

Codes for Reliable Transmission of Data over the Internet

This lecture covers:

- LT codes
- Raptor Codes

26.1 Introduction

Problem: Suppose we wish to send a large file to many receivers over the Internet. We can split the file into packets, some of which may be dropped or corrupted during transmission. Furthermore, different receivers will have various bandwidths and will receive subsets of these packets at different times. Here are three possible approaches to this problem:

Method 1: Use acknowledgements, as in TCP/IP, to determine which packets need to be retransmitted to each receiver. This can be very complex and inefficient when broadcasting to many receivers.

Method 2: Use erasure correction codes, in which redundancy is introduced by encoding a k -packet file as $n > k$ packets. This is problematic when different receivers have unknown loss rates that may vary over a large range.

Method 3: Produce an indefinitely long sequence of packets such that any collection of $k(1 + \epsilon)$ packets should suffice to recover the file, with high probability. This method should work regardless of the loss rates of the receivers and the period during which they are attentive to the arriving packets. Additionally, the encoding and decoding algorithms should be simple and efficient. Such a scheme is called a “fountain code”, which can be interpreted with the metaphor of getting water from a (digital) water fountain.

We will describe LT codes, Michael Luby’s implementation of Method 3.

26.2 LT Codes

Let a symbol be a string of t bits, $\{0, 1\}^t$. The original file is a sequence of k input symbols, x_1, x_2, \dots, x_k . The encoding process will produce a potentially unlimited stream of output symbols, $y_1, y_2, \dots, y_l, \dots$, which are generated by repeating a certain randomized algorithm. The algorithm is determined by a probability

distribution over the random variable $X \in \{1, 2, \dots, k\}$; we will call this the *degree distribution*, letting $a_d = \Pr[X = d]$.

Each output symbol y is independently generated as follows:

1. Draw d from the distribution over X
2. Choose a set S uniformly at random from the d -element subsets of $\{1, 2, \dots, k\}$
3. Set y equal to $\bigoplus_{i \in S} x_i$

Each y is the XOR of a random subset of input symbols. The output symbol also carries a label indicating the set S which generated it. We want the probability distribution X to be chosen so that upon receiving l randomly chosen output symbols, y_1, y_2, \dots, y_l where $l > k$, we can determine x_1, x_2, \dots, x_k with high probability. Thus, to decode, we must solve the system of equations:

$$y_j = \bigoplus_{i \in S_j} x_i \quad , j = 1, 2, \dots, l$$

Rather than performing Gaussian elimination over a 2-element field, we want to solve this system in linear time using *belief propagation*.

26.3 Belief Propagation

We will begin with an example to illustrate the belief propagation algorithm. Let $t = 1$ (i.e. each symbol is a 1-bit packet) and let the input consist of five symbols. We generate six output symbols as follows:

$$\begin{aligned} (1) \quad 1 &= x_2 \oplus x_3 \oplus x_4 \oplus x_5 \\ (2) \quad 0 &= x_1 \oplus x_3 \oplus x_4 \\ (3) \quad 0 &= x_2 \oplus x_3 \oplus x_5 \\ (4) \quad 1 &= x_3 \oplus x_4 \\ (5) \quad 0 &= x_3 \\ (6) \quad 0 &= x_2 \oplus x_3 \end{aligned}$$

We begin with Eq. (5), which asserts that $x_3 = 0$. Substituting this value into Eq. (6), we determine that $x_2 = 0$; similarly for Eq. (4), $x_4 = 1$. Proceeding in similar fashion, we insert the values for x_2 and x_3 into Eq. (3) to determine $x_5 = 0$. Finally, we can determine that $x_1 = 1$ by substituting in Eq. (2).

For the belief propagation algorithm, let A denote the set of indices of input symbols that have been determined and eliminated. Let R (the *ripple*) be the set of indices of input symbols that have been determined but not eliminated. \tilde{S}_i will denote the reduced set of symbols in S_i , and $|\tilde{S}_i|$ will be the *reduced degree*. The algorithm:

```

1:  $A \leftarrow \emptyset$ 
2: for  $i = 1, 2, \dots, l$  do
3:    $\tilde{S}_i \leftarrow S_i$ 
4: while  $A \neq \{1, 2, \dots, k\}$  do
5:    $R \leftarrow \bigcup_{\{i: |\tilde{S}_i|=1\}} \tilde{S}_i$ 
6:   if  $R = \emptyset$  then
7:     return failure
8:   choose an element  $k \in R$ ; solve for  $x_k$ 
9:    $A \leftarrow A \cup \{k\}$ 
10:   $\tilde{S}_i \leftarrow \tilde{S}_i - \{k\}$  for  $i = 1, 2, \dots, l$ 
11: return success (all variables determined)

```

26.4 Degree Distribution

We wish to design the algorithm above so that belief propagation is not likely to fail. We must choose l , the number of output symbols, and determine the distribution of X (the degree of an input symbol) so that the ripple is never empty while some variables remain undetermined. On the other hand, we don't want $|\tilde{S}_i|$ to be 1 for too many equations at the same time, because then several equations might determine the same variables, which would be wasteful of the output symbols.

Published information will tell us to decode x_1, x_2, \dots, x_k after we have collected y_1, y_2, \dots, y_l , where l is modestly greater than k . Clearly, if $l < k$ then the algorithm will always fail because the set of equations cannot have a unique solution; we need a set of k linearly independent equations (full rank) so that $l \geq k$.

Recall the coupon-collector problem: there are k coupons in a set, at each trial a coupon is drawn at random, and we wish to collect all of the coupons in a set; the expected number of trials needed is at least $k \ln k$. If the degree was always 1, we would need to collect about $k \ln k$ output symbols. Note that receiving an output symbol of degree d is like receiving d coupons. Thus, the sum of degrees of all received symbols must be $\Omega(k \ln k)$. Since we are transmitting about k symbols, we will want the average degree to be about $\ln k$.

Let a *stage* be an iteration of the “while” loop, during which one more input variable is eliminated. An output symbol contributes one of its input symbols to the ripple at the stage when all but one of its variables have been eliminated. For an output variable of degree d , that stage is a random variable concentrated around $\frac{d-1}{d+1}k$.

The degree distribution should be chosen so that, for any s , the number of input variables entering the ripple up to the end of stage s is slightly greater than s , with high probability.

26.4.1 The Ideal Soliton Distribution

Consider the following setup:

$$\begin{aligned}
 l &= k \\
 a_1 &= \frac{1}{k} \\
 a_d &= \frac{1}{d(d-1)}, \quad d = 2, \dots, k
 \end{aligned}$$

Note: $\sum_{d=1}^k a_d = 1$. This is called the *soliton distribution*.

The expected number of output symbols of degree 1 is $k(\frac{1}{k}) = 1$; for $d = 2, \dots, k$ the expected number of output symbols of degree d is $\frac{k}{d(d-1)}$.

Suppose the number of output symbols of degree 1 is 1 (the expected number). Then one variable is determined and eliminated. We now calculate the distribution of reduced degrees.

The expected number of output symbols of reduced degree 1 =

$$k \cdot a_d \left(\frac{2}{k} \right) = k \left(\frac{1}{2 \cdot 1} \right) \left(\frac{2}{k} \right) = 1$$

The expected number of output symbols of reduced degree d =

$$k \cdot a_d \left(1 - \frac{d}{k} \right) + k \cdot a_{d+1} \left(\frac{d+1}{k} \right) = \frac{k}{d(d-1)} + \frac{k}{(d+1)d} \left(\frac{d+1}{k} \right) = \frac{k-1}{d(d-1)}$$

Thus, if at each step the actual number of input variables joining the ripple is equal to the expected number, then the soliton distribution exactly reproduces itself, and the size of the ripple is just 1 at all times.

The soliton distribution is not usable in practice because the actual number of additions to the ripple will fluctuate around the expected number. However, by perturbing the soliton distribution so that the expected size of the ripple is $\Omega(\sqrt{k} \ln k)$ at all stages, Luby obtained a construction with

$$l = k + O(\sqrt{k} \ln^2(\frac{k}{\delta}))$$

with the average degree of an output symbol $\leq H(k) + O(\sqrt{k} \ln k \frac{k}{\delta})$, and failure probability δ , for any (small) positive δ .

26.4.2 Shokrollahi's construction

Shokrollahi gave an alternative construction by imposing a requirement that the expected size of the ripple after stage s should be at least $c\sqrt{k-s}$, where c is a positive constant. The intuition is that random fluctuations should be unlikely to produce a ripple of size 0 when the expectation is so large.

Define $\alpha = \frac{l}{k}$, along with a generating function Ω :

$$\Omega(x) = \sum_{d=1}^k a_d x^d$$

After s symbols have been decoded and eliminated, what is the probability that a given input variable is determined? This is the probability that, for some output symbol, the input variable is among the elements of the input symbol and the other inputs have all been determined by stage s . For an output symbol of degree d , this probability is $d \cdot \frac{1}{k} \cdot \left(\frac{s}{k}\right)^{d-1}$.

If d is unknown, then we can average over the possible degrees for the expectation:

$$\sum_{d=1}^k d \cdot a_d \cdot \frac{1}{k} \left(\frac{s}{k}\right)^{d-1} = \frac{1}{k} \Omega' \left(\frac{s}{k}\right)$$

Then the probability that the input variable has not been determined by stage s is $\left(1 - \frac{1}{k} \Omega' \left(\frac{s}{k}\right)\right)^{\alpha k}$, which is well approximated by $e^{-\alpha \Omega' \left(\frac{s}{k}\right)}$. So the expected number of input variables determined by stage s is $k \left(1 - e^{-\alpha \Omega' \left(\frac{s}{k}\right)}\right)$ and the expected size of the ripple is $k \left(1 - e^{-\alpha \Omega' \left(\frac{s}{k}\right)}\right) - s$.

By Shokrollahi's requirement, for all s ,

$$k \left(1 - e^{-\alpha \Omega'(\frac{s}{k})} \right) - s \geq \sqrt{k - s}$$

or, letting $x = \frac{s}{k}$:

$$1 - x - e^{-\alpha \Omega'(x)} \geq c \sqrt{\frac{1-x}{k}}$$

Rewritten as:

$$\Omega'(x) \geq \frac{\ln \left(1 - x - c \sqrt{\frac{1-x}{k}} \right)}{\alpha}$$

This requirement can only be satisfied if

$$1 - x - \sqrt{\frac{1-x}{k}} > 0$$

or equivalently:

$$x \leq 1 - \frac{c^2}{k}$$

For any $x \in [0, 1 - \frac{c^2}{k}]$, this condition is a linear inequality in the quantities a_d . By imposing these requirements, we can set up a linear program to minimize $\sum da_d$, the expected degree of an output symbol. Solving this linear program for a specified k, l, c will give us the desired degree distribution on X .

26.5 Conclusion

As an example, suppose that $k = 100,000$ and the encoding introduced a redundancy of $\alpha = 1.038$. Then the average degree of an input symbol would be 5.87. Note that this is much less than the $\ln k \approx 11.5$, which was the necessary average degree required by the coupon-collection analysis. Thus the decoder is very likely to stall, leaving about 500 symbols undetermined. To decode the remaining symbols, we introduce some added redundancy with a low-density parity check which will be able to recover the missing packets. Coupled together, these two coding schemes form a *Raptor* code, which performs better than the LT-codes and has constant, rather than logarithmic, average degree.