# Online codes
**(Extended Abstract)**

Petar Maymounkov*
petar@cs.nyu.edu
TR2002-833

November, 2002

**Abstract**

We introduce *online codes* – a class of near-optimal codes for a very general loss channel which we call the *free channel*. Online codes are linear encoding/decoding time codes, based on sparse bipartite graphs, similar to Tornado codes, with a couple of novel properties: *local encodability* and *rateless-ness*. Local encodability is the property that each block of the encoding of a message can be computed independently from the others in constant time. This also implies that each encoding block is only dependent on a constant-sized part of the message and a few preprocessed bits. Rateless-ness is the property that each message has an encoding of practically infinite size.

We argue that rateless codes are more appropriate than fixed-rate codes for most situations where erasure codes were considered a solution. Furthermore, rateless codes meet new areas of application, where they are not replaceable by fixed-rate codes. One such area is information dispersal over peer-to-peer networks.

*Secure Computer Systems Group, New York University, full-time PhD candidate, 715 Broadway, Room 715, New York, NY 10003

# 1 Introduction

Many communication settings entail a number of participants exchanging information over a channel that may lose packets. A typical example is communication via UDP over the Internet. One simple way to get around packet loss is to employ a protocol in which receiving parties acknowledge received data with the sender. This solution, although widely used (e.g. in TCP), proves awkward in some situations because it requires multiple rounds of communication, which is inappropriate for many situations such as multicast.

In the cases when there is a bound to how many packets a channel can lose, a more robust solution to the above problem is in place, often called *forward-error correction* or FEC. Forward-error correction refers to a 1-round (feedback-less) protocol where the sender prepares an encoding of the original message and sends it over the channel. In turn, the receiver is assured to be able recover the original message from the packets that it ends up receiving. The size of the encoded message is bigger than the size of the original message to make up for the expected losses. The receiver recovers the original message as soon as they receive at least as big a part of the encoding as the size of the original message. It is standard practice to think of the message as consisting of blocks. A *block* is a fixed-length bit string, which represents the smallest logical unit of information. For transmission over UDP, e.g., it makes sense to use blocks of size 512 bytes, because UDP channels lose information at the IP packet level.

A common model for a loss channel is a channel that loses no more than a fraction $\delta$ of the packets, also referred to as a channel of capacity $1 - \delta$. It is long known that there are codes of rate $R = 1 - \delta$ for transmission over such channels. This means that a message of size $n$ blocks can be encoded into $n/R$ transmission blocks, so that any $n$ of the transmission blocks can be used to decode the original message. An example of such codes are Reed-Solomon codes. Another model for a bounded-loss channel is the *erasure channel* introduced by Elias [8], where a channel is said to have capacity $1 - \delta$ if it loses each packet independently with probability $\delta$. Elias proved that there exist codes of rate $R$ for any $R < 1 - \delta$ that can be used to transmit over channels of capacity $1 - \delta$. For an erasure channel a rate $R$ code encodes a message of size $n$ into a transmission of size $n/R$, so that the original message can be recovered from most portions of the encoding of size $n$. A drawback of Reed-Solomon codes and Elias' codes is that they take at least $O(n \log n)$ time to encode and/or decode. To overcome this problem, a new class of codes for the Elias' erasure channel was introduced in [4, 5], called Tornado codes. Tornado codes require linear encoding and decoding time (with no hidden big constants) in return for being nearly-optimal, which means that they transmit at rates $R = 1 - \delta(1 + \epsilon)$ for any $\epsilon > 0$. The fast encoding and decoding times have made Tornado codes probably the only ones that can be used with big messages in practice.

In this work, we look at a more general erasure channel, which we call the *free erasure channel*, that has no constraints on its loss rate. Consequently we propose online codes, inspired by Tornado codes, but fit for this more general scenario. The free erasure channel has no constrain on which blocks it drops, as long as it is not based on the content on the blocks. Naturally, the original message of size $n$ blocks will not be recoverable until at least $n$ encoding blocks have been transmitted through the channel. No fixed-rate erasure code could be used for this channel. For any fixed-rate $R$ encoding there is no guarantee that $n$ blocks of the $n/R$ encoding blocks will be transmitted. This motivates the need for "rateless" codes. Formally, a *rateless* code is a function $\mathcal{C}(M, i)$ that takes as arguments the message and an integer $i$ and produces as a result the $i$-th encoding block of $M$'s encoding; furthermore, a rateless code has *recall* $R$, if all (or most) portions

of the infinite encoding of size $n/R$ can recover the original message. The recall of a rateless code is then a number in the interval $(0, 1]$ and a rateless code is optimal if it has recall of 1.

One example of an optimal rateless code is the following. Let the block size be of bit length $k$. Treat the message blocks as coefficients of a polynomial in $\mathbb{F}_{2^k}$, then define the $i$-th encoding block to be equal to the polynomial's value at $i \in \mathbb{F}_{2^k}$. Clearly, the decoding procedure will be able to recover the original message by interpolating the polynomial from any $n$ encoding blocks (and their locations). Even though this code is optimal, it is not practical as polynomial interpolation requires more than $O(n)$ time. Here we present *online codes* – a class of nearly-optimal, i.e. with recall $1 - \epsilon$ for any $\epsilon > 0$, rateless codes with constant-time encoding per block and linear time decoding algorithms. We call them "online", because one can encode on the fly. Our design and analysis is heavily inspired by Tornado codes [4, 5].

From a practical perspective, a free erasure channel could be any channel with an unknown or variable loss rate. In such situations one might be tempted to use a fixed-rate erasure Tornado code with a low enough rate to match the worst expected loss rate of the channel. With this solution, however, one quickly runs into problems as very low rate Tornado codes require memory proportional to the encoding size to encode. This requirement is based on the fact that the encoding blocks of Tornado codes are interdependent and in order to compute one of them one oftens needs to have computed many other ones. Furthermore, Tornado codes are based on sparse bipartite graphs that need to be in memory in order to compute random encoding blocks.[1] For a file of 1GB and a 50-fold expansion (i.e. rate of approximately 1/50) one faces the need for likely more than 5GB memory or impractically many accesses to slower secondary storage. For precisely these inconveniences [3, 2] examine different non-optimal compromise solutions. This drawback motivates the need for *locally-encodable* codes that can compute every encoding block independently, needing only a very small part of the message in memory and no additional space.

One novel and much stronger motivation for rateless codes is dispersal of information, also called multi-source download, in peer-to-peer and other distributed networks. In particular, we look at the following scenario. Some $p$ nodes in a peer-to-peer system have complete knowledge of a file which is being requested for download by some other $q$ nodes. In order to ensure maximal availability of the file, each source node starts uploading random blocks of a rateless encoding of the file. This way, in case the source nodes disappear from the system before any of the destination nodes have completed download, the destination nodes will be able to combine their partial knowledge of the file without any overlaps, and hence will be very likely to be able to recover the entire file. The technique of using erasure codes for information dispersal has been mentioned in [1] in the context of fixed-rate codes. Unfortunately, the use of a fixed-rate code in the above scenario will set a limit, roughly equal to the expansion of the code (1 over the rate of the code) on the number people who can download the file concurrently without having any encoding blocks in common. It will, hence, require very low rate codes which are, as we mentioned earlier, not easily feasible in practice. A more thorough explanation of how to use rateless codes for information dispersal and why they are not replaceable by fixed-rate codes appears in [9].

There is a richer variety of applications where rateless codes are much more appropriate than traditional fixed-rate erasure codes. We are only going to mention one more briefly. In [3, 2] the authors suggest that erasure codes are a very natural solution for multicast connections on the Internet or for downloading from multiple mirror-sites. To alleviate the need for very low rate erasure codes, they suggest continuously broadcasting different permutations of a higher rate

[1] Proper application of Tornado codes requires that the encoding blocks are permuted randomly before sent.

erasure encoding. Once again, rateless codes are a more natural solution for these settings. In general, rateless codes seem to be naturally apt for situations when there is no coordination among different encoding parties of the same message.

## 2 Related work

The only other work that deals with rateless codes that we are aware of is a forthcoming paper [6] by Luby. Since the author declined to provide us with a copy of his paper, we are unable to compare our results until his paper becomes public at FOCS'02. The final version of our paper will include a comparison with Luby's work.

## 3 Setting

We start with a message $M$ of size $n$ blocks that we want to encode. We describe a simple constant-time randomized algorithm for generating an encoding block. We call encoding blocks *check blocks*. A check block is the exclusive-OR of $i$ message blocks, that are selected uniformly and independently from the ordered set of all message blocks (duplicate edges could occur); $i$ is called the *degree* of the check block. The degree is chosen randomly according to a probability distribution $\rho = (\rho_1, \rho_2, \rho_3, \ldots, \rho_F)$ (which we explicitly describe later), such that degree $i$ is chosen with probability $\rho_i$. The maximum degree $F$ is a constant. We have $\sum_{i=1}^{F} \rho_i = 1$, so that $\rho$ is a valid distribution. The main contribution of this work is the following theorem.

**Theorem 1.** *For any message of size $n$ blocks and any parameters $\epsilon > 0$ and $\delta > 0$, there is a distribution $\rho$ that can recover a $1 - \delta$ fraction of the original message from $(1 + \epsilon)n$ check blocks in time proportional to $n \ln((\ln \delta + \ln(\epsilon/2))/\ln(1 - \delta))$.*

Furthermore, in Section 4.5 we describe a way to first preprocess the original message into a composite message of roughly the same size. After recovering $1-\delta$ fraction of the composite message from an encoding as described above, a simple procedure recovers the entire original message. As a result we get:

**Theorem 2.** *For any message of size $n$ blocks and any parameter $\epsilon > 0$, there is a rateless locally-encodable code that can recover the original message from $(1+\epsilon)n$ check blocks in time proportional to $n \ln(1/\epsilon)$.*

In Section 4 we derive constraints on $\rho$ which will ensure that a message is mostly recoverable from $(1 + \epsilon)n$ check blocks with high probability. In the same section we also explain how to preprocess a message into a composite message to ensure full recovery. Then in Section 5 we exhibit a specific class of distributions $\rho$ that fulfill these constraints. Section 6 explains the results from our experiments. Section 7 discusses a generalization of the construction and the connection to Tornado codes.

# 4 Analysis

Our approach to proving the main result will be as follows. We consider a message of $n$ blocks and $(1+\epsilon)n$ check blocks, generated according to the procedure described in Section 3. We then show that for any fixed $\epsilon > 0$ and $\delta \in (0,1)$, if the distribution $\rho$ conforms to certain constraints, we can recover all but a fraction $\delta$ of the message blocks with an overwhelming probability in $n$. We set $\beta = 1+\epsilon$ for the sake of convenience and conformity with the notation in [4, 5]. Finally, in Section 4.5 we describe two different techniques of preprocessing the message. In conjunction with the check blocks, these techniques ensure that the message will be decoded to completion.

## 4.1 The decoding process

The decoding process consists of one simple step: Find a check block $c$, all of whose message blocks are recovered, except for one. Recover the missing message block by solving for it. If we call this message block $m_x$, we have $m_x = c \oplus m_1 \oplus \cdots \oplus m_{i-1}$, where $m_1, \ldots, m_{i-1}$ are the recovered message blocks that are adjacent to $c$. Apply this step until no more message blocks can be decoded.

Following [5, 4], we can think of the message and check blocks as vertices of a bipartite graph, where the message blocks are the vertices on the left and the check blocks are the vertices on the right. We use the term *node* to stand for a block or a vertex. In this bipartite graph each check node is adjacent to exactly the message nodes that comprise it (in terms of the exclusive-OR). We call this graph $G$.

We think of the decoding process in terms of a graph process. Define an edge's *right* (respectively *left*) *degree* to be the degree of its right (respectively left) node in the context of $G$. Then the basic decoding step will be: Find an edge of right degree 1, mark its left node as recovered and remove all edges incident to this node. The process terminates when there are no more edges of right degree 1.

The main idea of the analysis will be to show that a randomly selected message node $v$ is recoverable with high probability by running the above graph process just on a small subgraph of $G$ around $v$. It then follows that $v$ is recoverable with at least as high a probability if the graph process is run on $G$. Furthermore, the subgraph we choose is of constant size with high probability in $n$, which later helps us argue that the total number of recovered message nodes is close to its expectation.

Formally, we are going to look at a sub-graph $G_l$ (for $l$ a constant) of $G$, which is chosen as follows:

1. Pick an edge $(v, w)$ of $G$ randomly and uniformly, and call $v$ the *root of* $G_l$.
2. Remove $(v, w)$ from $G$.
3. $G_l$ will consist of all vertices of $G$ that are reachable within $2l$ hops from $v$ and all edges of $G$ that connect any two of these vertices.

### 4.2 Left node degree distribution

Our first goal will be to show that with high probability (as $n$ goes to $\infty$) $G_l$ is a tree of constant size. As an intermediate step to this result, we start by showing that the degree of a fixed left node is Poisson-distributed with a constant mean. To see this, we can think of the way check blocks are created as follows. Step one: all $\beta n$ check blocks choose their degrees, i.e. every right node chooses how many edges to have. Step two: all $\beta n$ right nodes connect their edges uniformly at random to the left nodes. Step one determines the total number of edges $\mathcal{E}$, which is easily seen to be $\mathcal{E} = \beta\mu n \pm \mathrm{O}(\sqrt{n})$, after applying a Chernoff bound, where $\mu = \sum_{i=1}^{R} i\rho_i$ is the average right node degree. Once the number of edges is determined, we can equivalently think of step two as throwing $\beta\mu n$ balls (the edges) into $n$ bins (the left nodes). It is then a standard balls and bins result that a fixed bin (right node) will have $d$ balls (degree $d$) with probability asymptotically equal to $\lambda_d = e^{-\mu\beta}(\mu\beta)^d/d!$ as $n$ grows large. Since $\lambda = (\lambda_1, \lambda_2, \dots)$ is the Poisson distribution with mean $\mu\beta$, each left node has degree bounded by a constant, independent of $n$, with arbitrarily high (in the constant) probability.

Next, we observe that the probability that a fixed left node $v$ has degree $d$, after having revealed some constant number of edges of $G$, is asymptotically equal to $\lambda_{d-a}$, where $a$ is the number of revealed edges that are incident to $v$. This is so because it corresponds to having already thrown some constant number of balls in bins, which asymptotically doesn't affect the distribution of the number of additional balls to be thrown in the bin we are interested in.

At this point we are in position to argue that $G_l$ is tree-shaped. $G_l$ is created by first selecting and revealing a random edge $(v, w)$, then we reveal $v$'s neighbors, then their neighbors and so forth for $2l$ steps. In this process, each time we are about to reveal the edges of a left node that we've reached, it will have a constant number of edges. This follows from our discussion above. Overall, all left and right nodes revealed in the selection of $G_l$ will have constant-bounded degrees with arbitrarily high probability (in the constant). This means that $G_l$ will have constantly many vertices. Consequently, the probability that $G_l$ is not a tree will be the probability that there is a cycle in $G_l$, and hence will be proportional to $1/n$. Therefore, as $n$ grows large enough we can safely condition the rest of our analysis on the event that $G_l$ is a tree. The subgraphs $G_l$ for which this fails to be true will be expected to be no more than constantly many. A more detailed derivation of this fact can be found in the appendix.

### 4.3 The And-Or tree analysis technique

This section briefly explains the And-Or tree analysis technique introduced in [5], as it is the main tool we use for our analysis.

An *And-Or tree* $T_l$ is a randomly generated tree of maximum depth $2l$. The root of the tree is at depth 0, its children are at depth 1, their children at depth 2, and so forth. Internal nodes of the tree that are at depths $0, 2, 4, \dots, 2l-2$ are going to be labeled *OR-nodes*, whereas nodes at depths $1, 3, 5, \dots, 2l-1$ will be labeled *AND-nodes*. The tree will be generated top-to-bottom, starting with the root node. Each node will independently choose how many children to have. For the OR-nodes this choice is made according to a fixed distribution $(\alpha_0, \alpha_1, \dots, \alpha_A)$, such that an OR-node has $i$ children with probability $\alpha_i$. Similarly, each AND-node chooses how many children to have according to the distribution $(\beta_0, \beta_1, \dots, \beta_B)$. Nodes at depth $2l$ will be leaf nodes and so they won't have any children.

Furthermore, each leaf node is going to be assigned the value 0 or 1 independently, with $y_0$ being the probability that it is 0. We are going to be interested in the probability that the root node evaluates to 0, if we treat the tree as a boolean circuit. To add further generality to this construct, each OR-node is independently short-circuited to 1 with probability $a$, whereas each AND-node is independently short-circuited to 0 with probability $b$. OR-nodes with no children are assumed to have a value of 0, whereas AND-nodes with no children are assumed to have a value of 1.

Our goal is to compute $y_l$ – the probability that $T_l$'s root evaluates to 0. Since the OR-nodes at depth 2 in $T_l$ are independent $T_{l-1}$ And-Or trees, it makes sense to compute $y_l$ recursively as a function of $y_{l-1}$ – the probability that the root of a $T_{l-1}$ And-Or tree evaluates to 0. We use the following lemma from [5]:

**Lemma 1 (The And-Or Tree Lemma).** *The probability $y_l$ that the root node of a $T_l$ And-Or tree evaluates to 0 is $y_l = f(y_{l-1})$, where $y_{l-1}$ is the probability that the root node of a $T_{l-1}$ And-Or tree evaluates to 0, and*

$$f(x) = (1-a)\alpha(1-(1-b)\beta(1-x)), \text{ for}$$
$$\alpha(x) = \sum_{i=0}^{A} \alpha_i x^i \text{ and } \beta(x) = \sum_{i=0}^{B} \beta_i x^i.$$

The proof of this fact is straightforward. We now proceed to applying the And-Or tree technique to our codes.

### 4.4 Putting it all together

Our strategy will be to show that for a randomly selected $G_l$, we can recover the value of the message block, corresponding to the root of $G_l$ with constant, but arbitrarily high, probability $1-\delta$ depending on our choice of $\rho$ and $\epsilon$. We model the process of decoding $v$ in the context of $G_l$, via an And-Or tree. Informally, $v$ will be recoverable if at least one of its adjacent check nodes is useful to decode $v$. An adjacent check block will be useful to $v$, if all of its adjacent message blocks except for $v$ are recoverable. They are recoverable, if in turn their adjacent check blocks are useful to decode them, and so forth.

Formally, we think of $G_l$ as an And-Or tree in the following way. The root of the And-Or tree is $v$, its children are the check nodes adjacent to $v$, their children are the message nodes adjacent to them and so forth. This way the message nodes of $G_l$ map to the OR-nodes of the tree and the check nodes map to the AND-nodes. An AND-node is assigned the value 1 if either it has no children in the tree (meaning it has degree 1 in $G$ and is immediately good to recover its adjacent message node, which is its parent in the tree), or if all of its child OR-nodes have value 1. An OR-node is assigned the value 1, if we have recovered the value of the underlying message block, which can only be if at least one of its child AND-nodes is 1. Let $y_l$ be the probability that the root of the And-Or tree $v$ has the value 0, i.e. the message block $v$ is not recoverable in $G_l$. We would like to show that $y_l$ can get arbitrarily close to 0. In order to model the decoding process via this And-Or tree we first need to compute the distribution on the number of children of OR and AND nodes.

In our construction of the And-Or tree, the probability $\lambda_j^*$ that an OR-node has $j$ children equals the probability that the message node of a randomly-selected edge has node degree $j+1$ in $G$, conditioned on the fact that some constant number of other edges (not incident to $v$) might

have been revealed already. This conditioning is formally necessary when we discuss lower level OR-nodes, because by the time we have gotten to them, we would have already revealed the edges leading to them from the root $v$. Since as mentioned earlier, we can asymptotically disregard the fact that those edges have been revealed, we have $\lambda_j^* \approx \lambda_j = (j+1)\lambda_{j+1}/\mu$. On the other hand, the probability $\rho_j^*$ that an AND-node has $j$ children equals the probability that a check node $w$ has degree $j+1$ given that $(v,w)$ is in the edge-set of $G$ for some fixed message node $v$, again also condition on the fact that a constant number of other edges not incident to $w$ have been revealed already. Since $w$ generates its edges independently, those other revealed edges make no difference, hence for $\rho_j^*$ we get:

$$\begin{aligned}
\rho_j^* &= \Pr[\deg w = j+1 | (v,w)] \\
&= \frac{\Pr[\deg w = j+1 \text{ and } (v,w)]}{\Pr[(v,w)]} \\
&= \frac{\rho_{j+1}\left(1-\left(\frac{n-1}{n}\right)^{j+1}\right)}{\sum_{i=1}^{R}\rho_i\left(1-\left(\frac{n-1}{n}\right)^{i}\right)}, \text{ using L'Hopital's rule} \\
&\rightarrow \frac{(j+1)\rho_{j+1}}{\mu}
\end{aligned}$$

We then define the polynomials

$$\lambda(x) = \sum_{i=0}^{\infty}\lambda_i = e^{\mu\beta(x-1)}, \text{ and } \rho(x) \quad = \frac{1}{\mu}\sum_{i=0}^{R-1}(i+1)\rho_{i+1}x^i$$

which correspond to the polynomials $\alpha(\cdot)$ and $\beta(\cdot)$ from the And-Or lemma, respectively. Further, the probability of short-circuiting an OR-node to "1" is 0. And, the probability of short-circuiting an AND-node to "0" is also 0. As a result of a recursive application of the And-Or lemma on the And-Or tree induced by $G_l$, we get that

$$y_l = f^l(1), \text{ where } f(x) \quad = \lambda(1-\rho(1-x))$$

We would like to be able to make $y_l$ as small as possible, i.e. for any fixed $\delta$, we need to make sure that there exists a fixed $l$ such that for large enough $n$ we have $y_l = f^l(1) < \delta$. This condition is easily seen to be equivalent to

$$\lambda(1-\rho(1-x)) < x, \text{ for } x \in [\delta, 1] \tag{1}$$

Finally, to show that the number of recovered message blocks is close to its high expected value when inequality (1) holds, we need to show that the number of And-Or trees of each shape is close to its expected value. (Each And-Or tree is rooted at a message node.) This is achieved by a standard edge-exposure martingale [7].

In summary, if the distribution $\rho$ conforms with inequality (1), we can decode an arbitrarily high fraction $1-\delta$ of the message blocks with high probability in $n$. (The dependence on $n$ comes from Azuma's inequality when applying the edge-exposure martingale).

In section 5 we exhibit a specific class of distributions $\rho$ that possess the required properties. Now we turn our attention to what needs to be done in order to ensure that the entire message gets recovered to completion.

### 4.5 Decoding to completion

We present two ways to preprocess the original message into a *composite message*, such that any random $1-\delta$ fraction of the composite message can recover the original message. Moreover, the composite message will be no bigger than $(1+k\delta)n$ blocks for some constant $k$. We then apply the encoding and decoding procedures described so far to the composite message. This will ensure that the entire original message can be recovered, after we have recover $1-\delta$ fraction of the composite message. Furthermore, since $\delta$ can be made arbitrarily small, the number of check blocks necessary to decode the composite message (and hence the original message) can be made to equal $(1+\epsilon')n$ for any $\epsilon' > 0$.

The first approach for preprocessing the message is to encode it using a $1-\delta$ rate erasure code, which has linear encoding and decoding time. This can be achieved by using the version of Tornado codes described in [4]. This version of Tornado codes uses the XOR-based check blocks technique in conjunction with a conventional erasure code (e.g. Reed-Solomon) in order to ensure that the probability of failure to recover the entire message reduces to 0 as $n$ grows. Since the probability that our decoding process fails to recover $1-\delta$ fraction of the composite message also approaches 0 as $n$ grows, altogether the probability of not recovering the original message goes to 0 as $n$ gets large. It is also worth noting that $(1-\delta)$-rate Tornado codes are very cheap to compute, because the encoding is the message itself appended by an additional $\delta' n$ check blocks, where $\delta' \sim \delta$. This approach for preprocessing the original message is good in theory, but it is somewhat awkward in practice, because it requires the use of an additional conventional erasure code (e.g. Reed-Solomon), which complicates the implementation. One can get around this problem at the cost of sacrificing the property that the message fails to be recovered completely with probability approaching 0 in $n$. Our next approach does that.

The second approach is a simple recipe for preprocessing the original message into a composite message of size $(1+k\delta)n$. Such that the original message fails to be recovered completely with constant probability proportional to $\delta^k$. This solution is entirely sufficient in practice, as $\delta^k$ can easily be made smaller than $2^{-50}$. The composite message will consist of the original message appended by $k\delta$ *auxiliary blocks*. The auxiliary blocks are much similar to the additional check blocks used in the tail arguments of [4, 5]. Like the check blocks already described, each auxiliary block is going to be the exclusive-OR of some number of the original message blocks. Again, we think of the original message blocks and the auxiliary blocks as the left and right nodes, respectively, of a bipartite graph $G^*$. This time, each message block will have a fixed degree $k$ with respect to the auxiliary blocks, and it will choose its adjacent auxiliary blocks randomly and uniformly. The decoding process for recovering the original message from the auxiliary blocks will be the same as the one described in Section 4.1. It is then not hard to prove that a missing random $\delta$ fraction of the original message can be recovered completely from a random $1-\delta$ fraction of the auxiliary blocks with probability $1-\delta^k$. The idea is that any subgraph $H$ of $G^*$ consisting of no more than $\delta n$ message blocks and their neighbors is an expander, and so it can recover one more message block. Specifically, any $H$ that has $l$ left nodes and more than $kl/2$ right nodes must have at least one right node of degree 1 and hence at least one left node can be decoded. The proof is based on the argument that all subgraphs of $G^*$ consisting of no more then $\delta n$ left nodes and their right neighbors expand sufficiently. The details are given in Appendix B.

Finally, we want to comment on the fact that it is not possible to achieve linear-time decodable rateless codes that recover the entire original message with overwhelming probability in $n$, if they are based only on the XOR operation. This is the reason why in our first approach we relied on

Tornado codes, which use Reed-Solomon codes as a subroutine. This is in fact the case not just for rateless codes, but also for fixed-rate codes. As mentioned in [5], Tornado codes that don't use a conventional code cannot recover with overwhelming probability in $n$.

# 5 Asymptotically optimal codes

To substantiate our analysis so far, we describe a specific class of distributions $\rho$ that conform with inequality (1) and thus represent a class of rateless codes. For any given $\epsilon$ and $\delta$ we define $\rho$ as follows: $F = (\ln \delta + \ln(\epsilon/2))/\ln(1-\delta)$, $\rho_1 = 1-(1+1/F)/(1+\epsilon)$ and $\rho_i = (1-\rho_1)/\left((1-1/F)i(i-1)\right)$, for $i = 2, \ldots, F$. This will ensure that a fraction $1-\delta$ of a message of size $n$ can be decoded from $(1+\epsilon)n$ check blocks. Moreover, since $\epsilon \sim \delta$ (where the exact relationship depends on the preprocessing approach of choice), the total number of edges in the graph, and hence the decoding time, will be proportional to $\ln(1/\epsilon)n$. The details of this construction are purely mathematical and are given in Appendix C. This completes the proof of Theorems 1 and 2.

# 6 Experimental results

In order to verify that our asymptotic analysis actually works in practice, we implemented and tested our design. Our implementation was an unoptimized 150-line Java program that can encode and decode. We ran tests, using a code with $F = 2114$, $\delta = 0.005$, $\epsilon = 0.01$ and $k = 3$. We did multiple tests on messages of varying length. For messages of 5000 blocks the check block per message block ratio was in the interval $[1.025, 1.07]$; for message of size 32000 the interval was $[1.025, 1.04]$; and for messages of size 100,000 the interval was $[1.025, 1.028]$. The running time of each experiment was on the order of seconds. [2]

# 7 Discussion

We would like to mention briefly that one can consider a more general way in which check blocks are created. Instead of having every check block pick its adjacent message blocks uniformly from the set of message blocks, the choice can be skewed. In specific, one can use a continuous density function on the real interval $[0, 1]$, to represent the skew of the selection. The analysis of this variant is much the same. It gives more degrees of freedom to what the polynomial $\lambda(x)$ can be. This in turn changes the shape of the curve $\lambda(1-\rho(1-x))$. The shape of this curve determines how deep (in terms of $l$) the graph $G_l$ that we considered in our analysis should be, in order to make sure that we are close to the desired level of unrecovered message blocks. The bigger the depth of $G_l$, the bigger $n$ needs to be before the asymptotic analysis comes into effect. Ideally, one would want $\lambda(1-\rho(1-x))$ to be as concave as possible, but even when it is roughly linear (as it is in the case of the codes from Section 5), in practice, the codes converge for messages of as few as a 1000 blocks.

Also, we would like to mention the connection between online codes and Tornado codes [4, 5]. In Tornado codes, the bipartite graph of message and check blocks has **fixed** degree sequences. Which means that one is assured to have the desired fraction of nodes of each degree. Even though

[2] We used blocks of size 0 bits in order to measure the pure speed of the algorithm, without factoring in the actual XOR-ing cost, which depends on the block size.

in online codes, one only has an expectation for the fraction of nodes of each degree (on the left as well as on the right), asymptotically the outcome quickly approaches the expectation. (This is easy to see, using a martingale argument.) This makes the two kinds of codes very similar in nature despite the fact that their check blocks are generated in a different way.

# 8 Conclusion

The simplicity of implementation, their linear-time decoding, and most importantly their ratelessness, make online codes a great match for many practical scenarios. Furthermore, online codes seem to be a great solution for any setting in which multiple uncoordinated parties are encoding the same file. This makes them very promising for the peer-to-peer community, which is what motivated our work in the first place.

# 9 Acknowledgments

I would like to thank David Mazières and Yevgeniy Dodis for many discussions that helped simplify the design and fix problems. I also thank Srinivasa Varadhan for suggesting the right methodology for the proof in Appendix B. Thanks to Michael Mitzenmacher for being a great teacher and for introducing me to the exciting field of low-density parity-check codes. I would also like to thank Maxwell Krohn.

# References

[1] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed Content Delivery Across Adaptive Overlay Networks. In *SIGCOMM*, 2002.

[2] John Byers, Michael Luby, Michael Mitzenmacher. Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads. In *INFOCOM*, 1999.

[3] John W. Byers, Michael Luby, Michael Mitzenmacher, Ashutosh Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *SIGCOMM*, 1998.

[4] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann. Practical Loss-Resilient Codes. In *Symposium on Theory of Computation*, 1997.

[5] M. Luby, M. Mitzenmacher, and A. Shokrollahi. Analysis of Random Processes via And-Or Tree Evaluation. In *Symposium on Discrete Algorithms*, 1998.

[6] Michael Luby. LT Codes. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.

[7] Noga Alon, Joel Spencer. *The Probabilistic Method*. John Wiley & Sons, 2000.

[8] P. Elias. Coding for Two Noisy Channels. In *Information Theory, Third London Symposium*. Butterswort's Scientific Publicaions, pp. 61-76, 1955.

[9] Petar Maymounkov and David Mazieres. Rateless Codes and Big Downloads, 2002. Submitted for publication. Available at `http://www.scs.cs.nyu.edu/~petar`.

# A The left node degree distribution

This section details the discussion given in Section 4.2.

Let $\lambda = (\lambda_0, \lambda_1, \dots)$ be the degree distribution of a fixed left node, i.e. a fixed left node $v$ has degree $d$ in $G$ with probability $\lambda_d$ with respect to the random generation of the check nodes. Let $\mathcal{E} = \mathcal{E}(n)$ be a random variable equal to the number of edges in $G$. ($\mathcal{E}$ is random over the choices of degrees of the $\beta n$ check nodes.) Conditioned on $\mathcal{E}$, for $\lambda_d$ we have:

$$\lambda_d|\mathcal{E} = \binom{\mathcal{E}}{d} \left(\frac{1}{n}\right)^d \left(\frac{n-1}{n}\right)^{\mathcal{E}-d} \tag{2}$$

Next, we use a Chernoff bound to show that $\mathcal{E} = \mu\beta n + \mathrm{O}(\sqrt{n})$ with high probability. For $\mathcal{E}$ we have

$$\mathcal{E} = \sum_{i=1}^{\beta n} X_i, \tag{3}$$

where $X_i$ are iid[3] random variables with $\Pr[X_i = j] = \rho_j$. Set $\mu = E[X_i] = \sum_{i=1}^{R} i\rho_i$, and define $Y_i = (X_i - \mu)/R$; hence $E[Y_i] = 0$ and $|Y_i| \leqq 1$. Using a standard Chernoff bound from **Theorem A.1.16** in the Appendix of [7] (or equivalently, using Azuma's inequality) we get that:

$$\Pr\left[\left|\sum_{i=1}^{\beta n} Y_i\right| > a\right] < 2e^{-a^2/2\beta n} \tag{4}$$

By setting $a = \xi\sqrt{n}$ for some parameter $\xi$ and expanding the $Y_i$'s we get:

$$\Pr\left[\left|\sum_{i=1}^{\beta n}(X_i - \mu)\right| > R\xi\sqrt{n}\right] < 2e^{-\xi^2/2\beta} \tag{5}$$

Further, we can pick some constant but big enough $\xi$, so that we can safely assume from now on that:

$$\mathcal{E}(n) = \mu\beta n + \mathrm{O}(\sqrt{n}). \tag{6}$$

Now, we can plug $\mathcal{E}$ back into equation (2) to get an unconditional expression for $\lambda_d$:

$$\lambda_d = \binom{\mu\beta n + \mathrm{O}(\sqrt{n})}{d} \left(\frac{1}{n}\right)^d \left(\frac{n-1}{n}\right)^{\mu\beta n + \mathrm{O}(\sqrt{n}) - d} \tag{7}$$

Next, we asymptotically simplify the above equation for the case when $d$ is a constant independent of $n$. We get:

$$\lambda_d \approx \frac{1}{e^{\mu\beta}} \frac{(\mu\beta)^d}{d!} \tag{8}$$

[3] Independent and identically-distributed.

Since

$$\sum_{i=0}^{\infty} \frac{1}{e^{\mu\beta}} \frac{(\mu\beta)^i}{i!} = 1, \tag{9}$$

for any fixed $\delta > 0$, there is a fixed $f$, such that

$$\sum_{i=0}^{f} \frac{1}{e^{\mu\beta}} \frac{(\mu\beta)^i}{i!} > 1 - \delta. \tag{10}$$

This and the fact that $\lambda_d$ gets arbitrarily close to $((\mu\beta)^d/e^{\mu\beta}d!)$ as $n$ grows large mean that a fixed left node has degree bounded by a chosen constant with arbitrarily high probability in that constant. Applying a standard edge exposure martingale shows that the number of left nodes that have degrees no greater than a fixed constant is close to expected.

We now have that both left and right nodes have degrees bounded by constants (it is by construction for the right ones), and so it means that $G_l$ has no more than a constant number of vertices. That means that the probability that $G_l$ is not a tree, i.e. has a cycle, goes to 0 proportionally to $1/n$ as $n$ grows large.

# B Decoding to completion with auxiliary blocks

In this section of the Appendix, we will be proving that given a $\delta$ fraction loss of the composite message we can recover the original message with arbitrarily high, but constant, probability. Let the composite message consist of the $n$ message blocks and $\gamma n$ auxiliary blocks. We will show that the probability of failing to recover the original message will be proportional to $\delta^k$ for any $\gamma > k\delta$, where $k$ is the degree of the message blocks with respect to the auxiliary blocks.

First, a standard edge exposure martingale shows that both the original message blocks and the auxiliary blocks will be suffering a loss of a fraction $\delta$ each. We consider the bipartite graph $Q$ between the $\delta n$ missing original message blocks and **all** of the auxiliary blocks. Assuming that all auxiliary blocks are known, we bound the probability of failure to recover the missing message blocks. Later we argue that the same argument holds when we only consider the $1 - \delta$ auxiliary blocks that are actually known.

Generally, consider a bipartite graph $H$ that has $l$ left nodes, each of degree $k$, and more than $kl/2$ right nodes of non-zero degree. Then there has to be a right node of degree 1. Otherwise, we get a contradiction when we count the number of edges incident to the right nodes, because they come out to be more than $kl$. Since there is a right node of degree 1, we can decode one left node and then remove both nodes, ending up with a smaller graph. In view of this fact, we are going to prove that with overwhelming probability in $n$ there is no subset $U$ of the left nodes of $Q$ that *shrinks*. $U$ is said to shrink, if it has $|U|k/2$ or less neighbors on the right. In other words, we will be showing that all subgraphs of $Q$ are good expanders.

Let $l = |U|$. Fix a subset of size $kl/2$ of the $\delta n$ left nodes. The probability that all right neighbors of $U$ are within this subset is

$$\left(\frac{kl/2}{\gamma n}\right)^{kl}$$

Furthermore, the probability that $U$'s right neighbors are within any such subset is union-bounded by:

$$z(l) = \binom{\gamma n}{kl/2}\left(\frac{kl/2}{\gamma n}\right)^{kl}$$

This bound is biggest when $l = \delta n$, i.e. when $U$ is the set of all left nodes. And, finally, the probability that any subset $U$ of the left nodes shrinks is bounded by the union bound:

$$\begin{aligned} 2^{\delta n} z(\delta n) &= 2^{\delta n}\binom{\gamma n}{k\delta n/2}\left(\frac{k\delta}{2\gamma}\right)^{k\delta n} \\ &= \mathrm{O}\left(\left(2^{\delta}\frac{(1-\alpha)^{\alpha-1}}{\alpha^{\alpha}}\alpha^{k\delta}\right)^{n}\right), \text{ where } \alpha = k\delta/2\gamma \end{aligned}$$

When $\gamma > k\delta$ the base of the above exponent is smaller than 1. Hence, the probability that there exists a subset of the left nodes that shrinks vanishes as $n$ grows large. Which means that all missing message blocks can be decoded.

Last, we turn to the case when a $\delta$ fraction of the auxiliary blocks can be missing as well. In this case, much the same reasoning applies, but the probability that a subset of the left nodes is shrinking is even smaller. Therefore, the above analysis still provides a good upper bound. This time, however, with constant probability $\delta^k$ a message block is adjacent only to missing auxiliary blocks. So, we expect that a fraction $\delta^k$ of the missing $\delta n$ message blocks will not be recovered. Again, an edge-exposure martingale shows that the number of unrecovered blocks is close to expected. Overall, exactly a $\delta^{k+1}$ fraction of the message blocks end up not being recovered. If $\delta^{k+1} \ll 1/n$, the whole message will be recovered with sufficiently high probability, proportional to $1 - \delta^{k+1}$.

# C Good codes

Here we show why the class of distributions $\rho$ given in Section 5 fulfill inequality (1). Further, we show that the decoding time of these distributions is proportional to $\ln(1/\epsilon)n$.

For any given $\epsilon$ and $\delta$ we defined $\rho$ as follows:

$$F = \frac{\ln\epsilon + \ln\delta}{\ln(1-\epsilon)},\ \rho_1 = 1 - \frac{1+1/F}{1+\epsilon} \text{ and } \rho_i = \frac{1-\rho_1}{1-1/F}\frac{1}{i(i-1)}, \text{ for } i = 2,\ldots,F.$$

First, we verify that $\rho$ is a valid distribution, i.e. $\sum_{i=1}^{F}\rho_i = 1$. This is easy to check, using that:

$$\begin{aligned} \sum_{i=2}^{F}\frac{1}{i(i-1)} &= \sum_{i=2}^{F}\left(\frac{1}{i-1} - \frac{1}{i}\right) \\ &= 1 - 1/F \end{aligned}$$

Next, we proceed to find a good approximation of $\rho(x)$.

$$\begin{aligned}
\rho(x) &= \frac{1}{\mu}\sum_{i=0}^{R-1}(i+1)\rho_{i+1}x^i \\
&= \frac{1}{\mu}\left(\rho_1 + \frac{1-\rho_1}{1-1/F}\sum_{i=1}^{F-1}\frac{x^i}{i}\right) \\
&= \frac{1}{\mu}\left(\rho_1 + \frac{1-\rho_1}{1-1/F}\left(\sum_{i=1}^{\infty}\frac{x^i}{i} - \sum_{i=F}^{\infty}\frac{x^i}{i}\right)\right) \\
&= \frac{1}{\mu}\left(\rho_1 + \frac{1-\rho_1}{1-1/F}\left(-\ln(1-x) - \mathcal{E}(F,x)\right)\right), \text{ where } \mathcal{E}(F,x) = \sum_{i=F}^{\infty}\frac{x^i}{i}
\end{aligned}$$

Here $\mathcal{E}(F,x)$ is the error term in the Taylor series of $-\ln(1-x)$. Now we substitute the above expression for $\rho(x)$ in the left-hand side of inequality (1) to get:

$$\lambda(1-\rho(1-x)) = e^{-\beta\left(\rho_1 - \frac{1-\rho_1}{1-1/F}(\ln x + \mathcal{E}(F,1-x))\right)}$$

Then, inequality (1) becomes equivalent to:

$$\left(\frac{1-\rho_1}{1-1/F} - \frac{1}{\beta}\right)\ln x < -\frac{1-\rho_1}{1-1/R}\mathcal{E}(F,1-x) + \rho_1$$

To solve this inequality, we simply set the left-hand side to be less than 0, and the right-hand side to be greater than 0. This produces:

$$\frac{\mathcal{E}(F,1-x)}{1-1/F+\mathcal{E}(F,1-x)} < \rho_1 \leqq 1 - \frac{1+1/F}{\beta}$$

We set $\rho_1 = 1-(1+1/F)/\beta$ and find an $F$ for which

$$\frac{\mathcal{E}(F,1-x)}{1-1/F+\mathcal{E}(F,1-x)} < 1 - \frac{1+1/F}{\beta}$$

holds for all $x \in [\delta, 1]$. When $F > 2/\epsilon$, the above inequality reduces to $\mathcal{E}(F,1-x) < \epsilon/2$. To complete the analysis, we observe that

$$\mathcal{E}(F,x) < \xi, \text{ for all } x \in [0, 1-\delta] \text{ when } F > \frac{\ln\delta + \ln\xi}{\ln(1-\delta)}$$

and therefore, setting $F = (\ln\delta + \ln(\epsilon/2))/\ln(1-\delta)$ ensures that inequality (1) holds.

Furthermore, we set $\epsilon \sim \delta$, where the exact relationship depends on the preprocessing approach of choice. Then the number of edges will be:

$$\begin{aligned}
\sum_{i=1}^{F} i\rho_i &= n\frac{\epsilon - 1/F}{1+\epsilon} + n\frac{1}{1-1/F}\sum_{i=1}^{F}\frac{1}{i-1} \\
&= O(\ln(1/\epsilon)n).
\end{aligned}$$