Overlay Multi-hop FEC scheme for Point-to-Point Video Streaming over Peer-to-Peer Networks

Yufeng Shan and Shivkumar Kalyanaraman
Department of Electrical, Computer and Systems Engineering
Rensselaer Polytechnic Institute, 110 8th street, Troy, NY 12180

ABASTACT

Peer-to-peer networks have become extremely popular to provide data-storage (eg: Napster, Kazaa) or distributed computation facilities (eg: SETI@Home). We are interested in *extending the notion* of peer-to-peer networks to provide a variety of *overlay networking services*, i.e. by having peers also provide packet-forwarding capabilities. The performance characteristics (eg: loss, bandwidth, delay) of such a peer-based overlay network is likely to be very different and highly variable compared to the traditional internet end-to-end paths or even traditional managed overlay networks (eg: Akamai) because packets may cross the Internet several times (for each overlay hop!). However, the massive diversity, i.e. multiple peer-based overlay paths (eg: 100s of peer-to-peer overlay paths) harnessed could compensate for the performance variability of any one path. In addition, lightweight support at intermediate nodes can improve the single path performance. In this paper, we focus on the latter problem, i.e. providing lightweight support at *selected* intermediate peer forwarding nodes to achieve *dramatically increased error resilience* on a *single* peer-based path for point-to-point (not multicast!) video-streaming applications. Unlike traditional error correction that relies on end-to-end ARQ or FEC based upon the end-to-end error characteristics of the network path, our proposed scheme is a *flexible* scheme that also considers the error characteristics of each overlay hop. However, our scheme is *not* a heavyweight hop-by-hop error resilience scheme (like X.25); we segment the end-to-end overlay path into maximal sized "*segments*" and provide error resilience between the overlay nodes (i.e. peers or hosts) of those segments. Therefore we call our scheme an "overlay multi-hop FEC" (OM-FEC) scheme. Architecturally, this flexible design lies in between the end-to-end and hop-by-hop paradigms, and we argue that it is well suited to peer-based overlay networks. No support is expected from traditional Internet routers. We evaluate our work in two ways: simulations and real-world implementation on the worldwide Planetlab infrastructure using a real video streaming application. Both these evaluations confirm intuition, i.e., providing lightweight, and flexible multi-hop segment-based FEC can dramatically outperform a naive pure end-to-end strategy, and can be much more efficient than a heavyweight naive hop-by-hop resilience strategy.

Keywords: Video streaming, Peer-to-Peer, Overlay, Forward Error Correction, Multi-hop

1. INTRODUCTION

In this paper we envision a future where end-systems operate their own video streaming services using the assistance from other peers (i.e. end-systems). Why is this interesting? Consider this scenario: a grandchild in the U.S. plans to *spontaneously* stream his/her home video (captured on his camcorder) to his/her grandmother in another country (eg: China or India) as part of a 2-way chat session. All the ingredients necessary for this exciting application are coming into place: camcorders that provide MPEG data, large disk space on computers, broadband connections on both ends, initial IPv6 deployments by ISPs to help peer-to-peer applications. However, there is one hitch! Having a broadband access connection does not assure an end-to-end performance satisfactory for video streaming. End-to-end performance is still tied to the performance characteristics of the single default best-effort Internet path. The grandchild is also an avid user of peer-to-peer systems (eg: Kazaa) and wonders if a magical network of peers can form to help in his/her video streaming objective. In particular, could a large set of peer-based overlay paths complement the default Internet path, *even though* each peer-based overlay path would have low performance and high variability in performance.

We believe that this poses a fundamental design challenge: how to harness low performance and potentially transient or highly variable peer-based forwarding resources (i.e.) peer-based overlay paths to provide an

abstraction of a large end-to-end point-to-point virtual pipe. Fundamental design challenges can be tackled by revisiting fundamental architectural principles (eg: the end-to-end principle). In this paper, we focus on the problem of providing lightweight support at selected intermediate peer forwarding nodes to achieve dramatically increased error resilience on a single peer-based path for point-to-point (not multicast!) video-streaming applications. Unlike traditional error correction that relies on end-to-end ARQ or FEC based upon the end-to-end error characteristics of the network path, our proposed scheme is a flexible scheme that also considers the error characteristics of each peer-based overlay hop. However, our scheme is *not* a heavyweight hop-by-hop error resilience scheme (like X.25); we segment the end-to-end overlay path into maximal sized "segments" and provide error resilience between the overlay nodes (i.e. peers or hosts) of those segments. Therefore we call our scheme an "overlay multi-hop FEC" (OM-FEC) scheme. Architecturally, this flexible design lies in between the end-to-end and hop-by-hop paradigms, and we argue that it is well suited to peer-based overlay networks. No support is expected from traditional Internet routers. In this paper, we do not focus on overlay path construction and routing problems. We focus on a fixed constructed peer-based overlay path and how to efficiently utilize this path. We will henceforth use the term “overlay path” to mean the constructed path over a peer-to-peer network.

High-quality video streaming over the current best-effort Internet is a challenging proposition due to the characteristics of video data such as high bit rate requirement, delay and loss sensitivity. Streaming media distribution has been an intensively studied research topic in the past several years. A large amount of research has been done from all kind of aspects. From the network point of view, companies such as Akamai and Digital Island have deployed Content Delivery Networks (CDNs) by using a network edge-based architecture (edge servers) to achieve load balancing, lower latency and higher throughput. The content is duplicated to the edge servers in order to reduce the round trip time and avoid congestion in the Internet. Simultaneous use of multiple servers [1-2] and multi-paths [3-5] has been proposed in the context of video transmission over Internet. In [1] and [2], the authors propose the use of multiple servers to stream different components of the same content to a single client. Improvement in performance of the video transmission system due to reduction of burst losses is observed in both cases. Video transmission over multiple paths is discussed in [3-5][11]. The authors try to match the characteristic of video data with the path parameters, such as loss rate, delay and capacity, so that the video quality at receiver is maximized. From channel coding perspective, Forward Error Correction (FEC) and Automatic Retransmission reQuest (ARQ) schemes are intensively studied for video transmission. In case of network congestion (loss happens in the video receiver), error recovery scheme and congestion control scheme, such as FEC/ARQ and scheduling, may be deployed to recover the lost packets over the default network path. FEC/ARQ scheme calculates the end-to-end parameters of the default network transmission path and decides what kind of FEC/ARQ scheme should be deployed to combat this network condition.

Most recently, peer-to-peer (P2P) architectures and overlay networks are gaining attention. Padmanabhan et al [6] discuss the problem of distributing streaming media content, both live and on demand, to a large number of receivers in a scalable way. They propose a solution called *CoopNet*, an approach for content distribution that combines aspects of infrastructure-based and peer-to-peer based content distribution, where 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 MDC coded media content. Yeo et al in their multicasting streaming paper [7] propose an application level multicast overlay using peering technology and a lightweight gossip mechanism to monitor prevailing network conditions and to improve the tree robustness. Client can dynamically switch to other parents if they experience a poor QoS. In paper [8], Chu etc explore the possibility of video conferencing applications using an overlay multicast architecture. A redesigned Narada [9] protocol is used in [8] to organize the participating nodes into overlay spanning tree for data delivery. The constructed overlay is optimized according to the measurement of available bandwidth and latency among users, and can be modified by the addition of good links and the dropping of poor links. Their results indicate that End System Multicast can meet the stringent bandwidth and latency demands of conferencing applications in heterogeneous and dynamic Internet environments. The main goal of RON [10] 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 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, the RON will forward the packet by way of other RON nodes.

1.1 Scope and Assumptions

Most of the above papers talk about massive video data distribution or video conferencing using application layer *multicast* based on overlay or peer-to-peer network. When network is congested, the network chooses another better route for packet transmission according to its measurement. In contrast, our objective is to revisit the fundamental problem of efficiently utilize the resources a single overlay path, constructed over peers, i.e. having a number of "hops" between peers. 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 in larger time-scales [9][10]. In this sense, error resilience using FEC is *complementary* (i.e. does not compete) with resilience provided using overlay routing methods.

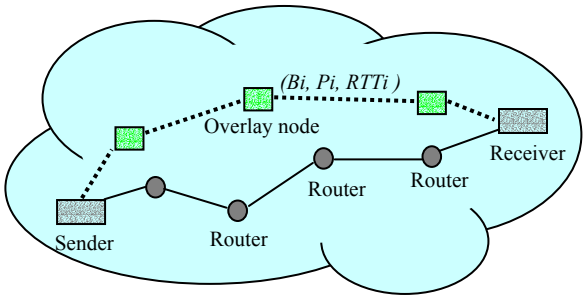

Figure 1: Streaming video using overlay network

We would also like to re-iterate that our Overlay Multi-hop FEC scheme is designed for Point-to-Point video streaming over peer-to-peer networks, while a lot of prior work on video streaming and overlay or peer-to-peer networks assume multicast (i.e. point-to-multipoint) and are subsequently more complex. The main goal of our proposed scheme is to efficiently utilize the characteristics {*bandwidth, delay and loss behavior*} of the constructed peer-based overlay transmission path, in order to maximize the video good-put at time of network congestion. We assume that we can always construct an overlay path by using Peer-to-Peer techniques, such as Chord [18] and Pastry [19] to obtain a set of intermediate forwarding peers (eg: Figure 1). The system consists of a set of participating nodes. The rectangles represent participating overlay nodes, circles denote routers and the dashed lines represent the virtual path between the nodes. The solid line is the default Internet path. The (B_i,P_i,RTT_i) represents the parameters of *i*th *virtual* link as *{bandwidth, loss rata, round trip time}*.

1.2 Motivations

The intermediate nodes of the overlay path receive and forward packets to its neighbor peer on the overlay. Unlike the Internet routers shared by thousands of data flows simultaneously, these overlay nodes may only be used by this application or shared by few other sessions. So the intermediate nodes may undertake some data-plane functions to help improve overall resilience. Forward Error Correction (FEC) is a widely used building block to recover packet loss for time-sensitive applications that cannot afford the delay for ARQ-style re-transmissions. Traditionally, FEC is designed according to the measured end-to-end parameters *{loss rate, bandwidth}* of the transmission path. We argue that doing FEC purely end-to-end in our peer-based overlay context is a dramatically sub-optimal strategy. For example, in Table 1 (a), we list the possible bandwidth and loss rate in a 6-hop overlay path, where {B_i,P_i}is the available bandwidth and the possible packet loss rate of *i*th virtual link, respectively.

Hop	1	2	3	4	5	6
B_i	300K	400K	550K	400K	600K	800K
P_i	0%	1%	10%	1%	0%	2%

(a)

FEC Method	End-to-End	OM-FEC
Good-put	258K	300K
Path loss rate	14%	14%

(b)

Table 1: (a) The example bandwidth and loss rate of an overlay path; (b) Good-put: OM-FEC vs End-to-End

In the above case, the end-to-end based FEC scheme would have to design its FEC overhead based on the end-to-end available bandwidth which is the bottleneck bandwidth *"300Kbps"* and the total end-to-end loss rate which is the sum of the individual loss rates, i.e. *"14%"* in this case. If Reed-Solomon codes are used

as FEC scheme, in order to fully recover the lost packets, 42kbps should be allocated to FEC, the good-put is reduced to 258Kbps. On the other hand, if a *heavyweight* hop-by-hop based FEC scheme is used, the FEC overhead is high only at links with high loss rates (eg: in link 3 which has a bandwidth of 550K and loss rate of 10%). Thus, the end-to-end good-put can be engineered to be 300Kbps in this case. Obviously, the hop-by-hop FEC scheme may induce more per-hop delay and use more computation power of the overlay nodes than necessary. To balance the performance considerations of delay and bandwidth efficiency with architectural complexity considerations (eg: end-to-end vs hop-by-hop), we propose a flexible and adaptive error resilience protocol called Overlay Multi-hop FEC (OM-FEC) for video streaming over peer-to-peer based overlay networks. Our proposed scheme aims to maximize the video good-put over the overlay path and to minimize the overall computation complexity in the intermediate nodes. Specifically, our OM-FEC scheme does not require FEC coding/decoding at each hop. Instead, OM-FEC scheme *optimally partitions* the end-to-end overlay path into sub-paths and performs FEC over these sub-paths, i.e. between the overlay nodes of these sub-paths. For example, in the case discussed in Table (a), OM-FEC partitions the overlay path into two sub-paths. The first sub-path consists only of Hop 1, but no FEC is added on this sub-path (i.e. the good-put is 300 kbps)! The second sub-path has 5 hops, from Hop 2 to Hop 6. Recall that we use the term "hop" to mean an overlay hop between peer nodes, not a router-to-router hop on the underlying Internet, and a segment is a sequence of such overlay hops. To finish our discussion in this toy example, we note that the second path segment has a minimum available bandwidth of 400k, and the same aggregate loss rate 14%, and therefore can accommodate an FEC addition of 42kbps to maintain an overall good-put of 300 kbps (limited by the first segment). Compare this to the 258 kbps achievable with end-to-end FEC. This example suggests that if different segments have substantial variability in available bandwidth and loss rates (as we can expect in peer-based overlay segments), our segment-based FEC scheme would dramatically outperform the naive end-to-end FEC strategy, while being considerably less complex than a heavyweight (overlay-) hop-by-hop FEC strategy.

The rest of our paper is organized as follows. In Section 2, we describe our protocol, rate allocation scheme, and algorithms for our proposed novel FEC strategy. Next, we describe the simulation, real Internet experiments and discuss results in Section 3. Finally, we conclude our work and provide the possible extensions in Section 4.

2. PROTOCOL OVERVIEW

Our proposed building blocks for the protocol include (1) a rate allocation algorithm for allocating different FEC rate for different virtual links (i.e. overlay hops) of an overlay path. (2) an Overlay Multi-hop FEC (OM-FEC) algorithm to determine optimal partitioning for the actual deployment of the FEC coding/decoding on the overlay path. In particular, the OM-FEC algorithm partitions the whole overlay path into sub-paths and adds FEC over these sub-paths. Figure 2 shows a typical peer-based overlay path, where N_i is the ith overlay node, the $\{B_i P_i RTT_i\}$ represents the parameters of *ith* virtual link between N_i and N_{i+1} as {*bandwidth, loss rate, round trip time*}. In this paper, we assume that the uplink and downlink use the same overlay nodes.

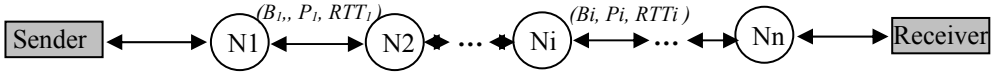

Figure 2: An overlay path for video streaming

The FEC scheme over our protocol operates in two modes as shown at Figure 3: (1) pure end-to-end mode; (2) OM-FEC mode. The default mode is end-to-end mode that is the same as the naive end-to-end FEC strategy, i.e., the end-to-end mode monitors the available end-to-end bandwidth and loss rate, and then decides how much FEC overhead should be added. In end-to-end mode, the overlay nodes receive and forward packets to the destination, without any extra computation functions. In the network experiences congestion such that the end-to-end FEC scheme cannot recover the lost packets, we trigger a transition to the OM-FEC mode. In this mode, the video server sends out an active probe packet every Δt time units to measure the performance characteristics of each hop of the overlay path. Each overlay hop assists in this measurement process and puts its measure in the probe packet and forwards it along. Each overlay node measures the loss rate, and puts a timestamp used to calculate the round trip time the suffix path from that node to the destination (this suffix path RTT computation is completed when the probe packet comes in the

reverse direction). The source can then use the sequence of suffix path RTTs to infer the per-overlay-hop RTTs. This series of per-hop RTT and loss rate estimates is used to infer the TCP-friendly available bandwidth (using the TCP formula) at each hop. Now, with this available bandwidth estimate and loss rate estimate, the optimal FEC strategy for each hop can be calculated. However, this does not imply that the optimal FEC must be implemented at each hop. The OM-FEC algorithm then calculates the optimal path partitioning consistent (or approximately consistent) with the above FEC estimates, so that the overall computation complexity at intermediate hops is minimized without sacrificing the FEC-based resilience gains. Overall, the final deployment of FEC in each time interval Δt would maximize the end-to-end realized good-put and minimize the computation complexity at intermediate peers (or overlay nodes). If the output of the OM-FEC algorithm suggests that the end-to-end strategy is optimal, the system transitions to the End-to-end mode (see figure below). The following sections will outline the details of the rate-allocation and path segmentation strategies in the OM-FEC scheme.

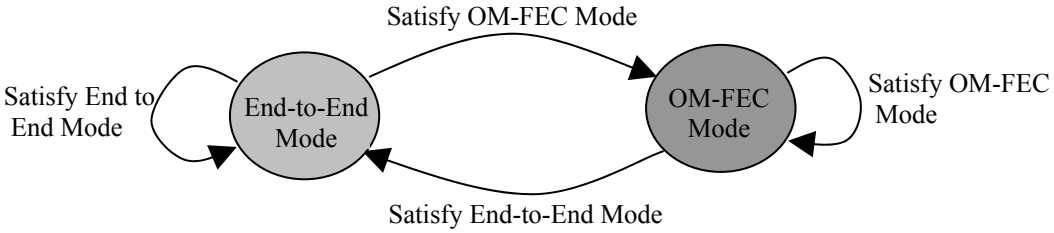

Figure 3: Two working mode transit status of our protocol

2.1 Rate Allocation Strategy in OM-FEC

The available bandwidth of the overlay path is allocated to both video data and FEC parity data. An adaptive end-to-end rate allocation scheme is described in our prior work [20]. In the OM-FEC mode, the problem of allocating optimal bit rate for FEC and video data to each virtual link can be stated as follows:

Find the FEC scheme that minimizes the total distortion: $\min(\sum_{i=0}^{N} D(i,t))$ (1)

$$\text{Subject to } \begin{cases} 0 \le R_{FEC}(i,t) \le R_{req}(i,t) \\ B_{data}(i,t) \le B(i,t) - R_{FEC}(i,t) \end{cases}$$

where N is the total number of virtual links, $R_{FEC}(i,t)$ is the actual FEC bandwidth at *ith* virtual link over an time interval $(t, t+\Delta t)$. $B_{data}(i,t)$ denotes the required video data rate. $B(i,t)$ is the estimated *TCP-friendly bandwidth* of *ith* virtual link. $R_{req}(i,t)$ is the required FEC bandwidth for *i*th virtual link. The required FEC bandwidth is the bandwidth with which the lost packets at receiver can be fully recovered. The available bandwidth of each virtual link is different with different loss rate and round trip time. Different virtual link may have different FEC bandwidth requirements. The goal of the rate allocation scheme is to find the suitable rate allocation $R_{FEC}(i,t)$ for the constructed overlay path to maximize the good-put of the video data in case of network congestion. Since we do not assume any scalable video coding scheme or any special scheduling scheme in this paper, the distortion of video quality is minimized when the amount of video data delivered is maximized. In other words, the problem of minimizing the distortion of video quality can be transformed to a problem of maximizing the video good-put in the overlay path. Our strategy proceeds as follows: for each virtual link, the algorithm assigns a portion of the available bandwidth for original video data, and then the rest is assigned to the FEC bandwidth until either the desired FEC rate is met or the rest of the available bandwidth budget is exhausted. In an extreme case, if the available bandwidth is less or equal to the required video data bandwidth, all the available bandwidth is assigned to the video data.

A key point to be noted is that even though our algorithm is open-loop (i.e. it has no closed loop control like TCP), it is TCP friendly! This is because the available bandwidth B_i of each virtual link is calculated based on a formula that gives an equivalent TCP-friendly rate (eg: see [12]) using the per-virtual-hop *{loss rate, round trip time}* estimates. In particular, we use the following TCP-friendly bandwidth equation:

$$B_i = \frac{S}{T_{rtt-i}\sqrt{\frac{2p_i}{3}} + T_{rto-i}(3\sqrt{\frac{3p_i}{8}})p_i(1+32p_i^{\,2})} \quad (2)$$

where B_i denotes the *estimated* available TCP-friendly bandwidth of the *ith* virtual link that is consistent with the loss rate estimate (p_i) and link round-trip time and timeout estimates (T_{rtt-i} and T_{rto-i}). S is the packet size in bytes. T_{rtt-i} is the estimated round trip time of the *ith* hop in seconds. T_{rto-i} is the TCP timeout of *ith* link, according to the analysis in reference [12], we choose $T_{rto-i} = 4T_{rtt-i}$ in our implementation. p_i is the estimated loss rate of the *ith* link. The end-to-end estimated bandwidth B from source to receiver is limited by the minimal per-hop TCP-friendly available bandwidth of the overlay path, i.e.,

$$B = \min(B_i); i \in \{1,2,.....,N\} \quad (3)$$

The estimated TCP-friendly available bandwidth of each hop and the end-to-end TCP-friendly available bandwidth is the input of rate allocation procedure.

2.2 Overlay-Multi-hop FEC (OM-FEC)

2.2.1 Forward Error Correction

Forward error correction codes are usually used for channel coding to protect the data from channel errors (e.g. losses, bit errors). The basic principle behind the use of FEC codes is that the original source data, along with the additional encoded parity packets, are transmitted by the sender, and the parity packets can be used to recover the lost original source packets at the receiver. A receiver can fully re-construct the original source data once it receives a sufficient number of packets. In this paper, we use systematic Reed-Solomon erasure codes as FEC. The RS(n,k) erasure code take *k* original packets and generate *n-k* packets parity packets. Given the position of the lost packets, the RS decoder can reconstruct packets loss up to *n-k* packets out of *n* packets. Without the position of the lost packets, the decoder can still reconstruct from lost packet up to *(n-k)/2*. Hence, larger ratio of n/k leads to higher level of protection for original data. Given a target loss rate P_{target} and the measured loss rate of P, The RS(n,k) can be determined by the following equation:

$$P_{target} = \sum_{i=n-k+1}^{n} \binom{n}{i} P^i (1-P)^{n-i} \quad (4)$$

If *n* is a given fixed number, *k* can be easily found using equation (4), vice versa, we can find *n* with a fixed *k*. In a video steaming system, *k* can not be chosen randomly, since video data is time sensitive. Bigger *k* means longer delay in the receiver side. *k* is related to the bit rate of the encoded video bit stream, packet size and buffering time in the receiver side. For a video bit stream, if the encoded bit rate is β bps, the packet size is η bytes and the buffer time at receiver side is λ seconds, then $k \le \frac{\lambda\beta}{8\eta}$. According to the results presented at [14], the viewing quality of MPEG-4 encoded video is acceptable at loss rate $1x10^{-5}$, good at loss rate $1x10^{-6}$. In this paper, we choose the residual end-to-end loss rate target $P_{target} \le 10^{-6}$.

Using a systematic code, the encoder picks groups of *k* source data blocks to generate *n - k* parity blocks. Thus, every source data block is used *n - k* times, and we can expect the encoding time to be a linear or approx linear function of *n – k*. In this paper, we use online adaptive FEC for the video streaming system. The data encoding and decoding is online processed. Due to the large amount of calculation, it is necessary to evaluate the performance of the RS encoder/decoder to see if it can work real-time. We test our RS encoder/decoder (based on Phil Karn's FEC code [16], our code implementation is not optimized) in a DELL PC with P4 CPU 2.0GHZ, 256M memory, RedHat 8.2. The test result is shown at Table 2.

N-K	5	10	15	20	25	30	35	40
256B/pkt	1.1ms	1.9ms	2.2ms	3.3ms	3.9ms	4.3ms	4.9ms	5.4ms
512B/pkt	2.0ms	3.7ms	4.3ms	6.5ms	7.8ms	8.6ms	9.8ms	10.8ms
1024B/pkt	4.1ms	7.3ms	8.6ms	13.0ms	15.6ms	17.2ms	19.5ms	21.6ms

Table 2: The encoding time of RS encoder

From Table 2, we observe that we can achieve very high FEC encoding rates even on commodity PCs (which are going to be the peers of a peer-to-peer overlay network). For example, the encoding bit rate of RS(255,245) code can be up to 274Mbps at packet size 1024bytes, and this code can recover the lost packets at random loss rate up to 3.92%. Erasure codes tested at [15] reported similar results as our tests above. The performance differences from [15] can be attributed to the implementation of the erasure CODEC. Since the decoder is much faster than the encoder, we do not list our decoding test results here.

2.2.2 Overlay-Multi-hop FEC (OM-FEC)

Based on the estimated parameters of the constructed overlay path, the server runs OM-FEC algorithm to decide what kind of FEC should be added to protect the video data, and also which overlay node should perform the FEC encoding/decoding and how much FEC should be added at the chosen nodes. The criteria of choosing FEC scheme for the overlay network are (1) Maximize the good-put of the constructed overlay path. (2) Minimize the overlay nodes computation complexity. In order to maximize the good-put of the constructed overlay network, we use the rate allocation algorithm described at Section 2.1. To minimize the computation burden of the overlay nodes, the OM-FEC scheme should use as few nodes as possible for the FEC encoding and decoding. Fewer nodes involved in the FEC encoding and decoding results in smaller jitter and transmission delay at receiver side. Our proposed OM-FEC algorithm is described as following.

(a) The server calculates the end-to-end based FEC (Reed-Solomon code RS(n,k)) based on equation (4). The end-to-end loss rate is approximately estimated as $P \approx \sum_{i=1}^{N} P_i$. Given a fixed *k, n* can be calculated. Since we know the *position* of the lost packets, the error recovery ability is doubled. The total bandwidth B_{total} needed for transmitting both the original data and parity packets is determined as.

$$B_{total} = B_{data} + B_{FEC} = B_{data} + \frac{(n-k)}{k} B_{data} = \frac{n}{k} B_{data} \quad (5)$$

(b) If $B_{total} \le B$, this means the bandwidth needed for original data and FEC is smaller than the available end-to-end bandwidth *B* of the overlay path, then FEC is added end-to-end. No intermediate overlay node is involved into the FEC encoding/decoding. The operating mode is therefore the End-to-End mode.

(c) If $B_{total} > B$, the available end-to-end bandwidth *B* is not large enough for both the original video data and end-to-end FEC overhead. If the current mode is End-to-End mode, then the protocol transitions to the OM-FEC mode. The easiest way to add OM-FEC is to *conceptually* add FEC at each overlay hop. Each node decodes the received packets and encodes them again according to its channel behavior to the next hop. Obviously, this scheme induces delay at every node and the computation burden of each node is large. In order to reduce the computation burden, the OM-FEC scheme tries to partition the overlay path into sub-paths as shown in Figure 4. For example, the OM-FEC algorithm partitions the overlay path into three sub-paths, which are the first J nodes as sun-path1, the next L nodes as sub-path2 and the last M nodes as sub-path3, respectively. FEC scheme is deployed over the 3 sub-paths. Thus, the overall computation burden is reduced compared to the one-by-one node FEC computation. *J,L,M* are the parameters dynamically determined by the OM-FEC algorithm.

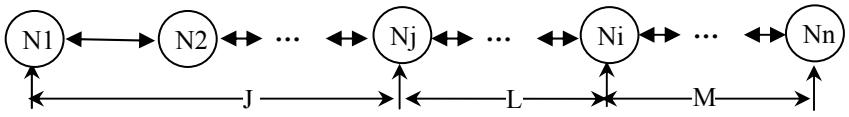

Figure 4 Overlay-Multi-hop FEC grouping

The algorithm of partitioning the overlay path into sub-paths is described as follows.

Start = 0; //begin calculation from the server
For (i = 1; i <= N; i++){
// calculate the FEC bandwidth should be allocated to the path from start node to ith node
Calculate $B_{FEC(start--i)}$*;*
// calculate the FEC bandwidth should be allocated to the path from start node to (i+1)th node
Calculate $B_{FEC(start--(i+1))}$*;*
// find the boundary node to partition the overlay into sub-paths
If (($\min\{B_{start}...B_i\} \geq B_{data} + B_{FEC(start--i)}$*)&&(*$\min\{B_{start}...B_{(i+1)}\} < B_{data} + B_{FEC(start--(i+1))}$*)){*
From start node to ith node is partitioned as one sub-path;
FEC is deployed over this sub-path, the FEC bandwidth allocated to this path is $B_{FEC(start--i)}$*;*
Start = i ; // start from the ith node to partition the rest of the path, the ith node is boundary node
}
}

In the above pseudo code, B_i is the estimated available bandwidth of *ith* virtual link; B_{data} is the required bandwidth for video data. *N* is the total number of links. The server runs the above algorithm to *partition* the overlay path into sub-paths and calculate how much FEC should be deployed on the different sub-paths. The boundary nodes of these sub-paths are the only ones involved in the encoding and decoding computation process. Based on the OM-FEC strategy, the biggest sub-path could include all the nodes over the overlay path that is the same as end-to-end scheme, and the smallest sub-path could be one hop (i.e. hop-by-hop). In other words, this is an automatically adaptive strategy that tunes the architectural complexity between the extremes of naive end-to-end and hop-by-hop operation.

2.3 Loss rate and Round Trip Time Estimation

The loss rate is the ratio of number of lost packets over the total number of packets sent during time interval Δt. The round trip time is computed using the moving average of round trip times. Since the video data packets need to pass multiple overlay nodes, it is difficult to calculate the loss rate and round trip time of each hop using the passive method (using sequence number and timestamp included in the video packet). In this paper, an active probing method is used to calculate the round trip time and loss rate of each virtual link. In order to synchronize overlay parameters calculation and reduce the bandwidth overhead, the proposed protocol uses a small active probing packet to synchronize the estimation procedure. The probe packet is sent from server every time interval Δt, each overlay node processes the probe packet and calculates the loss rate and the round trip time as shown in Figure 5.

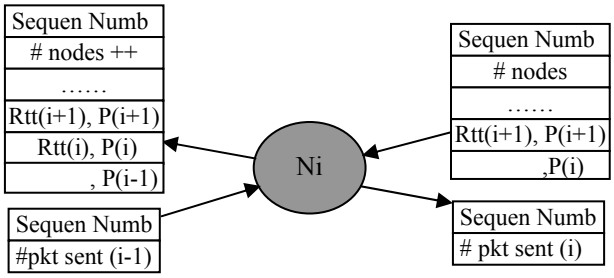

Figure 5: the probe information for each node

In the figure, *Sequen Numb* is the sequence number of the probe packet, *# nodes* denotes the number of node the probe packets passed, *#pkt sent(i)* denotes the number of packets the *ith* node sent. {*Rtt, P*} denote round trip time and loss rate, respectively. The probe packet goes through all nodes along the constructed overlay path. For downlink path parameter estimation, each node 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. The node recodes the time T_{send} for further round trip time calculation while the probe packet is sent. With the received item *"#pkt sent (i-1)"* and the measured received data packets *"#pkt recvd from (i-1)",*

the *ith* node can easily calculate the loss rate of the *(i-1)*th link as $P_{i-1} = \frac{\#\, pkt\_sent(i-1)}{\#\, pkt\_recvd\_from(i-1)}$. Since the N_i node may add FEC to the received packets or drop some packets according to the channel condition, the *"#pkt sent(i)"* may not equal *to "#pkt recvd from (i-1)"*. The probe packet is fed back to the server after it reaches the receiver. The feedback packet collects all information from these overlay nodes while going back to server. As for round trip time estimation, as soon as a probe packet arrives from the *(i+1)th* node, the *ith* node gets the arrive time of this packet T_{arrive}, then calculates the round trip time of *ith* link as follows:

$$RTT_i = T_{arrive} - T_{send} - \sum_{j=i}^{j=n} RTT_j \tag{6}$$

The *ith* node attaches the calculated loss rate of *(i-1)th* link and the round trip time of the *ith* link to the probe packet and then forwards the packet to *(i-1)th* node, until it arrives the server. The server analyzes the information brought back by the probe packet and calculates the available bandwidth for each virtual link.

2.4 Network Model

The issue of modeling packet loss over the Internet is discussed in [13], where it was shown that the sequence of data block success and failure can be approximated by means of a simple two-state Markov chain. In this paper, we assume that each link behaviors in the constructed overlay network can be described using the Gilbert model as shown in Figure 6. There are two states in this model; in this paper, state "1" represents a packet loss, state "0" represents a packet reaching the destination. Let *p* denote the probability of going from state '0" to state "1", let *q* denote the probability of going from state "1" to state "0".

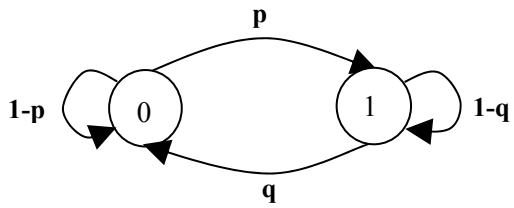

Figure 6 Gilbert model for every link of the overlay

The channel evolution of each link is completely specified by the channel state transition matrix: $M = \begin{bmatrix} 1-p & p \\ q & 1-q \end{bmatrix}$. The steady state probability that a packet error occurs, *P*, is: $P = \frac{p}{p+q}$,The average length of a burst of packet errors is L: $L = 1/q$. Based on this channel model, in the rest of this paper, we simulate the performance of our streaming protocol over peer based overlay network.

3 SIMULATIONS AND EXPERIMENTS

We now demonstrate the effectiveness of our approach by comparing with traditional end-to-end based FEC scheme, both simulations and experiments are performed. Based on our algorithm, we expect that as the number of virtual links increases and the variation of the loss rate becomes larger, our approach would outperform the naive end-to-end FEC scheme over the peer-to-peer network. This is confirmed by our simulations and real-world experiments. Since we use a receiver buffer to absorb the delay and jitter, we do not consider the delay and jitter induced by overlay node encoding and decoding in this paper. In this section, all the curves are the average of at least ten runs of simulations or experiments, i.e. we average out any randomness in the experimentation process.

3.1 Matlab simulations

The simulation configuration is shown at Figure 7. The topology setup includes one sender, one receiver and three intermediate overlay nodes, L1-L4 are four overlay virtual links. The sender sends out video packets through overlay nodes L1->L2->L3->L4 to the receiver, the feedback information is sent back using the same nodes but revise direction.

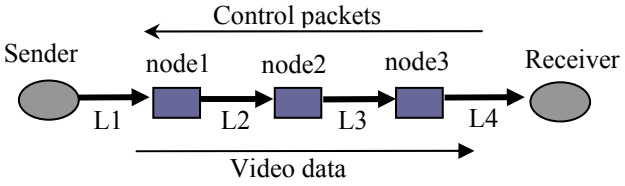

Figure 7: Simulation configuration

The FEC scheme is deployed based on end-to-end scheme or OM-FEC scheme according to the network conditions. We begin our simulation from starting the network in a marginal congestion condition. The simulation parameters are set as Table 3.

Test	Basic Test	Test A	Test B
L1	lossrate = [1% to 2%]	lossrate = [1% to 2%]	lossrate = [2% to 3%]
L2	lossrate = [1% to 4%]	lossrate = [3% to 5%]	lossrate = [3% to 6%]
L3	lossrate = [3% to 5%]	lossrate = [3% to 5%]	lossrate = [3% to 6%]
L4	lossrate = [2% to 4%]	lossrate = [2% to 4%]	lossrate = [3% to 4%]
RS(n,k)	K = 80, N is variable		
Network	condition changes very 5 seconds		
video	Encoded bitrate = 512kbps		

Table 3: Simulation Parameters

The round trip time parameters are chosen based on real Internet observations shown at Table 4. The source host is a computer at test-bed lab, RPI Campus. The approximate round trip time from west coast to east cost of US is about 60-80 ms, so our simulation parameter is set as close as real network, the total round trip time from sender to receiver is set to 70ms. We fix the round trip time of the four links as rtt1 = 10ms; rtt2 = 30ms; rtt3= 10ms; rtt4 = 20ms, respectively to simulate these realistic conditions.

Destination host	Average RTT (ms)	Destination host	Average RTT (ms)
www.berkeley.edu	64	www.gatech.edu	31
www.mit.edu	16	www.rice.edu	47
www.cornell.edu	<10	www.ucla.edu	79

Table 4: the measured average RTT within US from RPI

Given the parameters of the overlay network, such as round trip time and packet loss rate of each overlay virtual link, the bandwidth of each virtual link can be determined by Equation 2. In our simulation, we set the range of packet loss rate of each virtual link, the actual loss rate of each hop is random chosen within its range. The network condition is changed every 5 seconds. At time t = 0s, the sender begins to send out video data to receiver. The gathered network information 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 9, according to the measured loss rate (Figure 8) of each virtual link and round trip time. The sender determines what kind of FEC scheme should be deployed under this network condition. The traditional end-to-end scheme deploys FEC according to the measured round trip time and end-to-end loss rate. Our scheme fine grains the overlay path into sub-paths and deploys the FEC according to the characteristics of the overlay path and network conditions. Our OM-FEC scheme can use bandwidth more efficiently in case of network congestion as shown in Figure 10. In Figure 10, the video good-put is defined as bandwidth occupied by useful video data. The highest video good-put is 512 kbps that is the encoded video bit-rate in this simulation. To test our approach in a heavier congestion condition, we increase the loss rate of several links in the simulation setup Table 3, Test A and Test B. In Figure 11, we can see that our scheme out-performs the end-to-end scheme more at severe congestion.

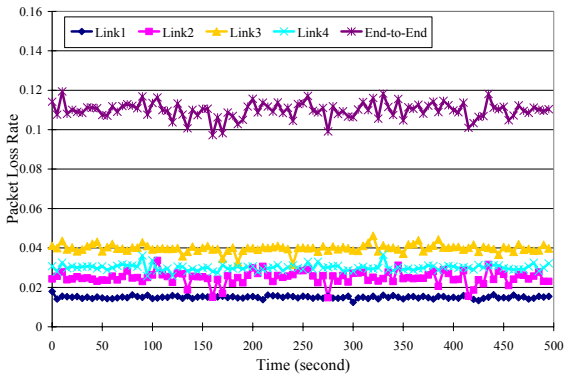

Figure 8: The packet loss rate of the overlay path

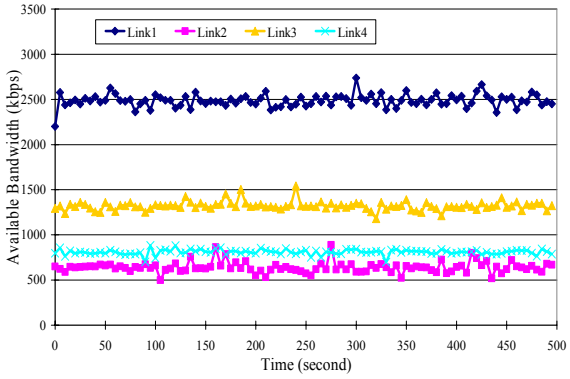

Figure 9: The available bandwidth of each virtual link

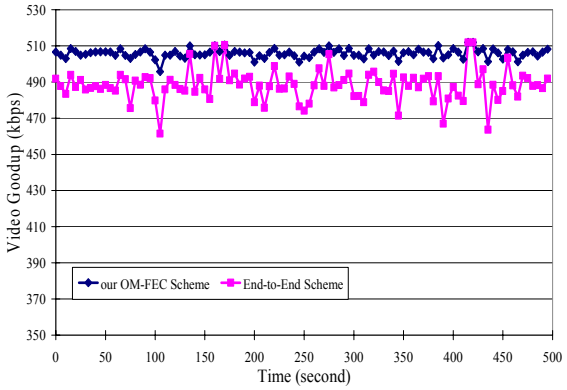

Figure 10: the video good put of our scheme vs end to end scheme

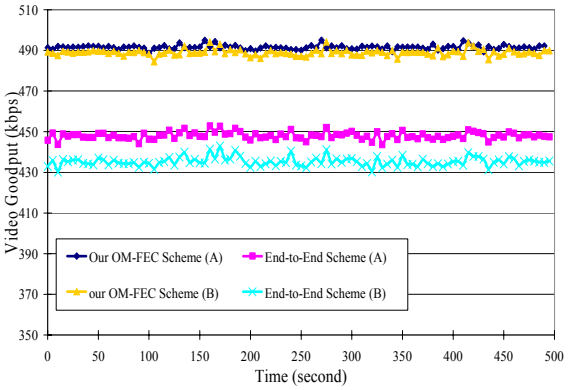

Figure 11: the video good put of our scheme vs end to end scheme in test condition A and B

3.2 Real-world Internet Overlay Experiments

We also implement our protocol over the real Internet (using the worldwide Planet-Lab [17] infrastructure). The implementation includes an overlay agent and the protocol itself. Our overlay agent can run at any Linux Planet-Lab node. The agent forwards video packet to next node until it arrives the destination. Every agent has a small buffer for FEC encoding and decoding according to the decisions from the protocol. The experimental topology is the same as Figure 7 and the Planet-Lab nodes involved are listed in Table 5.

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 5: Nodes involved in our experiments

The video sequence used in the experiments is “foremanQCIF”, 30f/s. The video bit-stream is encoded using H.263+ encoder with error-resilient option at 512kbps, Intra frame refresh at every second. At the receiver, we use a simple error-resilient technique to combat packet losses. Basically, the error-resilience technique replaces the lost group of block (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. Since there is virtually no congestion from UC Berkeley to RPI, packets are artificially dropped to simulate the congestion effect. The packet loss rate from Utah to CMU is set to 5%, other links are set to 1%. The upper bound of available bandwidth from Utah to CMU is also bounded, which is 550kbps. Since the dependency of the encoded video bit-stream is very high, one packet loss in an I or P frame may spread to its following frames. The quality degradation is very high in case of packets loss. In the above condition, the end-to-end scheme designs a FEC based on the 550kbps bandwidth and total loss rate 8%. Our OM-FEC scheme identifies the

bottleneck and patitions the overlay into 3 sub-paths, which are from Server to Node 1, from Node 1 to Node 2 and from Node2 to the receiver. The FEC is deployed within each sub-path. The OM-FEC designs FEC at the bottleneck as 550kbps, 5% loss rate. It can recover more packet loss than the End-to-End scheme, thus, the video quality is much higher as shown at Figure 12.

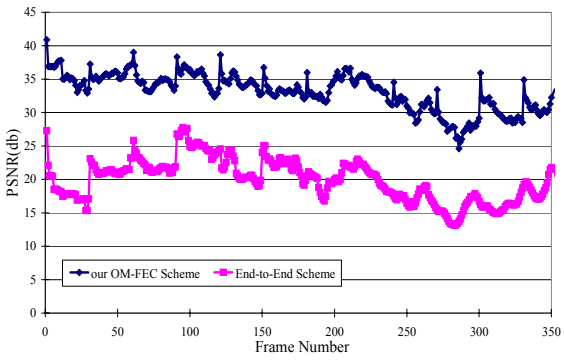

Figure 12: Video PSNR of our OM-FEC scheme vs End-to-End Scheme (4 virtual links)

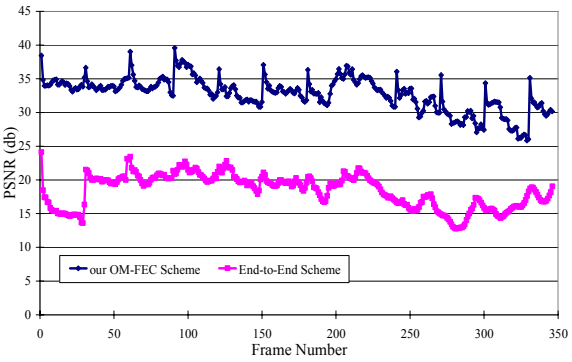

Figure 13: Video PSNR of our OM-FEC scheme vs End-to-End Scheme (5 virtual links)

We add one overlay node (Node4: planet1.ecse.rpi.edu) to the path at last hop with 1% loss rate. The experiment result is shown at Figure 13.In this case, for the End-to-End scheme, the FEC is designed based on 550kbps and loss rate 9%. Our OM-FEC scheme still partitions the overlay path into 3 sub-paths and the FEC at the bottleneck still as 550kbps, 5% loss rate. From Figure 13, more degradation is seen for End-to-End scheme. Our OM-FEC scheme has a slightly drop at around 150th frame. As more nodes involved in the transmission, our OM-FEC scheme performs dramatically better than End-to-End scheme.

4 CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed an error resilience scheme for streaming video over peer-to-peer network, that automatically adapts its architectural complexity between the extremes of pure end-to-end or pure hop-by-hop operation. Our protocol is TCP-friendly protocol, even though it does not use closed-loop control. We propose a rate allocation scheme and an adaptive transmission control scheme in order to achieve higher good-put of the constructed peer-based overlay transmission path. We have shown that video streaming using our approach outperforms the naive end-to-end FEC approach scheme without incurring per-hop complexity like in the hop-by-hop strategy. In our current work, we are incorporating ARQ techniques, buffer management, and multi-path routing to build up an overall network service abstraction for peer-to-peer video streaming and conferencing over peer-to-peer networks. We believe that this application has sufficient characteristics of a potential killer application for the future.

5. REFERENCES

[1] T. Nguyen and A. Zakhor, "Distributed video streaming with Forward Error Correction," *Proceedings of Packet Video Workshop (PV'02)*, April 2002

[2] A. Majumdar, R. Puri, and K. Ramchandran, "Distributed multimedia transmission from multiple servers," *Proceedings of IEEE ICIP'02*, vol. 3, pp. 177-180, September 2002

[3] J. G. Apostolopoulos and S. J. Wee, "Unbalanced multiple description video communication using path diversity," *Proceedings of IEEE ICIP'01*, vol. 1, pp. 966-969, October 2001.

[4] W. Xu and S. S. Hemami, "Efficient partitioning of unequal error protected MPEG video streams for multiple channel transmission," *Proceedings of IEEE ICIP'02*, vol. 2, pp. 721-724, September 2002

[5] J. Apostolopoulos, W. Tan, S. Wee, and G. W. Wornell, "Modelling Path Diversity for Multiple Description Video Communication," *Proceedings of IEEE ICASSP'02,* May 2002.

[6] V.N. Padmanabhan, H.J.Wang, P. A. Chou K. Sripanidkulchai "Distributing Streaming Media Content Using Cooperative Networking" Microsoft technical report MSR-TR-202-37, April 2002

[7] K. Yeo, B.S.Lee and M.H.Er "A Peering Architecture for Ubiquitous IP Multicast Streaming" ACM SIGOPS Operating Systems Review, Volume 36, Issue 3, pp 82-95, July 2002

[8] Yang-hua Chu, Sanjay G. Rao, Srinivasan Seshan and Hui Zhang “Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture” SIGCOMM 01 August 27-31, 2001 San Diego, CA, USA
[9] Y. Chu, S. Rao, and H. Zhang “A Case for End System Multicast” In *Proceedings of ACM Sigmetrics*, June 2000
[10] D.G. Andersen, H Balakrishnan, M.F. Kaashoek, and R. Morris *Resilient overlay networks*. In Proceedings of the Eighteenth ACM Symp on Operating Systems Principles, pages 131--145, Banff, Canada, October 2001
[11] T. Nguyen and A. Zakhor, "Path Diversity with Forward Error Correction (PDF) System for Packet Switched Networks" INFOCOM 2003, April 1-5, San Francisco CA, USA
[12] S. Floyd, M. Handley, J. Padhye, and J. Widmer, “Equation-based congestion control for unicast applications,” *Applications, Technologies, Architectures and Protocols for Computer Communication*, p.43-56, Oct. 2000.
[13] J. Bolot, S. Fosse-Parisis, and D. Towsley, “Adaptive fec-based error control for internet telephony,” in *Proceedings of IEEE INFOCOM*, 1999
[14] 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, http://wood-worm.cs.uml.edu/rprice/ep/gringeri
[15] L. Rizzo, “Effective erasure codes for reliable computer communication protocols”, ACM Computer Communication Review, vol. 27, pp.24-36, Apr. 1997
[16] http://www.ka9q.net/
[17] http://www.planet-lab.org/
[18] Stoica, I., Morris, R., Karger, D., Kaashoek, F.,and Balakrishnan, H. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications”. *Proceedings of the SIGCOMM, pp 149-160.* 2001
[19] A.Rowstron and P.Druschel “Pastry: Scalable, distributed object location and routing for large-scale peer to peer system” *In IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelburg, Germany, 2001
[20] Yufeng Shan and Avideh Zakhor “Cross Layer Techniques for Adaptive Video Streaming Over Wireless Networks" in *International Conference on Multimedia and Expo*, pp. 277 - 280. Lausanne, Switzerland, August 2002