# Towards Dynamic Adaptation in Broadcasting with Hybrid Rateless Codes

Carlos Faneca, José Vieira, André Zúquete
IEETA
Dep. Electronics, Telecom. and Informatics
University of Aveiro
Aveiro, Portugal
{carlos.faneca,jnvieira,andre.zuquete}@ua.pt

Julio Cano, André Moreira, Luis Almeida
Instituto de Telecomunicações
Dep. Electrical and Computer Engineering
Faculty of Engineering, University of Porto
Porto, Portugal
{jcano,andre.moreira,lda}@fe.up.pt

## ABSTRACT

There are many situations, such as in training and education, in which there is a frequent need to distribute large files to many clients, e.g., operating system boot images or raw data files. To carry out such distribution efficiently, we use wireless broadcast and a hybrid coding technique that combines forward coding using weak LT Codes with a feedback phase at the end, which allows concluding the process faster with lower computing cost than traditional LT codes. However, for the sake of scalability, in the feedback phase a scheduler bounds the maximum number of clients that can communicate feedback in each given cycle and schedules them. Moreover, using a shorter or longer feedback phase also impacts on the number of simultaneous clients in feedback mode, resulting in more or less impact of the scheduler and more or less effectiveness of the feedback itself. In this short paper we briefly describe a recently developed prototype and we summarize some preliminary results confirming the advantage of using scheduled feedback. Moreover, we discuss the interplay between duration of the feedback phase and clients scheduling as a line for future research in scheduling-coding co-design.

## Keywords

Wireless broadcast; Ad-hoc networks; Fountain codes; LT codes; Weak-LT codes; Traffic scheduling

## 1. INTRODUCTION

Distributing large files to a large number of clients using a wireless network is a challenging task. However, it is a typical requirement in educational and training scenarios where operating system boot image files or raw data files need to be distributed among the participants, or even in sensor networks where a similar code image has to be uploaded to many nodes. Scenarios of this kind, with large amounts of information disseminated among many clients in microcells will probably become more common as new networking paradigms, such as 5G networks, settle in.

Fountain Codes [1, 4], also called *rateless* such as LT [3, 4] codes, are an efficient way to preform this task. Using a properly designed degree distribution, it is possible to construct practical rateless codes such that the original data can be recovered from a number of codewords that is only slightly larger than the number of original symbols. As the information in the receiver grows, so does the probability that a new random packet is not informative, and thus useless from a code perspective. This suggests that the last part of the transfer can be enhanced with a feedback strategy in which the clients inform the server of the specific symbols they miss [2]. The encoder then gives preference to such symbols in the generation of the following codewords.

An application similar to ours has been considered in [2], where a feedback aided LT code is used to deploy a program (file) to multiple sensor nodes via wireless broadcast. Apart from code construction, one concern is the amount and location of feedback. To avoid congestion, one of the methods in [2] restricts acknowledgements to be uniformly distributed across the transmission, at the cost of some symbol overhead when compared to the non-uniform feedback case. To cope with multiple receivers, the scheme considers the worst case receiver. This inhibits the benefits for the majority of the clients, in case one receiver is much worse than the others or simply new clients join in the middle of the transmission.

Recently we proposed using a scheduler in the server that controls the transmission of the feedback packets to avoid this problem. The scheduler explicitly polls a bounded number of clients that are in feedback mode, e.g., three, according to an adequate scheduling criterion, e.g., those that miss less symbols. The use of a feedback phase also grants the server control over the transfer process to avoid producing code words when there is no client ready, thus alleviating the use of the wireless medium. However, there is an interplay between the duration of the feedback phase and the scheduler because the number of clients simultaneously in feedback mode is highly dependent on such duration.

In this short paper we present a summary of preliminary results that confirm the advantage of using our scheduled feedback approach and we discuss the use of different scheduling approaches as well as the interplay between the scheduler and the feedback duration, paving the way for future research along this line

## 2. CODING FRAMEWORK

In our approach, we consider LT Codes [3] with a lower degree distribution, which we call Weak-LT Codes. These are substantially simpler to compute but present an extra

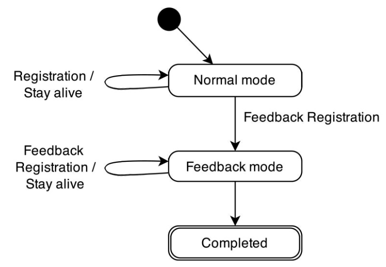

**Figure 1: Client state machine.**

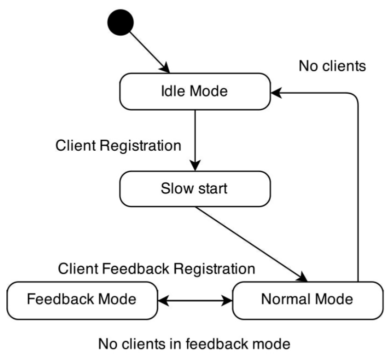

**Figure 2: Server state machine.**

overhead in codewords that we compensate with the use of feedback. According to this let us define a few variants that we shall use in our analysis. For Classic LT Codes, we use the robust soliton distribution with $c = 0.03$ and $\delta = 0.5$. For the Weak-LT Codes we further define two types. The Weak-LT (Type I) consists on the same distribution of the Classic LT but truncated to a maximum degree of 8. Combined with 20% remaining symbols communicated through feedback this approach aims at minimizing the mean and maximum codewords overhead. The Weak-LT (Type II) uses $c = 0.05$ and a maximum degree of 20. Combined with 2% feedback, this approach reduces the time clients spend in feedback mode.

When the server handles a client in *feedback mode* it starts to include in each codeword a missing symbol chosen randomly from the clients' feedback list, see Section 3, while maintaining the distribution degree, choosing the remaining symbols from the ones that are not in the feedback list. This way, the coding distribution is kept intact for the remaining clients that are receiving codewords at the same time. Due to the low degree distribution, the server can satisfy just a low number of clients simultaneously in feedback mode (three in the current prototype) and indexes of the symbols of the feedback list are explicitly sent on the broadcast packet.

## 3. SYSTEM ARCHITECTURE AND FEEDBACK PROTOCOL

We consider an ad-hoc network with one server storing and disseminating a file among several clients. The server and clients are configured to connect to the same SSID in ad-hoc mode, so that the server communicates directly to the clients. This reduces latency and eliminates distribution problems that the use of an Access Point in infrastructure mode could generate. The identification of all nodes in the system is carried out using their respective MAC addresses.

The protocol state-machine of the client side is shown in Fig. 1. The client starts in *normal mode* in which it receives codewords and periodically registers in the server to signal its presence. When only a few symbols are missing the client enters the *feedback mode.* In this state, each client registers in the server as a feedback mode client, so that the server considers this client when scheduling the feedback requests. When requested for feedback the client sends a unicast packet with a list of the symbols it needs to complete the download. When the download is completed the client sends a message to the server indicating so, and finishes.

The protocol state-machine of the server side is shown in Fig. 1. The server starts in *idle mode* where it simply sends periodic requests for registrations. It does not broadcast codewords to prevent flooding of the wireless channel while no client is present. When it receives the registration of at least one client it changes to the *slow start mode.* This mode is designed to receive a large amount of initial registrations. In this state, many clients are expected to initiate the download process simultaneously. This mode reserves a time slot to receive many registrations from clients. This time slot is progressively reduced to a minimum value, at which point the server changes to *normal mode.* In *normal mode* the server uses most of the time to broadcast codewords, while periodically requesting client registrations. Existing clients will still send their registration messages to the server to signal their presence. In case a client does not renew its registration, the server deletes it from its database. When no client is active the server goes back to *idle mode*, waiting for new clients.

While in *normal mode*, if at least one client registers in *feedback mode* the server enters its own *feedback mode.* In this state, the server additionally sends periodic feedback requests to clients in *feedback mode.* In each broadcast request, a limited number of clients is polled for feedback, to bound the bandwidth used by the feedback. The server is notified when a client completes the download. When no clients remains in *feedback mode*, the server changes back to *normal mode.*

## 4. PRELIMINARY RESULTS

In this section we show some preliminary results from simulations with one client, to verify the impact of the feedback approach. Then we show the results of practical experiments with a test bed of 10 clients. These results motivate us for the dynamic adaptation approaches we wish to research next and which we discuss in the following section.

### 4.1 Simulations

We implemented our coding framework for one client in MATLAB to analyze the behavior of the coding process with feedback and help tuning it. In particular, we simulated the decoding of a file with 10 000 symbols, using the Weak-LT Codes of Type-I (20% feedback) and of Type-II (2% feedback) as defined in Section 2, plus traditional LT codes for comparison. Figure 3 shows the distributions of the number of codewords needed to decode the complete file with

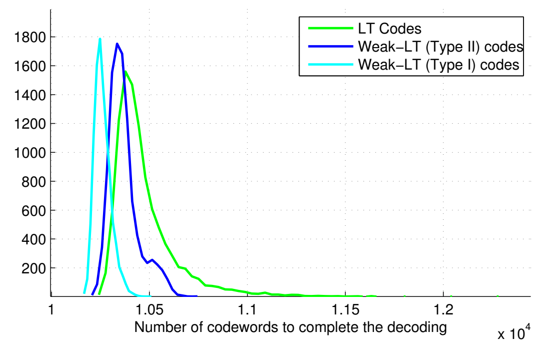

**Figure 3: Histogram of the number of codewords needed to decode a file with 10 000 symbols.**

the three types of coding when repeating the transfer 10 000 times.

As expected, the earlier the feedback starts, in this case, the Type I distribution is applied when just 20% of the symbols are missing, the less codewords are needed to complete the file transfer which ends sooner. Nevertheless, even with a very short feedback phase, triggered when just 2% of the symbols are missing we already have advantage in codewords overhead with respect to the traditional LT codes.

## 4.2 Practical Experiments

To see if these results also occurred in practice with more clients, we deployed a test bed with 10 clients scattered in a classroom with 38.5 $m^2$ and with a noisy Wi-Fi environment.

The server was a laptop running in single user mode to distribute a 100 MBytes file using a Thomson TG123g (chipset Realtek RTL8187B) Wi-Fi dongle interface. This Wi-Fi interface was chosen because it is able to broadcast at 54 Mbit/s, the maximum transmission speed of IEEE 802.11g and it is able to explore the TxOP feature of the QoS enhancements of IEEE 802.11, which increases significantly the transmission efficiency in unidirectional frame bursts.

The clients ran in ordinary laptops without any extra equipment. However, we built a USB stick with a bootstrap system that automatically joined the ad-hoc network created by the server and downloaded the file 100 times. For each download, a log was created containing the percentage of missed packets, the total codewords needed to decode the file, several time figures (elapsed time, time in feedback, user CPU time and system CPU time) and the maximum amount of memory used by the decoding process.

One aspect that we realized was that using more feedback caused an effective reduction of global throughput due to the extra contention in the wireless medium between communications from the clients and the server. This effect, however, is independent of the coding framework and depends on the efficiency of the QoS mechanisms of IEEE 802.11, only. Thus, to remove this effect we carried out two experiments, each at a different server throughput.

### *4.2.1 Server broadcasting at a limited throughput*

The first experiment considered a server broadcasting throughput of 20 Mbit/s and was used to compare LT with weak LT Type I codes. This throughput not only accommodated more feedback packets but also other spureous communica-

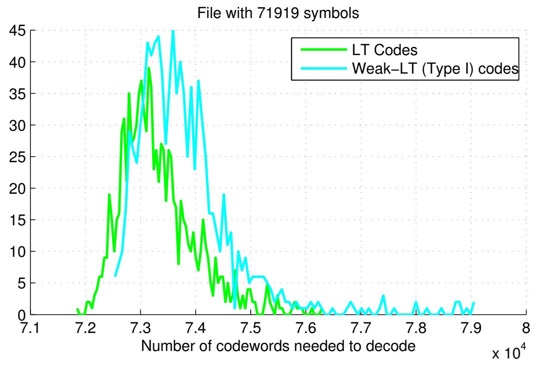

**Figure 4: Histogram of the number of codewords needed to decode a file with 10 clients and limited throughput.**

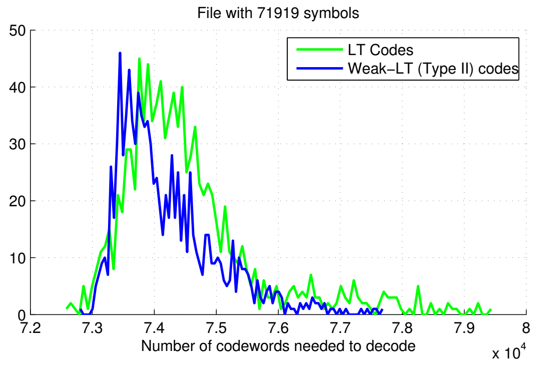

**Figure 5: Histogram of the number of codewords needed to decode a file using all measurements obtained by the 10 clients on the classroom.**

tions that could need to use the channel.

We took 100 independent measurements on each of the ten clients for broadcasting a file with 100 MB. The results obtained are presented in Fig. 4. Surprisingly, the number of codewords needed to decode the file was higher with weak LT codes, as opposed to what we had observed with a single client in simulation.

### *4.2.2 Server broadcasting at maximum throughput*

In this scenario we followed the same approach and took another 100 independent measurements on each of the ten clients but with the server transmitting at full speed (effective throughput close to 43Mbit/s). In this case, we used Weak-LT codes of Type II Code and again the classic LT codes. The obtained results are presented in Fig. 5. In this case, as expected from simulation the codewords overhead was in fact lower with the feedback approach, used for the last 2% of the symbols, only, when compared to the classic LT codes.

## 5. DYNAMIC ADAPTION AND SCHEDULING IN CODING

The preliminary results we achieved in practice generally confirmed the value of using feedback with weak LT codes to reduce the number of codewords needed to decode a file. However, they also showed some inconsistency between the situation with one client and many clients. In fact, when we used a longer feedback phase with multiple clients, we increased the number of clients simultaneously in feedback mode. Since the scheduling approach limits the number of clients that can be polled in each scheduler cycle to bound the associated bandwidth, when there are many clients beyond those that can be polled, several clients may have to wait until they transmit their feedback to the server.

### 5.1 Adapting the feedback strategy

This situation opens the possibility to use an adaptive technique that change the coding strategy online according to the number of clients simultaneously in the feedback phase. For example, when the first few clients connect to the server, it can use a Weak-LT distribution with a small average degree, say 3 similarly to the Type I referred before. This corresponds to a relatively large amount of feedback, around 20% of the symbols, leading to very small codewords overhead and thus to a quicker transfer.

However, when the number of clients in feedback mode grows beyond a certain number, e.g. 3, the server can change the average degree of the coding distribution in order to reduce the amount of feedback needed from each client. For example, it would change from Type I to Type II codes. If the number of clients in feedback mode grows even more, above a higher threshold, the server can end up using a plaint LT code avoiding this way the need of feedback.

This adaptive mechanism makes the best out of feedback in each concrete situation and should lead to minimizing the codewords overhead across a large range of numbers of clients. There are, still, several aspects to determine, such as the best codes and associated thresholds in number of clients.

### 5.2 Choosing a scheduling criterion

Another aspect that remains open is the specific scheduling strategy used during the feedback phase. Currently, the scheduler picks the clients that have the smallest number of missing symbols in an attempt to favor those that are closer to decode the whole file. However, other criteria can be used, such as random, time-driven or priority-driven or hybrid.

In a time-driven approach, the scheduler would favor, for example, the clients that have been waiting for a longer time. Alternatively, we could even have clients with different urgency, expressed as different deadlines, and schedule first the clients with earlier deadlines. This is a well-known criterion in real-time scheduling that maximizes meeting deadlines in an error-free scenario.

In a priority-based approach, the scheduler would pick the clients with higher priority. This could be used to create classes of service and favor the clients of the best service class first. Within each class, another criterion could be used, such as time-driven, resulting in a hybrid approach.

The specific impact of different scheduling policies on the coding process remains largely unexplored. However, when using an adaptive technique such as the one referred in Section 5.1, there will be less clients simultaneously in feedback mode, and thus the scheduler impact is expected to be small.

Finally, whenever a specific client scheduled for feedback informs the server of its missing symbols, we still need to decide which symbols to pick since only a small number can be encoded in low degree codewords. This is another scheduling problem but applied to the symbols within the codeword. The current approach is to choose randomly. However, other approaches are possible, such as analyzing all the feedback information the server has and picking the symbols that satisfy the largest number of clients, or those that satisfy the clients that are closer to finishing the file decoding. Again, these policies applied at this level are still largely unexplored and researching them seems an interesting direction.

## 6. CONCLUSION

In this work we addressed the case of disseminating a large file for many clients using a broadcast approach enhanced with rateless codes for reliability purposes. We showed that it is possible to use coding with low degree distributions combined with a feedback phase to reduce the code words overhead. In order to manage the feedback phase we developed a protocol that allows the server to be aware of the transfer process in the clients and thus stop the dissemination of codewords when no more clients are active.

We then characterized the process in simulation with one client and distributions of different degrees implying different amount of feedback. The results confirmed the expected benefits in reduction of codewords overhead. We then carried out practical experiments on a test bed with 10 clients. For small amount of feedback the resulted matched those of the simulation. However, with larger feedback, we obtained different results.

From these discrepancies we discussed possible causes and we drew the principles for an adaptive coding approach that uses different degree distributions according to the current number of clients in feedback mode. We also discussed the potential impact of different scheduler criteria for scheduling clients in feedback and pointed out to scheduling-coding codesign issues that seem to remain largely unexplored and which we plan to research in the sequel of this work.

## 7. ACKNOWLEDGEMENTS

This research is partly funded by the Portuguese Government through FCT grants *CodeStream - PTDC/EEI-TEL/3006/2012* as well as *PEst-OE/EEI/UI0127/2014* and *UID/EEA/50008/2013*.

## 8. REFERENCES

[1] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *ACM SIGCOMM Comput. Commun. Rev*, volume 28, pages 56–67. ACM, Oct. 1998.

[2] A. Hagedorn, S. Agarwal, D. Starobinski, and A. Trachtenberg. Rateless Coding with Feedback. In *IEEE INFOCOM 2009*, pages 1791–1799, Apr. 2009.

[3] M. Luby. LT codes. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 271–280, 2002.

[4] D. MacKay. Fountain codes. *IEE Proceedings-Communications*, 152(6):1062–1068, Dec. 2005.