PAPER *Special Issue on Internet Technology*

Streaming Video over TCP with Receiver-based Delay Control

Pai-Hsiang HSIAO[†], **H.T. KUNG**[†], ***and*** **Koan-Sin TAN**[††], ***Nonmembers***

SUMMARY Unicasting video streams over TCP connections is a challenging problem, because video sources cannot normally adapt to delay and throughput variations of TCP connections. This paper describes a method of extending TCP so that TCP connections can effectively carry hierarchically-encoded layered video streams, while being friendly to other competing connections. We call the method Receiver-based Delay Control (RDC). Under RDC, a TCP connection can slow down its transmission rate to avoid congestion by delaying ACK packet generation at the TCP receiver based on congestion notifications from routers. We present the principle behind RDC, argue that it is TCP-friendly, describe an implementation that uses 1-bit congestion notification from routers, and demonstrate by simulations its effectiveness in streaming hierarchically-encoded layered video.

key words: *video streaming, layered video, TCP, retransmission timeout, delay control*

1. Introduction

TCP is a dominant transport layer protocol in current Internet. It would be desirable if video and audio streams could be carried over TCP connections to take advantage of TCP's congestion control capabilities. However, it is well recognized that current TCP implementations are not suited for this purpose because TCP connections could introduce significant delay and throughput variations in the delivery of data [1].

There have been many proposals on new transport protocols for the purpose of solving this video transport problem, see e.g. the work by Rejaie et. al [2]. These protocols need to be TCP-friendly to ensure that they will not cause network collapse [1][3][4]. However, proving a new transport protocol to be TCP-friendly can be difficult, because the dynamics of TCP congestion control is extremely complex [5].

In this paper, we take a different approach: we extend TCP to make it suitable for transporting video, without modifying the TCP congestion control algorithm. In particular, we do not change the TCP sender code that governs TCP's behavior in the slow-start and congestion avoidance phases. The only change we make is on the TCP receiver side. In fact, our change is no more than extending the delayed ACK feature [6] in current TCP implementations, so that a longer delay can be imposed on ACKing (sending of ACK packets for received data packets) to avoid network congestion. For these reasons, we believe that our approach is, by design, TCP-friendly. We call this method "Receiver-based Delay Control" (RDC).

The three main contributions of this paper are summarized as follows:

- We propose RDC, which can slow down a TCP connection by extending ACKing delay, rather than shrinking its congestion window as in traditional TCP. As we shall argue this slowing down method increases the consistency of TCP performance.
- We demonstrate that RDC connections can behave like constant bit rate (CBR) pipes in the steady state. As a result, RDC connection is well-suited for video streaming.
- We describe a method of controlling the add and drop of video layers in streaming layered video over a TCP connection based on the buffer occupancy level of the TCP sending buffer. Our simulation results show improved performance of this layered streaming method when it is used together with RDC connections.

The rest of the paper is organized as follows. In Section 2, we describe the concepts and properties of a pure form of RDC ("exact RDC") that uses exact delay notification from routers in calculating ACKing delay, as well as an approximate version of RDC that uses 1-bit congestion notification from routers ("1-bit RDC"). Exact RDC is instructive in explaining the principle behind RDC, and serving as an ideal design point for performance comparison purposes, whereas 1-bit RDC represents a practical implementation of RDC. In Section 3, we present simulation results which establish the basic properties of RDC. Then in Section 4, we describe design and implementation of a source algorithm for transporting layered video streams [7] over TCP. In Section 5, we show our simulation results demonstrating the performance of RDC with the source algorithm in transporting layered video streams. In Section 6, we discuss some related work. Finally, in Section 7, we summarize and conclude the paper.

2. RDC Concepts

We first introduce the basic concepts and properties of exact RDC by comparing it with traditional TCP. We then describe two useful properties of exact RDC. Finally, we present 1-bit RDC that requires reduced network support similar to that of Explicit Congestion Notification (ECN) [8].

[†]Division of Engineering and Applied Sciences, Harvard University, USA
[††]Institute of Information Management, National Chiao-Tung University, Taiwan

2.1 Exact RDC

Consider a traditional FIFO-based router with incoming and outgoing links. As depicted by Fig. 1(a), each outgoing link has a FIFO buffer. Packets arriving on incoming links are forwarded to the FIFO buffer of an outgoing link. Packets are removed from that buffer and sent to its outgoing link at the link rate. The buffer occupancy, the number of packets that are currently stored in the FIFO buffer, increases when the arrival rate exceeds the departure rate. In the congestion avoidance phase of traditional TCP, a connection will grow its sending rate gradually until the FIFO buffer is exhausted and a packet is dropped.

In contrast, exact RDC depicted in Fig. 1(b) is able to keep the occupancy of the FIFO buffer low. The router will calculate a delay for each arriving packet using a token bucket based mechanism, and append a delay notification to the packet when it is forwarded to the next hop.

After receiving a data packet with a delay notification, the TCP receiver will forward the payload of the packet to the application immediately, but will impose a delay on the ACKing according to the received delay notification. However, if a data packet arrives out of order, an ACK packet will be sent immediately. Thus, duplicate ACK packets, triggered by out-of-order packets, are not delayed. This is essential for the proper working of fast retransmit and fast recovery [9].

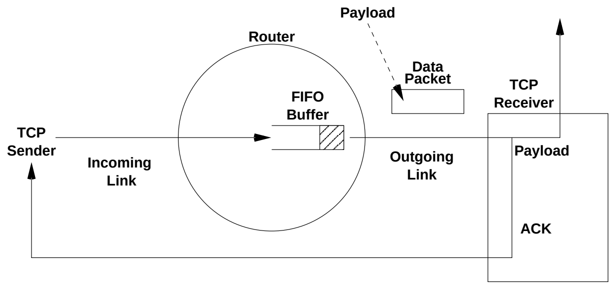
The router of Fig. 1(b) computes a delay for each arriving packet using a token bucket. The objective is that the computed delay for the packet should be the same as the delay the packet would experience if it was delayed in a FIFO buffer of a traditional router of Fig. 1(a). This ensures that exact RDC has the property described in the paragraph beneath Eq. (2). For simplicity, in the rest of the section we assume all packets in the FIFO buffer are of equal size, and a token in the token bucket represents a packet in the FIFO buffer. When packets are of different sizes, we may use a token to represent a byte of the packet.

More precisely, for each packet arriving at the router, not only it is forwarded to the outgoing FIFO buffer, a token is also inserted into the token bucket. We use token bucket level to represent the number of tokens that the token bucket currently has. The token bucket drains tokens at a rate (in tokens per unit time) smaller than the rate (in packets per unit time) that the outgoing link drains packets from the FIFO buffer. The token bucket may still drain even thought when the FIFO buffer is empty. The drain ratio ρ is defined as the ratio of the token bucket's drain rate over the link's output rate. The drain ratio ρ is always less than 1 for reasons to be explained later.

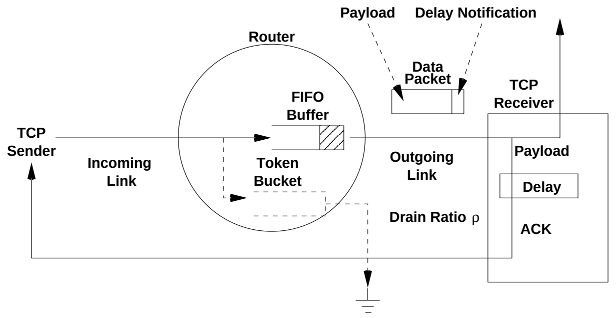
For each packet arriving at the FIFO buffer of Fig. 1(b), a delay D_{local} is computed as follows:

$$D_{local} = T_{transmission} \times (O_{tb} - O_{FIFO}), \qquad (1)$$

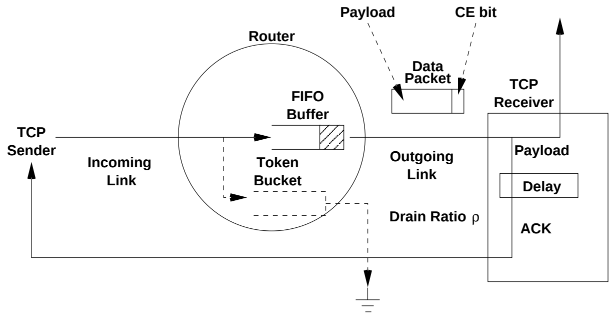
where $T_{transmission}$ is the packet transmission time over outgoing link, O_{tb} is the token bucket level in tokens, and O_{FIFO} is the FIFO buffer occupancy in packets.



(a) Traditional TCP Approach



(b) Exact RDC Approach



(c) 1-bit RDC Approach

Fig. 1 Compare RDC with traditional TCP. (a) In traditional TCP, packets are delayed in the FIFO buffer in the router during congestion; (b) in exact RDC, the router computes the delay of each arriving packet that it would experience in traditional TCP, using a token bucket method, and notifies the TCP receiver to impose the delay on the ACKing of the packet; and (c) in 1-bit RDC, the FIFO buffer in the router sets the Congestion Experienced bit (CE bit) with a marking probability determined by the calculated delay.

Suppose that the arriving packet is already appended with a delay notification $D_{incoming}$. A new delay notification $D_{outgoing}$ is calculated using

$$D_{outgoing} = \max(D_{incoming}, D_{local}), \qquad (2)$$

and is appended to the packet before it departs from the router.

After receiving a data packet with delay notification D, the TCP receiver will delay the ACKing of the data packet by D. A TCP connection under exact RDC behaves the same as the traditional TCP connection when the router has a large FIFO buffer and the outgoing link runs at a reduced speed that is ρ times the original. The only difference is, instead of delaying data packets in the FIFO buffer, the ACK packets are delayed at the receiver. Exact RDC, therefore, behaves like traditional TCP. Thus, it is TCP-friendly.

Token bucket level can grow without bounds, if the input rate is higher than the output rate for a long period of time. This could happen when the number of TCP flows is expanding. To prevent the calculated delay from growing unbounded, we limit the size of the token bucket to $size_{tb}$. When the token bucket level exceeds $size_{tb}$, the incoming packet is dropped. We note that although the computed delay may become large, it does not necessarily prevent RDC from utilizing available bandwidth. Large delay results in large round-trip time, in this case, by using a large congestion window size the connection still can achieve a high rate.

When the drain ratio is set to be less than 100% of the link rate, it helps to lower or even avoid the buffer occupancy. To illustrate this, consider the case when packets arrive at a rate that is higher than the drain rate but less than the link rate. In this case the token bucket level will arise while the buffer occupancy will not. Because of the raised token bucket level, departing packets will be marked to signal the TCP receiver to slow down the connection by extending the ACKing delay. Thus the slowdown can be achieved even without the associated increase in the buffer occupancy.

Choosing the ρ value is a matter of balancing between link utilization and buffer occupancy. As we will show later in the paper, when ρ is 90%, the FIFO buffer occupancy under exact RDC can be kept below a few packets. However, because the utilization on the outgoing link is bounded above by ρ times the bandwidth of the output link, we normally should not set ρ to be too low, e.g., below 90%. Although, for our simulation results reported in this paper, ρ is set to be 90%, we have observed that basically the same performance level can also be obtained if ρ is set to be 95%. Thus, when higher bandwidth utilization is required, we should choose a ρ value higher than 90%.

2.2 Properties of Exact RDC

A RDC connection is suited for transporting video. First, it can reduce the number of timeouts by allowing a larger congestion window size, resulting from extending round-trip time (RTT) by delaying ACKs. Second, it allows packets to experience reduced queueing delays in routers.

2.2.1 Reduced Number of Timeouts

We note that during the congestion avoidance phase, the rate of a TCP flow is determined by $cwnd$/RTT, where $cwnd$ is the congestion window size and RTT is the round-trip time. Thus, when the number N of TCP flows competing for the same network link increases, each flow must either decrease its $cwnd$ or increase its RTT.

Recall that $cwnd$ cannot be smaller than one packet. To avoid TCP timeouts, $cwnd$ needs to be larger than four packets to allow TCP fast retransmit and fast recovery to work [9][10]. In fact, to be "non-fragile," that is, resilient to retransmission timeouts, $cwnd$ needs to be about six packets if Explicit Congestion Notification (ECN) is not used [11][12].

Since it is undesirable to reduce $cwnd$ below certain limit, such as six packets, as noted above, increasing RTT becomes necessary when the number N of competing flows is sufficiently large. The RDC approach provides a way of extending RTT without introducing queueing delays in routers. That is, RDC delays the ACKing of packets at the TCP receiver instead.

2.2.2 Reduce Network Queueing Delays

As discussed above, under RDC a network does not build up queueing delays, and average queueing delays in a router can be kept below a few packets. This ensures low latency of packet delivery and allows the network to be responsive to congestion and flow control. Both of them are important for streaming applications. Keeping network queueing delay low is generally regarded as a good practice, as is often pointed out in the literature [13]–[15].

2.3 1-bit RDC

To simplify the router requirements, we suggest that RDC implementation use 1-bit Congestion Experienced (CE) notifications from routers, rather than notifications containing actual delays as in exact RDC described above. The CE bit is placed in the header of an IP packet and is used by ECN [8][16].

Recall that under exact RDC, the router calculates and appends delay for each packet, so the receiver can delay the ACK packet accordingly. As depicted in Fig. 1(b), the delay is calculated by the router using a token bucket based mechanism. An advantage of this approach is that the delay calculated reflects exactly the current congestion level at the router, so the receiver can quickly adjust to it. A disadvantage, however, is that there are no natural places in the TCP/IP headers to include the multi-bit delay value. We could use a header option field or the 16-bit ID field in the IP header as discussed in [17] for this purpose, but these are not standard methods.

As shown by 1-bit RDC of Fig. 1(c), RDC could be implemented using the CE bit in the IP packet header. The router could still employ a token bucket and update token bucket levels. But instead of appending each outgoing packet with the calculated delay, it only sets the CE bit in the IP packet header with a certain marking probability determined by the difference between the token bucket level and the FIFO buffer occupancy. That is, instead of using Eq. (1) and (2) to calculate D_{local} and $D_{outgoing}$, the router first calculates the difference to determine a marking probability. Then, if the incoming packet does not have the CE bit set, the router will set the bit with the marking probability. The marking probability increases linearly from 0 to 1, as the difference increases from 0 to a configured threshold, $thresh_{tb}$. When the difference is larger than $thresh_{tb}$, the CE bit is always set to the incoming packet. (As noted in Section 7, the CE bit can also be set by RED-like [14] algorithms without using the token bucket.)

The receiver will adjust the delay that is to be imposed on ACKing based on the percentage of received packets that have the CE bit set. The receiver estimates the round-trip time and uses it as an observation period over which the percentage is computed. That is, the observation period is the round-trip time of the flow, including the delay imposed to the ACK packet. We use two parameters, a and b, to denote some high and low thresholds, respectively. These thresholds will be used to determine whether to increase or decrease delay. For example, for $a = 0.9$ and $b = 0.1$, if 90% or above of the packets received in a period of time are set with the CE bit, the receiver will increase the delay for every future ACK packet. On the other hand, if only 10% or less of the packets received in the period of time are set with the CE bit, the receiver will decrease the delay.

When adjusting the delay, the receivers retain the same additive-increase and multiplicative-decrease (AIMD) [18][19] congestion window control behavior of traditional TCP. That is, during congestion avoidance phase, the TCP sender increases its sending rate additively by growing its window size additively (add one packet to the window size per round trip time) if it does not receive congestion signal during the round-trip time. It will reduce the window size by half to decrease its sending rate by half if there is a congestion signal. The AIMD behavior is important as it assures that TCP connections can reach equilibrium when they are in the congestion avoidance phase [18][20]. In RDC, the receiver maintains a value as the amount of time to delay the ACKing of each packet. This value will increase when sufficient congestion signals are received from the network, and will decrease when only few or no congestion signals are received. To retain the same behavior as AIMD congestion window control, we adjust the delay so that it is increased multiplicatively and decreased additively. This control method is thus additive-decrease and multiplicative-increase.

We use D_n to represent the amount of delay to be imposed on ACK packets in the nth observation period. Initially, the receiver imposes no delay for ACK packets, that is, $D_0 = 0$. When the observed percentage of packets with CE bit set is higher than the threshold a, the delay will be set to be the estimated round-trip time. Because ACKing to all packets received in the next observation period are delayed for this amount, it effectively doubles the round-trip time between the two end points. The length of the next observation period is then updated to be this new length, sum of the estimated round-trip time and the delay. The receiver continues to observe received packets and calculates the percentage for the next period of time. If the observed percentage is higher than the threshold a, the delay will be doubled. That is, $D_{n+1} = 2 \times D_n$, for $D_n > 0$, where D_{n+1} and D_n are new and old delays, respectively. On the other hand, if the observed percentage is lower than the threshold b, the delay will be reduced according to the following equation:

$$D_{n+1} = \left(\frac{cwnd}{cwnd + 1}\right) \times D_n \tag{3}$$

As shown in Eq. (3), the decrease in the delay is inversely proportional to the congestion window size ($cwnd$) plus 1. Thus, our delay update follows AIMD [18] principle.

In 1-bit RDC the CE bit is used as a signal for the receiver to estimate the value of the proper delay that would have been computed exactly in exact RDC. If the receiver underestimates the required delay on the ACKing of packets, then the sender's sending rate will still be higher than that the network allows. This means that packets with the CE bit marked will continue arriving at the receiver with high probability. The receiver, when still receiving packets with high marking probability, will further increase the amount of ACKing delay to decrease the sender's sending rate. On the other hand, when the sender's sending rate becomes lower than that the network can support, the receiver will not receive packets with high marking probability. The receiver will then decrease the delay to increase the sender's sending rate. Thus 1-bit RDC achieves the delay control effect that approximates that of exact RDC.

3. Simple Simulation for RDC

To study RDC, we have performed simulations for a simple network configuration in ns-2 [21]. Fig. 2 depicts the configuration, which is based on one of the configurations in [14]. Two flows, originating from two sources each with a 100 Mbps link to the gateway and having the same end-to-end round-trip delay of 42 ms, compete for the bottleneck link with a bandwidth of 45 Mb/s. The gateway has a buffer of 140 packets. The packet size is 1,000 bytes. The maximum window size of both flows is set to be 240 packets, which is slightly more than each flow's bandwidth-delay product (236.25 packets).

Note that if only one of the two flows is running, the flow's congestion window could reach the maximum window size of 240 packets. This is because some packets can

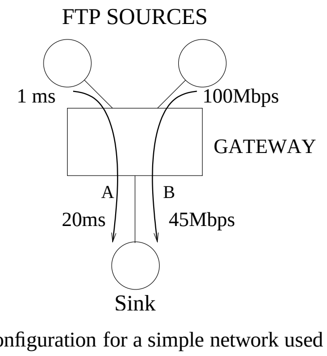


Fig. 2 The configuration for a simple network used in the simulation.

be queued at the gateway. There will be no loss of packets, since there is no buffer exhaustion. However, when the second flow starts running and competing for the bottleneck link, some packets will be dropped due to buffer exhaustion.

In the simulation, we start flow A at time zero and flow B one second later. One second is long enough for flow A to grow its congestion window to the maximum window size. Thus the rate of flow A stabilizes before flow B starts.

We compare four systems: DropTail, ECN, exact RDC and 1-bit RDC. We run simulations with their respective queue management algorithms implemented in the gateway and corresponding setups implemented in senders and receivers. For ECN, the parameters for RED in the gateway are min_{th} = 40, max_{th} = 120, $w = 0.002, max_p = 0.1$ and $gentle\_ = true$. Furthermore, senders and receivers process CE bit following the proposal in [16]. For both exact RDC and 1-bit RDC, $size_{tb}$ is set to be 6400 packets, and the drain ratio ρ equals to 0.9. For 1-bit RDC, the thresholds a and b are 0.9 and 0.1, respectively. The value of $thresh_{tb}$ is set to be 500.

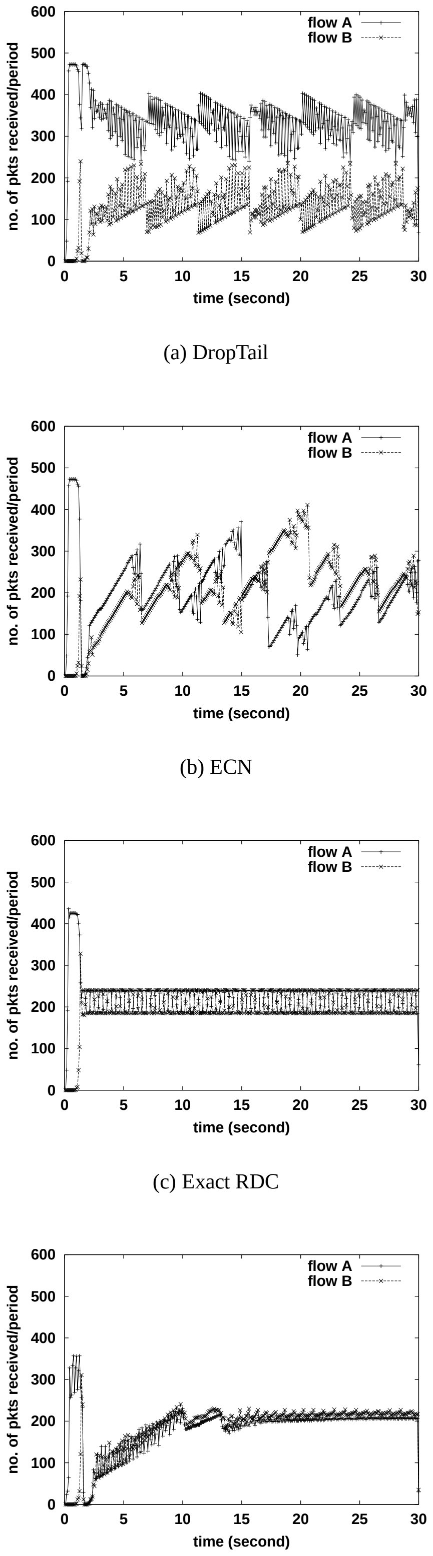
For 1-bit RDC, we need to estimate both RTT and $cwnd$ for each flow. For the RTT estimation, we use the interval between the arrival times of the first two data packets as an estimate. Other methods, such as the timestamp option as mentioned in [22], could be used to provide a more accurate RTT estimation. For the $cwnd$ estimation, we use the TCP sender's maximum window size as an estimate. We note that the use of maximum window size, instead of the current window size, makes 1-bit RDC less aggressive due to the reduced rate in decreasing delays based on Eq. (3).

Fig. 3 shows the number of packets sent by each flow in 84 ms periods. (We note that 84 ms is twice of the 42 ms end-to-end round-trip delay.) Under DropTail and ECN, both flows A and B continue exhibiting large throughput fluctuations even after 10 seconds. In contrast, under exact RDC and 1-bit RDC, the flows stabilize in less than 0.5 and 10 seconds, respectively.

The superior performance of RDC can be explained as follows. DropTail and ECN rely on TCP senders to control throughput by performing the AIMD control on $cwnd$. The throughput fluctuates between available bandwidth and half of the available bandwidth. In contrast, exact RDC continuously adjusts the sending rate by controlling the delay to be imposed on the ACKing of data packets. 1-bit RDC approximates the behavior of exact RDC.

Moreover, as shown in the figure, flow B in DropTail, ECN, and 1-bit RDC suffers from at least one timeout during the slow-start phase. This is caused by the fact that the router does not have a large enough buffer to absorb packet bursts introduced by the slow-start process. Consequently, some packets are dropped and it results in timeouts. Also in ECN, due to its use of average queue occupancy in determining CE marking probability, packets from both flows are dropped, and this results in the traffic phase effect [23]. Exact RDC does not seem to exhibit the phase effect, although 1-bit RDC sometimes does.

From Fig. 3(c) and 3(d), we see that exact RDC and 1-



(a) DropTail

(b) ECN

(c) Exact RDC

(d) 1-bit RDC

Fig. 3 Number of packets received at the router in 84ms periods over time for (a) DropTail, (b) ECN, (c) exact RDC, and (d) 1-bit RDC. (Simulation of the configuration of Fig. 2)

bit RDC can transport data with a relatively steady rate. In addition, we expect that they will experience only few or no timeouts, as discussed in Section 2.2.1. These properties are desirable for transporting video streams.

4. Layered Video Streaming

In this section, we describe our video encoding model, as well as the design and implementation of a streaming application. Beyond providing maximum available bandwidth for video delivery while remaining friendly to other flows, our goals include minimizing the playback latency.

4.1 Hierarchically-encoded Layered Video

We use a simple model for hierarchically-encoded layered video [24]–[26]. A video stream is hierarchically encoded into several layers, with every layer requiring the same delivery bandwidth. Data of a layer can only be played by the receiver when all data from its lower layers is received. The playback quality increases when data from additional layers is received. Streaming more layers delivers better quality of video, but requires more network bandwidth. Videos are encoded offline, with each layer stored separately.

4.2 Use of TCP as the Transport Protocol

An advantage of using TCP in transporting video is that the transport will be friendly to other flows sharing the same network and will not cause network collapse. However, it is not appropriate to use traditional TCP as is for streaming purposes, because it is designed for reliable data communication, not for real-time applications.

As discussed earlier, TCP connections may introduce significant bandwidth and delay variations. In particular, TCP connections may suffer retransmission timeouts. That is, TCP will cease transmission and wait for a retransmission timeout to expire, if sufficiently many packets are lost. Retransmission timeout can take seconds to expire, and this in turn can stop video playback for seconds. The receiver could buffer a large amount of data before it starts playback. However, this would increase the playback latency significantly. When timeouts happen frequently, even a large amount of buffering may not help. RDC addresses these obstacles as discussed in the previous sections.

4.3 Video Source Streaming Algorithm

When streaming video over a RDC connection, the multilayer video source decides dynamically when to add or drop a layer of encoded video. The decision will be based on the observed occupancy of the TCP sending buffer. Because network paths may experience different network conditions, the source needs to determine the highest layer N the network will allow at any given time. We say a streaming is at layer N if the source decides it is appropriate to send N layers of video to the receiver. Different streams may use different values of N at a given time, depending on their network condition. For each streaming session, the video source will monitor the TCP's sending buffer to detect change in network condition.

To support the monitoring required by our source application, we have extended the TCP agent in ns to support two additional variables: sending buffer size and sending buffer occupancy. Sending buffer size is defined to be the sending TCP agent's maximum window size. The sending buffer occupancy may change whenever the application writes data to the agent or the agent sends a data segment to the network. The application can read both variables. At any given time, the amount of data the application can write to the sending buffer is bounded by the sending buffer size minus the occupancy.

The source application is implemented as follows. The application is executed periodically when streaming video. If the TCP maximum segment size is X bytes, and the bandwidth requirement of each layer is B bytes/second, then the period is set to be X/B seconds. If a stream is currently at layer N, then the application will insert N segments to the buffer in each period, one from each of the N layers. The application monitors the buffer occupancy continuously to decide whether a layer should be added or dropped. If all the observed buffer occupancies over a predetermined interval are lower than a threshold T_{add}, an additional layer will be added to the stream. If an observed buffer occupancy is higher than a threshold T_{drop}, a layer will be dropped from the stream. Although the source algorithm has only been performed in the simulator at present, we believe it could be easily implemented in real-world systems. For example, on BSD-derived systems we can implement via `ioctl(2)` so the application can call and probe for the values of sending buffer size and buffer occupancy.

Fig. 4 illustrates sending buffer occupancy over time. The buffer occupancy decreases in the beginning, because the network can provide higher bandwidth than the streaming currently requires. At point 1, because all the observed buffer occupancies are lower than the threshold T_{add} in the period of I_1, an additional layer is added to the stream. The additional layer adds more data than the available bandwidth of the network can transport, so the buffer occupancy starts increasing. At point 2, the network is congested and packets are dropped, so sending data to the network is stopped

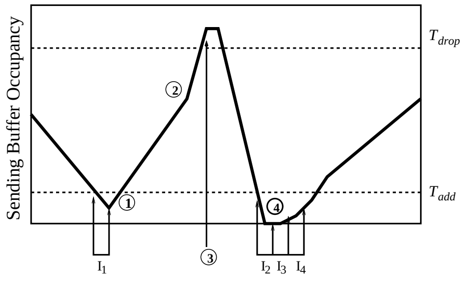


Fig. 4 Sending buffer occupancy over time. The two horizontal dashed lines are T_{add} and T_{drop}. The solid line gives the buffer occupancy over time.

because of retransmission timeout. As a result, the buffer occupancy increases rapidly. Later, at point 3, the buffer occupancy exceeds the threshold T_{drop}, so a layer is dropped from the stream. However, since the stream is still waiting for timeout to expire, some additional layers are dropped. When the timeout eventually expires and the TCP connection's congestion window opens up again, the TCP sender resumes transmission, and the buffer occupancy decreases. At point 4, three layers are added to the stream, one at a time.

The number of layers for a stream can be as low as zero, when the network is severely congested and no data can be delivered in time. In other words, the application can skip data for some streams during network congestion. This is necessary because once data are inserted into TCP's sending buffer, the source cannot cancel its delivery.

Note that our algorithm drops a layer immediately after one observation of high buffer occupancy, rather than several observations. This provides a rapid way of reducing the rate at which the source inserts data into the buffer when network congestion develops.

On the other hand, the predetermined observation period before adding a layer is set to be more than several seconds long. Since frequent fluctuation in the number of layers for a stream can cause the corresponding fluctuation in the playback quality, the purpose here is to minimize this fluctuation so as not to be annoying to video viewers.

5. Simulations of Video Streaming

In this section, we run three sets of simulations to study the performance of RDC connections in streaming video. The first two sets use similar network configurations, one set has 10 streaming connections, while the other has 100. We first describe our simulation setup for both cases, and then present results of each of the simulation sets in a subsection. Using the 10-stream simulation, we demonstrate that ECN, exact RDC, and 1-bit RDC have better stability than DropTail, and exact RDC and 1-bit RDC have smaller router buffer occupancy than DropTail and ECN. Using the 100-stream simulation, we demonstrate that 1-bit RDC performs better than ECN when there are many flows.

The third set of simulations, described in a separate subsection, uses a different configuration. This set of simulations compares the performance of ECN and 1-bit RDC in situations where streams with different round-trip times are mixed in the network.

5.1 Simulation Setup

Both sets of simulation use the network configuration depicted in Fig. 5, a configuration also used in [2]. In this network, the bottleneck link is the central link connecting routers G_0 and G_1. G_0 and G_1 have side links connecting to sender nodes (S_i) and receiver nodes (R_i), respectively. For each i, there is a TCP connection from node S_i to R_i. The variable n is the number of streaming connections, which is either 10 or 100. The bottleneck bandwidth is set to $0.64 \times n$ Mbps for reasons to be explained below. Both G_0 and G_1 have a FIFO buffer of 100 or 200 packets for the 10 or 100 connection setup, respectively.

We set all the data packets to have a fixed size of 1,000 bytes, and all TCP senders to have a maximum window of 40 packets. To reduce traffic phase effect among flows, we start all flows randomly in the first 20 seconds. We run the simulation for 800 seconds.

Our video data is multi-layer encoded as described in Section 4, with each layer requiring 20 KByte/s for its delivery. Each S_i is a source and delivers a stream to the corresponding R_i. For a 4-layer stream, a total of 80 KByte/s, or 0.64 Mbps, is required. Thus, to achieve the best streaming quality for all connections and maintain fairness, each source should continuously stream 4 layers of video on each connection. The period that the streaming application uses is 50 ms. The threshold T_{add} is set to zero packets, while T_{drop} is set to 20 packets, half of the TCP's maximum sending buffer size. The observation period for T_{add} is 15 seconds.

Four systems, DropTail, ECN, exact RDC, and 1-bit RDC, are compared in the 10-stream simulation, but only ECN and 1-bit RDC are compared for the 100-stream simulation. (For the 100-stream case, DropTail performs poorly.) When simulating with RED, we have min_{th} equal to $1/6$ of the FIFO buffer size (16.67 and 33.33, respectively, for 10-stream and 100-stream simulations), max_{th} equal to $1/2$ of the FIFO buffer size (50 and 100, respectively), w_q = 0.002, max_p = 0.1, and $gentle\_ = true$. For simulations with both exact RDC and 1-bit RDC, we use the same parameters as those used in simulations of Section 3, except that for the 100-stream simulation, $thresh_{tb}$ is increased from 500 to 5,000.

5.2 10-Stream Simulation

In this subsection we show that both exact RDC and 1-bit RDC can provide a relatively steady streaming quality, compared with DropTail and ECN. We also show that the packet delivery delay is significantly reduced for exact RDC and 1-bit RDC, because the FIFO buffer occupancy of the outgoing link from G_0 to G_1 is kept low in both cases.

In Fig. 6, results on achieved throughput in number of layers are presented. For each connection, the change in the

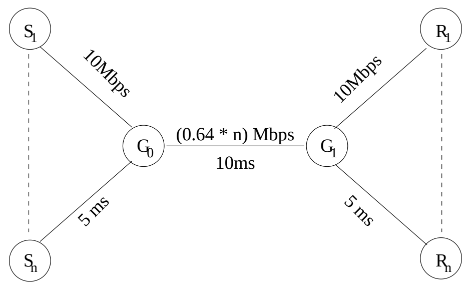
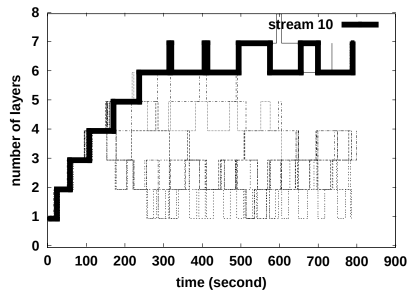
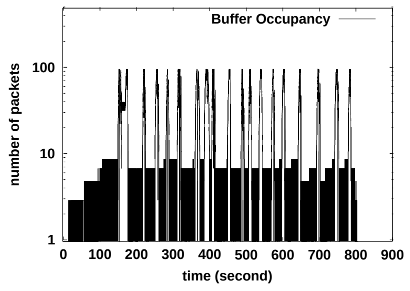


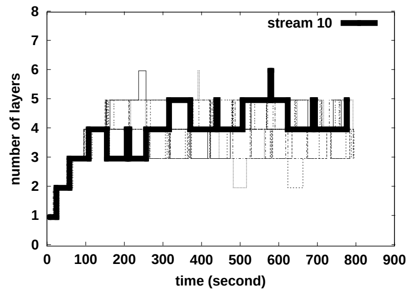
Fig. 5 Network configuration for video simulations.



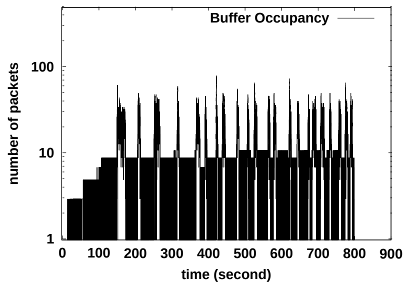
(a) DropTail



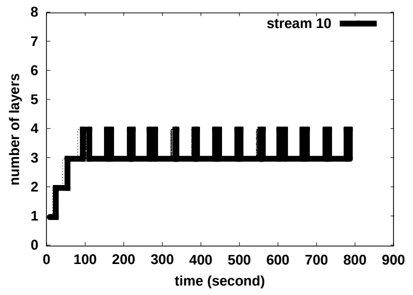
(a) DropTail



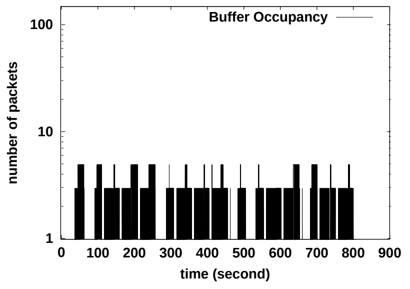
(b) ECN



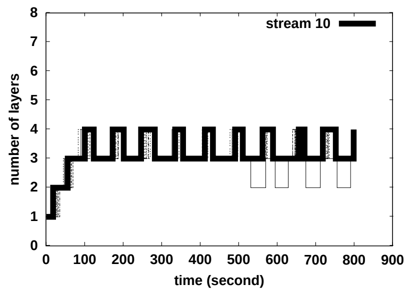
(b) ECN



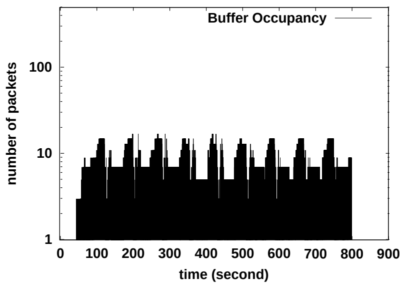
(c) RDC



(c) RDC



(d) 1-bit RDC



(d) 1-bit RDC

Fig. 6 10-stream simulation. Number of layers for each connection over time for (a) DropTail, (b) ECN, (c) exact RDC, and (d) 1-bit RDC. Only flow 10 is highlighted with a thick solid line, while the others are in thin dashed lines.

Fig. 7 10-stream simulation. Sampled FIFO buffer occupancy of the outgoing link from G_0 to G_1 for (a) DropTail, (b) ECN, (c) exact RDC, and (d) 1-bit RDC. Y-axis of the figures are in the log scale.

number of layers for each stream is shown. We note that the higher the number of layers, the better the quality of transported video. As expected, DropTail in Fig. 6(a) exhibits severe unfairness in the number of layers for different streams, and ECN in Fig. 6(b) also exhibits some fluctuation and unfairness. A random stream is highlighted with thick solid line for better presentation.

In contrast, both exact RDC and 1-bit RDC perform better in terms of fluctuation and fairness in the number of layers. However, since the throughput is bounded above at 90%, due to our choice of drain ratio $\rho = 0.9$ in the simulation, the average number of layers for both cases are lower than that of ECN's. The utilization will be increased accordingly if a larger value of ρ such as $\rho = 0.95$ is used.

In Fig. 7, we present results on the FIFO buffer occupancy. The buffer occupancy is sampled at every 50 FIFO buffer enqueue or dequeue events. The results show that exact RDC, in Fig. 7(c), has the lowest buffer occupancy of only a few packets. 1-bit RDC, in Fig. 7(d), has a lower occupancy and less fluctuation compared with both ECN and DropTail. These results imply that both delay and delay jitter are smaller with exact RDC and 1-bit RDC.

Finally, we note from our simulation that DropTail exhibits many packet loss and timeouts, while both ECN and 1-bit RDC exhibit only a few packet drops and timeouts. Exact RDC does not have any packet loss or timeout at all.

5.3 100-Stream Simulation

In this subsection, we present simulation results when the number of streams is increased from 10 to 100. We also increase G_0's FIFO buffer size from 100 to 200, and adjust RED's parameters accordingly. The parameters for 1-bit RDC remain the same. ECN and 1-bit RDC are compared, since the two methods depend on similar network support such as 1-bit congestion notification from routers. The results presented in this subsection demonstrate that 1-bit RDC performs well when the number of streams increases from 10 to 100, while the performance of ECN degrades.

Fig. 8 examines the performance results. In the top two figures of Fig. 8, we show the number of layers over time for 10 randomly selected streams among the 100 simulated streams. A random stream is highlighted with thick solid line in each figure to simplify presentation. We also present the sampled FIFO buffer occupancy in the bottom two figures. For the number of layers, both ECN and 1-bit RDC show slightly increased fluctuation compared with the 10-stream case. However, larger fluctuation in buffer occupancy is only observed under ECN.

The increased fluctuation in buffer occupancy of ECN connections is partly due to their relatively high packet loss rates and large numbers of timeouts. Table 1 shows the packet loss rate and the total number of timeouts during the lifetime of the 100 connections for both the ECN and 1-bit RDC cases. ECN exhibits significantly more timeouts than 1-bit RDC.

Table 1 Packet lost rate and number of timeouts in the lifetime of the 100-stream simulations.

	ECN	1-bit RDC
Packet lost rate (number lost)	8.5E-4 (5076)	3.5E-6 (17)
Number of timeouts	675	2

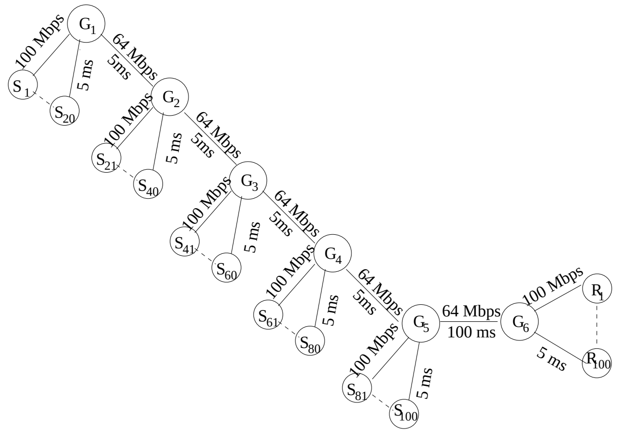


Fig. 9 Network configuration for video streams with different round-trip times.

5.4 Streams with Different Round-trip Times

We have studied, in previous subsections, performance of RDC and ECN when streams have homogeneous round-trip time. In this subsection, we compare their performance when streams have different round-trip times. Fig. 9 depicts the simulation network configuration.

There are 100 streams in this network, divided into 5 groups each with 20 streams. The round-trip times for streams in these groups are 220 ms, 230 ms, 240 ms, 250 ms and 260 ms. We note that these round-trip times are much longer than those considered in previous subsections. Parameters specific to ECN and RDC remain the same, as well as the queue size on the bottleneck link router.

As we expected, the performance, in terms of packet loss rate and buffer occupancy, of RDC and ECN are similar to the results of previous subsection. However, since streams now have different round-trip times, the performance of video layers over time are different.

Fig. 10 shows the number of layers for five randomly selected streams for ECN and RDC, one from each group. ECN exhibits higher layer disparity than RDC, due to different RTTs of streams. Streams with shorter RTT, for example, the one highlighted with thick line in Fig. 10(a), can grow its layers to 8 and starve other streams with longer RTTs.

5.5 Performance Summary

We have shown, by simulation, that the proposed multi-layered streaming can work well under ECN, exact RDC and 1-bit RDC. Traditional TCP with DropTail routers, on

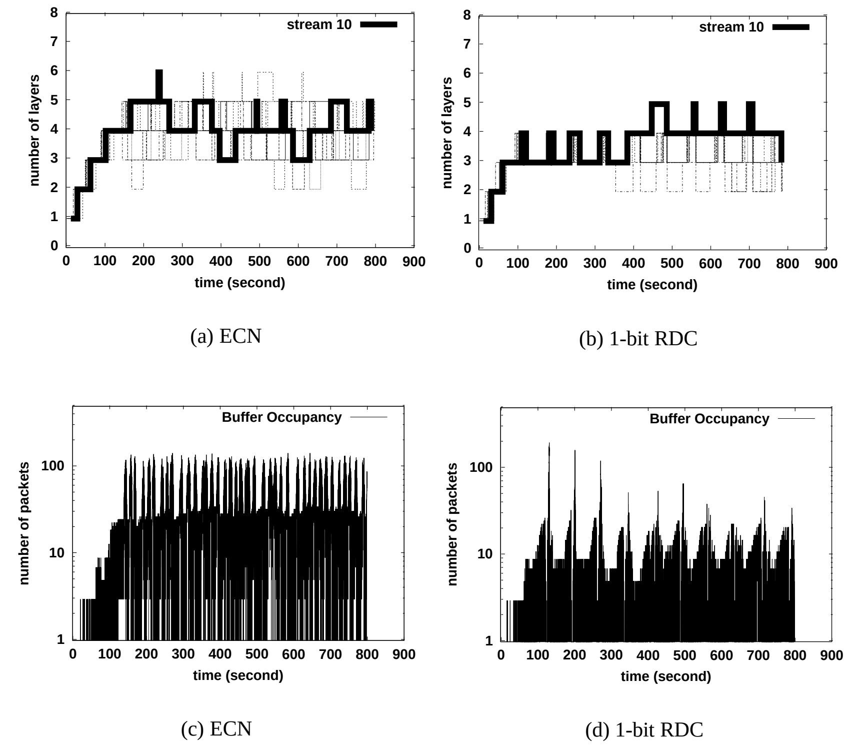


Fig. 8 100-stream simulation. The first row is the number of layers for each connection over time for (a) ECN, and (b) 1-bit RDC. 10 connections are randomly selected to simplify presentation, but only one is high lighted with a thick solid line. The second row is the sampled FIFO buffer occupancy of outgoing link from G_0 to G_1 over time.

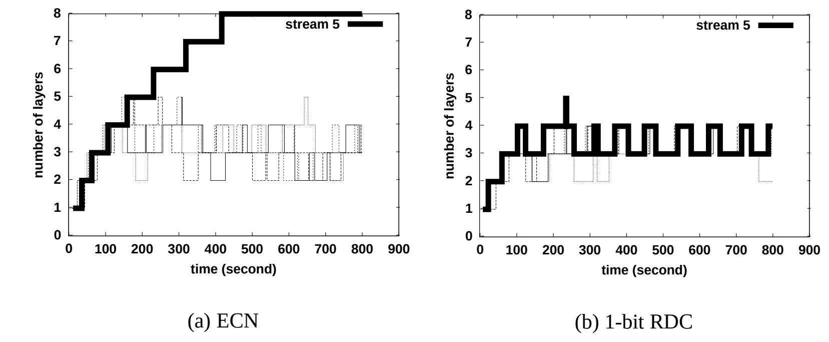


Fig. 10 100-stream simulation with different round-trip times. The number of layers for each connection over time for (a) ECN, and (b) 1-bit RDC are shown. Five connections are randomly selected to simplify presentation, one from each RTT group, but only one with the smallest RTT is high lighted with a thick solid line.

the other hand, performs poorly and is not appropriate for this purpose.

Exact RDC generally outperforms ECN and 1-bit RDC. However, as noted in Section 4, 1-bit RDC, which is an approximation of RDC, is more attractive than exact RDC for easy implementation and deployment.

When ECN and 1-bit RDC are compared, we note that when the number of competing streams is small (see the 10-stream simulation results of Fig. 6 and 7), 1-bit RDC performs better than ECN, but not by much. The performance difference becomes significant when the number of competing streams increases (see 100-stream simulation results of Fig. 8 and Table 1). This is because the RDC approach can reduce the number of timeouts as explained in Section 2.2.1.

6. Related Work and Discussion

In supporting exact RDC we require a router to manage the

FIFO buffer by dropping packets and sending out delay notifications based on the buffer occupancy and the token bucket level. Thus, it is an instance of active queue management (AQM) [13] as opposed to scheduling algorithm [27] that applies different treatments to different classes of packets in order to improve performance. Token-bucket based marking schemes similar to ours in Section 2.3 seem to have gained some popularity recently in the literature; see for example [28].

ECN [8] is a framework for sending explicit congestion signal from routers to TCP senders or receivers. Routers with AQM, such as RED, detect persistent high buffer occupancy and notify the senders that contributes to the congestion with such a signal. In response to the presence of congestion signal, these senders reduce their sending rates by half. In contrast, routers in the RDC framework notify receivers the need for delaying the ACKing of data packets.

ECN and RDC use different approaches in dealing with congestion. Under ECN, TCP senders adjust congestion window size to avoid congestion, whereas under RDC, TCP receivers control the delay of ACK packets. Other receiver-based congestion control methods manage the advertised window size, such as those discussed in [29].

Kanakia [30] studied the use of feedback information from the network, such as the router buffer occupancy, to modulate the source rate of a video encoder. The congestion control scheme used there is optimized for video quality and does not consider the TCP friendly requirement when there are other competing TCP flows. We consider in the paper not only optimizing video playback quality, which is achieved by reduced timeouts, steady delivery rate and layered encoding, but also being friendly with other competing TCP flows.

Real-time Transport Protocol (RTP) is a popular protocol that is used to stream multimedia over the Internet [31]. RTP marks packets with payload type identifiers, as well as time stamps and sequence numbers, so that audio and video data can be synchronized at the receiving host. But RTP does not itself have a feedback mechanism whereby the transmitter can be informed about the status of data delivery as seen at the receiver. An adjunct to RTP is a control protocol known as Real-Time Control Protocol (RTCP). This protocol allows information about sender identification, quality of service, packet losses and other factors to be interchanged between RTP sender and receiver, while the RTP data are being transmitted. In other words, RTCP provides a control channel for RTP. Feedback from the receivers is mainly for diagnosing delivery faults. RTCP produces sender and receiver reports, such as stream statistics and packet counts. However, neither RTP nor RTCP does congestion control for applications.

The main advantage of our streaming algorithm using RDC connections is that we do not need to develop a new congestion control algorithm for applications. The changes to the receiver are minimal while the requirements of intermediate routers are similar to that of ECN. As we mentioned in the paper, the ACK delaying mechanism at the receiver could be viewed as a natural extension of TCP delayed ACK. The token bucket for routers is not necessary, if one can set the parameters of RED or other active queue mechanisms properly.

7. Summary and Concluding Remarks

We have demonstrated by simulation that TCP connections with Receiver-based Delay Control (RDC) works well for carrying layered video streams. We have described an implementation, called 1-bit RDC, that only uses 1-bit in the IP header to carry congestion notifications generated by routers. Thus, 1-bit RDC could be implemented using the CE bit in the IP header.

For 1-bit RDC, accurate delays are actually not important for setting the CE bit in a packet header. A randomized algorithm for detecting congestion, such as RED, can generally achieve the same effect, if parameters are set correctly.

In addition to RDC, we have described a method for streaming hierarchically-encoded layered videos over TCP connections. The method includes an algorithm for a video streaming source to add or drop layers dynamically based on the buffer occupancy of the TCP sending buffer. Our simulation results show that the method works well with RDC connections.

While being able to utilize available bandwidth in delivering data and reduce playback latency, the RDC connection itself is friendly to competing TCP connections. Thus the combination of our RDC and layered video source algorithms offers an attractive approach to streaming video over IP networks.

8. Acknowledgment

This research was supported in part by NSF grant ANI-9710567, Air Force Office of Scientific Research Multidisciplinary University Research Initiative Grant F49620-97-1-0382, and grants from Microsoft Research, Nortel Networks, and Sun Microsystems.

References

[1] S. Floyd, M. Handley, J. Padhye, and J. Widmer, “Equation-based congestion control for unicast applications,” in *Proceedings of SIGCOMM 2000*, Aug. 2000.

[2] R. Rejaie, M. Handley, and D. Estrin, “Quality adaptation for congestion controlled video playback over the internet,” in *Proceedings of SIGCOMM ’99*, pp. 189–200, 1999.

[3] S. Floyd and K. R. Fall, “Promoting the use of end-to-end congestion control in the internet,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999.

[4] J. Mahdavi and S. Floyd, “Tcp-friendly unicast rate-based flow control.” Note sent to end2end-interest mailing list, Jan. 1997.

[5] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, “Modeling TCP throughput: A simple model and its empirical validation,” *ACM Computer Communication Review*, vol. 28, pp. 303–314, Sep. 1998.

[6] D. Clark, *Window and Acknowledgment Strategy in TCP*, Jul. 1982. RFC 813.

[7] M. Vishwanath and P. Chou, “An efficient algorithm for hierarchical compression of video,” in *Proceedings ICIP-94 (IEEE International*

Conference on Image Processing), vol. 3, (Austin, Texas), pp. 275–279, IEEE Computer Society Press, 1994.

[8] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, no. 5, pp. 10–23, 1994.

[9] M. Allman, V. Paxson, and W. Stevens, *TCP Congestion Control*, Apr. 1999. RFC 2581.

[10] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, MA, USA: Addison-Wesley, Dec. 1993.

[11] D. Lin. and H. T. Kung, "TCP fast recovery strategies: Analysis and improvements," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), Mar./Apr. 1998.

[12] P.-H. Hsiao, H. T. Kung, and K.-S. Tan, "Active delay control for TCP," in *Proceedings of the IEEE Conference on Global Communications (GLOBECOM)*, (San Antonio, TX), 2001.

[13] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, *Recommendations on Queue Management and Congestion Avoidance in the Internet*, Apr. 1998. RFC 2309.

[14] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[15] R. Jain, "Congestion control in computer networks: Issues and trends," *IEEE Network*, pp. 24–30, 1990.

[16] K. Ramakrishnan and S. Floyd, *A Proposal to add Explicit Congestion Notification (ECN) to IP*, Jan. 1999. RFC 2481.

[17] S. Savage, D. Wetherall, A. R. Karlin, and T. Anderson, "Practical network support for IP traceback," in *Proceedings of SIGCOMM 2000*, (Stockolm, Sweden), pp. 295–306, 2000.

[18] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, pp. 1–14, 1989.

[19] V. Jacobson, "Congestion avoidance and control," *ACM Computer Communication Review*, vol. 18, pp. 314–329, Aug. 1988. Proceedings of the SIGCOMM '88 Symposium in Stanford, CA, August, 1988.

[20] F. P. Kelly, "Stochastic modes of computer communication systems," *Journal of the Royal Statistical Society Series B*, vol. 47, no. 3, pp. 379–395, 1985.

[21] UCB/LBNL/VINT, "Vint network simulator - ns," 1997.

[22] I. M.-C. Tam, J. Su, and W. Wang, "Improving TCP performance over asymmetric networks," *ACM Computer Communication Review*, vol. 30, Jul. 2000.

[23] S. Floyd and V. Jacobson, "On traffic phase effects in packet-switched gateways," *Computer Communication Review*, vol. 21, pp. 115–156, Apr. 1991.

[24] J.-Y. Lee, T.-H. Kim, and S.-J. Ko, "Motion prediction based on temporal layering for layered video coding," in *Proc. ITC-CSC*, vol. 1, Jul. 1998.

[25] S. McCanne and M. Vetterli, "Joint source/channel coding for multicast packet video," in *Proceedings of IEEE International Conference on Image*, Oct. 1995.

[26] S. McCanne, *Scalable Compression and Transmission of Internet Multicast Video*. PhD thesis, University of California, Berkeley, 1996.

[27] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Internetworking: Research and Experience*, vol. 1, pp. 3–26, 1990.

[28] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue algorithm for active queue management," in *Proceedings of ACM SIGCOMM 2001*, Aug. 2001.

[29] N. T. Spring, M. Chesire, M. Berryman, V. Sahasranaman, T. Anderson, and B. N. Bershad, "Receiver based management of low bandwidth access links," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, pp. 245–254, 2000.

[30] H. Kanakia, P. P. Mishra, and A. Reibman, "An adaptive congestion control scheme for real-time packet video transport," in *Proceedings of the 1993 ACM SIGCOMM Annual Technical Conference*, Sep. 1993.

[31] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*, Jan. 1996. RFC 1889.

Pai-Hsiang Hsiao Pai-Hsiang Hsiao is a Ph.D student in the Division of Engineering and Applied Sciences at Harvard University. He is interested in congestion control, wireless networks and peer-to-peer systems.

H.T. Kung H.T. Kung is a William H. Gates Professor of Computer Science and Electrical Engineering at Harvard. Dr. Kung received his Ph.D. from Carnegie Mellon and served on their faculty before joining Harvard in 1992. He teaches and researches computer networks and their applications. Dr. Kung has pursued a variety of interests over his career, including complexity theory, database theory, VLSI design, parallel computing, computer neworks, and security. Dr. Kung co-chairs a joint Ph.D. program with the Harvard Business School on information technology and management.

Koan-Sin Tan Koan-Sin Tan is a Ph.D. student in Institute of Information Management, National Chiao-Tung University, Hsinchu, Taiwan. He is interested in congestion control and network security. He visited Harvard University from 2000–2002, during which he participated in research reported in this paper.