Chapter 11 Fountains Code for Erasure Channels

11.1 Introduction of Erasure Channels and Retransmission Mechanism

Channels with erasures are of great importance. For example, message files, static images or videos sent over the internet are first chopped into packets, and each packet is finally either received without error or not received in the terminal. Noisy channels to which good error-correcting codes have been applied also behave like erasure channels: In medium or high SNRs, the error-correction code performs perfectly which means that the decoder can recover almost all the received symbols corrupted by noise; while in low SNRs the decoder often fails and reports that it has been unsuccessful, so the receiver realizes that the whole packet has been lost.

Erasure channel and retransmission mechanism

A simple q-ary erasure channel is shown in Fig. 11.1, which has the input alphabet {0, 1, 2, …, $q-1$} with a probability $1-p$ of transmitting the input without error, and probability p of resulting in the output "?" representing an erasure symbol/packet. The alphabet size q is 2^l, where l is the number of bits in a packet.

The general methods for communicating over such channels employ a feedback channel from receiver to sender that is used to control the retransmission of erased packets. Basically, there are two protocols in realization of the retransmission mechanism.

(1) The receiver might send back messages that identify the missing packets, which are then retransmitted.

(2) Alternatively, the receiver might send back messages that acknowledge each received packet, the sender keeps a track of which packets have been acknowledged and retransmits the others until all packets have been acknowledged.

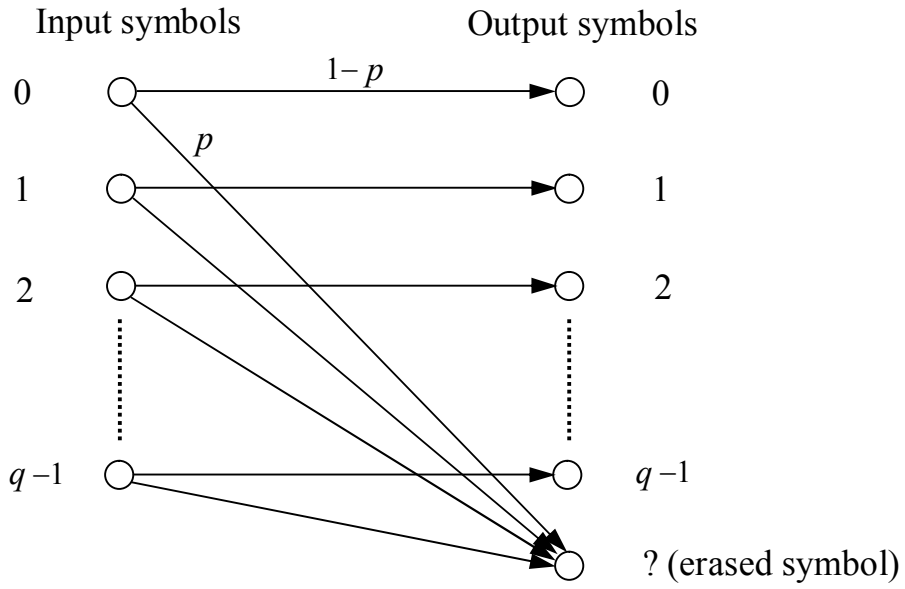


Fig. 11.1. An erasure channel with input alphabet $\{0, 1, \ldots, q-1\}$ and a probability $1-p$ of transmitting the input without error and probability p of delivering the output "?" representing an erasure.

Problems addressed

These simple retransmission protocols have the advantage that they will work regardless of the erasure probability p, but theorists who learned the renowned Shannon theory may feel that these protocols are wasteful. If the erasure probability p is large,

the number of feedback messages sent by the first protocol will be large. Under the second protocol, it is more likely that the receiver will end up by receiving multiple redundant copies of some packets through the heavy use of the feedback channel.

From another prospective view, according to Shannon Theorem, the error-free reliable transmission can be effectively realized with the help of an appropriate forward error-correction code provided that the code rate is less than the forward erasure channel capacity, namely, $C_E=(1-p)l$ bits per channel symbol, where l is the number of bits in a packet (or symbol). This conclusion is proved in the next section. Therefore, there is no need for the feedback channel for reliable transmission in theory.

The wastefulness of the simple retransmission protocols is especially evident in the case of a broadcast channel with erasures, where one sender broadcasts to many receivers and each receiver receives a random fraction $(1-p)$ of the packets. If every packet, which is missed by one or more receivers, has to be retransmitted, those retransmissions will be terribly redundant and lead to an interesting scene that every receiver will receive most of the retransmitted packets. Therefore, we would like to make erasure-correcting codes that require no feedback or almost feedback.

Reed-Solomon codes are classic block codes for erasure correction [1][2]. An (N, K) Reed-Solomon over Galois field $GF(q)$ ($q=2^l$) has the ideal property that if any K of the N transmitted symbols are received then the original K source symbols can be recovered. However, Reed-Solomon codes have the disadvantage that they are practical

only for small K, N and q. The standard implementations of encoding and decoding have a cost of order $K(N-K)\log_2 N$ symbol operations [1]. Furthermore, with a Reed-Solomon code, as with any block code, one must estimate the erasure probability p and choose the code rate $R=K/N$ before transmission. If unfortunately p is larger than expected and the receiver receives fewer than K symbols, the decoder cannot recover the K source symbols. Therefore, we would like a relatively simple way to extend the code on the fly to create a low or very low rate (N', K) code, which is pioneered by Michael Luby [3][4] and the team members of Digital Fountain (DF) Inc.

11.2 The Capacity of General *q*-ary Erasure Channel

Consider the q-ary erasure channel as shown in Fig. 11.1, let the input symbol $X \in \{0, 1, 2, \ldots, q-1\}$ and output symbol $Y \in \{0, 1, 2, \ldots, q-1, ?\}$. Then the average mutual information of X and Y is

$$\begin{aligned} I(X,Y) &= H(Y) - H(Y \mid X) \\ &= \sum_{i=0}^{q-1} \Pr\{Y=i\} \log \frac{1}{\Pr\{Y=i\}} + \Pr(Y=\text{"?"}) \log \frac{1}{\Pr(Y=\text{"?"})} \\ &\quad - \sum_{i=0}^{q-1} \Pr(Y=i \mid X=i) \Pr(X=i) \log \frac{1}{\Pr(Y=i \mid X=i)} \\ &\quad - \sum_{i=0}^{q-1} \Pr(Y=\text{"?"} \mid X=i) \Pr(X=i) \log \frac{1}{\Pr(Y=\text{"?"} \mid X=i)} \end{aligned} \tag{11-1}$$

where

$$\Pr(Y=i)=\sum_{j=0}^{q-1}\Pr(Y=i\mid X=j)\Pr(X=j)$$

$$=\Pr(X=i)\Pr(Y=i\mid X=i)$$

$$=(1-p)\Pr(X=i) \tag{11-2a}$$

$$\Pr(Y="?")=\sum_{i=0}^{q-1}\Pr(Y="?"\mid X=i)\Pr(X=i)$$

$$=p\sum_{i=0}^{q-1}\Pr(X=i)=p \tag{11-2b}$$

Substituting (11-2a) and (11-2b) into (11-1), it yields

$$I(X,Y)=\sum_{i=0}^{q-1}(1-p)\Pr(X=i)\log\frac{1}{(1-p)\Pr(X=i)}+p\log\frac{1}{p}$$

$$-\sum_{i=0}^{q-1}(1-p)\Pr(X=i)\log\frac{1}{1-p}-p\log\frac{1}{p}\sum_{i=0}^{q-1}\Pr(X=i)$$

$$=\sum_{i=0}^{q-1}(1-p)\Pr(X=i)\left[\log\frac{1}{1-p}+\log\frac{1}{\Pr(X=i)}\right]+p\log\frac{1}{p}$$

$$-\sum_{i=0}^{q-1}(1-p)\Pr(X=i)\log\frac{1}{1-p}-p\log\frac{1}{p}$$

$$=(1-p)\sum_{i=0}^{q-1}\Pr(X=i)\log\frac{1}{\Pr(X=i)} \tag{11-3}$$

Evidently, the channel capacity is

$$C_E=\max_{\Pr(X)}I(X,Y)=(1-p)\log q \tag{11-4a}$$

where the capacity C_E is achieved under the condition of equal probability for each discrete input symbol. If $q=2^l$ and using base-2 logarithm, then we obtain

$$C_E = (1-p)\log_2 q = (1-p)l \quad \text{bits/channel symbol} \tag{11-4b}$$

Furthermore, if base-q logarithm is adopted, then we get

$$C_E = (1-p)\log_q q = 1-p \quad \text{symbol/channel symbol} \tag{11-4c}$$

Example 11-1: Let the erasure probability $p=1/4$ for an erasure channel and suppose that we transmit 10000 packets (or symbols). Hence, approximate 7500 packets can be received successfully and 2500 packets lost (or erased).

(a) If we use a systematic block code of rate 3/5 less than $C_E = 1-p = 3/4$, then it results in 6000 packets that consist of information bits. By an appropriate decoding approach, we can correctly infer all the information packets from the aforementioned 7500 received packets with error-free.

(b) If we adopt a systematic block code of rate 4/5 greater than $C_E = 3/4$, then it yields 8000 information packets correspondingly. Evidently, the decoder cannot correctly infer these information packets from the 7500 received packets by any decoding method. □

11.3 The General Random Linear Fountain

An Interesting Metaphor

The encoder of a fountain code is a metaphorical fountain that produces an endless supply of the water drops, i.e., encoded packets. Suppose that the original source file has a size of Kl bits, and each drop contains l encoded bits. Interestingly, if anyone who

wishes to receive the encoded file, he/she just holds a bucket under the fountain and collects drops until the number of drops in the buckets is a little larger than K, which can be used to recover the original file.

Limitless and Rateless of Fountain Codes

Fountain codes are rateless in the sense that the number of encoded packets generated from the source message is potentially limitless. In other words, the number of encoded packets generated can be determined on the fly without any restrictions. For an erasure channel with any erasure probability p, the fountain encoder will produce as many coded packets as needed based on the K source (or input) packets in order to guarantee that the code rate R is less than the erasure channel capacity C_E given by (11-4c),

$$R < C_E = 1 - p \tag{11-5}$$

Suppose the number of coded packets transmitted over the erasure channel be N. Hence, $K = RN$. For the large N, the number K' of the received packets is

$$K' \approx C_E N > RN = K \tag{11-6}$$

Therefore, the subsequent task is to design a decoding strategy that can decode the K source packets from the K' received packets.

11.3.1 Encoding of the Linear Fountain Codes

Consider an encoder for a file of size K packets, s_1, s_2, …, s_K. A packet is the elementary unit that is either transmitted or erased by the erasure channel. Let the K-

dimensional row vector $\mathbf{S}$ associated with K source packets, $K \times N$ generator matrix $\mathbf{G}$ and N-dimensional column vector $\mathbf{T}$ related to N coded packets be given as follows

$$\mathbf{S}=[s_1, s_2, ..., s_K] \tag{11-7a}$$

$$\mathbf{G}=\begin{bmatrix} g_{11} & g_{12} & \cdots & \cdots & g_{1N} \\ g_{21} & g_{22} & \cdots & \cdots & g_{2N} \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ g_{K1} & g_{K2} & \cdots & \cdots & g_{KN} \end{bmatrix} \tag{11-7b}$$

$$\mathbf{T}=[t_1, t_2, ..., t_N] \tag{11-7c}$$

For binary case, g_{kn} ($1 \le k \le K$, $1 \le n \le N$) belongs to GF(2), s_k and t_n are source and coded packets, respectively, which also consist of binary digits from GF(2). The linear fountain is formed by

$$\mathbf{T}=\mathbf{SG} \tag{11-8}$$

where the bitwise sum is performed by mod-2 addition for the bits in the corresponding packets. After the coded packets traveling through the erasure channel, some of the transmitted packets are successfully received, but some are erased, which is equivalent to changing the generator matrix $\mathbf{G}$ used in the transmitter into another generator matrix $\mathbf{G}'$ processed by the receiver.

Example 11-2: Consider a generator matrix $\mathbf{G}$ of a linear fountain as follows

$$\mathbf{G}=\begin{bmatrix}
1&1&0&0&0&1&1&0&1&1&0&1&0&0&1&1&0&1&1&0&1&1&0&0\\
1&1&1&1&0&0&1&0&0&0&1&1&1&1&1&0&0&1&0&0&1&1&1&0\\
0&0&1&1&0&0&1&1&1&0&0&1&0&0&0&1&1&1&0&0&0&1&0&1\\
0&1&0&1&1&0&0&0&1&1&1&1&0&1&1&0&1&0&1&0&0&0&1&1\\
1&1&0&0&0&1&0&0&1&1&0&0&1&0&1&0&1&1&0&1&0&1&0&0\\
0&1&1&1&0&0&1&0&1&0&1&0&1&1&1&0&0&1&1&1&1&0&0&0\\
0&0&0&1&1&1&0&0&0&0&1&1&1&1&1&1&0&0&1&1&0&1&0&1\\
1&0&0&0&1&1&0&0&1&0&1&1&1&1&1&1&0&0&0&1&0&1&1&1\\
1&0&1&0&0&1&1&1&1&1&0&1&0&0&0&0&1&1&1&1&1&1&0&1\\
0&1&0&0&1&0&1&0&0&1&1&1&1&0&0&0&1&1&0&1&0&1&0&0\\
1&0&0&1&1&1&0&0&1&0&0&1&1&0&0&1&1&1&1&1&0&0&1&1\\
1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0
\end{bmatrix} \tag{11-9a}$$

which indicates that the source generates K=12 packets and the transmitter will send N=24 packets over the erasure channel.

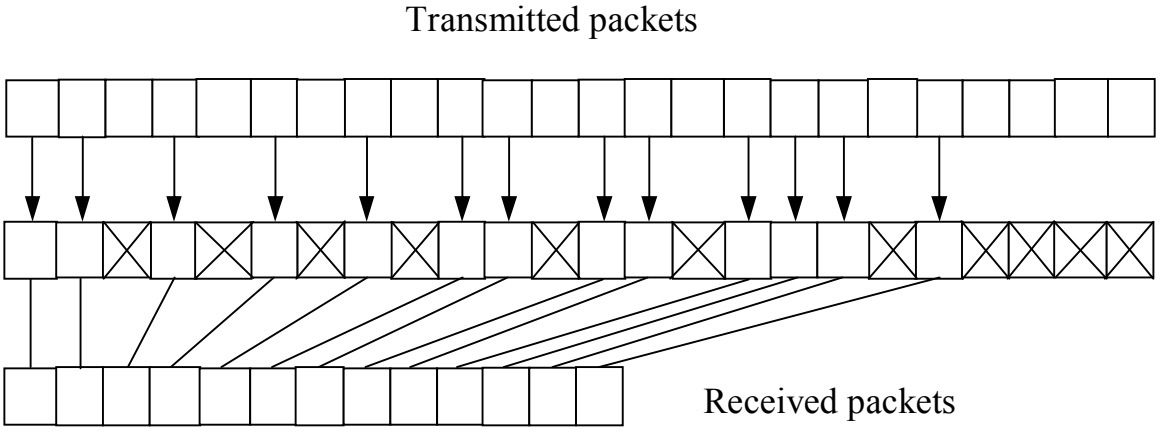


Fig. 11.2. When packets are transmitted based on the generator matrix **G** in (11-9a), some of them are not received due to the erasure as shown by the packets with crosses.

The transmitted packets formed by the generator matrix $\mathbf{G}$ in (11-9a) and the received packets are shown in Fig. 11.2. Evidently, we can realign the columns of $\mathbf{G}$ in order to define a new generator matrix $\mathbf{G}'$ from the viewpoint of the received packets, simply delete the columns of $\mathbf{G}$, which are corresponding to the erased packets. From Fig. 11.2 the number of the received packets is K'=13, thus $\mathbf{G}'$ is given as

$$\mathbf{G}' = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(11-9b)}$$

□

11.3.2 Decoding of the Linear Fountain Codes

As presented from the previous analysis in (11-5) and (11-6), when the number of received packets K' is larger that of the source packets K, then if there exists K independent column vectors from the total K' column vectors in the equivalent generator matrix $\mathbf{G}'$. These K vectors forms a K-by-K matrix $\mathbf{G}''$, which is invertible

and associated with a K-dimensional row vector $\mathbf{T}^{''}$ made up of K received packets selected for decoding of the source packets s_i (i=1, 2, …, K). Thus, we have

$$\mathbf{T}^{''}=\mathbf{S}\mathbf{G}^{''} \tag{11-10a}$$

Equivalently,

$$\mathbf{S}=\mathbf{T}^{''}(\mathbf{G}^{''})^{-1} \tag{11-10b}$$

For the last example, the matrix $\mathbf{G}^{''}$ can be formed by choosing the first 12 column vectors of $\mathbf{G}^{'}$ in (11-9b). Generally, as the fountain codes are rateless (very low code rate), we can easily find a K-by-K matrix $\mathbf{G}^{''}$ that is a sub-matrix of $\mathbf{G}$ (or $\mathbf{G}^{'}$) and also related to the K received packets from a large amount of received candidates.

11.4 The LT Codes and LT Process

Fountain codes are a class of sparse-graph codes that have received considerable attention [5][6][7] in the last few years. The first fountain codes were the erasure-correcting codes introduced by Luby [3] based on the Luby transform (LT), therefore, we give the name of this sort of codes as LT codes. The LT code retains the good performance of the random linear fountain code, while drastically reducing the encoding and decoding complexities.

In this section, we will first introduce LT encoder, then LT decoder, and finally propose an interesting balls and pins problem that is crucial for understanding and analyzing the degree distribution of LT codes.

11.4.1 LT Encoder

For an LT code, each encoded packet t_n is produced from the source packets s_1, s_2, s_3, …, s_K by the following two steps:

Step 1: Randomly choose the degree d_n of the encoded packet t_n from a degree distribution $\rho(d)$, the appropriate choice of $\rho(d)$ depends on the source file size K, which will discuss in the later section.

Step 2: Choose, uniformly at random, d_n distinct input packets, and set t_n equal to the mod-2 bitwise sum of those d_n packets.

This encoding operation defines a graph connecting encoded packets to source packets to source packets. Note that if the mean degree $\overline{d}$ is significantly smaller than K, then the graph is sparse. We can think of the resulting codes as an irregular low-density generator-matrix (LDGM) code as compared to well-known bit-based low-density parity-check (LDPC) codes of regular or irregular types.

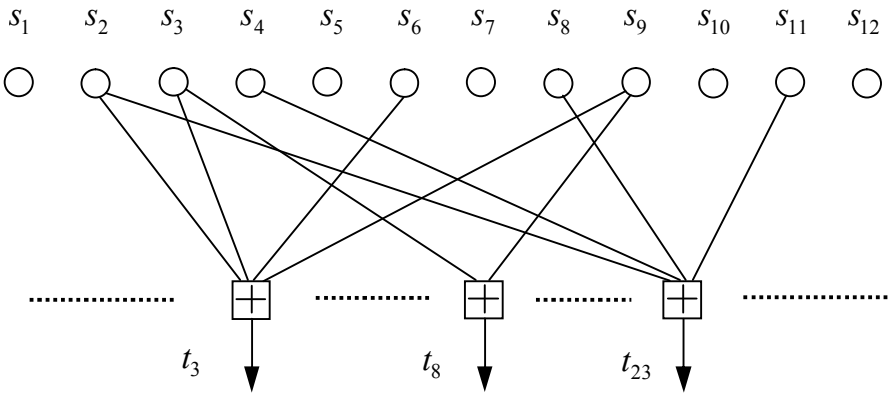


Fig. 11.3. The graph representation of three encoded packets t_3, t_8 and t_{23} in Example 11-2.

Example 11-3: Consider the generator matrix **G** in (11-9a) for the fountain code in Example 11-2. The graph representation of three encoded packets t_3, t_8 and t_{23} is depicted in Fig. 11.3 in terms of the source packets s_1, s_2, s_3, …., s_{12}. □

11.4.2 LT Decoder

The decoding of a sparse-graph code is relatively easy in the case of an erasure channel. The decoder's task is to infer the vector **S** of source packets from the vector $\mathbf{T}''$ of the received packets and the equivalent generator matrix $\mathbf{G}''$, where the relationship is explicitly expressed in (11-10a) and (11-10b). Note that in order to finalize the matrix $\mathbf{G}''$ from the generator matrix **G** used by the sender, the receiver must be capable of knowing the sequence of a received packet. Usually, this can be easily realized by appending a unique order to each transmitted packet.

A simple way to attempt to solve this problem is by message passing. But in the case of erasure channel, all the transmitted packets are either completely uncertain or completely certain in the receiver. Uncertain packet implies that a packet t_n could have any value of a candidate with equal probability, while certain packet means that a packet t_n has a particular value with probability one, which is exactly corresponding to the source packet s_n. We will henceforth call the encoded packets t_n check nodes.

A simple description of the decoding process is made by the following two steps:

Step 1: Find a check node t_n that is connected to only one source packet s_k. Note that if there is no such check node, this decoding algorithm halts at this point, and fails to recover all the source packets.

(a) Set $s_k = t_n$.

(b) Add s_k to all checks $t_{n'}$ ($n' \neq n$) that are connected to s_k. Let

$$t_{n'} := t_{n'} + s_k \tag{11-11}$$

for all n' such that $g_{kn'}=1$, where mod-2 sum is performed.

(c) Remove all the edges connected to the source packet s_k.

Step 2: Repeat *Step* 1 until all s_k are determined.

Further remarks on this algorithm:

(1) All the check nodes t_n defined are associated with the received packets, and all the packets erased by the channel cannot be taken into account in the decoding.

(2) In (a) of *Step* 1, the source packet s_n is given the value of t_n since t_n is only connected with s_k. In other words, t_n is the linear combination of only one source packet s_n.

(3) Further explanations for *Step* 1 are explicitly demonstrated in Fig. 11.4, where we can easily obtain

$$s_k = t_n \tag{11-12}$$

The simple process in (11-12) is shown in Fig. 11.4(a). For the check node $t_{n'}$ that connects three source packets s_k, $s_{k'}$ and $s_{k''}$, respectively, it result in

$$t'_n = s_k + s_{k'} + s_{k''} \tag{11-13a}$$

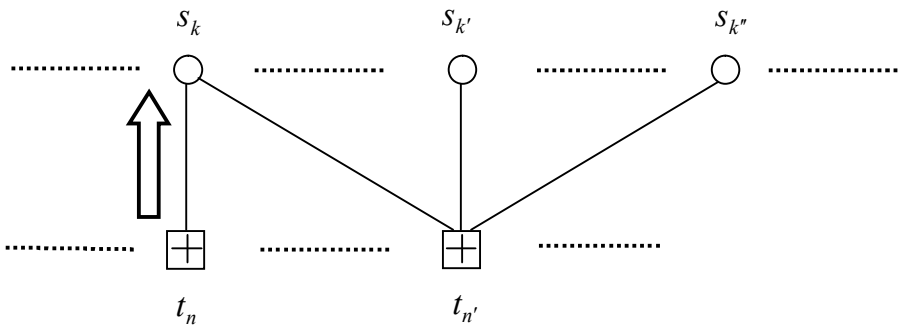
Adding s_k at both sides of (11-13a), we thus obtain

$$\underbrace{s_k + t'_n}_{Updated\ t'_n} = (s_k + s_k) + s_{k'} + s_{k''} \tag{11-13b}$$

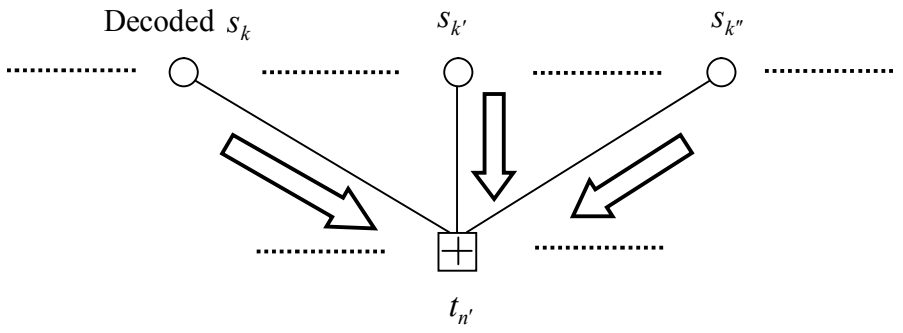
The process by (11-13b) is depicted by Fig. 11-4(b). Now the updated check node t'_n is

$$t'_n := s_k + t'_n = s_{k'} + s_{k''} \tag{11-13c}$$

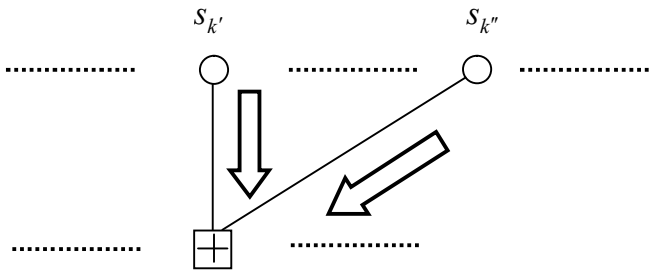
Therefore, it is equivalent to removing the edge that is connected to the source packet s_k as shown in Fig. 11.4(c).



(a)



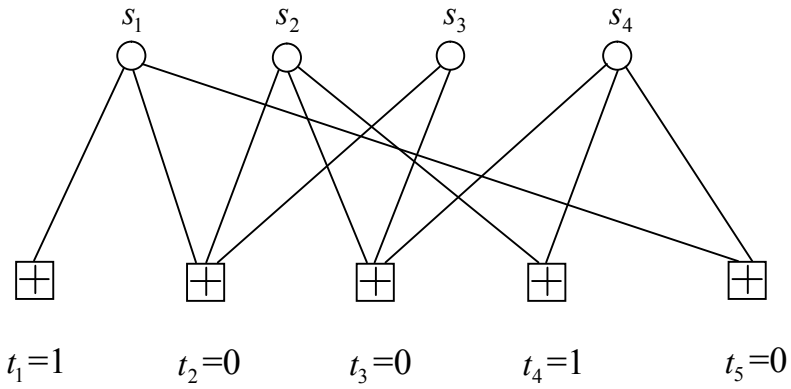
(b)



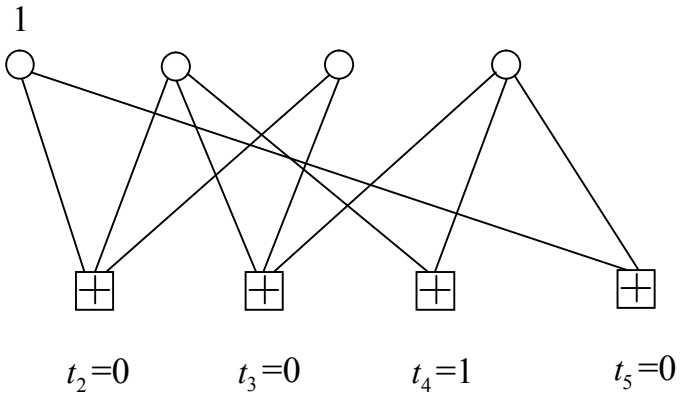
(c)

Fig. 11.4. The decoding process for *Step* 1 of the algorithm. (a) Set $s_k = t_n$. (b) Add s_k to the check node $t_{n'}$ that is connected to s_k. (c) Remove the edge connected to the source packet s_k.

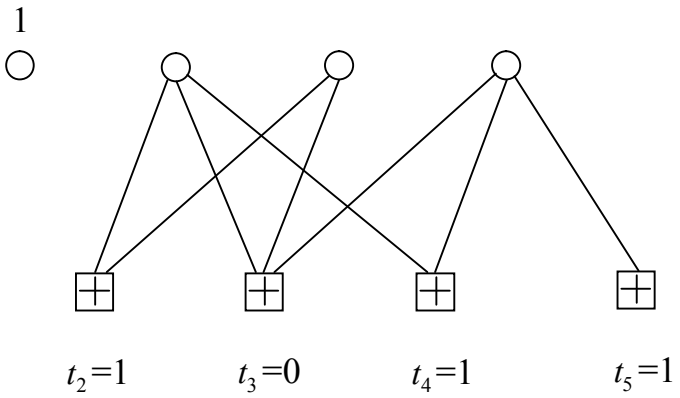
Example 11-4: Consider a very simple case shown in Fig. 11.5 where each packet is merely one bit. There are four source packets as shown by the upper circles and four received packets as shown by the lower check symbols, which has the values t_1, t_2, t_3, t_4=10010 at the start of the algorithm.



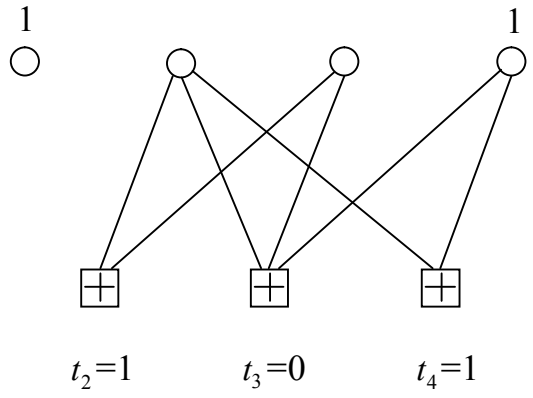
(a)



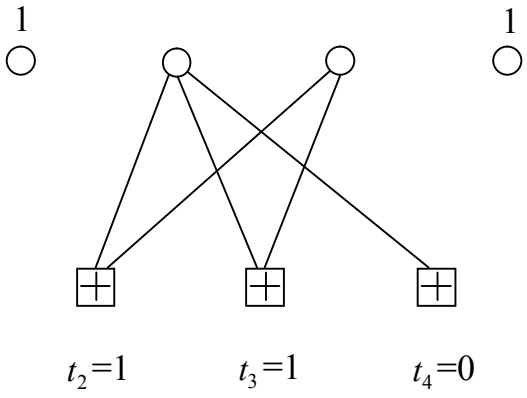
(b)



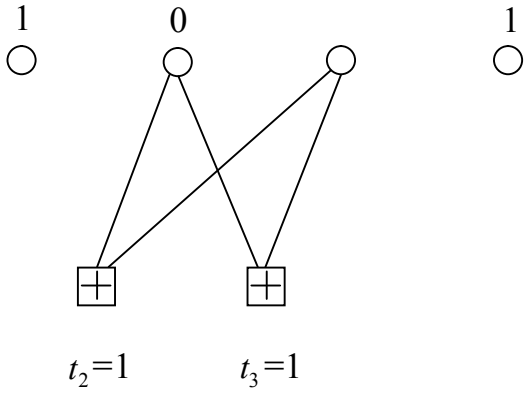
(c)



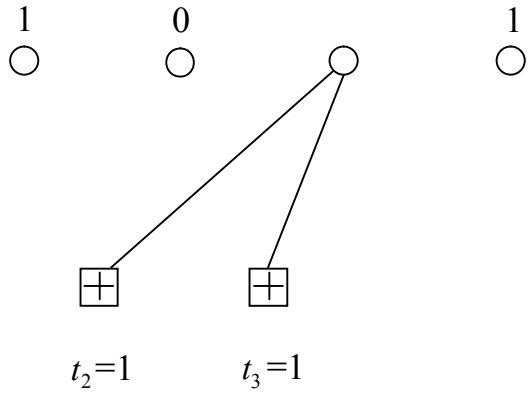
(d)



(e)



(f)



(g)

(h)

Fig. 11.5. An example of decoding for a fountain code with K=4 source bits and N=5 encoded bits.

At the first iteration, the only check node that is connected to a sole source bit (one bit packet) is the first check node t_1 shown in Fig. 11.5(a). Thus, we set $s_1 = t_1 = 1$ accordingly as depicted in Fig. 11.5(b), discard the check node t_1, then add the value of s_1=1 to the checks (t_2 and t_5) to which it is connected, and disconnecting s_1 from the graph as shown in Fig. 11.5(c). This ends the first iteration.

At the start of the second iteration, the fifth check node t_5 is connected to a sole source bit s_4. Then we set s_4=t_5=1, which has been updated in the first iteration, and also add

s_4 to the two checks (t_3 and t_4) it connects to in Fig. 11.5(d). Disconnecting s_4 from the graph, so far we have estimated the values of source bits s_1 and s_4 as shown in Fig. 11.5(e).

Using the same decoding process, we can determine the value of the source bit s_2=0. Finally, we find that two check nodes t_2 and t_3 are both connected to s_3 as shown in Fig. 11.5(g), and they agree with the value of s_3 as we exactly hope. Therefore, all the source bits are finalized and depicted in Fig. 11.5(h). □

Now we can summarize the general decoding rule for LT code as follows:

Decoding recovery rule:

If there is at least one coded symbol (packet) that has exactly one neighbor then the neighbor can be recovered immediately since it is a copy of the coded symbol. The value of the recovered input symbol is employed by an operation of exclusive-or into any remaining coded symbols that also have that input symbol as a neighbor, the recovered input symbol is removed as a neighbor from each of these coded symbols and the degree of each such coded symbol is decreased by one to reflect this removal.

11.4.3 The Balls and Pins Problem

Before introducing the Luby transform (LT) and designing a good degree distribution for LT codes, it will be helpful to consider an interesting problem of balls and pins.

Image that we throw N balls independently at random into K bins ($K<N$), where N and K are large numbers, such as 1000 or 10000. Fig. 11.6 illustrates this relationship between balls (coded symbols) and bins (input symbols).

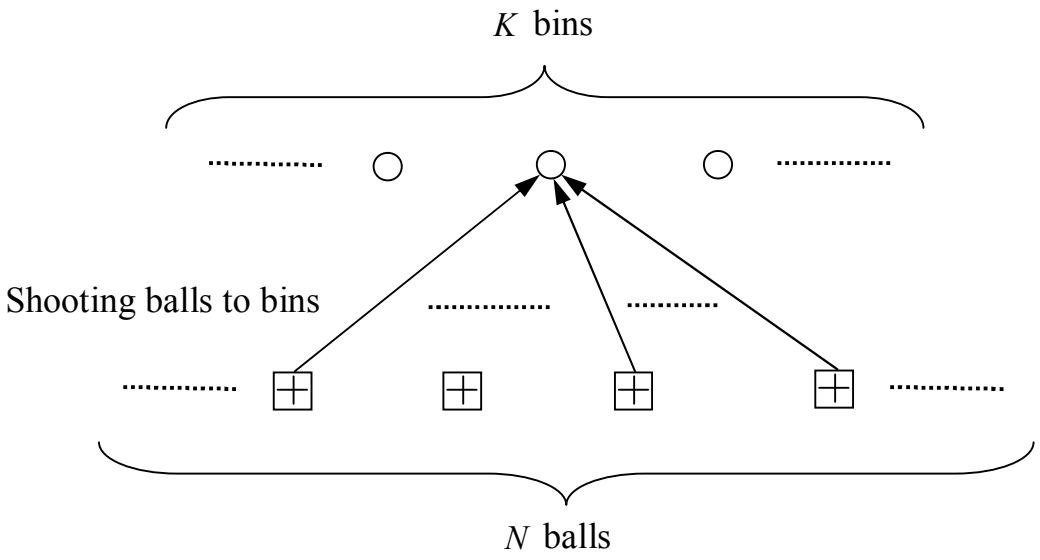


Fig. 11.6. The illustrative diagram for bins and balls. The N balls are thrown into the K bins.

For each ball, the probability of throwing a ball into one particular bin is equal likelihood as $1/K$. Thus, the probability that one particular bin is empty after N balls have been thrown is

$$\Pr\{\text{A bin is empty after } N \text{ shootings.}\}=\left(1-\frac{1}{K}\right)^{N} \tag{11-14}$$

Since the large number K, (11-14) can be approximated as

$$\left(1-\frac{1}{K}\right)^{N}=\left(1-\frac{1}{K}\right)^{-K(-N/K)}\approx e^{-N/K} \tag{11-15}$$

When $N=K$ the probability that one particular bin is empty is roughly $1/e$. Equivalently, the fraction of empty bins must be also roughly $1/e$. If we throw a total of $3K$ balls, the empty fraction drops to $1/e^3$. We have thrown a lot of balls to make sure all the bins have at least a ball. For a general N, the expected number of empty bins is

$$Ke^{-N/K} \tag{11-16}$$

The desirable scene is that this expected number is a very small number δ. Thus, we have

$$Ke^{-N/K} < \delta \tag{11-17}$$

which results in

$$N > K\ln(K/\delta) \tag{11-18}$$

This implies that we need to throw a lot of balls in order to achieve the goal of every bin with at least one ball in a high probability, which is roughly as $1-\delta$ [3].

11.4.4 LT Process

In the analysis of the LT codes, coded symbols are analogous to balls and input symbols are analogous to bins, and the process succeeds if at the end all input symbols are covered. Therefore, throwing a ball into a bin implies that the corresponding input symbol participates in the mod-2 sum in the outcome of the coded symbol. Now we present the steps of the following LP process.

LT process:

(**1**) All input symbols are initially uncovered, i.e., no value has been assigned to any input symbol.

(**2**) At the first step all coded symbols with one neighbor are released to cover their unique neighbors. The set of covered input symbols that have not yet been processed is called the *ripple* [3].

(**3**) At each subsequent step one input symbol in the ripple is processed. It is removed as a neighbor from all coded symbols that have it as a neighbor. All the coded symbols that have exactly one remaining neighbor are subsequently released to cover their remaining neighbor.

(**4**) Some of these neighbors may have been previously uncovered, causing the ripple to grow, while others of these neighbors may have already been in the ripple, causing no growth in ripple.

(**5**) The process ends when the ripple is empty at the end of some step, while the process fails if there is at least one uncovered input symbol at the end. The process succeeds if all input symbols are covered by the end.

Further comments on the LT process:

(**1**) In the classic balls and bins process, all the balls are thrown at once. Since the high probability of collisions, i.e., multiple balls covering the same bin, thus many more balls must be thrown to cover all bins, which is given by (11-18) to show the large number of N in case of small δ.

(**2**) In contrast, the proper design of the degree distribution ensures that the LT process releases coded symbols incrementally to cover the input symbols.

(**3**) Because the neighbors of a coded symbol are chosen at random independently of all other coded symbols, it is easy to image that the step, at which a particular coded symbol is released, is independent of all the other coded symbols.

(**4**) Once a coded symbol is released, it covers a uniformly chosen input symbol among the unprocessed input symbols, independent of all other coded symbols.

(**5**) The sum of the degrees of these coded symbols corresponds exactly to the total number of symbol operations needed to recover the data.

(**6**) These statistical properties make the LT process especially amendable to the design of good degree distributions and analysis of the process with respect to these distributions.

11.5 LT Degree Distributions

As we have known from previous chapters that the degree distributions of check and variable nodes dominate the performance of LDPC codes over an AWGN channel. Here the degree distribution of coded symbols also plays a key role to ensure success of LT process and recover the entire data.

Definition 11-1: (degree distribution): For all d, $\rho(d)$ is the probability that an coded symbol has degree d. □

According to the above analysis, the random behavior of the LT process is completely determined by the degree distribution $\rho(\cdot)$, the number of coded symbols N, and the number of input symbols K, respectively. The objective is to design degree distributions that meet the following design criteria.

(a) As few coded symbol as possible on average are required to ensure success of the LT process. Recall that the number of coded symbols that ensure success of the LT process corresponds to the number of coded symbols that ensure complete recovery of the data.

(b) The average degree of the coded symbols is as low as possible. The average degree is the number of symbol operations on average it takes to generate a coded symbol. The average degree multiplies N is the number of symbol operations on average it takes to recover the entire data.

The classical process of throwing balls into bins can be viewed as a special case of the LT process where all coded symbols have degree one (All-At-Once) [3] and thus are all released and thrown initially.

Definition 11-2: (All-At-Once distribution): $\rho(1)=1$. □

The analysis of the classical balls and bins process implies that $K\ln(K/\delta)$ coded symbols are needed to cover all K input symbols with probability at least $1-\delta$ with respect to the All-At-Once distribution. This result also means that for any distribution the sum of the degrees of the coded symbols must be at least $K\ln(K/\delta)$ to cover all input symbols at least once. Evidently, this number is unacceptable $\ln(k/\delta)$ times larger

than the minimum possible number that is $N=K$, i.e., K balls exactly throwing into K distinct bins.

In the subsequent section we will introduce the soliton distribution that ensures that both the number of coded symbols and the total number of symbol operations are near minimal.

11.5.1 The Ideal Soliton Distribution

The basic property required of a good degree distribution is that input symbols are added to the so-called ripple at the same rate as they are processed. This property is the inspiration for the name soliton distribution [3].

A desired effect is that as few released coded symbols as possible cover an input symbol that is already in the ripple. The reason is that the released coded symbols that cover the same input symbols already in the ripple are redundant. This implies that the ripple size should be small at all times to keep such coded symbols as few as possible. On the other hand, if the ripple vanishes before all K input symbols are covered, then the overall process ends in failure. The ripple size should be kept large enough to ensure that the ripple does not disappear prematurely. The desired behavior is that the ripple size never gets too small or too large.

The soliton distribution displays ideal behavior in terms of the expected number of coded symbols needed to recover the data, which is given as follows:

Definition 11-2: (Ideal soliton distribution) The ideal soliton distribution is $\rho(1)$, $\rho(2)$, …, $\rho(k)$, where

(1) $\rho(1)=1/K$

(2) For all $d=2, 3, \ldots, K$, $\rho(d)=\dfrac{1}{d(d-1)}$

where $\rho(d)$ is the proportion of the coded symbols that have degree d. □

Properties of the ideal soliton distribution:

(a) Obviously, we have

$$\sum_d \rho(d)=\frac{1}{K}+\sum_{d=2}^{K}\frac{1}{d(d-1)}=\frac{1}{K}+\sum_{d=2}^{K}\left(\frac{1}{d-1}-\frac{1}{d}\right)$$

$$=\left(1-\frac{1}{2}\right)+\left(\frac{1}{2}-\frac{1}{3}\right)+\ldots..+\left(\frac{1}{K-1}-\frac{1}{K}\right)+\frac{1}{K}=1 \qquad (11\text{-}19)$$

(b) The expected degree $\bar{d}$ of a coded symbol is

$$\bar{d}=E(d)=\sum_{d=1}^{K} d\,\rho(d)=\sum_{d=2}^{K}\frac{1}{d-1}+\frac{1}{K}$$

$$=1+\frac{1}{2}+\frac{1}{3}+\ldots..+\frac{1}{K-1}+\frac{1}{K} \qquad (11\text{-}20)$$

which is the harmonic sum up to K. Note that $\bar{d}\approx\ln K$ if K is sufficiently large.

(c) An efficient way to choose a sample distributed by the ideal soliton distribution is to first select a random real value $v\in(0, 1]$, and then for $i=2, 3, \ldots, K$, let the sample value be i if $1/i<v\le 1/(i-1)$, and let the sample value be 1 if $0<v\le 1/K$.

Since the probability that the random variable v falls into the interval $(1/i,\ 1/(i-1)]$ (i =2, 3, …, K) is

$$\Pr\{1/i < v \le 1/(i-1)\} = \int_{1/i}^{1/(i-1)} dv$$

$$= \frac{1}{i-1} - \frac{1}{i} = \frac{1}{i(i-1)} \tag{11-21a}$$

and the probability that v belongs to the interval $(0,\ 1/K]$ is

$$\Pr\{0 < v \le 1/K\} = \int_{0}^{1/K} dv = \frac{1}{K} \tag{11-21b}$$

Therefore, the aforementioned selection is reasonable.

The relationship between the ideal soliton distribution and All-At-Once distribution

Note that for the ideal soliton distribution the sum of the degrees of K coded symbols is on average $K\ln(K)$. Recall that for the All-At-Once distribution it roughly takes

$$K\ln(K/\delta) = K\ln(K) - K\ln\delta \tag{11-22}$$

coded symbols to cover all input symbols with probability $1-\delta$.

Interestingly, the number of symbol operations (sum of degrees) for the ideal soliton distribution and the number of coded symbols for the All-At-Once distribution coincide. But the numbers of coded symbols for both distributions are quite different.

For any distribution under the condition that the sum of the degrees of all the coded symbols needs to be around $K\ln(K)$ in order to cover all the input symbols, the ideal

soliton distribution compresses this into the fewest number of coded symbols possible, i.e., $N = K$.

Therefore, for all these distributions the total amount of computation, which is proportional to the sum of the degrees of all the coded symbols, is the same. However, the total amount of information, which is proportional to the number of coded symbols, is compressed to the minimum possible under the ideal soliton distribution.

11.5.2 The Robust Soliton Distribution

The problem with the ideal soliton distribution is that the expected ripple size (expected number of degree-one coded symbols) is only 1. Any variation in the ripple size is very likely to make the ripple disappear and the overall process fails.

The robust soliton distribution ensures that the expected size of the ripple is large enough at each point in the process so that it never disappears completely in high probability. On the other hand, the expected ripple size is also kept at a reasonable level so that not too many released coded symbols redundantly cover the input symbols already in the ripple.

The idea of robust soliton distribution is to design the distribution so that the expected ripple size is about $\ln(K/\delta)\sqrt{K}$ throughout the process, where the parameter δ is a bound on the probability that the decoding fails to run to completion after a certain number K' of symbols (packets) have been received.

Definition 11-3 (Robust soliton distribution [3][5]**)**: The robust soliton distribution $\mu(d)$ is defined as follows. Let $S = c\ln(K/\delta)\sqrt{K}$ for some suitable constant $c > 0$. Define a positive function as

$$\tau(d) = \begin{cases} S/(Kd) & for\ d = 1, 2, ..., (K/S) - 1 \\ \dfrac{S}{K}\log(S/\delta) & for\ d = K/S \\ 0 & for\ d > K/S \end{cases} \tag{11-23}$$

Add the ideal soliton distribution $\rho(d)$ to $\tau(d)$ and normalize to obtain the robust soliton distribution $\mu(d)$ as

$$\mu(d) = \frac{\rho(d) + \tau(d)}{Z} \tag{11-24a}$$

for all $d = 1, 2, \ldots, K$, where

$$Z = \sum_{d=1}^{K} [\rho(d) + \tau(d)] \tag{11-24b}$$

□

Evidently, the number of coded symbols (packets), which are required at the receiving end to ensure the completion of decoding with probability at least $1-\delta$, is $K' = KZ$.

Fig. 11.7 presents the distributions [5] $\rho(d)$ and $\tau(d)$ for the case $K = 10000$, $c = 0.2$ and $\delta = 0.05$, which gives $S = 244$, $K/S = 41$ and $Z \approx 1.3$. Obviously, for ideal soliton distribution $\rho(1) = 10^{-5}$ that is a very small number, and $\tau(1) \approx 0.025$. By (11-24a) and (11-24b) the value of robust soliton distribution relative to $d = 1$ is

$$\mu(1)=\frac{\rho(1)+\tau(1)}{Z}=\frac{10^{-5}+0.025}{1.3}$$

$$=0.01923>>10^{-5} \tag{11-25}$$

Therefore, the probability of degree-one symbols under robust soliton distribution is much higher than that under the ideal soliton distribution. □

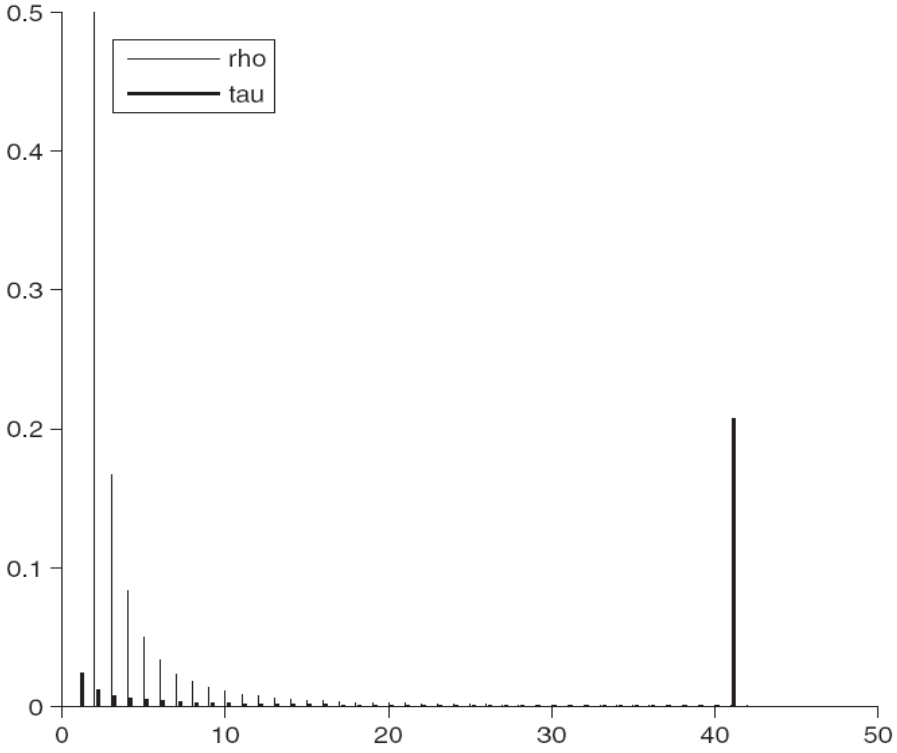


Fig. 11.7. The distributions $\rho(d)$ and $\tau(d)$ for the case K=10000, c=0.2 and δ=0.05, which gives S=244, K/S=41 and $Z\approx$1.3. The distribution $\rho(d)$ is largest at d=1 and $d=K/S$=41.

11.5.3 Introduction of Raptor Codes

One might think if we could do any better than LT codes, whose encoding and decoding cost scales as $K\ln K$. Raptor codes [5][7-9] can achieve linear time encoding

and decoding by concatenating a weakened inner LT code with an outer code that patches the gaps in the LT code.

LT codes have decoding and encoding complexity that are scaled as $\ln K$ per packet, because the average degree of the symbol in the sparse graph is $\ln K$. Raptor codes use an LT code with average degree $\bar{d}$ about 3. With this lower average degree, the decoder may work well in the sense that it can start process and cannot get stuck.

Fig. 11.8 presents an example of raptor code [5], where K=16 source symbols are first encoded by an outer code into $\tilde{K}$=20 pre-coded symbols and then encoded into N=18 coded symbols with a weakened inner LT code. The weakened LT code fails to connect some of the pre-coded symbols to any coded symbol. In this example, these 3 lost symbols are highlighted in back. The weakened LT code recovers the other 17 pre-coded symbols, then the outer code is used to deduce the original 16 original symbols.

For details of raptor codes, the interested reader can refer to the references [5][7-9].

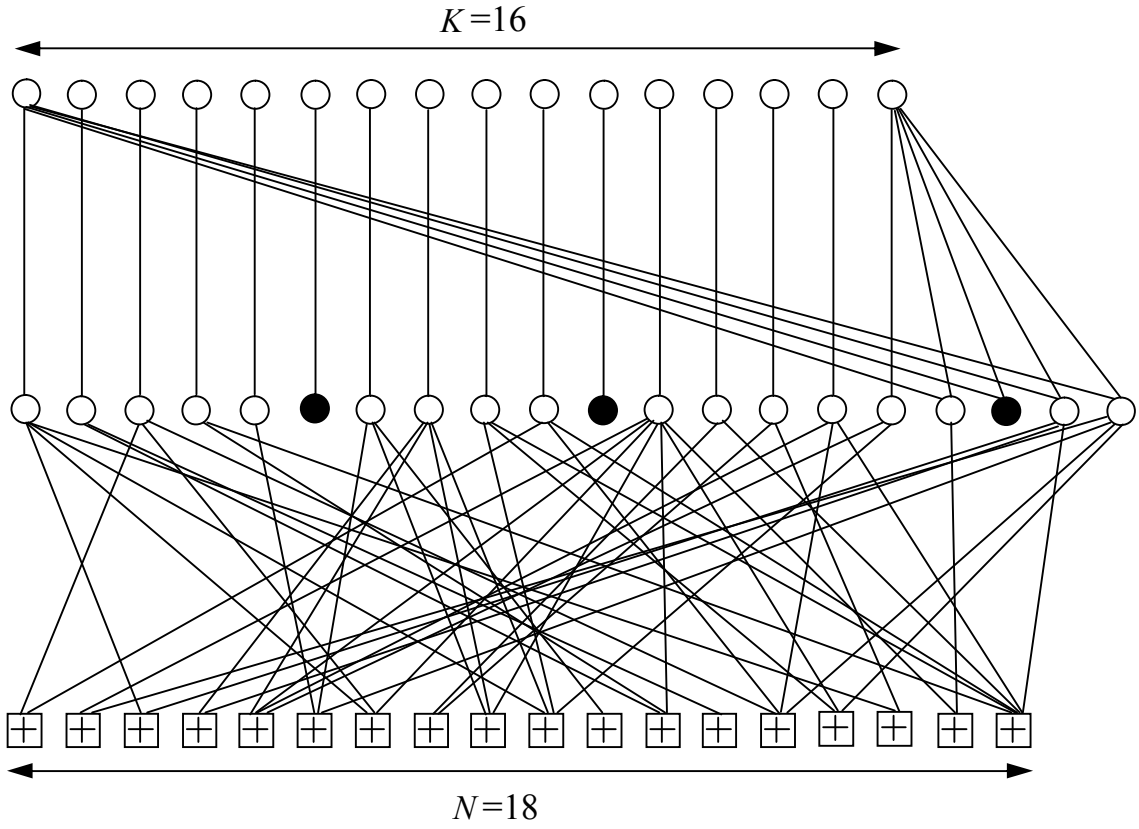


Fig. 11. 8. An example of a raptor code, K=16 source symbols (top row) are encoded by an outer code into $\tilde{K}$=20 pre-coded symbols (centre row). These symbols are encoded into N=18 coded symbols (bottom row) with a weakened inner LT code. Most of the coded symbols have degree 2 or 3. The average degree is 3. The weakened LT code fails to connect 3 pre-coded symbols. The LT code recovers the other 17 pre-coded symbols, then the outer code is employed to deduce the original 16 source symbols.

11.6 Applications of Raptor Codes in Multimedia Broadcast/Multicast Service

11.6.1 Fundamentals of Multimedia Broadcast/Multicast Service

The Multimedia Broadcast/Multicast Service (MBMS) technical specification defines mobile multimedia services over GSM-based 3rd generation (3G) cellular networks. MBMS multimedia content is delivered via IP packets, either as a streaming service or as a file download service to the end user. In streaming services, a continuous data flow of audio/video is delivered to the end user's handset; in download services, a finite amount of data is delivered to the user in its entirety as a file.

MBMS is a point-to-point IP-based service carrier by the 3G air interface of W-CDMA or the 2.5G air interface GPRS (General Packet Radio Service). The W-CDMA air interface provides bearer rates up to 256 kbps or greater, while the GPRS air interface provides bearer rates up to 128 kbps.

The MBMS technical specification requires mandatory use of advanced forward error correction codes in order to ensure that the one-way point-to-multipoint transmission of multimedia content is delivered to end users efficiently and with the best possible quality. Such FEC technology is raptor code invented by Digital Fountain Inc. [9][10][11], simply as DF Raptor. If the MBMS does not intrinsically provide some means to handle the effects of packet loss by using DF Raptor, streaming services would suffer dropouts and distorted audio/video, and download services would incur delays and waste bandwidth in delivering files to end users.

At the same time, however, the MBMS technical specification does not detail exactly how DF Raptor should be used, leaving it to the network operators and equipment vendors to choose the suitable parameters of DF Raptor.

11.6.2 Forward Error Correction for MBMS

MBMS employs FEC technology in two complementary ways, first at the physical layer, using the channel coding scheme of the underlying 3G air interface to correct bit errors, and, second at the application layer using DF Raptor to recover lost packets. While the FEC at the physical layer attempts to ensure that all the bits that are actually received are correct, the FEC provided by DF Raptor attempts to restore missing data.

FEC at the Physical Layer

To correct bit errors at the physical layer, FEC encoding and decoding algorithms are implemented as part of the W-CDMA or GPRS air interface. On the receiver side at each mobile receiver, the decoding algorithm attempts to correct bit errors that may have been induced by noise, interference, multipath, and other impairments over the wireless channel. If too many bit errors are present in a data block for the FEC decoding algorithm to correct, then the block is unrecoverable and the associated IP packets are lost. The block error rate represents the failure rate of the air interface FEC protection to recover the data blocks as partitioned at the physical layer.

FEC at the Application Layer

To provide packet-level protection at the application layer, DF Raptor is implemented at both the transmitter and each mobile receiver. The raptor code acts to complement the bit-level FEC protection by physical layer, recovering those IP packets that have been lost because the receiver's air interface FEC was unable to fully correct the underlying blocks of transmitted data. In addition to recovering packets that have been discarded because excessive physical layer errors are present, DF Raptor can also recover those packets that are lost because of the network congestion or simply never received due to temporary outages, receiver being turned off or out of coverage, etc.

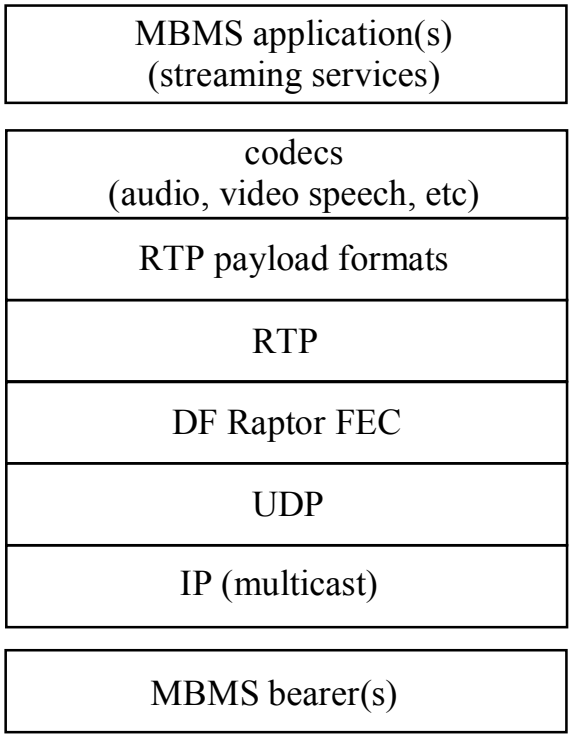


Fig. 11.9. The MBMS protocol stack for streaming services.

11.6.3 Packet Loss Protection for Streaming Services

MBMS streaming services are delivered using RTP (Real-time Transport Protocol) as specified by the Internet Engineering Task Force (IETF), operation over UDP/IP (User Datagram Protocol/Internet Protocol). The MBMS protocol stack for streaming services is shown in Fig. 11. 9.

RTP is designed to carry real-time content, providing timestamps, identification of the type of payload, sequencing numbering, and control mechanism to synchronize different streams while respecting their relative timing properties. The sequence numbers included in RTP allow the receiver to reconstruct the original packet sequence arranged by the sender, but RTP cannot guarantee successful delivery or prevent out-of-order delivery.

In order to improve the delivery quality of the multimedia stream, DF Raptor provides packet-level protection of the RTP data stream so that lost packets can be independently recovered by each receiver without requiring retransmission. At the sender (MBMS server) the original stream of RTP packets is partitioned into consecutive source blocks of data that are treated independently. Each successive source block is encoded by DF Raptor, generating more packets than were originally present in the source block in order to compensate for any potential packet loss. By transmitting the source packets along with the redundant packets in this way, playback applications can make use of whatever source data has been received even if DF Raptor is unable to completely recover the full

source block. The illustrative diagram of DF Raptor encoding of multimedia stream based on RTP is shown in Fig. 11.10, where DF Raptor as adopted by MBMS is termed as systematic code [10][11] because the encoding algorithm makes the original uncoded data available at the receiver directly.

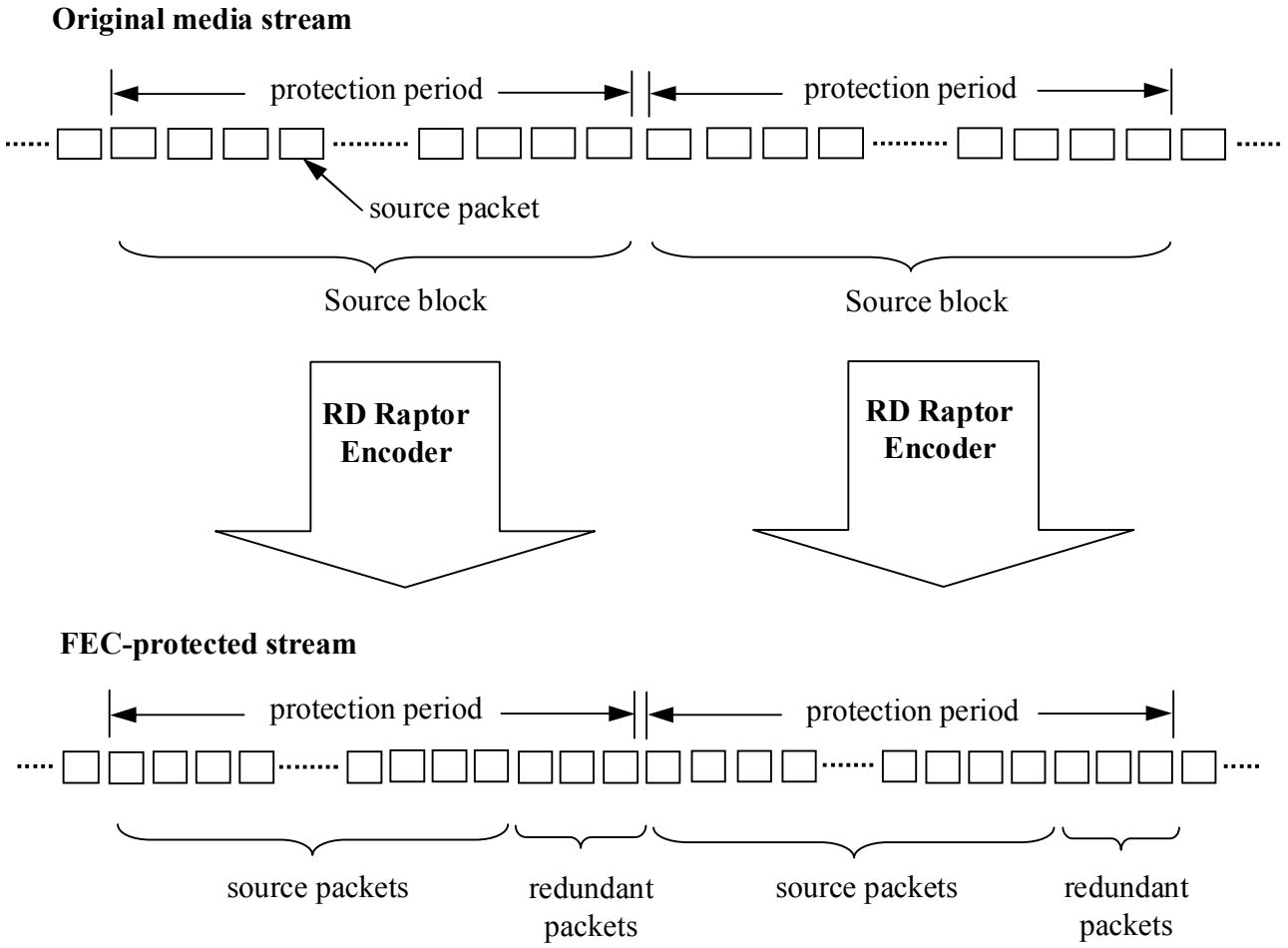


Fig. 11.10. DF Raptor encoding of media stream by RTP.

At each receiver, the received stream of source and redundant packets is processed as blocks and decoded by DF Raptor. It does not matter which of the original and

redundant packets have been lost provided that enough data has been successfully received, then each original block can be fully recovered. Successive blocks of the received stream of source and redundant packets are handled by the DF Raptor decoder in turn so that the original source stream is available for subsequent playback processing.

Meanwhile, a number of considerations are involved in configuring DF Raptor to protect MBMS streaming services. For example, both the size of the source block and the number of redundant packets generated from each source block are variables that can be set by the MBMS server as desired, which directly affect the performance of DF Raptor in protecting against packet loss. The trade-offs are related to the several facts: the source block size directly affects latency, while the number of redundant packets determines how much additional bandwidth is required to support the DF Raptor-protect stream.

Latency and Bandwidth Expansion

In general, each handset that contains a RF Raptor decoder must buffer the first block of received source and redundant packets in order to decode and obtain the first source block of the stream. For example, if N seconds of the original data stream comprises each source block, then in order to recover any lost source packets and avoid interruptions to the stream presented for playback, the delay between receiving the first packet and the start of playback at the receiver is equal to the N seconds of buffering

initially required at the decoder plus the negligible processing time required by the DF Raptor decoding algorithm.

The presence of latency in a one-way MBMS broadcast can potentially affect the user experience by perceptibly delaying the initial playback of a stream. Obviously, there are various techniques in MBMS that can effectively hide the effects: for example, a stream that is scheduled to begin playback at a specific time can actually begin transmission seconds earlier so that the latency is not apparent to the user, or multiple streams can be bundled and delivered as a single stream so that switching between streams does not incur any latency. Thus, minimizing latency by reducing the source block size is of great importance.

At the same time, large source block sizes imply that any burst errors will affect a smaller percentage of data than for a small source block. One the one hand, large source blocks can potentially increase the perceived latency, while on the other hand, large source blocks improve the ability of DF Raptor to protect against packet loss by various erasures.

A similar performance trade-off exists in setting the number of additional redundant packets produced by the encoding process for each source block. The overhead introduced by the redundant packets directly translates into channel bandwidth expansion. If we maintain the timing of the original stream, the source and redundant packets corresponding to a source block must be transmitted over the same duration as

the original data, which consequently leads to a faster data rate than the original stream so as to accommodate the additional redundant packets. On the one hand, more redundant packets mean the potential ability to tolerate more packet loss; on the other hand, more redundant packets must result in more bandwidth to support higher data rate transmissions.

An Example of Trade-off

In general, the quality of the delivered multimedia stream protected by DF Raptor can be quantified by the probability that DF Raptor cannot recover an entire source block. If the source block size employed to encode successive blocks of the original stream corresponds to n seconds and the probability of decoding failure for a block is P. Suppose that average m out of m_0 source blocks be decoded unsuccessfully, then we have

$$P \approx \frac{m}{m_0} \tag{11-26a}$$

The expected time $\overline{T}$ between the adjacent block decoding failures will be

$$\overline{T} = \frac{nm_0}{m} = \frac{n}{m/m_0} \approx \frac{n}{P} \tag{11-26b}$$

As long as the probability P of decoding failure (or the expected time $\overline{T}$ between decoding failures) for DF Raptor is below a desired threshold, then visible and audible artifacts in the playback of the video or audio stream will be few and far between. Hence, such decoding failures may not result in degraded quality as experienced by the end user.

In other words, the available source data and the decoding algorithm used by the multimedia stream may successfully mask the fact that the complete source blocks are not recovered. The trade-offs among loss protection performance, latency, and bandwidth expansion are considered in the following results [10][11].

Fig. 11.11 shows the simulation results [11] for UMTS (Universal Mobile Telecommunications Systems) Terrestrial Radio Access Network bearer rates of 256, 128 and 64 kbps when DF Raptor has been used to protect source block sizes associated with either 5 or 20 seconds of the original stream.

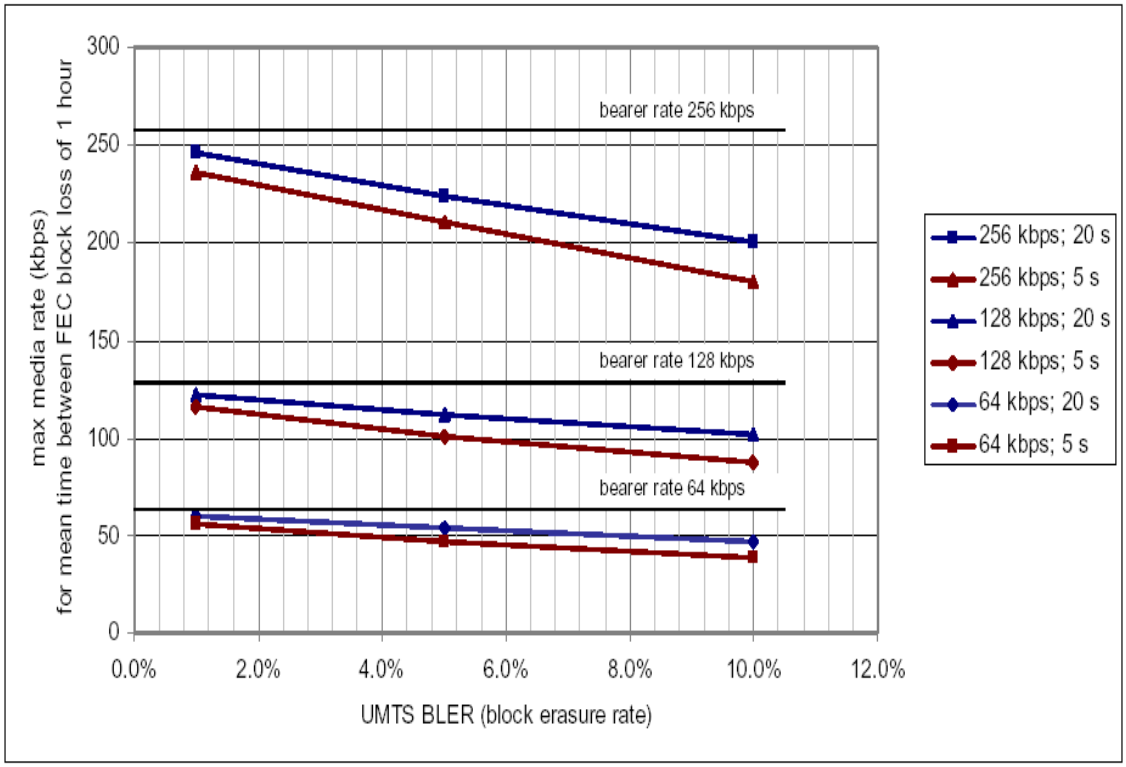


Fig. 11.11. Maximum MBMS streaming media rates supported by DF Raptor for different UMTS bearer rates and block erasure rates.

Fig. 11.11 offers the maximum media data rates that can be supported as a function of the block error rate of the channel so that DF Raptor statistically fails to fully recover a source block once an hour. Note that the mean time between decoding failures within one hour provides a quantitative figure of merit for the stream quality as delivered by MBMS using DF Raptor.

The maximum media rate shown in the figure corresponds to the minimum amount of redundant packets and bandwidth expansion necessary to provide the desired quality of one irrecoverable source block per hour. Note that if the media rate is actually less than the maximum value given in this example but DF Raptor is still employed to produce the redundant packets expanding the original stream bandwidth up to the available bearer rate, then the mean time between decoding failures would be longer than one hour and the delivered quality would be accordingly better.

Fig. 11.11 also presents the effects of different source block sizes. It can be seen that the larger the source block is (i.e., 20 seconds case), then the greater the maximum possible media rate since less redundant packets are required by DF Raptor to maintain a mean time of one hour between decoding failures.

11.6.4 Packet Loss protection for Download Services

Unlike the streaming service, MBMS download services are based on the FLUTE (File Delivery over Unidirectional Transport) protocol as specified by IETF carried over UDP/IP. The MBMS protocol stack for download services is shown in Fig. 11.12.

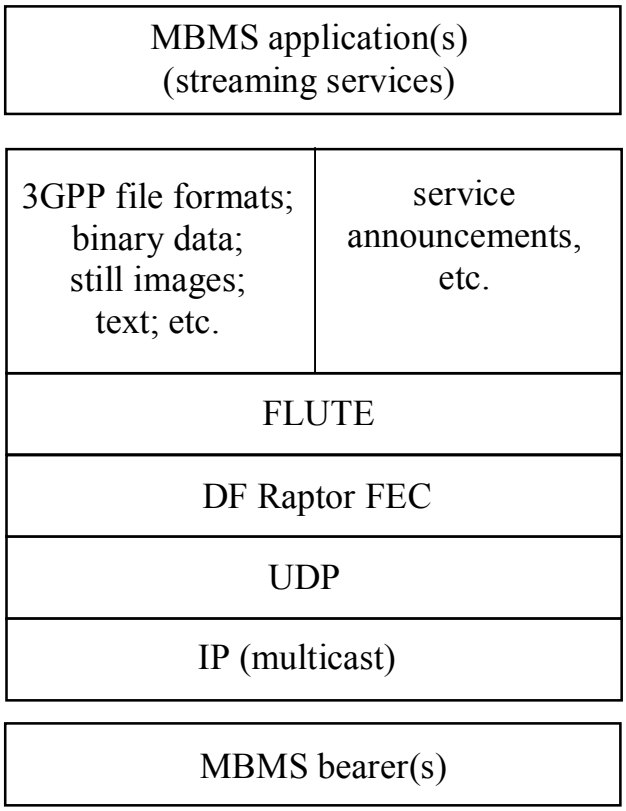


Fig. 11.12. The MBMS protocol stack for download services.

FLUTE is used to deliver both MBMS download services and MBMS service announcements via one-way point-to-multipoint transmissions. In the sender (or transmitter side), the MBMS server takes one or more data files (video clips, still images, software executables, etc.) and partitions them into data elements that can be constructed into FLUTE packet payloads. As part of the FLUTE transmission, the sender announces the total length of a data element, and the overall structure of how the data is to be transmitted, thus allowing each receiver to effectively manage the reception and reconstruction of the data.

FLUTE does not guarantee the data reliability for one-way transmissions. Hence, MBMS also need to integrate DF Raptor into the FLUTE protocol in order to provide packet-level protection of the data so that any lost packets can be recovered by each receiver without requiring retransmission. At the MBMS server, each data file is considered to be one or more source blocks, each of which is encoded by DF Raptor and transmitted using FLUTE as source and redundant packets. At each receiver, the received FLUTE packets are decoded by DF Raptor. It does not matter which of the source and redundant packets have been lost provided that enough data has been successfully received, then the original block can be fully recovered. The illustrative diagram of DF Raptor encoding of files based on FLUTE is shown in Fig. 11.13.

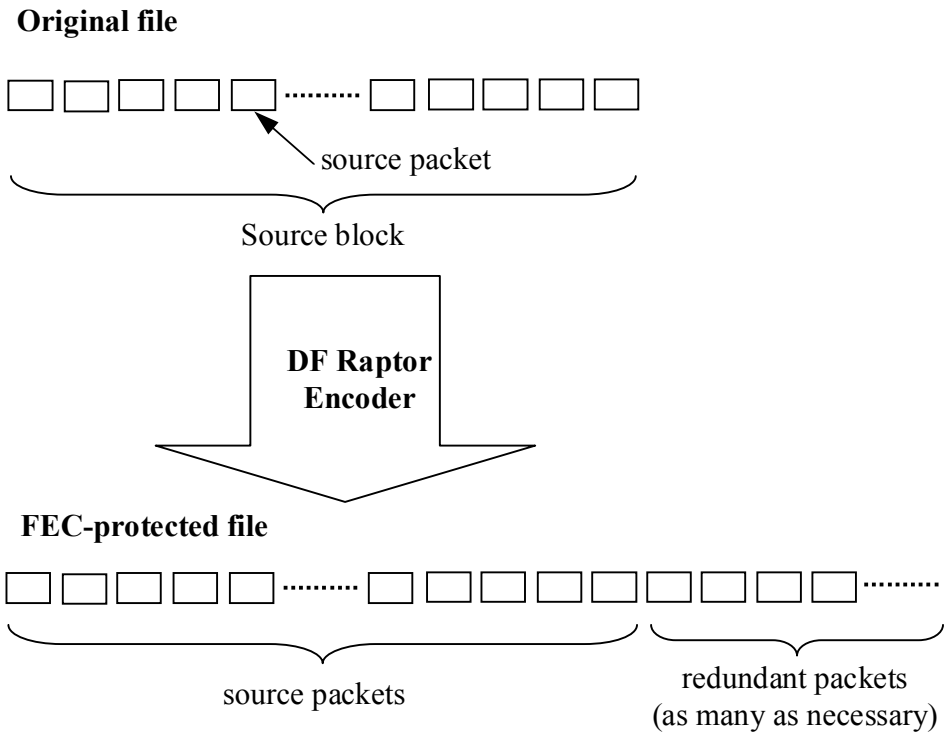


Fig. 11.13. DF Raptor encoding of files by FLUTE.

FEC Overhead

The number of additional redundant packets generated by DF Raptor is a variable that can be dynamically set at the MBMS server. A small number of redundant packets may not allow most MBMS receivers to fully recover the source block, while a large number of such packets will spend more additional transmission time that could be used for the support of other file downloads or services. DF Raptor provides the flexibility to optimize file download services by generating and transmitting just enough redundant packets as needed by individual receivers using post-delivery [10][11].

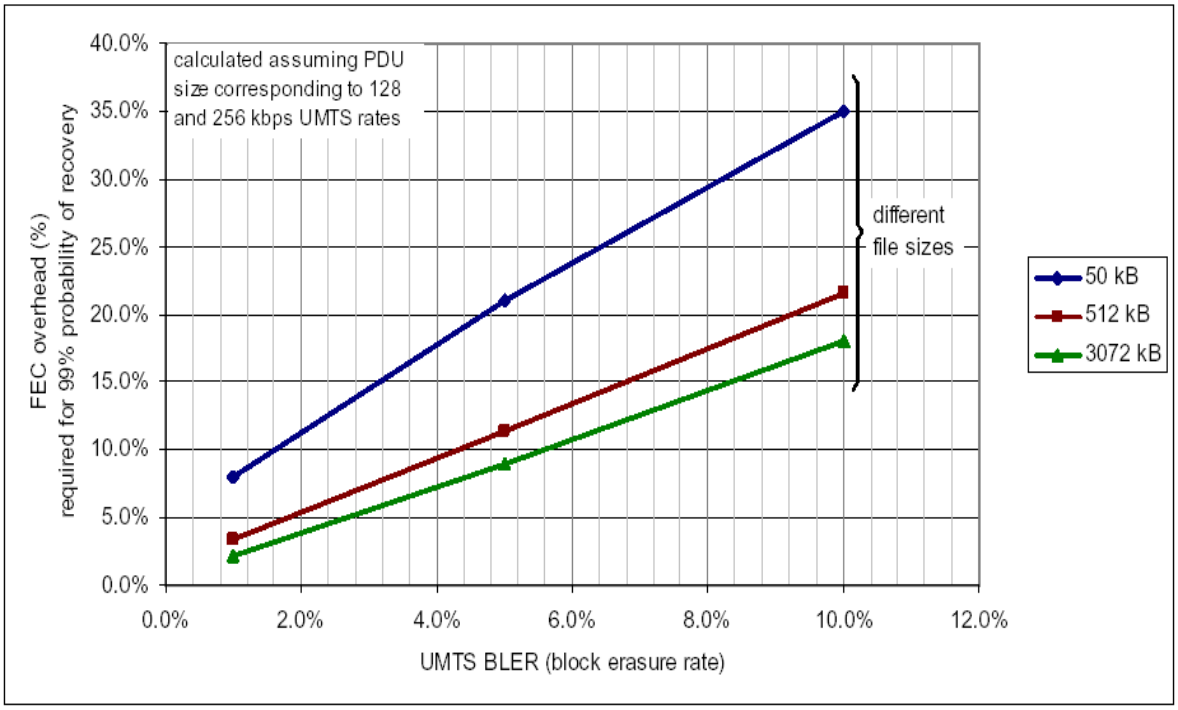


Fig. 11. 14. DF Raptor FEC overhead required for 99% decoding success as a function of different file sizes and UMTS channel conditions.

Fig. 11. 14 offers the number of additional packets in percentage [11] needed to deliver files of typical sizes over a UMTS channel as a function of the block error rate. The FEC overhead is defined as the ratio of the redundant packets to source packets, assuming the PDU (Protocol Data Unit) size related to UMTS Terrestrial Radio Access Network bearer rates of 128 and 256 kbps. The minimum FEC overhead required to realize a 99% probability of complete file recovery by a receiver is presented as a percentage. Note that if more redundant packets are available for a given block error rate, then the probability of success will be definitely higher.

Fig. 11.14 also shows that small data files require higher levels of FEC overhead that large data files for a given block error rate. This result reflects how the randomly distributed block errors are more likely to vary from the average rate when the source block size becomes small.

Finally, the simulation results indicate the need for relatively higher levels of FEC overhead on the presence of higher block error rates. In some scenarios, high levels of FEC overhead may be necessary if it allows MBSM download services to be operated more efficiently from an overall system perspective. Therefore, the flexible and dynamic FEC overhead is one of the key advantages of DF Raptor used in MBMS.

Post-Delivery File Repair

Unlike streaming services where occasional data errors may not be experienced by the user, download services require that each transmitted file must be obtained in its entirety

with error-free. In other words, a partially downloaded file usually is of no value to the user. If the FEC overhead supplied by the MBMS server is sufficient and each receiver can obtain enough source and redundant packets to fully recover the original data file, thus no further action is required.

However, some receivers may not have successfully obtained enough packets by the end of the file delivery session. In this case they may each employ a point-to-point connection over TCP/IP to request a file repair session from the file repair server, or if there are many such receivers then a point-to-multipoint repair session may be scheduled to deliver additional repair packets to the receivers via a new MBMS download service session. For details of the post-delivery file repair, the interested reader can refer to the references [9][10][11].

Summary

MBMS promises to deliver a rich array multimedia content to 3G subscribers. By using DF Raptor to protect against packet loss caused by channel conditions, such as network congestion, or intermittent receiver outages, etc., MBMS is able to deliver its streaming and download services with maximum quality and efficiency.

References

[1] Berlekamp, E. R. Algebraic coding theory, McGraw-Hill, New York, 1968.

[2] S. Lin and Costello, D. J. Jr. Error control coding: fundamentals and applications, Prentice-hall, Englewood Cliffs, New Jersey, 1983.

[3] M. Luby, "LT," codes, Proc. 43rd Ann. IEEE Symp. on Foundations of Computer Science, 16-19 November 2002, pp. 271-282.

[4] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," Proc. ACM SIGCOMM'98, 2-4 September 1998.

[5] D. J. C. MacKay, "Fountain codes," IEE Proceeding Communications, Vol. 152, No. 6, pp. 1062-1068, Dec. 2005.

[6] S. Shamai, I. E. Telatar and S. Verdu, "Fountain capacity", IEEE Trans on Inform. Theory, Vol. 53, No. 11, pp. 4372-4376, Nov. 2007.

[7] A. Shkrollahi, "Raptor codes," IEEE Trans. Inform. Theory, Vol. 52, No. 6, pp. 2551-2567, June, 2006.

[8] O. Etsami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," IEEE Trans. on Inform. Theory, vol. 52, No. 5, pp. 2033-2051, May 2006.

[9] J. W. Byer, M. Luby and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," IEEE Journal of Selected Areas in Communications, Vol. 20, No. 8, pp. 1528-1540, Oct. 2002.

[10] Press Release: Digital Fountain, Inc., 2005 [Online]. Available: http://www.digitalfountain.com/technology/standards/index/.cfm

[11] Multimedia Broadcast/Multicast Service (MBMS); Protocols and Codes. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/26346.htm

[12] G.Caire, S. Shamai (Shitz), A. Shokrollahi, and S. Verdu, "Fountain codes for lossless compression of binary source," in Proc. 2004 IEEE Workshop on Information Theory, San Antonio, TX, Oct. 2004.

[13] A. M. Tulino, S. Verdu, G. Caire and S. Shamai, "The Gaussian-erasure channel," in Proc. IEEE 2007 Int. Symp. Information Theory, Nice, France, June 2007, pp. 1721-1725.

[14] R. G. Gallager, Information Theory and Reliable Communication. New York: Wiley, 1968.

[15] T. M. Cover, "Comments on broadcast," IEEE Trans. Inf. Theory, vol. 44, No. 6, pp. 2524-2530, Oct. 1998.

[16] S. Verdu, "On channel capacity per unit cost," IEEE Trans. Inform. Theory, vol. 36, No. 5, pp. 1019-1030, Sept. 1990.

[17] S. Verdu and T. S. Han, "A general formula for channel capacity," IEEE Trans. Inform. Theory, vol. 40, No. 4, pp. 1147-1157, July, 1994.

[18] A. Lapidoth and P. Narayan, "Reliable communication under channel uncertainty," IEEE Trans. Inform. Theory, vol. 44, No. 6, pp. 2148-2177, Oct. 1998.