Merging Network Coding with Feedback Management in Multicast Streaming

André Moreira
Instituto de Telecomunicações
University of Porto
Porto, Portugal
andre.moreira@fe.up.pt

Luis Almeida
MRTC
Mälardalen University
Västeras, Sweden
lda@mdh.se

Daniel E. Lucani
Dept. of Electronic Systems
Aalborg University
Aalborg, Denmark
del@es.aau.dk

ABSTRACT

Reliable multicast over wireless poses interesting challenges arising from the unreliable nature of the wireless medium. Recovering lost packets is particularly challenging in multicast scenarios since different receivers lose different packets. For this reason, simply retransmitting packets does not scale well with the number of receivers and particularly with the packet loss rate. A more efficient alternative is to use erasure codes to generate packets that can help many receivers at the same time. In this paper, we propose using online network coding to send coded packets that repair losses according to feedback reports sent by the clients. In particular, we propose using a recently developed scheduler for controlling feedback reports, which also allows differentiating the QoS provided to clients, and combine it with an online coding approach to provide novel stochastic guarantees of worst-case delay as required for QoS sensitive applications. We show preliminary simulation results that confirm the bounded decoding delay of our approach in a streaming application.

1. INTRODUCTION

Video streaming is gaining more popularity, either to access pre-stored video materials or as a means for live broadcasting. A particular scenario of this latter case is when many wireless clients try to access the same video stream in the same hot spot, e.g., associated to a sports or cultural event. This case, for bandwidth efficiency and scalability reasons, requires multicast transmissions over a wireless medium which typically uses unacknowledged packets at the link layer, and are thus unreliable. In heavy loaded conditions, packet losses increase and the overall quality of the stream reception degrades, with strong asymmetries among clients. It is even possible that packet losses affect most of the video frames causing a complete failure of the streaming service.

In general, managing Quality of Service (QoS) provided to clients requires sending feedback to the streaming server so that it can resend the missing information and differentiate the QoS offered to each client. However, when many clients are involved there will be many feedback messages (acknowledgements), increasing the processing load in the server and the load in the wireless medium. For this purpose, we propose using the multicast protocol for video streaming that was proposed in [2], which synchronizes and schedules feedback from clients improving QoS. The proposed protocol allows fine tuning the balance between the bandwidth used by feedback, the number of clients served and the QoS of the streaming service. Preliminary results in [2] show the advantage of synchronizing the transmissions of the feedback reports. Moreover, these feedback reports also support individualized client Service Level Agreements (SLAs) actuating on the respective feedback rates and thus offering differentiated QoS.

In this paper, instead of using simple retransmissions to recover lost packets we use erasure codes to generate packets that can help many receivers at the same time. Erasure codes are usually designed to maximize throughput and ultimately achieve channel capacity. However, such design comes at the cost of additional delay resulting from the time needed to decode the original data. For delay sensitive applications such as streaming, this is a critical issue and better trade-offs between throughput and decoding delay are usually sought. A key technique that has been explored is online coding, a feature of network coding that allows packets to be encoded as they arrive to the encoder. This cuts on the latency when compared with other codes that need to wait for a whole block of packets to start encoding, and thus makes it suitable for streaming applications. Moreover, feedback can be used to dynamically adjust which packets should be encoded to better satisfy specific performance requirements.

Online coding for streaming applications was already addressed in [3]. However, the solution in [3] did not consider the use of a feedback reports scheduler. In this paper, we propose using this scheduler together with a different coding approach that provides stochastic guarantees of worst-case decoding delay, thus also allowing to compute the minimum buffer requirements that avoid missing deadlines in the streaming process.

In the following section, we briefly discuss some related work in coding, highlighting the novelty of our approach. Then we describe the system model organized in terms of the network model, coding mechanism and system architecture. The following section shows and discusses preliminary results focused on the impact of different coding schemes. We then quickly elaborate on the dynamic adaptation of different system parameters. Finally, the last section draws a conclusion and discusses the ongoing work.

2. RELATED WORK

In the seminal work of online network coding [7], the authors simply use feedback to manage the buffer at the sender. In [5, 1, 6], different authors study the decoding delay of

The paper was presented at APRES 2015.

online coding with feedback for the multiple receiver case. One common assumption in these works is the availability of perfect feedback, which is not feasible in wireless communications, especially in multicast scenarios. Periodic feedback is considered in [4], in terms of the asymptotic behavior of throughput and decoding delay, and analyzed for a single client. The metric of choice considered is the in-order delivery delay, which seems a much more suitable metric for multimedia streaming applications. Still, as most works, the analysis is based on average values. For more stringent requirements, such as those found in real-time applications, the second-order moment of the decoding delay should be also taken into account. A practical solution to massive multicast streaming with online network coding has been proposed in [3]. However, the proposed coding mechanism only provides a best-effort solution with no formal guarantees. This is the basis for the coding approach that we follow in this work, where we aim at providing stochastic guarantees of worst-case coding delay under realistic feedback settings.

3. SYSTEM MODEL

3.1 Network

We consider a streaming server that is in charge of multicasting a video to a group of N clients through a wireless medium, where the link between the server and each client i is modeled as a binary erasure channel with packet erasure probability e_i. We assume that time is organized in slots where each time unit corresponds to the time needed to transmit one packet, hence the server transmits one packet per slot. Feedback is periodically requested from the clients in multiples of T slots, where T can be regarded as the minimum feedback period. Hence, communication is organized in rounds of T packet transmissions, at the end of which feedback can be received.

3.2 Coding

We assume that network coding is performed at the server in an online fashion. An online encoder takes packets as they become available in the buffer and generates coded packets that are linear combinations of these original packets. Hence, a coded packet takes the form $c_j = \sum_i \alpha_{ij} p_i$, where the α_{ij} are coding coefficients. These coefficients are randomly picked from a chosen finite field and appended to the encoded packet. Coded packets can then be seen by a receiver as equations in a linear system, where each coded packet delivers a new degree of freedom, provided that it is linearly independent from all previously received packets, in which case it is said to be innovative.

Originally, the online coding mechanism does not restrict which packets can be mixed together. This means that new packets can be added indefinitely, potentially leading to high decoding delay. To mitigate this issue, we consider that the original data is partitioned in blocks of G packets called generations, such that packets can only be mixed with other packets in the same generation.

Since coded packets are a combination of multiple packets, a key question is how to represent the state of each client and what information should acknowledgments convey. This question has been answered in [7] and it relies on the definition of *seen packet* which we present in Definition 1.

DEFINITION 1 (SEEN PACKET). *A client is said to have seen packet p_k if it is able to compute a combination that only includes packet p_k and packets p_i with $i < k$, from all information it has received so far.*

Once all clients have seen a packet, the server no longer needs to include that packet in further coded packets, and thus clients need only to acknowledge seen packets. This particular definition of seen packet (taken from [6]) allows to easily establish a sufficient condition for a packet being decoded, which is presented in Lemma 2.

LEMMA 2 (DECODABILITY). *Under the terminology of Definition 1, if all packets $p_1, \ldots, p_k$ up to some k have been seen, then they have also been decoded.*

The next step is to define a coding scheme that describes how each coded packet is generated. Full-rank schemes always combine all packets available for a generation, achieving optimal throughput but poor decoding delay performance. Instead, we base our analysis on the time-invariant schemes presented in [4] which we describe in Definition 3.

DEFINITION 3 (TIME-INVARIANT SCHEME). *This coding scheme is characterized by a vector $x = [x_1 \cdots x_T]$ where x_i, for $1 \leq i \leq T$, are non-negative integers satisfying $\sum_i x_i = T$. In each round we transmit x_i independent linear combinations of the first i unseen packets in a generation.*

These coding schemes define how to generate the T coded packets for each round. Moreover, the maximum number of packets combined is T, which makes sense from a decoding delay perspective since no more than T packets can be delivered, and thus decoded, in each round. These schemes incorporate feedback in each round, by coding the first unseen packets reported. The design task consists on choosing the values composing x. In our analysis, we shall consider two instances of such schemes denoted as x_A and x_B and specified as

$$x_A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$
$$x_B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 \end{bmatrix}.$$

Comparing the structure of both codes, we see that x_A can potentially deliver 10 innovative packets in each round, whereas x_B can deliver at most 9. However, in average terms code x_B generates combinations with more packets than x_A. The intrinsic properties of each code shall induce different throughput and delay profiles which we seek to analyze.

3.3 Architecture

We now present the system architecture whose goal is twofold: manage feedback among many clients and recover lost packets in a timely fashion. The architecture of the proposed system is depicted in Fig. 1.

In the server side, video packets arriving from the application layer are assigned a unique sequence number and stored in the input buffer. An encoder then generates coded packets from the original packets available in the buffer in an online fashion. The scheduler receives the stream of coded packets, interleaves feedback request packets with periodicity T, and queues packets for transmission. Each feedback request can poll many clients. The scheduling of clients to be polled depends on their individual polling periods, i.e., T_i slots for client i with T_i being a multiple of T. The different periods T_i can be adjusted depending on the erasure

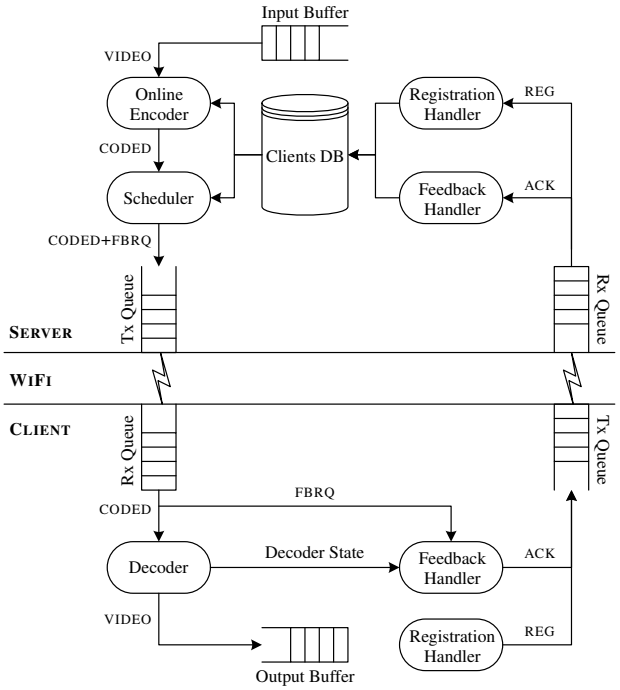

Figure 1: Architecture of the proposed system showing the main blocks in the server and client sides. Different types of packets are considered, namely video packets (VIDEO), coded packets (CODED), feedback request packets (FBRQ), acknowledgment packets (ACK) and registration packets (REG).

rate e_i and service level agreement SLA_i of client i. Moreover, the scheduler limits the maximum number of clients that can be included in a single poll, say to M, in order to limit the maximum bandwidth allowed for feedback (M acknowledgments every T). Within this constraint, the scheduler dynamically assigns this bandwidth among all clients, shifting feedback bandwidth from clients with low erasure levels to others with higher erasure levels. On top of that, the scheduler controls the relative phase of clients feedback transmissions (acknowledgments) to reduce collisions. Information about the clients is stored in a database which holds the registered clients together with respective SLA, erasure rate and missing packets. The database is updated whenever a client registration packet or acknowledgment arrives at the server.

On the client side, received coded packets are passed on to the decoder, which keeps track of seen packets. Whenever a client receives a feedback request, a feedback handler checks the decoder status and reports seen packets to the server in a short unicast acknowledgment packet. Whenever a new packet is decoded, it is placed in the streaming buffer to be consumed by the video application.

Choosing the design parameters for this system is a challenging task. On the feedback management side, proper polling frequencies must be found. Higher frequencies allow faster notification and recovery of missing packets, whereas lower frequencies imply lower overhead but less opportunities to recover. On the data recovery side, erasure codes are a bandwidth efficient alternative, albeit with an added cost associated with decoding delay. On this aspect, the coding should be designed in a way to strike a balance between throughput and decoding delay, which is the main focus of this work. For this reason, in the following we consider that $T_i = T$ for all clients and neglect the time taken by feedback requests and acknowledgments.

4. SIMULATION RESULTS

In this section we present preliminary results showing the evolution over time of the number of packets delivered in-order to the clients. In our experiments we considered $N = 100$ clients with packet erasure rates e_i randomly picked from an uniform distribution ranging from 10% to 40%. Each transmission round has a length of $T = 10$ slots, where one coded packet is multicast in each slot and after which feedback is received, acknowledging seen packets. The results are for a single generation of $G = 64$ packets. For each experiment we performed 1000 runs and present the results for the worst client, i.e. the client with highest packet erasure rate.

The results are shown in Fig. 2 and Fig. 3 for codes x_A and x_B respectively. The filled regions correspond to Tukey boxplots with the red, blue and green regions representing the median, boxes and whiskers, respectively. The outliers have been omitted for clearness. Since the whiskers encompass a very high percentage of the samples, we focus on their lower limit which provides a worst case scenario with high probability. The dashed line represents the requirements of the client application. We consider that the video client application consumes packets with a rate of $R = 0.3$ packets per slot. For this fixed rate, we compare the initial buffer B that is required to allow playback without interruption with very high probability, which is given by the interception of the dashed line with the Y-axis.

Looking at the results, one can observe that x_B provides better average goodput than x_A which implies that it supports better video quality. In turn, x_A presents less variation in the number of packets decoded in each instant. As a result, with code x_A the worst client needs to buffer $B = 6$ packets, whereas with code x_B the same client must buffer $B = 9$ packets. This gap may not seem significant but as we increase the feedback period to more realistic values, or increase the target video quality this difference is expected to grow.

5. DYNAMIC ADAPTATIONS

The system that we envision has several degrees of dynamic adaptation. For example, as referred in [2], the feedback requested to each client can be adapted dynamically so that clients with higher erasure rates are polled more frequently, i.e., T_i is an inverse function of e_i. This allows the server to better track more dynamic erasure processes.

Another degree of adaptation is the dynamic adjustment of the code structure in response to varying network conditions and system requirements. In fact, the time-invariant schemes used here can be used as a base for adaptive schemes whose code structure depends, for instance, on the erasure rates at the beginning of each round. This adaptation can be naturally achieved by using the feedback obtained from different receivers in order to determine the number of successfully received packets out of a total number of transmissions through the network, and thus the observed e_i for each

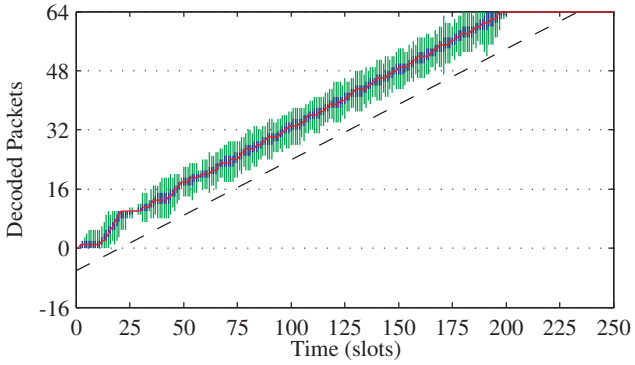

Figure 2: Evolution over time of the number of packets delivered in order when using code x_A. The simulation parameters are $N = 100$, $T = 10$, $G = 64$ and $e_i \sim U(0.1, 0.4)$. The dashed line represents the video client application requirements. For a packet consumption rate $R = 0.3$ the initial buffering required is $B = 6$.

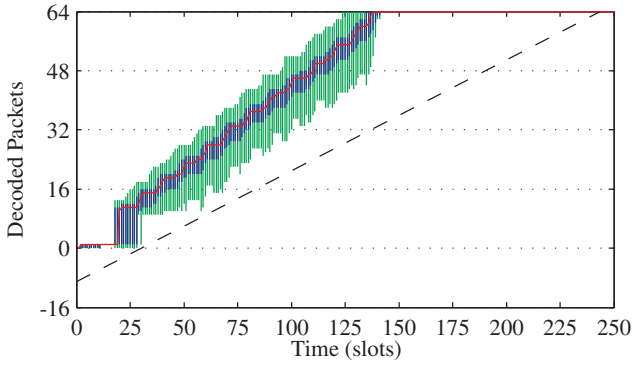

Figure 3: Evolution over time of the number of packets delivered in order when using code x_B. The simulation parameters are $N = 100$, $T = 10$, $G = 64$ and $e_i \sim U(0.1, 0.4)$. The dashed line represents the video client application requirements. For a packet consumption rate $R = 0.3$ the initial buffering required is $B = 9$.

receiver i.

Finally, it is equally relevant to analyze the interplay between the scheduling adaptations of the feedback information and the dynamic adjustment of the code structure to check whether there is any negative mutual interference.

6. CONCLUSION AND ONGOING WORK

In this paper, we discussed streaming for many clients. Given the necessary multicast nature of the transmissions, there is a need to mitigate packet losses since multicast transmissions are inherently unreliable. In previous works, such unreliability was mitigated with erasure codes with feedback or with retransmissions management. In this paper we proposed merging both techniques while using a coding configuration that provides novel stochastic guarantees of worst-case coding delay, a very significant feature for delay-sensitive applications. Preliminary simulation results confirm the correctness of the analysis. We concluded the paper highlighting two dimensions that can be used to grant dynamic adaptation and so improve performance, namely at the level of the feedback scheduler and the coding schemes.

As future work we intend to expand our analysis by moving towards more realistic settings. This includes considering larger feedback periods, incorporating the time needed to request and receive feedback, and performing adaptation under varying network conditions. Ultimately, we expect to perform real-world experiments to validate our ideas.

7. ACKNOWLEDGMENTS

This work was partially funded by Fundação para a Ciência e a Tecnologia under grant SFRH/BD/81726/2011 as well as the FCT projects PTDC/EEI-TEL/3006/2012 and UID/EEA/50008/2013. This work was also financed by the Swedish Foundation for Strategic Research (SSF), and by the Green Mobile Cloud project (Grant No. DFF-0602-01372B) granted by the Danish Council for Independent Research.

8. REFERENCES

[1] J. Barros, R. A. Costa, D. Munaretto, and J. Widmer. Effective delay control in online network coding. In *Proc. IEEE INFOCOM*, pages 208–216. IEEE, Apr. 2009.

[2] J. Cano and L. Almeida. Feedback management for scaling clients in streaming multicast. In *Proc. ACM Symposium On Applied Computing*, Apr. 2015. accepted for publication.

[3] D. Ferreira, R. A. Costa, and J. Barros. Real-Time network coding for live streaming in hyper-dense WiFi spaces. *IEEE Journal on Selected Areas in Communications*, 32(4):773–781, Apr. 2014.

[4] G. Joshi, Y. Kochman, and G. W. Wornell. The effect of block-wise feedback on the throughput-delay trade-off in streaming. In *IEEE INFOCOM Workshops*, pages 227–232. IEEE, Apr. 2014.

[5] L. Keller, E. Drinea, and C. Fragouli. Online Broadcasting with Network Coding. In *Proc. Workshop on Network Coding, Theory and Applications*, pages 1–6. IEEE, Jan. 2008.

[6] J. K. Sundararajan, P. Sadeghi, and M. Médard. A feedback-based adaptive broadcast coding scheme for reducing in-order delivery delay. In *Proc. Workshop on Network Coding, Theory, and Applications*, pages 1–6. IEEE, June 2009.

[7] J. K. Sundararajan, D. Shah, and M. Médard. ARQ for network coding. In *Proc. IEEE International Symposium on Information Theory*, pages 1651–1655. IEEE, July 2008.