Digital Fountain [Byers98a]

CSci551: Computer Networks
SP2006 Thursday Section
John Heidemann

Context

- SRM and Floyd et al did whiteboards
  - used ALF, application-customized naming of data objects, no guranteed ordering
  - receiver-driven, timer-based reliability
  - fairly low-rate (not bulk), rate-limited, somewhat latency tolerant
- other applications
  - **reliable bulk data transfer (Byers et al)**
  - videoconferencing
  - server location and anycast (uses anonymous addressing)

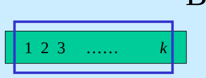
Key ideas

- coding techniques for reliable transfer
  - Tornado codes
  - send both data and "redundant" packets, so that some subset of packets is sufficient to rebuild original data
- layering
  - sender sends over several multicast groups
  - each group builds on the others
- keep cycling through data
- (multicast challenges)
  - (ack or nack) implosion
    - coding fixes by avoiding *any* nacks (no back channel)
    - risk is that you send extra data in the coding
  - receiver heterogenity—different receivers have different RTTs, buffer sizes, bottleneck bandwidth
    - layering lets receivers choose their bitrate
  - late joiners problem
    - solved by cycling through data
  - exposure: retx sent to whole group not needed by some
    - partly everyone has exposure now because of duplicate data
    - but no additional exposure due to specific receiver's losses

Requirements and Assumptions

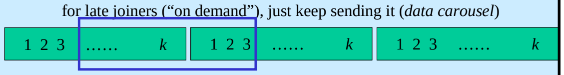
- reliable data transfer
- "on-demand"
  - clients can join at any time
- efficiency
  - minimal additional latency
  - minimal amount of extra data
- minimal assumptions about network
  - heterogeneous pipes and recievers
  - no backchannel needed
- error rates from 0-60%
- IP multicast model
  - would also work with EXPRESS

Basic Ideas

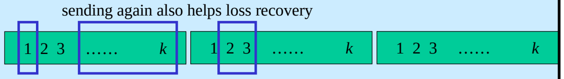


to send a file, split it into blocks and send them for reliability, clients can tell if they have all (and they can then do error recovery if not)

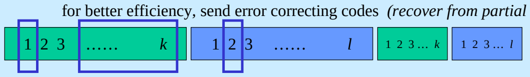
for late joiners ("on demand"), just keep sending it (*data carousel*)



sending again also helps loss recovery



for better efficiency, send error correcting codes *(recover from partial info)*



Coding Concepts

- source file
  - *k* packets
  - elements x_i
- encoded file
  - *n* packets (*k* source and *l* redundant)
  - elements y_i
- received file
  - (*k-e*) source packets and *e* redundant
  - regenerate original *k* packets

Coding Algorithms

- encoding the data
  - Reed-Solomon codes
  - Tornado codes
  - (or just send the source data multiple times)
- transmission timing
  - in-order (send *k* original, then *l* redundant)
  - interleaved (mix original and redundant to tolerate burst losses)

Coding Comparision

[Byers98a, Table 1]

	Tornado	Reed-Solomon
Decoding inefficiency	$1+\epsilon$ required	1
Encoding times	$(k+\ell)\ln(1/\epsilon)P$	$k\ell P$
Decoding times	$(k+\ell)\ln(1/\epsilon)P$	$k\epsilon P$
Basic operation	XOR	Field operations

Decoding Benchmarks		
SIZE	Reed-Solomon Codes	Tornado Codes
	Cauchy	Tornado Z
250 KB	2.06 seconds	0.18 seconds
500 KB	8.4 seconds	0.24 seconds
1 MB	40.5 seconds	0.31 seconds
2 MB	199 seconds	0.44 seconds
4 MB	800 seconds	0.74 seconds
8 MB	3166 seconds	1.28 seconds
16 MB	13629 seconds	2.27 seconds

[Byers98a, Table 3]

- R-S codes are much more computationally expensive than Tornado
- downside:
  - R-S codes reproduce *k* source blocks from *any k* encoded blocks
  - Tornado may require some extra (typically ~5%, worst case 10%)

Layered Coding

- problem:
  - different receivers have different bitrates
  - can't just send at one rate
    - too slow for some, too fast for others
- solution:
  - *layered encoding*
  - different clients subscribe to different layers
  - higher layers are much faster
  - ideal originally from [McCanne96a] (in supplementary papers)
    - see that paper for good explanation of how to find the right number of layers, etc.

Layered Strategy

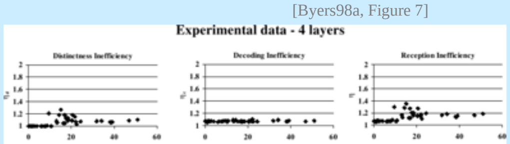
- higher layers are exponentially faster
- ordering is such that any horizontal or vertical slice gets all the data you need

[Byers98a, Table 5]

Layer	Bandwidth per Round	Packets sent during Rd 1	Rd 2	Rd 3	Rd 4	Rd 5	Rd 6	Rd 7	Rd 8
3	4	0-3	4-7	0-3	4-7	0-3	4-7	0-3	4-7
2	2	4-5	0-1	6-7	2-3	4-5	0-1	6-7	2-3
1	1	6	2	4	0	7	3	5	1
0	1	7	3	5	1	6	2	4	0

Evaluation of Layering

[Byers98a, Figure 7]



distinctness cost: how many fully dupicate pkts do you get?

decoding inefficiency: overhead due to tornado coding

reception cost: two things put together

Other questions/observations?

- xxx