HLDPC Codes – Low Density, Low Complexity, Efficient Erasure Correcting Codes

Kristian Nybom, Jerker Björkqvist

Department of Information Technologies, Faculty of Technology

Åbo Akademi University, Turku, Finland

***Abstract*—In this paper we introduce an erasure correcting code, which is designed to be used for providing error free data object transmissions over noisy wireless data casting systems, such as DVB-H. The code belongs to the class of low-density parity-check (LDPC) codes, and uses multiple parallel code structures, also called hyper codes. We also provide a straightforward algorithm for generating the parity-check matrix, which can generate instances of the code for different code sizes and coding capabilities, using only a few parameters. This gives us flexibility in designing the data transmission system. Using simulations and field measurements we show that the code performance is similar to that of the current state of the art codecs. The coding and decoding complexities are low, enabling software-based use in energy-constrained mobile devices.**

***Index Terms*—FEC, LDPC codes, hyper codes, application layer codes**

I. Introduction

CODING techniques are extensively being used to improve the quality of service in wireless networks. Environmental conditions are known to deteriorate the quality of reception and different coding techniques are therefore needed to correct corrupted information. Especially for mobile receivers, as in DVB-H networks [1], there is a need for developing error correcting codes that are capable of reconstructing transmitted objects both with as high probability and with as low complexity as possible.

Raptor codes [2, 3] are a fairly new class of codes, which have become popular in broadcasting scenarios, mainly because of their good erasure correcting capabilities, their small reception overheads, their low complexity, and their ability to produce an infinite stream of encoding symbols. In relatively small broadcasting scenarios, i.e. when the number of receivers is small and the receivers can inform the transmitter when they have received an object, the ratelessness of Raptor codes can efficiently be exploited. However, when the number of receivers is large, as will probably be the case in DVB-H broadcasting scenarios, there is no possibility to use a feedback channel to the transmitter. This implies that receivers cannot inform the transmitter when they have received the transmitted object and since there probably are several different objects that should be transmitted within a time frame, the amount of transmitted Raptor encoding symbols have to be limited. In effect, the Raptor code is changed from a rateless code to a code with a fixed code rate.

In this paper, we introduce the hyper low-density parity-check (HLDPC) code, which is based on some of the ideas presented in [2-7]. The HLDPC code is a more developed code than the Tornado code that was presented in [7], but since the only similarity, apart from the sparse graph structure, that the HLDPC and the Tornado code has is the Soliton distribution, we see it unfair to call this code a Tornado code. Furthermore, in [7] only the general idea of the hyper designed code is discussed, but no detailed information is given regarding the construction of the code. Therefore, the main contribution of this paper is to introduce the HLDPC code, to describe the methodology on how to create a HLDPC code and to give guidelines on how to specify the parameters for the code. We show with simulations and with field measurements that the HLDPC code has a similar performance as the Raptor code in DVB-H broadcasting scenarios.

II. Parity-Check Matrix Construction

HLDPC codes are systematic, fixed-rate, large block, and erasure correcting codes that are constructed with sparse parity-check matrices. The symbol sizes can be any predefined number of bits. The HLDPC code is a hyper code, utilizing several dimensions of encoding symbols. Hunt et al. [4] argue that by using several dimensions in the code, the minimum distance is increased when compared to the case where only one dimension, i.e. a standard code, is used. Throughout this paper, with dimension we refer to the hyper dimension and not to the code dimension.

In addition to the number of dimensions, the performance of the code is strongly related to the degree distribution of the message and parity symbols. By carefully choosing the degrees, the parity-check matrix can be made sparse, resulting in both an efficient decoding algorithm and a good erasure correcting performance. For these purposes, the HLDPC code uses the Soliton distribution for generating the degrees, which has been shown in [8] to result in both an efficient parity-check matrix generation and a good probability of successful decoding.

The overall algorithm for generating the parity-check matrix for an HLDPC code is given by the following steps:

1. Generate the degree distribution for the message symbols in the first dimension
2. Distribute the message symbol edges
3. Calculate parity symbol degrees for uncovered parity symbols so that all parity symbols are covered and distribute their edges
4. Permute the message symbols, once for each remaining dimension, so that the final parity-check matrix is obtained

These steps are described in detail in the following subsections.

A. The Hyper Structure

Hyper codes are a family of codes, which operate by arranging parity equations into multi-dimensional systems of equations. The multi-dimensionality gives the decoder several alternate possibilities to reconstruct missing symbols. In effect, hyper codes divide the parity-check matrix into several parts, where every part corresponds to one dimension. Therefore, when creating hyper codes, the number of dimensions needs to be specified. Every dimension contains the message symbols and a fraction of the total amount of parity symbols. Between the dimensions, the message symbols are permuted and in every dimension, the parity symbols are calculated based on the permuted message symbols in that dimension. Fig. 1 displays a parity-check matrix for a 3-dimensional HLDPC code. Fig. 1 also shows that the code can be viewed as a hyper-LDGM code.

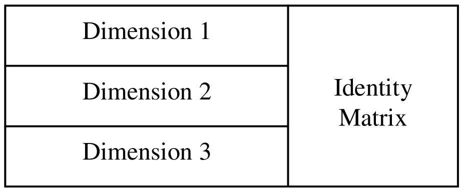

Fig. 1. The parity-check matrix structure for an HLDPC code with three dimensions. Every dimension utilizes a fraction of the parity symbols for error correction.

For a d-dimensional (n, k) HLDPC code, the number of parity symbols in every dimension, s, is given by (1), and is equal to the total number of parity symbols divided by the number of dimensions, rounded up. This calculation may result in an increasing of n so that the total number of parity symbols equals $d \cdot s$.

$$s = \left\lceil \frac{n-k}{d} \right\rceil \tag{1}$$

B. The Degree and Edge Distributions

The HLDPC code uses the ideal Soliton distribution [8] for generating the message symbol degrees. The number of elements in the distribution is equal to the number of message symbols.

$$\rho(i) = \begin{cases} 1/m & \text{if } i = 1 \\ 1/i(i-1) & \text{if } i > 1 \end{cases} \tag{2}$$

Equation (2) defines the ideal Soliton distribution, where ρ denotes the Soliton distribution and m is the number of elements in the distribution. Fig. 2 illustrates the Soliton distribution for 100 elements.

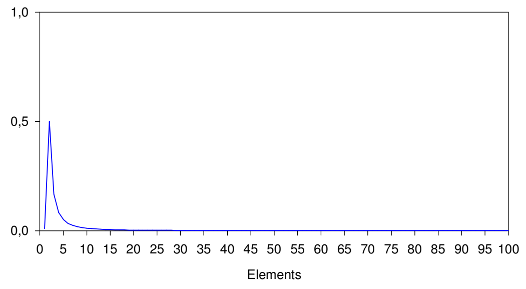

Fig. 2. The Soliton distribution for 100 elements.

When generating the degree distribution for the message symbols, the number of parity symbols in every dimension must be determined using (1). The minimal degree of message symbol i, $\underline{\deg(i)}$, is then calculated with (3), as the discrete Soliton distributed value i times the number of parity symbols in every dimension, rounded up.

$$\underline{\deg(i)} = \lceil \rho(i) \cdot s \rceil, \forall i \in k \tag{3}$$

This results in a small fraction of the message symbols having a relatively high degree, while the majority of the message symbols have the degree equal to one. The degree distribution needs only to be calculated in the first dimension, as will become clear in section II.C.

For distributing the edges, the HLDPC code relies on the algorithms presented by Harrelson et al. [9]. Using the limited randomness edge distribution algorithm, a degree number of edges is distributed for every message symbol, while knowing that the maximum edge value is equal to the number of parity symbols in the dimension. The adapted version of the limited randomness algorithm is given by (4), when the message symbols are indexed from zero to k-1 and the parity symbols in every dimension are indexed from zero to s-1.

$$\begin{aligned} &\text{1. Choose two integers } a \in \{1, ..., s-1\} \text{ and} \\ &\quad b \in \{0, ..., s-1\} \text{ uniformly at random} \\ &\text{2. The } i\text{th neighbor of the message symbol is then} \\ &\quad \text{the } ((ai+b) \bmod s)\text{th parity symbol} \end{aligned} \tag{4}$$

Harrelson et al. argue that as long as the maximum edge value is a prime number, all symbols can be covered. This, however, is rarely the case and therefore, the uncovered parity symbols have to be covered as well. Covering the parity symbols is done by determining a degree for every uncovered parity symbol and then by distributing the edges to the message symbols with the limited randomness algorithm, i.e. by substituting s with k in (4). The degree for the parity symbols should be chosen in the following manner: let f denote the discrete cumulative Soliton distribution and v be a random number, where $v \in [0,1]$. Find j such that $f(j-1) \le v < f(j)$. The degree of the parity symbol is then j.

C. *Finalizing the Parity-Check Matrix*

Once the degree and edge distributions have been generated for the first dimension, the remaining dimensions can be created by using the same edge and degree distributions as in the first dimensions and then permuting the message symbols. Once the message symbols have been permuted, new parity symbols can be calculated. This procedure is repeated until the parity symbols for all dimensions have been calculated.

As a permutation algorithm, a modified block interleaver can be used. The modification involves choosing a starting index from where the interleaving should begin. By employing this strategy, the permutation improves the possibility that no symbol has the same index in several dimensions. The starting index, b, is calculated with (5), where d_i is the dimension to which the permuted message symbols are being determined.

$$b = \left\lceil \frac{d_i - 1}{d} \cdot k \right\rceil, i \in \{2, ..., d\} \qquad (5)$$

Using (5) for a 5-dimensional HLDPC code with $k = 1000$, the starting indexes, are thus 200, 400, 600, and 800. Other permutation algorithms are certainly possible to use, as long as the algorithms strive to ensure that symbols do not share positions between dimensions. The sharing of positions results in replications of edge distributions, which is not desirable, because this results in reduced erasure correcting performance.

It is fairly easy to see that for each permutation in the code, the minimum distance is increased with one at most. This fact comes from the message symbol degree distribution: since the majority of the message symbols have degree one in the first dimension, then for every permutation performed, i.e. subsequent dimension, the final degree of most of the message symbols is increased with one. Because the minimum distance is defined as the minimum number of linearly independent columns plus one in the parity-check matrix, then for every dimension in the code, the minimum distance is increased with one at most. Recommendations on how to choose the number of dimensions will be discussed in section IV.

Because the code is an erasure correcting code, the decoding algorithm is simple. The decoding consists of finding those parity symbols, which know all their neighbors but one. The missing neighbor is then calculated as the modulo two sum of the parity symbol and all the parity symbols known neighbors. This eliminates the need of using an iterative decoding algorithm, which in turn yields a low complexity and efficient decoding algorithm. This also implies that the decoder can use received symbols for decoding as soon as they are received.

III. Implementation Guidelines

When implementing the HLDPC code there is no need to maintain the parity symbols as different sets of symbols, i.e. one set for every dimension. The complexity of the code can be significantly reduced if all the parity symbols are maintained as one set. By distributing the edges in a proper manner, the permutation of the actual message symbols can be avoided, while achieving the same result as if the message symbols were permuted. Fig. 3 illustrates an example of how a 2-dimensional (16, 8) HLDPC code is permuted with a block interleaver, without the starting index given in (5). In Fig. 3, the first row in every matrix indicates the symbol indexes in order to clarify the procedure. Note that after the message symbols have been permuted, the parity-check matrix in the second dimension is identical to the parity-check matrix in the first dimension, only with different symbol indexes. After the message symbols have been permuted, the columns are sorted according to the message symbol indexes to achieve the final parity-check matrix structure in the second dimension.

Dimension 1:

$$\left[\begin{array}{cccccccc|cccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ \hline
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1
\end{array}\right]$$

Dimension 2 after permutation of message symbols:

$$\left[\begin{array}{cccccccc|cccc}
1 & 4 & 7 & 2 & 5 & 8 & 3 & 6 & 13 & 14 & 15 & 16 \\ \hline
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1
\end{array}\right]$$

Dimension 2 after sorting the columns:

$$\left[\begin{array}{cccccccc|cccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 13 & 14 & 15 & 16 \\ \hline
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1
\end{array}\right]$$

Fig. 3. Permutation of a 2-dimensional (16, 8) HLDPC code. The first row in every matrix indicates the symbol indexes. Sorting the permuted dimension gives the final structure for the second dimension.

When all dimensions are merged into one parity-check matrix the parity-check matrix for the entire code is obtained. Fig. 1 illustrates such a parity-check matrix for an HLDPC code with three dimensions. The complexity of both the encoder and decoder is reduced by permuting the edges instead of the message symbols, and this also gives a straightforward encoding and decoding algorithm.

[10] specifies a triple generator, which can be used to efficiently generate pseudo-random numbers for the edge distributions. The triple generator calculates three numbers, *a*, *b*, and *d*, where *d* is a degree and *a* and *b* are the random

numbers used for the limited randomness algorithm in (4). When generating the message symbol edges, the *d* value should be omitted, since the degree is based on the Soliton distribution, while for the parity symbols, the *d* value should be used for the degree. The triple generator takes as input the number of message symbols and the symbol ID, which is the index of the symbol for which the values are generated. A benefit of using the triple generator is that both the transmitter and the receiver have the same random number generator. This eliminates possible problems caused by using different random number generators.

To improve the delivery of encoding symbols, each symbol can be prefixed with encoding symbol information, as described in [10]. In this case, each symbol is prefixed with the following three values:

- Source Block Number (SBN)
- Encoding Symbol ID (ESI)
- Source Block Length (SBL)

The SBN describes which encoding block the symbol belongs to. The ESI is the index of the symbol. If ESI is less than *k*, then the symbol is a message symbol. Otherwise it is a parity symbol. The SBL is the number of message symbols in the encoding block.

If the encoding symbol information is used in conjunction with the triple generator, the input parameters to the triple generator are obtained directly from the encoding symbol information, i.e. SBL and ESI. In this case, the SBL is used for generating the Soliton distribution, and thereby also the degrees. This implies that as soon as one encoding symbol is received, the entire parity-check matrix can be constructed, as long as the code rate, the number of dimensions and the permutation algorithm are known. As will be seen in section IV, the number of dimensions and the permutation algorithm can be fixed beforehand.

IV. HLDPC Parameter Specification

For complete generation of the parity-check matrix, three parameters must be specified: the number of message symbols, the code rate, and the number of dimensions. The first two parameters are clearly dependent on the transmission scheme, but the specification of the number of dimensions is not obvious. As mentioned in section II, the minimum distance increases with one at most with every dimension. From the minimum distance, the minimum number of symbols that can be corrected can be calculated. If the performance of the code is measured with the minimum distance, the HLDPC code may be viewed as a poor code because of its low minimum distance. However, as will be seen in section V, the code has a similar performance as the Raptor code. Hence, the minimum distance is not an adequate measurement of the performance of the code. What is more interesting to study is the average message symbol erasure rate after decoding and the amount of data that has to be transmitted in order for the receiver to be able to reconstruct an object, when measuring code performances.

Fig. 4 illustrates the decoded message symbol erasure rates vs. code rates for HLDPC codes with 3000 message symbols on a TU6 channel with a Doppler frequency of 79 Hz. The results are based on laboratory measurements of a DVB-H IP stream, made by Nokia. Similar results are obtained with other Doppler frequencies and message lengths, but with the maximum acceptable code rate shifted. The measured IP stream had an average IP packet erasure rate (IP PER) of 9.5 % and every IP packet contained exactly one symbol.

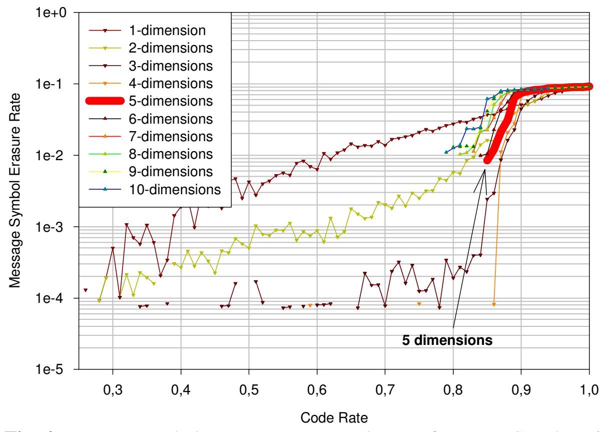

Fig. 4. Message symbol erasure rates vs. code rates for HLDPC codes with 3000 message symbols and up to 10 dimensions on a TU6 channel with Doppler frequency 79 Hz. 5-dimensional codes are the most beneficial ones.

As can be seen, using less than four dimensions does not produce an acceptable erasure correction capability, indifferently of the code rate. When the number of dimensions is increased beyond six dimensions, the erasure correcting performance is reduced, because the parity-check matrix becomes too dense. The remaining alternatives are hence four, five and six dimensions. Extensive testing has shown that five dimensions is the most beneficial and reliable alternative. Simulations have proved that the message length does not significantly affect the results. The message length only affects the maximal code rate that can be used but the number of dimensions is unaffected. It is therefore our recommendation to fix the number of dimensions to five in the HLDPC code, indifferently of the code rate and the code length.

The permutation algorithm should be chosen so that the symbols are spread out as far as possible from their original positions and so that no symbol has the same position in several dimensions. For this purpose, interleaving strategies can be employed. An example of a permutation algorithm is given in section II. Obviously, both the encoder and the decoder must use the same permutation algorithm. It is therefore imperative that the algorithm is determined either separately for each object or is fixed for all objects. If the algorithm is chosen separately for each object, the specification on which algorithm is used must be delivered to the receiver.

V. Simulations

The simulations were carried out on a TU6 DVB-H channel with a constant Doppler frequency of 10 Hz. The HLDPC code was used as an application layer code and every symbol was placed in exactly one IP packet. Hence, entire missing or corrupted IP packets could be corrected. The implementation of the HLDPC code was made using the triple generator and the encoding symbol information discussed in section III, and the permutation algorithm was the modified block interleaver given in section II.C. The code rate was fixed to 3/4 and five dimensions were used. The number of message symbols in the codes was chosen to 6000 and 3000 symbols.

For comparison, a non-systematic Raptor code was used. The Raptor code does not have a code rate because of its ratelessness, but rather uses a reception overhead. If $n' = k(1+\varepsilon)$ is the number of encoding symbols that have to be received in order to reconstruct a Raptor block, then ε is the reception overhead. By limiting n' to a fixed value n, i.e. allowing a maximum of n transmitted symbols for an object as discussed in the introduction, the Raptor code obtains a code rate. This is something that has to be done in large scale broadcasting scenarios in order to limit the time during which an object is being transmitted. For the simulations, n' was limited to $4/3*k$, i.e. the code rate was limited to 3/4, so that comparable results were obtained.

The modulations that were used on the TU6 channel were QPSK, 16QAM and 64QAM. The QPSK and 16QAM modulations were coded with convolutional code rates (CCR) 1/2 and 2/3, while the 64QAM modulation was coded with a CCR 1/2. No MPE-FEC error correction was utilized. In Fig. 5-9, the decoded message symbol erasure rates are higher than the original erasure rates at lower C/N values, because the Raptor code was non-systematic,

A. *Erasure Correction Performance Measurements*

Fig. 5 and Fig. 6 illustrate the erasure correcting performances for the codes on a QPSK-modulated TU6 channel with CCR 1/2 and 2/3 respectively. As can be seen, both the HLDPC and Raptor code produced quasi-error free objects for C/N values of approximately 7-8 dB and 10 dB, for CCR 1/2 and 2/3 respectively. For a criterion on the symbol erasure rate of 10^{-3}, the coding gain was 5-6 dB for CCR 1/2 and 6 dB for CCR 2/3.

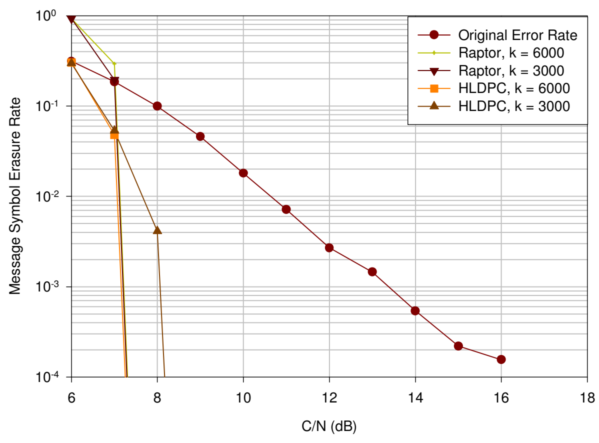

Fig. 5. Erasure correction performances of Raptor and HLDPC codes on a QPSK-modulated TU6 channel with convolutional code rate 1/2.

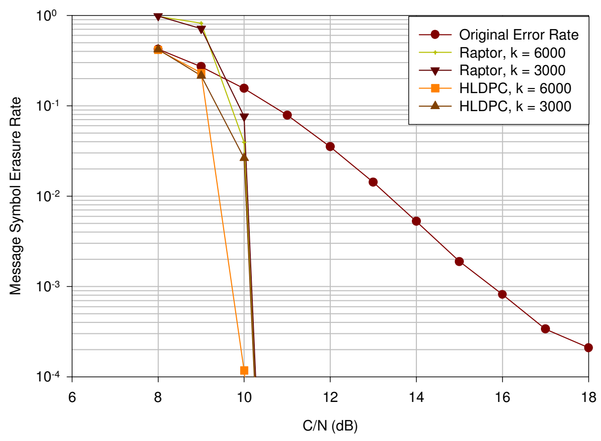

Fig. 6. Erasure correction performances of Raptor and HLDPC codes on a QPSK-modulated TU6 channel with convolutional code rate 2/3.

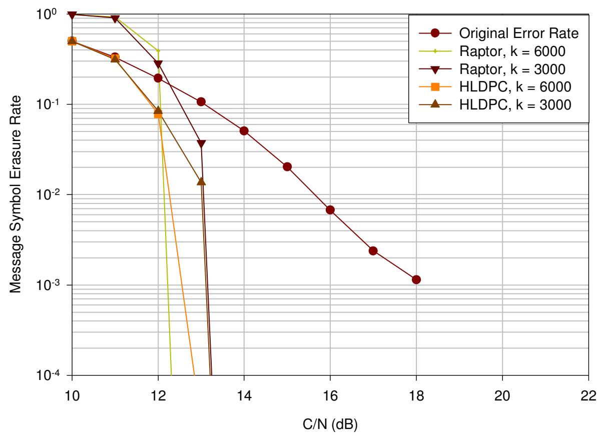

Fig. 7. Erasure correction performances of Raptor and HLDPC codes on a 16QAM-modulated TU6 channel with convolutional code rate 1/2.

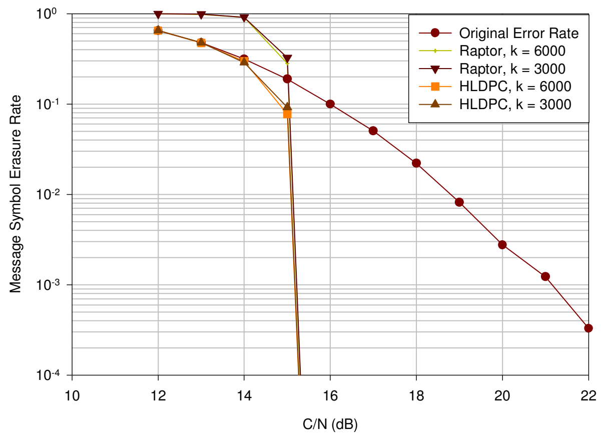

Fig. 8. Erasure correction performances of Raptor and HLDPC codes on a 16QAM-modulated TU6 channel with convolutional code rate 2/3.

For the 16QAM-modulated transmission, illustrated in Fig. 7 and Fig. 8, the required C/N values for quasi-error free correction was increased with approximately 5 dB for CCR 1/2 and 2/3 respectively. However, the equal erasure correction performances were retained. For the 16QAMmodulated simulation, the coding gains were approximately 5 dB and 6 dB for CCR 1/2 and 2/3 respectively.

Fig. 9 shows that there is no observable difference between the erasure correcting performances of the HLDPC code and the Raptor code for the 64QAM-modulated simulation. The codes produced quasi-error free objects at C/N values of approximately 17 dB and the coding gain was 5 dB.

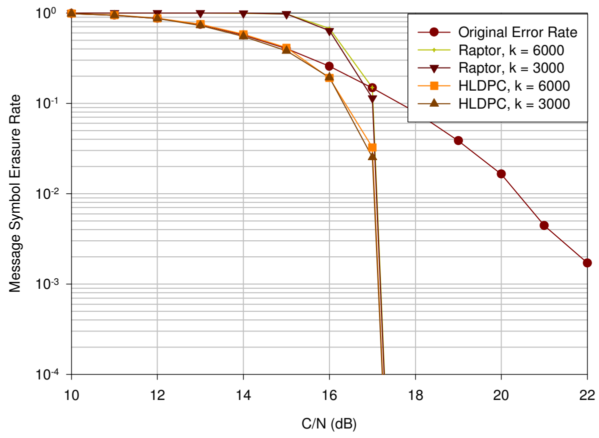

Fig. 9. Erasure correction performances of Raptor and HLDPC codes on a 64QAM-modulated TU6 channel with convolutional code rate 1/2.

What can be noted in Fig. 5-9 is that irrespectively of the modulations used and of the coding used prior to the HLDPC code or Raptor code, the erasure correction performances of the codes are similar.

B. Transmission Overhead Measurements

With transmission overhead, we refer to the ratio of data, proportional to the message length, that has to be transmitted for complete recovery at the receiver. We define the transmission overhead as the number of transmitted symbols divided by *k* minus one. The transmission overhead gives a measure of how long each receiver has to wait before it is able to reconstruct an object. The difference between transmission and reception overhead is that the transmission overhead measures the number of symbols that have to be transmitted, of which some are lost, while the reception overhead measures the number of symbols that have to be received, for complete recovery.

For measuring the transmission overhead, it was assumed that the objects were transmitted repeatedly over and over again in a carousel-like manner. The HLDPC code was interleaved with an IP interleaver, using a block interleaver with an interleaving depth equal to the number of encoding symbols. Because of the random generation of Raptor encoding symbols, the performance of the non-systematic Raptor code cannot be improved with interleaving. Therefore, the comparison is fair, although the Raptor code was not interleaved.

Fig. 10-12 illustrate the transmission overhead vs. C/N for the HLDPC and Raptor codes on a QPSK-, 16QAM-, and 64QAM-modulated TU6 channel, with CCR 1/2 and 2/3 for the QPSK- and 16QAM-modulations and CCR 1/2 for the 64QAM-modulation. The results show that for low C/N values, the Raptor code has a slightly lower transmission overhead than the HLDPC code. This is because almost every encoding symbol the Raptor code receives is different from the previous ones and can therefore be used for decoding with high probability. When the C/N increases there is no notable difference between the transmission overheads. Note that at the C/N values at where the transmission overheads differ notably, the IP PER is above $2*10^{-1}$, which is a relatively high IP PER for any network.

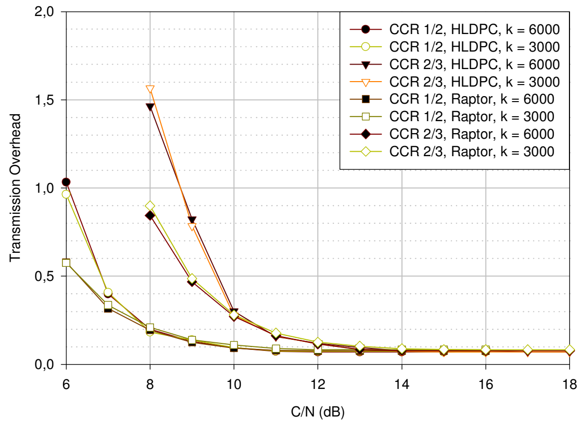

Fig. 10. Transmission overheads for HLDPC and Raptor codes on a QPSK-modulated TU6 channel with convolutional code rates 1/2 and 2/3.

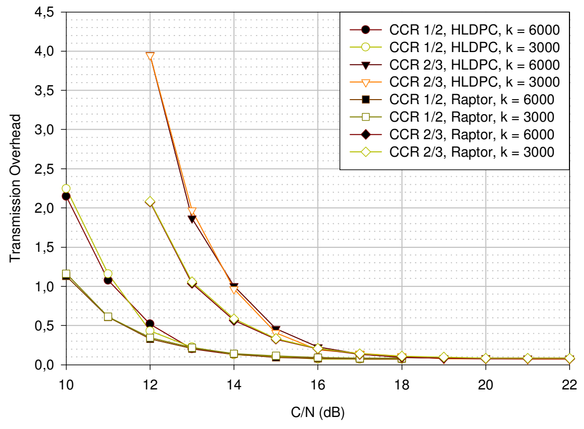

Fig. 11. Transmission overheads for HLDPC and Raptor codes on a 16QAM-modulated TU6 channel with convolutional code rates 1/2 and 2/3.

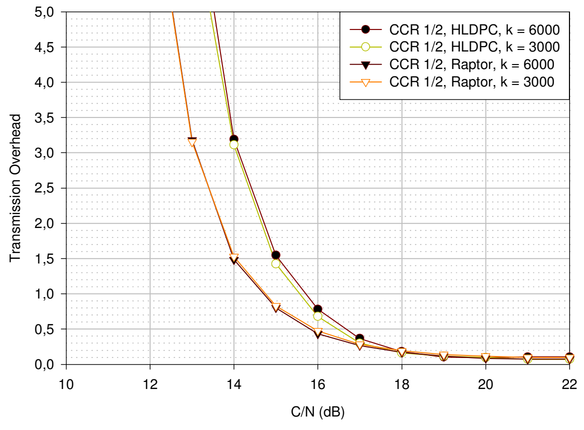

Fig. 12. Transmission overheads for HLDPC and Raptor codes on a 64QAM-modulated TU6 channel with convolutional code rate 1/2.

Fig. 13 shows an interesting property of the HLDPC code. The figure illustrates the relationship between the reception overhead and the IP PER for an IP interleaved 5-dimensional (8000, 6000) HLDPC code compared to a non-systematic Raptor code with equal message length. What is noteworthy is that the reception overhead for the HLDPC code is almost equal to the IP PER for IP PERs up to approximately $2*10^{-1}$. This implies that for every lost message symbol, a little more than one additional symbol is required for recovery on average. What is also noteworthy is that for IP PERs up to $8*10^{-2}$, the HLDPC code outperforms the Raptor code in terms of reception overhead.

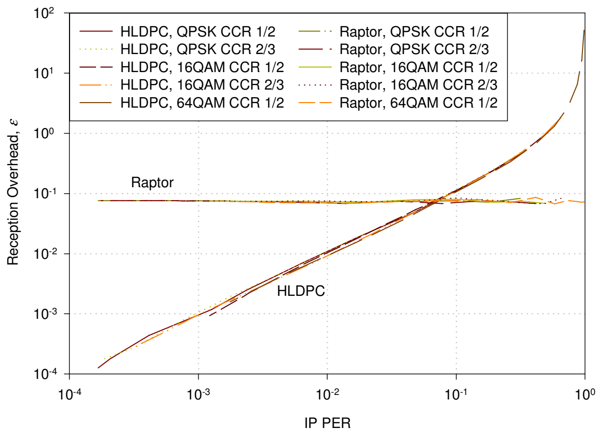

Fig. 13. Reception overhead vs. IP packet erasure rate for an interleaved 5-dimensional (8000, 6000) HLDPC code compared to an equal length Raptor code on a QPSK-, 16QAM-, and 64QAM-modulated TU6 channel, with convolutional code rates 1/2 and 2/3.

VI. Field Measurements

The HLDPC and Raptor codes were compared with field measurements. For the measurements, data was transmitted using a DVB-H test network and received with a DVB-T receiver, connected to a laptop, with implementations of the HLDPC and Raptor codes.

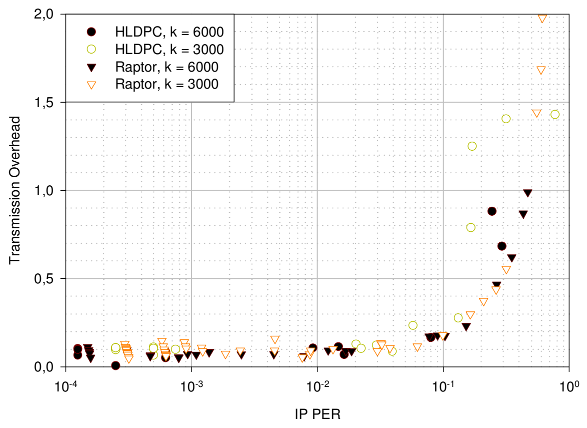

Fig. 14. Transmission overhead vs. received IP packet erasure rate for DVB-H mobile pedestrian outdoor measurements using HLDPC and Raptor codes.

Because the purpose of the application layer codes in DVB-H is file delivery, there was a zero error tolerance on the measurements and therefore, only the transmission overhead was measured. The measured results, which are illustrated in Fig. 14, resemble the simulation results showed in section V.B and there is no notable difference between the transmission overheads between the HLDPC and the Raptor code, except for the higher IP PERs.

VII. Conclusions

In this paper, we introduced an erasure correcting code, called HLDPC, for use in noisy wireless data casting systems. A common problem for LDPC codes, which this represents, is to generate the code structure. We presented a systematic algorithm for the construction of the parity-check matrix, which takes into account the requirements for degree and edge distributions of the code. The code uses a hyper code structure, i.e. multiple parallel codes, and permutations of data between the codes.

The code was analyzed primarily using simulations of a DVB-H system, based on the TU6 channel model, but also using data acquired from field measurements. The results show similar performance to Raptor codes.

The code is computationally of low complexity, the basic operations being bitwise operations on data, hence facilitating implementation in software layers of receiver devices. Additionally, in good reception conditions, the reception overhead is very low, making earlier receiver shut down possible, thus saving energy.

The code was not optimized against any particular channel model. The necessity of this remains open, and could be analyzed in future work.

References

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

[2] A. Shokrollahi, "Raptor Codes", in *Proceedings of the International Symposium on Information Theory* (ISIT 2004)

[3] A. Shokrollahi, "Raptor Codes", *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551-2567, 2006.

[4] A. Hunt, S. Crozier, and D. Falconer, "Hyper-Codes: High-Performance Low-Complexity Error-Correcting Codes", in *Proceedings of the 19th Biennial Symposium on Communications*, pp. 265-267, 1998.

[5] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Efficient Erasure Correcing Codes", *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569-584, 2001.

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

[7] K. Nybom and J. Björkqvist, "Designing Tornado Codes as Hyper Codes for Improved Error Correcting Performance", in *Proceedings of the Advanced International Conference on Telecommunications* (AICT 2006), 2006.

[8] M. Luby, "LT Codes", in *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science* (FOCS 2002), pp. 271-282, 2002.

[9] C. Harrelson, L. Ip, and W. Wang, "Limited Randomness LT Codes", in *Proceedings of the 41st Annual Allerton Conference on Communication, Control, and Computing*, 2003.

[10] "Universal Mobile Telecommunications System (UMTS); Mobile Broadcast/Multicast Service (MBMS); Protocols and Codecs", 3GPP TS 26.346 version 6.4.0 Release 6.