludwig, "Lifetime Packet Discard for Effi cient Real-Time Transport over Cellular Links," *ACM Mobile Computing and Communications Review*, vol. 7, no. 4, pp. 32–45, October 2003.

[7] M. Hemy, U. Hengartner, P. Steenkiste, and T. Gross, "MPEG System Streams in Best-Effort Networks," in *IEEE Packet Video*, 1999.

[8] Z.-L. Zhang, S. Nelakuditi, R. Aggarwal, and R. Tsang, "Effi cient Selective Frame Discard Algorithms for Stored Video Delivery across Resource Constrained Networks," in *IEEE INFOCOM*, 1999.

[9] "Location of MPEG4 Video Traces," `http://trace.eas.asu.edu/cgi-bin/main.cgi`, 2003.

[10] M. Reisslein, J. Lassetter, S. Ratnam, O. Lotfallah, F. H. P. Fitzek, and S. Panchanathan, "MPEG-4: Single Layer, Temporal and Spatial Scalability with PSNR Values," `http://trace.eas.asu.edu/pub.html`, 2002.