

Heterogeneous Gossip

Davide Frey

Rachid Guerraoui

Anne-Marie Kermarrec

Boris Koldehofe

Maxime Monod

Martin Mogensen

Vivien Quéma



Outline

- Context
 - Live Streaming
 - Gossip
 - Limitations
- Heterogeneous Gossip
 - Protocol
 - Evaluation
 - Conclusion

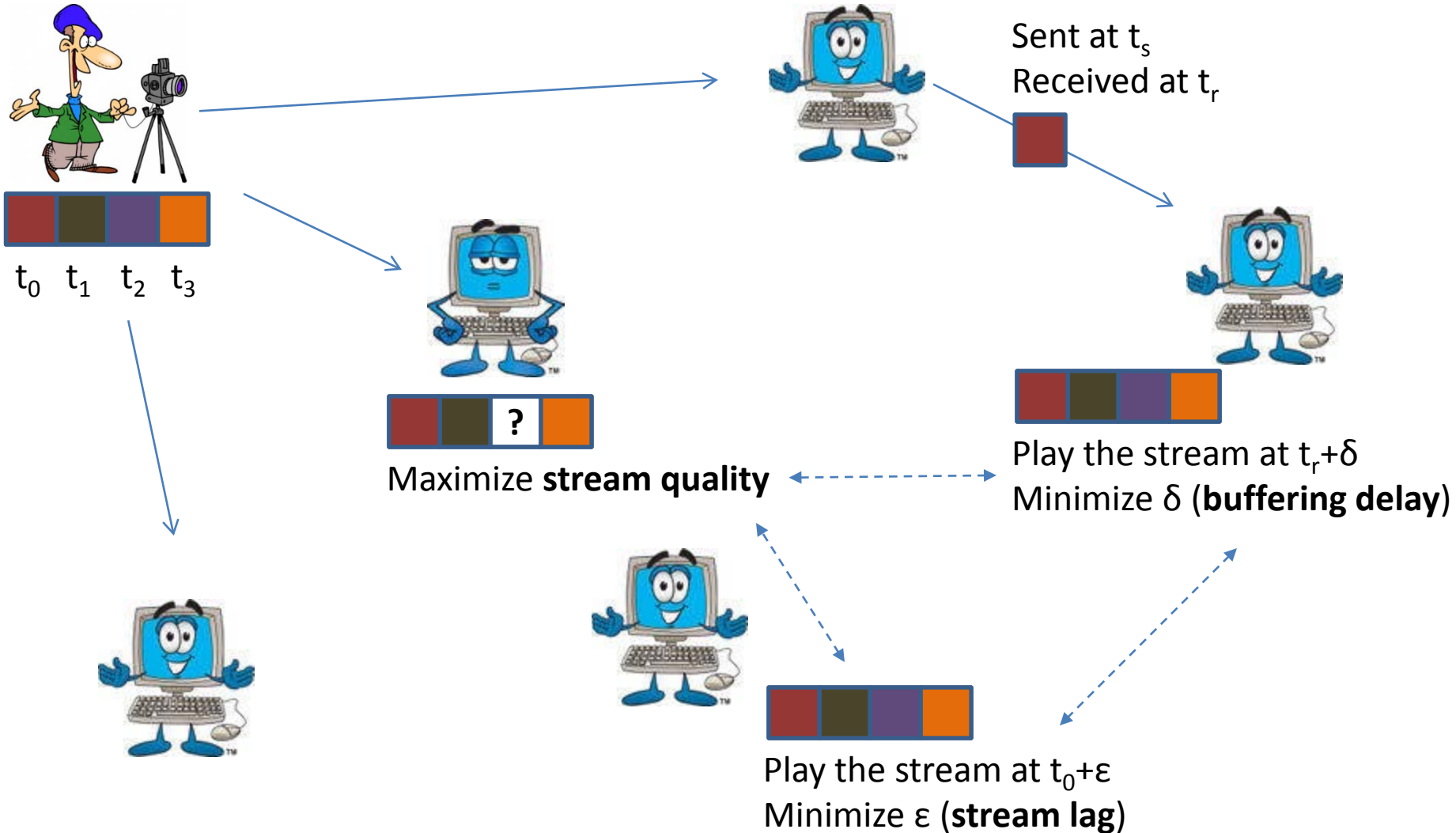


Context

- Large-scale (no IP-multicast)
- Churn/Failures
- Constrained, asymmetric and heterogeneous bandwidth

Target application: Live streaming

Live Streaming



Existing approaches

Structured overlay

Static overlay / **Reactive** repair

DHT-based systems

Trees

Multiple trees

Trees over mesh

Unstructured overlay

Dynamic overlay / **Proactive** repair

Mesh-based systems

Gossip

Existing approaches

Structured overlay

Static overlay / Reactive repair

SplitStream (2003)

Chunkyspread (2006)

Coolstreaming (2005)

Chainsaw (2005)

PULSE (2007)

PRIME (2007)

Coolstreaming (2007)

GridMedia (2007)

Unstructured overlay

Dynamic overlay / Proactive repair

BAR Gossip/FlightPath (2006/2008)

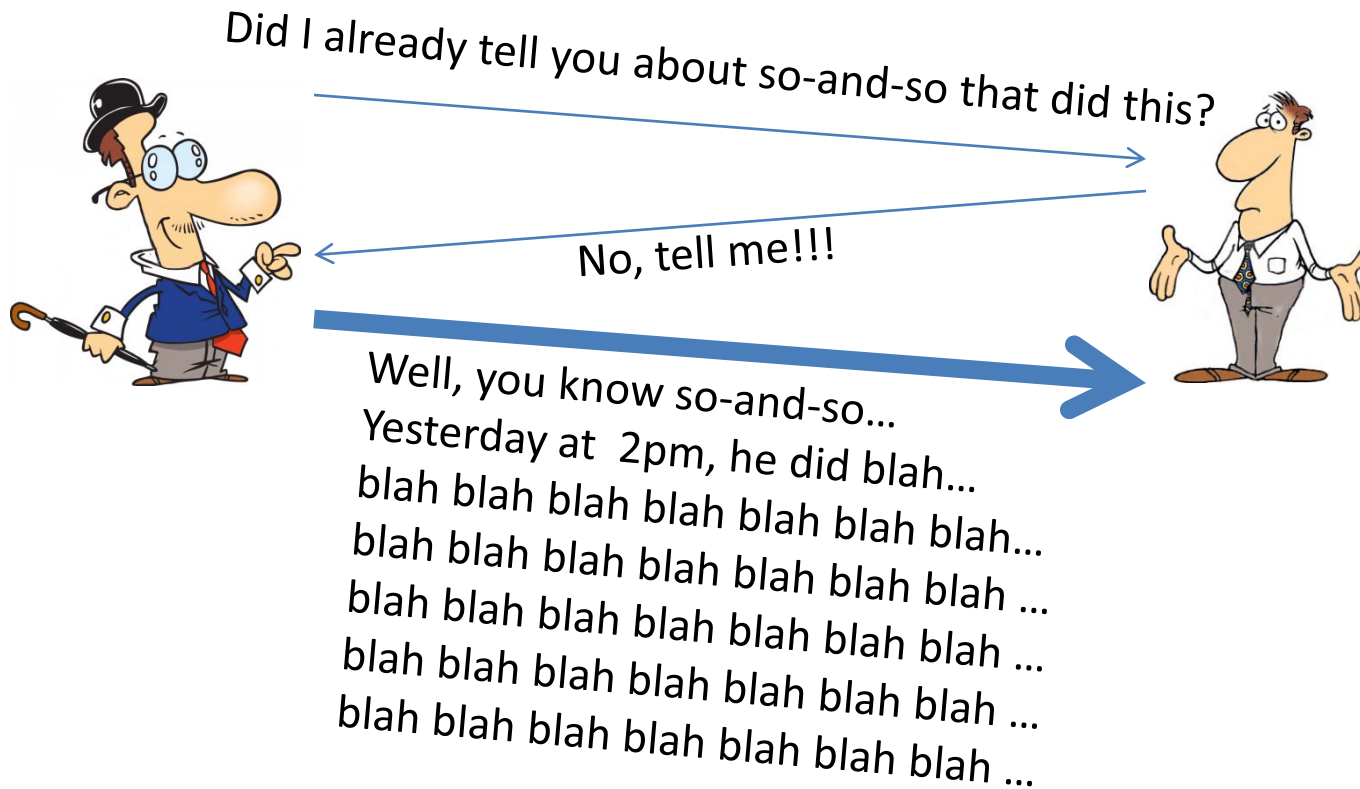
Gossip

Heterogeneity awareness?

Gossip in the real world

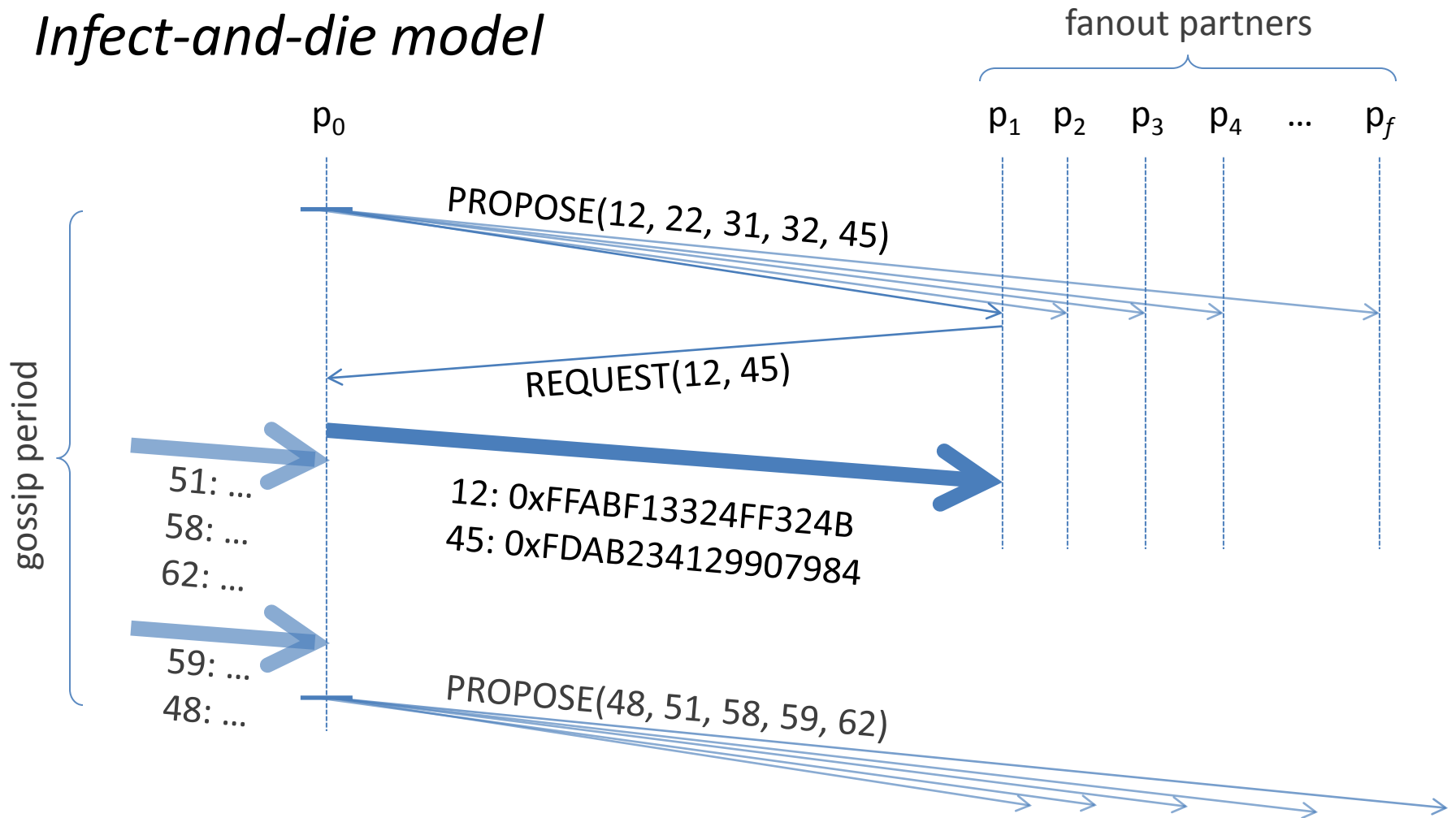


George meets Bob:



Gossip in computer science

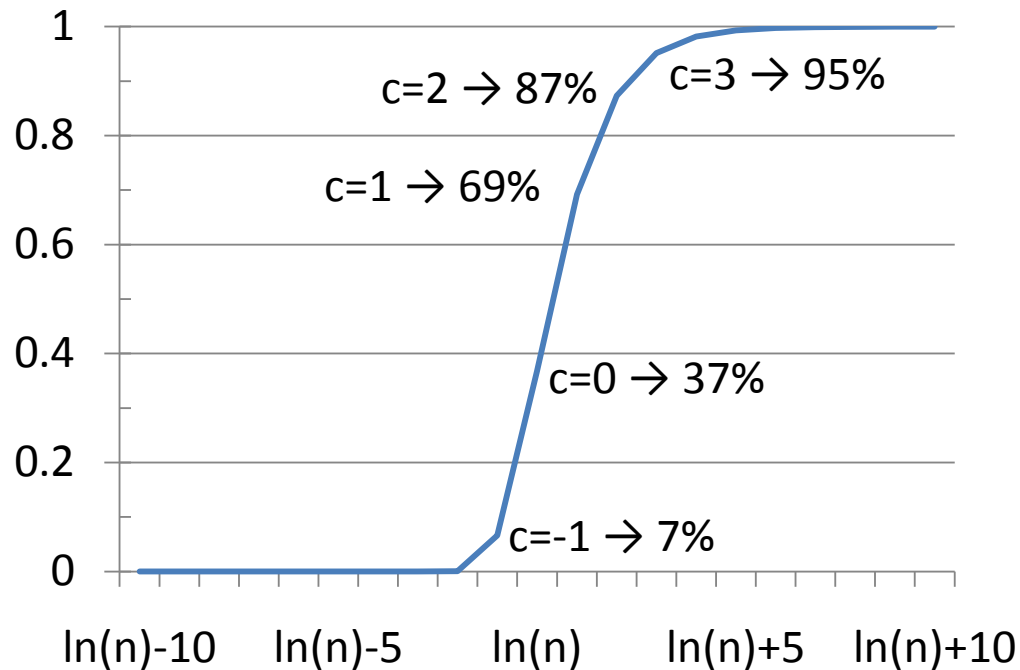
Infect-and-die model



Gossip - Theory

1. fanout = $\ln(n) + c$

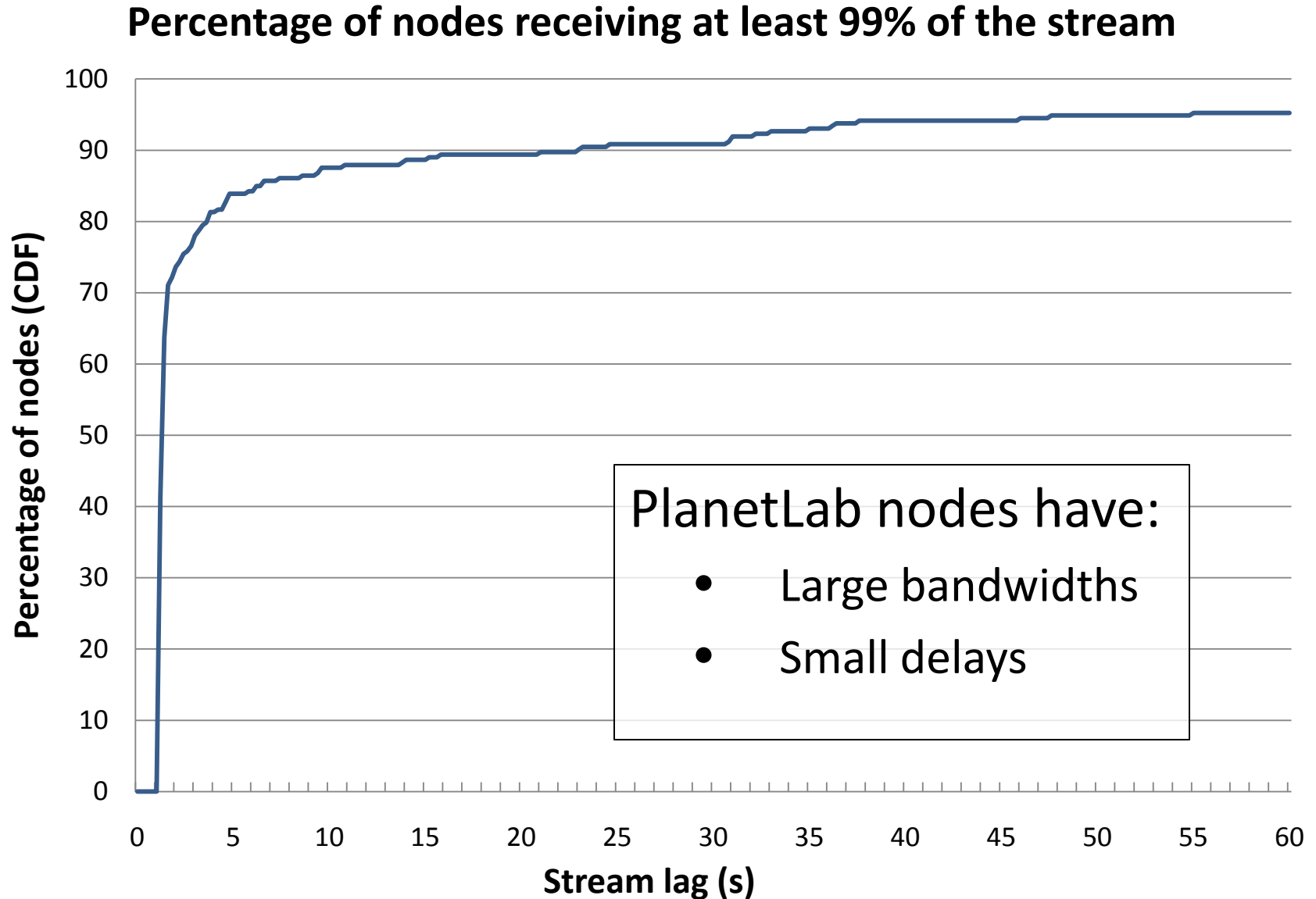
$P[\text{connected graph}]$ goes to $\exp(-\exp(-c))$



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

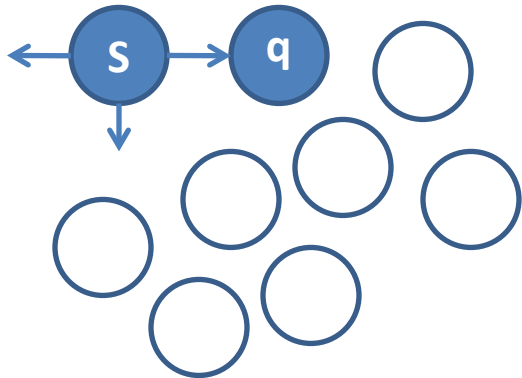
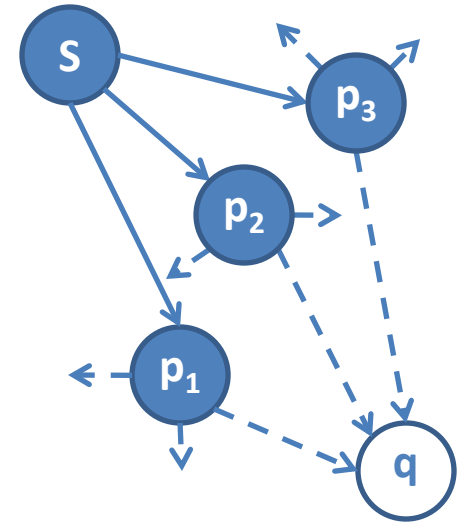
Gossip – Practice

(600kbps)

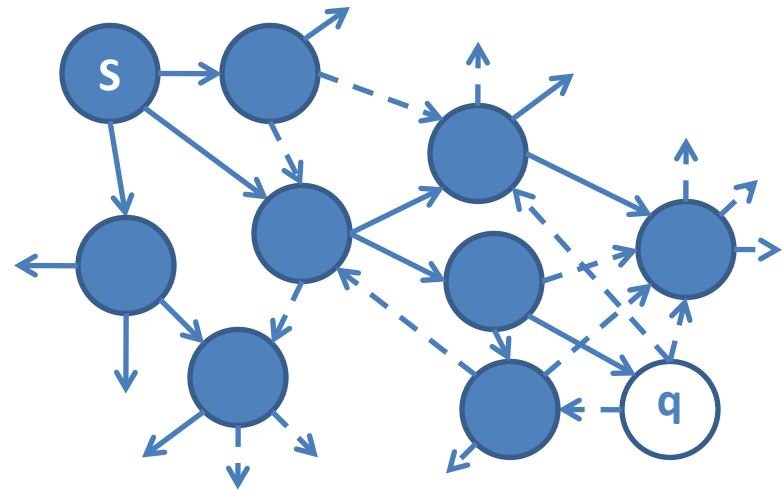


Gossip is load-balancing...

- Proposals arrive randomly
 - Nodes pull from the first proposal
- Highly-dynamic



Node q will serve f nodes whp



Node q will serve f nodes wlp

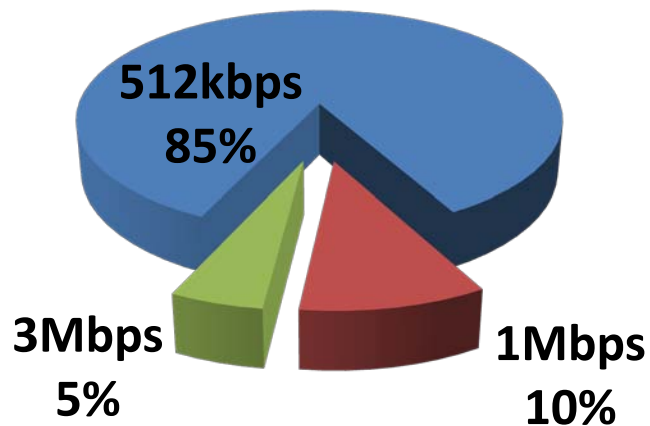
... but the world is heterogeneous!



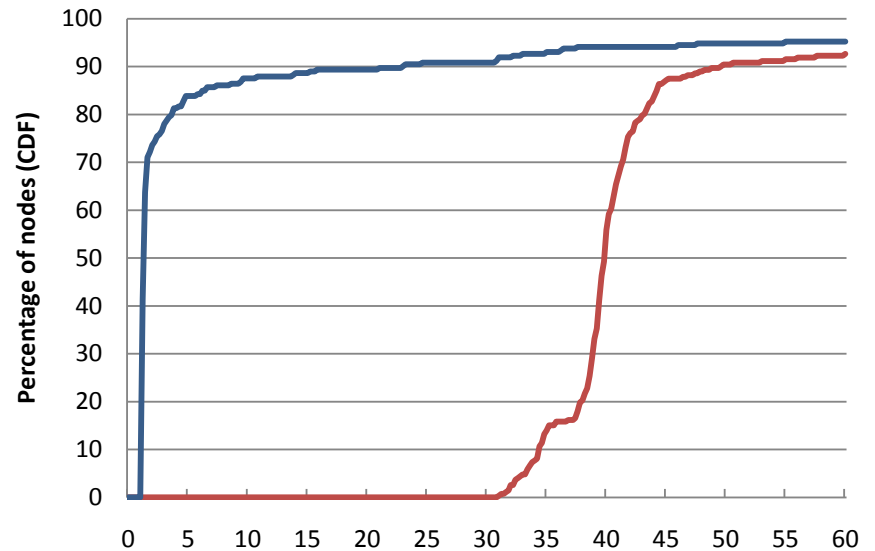
Load-balancing

Capability

3 classes (691kbps avg):



Percentage of nodes receiving at least 99% of the stream



How to cope with heterogeneity?

- **Goal:** contribute according to capability

- Advertise more = sell more:

- Propose more = serve more

- Increase fanout...

... and decrease it too!



vs



- **Challenges:**

- Preserve reliability of gossip

average fanout (f_{avg}) \geq initial fanout = $\ln(n) + c$

- Cope with dynamic capabilities

Heterogeneous Gossip - HEAP



Contribute according to capability

Capability

- 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

b_{avg}

Capability

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

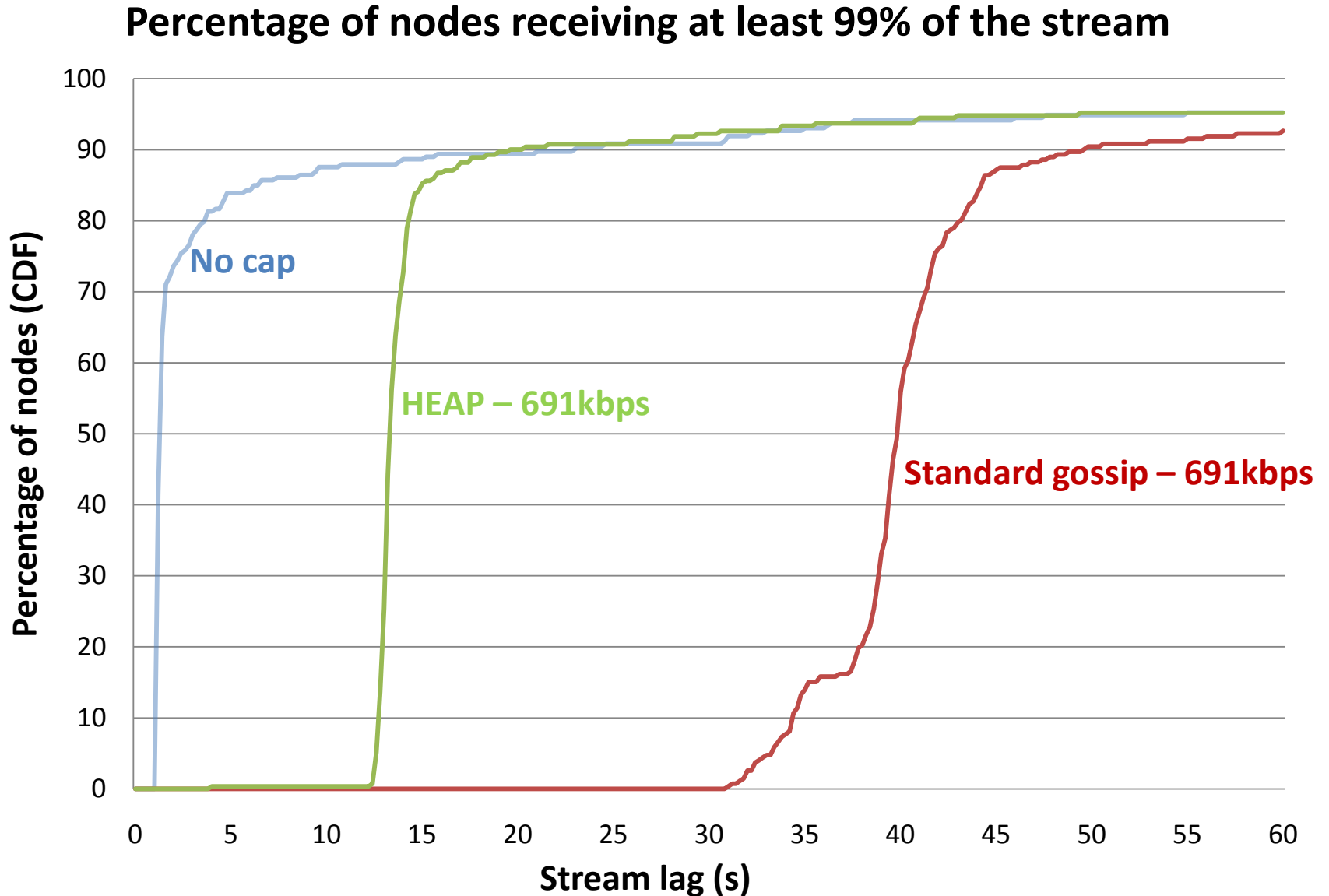
$$f_q = f_{init} \cdot b_q / b_{avg}$$

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

HEAP

- Get b_{avg} with (gossip) aggregation
 - Advertise 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



Experimental Setup

- 270 PlanetLab nodes
- Network capabilities
 - Bandwidth cap by throttling
 - Communication with UDP
- Stream rate of 600kbps
 - Windows of 110 events, including 9 FEC events
- Gossip
 - period of 200 ms
 - $f_{avg} = 7 (\ln(270) = 5.60)$



Evaluation Metrics

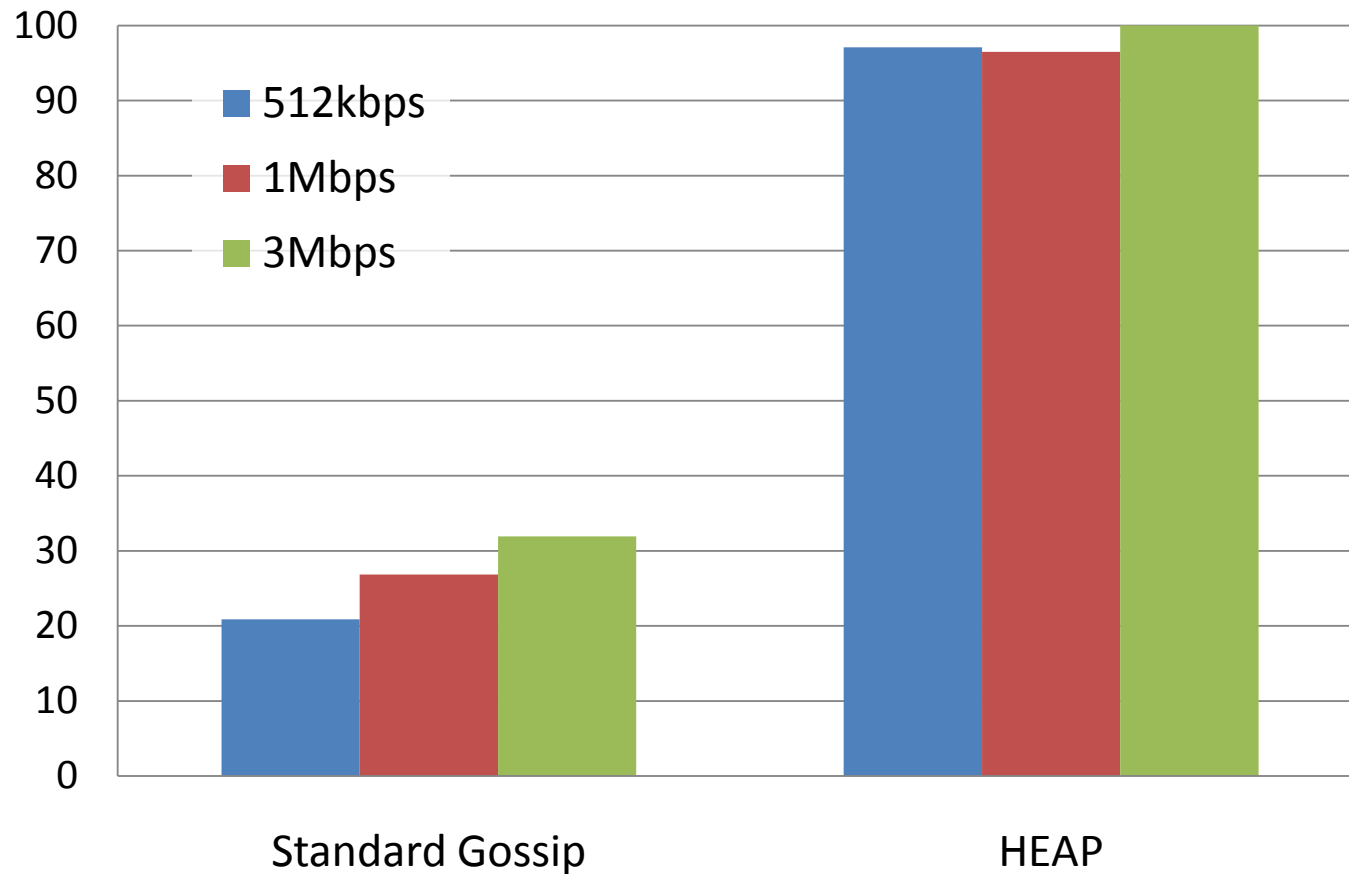


- Stream Lag
 - Time difference between creation at the source and delivery to the player
- Stream Quality
 - A window is considered jittered if < 101 events
 - Stream with maximum of 1% jitter means at least 99% of the windows are complete
 - Incomplete does not mean “blank”!

Quality improvement

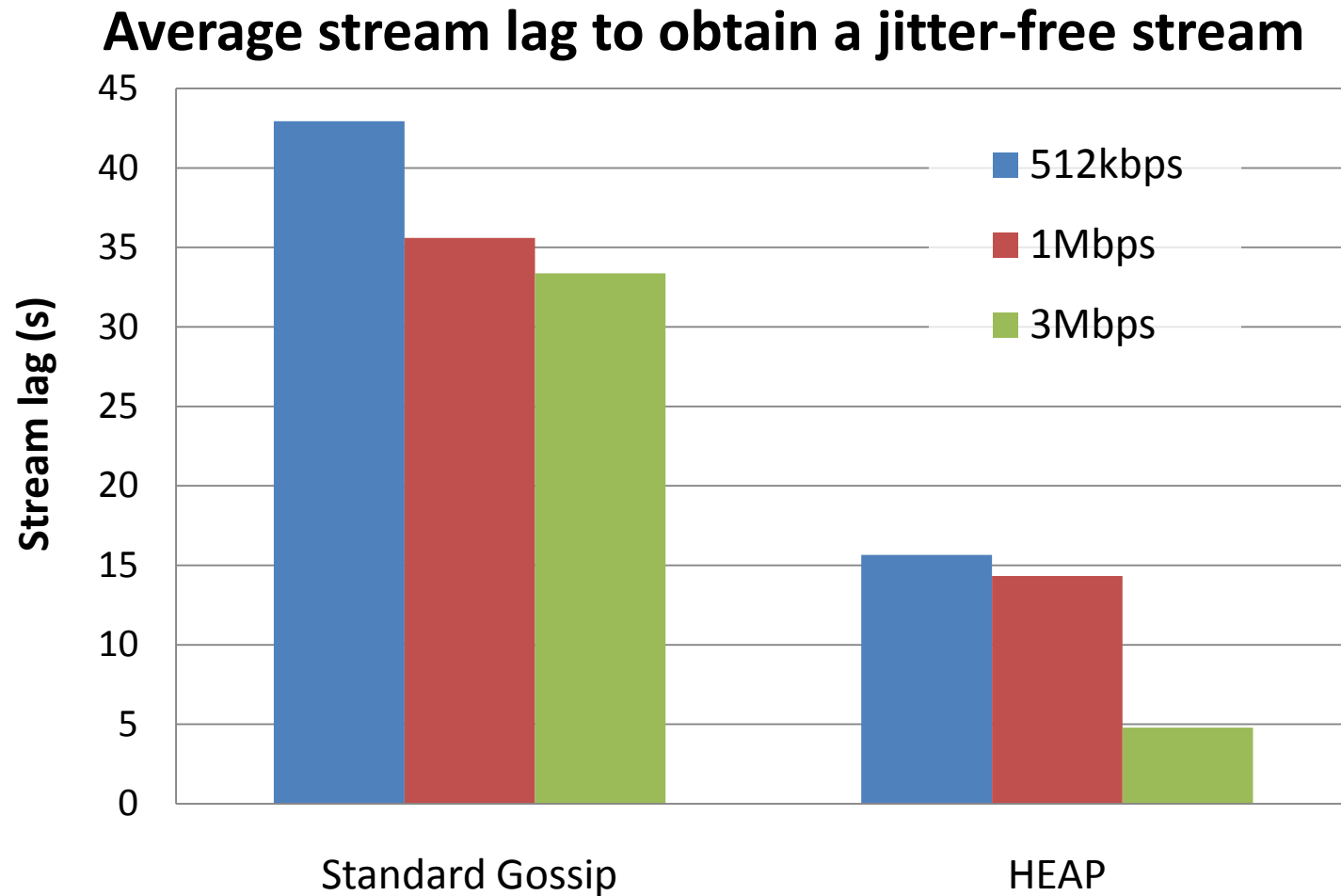
- Stream lag of 10s

Jitter-free percentage of the stream



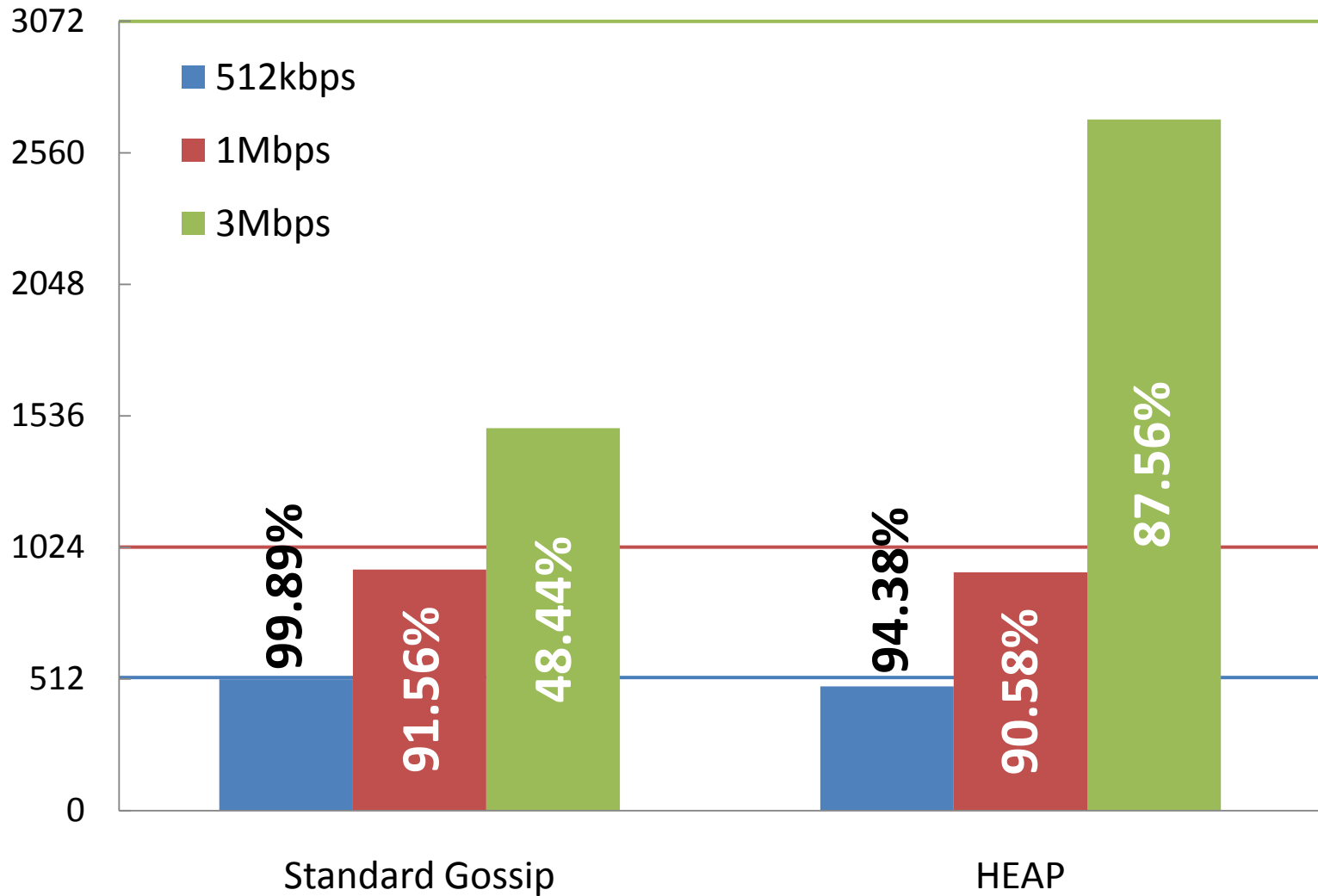
Stream Lag

- For those who can have a jitter-free stream



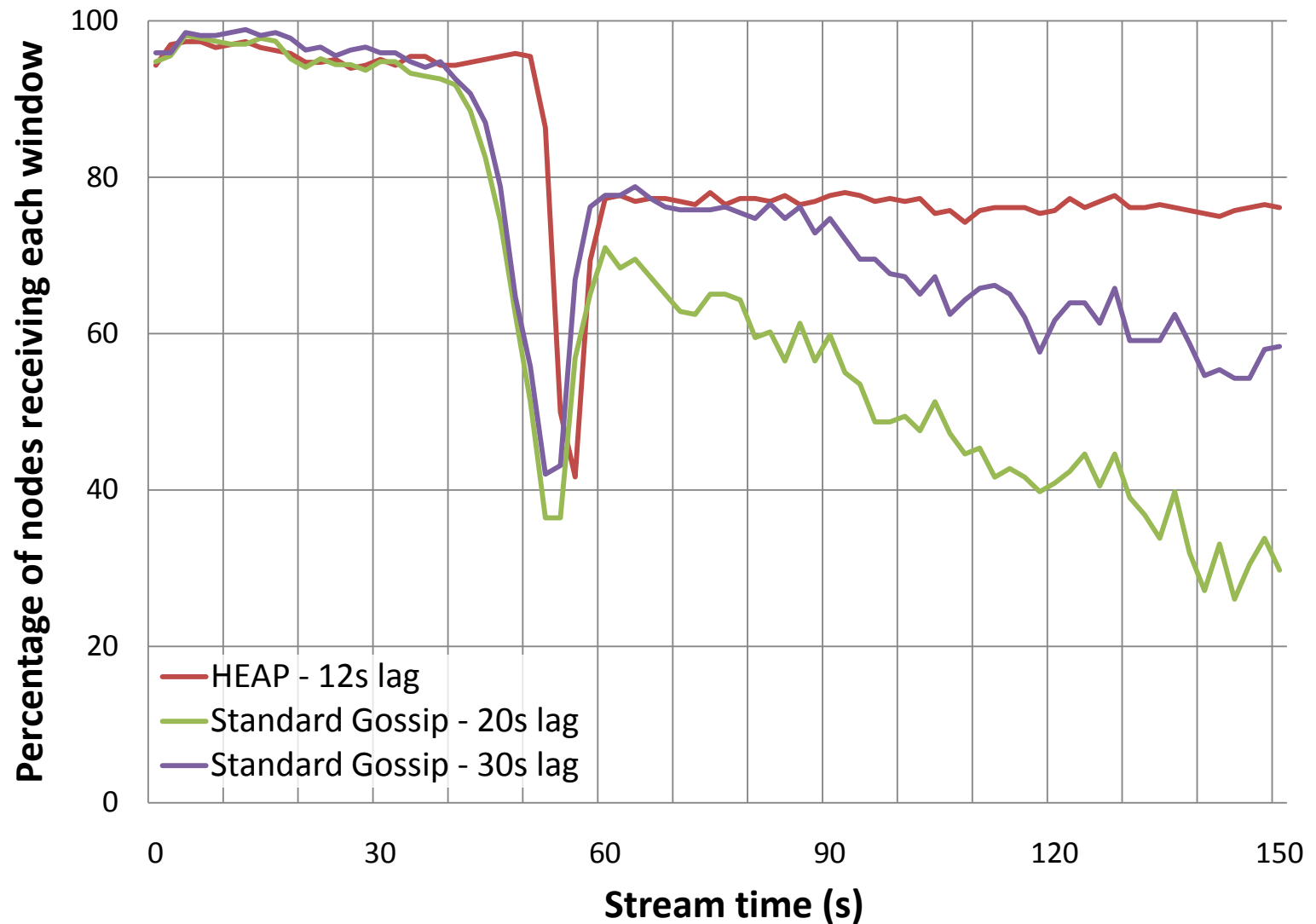
Proportional contribution

Average bandwidth usage by bandwidth class



20% nodes crashing

Failure of 20% of the nodes at time t=60s



Conclusion

- Limitations
 - UDP usage
 - TCP-unfriendliness
 - Incoming traffic
 - Probability of acceptance also depends on latency
- Future work
 - Compare with mesh systems
 - Freeriders
 - Biasing partner selection

