Digital Fountains, and Their Application to Informed Content Delivery over Adaptive Overlay Networks

Michael Mitzenmacher

Harvard University

The Talk

- Survey of the area
  - My work, and work of others
  - History, perspective
  - Less on theoretical details, more on big ideas
- Start with digital fountains
  - What they are
  - How they work
  - Simple applications
- Content delivery
  - Digital fountains, and other tools

Data in the TCP/IP World

- Data is an ordered sequence of bytes
  - Generally split into packets
- Typical download transaction:
  - "I need the file: packets 1-100,000."
  - Sender sends packets in order (windows)
  - "Packet 75 is missing, please re-send."
- Clean semantics
  - File is stored this way
  - Reliability is easy
  - Works for point-to-point downloads

Problem Case: Multicast

- One sender, many downloaders
  - Midnight madness problem – new software
  - Video-on-demand (not real time)
- Can download to each individual separately
  - Doesn't scale
- Can "broadcast"
  - All users must start at the same time?
  - Heterogeneous packet loss
  - Heterogeneous download rates

Digital Fountain Paradigm

Stop thinking of data as an ordered stream of bytes.

- Data is like water from a fountain
  - Put out your cup, stop when the cup is full.
  - You don't care which drops of water you get.
  - You don't care what order the drops get to your cup.

What is a Digital Fountain?

- For this talk, a *digital fountain* is an ideal/paradigm for data transmission.
  - Vs. the standard (TCP) paradigm: data is an ordered finite sequence of bytes.
- Instead, with a digital fountain, a k symbol file yields an infinite data stream; once you have received any k symbols from this stream, you can quickly reconstruct the original file.

Digital Fountains for Multicast

- Packets sent from a single source along a tree.
- Everyone grabs what they can.
  - Starting time does not matter – start whenever.
  - Packet loss does not matter – avoids feedback explosion of lost packets.
  - Heterogeneous download rates do not matter – drop packets at routers as needed for proper rate.
- When a user has filled their cup, they leave the multicast session.

Digital Fountains for Parallel Downloads

- Download from multiple sources simultaneously and seamlessly.
  - All sources fill the cup – since each fountain has an "infinite" collection of packets, no duplicates.
  - Relative fountain speeds unimportant; just need to get enough.
  - No coordination among sources necessary.
- Combine multicast and parallel downloading.
  - Wireless networks, multiple stations and antennas.

Digital Fountains for Point-to-Point Data Transmission

- TCP has problems over long-distance connections.
  - Packets must be acknowledged to increase sending window (packets in flight).
  - Long round-trip time leads to slow acks, bounding transmission window.
  - Any loss increases the problem.
- Using digital fountain + TCP-friendly congestion control can greatly speed up connections.
- Separates the “what you send” from “how much” you send.
  - Do not need to buffer for retransmission.

One-to-Many TCP

- Setting: Web server with popular files, may have many open connections serving same file.
  - Problem: has to have a separate buffer, state for each connection to handle retransmissions.
  - Limits number of connections per server.
- Instead, use a digital fountain to generate packets useful for all connections for that file.
- Separates the “what you send” from “how much” you send.
  - Do not need to buffer for retransmission.
- Keeps TCP semantics, congestion control.

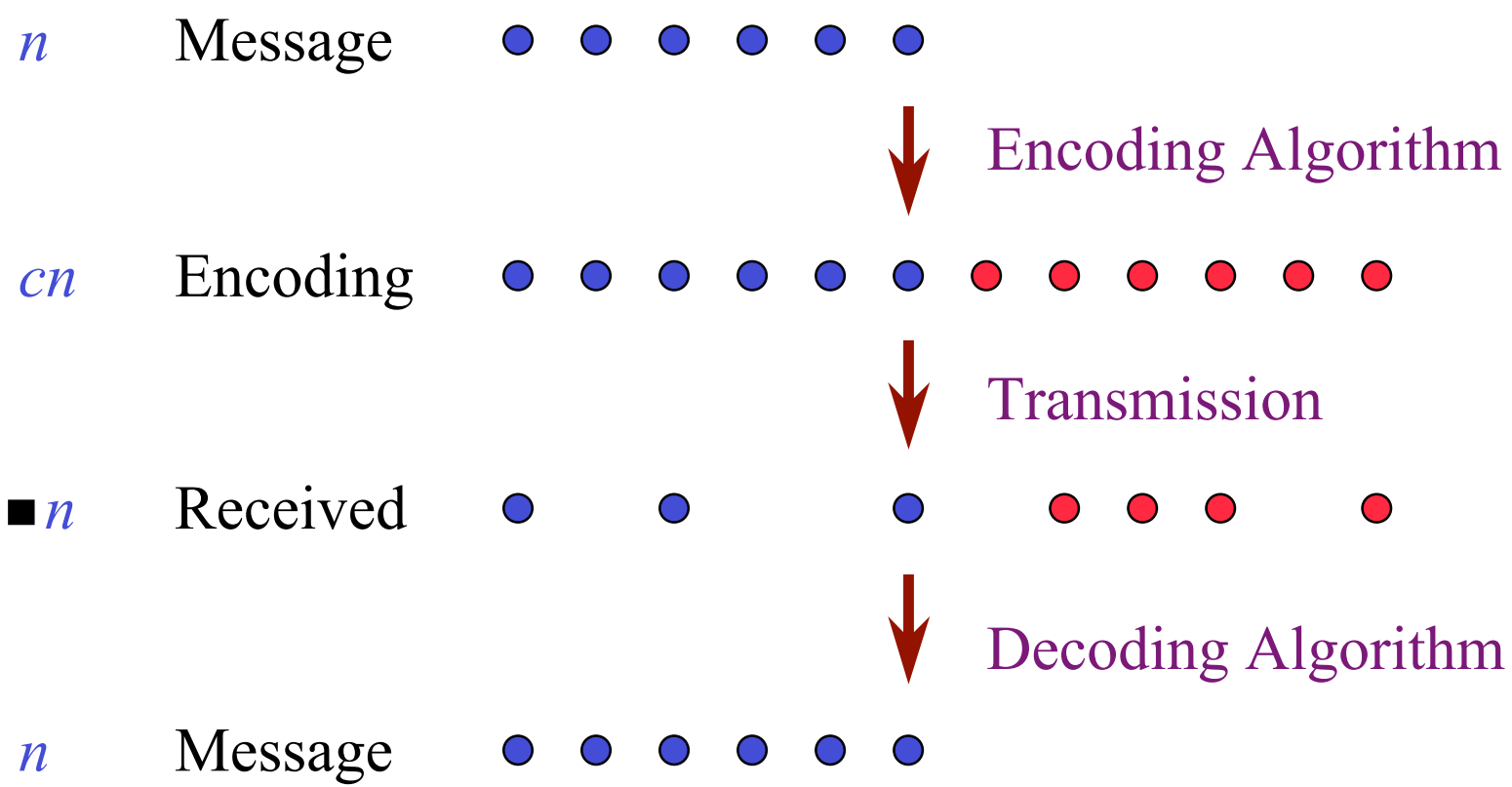
Digital fountains seem great!

But do they really exist?

How Do We Build a Digital Fountain?

- We can construct (approximate) digital fountains using erasure codes.
  - Including Reed-Solomon, Tornado, LT, fountain codes.
- Generally, we only come close to the ideal of the paradigm.
  - Streams not truly infinite; encoding or decoding times; coding overhead.

Digital Fountains through Erasure Codes



Reed-Solomon Codes

- In theory, can produce an unlimited number of encoding symbols, only need k to recover.
- In practice, limited by:
  - Field size (usually 256 or 65,536)
  - Quadratic encoding/decoding times
- These problems ameliorated by striping data.
  - But raises overhead; now many more than k packets required to recover.
- Conclusion: may be suitable for some applications, but far from practical or theoretical goals of a digital fountain.

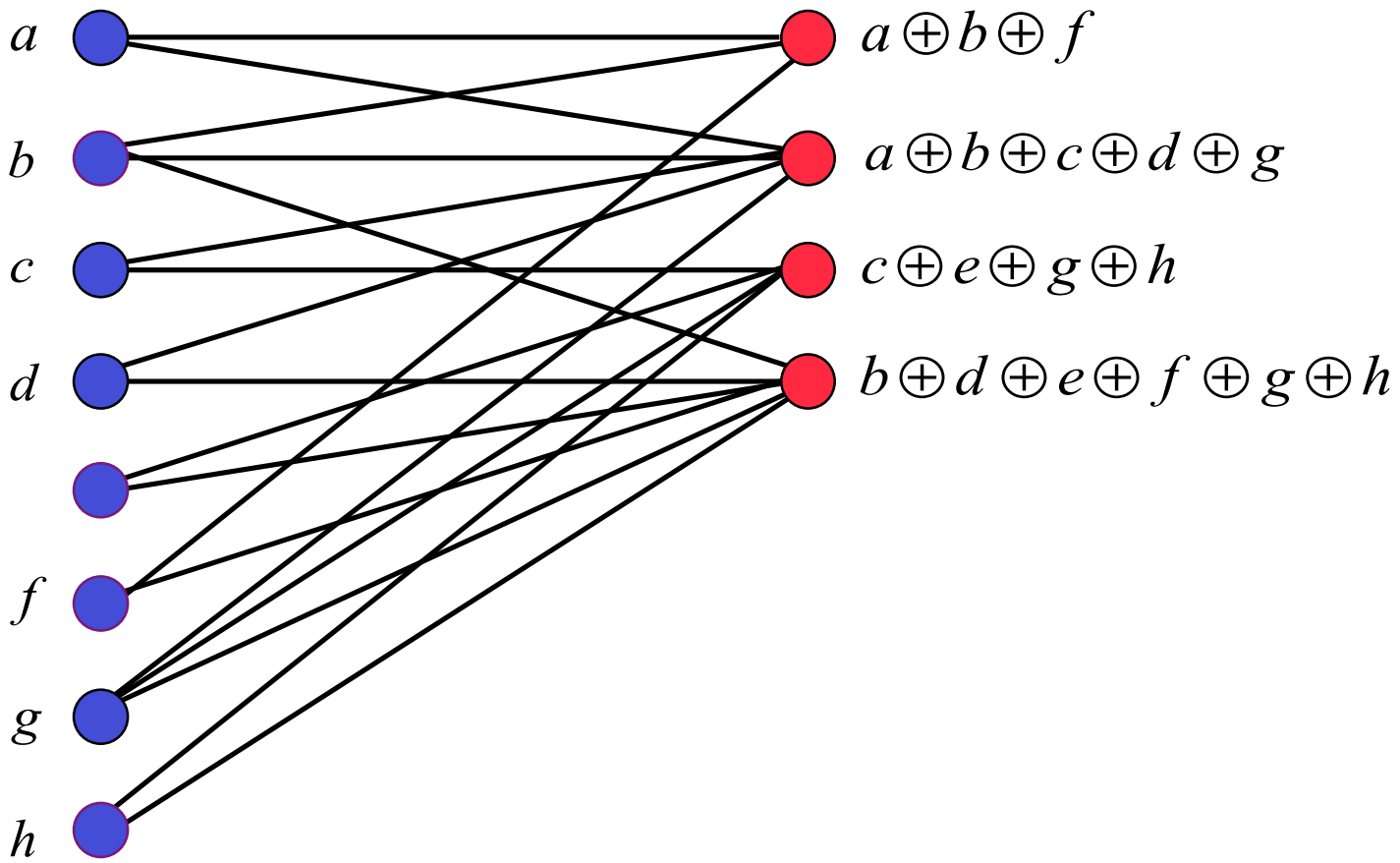
Tornado Codes

- *Irregular* low-density parity check codes.
- Based on graphs: k input symbols lead to n encoding symbols, using XORs.
  - Sparse set of equations derived from input symbols.
  - Solve received set of equations using back substitution.
- Properties:
  - Graph of size n agreed on by encoder, decoder, and stored.
  - Need $k(1+\varepsilon)$ symbols to decode, for some $\varepsilon > 0$.
  - Encoding/decoding time proportional to $n \ln (1/\varepsilon)$.

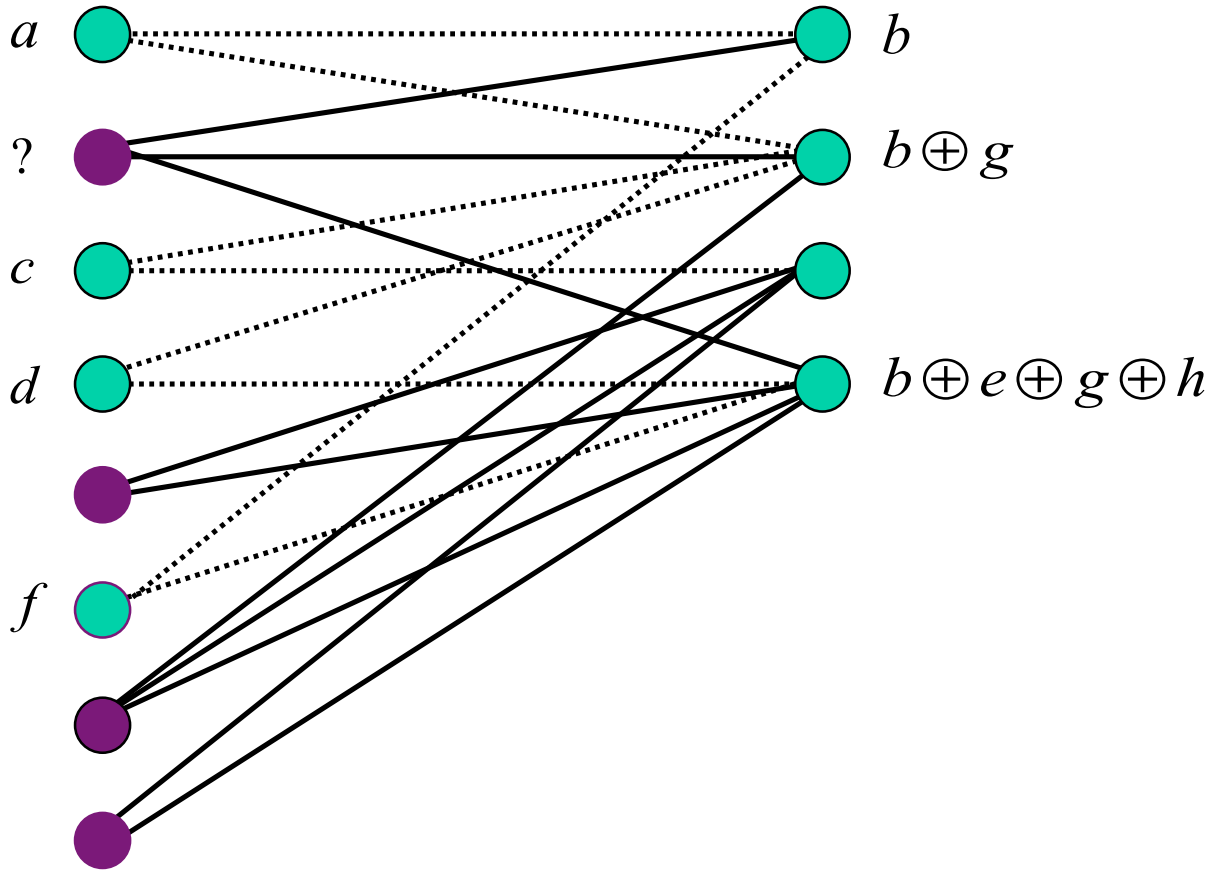
Tornado Codes

An Example

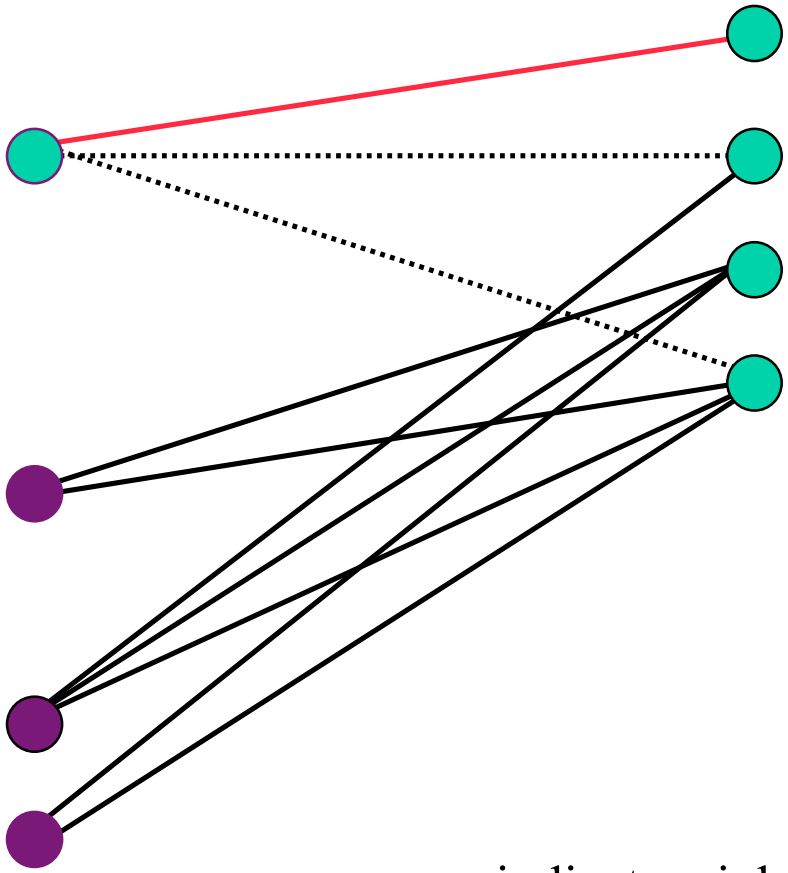
Encoding Process



Decoding Process: Direct Recovery



Decoding Process: Substitution Recovery

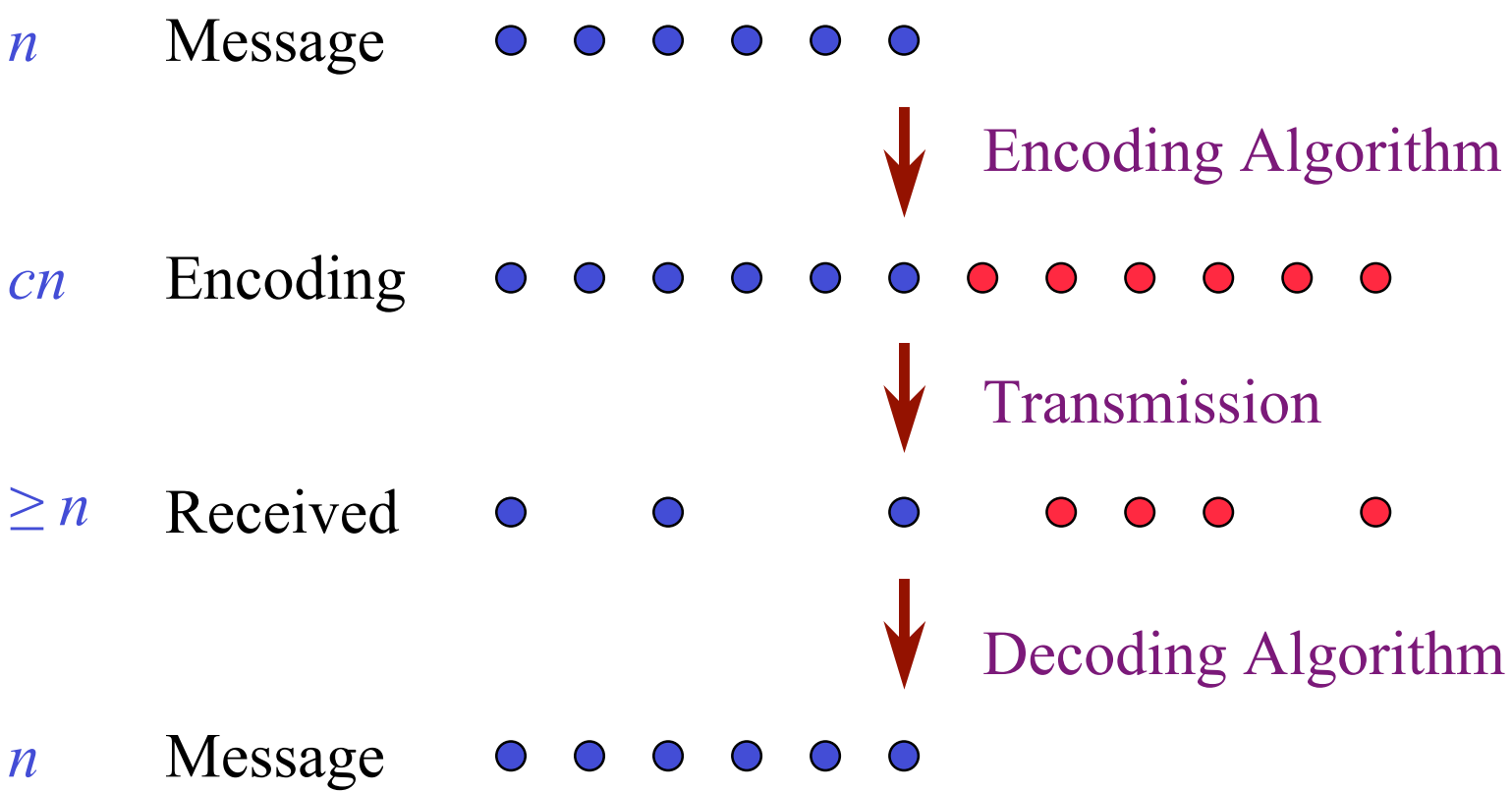


indicates right node has one edge

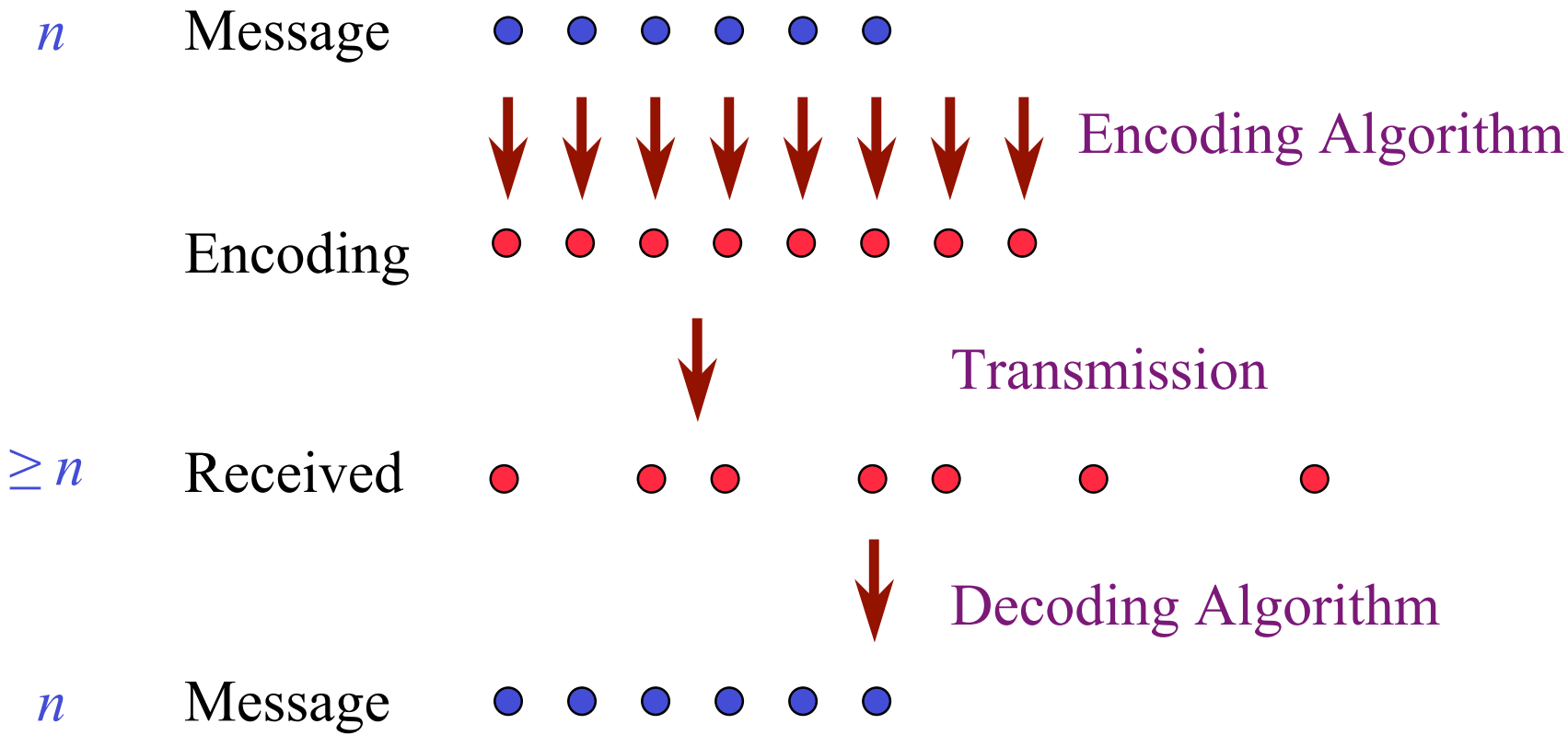
Tornado Codes: Weaknesses

- Encoding size n must be fixed ahead of time.
- Memory, encoding and decoding times proportional to n, not k.
- Overhead factor of $(1+\varepsilon)$.
  - Hard to design around. In practice $\varepsilon = 0.05$.
- Conclusion: Tornado codes a dramatic step forward, allowing good approximations to digital fountains for many applications.
- Key problem: fixed encoding size.

Digital Fountains through Erasure Codes : Problem



Digital Fountains through Erasure Codes : Solution



LT Codes

- Key idea: graph is *implicit*, rather than explicit.
  - Each encoding symbol is the XOR of a random subset of neighbors, *independent of other symbols*.
  - Each encoding symbol carries a *small header*, telling what message symbols it is the XOR of.
- No initial graph; graph derived from received symbols.
- Properties:
  - "Infinite" supply of packets possible.
  - Need $k + o(k)$ symbols to decode.
  - Decoding time proportional to $k \ln k$.
  - On average, $\ln k$ time to produce an encoding symbol.

LT Codes

- Conclusion: making the graph implicit gives us an almost ideal digital fountain.
- One remaining issue: why does average degree need to be around $\ln k$?
  - Standard coupon collector's problem: for each message symbol to be hit by some equation, need $k \ln k$ variables in the equations.
- Can remove this problem by *pre-coding*.

Rateless/Raptor Codes

- Pre-coding independently described by Shokrollahi, Maymoukov.
- Rough idea:
  - Expand original k message symbols to $k\,(1+\varepsilon)$ symbols using (for example) a Tornado code.
  - Now use an LT code on the expanded message.
  - Don't need to recover *all* of the expanded message symbols, just *enough* to recover original message.

Raptor/Rateless Codes

- Properties:
  - "Infinite" supply of packets possible.
  - Need $k(1+\varepsilon)$ symbols to decode, for some $\varepsilon > 0$.
  - Decoding time proportional to $k \ln (1/\varepsilon)$.
  - On average, $\ln (1/\varepsilon)$ (constant) time to produce an encoding symbol.
  - Very efficient.

Raptor codes give, in practice, a digital fountain.

Impact on Coding

- These codes are examples of low-density parity-check (LDPC codes).
- Subsequent work: designed LDPC codes for error-correction using these techniques.
- Recent developments: LDPC codes approaching Shannon capacity for most basic channels.

Putting Digital Fountains To Use

- Digital fountains are out there.
  - Digital Fountain, Inc. sells them.
- Limitations to their use:
  - Patent issues.
  - Perceived complexity.
    - Lack of reference implementation.
  - What is the killer app?

Patent Issues

- Several patents / patents pending on irregular LDPC codes, LT codes, Raptor codes by Digital Fountain, Inc.
- Supposition: this stifles external innovation.
  - Potential threat of being sued.
  - Potential lack of commercial outlet for research.
- Suggestion: unpatented alternatives that lead to good approximations of a digital fountain would be useful.
  - There is work going on in this area, but more is needed to keep up with recent developments in rateless codes.

Perceived Complexity

- Digital fountains are now not that hard…
- …but networking people do not want to deal with developing codes.
- A research need:
  - A publicly available, easy to use, reasonably good black box digital fountain implementation that can be plugged in to research prototypes.
- Issue: patents.
  - Legal risk suggests such a black box would need to be based on unpatented codes.

What's the Killer App?

- Multicast was supposed to be the killer app.
  - But IP multicast was/is a disaster.
  - Distribution now handled by contend distributions companies, e.g. Akamai.
- Possibilities:
  - Overlay multicast.
  - Big wireless: e.g. automobiles, satellites.
  - Others???

Conclusions, Part I

Stop thinking of data as an ordered stream of bytes.

Think of data as a digital fountain.

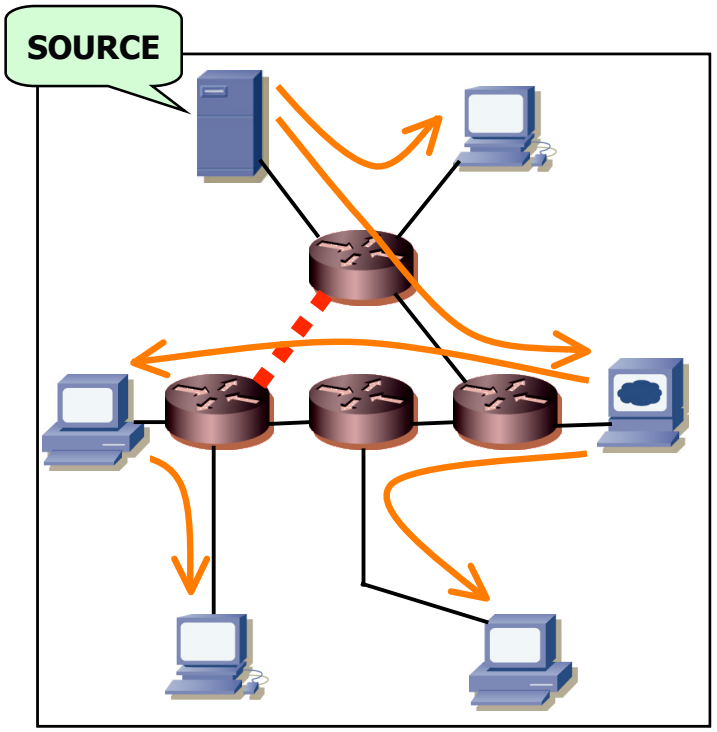
Digital fountains are implementable in practice with erasure codes.

A Short Breather

- We've covered digital fountains.
- Next up:
  - Digital fountains for overlay networks.
  - And other tricks!

Pause for questions, 30 second stretch.

Overlays for Content Delivery



- A substitute for IP multicast.
- Build distribution topology out of unicast connections (tunnels).
- Requires active participation of end-systems.
- Native IP multicast unnecessary.
- Saves considerable bandwidth over *N* * unicast solution.
- Basic paradigm easy to build and deploy.

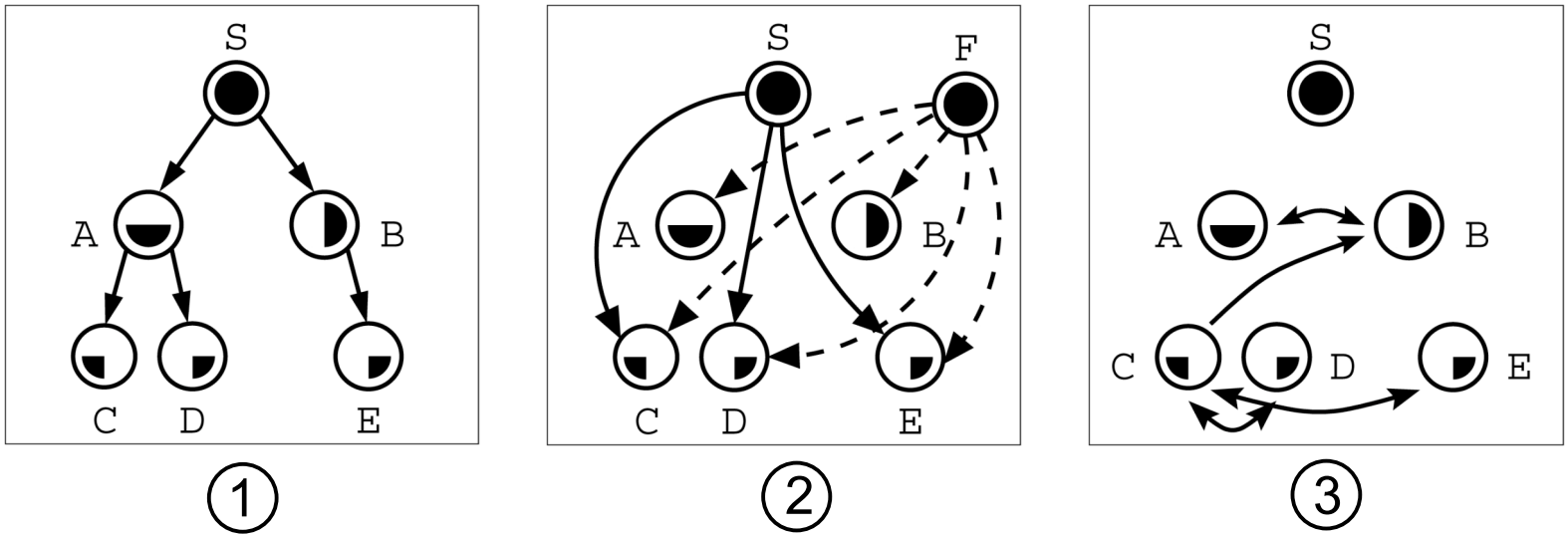
- Bonus:

  Overlay topology can *adapt* to network conditions by self-reconfiguration.

Limitations of Existing Schemes

- Tree-like topologies.
  - Rooted in history (IP Multicast).
  - Limitations:
    - bandwidth decreases monotonically from the source.
    - losses increase monotonically along a path.
- Does this matter in practice?
  - Anecdotal and experimental evidence says yes:
    - Downloads from multiple mirror sites in parallel.
    - Availability of better routes.
    - Peer-to-peer: Morpheus, Kazaa and Grokster.

An Illustrative Example



1. A basic tree topology.

2. Harnessing the power of parallel downloads.

3. Incorporating collaborative transfers.

Our Philosophy

- Go beyond trees.
  - Use additional links and bandwidth by:
    - downloading from multiple peers in parallel
    - taking advantage of "perpendicular" bandwidth
  - Has potential to significantly speed up downloads…
- But only effective if:
  - collaboration is carefully orchestrated
  - methods are amenable to frequent adaptation of the overlay topology

Suitable Applications

- Prerequisite conditions:
  - Available bandwidth between peers.
  - Differences in content received by peers.
  - Rich overlay topology.
- Applications
  - Downloads of large, popular files.
  - Video-on-demand or nearly real-time streams.
  - Shared virtual environments.

Use Digital Fountains!

- Intrinsic resilience to packet loss, reordering.
- Better support for transient connections via stateless migration, suspension.
- Peers with full content can always generate useful symbols.
- Peers with partial content are more likely to have content to share.

- But using a digital fountain comes at a price:
  - Content is no longer an ordered stream.
  - Therefore, collaboration is more difficult.

Informed Content Delivery:

Definitions and Problem Statement

- Peers A and B have working sets of symbols S_A, S_B drawn from a large universe U and want to collaborate effectively.
- Key components:
  - **Summarize**: Furnish a concise and useful sample of a working set to a peer.
  - **Approximately Reconcile**: Compute as many elements in $S_A - S_B$ as possible and transmit them.
- Do so with minimal control messaging overhead.

Approximate Reconciliation

- Suppose summarization suggests collaboration is worthwhile.
- Goal: compute as many elements in $S_A - S_B$ as possible, with low communication.
- Idea: we do not need all of $S_A - S_B$, just as much as possible.
  - Use Bloom filters.

Lookup Problem

- Given a set $S_A = \{x_1, x_2, x_3, \ldots x_n\}$ on a universe U, want to answer queries of the form:

$$\text{Is } y \in S_A?$$

- Bloom filter provides an answer in
  - "Constant" time (time to hash).
  - Small amount of space.
  - But with some probability of being wrong.

Bloom Filters

Start with an m bit array, filled with 0s.

B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hash each item x_j in S k times. If $H_i(x_j) = a$, set $B[a] = 1$.

B	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

To check if y is in S, check B at $H_i(y)$. All k values must be 1.

B	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

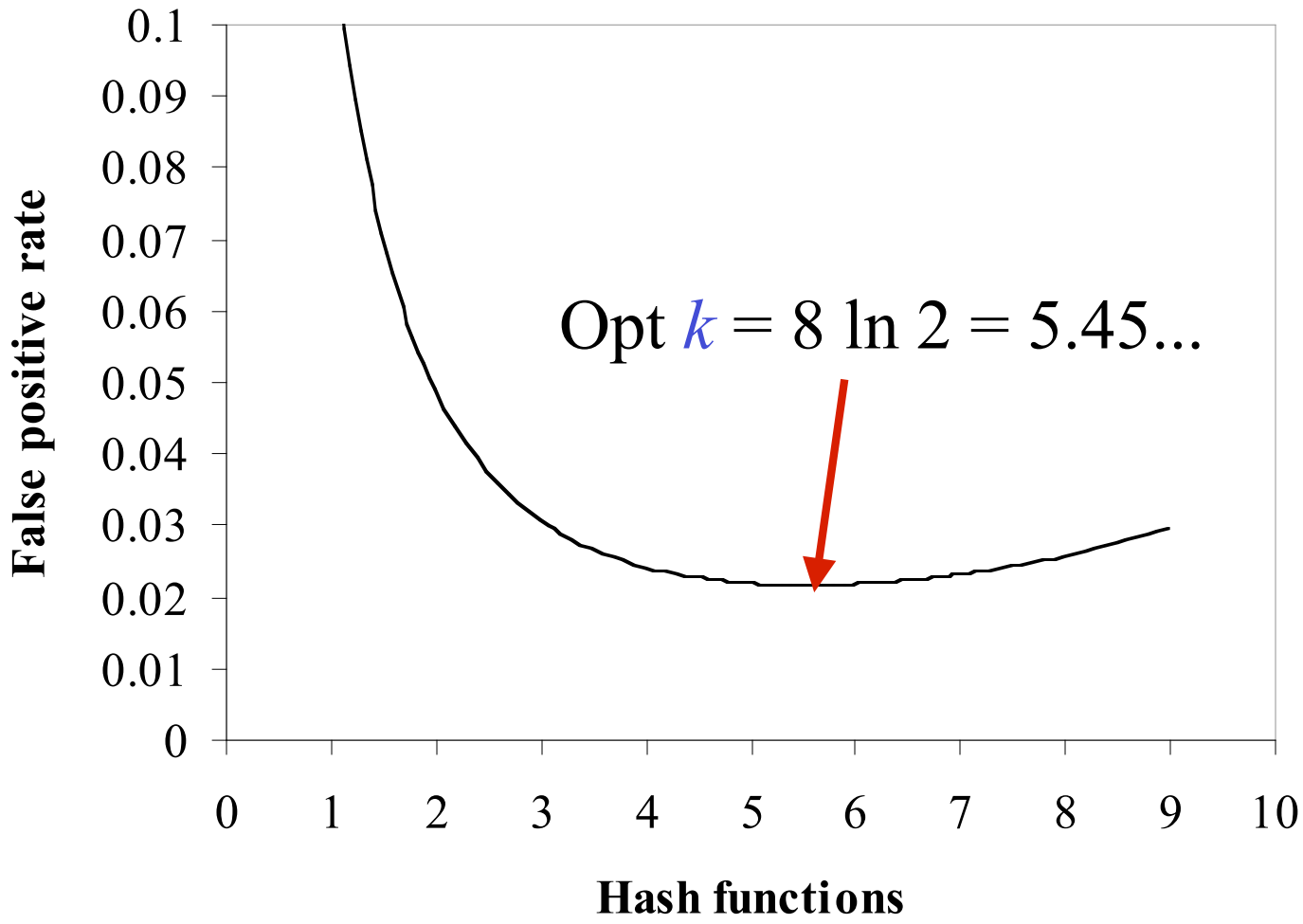
Possible to have a false positive; all k values are 1, but y is not in S.

B	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

Errors

- Assumption: We have good hash functions, look random.
- Given m bits for filter and n elements, choose number k of hash functions to minimize false positives:
  - Let $p = \Pr[\text{cell is empty}] = (1 - 1/m)^{kn} \approx e^{-kn/m}$
  - Let $f = \Pr[\text{false pos}] = (1 - p)^k \approx (1 - e^{-kn/m})^k$
- As k increases, more chances to find a 0, but more 1's in the array.
- Find optimal at $k = (\ln 2)m/n$ by calculus.

Example


0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
False positive rate
Opt $k = 8 \ln 2 = 5.45...$
0
1
2
3
4
5
6
7
8
9
10
Hash functions

$m/n = 8$

Bloom Filters for Reconciliation

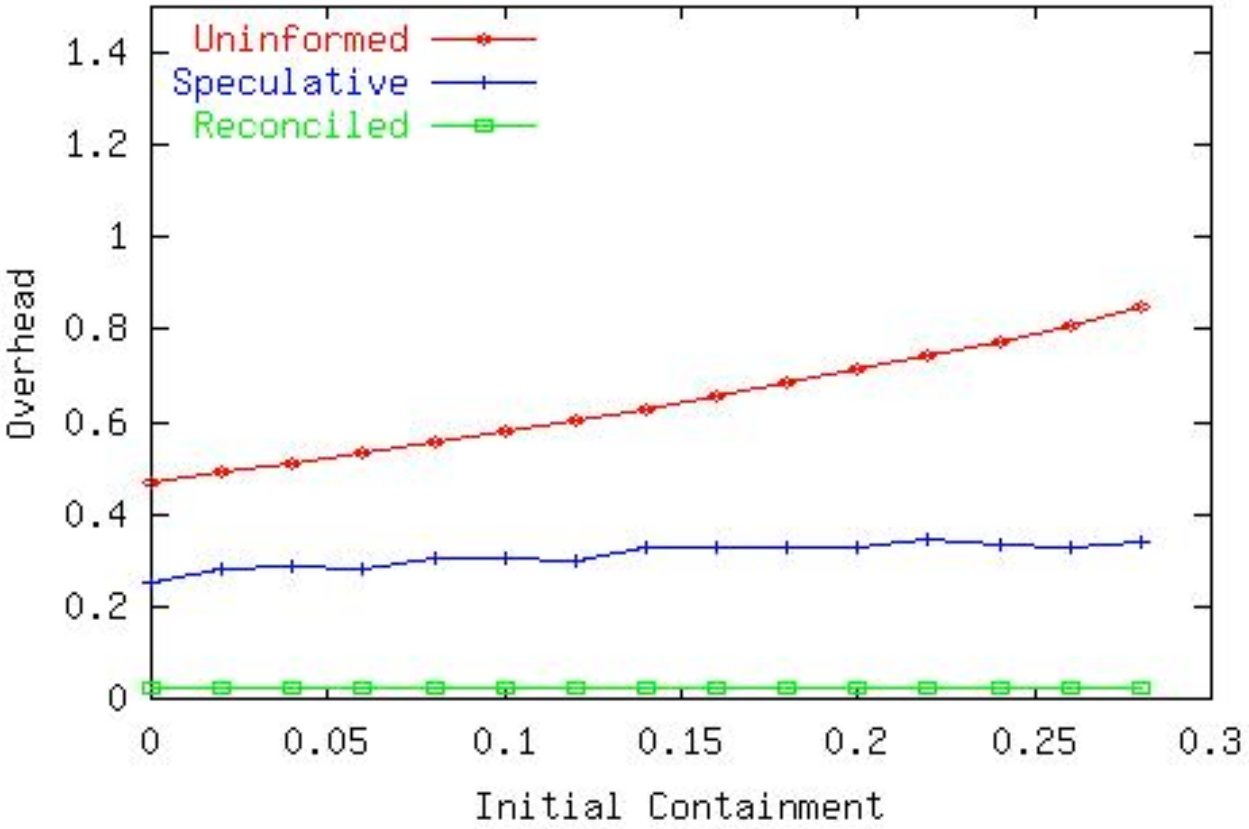
- *B* transmits a Bloom filter of its set to *A*; *A* then sends packets from the set difference.
  - All elements will be in difference: no false negatives.
  - Not all element in difference found: false pos.
- Improvements
  - Compressed Bloom filters
  - Approximate Reconciliation Trees

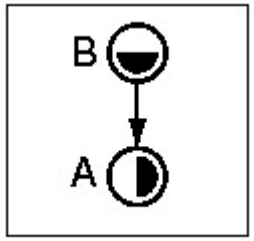
Experimental Scenarios

- Three methods for collaboration
  - Uninformed: *A* transmits symbols at random to *B*.
  - Speculative: *B* transmits a minwise summary to *A*; *A* then sends recoded symbols to *B*.
  - Reconciled: *B* transmits a Bloom filter of its set to *A*; *A* then sends packets from the set difference.

- Overhead: $\frac{\text{symbols received - symbols needed}}{\text{symbols needed}}$
  - Decoding overhead: with erasure codes, fixed 2.5%.
  - Reception overhead: useless duplicate packets.
  - Recoding overhead: useless recoding packets.

Pairwise Reconciliation



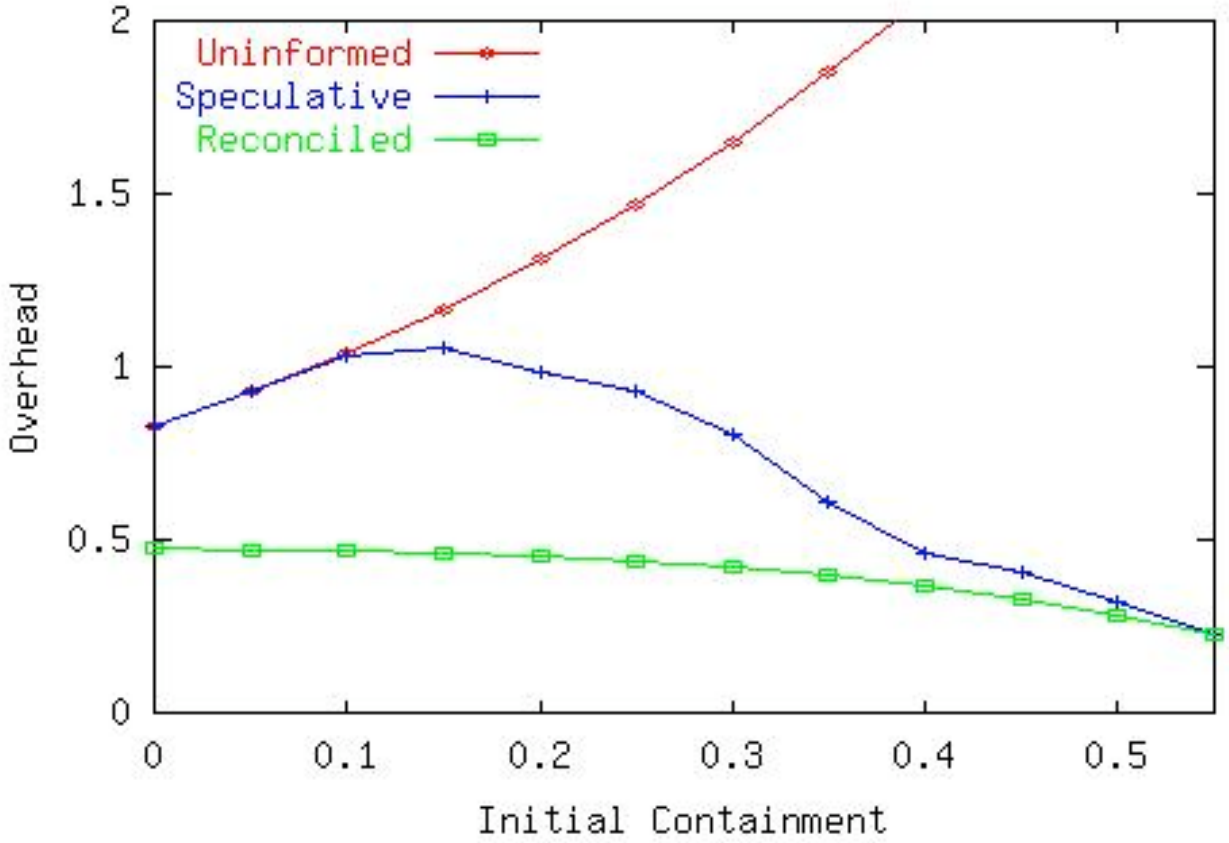


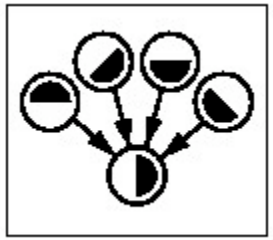
128MB file
96K input symbols

115K distinct symbols
in system initially

Containment of B in A: $\frac{|S_A \cap S_B|}{|S_B|}$

Four peers in parallel



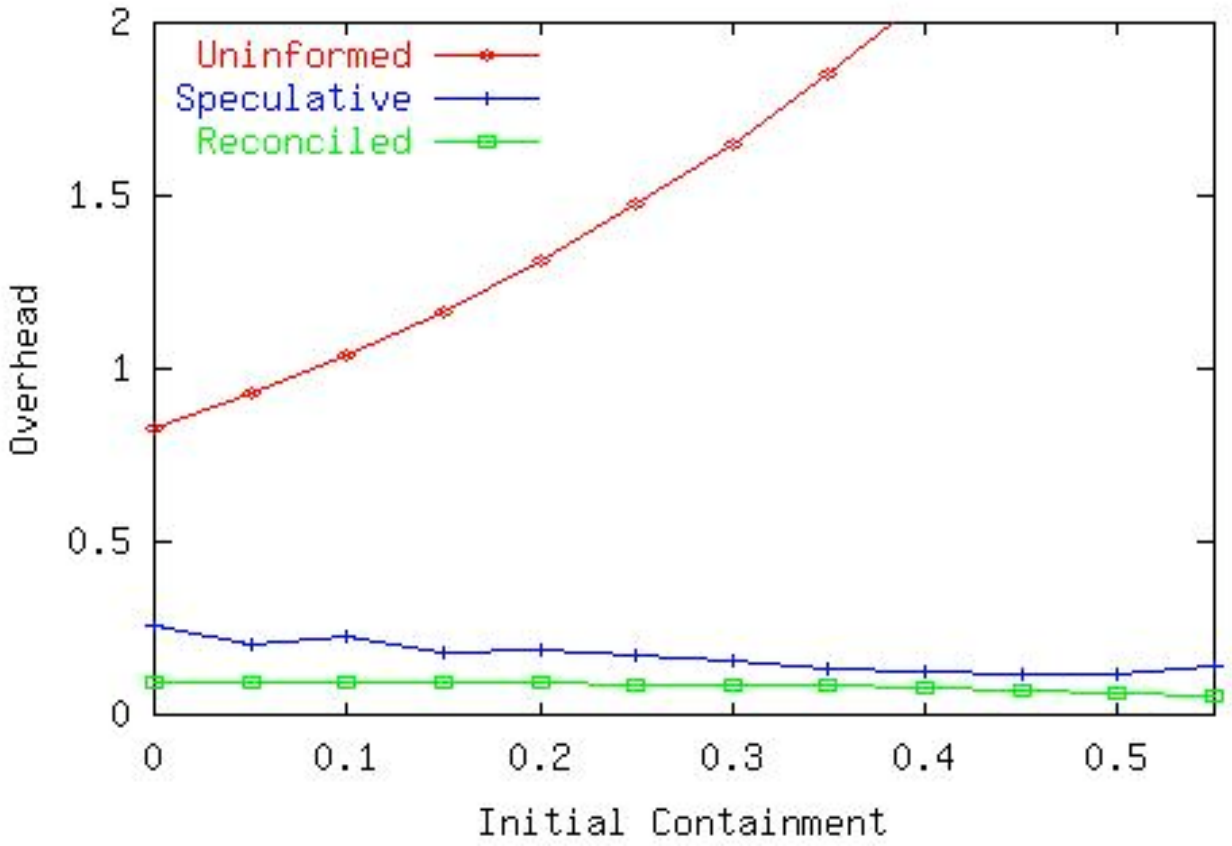


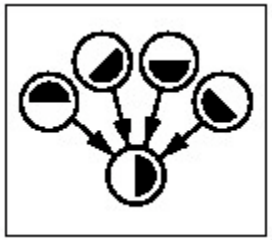
128MB file
96K input symbols

105K distinct symbols
in system initially

Containment of B in A: $\frac{|S_A \cap S_B|}{|S_B|}$

Four peers, periodic updates





128MB file
96K input symbols

105K distinct symbols
in system initially

Filters updated at
every 10%.

Containment of B in A: $\frac{|S_A \cap S_B|}{|S_B|}$

Subsequent Work

- Maymounkov: each source sends a stream of consecutive encoded packets.
  - Possibly simplifies collaboration, with loss of flexibility.
- Bullet (SOSP ’03):
  - An implementation with our ideas, plus purposeful distribution of different content.
- Network coding
  - Nodes inside the network can compute on the input, rather than just the endpoints.
  - Potentially more powerful paradigm
  - Practice?

Conclusions

- Even with ultimate routing topology optimization, the choice of **what** to send is paramount to content delivery.
- Digital fountain model ideal for fluid and ephemeral network environments.
- Collaborations with coded content worthwhile.
- Richly connected topologies are key to harnessing perpendicular bandwidth.
- Wanted: more algorithms for intelligent collaboration.