Digital Fountains: A Survey and Look Forward

Michael Mitzenmacher[1]
Harvard University
Division of Engineering and Applied Sciences
e-mail: michaelm@eecs.harvard.edu

Abstract — **We survey constructions and applications of *digital fountains*, an abstraction of erasure coding for network communication. Digital fountains effectively change the standard paradigm where a user receives an ordered stream of packets to one where a user must simply receive enough packets in order to obtain the desired data. Obviating the need for ordered data simplifies data delivery, especially when the data is large or is to be distributed to a large number of users. We also examine barriers to the adoption of digital fountains and discuss whether they can be overcome.**

I. Introduction

Most network communication is based on TCP (the Transport Control Protocol), which treats data as an ordered sequence of packets. TCP uses retransmissions to guarantee that, from the receiver's point of view, all packets are received in order.

For some applications, the ordered-sequence paradigm of TCP is too restrictive. An alternative paradigm was introduced in [3], based on the idealized solution of a *digital fountain* (similar in spirit but more expansive than the similar idea of the information dispersal algorithm [23, 27]). A digital fountain has properties similar to a fountain of water: when you fill your cup from the fountain, you do not care what drops of water fall in, but only want that your cup fills enough to quench your thirst. With a digital fountain, a client obtains encoded packets from one or more servers, and once enough packets are obtained, the client can reconstruct the original file. Which packets are obtained should not matter.

In more detail, an idealized digital fountain should have the following properties:

- A source can generate a potentially infinite supply of encoding packets from the original data. Ideally, encoding packets can be generated in constant time per encoding packet given the original data.[2]

- A receiver can reconstruct a message that would require k packets to send using TCP once *any* k encoding packets have been received. This reconstruction should also be extremely fast, preferably linear in k.

Approximations to a digital fountain can be obtained from the idealized version by loosening the requirements in various ways. For example, the number of encoding packets could be limited; the encoding or decoding times could be slower; or the number of encoding packets required could be greater than k. The utility of an approximation depends on the target application.

In what follows, we first briefly describe (without great technical detail) codes that can provide approximate digital fountains, including the recent advances in what are called *fountain codes*.[3] We then describe some of the applications that have been proposed for digital fountains, and end by considering some of the barriers that have prevented digital fountains from becoming more widespread in practice.

II. Digital Fountain Constructions

A. Reed-Solomon codes

In theory, Reed-Solomon codes [28] can be used to develop a useful approximation to a digital fountain: a message of k symbols can be recovered after receiving k distinct encoding symbols. In practice, however, there are several complications. The field size F gives a limitation on the number of distinct encoding symbols that can be created; usually F is 256 or 65,536, corresponding to symbols of 8 or 16 bits. Larger fields introduce nontrivial overhead for the resulting field arithmetic operations. More importantly, standard algorithms for decoding Reed-Solomon codes require quadratic time, which are too slow for even moderate values of k. (Standard quadratic time encoding algorithms can similarly be too slow, although encoding can often be done in preprocessing.) A natural approach for dealing with this problem involves encoding over small blocks of data and then striping data from various blocks to cope with bursty errors. This approach lessens the total computation, but can significantly increase the number of packets that must be obtained before complete decoding. Reed-Solomon codes may certainly be suitable for some applications, particularly when only small blocks of symbols need to be encoded. In general, however, they do not adequately fulfill the promise of digital fountains. A

[1]This work was supported by NSF Grants CCR-9983832, CCR-0118701, and CCR-0121154.

[2]In this setting we use a standard machine model where random access to the original data takes only constant time.

[3]We have chosen to use the term digital fountain to refer to the network abstraction of an essentially infinite supply of encoding packets, and use the term fountain codes to refer to a class of codes that are useful in implementing a nearly ideal digital fountain. The terms are, however, nearly interchangeable, and they may become substitutes for each other in the future.

more detailed argument is presented in [3], and computational experiments exploring the tradeoffs between Reed-Solomon codes and Tornado codes (discussed below) are presented in [26].

Instead, many implementations of digital fountains are based on variations of low-density parity-check (LDPC) codes (introduced by Gallager [5]), which we now describe.

B. Tornado Codes

A breakthrough in this area was the development of Tornado codes, a class of LDPC codes designed for erasures [17]. Tornado codes can naturally be described in terms of graphs. In the original construction, the graph consists of many layers of nodes. For the first layer, there is one node for each symbol in the message, where we can think of a symbol as being bits organized as a packet. The first layer is connected by edges to a second layer of redundant nodes. The edges of the graph determine the symbols corresponding to the second layer. In the standard framework, the symbol for a node at the second layer is the exclusive-or of the symbols corresponding to its neighbors. A recursive construction is used, deriving a third layer from the second, and so on.

The interpretation of this graph structure is that the symbols in the first layer are variables, and the symbols in the subsequent layers are constraints on these variables. By choosing these constraints properly in a random fashion, one can guarantee that with high probability, the message can be decoded once enough information about the variables and constraints arrive. This is similar in spirit to Reed-Solomon codes, with one important difference. With Reed-Solomon codes, every constraint (or redundant symbol) depends on every message symbol; this is what makes encoding and decoding expensive. With Tornado codes, each constraint depends only on a few message symbols; on average, only a constant number of exclusive-or operations are required to generate each redundant symbol. In graph language, the graph representing the code is *sparse*, with very few edges; hence the term "low-density" to describe these codes. This property gives Tornado codes their fast encoding and decoding time, at the cost that now more than k packets are generally required to obtain enough information to decode successfully.

In [17], the following results were shown. Let $\epsilon > 0$ be a fixed overhead (say, 0.05), k be the number of message packets, and n be the number of encoding packets. There exist carefully chosen graphs (equivalently, codes) such that only $(1 + \epsilon)k$ encoding packets (chosen uniformly from all encoding packets) are necessary to decode with high probability, and the time to encode and decode is proportional to $n \ln(1/\epsilon)$. That is, there exist codes with very low overhead that are still fast – time linear in n – to decode.

The original construction has been simplified greatly; in particular, there is a construction which just uses two graph layers while maintaining linear encoding and decoding times [29]. The first layer includes both the message packets *and* the necessary redundancy. The constraints implied by the second layer are that the exclusive-or of the neighbors of each node in the second layer must equal 0; these constraints are enforced by choosing the redundancy appropriately.

While Tornado codes, because they require time only linear in n for encoding and decoding, offer a better approximation to digital fountains than Reed-Solomon codes for many applications, they still suffer from a powerful drawback. When using a Tornado code, one must determine the number of encoding packets that can be generated, or equivalently the rate of the code, ahead of time, as the encoding packets are determined by the graph representing the Tornado code. While theoretically one could use a code with a very large number of encoding packets, because the running time and memory required are proportional to the number of encoding packets, this is not viable in practice.

Subsequent work has led to codes that circumvent this drawback, effectively making the underlying graph for the code implicit rather than explicit, as we now describe. As a result, the codes are called *rateless*. No fixed rate need be determined ahead of time, and essentially an infinite stream of encoding packets can be constructed from the message data.

C. LT codes

LT codes [13] were the first practical realization of a rateless code. LT codes have a very simple description. An encoding symbol is generated in the following manner:

- Choose a *degree* d for the encoding symbol, according to a predetermined distribution.
- Choose d distinct message symbols uniformly at random, and set the encoding symbol to be exclusive-or of these d symbols.

LT codes have a similar graph structure to Tornado codes, but the graph is implicit, instead of explicit, in the sense that each encoding symbol tracks its own neighbors, and there is no predetermined global view of the graph. For this to work, each encoding symbol must have associated with it a list of its associated message nodes, or neighbors. In practice, this is accomplished by using additional information, such as the packet identification number, as a seed to a pseudorandom number generator that is used to determine the degree and neighbors. The sender and receiver agree on this generator and the distribution ahead of time; they constitute part of the code.

The term fountain codes, informally defined (see, e.g. [32]), refers to codes of this form, where encoding symbols are (independently) determined by a distribution on the message symbols, and the encoding symbols are essentially sums of these message symbols.

As is the case with Tornado codes, a great deal of care is necessary to choose the appropriate underlying degree distribution in order to make this approach effective. LT codes do have an advantage over Tornado codes, in that with Tornado codes, even after designing the degree distribution, some care must be taken to design the actual graph used as well. With LT codes, there is no explicit graph to optimize.

It was shown in [13] that using encoding symbols with average degree $O(\ln k)$, one can devise codes such that only $k + o(k)$ encoding symbols are necessary to decode with high probability. Encoding symbols can be generated on the fly in time proportional to $\ln k$, and decoding can be done in time proportional to $k \ln k$. These codes allow us to construct near-ideal digital fountains. Perhaps the only additional thing one might hope for is to reduce the average degree to a constant, and the decoding time to $O(k)$. This cannot be done in the strict LT framework; one can easily show that simply for every message node to have at least one neighbor when there are $O(k)$ encoding symbols, the average degree must be at least $\Omega(\ln k)$. However, using pre-coding as described in the next section, the average degree can be reduced to a constant.

D. Raptor Codes

Raptor codes [33, 32] extend the idea of LT codes one important step farther. LT codes suffer in that an average degree of $\Omega(\ln k)$ is needed to cover every message node (with high probability). To circumvent this, suppose that we first pre-code the message M by encoding it with a fixed erasure code, such as a Tornado code. We now treat the encoded version M' as the message, so that encoding symbols are the exclusive-or of packets of M', in a manner similar to LT codes. Now, since we do not need to recover every packet of M' in order to recover M, but instead just a constant fraction of the packets of M', the $\Omega(\ln k)$ bound on the average degree no longer applies. Indeed, with an appropriate design, for any constant $\epsilon > 0$ and sufficiently large k the message M can be decoded after receiving only $(1 + \epsilon)k$ packets with high probability, with the degree of each encoding symbol being $O(\ln 1/\epsilon)$ and the total decoding time being $O(k \ln 1/\epsilon)$. This coding approach was first described in a patent application [33], and was published independently in [24, 32]. The paper by Shokrollahi [32] has many additional details, including useful descriptions of extremely effective practical constructions and analysis techniques for codes of finite length.

Raptor codes currently give the best approximation to a digital fountain. A virtually limitless supply of packets can be generated on the fly after some small initial preprocessing, with each packet taking only constant time to produce. Decoding can be accomplished after receiving just a few percent more than the minimum of k encoding packets (with high probability), and requires space and time linear in the size of the original message. Moreover, very efficient implementations are possible.

III. Network Applications of Digital Fountains

A. Multicast

The digital fountain paradigm was introduced in the context of reliable multicast [3], such as for the delivery of new software products. The assumption is that a large number of users, roughly but not exactly overlapping in time, want to download the same file without loss. Packets are distributed over one or more multicast trees to the receivers, with packets being copied only when necessary at branches in the route. This avoids the overhead of sending a distinct copy of each packet for each user over the network.

The benefits of using encoded content in the context of multicast are readily apparent. As each user may experience independent losses, if we have users request retransmission from the source, the result will be a feedback implosion when the set of receivers is large. Using encoded data prevents the need for retransmission and the corresponding feedback. Other benefits include simple means of handling heterogeneous users and disparate start time. For example, if two receivers share all but the last hop of their routing paths, but have different download rates, packets can be sent at the higher rate along the path, and are dropped at the last hop to match the rate of the slower receiver. Since every received packet is useful, dropping packets in this way does not hurt the slow receiver, and allows the faster receiver to maintain its higher rate. In a similar manner, if two such receivers begin downloading at different times, the later receiver can immediately receive useful packets. Since the order in which packets are received does not matter, joining a multicast session already in session is trivial.

Even using digital fountains, multicast presents many challenging problem, including the issue of designing appropriate congestion control schemes in order to make the resulting traffic network friendly. The use of digital fountains can simplify the resulting protocols; more information and examples can be found in [2, 3, 14].

B. Downloading in Parallel

Just as digital fountains can ease multicasting from one source to many receivers, they can simplify one receiver downloading in parallel from many sources. While downloading in parallel can also be done without digital fountains [4, 30], they greatly simplify the underlying network issues. Using digital fountains, each source can independently produce an endless stream of encoding packets; because the available supply of encoding packets is essentially infinite, no collisions where the same encoding packet is received from multiple sources need occur. The receiver can close all connections after receiving enough packets, without any concern of where

the packets are coming from, or the loss rate or sending rate of each source. Byers, Luby, and Mitzenmacher consider downloading from multiple sources in parallel using Tornado codes [4]. With Tornado codes, if the k message packets are used to create n encoding packets with $n = ck$ for some fixed constant c, then some fraction of the packets received from multiple senders will be duplicates, reducing the efficiency of the parallel download. The more recent LT and Raptor codes therefore yield even better results.

A scenario imagined in [4] combines multicast and downloading in parallel: mobile wireless receivers capable of receiving from multiple sources obtaining information seamlessly from fixed multicasting sources spread throughout a building or metropolitan area.

C. One-to-Many TCP

Rost, Byers, and Bestavros describe a way of using digital fountain codes in an architecture, dubbed the Cyclone architecture, that has many of the advantages of multicast while utilizing TCP [31]. This is a potential advantage, as TCP is still generally more trusted on the network than content sent by UDP.

The standard problem with handling large, popular files using TCP is that separate state must be maintained for each connection. Specifically, for performance reasons, each connection should cache recently sent packets that may have been lost and therefore could need to be retransmitted in a timely fashion. The effect of maintaining such state is to limit the number of connections that can be effectively handled, because of the memory requirements for each connection.

The main point of Rost, Byers, and Bestavros is that the memory for connections requesting the same file can be shared using content encoded with a digital fountain in place of the original content [31]. If we think of the cache as being a buffer with fresh encoded data corresponding to a file continuously being pumped into it, then transmissions and retransmissions for a file can all be handled from the same buffer, across connections. The only difference is that when a retransmission occurs, the encoded packet that was originally sent will not be retransmitted, but another encoded packet will be sent instead. Most importantly, the flow control and congestion control mechanisms of TCP are completely maintained by this approach.

D. Streaming Video

While downloading and storing a complete movie can naturally be done in conjunction with a digital fountain, handling streaming video or on-demand delivery is significantly more challenging. The issue is latency: if the entire file of k packets is encoded in one block, then essentially k packets must arrive before decoding can begin, incurring an unreasonable amount of latency for many video-based applications.

An overview of previously suggested techniques as well as a system that uses digital fountains is given by Mahanti, Eager, Vernon, and Sundaram-Stukel [22]. Most solutions are based on schemes that break the video file into segments. The first segment can be played while the second segment is downloading, and so on for the entire video. Each segment can be encoded separately; erasure coding enhances the system by reducing extreme performance variations that can occur in the face of non-trivial packet loss. If the segment lengths increase geometrically in size, then only a logarithmic number of distinct segments are necessary. Issues such as the relative sizes of the segments and the ratio of the download rate to the playback rate are important for designing robust systems, and discussed in detail in [22].

E. Transmission over Overlay Networks

An alternative to IP multicast, which requires the network to have IP multicast available, is to establish a data transmission system on top of an overlay network. The overlay allows more control, avoiding many problematic issues and allowing better management. Multicast can still be used when effective.

Byers, Considine, Mitzenmacher, and Rost argue for digital fountains when using multicast or simply downloading data to many users in this environment [1]. Overlay networks are expected to adapt to changing network conditions, and thus should be robust against congested or unstable areas of the network. Using encoded data and avoiding the need for an ordered stream enhances adaptability to transient network conditions. Furthermore, with encoding, the multicast distribution is not restricted to a multicast tree. Encoding easily allows not only taking full advantage of the overlay network by allowing data to be downloaded in parallel from multiple sources, but also taking advantage of what is called *perpendicular bandwidth* between the peers themselves.

More specifically, in many cases it may beneficial for peers to provide information to each other, potentially while they are still downloading. For example, two users downloading a file might start at slightly different times, suffer different losses, or obtain packets at different rates from different sources. In this case, they might be able to productively collaborate by sharing received encoded packets. If either peer obtained the entire file, that peer could produce new encoded content using the appropriate code. An interesting case considered in [1] is when neither peer has obtained enough encoding packets to construct the original file. In this case the peers' sets of encoding packets may have a great deal overlap, since the encoding packets may have come from the same sources. The question is then how to quickly determine what encoding packets might be useful to the other peer. Various techniques for this reconciliation problem are explored in [1], as well as in [25]. The Bullet system, which uses many of the techniques suggested in [1], was implemented and tested in [7].

Kwon and Byers suggest another approach for large-scale distribution on overlay networks, based on loosely connected TCP streams [8]. The idea is similar to that for the Cyclone architecture. Generally, the problem with using TCP in a multicast setting on overlay networks is that one is forced to slow the connection to the speed of the slowest link in the tree; otherwise, an intermediate point will have to buffer packets for this slow link, creating the potential for a buffer overflow. To avoid this problem, in the ROMA architecture [8], the TCP connections can use encoded packets. A buffer of limited size can be used, and if the sending rate is too large for this buffer, packets are simply discarded. This process maintains a fresh supply of encoded packets available for fast connections, while ensuring reliability for the slow connections as well.

IV. Barriers to Adoption

The concept of a digital fountain was described in the original digital fountain conference paper [3] back in 1998. Given their seeming promise, one might hope that they would be in widespread use today. Despite interest from parts of the networking community, uses of the digital fountain paradigm remain relatively scarce. We consider here some of the significant barriers to adoption.

Patent protections: Most of the work that has been described above has been undertaken by employees of or consultants with the company Digital Fountain, Inc. As should be clear, the company has both spurred innovation and made a great deal of their work public for the research community. On the other hand, the company has a number of patents issued and pending that appear to cover both the fundamental theory and specific implementation designs for Tornado, LT, and Raptor codes. Issued patents in the area include [6, 9, 10, 11, 12, 15, 16, 19, 20, 21]. (This list does not include several pending applications, some of which can be found through the U.S. Patent Office. For example, there is a patent application covering various techniques covered in the discussion of Raptor codes that has not issued at this time [33].) As far as I currently know, these patents have never been used against any company or individual. However, these patents may have in the past and hold the potential in the future to serve as a barrier to researchers wishing to work in this area. For example, it is not clear whether the creation of a publicly available implementation of various LDPC schemes would infringe on Digital Fountain patents; see, e.g., the discussion in [26].

These patent protections will eventually disappear, but not for well over a decade. On the other hand, the existing patents provide strong motivation for developing alternative designs for digital fountains or approximate digital fountains.

Perceived complexity: Although several papers describe various coding techniques that can be used to build (approximate) digital fountains, building a high-quality coding engine for a digital fountain is a non-trivial task, with a large number of pitfalls. For example, the random structure chosen for the code can have a significant impact on the average amount of overhead required before decoding; these structures often need to be fine-tuned carefully for top performance.

Network researchers would prefer to use "off-the-shelf" components rather than design such a coding engine themselves. At this time, however, I am not aware of a standard public coding library that implements a "black-box" digital fountain that can be used for network experiments. (While Digital Fountain, Inc., advertises such a library, it is naturally not free nor publicly available.) A standard tool of this form would allow more experimentation among network researchers, encouraging adoption of digital fountain technologies. Of course, any such public implementation would have to be built with intellectual property situation in mind, as discussed above.

Lack of a "killer application": The digital fountain framework was designed to deal with the problem of large-scale multicast over the Internet, and its advantages in this domain are substantial. Unfortunately, IP multicast as a technology has failed to get off the ground, never finding significant acceptance. Various reasons have been given, including overall complexity, the need for widespread deployment to make it effective, the lack of management tools, and issues with IP multicast addresses. In part, the lack of acceptance may be due to viable alternatives, including replication via hosting services such as Akamai. In any case, multicast was the "killer application" for digital fountains, and while digital fountains can simplify or enhance other network solutions, it is not clear that they offer such dramatic advantages for other problems.

There is hope that appropriate killer applications will arise in the future. Video-on-demand systems offer one opportunity. Large scale distribution of data to wireless devices is another. While cell phones or digital assistants are natural targets, as wireless networking becomes a standard feature on automobiles, the possibility of using digital fountains to effectively broadcast software to millions of vehicles may also become compelling.

V. Conclusions

The field of erasure codes has moved dramatically forward over the last decade. Many new codes with outstanding performance derived from the LDPC framework have been developed, including families of rateless codes, which allow for very close approximations to an idealized digital fountain. The networking community has recognized the potential of these codes for solving problems related to multicast, and further applications continue to arise. Widespread adoption, however, remains an elusive goal. The development of new approximations to digital fountains, unencumbered by potential patent protec-

tion and accompanied by freely available reference implementations, could greatly speed adoption, and provides a theoretical and technical challenge to the community. Regardless, we believe that both theory and practice of digital fountains will continue to grow, with an increasing emphasis on integrating digital fountains into existing and new network applications.

Acknowledgments

I would like to thank John Byers for helpful discussions as well as assistance in tracking down references.

References

[1] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery over adaptive overlay networks. In *Proceedings of ACM SIGCOMM Conference*, pp. 47-60, 2002. Journal version to appear in *IEEE/ACM Transactions on Networking*, 2004.

[2] J. Byers, G. Horn, M. Luby, M. Mitzenmacher, and W. Shaver. FLID-DL: congestion control for layered multicast. *IEEE Journal on Selected Areas in Communications*, 20(8), pp. 1558-1570, October 2002.

[3] J. Byers, M. Luby, and M. Mitzenmacher. A Digital Fountain Approach to Asynchronous Reliable Multicast. *IEEE Journal on Selected Areas in Communications*, 20(8), pp. 1528-1540, October 2002. (A preliminary version appeared in ACM SIGCOMM '98, pp. 56-67).

[4] J. Byers, M. Luby, and M. Mitzenmacher. Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads. In *Proceedings of INFOCOM 1999*, pp. 275-283.

[5] R. G. Gallager. *Low-density parity check codes*. PhD thesis, MIT, 1963. Manuscript published by MIT Press.

[6] A. Haken, M. Luby, G. Horn, D. Hernek, J. Byers, and M. Mitzenmacher. Generating high weight encoding symbols using a basis. U.S. Patent #6,411,223. Issued June 25, 2002.

[7] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: high bandwidth data dissemination using an overlay mesh. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pp. 282-297, 2003.

[8] G.-I. Kwon and J. Byers. ROMA: Reliable overlay multicast with loosely coupled TCP connections. *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communication Societies (Infocom 2004)*, Hong Kong, March 2004.

[9] M. Luby. Information additive code generator and decoder for communication systems. U.S. Patent #6,307,487. Issued October 23, 2001.

[10] M. Luby. Information additive code generator and decoder for communication systems. U.S. Patent #6,373,406. Issued April 16, 2002. (continuation to #6,307,487).

[11] M. Luby. Information additive code generator and decoder for communication systems. U.S. Patent #6,614,366. Issued September 2, 2003. (continuation to #6,373,406).

[12] M. Luby. Information additive group code generator and decoder for communication systems. U.S. Patent #6,320,520. Issued November 20, 2001.

[13] M. Luby. LT codes. In *Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 271-282, 2002.

[14] M. Luby, V. Goyal, S. Skaria, and G. Horn. Wave and equation based rate control using multicast round trip time. In *Proceedings of ACM SIGCOMM Conference*, pp. 191-204, 2002.

[15] M. Luby, G. Horn, J. Persch, J. Byers, A. Haken, and M. Mitzenmacher. On demand encoding with a window. U.S. Patent #6,486,803. Issued November 26, 2002.

[16] M. Luby and M. Mitzenmacher. Loss resilient code with double heavy tailed series of redundant layers. U.S. Patent #6,195,777. Issued February 27, 2001.

[17] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, February 2001.

[18] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. Spielman, and V. Stemann. Practical loss-resilient codes. In *ACM STOC '97*, pages 150–159, 1997.

[19] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, V. Stemann, and D. Spielman. Message encoding with irregular graphing. U.S. Patent #6,163,870. Issued December 19, 2000.

[20] M. Luby, M. A. Shokrollahi, V. Stemann, M. Mitzenmacher, and D. Spielman. Irregularly graphed encoding technique. U.S. Patent #6,081,909. Issued June 27, 2000.

[21] M. Luby, M. A. Shokrollahi, V. Stemann, M. Mitzenmacher, and D. Spielman. Loss resilient decoding technique. U.S. Patent #6,073,250. Issued June 6, 2000.

[22] A. Mahanti, D. L. Eager, M. K. Vernon, and D. Sundaram-Stukel. Scalable On-Demand Media Streaming with Packet Loss Recovery. *IEEE/ACM Transactions on Networking*, Vol. 11, No. 2, pp. 195-209, April 2003.

[23] N. F. Maxemchuk, Dispersity Routing in Store and Forward Networks. Ph. D. thesis, University of Pennsylvania, May 1975.

[24] P. Maymounkov. Online Codes. NYU Technical Report TR2002-833, November 2002.

[25] P. Maymounkov and D. Mazières Rateless codes and big downloads. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems*, February 2003. Available in *Lecture Notes in Computer Science 2735*, pp. 245-255, Springer, 2003.

[26] J. Plank and M. Thomason. On the practical use of LDPC erasure codes for distributed storage applications. Technical Report UT-CS-03-510, University of Tennessee, September 2003.

[27] M. O. Rabin. Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance. *Journal of the ACM*, Volume 38, pp. 335-348, 1989.

[28] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8:300–304, June 1960.

[29] T. Richardson and R. Urbanke. Efficient encoding of low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2), pp. 638-656, 2001.

[30] P. Rodriguez and E. Biersack. Dynamic parallel access to replicated content in the Internet. *IEEE/ACM Transactions on Networking*, 10(4), pp. 455-465, 2002.

[31] S. Rost, J. Byers, and A. Bestavros. The Cyclone Server Architecture: streamlining delivery of popular content. *Computer Communications*, 25(4), pp. 403-412, 2002.

[32] A. Shokrollahi. Raptor codes. Preprint available at `http://algo.epfl.ch/pubs/raptor.pdf`.

[33] A. Shokrollahi, S. Lassen, and M. Luby. Multi-stage code generator and decoder for communication systems. U.S. Patent Application #20030058958.