STREAMING OF SCALABLE VIDEO FROM MULTIPLE SERVERS USING RATELESS CODES

Jean-Paul Wagner, Jacob Chakareski and Pascal Frossard

Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Institute - LTS4
CH-1015, Lausanne

ABSTRACT

This paper presents a framework for efficiently streaming scalable video from multiple servers over heterogeneous network paths. We propose to use rateless codes, or *Fountain codes*, such that each server acts as an independent source, without the need to coordinate its sending strategy with other servers. In this case, the problem of maximizing the received video quality and minimizing the bandwidth usage, is simply reduced to a rate allocation problem. We provide an optimal solution for an ideal scenario where the loss probability on each server-client path is exactly known. We then present a heuristic-based algorithm, which implements an unequal error protection scheme for the more realistic case of imperfect knowledge of the loss probabilities. Simulation results finally demonstrate the efficiency of the proposed algorithm, in distributed streaming scenarios over lossy channels.

1. INTRODUCTION

Media streaming applications over the Internet often have to respect relatively tight effective bandwidth and delay constraints, and yet to achieve acceptable visual quality at the receiver. Server or path diversity help in achieving higher overall throughput to the client. For example, it has been shown in [1] that usage of multiple streaming servers provides better robustness in case one of the channels becomes congested. As the data packets most likely take different paths from their respective source to the client, the overall network load can be balanced, and the most reliable paths can be exploited more efficiently. However, one inherent problem of using multiple sources to send the same stream to a client is the coordination between servers. In order not to waste resources with redundant data packets, servers have to carefully coordinate their packet scheduling strategies [2]. It tends to render such a distributed streaming system overly complex and cumbersome, especially if conditions change on one of the source-client paths.

In this paper, we use rateless codes, or *Fountain codes*, in order to remedy to this coordination problem. We show that using rateless codes, it is feasible to efficiently stream scalable media from multiple sources to a client with no need of coordination among the sending servers. At the same time we make sure that each packet that is sent by any of the servers is not redundant for the client that receives it. This is in spirit similar to [3]. However we propose optimal sending schemes for a set of servers delivering a scalable media stream, and devise a heuristic-based algorithm that can provide close to optimum performance in realistic streaming scenarios. The proposed framework is generic and provides a low complexity distributed streaming solution. Building on the universal channel code properties of rateless codes, the system is able to adapt to any kind of channel loss, without adaptively transcoding the data at each sender, contrarily to [4].

This work has been partly supported by the Swiss National Science Foundation, under grant PP002-68737 and the Swiss Innovation Promotion Agency (CTI), under grant 7388.2 ESPP-ES.

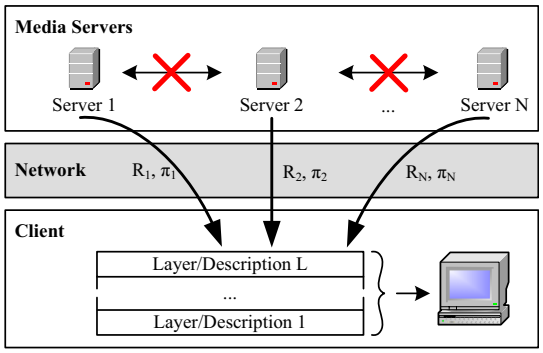

Fig. 1. Distributed streaming of scalable data streams.

The system under consideration is presented in Figure (1). N servers, which do not communicate among themselves, stream L layers of a media stream to a streaming client. The loss probabilities π_n and sending rates R_n are possibly different for each server-client path through the network. Even if our example scenario considers a layered encoded stream, it can be noted that the proposed distributed streaming framework applies to any scalable media encoding, and even to Multiple Description Coding schemes. The rest paper is organized as follows. In Section 2 we briefly introduce rateless codes by the example of Raptor codes, and show how they can be used to encode a scalable media bitstream. In Section 3 we solve the optimal rate allocation problem of an ideal scenario with static channel characteristics. In Section 4, we provide a distributed heuristic-based algorithm that performs efficiently in practical settings, and simulation results are given in Section 5. Finally we conclude with Section 6.

2. RATELESS CODES

2.1. Raptor Codes

With rateless codes, such as LT [5] and Raptor [6] codes, one can generate a potentially unlimited number of symbols from k original symbols. Ideal Raptor codes have the property of generating unique symbols with high probability, such that any $(k + \epsilon)$ packets can be used to decode the original k symbols. The notion of *Fountain code* comes from the analogy of a rateless code with a water Fountain (the unlimited number of symbols) from which any glass of volume $(k + \epsilon)$ satisfies the client needs, no matter which drops (symbols)

of water it has obtained. It does not even matter if the received symbols come from the same source, as long as different sources have encoded the same input symbols. This property is key in order to use multiple sources to provide the same stream to a client without any coordination among the sources. As long as the set of symbols they serve has been generated from the same input symbols, the encoded symbols will be different from one source to another with high probability, and so they can contribute in the same way to the client's decoding.

In practice, the number of symbols that can be generated from a set of source symbols is limited to the number of available *Encoding Symbol IDs*, or ESIs, which are coded with 2 bytes, thus providing a maximum of 2^{16} distinct encoded symbols. The symbol size T can range from 1 bit to several hundred bytes. If a *block* of K symbols of size T is encoded into a large number of encoded symbols of size T and if $1000 \leq K \leq 8192$, then the decoding overhead ϵ is typically of about 2 symbols. It is worth noting that Raptor codes induce linear complexity for both encoding and decoding, and therefore also allow for on-the-fly encoding if needed. For further details on Raptor codes and their implementation, we refer the interested readers to [6, 7, 8].

2.2. Coding Scheme for Layered Media

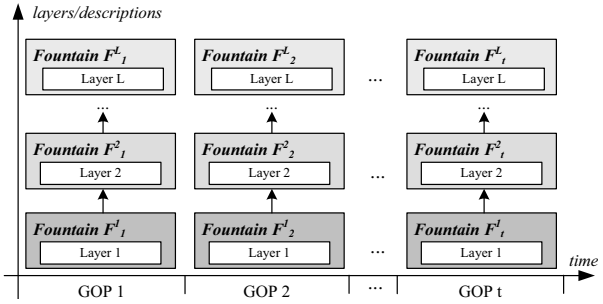

Fig. 2. Coding Scheme: a Fountain is created per GOP and per layer. The vertical arrows show the hierarchical dependencies in the bitstream.

A rateless code, applied blindly on a media bitstream, would mix the time-dependencies and the intra-layer dependencies that are present in the original scalable media stream. Therefore, we propose to create one Fountain per layer and per GOP of the original bitstream, as depicted in Figure (2). Such a Fountain is denoted $\mathcal{F}_t^l$, where l stands for the layer, $1 \leq l \leq L$ and t is the timestamp associated with the corresponding GOP. It encodes a set of K_t^l source symbols, which depends on the encoding rate of the layer l. Such a coding scheme allows to keep the hierarchical and temporal dependencies present in the original bitstream, which are essential for the scalable delivery of the stream. Then, each server sends different packets from the same Fountain $\mathcal{F}_t^l$, such that the client does not receive any duplicate packets. Even in the case of practical Raptor codes, there are several ways to guarantee that each server sends different symbols from the same Fountain. For example, the requesting client can provide a different random seed to each of the sources, determining a subset of ESIs (and thus encoded symbols), which the server has to transmit. Another option could be to centrally encode a large number of symbols for each Fountain, and to put disjoint subsets on different servers of a CDN.

3. OPTIMAL RATE ALLOCATION

In the generic case of non-systematic coding, the decoder needs to receive at least K_t^l symbols to be able to decode the layer l of the GOP t. The efficiency of the distributed streaming system is therefore clearly driven by an efficient rate allocation problem. The streaming rate distribution strategy has to maximize the probability that the client receives enough different packets for each layer, in order to optimize the quality after decoding. We now compute that probability, where we drop the GOP index t for the sake of clarity.

We consider a distributed streaming system with N servers. Each server S_n is connected to the client through a path n with a maximum rate of R_n symbols/GOP, and a loss probability π_n ($1 \leq n \leq N$). In the generic case, each path may have a different cost γ_n, inferred by transmitting a packet over path n. Let k_n^l denote the number of coded symbols of layer l, which are correctly received from server S_n. Further, let r_n^l denote the streaming rate for layer l on path n. If we assume an independent loss process, the probability density function (pdf) of k_n^l can be expressed as:

$$p_n^l(i) = Prob(k_n^l = i) = \binom{r_n^l}{i}(1-\pi_n)^i \pi_n^{(r_n^l - i)}, \qquad (1)$$

Let k^l represent the number of symbols from layer l, received from all the sources S_n together. The pdf of k^l is the convolution of N binomial density functions, and can be written as:

$$p^l(i) = Prob(k^l = i) = Prob\left(\sum_{n=1}^{N} k_n^l = i\right) = \bigotimes_{n=1}^{N} p_n^l. \qquad (2)$$

The corresponding cumulative density function $P^l(i)$ represents the probability that at least i symbols from layer l are correctly received. It is simply given by:

$$P^l(i) = 1 - \sum_{j=0}^{i-1} p^l(j). \qquad (3)$$

We are now able to derive conditions for optimal rate allocation among servers, and between layers. The set of *optimal* rates $r_n^l, 1 \leq n \leq N, 1 \leq l \leq L$ is such that it maximizes the probability $P^l(K^l)$ of receiving at least K^l symbols for each layer l. Interestingly, these probabilities only depend on the allocated rates per layer and per link, r_n^l, through equations (1) and (2). In the same time, the overall rate usage has to be minimized, while the individual path rate constraints have to be satisfied. Note that the difference between K^l and $\sum_n r_n^l$ corresponds to the error protection overhead, which is optimally distributed between servers when the rate allocation is optimal.

Optimization Problem 1 *Given the numbers of source symbols per layer $\{K^l\}$, the set of available rates $\{R_n\}$ and loss probabilities $\{\pi_n\}$, the optimal rate allocation $\vec{r^*}$ is given by:*

$$\vec{r^*} = \underset{\{r_n^l\}_{1\leq l\leq L, 1\leq n\leq N}}{arg\ max} \prod_{l=1}^{L} P^l(K^l) - \sum_{n=1}^{N} \gamma_n \sum_{l=1}^{L} r_n^l, \qquad (4)$$

under the constraints that $\sum_{l=1}^{L} r_n^l \leq R_n, \forall n.$

The solution of the Optimization Problem 1 is a priori combinatorial in the general case. We however outline a series of heuristics, which can be used to design a low-complexity rate allocation algorithm, which performs close to optimal. An example optimal rate allocation is illustrated in Figure (3), where $\vec{r^*}$ is computed by exhaustively searching for the optimum rate for each layer and each path. In this particular run, 3 servers and 2 layers are considered. The 3 links can

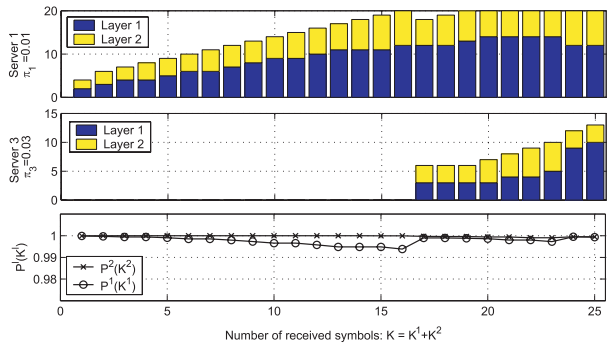

Fig. 3. An optimal rate allocation for 2 layers over 3 links. Link 2 is never used and therefore not shown. Results are shown for $K = K^1 + K^2$ between 1 and 25, with $K^1 \approx 2K^2$. Bottom: Probabilities $P^1(K^1)$ and $P^2(K^2)$ as computed by the optimal rate allocation. Each link has unit price.

support rates of maximum 20, 20 and 15 symbols/GOP, respectively, and the loss probabilities are $\pi_1 = 0.01$, $\pi_2 = 0.2$ and $\pi_3 = 0.03$. The figure shows the optimal allocation for $K = \sum_{l=1}^{2} K^l$ ranging from 1 to 25, and each time K^1 is approximately twice as large as K^2. Several observations can be noted on the solutions obtained for the Optimization Problem 1:

- As long as the channel with the lowest error probability can carry all the symbols that are needed, all the symbols are allocated to that channel.
- The channel with the second lowest error probability is used only when the one with the lowest error probability has been exhausted. The exceptions at $K = 17, 18$ are due to the fact that we use discrete pdfs, and only allow for integer symbols to be allocated.
- The optimal rate distribution allocates the optimal amount of error protection to all layers, as it takes into account the exact error probabilities on all the links.

The solution to Optimization Problem 1, $\vec{r^*}$, is an array of n lines and l columns, where each component $\vec{r^*}(n)(l)$ denotes the optimal number of symbols from layer l to send on link n. Let r_n^* denote the optimal total rate allocation for link n: it is the sum of the l components of the n^{th} line of $\vec{r^*}$. In general, $\vec{r^*}$ is not a unique solution, as for a given set of optimal per-link allocations r_n^*, there can be various per-layer allocations r_n^l among the links that are equivalent. In the following, we show that an allocation that allocates shares of each link bandwidth, proportionally to the respective layer sizes, is always among the solutions of Optimization Problem 1.

Any optimal allocation $\vec{r^*}$ makes sure that, on average, the total number of received symbols will be $\sum_{n=1}^{N} r_n^* \cdot \pi_n$. Out of this total number of received symbols, the optimal allocation also guarantees that, on average, the share of symbols received for each layer is proportional to the number of source symbols, K^l, as reflected by the weights κ^l given as:

$$\kappa^l = \frac{K^l}{K^{min}}, \tag{5}$$

where K^{min} is the smallest K^l. So, on average, $\bar{K}^l$ symbols are received for layer l:

$$\bar{K}^l = \frac{\kappa^l}{\sum_{l=1}^{L} \kappa^l} \left(\sum_{n=1}^{N} r_n^* \pi_n \right). \tag{6}$$

Consider now an allocation scheme that uses the same optimal per link rate allocations r_n^*. However, it proportionally allocates shares of each layer, as determined by the κ^l, to each link:

$$r_n^l = \frac{\kappa^l}{\sum_{l=1}^{L} \kappa^l} r_n^*. \tag{7}$$

Clearly, this allocation makes sure that the number of received symbols per layer is, on average, given by $\sum_{n=1}^{N} \left(\frac{\kappa^l}{\sum_{l=1}^{L} \kappa^l} r_n^* \right) \pi_n$. This is equivalent to Equation (6), which proves that the allocation scheme which affects shares that are proportional to the layer sizes to each used link, is always among the solutions to Optimization Problem 1. This property allows us to formulate a distributed algorithm in the next Section, as every server splits its sending rate in the same way among the streamed layers.

4. HEURISTIC-BASED ALGORITHM

4.1. Client side

We now propose a heuristic-based scheme, which offers a close to optimal rate allocation, with low complexity. We assume that a client has an approximation of both the available rates $\bar{R}_n$ and the error probabilities $\bar{\pi}_n$ for the N server-client paths. Based on these rates and loss probabilities, which are at best estimates of the actual path characteristics, it computes an overall rate allocation r_n per server (without considering the layers), by greedily attributing streaming rates to the path with the lowest error probability first. The allocation is finished once the sum of the effective rates that have been allocated satisfies the average rate of the video stream to be delivered, R^*:

$$R^* = \sum_{n=1}^{N} (1 - \bar{\pi}_n) r_n, \text{ with } r_n \leq \bar{R}_n, 1 \leq n \leq N. \tag{8}$$

4.2. Server side

Each server fills the rate r_n, as requested by the client, with shares for each layer that are reflected by the weights κ^l, as given in Equation (5). Each server then has the possibility of sending $u_n = (R_n - r_n)$ additional bits to the client. These bits may be used to perform Unequal Error Protection (UEP) on the delivered stream, which becomes beneficial when the client error probability estimates are not exact, as it is most likely the case in practice. The use of Raptor codes makes the implementation of UEP quite trivial: in order to better protect layer l, we just need to send more packets from fountain F^l. That is why we choose to use the available bitrate u_n in a way that reflects the hierarchical dependencies between the layers, which are present in the uncoded bitstream. To express these dependencies, we use a weight α^l for each layer, which reflects the number of layers that depend on layer l in order to be decoded correctly. For example, in the case of Figure (2), $\alpha^1 = L$, $\alpha^2 = L - 1$ and $\alpha^L = 1$. Note that in a balanced multiple description coding scenario, all weights are equal.

To summarize, each server for which $r_n > 0$ sends at rate r_n^{tot}, which can be written as:

$$r_n^{tot} = r_n + \mu u_n, \tag{9}$$

where the coefficient μ allows to control the rate that is used for unequal error protection against channel estimation mismatches. The rates per layer that sum up to the first term are expressed as $r_n^l = \frac{\kappa^l}{\sum_{l=1}^{L} \kappa^l} r_n$, similarly to Equation (7). Finally, the rates per layer that sum up to the second term are:

$$u_n^l = \frac{\alpha^l}{\sum_{l=1}^{L} \alpha^l} (R_n - r_n). \tag{10}$$

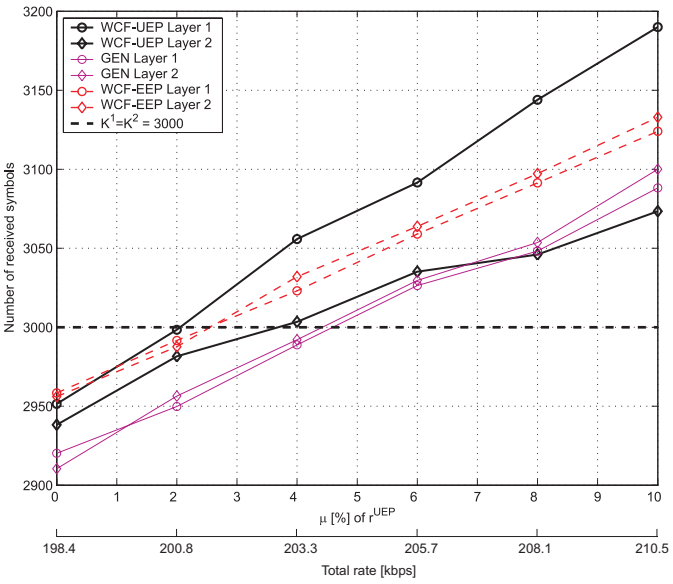

Fig. 4. Performance of distributed streaming algorithms, in number of received symbols versus overall streaming rate.

5. SIMULATION RESULTS

We consider the delivery of 2 layers of video, a base layer, and an enhancement layer, which depends on the base layer. Both layers have an equal average rate of 64kbps and a frame rate of 30Hz. The GOP size is 60 frames. One GOP is Raptor-encoded with symbol size $T = 8$ bytes. Thus the client needs to receive $K^1 = K^2 = 3000$ Raptor encoded symbols, in order to be able to decode one layer with high probability. The choice of parameters complies with the constraints imposed by practical Raptor codes, as mentioned in Section 2. We suppose that the servers can always provide enough aggregate rate to deliver the requested number of layers to the client. If this is not the case, each server can compute the number of layers to deliver based on the total allocated rate, which can be communicated by the client. This is however beyond the scope of this paper. Three

	Link 1	Link 2	Link 3
$R(kbps)$	128	128	96
π	0.04	0.06	0.08
$\bar{\pi}$	0	0.08	0.04
$r(kbps)$	128	70.4	0

Table 1. Top: link parameters. Bottom: loss probabilities as known by the client, and greedy rate allocation.

servers are used, and each server-client path is characterized by a maximum transmission rate of R_n, and loss probability π_n, as given in Table (1). The table also shows the estimated loss probabilities $\bar{\pi}_n$, as available at the client, and the rates r_n, which are allocated to each server, as given by Equation (8).

Simulations compare the performance of the proposed algorithm (WCF-UEP) with two other simple rate allocation schemes:

- (WCF-EEP) uses the same greedy rate allocation by using the reliable channels first, but implements equal error protection. i.e., $\alpha^1 = \alpha^2$. Note that this provides a solution close to the optimal algorithm proposed in Section 3.
- (GEN) is a generic scheme, allocating the overall rate in the network to all the links, by using a water-pouring strategy: all links get equal shares of rate, regardless of their loss probability. Once the lowest capacity link is full, the the procedure is continued on the remaining links. The allocated rate per link is then shared among all layers, in proportions reflected by Equation (5).

Figure (4) shows the number of received symbols for the various strategies versus the overall streaming rate, starting at r_{tot}, the overall rate allocated by the client. Each data point is the average over 10 simulation runs. For a given total rate, our scheme allocates more rate to the base layer than to the enhancement layer, while (WCF-EEP) provides an equal number of additional protection symbols to each layer. Thus, if there are random losses on any of the channels, due to a sudden unforeseen congestion for example, the client is more likely to receive at least the base layer using our scheme than using (WCF-EEP). Hence, our scheme is more robust as it implements graceful quality degradation. Note that the generic scheme (GEN) performs quite poorly: for the same total rate, more symbols get lost in the network due to the fact that more symbols are allocated to channels with higher loss probabilities.

6. CONCLUSIONS

This paper has presented a distributed streaming system for scalable video, based on rateless codes applied independently on each layer of each GOP. An optimal rate allocation strategy has been proposed in the ideal case of perfect knowledge of the network status. A low complexity heuristic-based algorithm has been designed, wich performs close to optimal. Additionally, it provides an increased robustness to incorrect channel characteristics estimation, by carefully allocating the remaining bandwidth as unequal error protection of the respective layers. Rateless codes are shown to provide a very interesting solution for low complexity distributed streaming systems, without need for complex synchronization schemes between servers.

ACKNOWLEDGEMENTS

The authors would like to thank Tiago Gasiba for the useful insights he provided on Raptor codes.

7. REFERENCES

[1] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, *"On Multiple Description Streaming with Content Delivery Networks"*, in Proc. of IEEE Infocom 2002.

[2] V. Agarwal and R. Rejaie, *"Adaptive Multi-Source Streaming in Heterogeneous Peer-to-Peer Networks"*, in Proc. of Multimedia Computing and Networking MMCN 2005, Jan 2005.

[3] C. Wu and B. Li, *"rStream: Resilient Peer-to-Peer Streaming with Rateless Codes"*, in Proc. of ACM Multimedia 2005, Nov 2005.

[4] A. Majumdar, R. Puri and K. Ramchandran, *"Distributed Multimedia Transmission from Multiple Servers"*, in Proc. of IEEE ICIP 2002.

[5] M. Luby, *"LT codes"*, in Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002.

[6] A. Shokrollahi, *"Raptor codes"*, Digital Fountain, Tech. Rep. DR2003-06-001, Jun 2003.

[7] J. Afzal, T. Stockhammer, T. Gasiba and W. Xu, *"System Design Options for Video Broadcasting over Wireless Networks"*, to appear in Proc. of IEEE CCNC 2006, Jan 2006.

[8] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, and W. Xu, *"Raptor Codes for Reliable Download Delivery in Wireless Broadcast Systems"*, to appear in Proc. of IEEE CCNC 2006, Jan 2006.