Paper title: A Digital Fountain Approach to Reliable Distribution of Bulk Data

Background and context

This paper was published in 1998 in ACM Sigcomm (acceptance rate 12% in that year), a highly respected top Tier conference in communications and computer networks which started in 1977.

At the time of publication, there had been a number of methods to deal with reliable data communication over networks. The dominant end-to-end method was using Jacobson's TCP in the transport layer or ARQ (Automatic Repeat Requests) when Unicast was used. When feedback was not possible when using storage media like CD's, for example, redundancy needed to be added to the transmitted information to allow for forward error correction. One of the most popular methods which operated on fixed-size blocks (block coding) was Reed Solomon codes. These had been made very popular due to the ubiquitous use in CD's, DVD's and hard disk drives.

One of the problems with traditional Reed Solomon codes was that they didn't behave well over erasure channels and were inherently poor at handling random bit errors. In 1995 Kafane et al devised Reed-Solomon erasure codes based on Cauchy matrices to deal with this limitation. This algorithm could encode and decode the bit stream in quadratic time and was the state of the art at the time of this publication.

The authors of this paper saw the potential to use an erasure code known as the Tornado code, developed by Luby, to distribute bulk data to a large number of clients over a multicast or broadcast network more efficiently and quickly then existing Reed-Solomon based methods of the day.

Overview

When distributing data to multiple clients over a large heterogeneous network like the Internet there will be a large amount of variety of reliability between paths from the source to the receivers. There was also the challenge of users choosing to access the data stream at different times. This article sought to provide a solution to this problem which was not scalable or adequate using retransmission or local repair. The solution was a digital fountain code which was new form of erasure code with the ability to deal with users joining a data stream at any point in time. It had the unique property that the source data could be reconstructed from any subset of the encoding packets as long as this was the same number of packets as the source data.

The key advantage of using this erasure code approach was that redundant data could be used to recover lost source data at the receivers and no feedback was required back to the source. This also meant that receivers experiencing different packet loss levels could recover the data.

The author's define the ideal protocol for bulk data distribution as follows:

- Reliable: Guaranteed delivery to all receivers
- Efficient: in terms of amount of time to encode/decode and total packets required to decode
- On demand: Download can be initiated at any time
- Tolerant: Should handle a heterogeneous population of receivers

A class of digital fountain codes known as Tornado codes was proposed previously by the 2nd author , Michael Luby, with the property that it was able to encode/decode symbols an order of magnitude faster than Reed-Solomon codes which were being used at the time. To achieve this it required slightly more than the number of source packets and so traded a small increase in decoding inefficiency for a substantial decrease in encoding and decoding times.

Tornado codes were described in two previous papers in 1997 and 1998 by Luby and Mitzenmacher. Here is a short summary: A Tornado code only used the xor-operation during decoding and encoding. It broek up the sender stream into k equal sized blocks of P bytes and checksums were computed on each of these blocks to determine if a block was damaged during transmission. It also calculated a set of parity blocks of size P which mapped to a subset of the k input blocks. Thus a total of n = k+l encoding packets were created and these all had a fixed length of size P. The subsets for each parity block were generated in a semi-random fashion using a specific distribution that allows the input file or message to be recovered with a very high probability.

The authors describe the intuition behind Tornado codes impressive encoding/decoding performance compared to Reed-Solomon codes and reduce it down to two important differences. Reed-Solomon codes make use of a very dense system of linear equations compared to Tornado codes random equations that are sparse. Reed-Solomon codes made use of matrix inversion and matrix multiplication field operations whereas Tornado codes only made use of exclusive-or operations. As a result Tornado codes encoding and decoding times grew linearly with an increase in size of source data (number of packets) and Reed-Solomon grew quadratically.

A set of experiments are performed which compare the best Reed-Solomon erasure codes available at the time, which where based on Cauchy matrices, and Tornado Z which was a particular distribution pattern for mapping source packets to redundant packet in Tornado codes. A set of files from size 250 KB to 16 MB were used and all runs used a packet length of 1KB. A stretch factor of 2 was used where k packets where encoded into n=2k packets. For a 250 KB file, Reed-Solomon encoded the data in 4.6 seconds whereas Tornado Codes performed this in 0.11 seconds. For a 16M file, Reed-Solomon encoded the data in 30802 seconds whereas Tornado Codes performed this in 3.93 seconds (3 orders of magnitude faster). Similar impressive results were shown for the decoding benchmark.

Due to the probabilistic nature of Tornado codes, it was important to understand the decoding inefficiency. Recall that Reed-Solomon has a decoding inefficiency of 1 as it could decode the k source packets using k encoded packets whereas Tornado codes need slightly more than k packets. Using 10000 trials on 16000 1KB packets, the average decoding inefficiency was 1.0536 with a standard deviation of 0.0073. This was considered a small price to pay for the drastically decreased encoding/decoding times of Tornado codes.

It was clear that Reed-Solomon erasure codes were not effective when encoding large files, and authors like J. Nonnenmacher and L. Rizzoc devised an interleaved coding mechanism to deal with this in the late 1990's. The concept was to group packets into blocks of length k and then stretch each block with m standard erasure codes. Reed-Solomon was then applied to these shorter blocks in the hope of decreasing the decoding time. The choice of the value of k became a tradeoff between decoding speed and decoding inefficiency which was caused by receiving occasional packets for blocks that had already been reconstructed successfully. The authors then did a comparison of Tornardo Z to this interleaved coding scheme where k was chosen so that decoding inefficiency was comparable to Tornado Z and k was chosen so that decoding time was comparable to Tornado Z.

When the length of the blocks for interleaved Reed-Solomon was chosen so that the efficiency was comparable to Tornado Z, Tornado Z still had a decoding time which was between 14 times and 212 times faster than interleaved Reed-Solomon for a 16MB file for loss rates between 1% and 50% respectively. When the block length (k=20) was chosen so that the decoding times were comparable, worst-case decoding inefficiency was comparable when using 1 receiver but worst-case decoding inefficiency grew quickly (1.2 to 1.5 for 10% loss) as the number of receivers grew (1 to 10000) in Interleaved coding and stayed fairly static in Tornado Z (1.06 to 1.1). A block length

of 50 was shown to achieve better decoding efficiency close to that of Tornado Z but no detail was given about the decoding time penalty except to say that it was reasonable in practice. The decoding inefficiency of Tornado Z as the file size grew was also shown to be almost constant compared to Interleaved Reed-Solomon which grew super-linearly.

In order to understand the impact of bursty loss patterns on Tornado and Reed-Solomon codes, some trace data from the Mbone multicast network, which had between 6 and 20 clients experiencing loss rates between 1% and 20%, were used to check some of the metrics already discussed. A random starting point from the trace data was chosen for each broadcast to simulate downloading a file of various lengths. The results with this trace were very similar to the results obtained with non-bursty data, with Tornado still maintaining near constant decoding inefficiency as file size grew and Reed-Solomon growing super-linearly.

The authors then go on to describe an experimental system which implements tornado codes on layered-multicast. Layered-multicast had been advocated by McCanne and Jacobson in 1996 to allow receivers to subscribe to a subset of multicast layers, which have increasing transmission rates, based on the width of the bottleneck link to the source and network congestion conditions. Congestion control was achieved by synchronization points which are specially marked packets in the stream and thus receivers are not required to provide any congestion control feedback or coordinate join attempts. These were important properties to use digital fountain in the context of layered-multicast as receiver-to-source and inter-receiver communication was undesirable.

Encoded packets in the Tornado code could now be scheduled across multiple layers and these had to be scheduled carefully in order to minimize the duplicate packets that a client would receive. In this implementation the transmission rates were geometrically increasing and so for example, layer 4 would have 4 packets scheduled and layer 3 would have 2 packets scheduled. So to receive 8 encoded packets. 2 rounds of packet reception would be required at layer 4 and 4 rounds would be required at layer 3. It was shown that it was better for a receiver to remains at a fixed subscription level when packet loss was sufficiently low and *Bhattacharyya et al* showed that a a transmission schedule always existed that could exploit this one level property. The authors acknowledged that if levels changed frequently, the optimal schedule was still an open research problem.

An experiment was set up which measures how well Tornado performs over a single layered multicast protocol and over a layered multicast protocol. The results measured the efficiency of the system using three metrics

- decoding inefficiency = # distinct packets received / # source data packets
- distinctness inefficiency = Total # packets received / # distinct packets received
- reception inefficiency = decoding inefficiency x distinctness inefficiency

A quicktime move of approximately 2 megabytes was sent between a server and 2 clients which all existed on 3 different subnets at Berkeley, Cornell and CMU with varying degrees of delay and loss rates sometimes as high as 20%. To generate higher loss rates, the base layer rate was pushed artificially high to cause the router along the path to drop packets. Decoding was only done after approximately 1.055 x (total encoding packets) was received to avoid redundant computation.

The results show that distinctness inefficiency remains at approximately 1 for loss rates less than 50% in the single layer case, but increases to approximately 1.2 thereafter. However in the case where 4-layer multicast was used, distinctness inefficiency begins to increase after about 15% packets loss. This was due to the congestion control in multi-layer multicast causing subscription levels to switch and packets being duplicated that were already received at other layers. But at high loss rates the distinctness inefficiency remains low for multi-layer multicast because receivers only subscribe to the base layer. The decoding inefficiency was approximately 1.07 for all packet loss levels, which was only slightly more than those in the standard experiments due to the mandatory

wait period imposed. This meant that reception inefficiency essentially tracked the distinctness inefficiency throughout.

The authors conclude by saying that Tornado codes are ideally suited to reliable multicast protocols and are closer to ideal digital fountain codes than previous systems based on Reed-Solomon erasure codes. Other applications of tornado codes are also mentioned such as dispersity routing in which a source injects packets along multiple paths in the network in order for the receiver to recover data once a sufficient number of packets has arrived. They also mention mirrored data, where clients can access multiple mirror sources simultaneously and aggregate packets they receive.

Paper analysis

This paper did a good job at convincing the audience that Tornado codes where a far better alternative to Reed-Solomon codes for reliable delivery of bulk data across lossy channels. This paper, together with the original paper describing tornado codes, was also beginning of series of other fountain codes based on a similar concept (Raptor codes, LT codes, Online codes). LT codes (Luby transform codes) were created by and named after Michael Luby himself. Raptor codes are now used for mobile cellular wireless broadcast and multicast and also used by the DVB-H standard.

It was useful to start out the paper with a description of an ideal theoretical protocol that meets all the requirements of reliable delivery of bulk data across lossy channels Similar to the way in which all coding schemes can be compared to the Shanon channel capacity limit, this allowed the reader to have this set of requirements in their head and understand the tradeoffs that would need to be made.

This paper was an attempt to bridge the original piece of theory work done in the first Tornado code paper by Luby to the real world of reliable data delivery in multicast and broadcast networks which may contain various degrees of packet loss. The experiments progress from extremely impressive numbers with orders of magnitude improvement when comparing Tornado codes with primitive Reed-Solomon codes to a more subdued improvement when comparing Tornado codes to interleaved codes to a rather lacklustre set of multicast experiments with only 2 clients at the end. The paper seems to loose its momentum at the end with the final experiments not really adding much value with so few clients involved and 4-layer multicast not showing that much value. Perhaps this would have looked better if a more intelligent packet scheduling system was used across the layers which took into account potential layer switching due to congestion.

The section which compared Tornado codes seemed to intentionally hide important information. For example why did they compare worst case decoding inefficiency rather than average decoding inefficiency, could it be because the numbers would have looked less impressive. The label on the y-axis doesn't even mention worst cast decoding inefficiency which may be intentionally deceptive. The plot of interleaved with k=50 in Figure 4 looks fairly comparable to Tornado coding but the authors don't want to mention exactly what the decoding time was for k=50. In the first paragraph in 6.2, they just say that it was reasonable in practice. This also seems like an information hiding strategy to avoid weakening their argument.

Even with its weaknesses, the paper was clearly good enough to start a whole field of research in fountain codes with publications starting in 1998 and still continuing on the topic today.