Lecture 8 — March 18, 2002

Lecturer: John Byers *BOSTON UNIVERSITY* *Scribe: S.Y.Zeng, X.Yuan*

This lecture is talking about a digital fountain approach to asynchronous reliable multicast, and accessing multiple mirror sites in parallel by using Tornado codes. Combine Tornado codes with layered multicast, to enable "digital fountain" from which you can start drinking at any time and adapt your drinking rate to your bandwidth, then stop after receiving any distinct l out of $(k + l)$ packets to reconstruct the original data.

8.1 Requirements of An Ideal Protocol

A protocol for distributing the software using multicast should be:

- **Minimize:** keep network traffic to a minimum.
- **Scalable:** Server load remains constant whether there are one or a million receivers.
- **Reliable:** An exact copy of the original file is reconstructed by each other.
- **Reception efficient:** The total number of packets each receiver needs to reconstruct the file is minimal.
- **Time efficient:** The amount of processing required to generate packets at the server and to reconstruct the file from received packets at the receiver is minimal.
- **Time independent:** Different receivers may start the download at disparate times and may sporadically download.
- **Server independent:** No coordination between servers.
- **Tolerant:** The protocol should tolerate a heterogeneous of receivers, especially a vatiety of packet loss rates and data rates.

8.2 Digital Fountain

- DF is the stream of encoding packets produced by one of the servers in idealized solution which achieves all the requirements mentioned above.

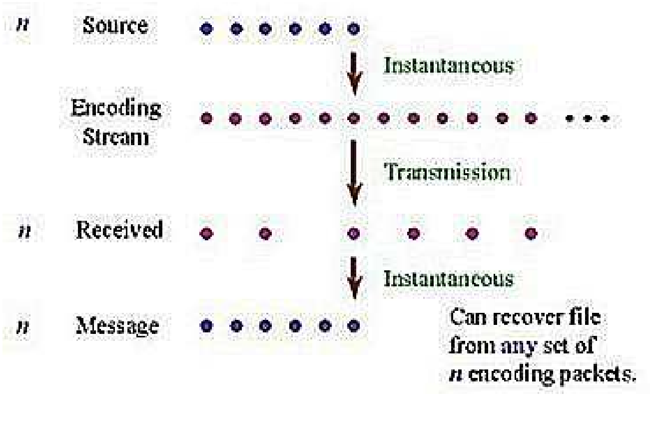

Figure 8.1. Digital Fountain

- **<u>Features:</u>**
  - Users can initiate the download at their discretion

- Users can continue download seamlessly after temporary interruption
- Tolerates moderate packet loss
- Low server load - simple protocol
- Does scale well
- Low network load

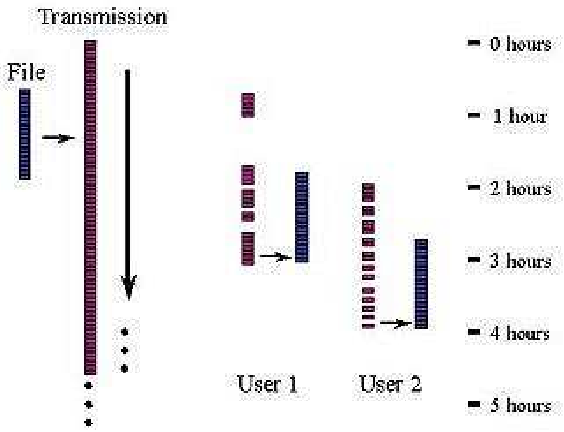

Figure 8.2. Digital Fountain Solution

8.3 RS vs. Tornado vs. LT

8.3.1 Reed-Solomon Codes

Features:

- $n = k + l$ where
  n: total number of encoding packets
  k: number of source data packets
  l: number of redundant packets, l redundant packets are a linear combination of the source data packets.
- *Decoding inefficiency:*
  - is 1.
  - This is a dense system which has a high decoding time.
- RS coding guarantees that reception of exactly k distinct encoding packets is enough to reconstruct the source data.
- *Limitations:*
  - Reception of redundant data
  - Inefficiency of encoding/decoding

8.3.2 Tornado Codes

Features:

- Faster encoding/decoding
- Need a little more than k packets to reconstruct source data
- Encoding/decoding time is dominated by number of *xor* in the system of equations
- Sparse random equations are used to build codes

- Equations are of the form y3 = x1 + x4 + x7 is bitwise xor
- Redundant packets may be created from other redundant packets
- The entire coding is performed ahead of time and then packets are sent out in random order from the source.
- *Decoding inefficiency:(1 + ϵ)k* packets are needed to reconstruct the data. ϵ is not fixed. It depends on loss patterns and random choices used to construct the codes.
- Trade-off a small degradation in decoding efficiency for a substantial improvement in encoding and decoding time.

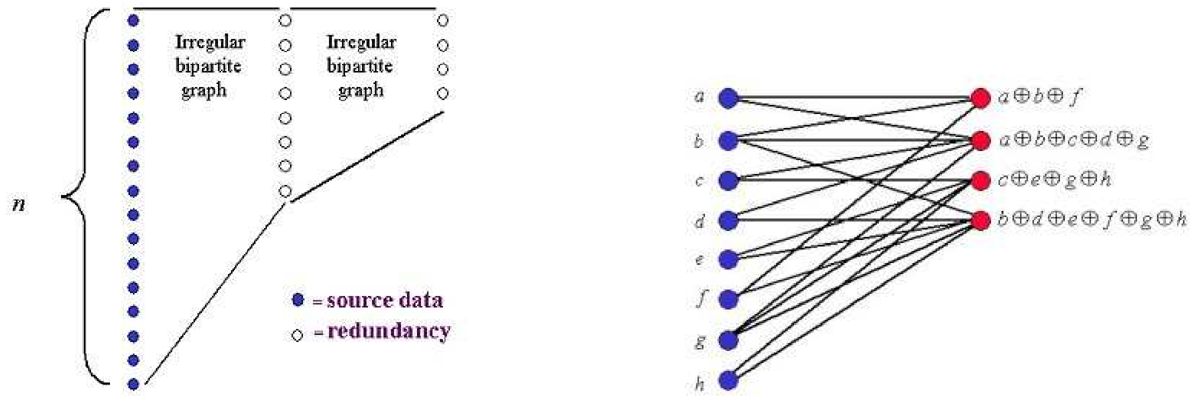

Figure 8.3. *Tornado Encoding Structure* and *Encoding/Decoding Process*

8.3.3 Luby's Transform

Features:

- No predetermined value of n, as the equations placed into each encoding packet are generated independently of all other encoding packets.
- Stretch factor is inherently unlimited.
- The average number of xors to generate each encoding packet and to reconstruct each source packet is upper bounded by the average number of variables in each equation.
- Each encoding packet is produced on-the-fly from an extremely large set of possibilities at the same average processing cost as every other encoding packet.

8.3.4 Comparisons

See Table 1 and Table 2.

8.4 Application 1 – Asychronous Reliable Multicast

8.4.1 Digital Fountain Prototype

- Build on top of IP multicast
- Tolerating heterogeneity
  - Layered multicast with multiple multicast groups
  - TCP-friendly, receiver-driven congestion control

	Tornado	Reed-Solomon
Decoding inefficiency	$1+\epsilon$ required	1
Encoding times	$n\ln(1/\epsilon)P$	$k(n-k)AP/2$
Decoding times	$n\ln(1/\epsilon)P$	$k(n-k)AP/2$

Table 1: Properties of Tornado vs. Reed-Solomon codes when a fixed stretch factor is employed

	LT	Reed-Solomon
Decoding inefficiency	Asymptotically 1	1
Per packet encoding times	$\ln(k)P$	$kAP/2$
Decoding times	$k\ln(k)P$	$k(n-k)AP/2$

Table 2: Properties of LT vs. Reed-Solomon on-the-fly codes.

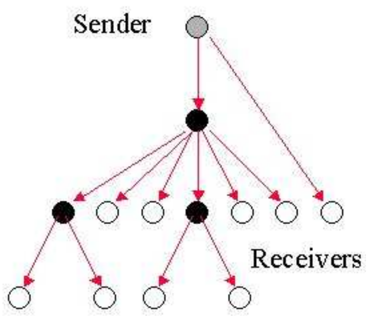

Figure 8.4. Multicast: save bandwidth

8.4.2 Implementation

A simulation for distributing bulk data to a large number of heterogeneous receivers who may access the data asynchronously.

Integrating with Layered Congestion Control

- The main idea of layered multicast is to enable the source to transmit data across multiple multicast groups, thereby allowing the receivers to subscribe to an appropriate subset of these layers.
- The server transmits data over multiple layers, where the layers are ordered by increasing transmission rate. e.g. transmission rates *Bi = 2(i-1)* is the rate of the *ith* layer. (note: there are other layered schemes, such as cumulatived layers is that a receiver subscribes to layer *i* also subscribes to all layers beneath it.)
- Using SP's - synchronization points to achieve congestion control. A receiver can attempt to join a higher layer only immediately after an SP. Lower bandwidth receivers are given more frequent opportunities to add higher layers.
- The server generates periodic bursts which create similar congestion conditions with an explicit join. If no packet losses during and after the burst, the receiver add a layer at the next SP. Otherwise, the receiver doesn't add a layer or drop to a lower level.

Scheduling Transmissions Across Multiple Multicast Groups

With constant stretch factor c, it is important to schedule transmissions across the multiple layers to minimize the duplicate packets on the receivers. With LT codes, the encoding process is on-the-fly and memoryless.

Reconstruction at the Receiver

Two methods:

- Incremental approach: decoding after each packet arrives.
- Patient approach: decoding waits until a fixed number of packets arrive.

8.5 Application 2 – Paraloading

8.5.1 Paradigms:

See Figure 8.5.

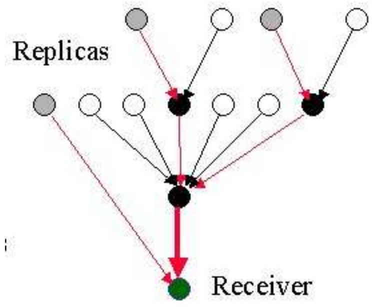

Figure 8.5. Parallel Download: reduces latency

Features:

- One receiver obtains file from multiple senders to improve downloading speed.
- Many to many distribution: simutaneous multicast/multigather.
  - Sources multicast or broadcast
  - Receivers download from multiple sources in parallel

8.5.2 The Techniques of Parllel Accesssing to Multiple Mirror Sites

- Using Tornado codes.
- Used in conjunction with TCP-friendly regulatory mechanisms.
- Requirements: Possible to create bottle-disjoint paths, means receivers should not be the bottlnect.
- Primary benefit: the speedup of the download.
- Main cost: bandwidth; A moderate number of additional packets may be injected into the network by the mirror sits.

8.5.3 Tornado code solutions

Given a set of k packets and produce a set of l redundant packets for a total of $n = ck = k+l$ encoding packets all of fixed length p. c is the stretch factor,$c = n/k$.

Feedback free solution

- Senders encode message the same way.
- Senders cycle through permutation of encoding
- When receiver obtain any 1.05k distinct packets, it can decode to obtain the message.
- **Pros:**
  - Simple
  - Loss protection
  - Good download speedups
  - No feedback, coordination
  - Solves many-to-many
- **Cons:**
  - Extra bandwidth for Tornado codes (5.5%)
  - Extra bandwidth from packet duplicates. (Depending on c, number of senders, variation in rates.)

Rare Feedback - Solutions with minimal control information

- Limit the distinctness inefficiency.
- Senders use same permutation of encoding.
- Receivers tell each of s senders to send $1/s$ of the encoding.
  - If $c \geq s$, each sender has 1 file worth of data
  - In rare cases, renegotiate, or have senders send the rest in random order.

Bibliography

[1] J. Byers, M. Luby, M. Mitzenmacher, "*A Digital Fountain Approach to Asynchronous Reliable Multicast,*" To appear in IEEE Journal on Selected Areas in Communications, 2002. (Journal version of the ACM SIGCOMM '98 paper.)

[2] J. Byers, M. Luby, M. Mitzenmacher, "*Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads,*" in Proceedings of IEEE INFOCOM '99.