Reduced-Overhead Multicasting of Different QoS Data Classes

Zohreh Abdeyazdan, Masoud Ardakani, Chintha Tellambura
Department of Electrical and Computer Engineering, University of Alberta
Email: {abdeyazd,ardakani,chintha}@ualberta.ca

***Abstract*—In this paper we study the problem of transmitting different quality of service (QoS) data classes on a broadcast erasure channel where each data class is intended for a group of users depending on their erasure rate. For real-time applications such as multimedia, data block length is typically small. Over a small data block length, the behavior of the erasure channel may significantly differ from its average. Thus, to guarantee a certain QoS, a considerable overhead may be needed. We provide a finite block length analysis of the effect of erasure on the overhead. We use this analysis to investigate a packet construction method that mitigated the effect of short block length.**

I. Introduction

Broadcasting data to a number of users with different channel quality can efficiently be done using fountain codes [1]–[3]. The transmitter, however, has to continue the transmission of a data block until all the receivers can decode it. Only at this point, the transmitter can start transmission of the next block of data. In this setup, although, the best user can receive one block very quickly, it must remain idle until the transmission of the next block has started.

In some applications, such as multimedia, different quality of service (QoS) data classes can be defined to be sent to different users. More specifically, users can be classified according to their reception quality to receive different quality multimedia. Since transmissions are done in packets, the user quality can be defined based on the packet erasure rate. [1] A user with a low erasure rate can potentially receive more QoS data classes than a user with a higher erasure rate. This is in contrast with slowing down the high quality users by the user with worst channel quality. As a result, users that experience good channel quality can receive high quality multimedia during the same time that users with poor channel condition receive the same multimedia at a lower quality. Another example is in vehicular networks. All the vehicles in a highway can receive the necessary information about the traffic and road conditions while those with better channel condition may receive other data classes such as data for entertaining applications or their requested data.

Here, we study the problem of multicasting different QoS data classes on erasure channels. The source node has data packets from different QoS classes intended for users that experience various erasure rates. Users with smaller erasure rates can receive more packets than users with poorer channel conditions. Different solutions have been suggested using rateless codes for multimedia multicasting with different QoS data classes. For instance, [4], [5] propose expanding window fountain codes. This method, however, alters the degree distribution of the fountain code whose increased complexity of decoding is not desirable.

[6], [7] introduce a solution to this problem using fountain codes and a scheduling algorithm based on data interleaving. Since this method is based on scheduling and data interleaving, we will refer to it as the SDI method. In this solution, there are two data classes where users with good channel quality will receive data from both classes. Users with poor channel quality will only receive one class of data. Since SDI is based on scheduling, data from different data classes are transmitted separately. As we will see later, for applications with small data block sizes such as realtime applications, this data separation may result in a considerable overhead needed to guarantee a certain probability of successful transmission. The source of this overhead is the fact that the behavior of the erasure channel may vary significantly from its average when used over a small block length.

Another solution to this problem that will be discussed in this paper is to mix the data of all QoS classes together [8]. In other words, the source can transmit the encoded data of all classes together over all time slots. We will refer to this method as MTS. This idea is used in [8] to formulate and solve the allocation problem of source bits of different data classes based on a defined cost criterion.

In this paper, we investigate the effect of this approach on the overhead. For this purpose, we first provide a finite block length analysis. Using, this analysis, we compare MTS with SDI method in different erasure rates to show that MTS enjoys a lower overhead. Thus, for applications such as multimedia which usually have a short block length, MTS should be the method of choice.

The rest of this paper is organized as follows. In Section II, the system model and the problem statement are provided. Section III will discuss problem solutions, starting with a review of SDI method, discussing the effect of finite block length on the system, and then discussing MTS solution. Section IV provides the numerical results, where MTS method is compared with SDI. The paper is concluded in Section V.

[1]For simplicity, we use "erasure rate" instead of "packet erasure rate" in the remainder of the paper, but we notice that an erasure means one packet is lost.

2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)
978-1-4673-1433-6/12/$31.00 ©2012 IEEE

II. System Model and Problem Statement

Consider a source that multicasts data to R users. In multicasting, a direct transmission from the source is not a desirable solution. This is because, even if one of the users fails to receive the packet, the packet must be retransmitted. Thus, as the number of users increases, direct transmission becomes less desirable. It is well known that fountain codes can avoid this problem [1]–[3]. With fountain codes, the source continues the transmission until the data for all intended users is provided. Another benefit of fountain coding over direct transmission is that instead of needing a feedback per each received packet per user, it only needs a feedback at the end of reception of the whole block by each user. Moreover, fountain codes can handle users with various or even unknown erasure rates. Thus, here we assume that data is transmitted using fountain codes.

The R users experience different erasure rates. Therefore, they can be classified into L different classes based on their erasure rates where $L \leq R$. Users in the same class are assumed to have equal erasure rate [2].

Let e_i be the erasure rate of users in class i. The receiving rate can then be defined as $r_i = 1 - e_i$. We also order user classes according to their erasure rate such that for any two user classes $i, j \in \{1, \ldots, L\}$, $i < j \Rightarrow e_i > e_j$. A typical user in class $i \in \{1, \ldots, L\}$ will be referred to as u_i. With this definition, if $i < j$, potentially u_j can receive more packets than u_i during the same period of time. Therefore, different QoS classes can be defined for users with various erasure rates.

Multimedia streaming is an application of the setup described above because different QoS levels of data can naturally be defined since users can receive multimedia with different qualities. Suppose the data stream is split into M classes, $C_1, C_2, \ldots, C_M$. Here, C_1 is the part of stream which provides the lowest quality multimedia and therefore necessary for any user who wants to receive the multimedia stream. Packets in $C_m, m \geq 2$ are intended for users that want higher quality of service. So, the more QoS classes a user receives, the better the quality of the stream it receives. The best quality is provided to the users that receive all M classes. Similar discussions are valid when these data classes are completely independent and from various applications with different priorities.

Without loss of generality we assume $L = M$. In other words, the number of classes of users is equal to the number of different QoS classes of data. With this assumption, by class i, we mean the users whose erasure rate is e_i and expect to receive data from classes $C_1, C_2, \ldots, C_i$. We also assume that all data classes are greedy, meaning that they always have data to send.

Similarly, one can view this as a system, where M data classes are broadcasted on the channel. Each data class has a predefined erasure threshold, and its data is intended for any user whose erasure rate is below the threshold. In this view, e_m represents the predefined threshold for class m. Notice that in this setup, the transmitter does not need to know the erasure rates of users. Moreover, users are naturally classified to different classes according to their erasure rates.

For the defined system, the problem is to devise a data transmission algorithm that $\forall i, 1 \leq i \leq M$ provides $C_1, \ldots, C_i$ to u_i with a failure rate guaranteed to be less than δ for all user classes. A failure at user class i is defined as not having enough received data from data classes $C_1, \ldots, C_i$ to be able to decode the data of all i classes.

Here, we assume erasures are the only reason for packet loss, so our results are valid for memoryless erasure channels.

The next section explains two solutions to this problem. One solution is SDI method and the other is the MTS method. To motivate MTS solution, the effect of finite block length on the system is also studied in the next section.

III. Problem Solution

As mentioned earlier, to handle the problem of various erasure rates of different users, the methods discussed in this section use fountain codes [1]–[3]. When N data packets are fountain coded, any user who received $N' = (1 + \epsilon)N$ encoded packets can decode the N data packets. Here, ϵ is the overhead of the fountain code and is typically due to the linear dependency of some received encoded packets and the suboptimal decoding. Since all methods discussed in this section use fountain codes, we treat N' as the block size when comparing these methods. In other words, the block size is defined as the number of encoded packets needed at the user side. This way, we can compare different methods without the need to consider the fountain coding. From this point of view, by one bit in data class m we mean a fountain encoded bit of this data class. In the remainder of this work, we use N instead of N' for the ease of notations.

A. SDI Method

For the defined system with M data classes and different erasure rates, [6] proposed an interleaving based method to transmit data. In this method, each class has its own Raptor [3] encoder and therefore data of each layer (class) will first be encoded internally. At each time slot, class $m \in \{1, \ldots, M\}$ will be chosen with probability α_m. Then an encoded packet from this class will be broadcasted over the channel. [6], [7] optimize probabilities $\alpha_1, \ldots, \alpha_M$ based on the channel erasure rates. The results show that these probabilities are proportional to the erasure rate of their corresponding classes.

Please note that in the time slots that the source transmits packets from C_i, $u_j, \forall j < i$, is in idle mode. Moreover, since u_j has a higher erasure rate than u_i, the total number of packets transmitted from C_j is more than what u_i needs. Thus, even u_i will be in idle mode for a portion of time when packets from C_j are transmitted.

Before discussing MTS [8], we provide a finite block length analysis of the system. While MTS is proposed in [8] for the first time, it is suggested for solving the allocation problem of various data classes. Here, we discuss another advantage of MTS, i.e., its lower overhead compared to SDI. For this

[2]In practice, users with almost equal erasure rates are grouped together

purpose, we first need a finite length analysis of the system. Also, in order to make this advantage more clear, we will review MTS from a new point of view in Section III-C.

B. Finite Block Length

On an erasure channel, erasures happen randomly and independently. For finite block length, the actual number of erasures may differ from the average expected number. Thus, if we need N received packets at the output of a channel with erasure rate e, $N/(1-e)$ transmissions may not be enough. In fact, to guarantee N received packets with high probability, the number of transmissions must be larger than $N/(1-e)$. Thus, the number of extra packets needed can be define as the transmission overhead. This overhead is especially important and fairly large in applications that have small block sizes such as realtime applications. Please note that this overhead is different from the overhead of fountain codes that we discussed earlier.

We previously defined failure for user in class i, $i \in \{1,\dots,M\}$ as "u_i does not receive enough encoded packets to decode the whole block of data from classes $1,2,\dots,i$." Then, to guarantee a probability of failure smaller than δ, one can find the needed transmission overhead of each data class.

Now, consider the data class m, with data blocks of size N_m packets. For users in any class l, $l \geq m$ that experience channel erasure rate e_l, the number of received packets X_l from data class C_m after K_m transmissions is a Binomial$(K_m, 1-e_l)$ random variable. For guaranteed transmission of C_m to user class l, we wish to have $X_l \geq N_m$ with probability at least $1-\delta$ or equivalently

$$p[X_l < N_m] < \delta.$$

When N_m is larger than a few hundreds, this Binomial distribution can accurately be approximated with a Gaussian distribution. Thus, $X_l \sim \mathcal{N}(K_m(1-e_l), K_m e_l(1-e_l))$ and $p(X_l < N_m)$ can be found using the Q function. Thus, the reception condition for u_l is

$$Q\left(\frac{K_m(1-e_l)-N_m}{\sqrt{K_m e_l(1-e_l)}}\right) < \delta. \tag{1}$$

This means that for finite block length, the number of transmissions K_m must be larger than $N_m/(1-e_l)$. Thus, a transmission overhead representing the number of extra encoded packets (compared to the expected number) can be defined as

$$k_{l,m} = K_m - \frac{N_m}{1-e_l}.$$

Although u_l is supposed to receive data from C_1 to C_l, since its erasure rate e_l is smaller than $e_1, e_2 \dots, e_{l-1}$, the overhead considered for those user classes would satisfy the reception condition of u_l. Thus, among the users that receive data from data class C_m, i.e., $u_m, u_{m+1}, \dots, u_M$, the highest erasure rate belongs to u_m. As a result, u_m needs the largest overhead among all user classes that need C_m. In other words, for each data class m we only need to satisfy the reception condition for u_m. Thus, the actual needed overhead for class m is

$$k_m = K_m - \frac{N_m}{1-e_m},$$

where K_m can be found using

$$Q\left(\frac{K_m(1-e_m)-N_m}{\sqrt{K_m e_m(1-e_m)}}\right) < \delta. \tag{2}$$

The overall overhead is the sum of the overheads of each data class, which can be found as

$$k_{\text{total}} = \sum_{i=1}^{M} k_i \tag{3}$$

Clearly as the block length increases, the needed overhead compared to the block length becomes smaller. For small block length, however, the transmitting time of overhead can be a significant portion of the total transmission time. We like to emphasize that by sending k_m extra encoded packets for each data class m, the transmitter guarantees a probability of failure less than δ.

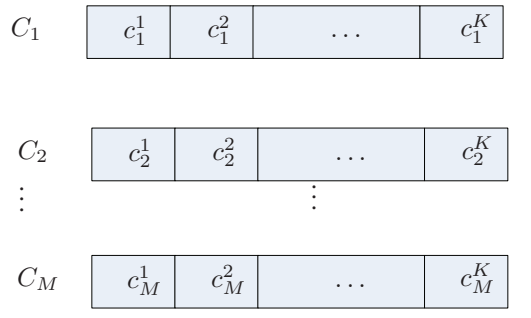
Now let us define a time slot, as the time period needed to transmit one packet over the channel. Here, the total of K time slots are available where $K = K_1 + \dots + K_M$. In SDI method, these time slots are first divided among M data classes, meaning that the transmission time allocated to data class m is $\alpha_m K$. Depending on the number of data classes, $\alpha_m K$ can be significantly smaller than K resulting in a significant needed overhead to combat the finite block length effect.

Our main insight in this work is to reduce the overhead by sending the data of all classes over all time slots, so that the needed overhead will be calculated for time K instead of $\alpha_m K$. As a result, the needed overhead is for a much larger data block and therefore is smaller. The details of this idea is provided in the next section.

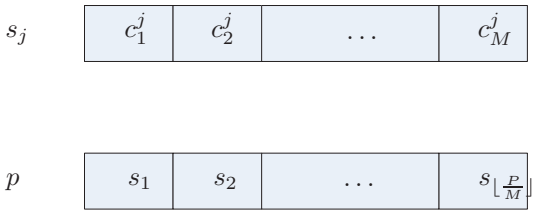
C. The MTS Method

Considering M QoS data classes, $C_1, C_2, \dots, C_M$, let c_i^j represent the jth encoded bit of data class C_i. We define symbol s_j as an ordered M-tuple constructed from fountain encoded bits of all data classes i.e. $s_j = (c_1^j, c_2^j, \dots, c_M^j)$. A symbol in this method is the smallest unit of data from which a packet is formed. A transmitted packet, therefore, contains $\lfloor P/M \rfloor$ symbols where P is the size of a packet. Fig. 1(a) shows different data classes and Fig. 1(b) shows a symbol s_j and a transmitted packet p in this method. We call this method MTS, because it works with M-tuple symbols.

In MTS, similar to SDI, each data class has its own fountain encoder. In other words, the data of each QoS data class is fountain encoded separately and then the encoded bits are used to generate symbols. Thus, as long as each user receives enough number of encoded packets from a data block of a certain data class, it will be able to decode the whole block. This means that our method does not effect the coding part.



(a) Blocks of data from different classes, $C_1, C_2, \ldots, C_M$ which contain encoded bits



(b) A generic symbol, s_j, and a transmitted packet, p, which is formed by defined symbols

Fig. 1. Generating transmitted packets in MTS method

Thus, comparisons can be made without considering the effect of fountain codes.

Although in MTS the same number of bits from all classes are put into each packet, it does not mean that users of all classes are receiving the same amount of data. This is because, each class is working independently and the symbols are created by encoded bits (not raw bits). To clarify this point, let us consider data class C_m. The number of transmitted encoded bits is determined based on three factors. The first one is the data block length V_m (similar to N_m in SDI method) which is determined based on the requirements of the application. The second factor is the erasure rate e_m of the worst case user that can receive this data. So, on average, $\frac{V_m}{1-e_m}$ encoded bits are required to be transmitted, but as we discussed in Section III, for guaranteed QoS, the actual number of needed transmissions is $K'_m = \frac{V_m}{1-e_m} + k'_m$, where k'_m is the overhead which depends on the acceptable probability of failure δ and the data block length. Here, K'_m which is the total transmission time for class m, is equal to K for $m = 1, \ldots, M$, because encoded bits of any class are present in all K transmitted packets.

To summarize, $\lfloor P/M \rfloor K$ fountain encoded bits from C_m are used in the construction of symbols during K time slots which include overhead bits as well. As soon as these symbols are constructed, the next data block from C_m will be used for construction of new symbols. Any other data class is performing a similar procedure in parallel and independently from class m. As we mentioned, it is assumed that all data classes are greedy and always have data to send. To clarify, Fig. 2 shows an example of different data classes, where each class (C_i) has a block of data (V_i) plus the needed overhead

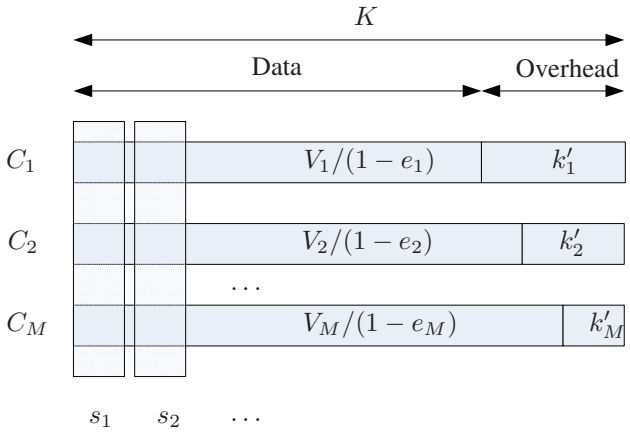


Fig. 2. The frame shows overheads and data of different classes for a fixed period of time K. It also shows the construction of symbols in MTS method.

(k'_i). As we can see, depending on the various erasure rates, the length of the overhead of different data classes are not the same. This figure also shows how symbols and consequently packets are constructed in MTS method.

The benefit of MTS, as discussed earlier, is that data class m, instead of being sent over only $\alpha_m K$, is sent over K, i.e., all time slots. This way, the data experiences an erasure channel whose behavior is closer to its average. Thus, the needed overhead for guaranteed reception is reduced. By using Eq. (2) for MTS, we have

$$Q\left(\frac{K(1-e_m) - V_m}{\sqrt{K e_m (1-e_m)}}\right) < \delta. \tag{4}$$

and V_m can be calculated for each class m. It is important to note that in Eq. (4) and in Fig. 2 , k'_m and V_m are number of bits not packets. This should be considered when comparing the overhead of two methods. Having V_m and K, the overhead of class m can be found as

$$k'_m = K - \frac{V_m}{1-e_m},$$

where k'_m is the number of overhead bits from data class C_m. To compare the overhead of MTS with that of SDI, the total number of overhead packets for MTS is calculated as the average of k'_i's which is

$$k'_{\text{total}} = \frac{\sum_{i=1}^{M} k'_i}{M} \tag{5}$$

The average arises since different data classes have various size overheads. The numerical results in the next section verify that MTS reduces the overhead of SDI. Moreover, MTS does not need any optimization and the implementation is fairly simple.

IV. Numerical Results

In this Section, we numerically compare the overhead of MTS and SDI methods where the number of data classes are

TABLE I

THE OVERHEAD OF SDI AND MTS METHODS FOR VARIOUS RECEIVING RATES OF DATA CLASS 1 AND 2, (r_1, r_2), WHEN $K = 1000$ AND $M = 2$ AND THE PERCENTAGE OF OVERHEAD REDUCTION

(r_1, r_2)	$(0.4, 0.45)$	$(0.4, 0.65)$	$(0.4, 0.83)$	$(0.5, 0.63)$	$(0.5, 0.75)$	$(0.5, 0.93)$	$(0.6, 0.71)$	$(0.6, 0.85)$	$(0.6, 0.93)$	$(0.7, 0.81)$	$(0.7, 0.89)$	$(0.7, 0.95)$
k_{SDI}	231	193	162	174	153	114	141	116	95	106	90	62
k'_{MTS}	158	132	110	119	105	79	97	80	66	74	62	43
OR%	32	31	31	31	31	31	31	31	31	30	30	30

TABLE II

THE OVERHEAD OF SDI AND MTS METHODS FOR VARIOUS RECEIVING RATES OF DATA CLASS 1 AND 2, (r_1, r_2), WHEN $K = 2000$ AND $M = 2$ AND THE PERCENTAGE OF OVERHEAD REDUCTION

(r_1, r_2)	$(0.4, 0.45)$	$(0.4, 0.65)$	$(0.4, 0.83)$	$(0.5, 0.63)$	$(0.5, 0.75)$	$(0.5, 0.93)$	$(0.6, 0.71)$	$(0.6, 0.85)$	$(0.6, 0.93)$	$(0.7, 0.81)$	$(0.7, 0.89)$	$(0.7, 0.95)$
k_{SDI}	324	270	226	243	215	160	198	163	134	149	126	88
k'_{MTS}	223	187	157	169	149	111	137	113	93	104	88	61
OR%	31%	31%	31%	31%	31%	31%	30%	30%	30%	30%	30%	30%

TABLE III

THE OVERHEAD OF SDI AND MTS METHODS FOR VARIOUS RECEIVING RATES OF CLASS 1, 2, AND 3, (r_1, r_2, r_3), WHEN M=3 AND $K = 1000$ AND THE OVERHEAD REDUCTION IN PERCENTAGE

(r_1, r_2, r_3)	$(0.3, 0.47, 0.6)$	$(0.3, 0.53, 0.72)$	$(0.3, 0.69, 0.94)$	$(0.4, 0.55, 0.63)$	$(0.4, 0.65, 0.73)$	$(0.4, 0.67, 0.9)$	$(0.5, 0.63, 0.76)$	$(0.5, 0.65, 0.84)$	$(0.5, 0.71, 0.88)$
k_{SDI}	300	266	178	255	241	194	239	203	194
k'_{MTS}	162	144	98	139	131	107	130	111	106
OR%	46%	46%	45%	45%	45%	45%	45%	45%	45%

$M = 2$ and $M = 3$ and the total transmission time, i.e., data plus overhead is equal for both methods. We consider total transmission time as $K = 1000$ and $K = 2000$ time slots. For a certain δ and a fixed K with various erasure rates, the overhead is found. Here we added the constraint that after K transmissions all classes should be done by transmission of one block. Thus, the block size of different classes vary. This has no effect on the overhead comparisons of these two methods, and is merely done for the ease of comparison.

In SDI, the overhead for data class m, k_m, $m = 1, \ldots, M$ is found for each class separately and the overall overhead is obtained from Eq. (3). For MTS method, by using Eq. (5) the overhead can be calculated as well.

Table I provides the results for $M = 2$ and $K = 1000$. (r_1, r_2) is the receiving rate of users in class one and two respectively. Table II represents the same results for $K = 2000$ and Table III shows the results for $M = 3$ and $K = 1000$ where in all tables r_i shows the receiving rate of class i. To compare these methods more directly, we also report the percentage of overhead reduction (OR) by MTS. If k is the overhead of SDI and k' is the overhead of MTS, then the overhead is reduced by

$$\text{OR} = \frac{\text{k} - \text{k}'}{\text{k}}$$

The results shows that the needed overhead of MTS method is smaller than SDI method in all three cases. As expected, by increasing K, the ratio $\text{overhead}/\text{K}$ for both methods is reduced. For asymptotically large K, since the channel behavior converges to its average, the overhead compared to the data block size will be negligible for both methods.

V. CONCLUSION

We studied the problem of transmitting different QoS data classes to users with various erasure rates. Each data class was intended for any user whose erasure rate was better than a predefined threshold. Since for some applications such as multimedia small block length is needed and in that case the number of erasures introduced by the channel can significantly be greater than its average, a large overhead may be needed for acceptable probability of success.We studied the effect of the block length on the overhead and provided an analysis to compare different solutions in terms of their overhead. Our results showed that MTS requires a much lower overhead compared to SDI.

REFERENCES

[1] D. MacKay, "Fountain codes," *Communications, IEE Proceedings-*, vol. 152, no. 6, pp. 1062–1068, Dec. 2005.

[2] M. Luby, "LT codes," in *Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science (FOCS)*, Vancouver, BC, Canada, Nov. 2002, pp. 271–280.

[3] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[4] D. Sejdinovic, D. Vukobratovic, A. Doufexi, V. Senk, and R. Piechocki, "Expanding window fountain codes for unequal error protection," *Communications, IEEE Transactions on*, vol. 57, no. 9, pp. 2510 –2516, Sep. 2009.

[5] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, and Z. Xiong, "Scalable video multicast using expanding window fountain codes," *Multimedia, IEEE Transactions on*, vol. 11, no. 6, pp. 1094 –1104, Oct. 2009.

[6] C. Yu, S. Blostein, and C. Wai-Yip, "Optimization of rateless coding for multimedia multicasting," in *Broadband Multimedia Systems and Broadcasting (BMSB), 2010 IEEE International Symposium on*, Shanghai, China, Mar. 2010, p. 1.

[7] ——, "Unequal error protection rateless coding design for multimedia multicasting," in *Int. Symp. on Inform. Theory (ISIT)*, Austin, TX, Jun. 2010, p. 2438.

[8] W. Sheng, W.-Y. Chan, S. D. Blostein, and Y. Cao, "Asynchronous and reliable multimedia multicast with heterogeneous QoS constraints," in *Communications (ICC), 2010 IEEE International Conference on*, Cape Town, South Africa, May 2010, pp. 1 –6.