A Rate Control Scheme for Adaptive Video Streaming over the Internet

Panagiotis Papadimitriou and Vassilis Tsaoussidis

Demokritos University, Electrical and Computer Engineering Department, Xanthi, 67100, Greece
E-mail: {ppapadim, vtsaousi}@ee.duth.gr

***Abstract*—In this paper, we propose a new streaming protocol, namely *Dynamic Video Rate Control* (*DVRC*), which enables adaptive video delivery over the Internet. DVRC operates on top of UDP providing a congestion-controlled flow of unreliable datagrams. The proposed rate control scheme is able to interact with new and existing video streaming applications which are capable of adjusting their rate based on congestion feedback. DVRC attempts to optimize the performance of video delivery with concern to friendliness with interfering traffic. Exploring DVRC's potential through extensive simulations, we identify notable gains in terms of bandwidth utilization and smooth video delivery. Furthermore, our results indicate that the protocol allocates a well-balanced amount of network resources maintaining friendliness with coexisting flows.**

***Keywords*—QoS, transport protocols, video streaming**

I. Introduction

Time-sensitive applications, such as streaming media, gain popularity and multimedia data is expected to compose a large portion of the overall data traffic traversing the Internet. These applications generally prefer timeliness to reliability. Video streaming, in particular, calls for strict requirements on end-to-end latency and delay variation. Long end-to-end delays commonly affect the timely delivery of video-data causing data unavailability and unintelligible real-time interaction with frustrating consequences to the end-user. Furthermore, reliability parameters, such as packet loss and bit errors, usually compose an impairment factor, since they cause a perceptible degradation in video quality. Unlike bulk data transfers, video streaming seeks to achieve smooth playback quality rather than simply transmit at the highest attainable bandwidth.

Multimedia applications commonly *rely* on the unreliable transport services provided by *User Datagram Protocol* (*UDP*). UDP is a fast, lightweight protocol without any transmission or retransmission control. The protocol does not have functionality to override application characteristics, such as its transmission rates. It simply transmits at application rate and pattern. However, the lack of congestion control poses a threat to the network: had such applications dominated the Internet, it would have faced risk of congestion collapse. In this context, Internetworking functionality evolves towards punishing free-transmitting protocols.

On the other hand, *Transmission Control Protocol* (*TCP*), based on the principles of congestion management [7], *Slow-Start* [16] and *Additive Increase Multiplicative Decrease* (*AIMD*) [3], provides a reliable data delivery service to Internet applications and is in large part responsible for the remarkable stability of the Internet. However, the protocol occasionally introduces arbitrary delays, since it enforces reliability and in-order delivery. Furthermore, the process of probing for bandwidth and reacting to observed congestion induces oscillations in the achievable transmission rate. Several TCP protocol extensions [1, 6] have emerged to overcome the standard TCP limitations providing more efficient bandwidth utilization and sophisticated mechanisms for congestion control, which preserve the fundamental *Quality of Service* (*QoS*) guarantees for time-sensitive traffic. *TCP-friendly* protocols, proposed in [6, 18, 19] achieve smooth window adjustments, while they manage to compete fairly with TCP flows. In order to achieve smoothness, they use gentle backward adjustments upon congestion. However, this modification has a negative impact on protocol responsiveness. In [12, 17] we showed that TCP-friendly protocols are unable to effectively recover from excessive congestion incidents resulting in increased packet drops, which eventually degrade application performance.

Overcoming the oscillatory nature of AIMD congestion control, as well as the risks arising by the extensive usage of UDP, we need sophisticated congestion control that interacts efficiently with other flows on the Internet. An overview of Internet's current congestion control paradigm reveals that routers play a relatively passive role: they merely indicate congestion through packet drops or *Explicit Congestion Notification* (*ECN*). It is the end-systems that perform the crucial role of responding appropriately to these congestion signals. Numerous video-streaming applications have implemented their own congestion control mechanisms, usually on a case-by-case basis on top of UDP. However, implementing application-level congestion control is difficult and not part of most applications' core needs. We believe that a new transport protocol is needed, which would combine unreliable datagram delivery with built-in congestion control. This protocol would act as an enabling technology: new and existing applications could use it to timely transmit data without destabilizing the Internet.

In this context, we have been working on an architecture for adapting outgoing video streams to the characteristics of the end-to-end network path. We had the option to rely on the unreliable UDP datagrams or modify TCP to provide unreliable semantics. However, the latter seems particularly inappropriate considering the TCP semantics and its reliance on cumulative acknowledgments. Consequently, we considered UDP as a better choice, due to its unreliable and out-of-order delivery. Along these lines, we developed a new streaming protocol, namely

Dynamic Video Rate Control (*DVRC*), operating on top of UDP. DVRC is intended to interact with a plethora of streaming applications which are capable of adjusting their rate based on congestion feedback. The protocol incorporates end-to-end congestion control and does not rely on QoS functionality in routers, such as *Random Early Drop* (*RED*), *ECN* or other *Active Queue Management* (*AQM*) mechanisms. Our objective is to provide efficient and smooth rate control while maintaining friendliness with interfering flows.

The remainder of the paper is organized as follows. In the sequel, we provide an overview of related work. In Section III we discuss the design and implementation details of the proposed rate control scheme. In Section IV we present our evaluation methodology, followed by Section V, where we demonstrate conclusive performance studies based on simulations. Finally, in Section VI we highlight our conclusions.

II. Related Work

The literature includes several studies and proposals towards efficient end-to-end congestion control for streaming applications in the Internet. *Rate Adaptation Protocol* (*RAP*) [13] is a rate-based protocol which employs an AIMD algorithm for the transmission of real-time streams. The sending rate is continuously adjusted by RAP in a TCP-friendly fashion by using feedback from the receiver. However, since RAP employs TCP's congestion control parameters (i.e. 1, 0.5), it causes short-term rate oscillations, primarily due to the multiplicative decrease. Furthermore, RAP occasionally does not result in inter-protocol fairness.

Datagram Congestion Control Protocol (*DCCP*) [8] is a new transport protocol that provides a congestion-controlled flow of unreliable datagrams. DCCP is intended for time-sensitive applications which have relaxed packet loss requirements. The protocol aims to add to a UDP-like foundation the minimum mechanisms necessary to support congestion control. DCCP provides the application with a choice of congestion control mechanisms via *Congestion Control IDs* (*CCIDs*), which explicitly name standardized congestion control mechanisms (i.e TCP-like and TFRC).

Authors in [4] propose a *Real-Time Transport* (*RTP*) [15] compatible protocol, called *SR-RTP*, which adaptively delivers high quality video in the face of packet loss. SR-RTP employs binomial congestion control [1] and responds to packet drops by reducing the congestion window (*cwnd*) size proportional to the square root of its value instead of halving it. However, binomial schemes, such as *IIAD* or *SQRT* are not able to achieve TCP-friendliness independent of link capacity [2].

Since TCP is rarely chosen to transport time-sensitive traffic over the Internet, TCP-friendly protocols constitute an elegant framework for multimedia applications. We consider as TCP-friendly any protocol whose long-term arrival rate does not exceed the one of any conformant TCP in the same circumstances [5]. TCP-friendly congestion control maintains network stability by promptly responding to congestion and is also cooperative with other flows, while it commonly provides more efficient QoS (e.g. smoothed sending rate and bounded latency for playback multimedia applications). *TCP-friendly Rate Control* (*TFRC*) [6] is a representative TCP-friendly protocol, which adjusts its transmission rate in response to the level of congestion, as estimated based on the calculated loss rate. Multiple packet drops in the same *Round Trip Time* (*RTT*) are considered as a single loss event by TFRC and hence, the protocol follows a more gentle congestion control strategy. More precisely, the TFRC sender uses the following response function:

$$T(p, RTT, RTO) = \frac{1}{RTT\sqrt{\frac{2p}{3}} + RTO\,(3\sqrt{\frac{3p}{8}})\,p\,(1+32p^2)} \quad (1)$$

where p is the steady-state loss event rate and *RTO* is the retransmission timeout value. Equation (1) enforces an upper bound on the sending rate T. However, the throughput model is quite sensitive to parameters (i.e. p, RTT), which are often difficult to measure efficiently and to predict accurately. Also, the long-term TCP throughput equation does not capture the transit and short-lived TCP behaviors, and it is less responsive to short-term network and session dynamics [17].

TCP Westwood [10] is a TCP-friendly protocol that emerged as a sender-side-only modification of TCP Reno congestion control. TCP Westwood exploits end-to-end bandwidth estimation in order to adjust the values of slow-start threshold and *cwnd* after a congestion episode. The protocol incorporates a recovery mechanism which avoids the blind halving of the sending rate of TCP Reno after packet losses and enables TCP Westwood to achieve high link-utilization in the presence of wireless errors. However, in [11] we showed that TCP Westwood tends to overestimate the available bandwidth, due to ACK clustering. *TCP Westwood+* is a recent extension of TCP Westwood, based on the *Additive Increase/Adaptive Decrease* (*AIAD*) mechanism. Unlike the initial version of Westwood, TCP Westwood+ computes one sample of available bandwidth every RTT using all data acknowledged in the specific RTT, therefore obtaining more accurate estimates.

III. Design and Implementation

The design principles of the proposed rate control scheme mainly rest on the assumption that a user's perception is sensitive to smooth and timely playback of the received video frames. Despite the degradation in visual quality, smooth video of lower bitrate is considered more preferable than inconsistent and jerky video of higher rate. In this context, the primary goal of DVRC is to properly adjust the rate of the transmitted video stream in accordance to the prevailing network conditions. Furthermore, our approach anticipates friendliness with interfering traffic.

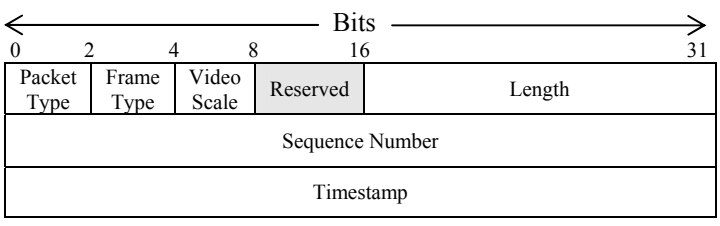


Figure 1. DVRC Header

Therefore, scalability is adjusted in terms of user-perceived video quality, as well as from the perspective of inter-protocol fairness.

A. Sender and Receiver Functionality

DVRC, in a complementary role, operates on top of UDP and supports end-to-end congestion control relying on sender and receiver interaction. DVRC acknowledges each datagram received by transmitting a control packet (containing no data). The protocol does not integrate reliability to UDP datagrams, so control packets do not trigger retransmissions. However, they are effectively used in order to determine bandwidth and RTT estimates, and properly adjust the rate of the transmitted video stream.

In this context, we have encapsulated additional header information to UDP packets (Fig. 1), including *packet type*, *length* and *sequence number*, *frame type*, *timestamp* and *video scale*. *Packet type* field denotes whether a segment with video-data or a control packet is transmitted. *Timestamp* field is used to handle RTT computation. More precisely, when the sender transmits a video-packet, it updates the specific field with current time. Upon the receipt of the corresponding control packet, the sender subtracts the included *timestamp* from current time in order to estimate the RTT sample. We discuss the *video scale* field in the following subsection.

Since UDP is an unreliable protocol, some datagrams may be lost due to congestion or inability of the receiving host from reading the packets rapidly enough. The receiver uses packet drops or re-ordering as congestion indicator. Consequently, congestion control is triggered, when:

- a packet is received carrying a sequence number greater than the expected sequence number
- the receiver does not acquire any packets within a timeout interval

Along these lines, the proper adjustment of the timeout interval is critical. A timeout interval that is set too short will claim false packet losses resulting in a wasteful reduction of the transmission rate. On the other hand, a long and consequently conservative timeout interval will inevitably impact the protocol responsiveness. In order to properly adjust the timeout, we need an accurate approximation of RTT. Hence, we measure *SampleRTT* and based on this quantity we further compute the weighted average of RTT:

$$\text{EstimatedRTT} = \alpha \times \text{EstimatedRTT} + (1-\alpha) \times \text{SampleRTT} \quad (2)$$

setting the smoothing factor *a* to 0.9. After RTT estimation, the timeout interval for DVRC (*DTO*) can be calculated by:

$$\text{DTO} = \text{EstimatedRTT} + \beta \times \text{Deviation} \quad (3)$$

where β is set to 4 and *Deviation* is the smoothed estimation of the variation of RTT. Deviation is represented as:

$$\text{Deviation}_n = \gamma \times \text{Deviation}_{n-1} + (1 - \gamma) \times \times | \text{EstimatedRTT} - \text{EstimatedRTT}' | \quad (4)$$

where $Deviation_{n-1}$ and *EstimatedRTT′* are the variation of RTT and the estimated RTT in the last round respectively, while γ is set to 0.25 inline with [6].

B. Rate Adjustment

The rate adjustment of the video stream is performed on a receiver-oriented fashion. Prior to the transmission of video-data, a selected number of scale values is assigned to diverse video encoding methods and transmission policies. The receiver detects the state of congestion and determines the next transmission rate in terms of a pre-defined scale value:

- if no congestion is sensed, the scale value is increased by 1
- in case of congestion, the scale value is decreased by 1

In order to explore the available bandwidth, transmission initiates with the lower scale value and increases linearly. The receiver uses control packets to periodically send feedback of reception statistics to the sender. Each control packet generated is updated with a scale value (using the *video scale* field), which signifies the proper transmission rate to the sender. In this context, the linear scale adjustments are automatically translated to gentle fluctuations in the transmission rate resulting in a smooth video flow.

From the perspective of inter-protocol fairness, although DVRC does not precisely halve its rate upon congestion, it allocates a well-balanced amount of resources and relinquishes them on the first indication of congestion. AIMD algorithm has significantly contributed to Internet stability; however, there is no proof that AIMD is the only approach to achieve this goal. On the other hand, it is a common sense that congestion control is essential to prevent congestion collapse. Furthermore, it is widely assumed that flows using end-to-end congestion control (even not compliant to AIMD) do not pose a threat to Internet stability. Along these lines, the proposed rate control scheme is viable, even in terms of large-scale deployment.

DVRC provides a framework upon existing or new streaming applications can effectively operate. Any video streaming server, which is able to adapt the quality of its transmission, can take advantage of DVRC's services. The desired scalability can be attained through various transcoding techniques [9]. *Simulcast* uses multiple versions of the stream, encoded at different bitrates. The versions of streams used are often limited in order to avoid high redundancy. The server transmits all the alternate streams and the client switches to the stream version that best matches its capacity. *Layered adaptation* has been proposed as a solution to bandwidth redundancy caused by simulcast. This approach is based on information decomposition. That is, the video stream is encoded at a base layer and one or more enhancement layers, which can be combined to render the stream at high quality. Layered adaptation is performed by adding or dropping enhancement layers depending on prevailing network conditions. The number of enhancement layers, as well as the selection of encoding bitrates is very important. In order to

Scale Value	Average Bitrate (Kbps)
4	340
3	256
2	170
1	86

Figure 2. Experimental Video Scale Assignment

handle scalability, DVRC supports a varying number of video scales which can be predetermined according to the transmission policies of the specific application. Furthermore, the field *frame type* in the DVRC header may augment a prioritized transmission (i.e. where I frames will be prioritized).

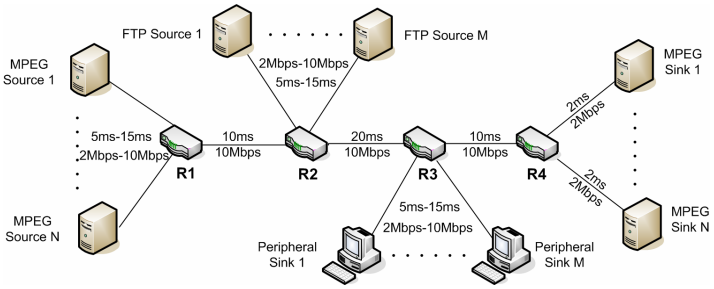
IV. Evaluation Methodology

A. Scenarios and Parameters

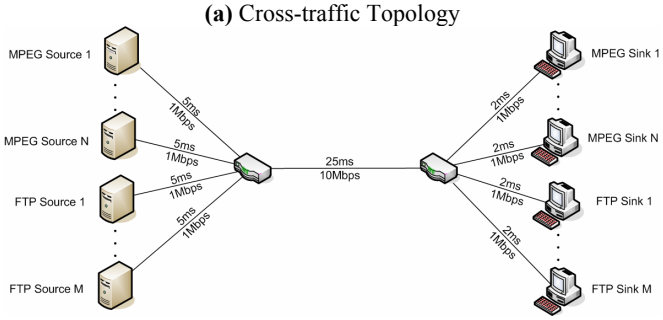
The evaluation plan was implemented on the *NS-2* network simulator. In order to assess the performance of DVRC, we implemented an experimental MPEG-4 video streaming server which is capable of video adaptation. The server transmits video data by dividing each frame into fixed size packets. The traffic generated closely matches the statistical characteristics of an original video trace. The model developed is based on *Transform Expand Sample* (*TES*). We used three separate *TES* models for modeling I, P and B frames, respectively. The resulting MPEG-4 stream is generated by interleaving data obtained by the three models. In the case of DVRC, the associated header (Fig. 1) is encapsulated in each segment during the packetization process. In our experiments, we relied on the approach of simulcast for video adaptation. We maintain 4 versions of the original video trace encoded at diverse bitrates. Each version is assigned at a predefined scale (1-4), as depicted in Fig. 2.

We performed extensive evaluations of DVRC versus the rate-based TFRC which is implied to yield remarkable efficiency on media delivery over a wide range of network and session dynamics. We also included the measurement-based TCP Westwood+ (TCPW) in our experiments. When the MPEG-4 server transmits over TFRC or TCPW, the video stream is adjusted at the optimal quality (i.e. scale value 4). As a result, we evaluate the transmission rate control performed by each protocol at the transport layer. However, in the case of DVRC, the video stream is delivered adaptively with the scale value initially set to 1. Under no signs of congestion, the scale increases linearly (until it has reached scale value 4).

The experiments were conducted based on realistic scenarios which address the heterogeneity of the Internet. More precisely, we used a network topology (Fig. 3a) which includes multiple bottlenecks, cross traffic, wireless links and varying RTTs. Cross traffic includes diverse FTP flows over TCP Reno. The propagation delays of the access links from all the source nodes, as well as the links to the FTP sink nodes range from 5 ms to 15 ms, while the corresponding bandwidth capacities range from 2 Mbps to 10 Mbps. NS-2 error models were inserted into the access links to the MPEG sink nodes. We used the Bernoulli model in order to simulate the link errors with packet error rate adjusted at 0.01.



(a) Cross-traffic Topology



(b) Dumbbell Topology

Figure 3. Simulation Topologies

We also enabled simulations on a single-bottleneck *dumbbell* topology with a bottleneck capacity of 10 Mbps in order to evaluate inter-protocol fairness (Fig. 3b). The bottleneck link is shared by competing MPEG and FTP connections. In both topologies, we set the packet size to 1000 bytes for all system flows and the maximum congestion window to 64 KB for all TCP connections. Each simulation lasts for 60 sec, and diverse randomization seeds were used in order to reduce simulation dynamics. All the results are collected after 2 sec in order to avoid the skew introduced by the warming up effect.

B. Measuring Performance

We hereby refer to the performance metrics supported by our simulation model. Since both topologies include competing MPEG and FTP connections, our performance metrics are applied separately to the MPEG and FTP traffic. *Goodput* is used to measure the overall system efficiency in bandwidth utilization. Long-term fairness is measured by the *Fairness Index*, derived from the formula given in [3], and defined as $(\sum_{i=1}^{n} \text{Throughput}_i)^2 / (n \sum_{i=1}^{n} \text{Throughput}_i^2)$, where $Throughput_i$ is the throughput of the i^{th} flow and n is the total number of flows. Inter-protocol fairness measurements were conducted based on *Normalized Throughput*, which is the ratio of the average throughput received by each flow over the bandwidth fair-share on each case.

Jitter composes a critical factor for the performance of video delivery. Based on [12], we use a metric for the performance evaluation on video delivery, called *Video Delivery Index*, which captures the joint effect of jitter and packet loss on perceived quality. The metric monitors packet inter-arrival times and distinguishes the packets that can be effectively used by the client application (i.e. without causing interruptions) from

delayed packets according to a configurable packet inter-arrival threshold. The proportion of the number of delayed packets is denoted as *Delayed Packets Rate*. Video Delivery Index is defined as the ratio of the number of *jitter-free* packets over the total number of packets sent by the application:

$$\text{Video Delivery Index} = \frac{\#\,\text{jitter - free packets}}{\#\,\text{sent packets}} \leq 1$$

In accordance with video streaming delay guidelines, we adjusted the packet inter-arrival threshold at 75 ms. Since MPEG traffic is sensitive to packet drops, we additionally define *Packet Drop Rate*, as the ratio of the number of lost packets over the number of packets sent by the application. For a system with multiple flows, we present the average of the Video Performance Index of each MPEG flow.

V. Results and Discussion

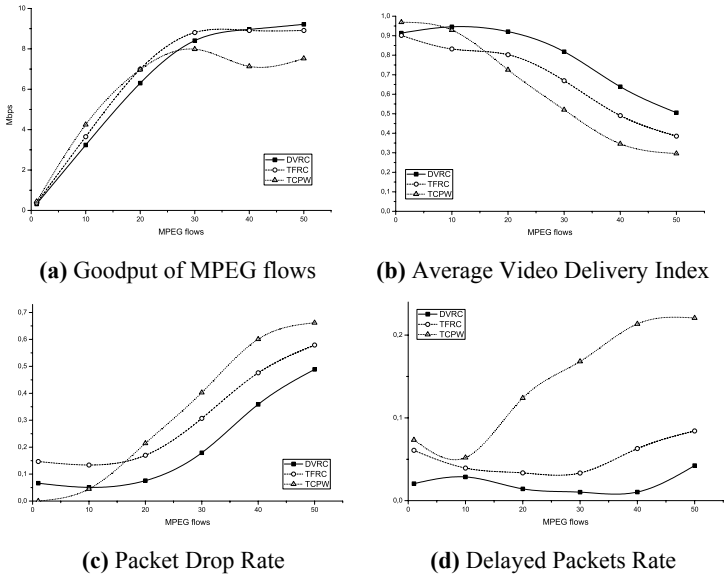
In the sequel, we demonstrate conclusive performance studies based on selected simulation results. More precisely, we evaluate the efficiency of DVRC in terms of video delivery and we further investigate DVRC's friendliness with interfering traffic.

A. DVRC Performance

Initially, we performed a series of experiments in order to assess the performance of our approach versus TCP-friendly traffic. We simulated a wide range of MPEG flows (1-50), adjusting the contention accordingly. Due to space limitations, we demonstrate DVRC, TFRC and TCPW flows competing with 10 FTP connections over TCP Reno, successively. The corresponding experiments were conducted on the cross-traffic topology. We measured *Goodput*, *Fairness Index* and *Video Delivery Index*, and we additionally demonstrate statistics from delayed and lost packets, since both compose influencing factors for perceived video quality (Figs. 4, 5). Furthermore, we present traces of the queue-length of router *R2* (Fig. 6) in the presence of 40 MPEG flows.

According to Fig. 4a, DVRC yields effective bandwidth utilization, especially for high link-multiplexing. The protocol occupies a balanced amount of resources achieving smooth video delivery and allowing interfering flows to obtain a fair share of the link. Inline with DVRC, TFRC manages to utilize a high fraction of the available bandwidth. Despite the improvements over the initial version of Westwood, TCPW's algorithm still does not obtain accurate estimates in heterogeneous environments, failing to achieve full utilization of the available bandwidth (Fig. 4a). This observation is profound in the case of scarce bandwidth (high contention), where the sending rate is diminished. Apparently, the protocol is sensitive to the disturbances caused by interfering FTP traffic.

From the perspective of video delivery, DVRC exhibits a remarkable efficiency, delivering smooth video which is slightly affected by contention (Fig. 4b). The protocol manages to control the trade-off between responsiveness and smoothness, exploiting the integrated rate control scheme. Upon congestion detection, the protocol triggers a gentle reduction in the transmission rate. Fig. 6 illustrates that DVRC results in variable



(a) Goodput of MPEG flows **(b)** Average Video Delivery Index

(c) Packet Drop Rate **(d)** Delayed Packets Rate

Figure 4. Performance on Video Delivery

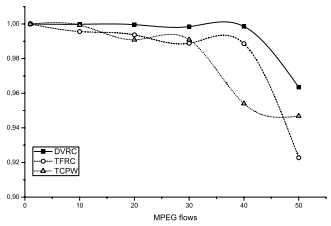


Figure 5. Fairness Index

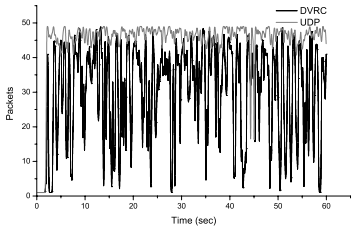
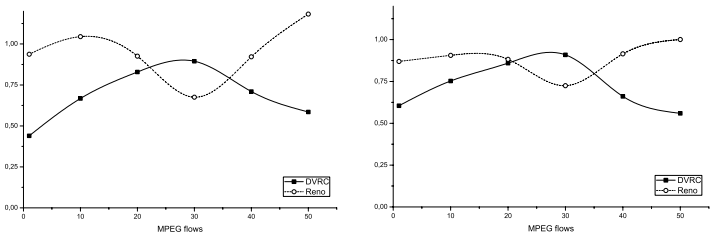


Figure 6. Queue Length of R2 (40 flows)

buffer conditions (due to the increased contention which inevitably leads to congestion), while UDP alone results in rapidly growing queues and eventually in buffer overflows. Along these lines, DVRC achieves the timely delivery of most packets enabling an uninterrupted and smooth receiving rate (Fig. 4d). Therefore, the transmitted video stream can be effectively reconstructed at the receiver. Such a performance does not necessitate the use of deep playback buffers in order to ameliorate the unpleasant effects of jitter. In addition, the low packet drop rate (Fig. 4c) in conjunction with the relaxed packet loss requirements of streaming video justifies our choice not to integrate reliability to DVRC.

According to Fig. 4d, TFRC delivers a smoothed flow and eventually confines the short-term oscillations in the sending rate, inline with DVRC. However, Fig. 4c illustrates relatively increased packet losses for TFRC, which deteriorate the perceived video quality (Fig. 4b). In dynamic environments with transient errors, TFRC occasionally fails to obtain accurate estimates of the loss event rate, invoking an inappropriate equation-based recovery, since TFRC's throughput model, as expressed in equation (1), is sensitive to the packet loss rate. TCPW is significantly affected by the increased queuing delays in our multi-hop network topology. The protocol exhibits increased packet drops (Fig. 4c), while a considerable proportion of the packets that are not dropped, reach the recipient later than required (Fig. 4d). Besides TCPW's tendency to overestimate the available bandwidth, the protocol slows down the transmission in



(a) 1-50 DVRC vs. 20 Reno flows **(b)** 1-50 DVRC vs. 40 Reno flows

Figure 7. Normalized Throughput

response to the transient errors. Consequently, the resulting transmission gaps induce interruptions in the receiving and playback rate of the video stream. The overall effect is long and variable delays, degrading the perceived video quality (Fig. 4b).

As depicted in Fig. 5, DVRC excels in bandwidths sharing, regardless of link-multiplexing. TFRC and TCPW achieve adequate levels of fairness, while for increased contention the *Fairness Index* for TFRC and TCPW slightly degrades, representing a perceptible unfairness to a small fraction of flows. We note that competing TFRC flows may have different throughput rates, since their observed loss event rates can differ.

B. DVRC Friendliness

We conclude our performance studies investigating the effect of DVRC on coexisting traffic. Based on the dumbbell topology, we simulated diverse MPEG/DVRC flows (1-50) competing with FTP traffic over TCP Reno. We demonstrate *Normalized Throughput* results from experiments with 20 and 40 FTP connections (Fig. 7).

Overcoming the *greedy* nature of UDP, DVRC monitors the prevailing network conditions and adjusts the video transmission rate maintaining friendliness with corporate flows. As depicted in Fig. 7, DVRC allows interfering TCP flows to obtain network resources close to the fair share of the link (i.e. *Normalized Throughput* of 1), which is critical for systems with multiple flows and different protocols. This observation is more profound in the situation of high link-multiplexing, where DVRC coexists fairly with TCP. In lightly loaded networks, the protocol allocates only the resources required to transmit video at the highest quality. In environments of moderate congestion, DVRC simply relies on sending lowest-quality video. Consequently, DVRC performs its task without implications on background traffic, preventing the potential of a congestive collapse.

On the contrary, recent studies (e.g. [14]) demonstrate a long-term throughput imbalance between competing TFRC and TCP connections. In particular, [14] reports that the throughput difference can be further amplified, as long as coexisting TCP and TFRC flows experience different loss event rates.

VI. CONCLUSIONS

We have proposed a rate control scheme for efficient video streaming delivery. DVRC is designed to interact with new and existing video streaming applications regardless of the selected scalability techniques or encoding policies. Our approach is able to adapt to the vagaries of the network reducing the oscillations in the transmission rate and eventually delivering smooth video. Through simulations, we identified significant gains for DVRC in highly-multiplexed dynamic networks. The corresponding performance studies reveal that the proposed rate control scheme compares very favorably with congestion control mechanisms that explicitly address time-sensitive traffic, such as TFRC. Finally, we demonstrated that DVRC effectively overcomes the greedy nature of UDP and maintains friendliness with interfering traffic. Future work will be focused on advanced error detection and the corresponding error recovery tactics in order to further enhance DVRC performance in heterogeneous wired/wireless environments.

REFERENCES

[1] D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithms", in *Proc. IEEE INFOCOM '01*, Anchorage, USA, April 2001.

[2] L. Cai, X. Shen, J. Pan, and J. Mark, "Performance analysis of TCP-friendly AIMD algorithms for multimedia applications", *IEEE Transactions on Multimedia*, 7(2), April 2005, pp. 339-355.

[3] D. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks", *Journal of Computer Networks and ISDN*, 17(1), June 1989, pp. 1-14.

[4] N. Feamster and H. Balakrishnan, "Packet Loss Recovery for Streaming Video", in *Proc. 12th IEEE Int/nal Packet Video Workshop*, Pittsburgh, USA, April 2002.

[5] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet", *IEEE/ACM Transactions on Networking*, **7**(4), August 1999, pp. 458-472.

[6] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-Based Congestion Control for Unicast Applications", in *Proc. ACM SIGCOMM 2000*, Stockholm, Sweden, August 2000.

[7] V. Jacobson, "Congestion avoidance and control", in *Proc. ACM SIGCOMM '88*, Stanford, USA, August 1988.

[8] E. Kohler, M. Handley, and S. Floyd, Designing "DCCP: Congestion control without reliability", in *Proc. ACM SIGCOMM 2006*, Piza, Italy, September 2006.

[9] J. Liu and Y. Zhang, "Adaptive Video Multicast over the Internet", *IEEE Multimedia*, 10(1), 2003, pp. 22-33.

[10] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", in *Proc. ACM MobiCom '01*, Rome, Italy, July 2001.

[11] P. Papadimitriou and V. Tsaoussidis, "Assessment of Internet Voice Transport with TCP", *Int/nal Journal of Communication Systems* (*IJCS*), 19(4), May 2006, pp. 381-405.

[12] P. Papadimitriou and V. Tsaoussidis, "On Transport Layer Mechanisms for Real-Time QoS", *Journal of Mobile Multimedia*, Rinton Press, 1(4), January 2006, pp. 342-363.

[13] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet", in *Proc. IEEE INFOCOM 1999*, New York, USA, March 1999.

[14] I. Rhee and L. Xu, "Limitations of Equation-based Congestion Control", in *Proc. ACM SIGCOMM 2005*, Philadelphia, USA, August 2005.

[15] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications", *RFC 1889*, IETF, January 1996.

[16] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", *RFC 2001*, January 1997.

[17] V. Tsaoussidis and C. Zhang, "The Dynamics of Responsiveness and Smoothness in Heterogeneous Networks", *IEEE Journal on Selected Areas in Communications* (*JSAC*), 23(6), June 2005, pp. 1178-1189.

[18] Y. R. Yang, M. S. Kim, and S. S. Lam, "Transient Behaviors of TCP-friendly Congestion Control Protocols", in *Proc. IEEE INFOCOM 2001*, Anchorage, Alaska, USA, April 2001.

[19] Y. R. Yang and S. S. Lam, "General AIMD Congestion Control", in *Proc. 8th IEEE Int/nal Conference on Network Protocols (ICNP)*, Osaka, Japan, November 2000.