A Digital Fountain Approach to Reliable Distribution of Bulk Data

John Byers, ***ICSI***

Michael Luby, ***ICSI***

Michael Mitzenmacher, ***Compaq SRC***

Ashu Rege, ***ICSI***

Application: Software Distribution

- New release of widely used software.
- Hundreds of thousands of clients or more.
- Bulk data: tens or hundreds of MB
- Heterogeneous clients:
  - Modem users: hours
  - Well-connected users: minutes

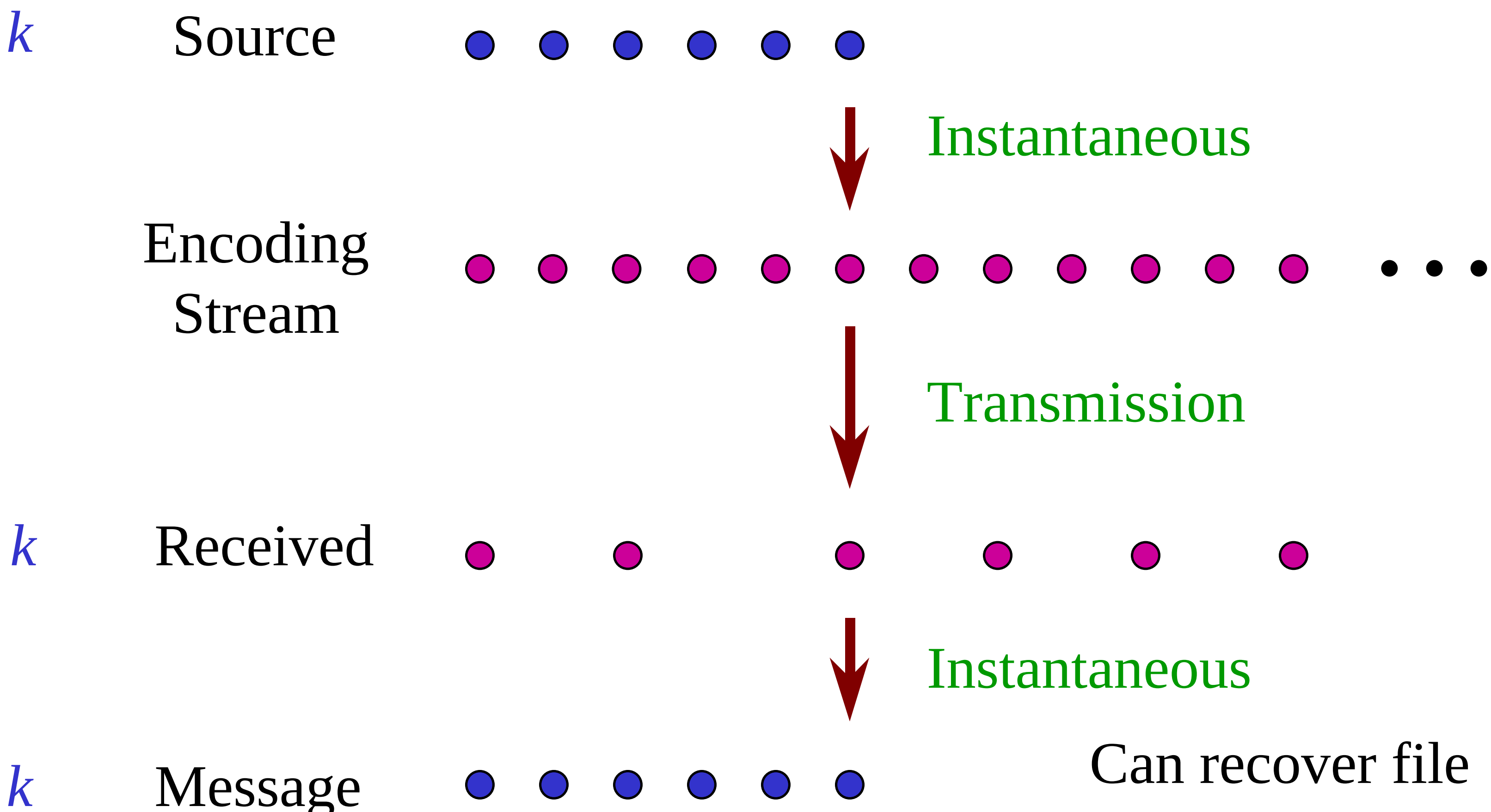
Primary Objectives

- Scale to vast numbers of clients
  - No ARQs or NACKs
  - Minimize use of network bandwidth
- Minimize overhead at receivers:
  - Computation time
  - Useless packets
- Compatibility
  - Networks: Internet, satellite, wireless
  - Scheduling policies, i.e. congestion control

Impediments

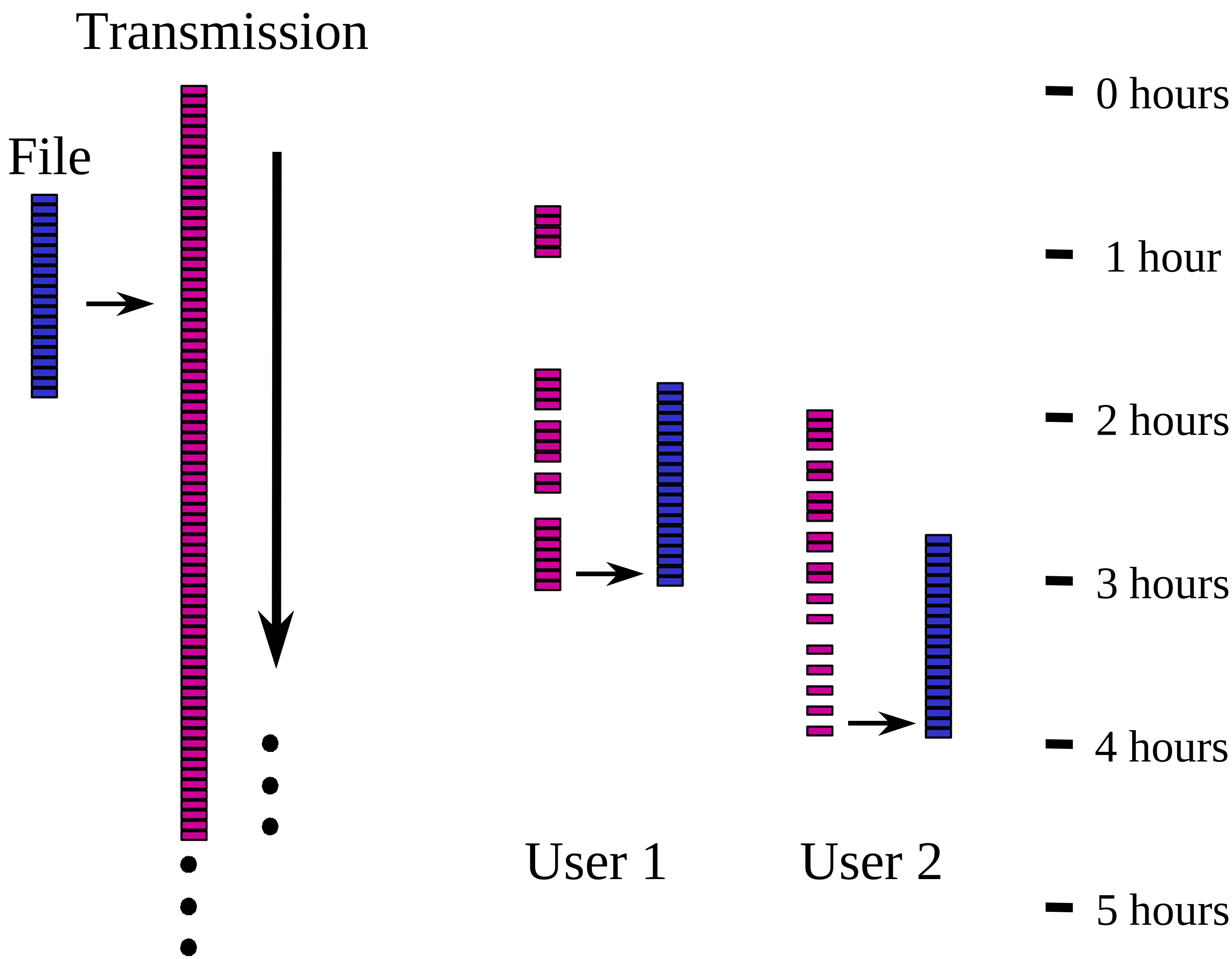
- Packet loss
  - wired networks: congestion
  - satellite networks, mobile receivers
- Receiver heterogeneity
  - packet loss rates
  - end-to-end throughput
- Receiver access patterns
  - asynchronous arrivals and departures
  - overlapping access intervals

Digital Fountain



Can recover file
from **any** set of
k encoding packets.

Digital Fountain Solution



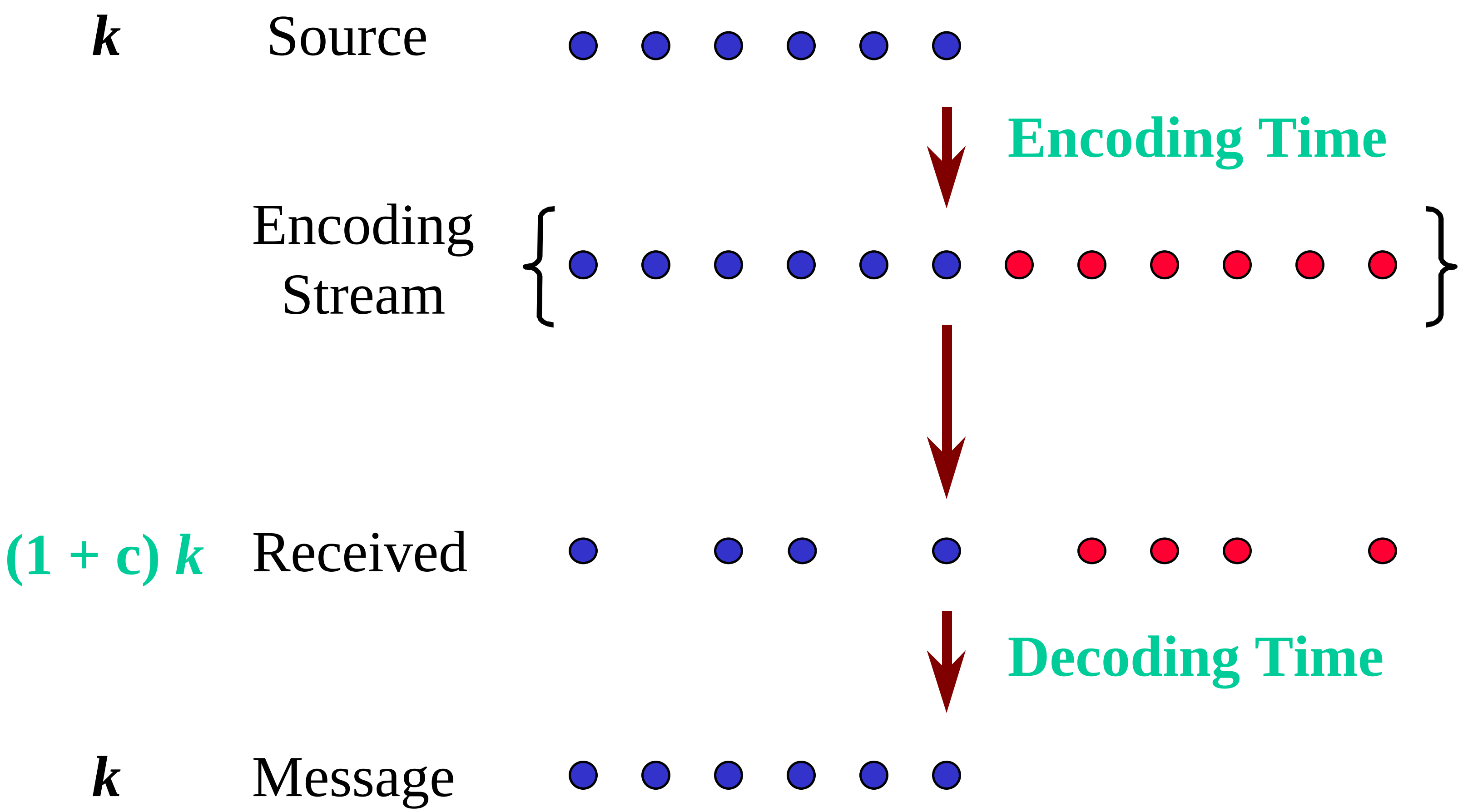
Is FEC Inherently Bad?

- Faulty Reasoning
  - FEC adds redundancy
  - Redundancy increases congestion and losses
  - More losses necessitate more transmissions
  - **FEC consumes more overall bandwidth**
- But…
  - Each and every packet can be useful to **all** clients
  - Each client consumes minimum bandwidth possible
  - **FEC consumes less overall bandwidth by compressing bandwidth across clients**

DF Solution Features

- 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.

Approximating a Digital Fountain



Approximating a DF: Performance Measures

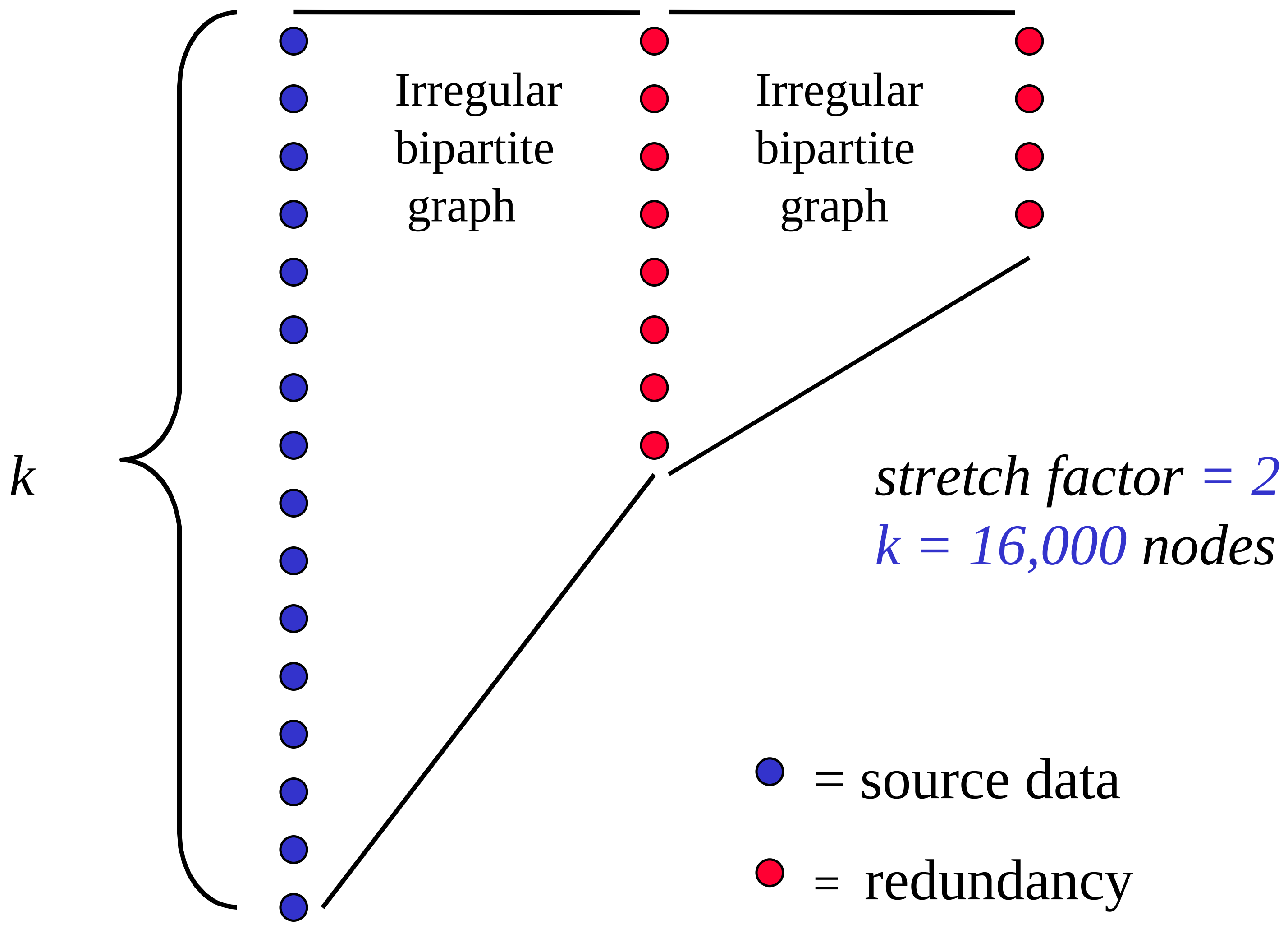
- Time Overhead:
  - Time to decode (or encode) as a function of k.
- Decoding Inefficiency:

$$\frac{\text{packets needed to decode}}{k}$$

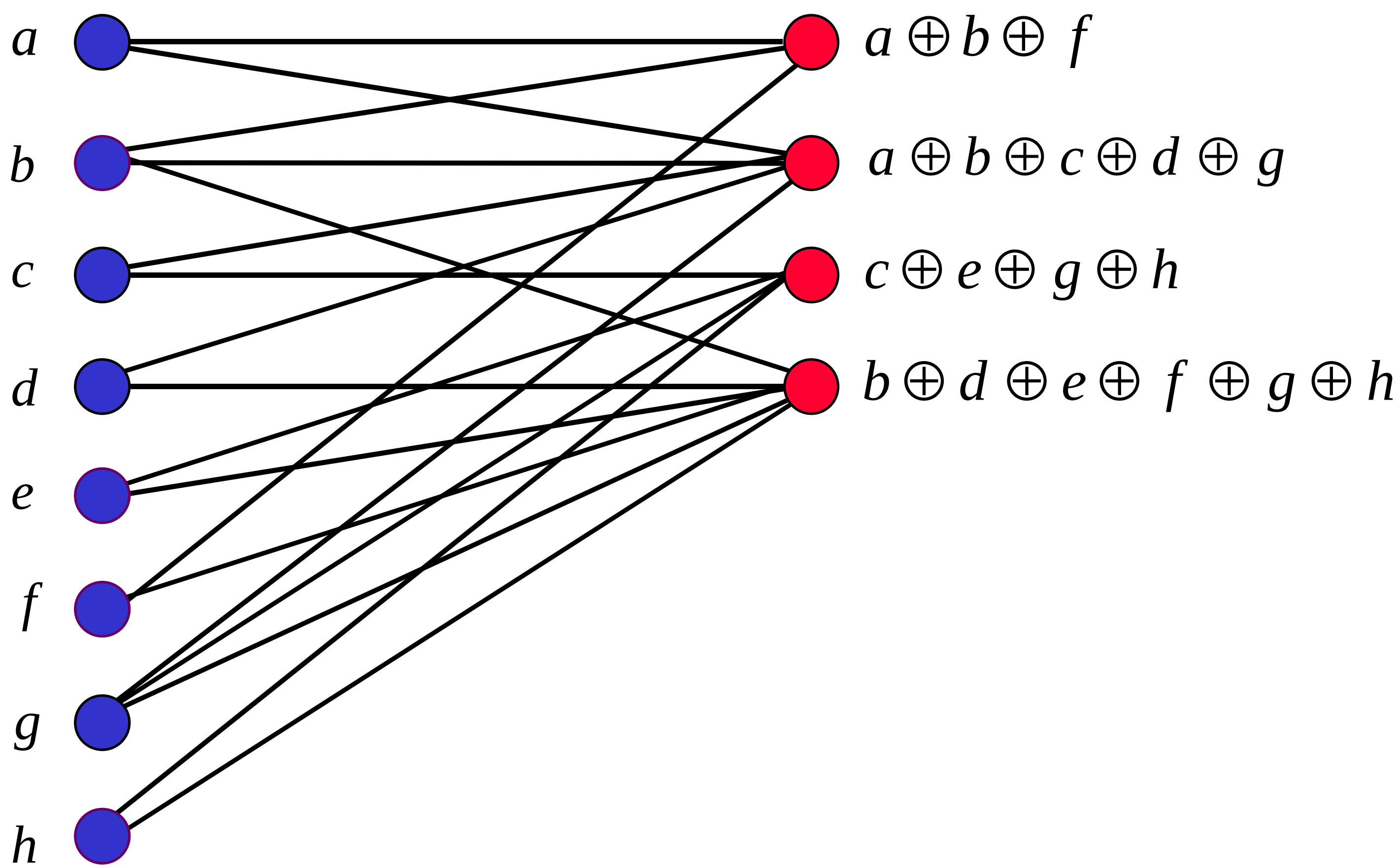
Work on Erasure Codes

- Standard Reed-Solomon Codes
  - Dense systems of linear equations.
  - Poor time overhead (quadratic in k)
  - Optimal decoding inefficiency of 1
- Tornado Codes [LMSSS ‘97]
  - Sparse systems of equations.
  - Fast encoding and decoding (linear in k)
  - Suboptimal decoding inefficiency

Tornado Z: Encoding Structure



Encoding/Decoding Process



Timing Comparison

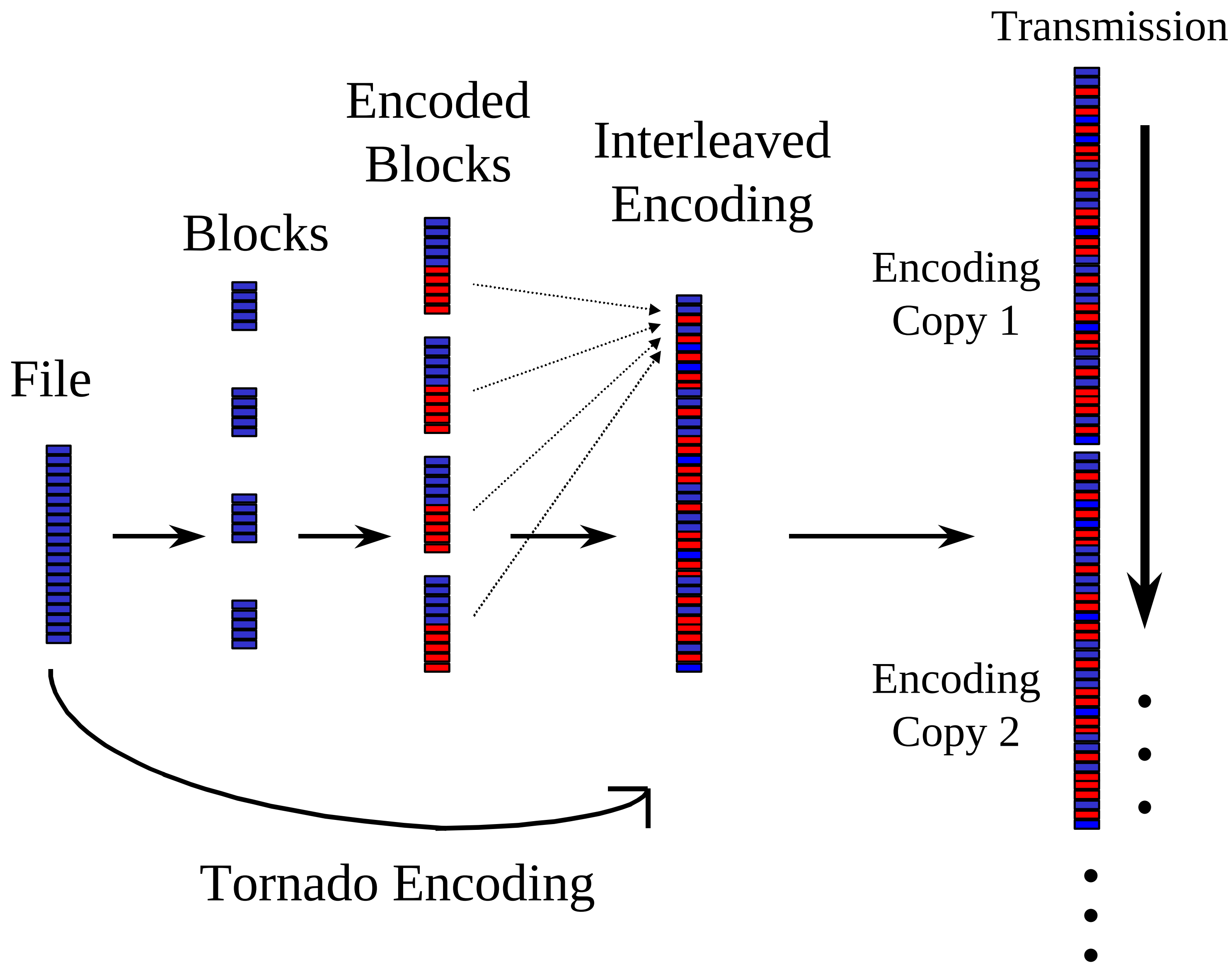
Encoding time, 1K packets		
Size	Reed-Solomon	Tornado Z
250 K	4.6 sec.	0.11 sec.
500 K	19 sec.	0.18 sec.
1 MB	93 sec.	0.29 sec.
2 MB	442 sec.	0.57 sec.
4 MB	30 min.	1.01 sec.
8 MB	2 hrs.	1.99 sec.
16 MB	8 hrs.	3.93 sec.

Decoding time, 1K packets		
Size	Reed-Solomon	Tornado Z
250 K	2.06 sec.	0.18 sec.
500 K	8.4 sec.	0.24 sec.
1 MB	40.5 sec.	0.31 sec.
2 MB	199 sec.	0.44 sec.
4 MB	13 min.	0.74 sec.
8 MB	1 hr.	1.28 sec.
16 MB	4 hrs.	2.27 sec.

Tornado Z: Average inefficiency = 1.055

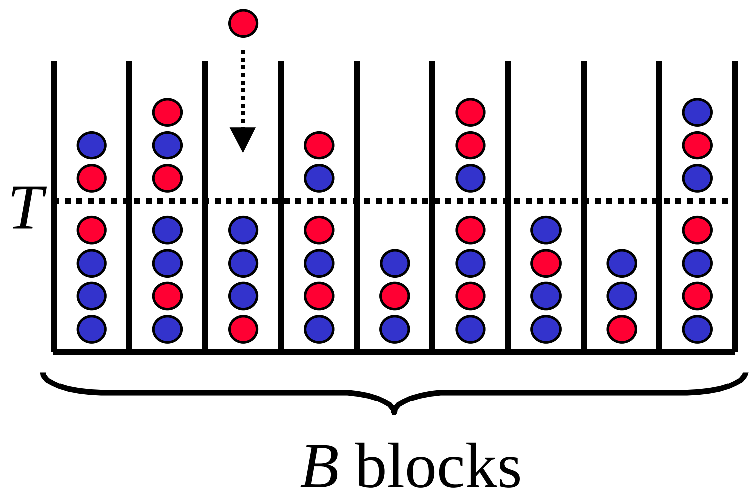
Both codes: Stretch factor = 2

Cyclic Interleaving



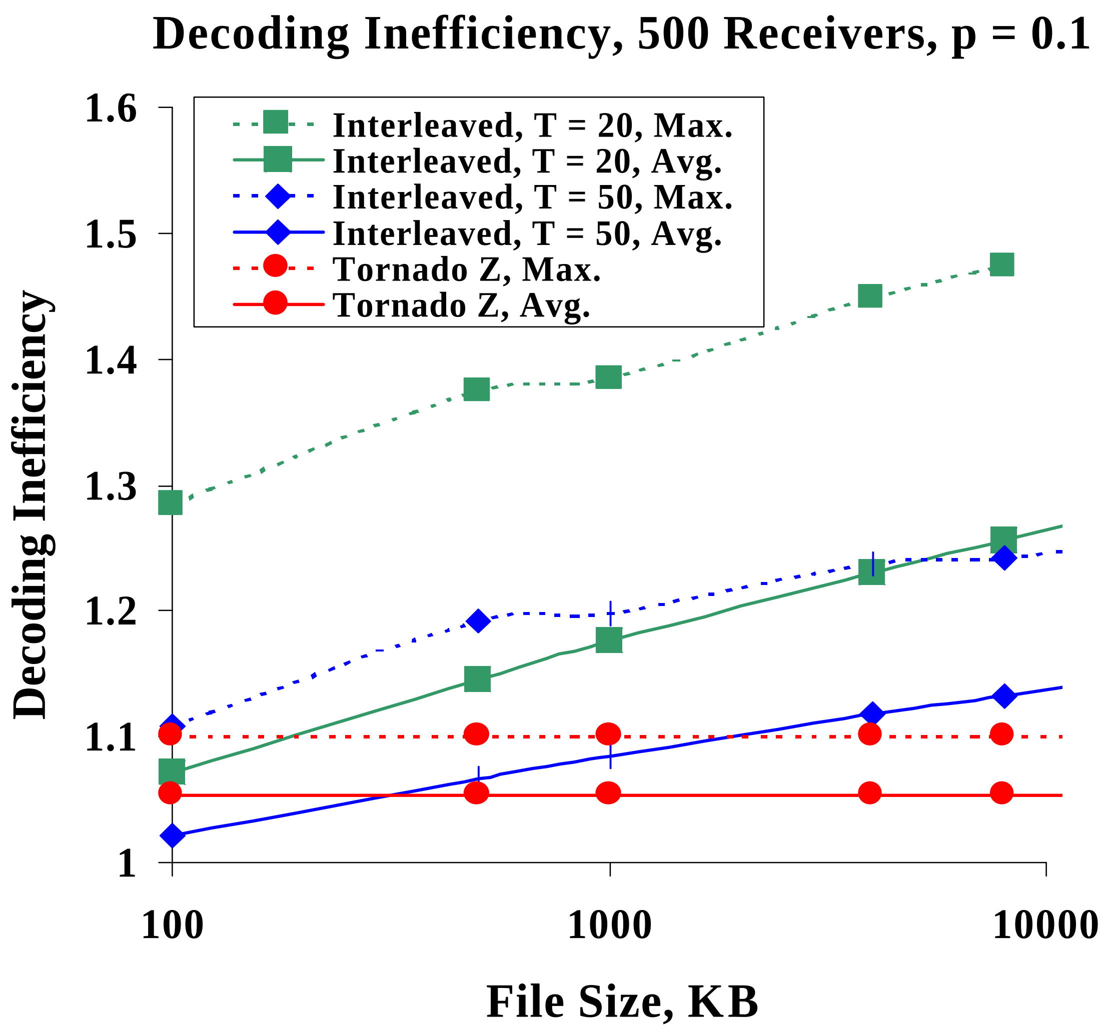
Cyclic Interleaving: Drawbacks

- The Coupon Collector's Problem
  - Waiting for packets from the last blocks:

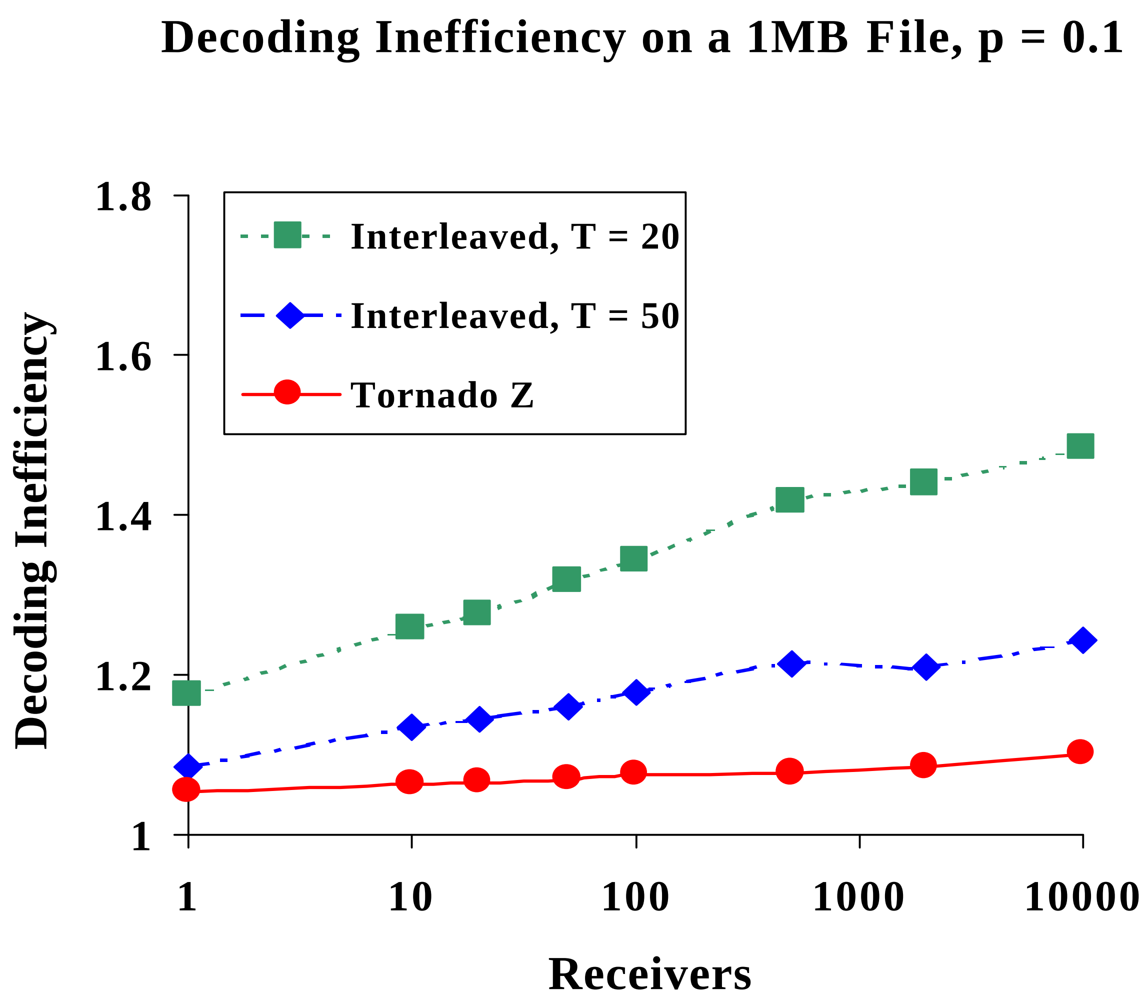


  - More blocks: faster decoding, larger inefficiency

Scalability over File Size



Scalability over Receivers



Digital Fountain Prototype

- Built on top of IP Multicast.
- Tolerating heterogeneity:
  - Layered multicast
  - Congestion control [VRC ‘98]
- Experimental results over MBONE.

Research Directions

- Other applications for digital fountains
  - Dispersity routing
  - Accessing data from multiple mirror sites in parallel
- Improving the codes
- Implementation and deployment
  - Scale to large number of clients
  - Network interactions