Nozzilla
A Peer-to-Peer Architecture for Video Streaming

Author: Alexandru Iosif Bikfalvi, Dipl. Eng.
Coordinator: Jaime García-Reinoso, Ph.D.

Outline

- Introduction
  - Motivation and Approaches of Video Streaming
- Overview of Application Level Multicast
- Selected Proposals
  - Scribe, SplitStream
- Nozzilla
  - Overview & Goals
  - P2P Architecture
  - Performance Analysis
- Summary
  - Conclusions and Future Work

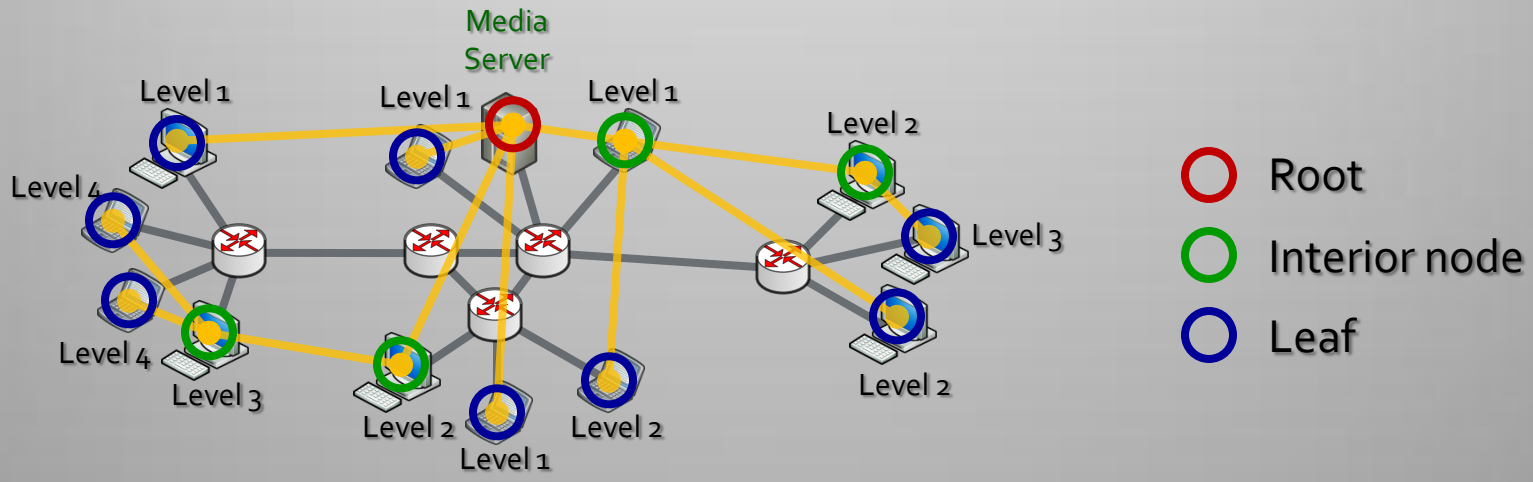
Motivation

- Media streaming is extremely expensive
  - Video streaming applications target a lot of receivers
  - Streaming servers need a lot of bandwidth and computing power
  - They may not be able to serve everybody
- Existing solutions are unfeasible or too costly

Solution	Pros	Cons
Client/Server	Simple	Not scalable
CDN	Server not overloaded	Complex and costly
IP Multicast	Good network utilization	Lack of deployment
P2P+ALM	Availability and cost	Utilization, reliability

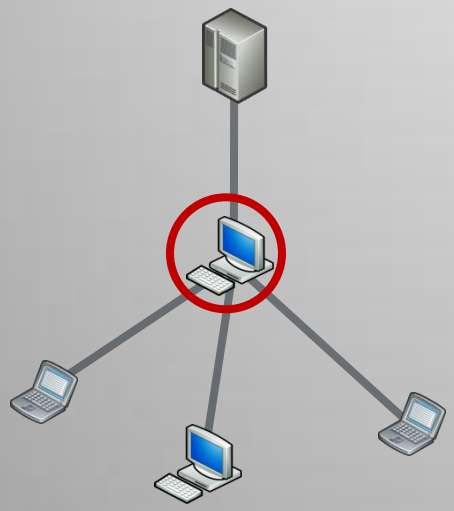
Application Level Multicast

- Packet replication is done by the peers
  - … meaning the same packets traverse same links several times
  - … but peer uplink bandwidth is (very) limited
  - … logical neighbors may be many hops away
  - … peers (i.e. nodes) come and leave as they wish
- Multicast overlay topology: tree
  - The root can be the media server or a client peer



Application Level Multicast

- Tree construction is very important
  - Tree level: determines the delay and stability
  - A peer accepts a limited number of children: fan-out
  - The fan-out of interior nodes is limited by their uplink capacity (from the peer to the network)

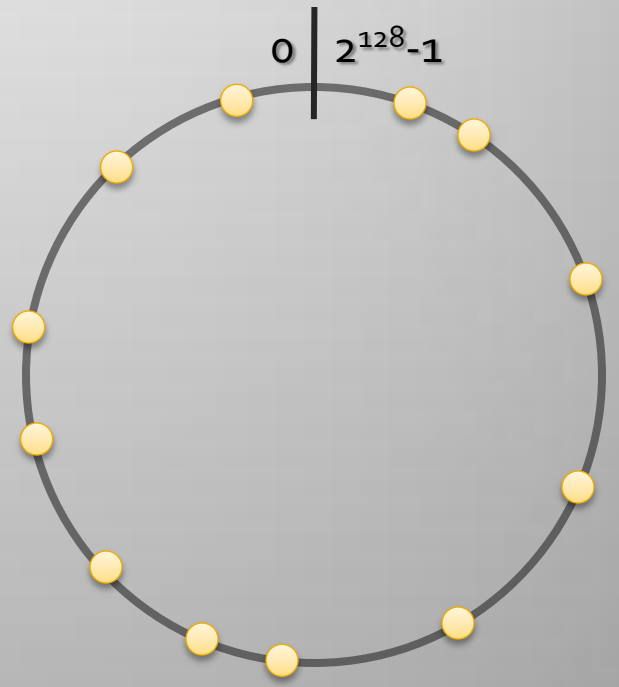


- For this multicast tree, the peers needs on the uplink three times as much bandwidth as is necessary for the downlink

Scribe (Overview)

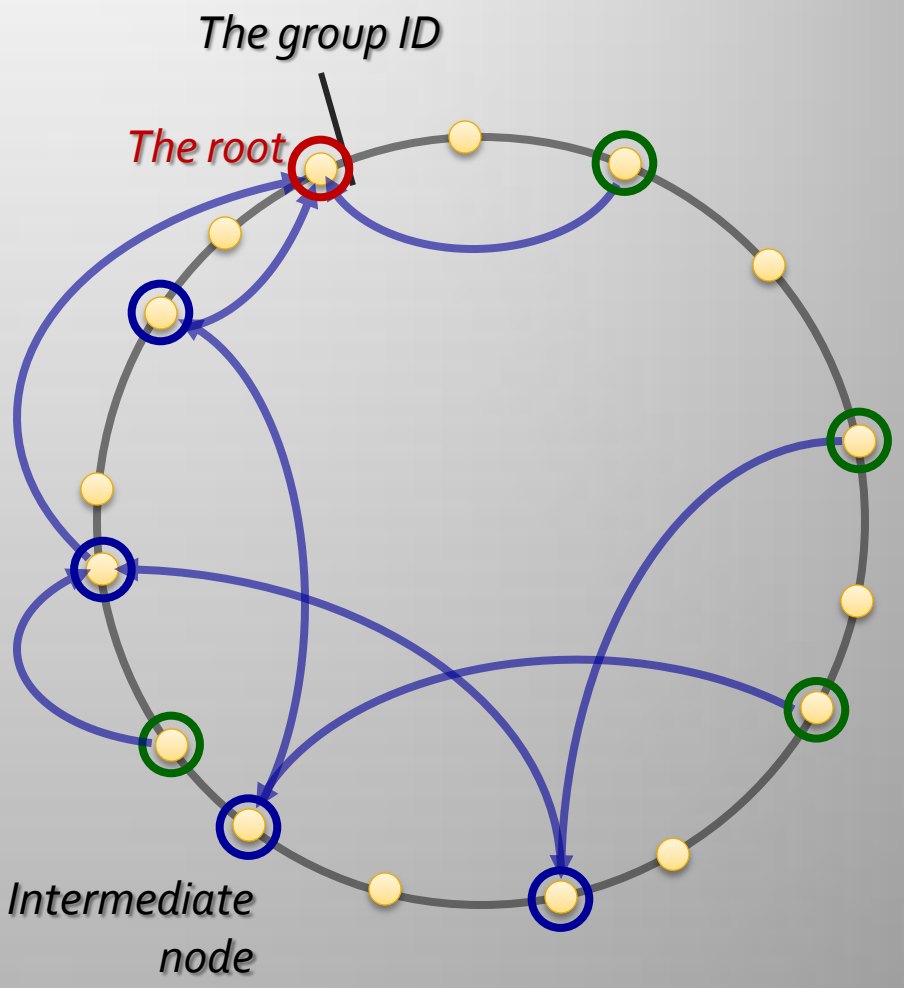
- Does not target video streaming
- Used to create an ALM tree in a P2P network
  - Used as foundation in many proposals including *Nozzilla*
  - Creates a multicast tree using Pastry

- Pastry
  - Structured P2P protocol
  - 128-bit circular hash space
  - Each peer has an ID in base $B = 2^b$ *(b is 3 or 4)*
  - Routes messages to the peer closest to a given target ID



Scribe (Multicast Tree)

- Each multicast group has a group ID
- The peer closest to the group ID becomes the root
- A peer joins the multicast group:
  - By sending a join message to the group ID
  - Joining finishes when a peer member of the group is found
  - Each intermediate peer also joins the group



SplitStream

- Enhances Scribe for video streaming
  - Takes into account the *fan-out*
  - Reduces the necessary *uplink bandwidth*
- How does it work?
  - Splits the stream into pieces (*stripes*)

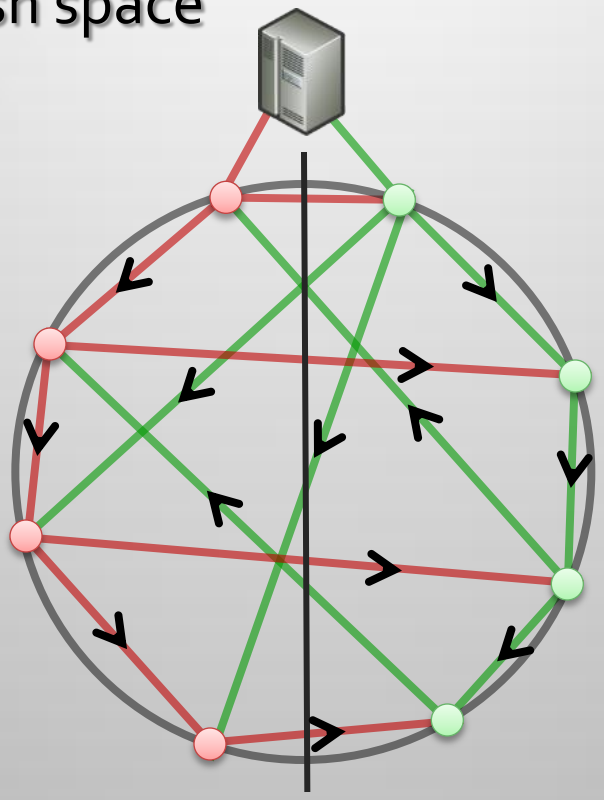
Example: 2 stripes

  - Creates one multicast tree for each stripe

SplitStream (cont'd)

- Divides the hash space

Example: 2 stripes



These peers can be interior nodes only for the red stripe

These peers can be interior nodes only for the green stripe

- The gain:
  - For each peer the downlink : 2 stripes (red and green)
  - For each peer the uplink: max 2 stripes (red or green)

Nozzilla

- Nozzilla is similar to SplitStream:
  - P2P protocol used to create multicast trees for video streaming
  - Based on Scribe/Pastry
  - Uses multiple stripe delivery (more robust, supports multiple description coding)
- However:
  - Takes into account the uplink resources at any time
  - Peers with resources are always considered interior nodes
  - Children can easily identify these peers
  - Peers re-compute resources whenever something changes

Nozzilla: Features

- Can be used with a QoS-enabled network
  - Each stripe can have a different priority
  - Peers compute resources per stripe considering QoS
- Improves Scribe peer distribution in the tree
  - Scribe/Pastry always forward messages to the peer closest to the *group ID*, which is the root
  - Hence, many peers will join the root

Nozzilla: Scenario

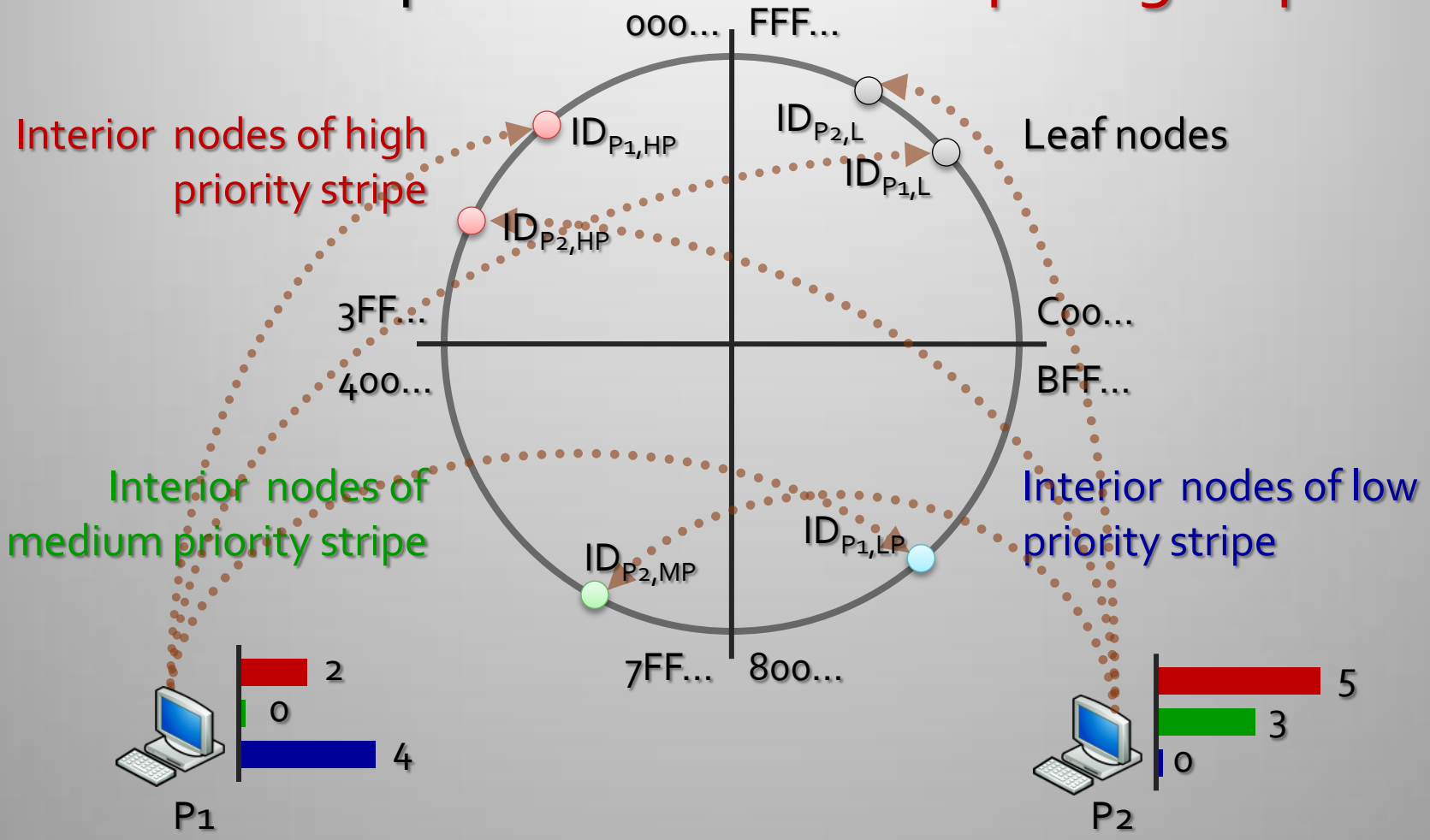
- For the purposes of this presentation
  - We have three stripes with a different priority

Example: 3 stripes

  - Use a slice in the hash space to contain *nodes* that can be interior nodes for each stripe
  - Use an extra slice to contain *nodes* that cannot be interior nodes
  - A peer computes its resources and can become a *node* in each slice

Nozzilla: P2P Architecture

- The hash space is divided into peer groups



Nozzilla: Modifying Pastry

- Each peer has:
  - Four node IDs, one for each peer group
  - The ID of a peer group is advertised if there are resources
  - The ID of the leaf peer group is always advertised
- The IDs are set when the peer is created

First two bits identify the peer group

Rest 126 bits are random and the same for all IDs

  - When a peer sends a message, it sends the 126 bits and specifies which IDs are advertised
  - Recipient adds advertised IDs and removes non-advertised IDs to/from routing table

Nozzilla: Modifying Pastry

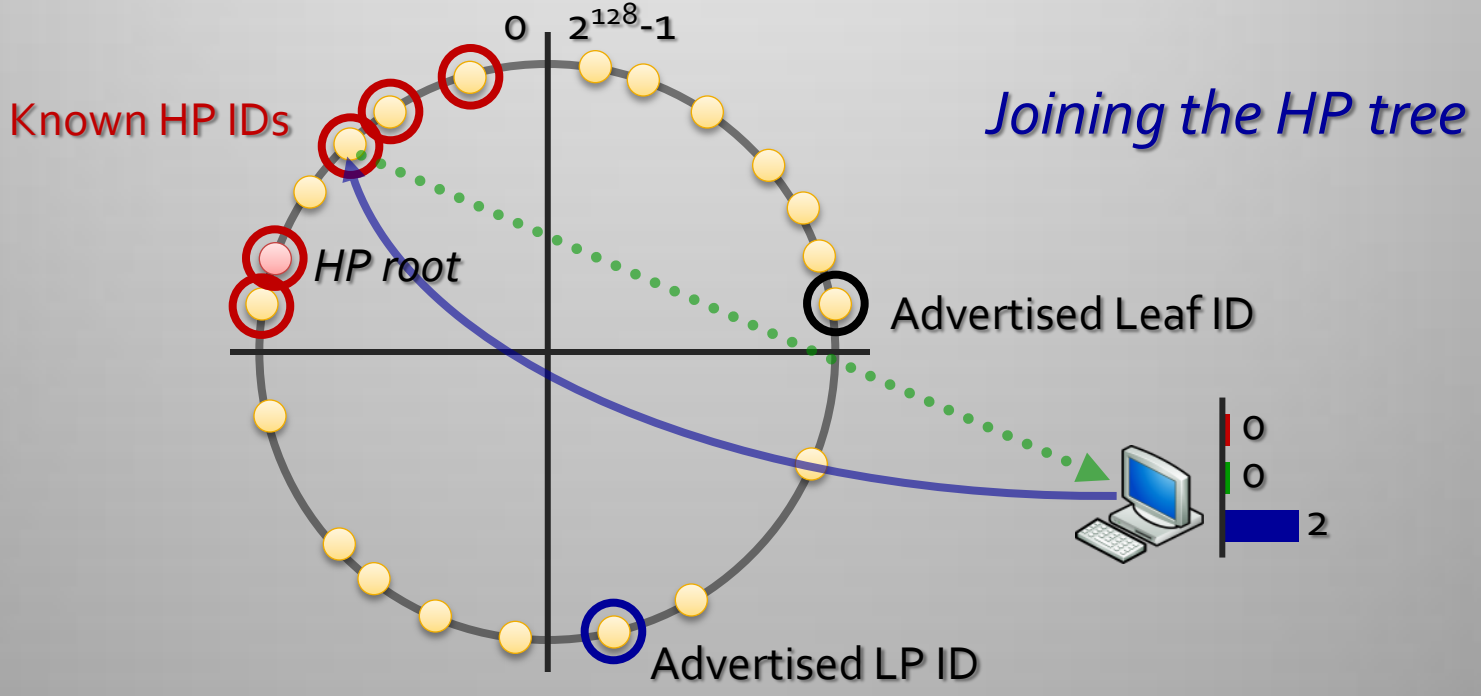
- When uplink resources become available
  - The ID of the associated peer group is advertised
  - A Pastry join is performed for that ID
- When uplink resources are exhausted
  - The ID of the associated peer group is *removed*
  - All neighbors are informed

Nozzilla: Modifying Scribe

- Scribe is modified in the following way
  - Intermediate nodes no longer join the multicast tree
  - First hop selected randomly from all known interior nodes
  - Unlike Scribe, these interior nodes are always known
- When a non-interior node receives a request
  - Will forward it to the next known node closer to the root
  - If none found, will return it to the last sender

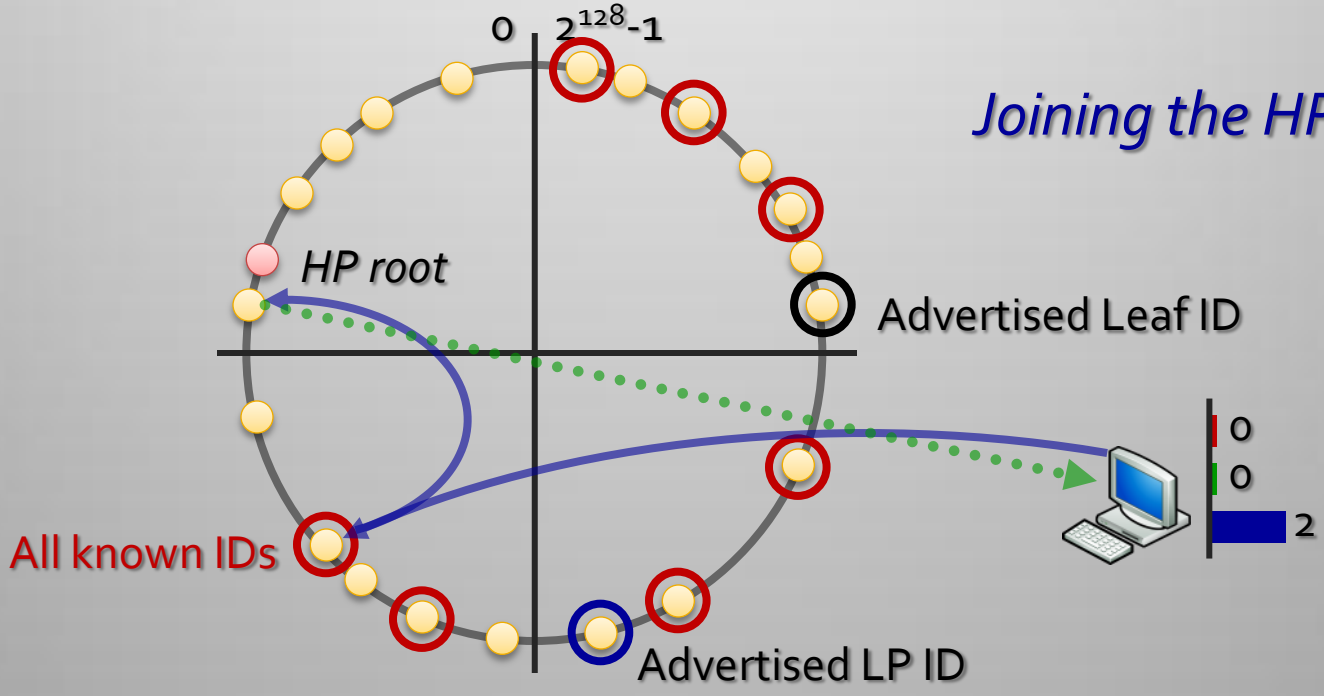
Nozzilla: Operations

- Joining the multicast tree
  - The initiator knows several interior nodes
  - It uses a random selection algorithm to choose first hop
  - Reduces load on the root



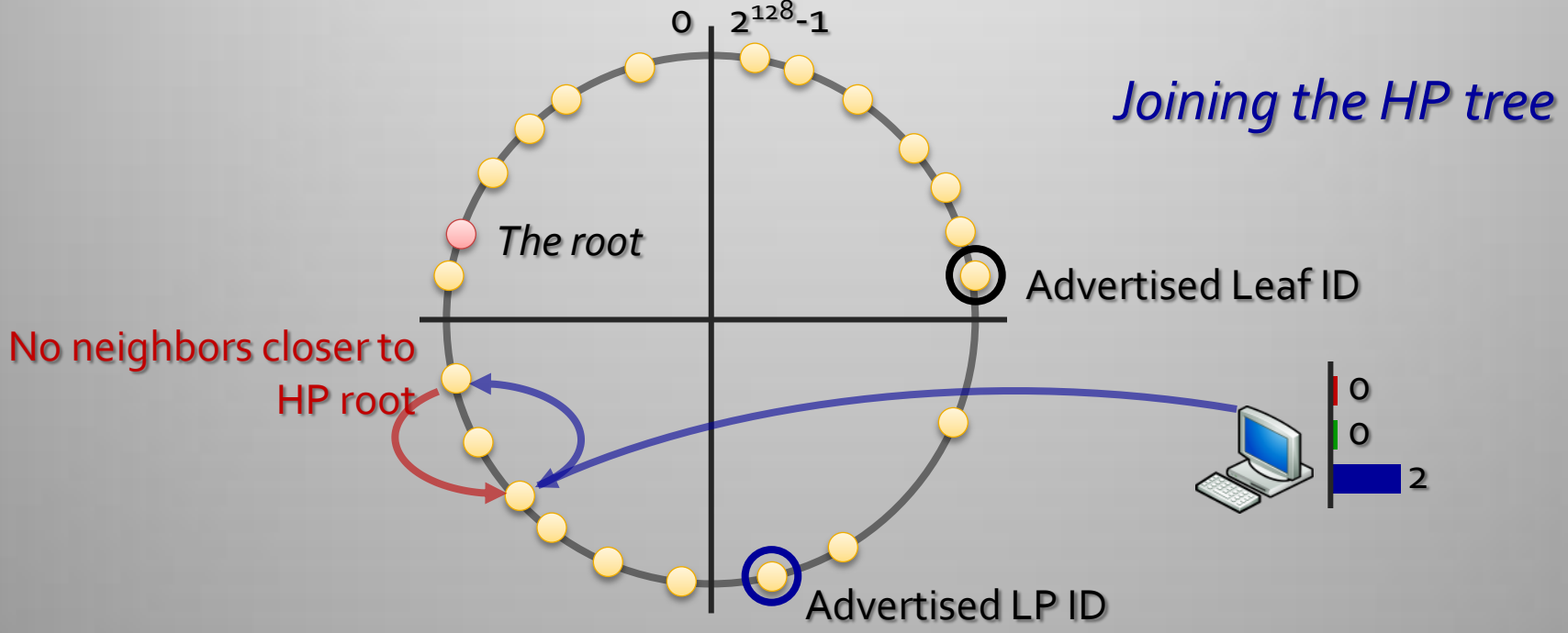
Nozzilla: Operations

- When no IDs of the target group are known
  - The initiator will use the closest neighbor
  - This neighbor is a passive peer an only forwards the request



Nozzilla: Operations

- If the one node does not have a next hop
  - It rejects the message to the last sender
  - The last sender will search an alternate path
  - If none exists, sends message back to the initiator



Performance Analysis

- Evaluate multicast tree behavior
  - In resource limited scenarios, but otherwise ideal conditions
  - Determine joining effort, geometry of multicast tree and success ratio
- Scenario
  - Each peer has resources: (0/*Res*, 0/*Res*, 0/*Res*)
  - Four scenarios: *Res* is 1, 3, 5, 7

A resource of 1 for one stripe ≈33% of the video stream bit rate

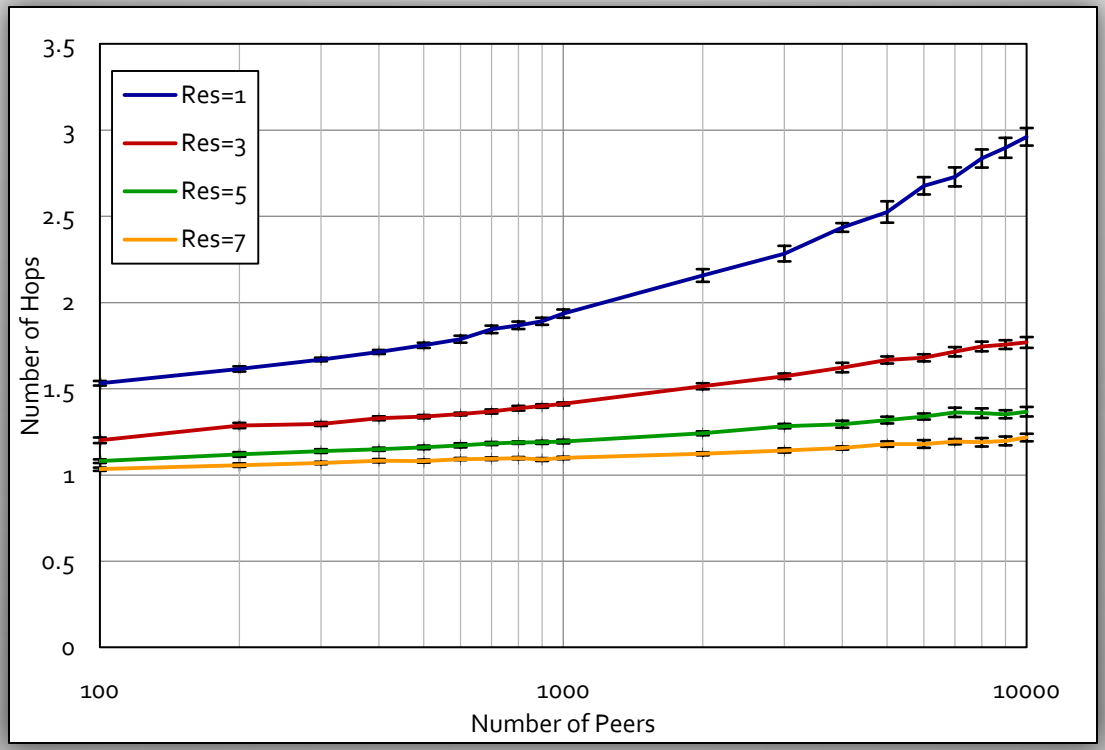
Res	Peer Total Average	Necessary Uplink
1	1.5	50%
3	4.5	150%
5	7.5	250%
7	10.5	350%

Success Ratio

- Assumed a hybrid scenario
  - The root is the media server (infinite resources)
- Does it work?
  - Join success ratio over 99.9%
  - In less than 0.1% cases the joining message was rejected back to the initiator
  - This happens mostly when *Res* is 1 and the number of peers is high
  - When Res is 1, each peer can be a parent for each stripe only once (most parents are new peers and the root)
  - After that, it leaves the peer group

Joining Tree Performance

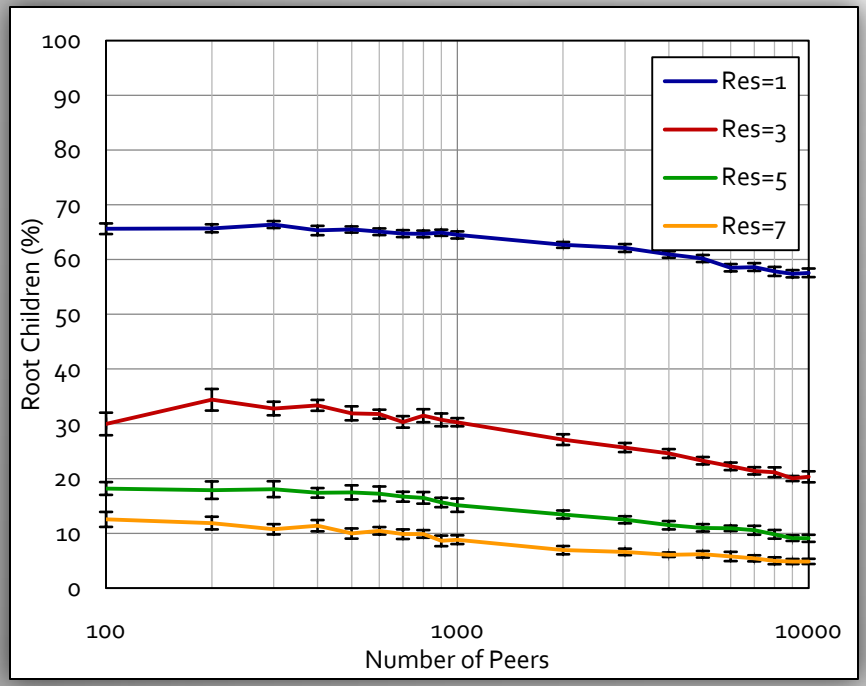
- Number of hops needed to join the tree

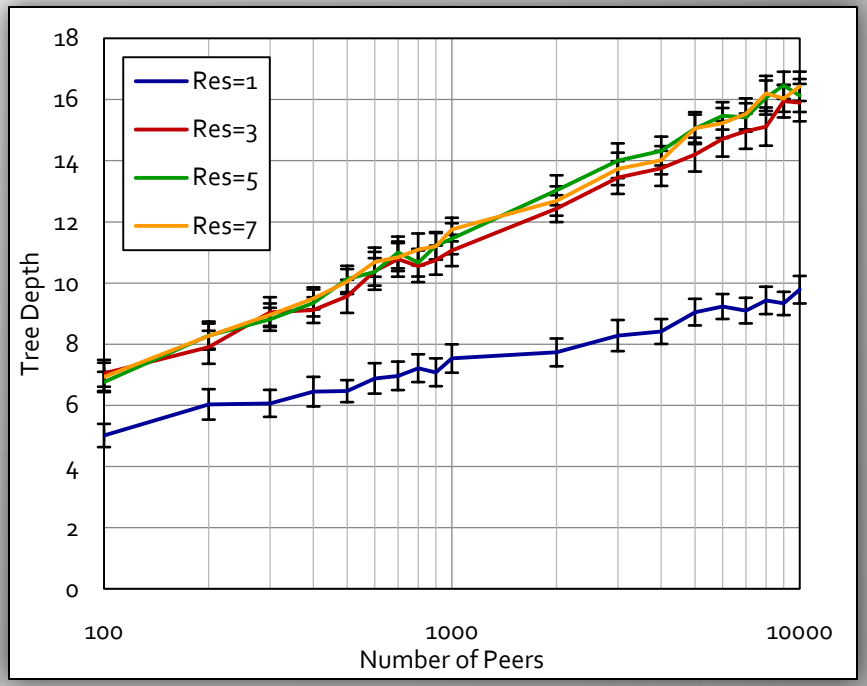


- Decreases with increasing the resources
- The improvement is significant when resources are low

Tree Geometry

- Let's see if we use P2P or client/server

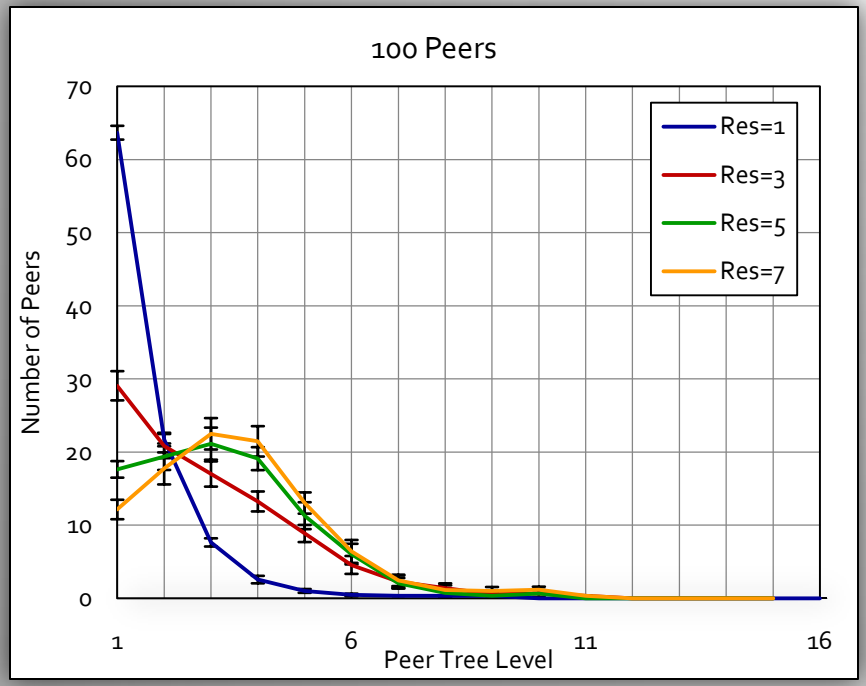


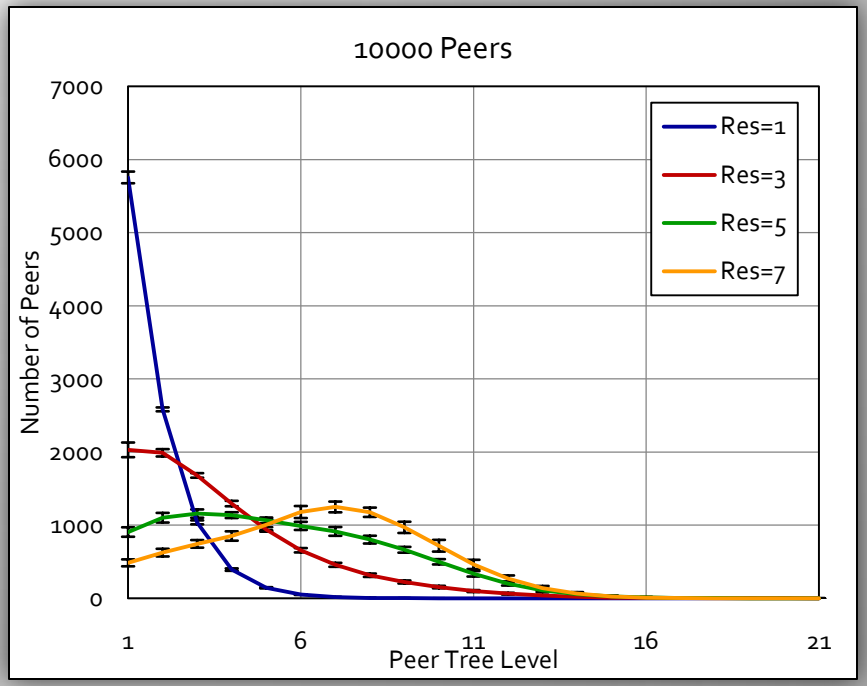


- Probably we don't want each peer to have 50% resources
- Otherwise, the root load is lower even for 10000 peers
- Tree depth is reasonable, but increases with the resources

Peer Level

- At what level are most of the peers?



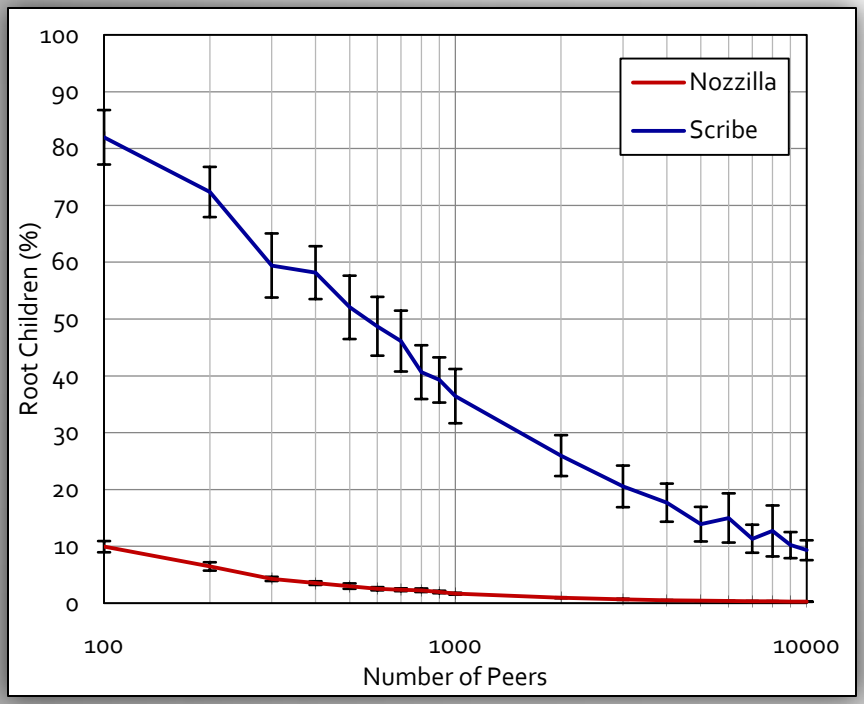


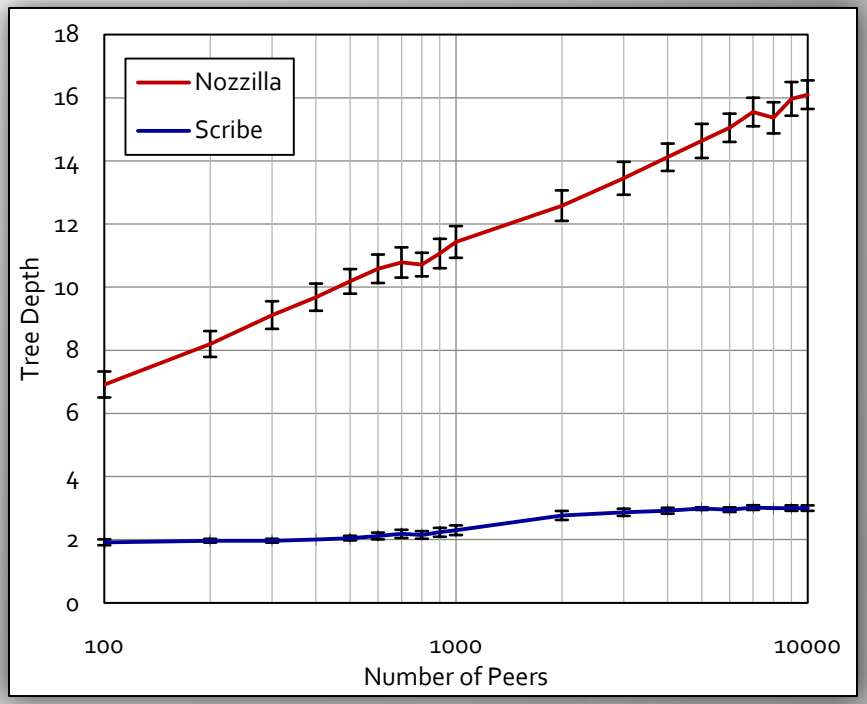
- Average peer level increases with the resources
- Takes the load off the root (media server)
- However, increases the tree depth

Nozzilla vs. Scribe

- Best-case vs. Best-case scenario
  - Scribe considers peers with infinite resources (it is built for general-purpose multicast, not video streaming)
  - When apply resource limitation, Scribe performs worse
  - If, we apply resources limitation only to Nozzilla, Scribe performs better
  - Do not apply resource limitation for both Scribe and Nozzilla
  - In this scenario the success ratio is 100% and joining performance is 1 hop

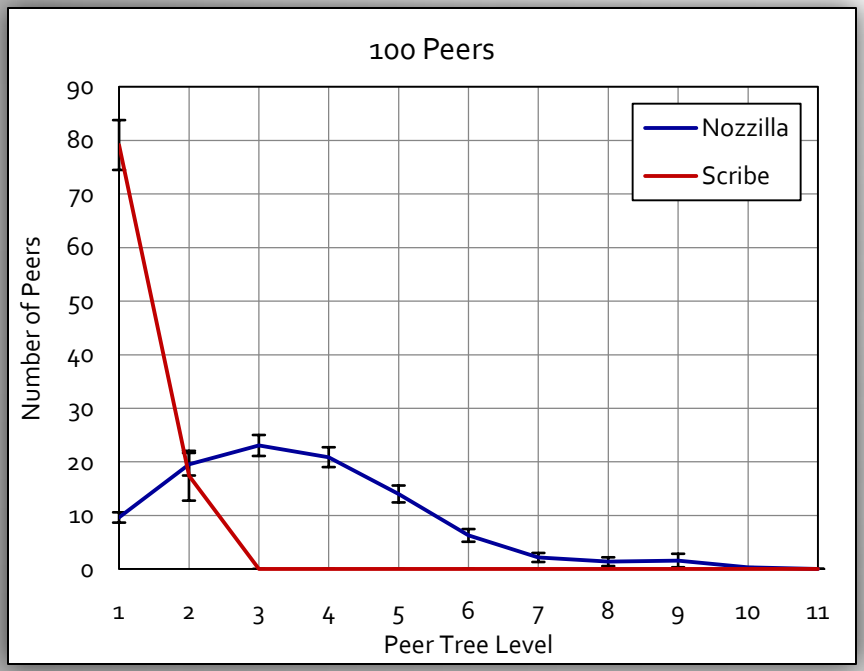
Nozzilla vs. Scribe (tree geometry)

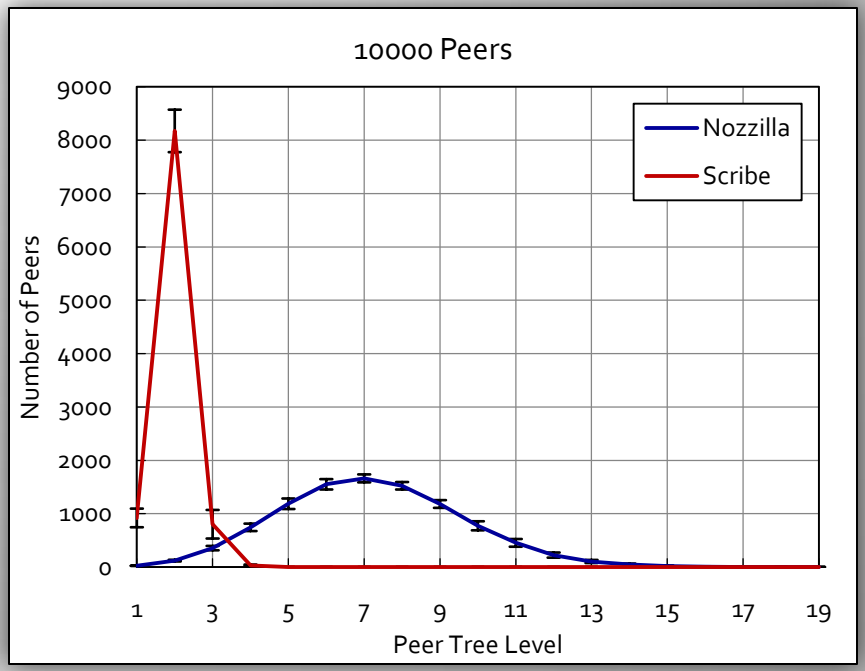




- For root children… Scribe is worse, Nozzilla is better
- The random first hop selection algorithm pays off
- But… tree depth is higher for Nozzilla

Nozzilla vs. Scribe (Peer Level)





- The average peer level is higher for Nozzilla
- Higher depth but lower root load

Summary

- Characteristics
  - P2P protocol to create multicast trees for video streaming
  - Multi-path video delivery (multiple stripes)
  - Takes into account uplink resources
  - Changes the geometry of the multicast tree to decrease the root load (enables hybrid topologies)
- Behavior
  - Excellent success ratio, low joining effort
  - Low root load for reasonable resources
  - Lengthier video path, may impact reliability

Future Work

- Improve responsiveness when peers leave
  - Unlike SplitStream, P2PCast no intelligent mechanism is used
- Extend random selection algorithm for intermediate hops
- True path diversity using a soft state
  - At least initiators should remember the path used and in case of rejection should retry with a different path
- Experimental analysis against other similar proposals


Q&A

ThankYou