Overlay Multi-hop FEC Scheme for Video Streaming over Peer-to-Peer Networks[1]

Yufeng Shan[1], Ivan V. Bajic[2], Shivkumar Kalyanaraman[1], and John W. Woods[1]

1/ ECSE Department, Rensselaer Polytechnic Institute, Troy, NY 12180-3590

2/ School of Engineering Science, Simon Fraser University, Burnaby, BC, V5A 1S6, Canada

Abstract

Overlay networks offer promising capabilities for video streaming, due to their support for application-layer processing at the overlay forwarding nodes. In this paper, we focus on the problem of providing lightweight support at *selected* intermediate overlay forwarding nodes to achieve increased error resilience on a *single* overlay path for video streaming. We propose a novel *Overlay Multi-hop FEC* (OM-FEC) scheme that provides FEC encoding/decoding capabilities at intermediate nodes in the overlay path. Based on the network conditions, the end-to-end overlay path is partitioned into segments, and appropriate FEC codes are applied over those segments. Architecturally, this flexible design lies between the end-to-end and hop-by-hop paradigms, and we argue that it is well suited to peer-based overlay networks. We evaluate our work by both simulations and controlled Planet-Lab network experiments. These evaluations show that OM-FEC can outperform a pure end-to-end strategy up to 10-15 dB in terms of video PSNR, and can be much more efficient than a heavyweight hop-by-hop strategy.

Index terms

peer-to-peer networks, overlay networks, video streaming, and forward error correction

[1] This work is partly supported by the ARO grants DAAD19-00-1-0559 and W911NF-04-1-0300

I. INTRODUCTION

Providing high-quality video streaming over the current best-effort Internet is a challenging problem due to video's characteristics such as high bit-rate, delay variation sensitivity, and loss sensitivity. Streaming video and other media has been intensively studied in the past several years. Most recently, *peer-to-peer* (P2P) architectures and overlay networks are gaining attention. Padmanabhan *et al*. [1] discussed the problem of distributing media content, both live and on demand, to a large number of receivers in a scalable way. They propose a solution called *CoopNet* for content distribution that combines aspects of infrastructure and peer-to-peer based content distribution, wherein clients cooperate to distribute content, thereby alleviating the load on the server. CoopNet builds multiple distribution trees spanning the source and all the receivers, for its multiple description coded media content. Yeo *et al*. [2] propose an application-level multicast overlay using peering technology and a lightweight gossip (i.e. active probing) mechanism to monitor prevailing network conditions and improve tree robustness. Clients can dynamically switch to other parents if they experience a poor QoS. In [3], Chu *et al*. explore the possibility of video conferencing using an overlay multicast architecture. Their constructed overlay-spanning tree is optimized according to measurements of available bandwidth and latency among users, and can be modified by the addition of good links and the dropping of the poor links. The main goal of *resilient overlay network* (RON) [4] is to enable a group of nodes to communicate with each other in the face of problems with the underlying Internet paths connecting them. RON detects such problems by aggressively probing and monitoring the paths connecting its nodes. If the underlying Internet path is the best one, that path is used and no other RON node is involved in the forwarding path. If the Internet path is not the best one, RON will forward the packet by way of other RON nodes.

Performance characteristics of peer-based overlay networks are likely to be very different and highly variable as regards the traditional Internet or even managed overlay networks. However, their massive diversity, i.e. multiple overlay paths can be harnessed, can compensate for the performance variability of any one path [8, 15]. In addition, lightweight support at intermediate nodes can improve single path performance. In this paper, we focus on the latter problem and propose a novel *Overlay Multi-hop FEC* (OM-FEC) scheme for video streaming over peer-based overlay networks. The OM-FEC scheme dynamically partitions the end-to-end overlay path into segments according to its error characteristic, and provides appropriate error resilience over each segment. Here, we do not focus on overlay path construction and routing problems. Rather, we assume a peer-based overlay path is pre-constructed and we will focus on how to efficiently utilize it. We will henceforth use the term "overlay path" to mean the constructed path over a peer-to-peer network.

A. Scope and Assumptions

Most prior work on video over peer-to-peer/overlay networks has focused on massive video data distribution or video conferencing using application-layer multicast. In contrast, our objective is to revisit the problem of efficiently utilizing the resources of a *single* overlay path. Our approach operates at small time-scales in the data-plane, and can be combined with overlay routing and topology management approaches that operate in the control-plane and over larger time-scales [4]. In this sense, OM-FEC is complementary to the prior work where resilience is provided using overlay routing methods. We assume that we can construct an overlay path with higher bandwidth [4] than the default Internet route by using P2P techniques such as Chord [7] or Pastry [12], to obtain a set of intermediate forwarding nodes as shown in Figure 1. In the figure, the dashed lines represent the virtual links between overlay nodes and the solid lines represent the

default Internet path. In this paper, we refer to one virtual link as one "overlay hop". The quantities B_i, P_i, and RTT_i represent, respectively, the bandwidth, loss rate, and round-trip time of the *i*-th hop.

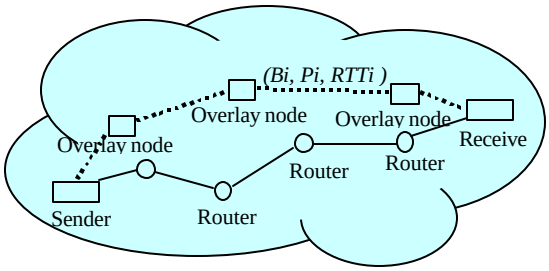


Figure 1: Streaming video using overlay network

B. Motivation

The advantage of application-layer overlay networks arises from two fundamental properties: (1) overlay nodes have capabilities of computation and storage power that are far beyond basic store and forward operations, and (2) the overlay topology can be constructed and manipulated to suit one's purposes. Based on these two considerations, we argue that applying error correction purely end-to-end or hop-by-hop in an overlay network is a sub-optimal strategy. For example, in Table 1 (a), we list a set of possible bandwidth and loss rates on a 6-hop overlay path, where B_i and P_i are, respectively, the bandwidth and loss rate on the *i*-th hop.

hop	1	2	3	4	5	6
B_i	360	325	315	400	600	800
P_i	3.5%	4%	4%	3.5%	2%	1%

Table1: (a) An example of possible bandwidth (Kbps) and loss rate of an overlay path

FEC Method	end-to-end	OM-FEC	hop-by-hop
Throughput	258	302	302
Path loss rate	18%	18%	18%

(b) Path throughput (Kbps) : OM-FEC vs End-to-End and Hop-by-Hop FEC

Given FEC using Reed-Solomon (RS) erasure-correcting codes, in order to fully recover lost packets, the end-to-end based FEC scheme would have to be designed based on the end-to-end

available bandwidth 315 Kbps and the end-to-end loss rate, approximately 18% in this case. Thus, the overall data throughput is reduced to around (1-0.18)×315 = 258 Kbps. On the other hand, if a heavyweight hop-by-hop based FEC scheme is used, the end-to-end data throughput will be 302 Kbps with the same path loss rate. However, the hop-by-hop FEC scheme induces more per-hop delay and uses more computational power of the overlay nodes than is necessary. Our proposed OM-FEC tries to minimize the overall computational complexity at the intermediate nodes, i.e. uses as few overlay nodes as possible to do FEC encoding/decoding, while still maintaining near highest video quality that can be obtained over the overlay path. OM-FEC partitions the whole overlay path into *segments* and performs FEC over each segment.

The paper is organized as follows. In Section II we describe our protocol, rate allocation scheme, and algorithms for the novel OM-FEC strategy. Then, we describe the simulation and real Internet experiments in Section III. Finally, in Section IV we conclude and suggest possible extensions.

II. OVERLAY MULTI-HOP FEC (OM-FEC)

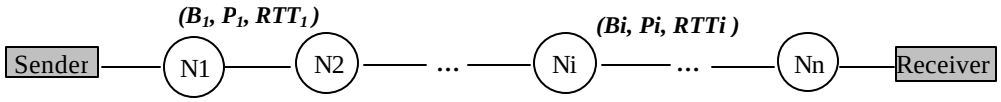


Figure 2: A sample overly path with n intermediate nodes

In our video streaming system, the video server sends a fixed rate bit stream to a user through an overlay path as shown in Figure 2. There are n intermediate overlay nodes (denoted as Ni), and hop i has available bandwidth B_i, packet loss rate P_i, and round trip time RTT_i. In order to protect packets from channel loss, forward error correction (FEC) codes are deployed. Obviously, applying FEC over each hop results in the best video quality at the receiver, but also the most

intermediate FEC encoding/decoding computation. On the other hand, an end-to-end based FEC scheme results in the worst video quality but the least FEC computation. Our objective is to efficiently utilize the resources of this overlay path and to reduce the overall FEC computational complexity while still maintaining near highest video quality.

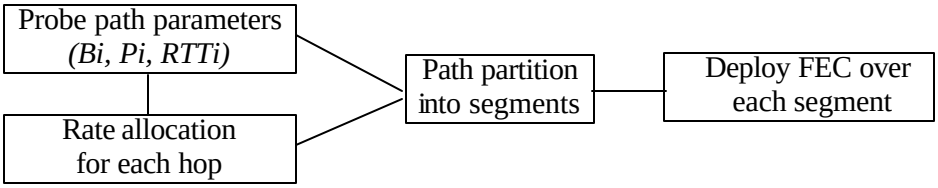


Figure 3: OM-FEC building blocks and the relationship of these blocks

The basic building blocks of the OM-FEC scheme include an algorithm to determine optimal partitioning of the overlay path, a rate allocation algorithm for allocating appropriate FEC rate for different hops of the overlay path, and the actual deployment of the FEC on the path as shown in Figure 3. The video server actively sends out a probe packet every ?*t* time units. Each overlay node measures the loss rate and round trip time (*RTT*) of its related hop using this small probe packet. The obtained per-hop *RTT* and loss rate estimates are used to infer the TCP-friendly [5] available bandwidth of each hop. With this available bandwidth and loss rate, the FEC rate for each hop can be calculated. However, FEC coding need not be implemented at each hop. Our server runs a greedy algorithm to partition the overlay path consistent with the above FEC rate estimates, so that the overall computational complexity at intermediate hops is minimized without sacrificing FEC based resilience gains. Partitioning splits the overlay path into segments, and separate FEC encoding/decoding is employed over the segments. Hence, only the boundary nodes between segments are involved in FEC encoding/decoding. When this path partition algorithm produces a single segment (equivalent to the entire end-to-end overlay path), then FEC is designed based on the end-to-end network characteristics, and the overlay nodes

simply receive and forward data and FEC packets on to the destination. The decision made by the server is conveyed to every node by a small command packet sent out from the server, so each node knows what it should do after it receives a command packet. The following sections outline the details of our OM-FEC scheme step-by-step.

A. Probe network parameters

Given the overlay path shown in Figure 2, the server first needs to know the available resources of the path and then must decide how to efficiently use these resources. In OM-FEC, as mentioned above, an active probing method is used to estimate the round trip time and loss rate of each hop. In order to synchronize overlay parameter calculation and reduce overhead bandwidth, the sender uses a small active probing packet to synchronize the estimation procedure. The probe packet is sent from the server every time interval $?t$. Each overlay node processes the probe packet, and calculates the loss rate and the round trip time.

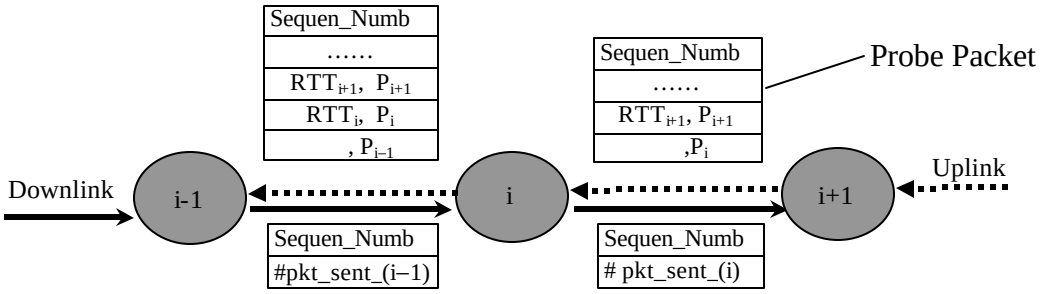


Figure 4: Probe packet information and processing; Server sends out a probe packet to downlink, the collected information of each hop is conveyed back to server along uplink

The structure of the probe packet is shown in Figure 4 where the circles denote overlay nodes, *Sequen_Numb* is the sequence number of the probe packet, *#pkt_sent*(*i*) denotes the number of packets the *i*-th node sent. The parameters {RTT_i, P_i} denote round trip time and loss rate of *i*th hop, respectively. This probe packet passes through all the nodes of the constructed overlay path. For *downlink* (link from server to receiver) path-parameter estimation, node *i* caches the probe

packet from its *up node*, replaces item *#pkt_sent*(*i*–1) with its own *#pkt_sent*(*i*), and then forwards the probe packet to the next node (*i*+1). Each node records the time T_{send} - the instant when it sends the probe packet to its *down node*, and this parameter is later used for round trip time calculation. With the received item *#pkt_sent*(*i*–1) and the measured received data packets *#pkt_recvd_from*(*i*–1), the *i*-th node can calculate the loss rate of the (*i*–1)-th link as $P_{i-1} = \frac{(\#\, pkt\_sent(i-1)) - (\#\, pkt\_recvd\_from(i-1))}{\#\, pkt\_sent(i-1)}$. The probe packet is fed-back to the server after it reaches the receiver. This feedback packet collects all information from these overlay nodes while going back to the server along the *uplink* (link from receiver to server). As for round trip time estimation, as soon as a probe packet arrives at the *i*-th node from the (*i+1*)-th node, the *i*-th node obtains the arrival time of this packet T_{arrive}, and then calculates the round trip time for the *i*-th hop as follows:

$$RTT_i = T_{arrive} - T_{send} - \sum_{j=i}^{j=n} RTT_j \qquad (1)$$

The *i*-th node attaches its calculated loss rate of (*i+1*)-th hop and the round trip time of the *i*-th hop to the probe packet and then forwards it to the (*i*–1)-th node as shown in Figure 4, until it arrives the server. Thus, the server has the loss rate P_i and RTT_i of each hop along the overlay path.

B. Rate allocation strategy of OM-FEC

At a certain time *t*, the server estimates the available bandwidth $B_{avail}(i,t)$, of each hop by using the information brought back by the probe packet. Then, OM-FEC uses a very simple method to allocate the available bandwidth for both FEC and video data on each hop: for the *i*-th hop, the algorithm assigns a portion of the available bandwidth $B_{avail}(i, t)$, to video data $B_{data}(i,t)$; and the remaining bandwidth is assigned to FEC $B_{FEC}(i,t)$, until either the desired FEC rate $B_{req}(i,t)$, is

met or the available bandwidth budget is exhausted. In an extreme case, if $B_{avail}(i, t) \le B_{data}(i, t)$, all the available bandwidth is assigned to the video data. The main goal of the rate allocation scheme is to find $B_{data}(t)$ for the whole path and $B_{FEC}(i,t)$ for each hop based on its measured loss rate and round trip time, for further utilization by OM-FEC (to decide what kind of FEC scheme should be deployed, OM-FEC, end-to-end FEC, or hop-by-hop FEC)

The $B_{avail}(i, t)$ is the estimated *TCP-friendly bandwidth* of the i-th hop and can be calculated using [5]:

$$B_{avail}(i,t) = \frac{S}{RTT_i\sqrt{\frac{2p(i,t)}{3}} + T_{rto-i}(3\sqrt{\frac{3p(i,t)}{8}})\,p(i,t)(1+32p(i,t)^2)} \qquad (2)$$

where S is the packet size in bytes, RTT_i is the estimated RTT of the i-th hop in seconds, T_{rto-i} is the TCP timeout of i-th link, and $p(i, t)$ is the estimated loss rate of the i-th link.

After the available bandwidths $B_{avail}(i, t)$, of all the hops are calculated, the end-to-end bandwidth $B_{e2e}(t)$, from source to receiver is the minimum of the per-hop TCP-friendly bandwidths of the overlay paths, i.e.

$$B_{e2e}(t) = \min_{1\le i\le n}\{B(i,t))\}\,. \qquad (3)$$

Thus, the server decides the bandwidth allocated to video data for each hop as:

$$B_{data}(t) = \min(\,B_{e2e}(t), B_{fixed}) \qquad (4)$$

We use systematic $RS(n_i,k_i)$ erasure codes to protect packets from channel losses on hop i. In order to fully recover the lost packets from hop i, the number of parity packets generated should follow

$$\frac{n_i - k_i}{n_i} \ge p(i,t)\,. \qquad (5)$$

Therefore, given the allocated video data bandwidth $B_{data}(t)$, and the estimated loss rate $p(i,t)$ on each hop, the required minimum FEC bandwidth $B_{req}(i,t)$ to fully recover the lost packets of each hop can be given as:

$$B_{req}(i,t) = \frac{B_{data}(t)}{1-p(i,t)} - B_{data}(t) = \frac{p(i,t)}{1-p(i,t)} B_{data}(t) \ . \quad (6)$$

Based on our statement at the beginning of this section, we can obtain the allocated bandwidth to FEC on each hop:

$$B_{FEC}(i,t) = \min\{ B_{req}(i,t), (B_{avail}(i,t) - B_{data}(t))\} . \quad (7)$$

If the inequality (8) holds for all the hops, that means none of the allocated FEC bandwidth of these hops can fully recover the lost packets on its link. Then FEC should be added hop-by-hop, and in this case, OM-FEC works the same as hop-by-hop FEC, i.e. every intermediate node performs an FEC encoding/decoding computation,

$$B_{FEC}(i,t) \le B_{req}(i,t), \forall i = 1,2,...,n . \quad (8)$$

Given the available loss rate on each hop $p(i,t)$, the end-to-end loss rate $p_{e2e}(t)$ can be estimated using:

$$p_{e2e}(t) = 1 - \prod_{i=0}^{i=n} (1 - p(i,t)) \quad . \quad (9)$$

Similarly to (6), the required end-to-end FEC bandwidth $B_{e2ereq}(t)$ to fully recover the end-to-end loss rate can be calculated as:

$$B_{e2ereq}(t) = \frac{p(t)}{1-p(t)} B_{data}(t) . \quad (10)$$

If the inequality (11) holds, it means the available end-to-end bandwidth $B_{e2e}(t)$ is large enough for both video data $B_{data}(t)$, and end-to-end required FEC bandwidth $B_{e2ereq}(t)$, then OM-FEC

adds FEC end-to-end, i.e. works the same as an end-to-end FEC scheme. No intermediate overlay nodes need FEC encoding/decoding.

$$B_{e2ereq}(t) \le B_{e2e}(t) - B_{data}(t) \tag{11}$$

In intermediate cases between the end-to-end and hop-by-hop extremes, OM-FEC must partition the overlay path.

C. Overlay Multi-hop FEC (*OM-FEC*)

We want to minimize the overall FEC computational complexity at the intermediate nodes, while still maintaining near highest video quality, i.e. the expected video distortion of using OM-FEC, $E[D_{OM\text{-}FEC}]$, should be the same as by using hop-by-hop FEC, $E[D_{hop\text{-}by\text{-}hop}]$. In order to maintain high video quality, we use the rate allocation algorithm described in Section II.B. To minimize the computational complexity of the overlay nodes, OM-FEC should use as few nodes as possible for the FEC encoding/decoding. The OM-FEC partition algorithm should find the minimum number of segments $N_{segment}$, of the overlay path. Mathematically, the problem can be stated as follows:

$$Minimize(N_{segment})$$

$$\text{Subject to} \quad \begin{cases} E[D_{OM-FEC}] = E[D_{hop-by-hop}] \\ B_{data}(t) = \min(B_{e2e}(t), B_{fixed}) \end{cases}. \tag{12}$$

Given a certain bitstream with bit rate at $B_{data}(t)$, OM-FEC attempts to find the best partition of the overlay path, i.e. the minimal number of partitioned segments $N_{segment}$, that can maintain the same video quality as hop-by-hop. We use a forward partition to find a good partition of the path.

In order to reduce computational burden, OM-FEC partitions the overlay path into segments according to the characteristics of each hop as shown in Figure 5. For example, the OM-FEC algorithm partitions the overlay path into *N* segments, which are the first *J* nodes as segment1, the next *L* nodes as segment2 and the last *M* nodes as segment*N*, respectively. FEC is then deployed over each segment. Parameters *J*, *L*, …, *M* will be dynamically determined by the OM-FEC algorithm as explained below.

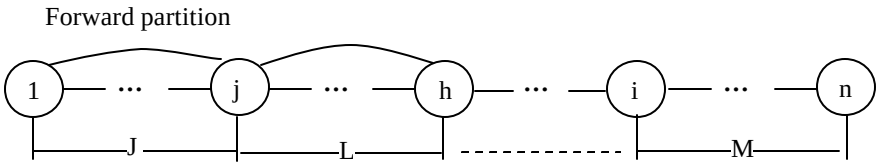


Figure 5: Overlay path is partitioned into segments by OM-FEC to reduce computational complexity at intermediate nodes, only boundary nodes perform FEC encoding/decoding, circles are overlay nodes;

Since OM-FEC does a forward search to find the best partition of the overlay path, the server should be the start node to partition the overlay path. The forward partition is defined as partitioning the overlay path along the direction from server to receiver. In Table 2, we define terms which will be used in the partition algorithm.

Terms	Meaning	Calculation equations
$B_{FEC(start+i)}$	allocated FEC bandwidth for the segment from start node to (start+*i*)th node	(2) – (7)
$B_{req(start+i)}$	required FEC bandwidth for the segment from start node to (start+*i*)th node	(9) - (10)
$N_{segmenf}$	number of segments partitioned	

Table 2: terms used in partition algorithm

The computation unit for this part is based on a segment. For instance, $B_{FEC(start+i)}$ is calculated based on the method of Section II.B and (2)–(7), but considering the segment from the *start* node to (*start*+*i*)-th node as one virtual link and using the accumulated loss rate and available bandwidth of this virtual link.

The Forward Partition algorithm is described by the following pseudo code given the server as the starting node.

```
//forward partition algorithm
Start = 0;    N_segment =   1     //begin calculation from the server
For (i = 1; i <= n; i++){
   // calculate the FEC bandwidth from start node to (start+i)-th node
   Calculate B_FEC(start+i);
   // calculate the FEC bandwidth to the segment from start node to (start+i+1)th node
   Calculate B_FEC(start+(i+1));
   // find a boundary node to partition the overlay into segments
   If ((B_FEC,(start+i) = B_req(start+i) ) && B_FEC,(start+i+1) < B_req(start+i+1))){
      Start node to (start+i)-th node is partitioned as one segment;
      N_segment ++
      FEC is deployed over this segment, the FEC bandwidth allocated to this segment is B_FEC,(start+i);
      Start = Start+i ;   // start from the boundary node to partition the rest of the path
   }
}
```

The server runs this algorithm to partition the overlay path into segments and deploys FEC over each segment. The decision is then conveyed to all intermediate nodes by a small command packet. For each boundary node, the command packet contains a 3-byte field specifying the node ID, and the *n* and *k* parameters of the chosen RS (*n*, *k*) code. The nodes whose IDs are not listed in the command packet will simply forward all the packets they receive, without FEC

coding/decoding. Thus, each node knows what it should do after it receives the command packet. Based on the OM-FEC strategy, the largest segment could include all the nodes of the overlay path (same as the end-to-end scheme), and the smallest segment could be one hop (i.e. hop-by-hop). In other words, OM-FEC is an adaptive strategy that tunes the architectural complexity between the extremes of end-to-end and hop-by-hop operation.

D. Feasibility of Intermediate FEC coding/decoding computation

In OM-FEC, since intermediate nodes perform FEC encoding/decoding, we would like to evaluate the limitations of applying FEC codes in video streaming and the feasibility of the decoding/coding computation at the intermediate nodes. The RS(n, k) encoder takes k data packets and generates $n - k$ parity packets. Given the position of the lost packets, the RS decoder can reconstruct up to $n - k$ lost packets out of a total of n packets. Hence, a larger ratio n/k leads to a higher level of protection for the video data. In a video steaming system, k cannot be chosen arbitrarily, since the video data is time sensitive. Larger values of k imply longer delays at the receiver side. The maximum value of k is related to the bit rate of the encoded video bit stream, packet size, and the buffering time at the receiver side. Let the encoded bit rate be b bps, the packet size be h bytes, and the receiver buffer size be l seconds. If the receiver buffer is full, the total amount of bits in the buffer is lb. The amount of bits in a network packet is $8h$, so the total amount of packets in buffer is $lb/8h$. If FEC is deployed over k packets, the receiver needs to wait for the arrival of at least k packets prior to RS(n, k) decoding. Therefore, we need $k \leq lb/8h$. Using a systematic code, the encoder picks groups of k source data symbols to generate $n - k$ parity symbols. Every source data symbol is used $n - k$ times, so we can expect the encoding time to be a linear or approximately linear function of $n - k$. Since our system relies on real-time FEC encoding/decoding, it is necessary to evaluate the performance of the RS codec. We tested

our implementation of the RS codec (based on Phil Karn's RS codec [10]) on a Dell PC with Pentium 4 CPU at 2.0 GHz, with 256 MB RAM, running Linux RedHat 8.2, with $n = 255$, and k variable. The time needed to produce $n - k$ parity packets, given k data packets, is shown (in ms) in Table 2, for various values of $n - k$ and packet size.

$n - k$	5	10	15	20	25	30	35	40
256 Bytes/packet	1.1	1.9	2.2	3.3	3.9	4.3	4.9	5.4
512 Bytes/packet	2.0	3.7	4.3	6.5	7.8	8.6	9.8	10.8
1024 Bytes/packet	4.1	7.3	8.6	13.0	15.6	17.2	19.5	21.6

Table 3: The RS encoding time (in ms) as a function of n – k and packet size.

From Table 3, we observe that very high FEC encoding rates can be achieved even on commodity PCs (which would most likely be the peers of the overlay network). For example, the encoding bit rate of the RS (255,245) code can be up to 274 Mbps at packet size 1024 bytes, and this code can recover lost packets at random loss rates up to 3.92%. Erasure codes tested in [11, 16] gave similar results. Since the decoding process is much faster than encoding, we do not list decoding test results here.

III. RESULTS

We now demonstrate the effectiveness of OM-FEC by comparing it with end-to-end based FEC and hop-by-hop based FEC in both simulations and controlled real Planet-lab network [6] experiments. Based on the OM-FEC algorithm, if the available end-to-end bandwidth is large enough for both video data and FEC, then OM-FEC works in end-to-end mode. In case of severe network congestion where none of the hops have enough bandwidth for both video data and FEC, OM-FEC partitions the overlay path into the smallest segments (i.e. hop-by-hop) and works the same as does hop-by-hop FEC. In this scenario, both hop-by-hop FEC and OM-FEC

outperform end-to-end FEC in terms of video quality. Here, we focus our simulations and experiments on the cases that lie between the above two extremes. We first test the bandwidth efficiency of OM-FEC versus the end-to-end scheme. This is followed by video simulations that compare the performance of OM-FEC against both end-to-end and hop-by-hop FEC. Finally, we perform a controlled Planet-lab network video experiment. We expect that, as the number of hops increases and the variation of their loss rates becomes larger, OM-FEC will outperform the end-to-end FEC scheme. This expectation is confirmed by our simulations and experiments. In this section, all the curves are the averages of at least ten simulation (experimental) runs.

A. Simulations -- bandwidth efficiency

In this section we compare OM-FEC and end-to-end FEC in terms of their provided video throughput. We assume the task is to transmit a video encoded at 512 Kbps (which also represents the highest possible video throughput) through a sequence of overlay nodes. The simulation configuration is shown in Figure 6. The topology includes one sender, one receiver, and three intermediate overlay nodes, with L1 through L4 denoting the overlay hops or links. Similarly to [13] and [14], we use the 2-state Markov (Gilbert) model to simulate packet loss on each hop. The sender sends out video packets through the three overlay nodes to the receiver, and the feedback information is fed-back via the same nodes but in the reverse direction. The probing packet is sent from the server once every 100ms.

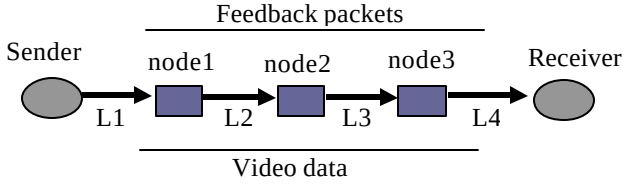


Figure 6: Simulation configuration for bandwidth efficiency.; We vary the loss rates on each hop L1 - L4 and compare the video throughput of OM-FEC vs. the end-to-end scheme.

We begin our simulation by starting the network in a state of slight congestion. The simulation parameters are shown in Table 4. The average burst lengths were in the range of 2-3 packets.

Test	Basic Test		Test A		Test B	
	Lossrate	RTT (ms)	Lossrate	RTT (ms)	Lossrate	RTT (ms)
L1	[1% to 2%]	10ms	[1% to 2%]	10ms	[2% to 3%]	10ms
L2	[1% to 4%]	30ms	[3% to 5%]	30ms	[3% to 6%]	30ms
L3	[3% to 5%]	10ms	[3% to 5%]	10ms	[3% to 6%]	10ms
L4	[2% to 4%]	20ms	[2% to 4%]	20ms	[3% to 4%]	20ms
RS(n, k)	k = 80, n is variable					
Network	conditions change every 300ms					
Video	Encoded video bitrate = 512 Kbps					

Table 4: Simulation parameters for three different tests: Basic Test, Test A, and Test B. Loss rates are randomly chosen from their defined range.

We set the round trip time and range of packet loss rates for each hop. The network condition is changed every 300 ms. At time t = 0, the sender begins to send out video data to the receiver. Information gathered by the probe packet from each hop is fed back to the sender. For the Basic Test, the sender calculates the available bandwidth of each link as shown in Figure 7(b), according to the measured loss rate (Figure 7(a)) and round trip time on each hop. The sender determines what kind of FEC scheme should be deployed for the current network condition. OM-FEC uses bandwidth more efficiently in case of network congestion as shown in Figure 8, where video throughput is defined as bandwidth occupied by the video data. To test our approach in a heavier congestion condition, we increase the loss rate of the links in the simulation setup shown in Table 4, for Test A and Test B. In Figure 9, we can see that OM-FEC outperforms the end-to-end scheme by a large margin at severe congestion. For end-to-end FEC, the increased end-to-end loss rate results in more FEC overhead over the entire overlay path. On the other hand, OM-FEC only considers the related segments where loss rate increases. Thus, the OM-

FEC scheme has better performance than the end-to-end FEC at severe congestion in terms of bandwidth utilization.

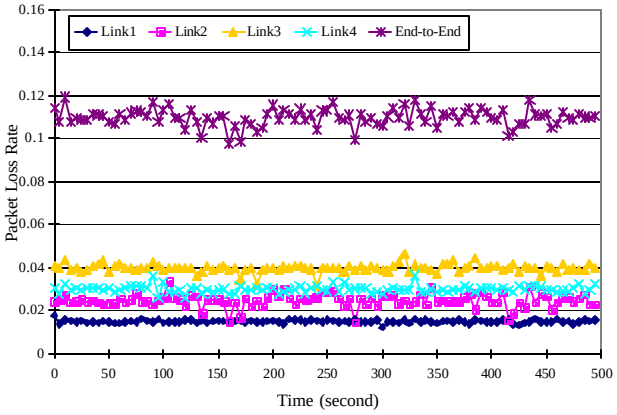
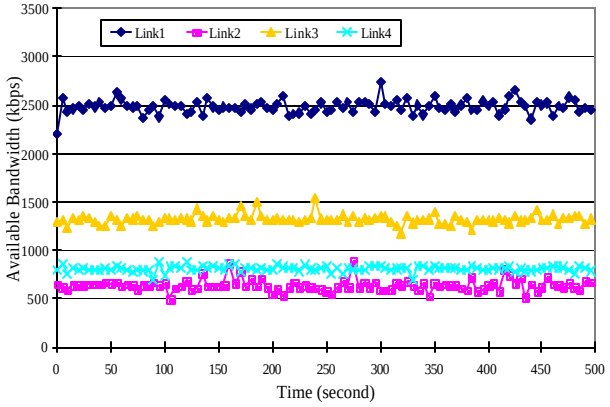


Figure 7: (a) packet loss rate on the overlay path



(b) available bandwidth of each hop

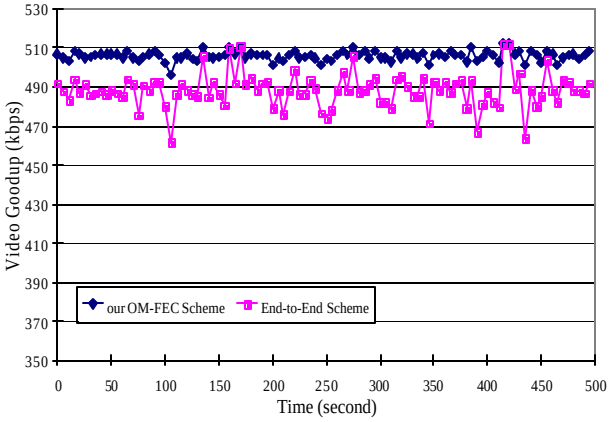


Figure 8: video throughput OM-FEC vs. end-to-end scheme

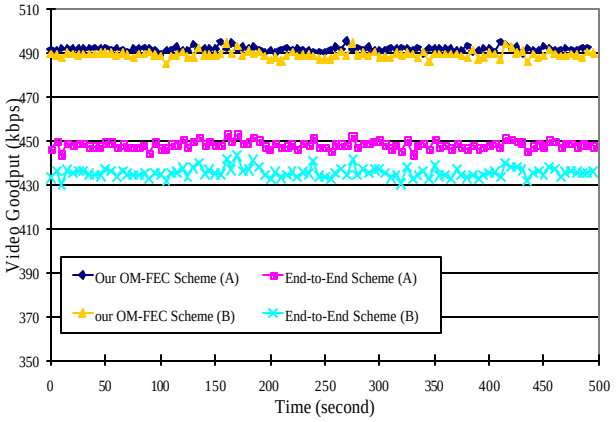


Figure 9: video throughput of OM-FEC vs. end-to-end scheme in Test A and Test B

B. Video simulations

In this part, we use a more complex overlay path to compare the performance of OM-FEC versus end-to-end and hop-by-hop. The path is shown in Figure 10, where ten hops are denoted as L1-L10. The packet loss rate for each hop is randomly chosen in the range [0.5%, 1.5%], thus the overall path loss rate is approximately in the range [5%, 15%]. The video sequence is *Foreman*, QCIF resolution, 30 frames per second (fps). The video bitstream is encoded using an H.263+ encoder with error-resilient option at 1530 Kbps, with intra frame refresh at every second. For

simplicity, the available bandwidth of each hop is fixed at 1656 Kbps, therefore, a maximum 126 Kbps bandwidth can be allocated to FEC, which can recover lost data up to a 7.6% packet loss rate. The network packet size is 512 bytes. Regarding the choice of *RS*(*n*,*k*) FEC codes, we fix *k*=85, while *n* is determined by the loss rate and the available bandwidth. The network conditions change every 300 ms. The probe interval is set to 100 ms.

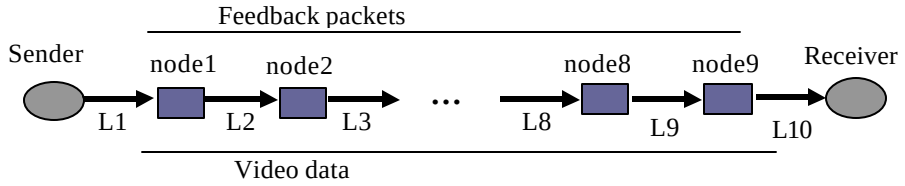


Figure 10: video simulation path configuration, with varying loss rate on each hop (*L1-L10*). *We compare the performance of OM-FEC vs. end-to-end and hop-by-hop*

Based on this setup, the server sends out video bit-team using OM-FEC, hop-by-hop based FEC, and end-to-end based FEC. The results are shown at Figure 11. Since each hop has a packet loss rate only between 0.5% and 1.5%, the hop-by-hop based FEC scheme has enough bandwidth for FEC to recover almost all the packet losses except for a very few observed burst losses. On the other hand, end-to-end FEC deploys FEC based on the end-to-end loss rate which is approximately ranging in [5%, 15%]. However, due to the limited available bandwidth, the maximum end-to-end FEC scheme can only recover a loss rate under 7.6%. Therefore, we observed very bad video quality on the end-to-end based FEC scheme. Our proposed OM-FEC partitions the overall path into segments based on the available bandwidth and loss rate in order to acquire the same video quality as hop-by-hop based FEC, with less computational complexity. OM-FEC partitions the overlay path in such a way that the added FEC codes can almost always fully recover the packet loss of each segment. We observed similar video results to those of hop-by-hop based FEC. The drops in the PSNR curve in OM-FEC are due to burst losses.

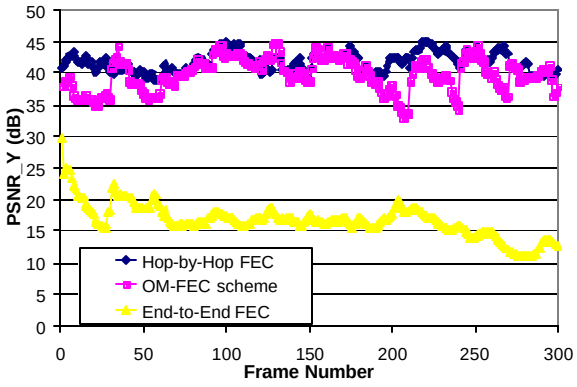


Figure 11: OM-FEC vs. hop-by-hop FEC and end-to-end FEC

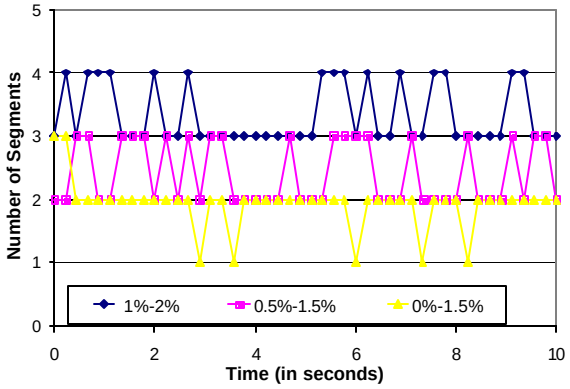


Figure 12: several sample partitionings of OM-FEC

In Figure 12, we show several partition samples of OM-FEC at different per hop loss rates. Though OM-FEC uses fewer nodes to do the FEC computations, it can achieve similar video performance to that of hop-by-hop FEC. In Table 5, we compare the computational complexity of OM-FEC with hop-by-hop and end-to-end. Obviously, the number of segments in the end-to-end scheme is always 1 (no intermediate node is involved in FEC computation) and the number of segments in hop-by-hop is always 10 (each overlay node does FEC decoding/re-encoding). On the other hand, OM-FEC partitions the overlay path based on the loss rate and the available bandwidth. It uses fewer nodes than hop-by-hop based, but with nearly the same performance in terms of video quality in the receiver side.

Per-hop loss range	Approximate total loss rate range	Number of OM-FEC segments	Avg. number of OM-FEC segments (in 10 runs)	Number of hop-by-hop segments	Number of end-to-end segments
0.0% -- 1.5%	0.0 -- 15%	1-2	1.94	10	1
0.5% -- 1.5%	5% -- 15%	2-3	2.27	10	1
1% -- 2%	10% -- 20%	3-4	3.20	10	1

Table 5. Computational complexity comparison of the three FEC schemes

C. Controlled Planet-Lab network experiments

We also implemented our protocol over the real Internet using the Planet-Lab infrastructure [6]. The implementation includes an overlay agent and the protocol itself. Our overlay agent can run at any Linux Planet-Lab node. Each agent forwards a video packet to the next node until it arrives at the destination. The experimental topology is the same as Figure 5 and the Planet-Lab nodes involved are listed in Table 6.

Server	nima.eecs.berkeley.edu
Node1	planetlab1.flux.utah.edu
Node2	planetlab-1.cmcl.cs.cmu.edu
Node3	planetlab1.cs.cornell.edu
Receiver	video.testbed.ecse.rpi.edu

Table 6: Nodes involved in Planet-Lab experiments

In the experiments described in this section we measure the objective video quality at the receiver in terms of the Peak Signal-to-Noise Ratio (PSNR). We use the same video sequence as in Section II.B except that the encoded bit rate is now 512 Kbps. Since there is virtually no congestion from UC Berkeley to RPI (Internet 2), packets are artificially dropped to simulate a realistic congestion effect. The packet loss rate from Utah to CMU is set to 5%, other links are set to 1% [13, 17]. The available bandwidth from Utah to CMU is also upper bounded to 550 Kbps. Under these conditions, the end-to-end scheme designs a FEC based on the 550 Kbps bandwidth and total loss rate 8%. OM-FEC identifies the bottleneck and partitions the overlay into three segments as follows: segment1 from Server to Node 1, segment2 from Node 1 to Node 2, and segment3 from Node2 to the receiver. Two nodes are involved in FEC encoding/decoding. The FEC is deployed within each segment. OM-FEC places FEC at the bottleneck for a bandwidth of 550 Kbps and 5% loss rate. It can recover more packet loss than the end-to-end

scheme and its video quality is much higher than that provided by end-end scheme FEC, as shown in Figure 9. The PSNR gains are on the order of 13 dB.

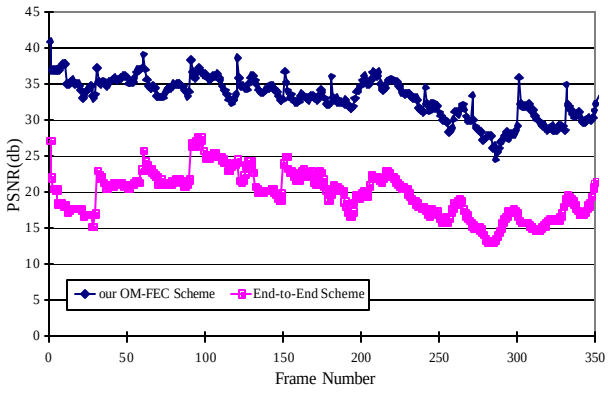


Figure 13: video PSNR of OM-FEC vs. end-to-end FEC (4 hops)

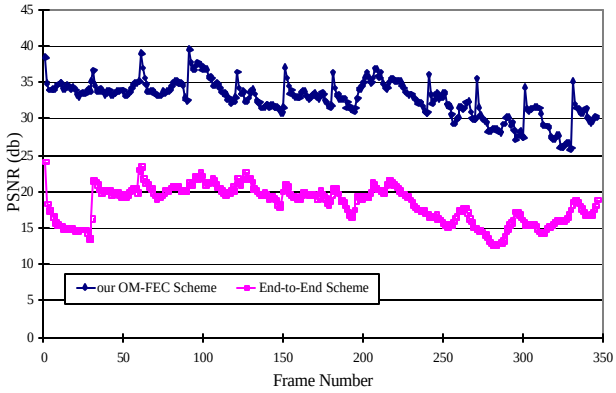


Figure 14: video PSNR of OM-FEC vs. end-to-end FEC (5 hops)

In the second set of experiments we add one overlay node (Node 4: planet1.ecse.rpi.edu) to the path at last hop with 1% loss rate. In this case, for the end-to-end scheme, the FEC is designed based on the bandwidth of 550 Kbps and loss rate 9%. OM-FEC still partitions the overlay path into segments as before, and the FEC at the bottleneck is still designed for the bandwidth of 550 Kbps and 5% loss rate. Still, two nodes are involved in FEC encoding/decoding. The PSNR results are shown in Figure 14, which shows that the advantage of OM-FEC over end-to-end FEC is increased compared to Figure 13. Here, the PSNR gains are on the order of 14 dB. As the number of nodes involved in the transmission increases, OM-FEC performs dramatically better than the end-to-end scheme. For visual comparison, we show a few decoded frames in Figures 15 and 16. These high PSNR improvement figures occurred because we have tried to send at a high video bit rate, relative to what is available on the links. Of course, if we had tried to send at a lower rate, there would be less difference between the various results.

Figure 15: video streaming over 4 hops: OM-FEC (left) vs. end-to-end FEC (right).

Figure 16: video streaming over 5 hops: OM-FEC (left) vs end-to-end FEC (right)

IV. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed an overlay multi-hop FEC (OM-FEC) approach for streaming video over peer-to-peer networks, which automatically adapts its architectural complexity between the extremes of pure end-to-end or pure hop-by-hop operation. The proposed OM-FEC improves the video throughput of the constructed peer-based overlay transmission path by dividing the overlay path into segments based on link characteristics, and applying the appropriate amount of FEC over each segment. We have shown that video streaming using our approach outperforms that of end-to-end FEC without incurring high per-hop complexity. In future work, we will look at incorporating ARQ and multi-path routing, as well as techniques for dealing with node failures [4], in an effort to build up an overall network service abstraction for video streaming and conferencing over peer-to-peer networks.

REFERENCES

[1] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, *Distributing Streaming Media Content Using Cooperative Networking*, Microsoft Technical Report MSR-TR-202-37, April 2002

[2] K. Yeo, B. S. Lee, and M. H. Er, "A Peering Architecture for Ubiquitous IP Multicast Streaming," *ACM SIGOPS Operating Systems Review*, vol. 36, pp 82-95, July 2002.

[3] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture," *Proc. SIGCOMM'01*, San Diego, CA, August 2001.

[4] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," *Proc. ACM SOSP'01*, pp. 131-145, Banff, Canada, October 2001.

[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," *ICSI Technical Report* , no. TR-00-03, March 2000.

[6] http://www.planet-lab.org/

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," *Proc. SIGCOMM'01*, pp 149-160, San Diego, CA, August 2001.

[8] A.C. Begen, Y. Altunbasak, O. Ergun and M.H. Ammar, "Multi-path selection for multiple description video streaming over overlay networks," *Signal Processing: Image Communication*, vol. 20/1, pp. 39-60, Jan. 2005.

[9] S. Gringeri, R. Egorov, K. Shuaib, A. Lewis, and B. Basch, "Robust compression and transmission of MPEG-4 video" ,*ACM MM 2000 Electronic Proceedings*, June 2000.

[10] Phil Karn, *General-purpose Reed-Solomon encoder/decoder in C*, version 4.0, available at http://www.ka9q.net/code/fec/

[11] L. Rizzo, "Effective erasure codes for reliable computer communication protocols, *ACM Computer Communication Review*, vol. 27, pp.24-36, April 1997.

[12] A.Rowstron and P.Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer to peer system," *Proc.IFIP/ACM International Conference on Distributed Systems Platforms* (*Middleware*), Heidelburg, Germany, 2001.

[13] J. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-based error control for internet telephony," *Proceedings of IEEE INFOCOM*, 1999, pages 1453-1460.

[14] K. Stuhlmü ller, N. Färber, M. Link, and B. Girod, "Analysis of video transmission over lossy channels," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 1012-1032, June 2000.

[15] J. Apostolopoulos, W. Tan, S. Wee, G. Wornell, "Modeling path diversity for multiple description video communication," *Proc. IEEE International Conference on Acoustics Speech Signal Processing* (*ICASSP*), 2002

[16] W. Tan and A. Zakhor, "Video Multicast using Layered FEC and Scalable Compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, pp. 373-386, March 2001

[17] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP throughput: A simple model and its empirical validation," *Proc. ACM SIGCOMM*, (Vancouver, CA), pp. 303-314, 1998.