Accessing Multiple Mirror Sites in Parallel Using Tornado Codes to Speed Up Downloads

This paper was published in the IEEE Infocom of 1999. The main objective, as the title suggested, is to speed up client download speed by accessing multiple mirror sites in parallel. The idea is to allow multiple senders in possession of the same file to simultaneously transmit data to the client using traditional end-to-end scheme, the data transmitted is the craftily encoded and sliced version of the file. Forward Error Correction code is used to achieve performance and avoid client feedback. The paper contains an overview on Tornado codes, the implementation, and the simulation results.

The introduction of the paper provided solid information in regard to the problem it is trying to solve and the potential usage of its implementation. It started by mentioning the targeted audiences of this paper as users whose performance bottleneck isn't in the last mile. However, the network infrastructure back in 1999 as well as today suggested the bottleneck for most users remained in the last mile, and a mechanism that requires massive change on the servers but would benefit only a limited amount of users is simply unpractical. In addition, the author suggested that the technique would be beneficial to users using wireless network because of its high tolerance against packet loss and to satellite networks because it requires little feedback, but there are numerous alternatives out there targeting specifically at these issues. Without further clarification and comparison, I find it confusing to what are the audiences that the paper is targeting at and what are some of its possible applications.

Another confusion arises in the last paragraph of the introduction when the authors claimed that their technique could be easily implemented on existing network infrastructure using TCP. Such claims seems to suggest that the simulation done in this paper transported packets over UDP rather than TCP, but the idea of establishing and terminating connection can be seen throughout such as under section V: "Tornado code solutions" It is unclearly to me whether the simulation used TCP or UDP. In addition, while the author claimed the technique could be easily implemented using TCP, the use of TCP would eliminate one of its biggest advantages as tremendous amount of feedback would be generated from client to the senders. As a result, it seems such technique would offer greater advantages in UDP environments.

Following the introduction, the paper discussed works that are related to their proposed technique. The authors claimed that the idea of parallel access to multiple mirror sites has received little attention, but the idea of using Forward Error Correction codes in multicast distribution has. While the paper briefly mentioned the reason for choosing the Tornado code over Reed-Solomon code here in the related work section, it failed to mention other FEC codes such as parity or Luby Transform code. The advantage of Tornado code over such FEC codes is also unclear.

The paper was very well organized in which the authors listed their assumptions in a separate section following related works. The first assumption assumed that each mirror site can efficiently produce a pool of encoded packets for the data either in advance or on demand. While such assumption might be reasonable back in 1999 when the most popular online activities are emails and simple web browsing, the growth of Internet over the decade has made it unreasonable. Under the assumption, the sender would be required to encode couple hundred megabytes large of files in advance or on demand. This would require massive amount of processing power. Another unreasonable assumption was made which assumed the set of paths from client to mirror sites are bottleneck-disjoint. With the introduction section seems to indicate that the simulation is done over UDP rather than TCP, massive amounts of data is almost guarantee to affect each connection if not cause congestion especially on the path directly connected to the receiver without any type of congestion control.

After clearly stated the necessary background information and assumptions, the paper went on to reveal its proposed solution, namely the idea of digital fountain with Tornado codes. The overview section briefly compares Tornado codes and Reed-Solomon codes which was summarized in table 1. The paper mentioned one of Tornado code's benefit as using only exclusive-or operations but failed to discuss why exclusive-or operations are better than field operations which are used by Reed-Solomon codes.

The mechanism of digital fountain with Tornado code was tested through simulation, but with the specification of the simulation environment and variables largely undisclosed, the simulation result and the actual performance of this mechanism remains questionable. As part of the simulation results, figure 1 demonstrates the ideal performance of digital fountain using Tornado code. The graph reveals a good tradeoff between resources and speedup. In fact, download time halved from a stretch factor of 3 to 4 and the speedup could be as high as 3.5 with 4 senders and a stretch factor of 6. However, figure 2 demonstrated a scenario that is more realistic where data arrive at different rates rather than uniform rates from senders. The performance substantially decreases in comparison to the ideal situation.

Digital fountain with Tornado codes could be vaguely summarized as a mechanism that trades resources for performances. While the graphs and simulation results did show a substantial gain in speed, it came with the price of large overhead and cost in resources. In fact, as figure 2 reveals, stretch factor of 3 seems to be the most optimal in terms of performance gain and the cost of resources. However, it would require encoding redundant data two times the size of original data in combine with two senders to gain a speedup of around 1.75. In addition, the paper assumed senders won't fall into cycles when a larger stretch factor is chosen, but this indicates a substantial amount of redundant data is needed so either way it won't work well with larger files. Overall, it is a very costly solution and while such mechanism probably would

have great usage back in 1999 when common files are small, it is not very realistic in the environment today.

Digital fountain with Tornado codes is in many ways, similar to that of the peer-to-peer technology. Both techniques sliced the file into multiple chucks of data and the client receives data from multiple senders. The only difference seems to be the underlying network: TCP and UDP, and the fact that peer-to-peer eliminates data redundancy but depends on the applications for reliability.

Overall, the paper was able to come up with a novel solution to an existing solution by combing technique designed for multicast with parallel mirror sites access. The information was well organized and written in ways that are very clear and easy to understand. However, the paper failed to clarify the intended audiences and possible applications. With the advance in technology over the decade, the mechanism proposed in this paper seems to be outdated but remains to be a good source of information nonetheless.