Multiple Sender Distributed Video Streaming

Thinh Nguyen and Avideh Zakhor

Department of Electrical Engineering and Computer Sciences

University of California, Berkeley, CA 94720

{thinhq, avz}@eecs.berkeley.edu

Abstract

With the explosive growth of video applications over the Internet, many approaches have been proposed to stream video effectively over packet switched, best-effort networks. In this paper, we propose a receiver-driven protocol for simultaneous video streaming from multiple senders to a single receiver in order to achieve higher throughput, and to increase tolerance to packet loss and delay due to network congestion. Our receiver-driven protocol employs a novel rate allocation scheme and packet partition algorithm. The rate allocation scheme, run at the receiver, determines the sending rate for each sender by taking into account available network bandwidth, channel characteristics, and a prespecified, fixed level of forward error correction, in such a way as to minimize the probability of packet loss. The packet partition algorithm, run at the senders based on a set of parameters estimated by the receiver, ensures that every packet is sent by one and only one sender, and at the same time, minimizes the startup delay. Using both simulations and Internet experiments, we demonstrate the effectiveness of our protocol in reducing packet loss.

To appear in IEEE Transactions on Multimedia April 2004

List of Figures

List of Tables

I. Introduction

Video streaming over best-effort, packet-switched networks is challenging due to a number of factors such as high bit rates, delay, and loss sensitivity. As such, transport protocols such as TCP are not suitable for streaming applications. To this end, many solutions have been proposed from different perspectives. From source coding perspective, layered and error-resilient video coding have been proposed. A layered video codec deals with heterogeneity and time-varying nature of the Internet by adapting its bit rate to the available bandwidth [1]. An error-resilient codec modifies the bit stream in such a way that the decoded video degrades more gracefully in lossy environments [1][2][3]. From channel coding perspective, Forward Error Correction (FEC) techniques have been proposed to increase reliability at the expense of bandwidth expansion [4][5][6][7]. From the protocol perspective, there are approaches based on multicast [5][8] and TCP-friendly protocols [9] for streaming multimedia data over the Internet. Multicast reduces the network bandwidth by not sending duplicate packets on the same physical link [8]; however, it is only appropriate for situations with one sender and many receivers. Meanwhile, TCP-friendly protocols use rate-based control to compete fairly with other TCP traffic for bandwidth, and at the same time, stabilize the throughput, thus reducing the jitter for multimedia streaming [10]. From networking perspective, content delivery network (CDN) companies such as Akamai employ edge architecture by pushing content to the edge of the network, and by strategically placing servers at the edge of the Internet in such a way that each client can choose the server resulting in shortest round-trip time and least amount of congestion.

An alternative to edge streaming for providing smooth video delivery, is to divide video among multiple streaming senders in order to effectively provide the required throughput [9]. Having multiple senders is in essence a diversification scheme in that it combats unpredictability of congestion in the Internet. Specifically, we assume independent routes from various senders to the receiver, and argue that the chances of all routes experiencing bursty packet loss at the same time is quite small. In addition if the route between a particular sender and the receiver experiences congestion during streaming, the receiver can re-distribute rates among the existing senders, or recruit new senders so as to minimize the effective loss rate.

In this paper, we propose a framework for streaming video from multiple mirror sites simultaneously to a single receiver in order to achieve higher throughput, and to increase tolerance to loss and delay due to network congestion. Our solution combines approaches from different perspectives including system architecture and transport protocols. From the systems perspective, we expand the edge architecture to allow simultaneous video streaming from multiple senders as shown in Figure 1. This is in contrast with the edge architecture where only one server is responsible for streaming video to its nearest clients. From protocol perspective, we use a receiver-driven protocol to coordinate simultaneous transmission of video from multiple mirror sites to a single receiver effectively. Our proposed distributed video streaming protocol, first introduced

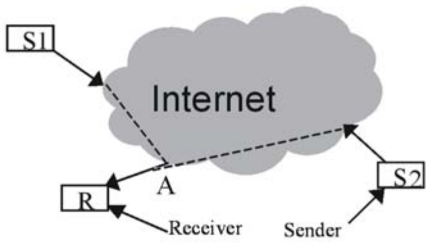


Fig. 1. *Distributed video streaming architecture.*

in [9][11], consists of rate allocation and a packet partition algorithms. The rate allocation scheme, run at the receiver, is used in conjunction with FEC to minimize the probability of packet loss in bursty channel environments by splitting the sending rates appropriately across the senders. The packet partition algorithm, run at individual senders based on a set of parameters estimated at the receiver, ensures that every packet is sent by one and only one sender, and at the same time, minimizes the startup delay. Senders and receiver communicate with each other through control packets that contain the required information to facilitate the rate allocation and packet partition algorithms.

In this paper, we employ FEC as part of our proposed distributed streaming protocol. A well known drawback of FEC though is that it results in bandwidth expansion, and hence reduces the amount of available bandwidth for the actual video bit stream. Since the level and burstiness of packet loss in the Internet fluctuates significantly, incorporating the optimal amount of FEC in any single route streaming application is a difficult task; too little redundancy cannot effectively protect the video bit stream, and too much redundancy consumes too much bandwidth unnecessarily. Thus for single route streaming, FEC level has to be closely matched to channel characteristics for it to be effective. In this paper, we show that FEC in a multiple sender scenario can combat bursty loss behavior in the Internet more effectively than in the single sender case. Specifically the above sensitivity mismatch between FEC level and network characteristics for single route streaming applications is reduced as compared with multi route streaming.

A. Related Work

Many diversity schemes have been proposed in wireless literature, ranging from frequency and time, to spatial diversity [12]. In wired networks, path diversity was first proposed in [13], and the theoretical work on information dispersion for security and load balancing was proposed in [14]. Recently, there have been other works dealing with simultaneous downloading of data from multiple mirror sites. If the data is not delay sensitive, it is possible to use multiple TCP connections to different sites, with each TCP connection downloading a different part of the data. For example, the authors in [15][16] use Tornado codes to download data simultaneously from multiple mirror sites. More recently, Digital Fountain has used an advanced class of linear-time codes to enable the receivers to receive any N linear-time coded packets from different senders, so as to recover N original data packets. There has also been a study on the throughput of TCP connection using multiple paths in [17]. In [18], the authors propose *CoopNet*, a tree structure for delivering video to the receivers from multiple servers. Also Peer-to-Peer file sharing system such as *Kazaa* allows downloading the media files simultaneously from multiple participating members. Another Peer-to-Peer system with adaptive layered streaming has also been proposed in [19]. In addition, multiple description coding of video (MDC) has been studied in detail in [20][21][22] [23][24]. In this approach, video source is partitioned into multiple descriptions, each one assumed to be sent along a different channel. The assumption is that visual quality of the video degrades gracefully as the number of received descriptions decrease due to channel impairments. The performance gain of using multiple servers together with MDC over traditional single server approach is compared and analyzed in [25]. Recently, rate distortion optimization in the framework of path diversity has been explored in [26]. Finally, authors in [27] have shown substantial improvement for real-time voice communication over the Internet using path diversity together with MDC, and sophisticated playback schedule.

B. Assumptions

To successfully stream video from multiple senders, we assume that the bandwidth bottleneck is not at the last hop. In streaming situations where the bottleneck is in the last hop, e.g. due to the physical bandwidth limitation of dial-up modem, our proposed distributed streaming approach is of little use. Indeed if a client is connected to the Internet through a low bandwidth connection, it is preferable to download the entire video in a non-real time fashion before playing it back. Our premise is that, asymptotically, as broadband connections to the Internet such as DSL become prevalent, the limiting factor in streaming is packet loss and delay due to congestion along the streaming path, rather than the physical bandwidth limitations of the last hop. Also, studies in [28] indicate that at least 60% of the Autonomous System (AS) are *multi-homed* to two or more providers, i.e. sufficiently disjoint paths to the receiver can be created using senders in different providers. Resilient Overlay Network (RON) [29] also demonstrates the existence of many sufficiently disjoint paths between two nodes on the Internet.

We now list additional assumptions about our distributed streaming framework:

1) The total needed video rate, S, is available from the aggregate bandwidth of all the senders. We make this assumption for sake of simplicity; If this assumption is not satisfied, either scalable video can be used to reduce the bit rate, or additional senders can be recruited.
2) The amount of FEC is assumed to be fixed for a given streaming session.
3) Video is pre-coded at a fixed rate, and residing at all servers.
4) Average packet loss rate over long term is independent of the instantaneous sending rate. i.e. there is no self congestion, and packet loss is only a result of cross traffic. This is a direct consequence of our network model, i.e. two-state continuous Markov chain to be described shortly. This assumption holds if the video bit rate is only a small part of the total traffic, i.e. there is a large degree of statistical multiplexing. Since speed of routers on the Internet are on the order of hundreds of megabits to tens of gigabits per second, and typical streamed video is less than 1Mbs, we believe this assumption is justified in practice.

5) Packet loss between two routes are independent. We make this assumption to simplify rate allocation analysis. In general, the distributed streaming framework, provides benefits over traditional uni-sender scheme even if packet loss correlation between routes is nonzero.

Our paper is organized as follows. In Section I-C, we provide an overall framework for distributed video streaming, and describe the proposed transport protocol. Next, in Section II we describe and analyze the proposed rate allocation scheme to be used with FEC. The goal of this algorithm is to split the sending rates appropriately across the senders in order to minimize the packet loss, taking into account the network conditions and a fixed amount of FEC. Beyond knowing the sending rate, each sender still needs to know which packet to send in order to avoid duplicate packets and minimize startup delay. We propose a packet partition algorithm in Section III to address this issue. Next, we present numerical and experimental results in Section IV. In Section V, we address the limitations of our currently proposed system, and propose possible extensions. Finally, we conclude in Section VI.

C. System Overview

Our transport protocol is a receiver-driven one in which, the receiver coordinates transmissions from multiple senders based on the information received from the senders. A high level description of our distributed streaming framework is shown in Figure 2. Each sender estimates and sends its round trip time (RTT) to the receiver. The receiver uses the estimated round trip times and its own estimates of senders' loss rates to calculate the optimal sending rate for each sender using the Rate Allocation Algorithm, in such a way as to minimize the overall probability of packet loss. The receiver monitors variations in route conditions of each sender in order to readjust rate distribution among senders. When the receiver decides to change any of the senders' sending rates, it sends identical control packets to all senders. The control packet contains the synchronization sequence number, each sender's RTT to the receiver, and the optimal sending rates as calculated by the receiver for all senders. Upon receiving the control packets, each sender sends UDP packets at the rate specified in the control packet. Each sender also runs a distributed packet partition algorithm to determine the sender for each packet in such a way that (a) every packet is guaranteed to be sent by one and only one sender, and (b) startup delay at the receiver is minimized. Before discussing

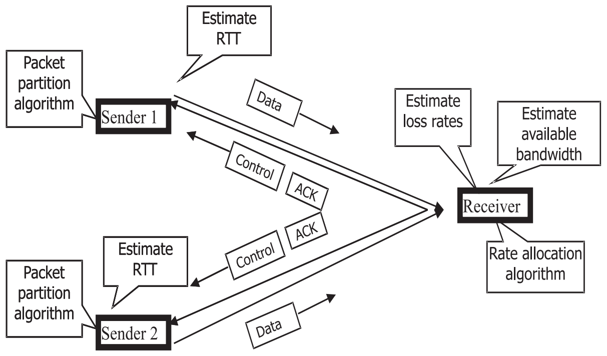

Estimate RTT
Packet partition algorithm
Sender 1
Data
Control
ACK
Estimate loss rates
Estimate available bandwidth
Receiver
Rate allocation algorithm
Packet partition algorithm
Estimate RTT
Sender 2
Control
ACK
Data

Fig. 2. *High level description of distributed streaming framework.*

the rate allocation and packet partition algorithms in detail, we state their objectives: *The goal of the rate allocation algorithm is to determine how to split the total rate S between M senders in order to minimize the probability of packet loss. The goal of the packet partition algorithm is to determine which packets should be sent by which senders in order to prevent duplicate packets and to minimize the startup delay.*

II. Rate Allocation Algorithm

In this section, we propose a rate allocation algorithm to be used as part of our distributed streaming protocol. The rate allocation algorithm is run at the receiver in order to determine the optimal streaming rate for each sender. We show that our optimal rate allocation scheme for distributed streaming significantly reduces the packet loss as compared to single route streaming. Before describing the rate allocation algorithm in detail, we argue intuitively as to why splitting packets across routes can reduce packet loss using FEC. First, over short time scales, sending packets at higher rate during a congestion period results in larger number of lost packets during that period than sending at a lower sending rate. Therefore, by splitting packets appropriately across two routes and lowering the sending rates on each route, one would expect to lower packet burst loss on each route. Second, FEC is more effective in non-bursty environments, i.e. in absence of large number of lost packets within a FEC block. Hence, splitting packets across routes reduces bursty packet loss, and increases the probability of successful recovery of the lost packets by FEC. Third, assuming congestion intervals of two independent routes rarely coincide, use of FEC allows one to recover packets lost on one route using the received packets on the other route.

A. *Optimal Rate Allocation*

We model packet loss on the Internet as a simple two-state continuous Markov chain which has been shown to approximate the packet loss behavior fairly accurately[30][31][32][33]. A two-state continuous Markov chain with state at time t denoted by X_t with $X_t \in \{g, b\}$, is characterized by μ_g and μ_b where subscript g stands for "good" and b for "bad". μ_g and μ_b can be thought of as rates at which the chain changes from 'good" to "bad" state and vice versa. When the chain is in good state, the probability of packet loss is much lower than when the chain is in bad state.

To simplify analysis, we replace the two-state continuous-time Markov chain with an equivalent two-state discrete one. Since we are only interested at the instances when a packet is sent, the transition probabilities for the discrete one denoted by p_{gg}, p_{gb}, p_{bb}, and p_{bg}, can be easily expressed as a function μ_g, μ_b, and τ, where τ is the sending interval [30]. At sending rate $S = 1/\tau$, we obtain the following transition probability matrix for an equivalent two-state discrete Markov chain: $\begin{pmatrix} p_{gg} & p_{gb} \\ p_{bg} & p_{bb} \end{pmatrix}$. With the discrete model, the discrete time step corresponds to the event of sending a packet. The process of the discrete Markov chain undergoing n discrete time steps is equivalent to the process of sending n packets through the network. To further simplify analysis, we only consider the case of two senders A and B, both assumed to be sending packets to the receiver along two routes with independent packet loss. The extension of analysis to the case with more than two senders is straightforward.

Our goal is to find the sending rates for the two senders in order to (a) minimize the probability of irrecoverable loss for a fixed level of FEC, and (b) to ensure that each sender sends packets only at available bandwidth so as not to compete unfairly with existing TCP traffic. To formally state our rate allocation problem, we use the notation in Table I. The rate allocation problem can now be stated as follows:

Given S, N, K, B_m, (μ_g^m, μ_b^m), we want to find N_m for $m \in \{A, B\}$ so as to minimize the probability of irrecoverable loss given by

$$C(K, N_0, N_1) = \sum_{j=N-K+1}^{N_A+N_B} \sum_{i=0}^{j} P(A, i, N_A) P(B, j-i, N_B) \tag{1}$$

N	Total number of packets in a FEC block
K	Number of data packets in a FEC block
B_m	Estimated available bandwidth for sender m in packets per second
(μ_g^m, μ_b^m)	Network parameters for route between sender m and the receiver
S	Aggregate bit rate of video and FEC in packets per second
$\lambda = \frac{N}{S}$	Interval between successive transmitted FEC blocks in seconds
N_m	Number of packets transmitted by sender m during λ seconds

TABLE I

Notations for rate allocation algorithm.

subject to

$$N_A + N_B = N, \quad \tfrac{N_A}{\lambda} \leq B_A, \quad \tfrac{N_B}{\lambda} \leq B_B \tag{2}$$

where $P(m, i, N_m)$ denotes the probability that i packets are lost out of the N_m packets sent by sender m, and $C(K, N_A, N_B)$ denotes the probability that more than $N-K$ packets are lost out of a total N_A+N_B packets sent by both senders. Since we assume independent packet loss along the two routes, the probability of j lost packets out of $N_A + N_B$ packets sent by both senders can be written as $\sum_{i=0}^{j} P(A, i, N_A)P(B, j-i, N_B)$. Therefore, the probability of more than $N-K$ lost packets out of N_A+N_B packets sent by both senders, or equivalently the irrecoverable loss probability, is $\sum_{j=N-K+1}^{N_A+N_B} \sum_{i=0}^{j} P(A, i, N_A)P(B, j-i, N_B)$. As indicated in inequality constraints (2), $\frac{N_m}{\lambda}$ is the sending rate of sender m, which is required to be less than or equal to the available bandwidth. Since the sum of the sending rates equals to the required sending rate for the video, we have $N_A + N_B = N$. In this paper, we assume that sum of the available bandwidth of all senders is always greater than or equal to the video bit rate.

The procedure to compute the $P(m, i, N_m)$ is shown in the Appendix. Using $P(m, i, N_m)$, we search over all possible values of N_A and N_B such that the constraints (2) are satisfied, and $C(K, N_A, N_B)$, the probability of irrecoverable packet loss is minimized. This search is fast since for the case of two senders, only N comparisons are required. For M senders, the straightforward exhaustive search has complexity $O(N^{M-1})$. This could indeed be problematic for large values of N and M. However, we believe that from an implementation point of view, having more than 10 connections results in too large of an overhead, and makes the coordination of the senders too difficult for distributed streaming to be feasible in practice.

III. Packet Partition Algorithm

A. Basic Description of PPA

In this section, we address the issue of packet selection for each sender. As described in Section I-C, after receiving the control packet from the receiver, each sender immediately decides the next packet in the video stream to be sent, using the packet partition algorithm (PPA). All the senders simultaneously run this algorithm in a distributed fashion in order to ensure that, all packets are sent by one and only one sender, and also to minimize the startup delay.

To show the advantages of our proposed PPA over other PPAs, we first briefly describe a PPA in peer-to-peer file sharing systems such as *Kazaa*. *Kazaa* file sharing system allows a single member to download a media file simultaneously from multiple participating members. To decide which packets to be sent by which sender, each sender is assigned to send a contiguous block of data of length proportional to its sending rate. For example, suppose there are two senders, the allowable sending rates for the first and second senders are 100 and 80 packets per second respectively, and total playback rate is 180 packets per second. In this case, the first sender is assigned to send the first 100 packets and the second sender, the next 80 packets. Therefore, the receiver has to wait until the entire 180 packets are received before attempting to playback since its playback rate is larger than the sending rate of the first sender. Even though, this strategy avoids duplicating packets between senders, it incurs unnecessary startup delay.

The main objective of our PPA is to ensure that the received packets arrive in an interleaved fashion from multiple senders, so as to reduce the startup delay. The algorithm can be described as follows. Each sender receives a control packet from the receiver through a reliable protocol at the beginning of a session or whenever the receiver determines there should be a change in any of the sending rates. The control packet contains two-byte fields $S(1)$-$S(5)$ to specify the sending rate in packets per second for each sender, and one-byte fields $D(1)$-$D(5)$[1] for the estimated delay from each sender to the receiver in multiples of 2 milliseconds. The quantized value of 2 milliseconds is chosen to specify up to 512 milliseconds using only

[1]For simplicity, here we assume the total number of senders not to exceed 5. In practice any number of senders can be deployed as deemed feasible.

one byte, and at the same time, is accurate enough for most practical purposes. The control packet also contains $Sync$ number which is used as the starting sequence number that all senders use in the PPA to determine the next packet to send, immediately upon receiving the control packet. Note that here, we only list the essential information in the control packet in order to describe our packet partition algorithm in a broad sense.

To describe the PPA in detail, we use the notation in Table II. If the reference time, $T_{k'} = 0$, is

$k^{'}$	Sequence number $Sync$ in the control packet which all senders use to initialize the PPA
$T_{k'}$	Time at which control packet with sequence number sync $k^{'}$ is sent by the receiver
N	Number of senders
$P_{k'}(k)$	Playback time for k^{th} packet with respect to $T_{k\prime}$
$S(j)$	Sending rate for sender j
$\sigma(j)$	Sending interval between packets for sender j
$n_{j,k,k'}$	Number of packets already sent by sender j since packet $k^{'}$, and up to packet k
$D(j)$	Estimated delay from the sender j to the receiver

TABLE II

Notations for packet partition algorithm.

conceptually chosen to be the departure time of the control packet from the receiver, the estimated arrival time of the k^{th} packet sent by sender j is $n_{j,k,k'}\sigma(j) + 2D(j)$. This is because it takes $D(j)$ for the control packet to arrive at the sender j, $n_{j,k,k'}\sigma(j)$ for the k^{th} packet to be sent by sender j, and $D(j)$ for it to arrive at the receiver. Since $P_{k'}(k)$ is the playback time of the k^{th} packet with respect to $T_{k'}$, the expression $A_{k'}(j,k) = P_{k'}(k) - [n_{j,k,k'}\sigma(j) + 2D(j)]$ can be interpreted as the estimated time difference between arrival and playback time of the k^{th} packet, if sender j is its originator.

The basic idea in our packet partition algorithm is that among all senders $j = 1, ...N$, the one that maximizes $A_{k'}(j,k)$ is assigned to send k^{th} packet. Specifically, each sender computes $A_{k'}(j,k)$ for each packet k, for itself, and all other senders, and only sends k^{th} packet if it discovers that $A_{k'}(j,k)$ is at a maximum for itself. If $A_{k'}(i,k)$ is not at a maximum for sender i, it will increase k by one, and repeats the procedure until it finds the packet for which is at a maximum among all other senders. Note that if $A_{k'}(j,k)$ is positive, k^{th} packet is on time, otherwise, k^{th} packet is late. In a way, by assigning sender j to send the k^{th} packet with maximum $A_{k'}(j,k)$, we minimize the probability of k^{th} packet being late.

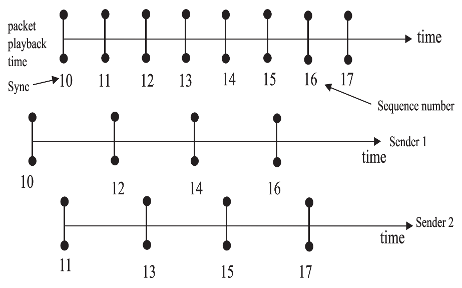


Fig. 3. *Illustration of packet partition algorithm.*

Each sender effectively keeps track of all the values of $A_{k'}(j, k)$ for all N senders, and updates every time a packet is sent. The values evolve in the same way at all senders even though they are computed at different locations. The reasons for this are that all the senders (a) receive the same control packet from the receiver, (b) only use the information in the control packet to update and (c) use the same equation to do so. Therefore, even though the actual delays from senders to the receiver might be different from what is reported in the control packet, as long as, all senders use the same information contained in the control packet, they all arrive at the same decision as to who sends what. There is also no need to synchronize all the senders'clocks to a global time, although they need to have more or less similar speed to prevent drift among senders.

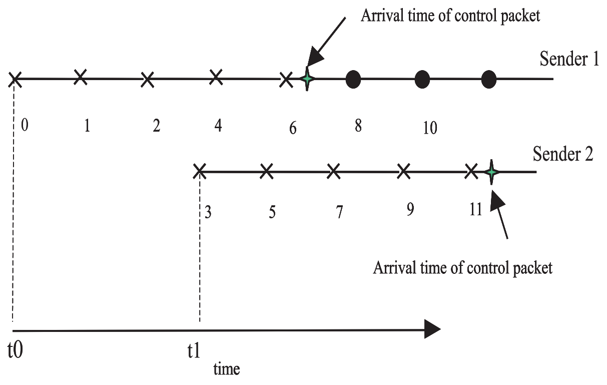
To illustrate the algorithm, we show a simple example in which, there are two senders, each sending packets at same rate. As shown in Figure 3, the $Sync$ sequence number is 10. The top line with vertical bars denotes the playback time for the packets. The middle and the bottom lines indicate the time to send packets for senders 1 and 2, respectively. In this scenario, packet 10 will be sent by sender 1 since the estimated difference between playback and its receive time for packet 10 is greater than that of sender 2. Next, packet 11 will be sent by sender 2 for the same reason.

In our implementation of a reliable protocol for the control packets, a batch of 5 identical control packets are sent with 5 milliseconds spacing between them whenever the receiver determines there should be a change in sending rates. Unlike TCP, our reliable protocol for control packets does not employ rate control or congestion avoidance mechanisms since the control packets are sent infrequently. If none of the control packets are acknowledged within two round-trip times from a particular sender, a new batch of control packets are sent to all the senders until the control packets are acknowledged by all senders. The interval of 5 milliseconds between the control packets is chosen in order to reduce potential loss of all 5 control packets due to burst loss, while ensuring that the control packets do not arrive at one particular sender too late. This value does not affect the overall performance of the system since in practice control packets are sent infrequently, e.g. on the order of once every several minutes, and the overall loss probability of all control packets is very small. Due to infrequent sending of control packets, our distributed protocol does not address the congestion control on a short time scale.

B. Practical Considerations with Packet Partition Algorithm

A practical consideration is the choice of the synchronization sequence number, k', in the control packet to signal new sending rates. In choosing k', one has to keep two objectives in mind. First the lag among the senders needs to be as little as possible so that received packets are in order as much as possible, thus minimizing required receiver buffer size. Second, we need to ensure that during rate transition, the aggregate bit rate for all senders remains constant, at the video rate, S, so as there is no or little buffer drainage at the receiver.

As an example, consider two sender scenario shown in Figure 4. The already sent packets by either sender are denoted by crosses, and the packets to be sent by circles. Suppose the receiver sends out control packets which arrive at sender one before sender two. This could be due to RTT difference between the two senders. Thus, by the time the control packet arrives, sender one's most recently sent packet is 6, and sender two's is 11. If k' is chosen to be 11, then the gap between senders one and two remains at 3 packets even after the rate change, unless sender one temporarily sends packets 8 and 10 faster. On the other hand, if k' is chosen to be 8, then senders one and two, will run the PPA beginning from packet 8 in order to determine who sends what, and packets 9 and 11 might be sent twice. This duplication results in the effective aggregate bit rate to temporarily go below the required sending rate S. One way to deal with this is for the sender to increase the sending rates temporarily after receiving the control packets.



Arrival time of control packet
Sender 1
0
1
2
4
6
8
10
Sender 2
3
5
7
9
11
Arrival time of control packet
t0
t1
time

Fig. 4. *Illustration of choosing synchronization sequence number.*

We have chosen to set $k' = min_j k''(j)$ [2], where $k''(j)$ is the estimated sequence number for the latest packet sender j has just sent, before receiving the control packet. Since the receiver knows about the last packet received from each sender, and the round trip time between itself and each sender, it can arrive at a fairly accurate estimate of $k''(j)$. In particular, a reasonable estimate of $k''(j)$ is $k^*(j) + 2D(j)S$, where $k^*(j)$ is the sequence number for the last packet receiver has received from sender j, and S is the total sending rate in packets per second. This estimate of $k''(j)$ is reasonable since $2D(j)S$ is the approximate number of packets sent by all senders during round trip time of sender j, i.e. during $2D(j)$ interval. For $k' = min_j k''(j)$, packet duplication might occur, resulting in temporarily lower aggregate bit rate than S at the receiver. To remedy this, the receiver draws upon its buffer in order to maintain playback rate of S.

In actual Internet experiments, where we change rates on the order of once every few minutes, and RTT difference of 40 milliseconds, we have found 500 milliseconds of buffer at the receiver to be large enough to absorb aggregate rate fluctuation at the transition time, for streaming periods of up to 20 minutes.

IV. Results

In this section, we first show the numerical results for our proposed optimal rate allocation scheme, followed by simulations and experimental results.

[2]We can also choose $k' = max_j k''(j)$, the details on using both strategies are discussed in [34]

Scenario	Average good time in seconds		Average bad time in seconds		Packet loss probability in the good state		Packet loss probability in the bad state		Average packet loss rate p	
	A	B	A	B	A	B	A	B	A	B
X	0.01	0.01	0.01	0.01-0.05	0	0	1	1	1%	1%- 5%
Y	5	5	0.5	0.5-2	0	0	0.2	0.2	1%	2%- 6%

TABLE III

Parameters chosen for numerical characterization in scenarios X and Y.

A. Numerical Characterization

We now present numerical results for various FEC protection levels and network characteristics using the optimal rate allocation between two senders versus that of using one sender. In the two sender scheme, we send packets on two "loss independent" routes A and B while in "one sender" scheme, packets are sent only on route A. Since end-to-end packet loss characteristics on the Internet vary widely depending on the locations of sender and receiver, we show the results for two common scenarios X and Y with the parameters shown in Table III. Note that in both scenarios X and Y, route A has lower packet loss rate than route B. The aggregate sending rate of both "two senders" and "one sender" scheme is $800kbps$, and the packet size is set to $500\ bytes$. We assume that available bandwidth for each route is greater than 800kbps in order to make a fair comparison for single route versus multiple route streaming.

Figures 5(a) and (b) show the probability of irrecoverable loss for scenarios X and Y, respectively, as a function of average bad times of route B for three FEC protection levels: $RS(30, 27)$, $RS(30, 25)$, and $RS(30, 23)$. As the average bad time of route B increases, as expected, the irrecoverable loss probability also increases for all three levels of FEC protection. Furthermore, a small increase in FEC protection level from $RS(30, 25)$ to $RS(30, 23)$ can reduce the optimal irrecoverable probability by a large factor such as 3.

Figures 6(a) and (b) show N_A, the optimal number of packets out of 30 that should be sent on route A as a function of the average bad time of route B for scenarios X and Y, respectively. As the average bad time of route B increases, more packets are sent on route A for all three levels of FEC protection. At the same average bad time of route B, the number of packets sent on the route A decreases with increased FEC

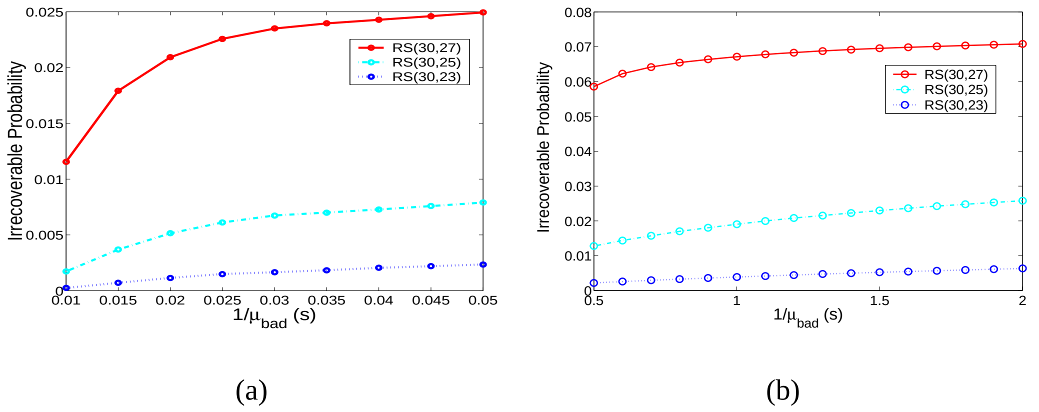
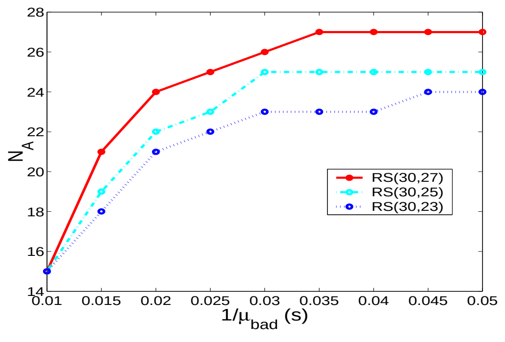


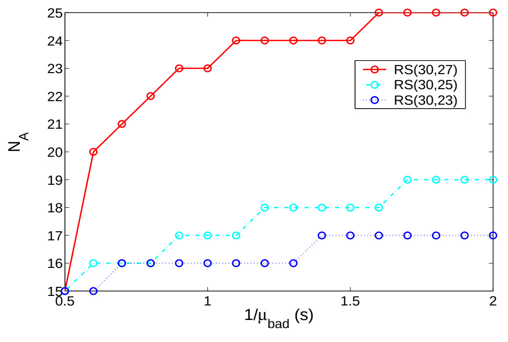
Fig. 5. *Irrecoverable loss probability for various of FEC levels as a function of average bad times of route B, using optimal partition for two senders for (a) scenario X and (b) scenario Y.*

protection level. This indicates that when stronger FEC protection is employed, more packets should be sent on the “bad” route. This is intuitively plausible by considering that in the reverse scenario in which weak FEC protection is employed, more packets should be sent on the “good” route. Specifically, in the extreme case where no FEC is used, no losses can be recovered by FEC, and hence, it is advantageous to primarily rely on the “good” route.

Figures 7(a) and (b) show the ratio of irrecoverable loss probabilities between the cases when all packets are sent on route A and when the optimal rate allocation is employed to send packets on both routes A and B for various FEC protection levels.

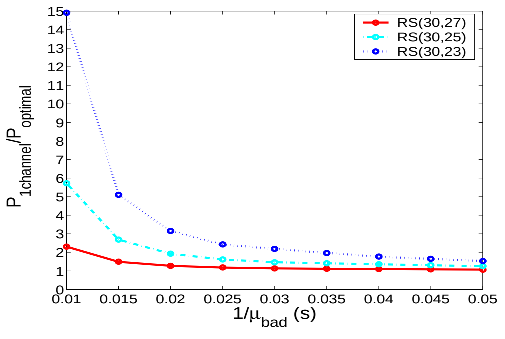
As seen, the irrecoverable loss probability improves by as much as a factor of 15, depending on network conditions and the amount of FEC. Also as expected, the curve corresponding to $RS(30, 23)$ is above that of $RS(30, 25)$ which in turn, is above that of $RS(30, 27)$. This indicates that the optimal rate allocation scheme is more effective with stronger FEC protection. Also, The above results show that using multiple senders can reduce the irrecoverable packet loss over that of single sender in two typical loss scenarios in the Internet. We should also emphasize that our framework for dividing the rates among senders to reduce irrecoverable packet loss is also applicable to other “bursty” network models which do not necessarily follow the exact “two-state” Markov model.

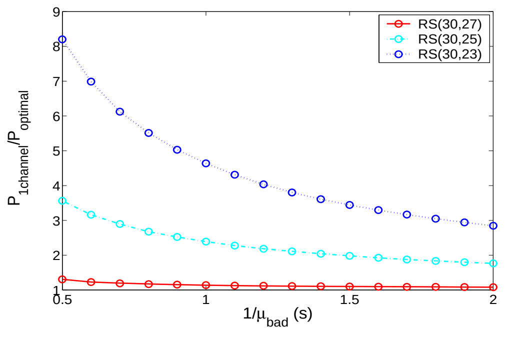




(a) (b)

Fig. 6. *Optimal partition for various FEC levels as a function of average bad times of route B for (a) scenario X and (b) scenario Y; N_A denotes the number of packets per 30 sent on route A.*



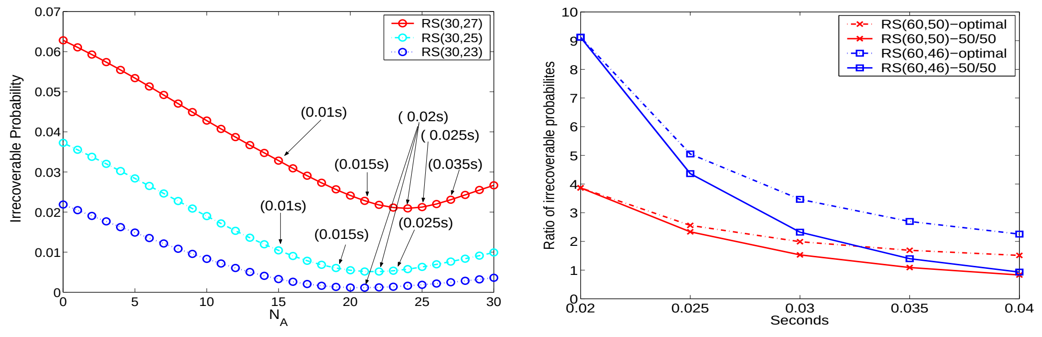


(a) (b)

Fig. 7. *Irrecoverable loss probability ratio of sending all packets using one sender to that of using two senders as a function of average bad times of route B for scenarios (a) X and (b) Y.*

B. Sensitivity Analysis of Optimal Sending Rate

In this section, we analyze the sensitivity of loss probability to deviations from optimal sending rates, arising from inaccuracies in route parameter estimation, or the limitation of the system to rapidly react to the changing network conditions. Figure 8(a) shows the irrecoverable probabilities for various rate allocations between two senders, when routes A and B have identical average good times at 1 second, and the average bad time of route A and B are at 10 and 20 milliseconds, respectively. The optimal sending rate for each FEC protection level is at the minimum of each corresponding curve. Assuming inaccurate estimation of average bad time of route B as shown in the parentheses, resulting sending rates on the x-axis vary around



(a) (b)

Fig. 8. *(a) Irrecoverable probability as the function of different partitions of sending rate between two senders; (b) irrecoverable loss probability ratios between sending all packets on route A over 50-50 and optimal rate splits.*

the optimal rate as shown in Figure 8(a). Note that the curves with stronger FEC protection levels are more flat at the minimum than those with weaker FEC protection levels. This indicates that when strong FEC protection is used, a slight variation in the sending rate around the optimal value results in a smaller change in irrecoverable probability than when weaker FEC protection is used. Hence, if delay and bandwidth requirements are satisfied, it is preferable to use stronger FEC protection level to be robust to slight deviations from the optimal sending rate values.

To further characterize these, Figure 8(b) shows the irrecoverable loss probability ratios of sending all packets on the better route A over the optimal and over 50-50 rate split between the routes. The average good times of routes A and B are now set to 1 second while the average bad time of route A is set to 20 milliseconds and that of route B varies from 20 to 40 milliseconds as shown in the x-axis of Figure 8(b). As expected, the optimal rate split results in the highest ratio, hence largest gain over the single route streaming. As seen, the 50-50 rate split also results in lower irrecoverable loss probability than sending all packets on single route A, especially when strong FEC is employed, and route B is not substantially worse than A. As expected, the more asymmetric the two routes become, the larger the gap between the optimal and 50-50 splits. The actual Internet experiments in Section IV-C set further light on these results.

C. Internet Experimental Results With Artificially Induced Packet Loss

We now demonstrate the effectiveness of optimal rate allocation scheme in reducing packet loss. Before describing the experiments, we emphasize that the reduction of packet loss using distributed video streaming scheme over the traditional single path scheme is attributed entirely to the rate allocation between senders, which reduces the bursty loss, hence, increasing the error correction capability of FEC. In essence, the packet partition algorithm provides the necessary machinery and logistics, to make the overall system, including the rate control algorithm to function properly, and hence is not directly responsible for reducing packet loss. Also the goals of PPA and rate allocation algorithms are different and in some sense complementary, and hence even though the two interact with each other, they cannot be combined. Therefore, we do not anticipate any improvements by combining the two, and examine their performance separately.

We perform the following experiments to compare traditional uni-sender streaming with distributed streaming using multiple senders. In experiment one, we use one sender in Belgium to stream all packets to U.C. Berkeley. In experiment two, one sender at Belgium and the other at Sweden simultaneously stream packets to the receiver at U.C. Berkeley. In both experiments, we use $RS(60, 46)$ with packet size of 500 bytes, and set the total sending rate to 200 packets per second, i.e. 800 kbps. In both experiments, we artificially induce packet loss at the senders using the Markov chain model mentioned earlier; however, other factors such as round trip time remains faithful to the characteristics of the routes. The observed average round trip times between Belgium and U.C. Berkeley is 152 milliseconds, and that between Sweden and U.C. Berkeley is 199 milliseconds. Throughout the experiments, we assume that 800kbps bandwidth is available at all times for a single sender to stream the packets.

For both experiments, initially, the average good and bad times of both connections are set at 1 and 0.02 seconds, respectively, and the packet loss probability in bad state is set to $p = 1$. These parameters result in an average packet loss rate of 2%. During the first 200 seconds in experiment one, all packets are sent using only Belgium sender, resulting in 8 irrecoverable loss events as indicated by the number of circles above and to the left of the horizontal and vertical lines in Figure 9(a). During the first 200 seconds in experiment two, sending rates are split equally between Belgium and Sweden senders, resulting in no irrecoverable loss events as seen in Figure 9(b). This is due to the reduction in burst loss by sending packets at lower rates on both routes, and hence, the increased error correction capability of FEC. From $t = 200$s to $t = 600$s, we increase the average bad time of Belgium connection to 0.04 seconds, resulting in 66 irrecoverable losses for experiment one. On the other hand, for experiment two, there are only 6 irrecoverable losses during $t = 200$s to $t = 400$s where sending rates are still split equally. This indicates that splitting sending rates between senders sup-optimally, can still provide some degree of robustness to sudden change for the worse in network characteristics.

Further reduction in packet loss can be achieved by sending packets at the optimal rates. In particular, for experiment two, during the period from 400 to 600 seconds, the Belgium and Sweden senders adjust their sending rates to optimal levels, i.e. 60 and 140 packets per second for Belgium and Sweden, respectively. During this period, there is no irrecoverable loss events as shown in Figure 9(b). This suggests that if accurate network parameters are available, then sending packets at the optimal rates results in further packet loss reduction as compared to dividing packets roughly evenly especially as the loss difference between the two routes widens. Figure 9(c) shows the throughputs of two senders in experiment two. As seen, the variation in throughputs is mainly due to packet loss. In particular, we have found the throughput dips of Belgium connection during the period from $t = 200$s to $t = 400$s to be on average larger than that of Sweden sender due to higher number of lost packets for Belgium sender during this period [3]. From $t = 400$s to $t = 600$s, since the sending rate of Belgium sender reduces to 60 packets per second, these throughput dips become smaller than before, providing FEC a better chance to recover the lost packets.

Next, we show the results of our packet partition algorithm for the same distributed streaming experiments. As discussed previously, the goal of the packet partition algorithm is for senders to send packets in an interleaved fashion such that that (a) packets are not duplicated and (b) arrive at the receiver more or less in playback order. During this experiment, the control packets are sent twice, once at the beginning, and the other at $t = 400$s in order to change the sending rates. We have observed no duplicate packets for the first 400 seconds. Immediately after the rate change, only one duplicate packet is observed at Sweden sender.

[3]This is best seen on the color printout of this plot.

However, in other experiments, depending on the accuracy of the round trip time estimates, the number of duplicate packets ranges from 0 to 3.

To show how closely interleaved packets are received, Figure 9(d) shows the difference in sequence number between two consecutive received packets. As seen, majority of the packets are no further than five packets apart. Some are as far as 17 packets apart, and can be attributed to burst losses occurring simultaneously for short periods of time on both routes. It is interesting to note the distinct lines rather than random patterns formed by the circles due to random network jitter. Upon close inspection of the trace file, we observe there is little network jitter. Ideally, without network jitter and with accurate estimates of the round trip times, the plot should show one horizontal line starting at one. However, since we use only one packet to probe for round trip time at the beginning, initially there is a slight inaccuracy in the estimated round trip time. Therefore, the first round trip time between U.C. Berkeley and Sweden is estimated at about 190 milliseconds, 9 milliseconds off the average value. This results in the line patterns appearing at other values than 1, making one sender send packets slightly ahead of the other.

After the second control packet is sent at 400 seconds, a more accurate round trip estimate is used, resulting in line patterns closer to 1. Also, since Sweden sender sends at higher rate, i.e. at 140 packets per second, the packet partition algorithm assigns a larger number of consecutive packets to the Sweden sender, and therefore, on average, the packets arrive at the receiver more or less in order. Figure 9(e) shows the histogram of the plot in Figure 9(d). As seen, most of the packets are within 5 packets of each other. This indicates the effectiveness of the proposed PPA to send closely interleaved packets.

D. Actual Internet Experiments

We also perform actual experiments over the Internet without artificially inducing packet loss. Rather, we run an off-line algorithm to be described shortly to determine network parameters. In experiment three, a sender at Purdue University streams a H.263 encoded video to a receiver at U.C. Berkeley at the rate of 200 packets per second. In experiment four, the same video is also streamed at 200 packets per second from a sender at Sweden to a receiver at U.C. Berkeley. In experiment five, both senders at Sweden and Purdue University simultaneously stream the video to a receiver at U.C. Berkeley with the optimal rates of

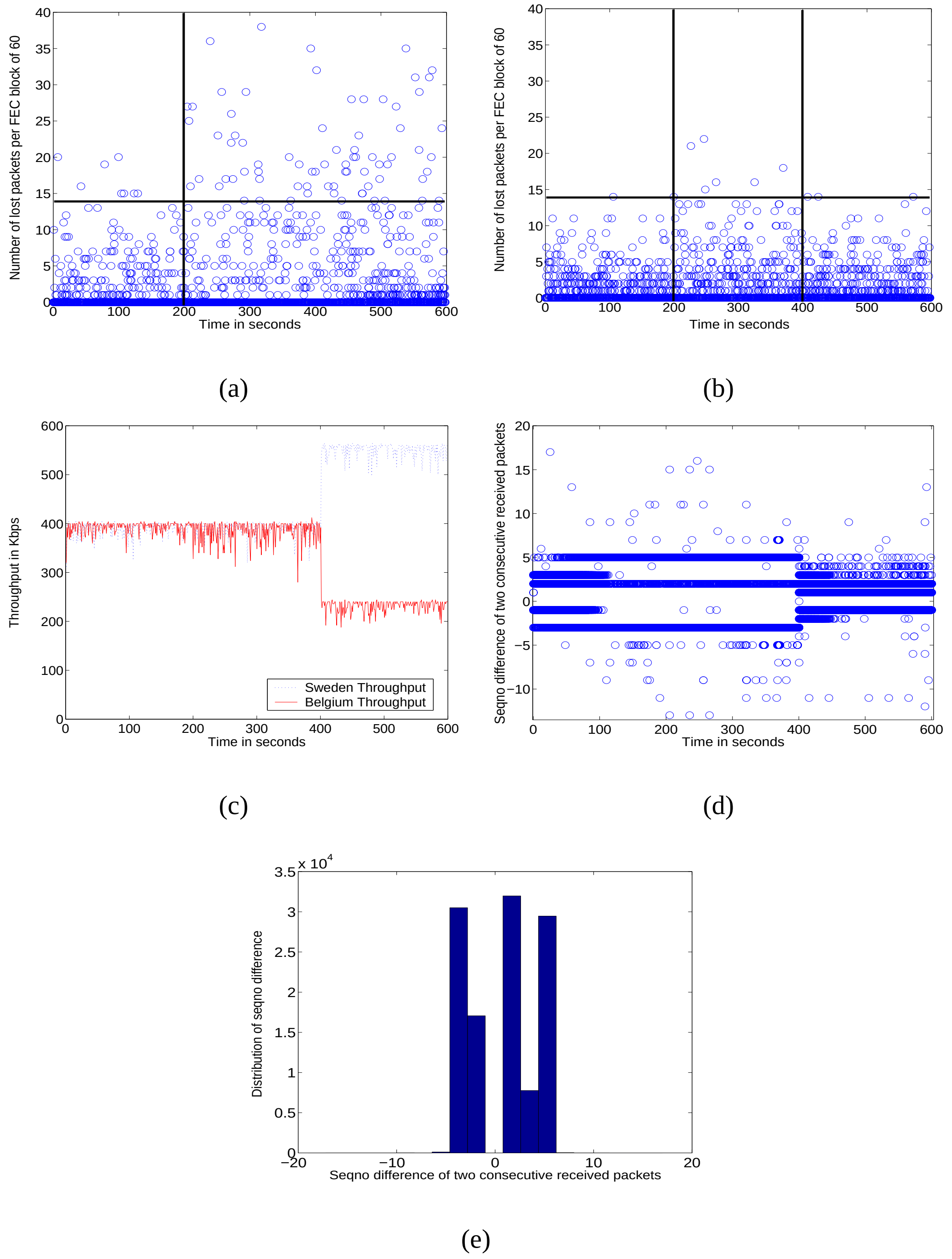


Fig. 9. *Internet simulations showing the number of lost packet per FEC block of 60 packets vs. packet sequence for (a) streaming from Belgium alone; (b) streaming from Belgium and Sweden; (c) Throughputs of two senders; (d) Variations in order of the received packets; (e) Histogram of variation in packet order.*

80 and 120 packets per second, respectively. In all three experiments, the streamed H.263 video has constant bit rate of $720 kbps$ and is packetized into 500 bytes packets which are then protected using $RS(100, 90)$ code. To compute the optimal sending rate, we estimate the network parameters for Sweden-Berkeley and Purdue-Berkeley routes, using a Hidden Markov Model inference algorithm [35] on the traces of packets over many hours offline [4]. To capture the bursty characteristics of the network, we ignore all the single loss events, and only use consecutive burst loss of 2 or more packets in our estimates of network parameters. Although many single losses may change the network characteristics, we observe that in our experiments, there are rarely multiple single losses within a FEC block, hence these losses can be recovered by FEC most of the times. Therefore, we only consider bursty characteristics to determine the sending rates. In our experiments, the average congestion intervals for Sweden-Berkeley and Purdue-Berkeley are estimated to be approximately 39 and 33 milliseconds while the average good times are 6.1 and 6.9 minutes, respectively.

Figure 10 plots the number of lost packets per 100 for experiments three, four, and five denoted with squares, circles, and crosses, respectively. The points above horizontal line represent irrecoverable loss events. Since we are using $RS(100, 90)$, irrecoverable loss happens when there are more than 10 lost packets per 100 sent packets. As seen, there are 5 instances of irrecoverable loss for experiments three and four where only one sender is used to stream video to receiver. On the other hand, in experiment five where both senders at Sweden and Purdue university stream video simultaneously to the receiver at U.C. Berkeley, all the lost packets are successfully recovered by FEC. Note that even though the average packet loss rates for experiments three, four, and five are 0.05%, 0.09%, and 0.08%, i.e. well below 10%, $RS(100, 90)$ code in experiments three and four cannot recover all the lost packets due to the bursty loss nature of Internet. Also, we have found uni-path schemes result in 13 to 20 dB instantaneous drop in PSNR at the time of irrecoverable loss events as compared to distributed video streaming scheme. To further show the robustness of the distributed streaming, we perform many uni-sender and and multi-sender experiments for different sending rates and two different levels of FEC protection: RS(60,46) and RS(60,50). The senders in the

[4]An online algorithm to estimate network parameters has also been developed and is discussed in detail in [34]. In particular, the average lengths of loss runs and loss-free runs times sending interval is used as the estimates of $1/\mu_b$ and $1/\mu_g$ respectively.

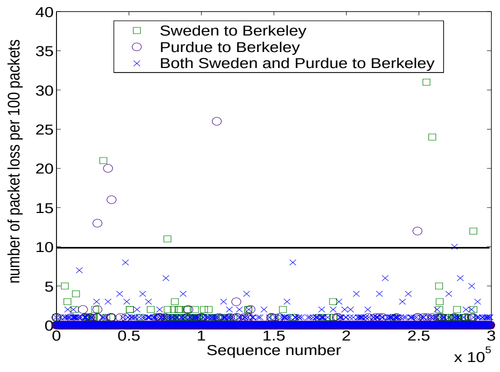


Fig. 10. *Actual Internet experiments showing the benefits of distributed video streaming over conventional approach.*

	Sweden	Hong Kong	Irrecoverable Loss		Gain over Hong Kong		Gain over Sweden	
			RS(60,46)	RS(60,50)	RS(60,46)	RS(60,50)	RS(60,46)	RS(60,50)
Rate	0	220	18	28	1	1	2.8	2.4
	20	200	14	31	1.3	0.9	3.6	2.2
	40	180	9	30	2	0.9	5.7	2.4
	60	160	5	9	3.6	3.1	10.2	7.6
	80	140	7	9	2.6	3.1	7.2	7.6
	100	120	12	20	1.5	1.4	4.3	3.5
	220	0	51	69	0.4	0.4	1	1

TABLE IV

Irrecoverable loss reduction using two senders for various sending rates and FEC levels.

experiments are at Sweden and Hong Kong, and the receiver is at U.C. Berkeley[5]. Packet size is set to 500 bytes and total sending rates is set to 220 packets per seconds. All the experiments are performed between 1 and 4 PM PT, and the duration of each experiment is set to 15 minutes. The results of these experiments are shown in Table IV. Sending rates in packets per second are shown in columns 1 and 2. The numbers of irrecoverable loss events are shown in columns 3 and 4, and the ratios of irrecoverable losses of multi-sender over uni-sender are shown in columns 5 through 8.

As seen in Table IV, distributed streaming from multiple senders clearly reduces the number of irrecoverable losses, up to 10.2 times over uni-sender streaming. Even though the average packet loss rate of Sweden sender, i.e. 1.3%, is lower than the average loss rate of Hong Kong, i.e. 1.8%, we have empirically found

[5]This Sweden site is not the same as the Sweden site in experiment four. Both sites planetlab-1.it.uu.se at Sweden and s1_803.ie.cuhk.edu.hk at Hong Kong are part of PlanetLab sites.

the Sweden's loss patterns to be more bursty; hence, it is advantageous to send packets at lower rate on the Sweden route. In this scenario, if one sends all the packets on the route with lower average loss rate, namely Sweden route, the number of irrecoverable loss is even larger than sending all the packets on the route with larger loss rate, namely Hong Kong route. This indicates that when appropriate amount of FEC is used, the lower average loss rate of a particular route is not necessarily a good indicator for sending packets at higher rate on that route, rather the burstiness loss patterns should be taken into account for setting the sending rates between routes.

Results in Table IV further verify our earlier numerical results in Section IV-B regarding graceful degradation of irrecoverable loss as we deviate from optimal partition of the packets between the two routes. Specifically, while the number of irrecoverable loss events is at its minimum for 60/160 split, deviation to 40/180 or 80/140 results in slight increase in irrecoverable loss events. This suggests that in a highly dynamic environment where network parameters change frequently, and therefore, it is hard to estimate them accurately, splitting packets sup-optimally between the senders may still provide advantages of uni-route streaming, even though optimal partitioning always results in best performance. Therefore, if enough FEC is used, control packets could be sent infrequently to avoid frequent adaptation to "noise" to achieve reasonable performance.

V. Extensions and Modifications to the Current System

In this paper, we have not addressed the issue of changing FEC levels during the session. This may be necessary when channel conditions for all senders get worse, and the redistribution of bit rates among senders is not sufficient to keep the irrecoverable loss probability at sufficiently low levels. If the video bit rate is fixed, this approach requires that the additional available bandwidth to exist for larger amount of FEC. Otherwise, reduced scalable video bitstream might be used to compensate for additional FEC. Assuming either approach, the receiver can run the rate allocation algorithm for the new amount of FEC, and send the control packets consisting the new FEC information to the senders. This approach however assumes that the senders can all either compute new levels of FEC on the fly, or have pre-stored redundant packets corresponding to different FEC levels ahead of time. In our current framework, we only consider the possibility of changing the sending rates among a fixed set of senders. It is possible to extend our system so that the receiver can dynamically request new senders in order to provide additional bandwidth as required.

VI. Conclusions

In this paper, we proposed a distributed video streaming framework using a receiver-driven protocol for simultaneous video streaming from multiple senders to a single receiver in order to achieve higher throughput, and to increase tolerance to packet loss due to network congestion. The receiver-driven protocol employs a rate allocation algorithm and a packet partition algorithm. The rate allocation algorithm determines the sending rate for each sender taking in account available network bandwidth, the given amount of FEC, and channel characteristics in order to minimize the probability of packet loss in bursty loss environments. The packet partition algorithm ensures no senders send the same packets, and at the same time, minimizes the startup delay. Our experimental results demonstrate the effectiveness of distributed video streaming framework in reducing overall packet loss rate. Future work involves investigating robustness of PPA under large network jitter, and inaccuracies in parameter estimation. We also plan to test the overall system under a wider range of parameters, and more realistic network conditions such as heavy traffic load.

References

[1] W. Tan and A. Zakhor, "Real-time internet video using error resilient scalable compression and tcp-friendly transport protocol," *IEEE Transactions on Multimedia*, vol. 1, pp. 172–186, june 1999.

[2] G. De Los Reyes, A. Reibman, S. Chang, and J. Chuang, "Error-resilient transcoding for video over wireless channels," *IEEE Transactions on Multimedia*, vol. 18, pp. 1063–1074, june 2000.

[3] J. Robinson and Y. Shu, "Zerotree pattern coding of motion picture residues for error-resilient transmission of video sequences," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 1099–1110, June 2000.

[4] H. Ma and M. El Zarki, "Broadcast/multicast mpeg-2 video over wireless channels using header redundancy fec strategies," in *Proceedings of The International Society for Optical Engineering (SPIE)*, November 1998, vol. 3528, pp. 69–80.

[5] W. Tan and A. Zakhor, "Error control for video multicast using hierarchical fec," in *Proceedings of 6th International Conference on Image Processing (ICIP)*, October 1999, vol. 1, pp. 401–405.

[6] P.A. Chou, A.E. Mohr, A. Wang, and S. Mehrotra, "Error control for receiver-driven layered multicast of audio and video," *IEEE Transactions on Multimedia*, vol. 3, pp. 108–22, March 2001.

[7] A. Mohr, E. Riskin, and R. Ladner, "Unequal loss protection: Graceful degradation over packet erasure channels through forward error correction," *IEEE Journal on Selected Areas in Communication*, vol. 18, pp. 819–828, April 2000.

[8] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "The pim architecture for wide-area multicast routing," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 153–162, April 1996.

[9] T. Nguyen and A. Zakhor, "Distributed video streaming," in *Proceedings of the SPIE - The International Society for Optical Engineering, Multimedia Computing and Networking (MMCN)*, San Jose, CA, January 2002, vol. 4673, pp. 186–95.

[10] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast application," in *Architectures and Protocols for Computer Communication*, October 2000, pp. 43–56.

[11] T. Nguyen and A. Zakhor, "Distributed video streaming with forward error correction," in *Packet Video Workshop*, Pittsburg, PA, April 2002.

[12] H.V. Poor and G.W. Wornell, *Wireless Communications: Signal Processing Perspectives*, Prentice Hall, 1998.
[13] N.F. Maxemchuk, *Dispersity Routing in Store and Forward Networks*, Ph.D. thesis, University of Pennsylvania, 1975.
[14] M.O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *Journal of the Association for Computing Machinery*, vol. 36, no. 2, pp. 335–348, April 1989.
[15] J. Byers, M. Luby, and M. Mitzenmacher, "Accessing multiple mirror sites in parallel: using tornado codes to speed up downloads," in *Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, March 1999, vol. 1, pp. 275–283.
[16] M. Luby, M. Mitzenmacher, M. Shokrollahi, Spielman D, and V. Stemann, "Practical loss-resilient codes," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, May 1997, pp. 150–159.
[17] E.G. Steinbach, Y.J. Liang, and B. Girod, "A simulation study of packet path diversity for tcp file transfer and media transport on the internet," in *Tyrrhenian International Workshop on Digital Communication (IWDC)*, September 2002.
[18] V.N. Padmanabhan, H.J. Wang, P.A. Chou, and K. Sripanidkulchai, "Distributed streaming media content using cooperative networking," in *ACM NOSSDAV*, Miami, FL, May 2002.
[19] R. Rejaie and A. Ortega, "Pals:peer to peer adaptive layered streaming," in *NOSSDAV*, June 2003.
[20] A. Reibman, H. Jafarkhani, Y. Wang, M. Orchard, and R. Puri, "Multiple description coding for video using motion compensated prediction," in *Proceedings of International Conference on Image Processing (ICIP)*, October 1999, vol. 3, pp. 837–841.
[21] J. Apostolopoulos, "Reliable video communication over lossy packet networks using multiple state encoding and path diversity," in *Proceeding of The International Society for Optical Engineering (SPIE)*, January 2001, vol. 4310, pp. 392–409.
[22] R. Puri, K. Ramchandran, K. Lee, and V. Bharghavan, "Forward error correction (fec) codes based multiple description coding for internet video streaming and multicast," *Signal Processing: Image Communication*, vol. 6, no. 8, pp. 745–762, May 2001.
[23] K. Goyal and J. Kovacevic, "Generalized multiple description coding with correlating transforms," *IEEE Transactions on Information Theory*, vol. 47, pp. 2199–2224, April 2001.
[24] Y. Wang, M. Orchard, V. Vaishampayan, and A. Reibman, "Multiple description coding using pairwise correlating transforms," *IEEE Transactions on Image Processing*, vol. 10, no. 3, pp. 351–366, March 2001.
[25] J. Apostolopoulos, "On multiple description streaming with content delivery networks," in *InfoComm*, June 2002, vol. 4310.
[26] J.Chakareski and B. Girod, "Rate-distortion optimized packet scheduling and routing for media streaming with path diversity," in *Data Compression Conference*, April 2003.
[27] Y.J. Liang, E.G. Steinbach, and B. Girod, "Real-time voice communication over the internet using packet path diversity," in *Proceedings ACM Multimedia 2001*, Sept 2001, pp. 431–440.
[28] S. Agarwal, C. Chuah, and H. Katz, "Opca: Robust interdomain policy routing and traffic control," in *IEEE Openarch*, April 2003.
[29] D.G. Andersen, *Resilient overlay networks, Master Thesis*, Massachusetts Institute of Technology, 2001.
[30] J. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive fec-based error control for internet telephony," in *Proceedings of IEEE INFOCOM*, 1999, vol. 3, pp. 1453–60.
[31] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of the temporal dependence in packet loss," in *INFOCOM*, 1999, vol. 1, pp. 345–52.
[32] M. Borella, D. Swider, S. Uludag, and G. Brewster, "Internet packet loss: Measurement and implications for end-to-end qos," in *Proceedings of ICPP Workshop on Architectural and OS Support for Multimedia Applications Flexible Communication Systems*, Minneapolis, MN, 1998, pp. 3–12.
[33] U. Horn, K. Stuhlmuller, M. Link, and B. Girod, "Robust internet video transmission based on scalable coding and unequal error protection," *Signal Processing: Image communication*, vol. 15, pp. 77–94, 1999.
[34] T. Nguyen, "Distributed media streaming," Ph.D. Thesis, U.C. Berkeley to be filed in 2003.
[35] M. Jordan and C. Bishop, *An Introduction to Graphical Models (In press)*.

Appendix

Procedure for computing $P(m, k, N_m)$

To compute $C(K, N_0, N_1)$ in Section II-A, we first compute $P(m, i, N_m)$ based on the given network parameters (μ_g^m, μ_b^m) of sender m as follows. We use notations in Table V: Note that p_{ij}^m depends not only on the parameters μ_g^m and μ_b^m, the rates at which the state of sender m changes from "good" to "bad" and vice versa, but also on the rate that sender m sends. Then,

$$\phi_{ij}^m(k,n) \triangleq Prob(L_m(n) = k, S_m(n) = j | S_m(0) = i)$$

denotes the probability that sender m is in state j, and there are k lost packets after it sends n packets, given that it is initially in state i. We can compute $\phi_{ij}^m(k,n)$ recursively by conditioning on the previous state l, and by using the total probability theorem to obtain

$$\phi_{ij}^m(k,n) \quad = \quad \sum_{l \in g,b} [\phi_{il}^m(k-1, n-1) p_{lj}^m P_m^{loss}(j) + \phi_{il}^m(k, n-1) p_{lj}^m (1 - P_m^{loss}(j))]$$

$S_m(n) \in \{g, b\}$	State of sender m after it sends n packets
$L_m(n)$	Number of lost packets out of n packets sent by sender m
$P_m^{loss}(i)$	Packet loss probability when sender m is in state i
p_{ij}^m	Transition probability from state i to state j for sender m

TABLE V

Notations for computing $P(m, kN_m)$.

for all $k \geq 0$ and $n \geq 0$, with the boundary conditions:

$$\phi_{ij}^m(0,0) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

$$\phi_{ij}^m(k,n) = 0 \text{ for } n < k$$

The above boundary conditions hold because of following arguments. If sender m does not send packet and hence does not change its state, there will certainly be no lost packets. Therefore $\phi_{ij}^m(0,0) = 1$ for $i = j$. On the other hand, by definition, it is impossible to have sender m change its state without sending a packet, hence $\phi_{ij}^m(0,0) = 0$ for $i \neq j$. Finally, $\phi_{ij}^m(k,n) = 0$ for $n < k$ since number of lost packets cannot exceed the number of sent packets. Now, since $P(m,k,N_m)$ is the probability of k lost packets out of N_m packets sent by sender m, regardless of the initial and final states, we marginalize $\phi_{i,j}^m(k,N_m)$ to obtain

$$P(m,k,N_m) = \sum_{i \in \{g,b\}} \sum_{j \in \{g,b\}} \pi_i^m \phi_{ij}^m(k,N_m)$$

where $\pi_g^m = \mu_b^m / (\mu_g^m + \mu_b^m)$ and $\pi_b^m = \mu_g^m / (\mu_g^m + \mu_b^m)$ are the steady-state probabilities of sender m being in “good” and “bad” states, respectively.