BSI
P2P Streaming

Davide Frey
WIDE Team
INRIA

Some slides by: Y. Chen, R. Karki, A-M Kermarrec, M. Monod, M. Zhang,

Large-scale broadcast/multicast

Application-level multicast (ALM)

1. Structured peer to peer networks
   - Flooding
   - Tree-based
2. Content streaming (today)
   - Multiple Trees
   - Mesh
   - Gossip

Setting

broadcastin
g

A source produces **multimedia content**

n viewers (*n* large)

VS

IP TV, Web TV, P2P TV, …

BBC iPlayer

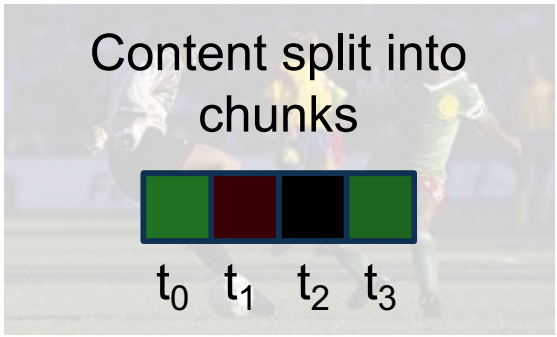
192K requests/day

78K users/day

244K simultaneous users (incl. VoD)

BBC iStats (April 2010)

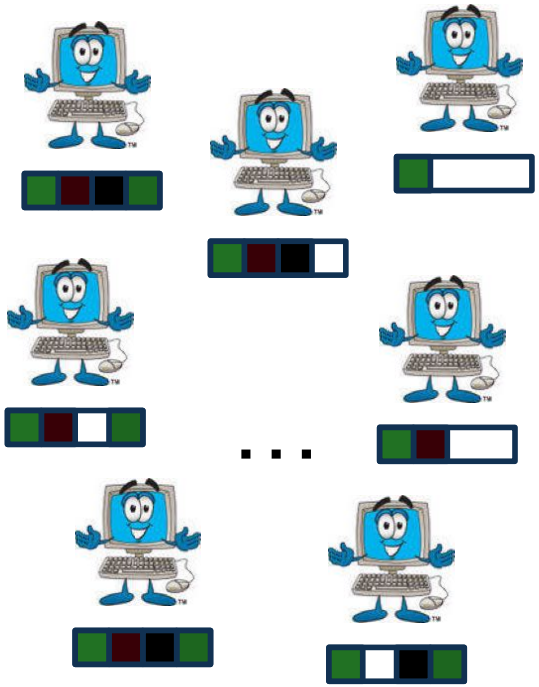
Streaming Basics



multimedia content

time-critical ordered large

disseminatio n



n viewers (*n* large)

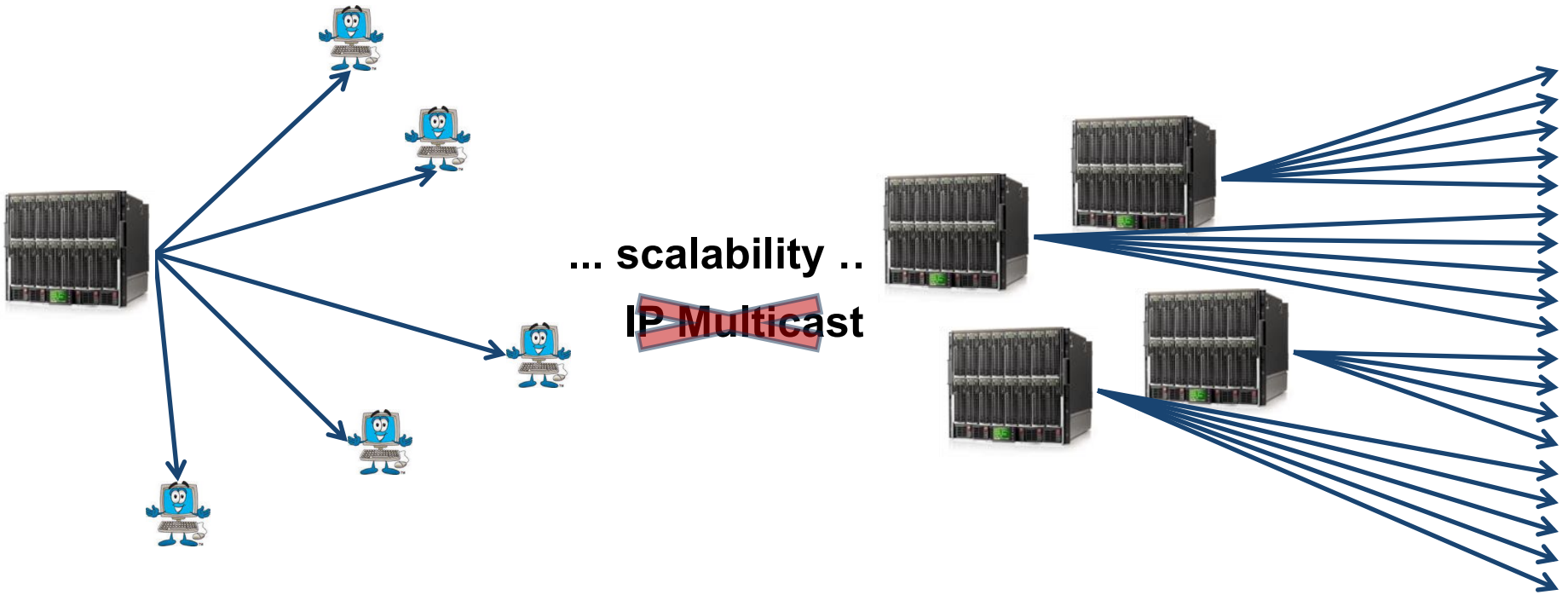
Stream rate ***s*** [kbps]

n viewers want to receive ***s***

Demand = Supply

Intuitive solution

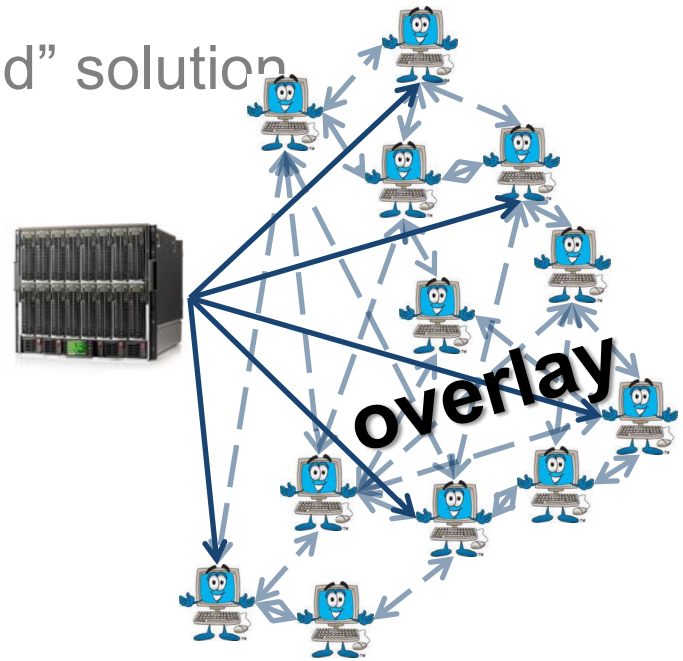
- “Centralized” solution



Participants are pure consumer

Let’s be smarter

“Decentralized” solution

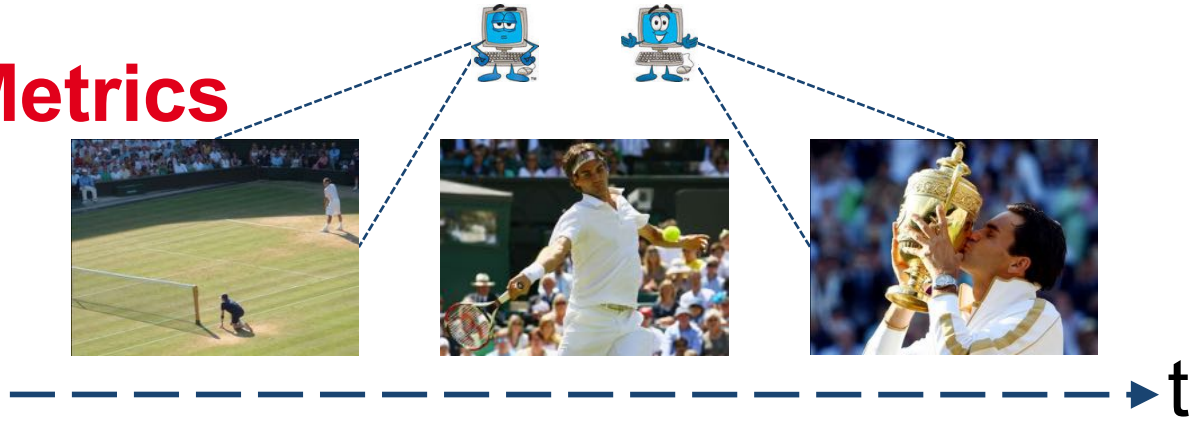


Participants **collaborate**

…most of them!

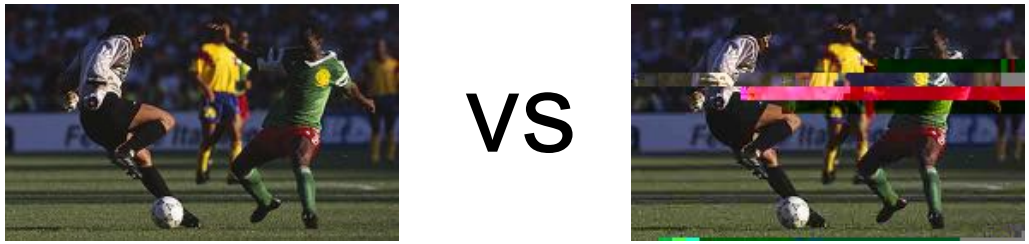
Evaluation Metrics

Stream lag



- Time difference between creation at the source and delivery to the clients' player
- Also:
  - delay penalty (delay wrt IP multicast)
  - Hop count

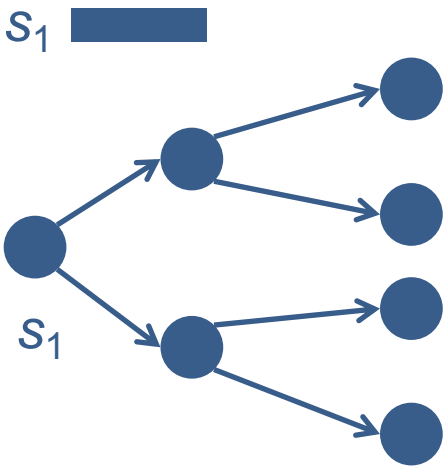
Stream quality

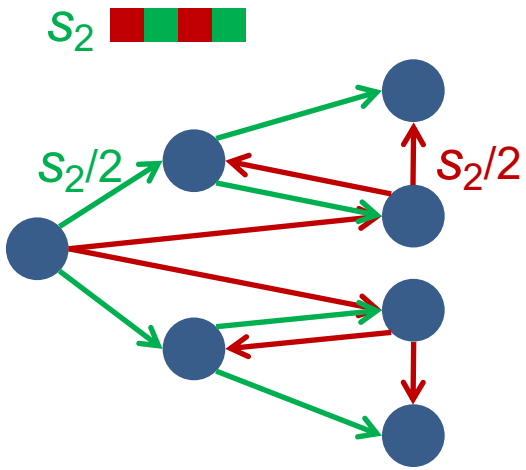


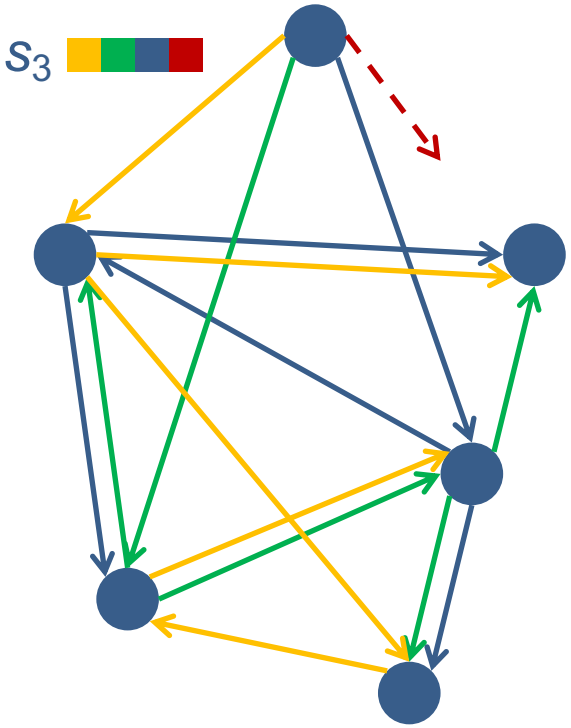
- Maximum 1% jitter means at least 99% of the groups are complete = 99%-playback
  - Incomplete groups does not mean "blank"
- Also: delivery-ratio or continuity index

Tree-based ALM

Streaming Approaches



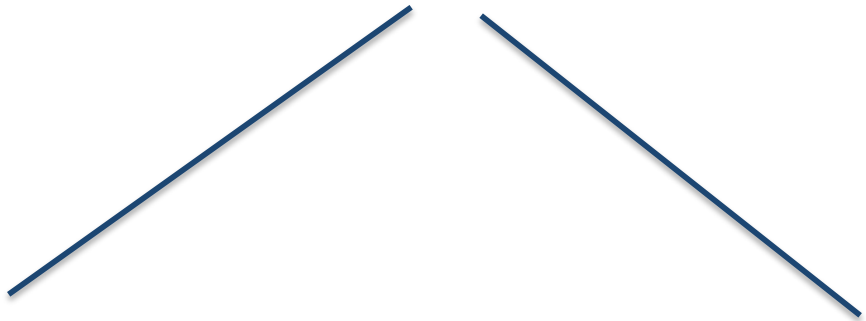




Single tree	Multiple trees	Mesh/Gossip
s_1 is constrained by design	Upload of nodes: multiple of s_2/z	s_3 optimal
Disconnection	Partial disconnection	Connected is not enough
Build/maintain tree	Build/maintain z trees	Peer selection, Packet scheduling

Addressing the Limitations of Trees

Some peers do not forward



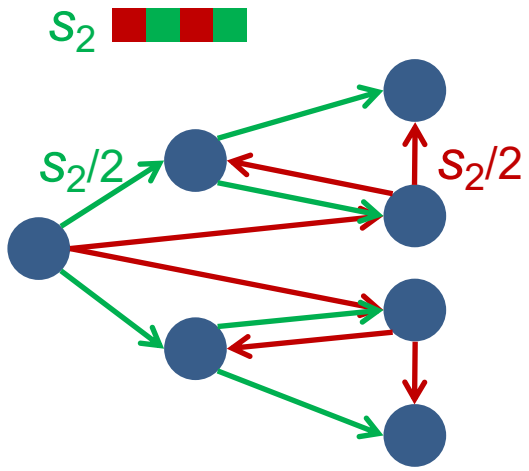
Multiple Trees

Mesh/Gossip

SplitStream approach

Content divided in *stripes*
Each stripe is distributed on an independent tree

- Load balancing
  - Internal nodes in one tree are leaves in others
- Reliability
  - Failure of a node leads to unavailability of x stripes if parents are independent and using appropriate coding protocols

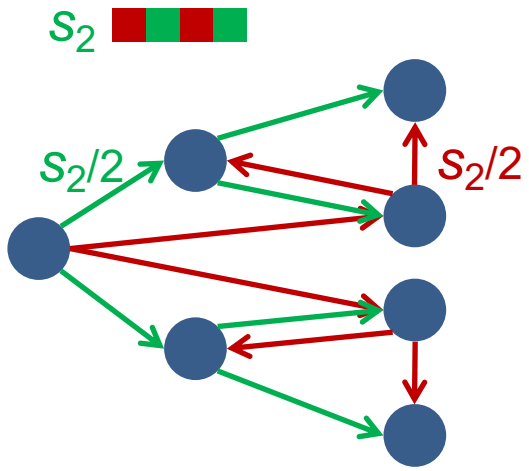


[SOSP 2003 « *SplitStream: High-Bandwidth Multicast in Cooperative Environment* »]

SplitStream approach

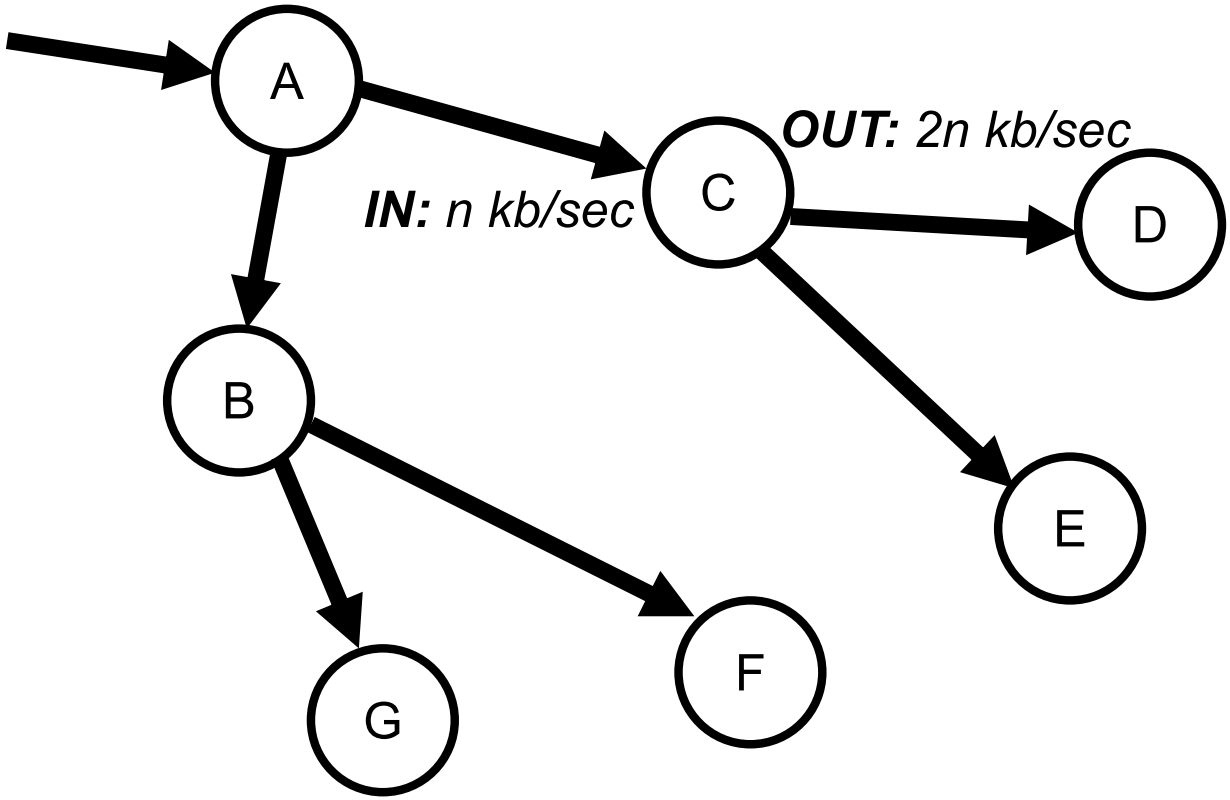
Content divided in *stripes*
Each stripe is distributed on an independent tree

- Load balancing
  - Internal nodes in one tree are leaves in others
- Reliability
  - Failure of a node leads to unavailability of x stripes if parents are independent and using appropriate coding protocols

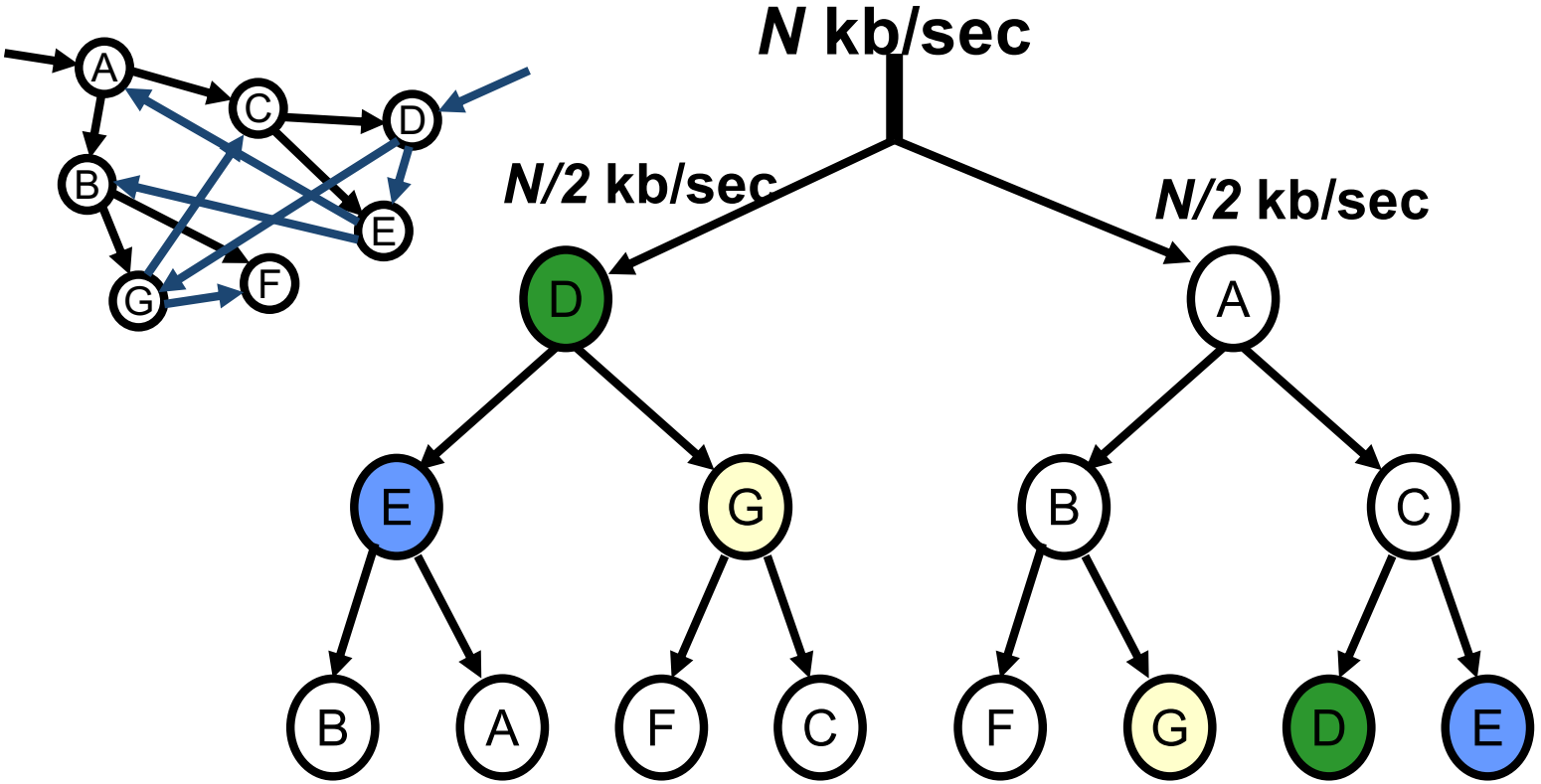


[SOSP 2003 « *SplitStream: High-Bandwidth Multicast in Cooperative Environment* »]

Tree-based ALM: unbalanced



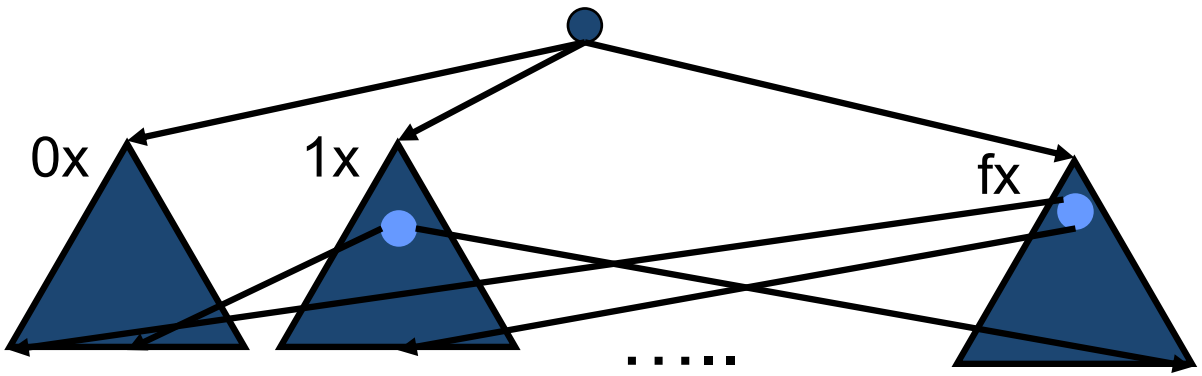
The SplitStream Forest



SplitStream

Construct one tree/group per data stripe

Each stripe identifier starts with a different digit (up to 16 independent stripes)



SplitStream

Main goal: build and maintain multiple multicast trees in a fully decentralized and reliable way so that

- Each client receives the desired number of stripes
- Independent trees
- Control upon bandwidth allocation
- Reasonable latency and network load

Leverage Scribe/Pastry

- Pastry: P2P routing infrastructure (structured, efficient, reliable)
- Scribe: decentralized and efficient tree-based protocol

SplitStream: forest management

Constraints

- Limited out-degree potentially increases the tree depth
- Load balancing to ensure within trees and between trees
- Failure independence of trees

Scribe So

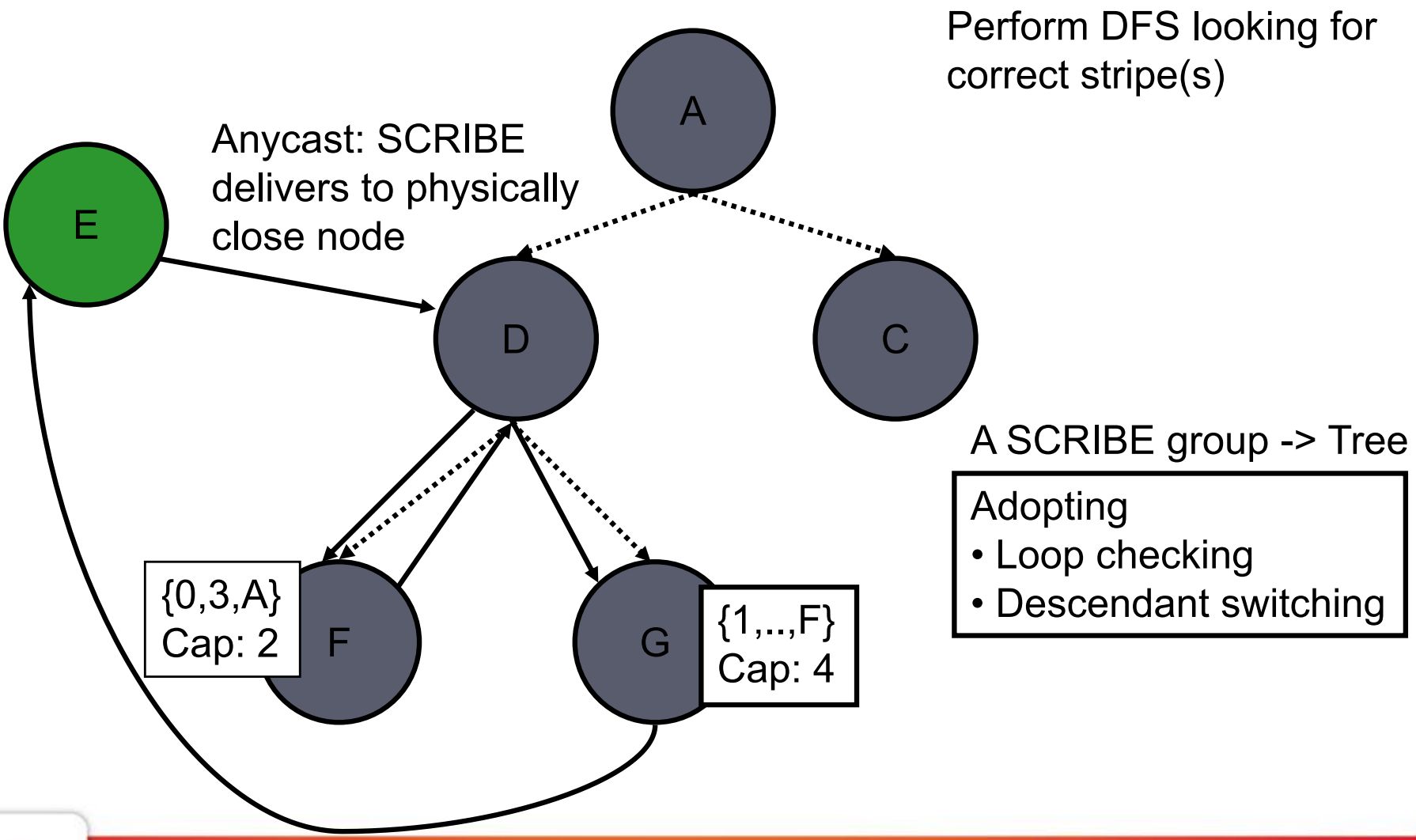
- Overloade

Ineffective because trees are correlated:
Leaf in one tree is interior in another

Splitstream Solution: spare capacity tree

- Always accept new children
- Underloaded nodes join spare capacity tree
- Overloaded nodes give up descendants
- Discard children with shortest prefix match
- Orphans *anycast* to the spare capacity tree to discover new parents

Spare capacity tree



Experiments

Simulations (average on 10 runs)
- Topologies **GT**, Mercator, MS Corp.
- 40000 nodes

Pastry (b=4, leafset = 16)

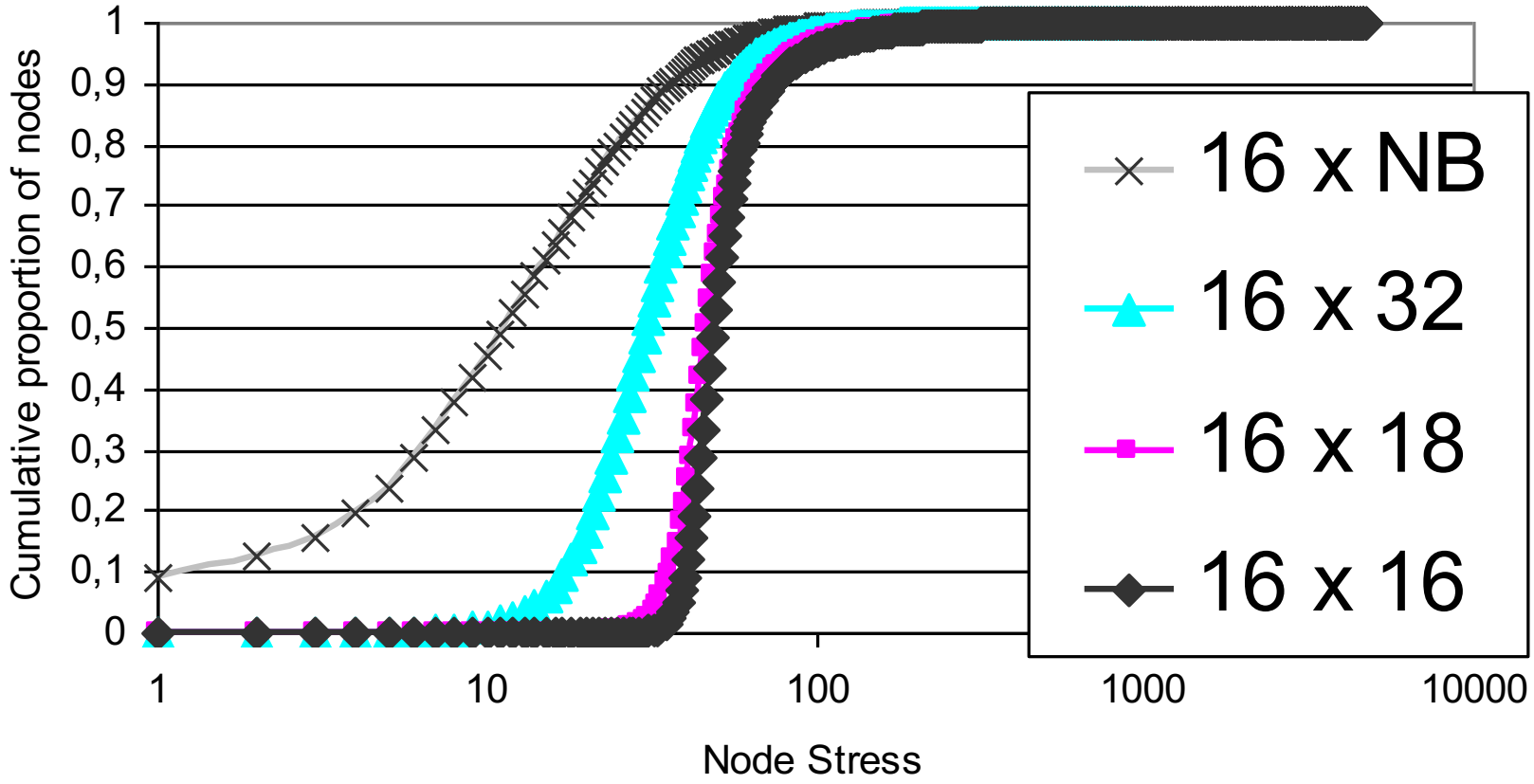
SplitStream : 16 stripes

Configurations in-degree x out-degree
- Impact of spare capacity 16x16, 16x18, 16x32 and 16xNB
- Impact of capacity/needs (Gnutella)

Failure resilience
- Path diversity
- Catastrophic failures (25% of faulty nodes) in a 10,000 node system

Results
- Forest construction
- Multicast performance

Forest construction: load on each node



Forest construction: load on each node

Configuration	16x16	16x18	16x32	16xNB
Max	2971	1089	663	472
Mean	57.2	52.6	35.3	16.9
Med	49.9	47.4	30.9	12

Load decreases as the spare capacity increases

16xNB: no *pushdown* nor orphans

- 16x16: each node contacts the spare capacity tree for 8 stripes on average
- Nodes with id close to the spare capacity tree get the highest load

Forest construction: network load

Measured as the number of msg on physical links

Configuration	16x16	16x18	16x32	16xNB
Max	5893	4285	2876	1804
Mean	74.1	65.2	43.6	21.2
Med	52.6	48.8	30.8	17

Load decreases as the spare capacity increases
Maximum approx. 7 times less than centralized system

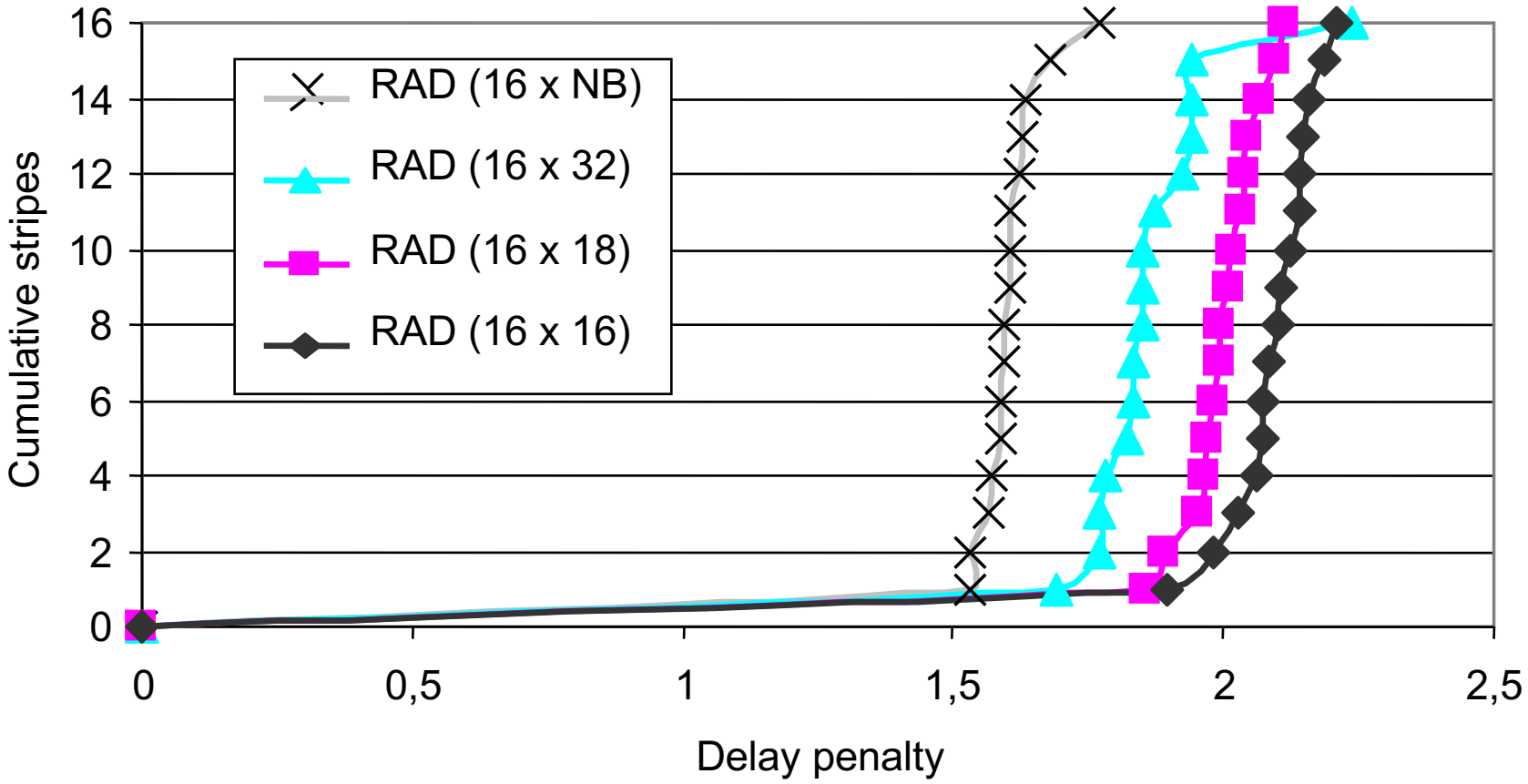
Multicast: link stress

One message/stripe, no failure

Configuration	Centralized (0.43)	Scribe (0.47)	IP (0.43)	16x16 (0.98)	16x18	16x32	16xNB
Max	639984	3990	16	1411	1124	886	1616
Mean	128.9	39.6	16	20	19	19	20
Med	16	16	16	16	16	16	16

- 16xNB : absence of forwarding bounds causes contention on a small
- Set of links
- Splitstream uses a larger fraction of links but load them less

Delay penalty during multicast

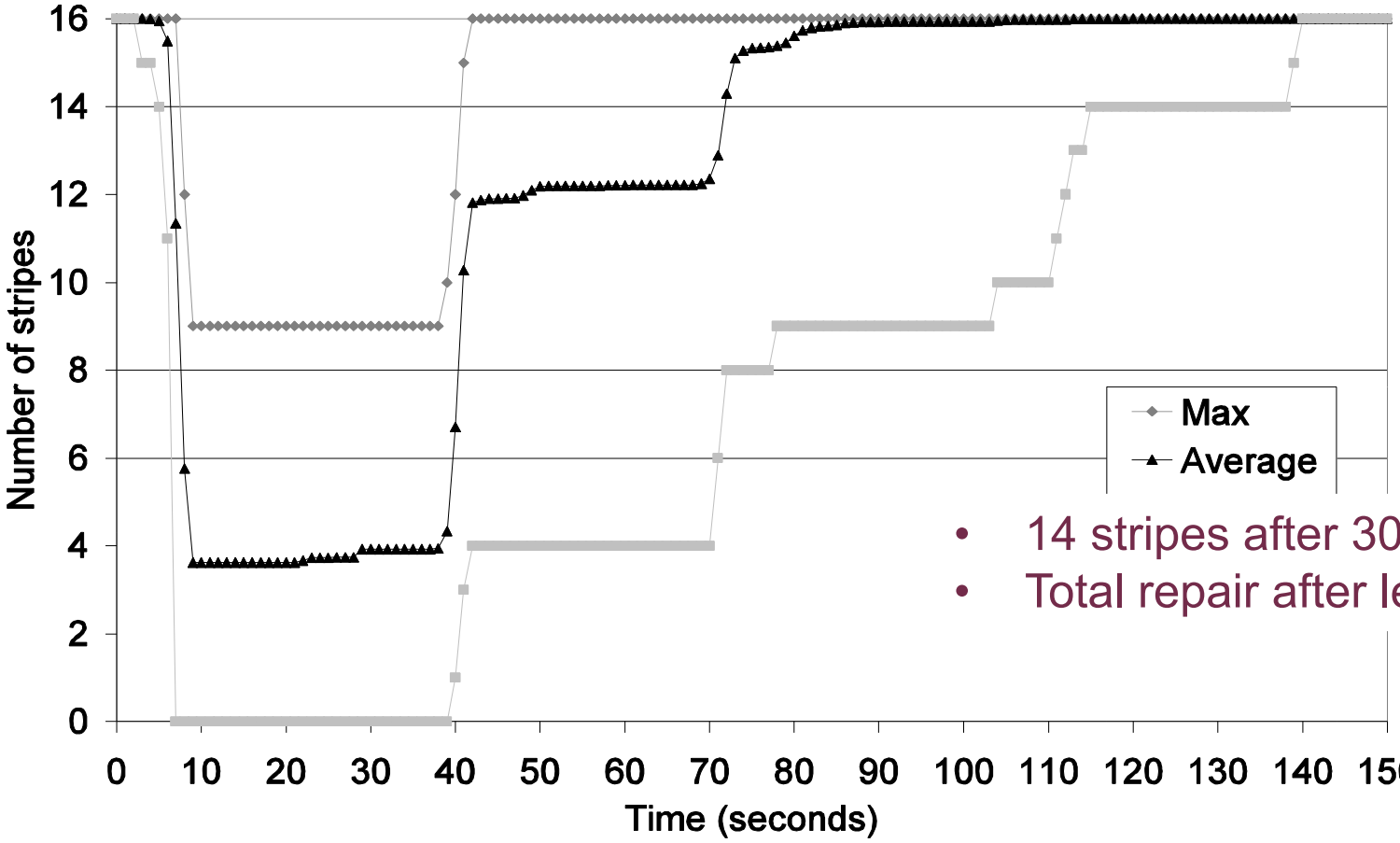


Path diversity

•Number of lost stripes (at most) on each node when the most significant ancestor is faulty (worst case scenario)

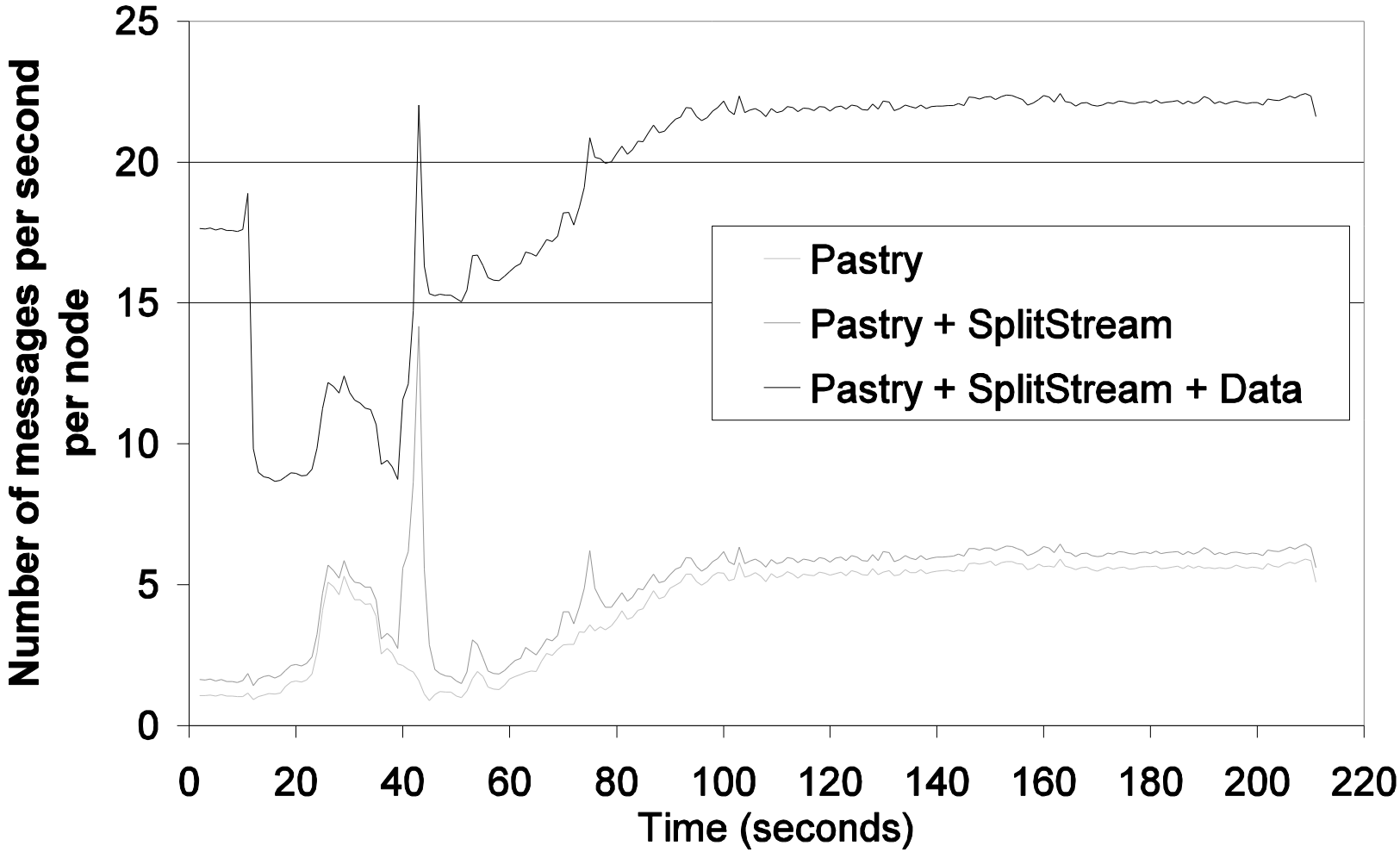
Configuration	16x16	16x32	16xNB
Max	6.8	6.6	1
Mean	2.1	1.7	1
Med	2	2	1

Catastrophic failure (25% of 10,000 nodes are faulty): number of received stripes



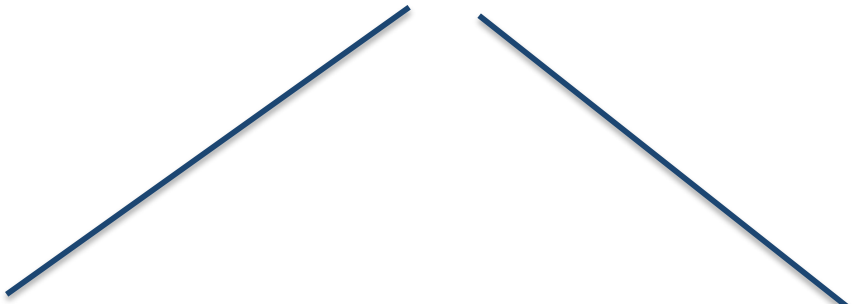
- 14 stripes after 30 s
- Total repair after less than 3mn

Catastrophic failure (25% of 10,000 nodes are faulty): number of messages



Addressing the Limitations of Trees

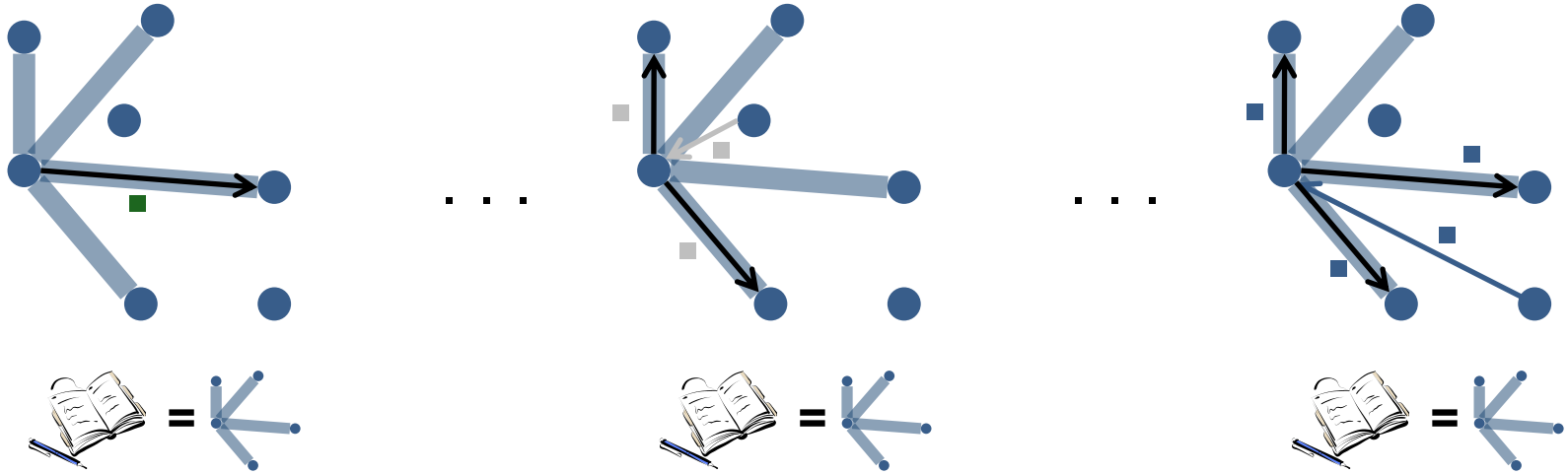
Some peers do not forward



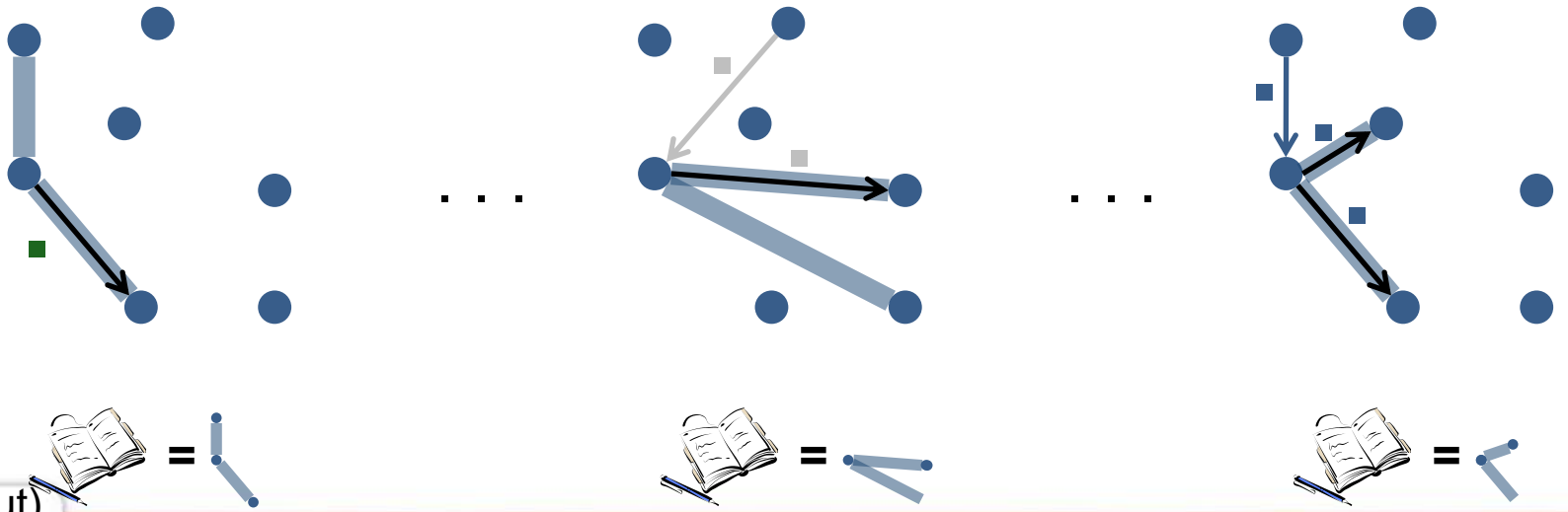
Multiple Trees

Gossip

Mesh vs Gossip

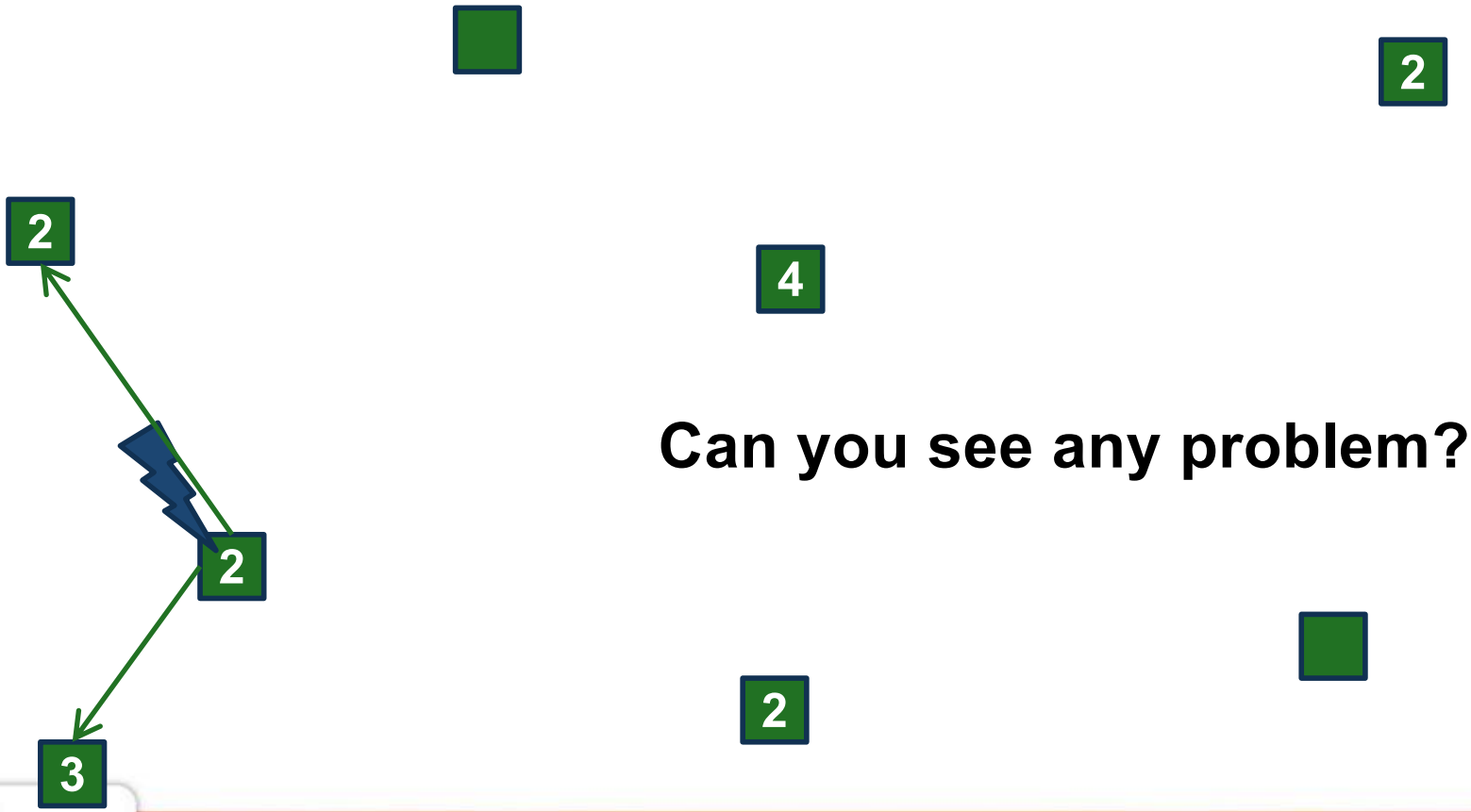


Gossip, $f = 2$



Beyond mesh: Gossip

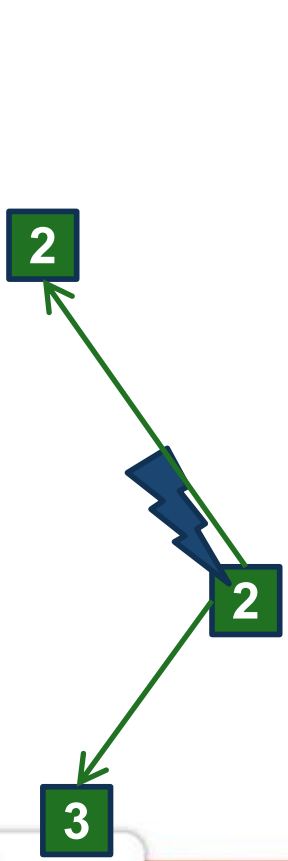
Gossip-based dissemination

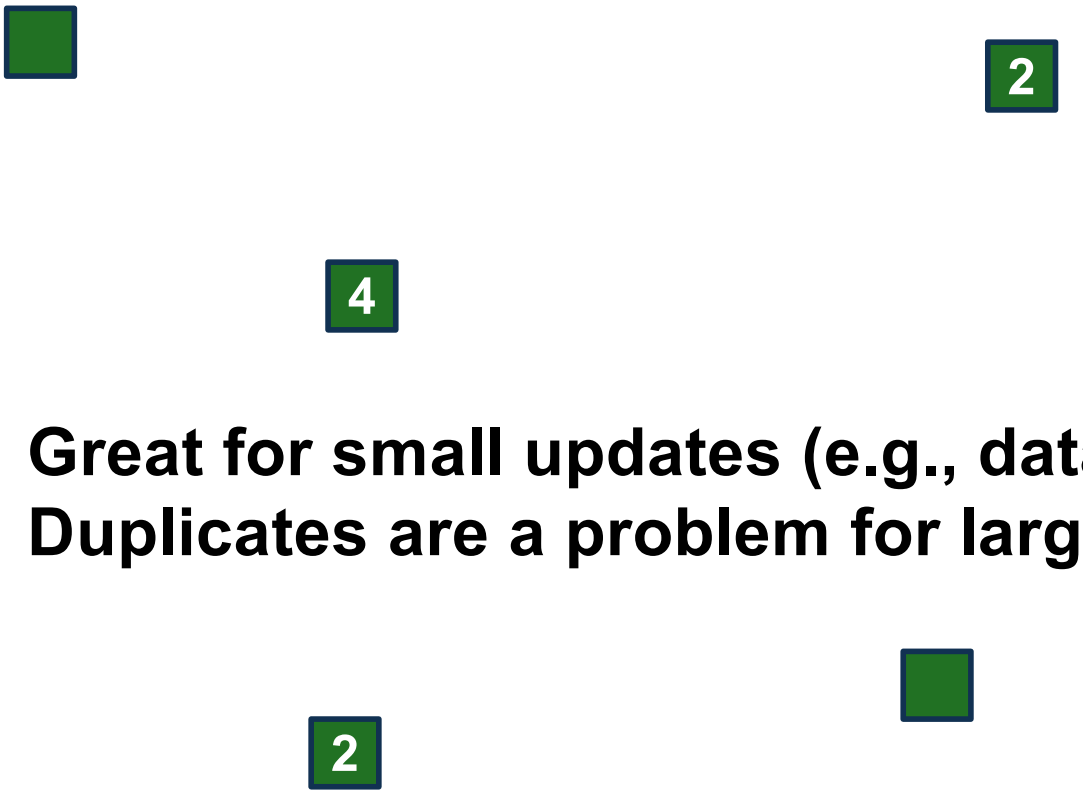


Can you see any problem?

Beyond mesh: Gossip

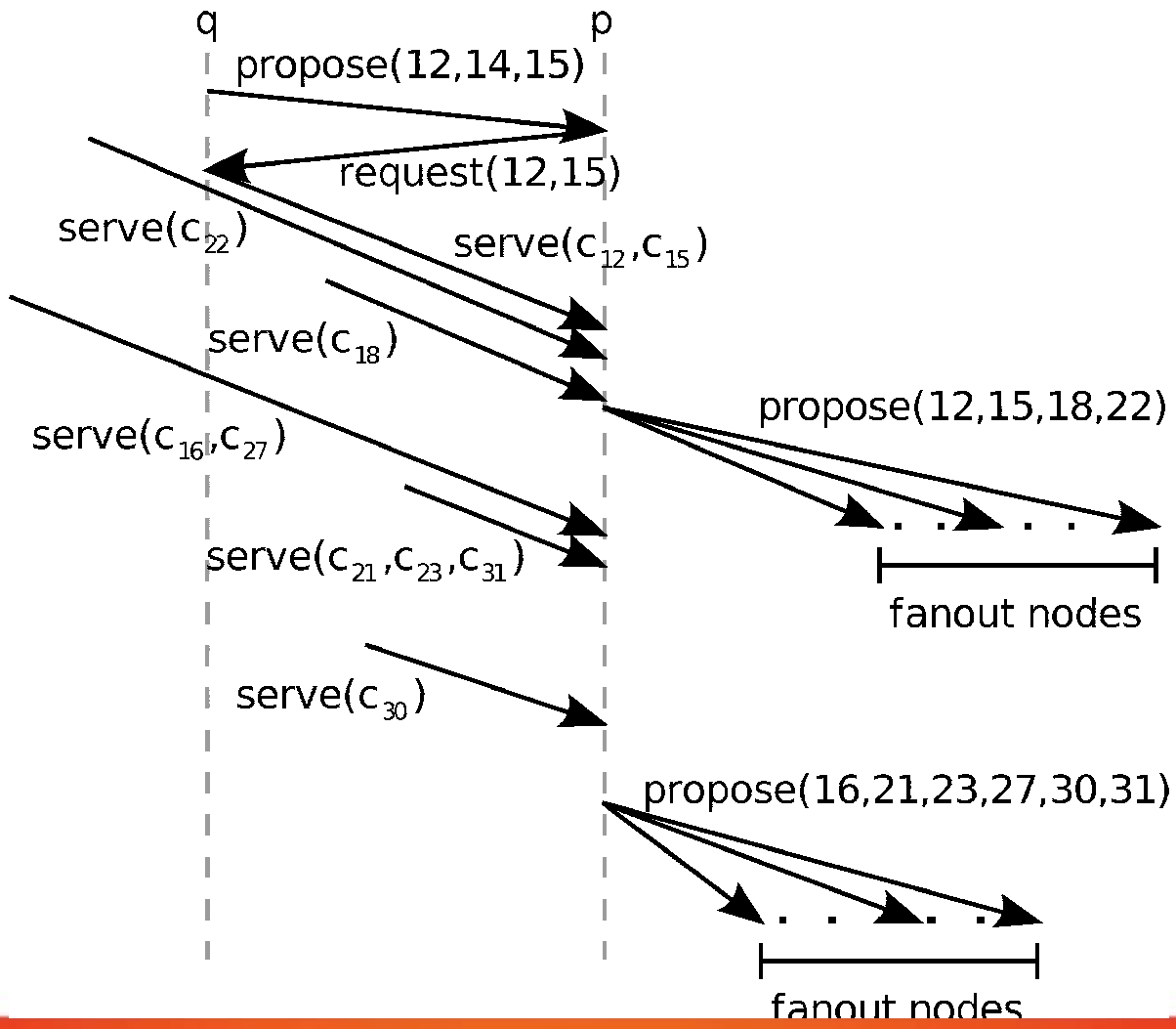
Gossip-based dissemination





Great for small updates (e.g., databases)
Duplicates are a problem for large content..

Three-Phase Gossip



Testing Gossip for Live Streaming

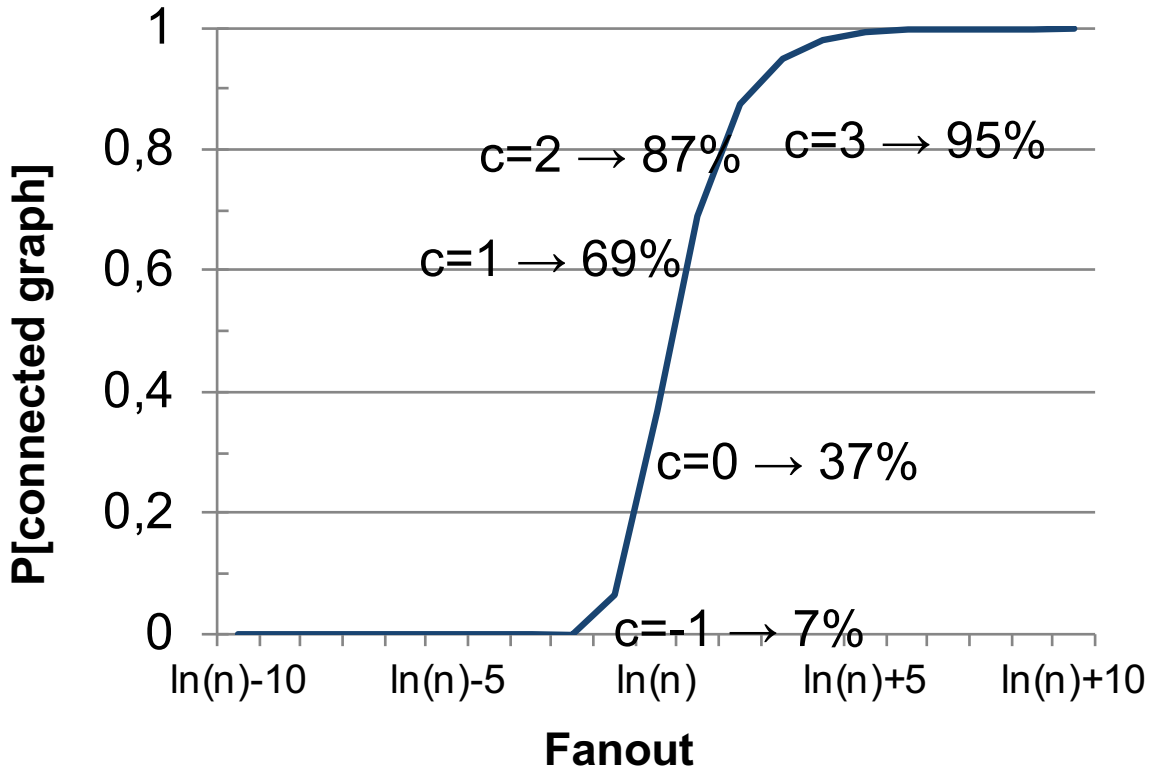
		Grid'5000	PlanetLab
Environment	Nodes	200 (40*5)	230-300
	BW cap	Token bucket (200KB)	Throttling
	Transport layer	UDP + losses (1-5%)	UDP
	Stream rate *s*	680 kbps	551 kbps
	FEC	5%	10%
	Stream (incl. FEC)	714 kbps	600 kbps
Gossip	T_g (gossip period)	200 ms	200-500 ms
	fanout (*f*)	8	7-8
	source's fanout	5	7
	Retransmission	ARQ/Claim	ARQ
	Membership	RPS (Cyclon) and full membership	

Gossip – Theory


Paul Erdős & Alfréd Rényi

1. **fanout = ln(*n*) + *c***

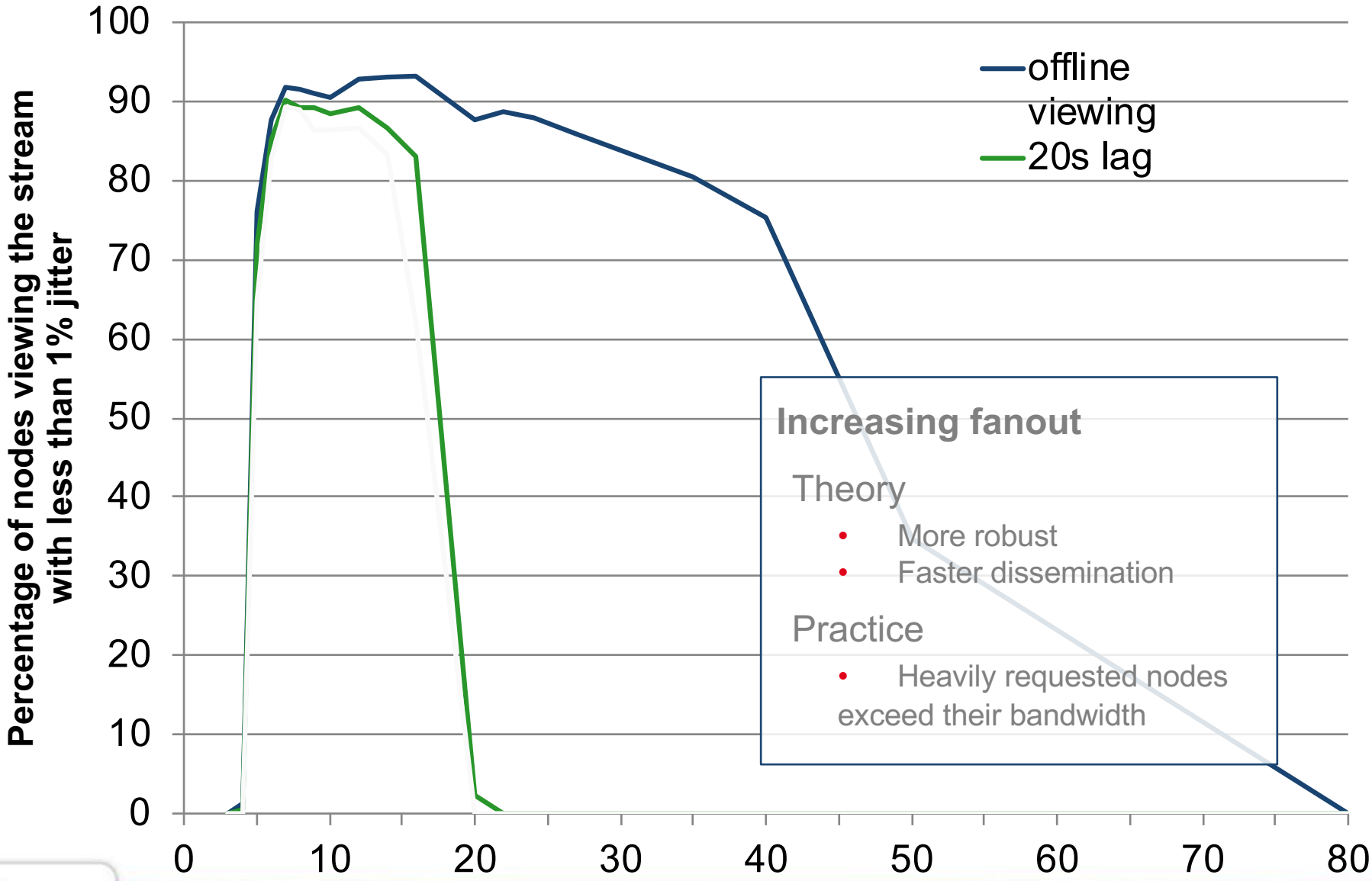
   P[connected graph] goes to exp(-exp(-*c*))



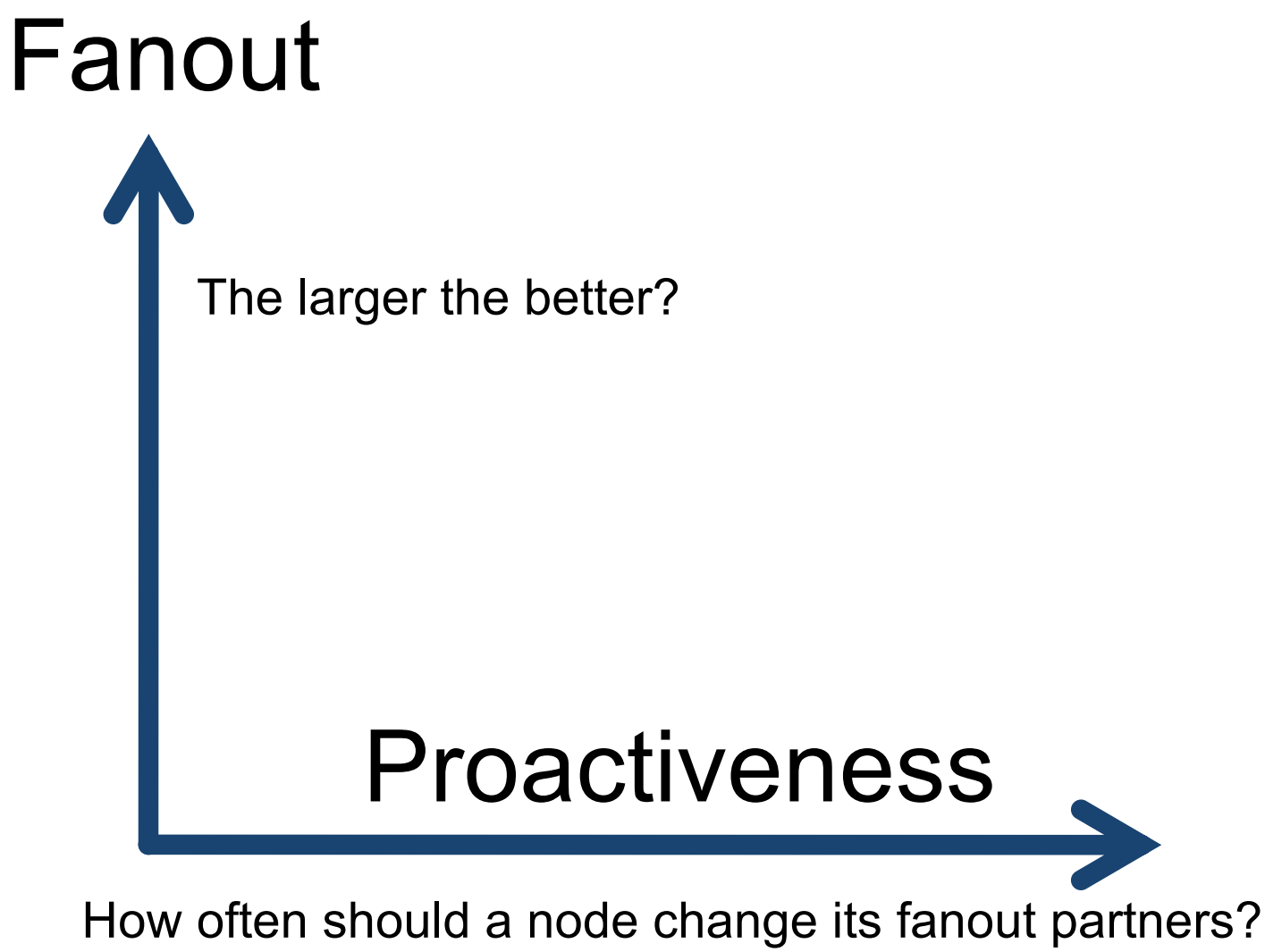
2. Holds as long as the fanout is ln(*n*) + *c* **on average**

Gossip Practice

PlanetLab (230)
700 kbps cap
s = 600 kbps

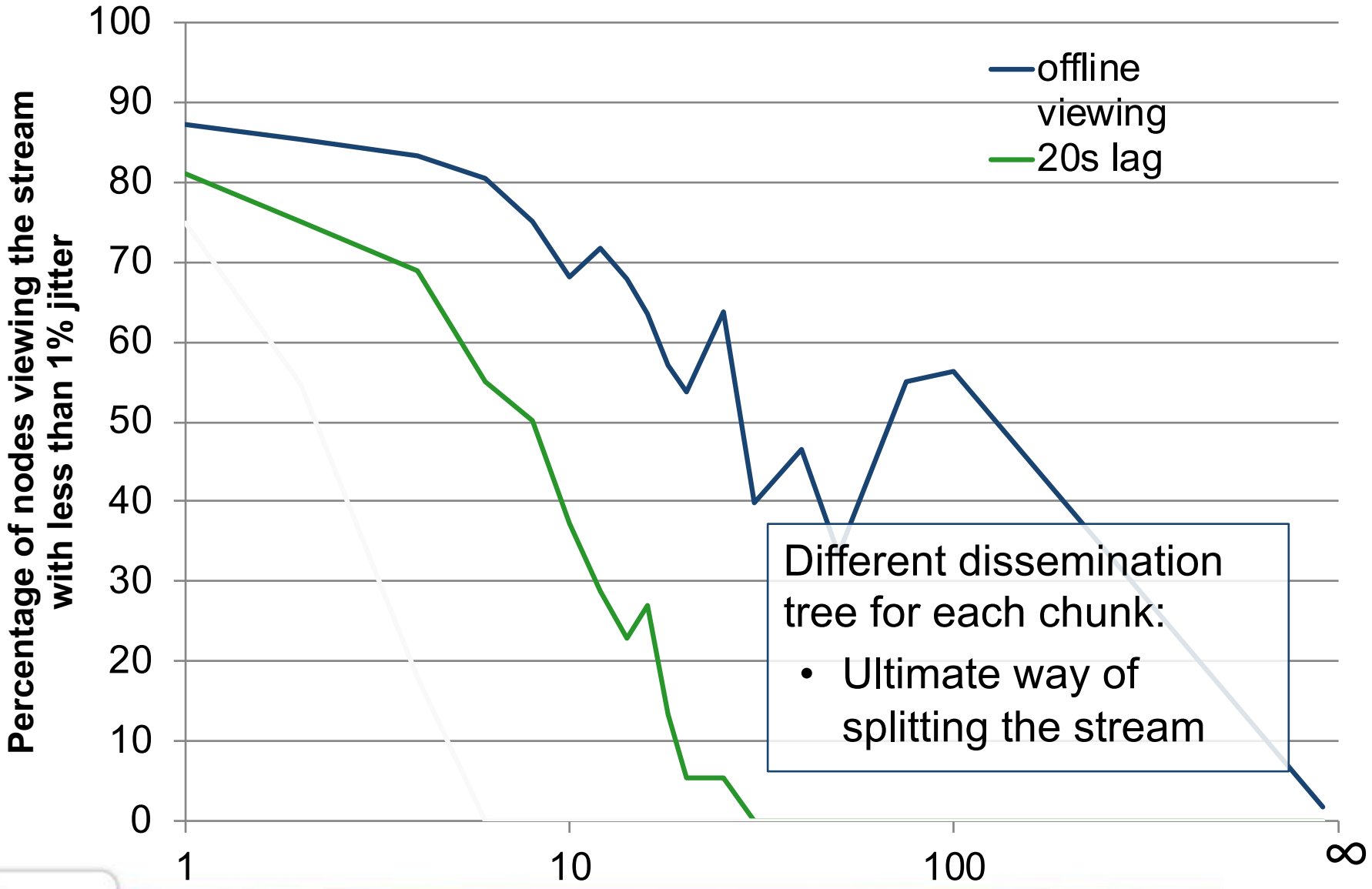


Stretching Gossip

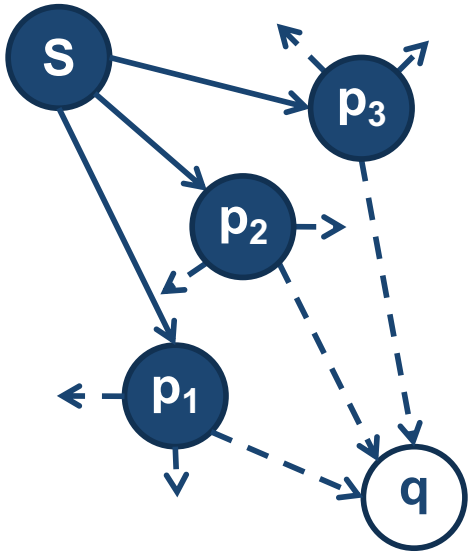


Optimal proactiveness

PlanetLab (230)
700 kbps cap
s = 600 kbps
f = 7



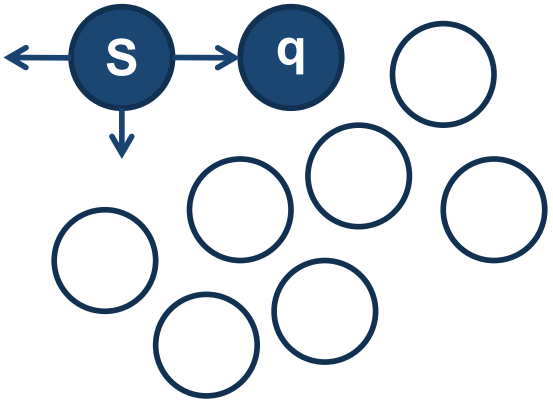
Gossip is load-balancing…



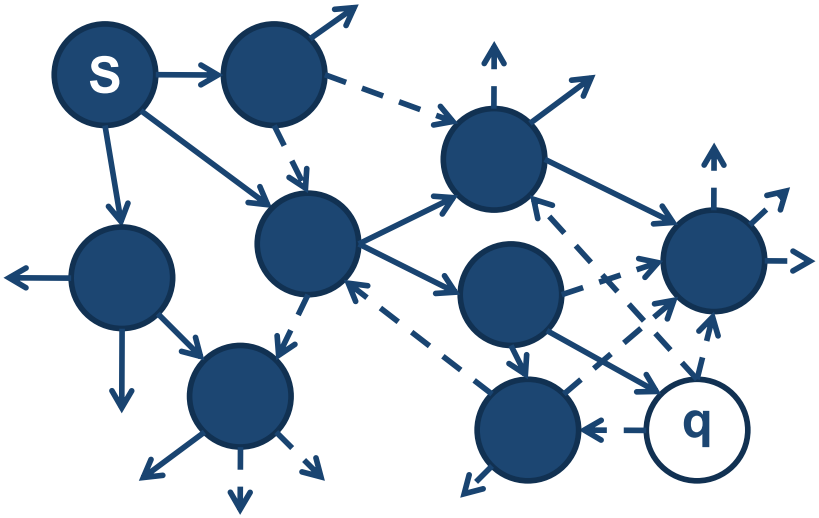
Proposals arrive randomly

- Nodes pull from first proposal

Highly-dynamic



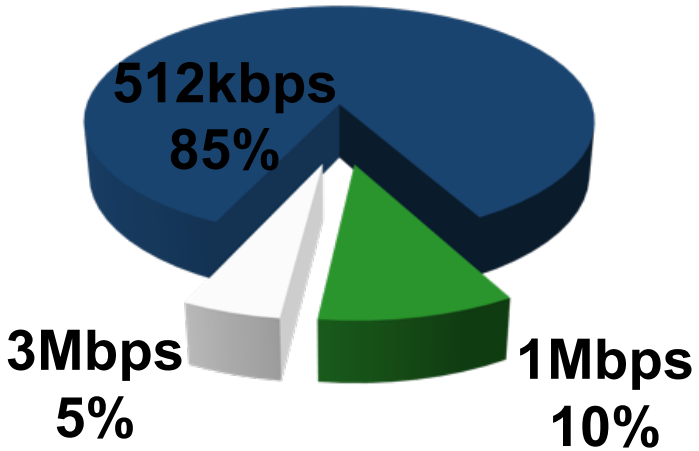
Node q will serve *f* nodes whp



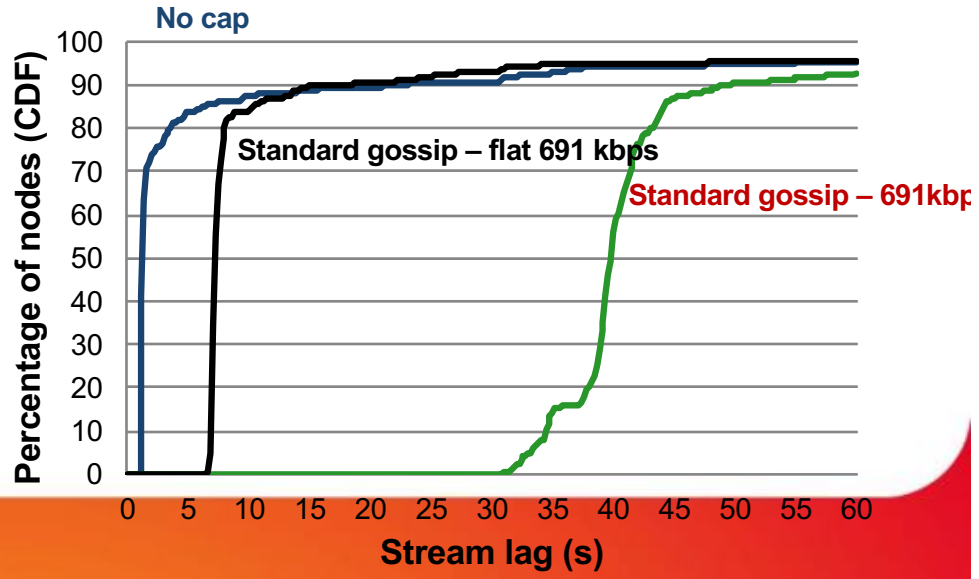
Node q will serve *f* nodes wlp

… but the world is heterogeneous!

3 classes (691kbps avg):



Percentage of nodes receiving at least 99% of the stream



How to cope with heterogeneity?

Goal: contribute according to capability

Propose more = serve more

- Increase fanout…

… and decrease it too!

Such that

- average fanout (f_{avg}) ≥ initial fanout = ln(n) + c

Heterogeneous Gossip - HEAP

q and r with bandwidths $b_q > b_r$

- q should upload b_q / b_r times as much as r

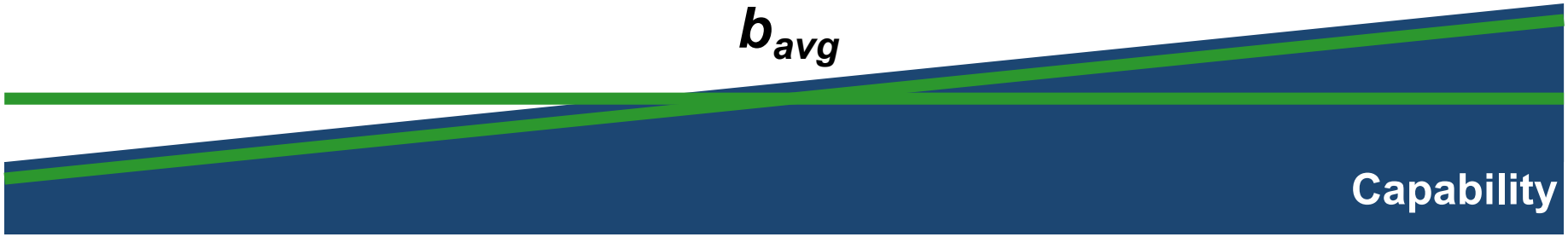
Who should increase/decrease its contribution?

… and by how much?

How to ensure reliability?

- How to keep f_{avg} constant?

HEAP



Total/average contribution is equal in both homogeneous and heterogeneous settings

$$f_q = f_{init} \cdot b_q / \mathbf{b_{avg}}$$

…ensuring the average fanout is constant and equal to f_{init} = ln(***n***) + c

HEAP

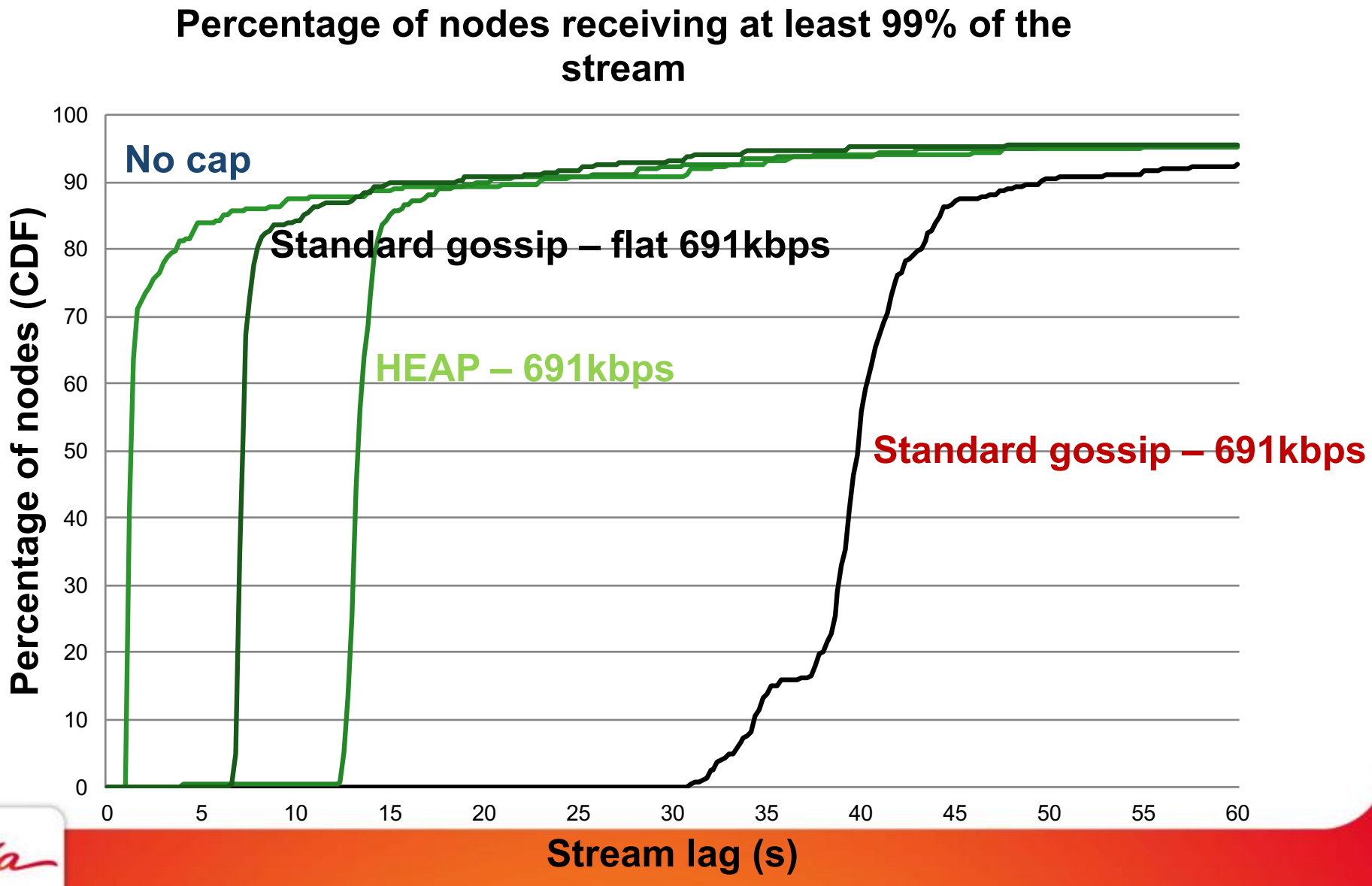
Get b_{avg} with (gossip) aggregation

- Advertize own and freshest received capabilities
- Aggregation follows change in the capabilities

Get n with (gossip) size estimation

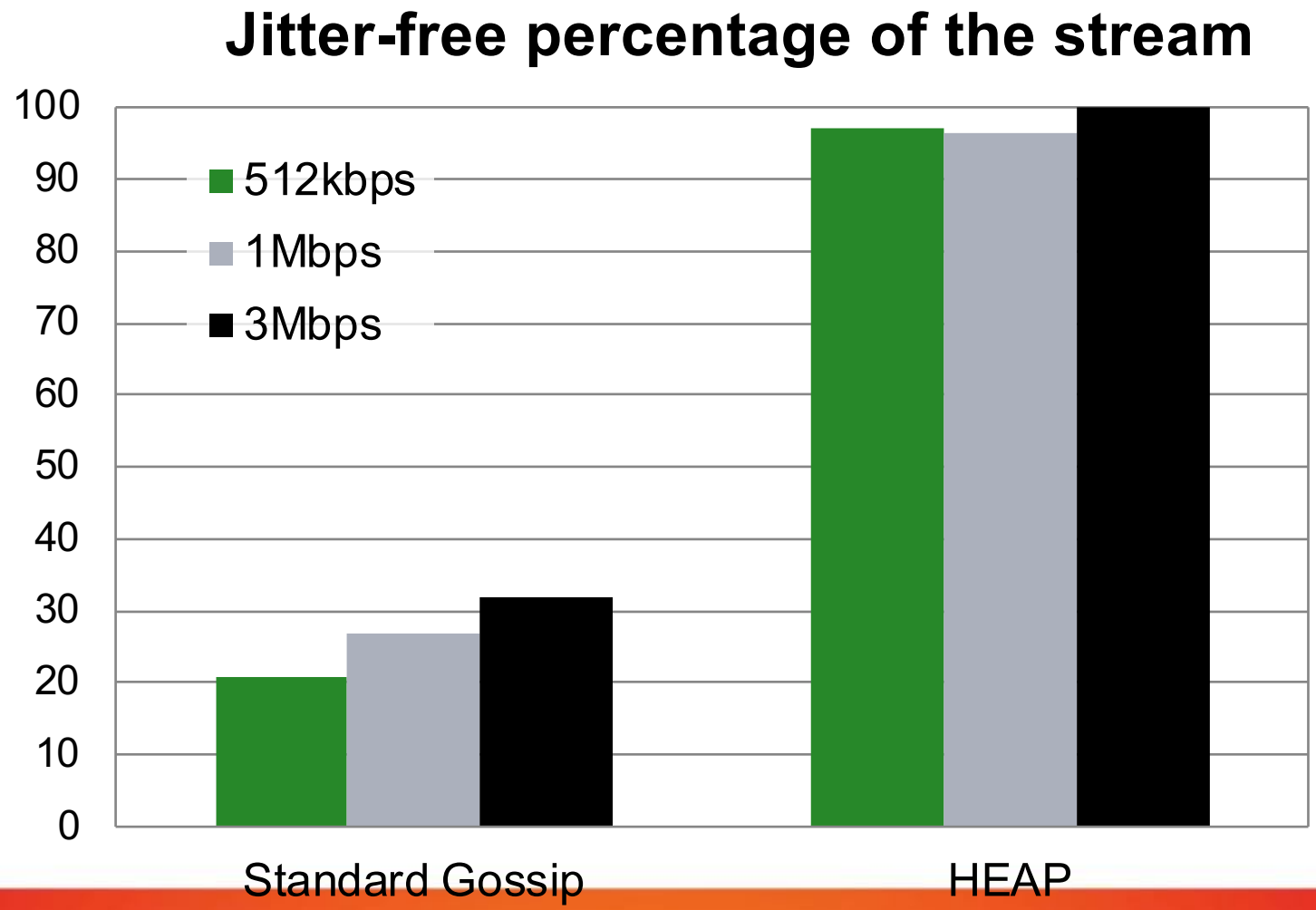
- Estimation follows change in the system
  - Join/leave
  - Crashes
  - …

Stream lag reduction



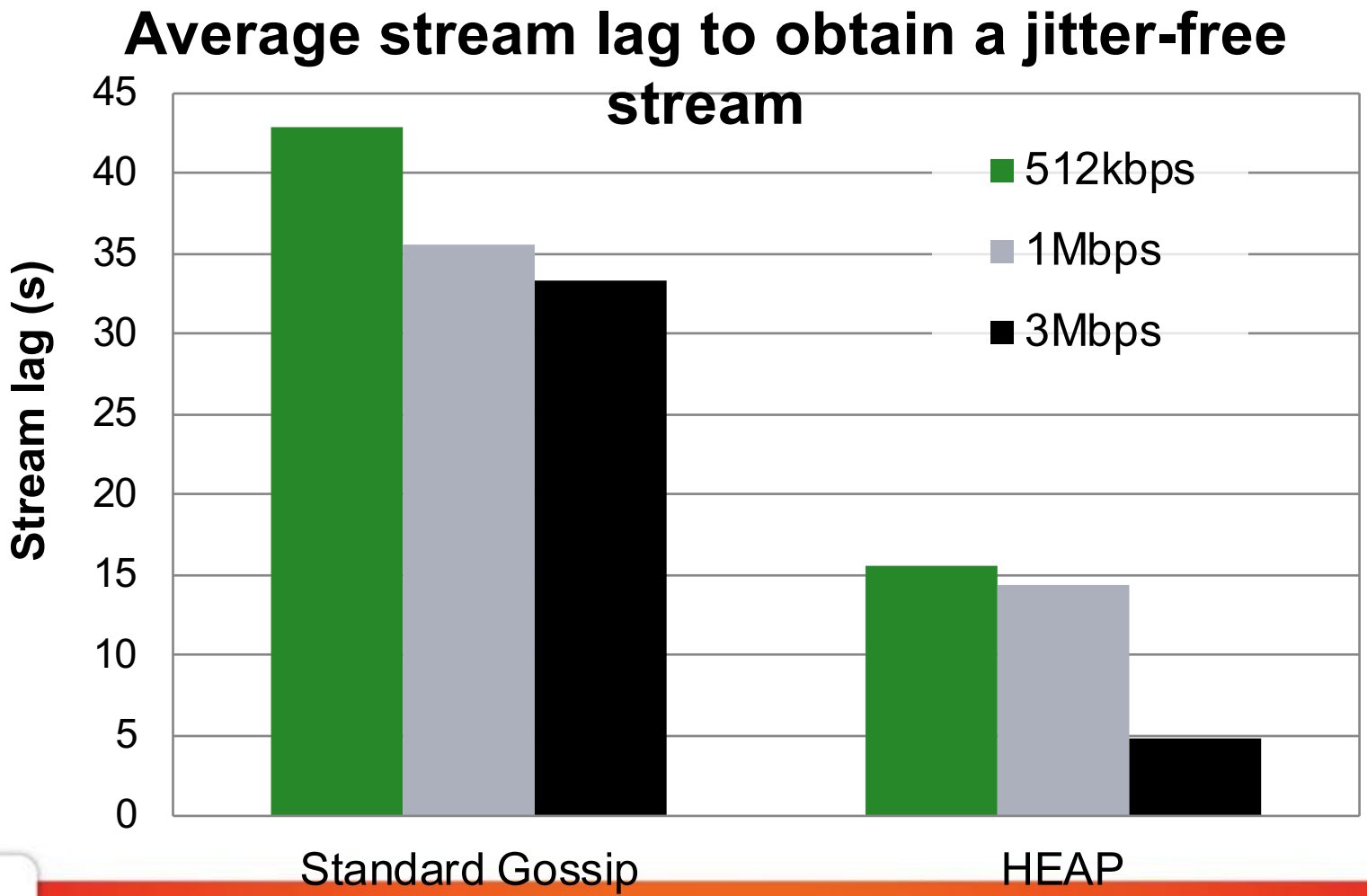
Quality improvement

Stream lag of 10s

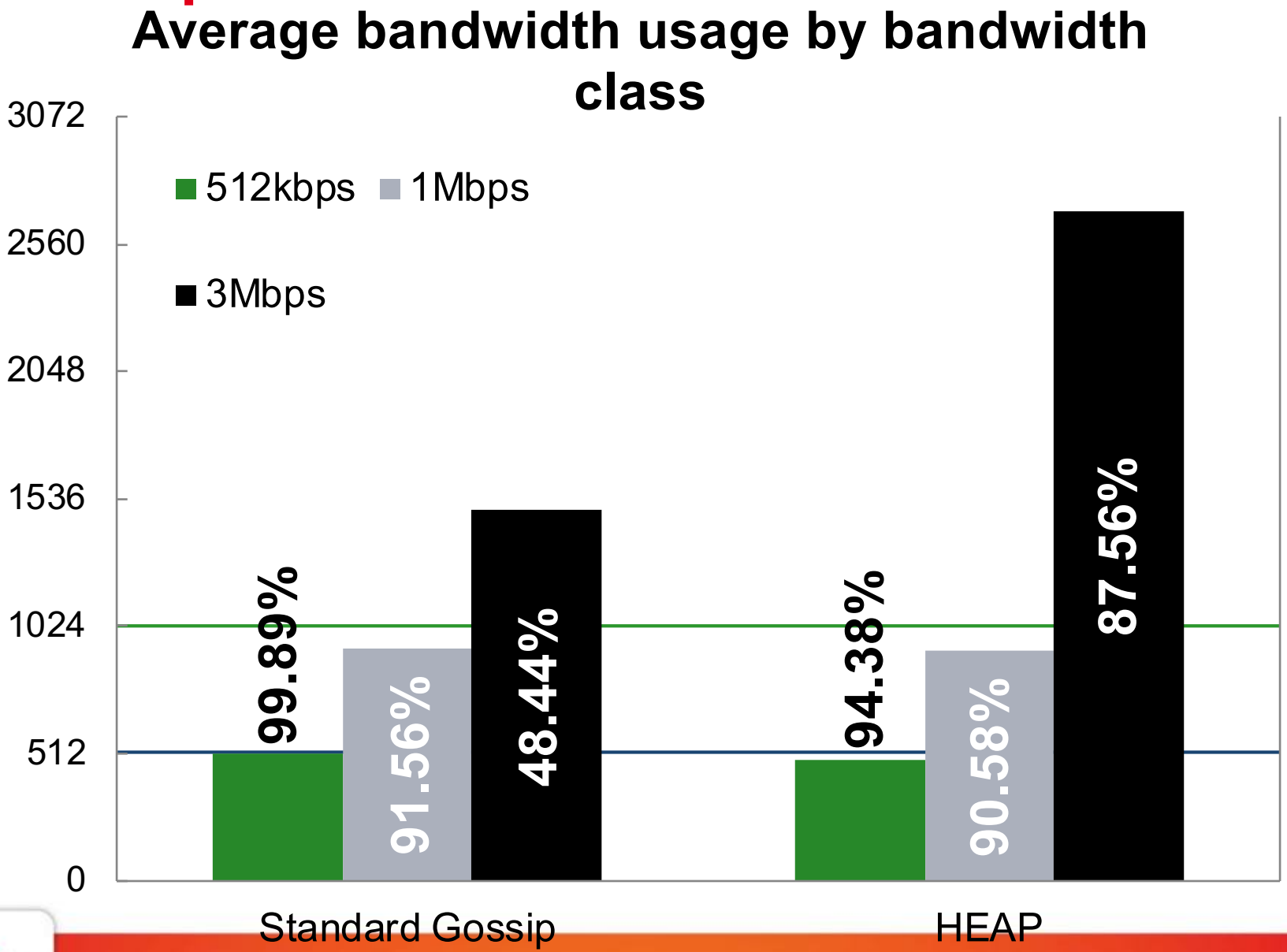


Stream lag

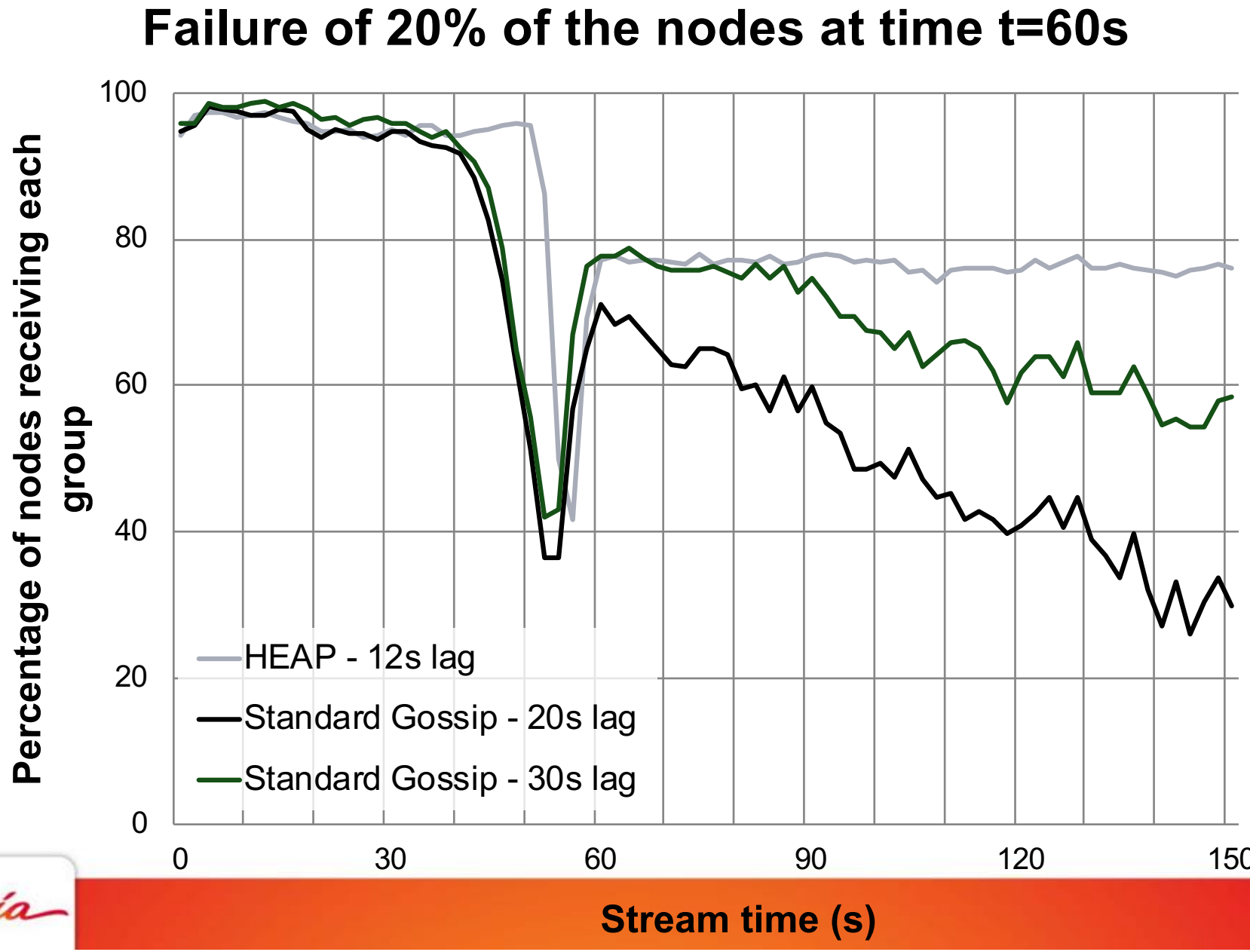
For those who can have a jitter-free stream



Proportional contribution

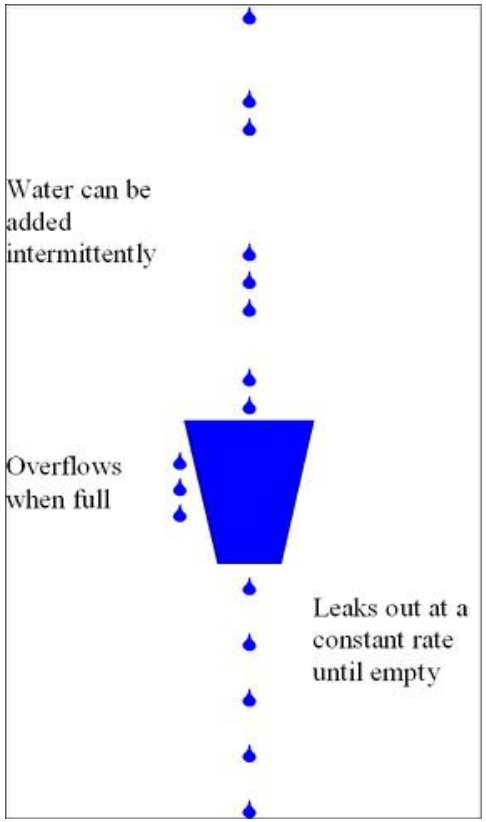


20% nodes crashing

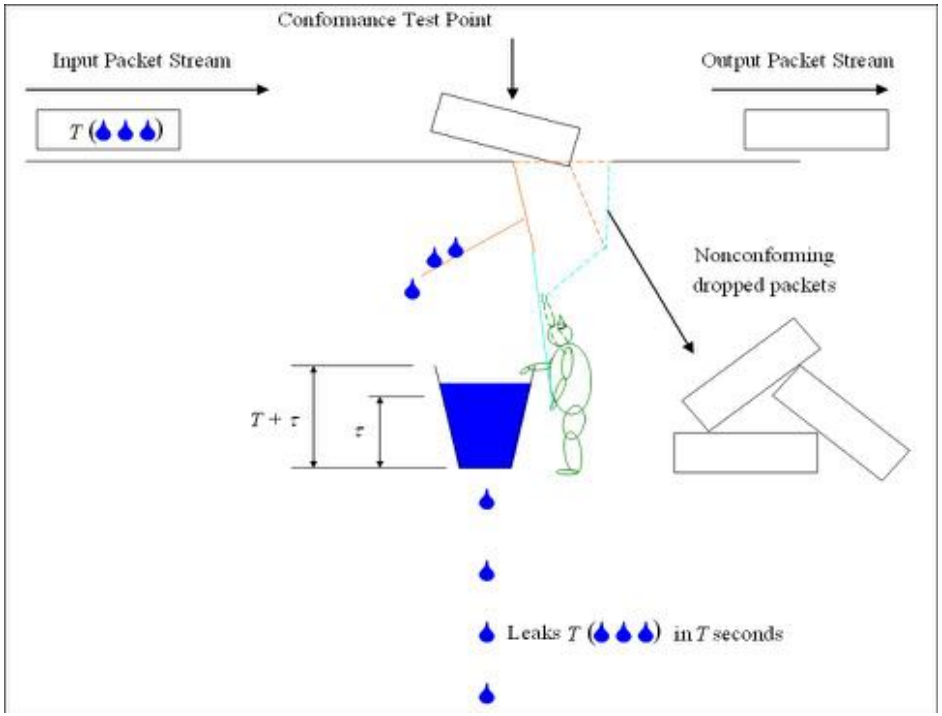


About Bandwidth Limitation

- Leaky Bucket
- Token Bucket

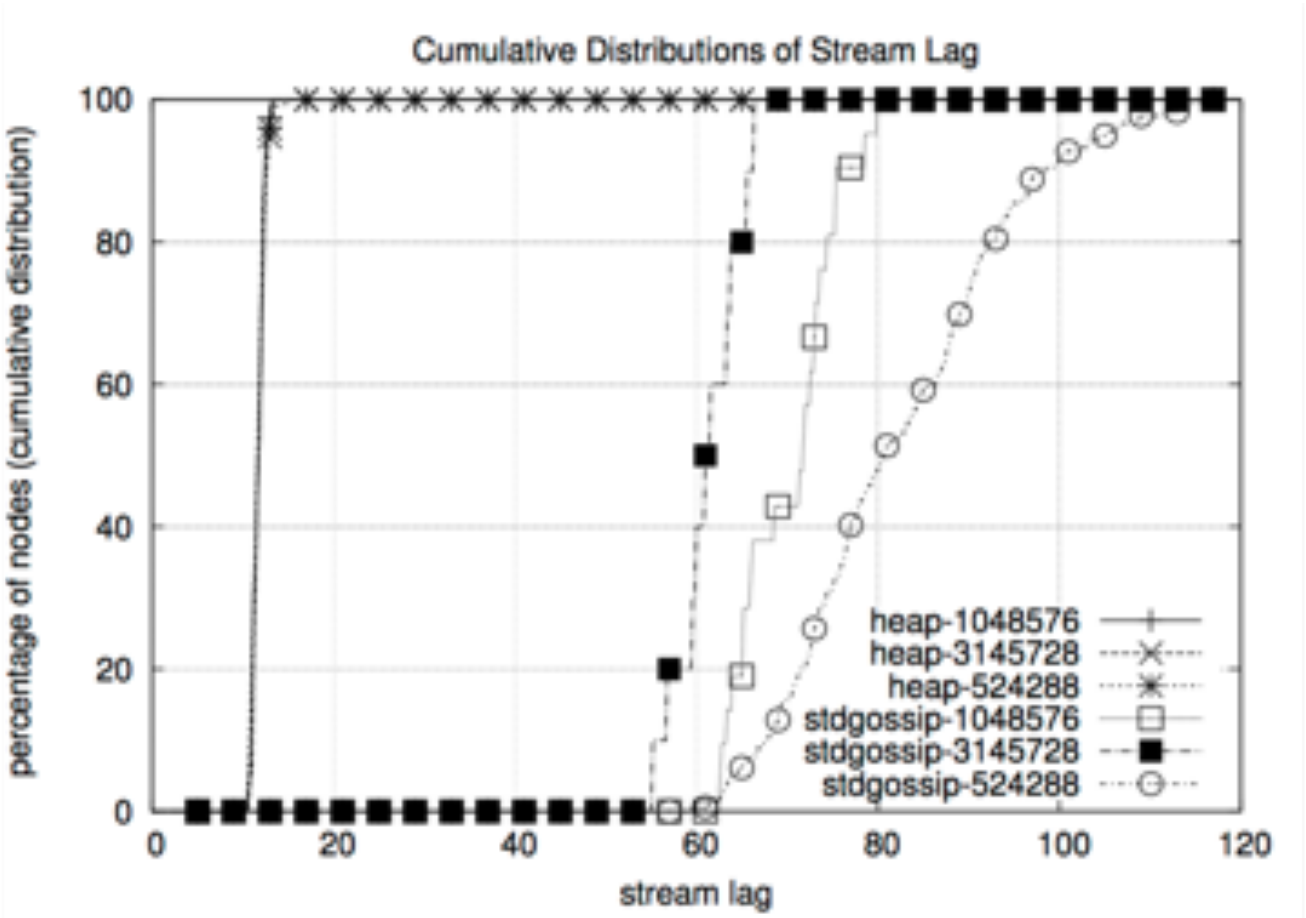


By Graham.Fountain [CC BY 3.0 (http://creativecommons.org/licenses/by/3.0)], via Wikimedia Commons



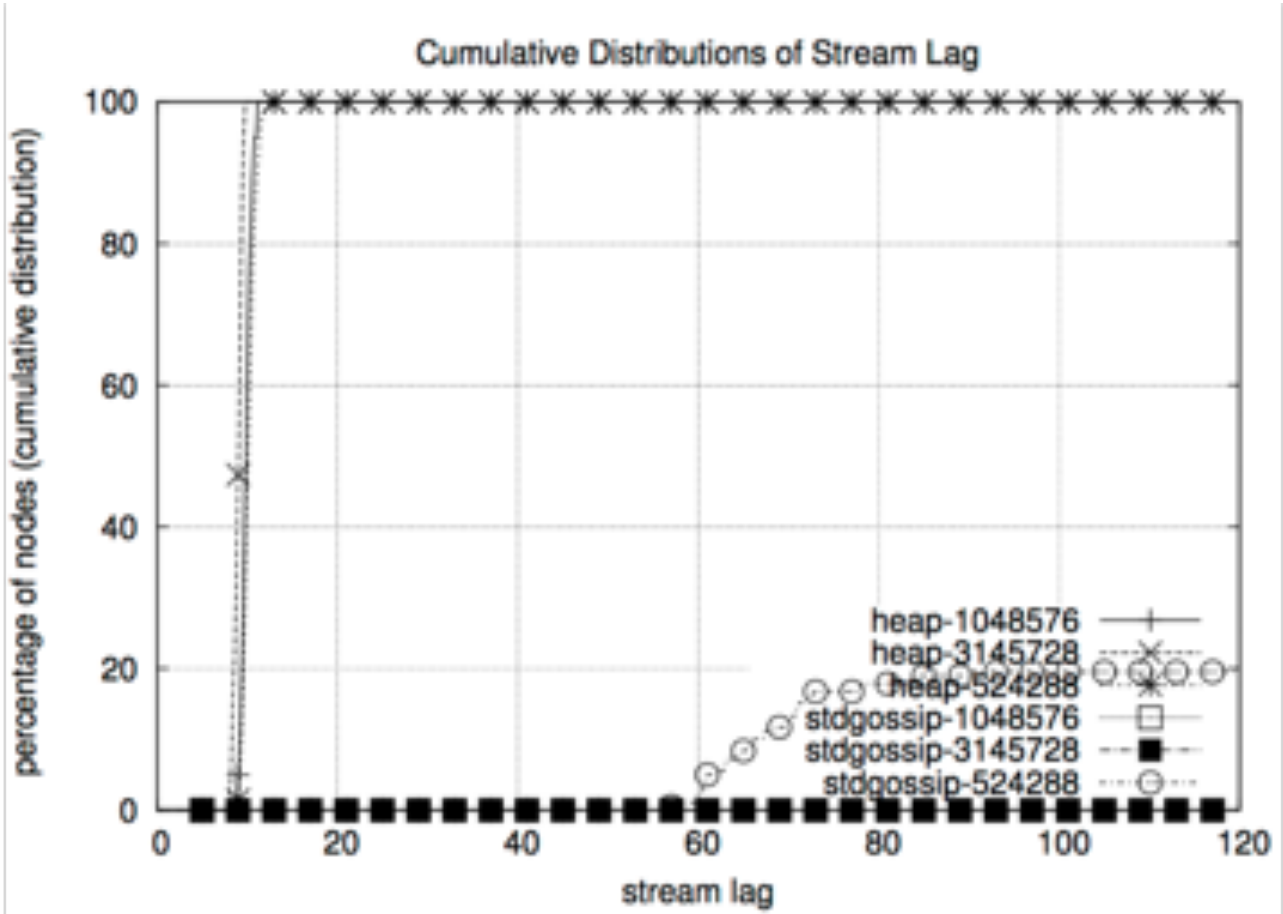
By Graham.Fountain at English Wikipedia, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=35271394

Unbounded Leaky Bucket



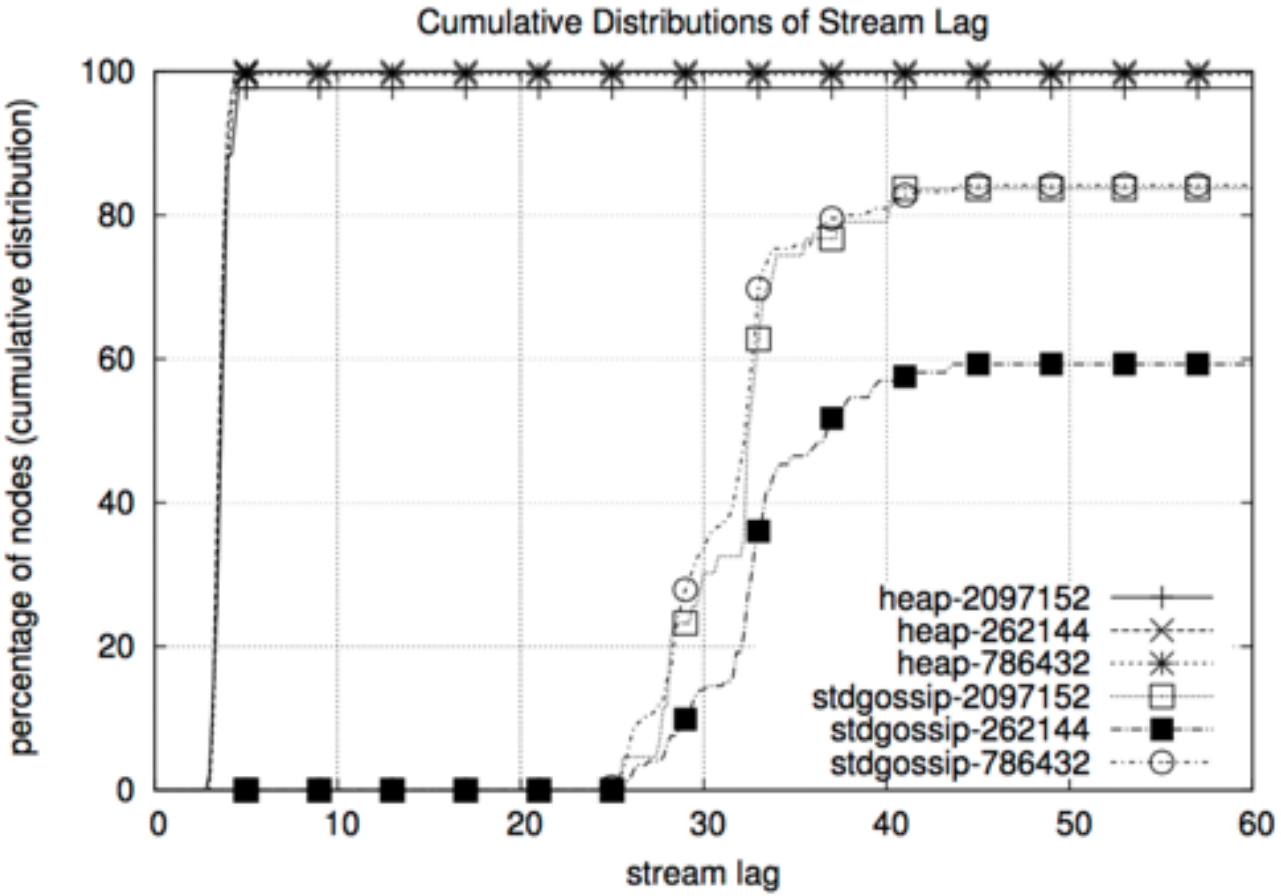
(b) ms691 Infinite Leaky Bucket

Bounded Leaky Bucket



(a) ms691 200KB Leaky Bucket

Token Bucket



(a) ref691

Summary

Multiple Trees

- Effective but hard to split bw perfectly

Mesh

- Easier to build but efficiency – delay tradeoff
- Packet scheduling can improve performance

Gossip

- Improves over mesh by making it dynamic

Pull-Push (we have not seen this in the course, but you can read the following slides)

- Use mesh to identify trees

iGridMedia

Pull-based protocols are effective

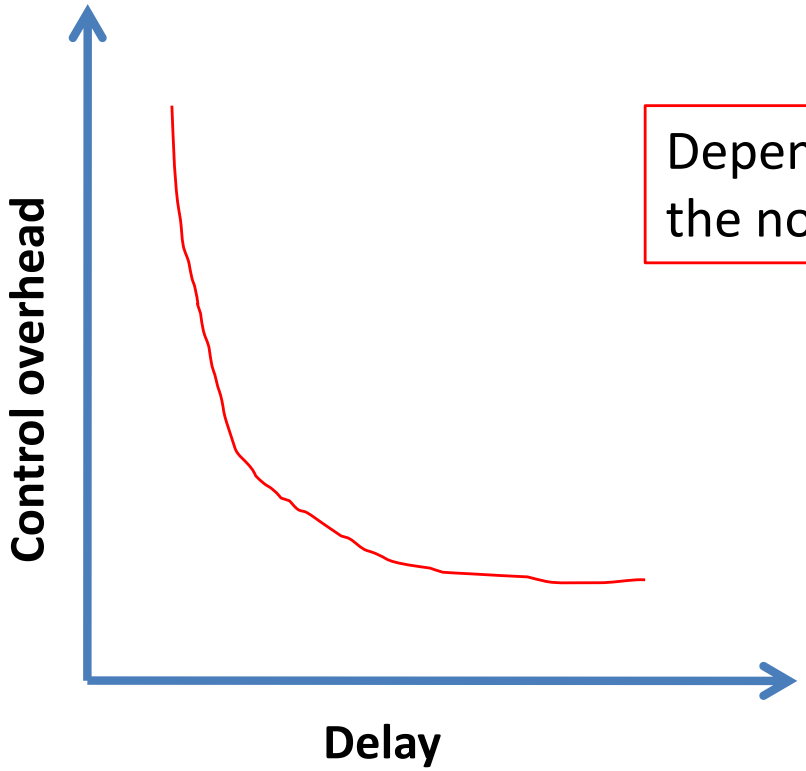
- Select neighbors from unstructured overlay
- Periodically notify neighbors of available packets
- Neighboring nodes request packets

Nearly optimal

- bandwidth utilization
- Throughput

Without intelligent scheduling and bw measurement

Tradeoff



Depends on how frequently the notifications are sent.

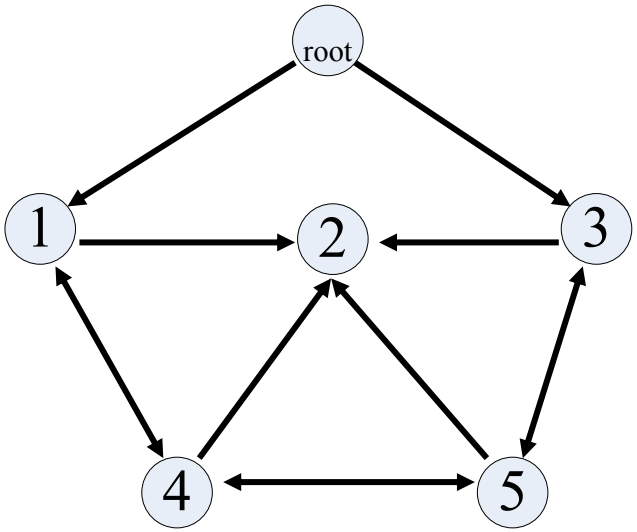
Overlay Construction

Contact rendezvous point

Randomly find set of partners

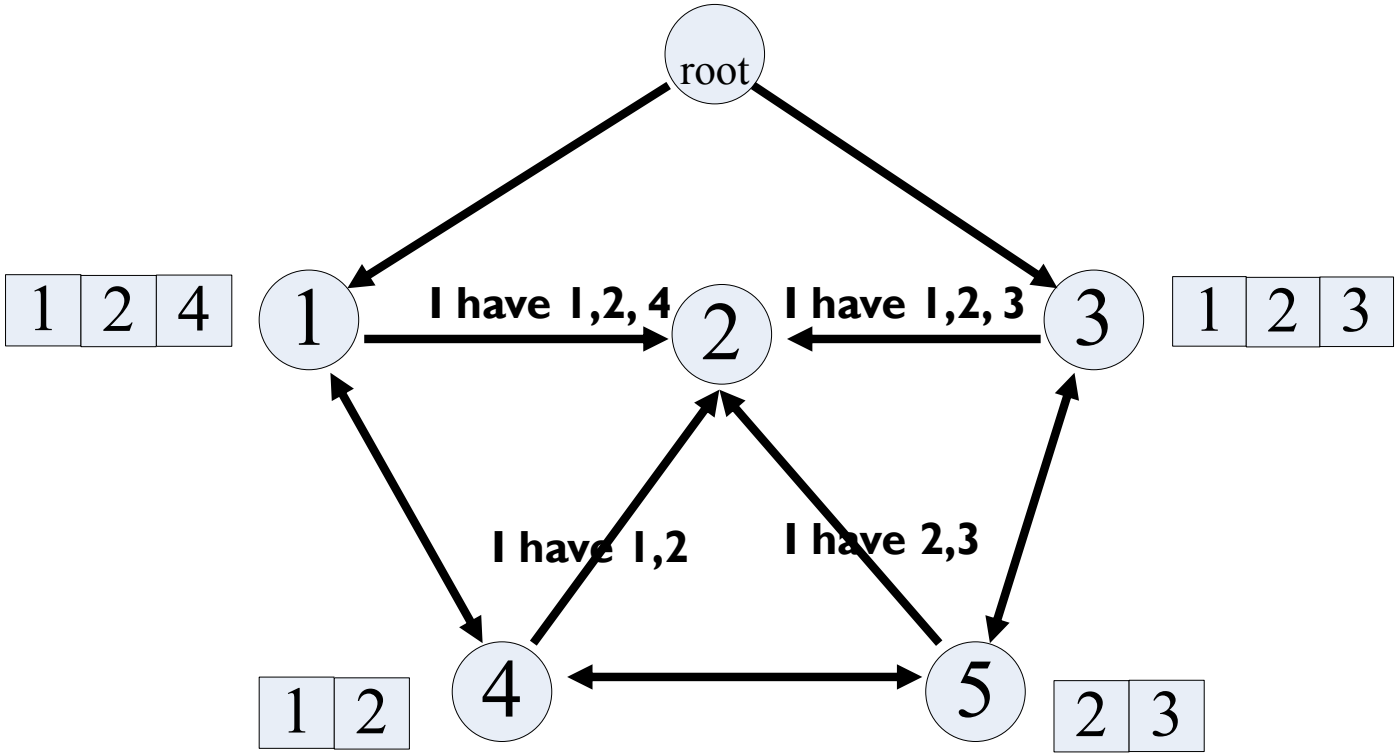
- RPS can be used

Build (static) random graph



Push/Pull Method

- Pull Part



Pull-Push method

Split stream as in Splitstream

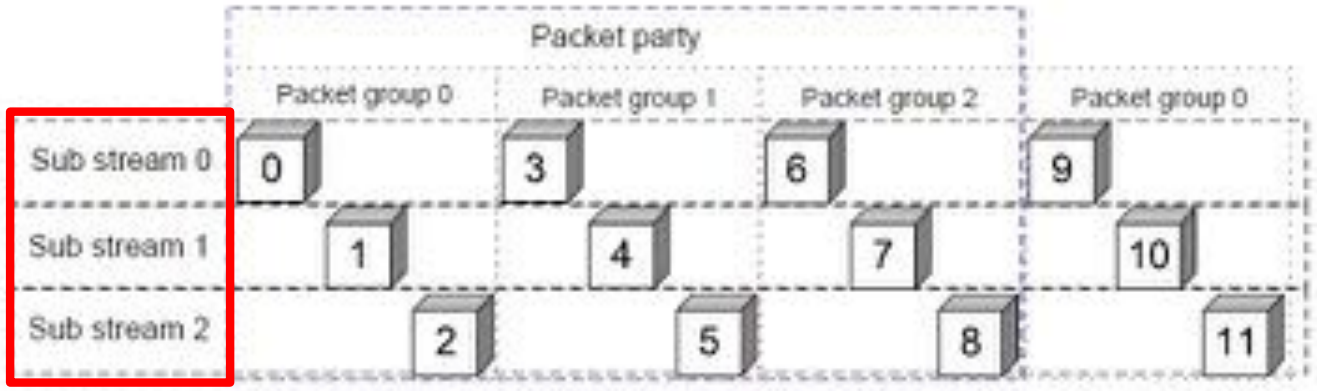


Fig. 18

AN EXAMPLE THAT HAS 3 SUB STREAMS. EVERY PACKET GROUP HAS 3 PACKETS, AND 3 PACKET GROUPS MAKE UP A PACKET PARTY

Pull-Push method

Peers periodically ask for buffer maps

Pull according to buffer maps

Once a node received a packet in group 0 of one packet party

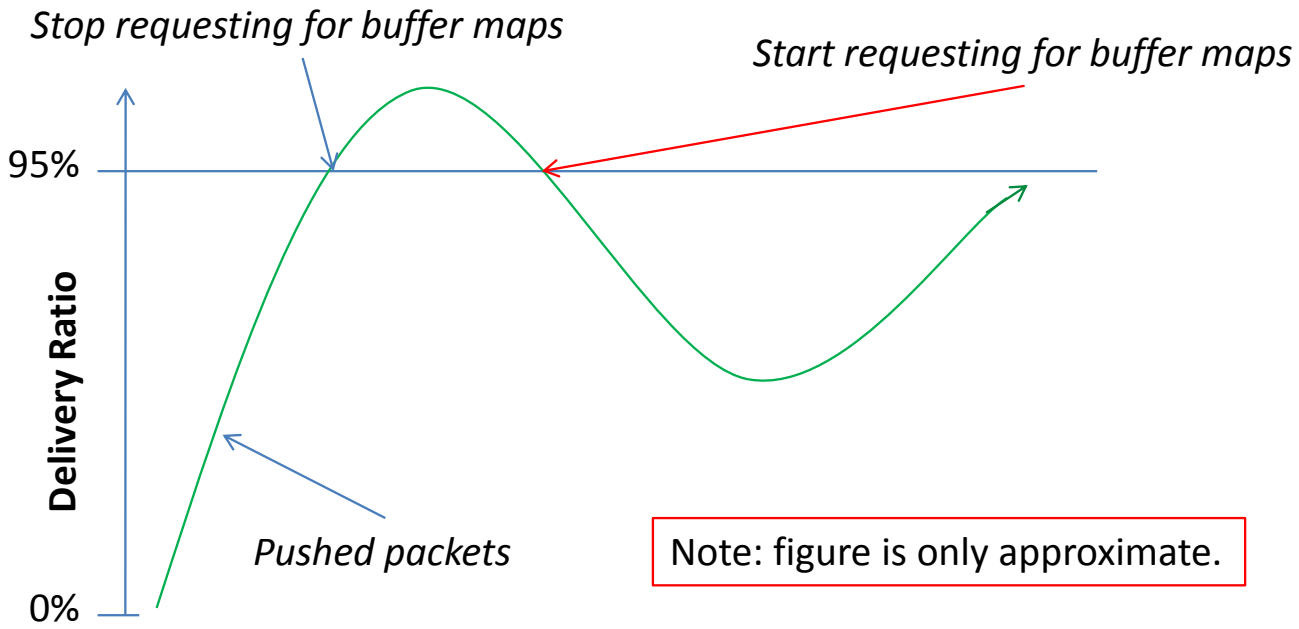
- Send subscription for corresponding substream

Sender will push all packets in the same substream

Pull-Push method

Stop requesting maps when 95% delivery rate with pushed packets

If delivery rate drops, request again



Performance

Considerably

smaller delays

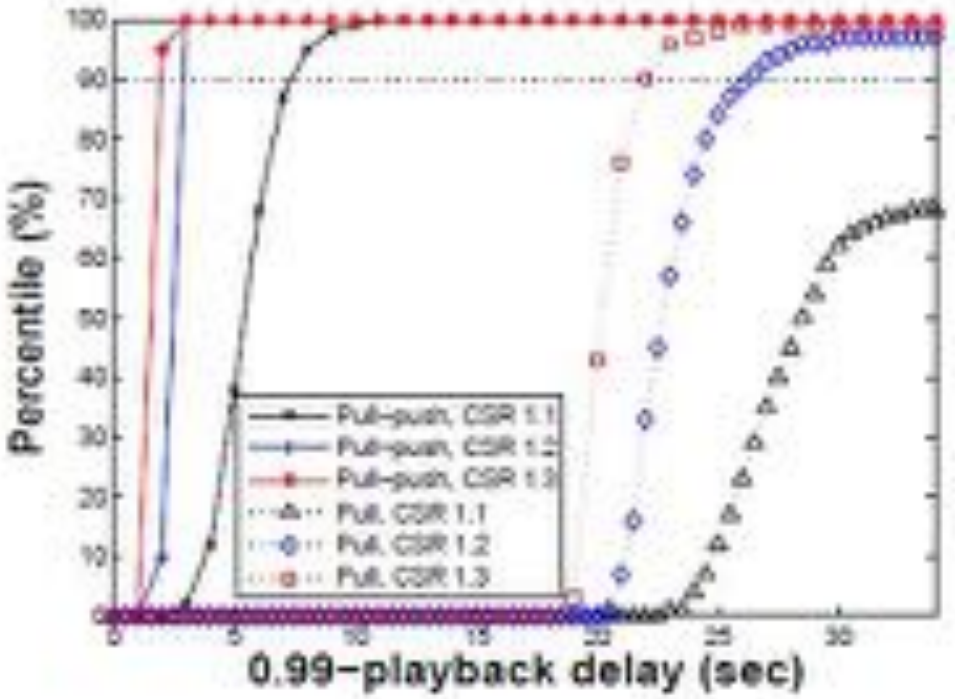


Fig. 20

CDF OF 0.99-PLAYBACK DELAY AMONG ALL PEERS IN PULL-PUSH HYBRID PROTOCOL AND PULL-BASED PROTOCOL

Overhead

Much smaller than

for pull-only

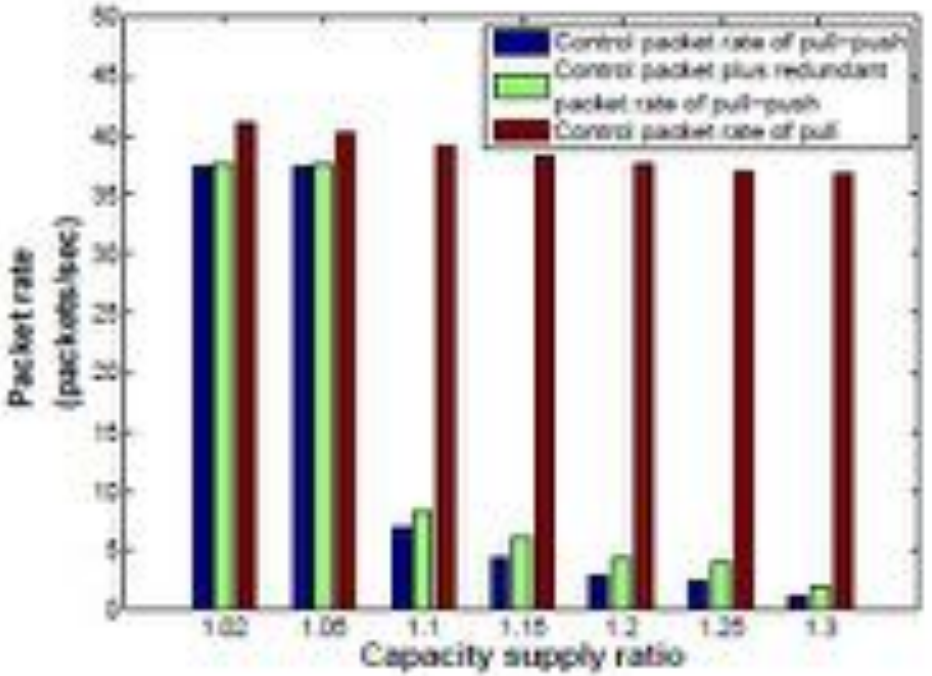
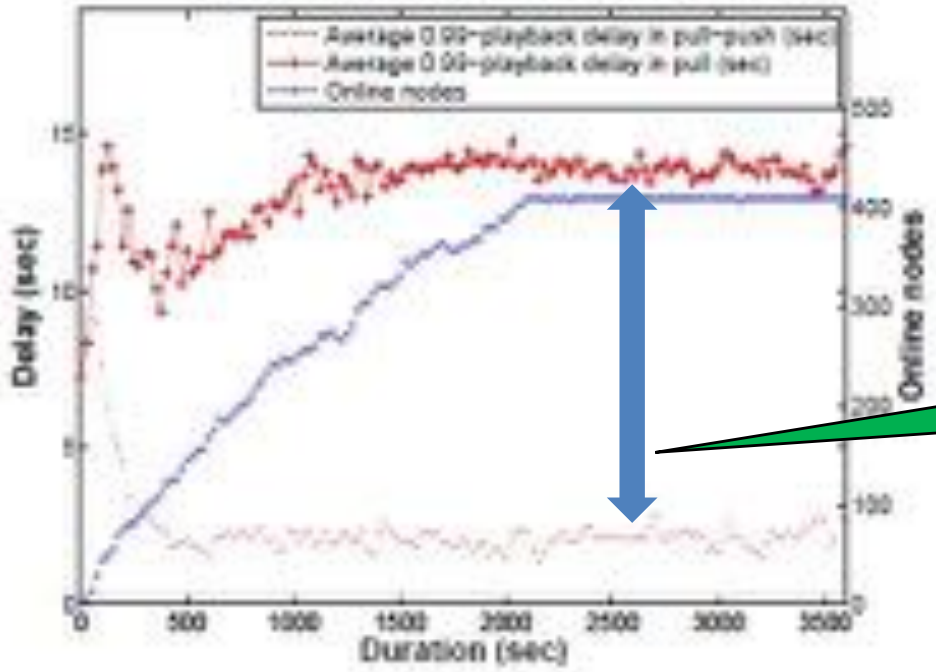


Fig. 21

AVERAGE CONTROL PACKET RATE COMPARISON BETWEEN PULL-BASED AND PULL-PUSH HYBRID PROTOCOL

PlanetLab



Significant delay reduction!

Fig. 28

PLANETLAB EXPERIMENT WITH 409 NODES. COMPARISON OF AVERAGE PLAYBACK DELAY AND PACKET ARRIVAL DELAY

Summary

Multiple Trees

- Effective but hard to split bw perfectly

Mesh

- Easier to build but efficiency – delay tradeoff
- Packet scheduling can improve performance

Gossip

- Improves over mesh by making it dynamic

Pull-Push

- Use mesh to identify trees

References

M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: High-bandwidth multicast in a cooperative environment", SOSP'03, Lake Bolton, New York, October, 2003.

M. Castro, M. B. Jones, A-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang and A. Wolman, "*An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer overlays*", Infocom 2003, San Francisco, CA, April, 2003.

Zhang, X.Z.X. et al., 2005. CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, 3(c), p.2102-2111. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1498486.

Picconi, F. & Massoulie, L., 2008. Is There a Future for Mesh-Based live Video Streaming? *2008 Eighth International Conference on PeertoPeer Computing*, p.289-298. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4627291.

Zhang, M.Z.M. et al., 2008. iGridMedia: Providing Delay-Guaranteed Peer-to-Peer Live Streaming Service on Internet. *IEEE GLOBECOM 2008 2008 IEEE Global Telecommunications Conference*, p.1–5. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4698112.

Meng Zhang; Qian Zhang; Lifeng Sun; Shiqiang Yang; , "Understanding the Power of Pull-Based Streaming Protocol: Can We Do Better?," *Selected Areas in Communications, IEEE Journal on* , vol.25, no.9, pp.1678-1694, December 2007

Davide Frey; Rachid Guerraoui; Anne-Marie Kermarrec; Maxime Monod. **Boosting Gossip for Live Streaming.** P2P 2010, Aug 2010, Delft, Netherlands.

Davide Frey; Rachid Guerraoui; Anne-Marie Kermarrec; Maxime Monod; Koldehofe Boris; Mogensen Martin; Vivien Quéma. **Heterogeneous Gossip.** Middleware 2009, Dec 2009, Urbana-Champaign, IL, United States.

Davide Frey; Rachid Guerraoui; Anne-Marie Kermarrec; Maxime Monod; Vivien Quéma. **Stretching Gossip with Live Streaming.** DSN 2009, Jun 2009, Estoril, Portugal.