A Digital Fountain Retrospective

John W. Byers
Boston University

Michael Luby
ICSI, Berkeley, CA

Michael Mitzenmacher
Harvard University

ABSTRACT

We introduced the concept of a digital fountain as a scalable approach to reliable multicast, realized with fast and practical erasure codes, in a paper published in ACM SIGCOMM '98. This invited editorial, on the occasion of the 50th anniversary of the SIG, reflects on the trajectory of work leading up to our approach, and the numerous developments in the field in the subsequent 21 years. We discuss advances in rateless codes, efficient implementations, applications of digital fountains in distributed storage systems, and connections to invertible Bloom lookup tables.

CCS CONCEPTS

• **Networks** → **Network algorithms**; *Transport protocols*;

KEYWORDS

erasure codes, fountain codes, reliable multicast

1 OVERVIEW

The original inspiration for working on designing efficient erasure codes was the problem of streaming video over lossy broadcast and multicast networks. The prevailing solutions at the time (the mid '90s) used feedback and retransmission schemes, each of which ran into scalability problems as the number of receivers grew large. An initial project at the International Computer Science Institute (ICSI) in Berkeley explored an orthogonal approach: use erasure codes to protect against packet loss [1], [2], [17].

The erasure code approach had a different scalability issue: the encoding and decoding complexity of the existing erasure codes at the time were too high to be practical as the size of the source data grew large. Another ICSI-led project tackled this issue, designing and implementing erasure codes, informally known as Tornado codes [21], [22], which are the first channel capacity achieving erasure codes with linear time encoding and decoding algorithms.

A growing realization was that erasure codes (including Tornado codes) had yet another scalability limitation: erasure codes were designed for a fixed *code rate*, defined to be the ratio of the source data size to the encoded data size. For example, a 1MB source file encoded with a rate $2/3$ erasure code produces a 1.5MB encoding. A problem with a fixed code rate is that it is difficult to predict in advance the amount of data loss, i.e., since at least 1MB of the 1.5MB encoding is needed to recover the 1MB source file, the receiver is not able to decode if the packet loss rate exceeds 33%.

For point-to-point delivery, a conservative overestimate of the packet loss rate can be used, and the receiver can explicitly signal the sender once enough encoding has been received. This can cause inefficiencies, i.e., if an excessive amount of encoding is lost then decoding can not occur and the receiver falls back to retransmission based protocols.

More fundamentally, in a multicast setting where concurrent receivers experience different packet loss patterns, efficiently orchestrating transmissions from a fixed amount of encoding (see, e.g., [31]), becomes unwieldy at best, and runs into significant scaling issues as the number of receivers grows.

The research described in [6], [7] (awarded the ACM SIGCOMM Test of Time award) introduced the concept of an erasure code *without* a predetermined code rate. Instead, as much encoded data as needed could be generated efficiently from source data on the fly. Such an erasure code was called a *digital fountain* in [6], [7], which also described a number of compelling use cases. For example, streaming applications naturally need to dynamically adjust in real-time the amount of encoded data to generate and send, particularly when the data loss rate (or rates) is not known in advance. As another example, independently generated fountain-encoded data for the same source data at independent locations could then be transmitted in parallel to the same receiver, without need for coordination [5]. Today, we call a digital fountain a *fountain code* (also sometimes called a *rateless code*).

2 INTELLECTUAL PRECURSORS TO FOUNTAIN CODES

In addition to work on rate-based erasure codes from the coding theory community, the theoretical computer science community had developed several core concepts that served as precursors to the digital fountain approach. Notably, Shamir's secret-sharing scheme [36] provided a means to share an encoded message amongst n parties, such that *any* subset of k parties could jointly decode the original message exactly (and where zero information was revealed to colluding subgroups of smaller size). His method elegantly involves viewing a secret as coefficients of a degree k polynomial, where sharing a secret and recovering a secret amounts to sampling n points from, and uniquely solving for, a degree k polynomial, respectively. Subsequently, Rabin's information dispersal algorithm (IDA) [35] devised a method to disperse a file F into $n > k$ encoded pieces, each of size $\frac{|F|}{k}$, such that any k of the n pieces suffice to reconstruct F. The key to this construction is to view the file symbolically, as a sequence of elements from a finite field, e.g., numbers mod p, and then use carefully chosen linear combinations of symbols of the file to produce the encoded pieces. Recovery of the file from k encoding pieces then amounts to inverting an $k \times k$ matrix. Ensuring that this is always possible necessitates that all possible recovery matrices are full rank. By encoding with a Cauchy matrix, this guarantee can be achieved, along with $O(k^2)$ decoding time.

As with these prior works, the digital fountain approach also had the *any* k of n concept at its core, viewing a reliable network transfer as an ordered sequence of k packet-sized chunks, but facilitating it through the transmission of $n > k$ *unordered* encoding packets. At network scale, k could be many orders of magnitude larger than envisioned in the IDA approach, so quadratic decoding was problematic. Also, as we have stated, it was desirable to be able to generate encoding symbols on the fly, so that n could be essentially unlimited and not specified in advance, but coding theory did not have efficient methods for this case. In particular, generating a new symbol should take only a small amount of time, with $o(k)$ operations, and ideally a constant number of operations (or constant on average).

Our ideal digital fountain had four properties:

- scalability up to very large k;
- ability to generate $n >> k$ encoded packets on the fly, with low per packet generation cost;
- exactly k of n encoded packets needed for decoding; and
- linear-time encoding and decoding.

Absent methods to achieve all the desiderata at once, our first realization of fountain codes, Tornado codes, could scale to large m and provided linear time decoding. Tornado codes allowed for $n > c \cdot k$ encoded packets for any desired constant $c > 1$, but the value for c would have to be chosen in advance, and the n packets were highly structured, so they were not naturally generated on the fly. Also, Tornado codes compromised on the exactly k of n requirement, and instead required receipt of $(1 + \epsilon) \cdot k$ packets to decode, where the constant ϵ would affect the encoding and decoding time through a $\ln(1/\epsilon)$ multiplicative factor. Several generations of fountain codes have made great strides over our original Tornado code implementation (some described below), with the standardized RaptorQ code [48], [37] providing a realization of fountain codes that achieves all four criteria essentially optimally.

3 PRACTICAL FOUNTAIN CODES

In a series of research papers [19], [27], [37], increasingly better practical designs of fountain codes were invented. This work developed not only the theoretical foundations of the new codes but also practical implementations.

The company Digital Fountain, Inc. developed a suite of products around fountain code technology, e.g., [38], [24]. Fountain codes have been used in deployments by major IPTV providers in Asia and Europe; as well as by movie studios, branches of the armed services, defense contractors, the postal service, and satellite system providers in the United States.

Fountain codes have also been widely adopted in network standards. A protocol suite for delivering multimedia data over broadcast and multicast networks based on fountain code technology was standardized through the Internet Engineering Task Force (IETF); see, for example [48], [49], [45], [44], [50], [43], [46], [47], [42]. The protocol suite based on the Raptor code specified in [45] is an integral part of the 3G/4G/5G Multimedia Broadcast/Multicast Service (MBMS) standard [51]. The more advanced RaptorQ code [48], [37], developed by a team at Qualcomm Technologies, Inc. (which acquired Digital Fountain) is integrated into the ATSC 3.0 standard to provide packet loss protection at the Internet Protocol (IP) layer for streaming and object data delivery [41].

4 FOUNTAIN CODE INSPIRED RESEARCH

There has been a large body of academic research related to fountain code technology. The surveys [26], [37], [30], provide general overviews of fountain code technologies.

We see many of the themes in the work on digital fountains reappear several years later in the work on network coding. In our work on digital fountains, we were focused on source encoding, treating the internal network as devices that just passed packets along, as had historically been the case. Network coding, taking advantage of the the growth of computing power and corresponding "smarts" within the network, allowed internal nodes in the network to participate in the encoding, creating new (linear) combinations of encoded data within the network, not just at the source [39], [25]. Content distribution was also one of the first applications suggested for network coding, and early application papers noted the natural connection [12]. Conceptually similar work employed variants of online coding at intermediaries in overlay networks and on wireless mesh networks. For example, COPE [15] demonstrates how nodes in a wireless mesh network can improve throughput by opportunistically XORing packets together and retransmitting them, using a flavor of online coding. A similar opportunistic coding strategy across data chunks was used for fast set reconciliation across overlay networks in content delivery applications [4].

Further, the concept of rateless codes has led to novel constructions of rateless codes for additional channels, including standard error channels. Examples include spinal codes [33] and rateless codes based on polar codes [18]. Rateless codes have also helped advance new ideas in coding, such as protographs and spatially coupled codes [8], [28].

Outside of coding, an outgrowth of the digital fountain work was a greater understanding of what has been referred to as *peeling algorithms* [14]. A peeling algorithm can be set up as a graph (or hypergraph), where vertices and the adjacent edges can be removed step by step according to some rule, until at the end of the process the graph is empty. For simple removal rules and sparse graphs, this leads to linear time algorithms in the greedy style. The Tornado codes used in the digital fountain paper provided a prominent example of a peeling algorithm.

A notable further example of a peeling based algorithm is the Invertible Bloom Lookup Table, or IBLT, which can be used to synchronize distributed data [11], [13], [10]. Suppose Alice and Bob each have a set of keys K_A and K_B, which are each very large, but the set difference $(K_A \cup K_B) - (K_A \cap K_B)$ is itself very small. It would be useful if both parties could determine the set difference with a message of size proportional to the size of the set difference, instead of the size of the actual sets. IBLTs allow this to happen, using a mechanism reminiscent of Tornado codes: Alice XORs her keys into a small array, each key being XORed into a small number of cells based on a shared hash function; Bob receives the results and XORs his keys into the array using the same hash function; in the final array, only keys in the set difference remain, and they are then peeled out of the structure. The connections between IBLTs and coding have been noted [29]. IBLTs are now starting to find uses in networking as a foundational data structure; they have been suggested as a means for synchronizing blockchains [40], [32].

Fountain codes have frequently been suggested as a building block for distributed storage systems, as in [3] [16],[34] [9]. In a recent example, a *liquid system* [23] proposes using fountain codes, lazy repair, and flow storage organization to provide durable object storage based on spreading redundantly generated data across a network of hundreds to thousands of potentially unreliable storage nodes. Liquid system can be operated to enable flexible and essentially optimal combinations of storage durability, storage overhead, repair bandwidth usage, and access performance. Recent work [20] proves information theoretic lower bounds on the tradeoffs between storage overhead and repair bandwidth, showing that a liquid system is close to optimal in these tradeoffs.

5 THE FUTURE OF FOUNTAIN CODES

Although fountain code technology has been around for several years, and has been useful in many deployments, it is still not ubiquitous. One potential reason for this is that there are patents on the underlying fountain code technology, including several patents owned by Qualcomm Technologies, Inc. Qualcomm has published an IPR statement in the IETF for the RaptorQ code[1], which mirrors the licensing commitment Qualcomm, Inc. has made with respect to the MPEG DASH standard. This may resolve significant patent issues.

A second issue is that developing high-performance implementations of such codes are not straightforward. Projects to develop implementations suitable for deployment are therefore useful; one such ongoing project is Codornices [52]. At the current time, the source code for the Codornices project implementation of the RaptorQ fountain code is available to commercial entities for unlimited deployment for a flat annual license fee.

ACKNOWLEDGMENTS

The authors acknowledge the following support. Byers is supported by the National Science Foundation under grant CNS-1815016. Luby is supported by the National Science Foundation under grant CNS-1936572. Mitzenmacher is supported by the National Science Foundation under grants CCF-1535795 and CCF -1563710.

REFERENCES

[1] A. Albanese, J. Blömer, J. Edmonds, M. Luby, M. Sudan. Priority Encoding Transmission. *IEEE Symposium on Foundations of Computer Science*, 1994.

[2] A. Albanese, J. Blömer, J. Edmonds, M. Luby, M. Sudan. Priority Encoding Transmission. *IEEE Transactions on Information Theory*, Vol. 42, No. 6, November 1996.

[3] C. Anglano, R. Gaeta, M. Grangetto. Exploiting Rateless Codes in Cloud Storage Systems. *IEEE Transactions on Parallel and Distributed Systems*, pp. 1313-1322, Vol. 26, No. 5, 2014.

[4] J. W. Byers, J. Considine, M. Mitzenmacher, S. Rost Informed content delivery across adaptive overlay networks *ACM SIGCOMM*, pp. 47-60, 2002.

[5] J.W. Byers, M. Luby, M. Mitzenmacher. Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads. *IEEE INFOCOM*, New York, NY, pp. 275 – 283, 1999.

[6] J.W. Byers, M. Luby, M. Mitzenmacher, A. Rege. A digital fountain approach to reliable distribution of bulk data. *ACM SIGCOMM*, pp. 56-67, 1998.

[7] J.W. Byers, M. Luby, M. Mitzenmacher. A Digital Fountain Approach to Asynchronous Reliable Multicast. *IEEE J. on Selected Areas in Communications*, Special Issue on Network Support for Multicast Communication, Vol. 20, No. 8, October 2002, pp. 1528 – 1540.

[8] T. Chen, K. Vakilinia, D. Divsalar, and R. Wesel. Protograph-based Raptor-like LDPC codes. *IEEE Transactions on Communications*, Vol. 63, No. 5., pp. 1522-1532, 2015.

[9] A. Dimakis, V. Prabhakaran, K. Ramchandran. Distributed Fountain Codes for Networked Storage. 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings.

[10] D. Eppstein, M. Goodrich. Straggler identification in round-trip data streams via Newton's identities and invertible Bloom filters. *IEEE Transactions on Knowledge and Data Engineering*, pp 297-306, Vol. 23, No. 2, 2010.

[11] D. Eppstein, M. Goodrich, F, Uyeda, G. Varghese. What's the difference?: efficient set reconciliation without prior context. *ACM SIGCOMM*, pp. 218-229, 2011.

[12] C. Gkantsidis, P. Rodriguez. Network coding for large scale content distribution. *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, pp 2235-2245, 2005.

[13] M. Goodrich, M. Mitzenmacher. Invertible Bloom lookup tables. *2011 49th Annual IEEE Allerton Conference on Communication, Control, and Computing (Allerton)*, pp 792-799, 2011.

[14] J. Jiang, M. Mitzenmacher, J. Thaler, Parallel peeling algorithms. *ACM Transactions on Parallel Computing (TOPC)*, Vol. 3, No. 1, 2016.

[15] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, J. Crowcroft, XORs in The Air: Practical Wireless Network Coding. *ACM SIGCOMM*, Pisa, Italy, 2006.

[16] Z. Kong, S. Aly, E. Soljanin. Decentralized Coding Algorithms for Distributed Storage in Wireless Sensor Networks. *IEEE Journal on Selected Areas in Communications*, Vol. 28, No. 2, pp. 261-267, 2010.

[17] B. Lamparter, A. Albanese, M. Kalfane, M. Luby. PET - Priority Encoding Transmission: A New, Robust and Efficient Video Broadcast Technology. *ACM Multimedia*, San Francisco, CA, November 1995.

[18] B. Li, D. Tse, K. Chen, H. Shen. Capacity-Achieving Rateless Polar Codes, *IEEE International Symposium on Information Theory (ISIT)*, pp. 46-50, 2016.

[19] M. Luby. LT Codes. *IEEE Symposium on Foundations of Computer Science*, November 16-19 2002, pp. 271-282.

[20] M. Luby. Repair rate lower bounds for distributed storage. Submitted to *IEEE Transactions on Information Theory*, July 2019.

[21] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, V. Stemann. Practical Loss-Resilient Codes. *ACM Symposium on Theory of Computing*, 1997.

[1] Available at https://datatracker.ietf.org/ipr/2554/

[22] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman. Efficient Erasure Correcting Codes. *IEEE Transactions on Information Theory*, Special Issue: Codes on Graphs and Iterative Algorithms, pp. 569-584, Vol. 47, No. 2, February 2001.

[23] M. Luby, R. Padovani, T. Richardson, L. Minder, P. Aggarwal. Liquid cloud storage. *arXiv:1705.07983 [cs.DC]. ACM Transactions on Storage*, Vol. 15, Issue 1, Article 2, April 2019.

[24] M. Luby, M. Watson, T. Gasiba, T.Stockhammer, W. Xu. Raptor Codes for Reliable Download Delivery in Wireless Broadcast Systems. *IEEE Consumer Communications and Networking Conference*, Las Vegas, Nevada, USA, January 2006

[25] D. Lun, D. Shah, M. Médard, R. Koetter. Efficient operation of wireless packet networks using network coding. *First International Workshop on Convergent Technology*, Vol. 5, 2005.

[26] D. MacKay. Information Theory, Inference, and Learning Algorithms. *Chapter 50: Digital Fountain codes*. Cambridge University Press, September, 2003.

[27] P. Maymounkov. Online codes. Technical report, New York University, 2002.

[28] D. Mitchell, M. Lentmaier, D. Costello. Spatially Coupled LDPC Codes Constructed from Protographs. *IEEE Transactions on Information Theory*, Vol. 61, No. 9, pp. 4866–4889, 2015.

[29] M. Mitzenmacher, G. Varghese. Biff (Bloom filter) codes: Fast error correction for large data sets. *2012 IEEE International Symposium on Information Theory Proceedings*, pp 483-487, 2012.

[30] M. Mitzenmacher. Digital fountains: A survey and look forward. *IEEE Information Theory Workshop*, pp 271-276, 2004.

[31] J. Nonnenemacher, E. Biersack, D. Towsley. Parity-based loss recovery for reliable multicast transmission. *ACM SIGCOMM*, pp. 289 – 300, 1997.

[32] A. Ozisik, B. Levine, G. Bissias, G. Andresen, D. Tapp, S. Katkuri. Graphene: Efficient Interactive Set Reconciliation Applied to Blockchain Propagation. *ACM SIGCOMM*, 2019.

[33] J. Perry, P. Iannucci, K. Fleming, H. Balakrishnan, D. Shah. *ACM SIGCOMM 2012*, pp. 49-60, 2012.

[34] J. Plank and M. Thomason. On the Practical Use of LDPC Erasure Codes for Distributed Storage Applications. Technical Report CS-03–510, University of Tennessee, 2003.

[35] M. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM (JACM)*, 36(2), 1989, pp. 335-348.

[36] A. Shamir. How to share a secret. *Communications of the ACM*, 22 (11), 1979, pp. 612-613.

[37] A. Shokrollahi, M. Luby. Raptor Codes. *Foundations and Trends in Communications and Information Theory*, 2011, Vol. 6: No 3-4, pp 213-322.

[38] T. Stockhammer, M. Luby, M. Watson. Application Layer FEC in IPTV Services. *IEEE Communications Magazine*, Vol. 45, No. 5, May 2008, p. 94-101

[39] J. Sundararajan, D. Shah, M. Médard. ARQ for network coding. *2008 IEEE International Symposium on Information Theory*, pp 1651-1655, 2008.

[40] A. Widrum. How the Magic of IBLTs Could Boost Bitcoin's Decentralization. Bitcoin Magazine, Nov. 2015.

[41] Advanced Television and Systems Committee. ATSC Standard: Signaling, Delivery, Synchronization, and Error Protection, (A/331), June 2019.

[42] Internet Engineering Task Force (IETF) RFC3453. The Use of Forward Error Correction (FEC) in Reliable Multicast. M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, J. Crowcroft. December 2002.

[43] IETF RFC3738. Wave and equation based rate control building block. M. Luby, V. K Goyal. April 2004.

[44] IETF RFC5052. Forward Error Correction (FEC) Building Block. M. Watson, M. Luby, L. Vicisano. August 2007.

[45] IETF RFC5053. Raptor Forward Error Correction Scheme for Object Delivery. M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer. October 2007.

[46] IETF RFC5775. Asynchronous Layered Coding (ALC) Protocol Instantiation. M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, J. Crowcroft. April 2010.

[47] IETF RFC5651. Layered Coding Transport (LCT) Building Block. M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, J. Crowcroft. October 2009.

[48] IETF RFC6330. RaptorQ Forward Error Correction Scheme for Object Delivery. M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, L. Minder. August 2011.

[49] IETF RFC6681. Raptor Forward Error Correction Schemes for FECFRAME. M. Watson, T. Stockhammer, M. Luby. August 2012.

[50] IETF RFC6726. FLUTE - File Delivery over Unidirectional Transport. T. Paila, M. Luby, R. Lehtonen, V. Roca, R. Walsh. November 2012.

[51] Multimedia Broadcast Multicast Service (3GPP TS 26.346). 3rd Generation Partnership Program technical specification, Ver. 6.0.0, April 2005.

[52] Codornices Project, International Computer Science Institute. www.codornices.info.