Tornado codes

Scribe notes for 15-853[*]

Introduction

Similar to Hamming codes and Reed Solomon (RS) codes, Tornado codes belong to the family of forward error correction (FEC) codes. While the former two codes focus on efficiency, Tornado codes serve for a different purpose. Designed by Michael Luby, Michael Mitzenmacher, Amin Shokrollahi *et al.*, Tornado codes give up some efficiency in exchange for linear time encoding and decoding. Software-based implementations of Tornado codes are about 100 times faster on small lengths and about 10,000 times faster on larger lengths than other software-based Reed-Solomon erasure codes.

In this notes, we focus on the ability of Tornado codes to correct erasures. Erasures are more common than errors in the Internet environment, because bitstreams are organized into packages and packages are often either received or lost (erased). The following discussion is thus very suitable for applications such as multicast of streaming media over the Internet.

The notes are organized as follows. We first describe *expander graph*, which is the foundation of Tornado codes. We then build Tornado codes on top of expander graph and show that under certain assumptions, with a high probability Tornado codes can correct most of the erasures.

Expander Graph

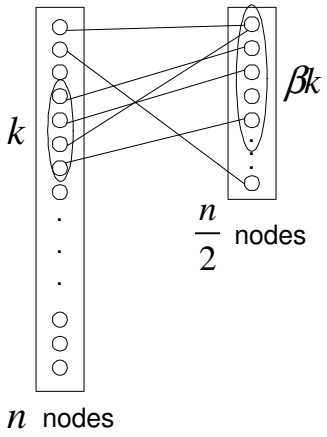

Figure 1 An expander graph.

We consider a special graph that is composed of two sets of nodes, as shown in Figure 1. The left set has n nodes, while the right set has $n/2$ notes. Given any two constants $0 \le \alpha \le 1$ and $\beta > 0$, we define (α, β)-*expansion* as:

[*] This notes is based on the lecture given by Prof. Bruce Maggs, Sep. 23, 2002. The author of the notes is Cha Zhang.

Definition 1: A graph has (α, β)-*expansion* if every subset of $k \le \alpha n$ nodes on the left has at least βk neighbors on the right, where α and β are fixed constants which satisfies $0 \le \alpha \le 1$ and $\beta > 0$.

Conclusion 1: In the expander graph in Figure 1, α and β must satisfy $\alpha\beta \le \frac{1}{2}$.

Proof: From the definition of (α, β)-expansion, we know that every subset of $k \le \alpha n$ nodes on the left has $\beta k = \alpha\beta n$ neighbors on the right. However, the right set has maximally $n/2$ nodes. Therefore:

$$\beta k = \alpha\beta n \le \frac{n}{2} \Rightarrow \alpha\beta \le \frac{1}{2}. \qquad \square$$

Given a graph as Figure 1, the existence of an (α, β)-expansion can be verified by Upfal's theorem [1].

Theorem 1: (Upfal 89) Suppose each node on the left has degree *d* (*d* is a positive integer), and each node on the right has degree 2*d*. View each node as a collection of *d* (or *2d*) mini-nodes. Select a random permutation on the *dn*×*dn* bipartite graph of mini-nodes. Collapse mini-nodes back into original nodes. If

$$d > \beta + 1 + \frac{\beta + 1 + \ln(2\beta)}{\ln\left(\frac{1}{2\alpha\beta}\right)} \tag{1}$$

then with probability > 0, the graph has (α, β)-expansion.

We discuss two interesting cases for the above theorem.

Case 1: When we have a very small α, i.e., $\alpha \to 0$, we have $\ln\left(\frac{1}{2\alpha\beta}\right) \to \infty$. From Equation (1) we know that $d > \beta + 1$. Therefore the maximum β we may take is:

$$\beta \approx d - 1.$$

Case 2: If α is fixed and we have a large enough degree *d*, so that β is neglectable compared with *d*, Equation (1) can be simplified as:

$$d > \frac{\beta + 1 + \ln(2\beta)}{\ln\left(\frac{1}{2\alpha\beta}\right)}$$

so that we are allowed to take β such that $\alpha\beta \to \frac{1}{2}$.

In this lecture we will need expanders where $\beta > \frac{d}{2}$. To see how we should choose for α, we let $\beta \approx \frac{d}{2}$ and plug it into Equation (1). We have:

$$\beta \approx d - 1 - \frac{\beta + 1 + \ln(2\beta)}{\ln\left(\frac{1}{2\alpha\beta}\right)} \approx \frac{d}{2}$$

Replace all the β with $\frac{d}{2}$ and assume $d >> 1$, we get approximately:

$$d - \frac{\frac{d}{2} + \ln(d)}{\ln\left(\frac{1}{\alpha d}\right)} \approx \frac{d}{2} \text{ or } \frac{d}{2}\ln\left(\frac{1}{\alpha d}\right) \approx \frac{d}{2} + \ln(d)$$

Therefore:

$$\ln\left(\frac{1}{\alpha d}\right) \approx \frac{\frac{d}{2} + \ln(d)}{\frac{d}{2}} = 1 + \frac{\ln(d)}{\frac{d}{2}}$$

$$\Rightarrow \frac{1}{\alpha d} \approx e \cdot e^{\frac{\ln(d)}{d/2}} \approx e \cdot d^{\frac{1}{d/2}} \approx e \cdot d^{\frac{2}{d}}$$

Notice that $d^{\frac{2}{d}} \approx 1$ for large d[1]. We obtain that

$$\alpha \approx \frac{1}{e \cdot d} \text{ or } \alpha\beta \approx \frac{1}{e \cdot d} \cdot \frac{d}{2} \approx \frac{1}{2e}.$$

This is not bad because we know that the maximum value of $\alpha\beta$ is $\frac{1}{2}$ based on Conclusion 1.

Recently, O. Reingold gave a simple deterministic construction of an expander in [2], where β can be almost as large as possible. That is, $\beta = (1-\varepsilon)d$ where ε is an arbitrary small number.

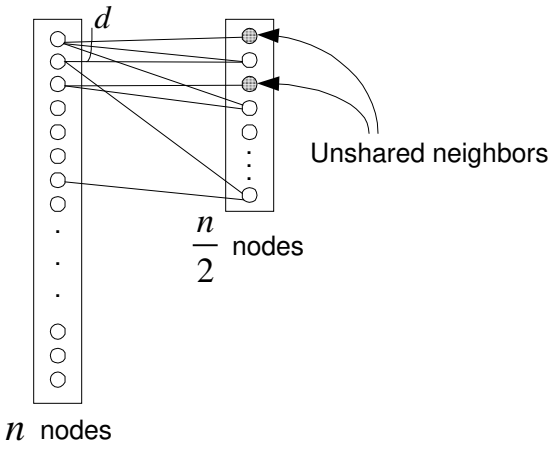

Figure 2 Unshared neighbors for a graph.

---

[1] This might not be obvious but can be justified as follows. We know that $d^{\frac{1}{\ln(d)}} = \left(e^{\ln(d)}\right)^{\frac{1}{\ln(d)}} = e$. For large d, we have $\frac{2}{d} << \frac{1}{\ln(d)}$. Therefore, $d^{\frac{2}{d}}$ should be close to 1.

There is one property for expander graph that is extremely important to Tornado codes. It is called the *unshared neighbor property*. An unshared neighbor is a node on the right that is only the neighbor of one node on the left, as shown in Figure 2. In other words, it is not shared as neighbors for multiple nodes on the left. The (α,δ) unshared neighbor property is defined as follows.

Definition 2: A graph has (α,δ) unshared neighbor property if for any subset of $k \le \alpha n$ nodes on the left, at least δk have an unshared neighbor on the right.

We have the following theorem on the choice of δ.

Theorem 2: For a graph that has degree d on the left, if it has both (α,β)-expansion and (α,δ) unshared neighbor property, it must be true that:

$$\delta \ge \frac{2\beta}{d} - 1 \tag{2}$$

Proof: Let γ be the fraction of nodes on the right that have only one neighbor on the left. Given a subset of k nodes on the left, from the definition of (α,β)-extension we know that it has at least βk neighbors on the right. Among them $\gamma\beta k$ are unshared neighbors. The left $(1-\gamma)\beta k$ have at least 2 neighbors on the left. If we count the edges between the left subset and its neighbors on the right, it must be less than dk, because the graph has degree d on the left. Therefore, we have:

$$\gamma\beta k + 2(1-\gamma)\beta k \le dk \tag{3}$$

where the second item on the left has a scale factor of 2 because $(1-\gamma)\beta k$ nodes have at least 2 neighbors (therefore 2 edges) on the left. Simplify Inequality (3) and we have:

$$\gamma \ge \frac{2\beta - d}{\beta} \text{ or } \gamma\beta k \ge (2\beta - d)k$$

Since on the right $\gamma\beta k$ nodes are unshared neighbors, on the left there must be more than $\frac{\gamma\beta k}{d}$ nodes that have an unshared neighbor on the right. Here we divide $\gamma\beta k$ by d because a node on the left has maximally d neighbors and they can all be unshared neighbors. Based on the definition of (α,δ) unshared neighbor property, we know that

$$\delta k \ge \frac{\gamma\beta k}{d} \ge \frac{(2\beta - d)k}{d} \text{ or } \delta \ge \frac{2\beta}{d} - 1. \qquad \square$$

Tornado codes

Consider a simple coding method based on the expander graph. As shown in Figure 3, let the left n nodes be the n message bits, and let the right $\frac{n}{2}$ nodes be the parity bits. Denote the message bits as $b_0, b_1, b_2, \cdots, b_{n-1}$, and the parity bits as $p_0, p_1, p_2, \cdots, p_{\frac{n}{2}-1}$. Each parity bit on the right is set to the exclusive or of its neighbors on the left. For example, parity bit p_0 can be defined as:

$$p_0 = b_0 \oplus b_2 \oplus b_4$$

where $\oplus$ stands for exclusive or. Such codes are called Tornado codes. Let the graph have degree d on the left. It is obvious that in order to get all the parity bits, the number of exclusive or operations is less than nd. In other words, encoding Tornado codes has complexity $O(nd)$, which is linear.

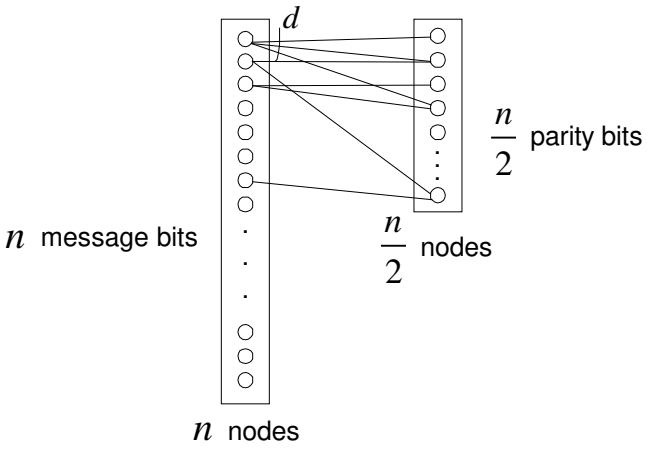

Figure 3 Tornado codes.

We next work on the decoding of Tornado codes. We assume that the graph in Figure 3 has (α, δ) unshared neighbor property, where $\delta > 0$. In the simplest case, suppose no parity bits are lost, and $k \le \alpha n$ message bits are lost. The idea is to decode the message bits recursively by making use of the (α, δ) unshared neighbor property.

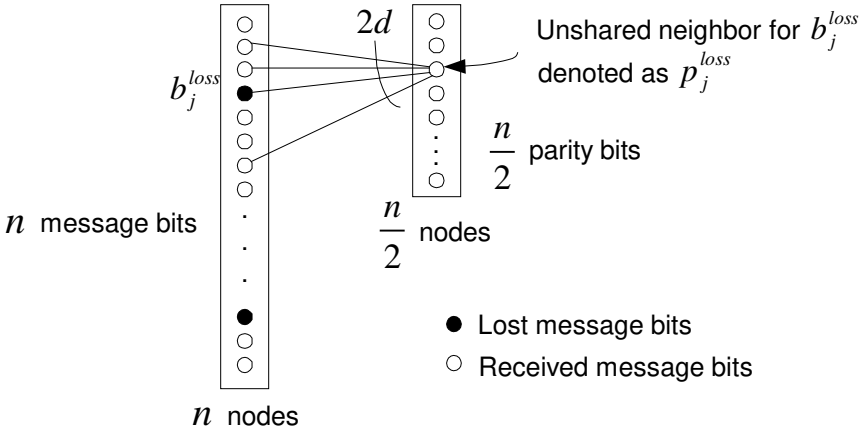

Figure 4 Decoding Tornado codes when all the parity bits are received.

Given the subset $k \le \alpha n$ message bits that have been lost, we know that at least δk of them have an unshared neighbor on the right. Take any message bit b_j^{loss} $(0 \le j < \delta k)$ among the δk ones and consider it jointly with its unshared neighbor p_j^{loss}. As is shown in Figure 4, if p_j^{loss} has degree $2d$, p_j^{loss} is the exclusive or of $2d-1$ received message bits and the lost bit b_j^{loss}. Therefore, b_j^{loss} is recoverable. After we recover the δk lost

message bits that have an unshared neighbor on the right, $k-\delta k$ lost message bits are still left. We can apply the above method again and again until all the lost message bits are recovered. The key really is that after we recover the δk lost message bits, some of the left $k-\delta k$ bits will have unshared neighbors that they did not own before. The above decoding method has linear time complexity, because every time we find a lost message bit, we only need to do at most $2d$ exclusive or operations.

The above algorithm helps decoding Tornado codes when no parity bits are lost. However, in practice this is not true because parity bits have the same probability to be lost as the message bits. We need to design a new scheme to encode and decode the message bits. Figure 5 shows such a scheme based on a recursive structure.

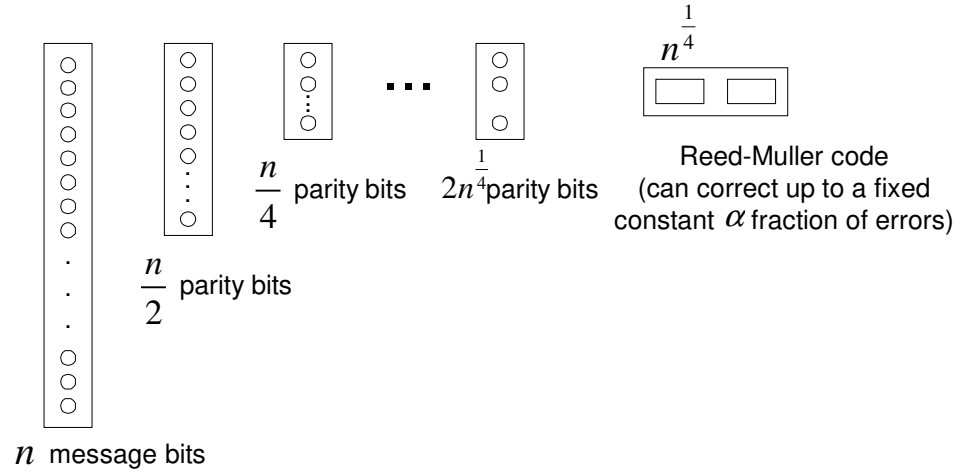

Figure 5 Enhanced Tornado codes in a recursive manner.

The idea is pretty straightforward. To protect the $\frac{n}{2}$ parity bits from erasures, we use another $\frac{n}{4}$ parity bits to protect them. We further protect the $\frac{n}{4}$ parity bits with $\frac{n}{8}$ parity bits, etc. At a certain point, e.g., when the number of nodes reduces to $n^{\frac{1}{4}}$, we stop the recursion and apply a Reed-Muller code to protect the $n^{\frac{1}{4}}$ nodes. The decoding process start from the Reed-Muller code and propagate to the left until we get all the lost message bits. Even the decoding of Reed-Muller code is in quartic complexity, the overall decoding time is still linear in n.

In the above scheme, we used $\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+\cdots+2n^{\frac{1}{4}}+n^{\frac{1}{4}}+\mathrm{O}\left(n^{\frac{1}{4}}\right)\approx n$ parity bits to protect the n message bits. Here $\mathrm{O}\left(n^{\frac{1}{4}}\right)$ is the number of parity bits used in Reed-Muller codes to protect $n^{\frac{1}{4}}$ bits. Since the decoding process is from right to the left, if we fail at a certain level, we will fail to decode the erased message bits. In worst case, if all the errors

are in the Reed-Muller code, only $O\left(n^{\frac{1}{4}}\right)$ errors can crash the code. If the Reed-Muller code is decoded successfully, due to the (α,δ) unshared neighbor property we know that at a level with *m* message or parity bits, we can correct at most αm erasures.

We next study for a certain level with *m* message or parity bits, what is the probability that we will have more than αm erasures. If such a probability is low, we can claim that with a high probability Tornado codes are able to correct the erasures.

Assume that the erasures happen at random positions[2]. Let the probability of each bit being erased is *p*. Let X be a random variable that denotes the number of erasures happened in the *m* bits. Obviously, X follows a *Binomial distribution*. The expectation of X is:

$$E(X)=mp\,.$$

Given a certain value *r*, we have:

$$\Pr(X=r)=\binom{m}{r}p^r(1-p)^{m-r}$$

where $\Pr(X=r)$ stands for the probability of random variable X taking value *r*. The probability that we have more than *r* erasures can be represented by:

$$\Pr(X\geq r)=\sum_{k=r}^{m}\binom{m}{k}p^k(1-p)^{m-k}$$

It can be proved[3] that:

$$\Pr(X\geq r)=\sum_{k=r}^{m}\binom{m}{k}p^k(1-p)^{m-k}\leq\binom{m}{r}p^r$$

Moreover, given the inequality that $\left(\frac{m}{r}\right)^r\leq\binom{m}{r}\leq\left(\frac{me}{r}\right)^r$, we can have:

$$\Pr(X\geq r)\leq\binom{m}{r}p^r\leq\left(\frac{me}{r}\right)^r p^r=\left(\frac{mpe}{r}\right)^r=\left(\frac{e\cdot E(X)}{r}\right)^r$$

Suppose we choose value *r* that is greater than $2e\cdot E(X)$, we get:

---

[2] This is a tricky assumption and we know that this is not true in the real environment. That is, in practice erasures are most likely to be bursting. However, we can assume that we randomly permute the order of the message bits before sending so that the erasures are random to the original sequence of message bits.

[3] We can go through the math to prove this, but here is how to do it conceptually. $\binom{m}{r}p^r$ is the probability that we choose *r* bits out of *m* bits and all of them are erased. This is an over-count of $\Pr(X\geq r)$ because the same case in $X\geq r$ may be counted may times in the above probability. For example, let $m=3$, $r=1$. Let the 3 bits be b_0,b_1,b_2. The case b_0 and b_1 are both erased (b_2 not erased) will be counted twice in the above probability (once when we choose b_0 and say it is erased and again when we choose b_1 and say it is erased).

$$\Pr(X \geq r) \leq \left(\frac{e \cdot E(X)}{r}\right)^r \leq \left(\frac{e \cdot E(X)}{2e \cdot E(X)}\right)^r = \left(\frac{1}{2}\right)^r$$

If $r \leq \log_2 m$, $\Pr(X \geq r) \leq \frac{1}{m}$, which is a very small number given m is large. In the Tornado code, at any level, the number of erasures we can correct has the order of $r = \Theta\left(n^{\frac{1}{4}}\right)$. Therefore, with high probability we will be able to correct the erased codes.

The above analysis gives a quick bound on the ability Tornado codes can correct the erased codes. We can give a better bound by applying the inequality:

$$\Pr(X \geq (1+\varepsilon)E(X)) \leq e^{-\frac{\varepsilon^2 E(X)}{3}}.$$

The probability that Tornado codes fail at a certain level with m message or parity bits is:

$$\Pr(X \geq \alpha m) = \Pr(X \geq (1+\varepsilon)pm) \leq e^{-\frac{\varepsilon^2 pm}{3}}$$

where $\varepsilon = \frac{\alpha}{p} - 1$ and $m \geq n^{\frac{1}{4}}$. It can be easily shown that if

$$\varepsilon \geq \sqrt{\frac{6}{\alpha}} \cdot \frac{\sqrt{\ln n}}{n^{\frac{1}{8}}} = O\left(\frac{1}{n^{\frac{1}{8}}}\right)$$

then the failure probability is bounded as:

$$\Pr(X \geq \alpha m) \leq e^{-\frac{\varepsilon^2 pm}{3}} \leq \frac{1}{n}.$$

Roughly speaking, as ε is very small, we can conclude that Tornado codes have a high probability to be successfully decoded when $p \approx \alpha$.

References

[1] E. Upfal. An O(logN) *deterministic packet routing scheme*. In 21st Annual A CM Symposium on Theory of Computing, pages 241-250. ACM, May 1989.

[2] M. Capalbo, O. Reingold, S. Vadhan and A. Wigderson, "Randomness Conductors and Constant-Degree Lossless Expanders", To appear in STOC '02, May 19-21, 2002, Montreal, Quebec, Canada.