JOINT SENDER/RECEIVER OPTIMIZATION ALGORITHM FOR MULTI-PATH VIDEO STREAMING USING HIGH RATE ERASURE RESILIENT CODE

Changxi Zheng[†] *Guobin Shen*[‡] *Shipeng Li*[‡] *Qianni Deng*[†]

[†] Shanghai Jiaotong University, Shanghai, P.R. China, 200240
[‡] Microsoft Research Asia, Beijing, P.R. China, 100080

ABSTRACT

In this paper we present a joint sender/receiver optimization algorithm and a seamless rate adjustment protocol to reduce the total number of packets over different paths in a streaming framework with a variety of constraints such as target throughput, dynamic packet loss ratio, and available bandwidths. We exploit the high rate erasure resilient code for the ease of packet loss adaption and seamless rate adjustment. The proposed algorithm and adjustment protocol can be applied at an arbitrary scale. Simulation results demonstrate that the overall traffic is significantly reduced with the proposed algorithm and protocol.

1. INTRODUCTION

Streaming with path diversity has been proposed as an effective technology to combat the unpredictability and congestion on the network, and to achieve better bandwidth utilization [1] [2] [3]. In multi-path streaming framework, packets are sent out over multiple delivery paths. Packets from different paths are used either to recover the lost packets or to increase the effectively aggregated throughput (so as to satisfy the bandwidth requirement which would otherwise can not be fulfilled). In addition to path diversity, Forward Error Correction (FEC) techniques, which were originally introduced to reduce the delay due to retransmission for lost packets, was adopted to improve the overall performance. Most frequently used FEC code is the (N, K) Reed-Solomon code where N is the number of generated messages and K is the original messages. Here we use the high rate RS code instead of the traditional one in order to facilitate packets recovery and rate adjustment over different paths.

In order to improve the utilization of available bandwidth resources, the rate allocation among different paths were studied in [3] and [4]. The rate allocation is formulated as an optimization problem where the target is to minimize the irrecoverable loss probability, and solved from the receivers' perspectives with an elegant receiver-driven protocol. Specifically, given N encoded packets, they will determine the optimal packet number, N_i, for the i^{th} path with constraint $\sum N_i = N$. However, in these works, the erasure resilient code is applied on a per-client perspective and the protection ratio is usually determined according to an average packet loss ratio and is fixed throughout the streaming session. That means the protection ratio N/K is fixed. As a result, it is critical to determine a proper protection ratio, which is extremely difficult, if not impossible, due to the network dynamics. Moreover, due to the limited knowledge on receiver side, the optimization only considers the benefits for the client itself, ignoring the affect on other peers. Therefore, when applied to a peer-to-peer streaming framework, this algorithm will inevitably lead to at most a local optimization.

In this paper, we seek to minimize the total number of packets sent over different paths subject to various constraints such as target throughput, dynamic packet loss ratio, and available bandwidths. We formulate the problem as a joint sender/receiver optimization problem and propose a seamless rate adjustment protocol. To achieve easier packet loss adaptation, we exploit high rate erasure resilient code [5] instead of the normal RS code [6]. Because of the difficulty to achieve the global optimization, the proposed protocol tries to achieve optimal performance in a small neighborhood, which can be seen as achieving a tradeoff between global optimization and local optimization.

The rest of paper is organized as follows: high rate erasure resilient code is briefly reviewed in Section 2. In Section 3, we present the joint sender/receiver optimization algorithm and propose a seamless rate adjustment protocol to solve it. Simulation results and discussions are presented in Section 4. Finally, Section 5 concludes the paper.

2. HIGH RATE ERASURE RESILIENT CODE

Mathematically, the erasure resilient code is generated through matrix multiplication on the Galois Field GF(p):

$$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{bmatrix} = \boldsymbol{G} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{K-1} \end{bmatrix} \tag{1}$$

where $\boldsymbol{G}$ is the generator matrix, $\{x_0, x_1, \ldots, x_{K-1}\}$ are the original K packets, and $\{c_0, c_1, \ldots, c_{N-1}\}$ are the N coded packets. Anytime as long as the client receives no less than K packets, the original K packets can be decoded without retransmission.

High rate erasure resilient code is a kind of FEC code with a large coded message space, whose parameters, (N, K), satisfies the property that N is much larger than K ($N \gg K$), and thus the amount of rows in generator matrix $\boldsymbol{G}$ is quite large. In practice, we use the Reed-Solomon Code on Galois Field, $GF(2^{16})$, to generate mutually parity packets on sender side [5]. With the RS code on $GF(2^{16})$, the amount of original packets in a FEC block is K, and that of potential coded packets is 2^{16}=65536, which are like 65536 different *colors*, and each corresponds to a row in $\boldsymbol{G}$. A coded packets is generated according to the coefficients in a row. Note that it is not necessary to generate all of the coded packets using the so large $\boldsymbol{G}$. Instead, the 65536 colors are distributed by the receiver to its candidate senders. Each sender obtains a set of colors that are distinctive with each others. The senders encode and send packets according to the colors assigned. Without loss of generality, each sender is distributed with the same number of colors in this work while there are no such constraints in practice. In our implementation, each sender has 256 colors, and thus a receiver could have at most 65536/256=256 candidate senders. That is enough even for a large-scale P2P network. Since the aggregated coded packets to a receiver through different paths are generated by the same generating matrix, they can be decoded cooperatively yet straightforwardly on the receiver. The packet rates along different paths can be dynamically adjusted.

3. JOINT SENDER/RECEIVER OPTIMIZATION ALGORITHM

3.1. Encoding Packets

The video streaming is sent out by packing the streaming data into packets. If the streaming is packed into M packets before encoding using erasure resilient code. We divide the M packets into blocks , each of which contains K packets. The k packets are encoded according to the colors assigned to the sender. Blocks are numbered from 1 to $\lceil \frac{M}{K} \rceil$, and colors are numbered from 0 to 65535. The receiver knows them for any received packets by detecting the color identification field in the packet headers. Since any K received packets in a block is encoded from K different rows in $\boldsymbol{G}$, the K rows consist of a sub-matrix with a full rank of K. Thus the K original packets could be easily recovered on receivers' side.

3.2. Rate Allocation Algorithm

The joint sender/receiver optimization algorithm is achieved via two optimization algorithms, namely receiver-driven optimization algorithm and the server-driven optimization problem, that operate on the receiver side and the sender side, respectively. All the notations involved are defined in Table 1. In the following derivation, we always use the p_i instead of $p_{loss(i)}$ for simplification.

Notation	Definition
K	The number of original packets in each of FEC block
λ	The number of senders for a receiver
s_i	The λ candidate senders of a receiver ($i = 1 \ldots \lambda$)
B_i	The output capability from s_i to this receiver, represented by number of packets in each of FEC block
$p_{loss(i)}$	The estimated average packets loss rate from s_i
p_i	The average packet success rate from s_i, which is calculated from $(1 - p_{loss(i)})$
T_i	The number of encoded packets send from s_i
r_i^s	The i^{th} downstream receivers of sender s ($i = 1 \ldots \delta$)
p_i^s	The packet success rate from s to its receivers
s_i^*	The *next* sender of r_i^s, from which packets success rate is less than the rate from s
p_i^*	The packet success rate from sender s_i^* to r_i^s

Table 1. Predefined Notations

3.2.1. Receiver-Driven Optimization Algorithm

From the perspective of a receiver r, the basic optimization target is to determine T_i to minimize the total number of sent packets D which is given by

$$D = \sum_{i=1}^{\lambda} T_i$$

subject to

$$\begin{cases} \sum_{i=1}^{\lambda} p_i T_i \geq K \\ T_i \leq B_i \qquad i = 1 \ldots \lambda \end{cases} \tag{2}$$

where p_i is calculated from $p_{loss(i)}$ that could be estimated in many ways, B_i could be obtained using TCP-friendly bandwidth estimation, or negotiated by the sender and the receiver at initial state exchanges. Similar modelling has been proposed in [3], the optimization algorithm is solved via an intuitive and greedy algorithm, which assigns more packets to senders with lower loss rate and fewer packets to higher ones until the constraints(2) is satisfied. In practice, this algorithm runs periodically to adjust the packets rate from the receiver's perspective. Even though their optimization target is different from ours, their solution works well. We adopt their algorithm in this work.

3.2.2. Sender-Driven Optimization Algorithm

For the sender-driven algorithm, we first consider a simple case with three senders and two receivers. Receiver R_1 receives stream from S_1 and S_2, and R_2 receives stream from S_1 and S_3. p_{ij} and T_{ij} represent the estimated packets success rate and number of encoded packets for each block sent

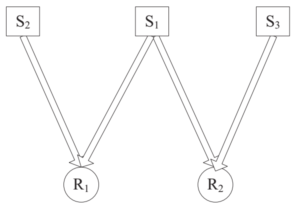

Fig. 1. A simple case for sender-driven algorithm

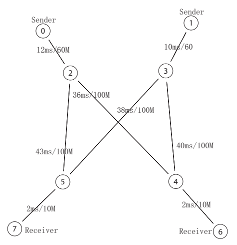

Fig. 2. Topology in scenario 1

from S_i to R_j respectively ($i = 1, 2, 3$ and $j = 1, 2$). Furthermore, we define $\hat{B}_i$ as the currently free output capability (in the unit of packets) of S_i. In order to decode the received packets smoothly, the following constraints have to be met:

$$\begin{cases} T_{12}p_{12} + T_{32}p_{32} \geq K \\ T_{11}p_{11} + T_{21}p_{21} \geq K \end{cases} \tag{3}$$

If S_1 reduces μ packets sent to R_2 for each block, and increases μ packets sent to R_1, the output rate does not increase for itself. But in order to keep smooth decoding on all receivers, the rate from other two senders should be changed accordingly and meet:

$$\begin{cases} (T_{12} - \mu)p_{12} + (T_{32} + \frac{p_{12}}{p_{32}}\mu)p_{32} \geq K \\ (T_{11} + \mu)p_{11} + (T_{21} - \frac{p_{11}}{p_{21}}\mu)p_{21} \geq K \end{cases} \tag{4}$$

After adjustment, the total number of sent packets is

$$C = T_{12} + T_{32} + T_{11} + T_{21} - (\frac{p_{11}}{p_{21}} - \frac{p_{12}}{p_{32}})\mu \tag{5}$$

Hence the totally sent packets could be decreased with the precondition of smooth decoding, as long as

$$\frac{p_{11}}{p_{21}} > \frac{p_{12}}{p_{32}} \tag{6}$$

and

$$\mu \leq \min\{T_{21}\frac{p_{21}}{p_{11}}, \hat{B}_3\frac{p_{32}}{p_{12}}, T_{12}\} \tag{7}$$

For the general case, different senders have different number of downstream receivers. Since the sender-driven optimization algorithm will be applied to each sender independently, we focus on the formulation for a single sender with δ downstream receivers. Note that p_i^* could be obtained via periodically state exchange between senders and receivers.

The sender calculates $\Gamma_i = \frac{p_i^s}{p_i^*}$ for each of its receiver, and ranks them in descending order. Without loss of generality, we assume $\Gamma_1 \geq \Gamma_2 \geq \ldots \geq \Gamma_\delta$. According to the formula (5), the sender has the *benefit matrix*:

$$\boldsymbol{E} = \begin{pmatrix} 0 & e_{12} & e_{13} & \cdots & e_{1\delta} \\ 0 & 0 & e_{23} & \cdots & e_{2\delta} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & 0 & e_{(\delta-1)\delta} \\ 0 & \ldots & \ldots & \ldots & 0 \end{pmatrix}$$

where $E_{ij} = \Gamma_i - \Gamma_j \quad (i < j)$.

So, the target of server-driven optimization algorithm is to find an *adjustment matrix*:

$$\boldsymbol{\Phi} = \begin{pmatrix} 0 & \mu_{12} & \mu_{13} & \cdots & \mu_{1\delta} \\ 0 & 0 & \mu_{23} & \cdots & \mu_{2\delta} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & 0 & \mu_{(\delta-1)\delta} \\ 0 & \ldots & \ldots & \ldots & 0 \end{pmatrix}$$

such that the total number of reduced packets, Q, is maximized where

$$Q = \sum_{1 \leq i,j \leq \delta} e_{ij}\mu_{ij} \tag{8}$$

and consistent with constraint (7), each μ_{ij} subjects to the constraints:

$$\mu_{ij} \leq \min\{\frac{T_i^*}{\Gamma_i}, \frac{B_j^*}{\Gamma_j}, T_j\} \tag{9}$$

Note that where T_i^* is the amount of packets sent from the *next* sender of the receiver r_i^s, B_j^* is the current *free* output capability of the *next* sender of r_j^s, and T_j is the amount of packets sent from the current sender s. This constraint is a generalization of Eqn. (7).

We also use an intuitive and greedy algorithm to determine each μ_{ij} in matrix $\boldsymbol{\Phi}$. The server ranks all e_{ij} in an descending order such that its receivers related to former e_{ij} are granted higher adjustment priority than the ones related to latter e_{ij}. The sender then communicates with the *next* sender of the receivers to determine how to adjust the packets rate. Note that the second term in Eqn. (9) is very important since it prevent the adjustment from a chain reaction to all the servers. In other words, only the servers that have free output capacity will react to the adjustment. Moreover, a server may selectively optimize only for several most significant e_{ij}. As a result, the proposed algorithm can be applied on an arbitrary scale.

3.3. Seamless Rate Adjustment Protocol

Sender s who has calculated the benefit matrix runs the adjustment protocol for a e_{ij} as follows.

1) Send control packets to s_i^* and s_j^* to ask how many packets in each block could be reduced and increased respectively. After receiving the control packets, s_i^* and s_j^*

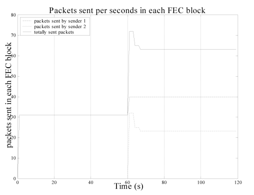

Fig. 3. Sent Packets in Scenario 1

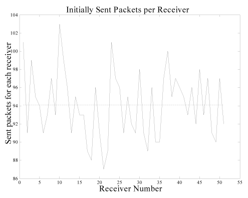

Fig. 4. Total packets for each FEC block

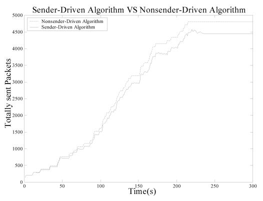

Fig. 5. Packets Reduction

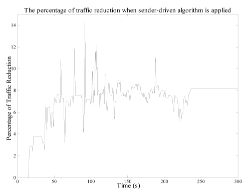

Fig. 6. Percentage of Traffic Reduction

response the capability that it could reduce and increase respectively, which satisfies the constraints in Eqn. (9).

2) After receiving the response from s_i^* and s_j^*, sender s determine the value of μ_{ij} according to the Eqn. (4) and constraints (9). It then sends the request to s_j^* to ask for increasing the packets sent to r_j^s.

3) After receiving the successfully increased response from s_j^*, the sender makes a self-adjustment. It reduces the packets sent to r_j^s, and increases the packets to r_i^s.

4) Finally, sender s sends requests to s_i^* to ask for decreasing packets to r_i^s.

From the above protocol, senders always decrease the sent packets after receiving the successfully increased response from another sender. This mechanism ensures that the receiver can always receive enough packets for decoding. On the other hand, the receiver doesn't care how many packets are sent out by a special sender, it only cares about the total number of received packets. Clearly, this packets rate adjustment is transparent for receivers.

4. SIMULATION AND RESULTS

In this section, we present the simulation results for the proposed algorithm using NS2 network simulator[7]. In the simulation context, we use a video with an actual bit rate of 960kbps and a packet size of 600 bytes.

In the first scenario, we use a simple topology as shown in Figure 2 to observe the rate adjustment among senders. Here each FEC block contains 30 packets. The first receiver joins into the session at time 0s, and the second one joins at time 60s. As shown in Figure 3, the total number of sent packets is more than 70 after the second joining. Then as senders run the adjustment protocol, the packets sent by sender 1 increases, but sender 2 reduces, so that the total number of sent packets reduces. Furthermore, Sender 2 reduces packets always after sender 1 increased the packets, which ensures the video quality on receivers's side.

At last, we demonstrate the effects on many receivers case. We use Brite to generate a *Flat Albert Barabasi* topology, and assign 30 senders and 50 receivers. Each receivers has from 2 to 6 candidate senders as an Uniform Distribution. All receivers come as a Poisson Distribution with the rate of 5 seconds for a joining. Each FEC block contains 80 original packets. The average loss rates follow a uniform distribution from %1 to %30. Figure 4 shows the required packets for each receiver. We also plot the overall sent packets for each FEC block on the network in Figure 5. It shows that as more and more receivers joining into the session, the optimization with sender-driven algorithm makes a more remarkable effect for saving packets. This demonstrates our approach could reduce the total network burden especially in large-scale network.

5. SUMMARY AND CONCLUSIONS

In this paper, we proposed an joint sender/receiver optimization algorithm and a seamless rate adjustment protocol to minimize the overall traffic in a streaming framework with path diversity. The proposed algorithm and adjustment protocol can be applied at any network scale. As a result, a good tradeoff between local optimum and global optimum can be achieved. We adopted high rate erasure resilient code for easy packet loss ratio adaptation. Several simulations were carried out and the results demonstrate the effectiveness of the proposed algorithm and the adjustment protocol.

6. REFERENCES

[1] J. Kim, R. Mersereau, and Y. Altunbasak, "Network-adaptive video streaming using multiple description coding and path diversity," in *IEEE Proceedings of the ICME*, 2003.

[2] T. Nguyen and A. Zakhor, "Path diversity with forward error correction (pdf) system for packet switched networks," in *IEEE Proceedings of the INFOCOM*, 2003.

[3] T. Nguyen and A. Zakhor, "Distributed video streaming over internet," in *Multimedia Computing and Networking (MMCN), San Jose, CA*, Jan. 2002.

[4] T. Nguyen and A. Zakhor, "Distributed video streaming with forward error correction," in *Proc. Packet Video Workshop, Pittsburgh, USA*, Apr. 2002.

[5] J. Li, "Peerstreaming: A practical receiver-driven peer-to-peer media streaming system," Tech. Rep. MSR-TR-2004-101, Sept. 2004.

[6] N. Alon and M. Luby, "A linear time erasure-resilient code with nearly optimal recovery," *IEEE Transactions on Information Theory*, vol. 42, pp. 1732–1736, 1996.

[7] K. Fall, "Ns notes and documentation: The vint project," 2000.