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

t_0 t_1 t_2 t_3

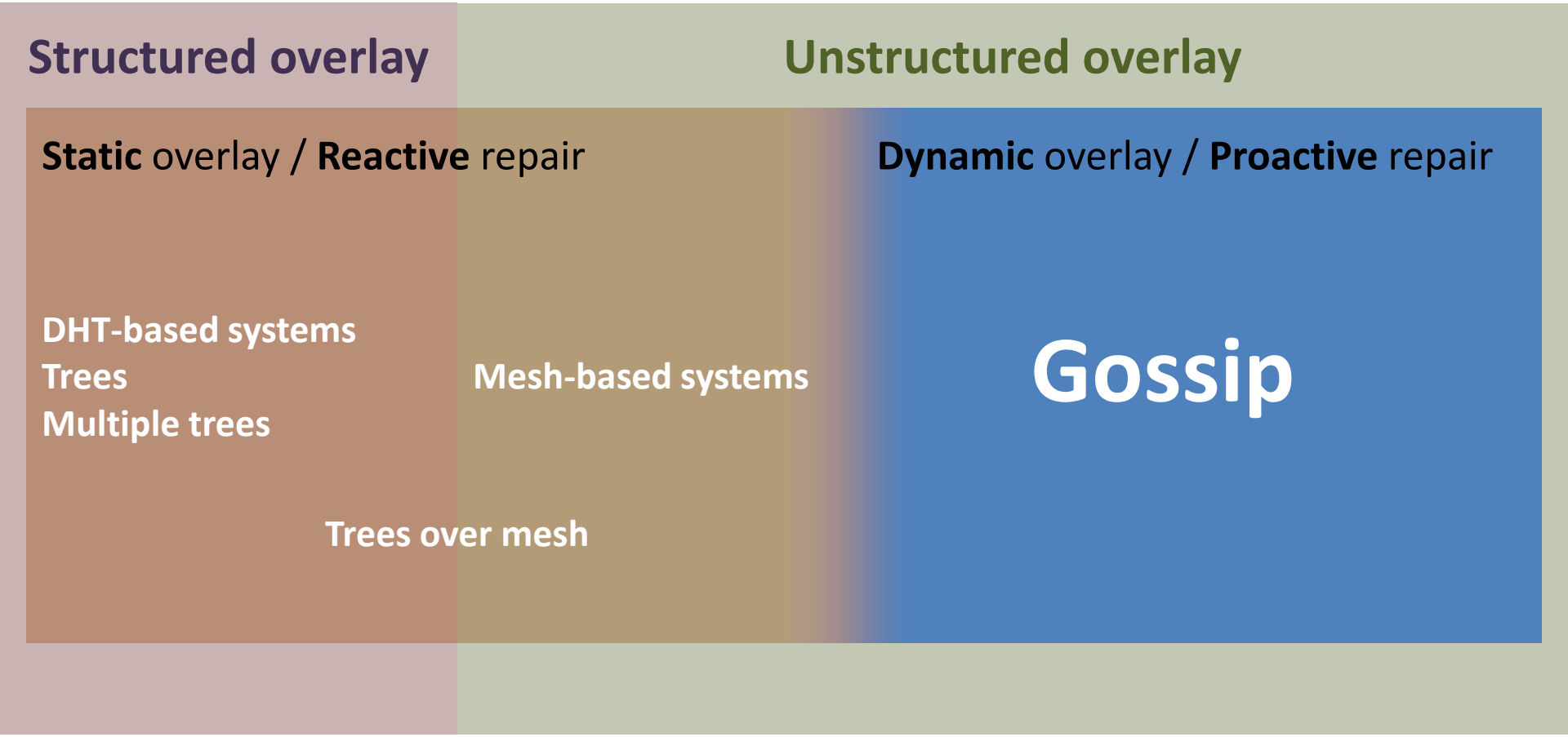
Sent at t_s
Received at t_r

Maximize **stream quality**

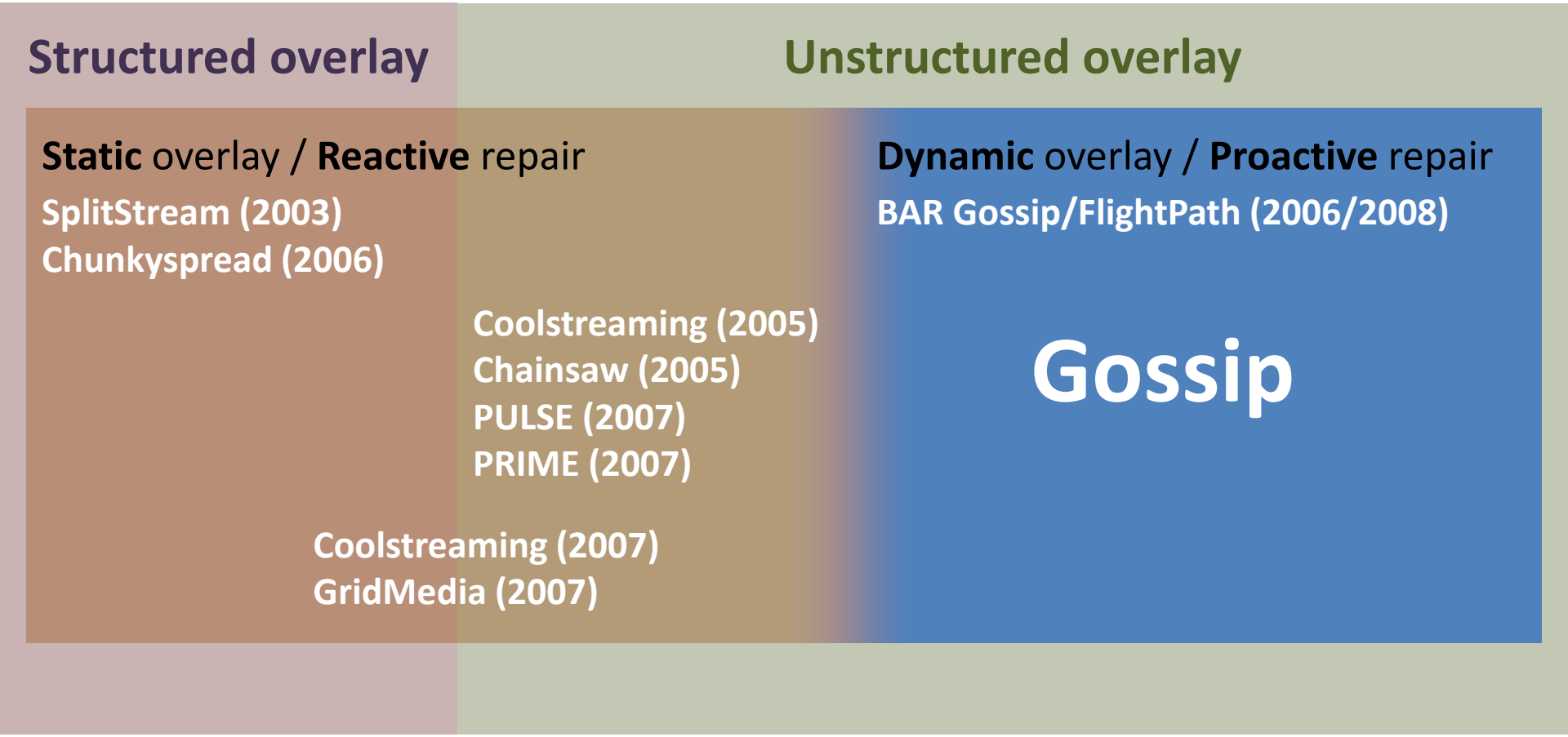
Play the stream at $t_r+\delta$
Minimize δ (**buffering delay**)

Play the stream at $t_0+\varepsilon$
Minimize ε (**stream lag**)

Existing approaches



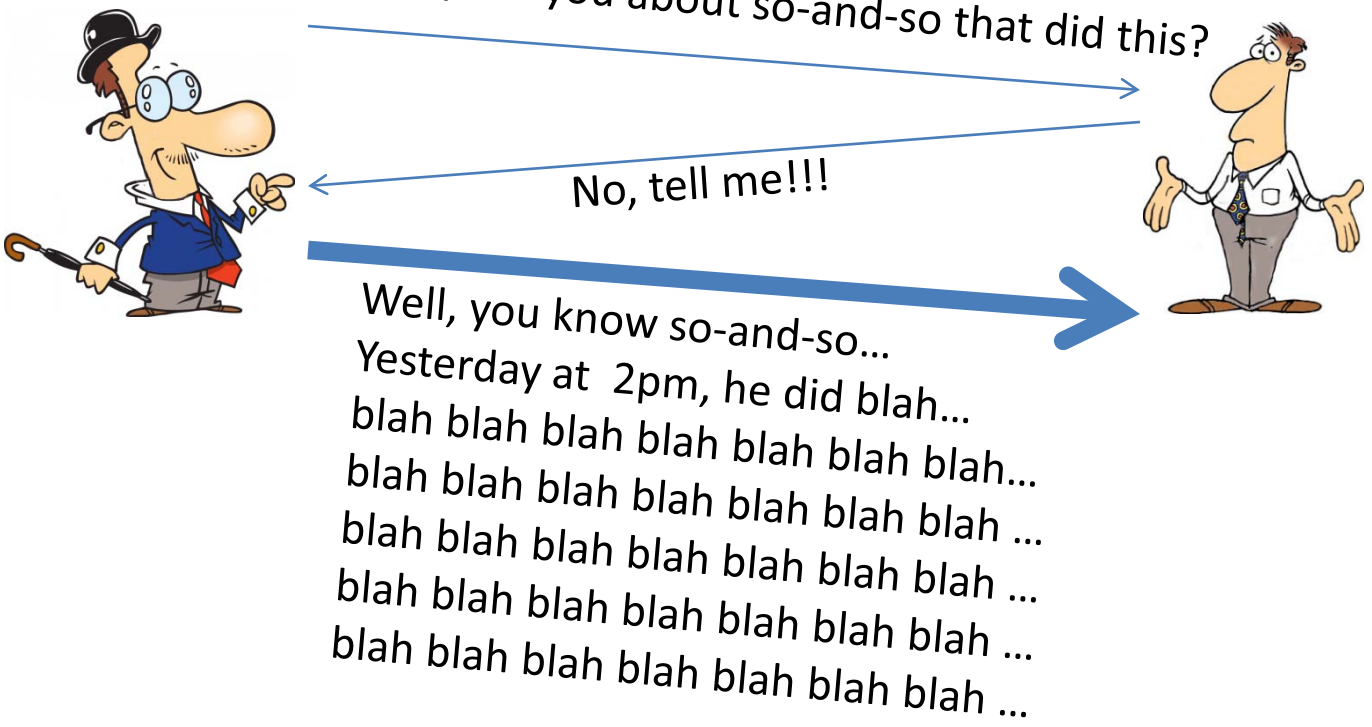
Existing approaches



Heterogeneity awareness?

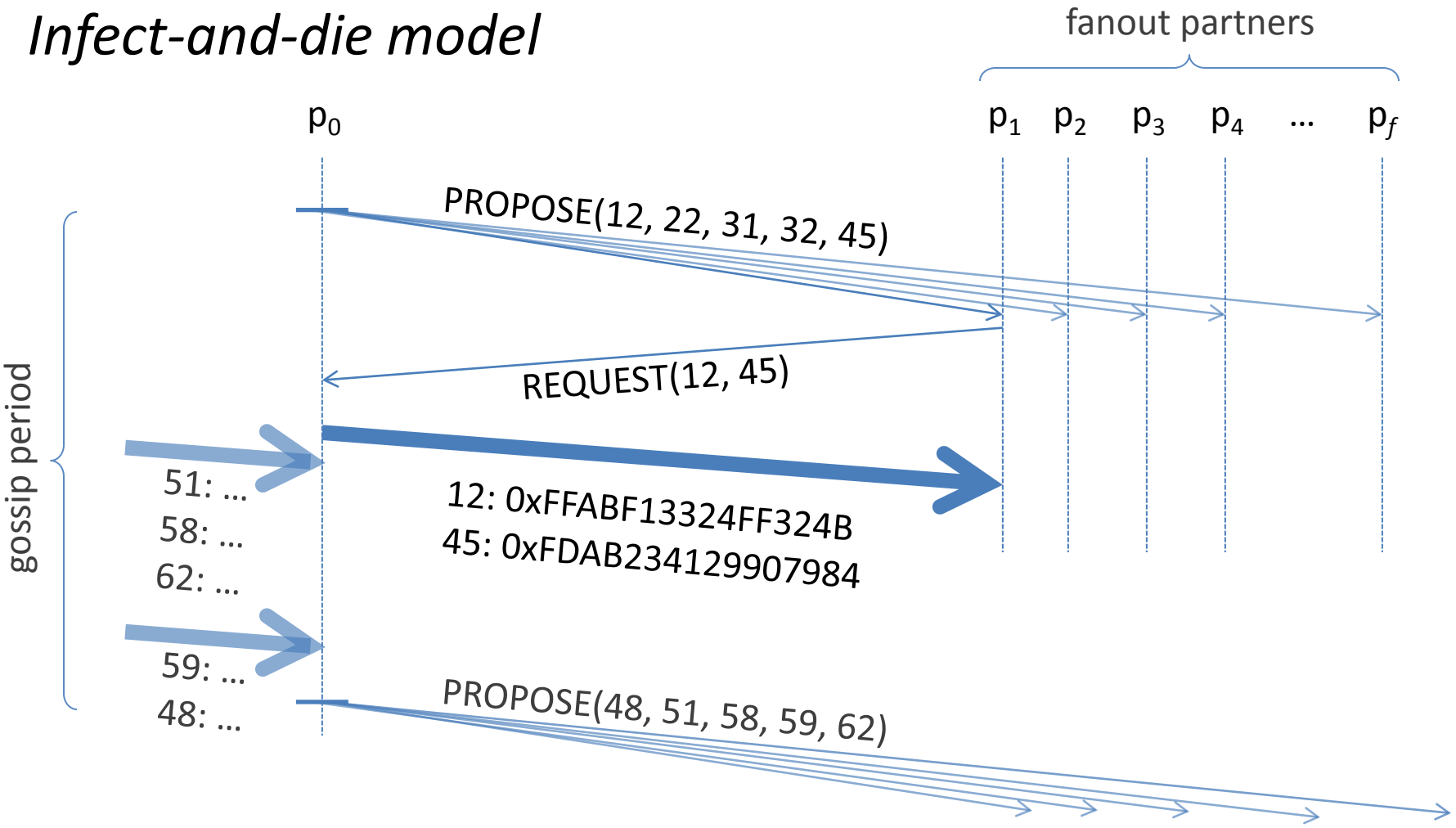
Gossip in the real world

George meets Bob:



Gossip in computer science

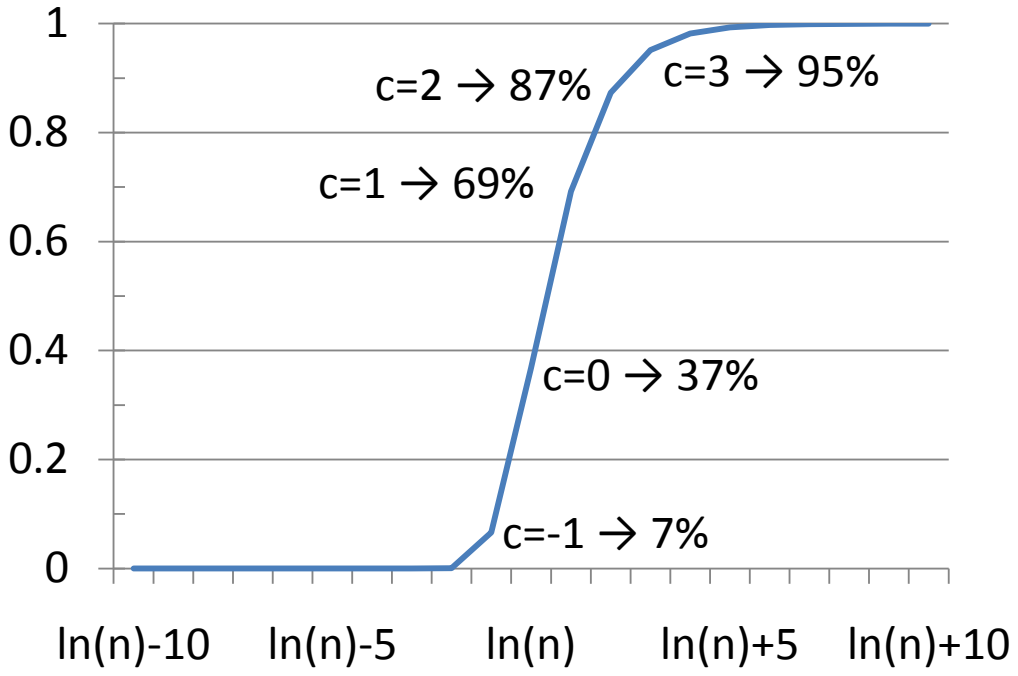
Infect-and-die model



Gossip - Theory

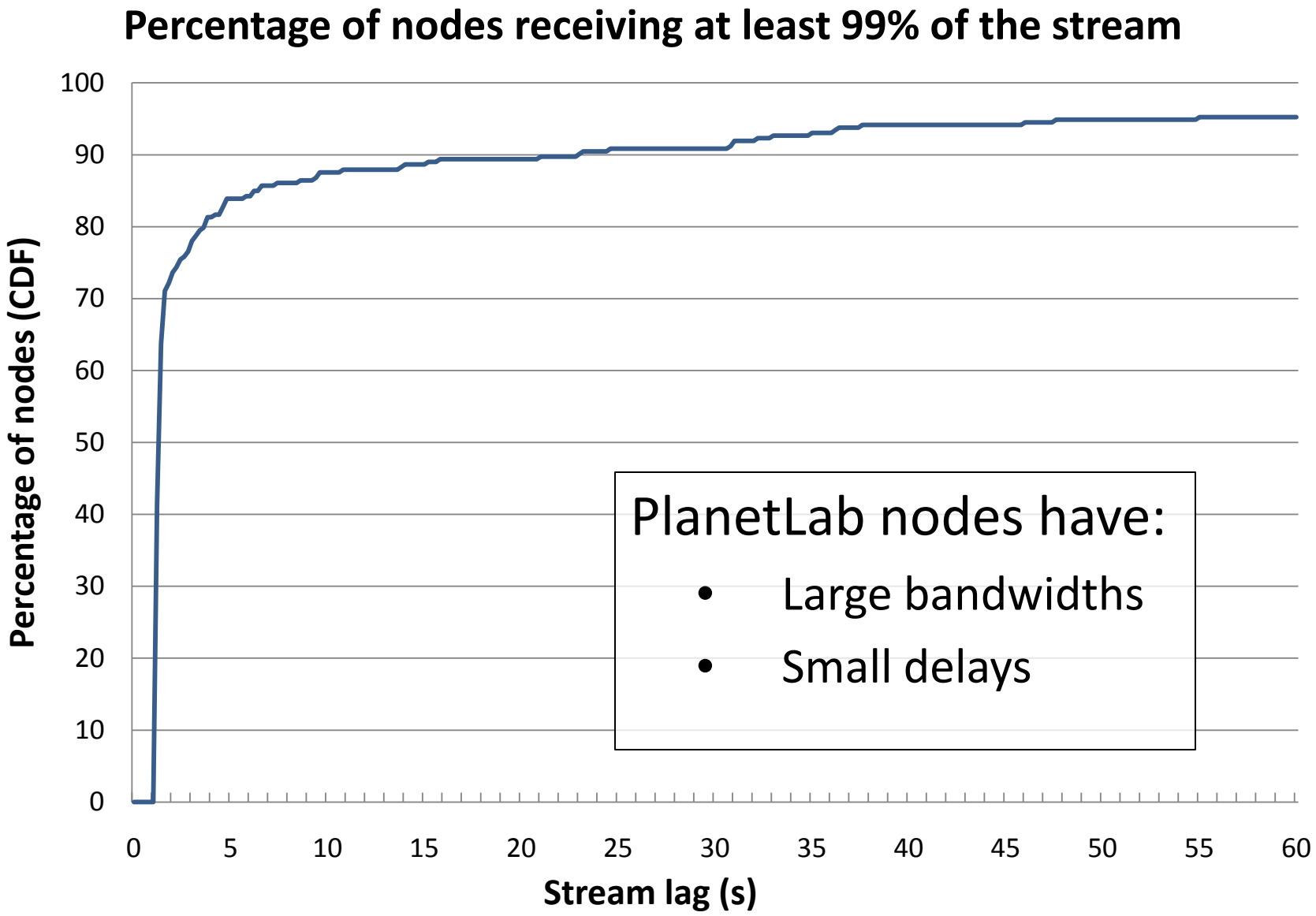
1. **fanout = ln(*n*) + *c***

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

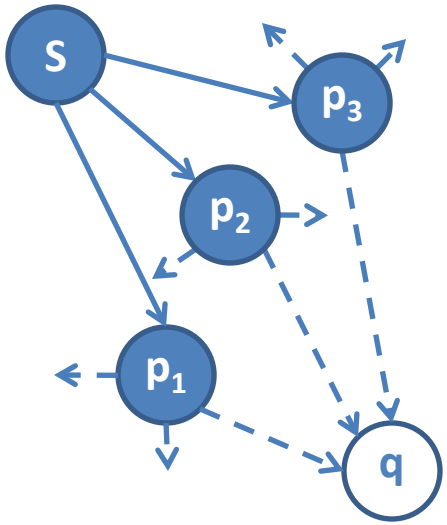


2. Holds as long as the fanout is ln(*n*) + *c* **on average**

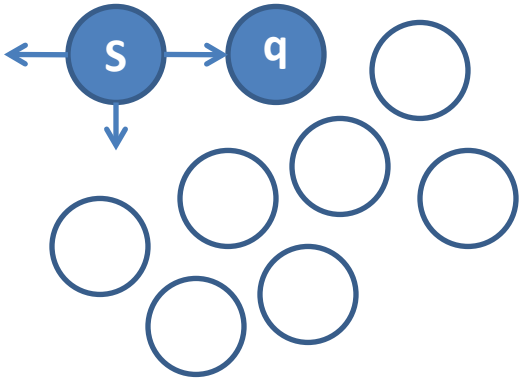
Gossip – Practice (600kbps)



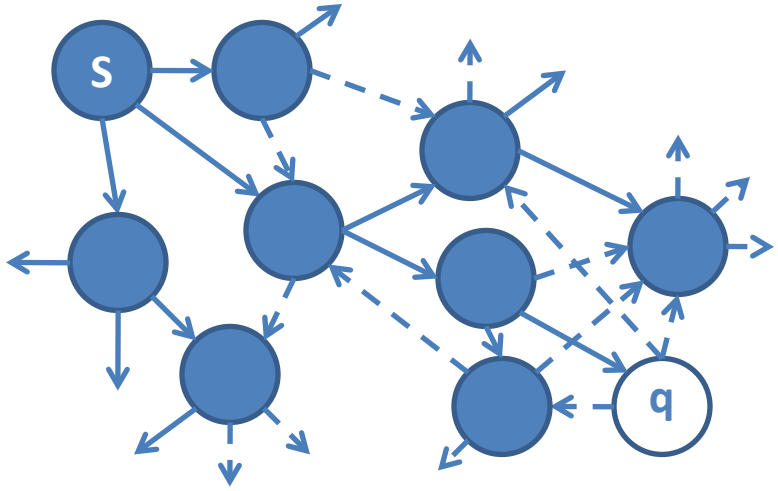
Gossip is load-balancing…



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

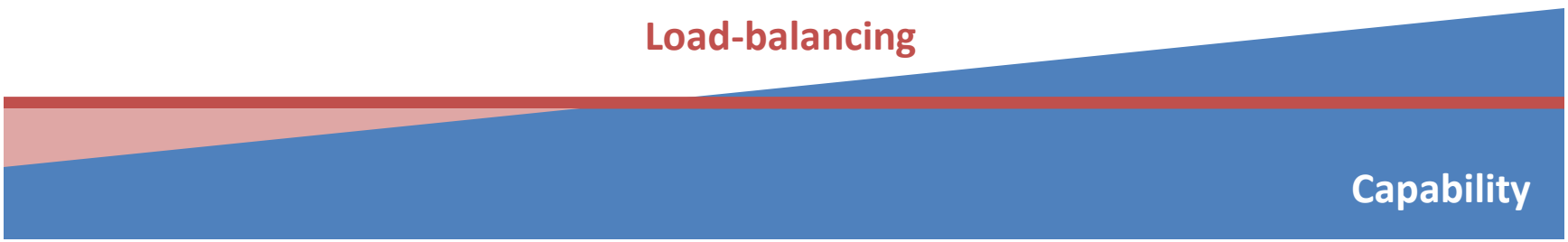


Node q will serve f nodes whp

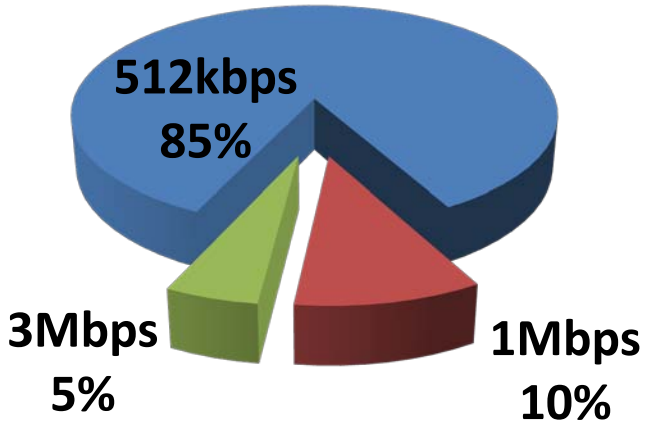


Node q will serve f nodes wlp

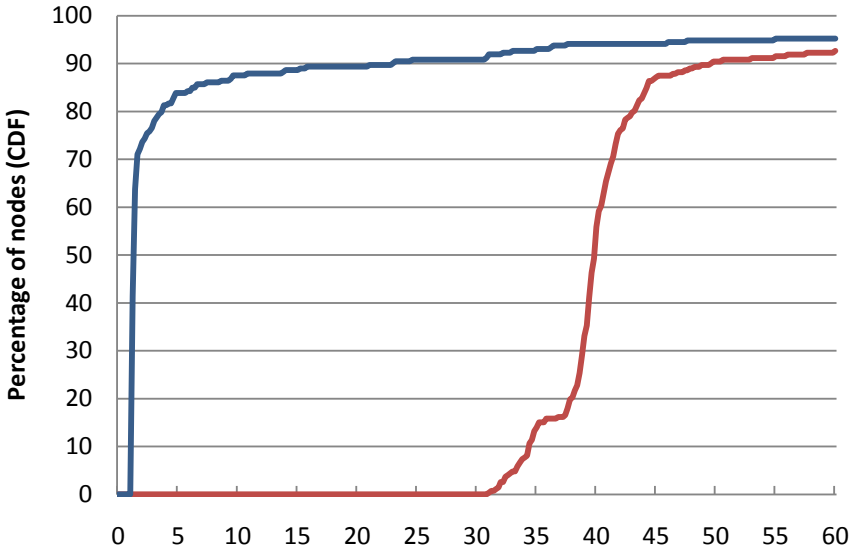
… but the world is heterogeneous!



3 classes (691kbps avg):



Percentage of nodes receiving at least 99% of the stream



How to cope with heterogeneity?

- **Goal:** contribute according to capability

- Advertize more = sell more:
  - Propose more = serve more
    - Increase fanout…

vs

… and decrease it too!

- **Challenges:**
  - Preserve reliability of gossip

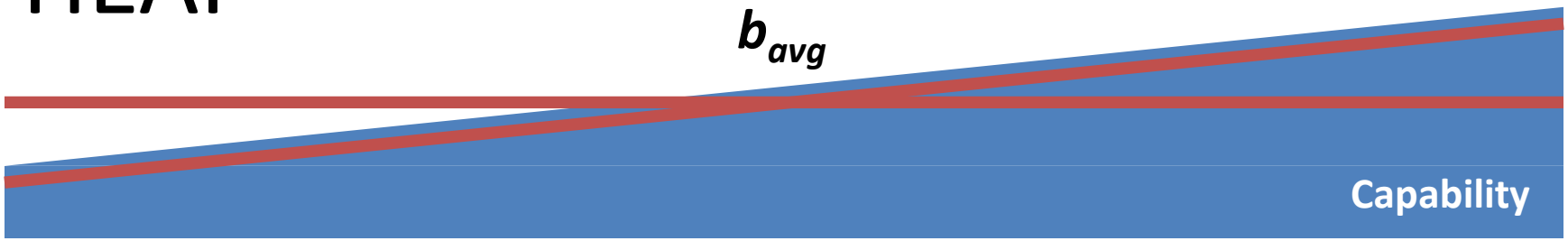
    average fanout (f_{avg}) ≥ initial fanout = $\ln(n) + c$
  - Cope with dynamic capabilities

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

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

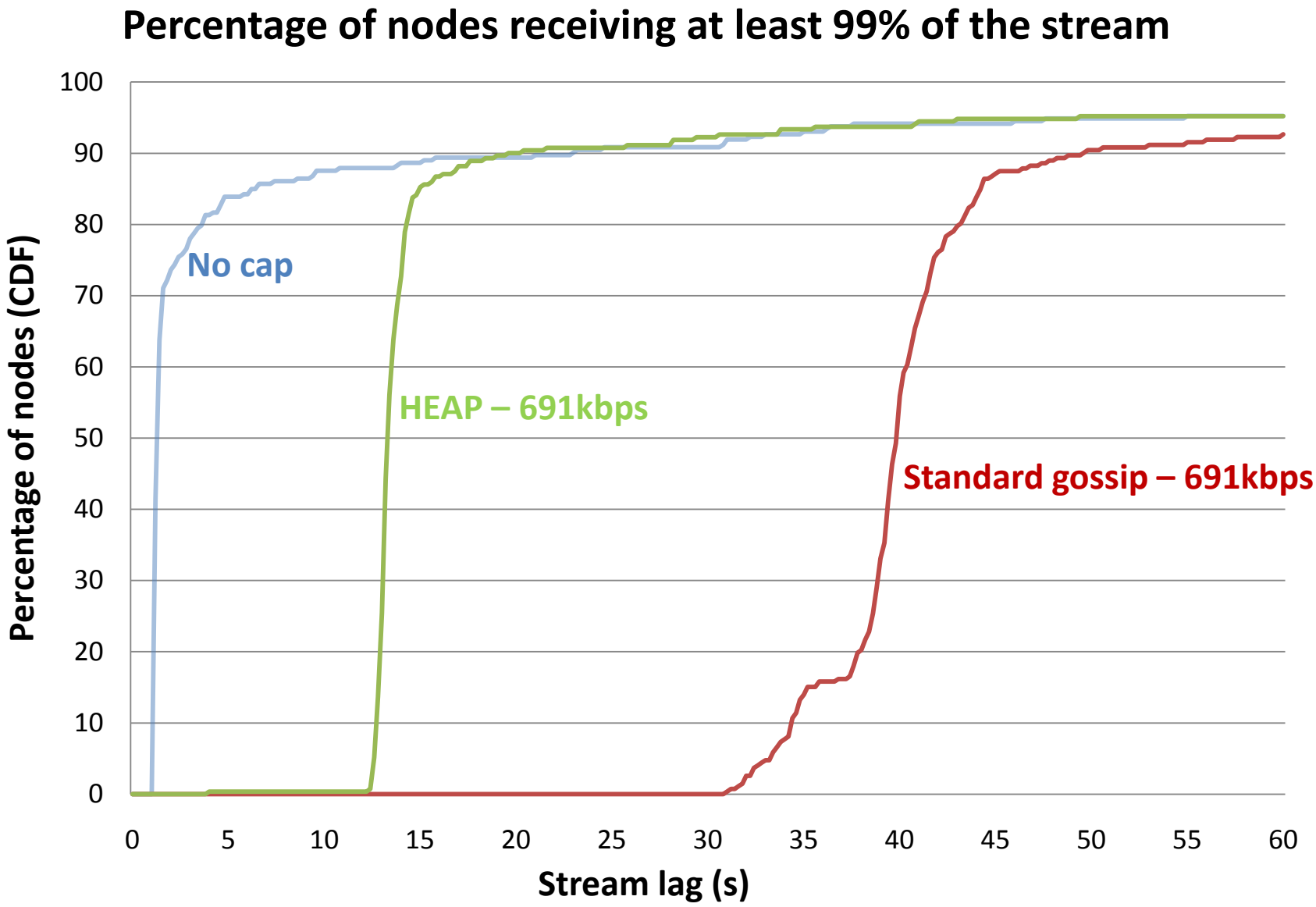
…ensuring the average fanout is constant and equal to $f_{init} = \ln(\mathbf{n}) + c$

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



Experimental Setup

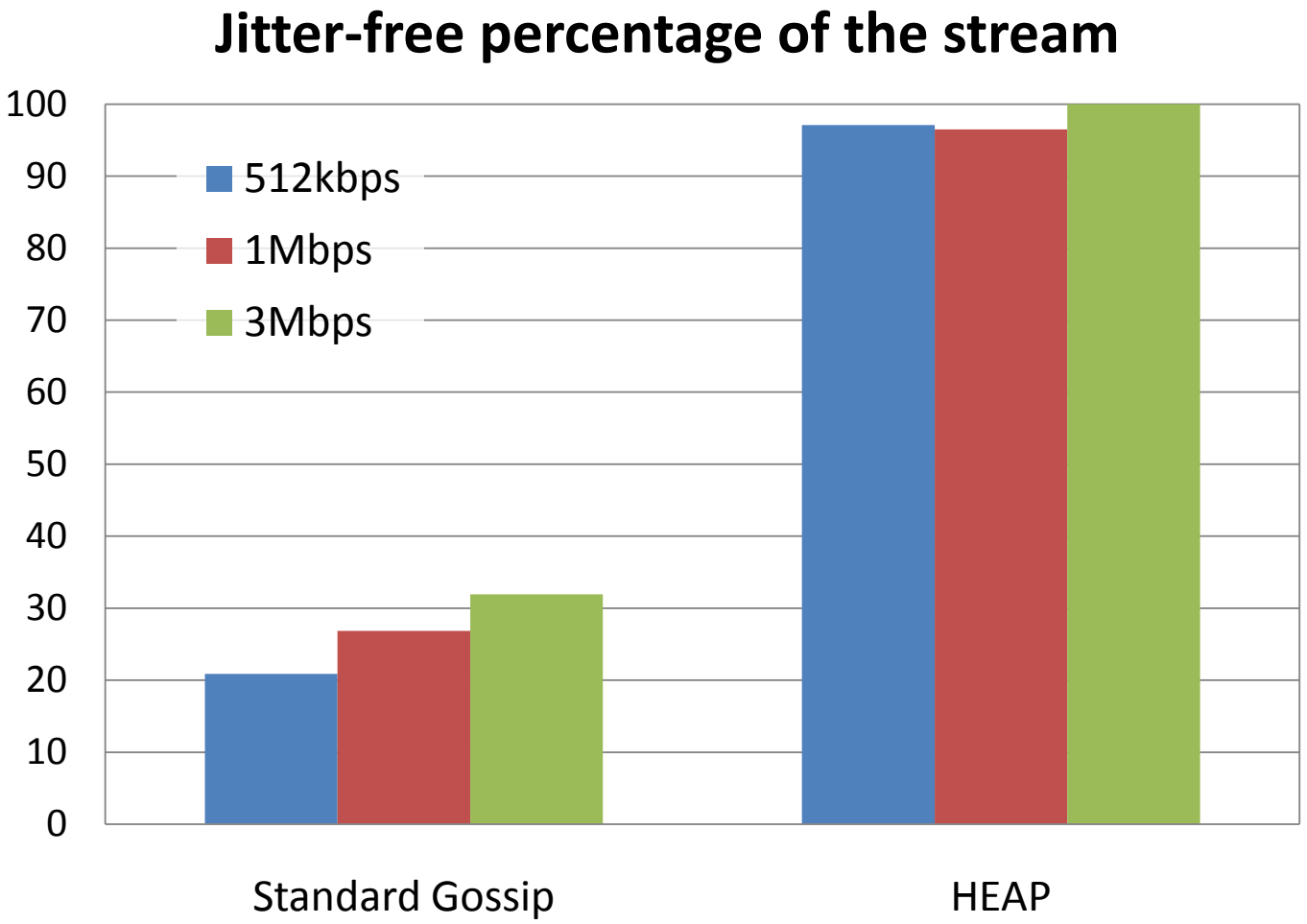
- 270 PlanetLab nodes
- Network capabilities
  - Bandwidth cap by throttling
  - Communication with UDP
- Stream rate of 600kbps
  - Windows of 110 events, including 9 FEC events
- Gossip
  - period of 200 ms
  - f_{avg} = 7 (ln(270) = 5.60)

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”!

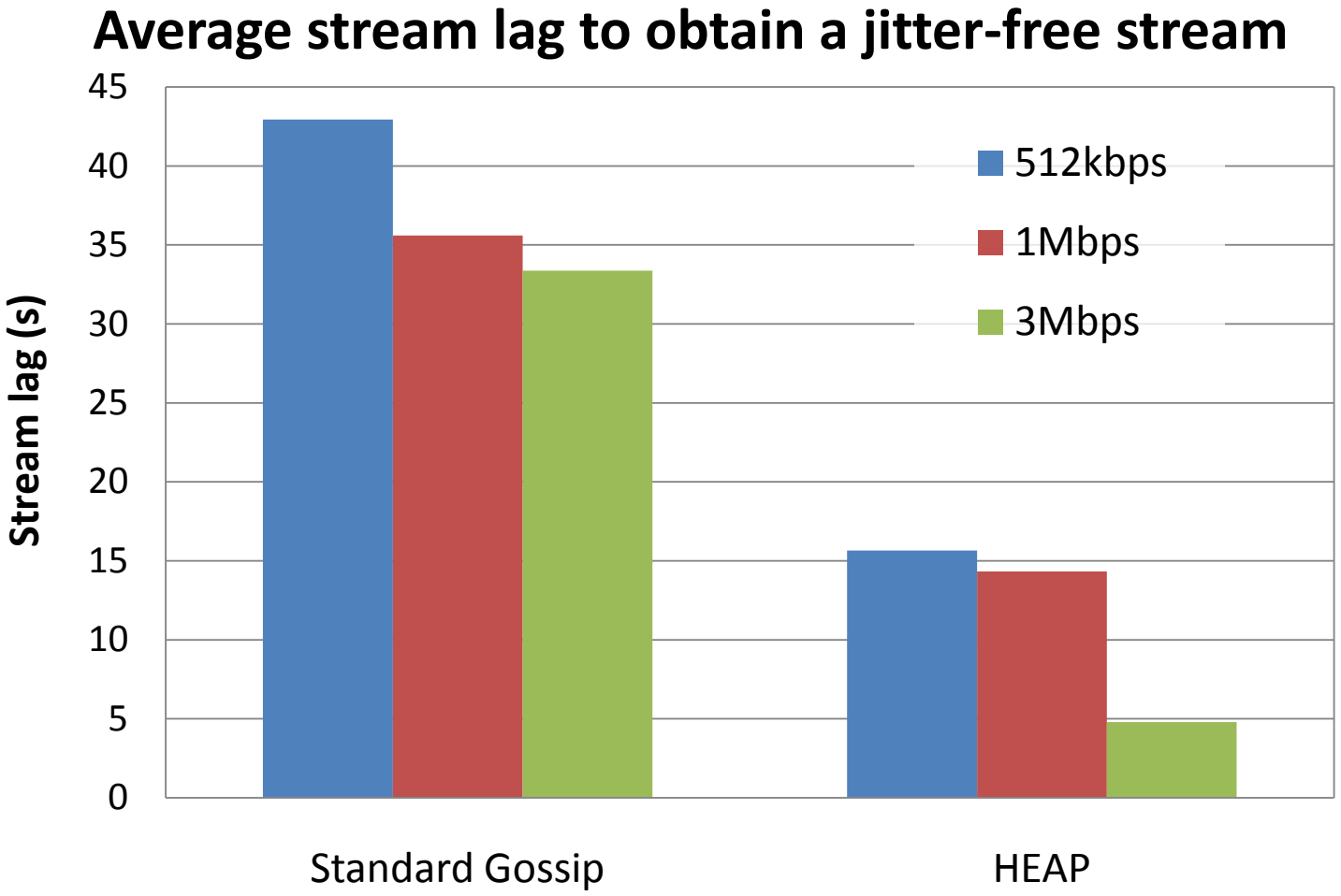
Quality improvement

- Stream lag of 10s

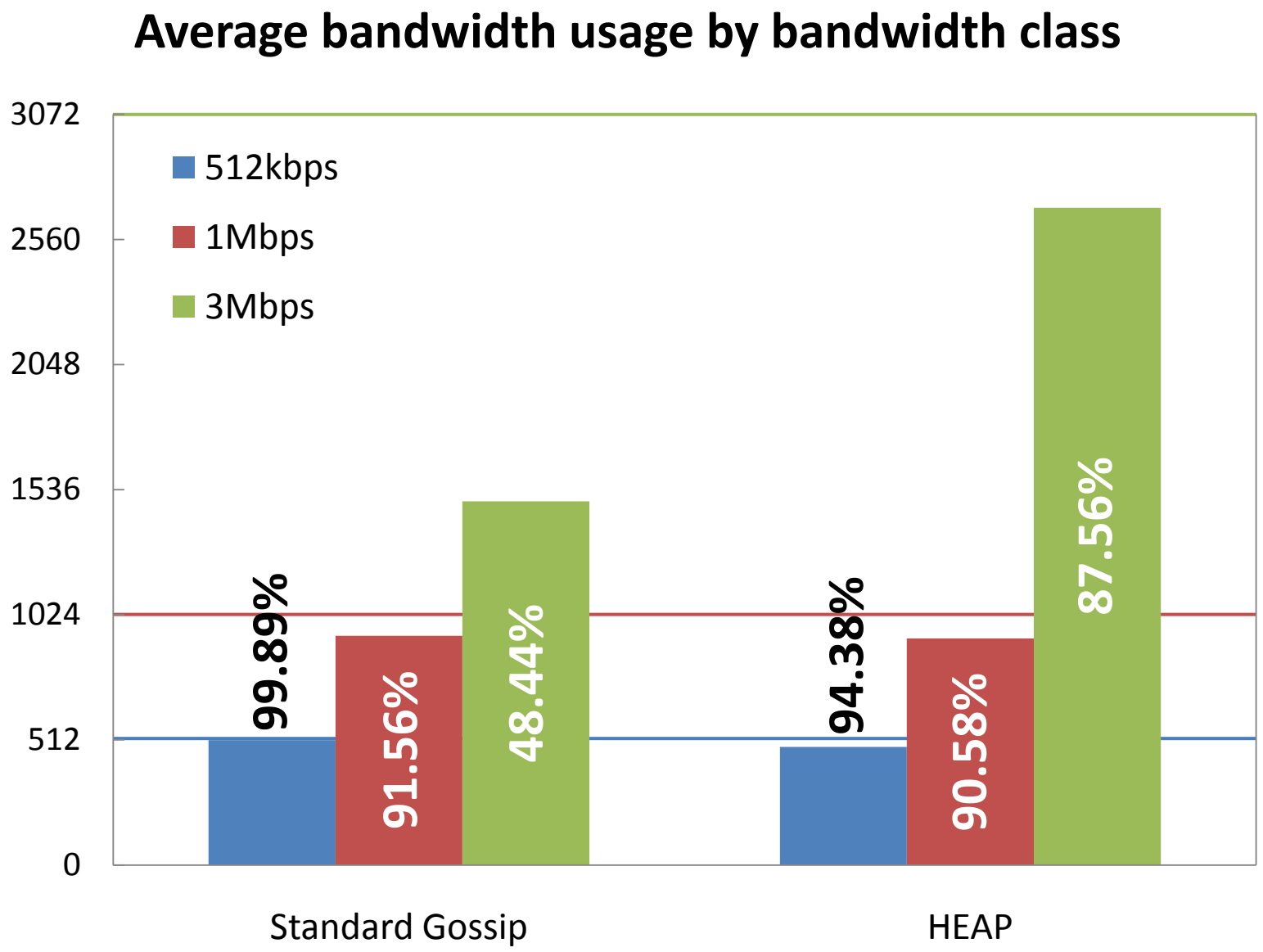


Stream Lag

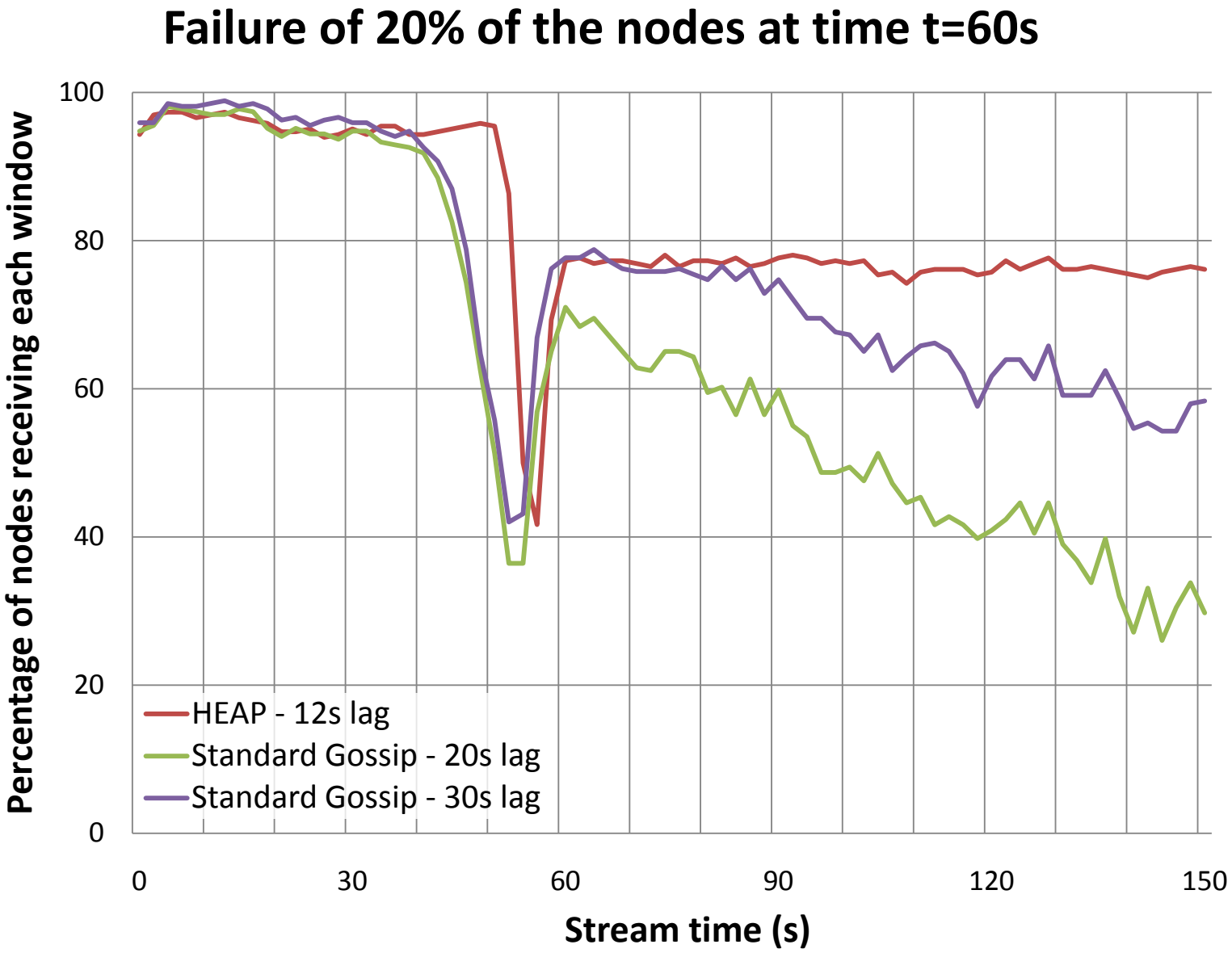
- For those who can have a jitter-free stream



Proportional contribution



20% nodes crashing



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