CS 559: Algorithmic Aspects of Computer Networks **Fall 2007**

Lecture 4 — September 17

Lecturer: John Byers *BOSTON UNIVERSITY* *Scribe: Sowmya Manjanatha*

In today's lecture, we discussed and characterized the implications of building *Digital Fountain* with *Reed Solomon (RS)*, *Tornado* and *LT* codes. We also compared the use of interleaving to increase data reconstruction performance, and concluded the topic of digital fountain with its application to layered congestion control. We also briefly touched upon the concept of Network coding.

4.1 Digital Fountain Characterization

A *Digital Fountain* is an approach used to provide reliability in multicast networks without the use of receiver initiated packet retransmissions. Prior work to *Digital Fountain* has shown that use of receiver initiated retransmissions to recover lost data is unscalable in multicast networks and the impact is significant in lossy networks. Instead, like many other techniques that transmit redundant codewords, *Digital Fountain* uses the concept of encoding data using Tornado and LT codes such that reconstruction of data at the receiver is done with minimal overhead. The intuition behind *Digital Fountain* is that there is a tradeoff between decoding time and decoding efficiency. By accepting an almost negligible penalty of decoding inefficiency, significant improvements are achieved for encoding and decoding times [1]. In the following section, we discuss the limitations of RS codes to make the advantages of Tornado and LT codes clear.

4.1.1 Limitations of building Digital Fountain with RS codes

In analyzing RS codes, two properties namely *Running Time* i.e. encoding and decoding times and *size of encoding* is considered. RS codes display the following features:

- encoding time is quadratic because the operations needed to generate n encoding for a stream of k packets is $\frac{k(n-k)A}{2}$. Decoding time is comparable to encoding time (A is symbol size).
- There is no decoding overhead i.e. decoding efficiency is 1 provided the matrix used for reconstruction can be inverted.
- Decoding/Encoding times are impractical for a file size of $1MB$.

[1] The details of implementing digital fountain with RS, Tornado and LT codes was discussed in the previous lecture. This description uses concepts from previous lecture as a basis for explanation here

4.1.2 Comparison of RS Vs Tornado and LT codes

	Reed Solomon	**Tornado**	**LT**
Encoding/ Decoding Times	quadratic: too high for file sizes such as 1MB	nlog(1/ϵ)P	roughly $O(klogk)P$
Decoding Efficiency	1	1 + ϵ	1 + $O(ln^2(k/\delta)/\sqrt{k})$

Table 4.1. Performance of RS Vs Tornado Vs LT

Table 4.1 provides a quick comparison of how Reed Solomon, Tornado and LT codes perform. The exact bounds for all the running times are provided in the Digital Fountain paper [1]. The authors of the Digital Fountain paper found a small variation in the decoding efficiency depending on the set of encoding packets arrived. The CCDF plot shown in figure 4.1 illustrates how the decoding efficiency performs over 10,000 trials. They noted that the decoding inefficiency is roughly 1.06 on average which is fairly low.

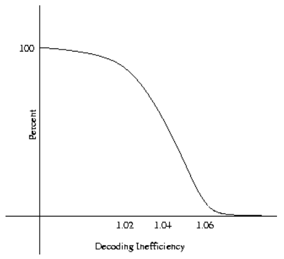

Figure 4.1. Decoding Efficiency of Tornado over several trials

4.1.3 Interleaving

In such multicast data dissemination scenarios, interleaving becomes necessary and has two major advantages, (1) If a receiver missed a block either because of a lost packet or because it joined the multicast group in the middle of a transmission, then it must wait for the entire length of the file to be disseminated before the source starts to recycle through the carousel again. The problem is worse if the file sizes are larger. (2) If a malicious source starts to send a burst of data which the receiver cannot handle, then again the receiver must wait its long turn.

The approach to remedy this situation is interleaving and is done as follows: A file is divided into blocks of size B as shown in Figure 4.1.3 and then stretched to a factor of 2 in the example shown and the encoding is formed by interleaving packets from different blocks. That is basically the packets are shuffled so that the probability that a receiver who missed a packet will receive expected packets sooner increases. The encoding time in this case is $O(B^2)$*k/B = $O(kB)$ where k is the size of the file.

The problem with interleaving is that decoding inefficiency then becomes a random variable which depends on several factors such as loss rate, block size etc. In order to understand

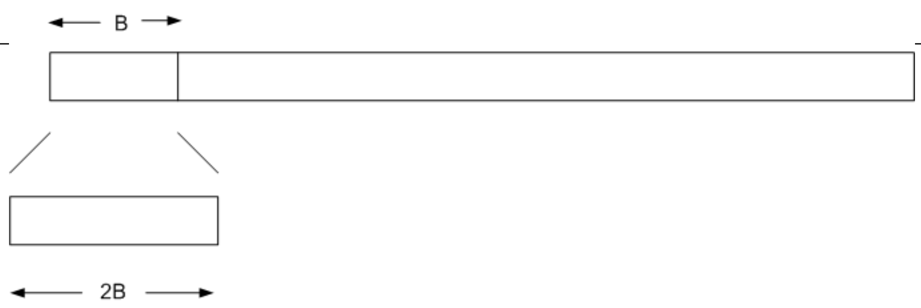

Figure 4.2. Splitting file into blocks.

the tradeoff between decoding inefficiency and coding time, the authors of Digital Fountain paper conducted experiments by varying different parameters and for two sets of experiments i.e file size in one, and, receiver count in the other. The random nature of last blocks to fill is depicted in Figure 4.3

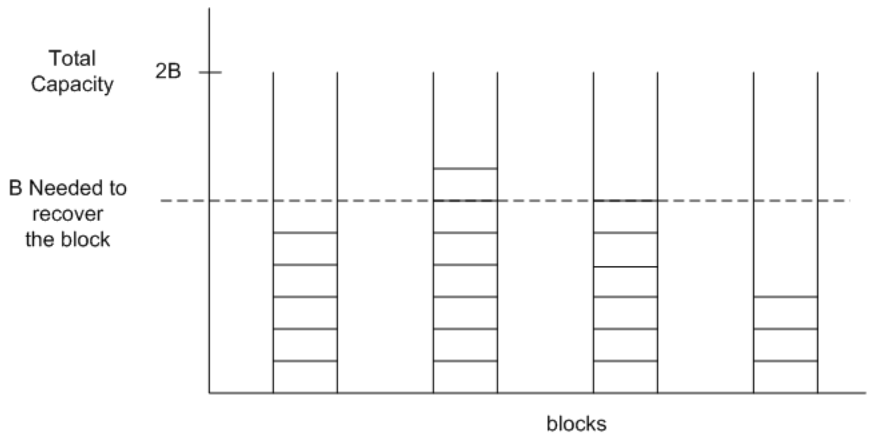

Figure 4.3. Waiting for blocks to fill..

From the experiments, RS and LT performed almost the same since interleaving increases the decoding time. The problem gets worse with increase in file sizes.

4.1.4 Layered Congestion Control

The key idea here is that the source transmits data across multiple multicast groups. Receivers subscribe to a specific layer based on the their reception rates. A receiver's subscription level depends on the bottlenecks in the path since a flow may use different congestion paths as shown in Figure 4.4 (left). The total number of layers is

$$Total\ number\ of\ layers \ = \ log\frac{highest\ bandwidth}{lowest\ bandwidth}$$

A receiver can attempt to subscribe to a higher layer only after receiving *synchronization points* which are specially marked packets in stream. In the implementation used in the paper, the server generates periodic bursts during which packets are sent at twice the normal rate of the layer. If a receiver experiences packet losses, then it will subscribe back to the previous layer. This approach enables scalability and especially suitable for video coding because of the natural dynamics of video coding application.

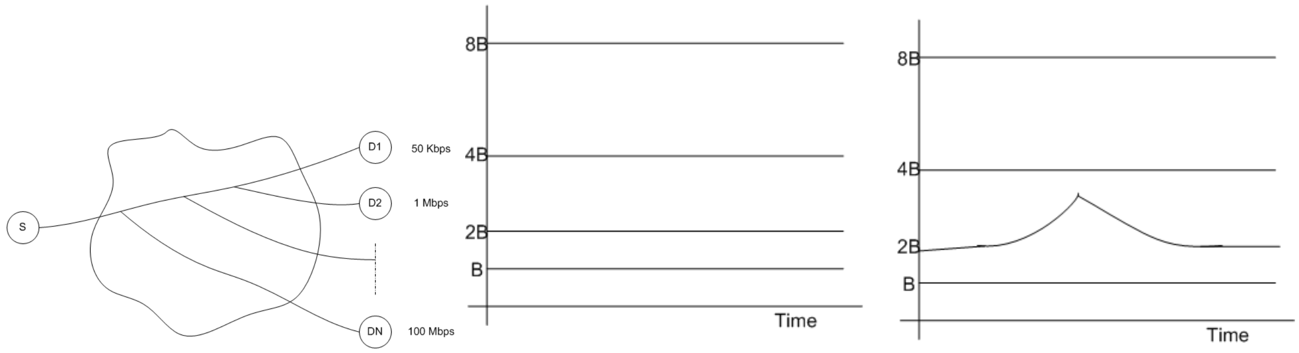

Figure 4.4. Different Receiver Paths (left), Geometrically increasing Layers (middle), Bursty increase approach (right)

4.2 Network Information Flow

Network Coding refers to applying coding at the routers/switches i.e. as opposed to coding at the source. Ahlswede et al [2] introduced the concept of *Network Coding* in their seminal work published in 2000. In systolic networks, a *Network Information Flow* can be defined as a weighted subgraph of the original network graph as shown in Figure 4.2. The *Network Information Flow* displays the following set of properties:

- Weights respect the capacities of the original graph.
- Weights also respect the conservation of flow i.e. inflow(v) = outflow(v) except when v is the designated source or destination, v being the vertex of the subgraph.
- the value of flow val(flow) = outflow(source) = inflow(destination).

Contrary to the prevailing philosophy, Ahlswede et al displayed a remarkable idea where, in network scenarios with single source and multiple destinations, a flow could be xor'ed into one at vantage points to cut the number of transmissions as seen from the Figure 4.2(right). More on this topic will be discussed in the next lecture.

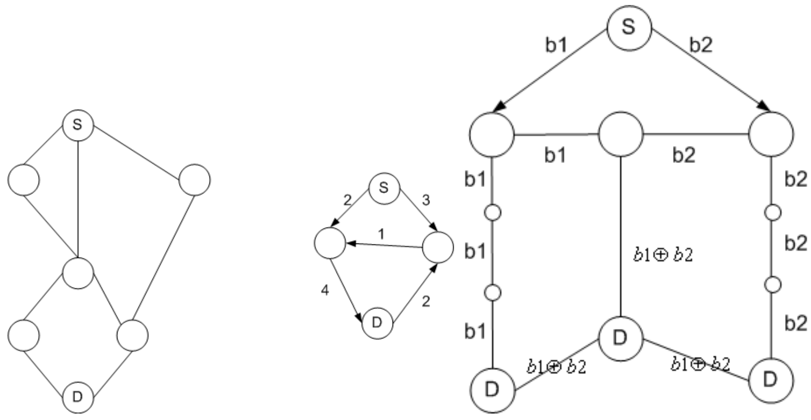

Figure 4.5. Flow represented by a weighted subgraph(left) and Network graph where Network Coding could be beneficial (butterfly graph on right)

.

Bibliography

[1] John W. Byers, Michael Luby and Michael Mitzenmacher, "A Digital Fountain Approach to Asynchronous Reliable Multicast", IEEE Journal on Selected Areas in Communication, 2002.

[2] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, Raymond W. Yeung, "Network Information Flow", IEEE Transactions on Information Flow, 2000.