International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2001), June 2001, pp. 199-208

Video over TCP with Receiver-based Delay Control

Pai-Hsiang Hsiao
Harvard University
33 Oxford Street
Cambridge, MA 02138, USA
shawn@eecs.harvard.edu

H. T. Kung
Harvard University
33 Oxford Street
Cambridge, MA 02138, USA
kung@harvard.edu

Koan-Sin Tan
National Chao-Tung University
1001 University Road
Hsinchu, 300, Taiwan
freedom@iim.nctu.edu.tw

ABSTRACT

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 points out a direction on how TCP can be modified such that TCP connections can carry hierarchically-encoded layered video streams well, while being friendly to other competing flows. The method is called *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 notifications from routers. The paper presents the principle behind RDC, argue that it is TCP-friendly, describe an implementation that uses 1-bit congestion notification from routers, and give our simulation results.

1. INTRODUCTION

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 [8] because TCP connections can significantly increase delay and throughput variations.

There have been many proposals on new transport protocols (see, e.g., [20]), for the purpose of solving this video transport problem. These protocols need to be TCP-friendly [8,15], to ensure that they will not cause network collapse. However, proving a new transport protocol to be TCP-friendly is fundamentally difficult, because of the high complexity of TCP dynamics.

In this paper, we take a different approach: we modify TCP to make it suitable for transporting video, without modifying TCP congestion control algorithms. 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 in the current TCP implementation, so that a longer delay can be imposed to avoid network congestion. For these reasons, we believe that our approach is, by design, TCP-friendly. We call this method "TCP with *Receiver-based Delay Control*" (RDC), or, in short, TCP RDC.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
NOSSDAV'01, June 25-27, 2001, Port Jefferson, New York, USA.
Copyright 2001 ACM 1-58113-370-7/01/0006…$5.00.

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, as well as an approximate version of RDC that uses 1-bit congestion notification from routers ("1-bit RDC"). Exact RDC is useful in explaining the principle behind RDC, and serving as an ideal design point for performance comparison purposes. 1-bit RDC represents a practical implementation of RDC. Simulation results which establish the basic properties of RDC are given in Section 3. Design and implementation of a source algorithm for transporting layered video streams [26] over TCP are described in Section 4. Finally, in Section 5 we show our simulation results demonstrating the performance of TCP RDC in carrying layered video streams. Discussions on related work and concluding remarks are given in Section 6 and Section 7, respectively.

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

- We demonstrate that TCP RDC connections can behave like constant bit rate (CBR) pipes in the steady state, and, as a result, can be well-suited for video streaming.
- We describe a method of controlling the add/drop of video layers in streaming video over a TCP connection based on the buffer occupancy level of the TCP sending buffer. Our simulation results show superior performance of this layered streaming method when it is used together with TCP RDC connections.

2. TCP 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 RDC. Finally, we describe 1-bit RDC that requires network support similar to that of Explicit Congestion Notification (ECN) [7].

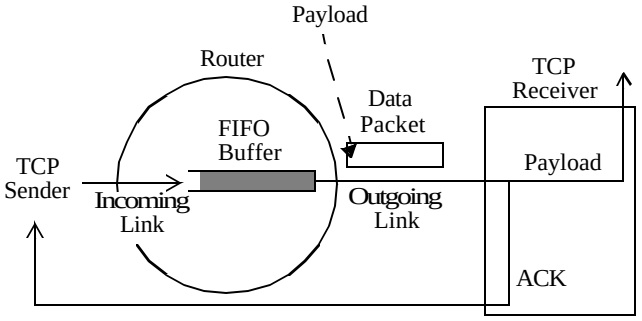
2.1. Exact RDC

Consider a traditional FIFO-based router with incoming and outgoing links. As depicted by Figure 1 (a), each outgoing link has a FIFO buffer. Packets arriving on incoming links are forwarded to the FIFO buffer of an outgoing link via some routing or forwarding mechanisms. Packets are removed from that buffer and sent on to its outgoing link at the link rate. The occupancy of the FIFO buffer in Figure 1 (a) will build up when the arrival rate exceeds the departure rate. In the congestion avoidance phase of traditional TCP, a connection grows its sending rate gradually until the router buffer is exhausted and a packet is dropped.

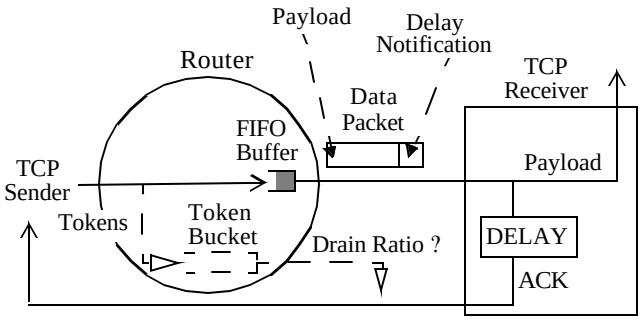
In contrast, exact RDC depicted in Figure 1 (b) is able to keep the occupancy of the FIFO buffer in the router low. The router will calculate a delay for each arriving packet using a token-bucket based mechanism, and append a delay notification to the packet when it is forwarded to the next hop.

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 (generation of ACK packets for the data packet) according to the received delay notification. However, if the 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 [1].

(a) *Traditional TCP*



(b) *Exact RDC*



(c) *1-bit RDC*

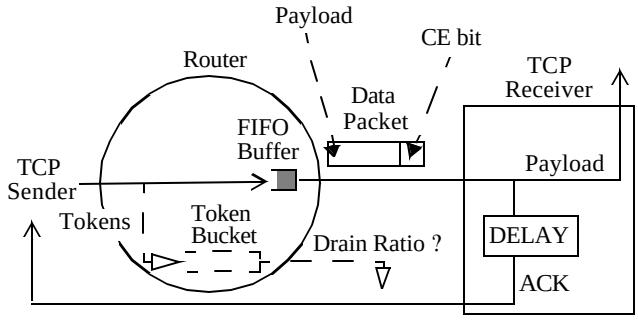


Figure 1: Compare RDC with traditional TCP. (a) In traditional TCP, packets are delayed in the FIFO buffer in the router; (b) in exact RDC, the router computes the delay of each arriving packet using a token bucket method, and notifies the TCP receiver to impose the delay on the ACKing of the packet;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

The token bucket in the router of Figure 1 (b) computes a delay for each arriving packet as follows. 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 Figure 1 (a).

More precisely, for each packet arriving at the router, a token is inserted into the token bucket. 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. The drain ratio ? is expressed as the ratio of the token bucket's drain rate (in tokens per unit time) over the link's output rate (in packets per unit time). ??is?always smaller than 1.

For each packet arriving at the FIFO buffer of Figure 1 (b), a delay *D-local* is computed as follows:

$$\begin{aligned} \textit{D-local} = {} & \text{Packet\_Transmission\_Time over outgoing link} \\ & * (\text{Token\_Bucket\_Level in Tokens} \\ & - \text{FIFO\_Buffer\_Occupancy in Packets}) \end{aligned} \quad (1)$$

Suppose that the arriving packet already has a delay notification *D-incoming*. A delay notification *D*, which is:

$$\textit{D-outgoing} = \text{maximum}(\textit{D-incoming}, \textit{D-local}) \quad (2)$$

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

After receiving the 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 of ? 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 and thus 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. To prevent the calculated delay from growing unbounded, we limit the size of the token bucket (*tb_size*). When the token bucket level exceeds *tb_size*, the incoming packet is dropped.

Choosing the ? value is a matter of balancing regarding a trade-off between link utilization and buffer occupancy. As we will show later in the paper, when ? is 90%, the FIFO buffer occupancy under exact RDC of Figure 1 (b) can normally be kept below a few packets. However, because the utilization on the outgoing link is bounded above by ?, we normally should not set ? to be below 90%. Although, for our simulation results reported in this paper, ? is set to be 90%, we have observed that basically the same performance level can also be obtained if ? is set to be 95%. Thus, when higher bandwidth utilization is required, we could choose a higher value for ?.

2.2. RDC Properties

RDC possesses two properties that make RDC connections suitable for transporting video streams. First, the number of time-outs is reduced by allowing a larger congestion window size, resulting from extended RTT via the RDC mechanism. Second, packets experience smaller network queueing delays in routers.

2.2.1. Reduced Number of Time-outs

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 competing TCP flows 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. In fact, to avoid TCP time-outs, CWND needs to be larger than five or six packets to allow TCP fast retransmit and fast recovery to work [23]. To be truly "non-fragile," that is, resilient to TCP time-outs, CWND in fact needs to be a few packets larger than five or six packets [14].

Since it is undesirable to reduce CWND below certain limit 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 to increase RTT.

2.2.2. Small Network Queueing Delays

As stated 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 [2,6,10].

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 an IP packet. The CE bit for 1-bit RDC is similar to that for Explicit Congestion Notification (ECN) [7,19].

Recall that under exact RDC of Figure 1 (b), the router calculates and appends delay for each packet, so the receiver can delay the ACK packet accordingly. The delay is calculated by the router using the 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 information. We could use a header option field or the 16-bit ID field in the IP header as discussed in [21] for this purpose, but these are not standard methods.

As shown by 1-bit RDC of Figure 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 token bucket level and FIFO buffer occupancy. That is, instead of using Equations (1) and (2) to calculate *D-local* and *D-outgoing*, the router first calculates the difference between token bucket level in tokens and FIFO buffer occupancy in packets to determine a marking probability. Then, if the incoming packet does not have the CE bit set, the router will set the bit with that marking probability. The marking probability increases linearly from 0 to 1, as the difference of token bucket levels and buffer occupancy increases from 0 to a configured threshold, *tb_threshold*. When the difference is larger than *tb_threshold*, the CE bit is always set to the incoming packet. (As noted in Section 7, the CE bit can also be generated by RED-like algorithms without using the token bucket.)

The receiver can adjust the delay that is imposed on ACK packets based on the percentage of received packets that have the CE bit set. The receiver estimates the round-trip time and uses it as a period over which the percentage is computed. We use two parameters, ? and ?, to denote the high and low thresholds, respectively. These thresholds are used to increase or decrease delay. For example, for ? = 0.9 and ??? 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 a period of time are set with the CE bit, the receiver will decrease the delay.

To adjust the delay, the receiver uses an additive decrease and multiplicative increase strategy analogous to that used in the TCP congestion window update algorithm [3,9]. Initially, the receiver imposes no delay for ACK packets. When the observed percentage of packets with CE bit set is higher than the threshold ?, the delay will be set to be the estimated round-trip time. The observation period is then updated to be the sum of the estimated round-trip time and the new delay. The receiver continues to observe packets and calculate the percentage for the next period of time. If the observed percentage is higher than the threshold ?, the delay will be doubled. That is, $D_{n+1} = 2 ? D_n$, 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 ?, the delay will be reduced according to the following equation:

$$D_{n+1} = ? \frac{CWND}{CWND + 1} ? ? D_n \qquad (3)$$

As shown in Equation(3), the decrease in the delay is inversely proportional to the congestion window size (CWND) plus 1. Thus, our delay update follows the principle of "Additive Increase, Multiplicative Decrease" (AIMD) [3].

For methods that the TCP receiver can use to estimate both the round-trip time and congestion window size, see [18]. Section 3 describes the particular methods we used for the simulation implementation of this paper.

3. SIMPLE SIMULATION FOR RDC

To study the basic properties of RDC, we have performed *ns* [25] simulations for a simple network configuration. Figure 2 depicts the configuration, which is based on one of the configurations in [6]. Two flows, originating from two sources with 100Mbps links and having the same end-to-end round-trip delay of 42ms, compete for the bottleneck link with a bandwidth of 45Mbps. The gateway **5** has a buffer of 140 packets. The maximum window size of both

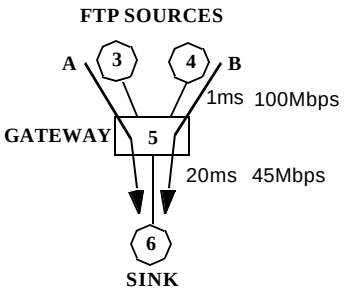


Figure 2: The configuration for a simple network

flows is set to be 240 packets (packet size is fixed at 1,000 bytes), 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 (CWND) could reach the maximum window size of 240 packets. This is because some packets can be queued at the router, gateway 5. 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 0 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 methods: DropTail, ECN, exact RDC and 1-bit RDC. We run simulations with their queue management algorithms implemented in the gateway and corresponding setups implemented in senders and receivers. For RED, the parameters used in ECN gateway are *min_th* = 40, *max_th* = 120, *w* = 0.002, *max_p* = 0.1 and *gentle* = true. For ECN, senders and receivers process CE bit following the proposal in [19]. For both exact RDC and 1-bit RDC, *tb_size* is set to be 6400 packets, and drain ratio ? equals to 0.9. For 1-bit RDC, the thresholds ? and ??are 0.9 and 0.1, respectively. The value of *tb_threshold* is set to be 500.

For 1-bit RDC, we also need to estimate both round-trip time and CWND for each flow. For round-trip time, we use the interval between the arrival times of the first two data packets as an estimate. For CWND, we use the TCP sender's maximum window size as an estimate. Other methods, such as using the timestamp option for round-trip time as mentioned in [18], could be used to provide a more accurate estimation. We note that the use of maximum window size makes 1-bit RDC less aggressive, due to the reduced rate in decreasing delays based on Equation(3).

Figure 3 shows the number of packets sent by each flow in 84ms periods. (We note that 84ms is twice of the 42ms 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 adjusts the sending rate with a

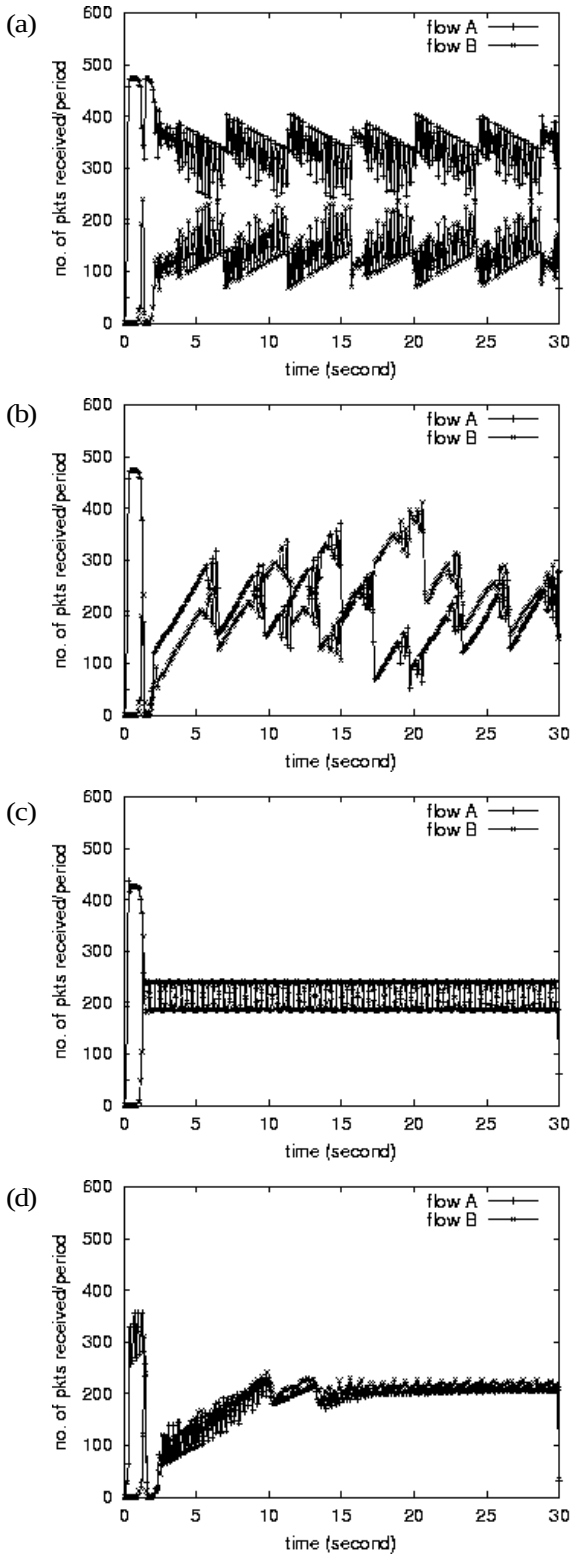


Figure 3: Number of packets received at the router in a 84ms period over time for configurations of (a) DropTail, (b) ECN, (c) exact RDC, and (d) 1-bit RDC. (These are simulation results for the configuration of Figure 2.)

continuous function by controlling the delay to be imposed on the ACKing of data packets by the TCP receiver. 1-bit RDC approximates the behavior of exact RDC.

Moreover, as shown in the figure, the second flow in DropTail, ECN, and 1-bit RDC suffers from at least one time-out 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. As a result, some packets are dropped. Also in ECN, due to its use of average queue occupancy, packets from both flows are dropped, and this results in traffic phase effect [5]. Exact RDC does not seem to exhibit phase effect, although 1-bit RDC sometimes does.

From Figure 3 (c) and (d), we see that exact RDC and 1-bit RDC can transport data with a relatively steady rate. In addition, we expect that they will experience only few or no time-outs, 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 and the design and implementation of the streaming application. In addition to providing maximum available bandwidth for video delivery while remaining friendly to other flows, our goals also include minimizing the playback latency.

4.1. Hierarchically-Encoded Layered Video

We use a simple model for hierarchically-encoded layered video, [13,16,17,24,26]. 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 to deliver the video. Videos are encoded offline, with each layer stored separately.

4.2. The Use of TCP as the Transport Protocol

An advantage of using traditional 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 easy to use traditional TCP as is for streaming purposes, because it is designed for reliable data communication, not for real-time applications.

Another major obstacle for using TCP in streaming applications, in addition to bandwidth variation and delay jitter, is retransmission time-outs. TCP will cease transmission and wait for a retransmission time-out to expire, if sufficiently many packets are lost. Retransmission time-out can take seconds to expire, and this can stop video playback for seconds. A solution is for the receiver to buffer a large amount of data before it can start playback. However, this can increase the playback latency significantly. When time-outs happen frequently, even the buffering cannot help. TCP RDC addresses these obstacles as discussed in the previous sections.

4.3. Video Source Streaming Algorithm

When streaming video over a TCP RDC connection, the multi-layer video source decides dynamically when to add or drop a layer of encoded video. The video source's decision of adding or dropping a layer will be based on the observed occupancy of the TCP sending buffer.

Because network paths to receivers may experience different network conditions, the source needs to determine the highest layer N the network will allow for a given stream at any time. We say a streaming is at layer N if the source decides it is appropriate to send video of N layers to the stream's receiver. Different streams may use different values of N at a given time, depending on their network condition. For each stream, the video source will monitor the TCP's sending buffer to detect network condition.

We extend 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 changes whenever the application writes data to the agent or the agent sends a data segment to the receiving agent. 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 application can read both variables. Although this is only done in the simulator at present, we believe it could be easily implemented in real-world systems.

The source application is implemented as follows. The application is executed periodically when streaming video. If the TCP maximum segment size is *MSS* bytes, and the bandwidth requirement of each layer is B bytes/second, then the period is set to be MSS / 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.

Figure 4 illustrates sending buffer occupancy over time. The buffer occupancy decreases in the beginning, because the network can

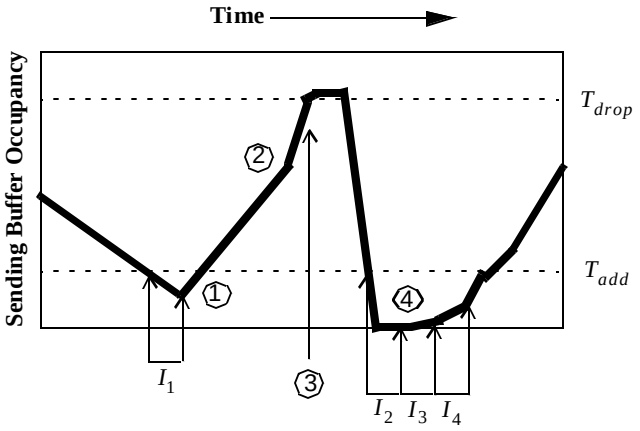


Figure 4: Sending buffer occupancy over time. The two horizontal dashed lines are T_{add} and T_{drop}. The solid line is the buffer occupancy.

provide higher bandwidth than the streaming requires. At point 1, because all the observed 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 many packets are dropped, so the streaming over the network is stopped because of retransmission time-out. 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 time-out to expire, some additional layers are dropped from the stream. When the time-out eventually expires and the TCP connection's congestion window opens up again, the network 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 0, 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 users.

5. SIMULATIONS OF VIDEO STREAMING

In this section, we run two sets of simulations to demonstrate the performance of RDC connections in streaming video. One set has 10 streaming connections, while the other has 100 ones. 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.

5.1. Simulation Setup

Both sets of simulation use the network configuration depicted in Figure 5, a configuration also used in [20]. In this network, the bottleneck link is the central link connecting routers *G0* and *G1*. *G0* and *G1* have side links connecting to sender nodes (*Si*) and receiver nodes (*Ri*), respectively. For each *i*, there is a TCP connection from node *Si* to *Ri*. The variable *n* is the number of streaming connections, so it is either 10 or 100. The bottleneck bandwidth is set to 0.64 ? *n* Mbps for reasons to be explained below. Both *G0* and *G1* have a FIFO buffer of 100 or 200 packets for the 10 or 100 connection setup, respectively.

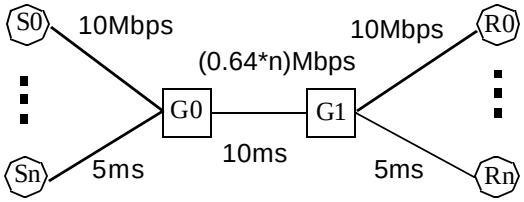


Figure 5: Network configuration for video simulations

Our video data is multi-layer encoded as described in Section 4, with each layer requiring 20KByte/s for delivery. Each *Si* is a source and delivers a stream to the corresponding *Ri*. For a 4-layer stream, a total of 80KByte/s, or 0.64Mbps, is required. Thus, to achieve the best streaming quality for all connections and maintain fairness, the source should continuously stream 4 layers of video on each connection. The period that the streaming application uses is 50ms. The threshold T_{add} is set to 0 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 methods, 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, 33.33, respectively, for 10-stream and 10-stream simulations), *max_th* equal to 1?2 of the FIFO buffer size (50 and 100, respectively), *w* = 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, *tb_threshold* is increased from 500 to 5,000.

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 synchronizations among flows, we start all flows randomly in the first 20 seconds. We run the simulation for 800 seconds.

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 *G0* to *G1* are kept low in both cases.

In Figure 6, results on achieved throughput in number of layers are presented. For each connection, the change in the 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 Figure 6 (a) exhibits severe unfairness in the number of layers for different streams, and ECN in Figure 6 (b) also exhibits some fluctuation and unfairness.

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 ? = 0.9 in the simulation, the average number of layers for both of them are lower than that of ECN's. The utilization will be increased accordingly if a larger value of ? such as ? = 0.95 is used.

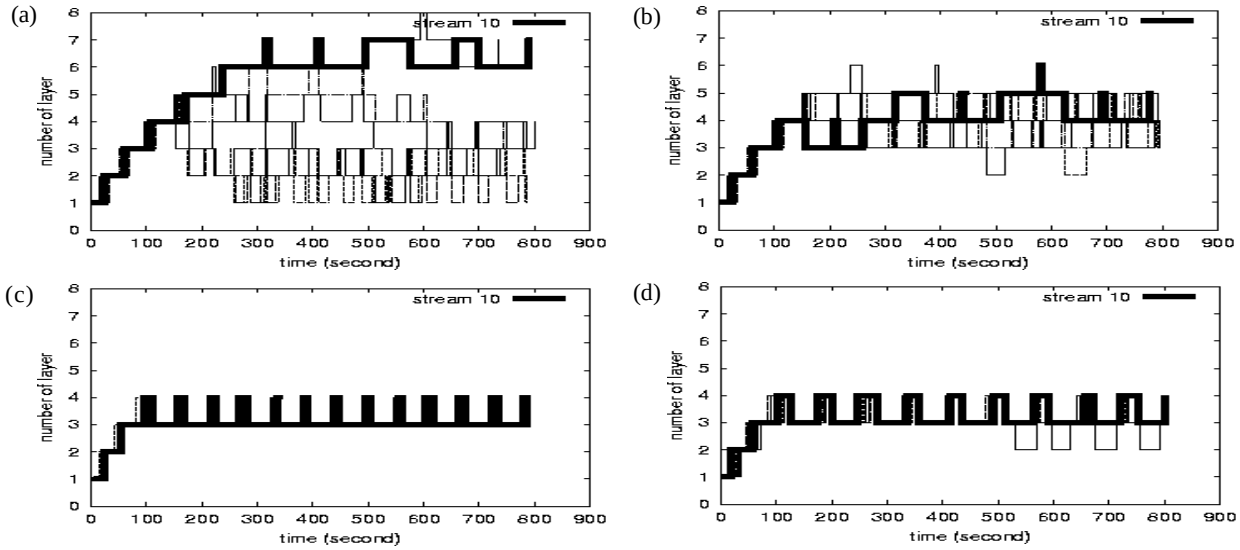


Figure 6: 10-stream simulation. Number of layers for each connection over time for (a) DropTail, (b) ECN, (c) exact RDC, and (d) 1-bit RDC. Only flow 10 is highlighted with a thick solid line, while the others are in thin dashed lines.

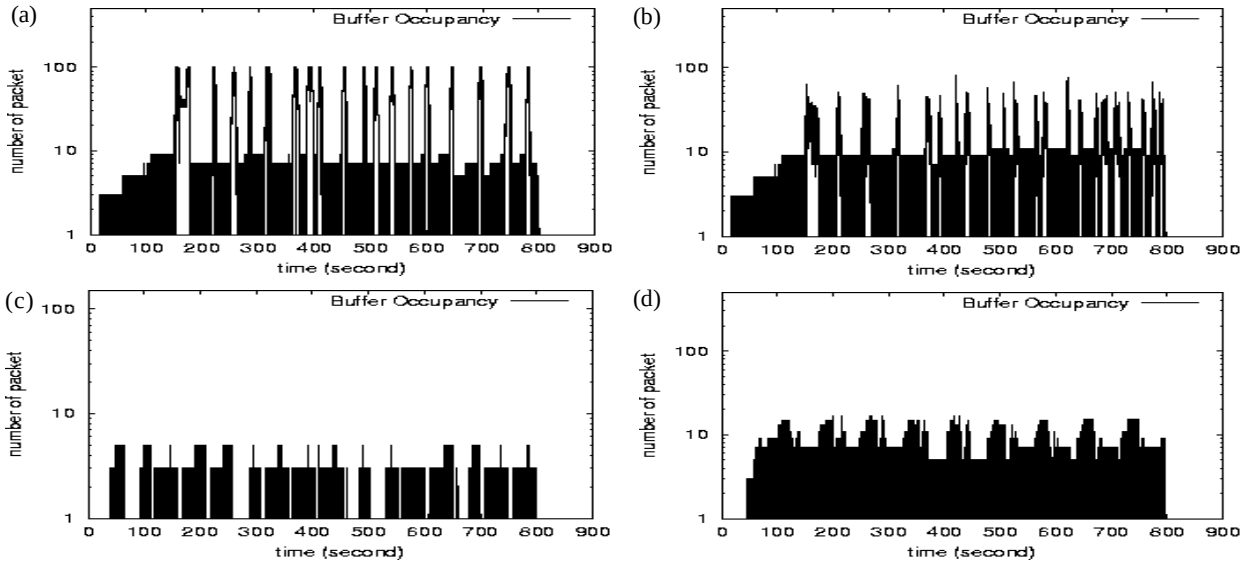


Figure 7: 10-stream simulation. Sampled FIFO buffer occupancy of the outgoing link from *G0* to *G1* for (a) DropTail, (b) ECN, (c) exact RDC, and (d) 1-bit RDC. Y-axis of the figures are in the log scale.

In Figure 7, results on the FIFO buffer occupancy are presented. The buffer occupancy is sampled at every 50 FIFO buffer enqueue or dequeue events. The results show that exact RDC (Figure 7 (c)) has the lowest buffer occupancy of only a few packets. 1-bit RDC (Figure 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 time-outs, while both ECN and 1-bit RDC exhibit only a few packet drops and time-outs. Exact RDC does not have any packet loss or time-out at all.

5.3. 100-Stream Simulation

In this subsection, we increase the number of streams from 10 to 100. We also increase *G0*'s FIFO buffer size from 100 to 200, and adjust RED's parameters accordingly. The parameters for 1-bit RDC remains 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. It is instructive to compare the performance between sender-based congestion control used by ECN and receiver-based delay control used by 1-bit RDC. The results presented in this subsection demonstrate that 1-bit RDC performs well even when the number of streams increases from 10 to 100, while the performance of ECN degrades.

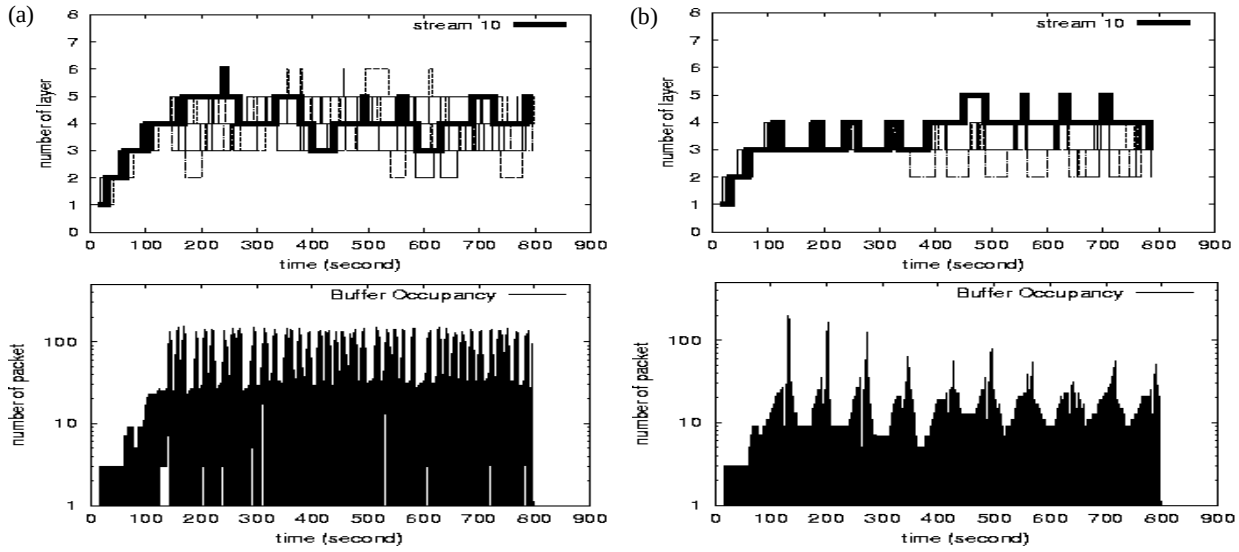


Figure 8: 100-stream simulation. The first row is the number of layers for each connection over time for (a) ECN, and (b) 1-bit RDC. 10 connections are randomly selected for presentation, but only one is highlighted with a thick solid line. The second row is the sampled FIFO buffer occupancy of outgoing link from *G0* to *G1* over time.

Figure 8 examines performance results of 10 randomly selected streams among the 100 simulated streams. In the first row of Figure 8, we show the number of layers for each of the 10 streams over time. We also present the sampled FIFO buffer occupancy in the second row of the figure. The left column contains the results for ECN, while the right column contains those for 1-bit RDC. 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 time-outs. Table 1 shows the packet loss rate and the total number of time-outs during the lifetime of the 100 connections for both the ECN and 1-bit RDC cases. ECN exhibits significantly more time-outs than 1-bit RDC.

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

Table 1 : Packet lost rate and number of time-outs 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)*
# Time-outs	675	2

Exact RDC generally outperforms ECN and 1-bit RDC. However, as noted in Section 2.3 for easy implementation and deployment, 1-bit RDC, which is an approximation of RDC, can be more attractive than exact RDC.

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 Figures 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 Figure 8 and Table1). This is because the RDC approach can reduce the number of time-outs as explained in Section 2.2.1.

6. RELATED WORK AND DISCUSSION

To support RDC, a router manages 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) [2] as opposed to scheduling algorithm [4] 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 [12].

ECN [7] is a framework for sending explicit congestion signal from routers to TCP senders and receivers. Routers with AQM, such as RED, implemented as their queue management algorithm detect persistent high buffer occupancy and notify the senders of congestion with a signal. This signal can be carried out by forward signals such as the CE bit in the IP packet. In response to the presence of congestion signal, senders reduce their sending rates by half. In contrast, routers in the RDC framework notify the receivers the need for delaying the ACKing of data packets. Receivers delay ACK packets accordingly, and can reduce senders' sending rates by increasing the ACKing delay.

ECN and RDC use different approaches in dealing with congestion. Under ECN, TCP senders adjust congestion window size to avoid congestion; under RDC, TCP receivers control the delay of ACK packets. Other receiver-based congestion control methods manage advertised window size, such as those discussed in [22].

Kanakia [11] 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 they used is optimized for video quality and does not consider the friendly requirement when there are other competing streams. In this paper, we consider not only the problem of optimizing video playback quality, which is achieved by reduced time-outs, steady delivery rate and layered encoding, but also the interactions with other competing streams.

7. 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 uses 1-bit congestion notifications generated by routers.

For 1-bit RDC, accurate delays are actually not important for setting the Congestion Experienced (CE) bit in a packet header. A randomized algorithm for detecting congestion, such as RED, can generally achieve the same effect. We have preliminary results that validate this claim. In a future paper, we plan to demonstrate the use of RED-like methods in generating CE bits, without using the token bucket method of Section 2.3.

In this paper, because we wanted to focus on comparing TCP RDC with other schemes such as ECN and traditional TCP, we did not address their interaction when they co-exist in the same network. However, our empirical results have shown that when competing with 1-bit RDC flows, traditional TCP flows can usually gain more bandwidth than RDC flows, because they don't impose delays to ACK packets. It is our belief that by assuming a properly randomized AQM scheme in routers, such as RED, and by tuning the delay control at the TCP receiver, 1-bit RDC will be able to handle this unfairness problem.

In addition to RDC, in this paper 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 TCP RDC connections.

Thus the combination of our TCP RDC and layered video source algorithms offers an attractive approach to streaming video over IP networks. The approach can provide maximum available bandwidth for video delivery, while being friendly to other competing streams and minimizing playback latency.

Although it is not clear how RDC can become part of current standards, the proposed framework does provide a new direction for the design of future TCP-like transport protocols suitable for video streaming.

8. ACKNOWLEDGMENTS

This research was supported in part by NSF grant ANI-9710567, and in part by a grant from Sun Microsystems.

9. REFERENCE

[1] Allman, M., Paxson, V., and Stevens, W. R. "TCP congestion control," RFC 2581, April 1999.

[2] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., Zhang, L. "Recommendations on queue management and congestion avoidance in the Internet," RFC 2309, April 1998.

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

[4] Demers, A., Keshav, S., and Shenker, S. "Analysis and simulation of a fair queueing algorithm," Internetworking: Research and Experience, Volume 1, 1990, p. 3-26.

[5] Floyd, S., and Jacobson, V. "Traffic phase effects in packet-switched gateways," Computer Communication Review, V.21 N.2, April 1991.

[6] Floyd, S., and Jacobson, V. "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, p. 397-413.

[7] Floyd, S. "TCP and explicit congestion notification," ACM Computer Communication Review, V. 24 N. 5, October 1994, p. 10-23.

[8] Floyd, S., Handley, M., Padhye, J., and Widmer, J. "Equation-based congestion control for unicast applications," In Proceedings of the 2000 ACM SIGCOMM Annual Technical Conference, August 2000.

[9] Jacobson, V. "Congestion avoidance and control," Computer Communication Review, 18(4), 1988, p. 314-329.

[10] Jain, R. "Congestion control in computer networks: issues and trends," IEEE Network, 4(3), May 1990, p. 24-30.

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

[12] Kunniyur, S., and Srikant, R. "Analysis and design of an adaptive virtual queue algorithm for active queue management," To appear in the Proceedings of ACM SIGCOMM 2001.

[13] Lee, J.-Y., Kim, T.-H., and Ko, S.-J. "Motion prediction based on temporal layering for layered video coding," In Proceedings of the ITC-CSCC, 1:245-248, July 1998.

[14] Lin, D., and Kung, H. T. "TCP fast recovery strategies: analysis and improvements," In Proceedings of the IEEE INFOCOM'98, April 1998.

[15] Mahdavi, J., and Floyd, S. "TCP-friendly unicast rate-based flow control," Note sent to end2end-interest mailing list, January 1997.

[16] McCanne, S., and Vetterli, M. "Joint source/channel coding for multicast packet video," In Proceedings of the IEEE International Conference on Image Processing, October 1995, p. 776-785.

[17] McCanne, S. "Scalable compression and transmission of Internet multicast video," Ph.D. thesis, University of California Berkeley, UCB/CSD-96-928, December 1996.

[18] Ming-Chit, I.T., Jinsong, D., and Wang, W. “Improving TCP performance over asymmetric networks,” ACM Communication Computer Review, July 2000.

[19] Ramakrishnan, K.K., and Floyd, S. “A Proposal to add explicit congestion notification (ECN) to IP,” RFC 2481.

[20] Rejaie, R., Handley, M., and Estrin, D. “Quality adaptation for unicast audio and video,” In Proceedings of the 1999 ACM SIGCOMM Annual Technical Conference, September 1999.

[21] Savage, S., Wetherall, D., Karlin, A., and Anderson, T. “Practical network support for IP traceback,” In Proceedings of the 2000 ACM SIGCOMM Annual Technical Conference, August 2000, p. 295-306.

[22] Spring, N. T., Chesire, M., Berryman, M., Sahasranaman, V., Anderson, T., Bershad, B. “Receiver based management of low bandwidth access links,” In Proceedings of the IEEE INFOCOM 2000, March 2000.

[23] Stevens, W. R. “TCP/IP Illustrated, Volume 1: The Protocols,” Addison-Wesley, Reading, MA, USA, 1994.

[24] Turletti, T., Fosse-Parisis, S., and Bolot, J. “Experiments with a layered transmission scheme over the Internet,” Research report, INRIA, B.P.93, Sophia-Antipolis Cedex, France, November 1997.

[25] UCB/LBL/VINT. “Network simulator — ns,” http://www.isi.edu/nsnam/ns/.

[26] Vishwanath, M., and Chou, P. “An efficient algorithm for hierarchical compression of video,” In Proceedings of the IEEE International Conference on Image Processing, November 1994.