Distributed Video Streaming with Forward Error Correction

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. Many use techniques from source and channel coding, or implement transport protocols, or modify system architectures in order to deal with delay, loss, and time-varying nature of the Internet. In our previous work , we proposed a framework with a receiver driven protocol to coordinate 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. The receiver-driven protocol employs two algorithms: rate allocation and packet partition. The rate allocation algorithm determines the sending rate for each sender; the packet partition algorithm ensures no senders send the same packets, and at the same time, minimizes the probability of late packets. In this paper, we propose a novel rate allocation scheme to be used with Forward Error Correction (FEC) in order to minimize the probability of packet loss in bursty loss environments such as those caused by network congestion. Using both simulations and actual Internet experiments, we demonstrate the effectiveness of our rate allocation scheme in reducing packet loss, and hence, achieving higher visual quality for the streamed video.

1   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 codecs 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 reduce delay due to retransmission at the expense of increased bit rate [4, 5]. From protocol perspective, there are approaches based on multicast [5, 6] and TCP-friendly protocols [1] for streaming multimedia data over the Internet. Multicast reduces the network bandwidth by not sending duplicate packets on the same physical link [6], but 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 [7]. From network perspective, content delivery network (CDN) companies such as Akamai, iBeam, and Digital Island use edge architecture as shown in Figure 1(a) to achieve better load balancing, lower latency, and higher throughput. Edge architecture reduces latency by moving content to the edge of the network in order to reduce round-trip time and to avoid congestion in the Internet. Companies such as FastForward Networks and Akamai strategically place a large number of the servers around the Internet so that each client can choose the server that results in shortest round-trip time and least amount of congestion. A number of these schemes assume a single fixed route between the receiver
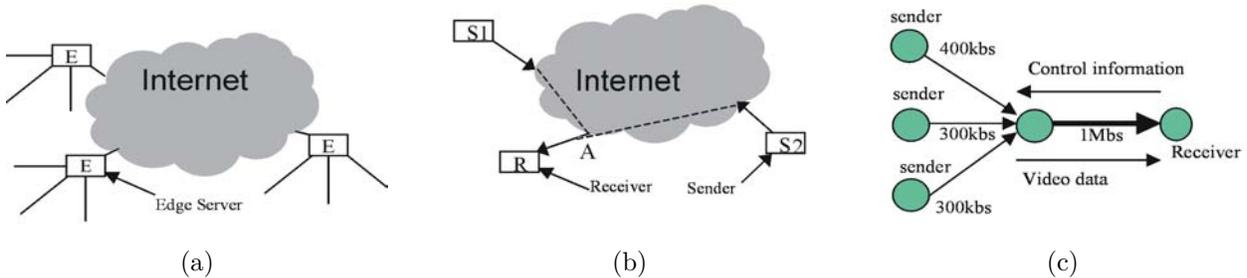
Figure 1: *(a) Edge server architecture; (b) Distributed video streaming architecture; (c) Distributed video streaming.*

and the sender throughout the session. If the network is congested along that route, video streaming suffers from high loss rate and jitter. Even if there is no congestion, as the round-trip time between the sender and the receiver increases, the TCP throughput may reduce to unacceptably low levels for streaming applications. Furthermore, authors in [8, 9] have revealed the ill-behaved systematic properties in Internet routing, which can lead to sub-optimal routing paths. Based on these, it is conceivable to make content available at multiple sources so that the receiver can choose the "best" sender based on bandwidth, loss, and delay. In a way, this is what Akamai does by moving the server closer to the client. However, if no sender can support the required bit rate needed by the application, it is conceivable to have multiple senders simultaneously stream video to a single receiver to effectively provide the required throughput. Having multiple senders is also a diversification scheme in that it combats unpredictability of congestion in the Internet. If the route between a particular sender and the receiver experiences congestion during streaming, the receiver can redistribute streaming rates among other senders, thus resulting in smooth video delivery.

In previous paper [10], we proposed 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 combined approaches from different perspectives including system architecture and transport protocols. From systems perspective, we expanded the edge architecture to allow simultaneous video streaming from multiple mirror sites as shown in Figure 1(b). 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 used a TCP friendly protocol to coordinate simultaneous transmission of video from multiple mirror sites to a single receiver effectively.

In this paper, we extend our previous work by proposing a novel rate allocation scheme to be used with FEC to minimize the probability of packet loss in bursty channel environments. In general, FEC has been shown to be an effective tool in combatting packet loss in low delay multimedia communications as well as streaming applications on the Internet [11]. It can be argued that for streaming applications, retransmissions together with large buffer sizes would obviate the need for FEC. However, in practice, retransmissions result in large start up delay, and limit interactive VCR-like functionalities such as fast forward and rewind. Based on delay considerations, in this paper, we will focus on FEC rather than retransmission schemes.

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 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 FEC level has to be closely matched to channel characteristics for it to be effective in single route streaming applications. In this paper, we show that by combining path diversification and FEC, we can combat bursty loss behavior in the Internet more effectively. Specifically the above mismatch of FEC level and network characteristics for single route streaming application becomes more relaxed in distributed streaming applications due to the additional redundancy introduced by multiple routes.

1.1 Related Work

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 [12] 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. Another example is multiple description coding of video [13, 14, 15, 16, 17], in which a video source is partitioned into multiple descriptions , each one assumed to be sent along a different route in the network. This approach assumes that the visual quality of the video degrades gracefully as number of received description decreases due to network congestion on different routes.

1.2 Assumptions

To successfully stream video from multiple senders, we assume that the available aggregate bandwidth from all the senders to the receiver exceeds the required video bit rate. As an example, consider a network configuration shown in Figure 1(b). The available bandwidth from router A to receiver R, sender $S1$ to router A, and sender $S2$ to router A is assumed to be $2Mbps$, $0.8Mbps$, and $0.4Mbps$, respectively. Router A can be an edge router of a particular network domain of a university or a company, and is assumed to be the only router receiver R is connected to. In this scenario, it is not possible to stream a $1Mbps$ video from sender $S1$ to receiver R since the available bandwidth for streaming video from $S1$ to R is only $0.8Mbps$. However, it is possible to stream a $1Mbps$ video simultaneously from both senders $S1$ and $S2$ to receiver R since the aggregate bandwidth from both senders $S1$ and $S2$ to receiver R is $1.2Mbps$ which exceeds the required bit rate of $1Mbps$. On the other hand, if the available bandwidth from router A to receiver R is only $0.9Mbps$, then the link between A and R becomes a bottleneck, and video streaming at $1Mbps$ becomes infeasible both for multiple senders and single sender scenarios. We also assume that the routes from the client to the senders do not share the same congestion link. If there is congestion on a shared link between two senders, the lost packets between different senders are correlated, and our analyses on optimal sending rates will no longer hold. Based on the above assumptions, in streaming situations when 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.

Finally, there has been work on detecting the shared congestion points of different routes [18] based on the correlation of packet loss and delay between routes. These correlations can be used ahead of time to improve the performance of our approach.

1.3 Scope

In this paper, we propose a novel rate allocation scheme to be used with FEC in a bursty loss environment such as the Internet. The rest of our paper is organized as follows. In Section 2, we briefly describe our previously proposed transport protocol, rate allocation, and packet partition algorithms [10]. Next, in Section 3 we propose a novel rate allocation scheme to be used with FEC in order to minimize the probability of packet loss. In Section 4, we present simulations and actual Internet experimental results. Finally, we conclude and provide possible extensions in Section 5.

2 Protocol Overview

2.1 Transport Protocol

In this section, we briefly describe our protocol originally proposed in [10]. 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. Each sender estimates and sends its round trip time to the receiver. The receiver uses the estimated round trip times and its estimates of senders' loss rates to calculate the optimal sending rate for each sender. When the receiver decides to change any of the senders' sending rates, it sends an identical control packet to each sender. The control packet contains the synchronization sequence number and the optimal sending rates as calculated by the receiver for all senders. Using the specified sending rates and synchronization sequence number, each sender runs a distributed packet partition algorithm to determine the next packet to be sent. Figure 1(c) shows the block diagram of an example of a system deploying our approach.

2.2 Loss and Bandwidth Estimation

In our protocol, the receiver estimates available bandwidth for each sender based on the TCP-friendly rate control algorithm (TFRC) proposed in [7]. TFRC protocol is designed to be fair with TCP traffic, and

results in less fluctuation in sending rate than TCP does. It calculates the available bandwidth according to the following equation:

$$B = \frac{s}{R\sqrt{\frac{2p}{3}} + T_{rto}(3\sqrt{\frac{2p}{3}})p(1 + 32p^2)} \tag{1}$$

where B denotes the current available TCP-friendly bandwidth between each sender and the receiver, T_{rto} is TCP time out, R is the estimated round-trip time in seconds, p is the estimated loss rate, and s is the TCP segment size in bytes. The estimated round trip time is computed using the moving average of round-trip times over a fixed time interval. Similarly, the estimated loss rate is the ratio of number of lost packets over the total number of packets sent during a fixed time interval. The estimated bandwidth B forms an upper bound for the TCP-friendly sending rate.

2.3 Rate Allocation Algorithm

In our proposed protocol in [10], the receiver computes the optimal sending rate for each sender based on its loss rate and estimated available bandwidth. The problem of allocating optimal sending rate to each sender can be stated as follows. Let N be the total number of senders, and $L(i,t)$ and $S(i,t)$ be the estimated loss and sending rates, respectively for sender i over an interval $(t, t+\delta)$. Our goal is to find $S(i,t)$, $i = \{1...N\}$, that minimize the total lost packets during interval $(t, t+\delta)$ given by

$$F(t) = \sum_{i=1}^{N} L(i,t)S(i,t)$$

subject to

$$0 \leq S(i,t) \leq B(i,t) \quad \text{and} \quad \sum_{i=1}^{N} S(i,t) = S_{req}(t)$$

where S_{req} is the required bit rate for the encoded video during the interval $(t, t+\delta)$, and $B(i,t)$ is the TCP-friendly estimated bandwidth for sender i during the interval $(t, t+\delta)$. In [10], we have proposed an algorithm to minimize $F(t)$, the number of lost packets during interval $(t, t+\delta)$, given instantaneous feedback, and assuming that the estimated loss rate and TCP-friendly available bandwidth are accurate. The idea of the algorithm is as follows. At time t, we sort the senders according to their estimated loss rates from lowest to highest. We start with thes lowest loss rate sender and assign its sending rate to be its TCP friendly estimated bandwidth as described in equation (1). We then continue to assign the available bandwidth of each sender to its sending rate, beginning with the ones with lower loss rates and moving to the ones with higher loss rates, until the sum of their available bandwidths exceeds the bit rate of the encoded video.

2.4 Packet Partition Algorithm

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. All the senders simultaneously run this algorithm to ensure that no sender sends the same video packet, and also to minimize the probability of packets arriving late at the receiver due to network jitter. The algorithm can be described as follows. Each

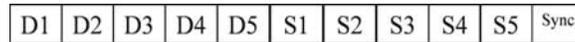| D1	D2	D3	D4	D5	S1	S2	S3	S4	S5	Sync

Figure 2: *Packet format*

sender receives control packet from the receiver through a reliable protocol whenever the receiver determines there should be a change in any of the sending rates. The format of the control packet is shown Figure 2. For simplicity, we do not show the IP header and sequence number. $S1 - S5$ are two-byte fields to specify the sending rate in packets/second for each sender. Packet size is constant for all senders. $D1 - D5$ are one-byte fields to denote the estimated delay from each sender to the receiver. This delay is expressed in multiples of 2 milliseconds interval. The $Sync$ field is the starting sequence number that all senders use in the packet partition algorithm to determine the next packet to send, immediately upon receiving the control packet. The entire copy of the video is assumed to reside at all the senders.

The basic idea in our packet partition algorithm is that among all senders $i = \{1...N\}$, the one that maximizes the time difference $A(i,k)$ between the estimated receive and playback time for k^{th} packet is chosen to send that packet. Hence, maximizing $A(i,k)$ is equivalent to minimizing the probability that the kth packet is late. Specifically, each sender i computes $A(i,k)$ for each packet k for itself and all other senders,

and only sends packet k if it discovers that is at a maximum for itself. If $A(i, k)$ 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 it is at a maximum among all other senders. The details on estimating $A(i, k)$ and other practical issues can be found in [10].

3 Proposed Distributed Streaming with FEC

The rate allocation algorithm in Section 2.3 minimizes number of lost packets, assuming the lost packets are identically and independently distributed. It is relatively simple to implement since only the estimated average loss rates and round-trip times of senders are required to compute the optimal sending rate for each sender. However, this rate allocation algorithm is based on a uniformly random packet loss model which has been shown to be inadequate for bursty packet loss due to congestions in the Internet; in addition, it does not use FEC to recover the lost packets. In this section, we show that incorporating FEC results in a fundamentally different rate allocation algorithm from the case where FEC is not incorporated. Specifically we show that use of FEC reduces peak loss rate in a bursty loss environment, and hence improves the overall visual quality as compared to our previous approach in Section 2.3. We begin with a brief discussion on erasure codes and network model.

3.1 Erasure Codes

Erasure codes are a form of FEC [19] used for communication between senders and receivers through a lossy medium. When decoding the encoded data using erasure codes, the receiver is assumed to know the exact location of the lost packets, while this information is not needed in a general FEC technique. Erasure codes are typically used for sending packets through the Internet since the receiver can detect the location of the lost packets by noting the skipped packet sequence number. In a typical erasure code, sender encodes redundant packets before sending both the original and redundant packets to the receiver. Receiver can reconstruct the original packets upon receiving a fraction of the total packets. Standard erasure codes such as the (N, K) Reed-Solomon erasure codes, take K original packets and produces $(N-K)$ redundant packets, resulting in a total of N packets. If K or more packets are received, then all the original packets can be completely reconstructed. Hence, larger $\frac{N}{K}$ ratio leads to higher level of protection for data. In this paper, we use Reed-Solomon codes for our analysis and simulations.

3.2 Network Model

An accurate model for packet loss over the Internet is quite complex. Instead, we model our network as a simple two-state continuous-time Markov chain, which has been shown to approximate the behavior of packet loss over the Internet fairly accurately [11, 20]. A two-state continuous Markov chain with state at time t denoted by X_t where $X_t \in \{g, b\}$, is characterized by μ_g and μ_b. 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 having lost packets is much smaller than that of when the chain is in bad state. A further simplified model assumes that a packet transmitted at time t is successfully received if $X_t = g$, and is lost otherwise. The stationary probabilities of lost packet and received packet are π_b and π_g where $\pi_b = \frac{\mu_b}{\mu_g + \mu_b}$ and $\pi_g = \frac{\mu_g}{\mu_g + \mu_b}$.

We now provide intuition for using a two-state continuous-time Markov model to approximate the packet loss behavior of the Internet traffic. It is known that lost packets in the Internet often happen in bursts. This loss of successive packets is due to the way packets are dropped at the network routers during congestion. Most routers employ First-In-First-Out (FIFO) policy in which, the successive arrived packets are dropped if the network buffer becomes full. Hence, a Markov chain which models a bursty loss environment, approximates the Internet traffic reasonably well. Also the average congestion period during which,the amount of aggregate traffic exceeds the link's capacity, can be thought of as $1/\mu_b$, i.e. the average time that the Markov chain is in the bad state. Clearly, sending more packets during congestion results in larger number of lost packets.

To support the validity of the Markov model, we have performed a number of streaming experiments over the Internet, the results of which are shown in Figure 3. The experiments show results of sending packets from Hong Kong University to UC Berkeley at the rates of 200, 400, and 800$kbps$, respectively. The vertical axis shows the number of lost packets out of every 70 packets, and the horizontal axis shows the packet sequence number. As expected, larger sending rates result in longer loss bursts when the route is in congestion. The largest numbers of lost packets out of 70 packets for 200, 400, and 800$kbps$ are 6, 23, and 31 packets, and

the average loss rates are 0.13%, 0.18%, and 0.21%, respectively. Figure 3(a) indicates that if $RS(70, 64)$ codes are used to protect the data, then all packets sent at the rate of $200kbps$ will be completely recovered, while there will be 5 and 14 instances of irrecoverable loss for sending packets at the rates of $400kbps$ and $800kbps$, respectively. Irrecoverable loss occurs when the number of lost packets out of N encoded packets exceeds $N - K$ or 6 in these experiments. These experiments also indicate that even though the average loss rate is small, a substantial amount of redundancy in erasure codes is needed to recover the data completely. For instance, based on Figure 3(c), a strong $RS(70, 39)$ code is needed to completely recover all the data. In general, FEC codes do not combat bursty loss efficiently. Given the same ratio N/K and the bursty loss characteristics, the efficiency of $RS(N, K)$ code increases as N and K increase. However, the encoding and decoding computational complexities and delay also increase. To alleviate this drawback, interleaving techniques [21] are often used, even though they result in increased delay.
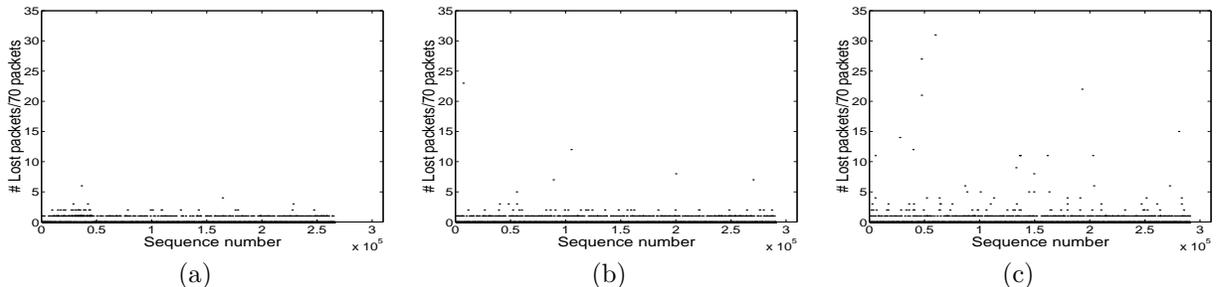

(a)   (b)   (c)

Figure 3: *Hong Kong to U.C. Berkeley, (a) rate = 200kbps; (b) rate = 400kbps; (c) rate = 800kbps*

The above experiments provide an insight on how to combat bursty loss efficiently without introducing increased delay and complexity. Instead of streaming a video at $800kbps$ from the sender at Hong Kong University to a client at U.C. Berkeley, we can use 4 senders, each streaming the video simultaneously at $200kbps$ to the client at U.C. Berkeley. The senders are assumed to be located in such a way that the packet loss along the routes between each sender and the client are uncorrelated. Assume that the routes between all the senders and the client have the same loss behavior as the route between Hong Kong University and U.C. Berkeley; then we expect that the number of instances of irrecoverable loss to be fewer than streaming a video at $800kbps$ from only one sender. The effect of sending packets at a lower rate on multiple independent routes transforms the bursty loss behavior into a more uniform loss behavior, thus increasing the efficiency of FEC techniques. Naturally, given a number of routes each with a different loss behavior, the video bit rate, and the total amount of FEC protection, there should be an optimal partition of sending rates for each route. We address this rate allocation problem in the next section.

3.3   Optimal Rate Allocation

To simplify analysis, we replace the two-state continuous-time Markov chain with an equivalent two-state discrete one. To see the equivalence, note that since we are only interested at the instances when a packet is sent, equations for computing the transition probabilities for the discrete one can be derived as:

$$p_{gg} \triangleq P(X_{n+1} = g | X_n = g) = \pi_g + \pi_b e^{-(\mu_g + \mu_b)\tau} \tag{4}$$

$$p_{gb} \triangleq P(X_{n+1} = g | X_n = b) = 1 - p_{bb}(\tau) \tag{5}$$

$$p_{bb} \triangleq P(X_{n+1} = b | X_n = b) = \pi_b + \pi_g e^{-(\mu_g + \mu_b)\tau} \tag{6}$$

$$p_{bg} \triangleq P(X_{n+1} = b | X_n = g) = 1 - p_{gg}(\tau) \tag{7}$$

where τ is the sending interval. 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, 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. Furthermore, we assume that the total estimated TCP-friendly bandwidth of the two senders, as computed by equation (1), exceeds the required video bit rate.

Our goal is to find the sending rates for the two senders in order to (a) minimize the probability of irrecoverable loss for a given protection level of FEC, and (b) to ensure that each sender sends packets at a TCP-friendly rate. To formally state our rate allocation problem, we use the following notation:

S_{req} Required video sending rate in packets per second

N Total number of packets in FEC block.

K Number of data packets in FEC block.

B_m Estimated TCP-friendly bandwidth for sender m in packets per second

(μ_g^m, μ_b^m) Network parameters for route between sender m and the receiver as defined in Section 3.2

$\lambda = \frac{N}{S_{req}}$ Interval between successive transmitted FEC block in seconds

N_m Number of packets transmitted by sender m during λ seconds

The rate allocation problem can now be stated as follows:

Given S_{req}, N, K, B_m, (μ_g^m, μ_b^m), we want to find N_m for $m = 0, 1$ so as to minimize the probability of irrecoverable loss given by

$$C(K, N_0, N_1) = \sum_{j=N-K+1}^{N_0+N_1} \sum_{i=0}^{j} P(0, i, N_0) P(1, j-i, N_1) \tag{8}$$

subject to

$$N_0 + N_1 = N, \quad \frac{N_0}{\lambda} \le B_0, \quad \frac{N_1}{\lambda} \le B_1 \tag{9}$$

where $P(m, i, N_m)$ is the probability that i packets are lost out of the N_m packets sent by sender m. $C(K, N_0, N_1)$ is the probability that more than $N - K$ packets are lost out of a total $N_0 + N_1$ packets sent by both senders. Since we assume independent packet loss along the two routes, the probability of j lost packets out of $N_0 + N_1$ packets sent by both senders can be written as $\sum_{i=0}^{j} P(0, i, N-0) P(1, j-i, N_1)$. Therefore, the probability of more than $N - K$ lost packets out of $N_0 + N_1$ packets sent is $\sum_{j=N-K+1}^{N} \sum_{i=0}^{j} P(0, i, N_0) P(1, j-i, N_1)$. As indicated in constraints (9), $\frac{N_m}{\lambda}$ is the sending rate of sender m, which is required to be less than or equal to the estimated TCP-friendly bandwidth. Since the sum of the sending rates equals to the required sending rate for the video, we have $N_1 + N_0 = N$. In this paper, we assume that the aggregate TCP-friendly rate is always greater than or equal to the video bit rate. When the bandwidth constraints shown in (9) are not satisfied due insufficient network bandwidth, a scalable video bitstream can be used to stream the video at a lower 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_0 and N_1 such that the constraints (9) are satisfied, and $C(K, N_0, N_1)$, the probability of irrecoverable packet loss is minimized. This search is fast since only N comparisons are required for two senders. The number of comparisons required for M senders are characterized in [22]. In practice, other system factors such as reordering of the received packets, and the number of ports limit the number of senders, thus reducing the size of the search space and computational complexity.

Note that in our rate allocation formulation, the amount of FEC is assumed to be known in advance. The constraints on this amount of FEC are the available bandwidth, complexity, and delay associated with decoding and encoding a FEC block. For applications that can tolerate high delay, more FEC can be used, provided that (a) the available TCP-friendly bandwidth is greater than aggregate bandwidth of the application and FEC overhead, and (b) sufficient computing power for encoding and decoding exists at the sender and receiver.

3.4 Numerical Characterization

To compare the capability to recover lost packets using our optimal sending rate allocation for two senders against that of using one sender, we numerically compute and compare the probabilities of irrecoverable loss across different model parameters (μ_{good}, μ_{bad}) for the two following scenarios. In "two senders" scenario, packets are simultaneously sent along two "loss-independent" routes A and B while in "one sender" scenario, all packets are sent along route A. Recall that $1/\mu_{good}$ and $1/\mu_{bad}$ can be thought of as the average time that a sender is in "good" and "bad" states, respectively. We vary these model parameters in our computations to determine the robustness of the optimal sending rate allocation under different network conditions. For convenience, we refer to the average time that a sender spends in "good" and "bad" states as average good and bad times, respectively. The parameters of the two routes vary as follows. The average good time of routes A and B are identical and they vary from $1s$ to $5s$. The average bad time of route A remains constant at $0.02s$ while average bad time of route B varies from $0.02s$ to $0.2s$. The probabilities that a packet is lost while in "good" and "bad" states are 0 and 1, respectively. The aggregate sending rate of both "two senders" and "one sender" scenario is $500kbps$, and the packet size is set to 500 bytes. In both scenarios, packets are protected using $RS(100, 88)$ codes.

Figure 4(a) shows the probability of irrecoverable loss for for the two sender scenario using our rate allocation algorithm in Section 3.3. The y-axis shows the average good times for both routes A and B ranging from $1s$ to $5s$, while the x-axis shows the average bad time of route B ranging from $0.02s$ to $0.2s$. The z-axis in the same figure shows the probability of irrecoverable loss using optimal rate partition between two routes A and B at different average good and bad times. As an example, the point $(0.05, 2, 0.01)$ in the graph indicates that the minimum probability of irrecoverable loss is 0.01 when the average good times for both routes A and B are $2s$, while the average bad time for routes A and B are $0.02s$ and $0.05s$, respectively. Figure 4(a) indicates that the probability of irrecoverable loss varies mostly with the average bad time while it remains relatively constant with respect to the average good time except when the average good time is small. This observation is intuitively plausible since we would expect a route with long average bad time to have longer bursts of lost packets, leading to higher probability of irrecoverable loss. The z-axis in Figure 4(b) shows N_A, the optimal number of packets out of 100 that should be sent on route A, with the remaining $100 - N_A$ packets sent on route B. This graph shows that as the average bad time of route B increases from $0.02s$ to $0.2s$, more packets should be sent on route A. This result makes sense since the average bad time of channel A is only $0.02s$, and therefore sending more packets on route A will result in smaller probability of irrecoverable loss. An interesting point to note that when the model parameters of both routes A and B are identical, it is optimal to split the sending rates equally between the two routes as indicated in Figure 4(b).

Figure 4(c) shows the probability of irrecoverable loss as a function of model parameters for the "one sender" scenario in which, all the packets are sent on route A. Since no packet is sent on route B, the probability of irrecoverable loss remains constant as the average bad time of route B increases along x-axis. As expected, the probability of irrecoverable loss decreases as the average good time for route A decreases. Even though route A has a lower average bad time than that of route B, sending all packets in the route A, is not always a good idea as shown Figure 4(d). Figure 4(d) shows the ratio of irrecoverable loss probabilities between the case when all packets are sent on route A and the case when the optimal rate allocation is employed to send packets on both routes A and B. When the average bad time of route B is greater than $0.07s$, the performances of the multi-senders and uni-sender schemes are almost identical as the ratio of irrecoverable loss probabilities is approximately 1. If however, the average bad time of route B is less than $0.07s$, it is advantagous to use both routes A and B to send packets at the appropriate rates. For certain model parameters, the irrecoverable loss probability using optimal rate partition scheme is almost 35 times less than that of the uni-sender one.

Naturally, the question arises as to why it is beneficial to also use the "worse" route to send any packet on. By "better" and "worse" routes, we mean one with shorter and longer average bad times, respectively. As noted previously in Section 3.2, sending more packets on a route while it is in the bad state will result in larger number of lost packets. Therefore, when the "better" route A is in the bad state and the "worse" route B is in good state, sending all packets in route A will result in larger number of lost packets than splitting the packets between the two routes. It is true that when route A is in good state and route B is in bad state, splitting packets between the two routes is likely to cause larger number of lost packets than sending all packets in the "better" route A. However, if an appropriately small fraction of packets are sent in route B while it is in bad state, the total number of lost packets per 100 packets is likely to be small enough to be recovered by FEC. Generally, when at least one of the routes is in good state, splitting packets appropriately between two routes will provide a good chance for recovering all the lost packets using FEC. If both routes A and B are in bad state at the same time, then FEC is not likely to be able to recover the lost packets; the probability of this however is quite small. Therefore in most situations, it is advantageous to split the sending rates between the routes.

It can be argued that the optimal scheme is one that can predict accurately the "better" channel at any instant and send all the packets through that channel. However, this scheme is impractical because the bursts of lost packets often last only on the order of tens to hundreds of milliseconds which is too short for any scheme to adapt, given that round trip time is already on the order of hundreds of milliseconds.

4   Simulations Results

In this section, we perform both MatLab simulations and actual Internet experiments to show that our optimal rate allocation scheme results in fewer lost packets and leads to higher visual quality for the streamed video.
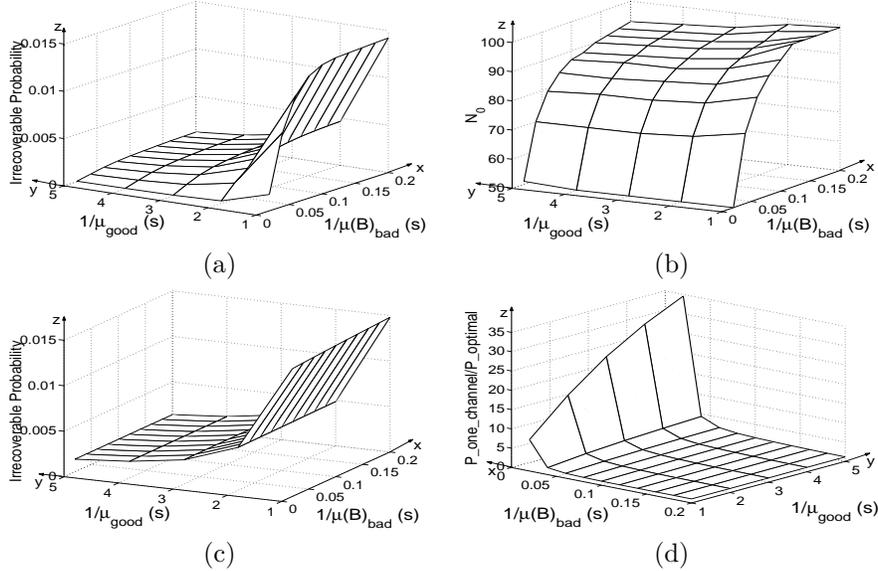
Figure 4: *(a) Probability of irrecoverable loss for RS(100, 88) using optimal partition for two senders; (b) Optimal partition for RS(100,88) for two sender; (c) Probability of irrecoverable loss for RS(100,88) using only one route to send packets; (d) Ratio of irrecoverable loss probability using only one sender to that of multi-senders.*

4.1 MatLab Simulations

To validate our numerical results of the optimal rate allocation, we perform the following two simulations. In the first simulation, we simulate a single sender and receiver by sending all the video packets using a single route in which, the packet loss behavior is modeled as a two-state continuous Markov chain. In the second simulation, we simulate multiple senders and single receiver by sending the video packets along two independent, identical routes similiar to the one in the first simulation. The simulation parameters are shown below

$\mu_{good} = 0.15$, rate at which route's state changes from "good" to "bad"
$\mu_{bad} = 30$, rate at which route's state changes from "bad" to "good"
$S_{req} = 800 kbps$, $720 kbps$ actual video bit rate + $80 kbps$ FEC rate
$packet\ size = 500$ bytes
$N = 100$ total number of packets in FEC block
$K = 90$ number of data packets in FEC block

The video in the simulations is a $1.5 Mbps$ MPEG-1 video sequence taken from MPEG-7 test suite, which is then transcoded using H.263 encoder with error-resilient option at $720 kbps$. The video is then packetized into $500 bytes$ packets, and all packets are further protected by $RS(100, 90)$ codes, making the total video bit rate approximately $800 kbps$. After FEC decoding at the receiver, if there is still an irrecoverable loss, we use a simple error-concealment technique to conceal the visually degraded frames. Basically, the error-concealment technique replaces the lost group of blocks (GOB) of the current frame with GOB of the previous frame, and copies the motion vectors of the lost GOB from the GOB above it. Using optimal rate allocation for two independent, identical routes with the above simulation parameters, we obtain the equal sending rates of $400 kbps$ for each route. Figure 5(a) shows the number of lost packets per 100 versus packet sequence number in the first simulation where packets are sent at $800 kbps$ using only one route. A point above the horizontal line represents an irrecoverable loss event. As seen, there are 7 instances of irrecoverable loss in which the number of lost packets exceeds 10. Figure 5(b) shows the number of lost packets out of 100 versus packet sequence number in the second simulation where packets are sent simultaneously at the optimal rate of $400 kbps$ on each route. In this simulation, there is only one instance where FEC cannot recover the lost packets. Clearly, using the optimal sending rate allocation scheme to send packets on two "loss independent" routes results in fewer lost packets than that of sending all packets on only one route.

Next, we compare the mean squared error (MSE) of pixel values between the sent and the received frames as a function of time. Higher MSE represents lower fidelity of the video. Figure 5(c) shows the MSE

in dB, resulting from both experiments, with the dotted and solid lines representing the MSE from first and second simulations, respectively. Since MSE for a successfully recovered frame is 0 and $\log(0) = -\infty$, Figure 5(c) only shows the instances when MSE is greater than 0. As seen, peaks in MSE closely reflect instances at which irrecoverable loss occur. Visual inspection has shown that these peaks in MSE result in noticeable visual degradation of the video. Hence, the multi-sender scheme results in better visual quality than uni-sender scheme for the streamed video.
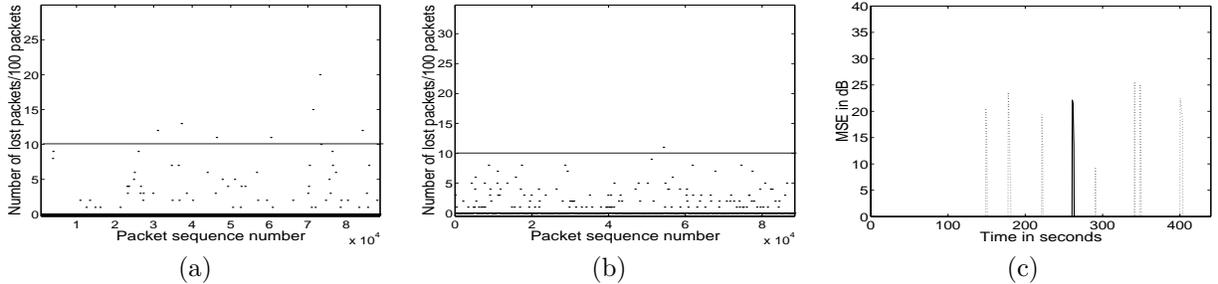

Figure 5: *(a) Number of lost packets per 100 packets using one route; (b) Number of lost packets per 100 packets using two routes; (c) MSE from two simulations*

4.2   Internet Experiments

In addition to MatLab simulations, we have developed an actual system for distributed video streaming with real-time FEC decoding and displaying capabilities. We now demonstrate the effectiveness of the optimal rate allocation scheme in reducing the number of lost packets by performing the following three Internet streaming experiments. In experiment one, a sender at Purdue university streams a H.263 video to a receiver at U.C. Berkeley at the rate of 200 packets per second. In experiment two, 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 three, both senders at Sweden and Purdue university simultaneously stream the video to a receiver at U.C. Berkeley with the optimal rates of 80 packets per second and 120 packets per second, respectively. In all three experiments, the streamed H.263 video has 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 [23] on the traces of packets over a period of one hour offline. 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. All experiments are done one after another within 80 minutes interval to order to keep the network traffic relatively constant for all three experiments.

Figures 6(a) and 6(b) show the number of lost packets per 100 for experiments one and two, respectively. The points above horizontal line in Figure 6 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 one and two where only one sender is used to stream video to receiver. On the other hand, in experiment three as shown in Figure 6(c), when 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. The average packet loss rates for experiments one, two, and three are 0.05%, 0.09%, and 0.08%, respectively. An interesting point to note is that even though the average loss rates in all three experiments are well below 10%, $RS(100, 90)$ code in experiments one and two cannot recover all the lost packet due to the bursty loss nature of Internet.

5   Conclusions and Future Work

In this paper, we propose a novel rate allocation algorithm to be used with FEC in order to minimize the probability of lost packets in a bursty loss environment due to the network congestion. Using both MatLab simulations and actual Internet experiments, we demonstrate the effectiveness of our rate allocation scheme in reducing packet losses, and hence, achieving higher visual quality for the streamed video. Our current system uses an offline algorithm to estimate the network parameters. We are planning to incorporate an online algorithm to estimate the these parameters to reflect the current network conditions. In addition, our rate allocation scheme has not made use of the relative importance of different bits in the video such as
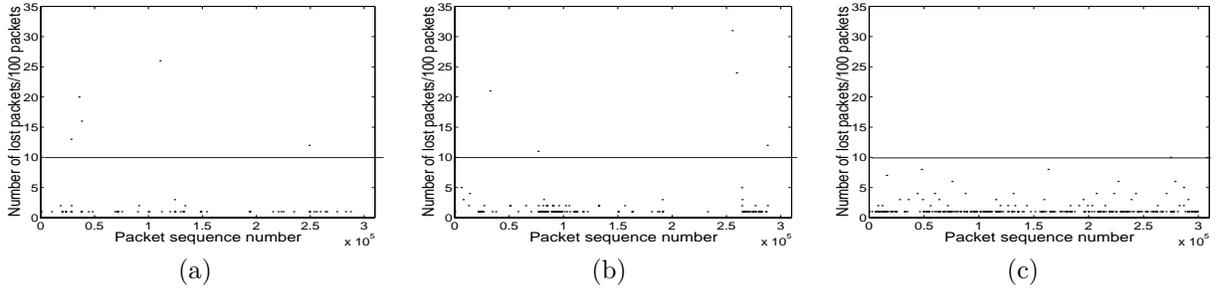
Figure 6: *(a) Video streaming from Purdue university to U.C. Berkeley; (b) Video streaming from Sweden to U.C. Berkeley; (c)Simultaneous video streaming from Sweden and Purdue university to U.C. Berkeley*

motion vectors and texture bits. We speculate that the distortion will be even smaller if the rate allocation scheme can be extended to allocate video bits based on their importance. Also, for live streaming and interactive applications, we are investigating source based routing techniques to send packets on optimal multiple routes from a single sender to the receiver.

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. Zarki, "Broadcast/multicast mpeg-2 video over wireless channels using header redundancy fec strategies," in *Proceedings of The International Society for Optical Engineering*, 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*, October 1999, pp. 401–405.

[6] S. Deering et al., "The pim architecture for wide-area multicast routing," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 153–162, April 1996.

[7] 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.

[8] C. Labovitz, G. Malan, and F. Jahanian, "Internet routing instability," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 515–528, October 1998.

[9] V. Paxson, "End-to-end routing behavior in the internet," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 601–615, October 1997.

[10] T. Nguyen and A. Zakhor, "Distributed video streaming," in *Multimedia Computing and Networking*, San Jose, CA, January 2002.

[11] J. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive fec-based error control for internet telephony," in *Proceedings of IEEE INFOCOM*, 1999.

[12] 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.

[13] 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*, October 1999, vol. 3, pp. 837–841.

[14] 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*, January 2001, vol. 4310, pp. 392–409.

[15] 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.

[16] K. Goyal and J. Kovacevic, "Generalized multiple description coding with correlating transforms," *IEEE Transactions on Information Theory*, vol. 47, pp. 2199–2224, April 2001.

[17] Y. Wang, M. Orchard Vaishampayan, and V. Reibman, "Multiple description coding using pairwise correlating transforms," *IEEE Transactions on Image Processing*, vol. 10, no. 3, pp. 351–366, March 2001.

[18] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measuremen," in *International Conference on Measurement and Modeling of Computer Systems*, June 2000, pp. 145–155.

[19] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *Computer Communication Review*, vol. 27, no. 2, pp. 24–36, April 1997.

[20] 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*, 1999.

[21] B. Wah and D.Lin, "Transformation-based reconstruction for real-time voice transmissions over the internet," *IEEE Transactions on Multimedia*, vol. 1, pp. 342–351, December 1999.

[22] T. Nguyen and A. Zakhor, "Distributed video streaming," *To be submitted to IEEE/ACM Transactions on Multimedia and Networking.*

[23] M. Jordan and C. Bishop, '*An Introduction to Graphical Models.*

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

To compute $C(K, N_0, N_1)$ in Section 3.3, 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 denote

$S_m(n) \in \{g, b\}$ 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

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 the packets according to equations (4) through (7). 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 intially 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

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

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

$$\phi_{ij}^m(0,0) = \left\{ \begin{array}{ll} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{array} \right.$$

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

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.