

Is there a future for mesh-based live video streaming?

Fabio Picconi

Laurent Massoulié

Thomson Research Lab
Paris, France

P2P'08 – Sep 11, 2008

P2P livestreaming today

- **many widely deployed systems**
 - PPLIVE, SopCast, TVants, Joost, Zattoo, etc.
- **but...**
 - most streams ~ 400 kbps, a few ~ 800 kbps
- **many mesh algorithms suffer from content bottleneck**
 - 2002 Gnutella study showed mean uplink capacity ~ 1.1 Mbps [Saroiu et al., 2002]
 - efficient systems should support ~ 1 Mbps streams

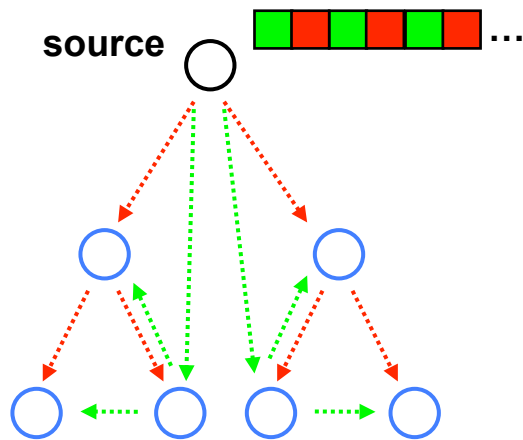
need for efficient P2P streaming algorithms

Outline

- **existing P2P streaming algorithms**
 - tree-based, mesh-based
 - rate optimality
- **mesh-based prototype**
- **evaluation**
- **conclusions**

Streaming algorithms

multi-tree/pull-push



low signaling overhead
low delay



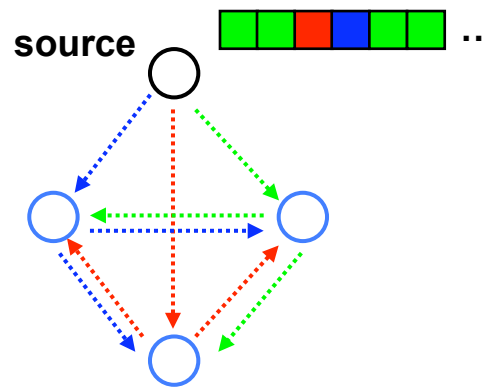
tree maintenance
peer churn

Splitstream [2003]
Chunkyspread [2006]
Coolstreaming* [2007]
GridMedia* [2007]

4

* pull-push

N trees of depth 2

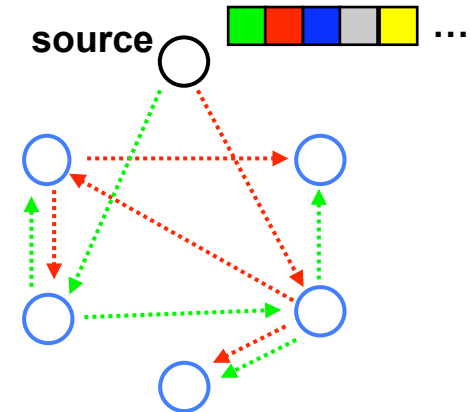


high rates
no neighbor selection

not scalable
slow peers increase delay

AQCS [2007]

mesh



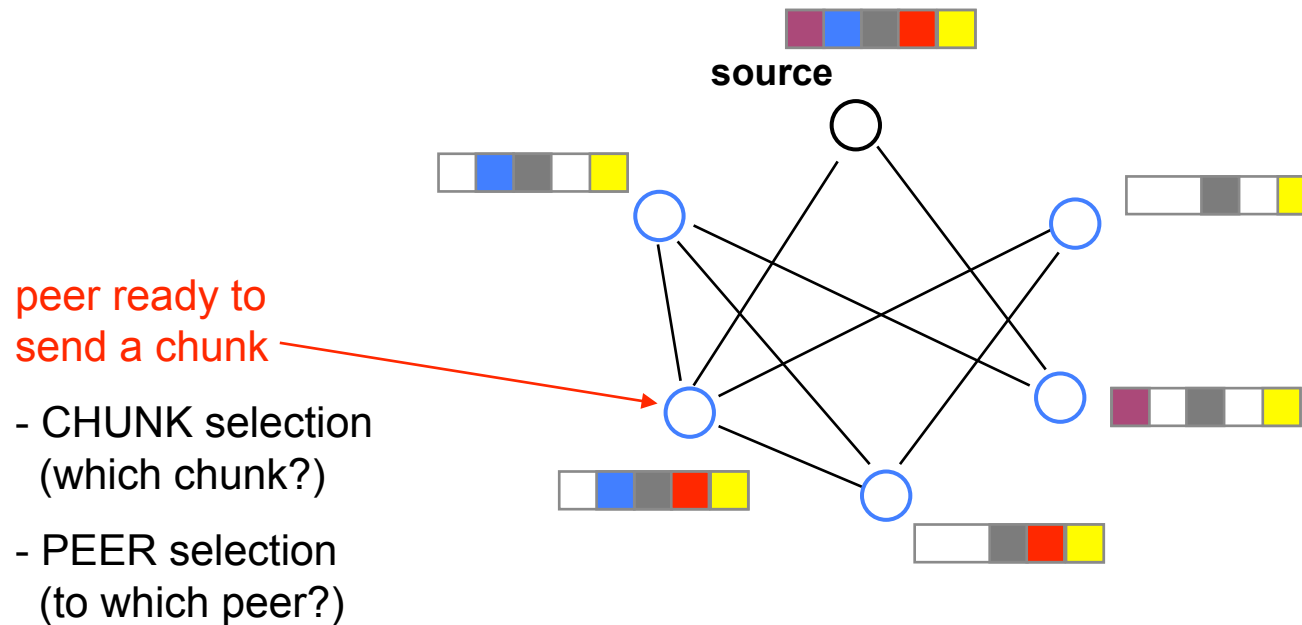
epidemic diffusion
churn tolerant

signaling overhead
choice of scheduling alg.

Chainsaw [2005]
Coolstreaming [2005]
PULSE [2007]
DP/RU [2007]

Mesh algorithms

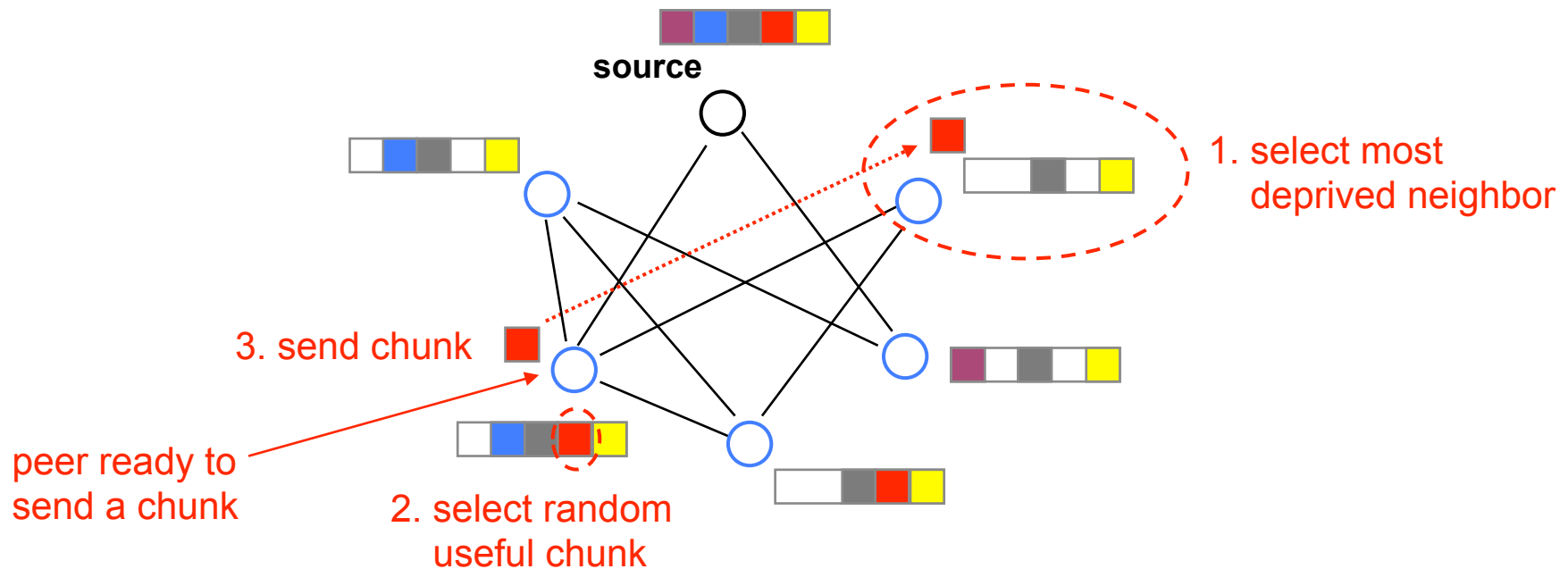
- each chunk transfer requires a local scheduling decision



- CHUNK selection (which chunk?)
 - PEER selection (to which peer?)
- select CHUNK first, then PEER ?
select PEER first, then CHUNK ?
at random?
chunk closest to deadline? furthest from deadline?

Mesh algorithms (cont'd)

- Example: Most **D**epriver **P**eer, **R**andom **U**seful Chunk (DP/RU) [Massoulié et al., INFOCOM 2007]



Efficiency: rate and delay

- **maximum achievable streaming rate**

$$r_{\max} = \min \left\{ u_s, \frac{u_s + \sum_{i=1}^N u_i}{N} \right\}$$

u_s : upload capacity of source
 u_i : upload capacity of peer i

- **in theory, rate optimality proven for**

- AQCS (N trees) [Guo et al., Tech.Rep. 2007]
- DP/RU (mesh) [Massoulié et al., INFOCOM 2007]

- **in practice, Planetlab studies show**

	delivery rate [% r_{\max}]	delay [sec]	
AQCS (N trees)	~ 90%	15	[Liang, ICDCS'08]
DP/RU (mesh)	~ 80%	30	[Liang, ICDCS'08]
GridMedia (pull-push)	~ 80%	2	[Zhang, JSAC Dec'07]

should we prefer tree-based to mesh-based?

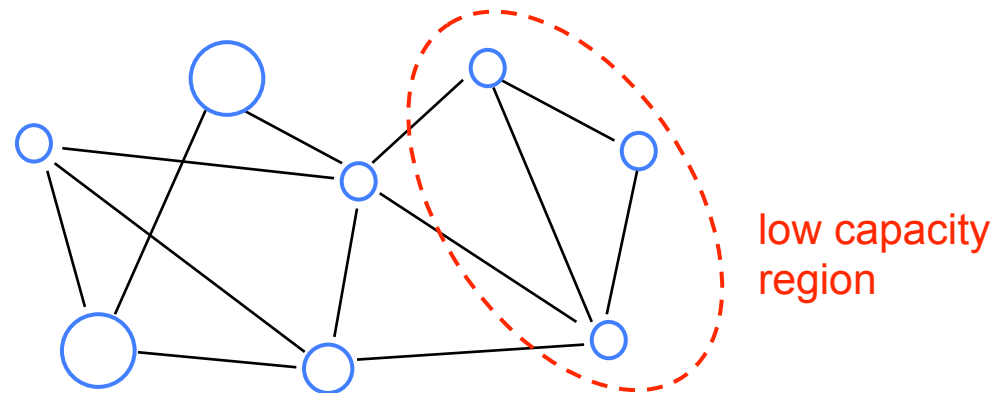
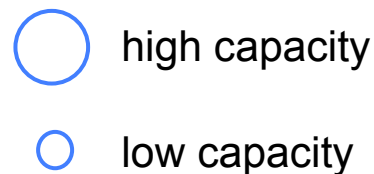
Is there a future for mesh?

- **developed a mesh-based prototype**
- **applied several optimizations to maximize performance**
 - scheduling
 - DP/LU (similar to DP/RU, but sends *latest* instead of *random* chunk)
 - simulations show DP/LU has lower delay [Bonald et al., 2008]
 - neighbor selection algorithm
 - distributes high bandwidth peers uniformly across overlay
 - source push strategy
 - gives preference to fast nodes

Prototype

- **neighbor selection**

- fixed maximum number of neighbors
- average neighborhood upload capacity U_N
- choose neighbors such that U_N is close to a reference value r
 - if $U_N < 0.9 r$ → peer replaces slowest neighbor
 - if $U_N > 1.1 r$ → peer replaces fastest neighbor
- $r =$ stream rate



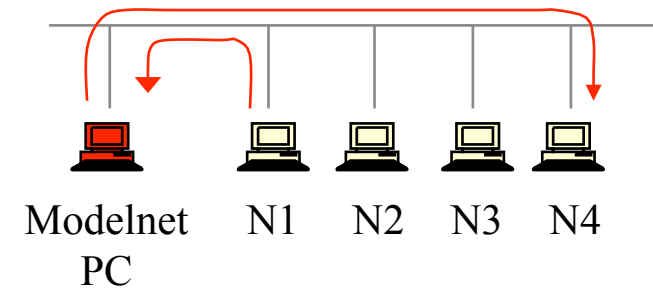
Prototype (cont'd)

- **source strategy**
 - peer receives number of chunks proportional to its uplink capacity
 - motivated by intuition and analysis of delay
- **implementation: peers declare their upload capacity**
 - assumption: bottleneck at access links
 - advantage: fast (optimistic) overlay rewiring
 - actual download bandwidth can be verified later

Evaluation

- **use network emulation**

- deploy in local cluster
- run 10-20 clients per physical machine
- route traffic through Modelnet PC



- **configuration**

- client upstream capacities: {4000, 1000, 384, 128} kbps (same as Liang et al., ICDCS'08)
- source upstream capacity: 1.1 Mbps
- overlay size: 80 to 640 clients
- maximum neighbors: 20
- download window (delay): 15-30 seconds

Delivery rate

Configuration

overlay size: 80 clients

$r_{max} = 1040$ kbps

5-minute experiment

- mean delivery rate:

99% @ $0.9 r_{max}$

95% @ r_{max}

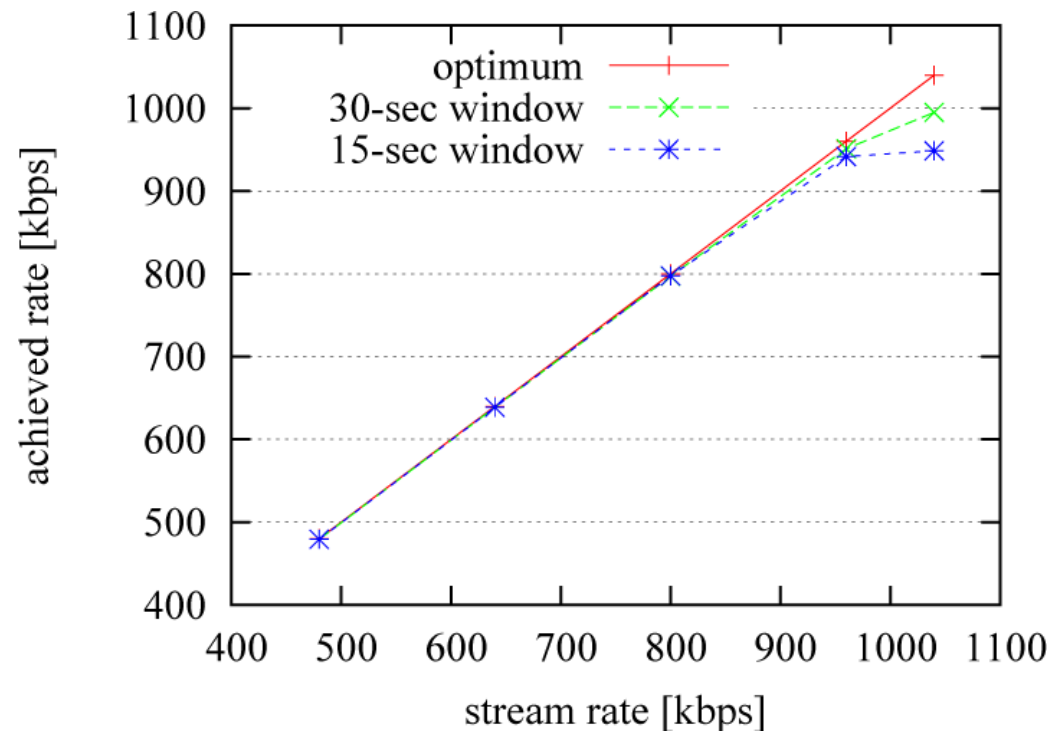
(30-sec window)

Indirect comparison

Our system: 99% @ $0.9 r_{max}$

AQCS*: 100% @ $0.9 r_{max}$

GridMedia: 99% @ $0.8 r_{max}$



*40-client experiment

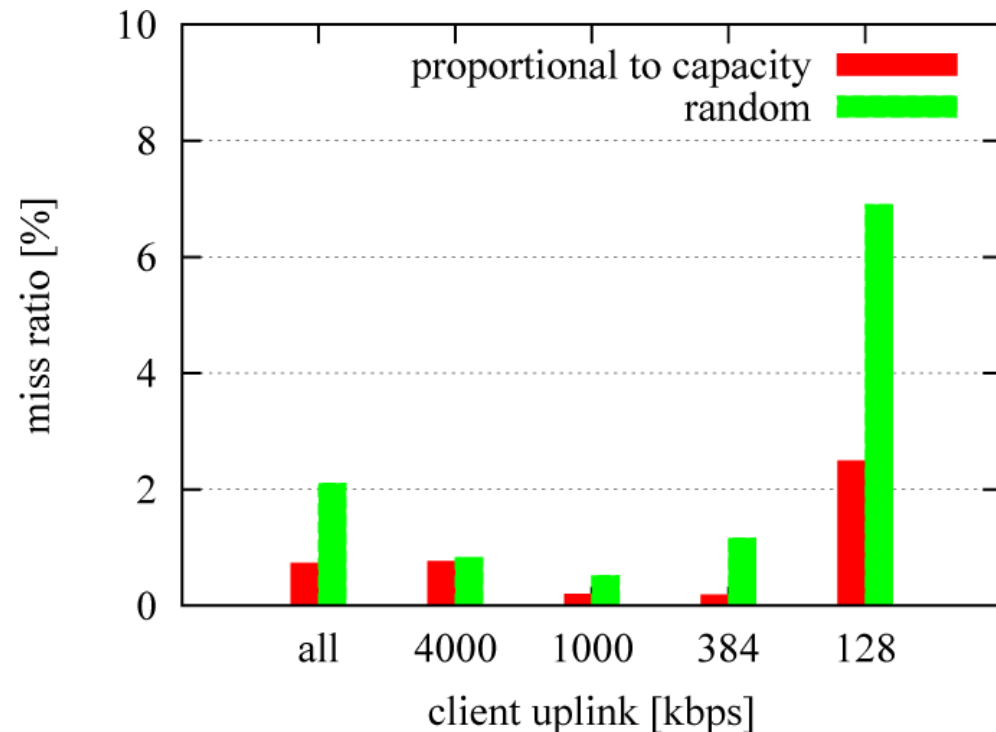
Source strategy

Configuration

$r = 960$ kbps ($\sim 0.9 r_{max}$)

30-sec window

- capacity-aware source reduces misses by factor of 3
- consistent with analysis of propagation delay



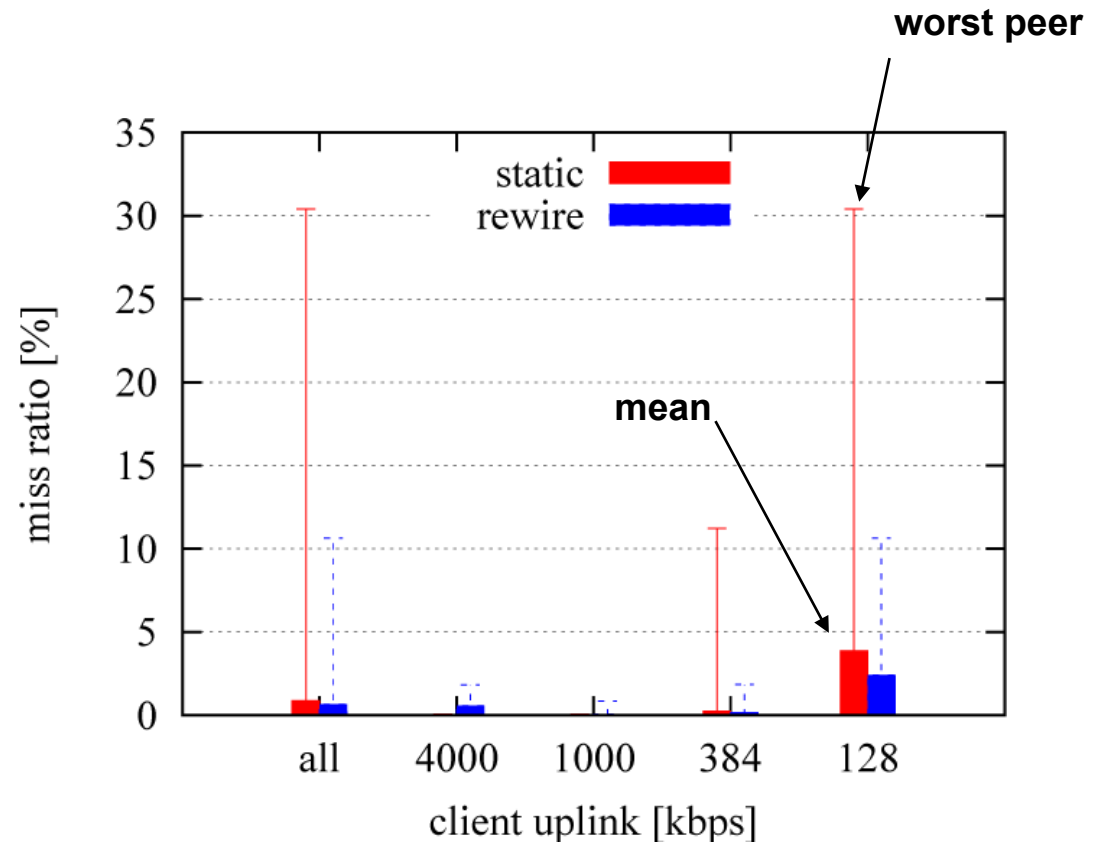
Neighbor selection

Configuration

$r = 960$ kbps

30-sec window

- static overlay yields good mean performance
- neighbor selection reduces worst-case miss ratio



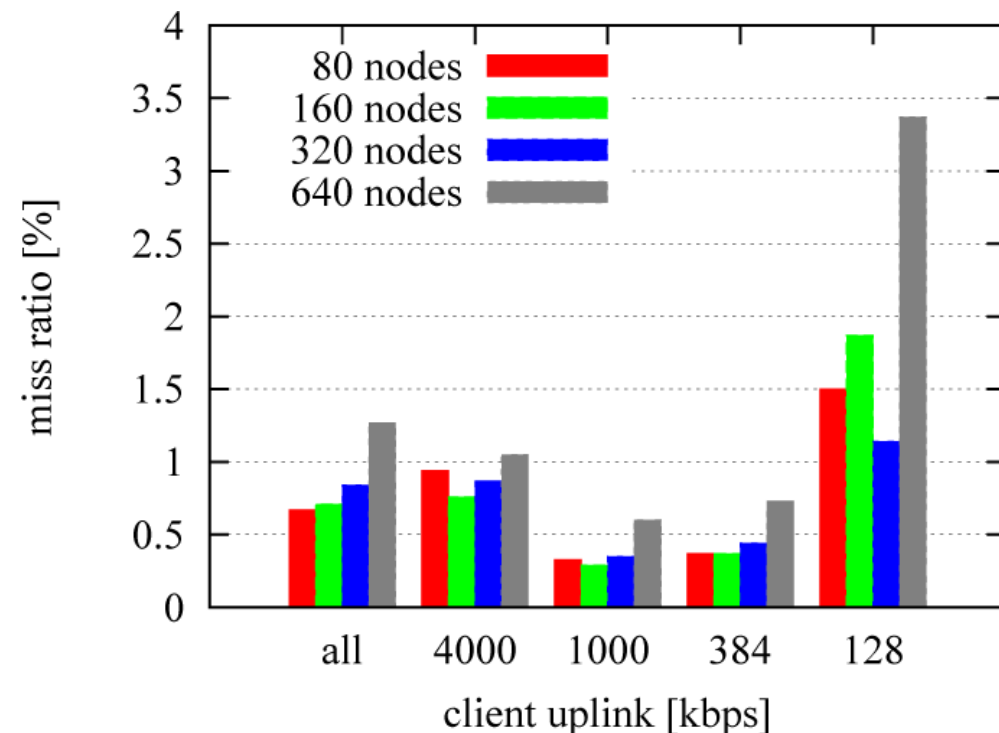
Scalability

Configuration

$r = 960$ kbps

30-sec window

- 8-fold increase in overlay size produces small increase in miss ratio
- shows that delay grows slowly with overlay size



Churn tolerance

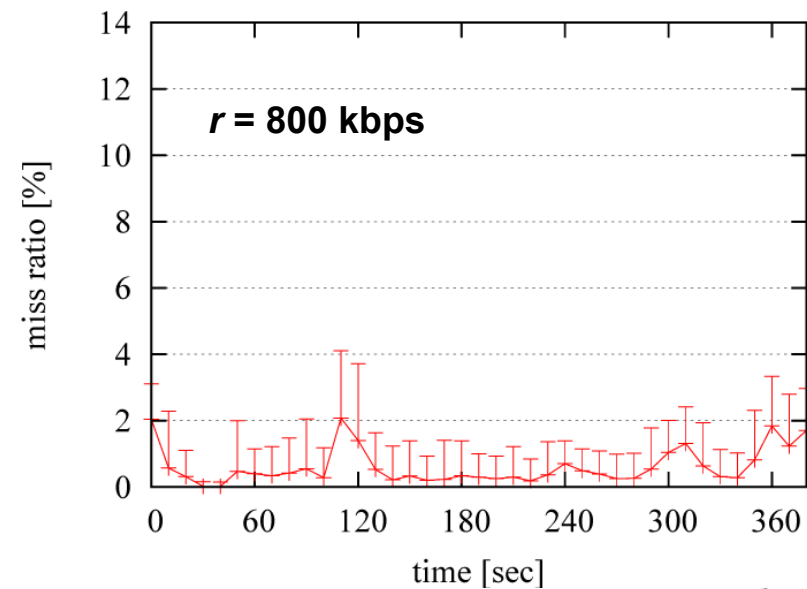
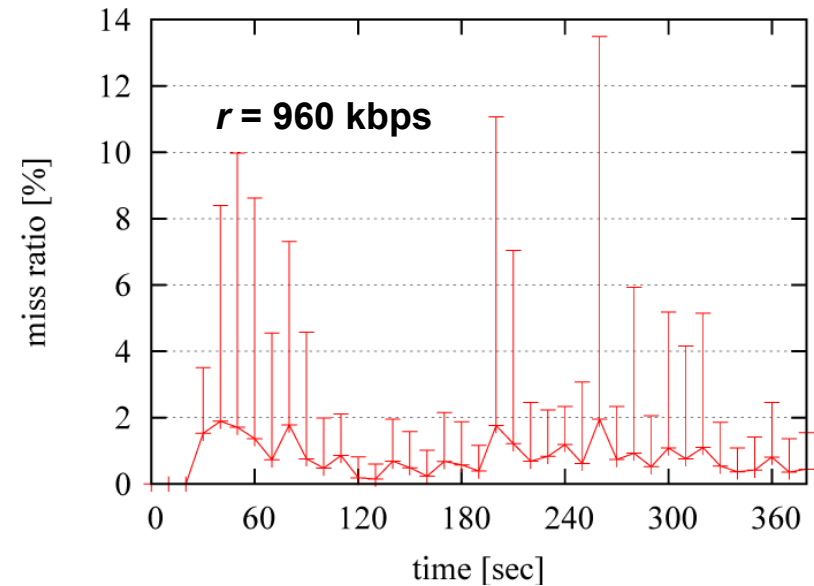
Configuration

overlay size: 80 clients

$r = 800\text{-}960$ kbps

one leave+join event every 10 sec

- average miss ratio stays below 2%
- lower stddev peaks for 800 kbps thanks to some excess upload capacity



Conclusions

- **mesh systems can achieve near-optimal rates in practice**
 - negligible chunk misses (1%) at 90% max stream rate
 - comparable or better rate than other tree-based systems (AQCS, GridMedia)
- **mesh is still an attractive choice thanks to:**
 - high rates
 - simple unstructured overlay (no tree-like structure)
 - scalability (delays grows slowly with overlay size)
 - churn-tolerance
- **disadvantage**
 - delay still higher than some tree systems (GridMedia)

Future work

- **delay**
 - analysis of optimal delay suggests further optimizations possible
 - integrate bandwidth-aware scheduling
- **QoS**
 - source coding
 - integration with CDN
- **network awareness**
 - modify neighbor selection and scheduling algorithm to reduce backbone traffic

Questions ?