ADAPTIVE AND ROBUST MEDIA STREAMING OVER MULTIPLE CHANNELS WITH BURSTY LOSSES

Jean-Paul Wagner and Pascal Frossard

Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Institute - LTS4
CH-1015, Lausanne
Email: {Jean-Paul.Wagner, Pascal.Frossard}@epfl.ch

ABSTRACT

This paper addresses the problem of efficiently delivering a layered media stream from multiple senders to a single receiver, over channels that present correlated packet loss patterns. Using a digital fountain approach, the performance of a distributed streaming system is driven by the probability of receiving a given number of packets on aggregate over the multiple channels. In addition, such a system permits to avoid the need for communication between streaming servers. We devise an optimization problem whose solution provides the optimal number of packets that need to be transmitted per channel, in order to maximize the probability of correct decoding for a given media stream. Our findings indicate that it is in general important to consider both the Packet Loss Ratio (PLR) and Average Burst Length (ABL) in channel selection problems such as multipath routing or rate aggregation on multiple bursty channels. Finally we present a low-complexity algorithm which is able to quickly find a suboptimal yet effective solution to the combinatorial optimization problem.

1. INTRODUCTION

One of the main challenges in delivering video streams over the Internet, is to adaptively transmit video packets in order to match the available effective bandwidth, and provide a good quality of service to the media terminal. Today, the network does not give any hard quality of service guarantees. The channel parameters such as the available transmission rate or the loss process, are also prone to vary in time, hence to affect the quality of the received media. In order to deal with this situation, it has been proposed to aggregate the transmission rate using multiple channels, either from one server to one client using multiple distinct paths, or even from multiple distinct servers to one client. For example, it has been shown in [3] that the usage of multiple streaming servers in different network locations 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. Similarly, sources in modern peer-to-peer (P2P) systems may not be able or willing to commit to send the full video bitstream to a single client down-stream, especially if the rate of the stream is high. In such a scenario, aggregation from multiple peers is the only way to effectively deliver the requested stream at the desired quality. The recent emergence of multi-homed devices [5], which are able to access a networked resource simultaneously over multiple access channels, also calls for efficient channel aggregation methods.

An inherent problem of using multiple sources for the delivery the same stream however resides in the coordination between servers. In an effort to use the network resources efficiently, servers need to carefully coordinate their packet scheduling strategies [2], and avoid wasting bandwidth by duplicate packets. This in turn tends to render such a distributed streaming system overly complex and cumbersome, especially if channel parameters are dynamic. 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 layered 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 [11]. We extend our previous work in [10] by considering more realistic channel models that typically exhibit correlated loss patterns in the form of error bursts, as it is the case in most transmission scenarios. Further we assign a cost to every packet transmission, which depends on the used channel. Given this setup, we propose optimized sending schemes for a set of servers delivering a given media stream, and devise a heuristic-based algorithm that provides 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 prior to transmission [8].

The remainder of the paper is organized as follows. In Section 2 we explicit the considered framework and give a brief introduction to rateless codes by the example of Raptor codes. In Section 3 we devise an optimization problem whose solution drives the optimal performance of the considered system. In Section 4, we provide and validate a distributed heuristic-based algorithm to solve the optimization problem under complexity constraints. Section 5 provides simulation results before we conclude with Section 6.

2. FRAMEWORK

2.1 Network model

In an effort to make our notation transparent, we use capital letters for system-wide parameters, e.g., the rate of the delivered video, and small letters for parameters that depend on a specific channel. See Table (1) for examples. Further we use bold-face letters to denote vectors, i.e., $\mathbf{r} = (r_1, \ldots, r_N)$. All vectors are of dimension N, which is the number of peers/channels that are available to serve the client. For the sake of simplicity we will use the term *rate* throughout what follows to denote a number of packets per time unit. Hence we suppose packets of fixed size, and rates (in bit/s) that are multiples of the packets size.

R^T	video target rate
R^j	rate of layer j
R	total allocated rate
r_n	rate allocated on channel n
r_n^{max}	maximum available rate on channel n
γ_n	cost of sending a packet on channel n
p_n, q_n	parameters of the loss process on channel n
π_n, α_n	packet loss rate (PLR) and average burst length (ABL) of channel n

Table 1: Notation

Figure (1) illustrates the framework we consider in this paper. A client wants to retrieve a layered media stream from the network.

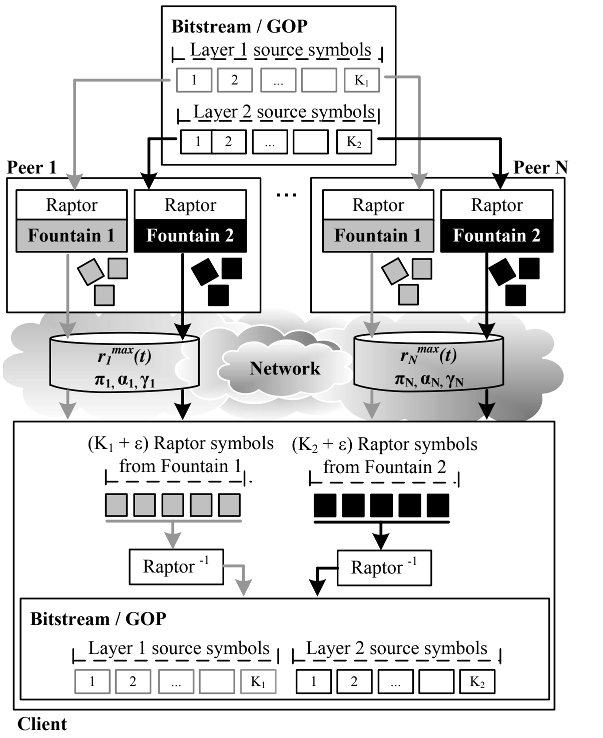

Figure 1: Streaming from multiple sources using Fountain Codes.

To do so, it has access to a set of N servers/peers all of which hold the video asset that is to be delivered. We suppose that the client can connect to each of the serving peers through a distinct channel [1]. The channel is characterized by an available rate $r_n^{max}(t)$ at time instant t, a packet loss rate π_n and the average packet loss burst length α_n. Finally, a cost γ_n is inferred when a packet is transmitted over the channel n. Note that we assume that the parameters $(\pi_n, \alpha_n, \gamma_n)$ do not depend on the transmission rate.

We consider that the loss process on each channel is governed by a Gilbert-Elliot model, which is a two-state Markov chain: in the *ON* state a packet is delivered to the client, whereas in the *OFF* state the packet is lost, see Figure (2). The transition probabilities p_n and q_n completely characterize the loss process on channel n, and the transition probability matrix reads as:

$$P_n = \begin{bmatrix} 1-p_n & p_n \\ q_n & 1-q_n \end{bmatrix} \qquad (1)$$

A transition is triggered at each time that a packet is sent over the channel. A high transmission rate on channel n results in more packets being transmitted, and hence implies a higher transition frequency of the corresponding Markov chain. Finally, referring to the outlined model, the PLR and ABL for channel n are given by the stationary probability of being in the *OFF* state and by the average residence time in the *OFF* state, respectively $\pi_n = \frac{p_n}{p_n+q_n}$ and $\alpha_n = \frac{1}{q_n}$.

2.2 Rateless Codes

With rateless codes, such as LT [6] and Raptor [9] 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 $(K+\varepsilon)$ packets can be used to decode the original K symbols. The notion of *Fountain code* stems from the analogy of a rateless code with a water Fountain (the unlimited number of symbols) from which any volume $(K+\varepsilon)$ satisfies the client needs, no matter which drops (symbols) of water it has obtained. It does not 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 servers. As long as the set of symbols they provide has been generated from the same input symbols, the encoded symbols will be different for each source with high probability, which means that every delivered symbol in the system gives the same amount of novel information to the client.

[1]Our model also extends to partially disjoint paths, where the bandwidth on shared network segments is adequately distributed between the different packet flows.

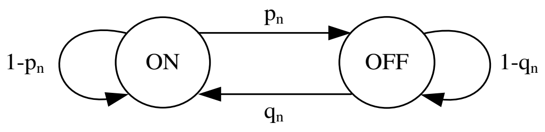

Figure 2: Two-state Markov model for channel n. A packet is transmitted if the chain is in state *ON*, and lost in state *OFF*.

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 [9, 1, 7].

We make the assumption that the media source has been encoded into a layered bitstream, and that it is made up of independently decodable segments. Adding a layer to the bitstream will increase either the quality of the decoded stream, its framerate or its resolution, depending on the source encoder that has been used. We will refer to an independently decodable segment of the bitstream as a Group of Pictures (GOP), borrowing the term without loss of generality from the MPEG terminology.

We propose to create one Fountain per layer and per GOP of the original bitstream, as depicted in Figure (1). This 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. As long as the client receives $K+\varepsilon$ distinct symbols on aggregate from all of the available sources, it will be able to decode the corresponding video data. 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 or source peers.

Given the above considerations it is clear that the performance of the framework will rely on the solution to two distinct problems: *i)* the rate r_i assigned to each peer needs to be split in an efficient manner among the available layers, and *ii)* we need to find a rate allocation $\mathbf{r}$ among the N available channels that maximizes the probability of receiving the number of symbols needed for decoding.

2.3 Peer rate distribution

In this subsection we suppose that there is a rate allocation $\mathbf{r}$ among the N available channels that maximizes the probability of receiving the number of symbols needed for decoding the bitstream, and hence minimizes the distortion at the decoder. In the next subsection we will present how $\mathbf{r}$ can be computed. Given $\mathbf{r}$, we need to assign a way of partitionning the streaming rate r_n between the

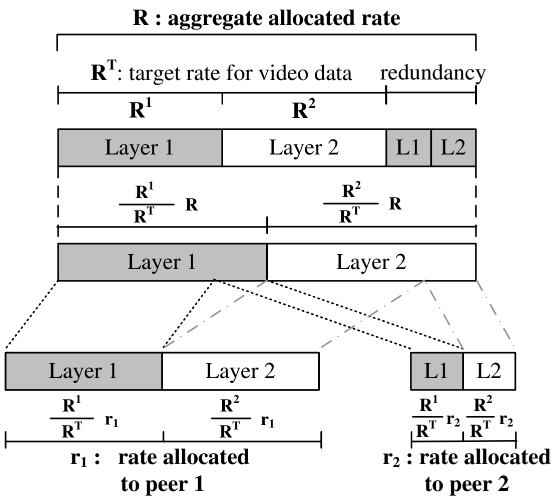

Figure 3: A simple example with two senders and two layers to transmit: each sender splits the rate that got allocated to it in a similar way between the video layers that are available, in order to maximize the delivered quality.

different layers of the video stream to each used source n. Recall that our primary objective is to keep servers synchronized while avoiding communication between streaming peers, and to provide a robust distributed streaming solution that benefits from server diversity. Without loss of generality, we assume first that the target rate R^T, which has to be received by the terminal, can be partitioned into a set of rates that correspond to the rates of the video layers, i.e.:

$$R^T = \sum_{j=1}^{L} R^j. \tag{2}$$

where R^j denotes the rate, including Raptor overhead, that needs to be received in order to decode layer j. We denote by R the aggregate allocated rate, which is the sum of the allocated rates satisfying Equation (5):

$$R = \sum_{n=1}^{N} r_n. \tag{3}$$

Clearly R includes the target rate R^T, plus some amount of redundant Raptor symbols, which are allocated to cope with the loss processes on the different channels. Under the assumption that the full target rate R^T should be delivered to the client, all layers are to be protected equally. Hence, the redundant rate $(R - R^T)$ is split among layers in order to reflect the relative size of each layer. This means that the total amount of Raptor symbols sent on aggregate for each layer, will be proportional to the fraction of rate each layer takes from the target rate R^T. Therefore, we finally get :

$$R = \sum_{n=1}^{N} \left(\sum_{j=1}^{L} \frac{R^j}{R^T} r_n \right), \tag{4}$$

which indicates that the distribution among layers of the allocated rate r_n at each peer n should be proportional to the relative size of each layer in the target rate R^T. It is important to note that each peer knows the sizes of each layer R^j. Each peer can thus optimally allocate the rate to each layer in a completely distributed way, when it knows the rate r_n that has been allocated to it, as well as the target rate R^T (see Figure (3) for an illustration). Although the proposed partition of r_n is not the only possible one, it has the advantage of being inherently distributed.

3. OPTIMAL RATE ALLOCATION PROBLEM

Let R^T be the target video rate that needs to be delivered to the client, including the Raptor overhead. Further let $P(R^T)$ be the probability of receiving at least the target rate. We want to find the rate allocation $\mathbf{r}^* = (r_1^*, \ldots, r_N^*)$ that achieves the optimal trade-off between maximizing $P(R^T)$ and minimizing the resulting cost, i.e.:

$$\mathbf{r}^* = \arg \max_{r_i \le r_i^{max}, \forall i} \left(P(R^T) - \lambda \frac{\mathbf{r} \cdot \gamma^\dagger}{\mathbf{r^{max}} \cdot \gamma^\dagger} \right), \tag{5}$$

where $\gamma^\dagger$ denotes the transpose of the cost vector $\gamma = (\gamma_1, \ldots, \gamma_N)$, and λ is a Lagrangian factor.

As stated, the problem exhibits combinatorial complexity. It is not trivial to compute $P(R^T)$, which is the probability of receiving at least R^T packets on aggregate over the N channels, each of which is defined by a Gilbert-Elliot loss process. We can express this probability in terms of the corresponding probability density function (pdf):

$$P(R^T) = 1 - \sum_{j=0}^{R^T - 1} p_R(j), \tag{6}$$

where $p_R(j)$ is the probability of losing j out of the R packets that are transmitted on aggregate over the N independent channels. This can in turn be computed by the convolution of the N probability density functions that give, for each channel n, the probability of losing i out of the r_n packets:

$$p_R = \bigotimes_{n=1}^{N} p_{r_n}. \tag{7}$$

Note that there is no closed form to express p_{r_n} in the case of a bursty loss channel, especially if the number of packets that are transmitted is relatively small and the pdf is thus ill approximated by a Normal density. These probability density functions can however be computed using the exact but iterative solution proposed by [4], at the price of increased computational complexity. In the next section we propose a suboptimal rate allocation solution, which achieves close to optimal performance with a reduced complexity.

4. HEURISTICS BASED ALGORITHM

As there is no simple constructive algorithm to find the optimal solution to the optimal rate allocation problem, we propose a low complexity client-driven rate allocation algorithm based on heuristics. The client has to determine the rates that need to be sent from each of the N streaming peers it has access to, and eventually to communicate the results to the servers. The allocation problem can be decomposed as follows :

- the client should first classify channels, in order to select *good* channels, whose usage minimizes the resulting overall cost, and simultaneously maximizes the likelihood of receiving allocated packets.
- it then determines how these channels should be used. It uses in priority the *good* channels, and fills them up successfully, until the joint likelihood of receiving the allocated packets satisfies a chosen termination condition:

$$\sum_{n=1}^{N} r_n \overline{P}(r_n) \ge R^T P^T. \tag{8}$$

Here P^T is the target probability of success, and $\overline{P}(r_n)$ is the probability of receiving all of the r_n allocated packets on channel n.

In the remainder of this section, we present two methods for classifying *good* channels, and we validate the respective heuristic-based algorithms in different streaming scenarios. Throughout this section we will assume that $\lambda = 1$ for the sake of clarity.

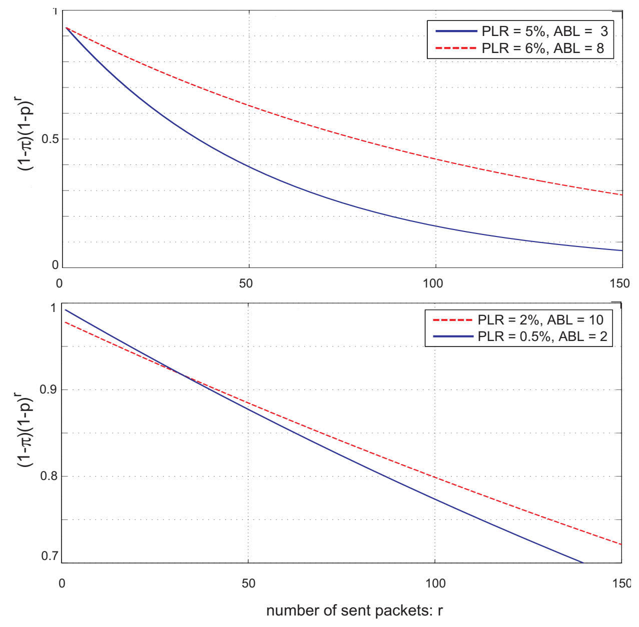

Figure 4: Two cases that show the importance of the ABL. The plots show $(1-\pi)(1-p)^r$ on the y-axis, and the number of sent packets r on the x-axis. *Top:* the probability of receiving all of the sent packets is consistently higher for the channel with *higher* PLR. *Bottom:* The probability curves for 2 channels may intersect.

4.1 Baseline

We first consider a *baseline* algorithm where channels are simply classified according to the average loss probability, and to the transmission cost. In other words, the channels are sorted according to $\frac{1-\pi_n}{\gamma_n}$.The average burst length is not considered in the baseline scheme, and the probability of receiving r_n packets is approximated as :

$$\overline{P}(r_n) = (1-\pi_n)^{r_n}. \tag{9}$$

Given this expression, which decreases with r_n, it is clear that the hierarchy that is induced by computing $\overline{P}(1)$ for each channel is preserved when more packets are transmitted. It follows that a selected channel is fully used, unless the termination constraint is met earlier.

4.2 ABL based Algorithm

The second algorithm is based on a more accurate estimate of receiving all of the allocated packets over channel n, which consider the burstiness of the loss process. Indeed, based on the Gilbert-Elliot model, the probability of receiving all the r_n packets is given by:

$$\overline{P}(r_n) = (1-\pi_n)(1-p_n)^{r_n-1}. \tag{10}$$

It becomes clear in this case that a simple water-filling algorithm that considers the loss probability π_n, but not the average length of bursts of errors, will fail in selecting the *good* channels first. For example, a channel with higher PLR can provide consistently higher success probabilities (see Figure (4)-top) depending on the respective ABL values. There are even cases where the choice of a particular channel depends on the number of packets to be sent (see Figure (4)-bottom). We propose to take this phenomenon into account in the selection of the *good* channels. We first classify the channels according to $\frac{\overline{P}(1)}{\gamma_n}$, with $\overline{P}(1)$ as given by Equation (10), in a similar way as the baseline algorithm. However, this hierarchy is now allowed to vary with the number of packets assigned to the channels. Using Equation (10) and considering a pair of channels (i,j) we compute the number of packets r_{ij} at which the hierarchical order of channels i and j in the channel classification changes.

Algo	prob	cost	metric
Baseline (eq)	$2.1 \cdot 10^{-2}$	n/a	n/a
ABL-based (eq)	$6.8 \cdot 10^{-4}$	n/a	n/a
Baseline (rand)	$6.5 \cdot 10^{-4}$	$2.1 \cdot 10^{-4}$	0.023
ABL-based (rand)	$5.0 \cdot 10^{-4}$	$4.48 \cdot 10^{-4}$	0.041

Table 2: Difference between heuristics-based algorithms and optimal rate allocation [MSE].

The intersection points are computed as:

$$\frac{(1-\pi_i)(1-p_i)^{r_{ij}-1}}{\gamma_i} = \frac{(1-\pi_j)(1-p_j)^{r_{ij}-1}}{\gamma_j} \tag{11}$$

$$r_{ij} = 1 + \frac{\log\left(\frac{1-\pi_j}{1-\pi_i}\right) + \log\left(\frac{\gamma_i}{\gamma_j}\right)}{\log\left(\frac{1-p_i}{1-p_j}\right)} \tag{12}$$

It should be noted that there is at most one such switching point for each channel pair in the rate interval of relevance $[0, max(r_i^{max}, r_j^{max}]$. Also, as $r_{ij} = r_{ji}$, we only need to compute at total of $\frac{N(N-1)}{2}$ switching points. The rest of the algorithm is a straightforward generalization of the water-filling method, with the additional step however that, while filling channel i, we switch the ongoing filling operation entirely to channel j if we need to allocate more than r_{ij} packets. The algorithm proceeds until the stopping criterion of Eq. (8) is satisfied.

4.3 Validation

In order to validate both algorithms, we have considered a scenario in which a client has access to three serving peers, each offering to provide a maximum rate of $r_n^{max} = 40$ packets. The target rate to be received has been set to $R^T = 30$ packets. We have computed the optimal rate allocations for 500 realizations on the three channels, with respect to Equation (5). In each realization, the PLR for the three channels has been selected uniformly in the interval from 1% to 10%. Similarly the ABL for each realization has been uniformly chosen for each channel in the interval from 2 to 20. We have also computed the rate allocation given by both the baseline algorithm and the ABL-based algorithm, in each network realization. In order to verify whether the optimal channels are chosen by the algorithms, we constrain the heuristics-based solutions to allocate a total number of packets that is equal to the total number of packets used under the optimal rate allocation. The distribution of the rate among the different channels may however differ significantly. The resulting rate allocations have been used to compute the effective success probability, the induced cost and the value of the optimization metric according to Equation (5). In Table 2 we report the average difference (in MSE) for the values of these metrics with respect to the performance of the optimal rate allocation, over 500 realizations. We propose two different sets of simulations: *i)* in the first set (eq), all channel costs were always equal, *ii)* in the second set (rand), the costs for each channel were chosen randomly between 0.01 and 1 in each realization (with $\lambda = 1$). Based on these experiments, we conclude that the algorithm we propose provides a robust channel selection algorithm when the channels are characterized by both ABL and PLR, and that consideration of the ABL in the channel selection is quite important. However, when the optimization function includes transmission costs, similar performance is achieved for both channel selection schemes.

However, in a realistic scenario neither of the two algorithms is aware of how many packets need to be allocated on either of the chosen channels. The termination condition is effectively given by Equation (8). As the baseline algorithm does not take into account the ABL, it makes a consistent error in estimating the probability of effectively receiving the allocated packets: hence it does not know when to terminate. This behavior is illustrated using a simple yet

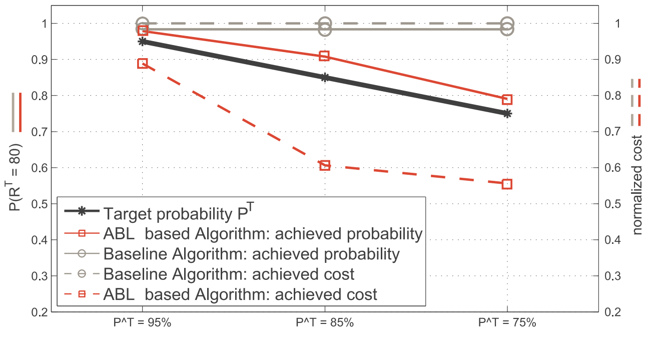

Figure 5: The probability of receiving $R^T = 80$ packets (left axis) and the normalized costs induced by the two rate allocations (right axis) are shown for different termination conditions. As opposed to the baseline algorithm, the ABL-based algorithm is capable of terminating correctly.

representative example in Figure (5). The considered scenario is as follows. A client aims at retrieving a total of $R^T = 80$ packets. There are 3 available peers, each willing to transmit a maximum of 60 packets. The loss parameters for the 3 channels are $(\pi_1 = 5\%, \alpha_1 = 3)$, $(\pi_2 = 6\%, \alpha_2 = 8)$ and $(\pi_3 = 7\%, \alpha_1 = 5)$ respectively. We vary the target probability of success P^T in the right-hand side of Equation (8) from 95% down to 75%. It can be observed that the baseline algorithm always ends up using all the available channels at full rate and implying maximal cost. The ABL-based algorithm, through its more accurate estimation of the success probability, is able to tune the amount of packets that is injected in the network so that the effective success probability induced by the resulting rate allocation follows the target probability P^T.

5. SIMULATION RESULTS

We have encoded the SOCCER test sequence (CIF, 30Hz) using the H.264-SVC reference codec into a base layer ($\simeq 300kbps$) and an SNR-enhancement layer ($\simeq 700kbps$). The GOP size is set to 32 frames. Each GOP/Layer of the bitstream has then been encoded into a digital fountain using a Raptor code, where the Raptor symbol size has been set to 16 bytes. The client retrieves the stream by aggregating 3 available channels that all have a capacity of 512 kbps. Clearly, only the base layer could possibly be transmitted in the absence of channel aggregation, resulting in a maximum average (Y)-PSNR of 33.03dB. Two of the available channels have similar loss characteristics given by the tuple $(\pi_{1,2} = 0.03, \alpha_{1,2} = 2)$, whereas the third channel is given by $(\pi_3 = 0.01, \alpha_3 = 6)$. The cost for sending a packet on either channel is equal to $\gamma_{1,2,3} = 1$ and the network packet size was set to 1280 bytes. Following the above considerations, both layers can be correctly decoded whenever 105 packets have been received, 29 of which originating from the base layer fountain, the remaining 76 from the enhancement layer fountain. We have run both rate allocation algorithms on the above scenario, using $R^T = 105$ and $P^T = 0.9$, and all three senders have split their respective allocated rate among the 2 layers as given by Equation (4).

As both algorithms tend to over-provision the system, it is not surprising that the client is always able to retrieve the 2 available layers over the 3 channels, thus receiving the best available quality. However, as already observed above, the baseline algorithm tends to being unable to terminate correctly as it underestimates the reception probability of the allocated packets. It hence allocates all the available resources to the streaming process, resulting in a waste of bandwidth.

The results are given in Table (3) where the optimal allocation given by Equation (5) is given as reference. The *cost* column indicates the number of packets injected into the network. The *redundancy* column shows the percentage of redundant Raptor symbols

Algo	Y-PSNR [dB]	cost	redundancy [%]
Baseline	38.75	150	38.09
ABL-based	38.75	125	15.23
Optimal	38.75	115	4.76

Table 3: PSNR, inferred cost and received redundancy for the two proposed algorithms and the optimal rate allocation.

that are received for either rate allocation, as compared to the 105 symbols necessary for decoding. Note that in this case, the redundancy ratio for the baseline algorithm is only bounded through the available channel capacity, as the algorithm does not terminate correctly. These results confirm that, using the ABL-based algorithm, the proposed framework is able to provide close to optimal performance for delivering scalable encoded media streams.

6. CONCLUSIONS AND FUTURE WORK

We have presented a low-complexity framework for the delivery of a given layered media stream from multiple senders to a single receiver, over channels that present correlated packet loss patterns. As our distributed streaming system relies on the aggregate reception of a sufficient number of packets for correct decoding, we have devised an optimization problem whose solution gives the optimal rate allocation that maximizes the probability of lossless decoding. We have provided and validated a heuristics-based algorithm, which is able to quickly provide a sub-optimal solution to the combinatorial problem. We intend to further investigate the proposed framework with respect to dynamically changing network parameters in future work. Our findings finally indicate that it is important to consider both the Packet Loss Ratio (PLR) and Average Burst Length (ABL) when addressing channel selection in multipath routing or rate aggregation over bursty channels.

REFERENCES

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

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

[4] P. Frossard, *"FEC performance in multimedia streaming"*, IEEE Commun. Lett. vol. 5, pp. 122–124, Mar. 2001.

[5] H.Y. Hsieh and R. Sivakumar, *"A Transport Layer Approach for Achieving Aggregate Bandwidths on Multi-Homed Mobile Hosts"*, in Springer, Wireless Networks, Volume 11, Number 1-2, Jan 2005.

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

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

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

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

[10] J.-P. Wagner, J. Chakareski and P. Frossard, *"Streaming of Scalable Video from Multiple Servers using Rateless Codes"*, in Proc. of ICME 2006.

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