Joint Source-Channel-Fountain Coding for Asynchronous Broadcast

Nicolas Dütsch, Hrvoje Jenkač, Timo Mayer, and Joachim Hagenauer
Lehrstuhl für Nachrichtentechnik (LNT)
Technische Universität München (TUM)
Arcisstraße 21, D-80290 München, Germany
E-mail: {Nicolas.Duetsch, H, Timo.Mayer, Hagenauer}@tum.de

***Abstract*— A novel concept of joint source-fountain coding for asynchronous wireless broadcast is presented. The traditional field of joint source-channel coding aims in both lossless data compression as well as reliable data delivery. Besides, fountain coding allows reliable broadcast without the need of feedback, and furthermore, asynchronous data access, i.e., each receiver is able to decide when it wants to subscribe to the ongoing broadcast session. Our approach combines these traditional distinct subjects. We introduce an appropriate system model and present a realization for a joint source-fountain code by applying the capacity approaching, parallel concatenated turbo-codes. Simulation results are provided, showing that our joint approach outperforms a separated scheme composed of source coding and subsequent fountain coding.**

I. Introduction

Traditionally, source and channel coding are performed separately. First, the redundancy of the source is removed. Then, the compressed representation is protected against transmission errors by channel coding. Optimality is guaranteed though only for asymptotically large block lengths. However, in many practical applications the conditions of Shannon's separation theorem neither hold, nor can be used as a good approximation.

Because of the duality between source and channel coding, channel codes originally designed for error protection, also achieve good performance in noisy data compression scenarios [1] and even allow lossless compression [2], [3]. Typically, these approaches are based on two powerful code classes: turbo codes [4] and low-density parity-check (LDPC) codes [5]. Their common success lies in the exchange of messages at the decoder. Recently, these compression techniques have been extended. Very efficient joint source-channel codes providing incremental redundancy techniques have been proposed in [6] and [7]. Both lossless compression and reliable transmission is achieved at the same time, exploiting appropriate feedback information.

Unfortunately, feedback oriented approaches, aiming in full reliability, cannot be efficiently applied in broadcast environments. Especially for a large number of receivers, the feedback implosion problem is observed. Motivated from this drawback, a new class of codes, namely the *fountain codes*, has been discovered [8], [9], which provides an infinite amount of redundancy. It was shown that such codes guarantee full reliability in broadcast environments without requiring feedback channels. Moreover, segment asynchronous data reception is possible, from the view of each receiver (a segment will be specified in Section II). Practical realizations of fountain codes have been obtained by the introduction of, e.g., LT-codes [10], Raptor-codes [11], and the Turbo-Fountain [12].

In this work we investigate the novel concept of joint source-fountain coding. A joint code should provide the following properties at the same time: (i) lossless data compression (ii) fully reliable data delivery without feedback (iii) asynchronous data access in broadcast environments. We introduce a method based on parallel concatenated turbo codes to accomplish this goal. We show that for binary memoryless sources joint source-fountain coding outperforms a separated scheme consisting of source coding first and fountain coding afterwards.

The paper is organized as follows: Section II introduces the system model, as well as some definitions and basics. In Section III, we introduce the encoding and the decoding structure of our proposal, along with some tools needed for lossless data compression. In the subsequent section, simulation results are presented and discussed.

II. System Model

We consider a wireless broadcast system with a single transmitter and multiple receivers. The goal is to compress a message and broadcast it reliably without losses to all receivers, i.e., without any residual errors or missing data. Furthermore, we assume that no feedback channel is available to request retransmissions for lost data. Each receiver is able to check the integrity of the data, typically consisting of several thousand bits, by a CRC-check. However, every single receiver should be able to receive the message error-free, independent of its receiving conditions. Furthermore, every receiver should be able to decide when it starts to receive the message, independent of other receivers. Fig. 1 shows the system consisting of a single encoder, i.e., the *joint source-fountain code*, which accomplishes all of these requirements at the same time. Consider a source message $\boldsymbol{u} = (u_1, \ldots, u_k)$ of length k bits to be compressed and reliably distributed. The source is assumed to output statistically independent symbols of the binary alphabet $\{+1, -1\}$ with probabilities $\Pr\{u = +1\} = p$ and $\Pr\{u = -1\} = 1 - p$, resulting in a

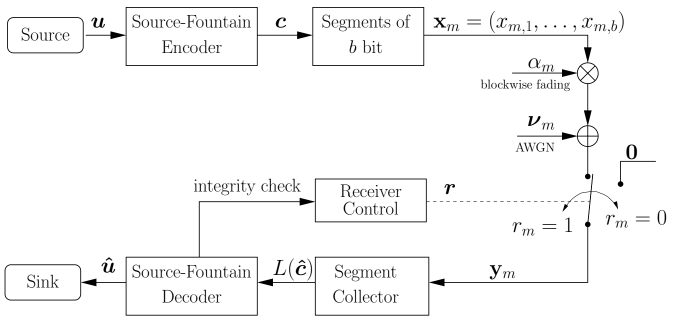

Fig. 1. System model: wireless broadcast system with joint source-fountain coding and asynchronous data access.

source entropy $H = -p\log_2(p) - (1-p)\log_2(1-p)$, measured in bit/symbol. Further, we define

$$L_{SSI} \triangleq \log\left\{\frac{\Pr\{u=+1\}}{\Pr\{u=-1\}}\right\} \tag{1}$$

as the *source state information* (SSI), with $\log$ denoting the natural logarithm.

The finite information sequence $\boldsymbol{u}$ is encoded with a joint source-fountain code, which is assumed to produce a code word $\boldsymbol{c}$ of n encoded bits. The specific property of this code results in an infinite sequence of code symbols, $n \to \infty$, likewise to the property of fountain codes [8]. We assume that the reader is familiar with the idea of fountain coding and refer to [8], [9], [12]. After appropriate segmentation, where b bits are grouped to a transmission segment $\mathbf{x}_m \triangleq (x_{m,1}, \ldots, x_{m,b})$, the coded data is transmitted as a sequence of ± 1, i.e., BPSK modulated. It is straightforward to extend the scheme to higher order modulation, e.g., M-ary QAM with $M = 2^B$, where a segment would consist of $b \cdot B$ bits and b QAM symbols. Assume that the infinite segment sequence $\boldsymbol{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m, \ldots)$ is sequentially broadcasted over the channel, i.e., the transmitter serves as perpetual fountain.

Now assume that the smallest entity the receiver is able to receive is a segment $\mathbf{x}_m$, typically of the order of 100 bits. Furthermore, assume that receivers tune into the ongoing broadcast session at any segment time with index m, without any coordination among receivers or any information being sent to the transmitter. Some receivers even might decide to interrupt their listening for some time. To formalize the random receiver behavior, we assign to each receiver a receiver pattern $\boldsymbol{r} = (r_1, r_2, \ldots, r_m, \ldots)$ with $r_m \in \{0,1\}$. Thereby, $r_m = 1$ indicates that the receiver subscribes to the channel at segment index m, whereas $r_m = 0$ denotes that the receiver does not listen to the m-th segment of the broadcast session. The receiver pattern $\boldsymbol{r}$ is determined by the *receiver control*.

Let $\mathbf{x}_m \triangleq (x_{m,1}, \ldots, x_{m,b})$, $\mathbf{y}_m \triangleq (y_{m,1}, \ldots, y_{m,b})$ and $\boldsymbol{\nu}_m \triangleq (\nu_{m,1}, \ldots, \nu_{m,b})$ be the transmitted signal, with $x_{m,i} \in \{\pm 1\}$, the received signal and the noise sample during segment m, respectively. Noise is assumed to be Gaussian, i.i.d., and the channel signal-to-noise ratio (SNR) is defined as E_s/N_0. The propagation channel is assumed slowly time-varying and frequency-flat for each segment m. In particular, the channel is assumed to be constant over the entire segment m with fading coefficient α_m. Then, the received signal over segment m is given as $\mathbf{y}_m = r_m(\alpha_m \mathbf{x}_m + \boldsymbol{\nu}_m)$. We assume that each receiver has perfect knowledge of the channel gain α_m and has access to the SNR.

The L-value [13] of a binary random variable $x \in \{\pm 1\}$ is defined as

$$L(x) \triangleq \log\left\{\frac{\Pr(x=+1)}{\Pr(x=-1)}\right\}, \tag{2}$$

and the soft-bit of the realization x as

$$\lambda(x) = \mathrm{E}\{x\} = \tanh\left\{\frac{L(x)}{2}\right\}. \tag{3}$$

The L-value is also denoted as log-likelihood ratio (LLR). Then, channel decoding is based on the LLR $L(x|y)$. Following this and assuming a fading AWGN channel the LLR of each received symbol i within segment m can be expressed as [13]

$$L(x_{m,i}|y_{m,i}) = 4\alpha_m \frac{E_s}{N_0} y_{m,i}. \tag{4}$$

In the case $r_m = 0$, the reception of the corresponding segment is bypassed, as shown in Fig. 1. This definition is already included in (4) where the L-value at the receiver becomes zero. The *segment collector* concatenates the received segments to a continuous sequence $\hat{\boldsymbol{c}} = (\hat{c}_1, \hat{c}_2, \ldots)$, or more specifically, the corresponding LLRs are collected. The gathered LLR $L(\hat{\boldsymbol{c}})$ is passed to the joint source-fountain decoder which outputs $\hat{\boldsymbol{u}} = (\hat{u}_1, \ldots, \hat{u}_k)$ as an estimate of $\boldsymbol{u}$. An error detection mechanism, e.g., a CRC, performs an integrity test and triggers the receiver control: If the information collected from the channel is still not sufficient, the joint source-fountain decoder waits for further information.

III. Joint Source-Fountain Coding: The Turbo-Fountain

In this section, we describe an appropriate encoder and decoder structure which is able to fulfill the requirements defined in Section I. We employ the capacity approaching parallel-concatenated turbo-codes [4] in order to produce the infinite amount of redundancy.

A. Turbo-Fountain Encoder

Fig. 2 shows the encoder structure of the *Turbo-Fountain*, as we term our realization of a joint source-channel-fountain encoder. In a first step, the information symbols $\boldsymbol{u}$ are encoded by a traditional parallel concatenated turbo code, consisting of two recursive systematic convolutional (RSC) component codes, producing the parity blocks $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$. The length of each parity sequence depends on the rate R_{cc} of the component codes, as well as on k. Depending on the source statistics, compression can be achieved in the case $H < 1$. This is specified in more detail in the next subsection.

In the second step, the encoded symbols are applied to a potentially infinite number of random interleavers π_i, resulting

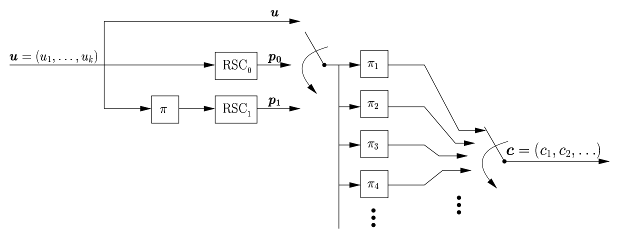

Fig. 2. Encoder structure of the Turbo-Fountain based on a parallel concatenated turbo code and an infinite number of random interleavers to obtain an infinite amount of redundancy.

in a potentially infinite amount of output symbols $\boldsymbol{c}$. For practical implementations the number of incorporated interleavers might be fixed. Then, the interleavers are read out cyclically in order to produce a limitless output sequence, known under the acronym *broadcast disk* [14]. Alternatively, pseudo-random interleavers are generated randomly on the fly, similar to the random LT connections in the original fountain codes [10], [11].

B. Turbo-Fountain Decoder

The corresponding decoder structure is illustrated in Fig. 3. The collected LLRs $L(\hat{\boldsymbol{c}})$ are mapped to single interleaver

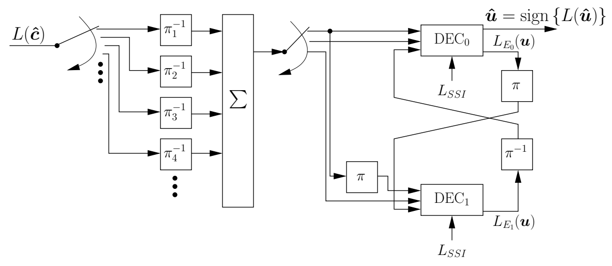

Fig. 3. Decoder structure of the Turbo-Fountain.

branches, inversely to the distribution procedure at the encoder. Concerning the synchronization issue and the interleavers, we assume that synchronized pseudo-random generators at the transmitter and the receivers are used. Usually, in wireless systems data is transmitted block-wise and hence, sequence numbers are available to initialize the pseudo-random generators at the receivers. The synchronization issue is necessary for any fountain code, and was already discussed extensively, e.g., in [8]. After appropriate de-interleaving using π_i^{-1} in every branch, the corresponding L-values are additively combined and finally passed to a turbo decoder. The success of turbo decoding depends on the exchange of extrinsic information $L_E(\boldsymbol{u})$ between both APP component decoders. After a maximum number of I iterations a hard decision on the a posteriori LLR of the source bits is made: $\hat{\boldsymbol{u}} = \mathrm{sign}\{L(\hat{\boldsymbol{u}})\}$.

In order to obtain the best performance with iterative source-channel decoding, the SSI , defined in (1), has to be taken into account. This can be estimated on-the-fly during the iterative decoding as described in the following subsection.

Prior investigations in the field of turbo source coding have shown that perfect reconstruction at the receiver can be predicted by observing the width of the tunnel in the extrinsic information transfer (EXIT) chart [15]. At the beginning of iterative decoding, the tunnel is either open, depending on the SSI, or has to be opened be means of *systematic doping* (i.e., systematic symbols are made available at the receiver in addition to the parity symbols) depending on the distribution of the source symbols. The result is that in the first case, all systematic symbols can be discarded at the turbo encoder output and do not have to be transmitted, which was mentioned in the previous subsection. However, if the entropy H is close to one (i.e., $L_{SSI} \approx 0$), some of the systematic symbols should be broadcasted. The dependencies between SSI and the amount of doping is part of our ongoing research.

C. Estimation of the Source State Information

In order to profit from unbiased source symbols during iterative decoding, the SSI has to be known at the receiver. In point-to-point communication systems this additional knowledge can be calculated on the basis of the original source message at the transmitter and send to the counterpart before transmitting the source information. Because of the asynchronous data access this is not feasible and thus, we have to estimate the SSI during iterative decoding.

An algorithm to guess the SSI on-the-fly was published in [1]: After each iteration j, the extrinsic information on the source symbols determined by each component code as well as the estimated source state information are added

$$L^{(j)}(u_i) = L_{E_0}^{(j)}(u_i) + L_{E_1}^{(j)}(u_i) + L_{SSI}^{(j)} \tag{5}$$

and transfered to the soft-bit domain

$$\lambda^{(j)}(u_i) = \tanh\left\{\frac{L^{(j)}(u_i)}{2}\right\}. \tag{6}$$

We take the average soft-bit as we assume the source to be i.i.d. and go back to the L-value representation. Finally we obtain the estimated SSI for the next iteration $(j+1)$

$$L_{SSI}^{(j+1)} = 2\tanh^{-1}\left\{\frac{1}{k}\sum_{i=1}^{k}\lambda^{(j)}(u_i)\right\}. \tag{7}$$

As shown in [1] no significant degradation is observed if the source state information is not known a priori and has to be estimated. The price to pay is that more iterations are needed until decoding converges.

IV. Performance Evaluation

A. Simulation Environment

We implemented a simulation environment as shown in Fig. 1, incorporating the Turbo-Fountain, which serves as the joint source-fountain code. The corresponding simulation parameters are given in Table I. The random start of data reception was simulated by assigning appropriate receiver patterns $\boldsymbol{r}$. Theoretically, any random receiver pattern $\boldsymbol{r}$ could be possible. However, it is not feasible to consider all theoretically possible receiver patterns in the simulation. Therefore, we restricted to practical receiver behaviors only, i.e., we assume a continuous reception until the source message is decoded, whereby the begin of the reception was chosen by random. The receiver control tracks decoding success by an CRC integrity test, and stops reception from the channel when sufficient information is collected. Let $\rho = \sum_{m=1}^{\infty} r_m$ denote the number of observed segments by the receiver in order to achieve decoding success. Since we are interested in the performance of the joint source-fountain code an appropriate performance measure is required. Aiming at lossless recovery of the source message, a valid measure is the average amount of symbols $\overline{n} \triangleq b \cdot \mathrm{E}\{\rho\}$ required to achieve decoding success or to be more specific the average receiver throughput $k/\overline{n}$. Note that, since the Turbo-Fountain provides source compression, $k/\overline{n} > 1$ can be achieved. However, from Shannon's theory it is well known that the average rate $k/\overline{n}$ is upper bounded by C/H, with C denoting Shannon's capacity of a binary-input/soft-output channel, and H the entropy of the source. Note that if the entropy H is one, i.e., for equally likely input symbols, the average receiver throughput $k/\overline{n}$ is upper bounded by C. On the other hand, for a perfect channel, i.e., $C = 1$, $k/\overline{n}$ is upper bounded by $1/H$.

We compare our proposed Turbo-Fountain system with a system consisting of separated source coding and fountain coding. The binary symbols of the source sequence ($k = 16000$) were grouped to 8-bit symbols and compressed using the standard Unix compression tool 'bzip2' with an implementation of the Huffman-Algorithm. Afterwards, traditional fountain coding using LT-codes [8], [10] was performed and transmitted over the channel. In order to obtain a fair comparison, the LT-code was iteratively soft-decoded with the sum-product algorithm [16], and not hard-decoded as in the original LT-code proposal [10]. The extension of the soft-decoded LT-code to joint source-channel-fountain coding is straighforward and part of our ongoing research.

TABLE I
SIMULATION PARAMETERS

Information message length	$k = 16000$ bit
Radio segment size	$b = 160$ bit
Rate of component codes	$R_{cc} = 1/3$
Polynomial of component encoders	$[1, 33/37, 25/37]_8$
Maximum number of iterations	$I = 20$
Component decoding algorithm	Max-Log-MAP BCJR [17], [18]
Interleaver size π	k
Max. interleaver size π_i	$(2/R_{cc} - 1)\,k$

B. Simulation Results

Fig. 4 shows simulation results on the AWGN channel for $p = 0.90$ (i.e., $H = 0.469$ bit/symbol). The average receiver throughput $k/\overline{n}$ is plotted versus E_s/N_0 in dB. The Turbo-Fountain, which performs joint source-fountain coding, is indicated with the bold line. As can be observed, our approach outperforms the reference system, with separate source coding followed by traditional fountain coding over the entire investigated E_s/N_0 region. In the low E_s/N_0-region, we obtain a gain of approximately 2 dB when applying our proposed system. However, the Turbo-Fountain still remains a small gap to the upper bound C/H, which leaves room for future research in order to approach C/H.

Fig. 5 shows the same system but on the block fading channel with Rayleigh distributed channel gains α_m, e.g., realized by frequency hopping. Basically, the same tendencies as explained for the AWGN channel can be observed. This is due to the fact that both systems exploit soft-information in a similar way. Again, our proposed system comes quite close to the upper bound for the lower average SNR-region. Likewise, the gain in the lower E_s/N_0 region is again approximately 2 dB.

Fig. 6 shows simulation result for fixed E_s/N_0, but for a

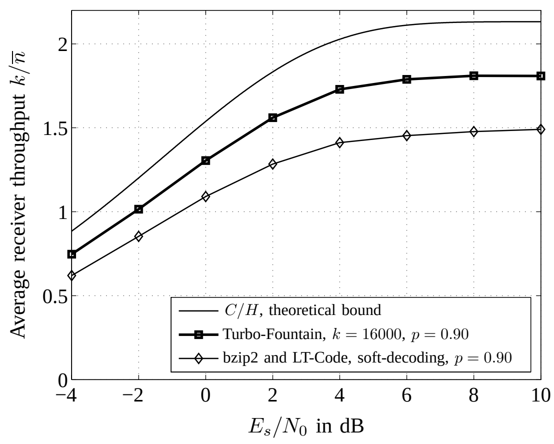

Fig. 4. Simulation results on the AWGN channel. Average receiver throughput $k/\overline{n}$ vs. E_s/N_0.

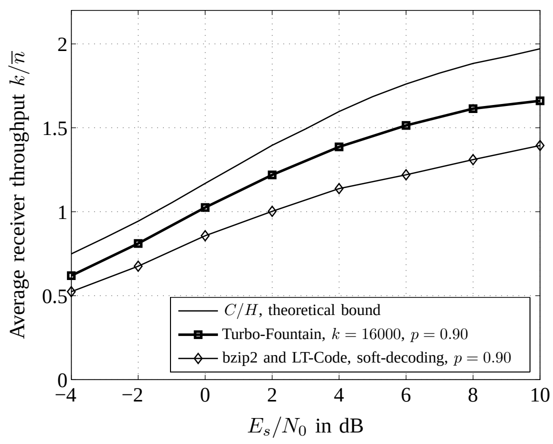

Fig. 5. Simulation results on the block-wise Rayleigh fading channel. Average receiver throughput $k/\overline{n}$ vs. average E_s/N_0.

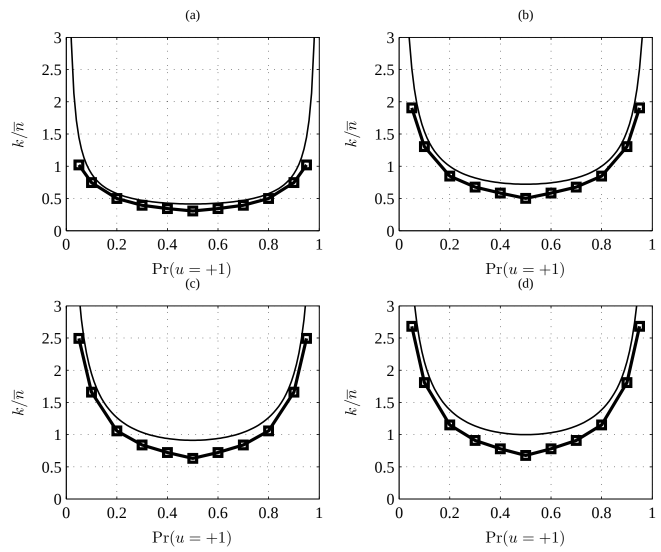

Fig. 6. Simulation results on the AWGN channel. Average receiver throughput $k/\overline{n}$ vs. $\Pr(u = +1)$ for fixed E_s/N_0. (a): $E_s/N_0 = -4$ dB (b): $E_s/N_0 = 0$ dB (c): $E_s/N_0 = 3$ dB (d): $E_s/N_0 = 10$ dB

variable input distribution $\Pr\{u = \pm 1\}$. The average receiver throughput $k/\overline{n}$ is plotted versus the distribution of the input-symbols $\Pr\{u = +1\}$. The result of our proposal is marked with the bold line and is compared to C/H. Following the ideas of Section III, all systematic bits from the component encoders were punctured. This results in slightly decreased performance in the region $p \approx 0.5$, but provides better results outside this region. If we avoid to puncture the systematic bits, we obtain an inverse behavior, not presented here. As expected, reliable transmission and compression at the same time, i.e., $k/\overline{n} > 1$, can be achieved especially in the regions where the source symbols are distributed unequally, meaning $H \ll 1$.

V. Conclusions

We investigated the novel concept of joint source-fountain coding in a wireless broadcast environment. Joint source-fountain coding aims at lossless data compression, reliable broadcast, and asynchronous data access at the same time. No feedback information from the receivers to the transmitter is required. The transmitter broadcasts a perpetual data stream. Receivers tune-in into the ongoing broadcast session at any segment and just collect sufficient information in order to be able to reconstruct the information message. We presented the *Turbo-Fountain*, which serves as a possible realization of a joint source-channel-fountain code. The Turbo-Fountain, based on parallel concatenated codes, produces an unlimited amount of redundancy by appropriate interleaving. An appropriate decoder structure was presented. Simulation results for both the AWGN channel and the fading channel have been provided. We have shown that our joint source-fountain coding approach outperforms a separate approach, consisting of source coding, applying the Huffman algorithm, and subsequent fountain coding with LT-codes. Due to the remaining gap to the optimal bound (C/H), further optimization of the Turbo-Fountain is part of our future work.

References

[1] J. García-Frías and Y. Zhao, "Compression of binary memoryless sources using punctured turbo codes," *IEEE Communications Letters*, pp. 394–396, Sept. 2002.

[2] J. Hagenauer, J. Barros, and A. Schaefer, "Lossless turbo source coding with decremental redundancy," in *Proc. International ITG Conference on Source and Channel Coding*, Erlangen, Germany, Jan. 2004, pp. 333–340.

[3] G. Caire, S. Shamai, and S. Verdú, "A new data compression algorithm for sources with memory based on error correcting codes," in *Proc. IEEE Information Theory Workshop*, Paris, France, Mar. 2003, pp. 291–295.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE International Conference on Communications*, Geneva, Switzerland, May 1993, pp. 1064–1070.

[5] R. G. Gallager, "Low-Density Parity-Check codes," *IRE Transactions on Information Theory*, pp. 21–28, Jan. 1962.

[6] G. Caire, S. Shamai, and S. Verdú, "Almost-noiseless joint source-channel coding-decoding of sources with memory," in *Proc. International ITG Conference on Source and Channel Coding*, Erlangen, Germany, Jan. 2004.

[7] N. Dütsch and J. Hagenauer, "Combined incremental and decremental redundancy in joint source-channel coding," in *International Symposium on Information Theory and its Applications*, Parma, Italy, Oct. 2004, pp. 775–779.

[8] J. Byers and M. Luby and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1528–1540, Oct. 2002.

[9] M. Mitzenmacher, "Digital fountains: A survey and look forward," in *Proc. IEEE Information Theory Workshop 2004, San Antonio, TX, USA*, Oct. 2004, pp. 271–276.

[10] M. Luby, "LT codes," in *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science*, Nov. 2002, pp. 271–280.

[11] A. Shokrollahi, "Raptor codes," Digital Fountain, Tech. Rep. DR2003-06-001, Jun. 2003.

[12] H. Jenkac, J. Hagenauer, and T. Mayer, "The Turbo-Fountain and its application to reliable wireless broadcast," in *Proc. European Wireless 2005*, Nicosia, Cyprus, Apr. 2005, pp. 1–7.

[13] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, pp. 429–445, Mar. 1996.

[14] S. Acharya, M. Franklin, and S. Zdonik, "Dissemination based data delivery using broadcast disks," *IEEE Pers. Communications*, vol. 2, pp. 50–60, Dec. 1995.

[15] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[16] H. Jenkac, T. Mayer, T. Stockhammer, and W. Xu, "Soft decoding of LT-codes for wireless broadcast," in *Proc. Mobile Summit 2005*, Dresden, Germany, June 2005.

[17] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. IT-20, pp. 284–287, Mar. 1974.

[18] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," *IEEE Transactions*, pp. 1009–1013, Feb. 1995.