DNA
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

broadcasting

A source produces **multimedia content**

n viewers (*n* large)

VS

IP TV, Web TV, P2P TV, …

BBC iPlayer

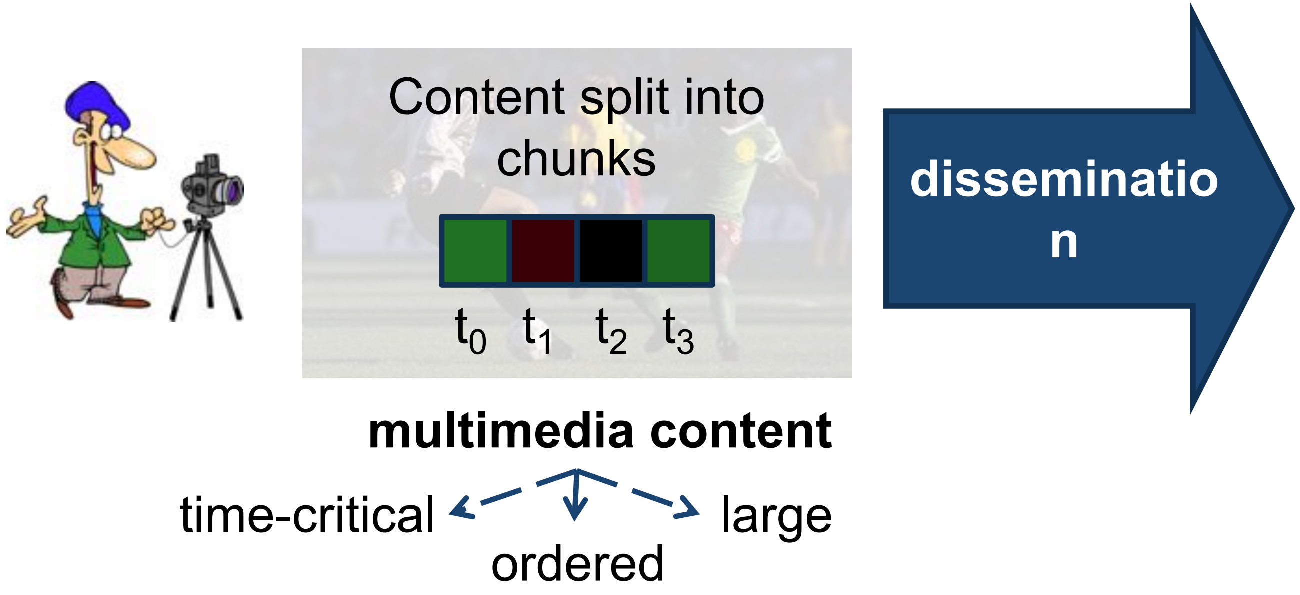
192K requests/day

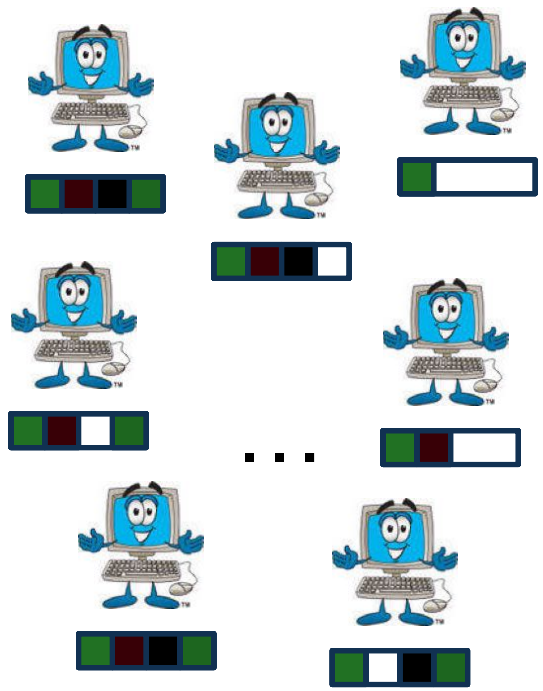
78K users/day

244K simultaneous users (incl. VoD)

BBC iStats (April 2010)

Streaming Basics





n viewers (*n* large)

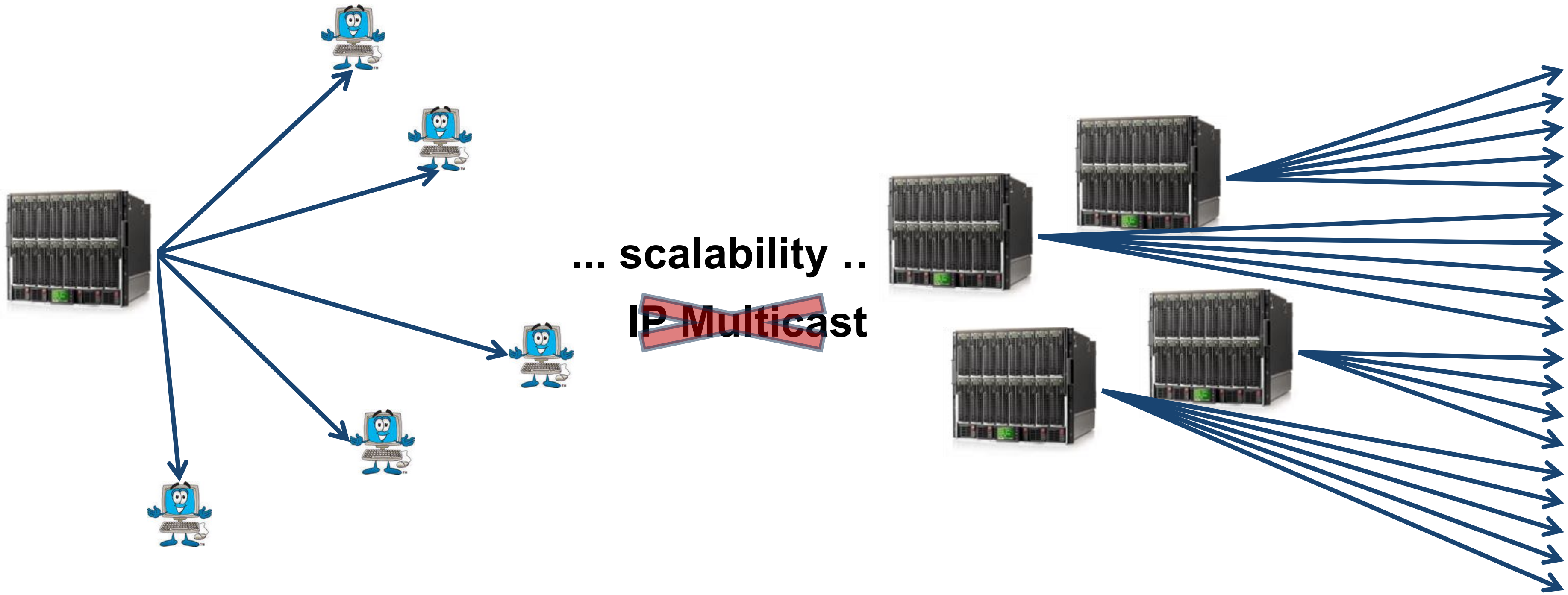
Stream rate ***s*** [kbps]

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

Demand = Supply

Intuitive solution

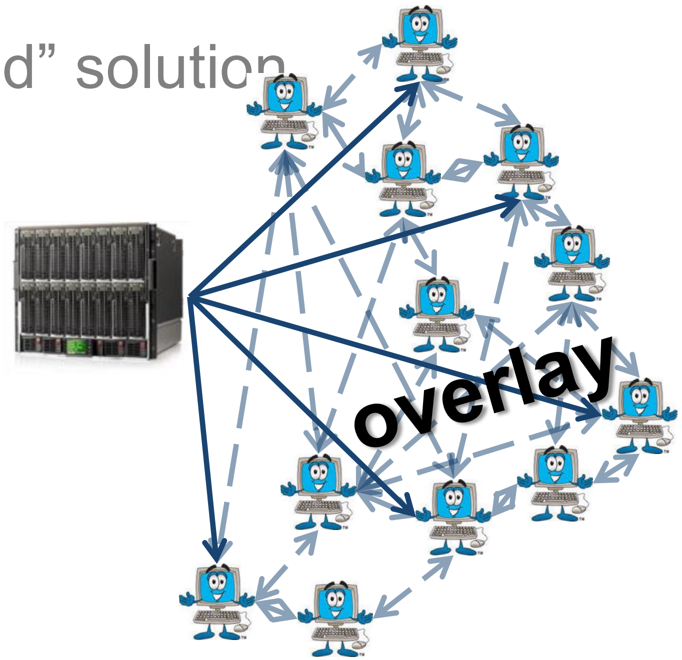
- "Centralized" solution



Participants are pure consumer

Let's be smarter

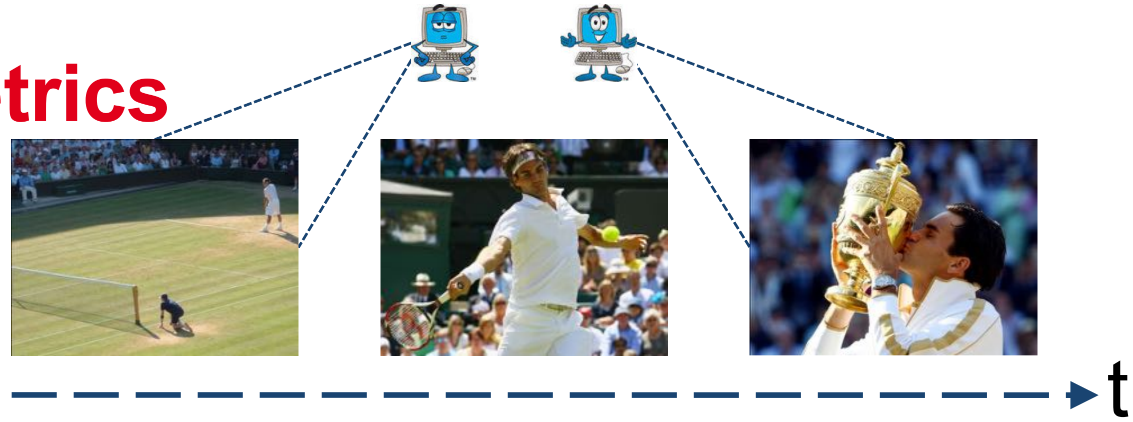
"Decentralized" solution



Participants **collaborate**

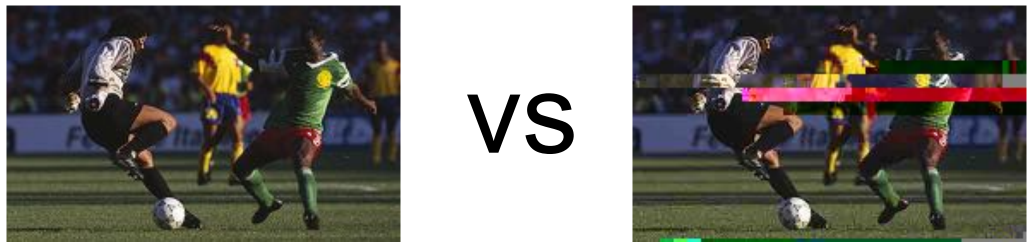
…most of them!

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

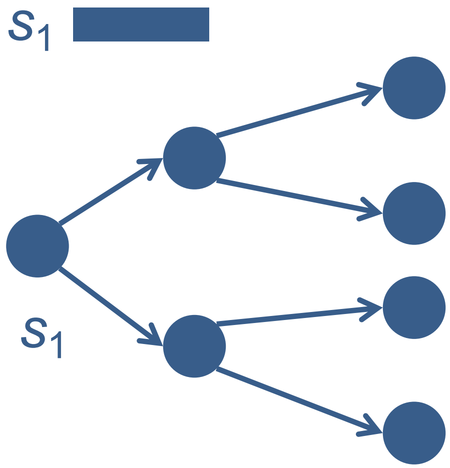


Stream quality

- 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

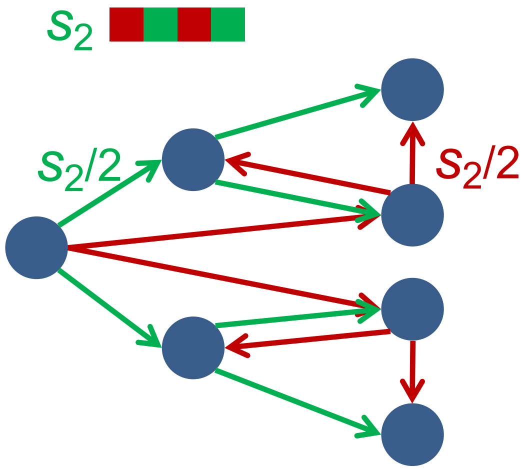
Tree-based ALM

Streaming Approaches



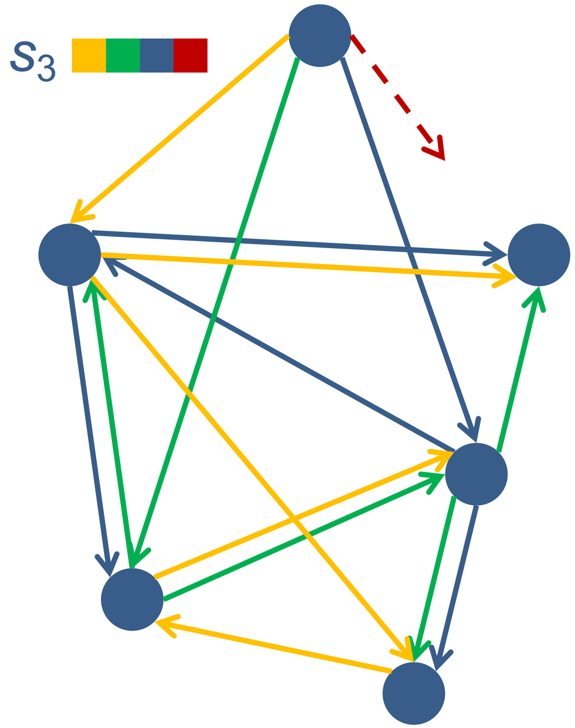
Single tree

- s_1 is constrained by design
- Disconnection
- Build/maintain tree



Multiple trees

- Upload of nodes: multiple of s_2/z
- Partial disconnection
- Build/maintain z trees



Mesh/Gossip

- s_3 optimal
- Connected is not enough
- Peer selection, Packet scheduling

Addressing the Limitations of Trees

Some peers do not forward

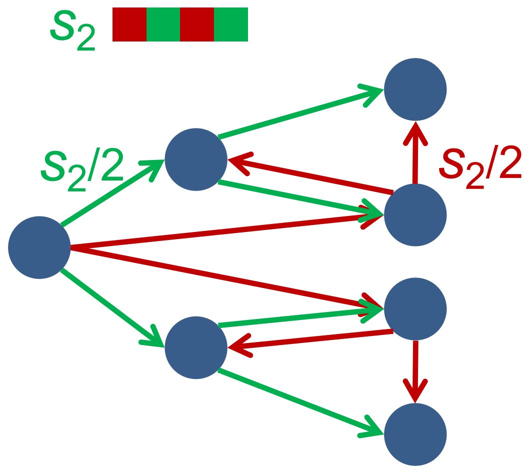
Multiple Trees

Mesh/Gossip

SplitStream approach

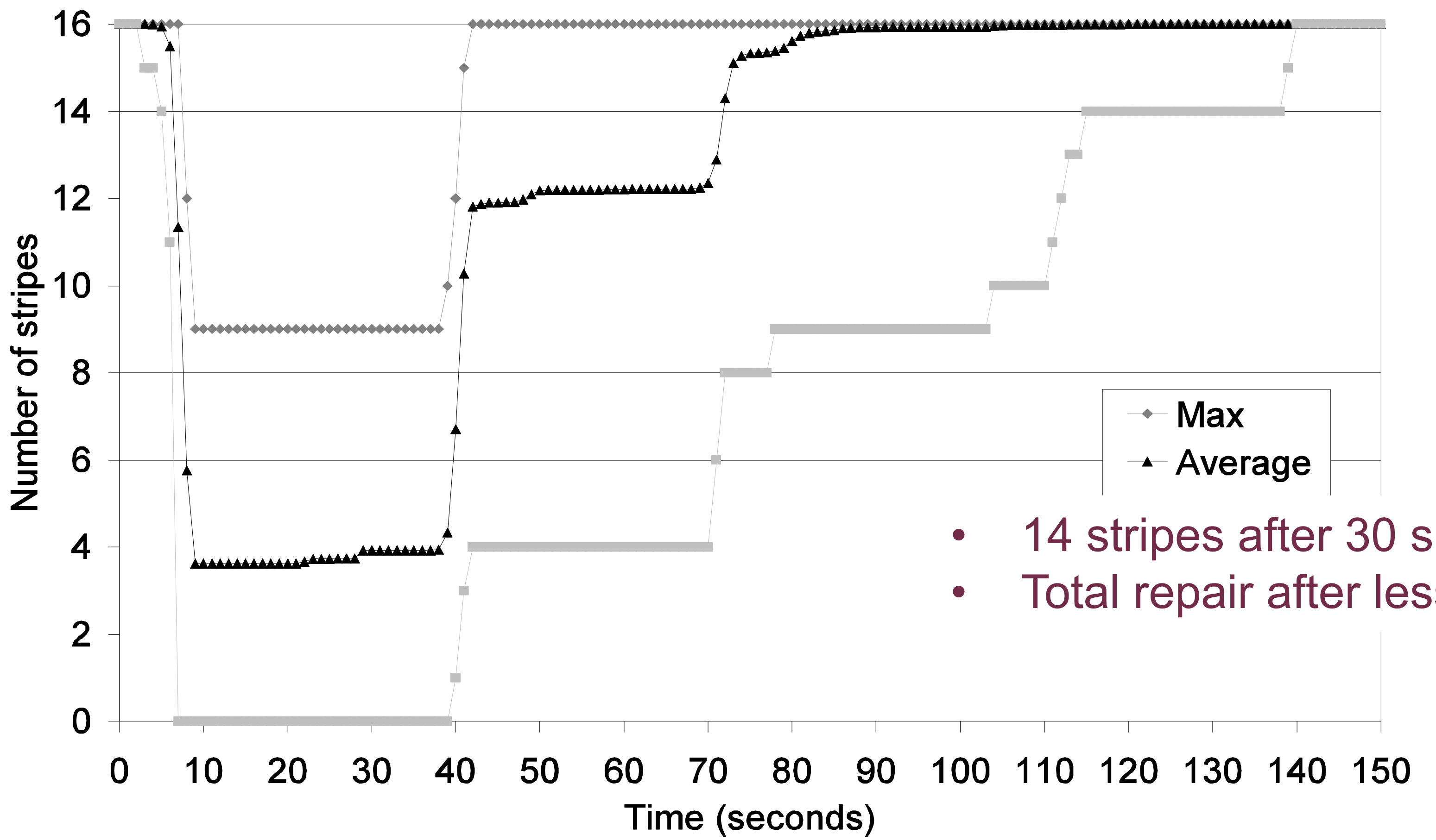
Content divided in *stripes*
Each stripe is distributed on an independent tree

- 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



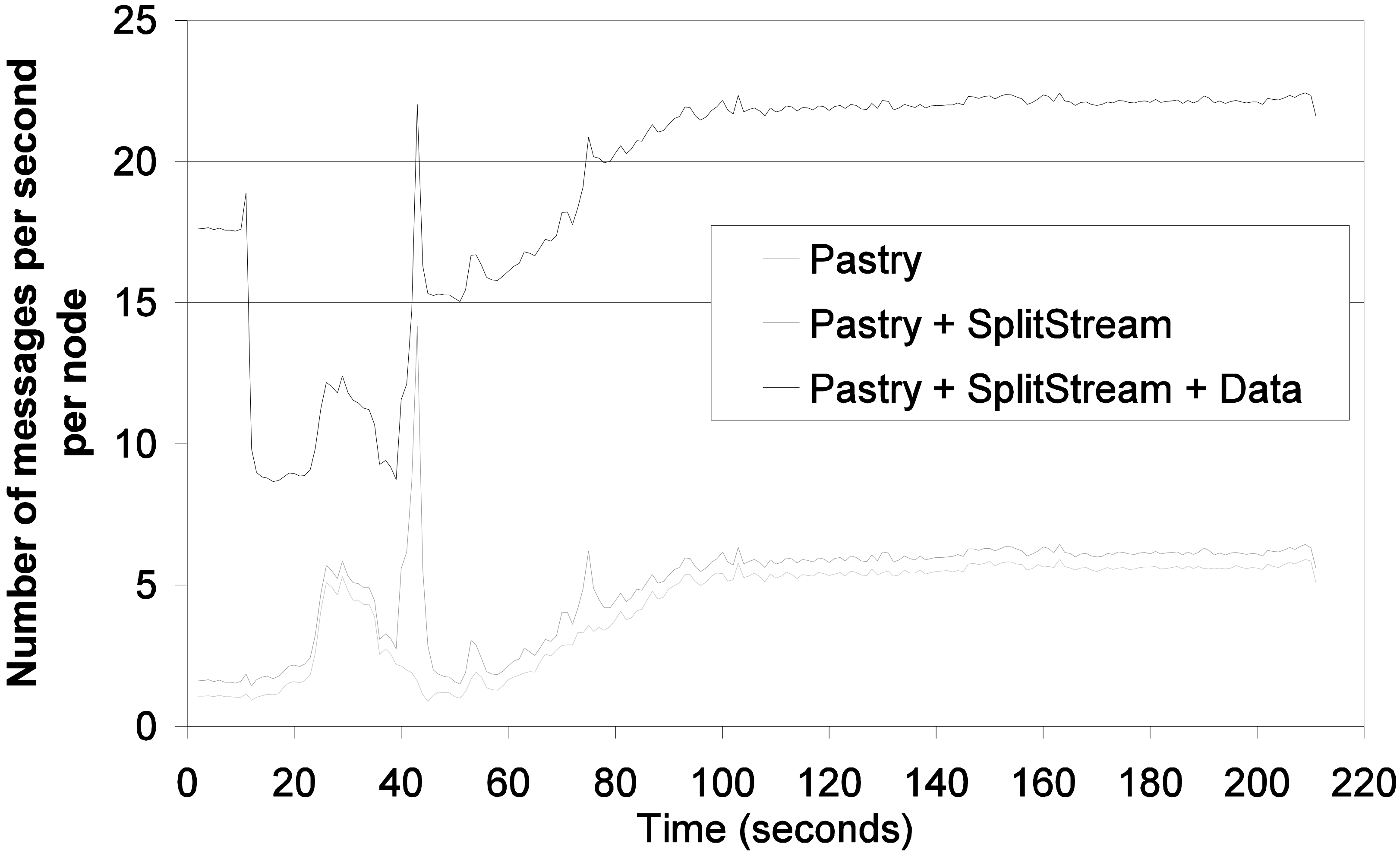
[SOSP 2003 « *SplitStream: High-Bandwidth Multicast in Cooperative Environment* »]

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



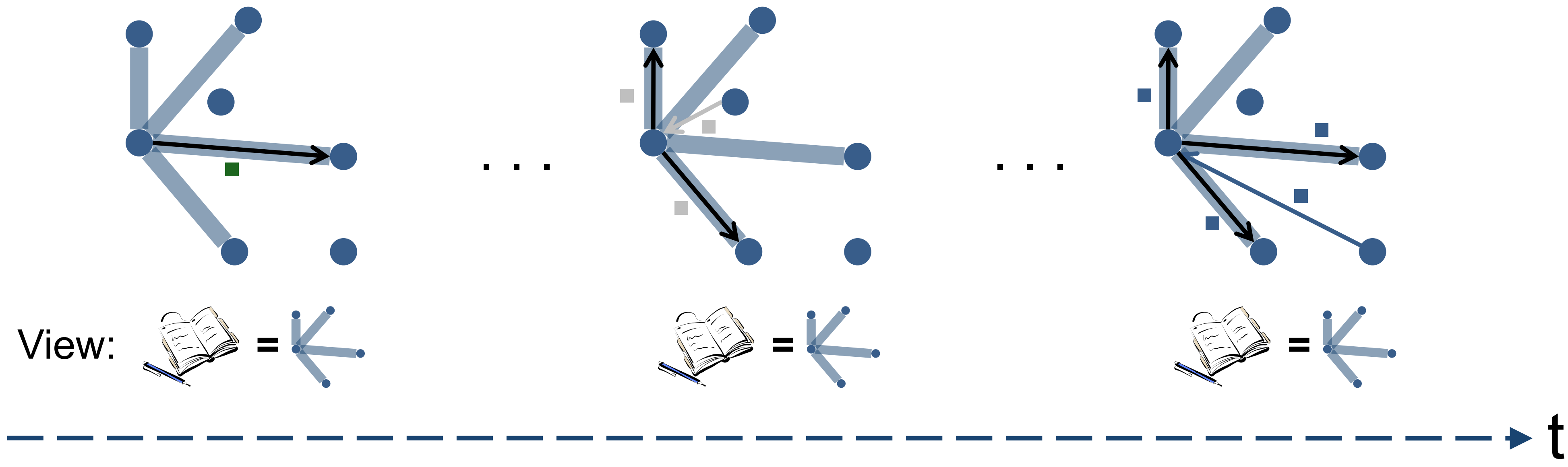
Addressing the Limitations of Trees

Some peers do not forward

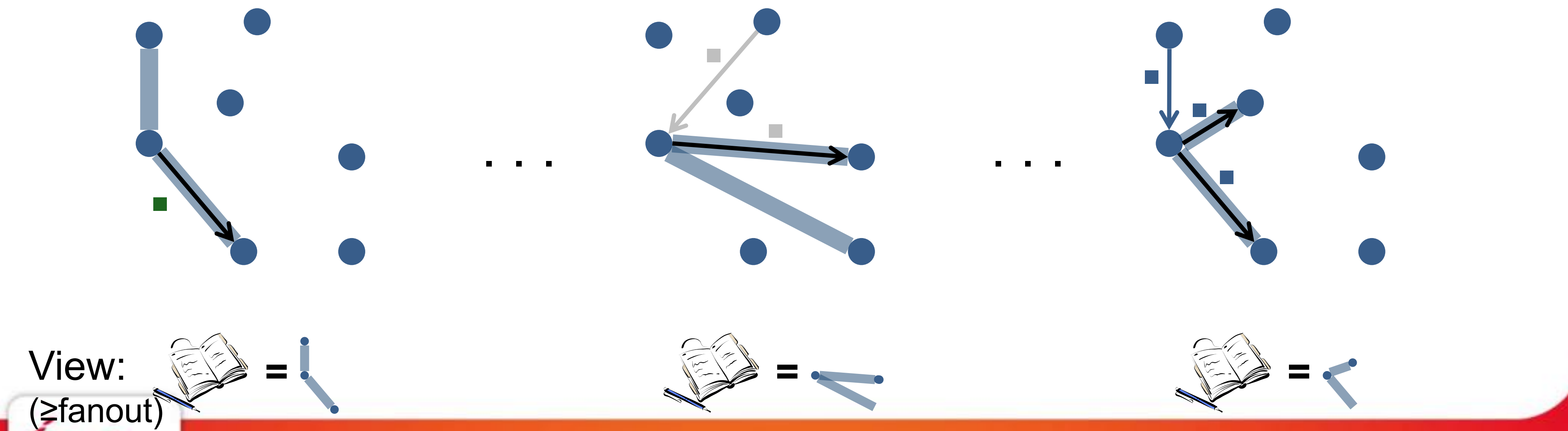
Multiple Trees

Gossip

Mesh vs Gossip



Gossip, $f = 2$



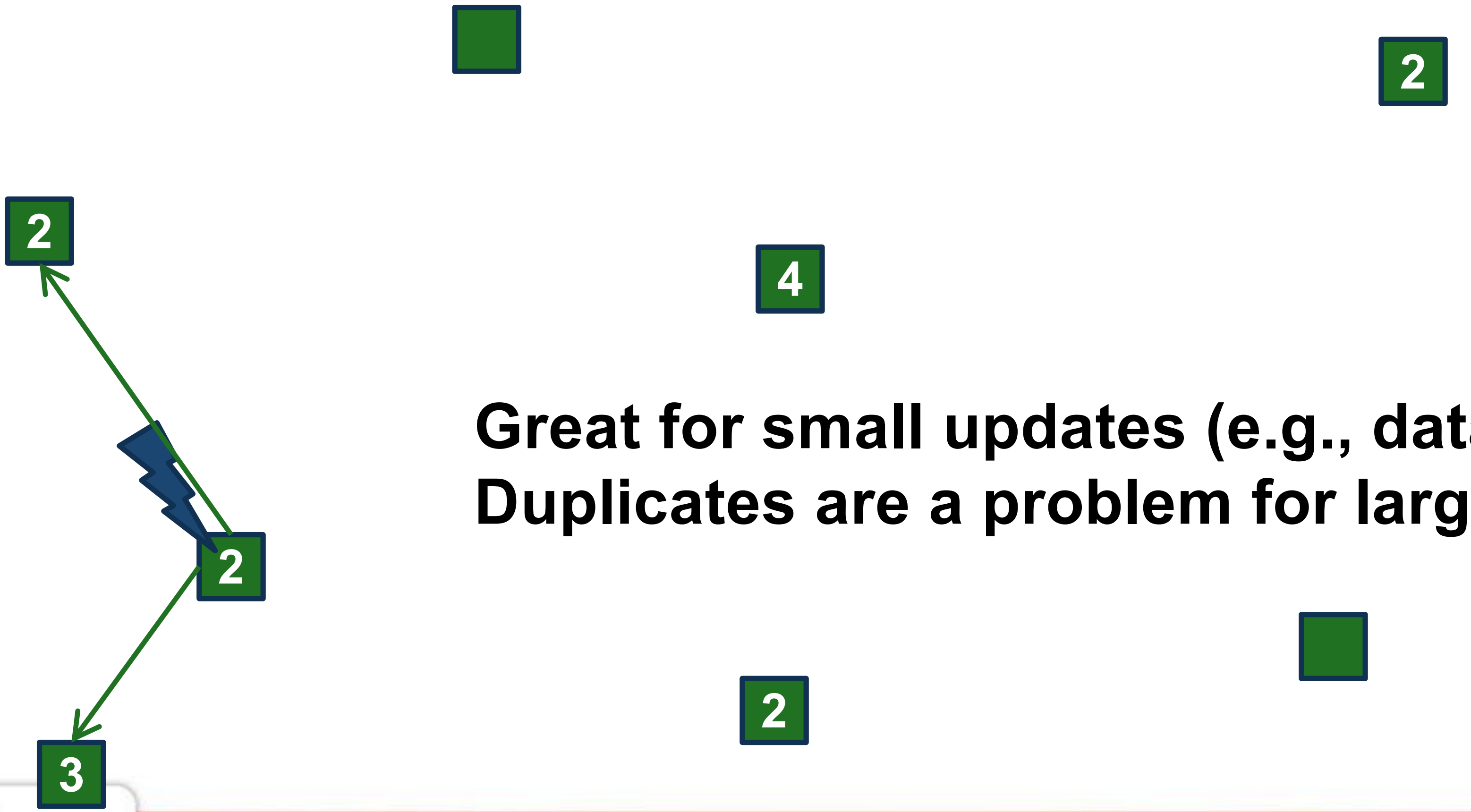
Beyond mesh: Gossip

Gossip-based dissemination

Can you see any problem?

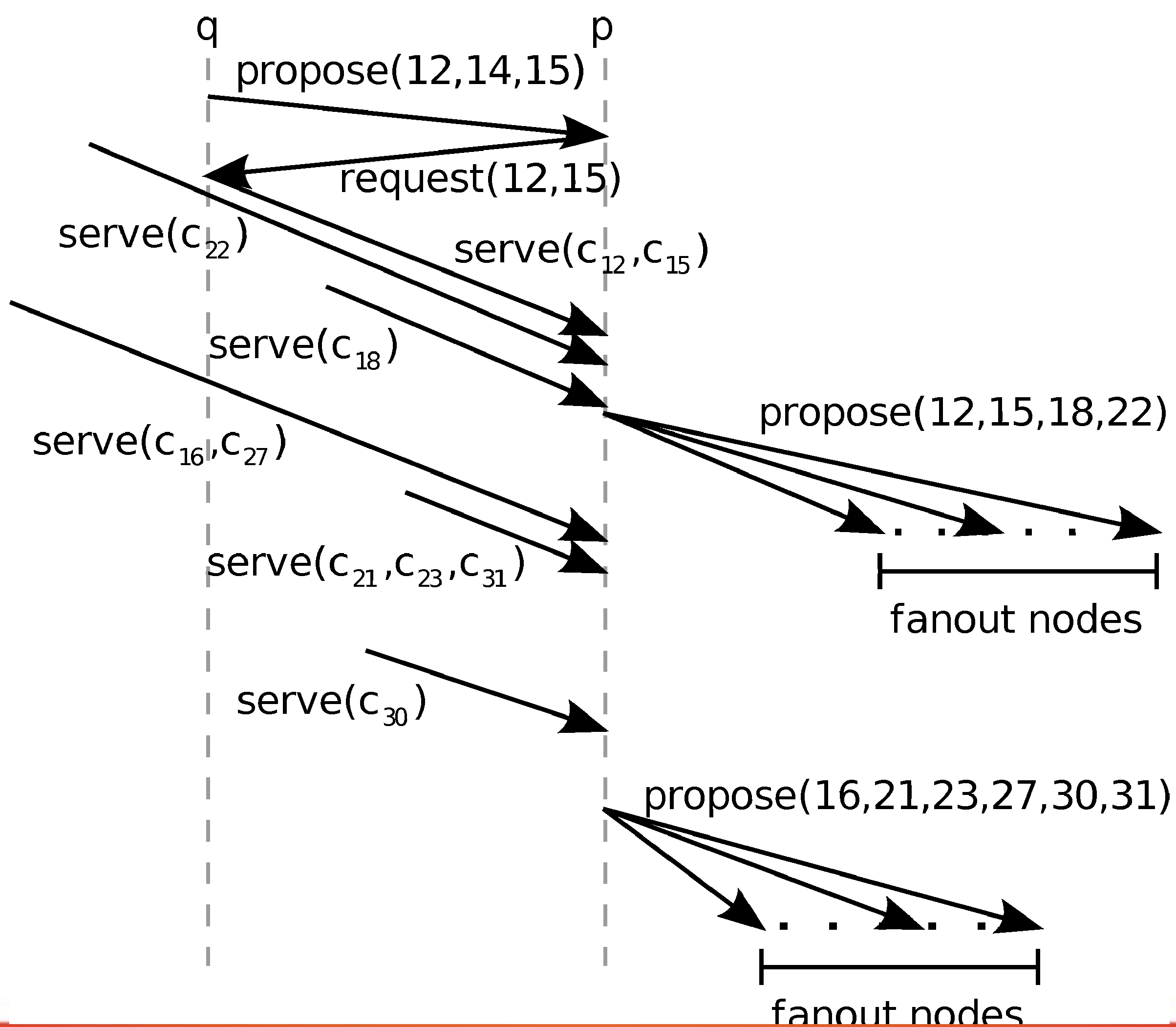
Beyond mesh: Gossip

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
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
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	

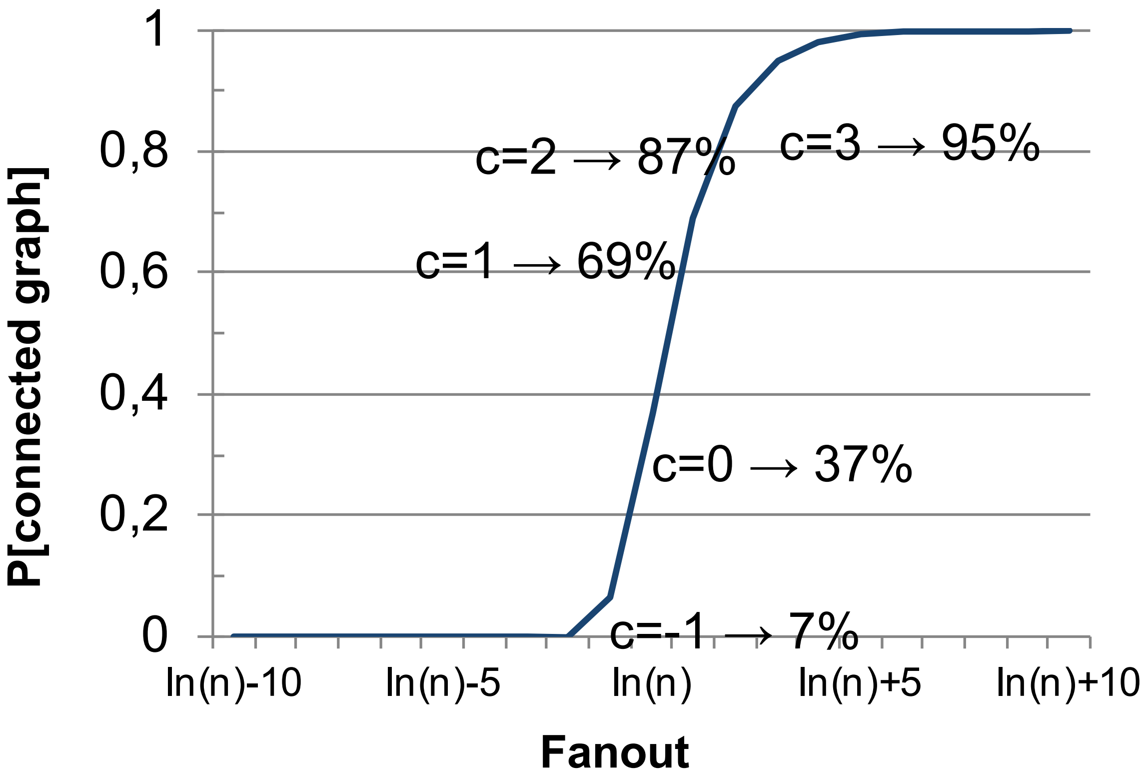
Environment (BW cap – Stream rate *s*)

Gossip (T_g – Membership)

Gossip – Theory

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

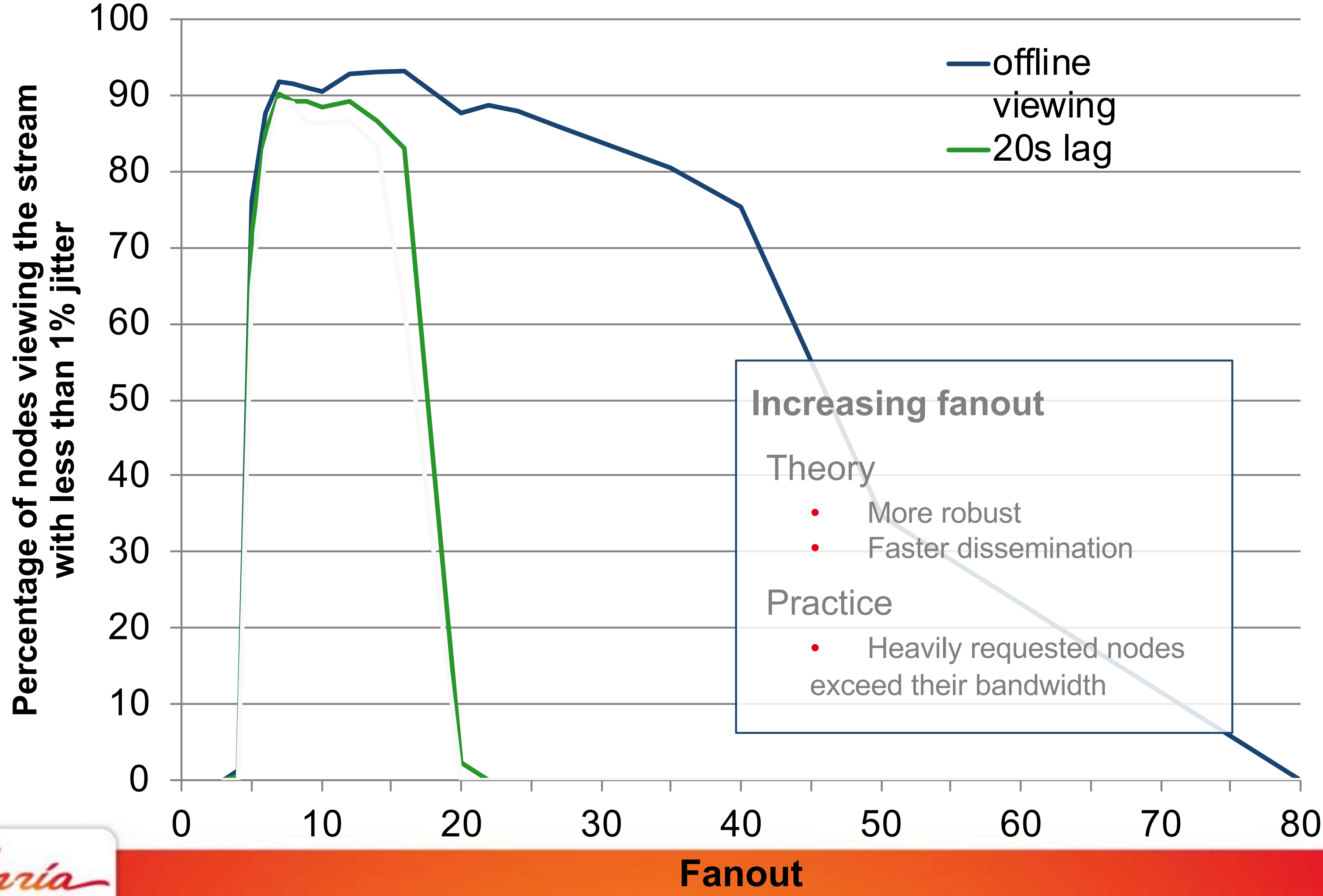
   P[connected graph] goes to exp(-exp(-c))



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

Gossip Practice

PlanetLab (230)
700 kbps cap
s = 600 kbps



Increasing fanout

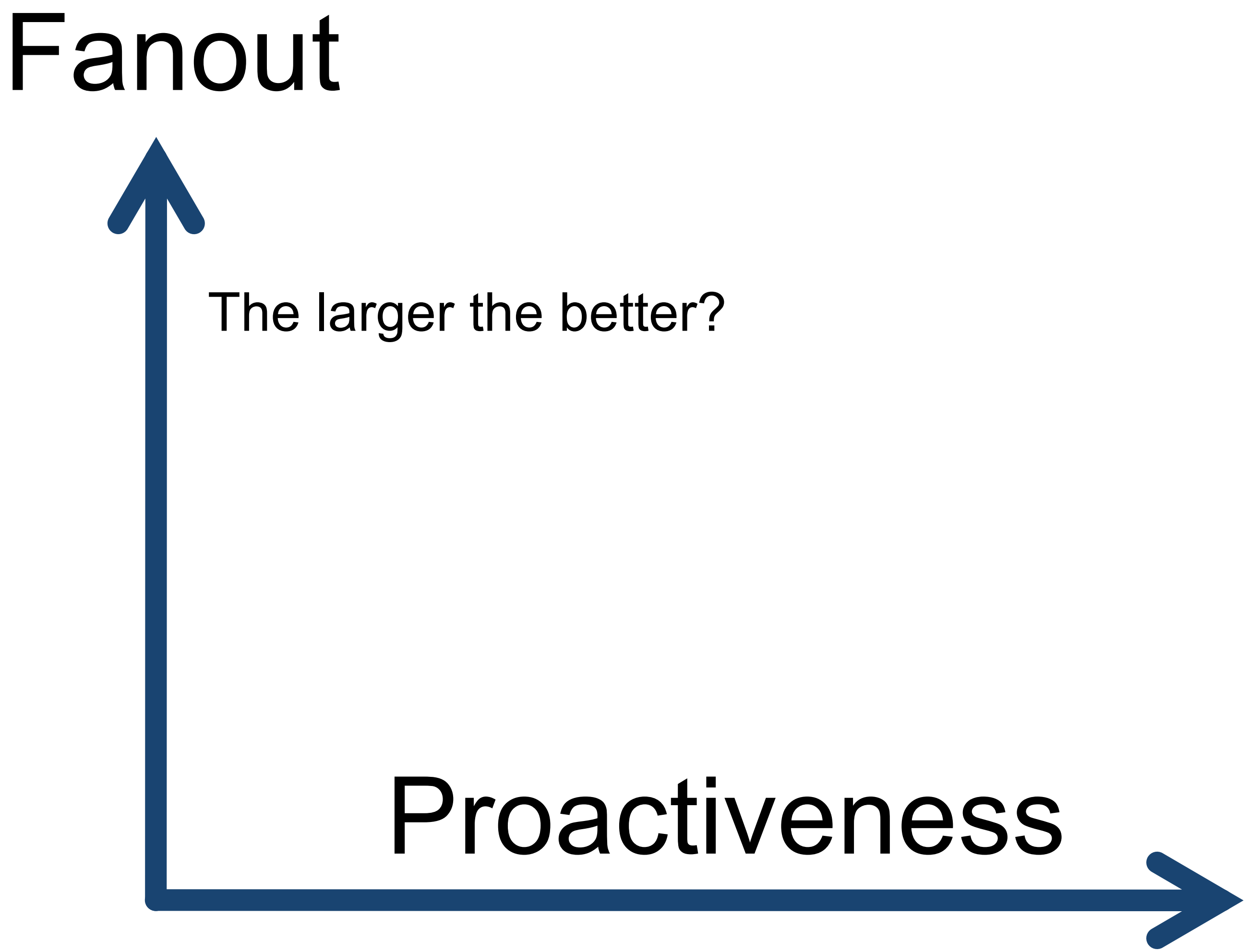
Theory

- More robust
- Faster dissemination

Practice

- Heavily requested nodes exceed their bandwidth

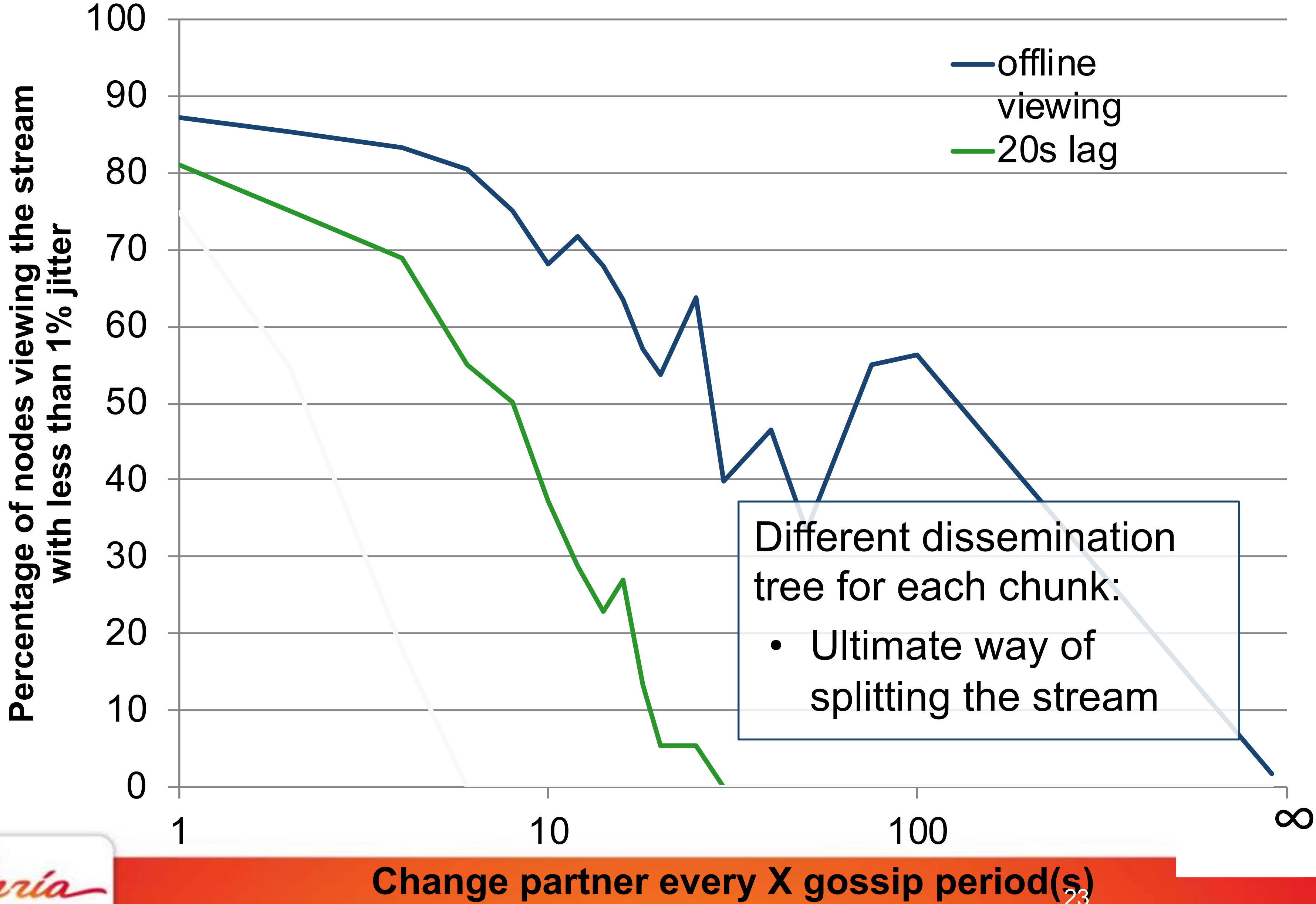
Stretching Gossip



How often should a node change its fanout partners?

Optimal proactiveness

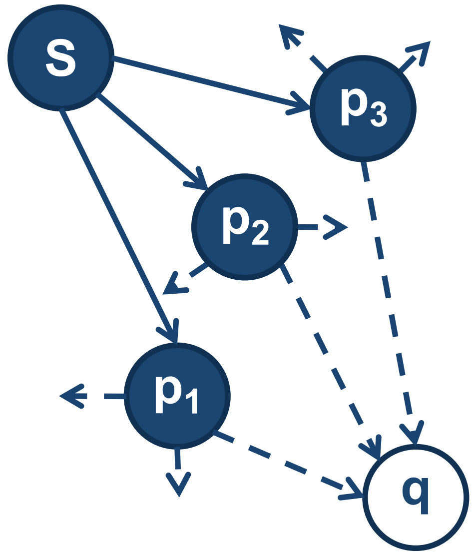
PlanetLab (230)
700 kbps cap
s = 600 kbps
f = 7



Different dissemination tree for each chunk:

- Ultimate way of splitting the stream

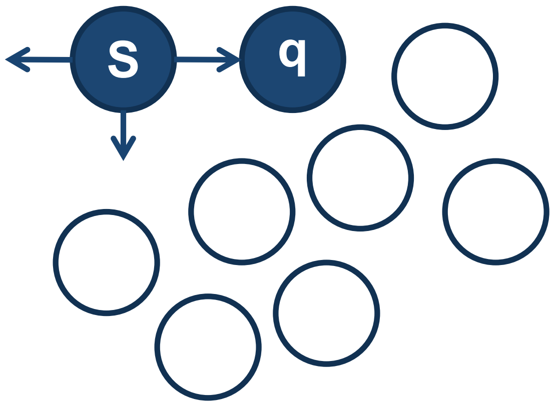
Gossip is load-balancing…



Proposals arrive randomly

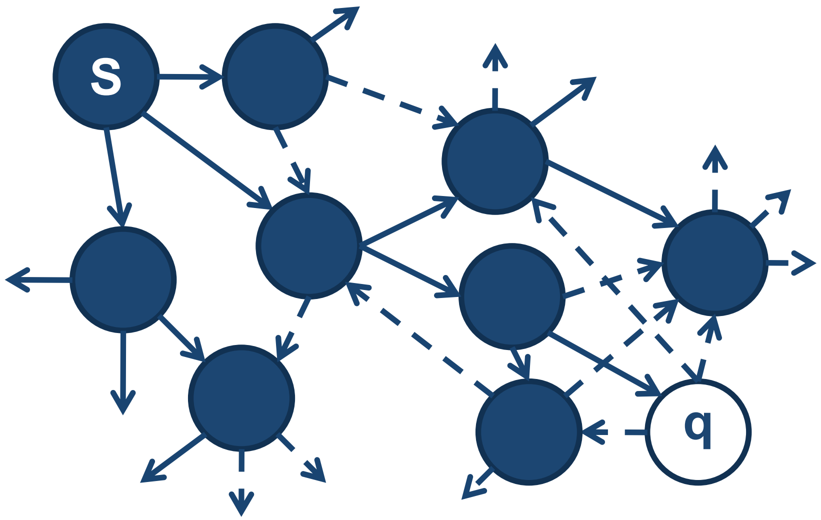
- Nodes pull from first proposal

Highly-dynamic



Node q will serve *f* nodes whp

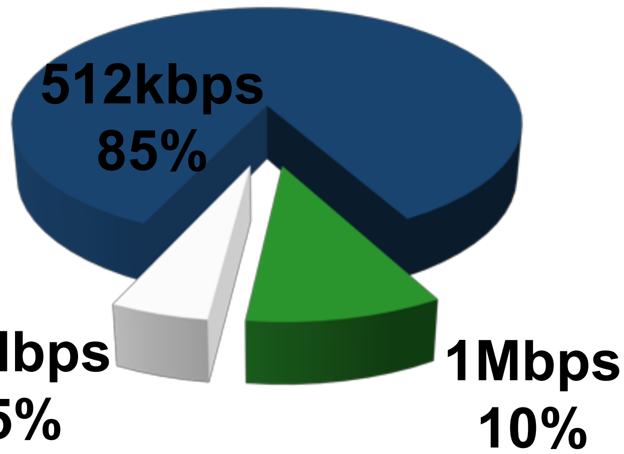
…



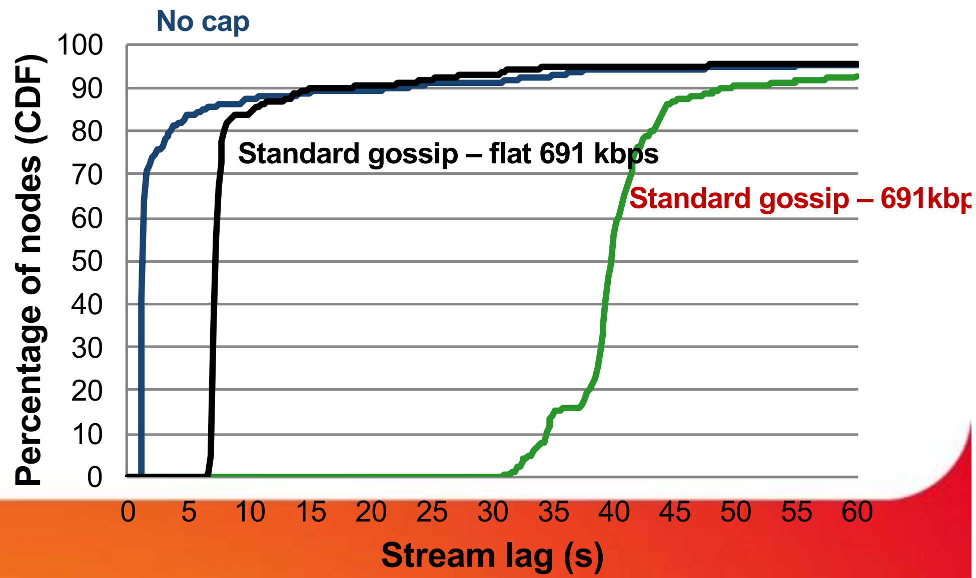
Node q will serve *f* nodes wlp

… 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

VS

- Increase fanout…

… and decrease it too!

Such that

- average fanout (f_{avg}) $\geq$ 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

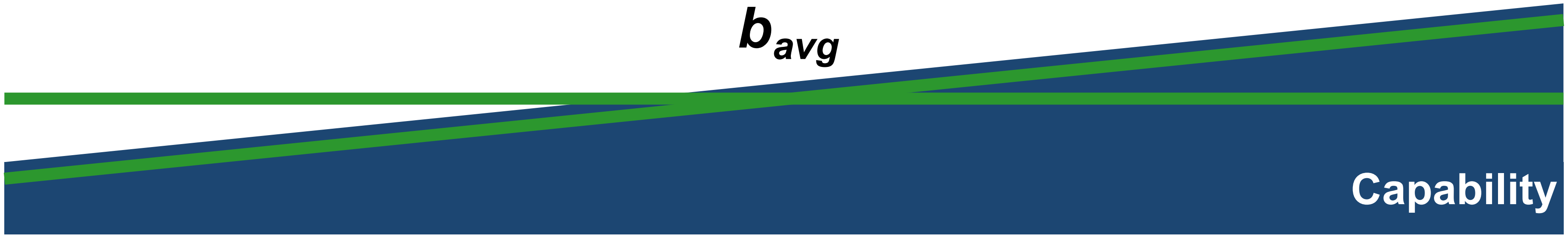
Who should increase/decrease its contribution?

… 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 $f_{init} = \ln(\boldsymbol{n}) + c$

HEAP

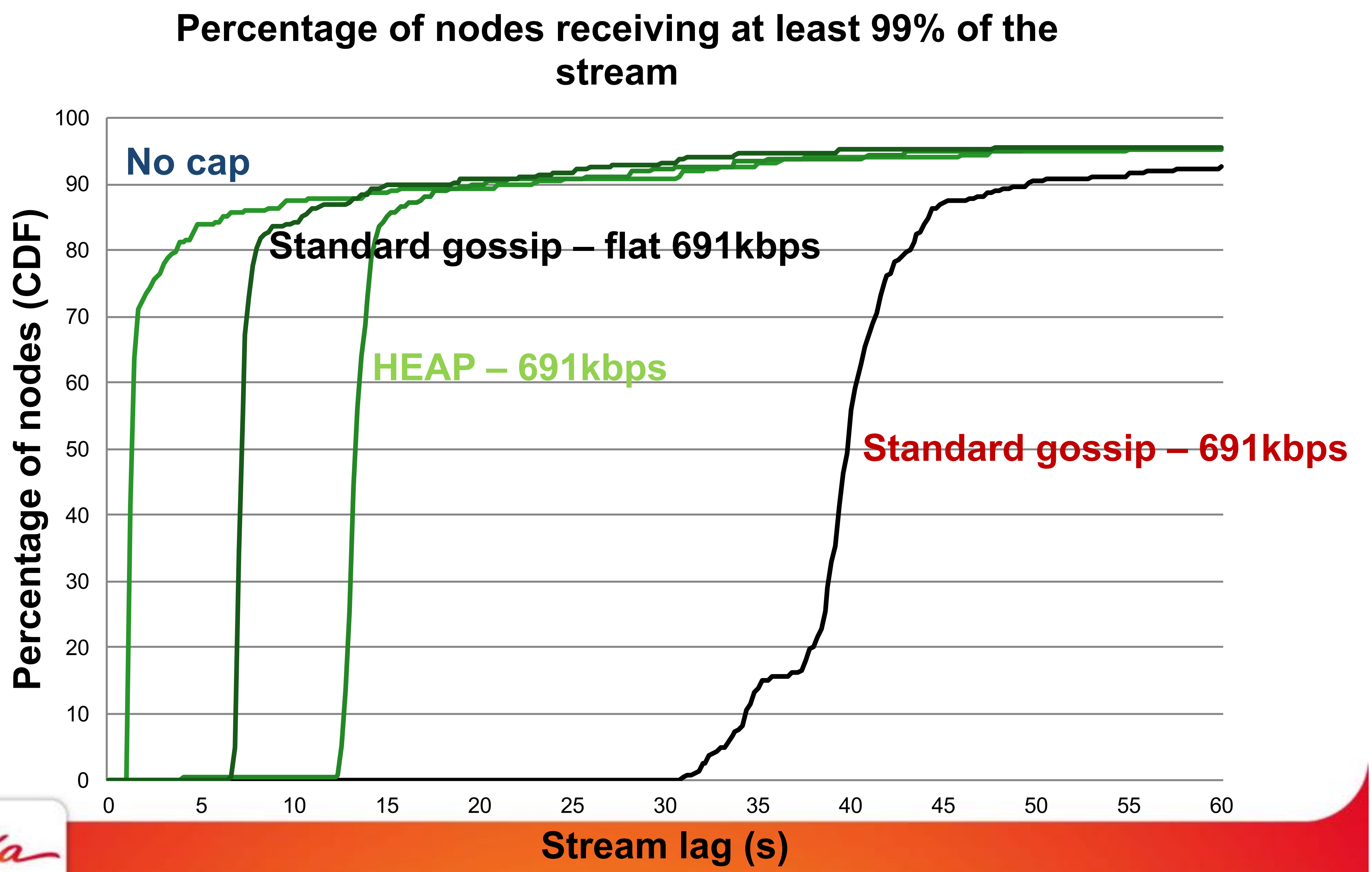
Get b_{avg} with (gossip) aggregation

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

Get n with (gossip) size estimation

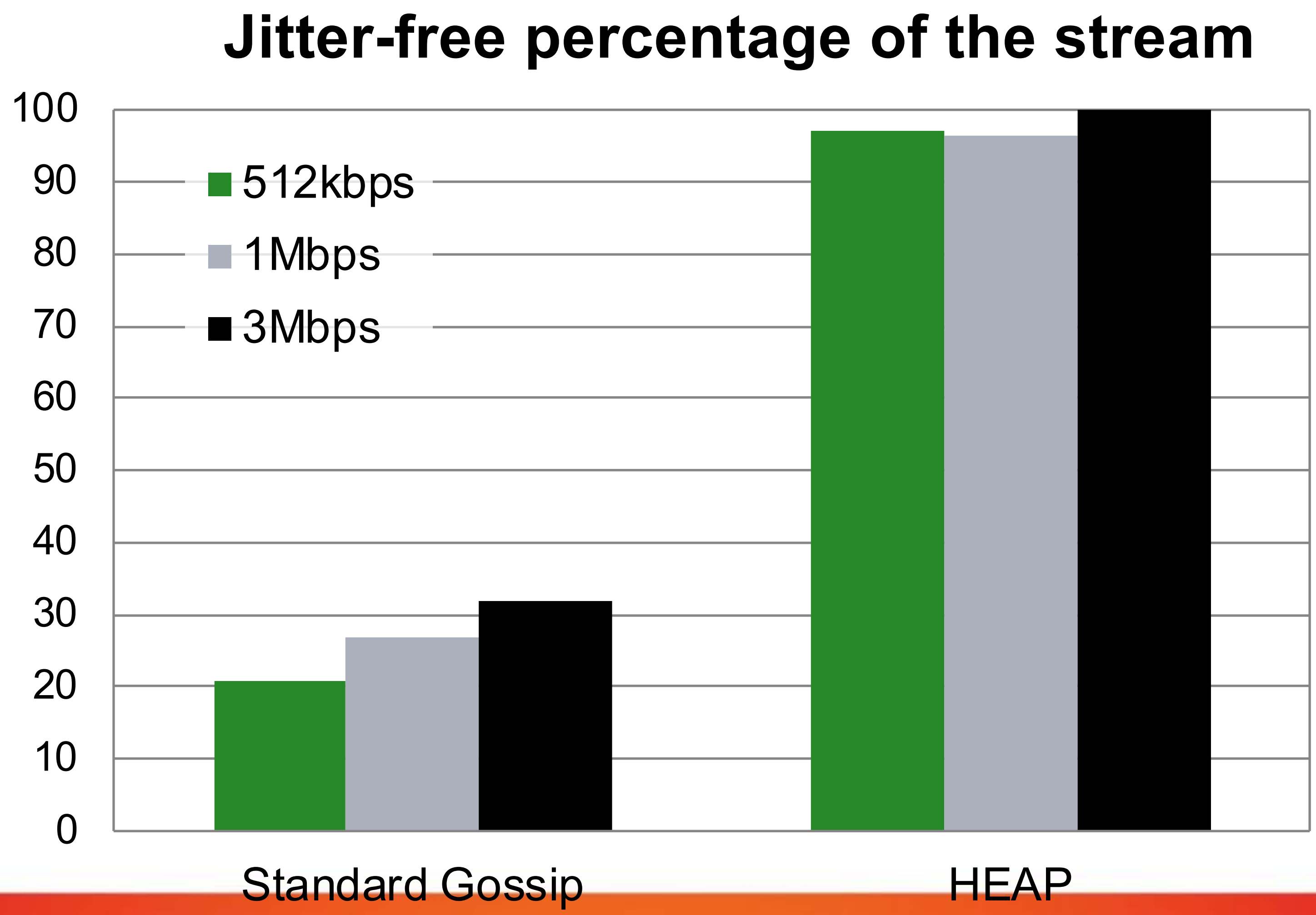
- Estimation follows change in the system
    - Join/leave
    - Crashes
    - …

Stream lag reduction



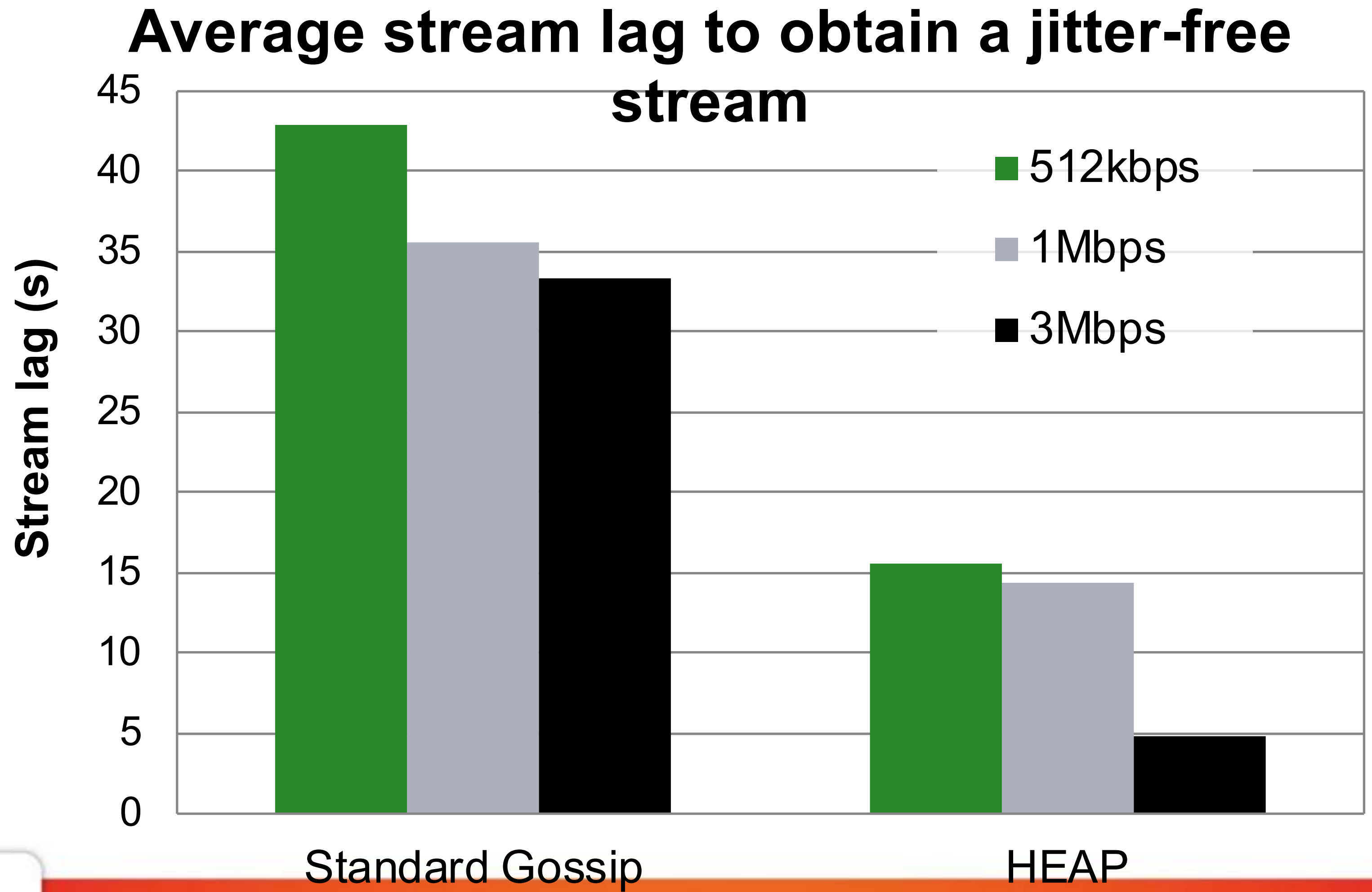
Quality improvement

Stream lag of 10s

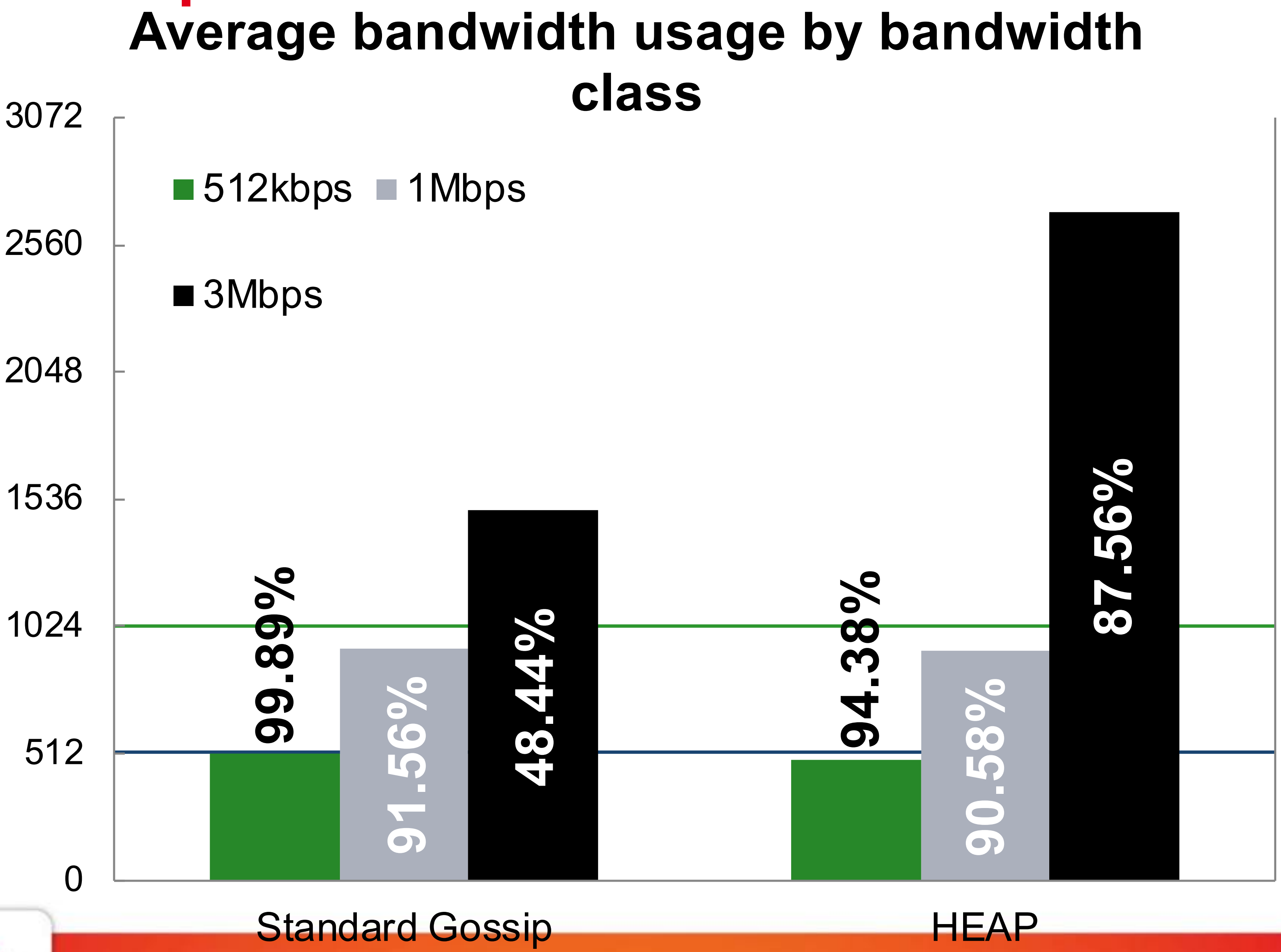


Stream lag

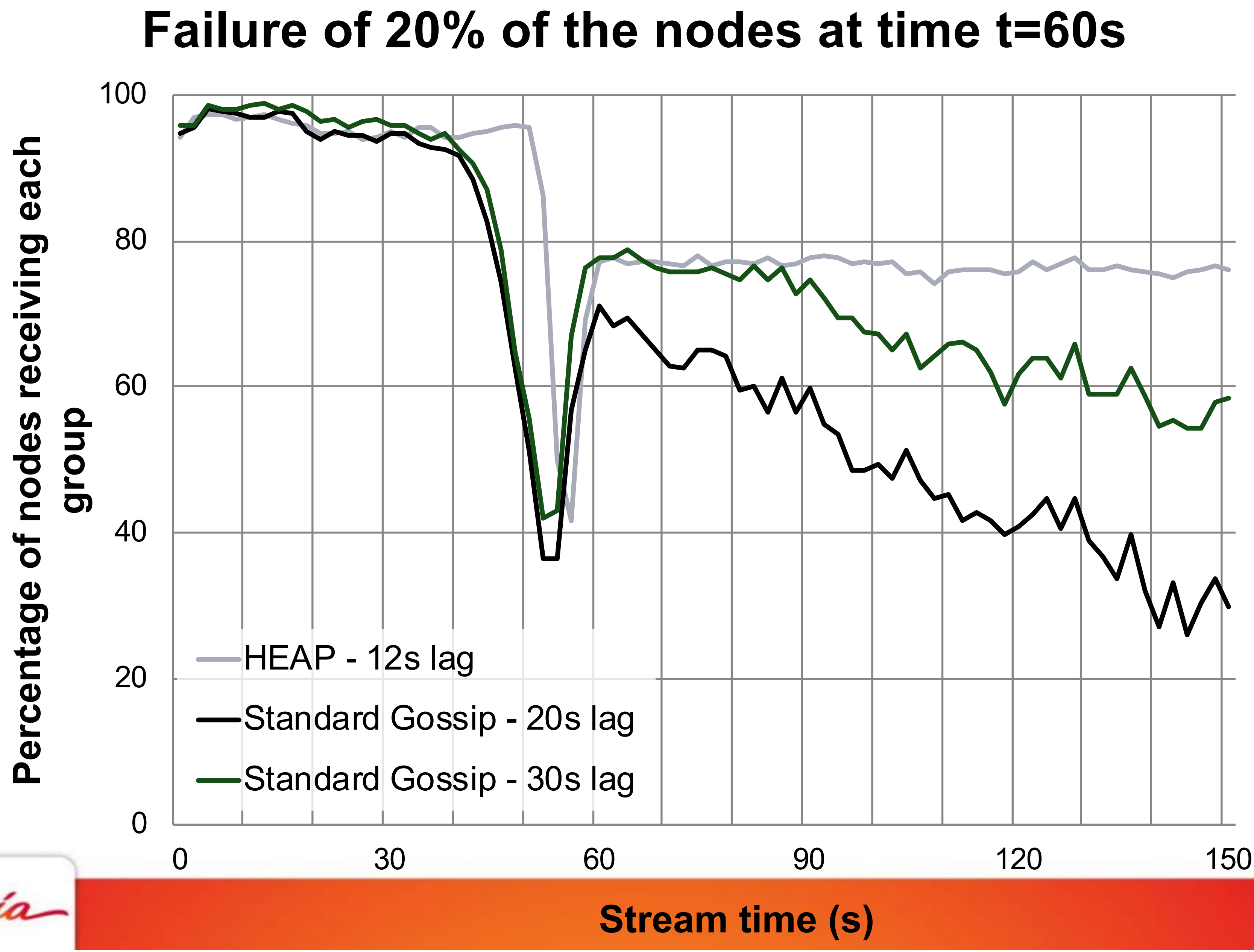
For those who can have a jitter-free stream



Proportional contribution

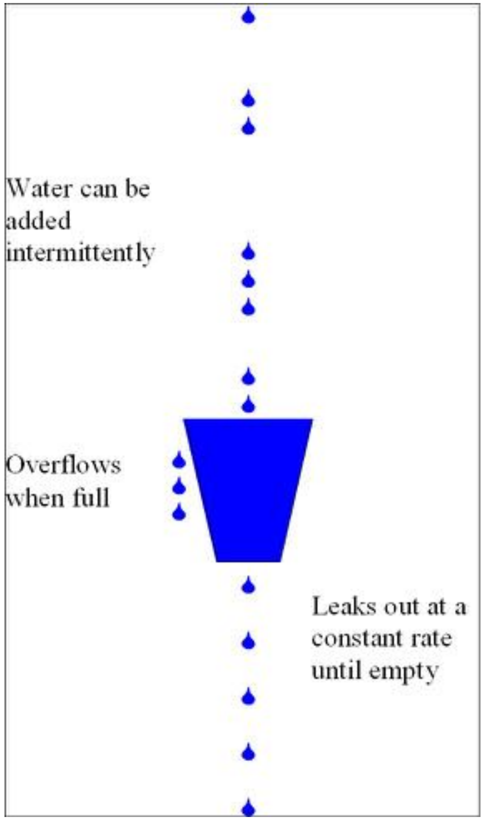


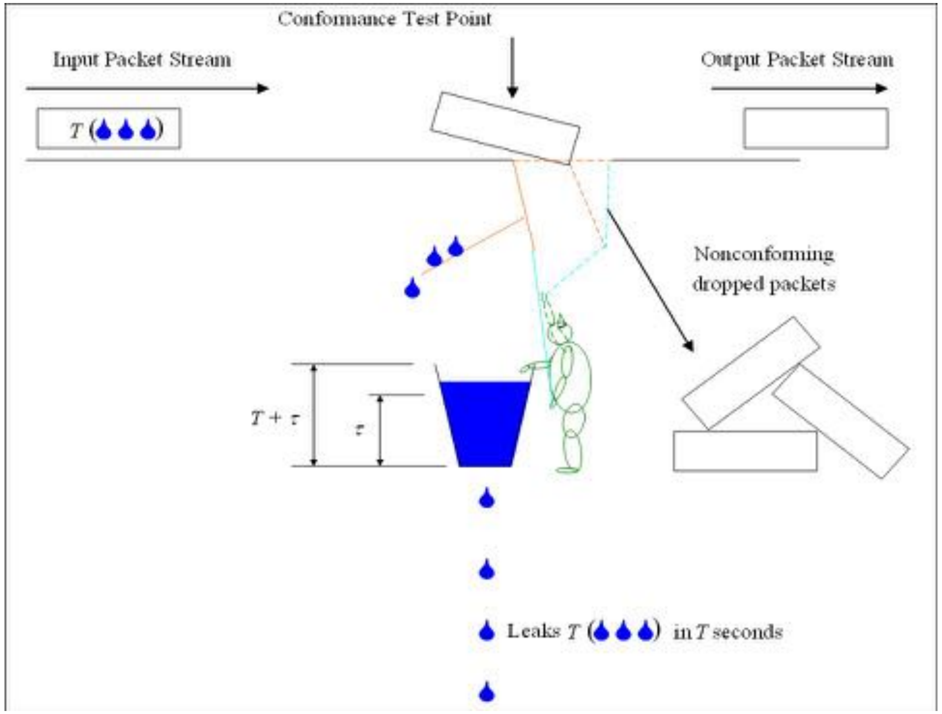
20% nodes crashing



About Bandwidth Limitation

- Leaky Bucket
- Token Bucket

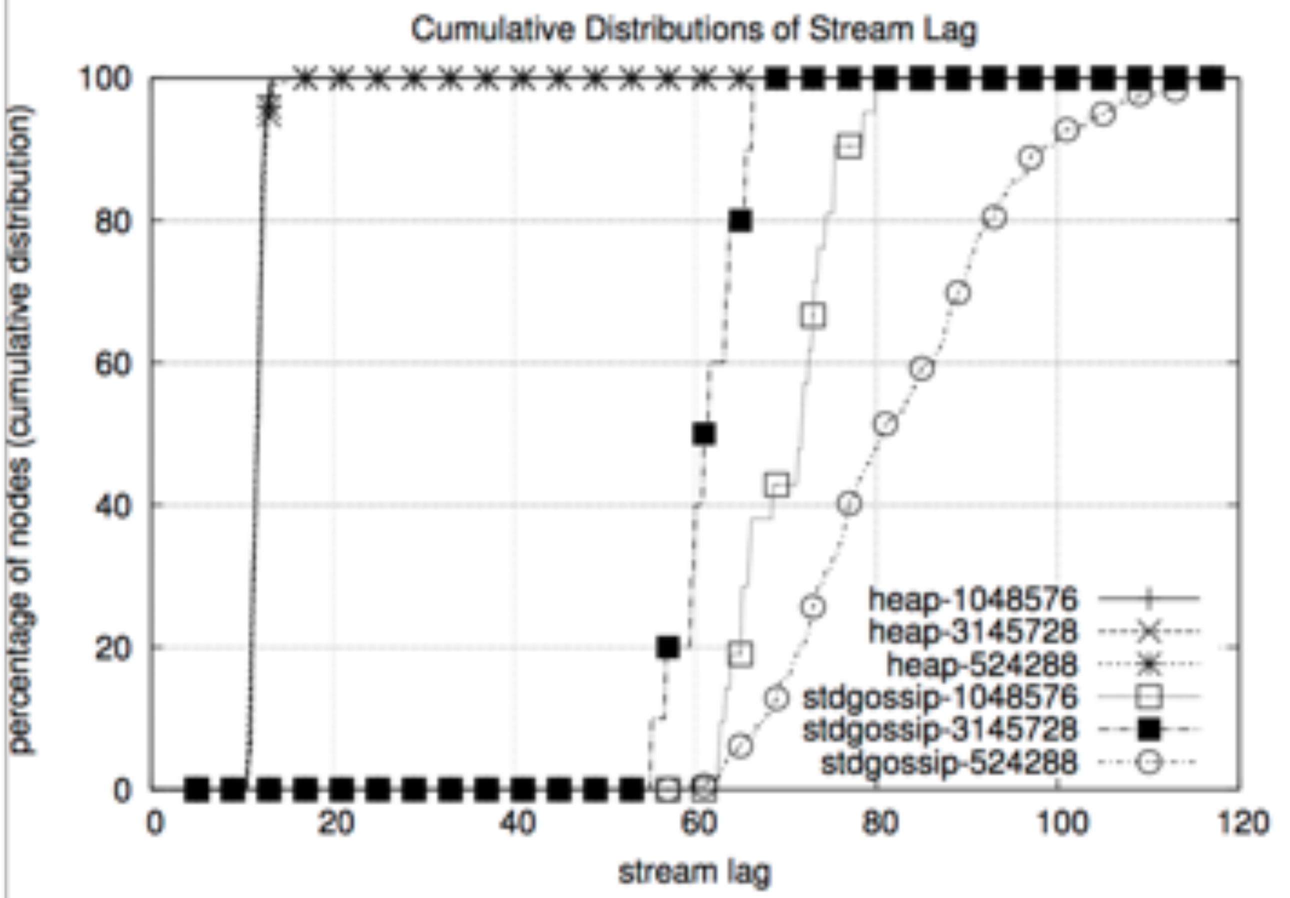




By Graham.Fountain [CC BY 3.0 (http://creativecommons.org/licenses/by/3.0)], via Wikimedia Commons

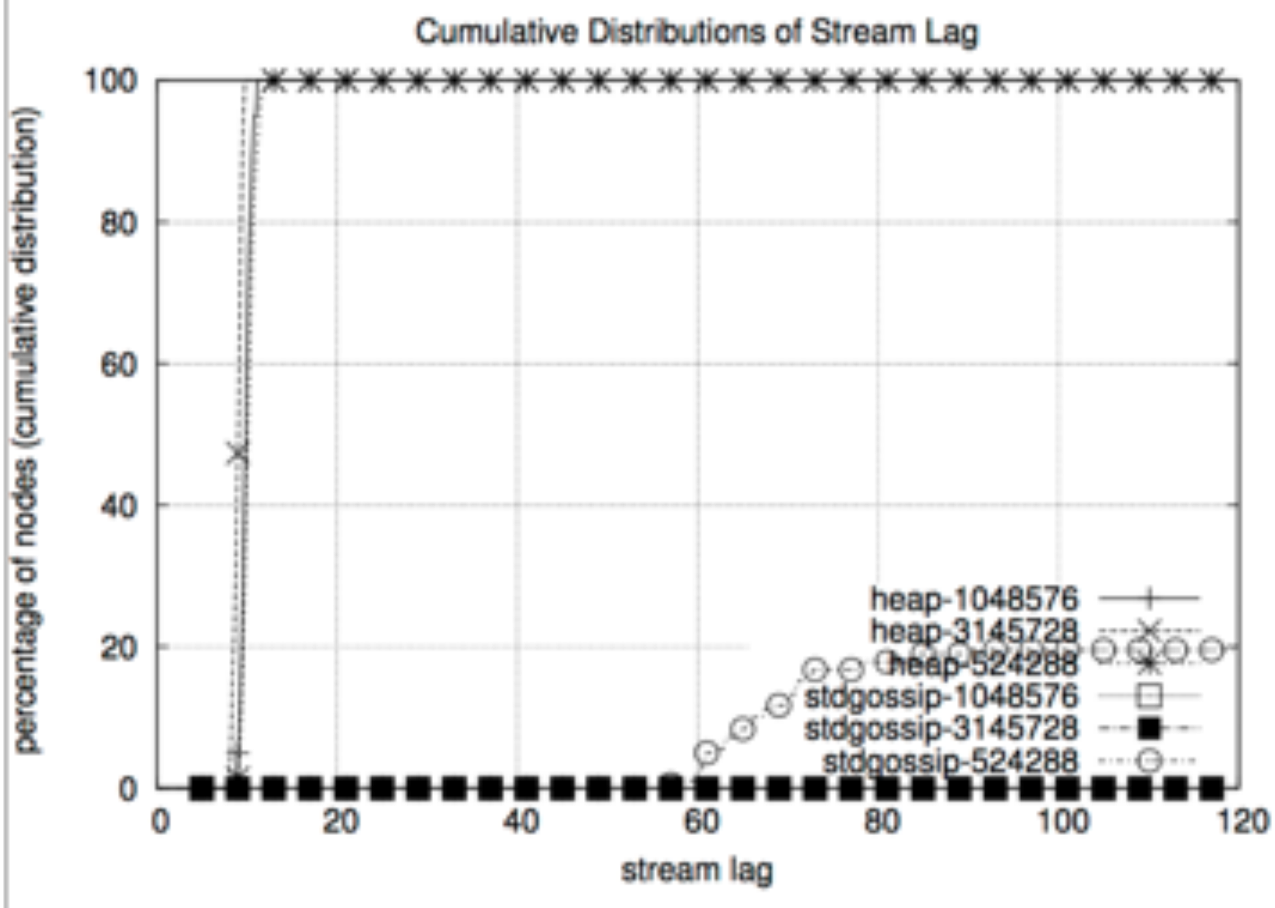
By Graham.Fountain at English Wikipedia, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=35271394

Unbounded Leaky Bucket



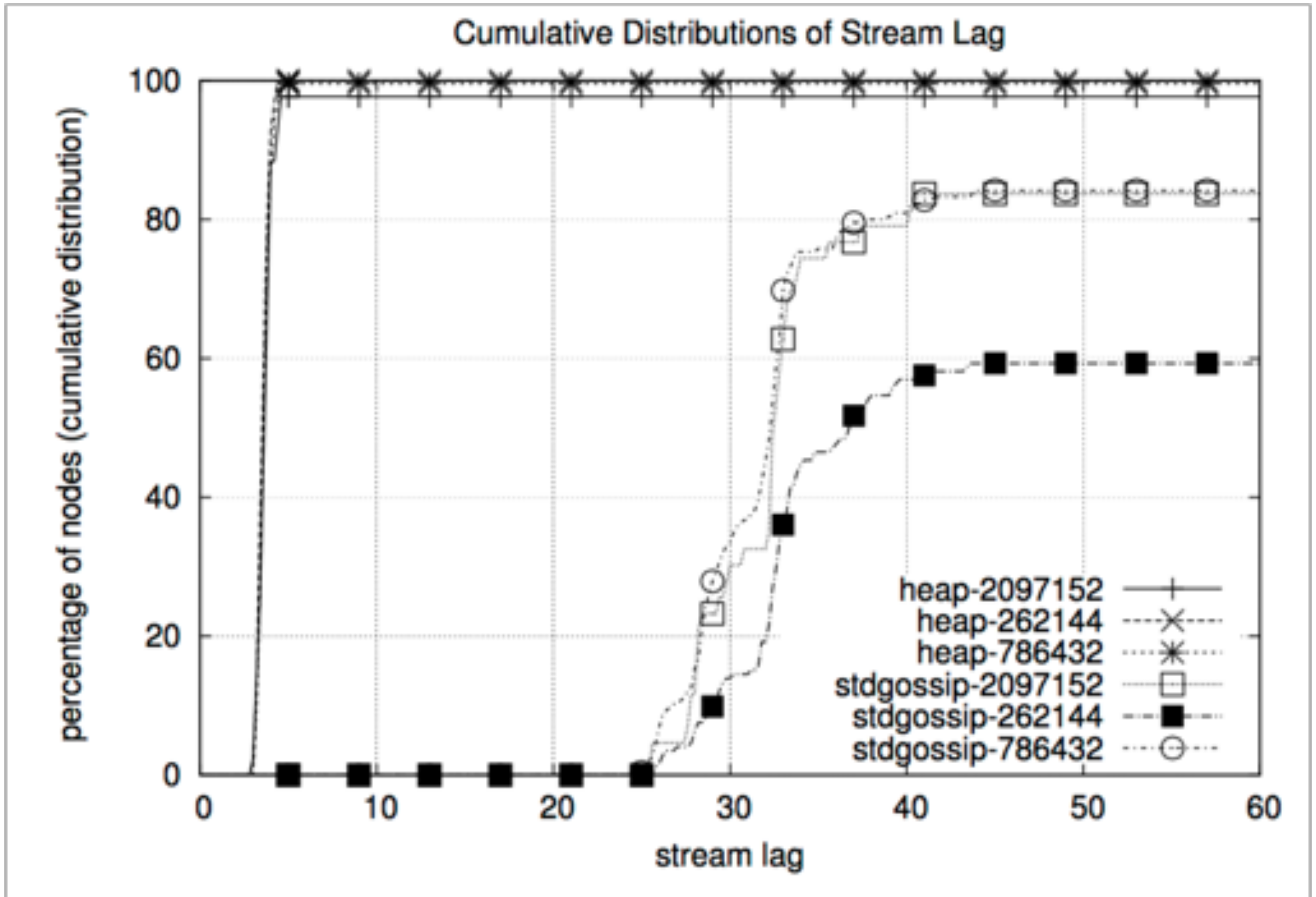
(b) ms691 Infinite Leaky Bucket

Bounded Leaky Bucket



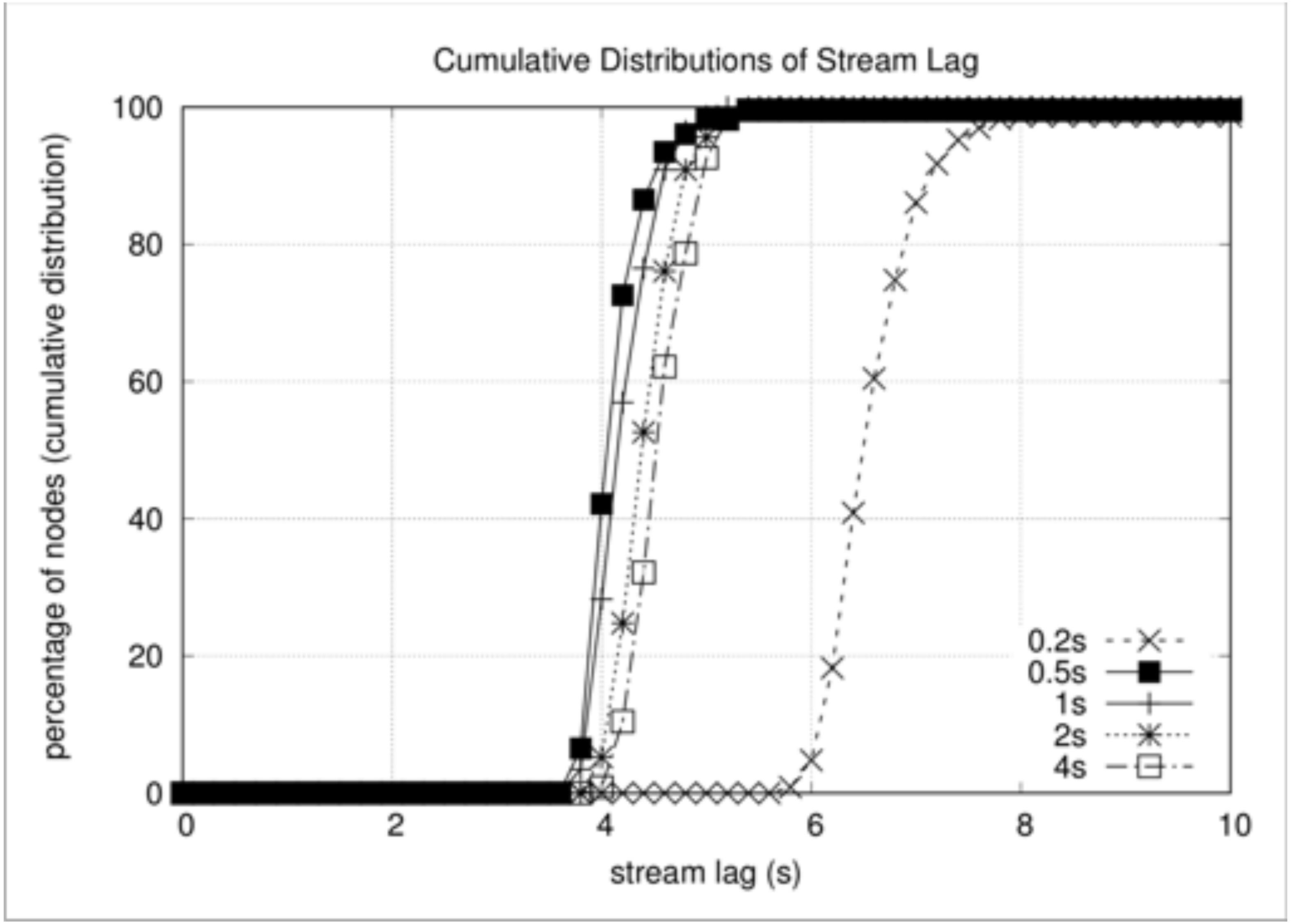
(a) ms691 200KB Leaky Bucket

Token Bucket

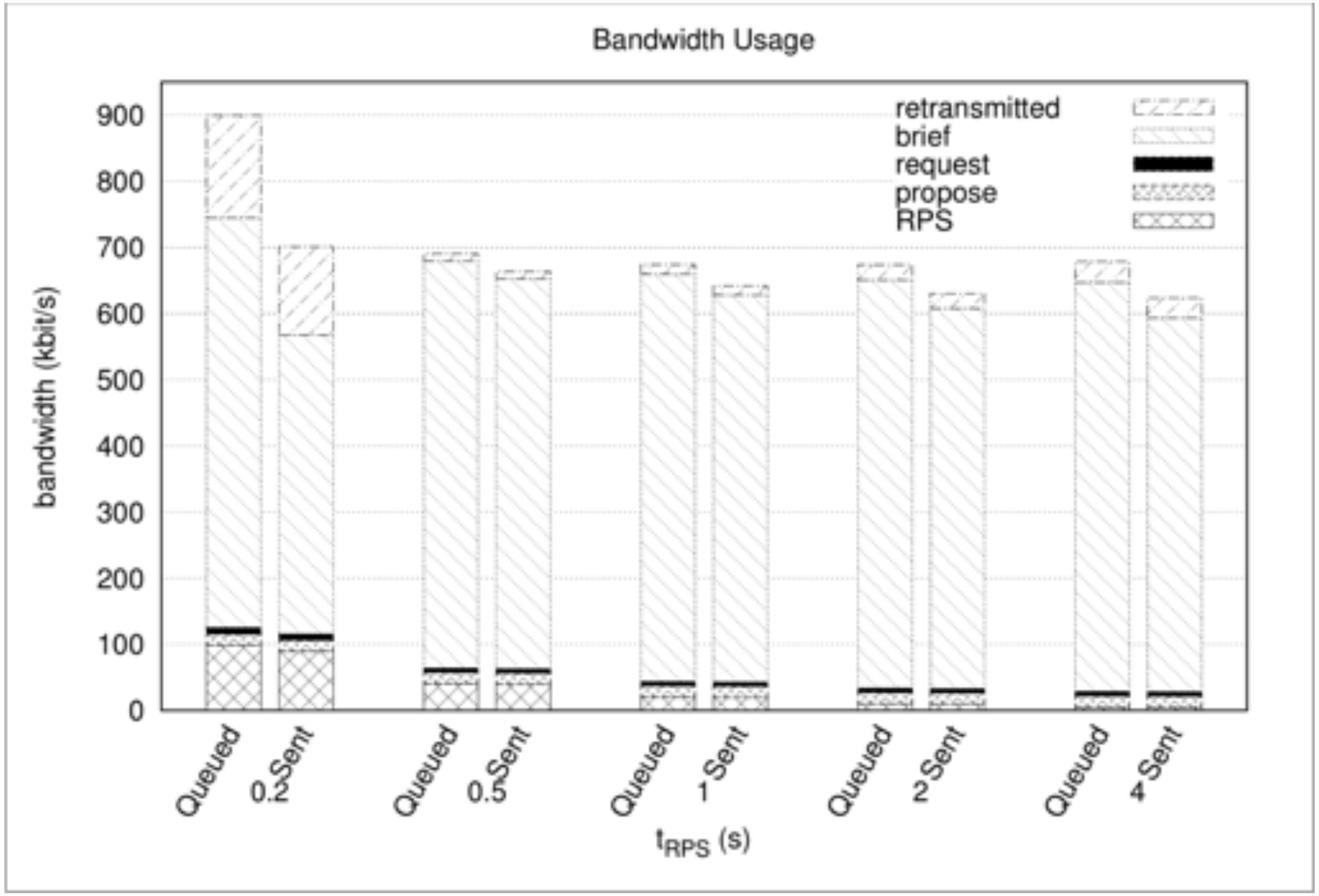


(a) ref691

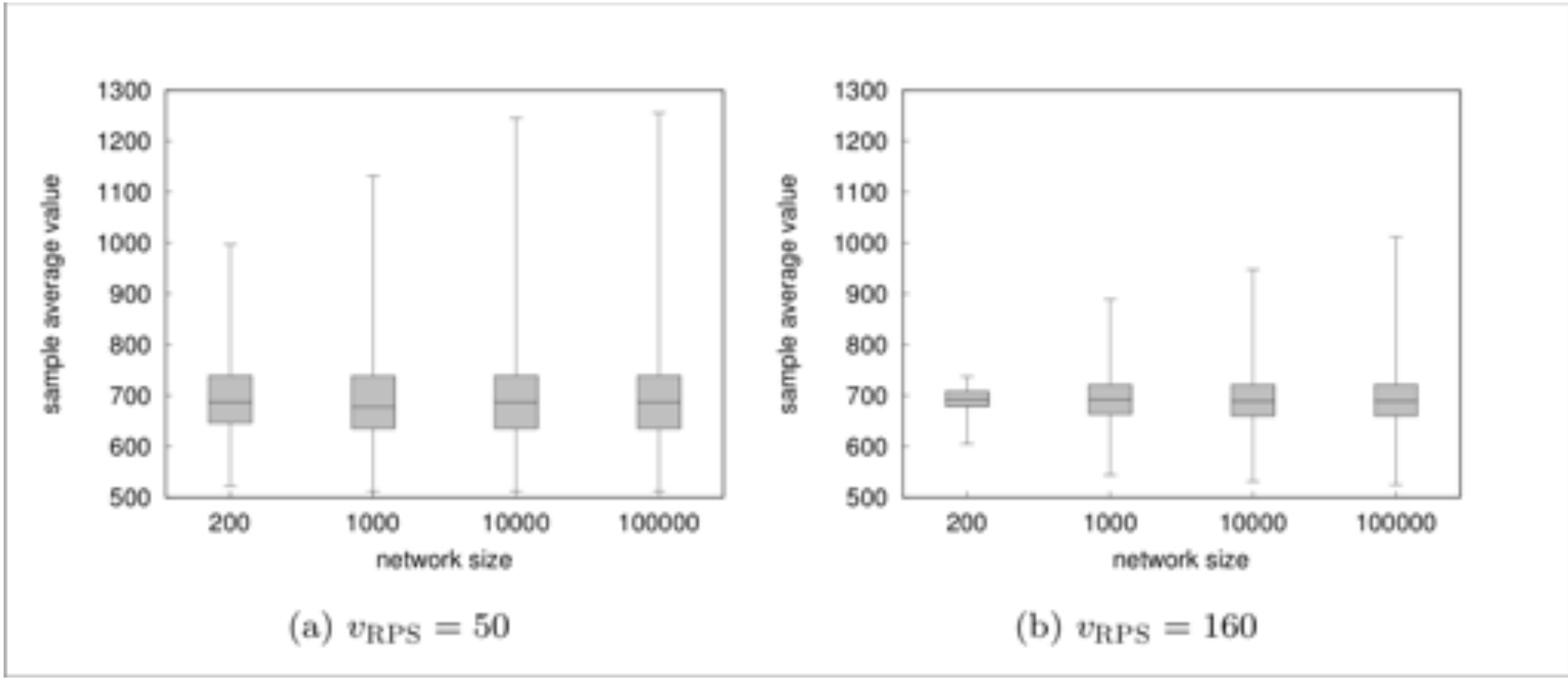
Stream Lag vs RPS frequency



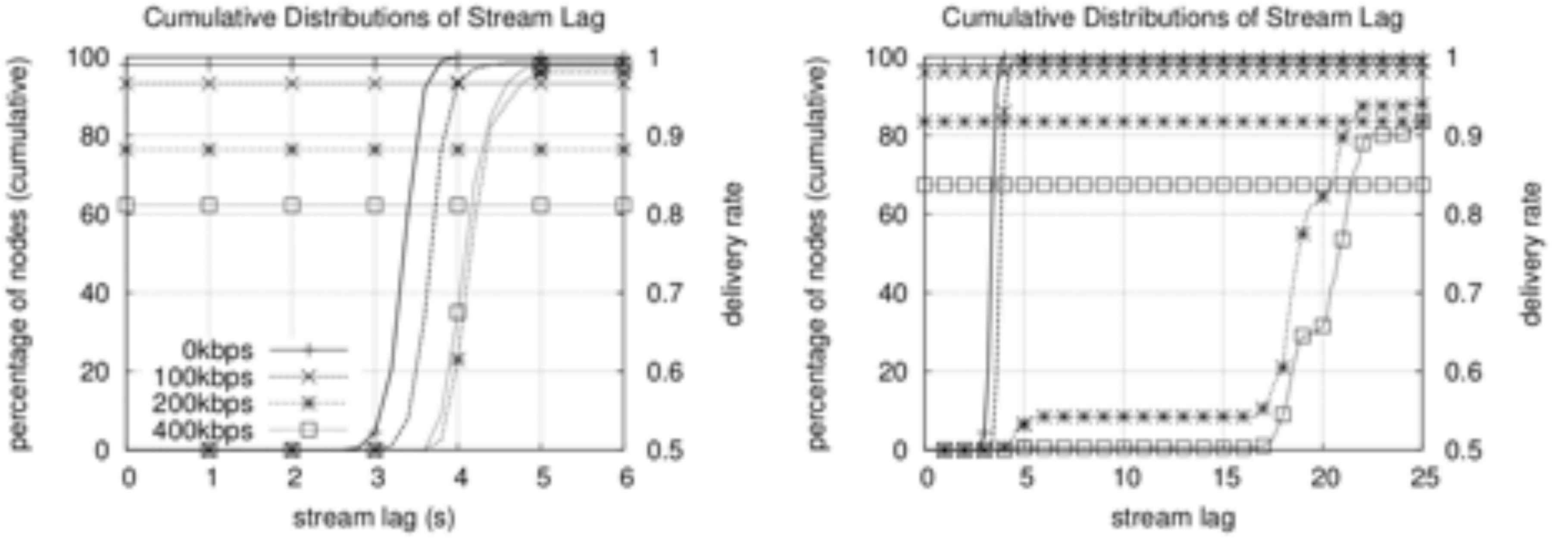
Bandwidth vs RPS Frequency



RPS-based Averaging

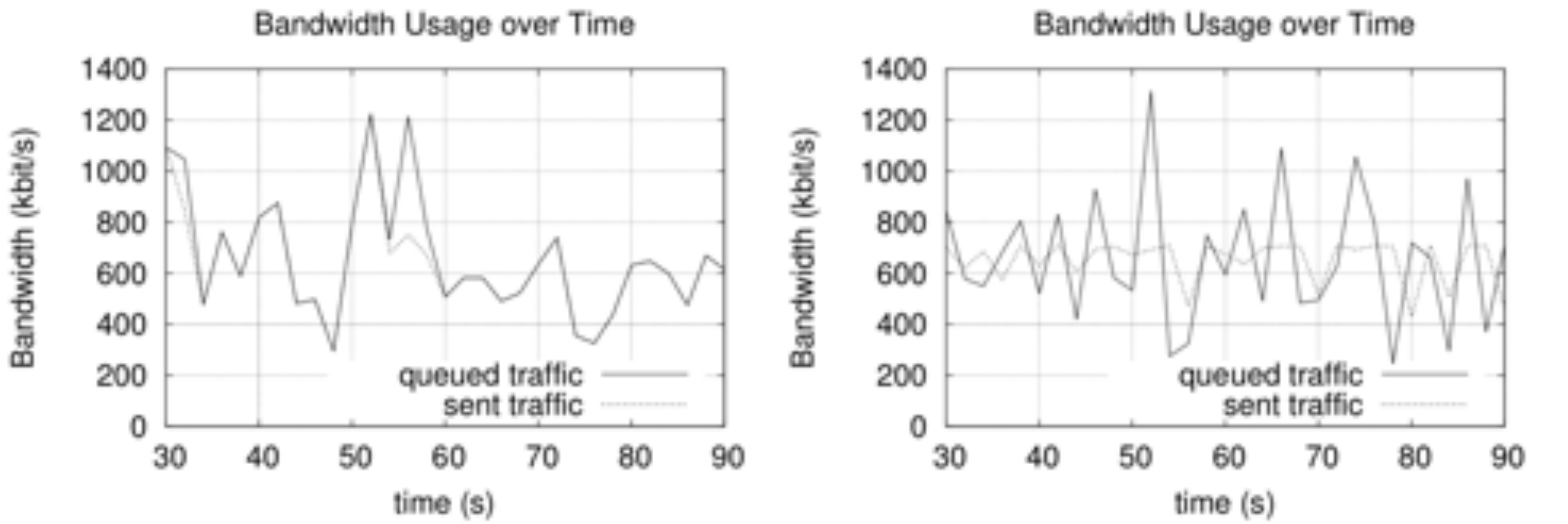


Cohabitation with External Applications



(a) $T_{app} = 200\,\text{ms}$ (b) $T_{app} = 100\,\text{ms}$

Ideas for Improvement



(a) Token Bucket 200k

(b) Leaky Bucket 200k

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

DoNet

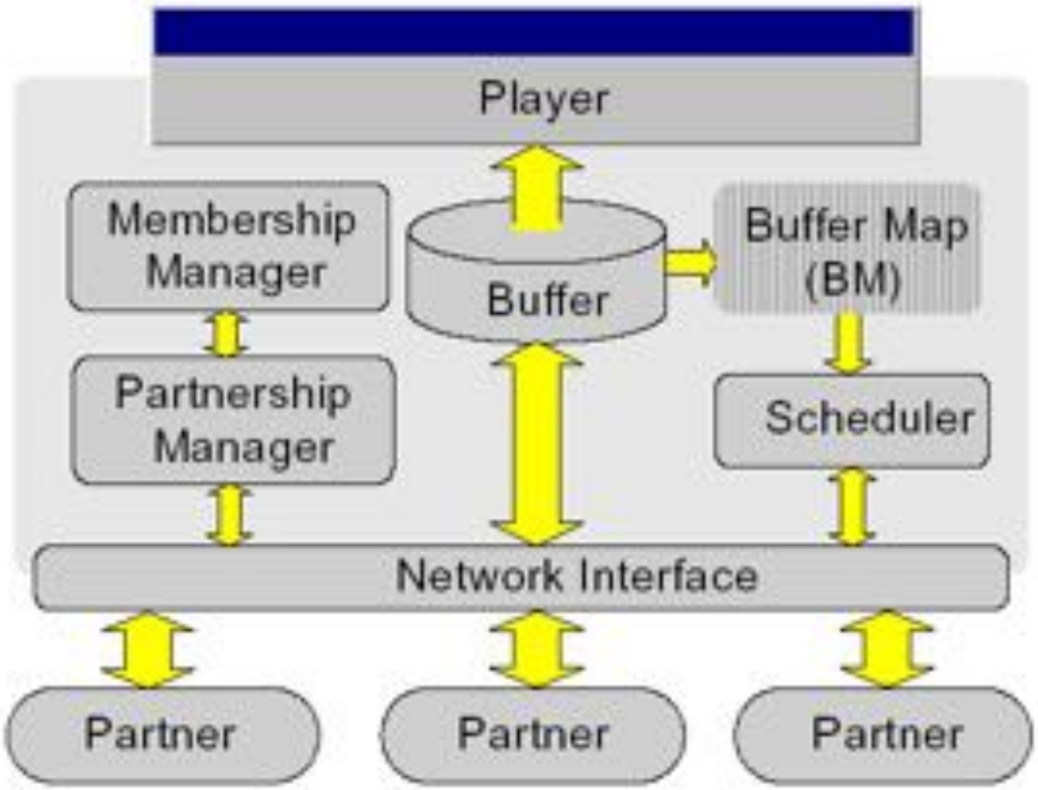
DoNet

- Data availability guides flow direction
  - no specific overlay structure
- Greater flexibility in application layer
  - larger buffers, determine the data forwarding directions adaptively and intelligently
- Semistatic structure
  - constantly rendered to suboptimal by node dynamics
  - suitable for overlay with high dynamic nodes:

Key Design Issues of DONet

- how to form partnerships
- how to encode and exchange data availability information
- how to supply video data to partners and receive it from them

Design & Optimization



A generic system diagram for a DONet node.

DoNet Roles

- membership manager:
  - maintain a partial view of other overlay nodes
- partnership manager:
  - establishes and maintains the partnership with other nodes
- Scheduler:
  - schedules the transmission of video data
- DONet node
  - Receiver and supplier
  - Origin is only supplier

Node Join and Membership Management

- Membership cache
  - contains a partial list of the unique identifiers for nodes
- redirect to obtain list
  - new node → origin node → deputy node
- Scalable Gossip Membership protocol
  - periodically distribute membership messages
  - similar to RPS
- Decrease TTL in mCache when
  - Node forwards membership message
  - Node serves as a deputy and includes entry in partner candidate list

Resulting Overlay

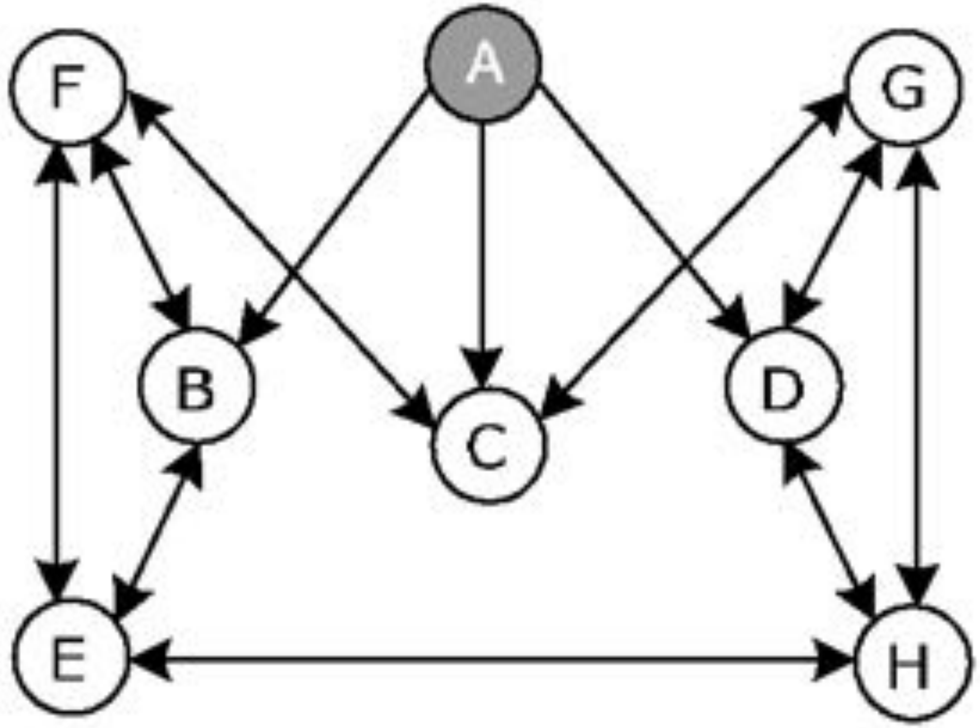
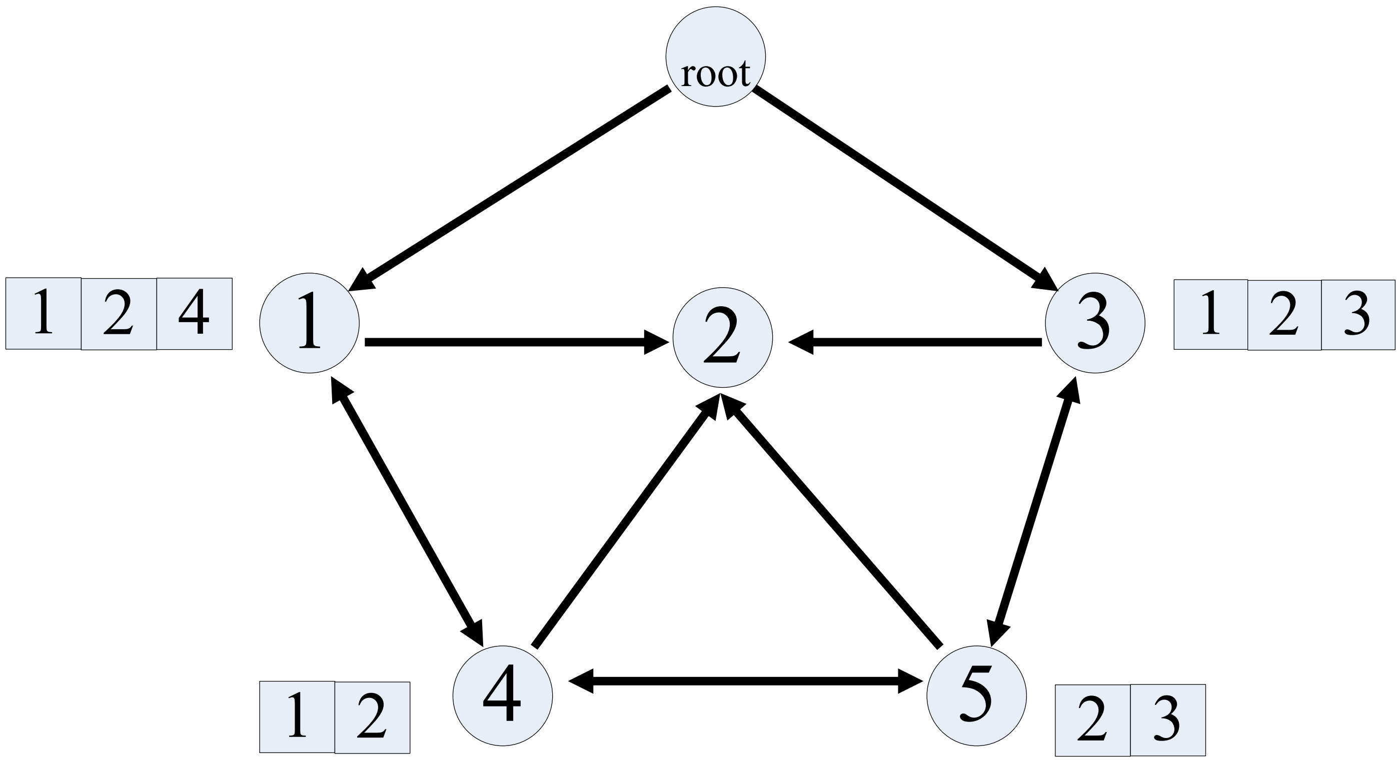


Illustration of the partnership in DONet (origin node: A).

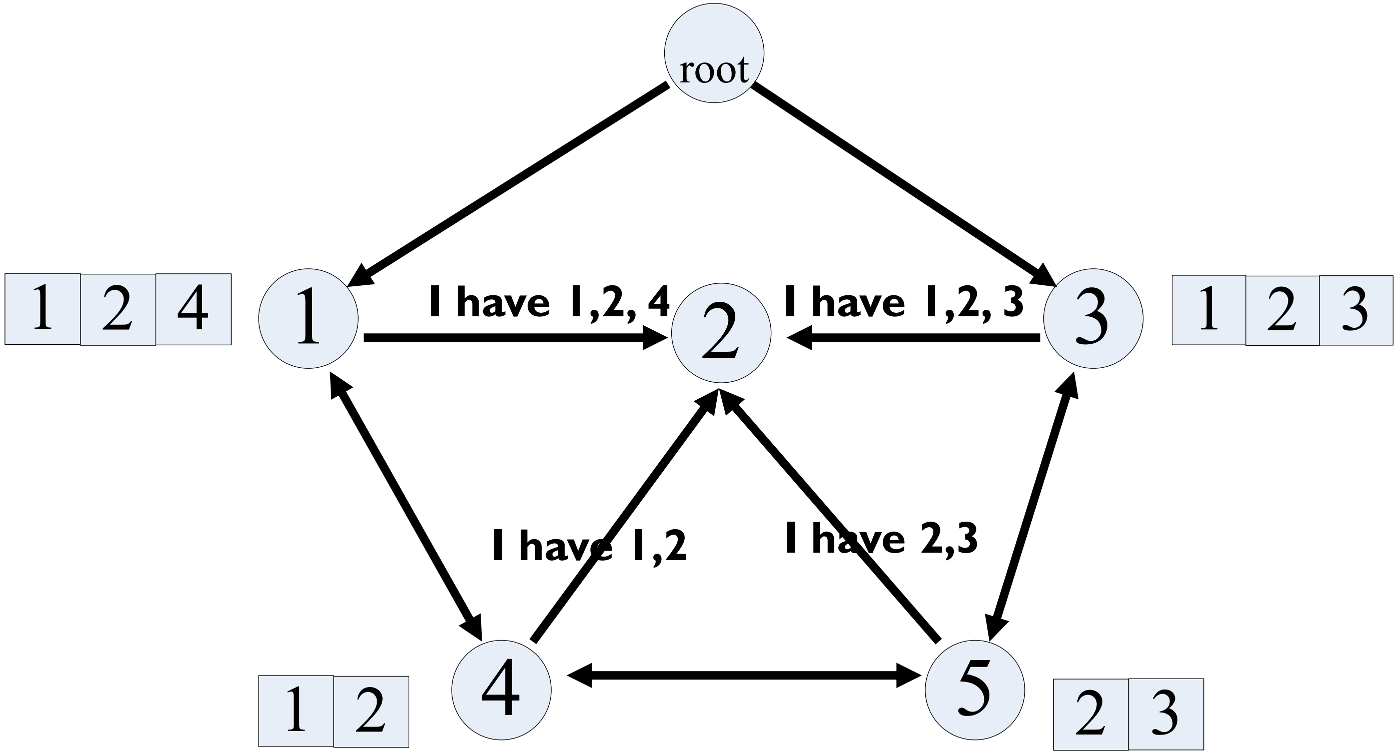
Buffer Map Representation and Exchange

- Divide video stream into segments
- Buffer Map represent their availability
- Playback progresses of the nodes are semi-synchronized
- Sliding window of 120-segment

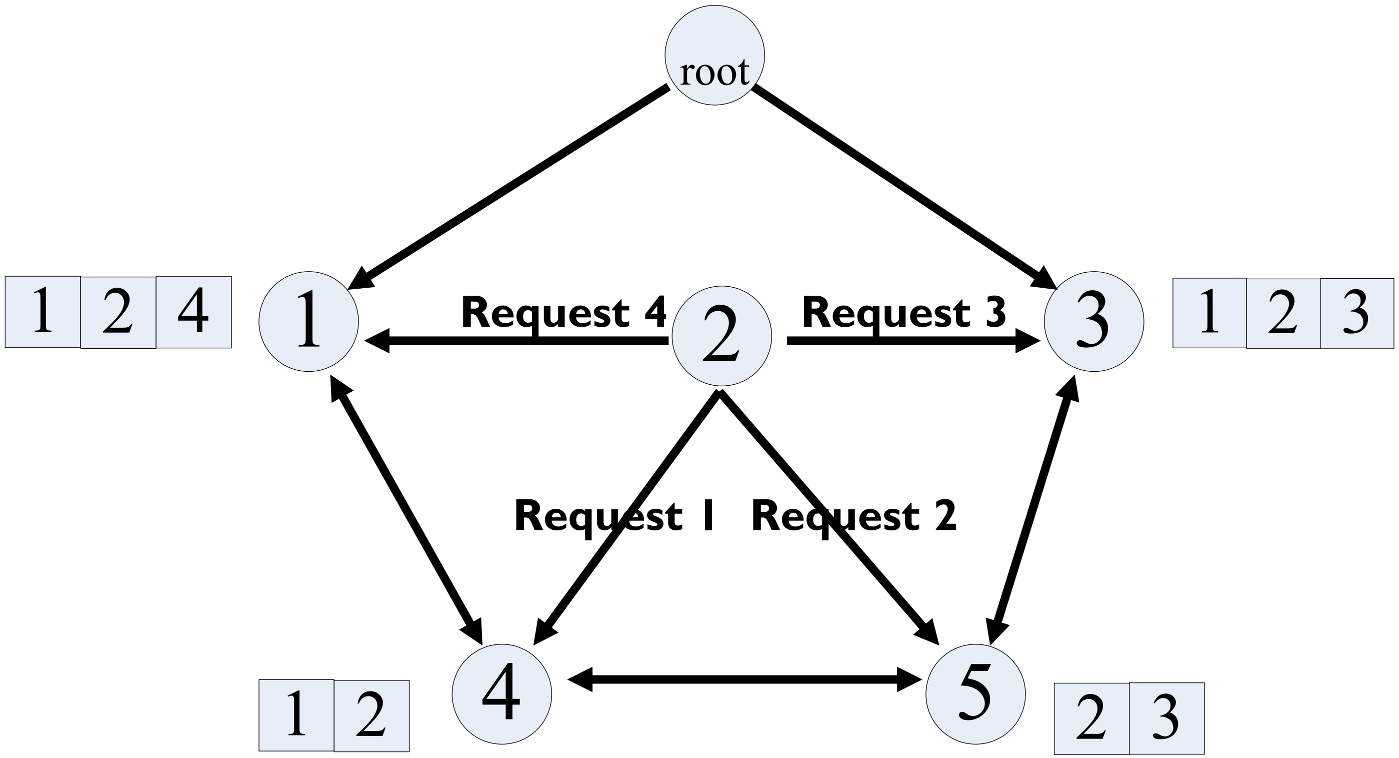
Buffer Map Representation and Exchange



Buffer Map Representation and Exchange



Buffer Map Representation and Exchange



Scheduling Algorithm

- round-robin scheduler
  - Good for homogenous and static network
- RR inadequate in Reality
  - playback deadline for each segment
  - heterogeneous streaming bandwidth from the partners
  - Parallel machine scheduling, NP-hard!
- Smarter Scheduler 15ms / execution
  - Sort segments by the number of potential suppliers
  - Select rarest from node with highest bandwidth and enough available time
- Origin node advertises conservative BM if needed

Failure Recovery and Partnership Refinement

- Departure can be detected after an idle time of TFRC or BM exchange
- an affected node can quickly react through re-scheduling using the BM information of the remaining partners if the probability of concurrent departures is small
- each node periodically establishes new partnership
  - maintain a stable number of partners
  - better quality
  - Reject the one with the lowest score $\max\{\bar{s}_{i,j}, \bar{s}_{j,i}\}$

Analysis of Overlay Radius

- overlay radius d $< \log_{M-1} N + 3.$
- As an example, for a DONet of 500 nodes and M = 4, almost 95% of the nodes can be reached within 6 hops.

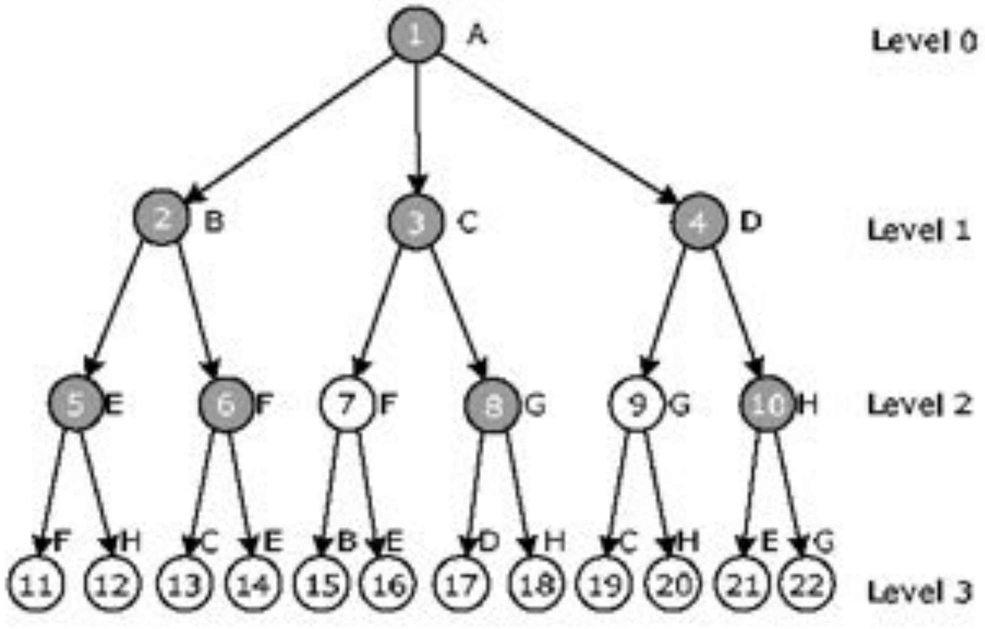


Fig. 4. A Breath-First Search (BFS) tree till level 3. Dark nodes: $\delta(t) = 1$ (first appearance); White nodes: $\delta(t) = 0$.

Performance under Stable Environment

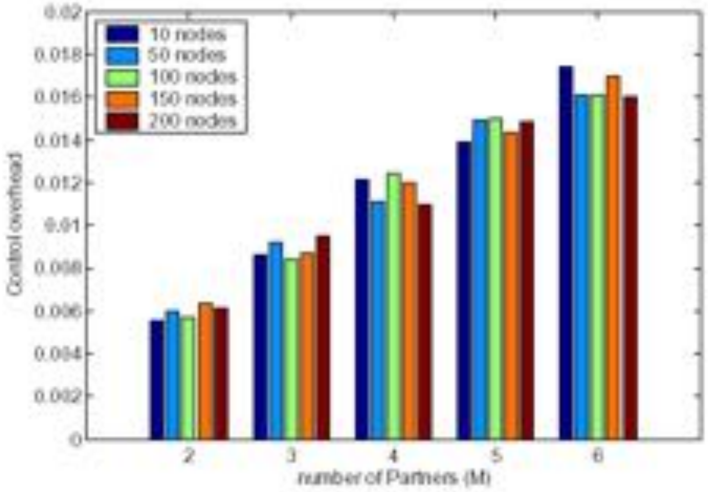


Fig. 7. Control overhead as a function of the number of partners for different overlay sizes. (Control overhead= Control traffic volume/Video traffic volume at each node).

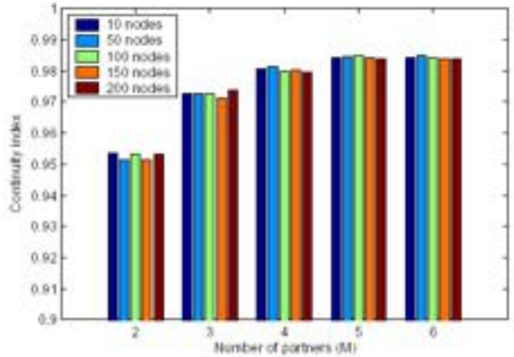


Fig. 8. Continuity index as a function of the number of partners.

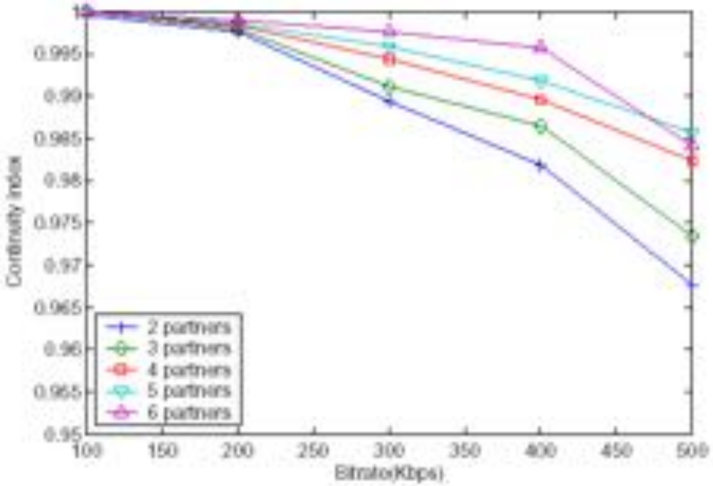


Fig. 9. Continuity index as a function of the streaming rate. Overlay size = 200 nodes.

Performance under Dynamic Environment

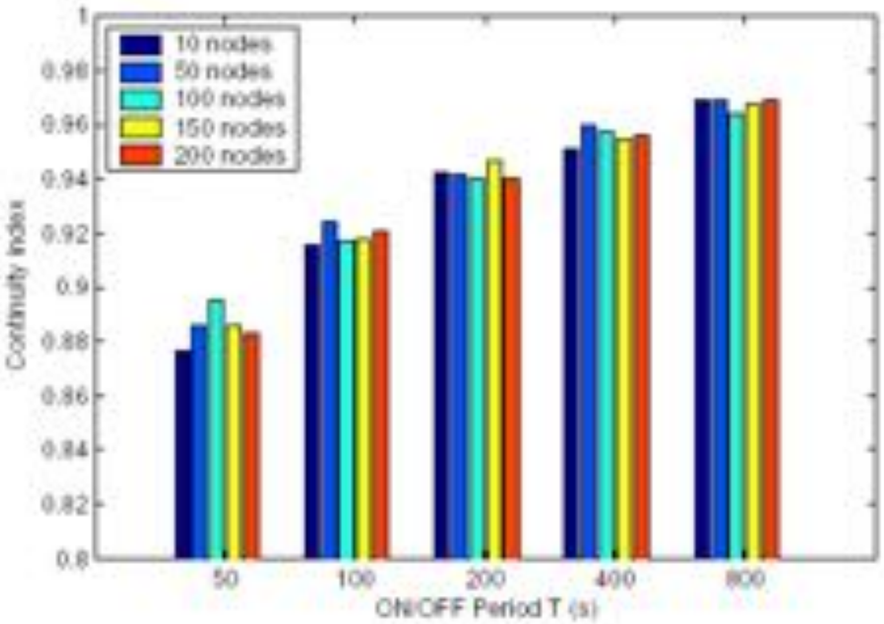


Fig. 11. Continuity index as a function of the average ON/OFF period for different overlay sizes.

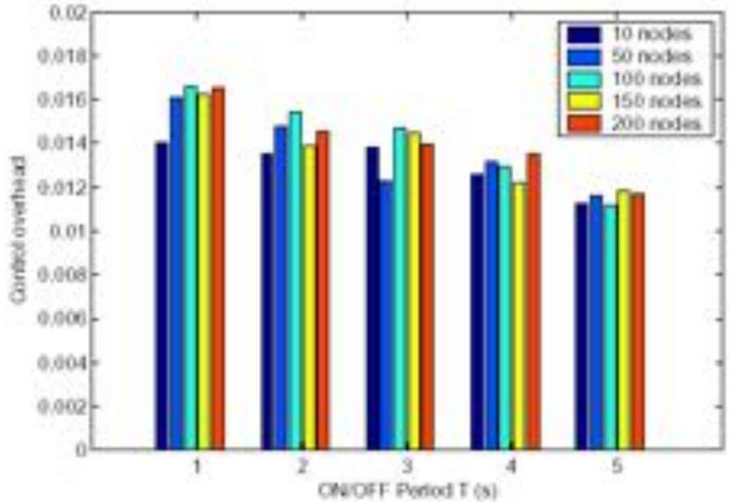


Fig. 10. Control overhead as a function of the average ON/OFF period for different overlay sizes.

Comparison with Tree-based Overlay

- Single Tree: no splitstream!!!
- 3 children per node, except source with 4
- Yields same degree as M=4
- Some children moved down one level until bw constraint satisfied
- Tree repair grafting nodes to upstream neighbor when parent fails

Comparison with Tree-based Overlay

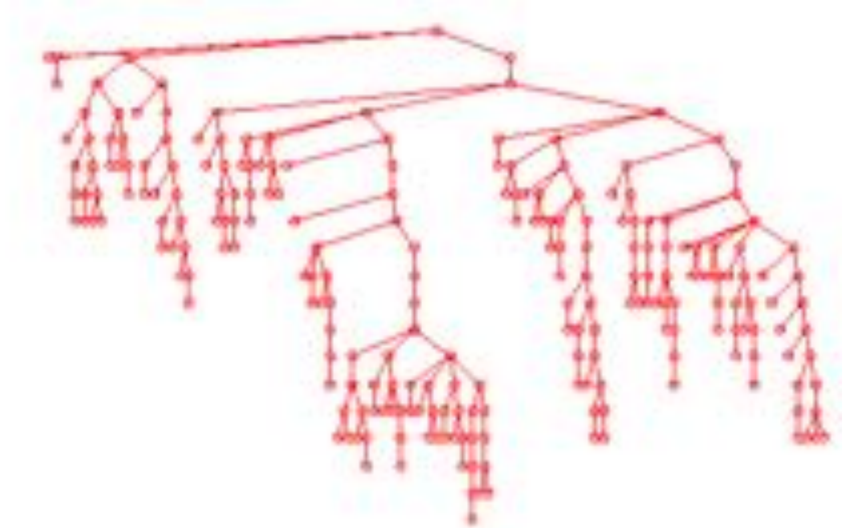


Fig. 12. A snapshot of a tree-based overlay with 231 nodes.

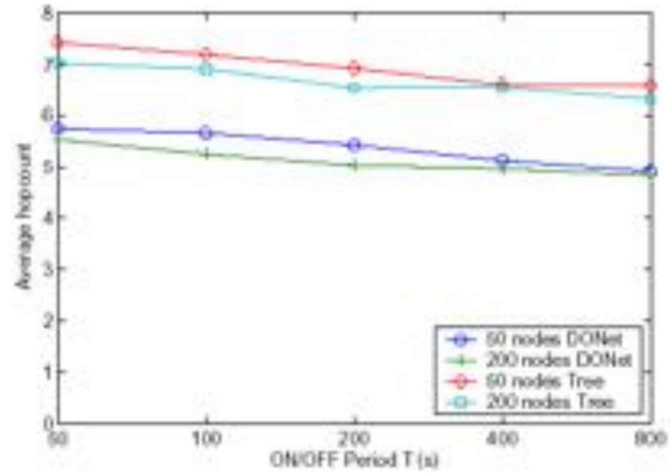


Fig. 13. Average overly hop-count of DONet and tree-based overlay.

Comparison with Tree-based Overlay

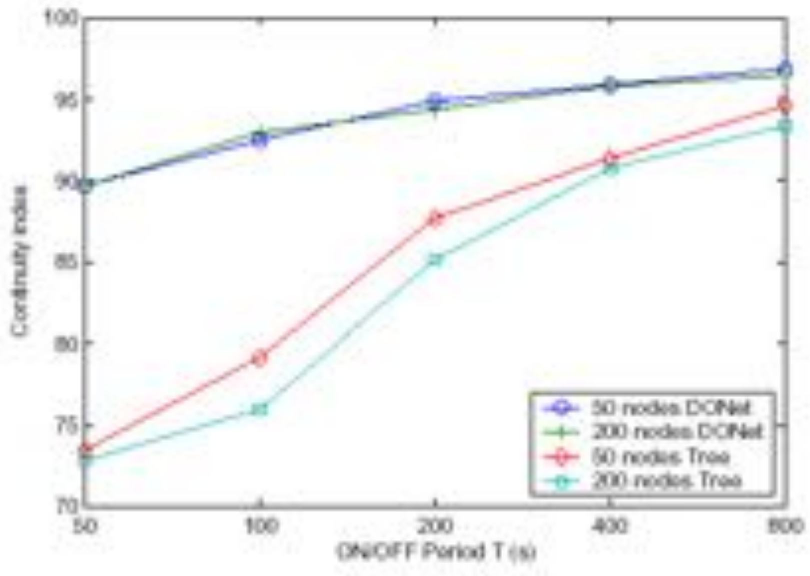


Fig. 14. Comparison of continuity indices for DONet and the tree-based overlay. (Y-axis: index; X-axis : ON/OFF period ; for overlay size of 50 and 200).

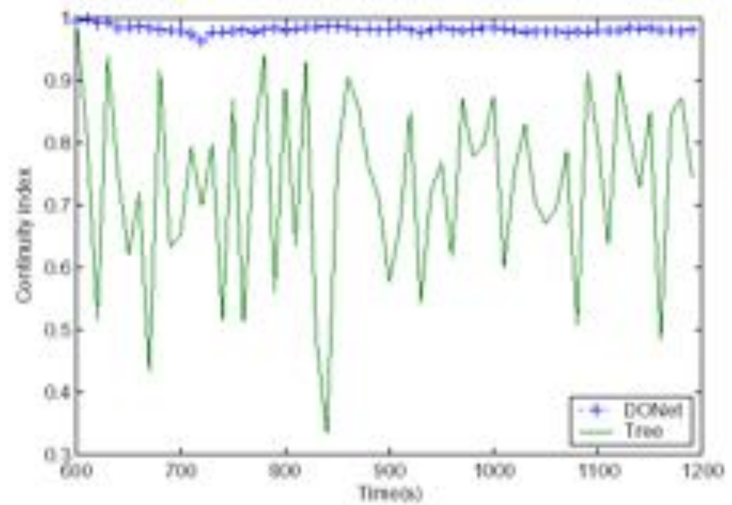


Fig. 15. Samples of continuity indices for DONet and a tree-based overlay in a experiment (from 10 min to 20 min).

A Practical DONet Implement

- Broadcast live sports programs (450 - 755Kbps RealVideo/Windows Media format)

TABLE I

USER IP DISTRIBUTION OF COOLSTREAMING V.0.9. (APPROXIMATION)

Time	Total	CN	HK	US	Other
June 17	1500	300	400	250	50
June 22	2400	400	1000	900	100
June 25	3000	600	1300	1000	100
June 27	4000	1000	1500	1400	100

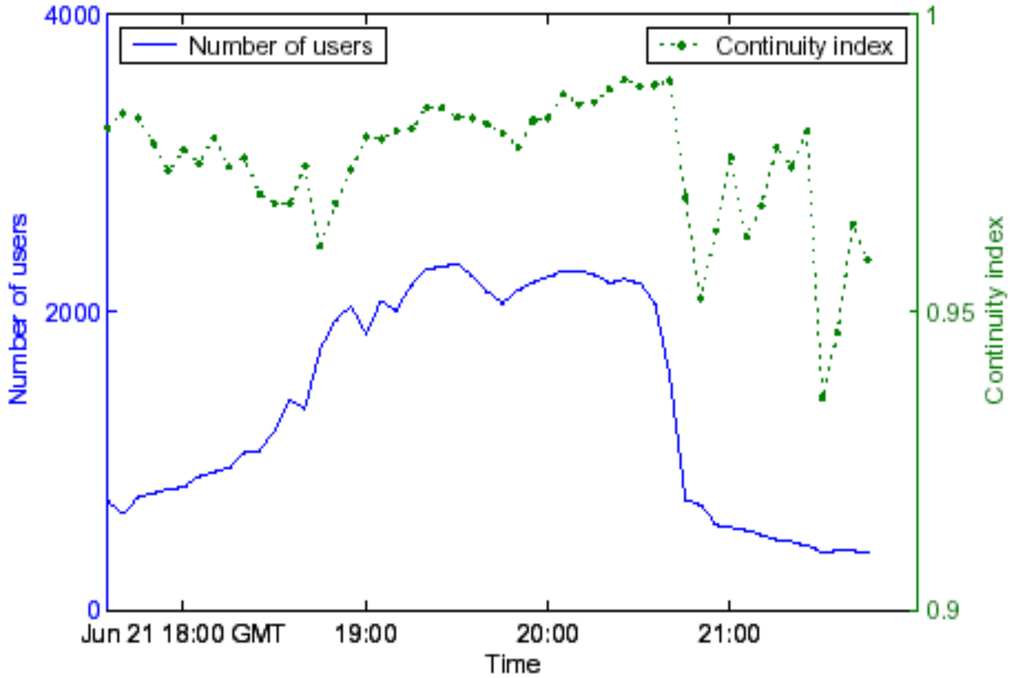


Fig. 17. Number of users and Continuity index over time.

iGridMedia

iGridMedia

Pull-based protocols are effective

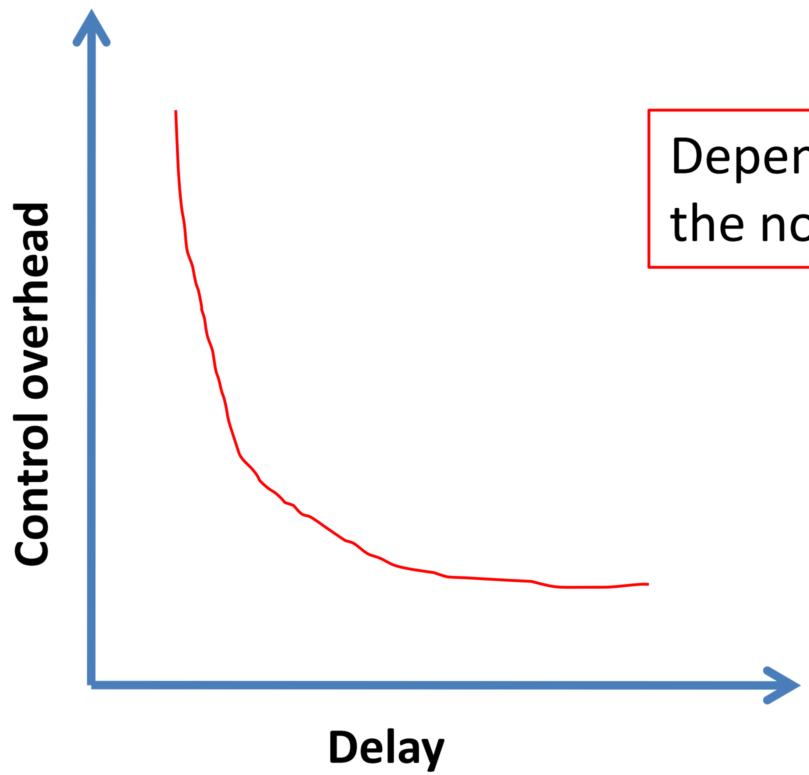
- Select neighbors from unstructured overlay
- Periodically notify neighbors of available packets
- Neighboring nodes request packets

Nearly optimal

- bandwidth utilization
- Throughput

Without intelligent scheduling and bw measurement

Tradeoff



Depends on how frequently the notifications are sent.

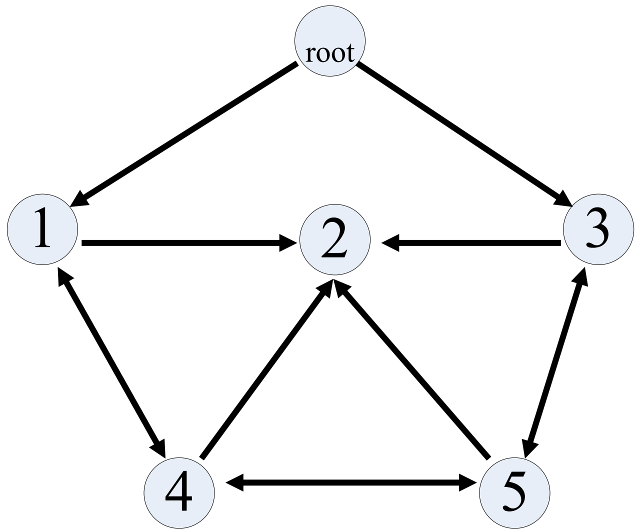
Overlay Construction

Contact rendezvous point

Randomly find set of partners

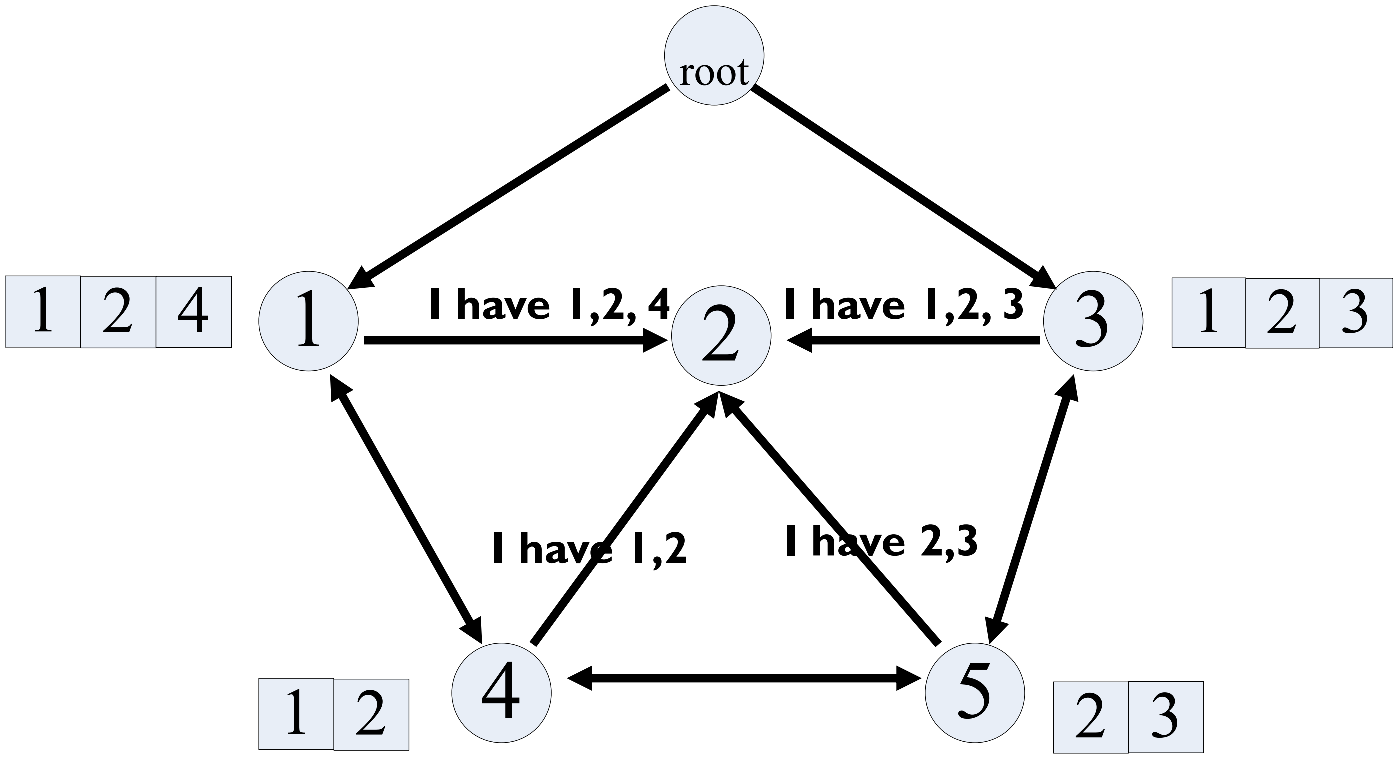
- RPS can be used

Build (static) random graph



Push/Pull Method

- Pull Part



Pull-Push method

Split stream as in Splitstream

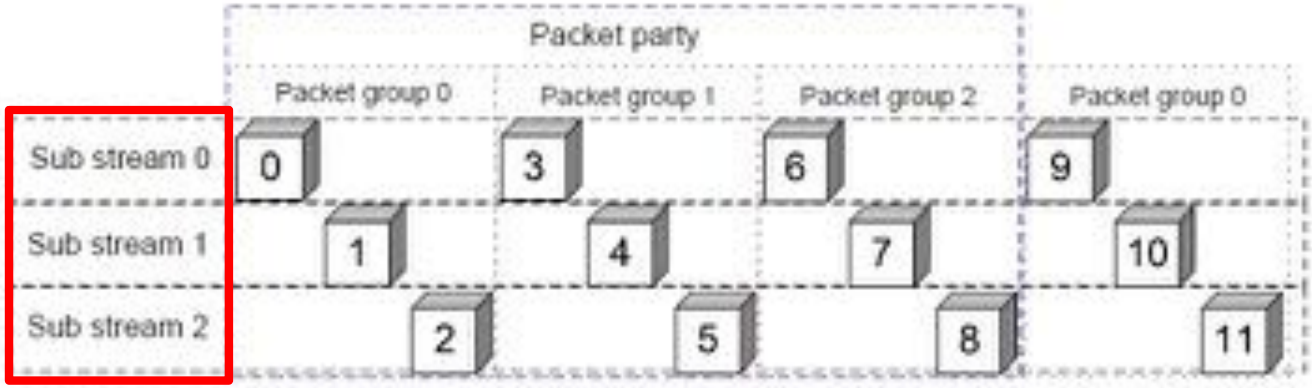


Fig. 18

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

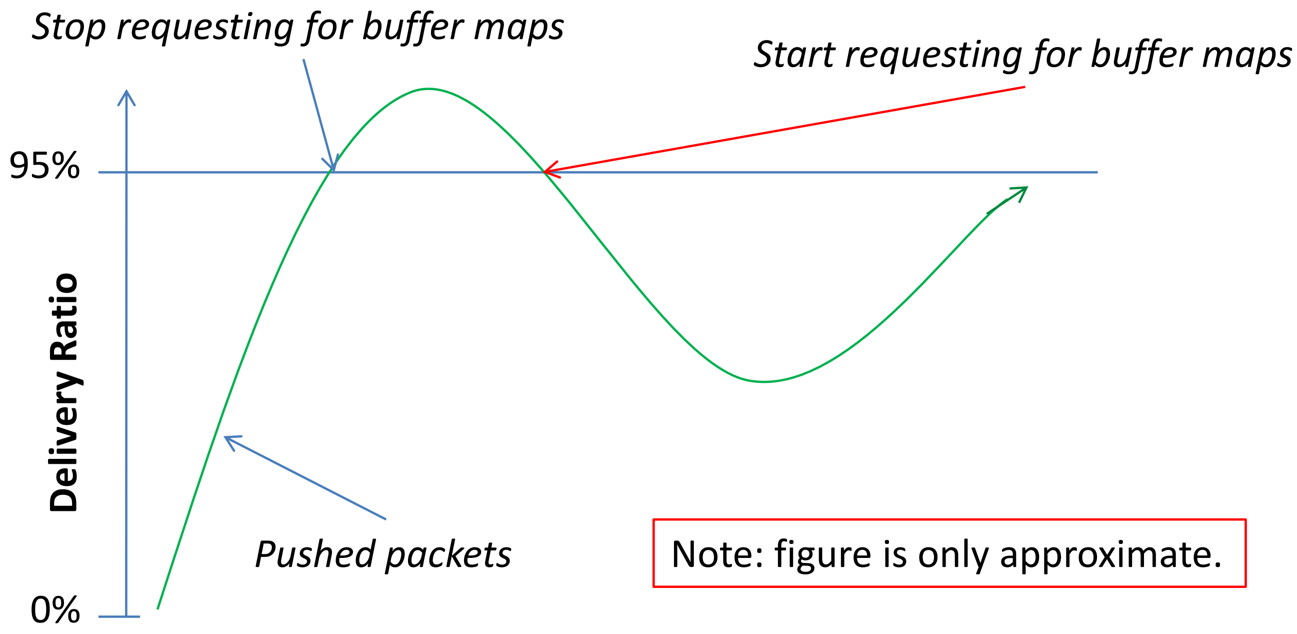
- Send subscription for corresponding substream

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

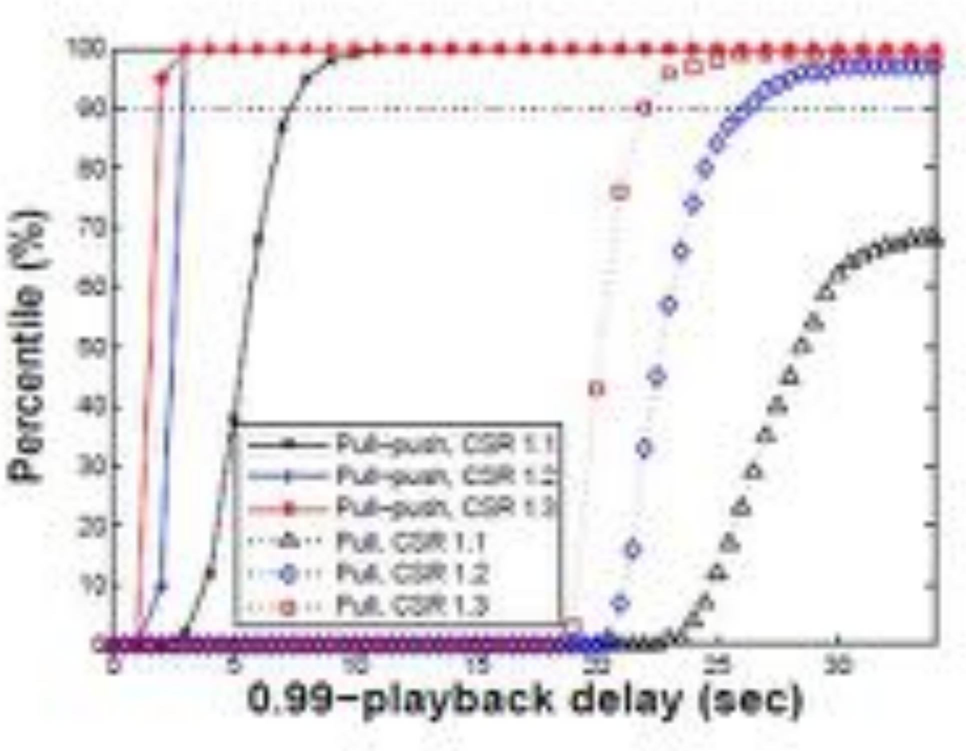


Fig. 20

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

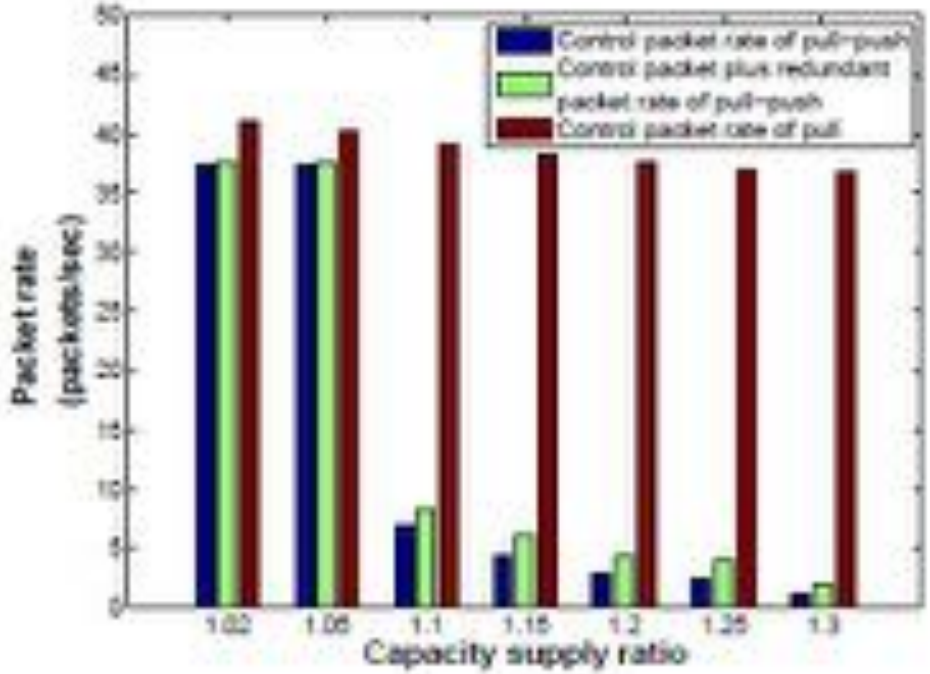
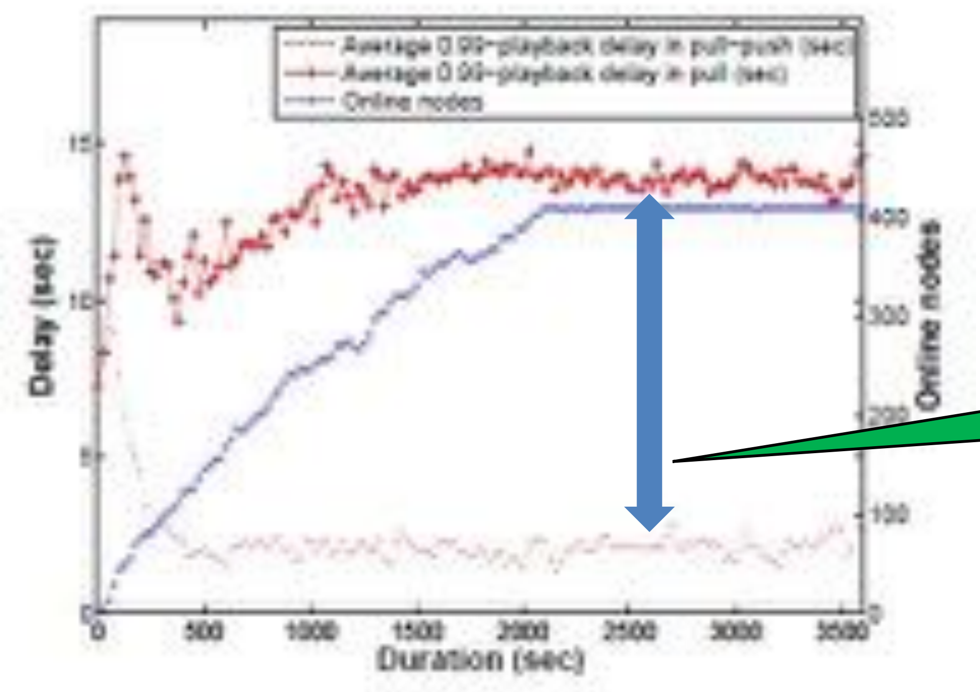


Fig. 21

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.