ForestCast for P2P Live Streaming

Amir H. Payberah (amir@sics.se)

Fatemeh Rahimian (fatemeh@sics.se)

Supervisor: Ali Ghodsi (ali@sics.se)

Examiner: Seif Haridi (seif@sics.se)

What is the Problem?

- Media Streaming over Internet is getting more popular everyday.
- Conventional solution:
  - Client-Server model
  - Very expensive

- Another solution:
  - Peer-to-Peer overlay

Outline

- Introduction
- Related Work
- Our Solution (ForestCast)
- Our Simulator (SICSSIM-B)
- Evaluation
- Conclusion

Peer-to-Peer Overlay

- A type of overlay in which each peer simultaneously functions as both client and server to the other peers on the network.

- Each peer contributes its own resources.
  - The capacity of whole system grows when the number of peers increases

P2P Media Streaming

- The peers who have parts of the media can forward it to other requesting peers.

- Media streaming
  - Live media streaming
  - Video on Demand (VoD)

P2P for Live Media Streaming

- Bandwidth intensive.
- Data should be received with respect to certain timing constraints.
  - A negligible startup delay
  - Smooth playback
  - A negligible playback latency
- Nodes join, leave and fail continuously.
  - Called churn
- Network capacity changes.

Our Contribution

- We present ForestCast, a peer-to-peer live media streaming system.
- Within ForestCast, we have proposed a number of heuristics and examined their impact on the quality of service experienced by clients.
- To evaluate ForestCast, we have implemented a peer-to-peer simulator, called SICSSIM-B.

Outline

- Introduction
- Related Work
- Our Solution (ForestCast)
- Our Simulator (SICSSIM-B)
- Evaluation
- Conclusion

Related Work

- SplitStream
- Coolsteraming
- CoopNet
- Orchard
- Bullet
- Prime
- Pulsar
- NICE
- Zigzag
- DirectStream
- MeshCast

- mTreeBone
- PULSE
- GnuStream
- SAAR
- ChainSaw
- ChunkySpread
- BulkTree
- GoCast
- AnySee
- DagStream
- Climber

- CollectCast
- HyMoNet
- GridMedia
- ScatterCast
- SpiderCast
- Yoid
- Narada
- Zebra
- ...

peercast.org
p2p broadcasting for everyone

PPStream
www.ppstream.com

Two Main Questions

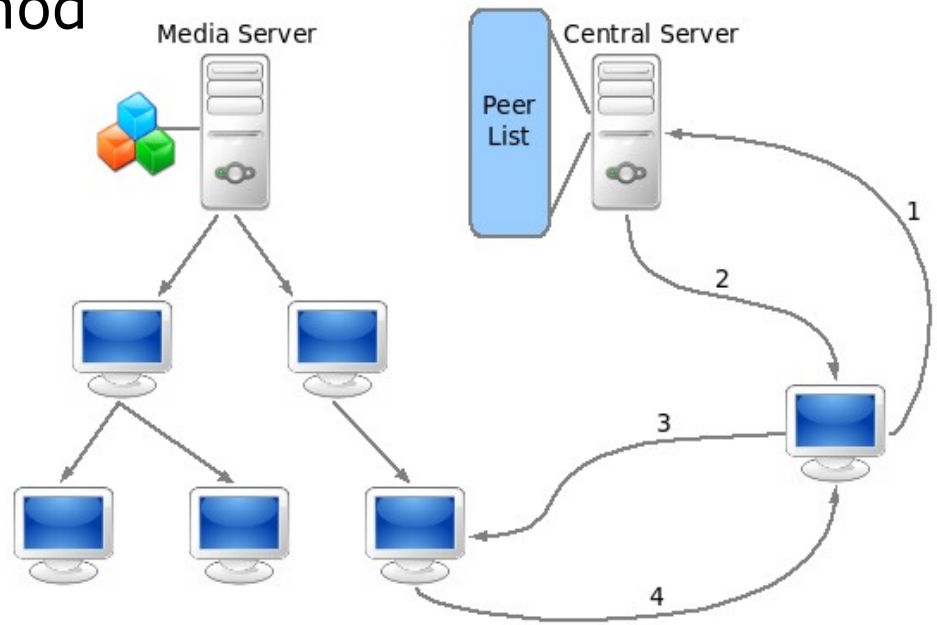
- How to find supplying peers?
- How to deliver content to peers?

Finding Supplying Peers

- Centralized method
- Hierarchical method
- DHT-based method
- Controlled flooding method
- Gossip-based method

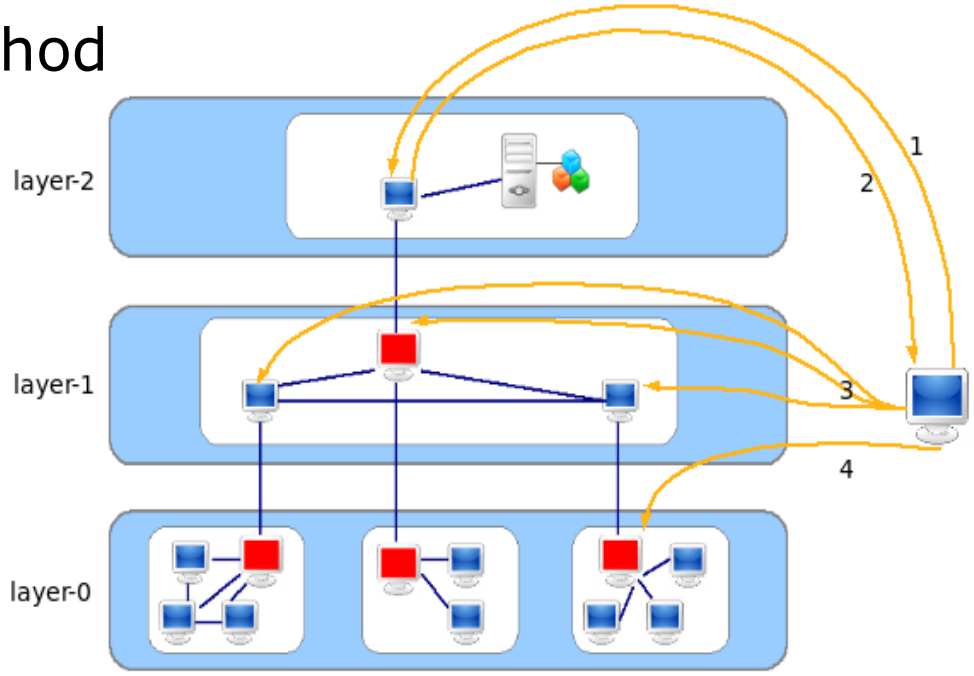
Finding Supplying Peers

- Centralized method
- Hierarchical method
- DHT-based method
- Controlled flooding method
- Gossip-based method



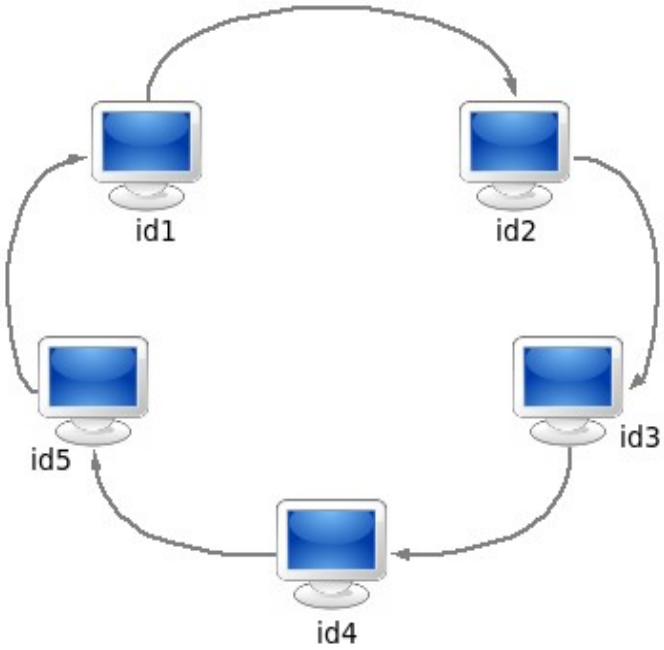
Finding Supplying Peers

- Centralized method
- Hierarchical method
- DHT-based method
- Controlled flooding method
- Gossip-based method



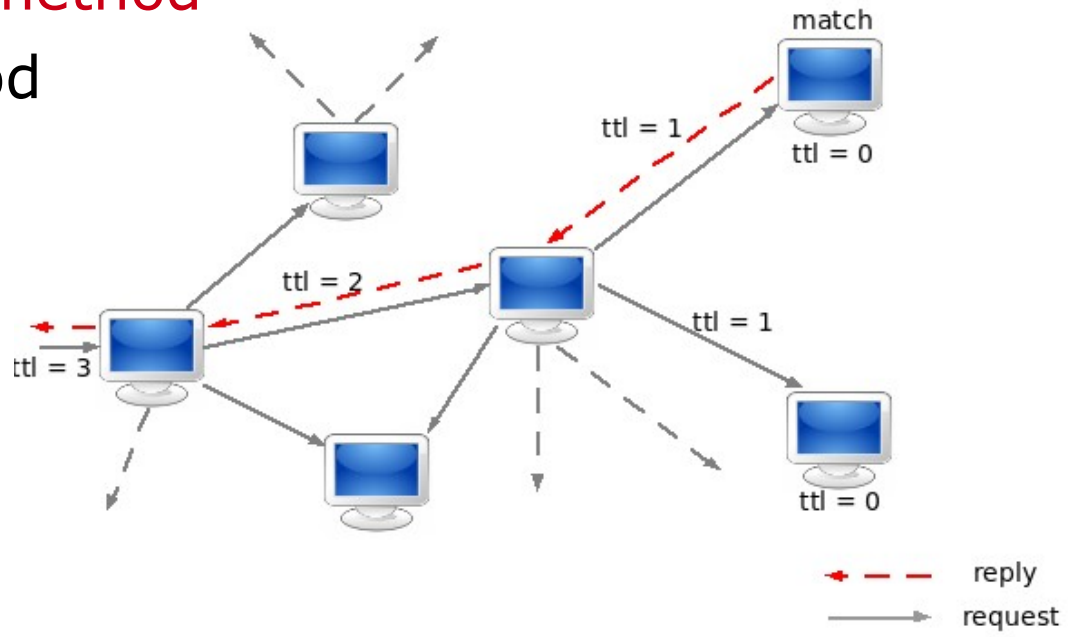
Finding Supplying Peers

- Centralized method
- Hierarchical method
- DHT-based method
- Controlled flooding method
- Gossip-based method



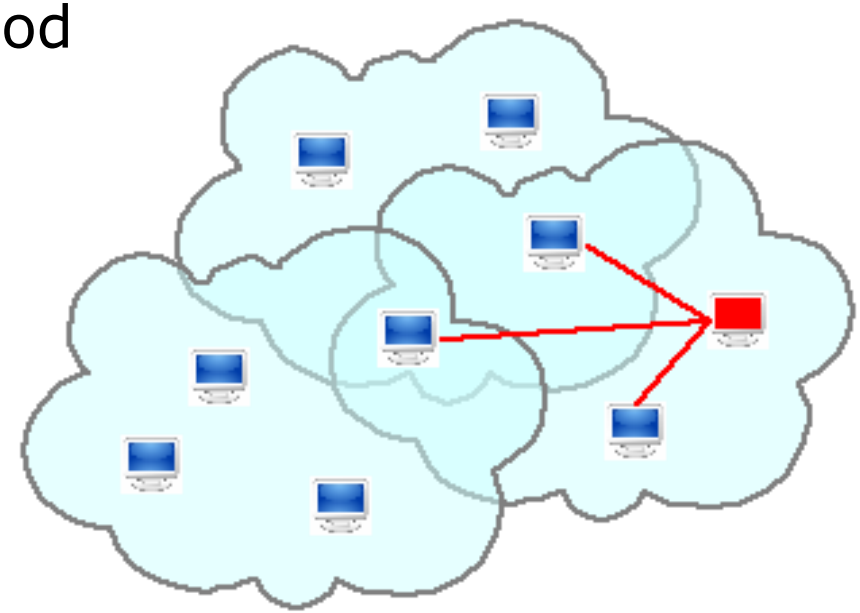
Finding Supplying Peers

- Centralized method
- Hierarchical method
- DHT-based method
- Controlled flooding method
- Gossip-based method



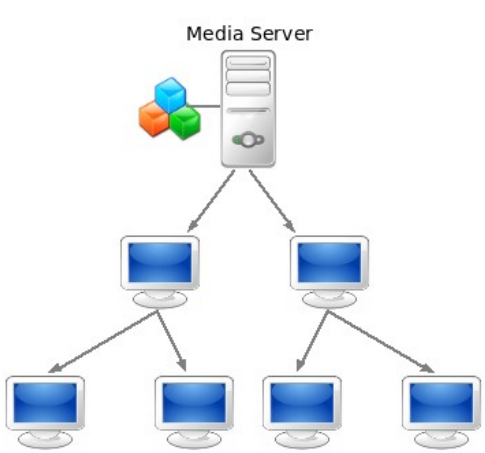
Finding Supplying Peers

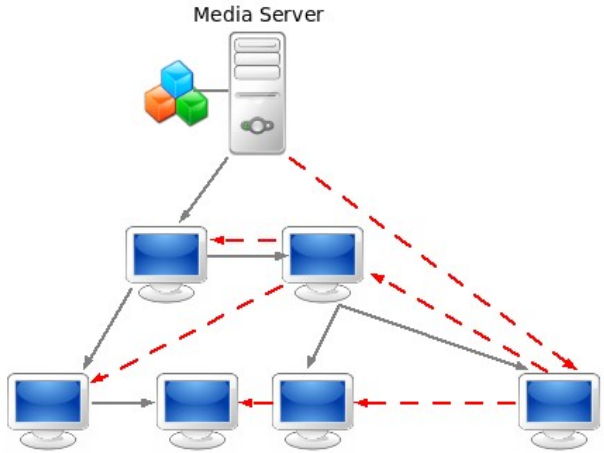
- Centralized method
- Hierarchical method
- DHT-based method
- Controlled flooding method
- Gossip-based method



Data Delivery

- Push method
  - Single tree
  - Multiple trees





- Pull method
- Push-Pull method

Related Work

Finding supplying peers \ Data Delivery	Push method (Single tree)	Push method (Multiple trees)	Pull method	Push-Pull method
Centralized method	DirectStream (2006)			Prime (2007) mTreeBone (2007)
Hierarchical method	ZigZag (2003)			mTreeBone (2007)
DHT-based method	SAAR (2007)	SAAR (2007) SplitStream (2003)	SAAR (2007)	Pulsar (2007) mTreeBone (2007)
Controlled flooding method			GnuStream (2003)	
Gossip-based method		Orchard (2006) ChunkySpread (2006)	CoolStreaming (2005) PULSE (2006) ChainSaw (2005) PPLive (2004)	Bullet (2003)

Outline

- Introduction
- Related Work
- Our Solution (ForestCast)
- Our Simulator (SICSSIM-B)
- Evaluation
- Conclusion

ForestCast

Finding supplying peers \ Data Delivery	Push method (Single tree)	Push method (Multiple trees)	Pull method	Push-Pull method
Centralized method	DirectStream (2006)	ForestCast		Prime (2007) mTreeBone (2007)
Hierarchical method	ZigZag (2003)			mTreeBone (2007)
DHT-based method	SAAR (2007)	SAAR (2007) SplitStream (2003)	SAAR (2007)	Pulsar (2007) mTreeBone (2007)
Controlled flooding method			GnuStream (2003)	
Gossip-based method		Orchard (2006) ChunkySpread (2006)	CoolStreaming (2005) PULSE (2006) ChainSaw (2005) PPLive (2004)	Bullet (2003)

Objectives

- Maximize the total utilization of upload bandwidth
- Maximize the received quality
- Minimize the playback latency
- Minimize the startup delay

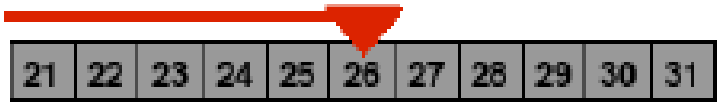
ForestCast is a solution to heuristically move towards these objectives, and enable providers to discover a solution which best fits their goals.

Some Definitions

- Three types of node:
  - Media server
  - Central server
  - Peer
- The media server uses MDC to split the stream into substreams of equal-size and equal-value, called stripes.
  - One tree to multicast each stripe.
- The stripes are fragmented into equal-size segments.
  - These segments are sequentially numbered by the media server.

Some Definitions

- Startup segment
- Head of buffer
- Playback point
- Head-to-play distance
- Media point
- Playback latency



Some Definitions

- Open node
- Node profile
  - Playback point
  - Stripes the node is receiving
  - Head of buffer for each stripe
  - Parent for each stripe
  - Latency to each parent
  - ...

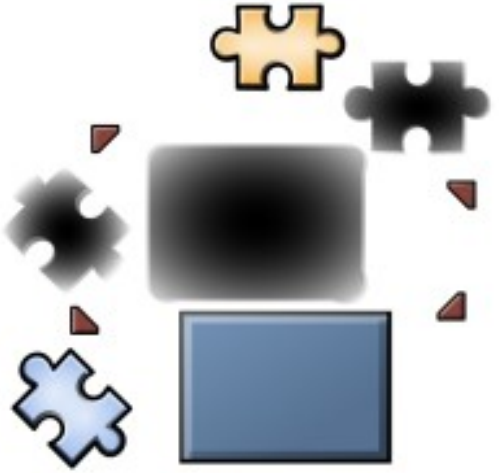
Join Procedure

Join Procedure

- Step 1. Server finds a list of open nodes in each stripe tree,
- Step 2. It assigns a priority to the selected open nodes and selects the most appropriate parent for each stripe,
- Step 3. Decides about a startup segment,
- Step 4. Decides about the head-to-play distance of the peer.

Heuristics for Join Procedure

- Node collection
- Parent Selection
- Startup segment
- Buffering delay



Leave Procedure

Leave Procedure

- Step 1. For each child, server finds a substitute parent, such that the child does not miss any segment.
- Step 2. If a substitute parent could be found for all the children, grants the requesting node to leave the system. Otherwise retries after a short interval.

Failure Handling

Failure Handling

- Step 1. For each child, if possible server finds a substitute parent, such that the child does not miss any segment, otherwise it finds a parent which causes the least missing segments.
- Step 2. If some of the children are still parent-less, retries after a short interval.

Incremental Improvement

Main Idea

- Nodes with higher bandwidth are better to be close to the media source.
- Stable nodes are better to be close to the media source.
- Node strength
  - $Strength(A_i) = age_A . (fanout_{Ai} + freeBw_A)$
- The stronger nodes gradually bubble up the trees.
  - Eventually we end up in a layout, in which node distances from the root of trees are in the order of their decreasing strength.
- The algorithm has two steps:
  - Promotion
  - Reconfiguration

Outline

- Introduction
- Related Work
- Our Solution (ForestCast)
- Our Simulator (SICSSIM-B)
- Evaluation
- Conclusion

SICSSIM-B

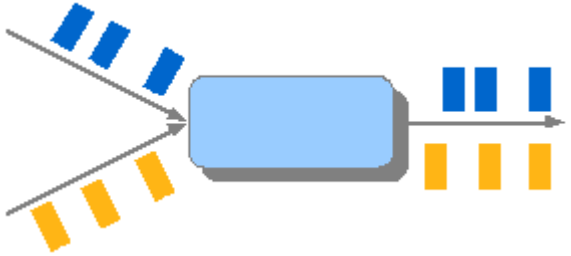
- SICSSIM-B
  - Discrete-event
  - Flow-level
  - Models
    - Link latencies
    - Bandwidth
    - Congestion

Discrete-event Modelling

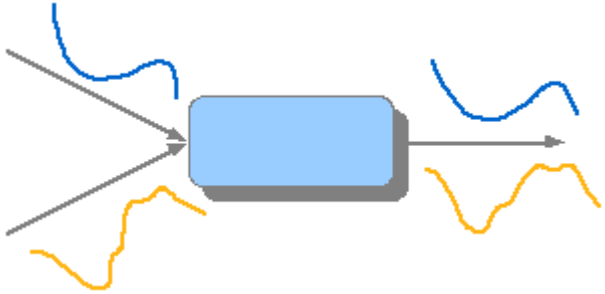
- A common method of simulating networks
- The operation of a system is represented as a sequence of events in time order.
- Each event occurs at a point in time and makes a change of state in the system.

Packet-level vs. Flow-level

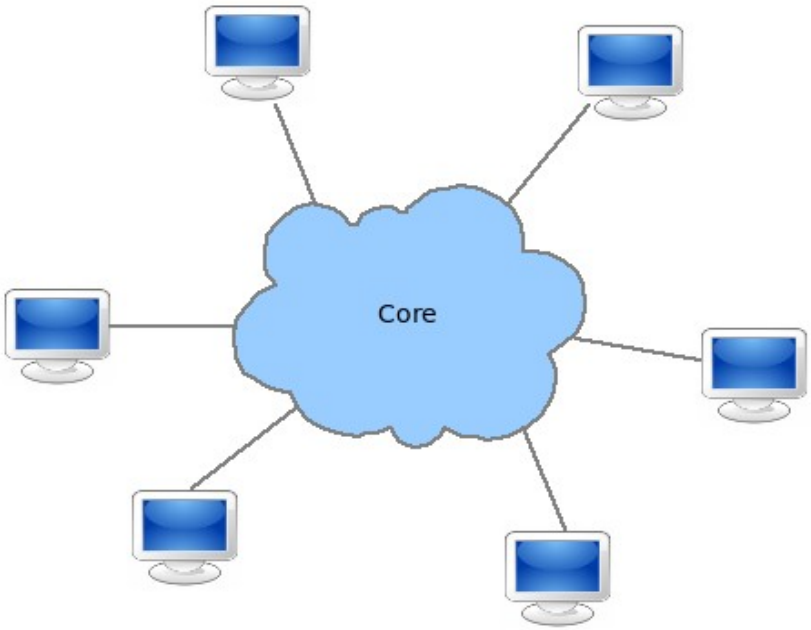
- Packet level modelling
  - For each packet departure or arrival one event will be generated.



- Flow level modelling
  - The events are generated only when the rate of flows changes.



Modelling Latency, BW and Congestion



	A	B	C	D	Total Upload
A		56	0	0	56
B	128		1000	56	2000
C	256	1500		0	5000
D	0	0	56		56
Total Download	512	2000	10000	56	

Messages in SICSSIM-B

- Control messages
  - Join, leave, failure, send data, receive data, congestion and ...
  - They are very small size packets (zero bandwidth).
  - To handle them we put them in FEL.

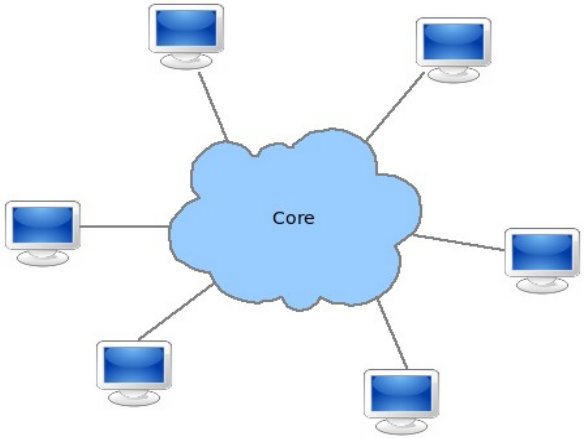
- Data messages
  - Data messages carry the real media.
  - We don't transfer data in the simulator.
  - We just assume there is a flow of data from nodes to nodes which is marked by a number of control messages.
    - Send data control message
    - Receive data control message
    - Change the rate control message

Outline

- Introduction
- Related Work
- Our Solution (ForestCast)
- Our Simulator (SICSSIM-B)
- Evaluation
- Conclusion

Experimental Setting

- The stream is split into four stripes by MDC.
  - Rate of each stripe: 128Kbps
- The delay between two peers:
  - 95% of delays are between 20ms and 160ms with normal distribution.
  - The value is almost the same for a pair of nodes.



Experimental Setting (Cont'd)

- Three types of event:
  - Join, Leave and Failure
- We use poisson distribution to model the interval between each two events.
  - Averagely 10 events per second.
- Three types of scenario:
  - Join only
  - Low rate departure (Low churn)
  - High rate departure (High churn)

Experimental Setting (Cont'd)

- Upload bandwidth distribution follows the real measurements in live streaming.

Type	Degree bound	Number of hosts	Upload bandwidth (Kbps)
Free riders	0	58646 (49.3%)	0 - 249
Contributors	1	22264 (18.7%)	250 - 499
Contributors	2	10033 (8.4%)	500 - 749
Contributors	3 - 19	6128 (5.2%)	750 - 4999
Contributors	20	8115 (6.8%)	5000
Unknown	-	13735 (11.6%)	-
Total	-	118921 (100%)	-

Different Simulations

- Effect of different heuristics.
- Select the best heuristics and investigate the system property in scale.
- Examine the impacts of adding an incremental improvement.

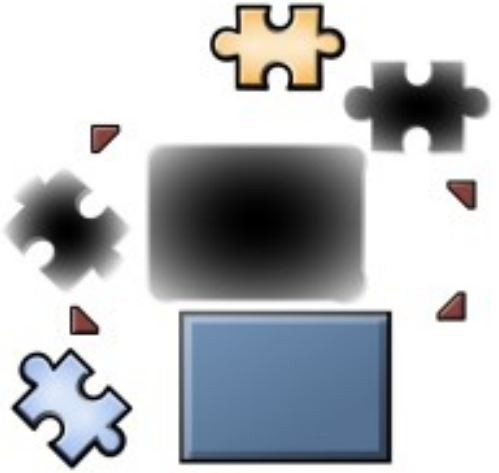
Some of the Metrics

- Bandwidth utilization
- Tree depth
- Startup delay
- Playback latency
- Quality
- Disruption

Different Heuristics

Heuristics for Join Procedure

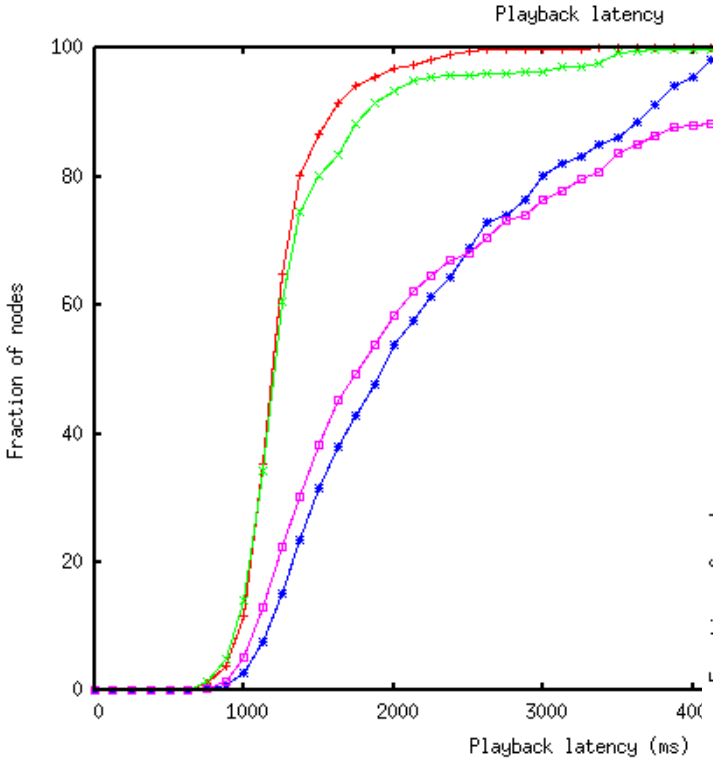
- Node collection
- Parent selection
- Startup segment
- Buffering Delay

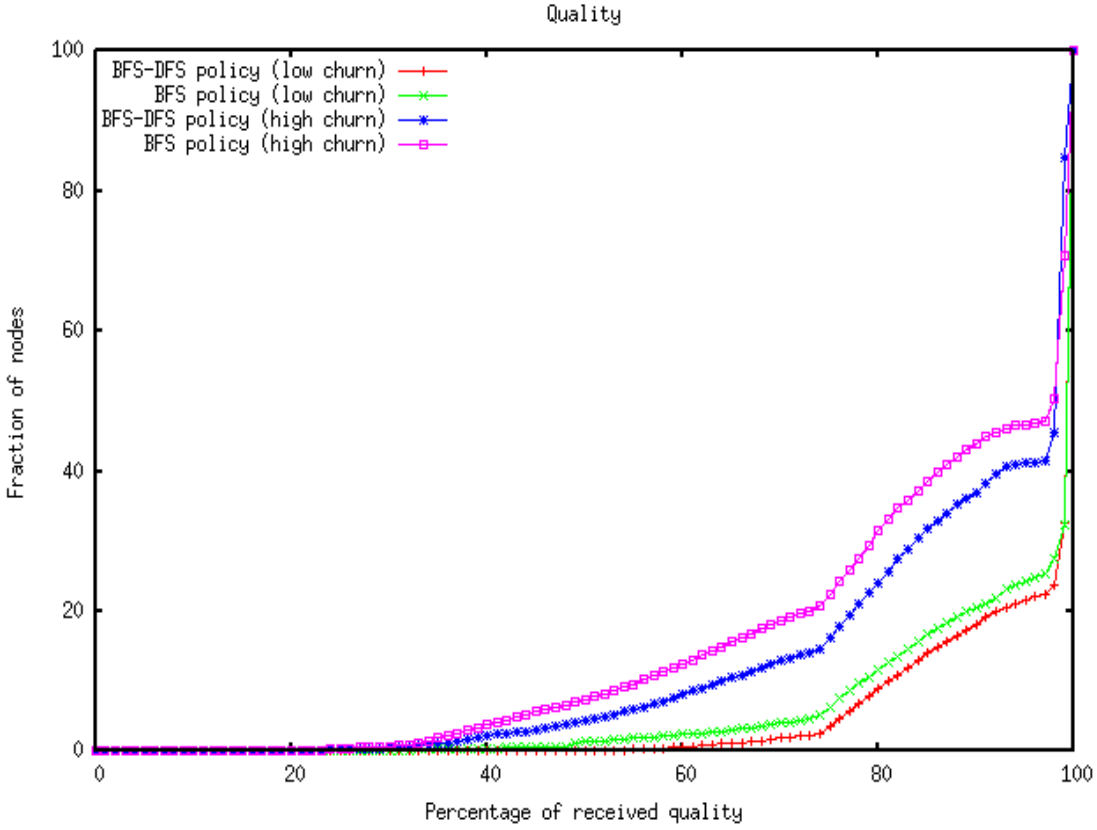


Node Collection

- Two policies:
  - BFS policy
  - BFS-DFS policy
    - DFS for free riders
    - BFS for the others

Node Collection





Parent Selection (Priority Function)

- Two Policies:
  - Any-Stripe policy: $\Phi(b, l) = b^{\alpha} / l^{\gamma}$
  - Rarest-Stripe policy: $\Phi(b, f, l) = b^{\alpha} / (f^{\beta} . l^{\gamma})$

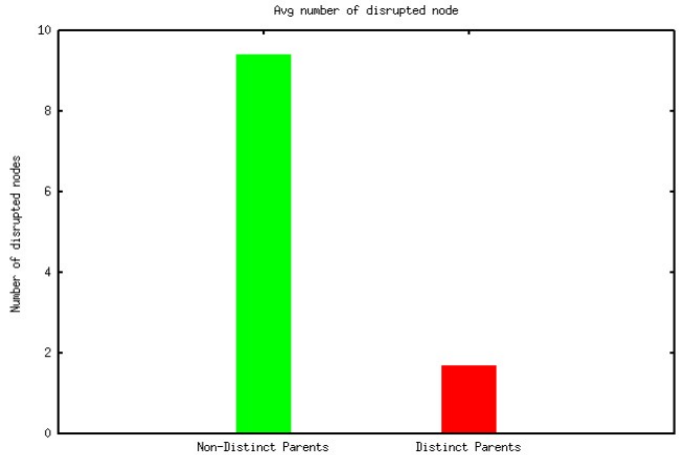
- Example scenario:
  - Suppose two stripes are in system: stripe1 and stripe2.
  - Peer P is the only open node in system that can provide stripe2.
  - If P uses all its upload bandwidth to transfer stripe1 then stripe2 will become unavailable in system.

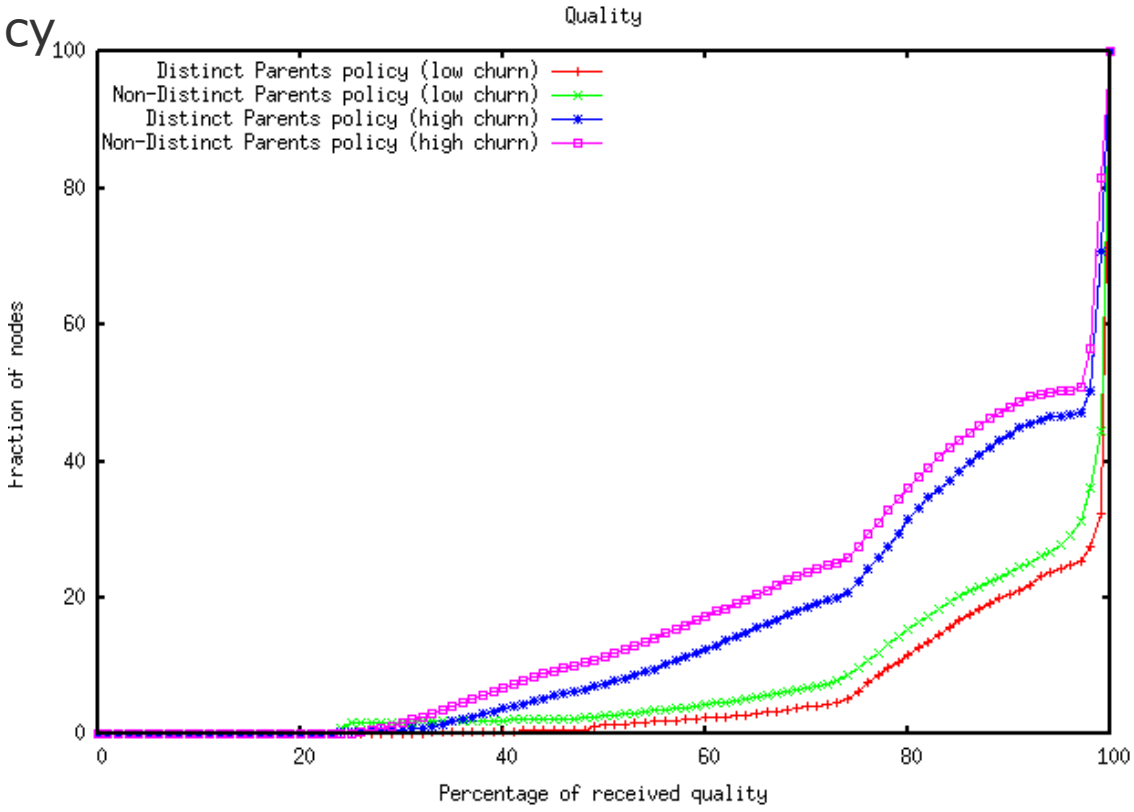
Parent Selection (Priority Function)

- By using Any-Stripe policy, in 30% of experiments at least one tree was saturated.
  - Degraded quality
  - Un-utilized bandwidth

Parent Selection (Distinctness)

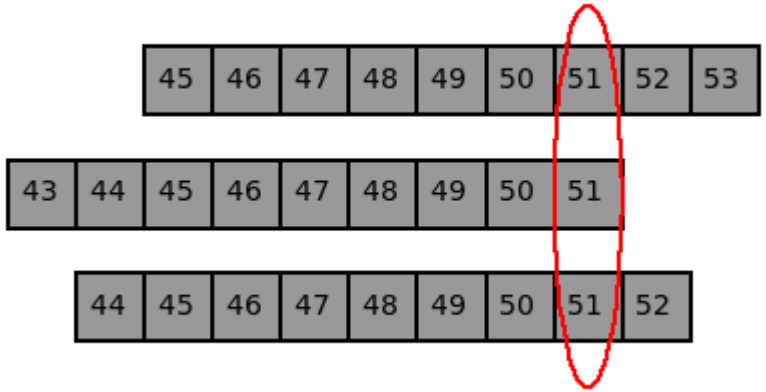
- Two Policies:
  - Non-Distinct Parents policy
  - Distinct Parent policy

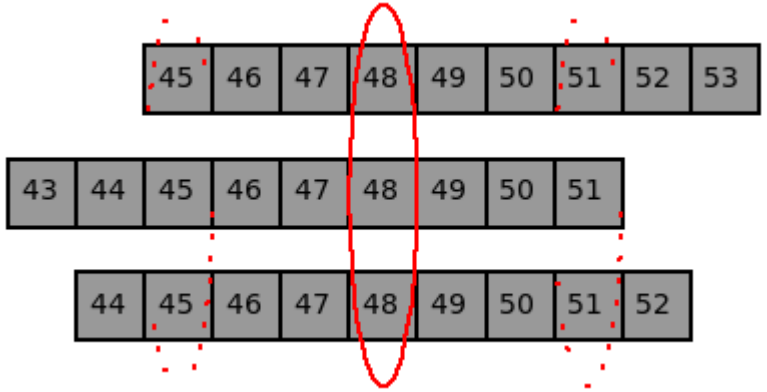




Startup Segment

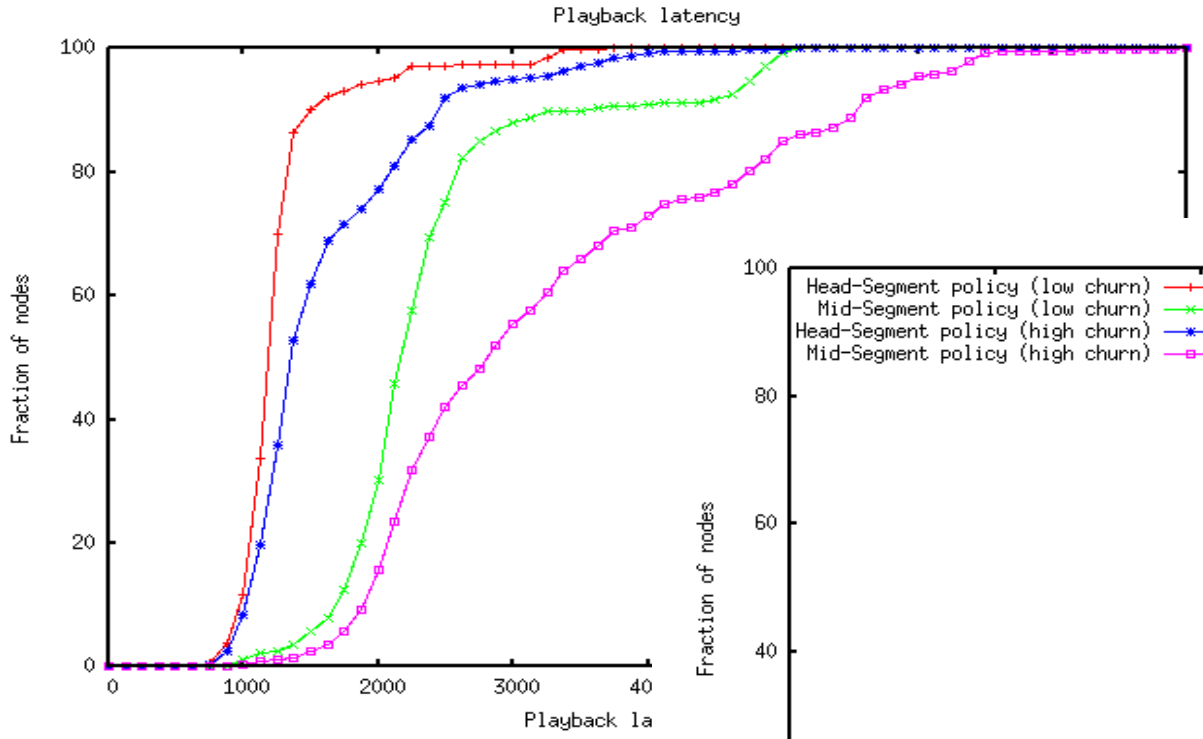
- Head-Segment policy

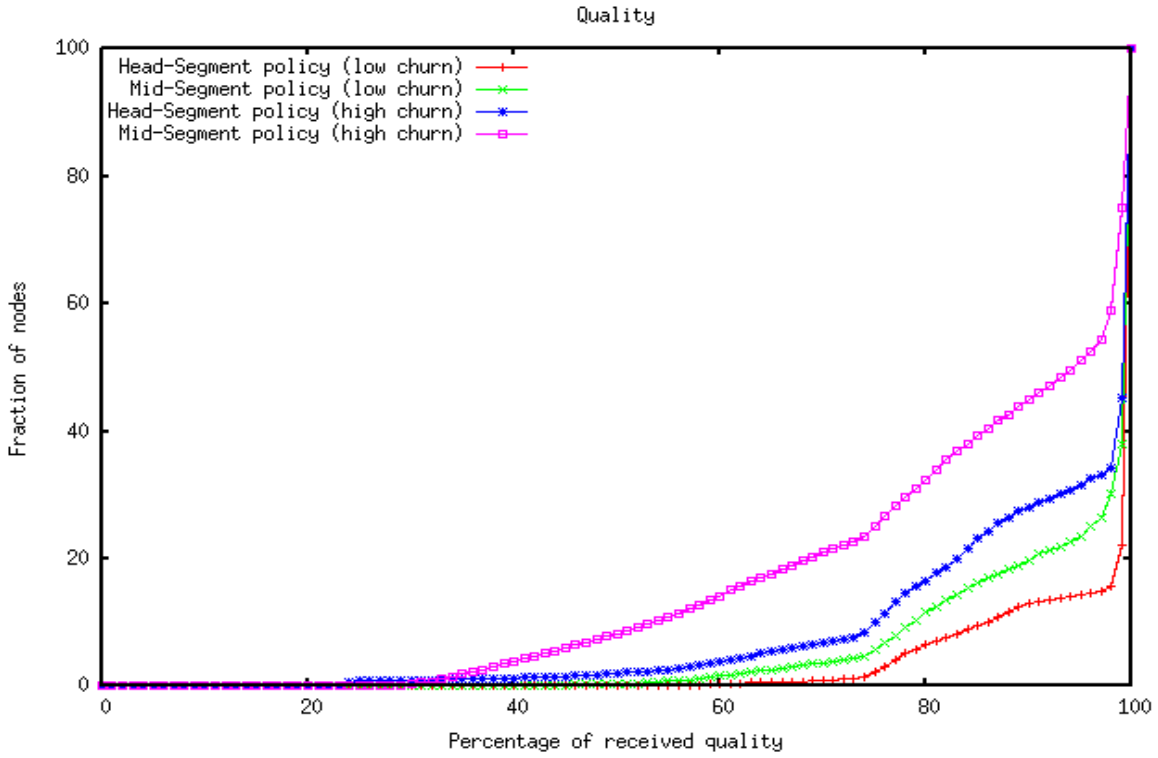




- Mid-Segment policy

Startup Segment





Measurement at Scale

Selected Heuristics

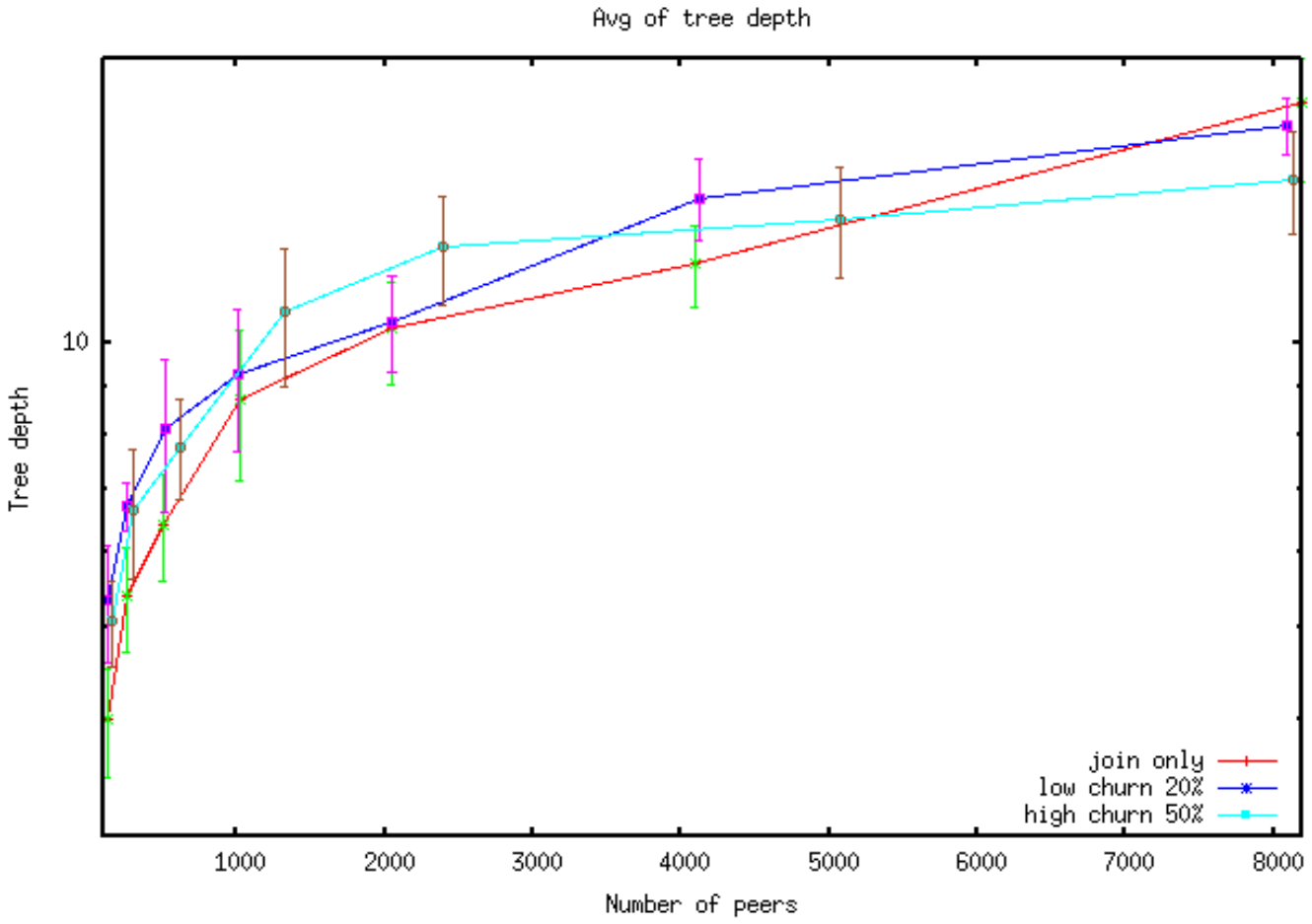
- Node collection:
  - BFS-DFS policy
- Parent Selection:
  - Rarest-Stripe policy
  - Distinct parent policy
- Startup Segment:
  - Head-Segment policy

Bandwidth Utilization

- It is defined as the ratio of total utilized upload bandwidth to the total demanded download bandwidth.

- Bandwidth utilization in ForestCast is almost 100%.

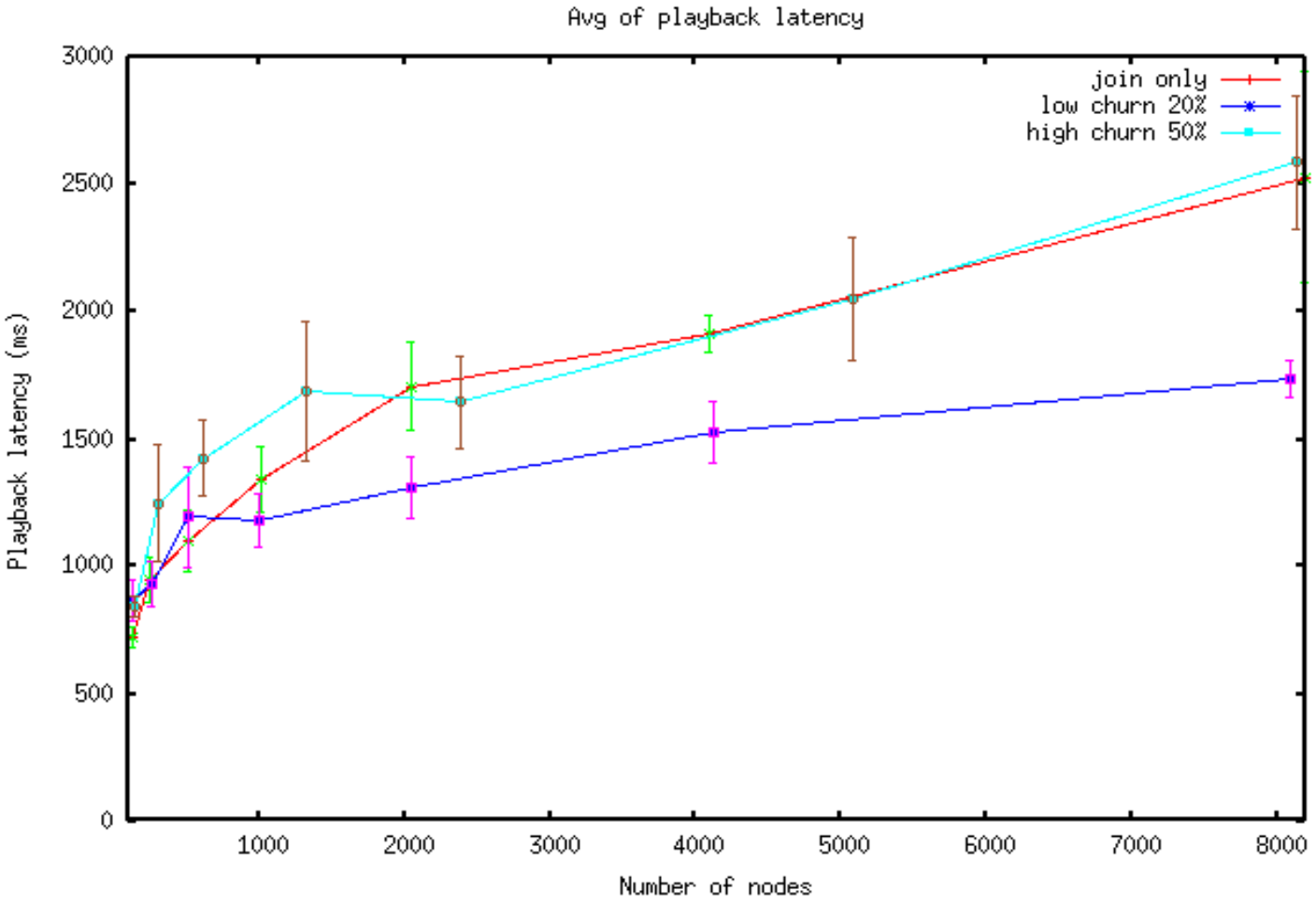
Tree Depth


Avg of tree depth
Tree depth
10
1000
2000
3000
4000
5000
6000
7000
8000
Number of peers
join only
low churn 20%
high churn 50%

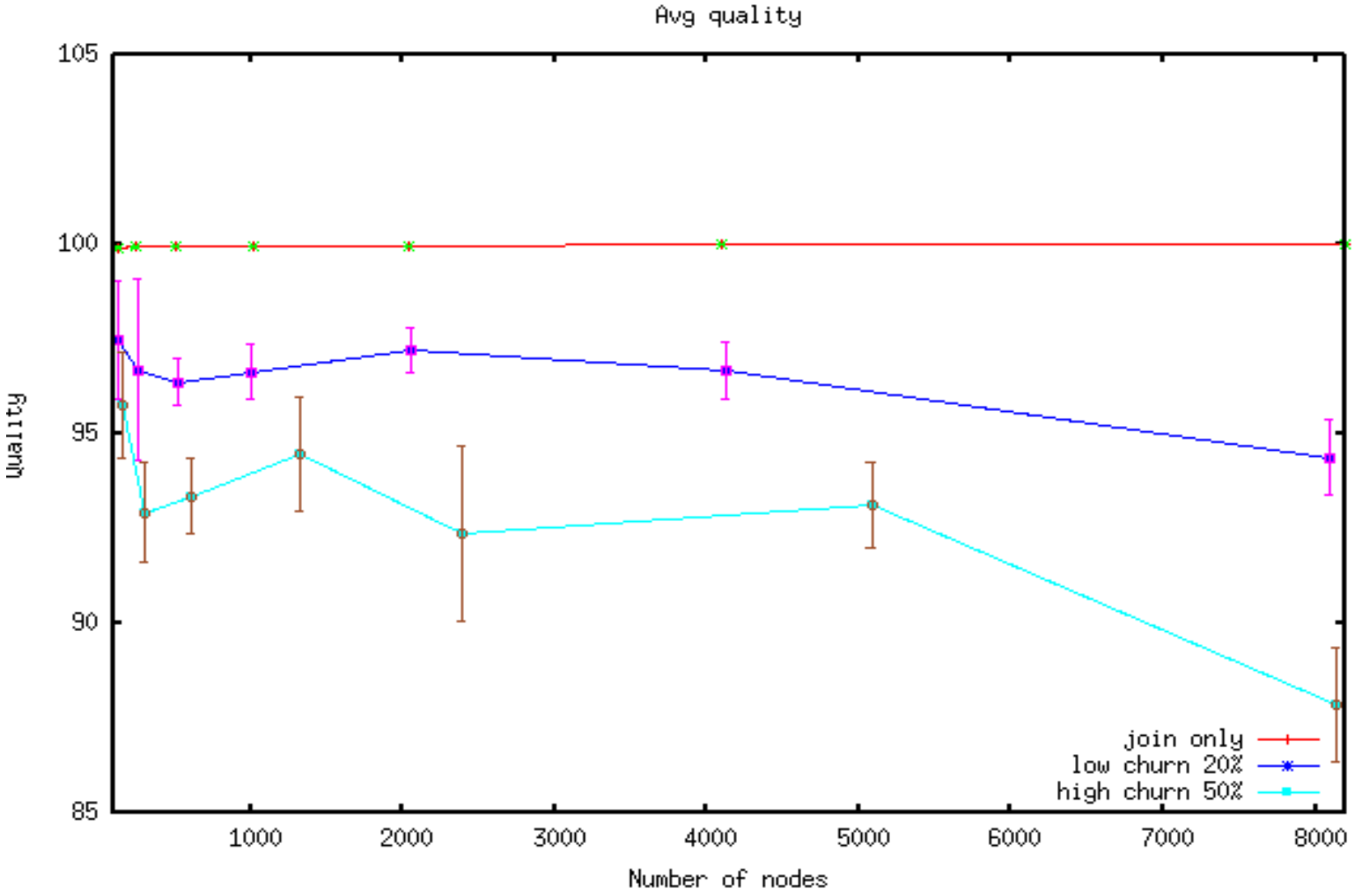
Startup Delay

- Startup delay = Time to join + Buffering delay.
- Time to join is almost independent of the network size.
  - Less than 200ms in worse case.
- Buffering delay is a trade-off between playback latency and received quality.

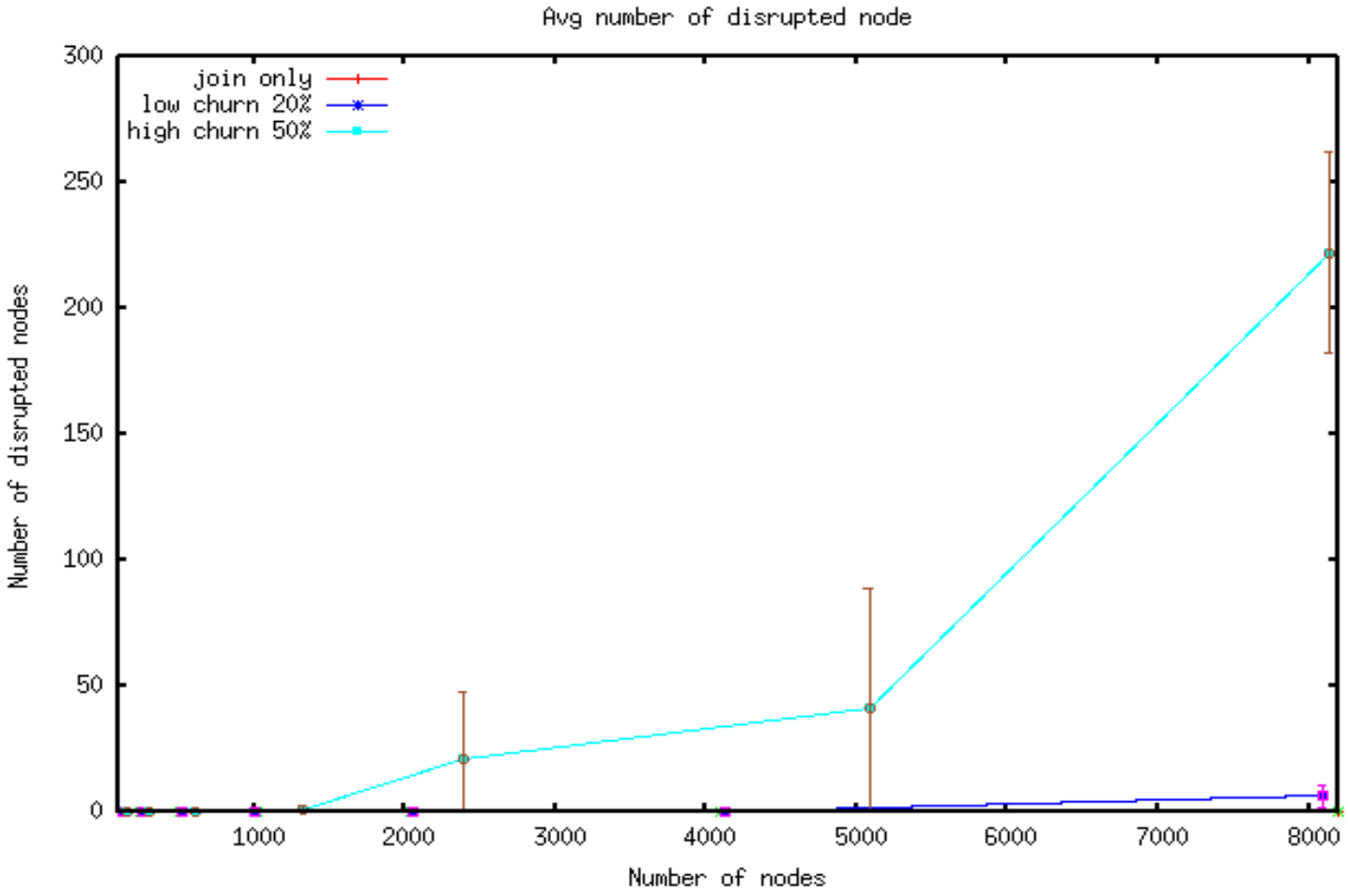
Playback Latency



Received Quality



Disrupted Nodes



Impact of Adding Incremental Improvement

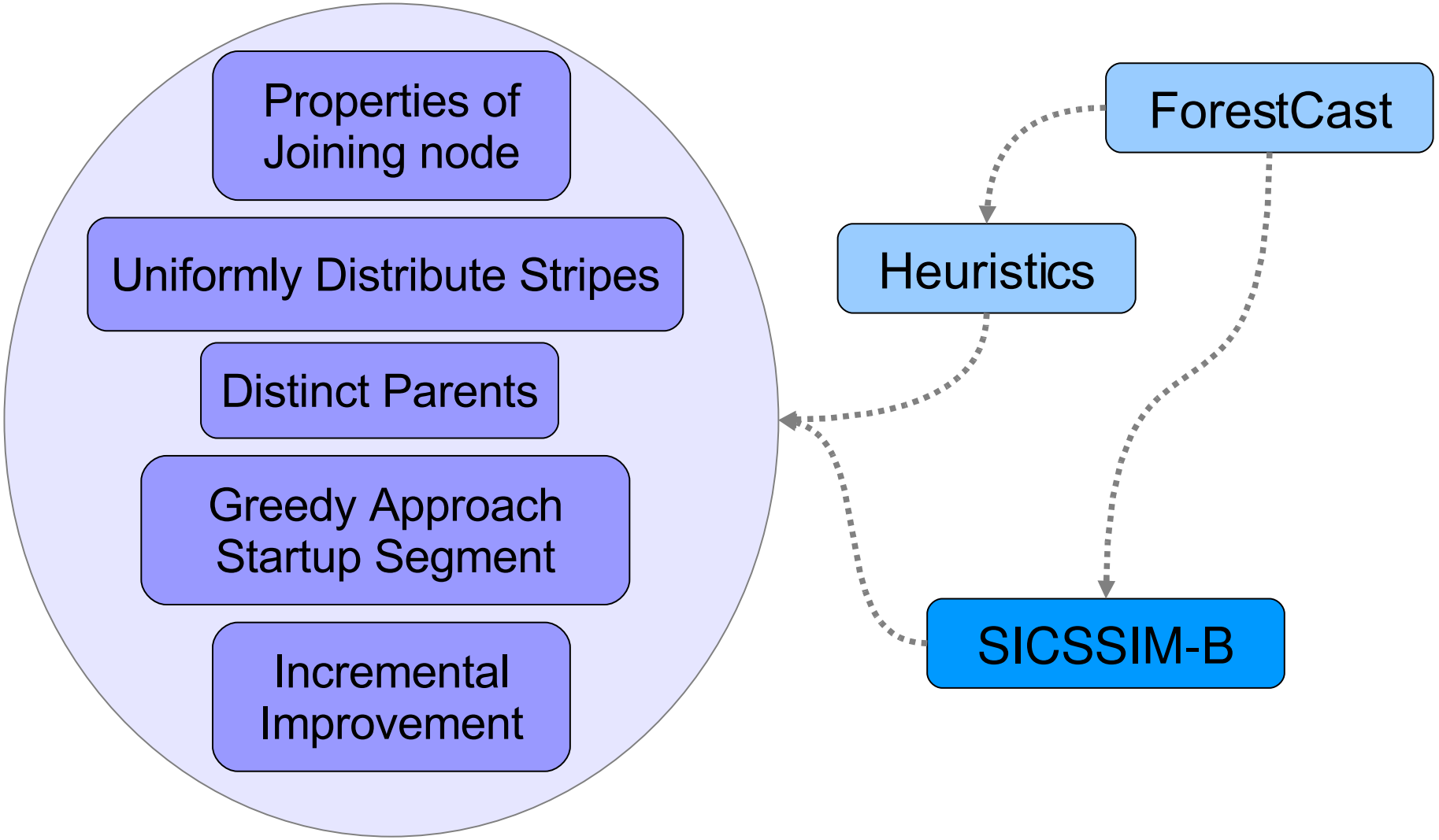
Impact of Incremental Improvement

- Tree depth is nearly less than half compared to when we had no improvements.
- The average value for received quality improves.

Outline

- Introduction
- Related Work
- Our Solution (ForestCast)
- Our Simulator (SICSSIM-B)
- Evaluation
- Conclusion

Conclusion


Properties of Joining node
Uniformly Distribute Stripes
Distinct Parents
Greedy Approach Startup Segment
Incremental Improvement
ForestCast
Heuristics
SICSSIM-B

Thanks for your attention :)