The Turbo-Fountain and its Application to Reliable Wireless Broadcast

(Invited Paper)

Hrvoje Jenkač, Joachim Hagenauer, and Timo Mayer

Institute for Communications Engineering (LNT)
Munich University of Technology (TUM), Germany
email: **h@tum.de, hagenauer@tum.de, timo.mayer@tum.de**

***Abstract*: Reliable wireless broadcast with asynchronous data access based on fountain coding is investigated. We review the traditional problem formalization for fountain codes operating on erasure channels and we generalize the problem formalization to arbitrary types of channels. We introduce a novel type of rateless codes based on the turbo principle: the *Turbo-Fountain*. The Turbo-Fountain is able to consider soft information from the channel in the decoding process. Two realizations for the Turbo-Fountain are introduced. We show simulation results for the Turbo-Fountain realizations on the AWGN and on fading channels. Additionally, we compare the achievable Turbo-Fountain performance with traditional fountain codes designed for the erasure channel, both on the AWGN and on fading channels, considering appropriate erasure declaration. It is shown that the Turbo-Fountain provides significant performance gains, due to exploitation of soft information, and approaches capacity.**

1. Introduction

Recently, reliable wireless broadcast has gained significant interest with the standardization and introduction of Multimedia Broadcast and Multicast Services (MBMS) into wireless cellular networks. Whereas many types of multimedia data tolerate residual errors to some extent, like video streaming or music distribution, in general, file download must be performed error-free, e.g., the distribution of executable programs. Broadcasting to wireless receivers in a cellular environment results in diverse receiving conditions for different receivers. Furthermore, link adaptation cannot be utilized in the broadcast mode.

Therefore, approaches based on Automatic Repeat reQuest (ARQ) have been proposed and protocols for broadcast ARQ elaborated. However, all retransmission strategies require feedback channels from the receivers to the transmitter. This is not feasible for a multitude of receiving entities, as it is the case within a football stadium where replays are distributed to a vast number of cell phones. Furthermore, the system throughput applying ARQ is degenerating with a growing number of receiving entities. This phenomenon is also known as feedback implosion.

Alternatively, Forward Error Correction (FEC) with a fixed code rate can be applied. However, this results in unnecessary reception overhead for receivers with good channel conditions, and residual errors for receivers with bad or moderate channel states. Moreover, both solutions do not allow asynchronous access of receivers to the data. However, in [1] and [2] FEC for reliable broadcast was investigated and, furthermore, in [3], it was elaborated that *fountain codes*, like, e.g., LT-Codes [4] or Raptor codes [5], solve the reliable broadcast problem without requiring feedback channels. Traditionally, these codes were investigated on erasure channels and have been proposed for Internet communications. However, in wireless broadcast other channel models apply, like the Gaussian channel or the fading channel. These types of channel models provide soft-information about each received bit. Recently in [6] the author encourages the research community to look for novel types of *Digital Fountain* codes and Digital Fountain approximations.

In this work we present an alternative method to approximate a Digital Fountain by applying turbo codes: the *Turbo-Fountain*. Moreover, we investigate the performance of the Turbo-Fountain on the AWGN and on fading channels, and provide a powerful decoding technique which exploits soft-information of the channel. We show how the *Turbo-Fountain* can be applied to reliable broadcast in a wireless environment.

This paper is organized as follows. In Section 2. we introduce the framework and give a short overview on related an prior work. Subsequently, we introduce the notation and formalize the problem. In Section 4. two realizations of the Turbo-Fountain are presented. In Section 5. we introduce our simulation setup and provide simulation results when applying the Turbo-Fountain. Finally, we summarize the major results.

2. Overview

2.1. Framework

We consider a wireless transmission system with a single transmitter and multiple receivers. The goal of this system is to broadcast a message reliably to all receivers, i.e., without residual errors or missing data within the message. We assume that no feedback channel is available to request retransmissions for lost data. However, every single receiver should be able to receive the message error-free, independent of its channel quality. Moreover, the receivers are assumed to tune-in into the ongoing broadcast session at arbitrary time, without any coordination among receivers. In the following, this is referred to as *asynchronous data access*. Asynchronicity among receivers can originate from various reasons. Some examples are i) different service request times of individual receivers, ii) reception pauses initiated by receivers, e.g., in order to receive other services in the mean time, or iii) cell reselection in wireless cellular system, i.e., the selection of a new transmitter. Many other scenarios resulting in a non-synchronous start of reception are imaginable. However, the receivers in our system should be able to reconstruct the message inde-

pendent of their access times or access patterns, and benefit from any received portion of the ongoing broadcast.

2.2. Related and Prior Work

In [3] the idea of a *Digital Fountain* was introduced in order to solve the reliable multicast problem with asynchronous data access. Recently, a survey on *fountain codes* and their applications has been given in [6]. A protocol was presented in order to solve the reliable multicast problem over erasure channels by applying these codes. A fountain code produces a potentially limitless number of code symbols from a finite information message, and theoretically provides a limitless amount of redundancy. Therefore, fountain codes have been denoted as *rateless*. The code symbols are broadcasted by the transmitter perpetually. Usually, it is assumed that the transmitter is at least aware of receivers being present in the serving area. If no receiver is present, the transmitter is assumed to interrupt the transmission in order to save resources. Let k denote the size of the information message in number of symbols. The code property of an ideal fountain code allows to reconstruct the original information message from any k code symbols out of the infinite code sequence. Hence, fountain codes allow multiple receivers to recover from different loss patterns, since each receiver just collects k symbols in order to be able to reconstruct the information message. Asynchronous data reception among the receivers is supported inherently by the same property since any k symbols yield in reconstruction success. Entailed with the application of fountain codes, there are many further nice properties, like the reception from multiple transmitters or the arbitrary number of possible receivers. Furthermore, a Digital Fountain allows reception pauses. A small drawback of the fountain concept is the limitation on download-and-play services, i.e., this concept is not applicable for real-time service like video streaming, maintaining the same marvelous properties.

Recently, there are many publications in many research areas adopting the fountain idea where rateless codes are applied to solve various problems. Practical approximations of a *Digital Fountain* have been obtained by the introduction of LT-Codes [4] and Raptor codes [5], which come close to the theoretical limit on the erasure channel. A formalized description of fountain codes on the erasure channel will be given in Section 3.2. Because fountain codes were traditionally proposed to solve the multicast problem over the Internet, these codes have been optimized and investigated on erasure channels. The loss behavior of the Internet can be basically modeled by an erasure channel since transmitted packets do not arrive at the receiver at all due to router congestion. In [7] investigations of LT-Codes and Raptor codes on noisy channels other than erasure channels have been performed. However, the work only studied bit error rates and block error rates, rather then reliable transmission in combination with asynchronous data access.

2.3. Extension to Wireless

In wireless communications data packets are not lost, but are subject to fading and Gaussian noise. Basically, traditional erasure based fountain codes can be applied in wireless systems on higher layers operating on a virtual erasure channel. In this case, radio blocks or higher layer packets, which have been received in error, are declared as erased. A significant amount of information, which is even available in erroneous radio blocks, is not exploited. However, it is well known that for channel decoding soft-information about each bit within a received radio block improves system performance significantly.

In this work we provide a framework which generalizes the fountain concept to channels other than erasure channels. We support our theoretical concept by proposing a novel type of Digital Fountain approximation applying Turbo codes and introduce the Turbo-Fountain. We investigate a wireless broadcast system, which allows both fully reliable and asynchronous data access by applying the Turbo-Fountain. The soft-information available at the wireless receiver is incorporated in the decoding algorithm yielding significant performance gains compared to traditional fountain codes designed for erasure channels.

3. Problem Formalization

3.1. Setup and Notations

Consider the information message $\boldsymbol{u} = (u_1, \ldots, u_k)$ of length k symbols, where all symbols u_i are of the same cardinality. For practical purposes u_i can represent a single bit or a tuple of bits. This information message $\boldsymbol{u}$ is encoded with a fountain code $\mathcal{F}$, which is assumed to produce an infinite sequence of code symbols $\boldsymbol{c} = (c_1, c_2, c_3, \ldots)$ from the finite information sequence $\boldsymbol{u}$, where $\boldsymbol{c} = \mathcal{F}(\boldsymbol{u})$. The code symbols c_i are assumed to be broadcasted over the broadcast channel in a successive manner. At the receiver of interest, the symbol sequence $\boldsymbol{y} = (y_1, y_2, y_3, \ldots)$ is received. Note that c_i and y_i need not to be from equal alphabets, this depends on the channel only. Now, let us define the receiver pattern $\boldsymbol{r} = (r_1, r_2, \ldots)$, containing random variables r_i, where $r_i \in \{0, 1\}$. $r_i = 1$ indicates that the receiver listens to the channel, whereas $r_i = 0$ is equivalent to the fact that the receiver does not listen to the channel, i.e., it does not receive the symbol i. In fact, this corresponds to the well known puncturing and $\boldsymbol{r}$ represents a puncturing pattern. Let us define $\hat{\boldsymbol{c}} = (\hat{c}_1, \hat{c}_2, \hat{c}_3, \ldots)$, with

$$\hat{c}_i = \begin{cases} y_i & r_i = 1 \\ \mathfrak{p} & r_i = 0 \end{cases},$$

and $\hat{c}_i = \mathfrak{p}$ indicating that a certain symbol c_i was punctured as a consequence of not subscribing to the channel. Moreover, let us define $|\hat{\boldsymbol{c}}| \triangleq \|\{i \,|\, \hat{c}_i \neq \mathfrak{p}\}\|$ as the number of overall available, un-punctured symbols at the receiver. Finally, we define the decoding process as $\hat{\boldsymbol{u}} = \mathcal{F}^{-1}(\hat{\boldsymbol{c}})$, with $\hat{\boldsymbol{u}} = (\hat{u}_1, \ldots, \hat{u}_k)$ denoting the decoding decision on the information sequence.

3.2. Erasure Fountain Codes (EFC)

Consider the memoryless erasure channel with its transition probabilities

$$\begin{aligned} \Pr\{y_i = \mathfrak{e}\} &= p_e, \\ \Pr\{y_i = c_i\} &= 1 - p_e, \end{aligned} \tag{1}$$

with $\mathfrak{e}$ denoting an erasure marker and p_e the erasure probability. Hence, $\hat{c}_i \in \{c_i, \mathfrak{e}, \mathfrak{p}\}$ contains a code symbols c_i, an erasure marker $\mathfrak{e}$ or a puncturing indicator $\mathfrak{p}$, respectively. We define $\kappa \triangleq \|\{i \,|\, (\hat{c}_i \neq \mathfrak{p} \wedge \hat{c}_i \neq \mathfrak{e})\}\|$ as the number of overall available un-punctured and unerased symbols at the receiver. Now, the Ideal Erasure Fountain Code (IEFC) $\mathcal{F}_I$ is defined by the following property

$$\mathcal{F}_I : \left\{ \mathcal{F}_I^{-1}(\hat{\boldsymbol{c}}) \equiv \boldsymbol{u} \; \forall \hat{\boldsymbol{c}} \,|\, \kappa \geq k \right\}, \qquad (2)$$

i.e., the reception of at least k arbitrary code symbols is sufficient to guarantee decoding success. Note, this corresponds to the well know Maximum Distance Separable (MDS) property. Unfortunately, practical codes do not achieve the MDS property. Usually a slightly increased number $\kappa = k'$ symbols ($k' > k$) is required to be received. The inefficiency of a practical Erasure Fountain Code (EFC) is expressed by ϵ, with $k' = (1+\epsilon)k$. Currently, the research community is working towards $\epsilon \to 0$ with reasonable complexity [4], [5], [8]. A nice property for the code design of erasure fountain codes is the fact that code optimization can be performed without knowledge about the channel loss rate.

3.3. Problem Generalization to Arbitrary Channels

In wireless communications, transmitted symbols are not lost like in Internet communications. In this work we extend the fountain idea to more appropriate channel models, concerning wireless communications, like the Gaussian channel or fading channels. These types of channels are commonly characterized by their probability density functions $p_{y_i}(y_i|c_i)$ and provide soft-information [9] about the received symbols.

We generalize (2) and define the capacity achieving, ideal fountain code $\mathcal{F}_S$ to fulfill the property

$$\mathcal{F}_S : \left\{ \mathcal{F}_S^{-1}(\hat{\boldsymbol{c}}) \equiv \boldsymbol{u} \; \forall \hat{\boldsymbol{c}} \,|\, |\hat{\boldsymbol{c}}| \geq \left\lceil \frac{k}{C} \right\rceil \right\}, \qquad (3)$$

where C is the Shannon's channel capacity which can be calculated from $p_{y_i}(y_i|c_i)$. This means, the reception of at least $\lceil \frac{k}{C} \rceil$ arbitrary symbols is sufficient to guarantee decoding success. It can easily be shown that (2) is obtained from (3) by inserting the channel capacity of the erasure channel. However, the formulations in (2) and (3) do not give any practical instructions how to find such codes, but just state the desired properties.

4. Turbo-Fountain

In Section 3.3., the properties of a capacity achieving fountain code have been formalized. Though no proof is given, intuitively, it seems to be a challenging and complex task to find such codes. In this section, we apply the capacity approaching turbo codes [9], [10], [11] to approximate an ideal fountain. We describe two different realizations which have in common to produce an infinite sequence of code symbols from a finite information sequence, according to the fountain idea. Furthermore, we introduce corresponding decoder structures based on soft-in/soft-out component decoders, in order to exploit the available soft information from the channel efficiently. In the following, we restrict to $u_i \in \{0, 1\}$ and $c_j \in \{0, 1\}$.

4.1. Parallel Turbo-Fountain

Fig. 1 shows the block diagram of the Turbo-Fountain encoder, based on parallel concatenated recursive convolutional codes. In the following, we denote this structure as *Parallel Turbo-Fountain*.

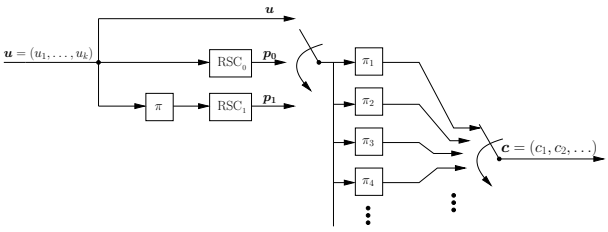

Figure 1: Encoder structure for the Parallel Turbo-Fountain based on parallel concatenated convolutional codes and infi nite number of random interleavers at the output.

The information bits $\boldsymbol{u}$ are successively applied to the encoder input, and directly passed to the first component encoder RSC_0. Additionally, an interleaved version of $\boldsymbol{u}$ is passed to the second component encoder RSC_1 in the same manner, where π is denoting a random interleaver. Both component encoders are rate $R_c = 1/2$ recursive systematic convolutional encoders. Only the parity symbols are used for further processing.

After a parallel-serial conversion, all output bits from both encoders, which are parity bits from both encoders and the termination bits, as well as the systematic bits, are applied to a potentially limitless number of random interleavers. In practice, the number of required interleavers depends on the number of required redundancy. Finally, the interleaved bits from each interleaver branch are mapped successively onto the output $\boldsymbol{c}$.

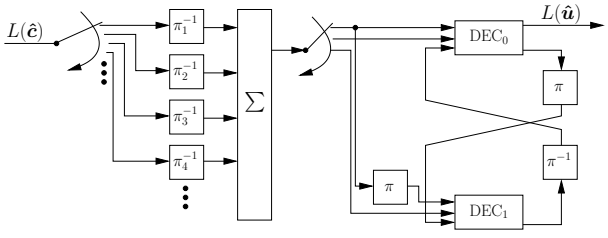

Figure 2: Soft-in/soft-out decoder for Parallel Turbo-Fountain.

Fig. 2 shows the corresponding decoder structure. The L-values [9] $L(\hat{\boldsymbol{c}})$ of the received bits are distributed to different branches inversely to the procedure at the encoder output. In general, we assume that the transmitter and the receivers utilize synchronized pseudo random generators, which can easily be realized in wireless systems since the data is transmitted block-wise and, hence, sequence or frame numbers are available to initialize the pseudo random generator at the receivers. In general, the synchronization issue applies to any type of fountain codes and is not a special problem of the Turbo-Fountain. After distributing the L-values to the differ-

ent branches, appropriate de-interleaving π_i^{-1} in each branch i is performed. Then, the de-interleaved L-values from all branches are combined (bit-wise summation) and passed to a state-of-the art turbo decoder [10] with two soft-in/soft-out component APP decoders. Within each iteration extrinsic information is exchanged between the component decoders, which serves as a-priori information for the other decoder. The L-values on the decoding decision $L(\hat{\boldsymbol{u}})$ are obtained at the decoder output, where $\mathrm{sign}\{L(\hat{\boldsymbol{u}})\}$ is the hard decision on each bit.

4.2. Multiple Turbo-Fountain

With the Parallel Turbo-Fountain structure, code bits output at time index i are output again later at time index $i' > i$. Therefore, we looked for an encoder structure which avoids repetition of code bits, in order to avoid collecting identical code bits several times at the receiver. Now, we present a solution based on multiple turbo codes, as introduced and proposed for deep-space communication in [12] or in [13], and denote the structure as *Multiple Turbo-Fountain* in the following. Multiple turbo codes are turbo codes with more than two parallel concatenated component codes. We extend the idea of multiple concatenated encoders and allow a potentially limitless number of parallel encoders.

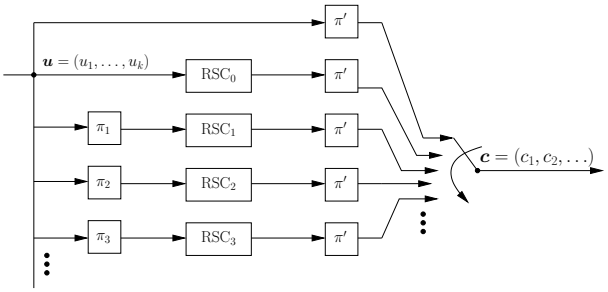

Figure 3: Encoder structure for Multiple Turbo-Fountain based on an infinite parallel concatenation of recursive systematic convolutional codes.

Fig. 3 shows the block diagram of the encoder. The k information bits $\boldsymbol{u}$ are directly passed to a random interleaver π' on the upper branch, as well as to the first component encoder RSC$_0$. All other encoders RSC$_i$, with $i > 0$, obtain an interleaved version $\pi_i(\boldsymbol{u})$ of the information bits. $(\pi_1, \pi_2, \ldots)$ are random interleavers, with $\pi_i \neq \pi_j \ \forall i \neq j$ and again, the component encoders are rate $R_c = 1/2$ recursive systematic convolutional encoders, only outputting the parity symbols. Then, the outputted bits from all encoders RSC$_i$ are interleaved using random interleavers π', as well. Hence, systematic bits are output only at the beginning of the fountain. After random interleaving, the bits from all branches are mapped consecutively onto the output $\boldsymbol{c}$.

Fig. 4 shows a possible decoder structure. The L-values $L(\hat{\boldsymbol{c}})$ of the received bits are distributed to the different branches, inversely to the procedure at the encoder output. The first branch contains the systematic bits. In order to pass systematic information to all component decoders, the systematic bits from the first branch are interleaved appropriately with the interleavers

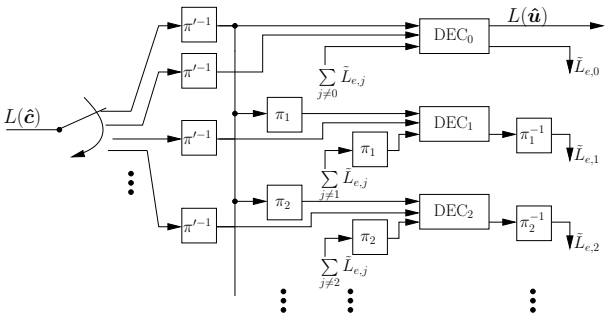

Figure 4: Soft-in/soft-out decoder for Multiple Turbo-Fountain.

π_i. As mentioned in the previous section, we assume that the corresponding random interleavers are generated by synchronized pseudo random generators. The parity bits designated for the different component decoders, which are received successively after the systematic bits, are passed after de-interleaving with π'^{-1} on the corresponding branches to the dedicated component decoders. In [12] several iterative decoding strategies for multiple turbo codes were presented. We apply parallel decoding, where all component decoders operate at the same time in parallel. Each component decoder DEC$_i$ is fed with a-priori information $L_{a,i}$ which is obtained by adding the extrinsic information $L_{e,j}$ from all other decoders $j \neq i$. As shown in the figure, an appropriate interleaving is required in order to allow the summation of extrinsic information. This can be formalized as

$$L_{a,i} = \pi_i\Big(\sum_{j\neq i} \tilde{L}_{e,j}\Big) = \pi_i\Big(\sum_{j\neq i} \pi_j^{-1}(L_{e,j})\Big). \qquad (4)$$

Note that for practical decoding, only those component decoders have to be considered in the decoding process which are fed with received symbols from the channel. Hence, there is always only a finite number of decoders involved. The actual number of required component decoders depends on the receiver pattern.

5. Performance Evaluation

5.1. Simulation Environment

In order to evaluate the performance of the presented Turbo-Fountain realizations we modeled a simplified wireless system and we performed extensive simulations. Usually, in state-of-the-art wireless systems like, e.g., GPRS or UMTS, data to be transmitted is segmented and mapped onto radio blocks. Data segments carried by these radio blocks are optionally protected with an error correcting code, e.g., convolutional code in GPRS. Usually a Cyclic Redundancy Check (CRC) sequence is inserted in order to perform error detection. Various strategies can be performed if residual bit errors are detected within received radio blocks. Usually erroneous radio blocks are declared as erased and are not forwarded to upper layers. Alternatively, the erroneous blocks can be used for further processing, or soft information, i.e., L-values about each bit within the radio

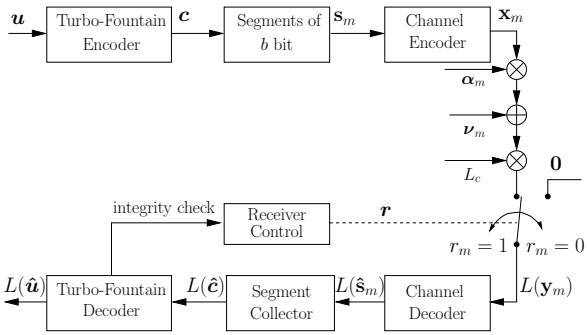

Figure 5: System diagram of the simulation setup.

block, are made available[1].

In the following we introduce a simulation environment, which models the characteristics of segmentation and radio block protection in a simplified but still meaningful way in order to determine the performance of the Turbo-Fountain for asynchronous data access. Fig. 5 shows the block diagram of our simulation environment. The information sequence $\boldsymbol{u}$ of length k bit is encoded with the Turbo-Fountain encoder which outputs the infinite code sequence $\boldsymbol{c}$. After appropriate segmentation, where b bits are grouped to a segment $\mathbf{s}_m \triangleq (s_{m,1}, \ldots, s_{m,b})$, a convolutional encoder with rate $R = b/B$ is applied to encode each segment $\mathbf{s}_m$, resulting in the radio block sequence $\boldsymbol{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m, \ldots)$. The symbols in the encoded radio block sequence are BPSK modulated and transmitted over the channel. 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 radio block m, respectively. The fading coefficients $\boldsymbol{\alpha}_m$ over radio block m depend on the channel: i) $\boldsymbol{\alpha}_m = (1, \ldots, 1)$ for AWGN, ii) $\boldsymbol{\alpha}_m = (\alpha_{m,1}, \ldots, \alpha_{m,B})$ for symbol-wise fading, and iii) $\boldsymbol{\alpha}_m = (\alpha_m, \ldots, \alpha_m)$ for block-wise fading, with Rayleigh distributed channel gains α, i.e., $f_\alpha(\alpha) = 2\alpha \exp\{-\alpha^2\}$, $\alpha > 0$. The additive noise is assumed to be Gaussian, i.i.d., $\nu_{m,i} \sim \mathcal{N}(0, N_0/(2E_s))$, and the channel signal-to-noise ratio (SNR) is defined as E_s/N_0. L-values [9] are obtained after the multiplication with the channel state information $L_c = 4\alpha(E_s/N_0)$. We assume that each receiver has perfect knowledge of the channel gain $\alpha_{m,i}$ and has access to the SNR. As introduced in Section 3.1., the receiver pattern decides which radio blocks $\mathbf{x}_m$ the receiver observes from the channel. Thereby, $r_m = 1$ indicates that the receiver listens to the channel at radio block index m, whereas $r_m = 0$ denotes that the receiver does not listen to the m-th segment. This is controlled by the *receiver control*. The received radio blocks $\mathbf{y}_m$ are decoded with a Max-Log-Map decoder [14], [15] in order to obtain estimates on the segments $\hat{\mathbf{s}}_m$. The segment collector arranges the received segments to a stream of symbols which is passed to the Turbo-Fountain decoder, which outputs $\hat{\boldsymbol{u}} = (\hat{u}_1, \ldots, \hat{u}_k)$ as an estimate of $\boldsymbol{u}$. The decoder indicates decoding success to the receiver control, by observing a successful CRC-check. Let d denote the time index at which decoding is successful, i.e., enough symbols are received. Hence, the receiver control sets $r_i = 0 \ \forall i > d$.

Theoretically, any arbitrary realization of the receiver pattern could be selected. However, it is infeasible to consider all possible random receiver patterns within the simulation since the receiver pattern is of infinite length. Furthermore, many receiver patterns which indeed spread the reception over a finite time period are impractical, though theoretically possible. Hence, we reduce the set of possible receiver patterns. Only receiver patterns which initiate a continuous reception of segments from the channel are selected. This describes a setup where a receiver randomly joins the data reception (asynchronous access) but does not interrupt the reception until it is able to decode the information message. This is the case within a typical wireless scenario without reception interrupts. Within a large number of simulation runs we tracked the minimum number of symbols required to be received at the receiver in order to achieve decoding success. Let $\overline{n}$ denote the mean value of the number of required symbols to be obtained from the channel at a specific E_s/N_0. Hence, we define $k/\overline{n}$ as the average receiver throughput. In order to avoid useless decoding attempts after each symbol reception at the receiver, we initiate the decoding procedure the first time after reception of k bits.

5.2. Simulation Parameters

Before we present the simulation results, we briefly introduce the system parameters. For all performed simulations we selected a radio block size of $B = 160$ bit. We investigate the system performance for two different radio protection modes, uncoded transmission, with $R = 1$, and alternatively coded transmission with code rate $R = 0.5$. In the latter case, we applied a memory $M = 4$ recursive systematic convolutional code with generator polynomial $[37, 21]_8$ in octal representation. The segment size b is adjusted appropriately with the applied code rate R. For the component encoders of both Turbo-Fountain structures, we selected memory $M = 2$ recursive systematic convolutional codes with generator matrix $[5, 7]_8$. Since broadcast solutions with fountain codes should allow transmission of long information messages, at least in the order of kilobytes up to several megabytes, we selected sliding window Max-Log-Map component decoders [14], [15], [16]. A sliding window decoder allows decoding of long code sequences with manageable memory requirements. Furthermore, recently efficient hardware implementations based on analog circuits have been presented [17], which allow decoding speeds upto several Gbit/s [18]. We present performance results for information block length $k = 16000$ bit. Longer information block lengths have also been simulated showing similar results, but are not presented here. The number of performed iterations at the receivers was fixed to $I = 15$. We investigate the

[1] We assume that header information for each received radio block is always correctly available. This can be accomplished by protecting headers with lower code rate, like e.g in EGPRS. We do not consider header overhead in our performance evaluation.

performance on the AWGN, the symbol-wise Rayleigh fading, and the block-wise Rayleigh fading channel.

5.3. Simulation Results

Fig. 6 shows simulation results on the AWGN channel. The average receiver throughput $k/\overline{n}$ is shown versus E_s/N_0. As can be observed, the throughput is increasing with increasing E_s/N_0 for all compared schemes.

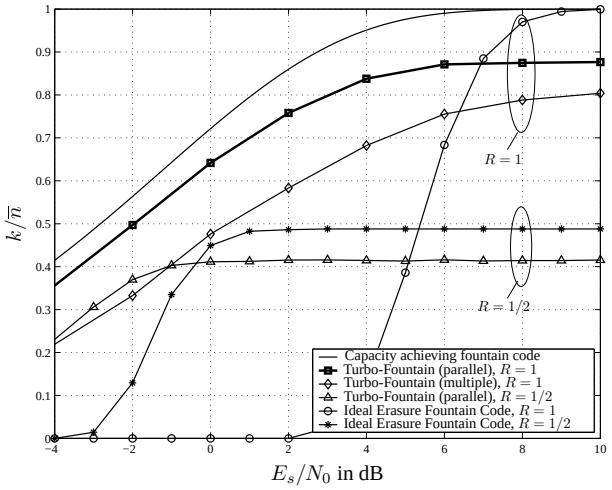

Figure 6: Simulation results on the AWGN channel. Average receiver throughput $k/\overline{n}$ vs. E_s/N_0.

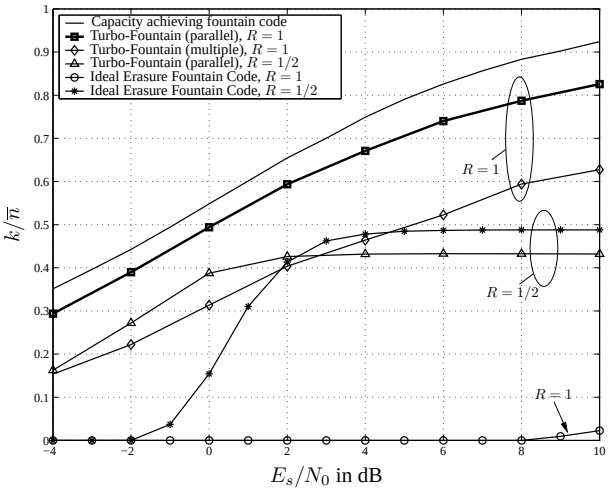

Figure 7: Simulation results on symbol-wise Rayleigh fading channel. Average receiver throughput $k/\overline{n}$ vs. E_s/N_0.

First, we investigated the performance of an Ideal Erasure Fountain Code (IEFC), which serves as an reference system in the following. Erroneously received radio blocks are declared as erased at the receiver, according to Section 5.1. Note, since IEFCs have not been found, we just assume that decoding of the information message would be successful, if k bits were received un-erased and un-punctured at the receiver. This serves as an upper bound for all practical erasure based fountain codes. As can be observed, without additional radio block protection, i.e., $R = 1$, sufficient throughput can only be achieved for $E_s/N_0 > 4$ dB. However, for increasing E_s/N_0 the average throughput reaches $k/\overline{n} = 1$. Applying radio block protection with rate $R = 0.5$, the system is able to support receivers with lower E_s/N_0 as well, since less radio blocks are declared as erased. However, the maximum achievable throughput is limited to $k/\overline{n} = R$ since in a broadcast scenario link adaptation, i.e., swiching of coding schemes, cannot be performed. Unfortunately, an Erasure Fountain Code (EFC) remains a huge gap compared to a fountain code operating at the capacity of the AWGN chanel.

In the contrast, the Parallel Turbo-Fountain, exploiting soft information at the decoder without erasure declaration, in combination with $R = 1$ follows the capacity over the entire investigated region, only with a small remaining gap. For $E_s/N_0 > 7$ dB the IEFC outperforms the Parallel Turbo-Fountain. However, practical EFC codes are below the IEFC bound and furthermore, the point of operation in an wireless systems is expected to be located for many receivers somewhere in the lower E_s/N_0 region. Interestingly, the performance of the Multiple Turbo-Fountain is noticeably weaker, compared to the Parallel Turbo-Fountain, though the encoder and decoder complexity is much higher, incorporating multiple component codes. However, it still shows huge performance gains compared to IEFC for low E_s/N_0. Radio block protection ($R = 0.5$) in combination with Turbo-Fountain does not show any advantages over the entire E_s/N_0 region.

Fig. 7 and Fig. 8 show the simulation results for the symbol-wise Rayleigh fading and the block fading channel, respectively. Basically, the same tendencies as explained for the AWGN channel can be observed. However, for symbol-wise fading, which is a fairly good channel model for fast moving wireless receivers, the IEFC with $R = 1$ cannot provide any throughput since in every radio block, bits are received erroneously with high probability. Hence, EFCs are only practical with appropriate radio block protection. However, now the Parallel Turbo-Fountain without radio block protection outperforms all presented schemes significantly in the entire investigated E_s/N_0 region.

On the block fading channel, the performance of the Turbo-Fountain is mostly equivalent to the performance on the symbol-wise fading channel. This comes from the fact that for both channel types the ergodic capacity is equivalent. However, since with block fading the fading gain varies only from block to block, the IEFC performance is different, but provides a huge gap to the Turbo-Fountain.

5.4. Discussion

As shown in the previous subsection, the Parallel Turbo-Fountain shows impressive results, but still remains a gap to capacity. However, this encourages for future work to close this gap. Interestingly, the performance of the Multiple Turbo-Fountain is worse compared to the Parallel. The reason for this is that the encoder structure outputs systematic information only at the beginning. This indeed reduces the performance since turbo codes require systematic bits in order to achieve good performance, although the collection of

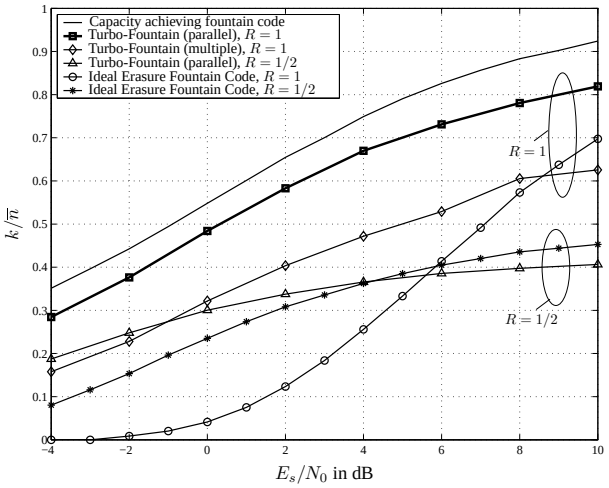

Figure 8: Simulation results on the block-wise Rayleigh fading channel. Average receiver throughput $k/\overline{n}$ vs. signal-to-noise ratio E_s/N_0.

identical code bits is avoided with this structure. On the other hand, the Parallel Turbo-Fountain cannot avoid the collection of identical code bits multiple times at the receiver, due to its interleaved repetition structure. However, the results show that a Digital Fountain can already be very well approximated with an interleaved repetition structure, based on a rate $1/3$ turbo code. In the case of soft decoding, multiple collection of L-values belonging to the same bits are not useless, but increase the likelihood for these bits, which is not the case with erasure based decoding.

The results show that exploiting soft information is inevitable in a wireless environment, which is the case with the Turbo-Fountain. However, many erasure codes allow decoding based on soft information as well, e.g., by belief propagation, but at the expense of increased decoding complexity. This was out of the scope of this work.

6. Conclusions and Future Work

We investigated fountain coding as a solution for the problem of reliable wireless broadcast with asynchronous access. We extended the traditional problem formalization of ideal fountain codes on erasure channels to arbitrary channel types. We introduced the *Turbo-Fountain*, based on turbo codes, as a possible approximation of a fountain code. We presented two different structures of the Turbo-Fountain and compared the performance of the Turbo-Fountain to an ideal erasure based fountain code with appropriate erasure declaration on the AWGN, the symbol-wise Rayleigh fading, and the block-fading channel. We considered decoding based on soft-information from the channel for the Turbo-Fountain. Huge performance gains of the Turbo-Fountain have been shown compared to ideal erasure based codes over the E_s/N_0 region of interest. Moreover, the Turbo-Fountain was shown to operate close to capacity, only with a small remaining gap. We showed that additional radio block protection is not required if the Turbo-Fountain is applied. Since there is a remaining gap to the capacity, future work will consider optimization of the Turbo-Fountain towards capacity. Furthermore, we will extend the system model, in order to consider reception pauses, and will consider more arbitrary receiver patterns.

Acknowledgments

We would like to thank Nicolas Dütsch for making some useful parts of his software platform available.

REFERENCES

[1] J. Nonnenmacher and E. Biersack. Reliable multicast: Where to use forward error correction. *Proc. IFIP 5th Int. Workshop Protocols High-Speed Networks, Sophia Antipolis, France*, pages 134–148, October 1996.

[2] L. Rizzo and L. Vicisano. A reiliable multicast data distribution protocol based on software FEC techniques. *Proc. HPCS*, 1997.

[3] J. Byers, M. Luby, and M. Mitzenmacher. A digital fountain approach to asynchronous reliable multicast. *IEEE Journal on Selected Areas in Communications*, 20(8):1528–1540, October 2002.

[4] M. Luby. LT codes. In *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.

[5] A. Shokrollahi. Raptor codes. Technical Report DR2003-06-001, Digital Fountain, Jun. 2003.

[6] M. Mitzenmacher. Digital fountains: A survey and look forward. *Proc. of the IEEE Information Theory Workshop 2004, San Antonio, TX, USA*, pages 271–276, October 2004.

[7] R. Palanki and J. Yedidia. Rateless codes on noisy channels. *Proc. International Symposium on Information Theory (ISIT) 2004, Chicago, IL, USA*, page 37, June 2004.

[8] P. Maymounkov and D. Mazieres. Rateless codes and big downloads. *Proc. of the 2nd International Workshop on Peer-to-Peer Systems*, February 2003.

[9] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Transactions on Information Theory*, 42(2):429–445, March 1996.

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

[11] J. Hagenauer. The turbo principle in mobile communications. *Proc. International Symposium on Nonlinear Theory and its Applications, Xi'an, China*, October 2002.

[12] D. Divsalar and F. Pollara. Multiple turbo codes for deep-space communications. *TDA Progress Report 42-121*, pages 66–77, May 1995.

[13] C. He, D. Costello, A. Huebner, and K. Zigangirov. Joint interleaver design for low complexity multiple turbo codes. *ITW2003, Hong Kong*, July 2003.

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

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

[16] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. Soft-output decoding algorithms in iterative decoding of turbo-codes. *JPL TDA Progress Report*, pages 42–124, February 1996.

[17] M. Moerz. Analog sliding window decoder core for mixed signal turbo decoder. *Proc. of the 5th International ITG Conference on Source and Channel Coding (SCC'04), Erlangen, Germany*, January 2004.

[18] M. Moerz. Analog turbo-decoding in VLSI. *PhD Thesis, Munich University of Technology, to be submitted*, February 2005.