IMPROVING OBJECT DELIVERY USING APPLICATION LAYER TORNADO CODES IN DVB-H SYSTEMS

Kristian Nybom Janne Kempe Jerker Björkqvist
Department of Computer Science, Faculty of Technology,
Åbo Akademi University
Turku, Finland

ABSTRACT

This paper analyzes the effects of applying application layer Forward Error Correcting (FEC) coding to a wireless filecasting system, here represented by DVB-H. We show the benefit of moving error correcting coding to higher levels in the data transmission protocol stack. The data is delivered using a data carousel and the FEC code utilized is a hyper-Tornado code for which the main characteristics are long codewords and near linear complexity. Because the hyper-Tornado code provides internal data interleaving and supports long code lengths, we can observe a significantly decreased number of mean data carousel iterations before an error-free carousel object is received compared to the basic DVB-H system. This reduction is for the user observed as decreased download-time, but also provides energy saving in the receiver due to shorter receiver on-times.

I. INTRODUCTION

Wireless digital broadcasting technologies are evolving into the mobile convergence market. Technologies such as DVB-H [1] are used for streaming digital video and audio to mobile handsets. Aside from audiovisual streaming services, other file-type services can be offered utilizing the same technology. Such file download services may contain for instance video clips, documents and applications. The quality of service for a file downloading service can be measured by the time difference between the point in time when download is issued and the point when the file becomes usable. For file delivery services, the requirement is usually that the entire object must be present without a single error before it is useful to the end-user or the terminal system. Applying this criterion to unidirectional transmission, where retransmission requests are generally not possible, the only possibilities to receive the missing pieces from a file are to use strong enough forward error correcting codes (FEC), to wait for the recurring transmission from the file carousel, or to use a combination of these two. Apart from the user experience of the download time, energy consumption of the mobile terminal is also an important issue. If the downloading system is used efficiently it can in poor reception conditions provide robust protection against transmission errors, and in good reception conditions minimize the downloading time and the receiver energy consumption.

The main contribution of this paper is to show by simulations that by using an erasure correcting code, with source block length matched to the size of the transmitted file, it is beneficial in DVB-H file downloading services to move the error correction coding up from lower layers (mainly the link layer MPE-FEC in this paper) to the application layer. The gain from this arrangement is that the error correction capability can be utilized more efficiently. Additionally, in good reception conditions the terminal may choose to ignore redundant and already received data and therefore save energy by shutting down the receiver. In [2] similar indications have been shown by using a Raptor [3] code as application layer FEC. Raptor codes are standardized as application layer FEC in DVB-H, but the Hyper Tornado code is investigated here because of its low complexity and good error correcting capability.

II. SYSTEM OVERVIEW

The object downloading system proposed in this paper operates at the transport level above the IP and UDP protocols used for broadcasting the data. Since DVB-H is used here as the broadcasting technology, IP packets are encapsulated into Multiprotocol Encapsulation (MPE) sections protected with a CRC-32 checksum. If the CRC-32 check fails, the section data is discarded. The DVB-H optional MPE-FEC is omitted from this system. The UDP datagram payload contains exactly one object fragment (OF) consisting of an object delivery protocol (ODP) header and one code symbol. This protocol is designed to provide recognition of individual object fragments and their data offset inside a specific object. The size of one OF coincides with the payload size of one UDP datagram, which means that IP packets containing OFs may be received in arbitrary order. This allows arbitrary interleaving of IP packets at the transmitting side. DVB-H specifies the use the FLUTE protocol for file delivery, but the ODP approach was chosen for its simplicity.

The ODP header contains the following fields: uncoded file size, encoded file size, data offset, object ID and symbol size. The encoded file size gives the receiver the total memory size to allocate upon reception of the first OF. The data offset field describes the byte offset of the received data inside the object. The data offset is used for finding the position of the OF within the object. The uncoded file size field is only used for determining the zero padding needed in the last code symbol. The object ID field provides a unique identifier for an object. It also carries information about which FEC is used and contain a 1-bit flag indicating whether the OF is a message symbol or a repair symbol. For simplicity, object sizes and FEC information is transmitted inside every OF, but in a real system this information would naturally be transmitted for instance in a service announcement header. This would result in less protocol header overhead.

As shown in Fig. 1, a file is encoded using the application layer FEC into a transport object, a concept used also in ALC/LCT [4]. The transport object is packetized into ODP/UDP/IPv6 packets, which are interleaved using a block interleaver, specified in Section V. This interleaver serves two purposes: distribution of long transmission burst errors over the entire object, and distribution of the repair symbols in-between the source symbols.

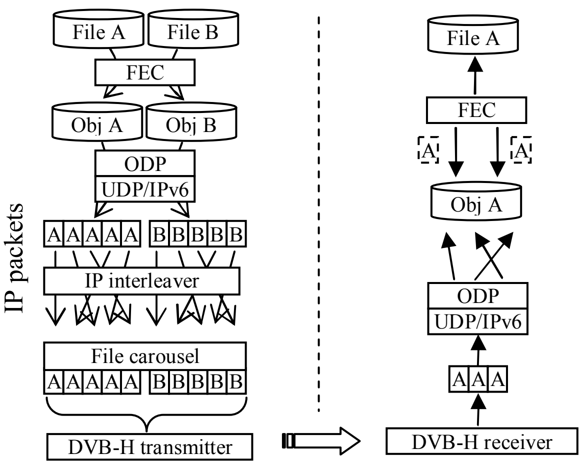

Figure 1: Block diagram over the system components for both transmitter and receiver.

Objects are transmitted using a file carousel model. The download application identifies the desired object using the object ID field and allocates memory according to the encoded object size field. The receiver application reconstructs the object by receiving object fragments and by invoking the FEC to calculate missing fragments.

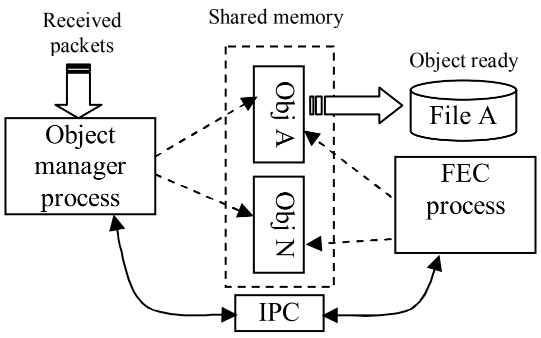

Figure 2: Receiver application.

The receiver application, shown in Fig. 2, is realized using two processes working in a shared memory area. The ODP receives fragments from the UDP endpoint and inserts them into the correct position in shared memory. Using IPC the FEC is invoked to work on the received data according to the FEC invocation policy. This allows the FEC to be executed only when certain conditions are met, e.g. when a certain amount of data has arrived. The aim is not only to correct data that is missing due to transmission errors, but also to reconstruct, whenever possible, any missing coding symbols before they are actually received.

III. Error Correcting Coding

Tornado codes [5, 6, 7] are efficient erasure correcting codes, suitable for multicasting of bulk data. The Tornado code consists of a set of symbols containing the message data, followed by several sets of symbols, in a cascading manner, containing repair data. The size of the symbols can be chosen arbitrarily, but typically they are chosen to equal the IP packet payload. The code is an exclusive-OR based erasure correcting code, which error correcting performance is critically dependent on the dependencies between the nodes. In [6] an analysis of the Tornado code structure is given.

Tornado codes have the property that they require only a small fraction of the repair data, in order to successfully reconstruct the missing message nodes. The code is, however, quite vulnerable to burst errors and without interleaving mechanisms, the decoding process is prone to fail for higher error rates. When designed as a hyper code [8], the Tornado code has several dimensions, where each dimension is a high code rate Tornado code with the repair symbols calculated on permutations of the message symbols. This design methodology results in a better resistance against burst errors, compared to standard Tornado codes.

Tornado codes have the property that they can detect when enough data has arrived in order to reconstruct the entire object. Moreover, the codes work as real-time codes, being able to use data as soon as it is received and, hence, detecting when enough data has arrived in order to reconstruct the object.

IV. Channel Approximation with a Two-State Markov Model

Wireless channels have an unreliable and error-prone nature that seriously affects the quality of transmitted packets. To model the time-varying fading and interference conditions that affect the channel, a two-state Markov model is used [9]. This model introduces bursts of errors into a stream of packets. The two-state Markov model is illustrated in Fig. 3.

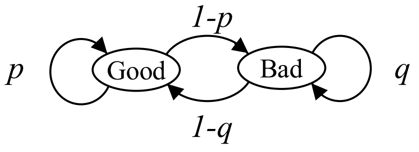

Figure 3: A two-state Markov model.

In Fig. 3, p is the probability of successfully transmitting a packet given the previous packet was successfully transmitted. Likewise, q is the probability of losing a packet given the previous packet was lost.

It should be noted that the justification for using this model is not, in any way, to accurately model a mobile channel, but rather to obtain an error pattern resembling the bursty nature of such a channel. This approach has also been used in [10].

V. Simulations and Test Results

The test system was setup on a 2 GHz Pentium 4 running Linux. IP packet erasures were introduced into the transmission channel by using a two-state Markov model, with an average burst length of 5 IP packets. The tests were run for IP packet error rates (IP PER) from 0%-80%. For every IP PER up to 25%, 1000 downloads were simulated.

For the remaining IP PERs, only 10 downloads were simulated, simply to obtain a trend of the performance of the system.

As object data, a 4.308 MB file containing random data was used. This file size matched perfectly for a 5-dimensional (4000, 3000) Tornado code, as described in [8], with symbol sizes equal to 1436 bytes. This code had a code rate of ¾ and thereby, the transmitted object consisted of one block. The symbol size was chosen so that 1500 byte IP datagrams were obtained. The encoded object was transmitted both as interleaved, using a block interleaver with the dimensions 63*64, and uninterleaved.

The Tornado code was set to start decoding as soon as 50% of the object was received in order to assure that the decoder would reconstruct the transmitted object as early as possible, thereby minimizing the terminals receiver on-time. The FEC invocation policy should be optimized with regard to power consumption, taking into account the CPU-load of the decoding algorithm.

In the same manner, an MPE-FEC was tested, using 256, 512 and 1024 rows and IP datagrams of 1500 bytes. In order to obtain comparable results, the code rate of the MPE-FEC was chosen to ¾. The code length of the Reed-Solomon code used in each MPE-FEC frame is considerably shorter than the code length of the Tornado code. Therefore, a single Tornado code block encapsulates the object whereas several MPE-FEC frames are required for encapsulation of the file data. This gives a comparison between different system setups, both employing same amount of data.

In Fig. 4, 5, and 6, a concept of carousel rounds needed for complete download of the source file is used. This comparison is fare for the Tornado and MPE-FEC results because they incorporate the same amount of data. In Fig. 5 and 6 an uncoded object is also included. For the uncoded file there is 25% less data in the transmission object due to absence of the repair data. Even if the uncoded object transmission contains less data than the encoded, the comparison gives an idea of the impact of the encoding. Consider the case where the transmission time of the uncoded source file t_u is significantly lower than the total carousel round time t_C ($t_u << t_C$) due to many other files in the carousel. If the first reception of the object was unsuccessful the receiver has to wait $t_{wait}=t_C-t_u$ before the second transmission starts. This waiting time t_{wait} is equal also for the case of the encoded object in the same data carousel. This time t_{wait} is accumulated for each additional carousel round required for successful download of the object. The impact of the accumulating t_{wait} each required additional carousel round stresses even further the importance of receiving the object in small number of carousel rounds. Furthermore, if $t_u << t_C$, the comparison between the uncoded and the encoded object downloads becomes more fare since t_{wait} is equal in both cases.

In Fig. 4, 5 and 6, the results are calculated as the average of all carousel rounds required for successful object download with a given IP PER (0%, 1%, 2%, etc.). In addition, the minimum and maximum of observed carousel rounds for each average value is given as error bars. The highest concentration of test results was observed close to the average value. A single carousel round denotes one transmission of the encoded object. In Fig. 5 and 6, the results are shown for the data encoded with the Tornado code. Here the data is interleaved as mentioned earlier. The difference in the number of carousels required for error free downloading between the interleaved and uninterleaved data were insignificant. This is mainly due to the relatively short average burst error length. Our tests also showed unsurprisingly that for longer average error burst length the significance of the interleaving is enhanced.

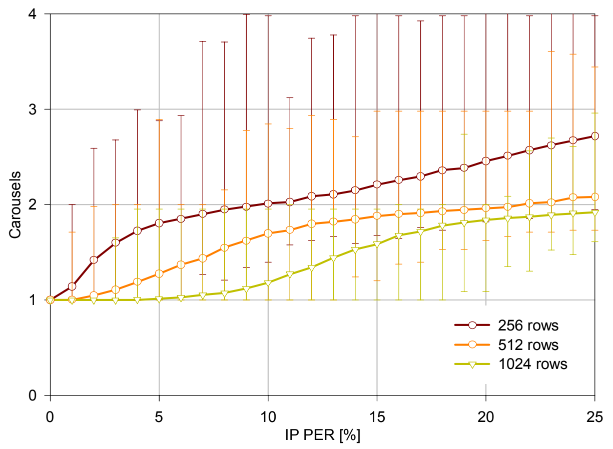

Figure 4: Carousels required for error free downloading using different number of rows in the MPE-FEC.

Fig. 4 illustrates the average number of carousel rounds required for error free downloading of the object, when MPE-FEC is utilized. The figure clearly shows the benefit of using 1024 rows in the MPE-FEC, with regard to required carousel rounds, compared to 256 and 512 rows. The MPE-FEC with 1024 rows provides the system with the ability to download objects with IP PERs up to 25% in less than two carousel rounds on average. However, when fewer rows are utilized, the applicability of MPE-FEC for file downloading services is diminished.

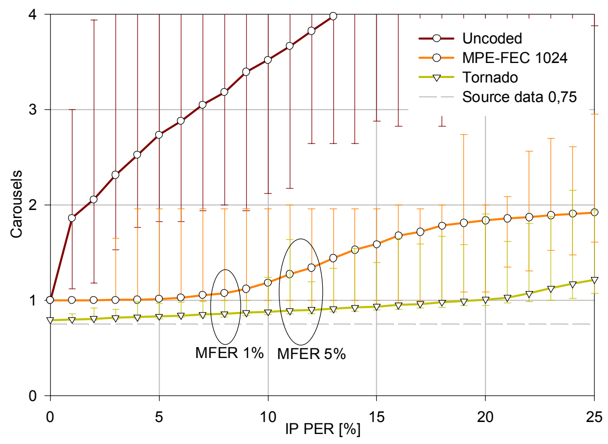

Figure 5: Carousels required for error free downloading for uncoded data, MPE-FEC and Tornado.

Fig. 5 shows the average number of carousel rounds that are required for downloading an object when the object is transmitted as uncoded, as coded with an MPE-FEC with 1024 rows, and as coded with a 5-dimensional (4000, 3000) Tornado code. As mentioned already earlier, a single carousel round for the uncoded object includes 25 % less data as for the coded objects.

From Fig. 4 and 5 can be seen that downloading an object, coded with MPE-FEC with 1024 rows, in approximately one carousel round is possible for IP PERs up to 5%. Comparing this with the application layer coding, one can see that the application layer code gives far better results. An object coded with the Tornado code, as specified above, can be received in less than one carousel round for IP PERs up to 17% with high probability. This gives the terminal the possibility to turn off the receiver as soon as enough data has arrived, even before the end of the first carousel round. For clarification, a reference line is included in Fig. 5, showing how much of the transmitted data that was source data. Obviously, the reference line does not involve the uncoded transmission.

Fig. 6 shows the same results as Fig. 5 but on a larger scale. The figure demonstrates that indifferently of the received IP PER, the application layer code will give better results than the MPE-FEC.

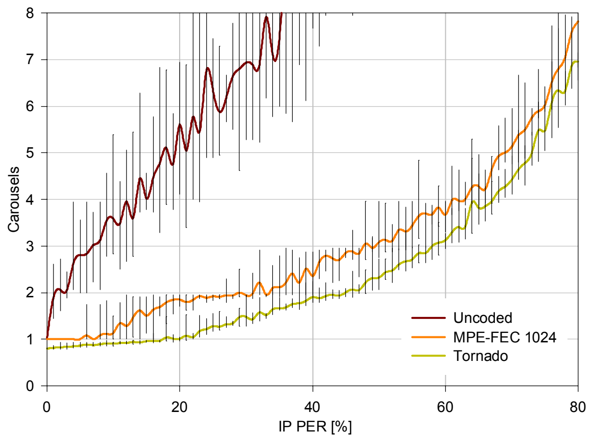

Figure 6: Required carousels for error free downloading for higher IP packet error rates.

The test results show that the MPE-FEC code results in a “staircase”-like curve with growing IP PERs with large variations, compared to the Tornado code, which results in a curve resembling a continuous curve with small variations. This is another desirable property of the application layer code, as it gives a more predictable outcome.

Furthermore, Fig. 5 shows the error rates corresponding to MPE-FEC frame error ratios, MFER 1% and MFER 5%, respectively. [11] defines a subjective failure point (SFP) for a streaming video service. The SFP corresponds fairly well to a packet error ratio of 10^{-4} on Transport Stream level. The SFP is however not directly applicable to DVB-H [12], but instead it is suggested that MFER 5% should be used instead to measure Quality of Restitution, which is also known as the Objective Failure Point.

Fig. 7 shows a conceptual overview of the implications of moving the error correction from the MPE-FEC to the application layer FEC in case of a file downloading service. The dashed line represents an imaginary coverage area for a video streaming service. The criteria for coverage here is MFER 5%. The graph in Fig. 5 shows that the MPE-FEC coded object cannot be received, on average, in 1 carousel round under these conditions. The coverage area will hence be smaller. The application layer FEC still manages quite well under these conditions and will, on average, manage even higher error ratios than the IP PER of 12-13% corresponding to MFER 5%. Hence, the coverage area for the application layer FEC is larger than the area for the MPE-FEC.

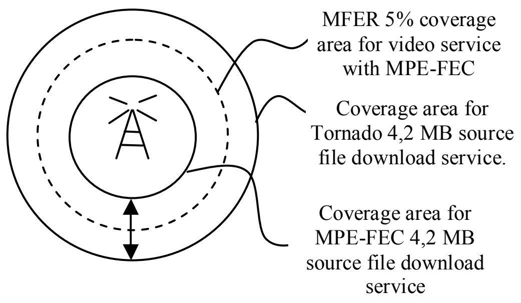
MFER 5% coverage area for video service with MPE-FEC
Coverage area for Tornado 4,2 MB source file download service.
Coverage area for MPE-FEC 4,2 MB source file download service

Figure 7: Service coverage for file downloading and video streaming.

VI. Conclusions

In this paper, we have presented a system for filecasting in wireless datacasting networks. The system consists of a carousel-based filecasting protocol co-operating with a forward error correcting code. Both the filecasting protocol and the FEC code were designed for filecasting over mobile wireless networks, which typically show very dynamical radio channel characteristics, observed by the receiver as varying burst error length profiles. As filecasting systems are sensitive to single bit errors in the delivered objects, we show that using long erasure correcting codes, a considerable increase in resistance against these channel errors can be achieved, compared with the optional MPE-FEC error correcting with the same amount of redundancy. Moreover, we have shown that in good reception conditions this redundancy can be disregarded at the terminal end by shutting down the receiver. This results in more economical use of energy resources as the system works as if no redundancy was added.

This, however, does not directly affect the bandwidth utilization from the network point of view, only in terms of reduced need for transmitting several carousel rounds. In future filecasting systems, where the bulk data is delivered using DVB-H and the lost data using e.g. UMTS, there can be a large gain in using strong error correcting in the filecasting system.

In this paper, a Markov-chain model was used for modeling the wireless channel characteristics. This model might not be sufficient to describe the typical DVB-H channel, as the characteristics greatly vary over time, hence producing longer burst errors than the model suggest. In this case the proposed system would be even more beneficial, but this should however be verified using field trials and more simulations.

REFERENCES

[1] "Digital Video Broadcasting (DVB); Transmission System for Handheld Terminals (DVB-H)", ETSI EN 302 304 V1.1.1, 2004.

[2] M. Luby, M. Watson, T. Gasiba, T. Stockhammer and W. Xu, "Raptor Codes for Reliable Download Delivery in Wireless Systems", in Proceedings CCNC 2006, to appear.

[3] M. A. Shokrollahi, "Raptor Codes", in Proceedings of ISIT 2004.

[4] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo and J. Crowcroft, "RFC 3451 – Layered Coding Transport (LCT) Building Block".

[5] M. Luby, M. Mitzenmacher, M.A. Shokrollahi, D.A. Spielman and V. Stemann, "Practical Loss-Resilient Codes", in Proceedings of the 29th Annual Symposium on Theory of Computing, 1997, pp. 150-159.

[6] M. Luby, M. Mitzenmacher, M.A. Shokrollahi and D.A. Spielman, "Efficient Erasure Correcting Codes", IEEE Transactions on Information Theory, 2001, 47(2). pp. 569-584.

[7] J.W. Byers, M. Luby, M. Mitzenmacher and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data", in Proceedings of ACM SIGCOM'98, 1998, pp. 56-67.

[8] K. Nybom and J. Björkqvist, "Designing Tornado Codes as Hyper Codes for Improved Error Correcting Performance", in Proceedings of AICT'06, 2006, to appear.

[9] J. McDougall and S. Miller, "Sensitivity of Wireless Network Simulations to a Two-State Markov Model Channel Approximation", in Proceedings of Globecom'03, 2003, pp. 697-701.

[10] J. Peltotalo, S. Peltotalo and J. Harju, "Analysis of the FLUTE Data Carousel", in Proceedings of EUNICE 2005, pp. 138-142.