# Bulk Data Distribution Paper Review

“A Digital Fountain Approach to Reliable Distribution of Bulk Data” was published by Ashutosh Rege, Michael Mitzenmacher, Michael Lubyt, and John W. Byers at the Association for Computing Machinery's Special Interest Group on Data Communication in Vancouver, British Columbia, Canada, September 1998. The paper is divided into two major parts: An overview of an ideal multicast / broadcast protocol for large data distribution and then a close approximation to the ideal is implemented and evaluated.

The paper starts with a normal introduction of multicast, starting with the problems that unicast presents and the current solutions being implemented. It introduces the idea of an ideal multicast protocol which the paper calls a “Digital Fountain”, which allows the source data can be reconstructed by any subset of the encoded packets equal to the length of the source data.

Next, requirements for a protocol for distributing data over multicast are laid out: Reliable, Efficient, On Demand, and Tolerant. Another requirement is that there should be little or no responses from the clients to the multicast server. This is to prevent the flooding of the network and more specifically the server.

A Digital Fountain is a protocol where the server wishes to distribute K packets of data to clients, where packets that are sent out are called encoding packets. Once a client has any K number of encoding packets, it is able to reconstruct the original data.

The bulk of the paper is devoted to the introduction of Tornado code and its benefits versus current implementations of Reed-Solomon. Reed-Solomon guarantees that after receiving K distinct packets that the source material can be reconstructed. Tornado code is introduced offering the same type of guarantees that Reed-Solomon Error Correction offers. The major advantage of Tornado codes over standard codes is that they trade off a small increase in decoding inefficiency for a substantial decrease in encoding and decoding times. Table 1 compares the properties of Tornado code vs Reed-Solomon.

The paper then shows a fast implementation of the Tornado code compares with Reed-Solomon. An implementation of Tornado, Tornado Z, is compared against an implementation of Reed-Solomon, Cauchy, with Table 2 and Table 3 showing the results. These charts help the reader clearly see that Tornado Z outperforms Cauchy in both encoding and decoding. Table 4 shows that as file size increases, the speedup of Tornado Z increases with respect to interleaved codes.

Next, a simulation is setup to evaluate the real world performance of the Tornado code. Congestion control is implemented by layering multicast, where increasing layer number relates to increasing transmission rate. Table 5 shows the packet transmission scheme for four increasing transmission rate multicast layers with Figure 7 showing the results of the experiment. Figure 7 shows that using multiple layers only hinders the ability of the Tornado code to be efficient.

The experimentation used a file size of just over two megabytes. Including other file sizes would have increased the authors ability to show Tornado's ability to efficiently and effectively transmit over multicast.

The paper states that with such a high loss rate, a larger stretch factor would have been beneficial to the reception inefficiency. Running the experiment a second time choosing a longer stretch factor would prove the papers statement with not much effort.

The conclusion of the paper explains other potential uses for the Digital Fountain style Tornado code, citing dispersity routing of data from endpoint to endpoint in a packet-routing network and mirrored data.

The paper follows well-formed structured format, starting with the ideals necessary in a protocol, introducing an implementation of those ideals, and wrapping up with a real world experiment. The papers use of Tables and Figures throughout increase its readability and help the audience visualize the benefits.