P2P Live Streaming

Amir H. Payberah
amir@sics.se

Fatemeh Rahimian
fatemeh@sics.se

Outline

- Introduction
- Related Works
- ForestCast
- Simulation
- Summary

Outline

- Introduction
- Related Works
- ForestCast
- Simulation
- Summary

What is the Problem?

- Growing interest in networked multimedia streaming application.
- Simplest solution:
  - Allocate server and network resources for each client request.
  - Does not scale well.
- Better solution:
  - Peer-to-Peer technologies

Peer-to-Peer Technology

- A type of network in which each peer has equivalent capabilities and responsibilities.

- Popular for many scalable applications
  - Multicasting
  - File sharing

P2P Media Streaming

- The peers who have all or part of the requested media can forward the data to requesting peers.

- The requesting peers can become supplying for other requesting peers.

- Each peer contributes its own resources.
  - The capacity of whole system becomes much more than the client-server model.

PPStream
www.ppstream.com

PPLIVE

P2P Media Streaming Challenges

- Dynamic uptime
- Limited and dynamic bandwidth

Outline

- ✓ Introduction
- ➡ Related Works
- ForestCast
- Simulation
- Summary

Two Main Questions

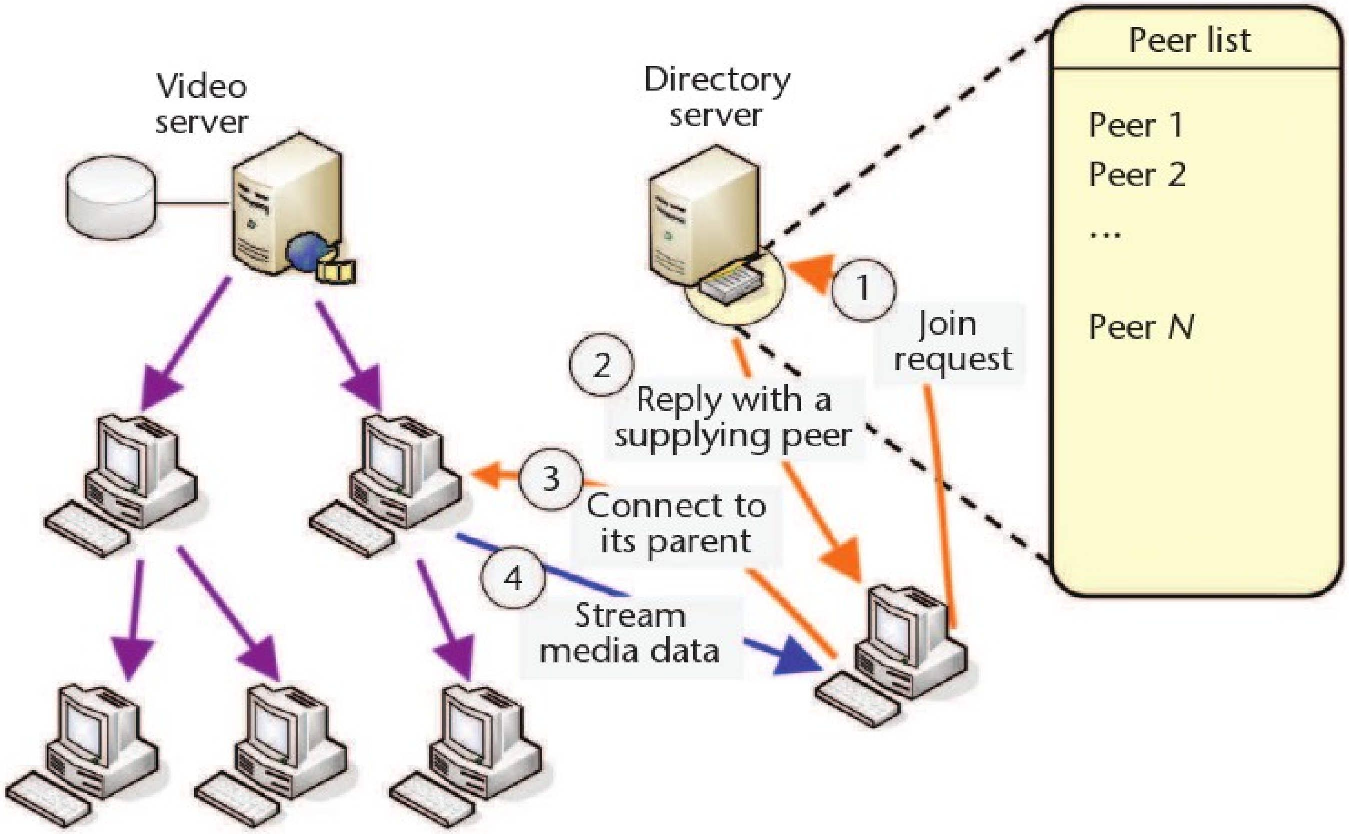
- How to find supplying peers?
- How to maintain content delivery paths?

Locating Supplying Peers

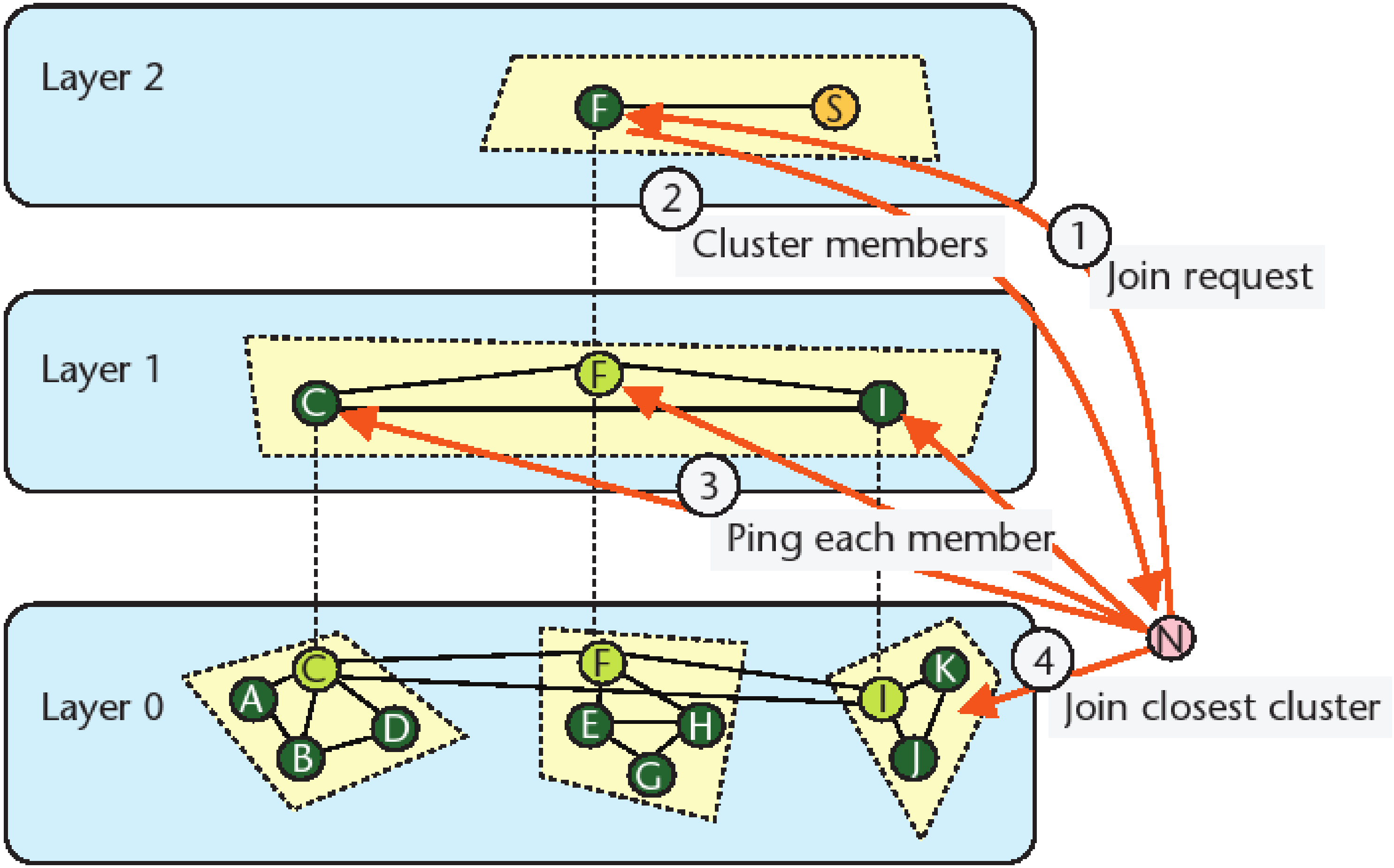
Locating Supplying Peers

- Centralized directory
- Hierarchical overlay structure
- DHT-based approach
- Controlled flooding
- Gossip-based approach

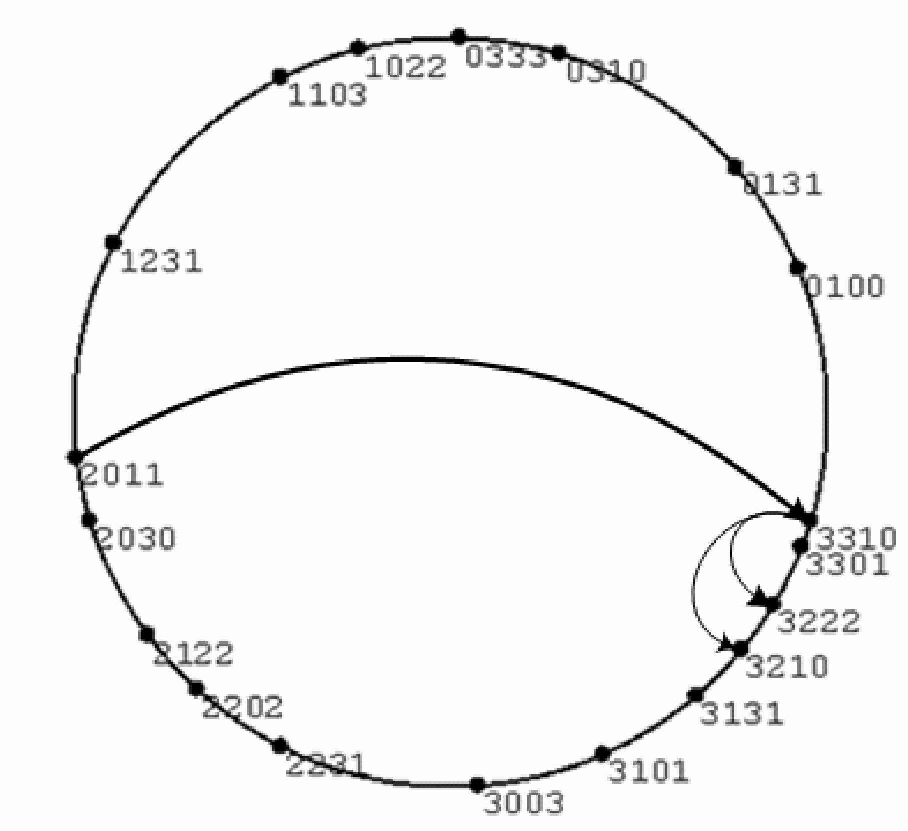
Centralized Directory



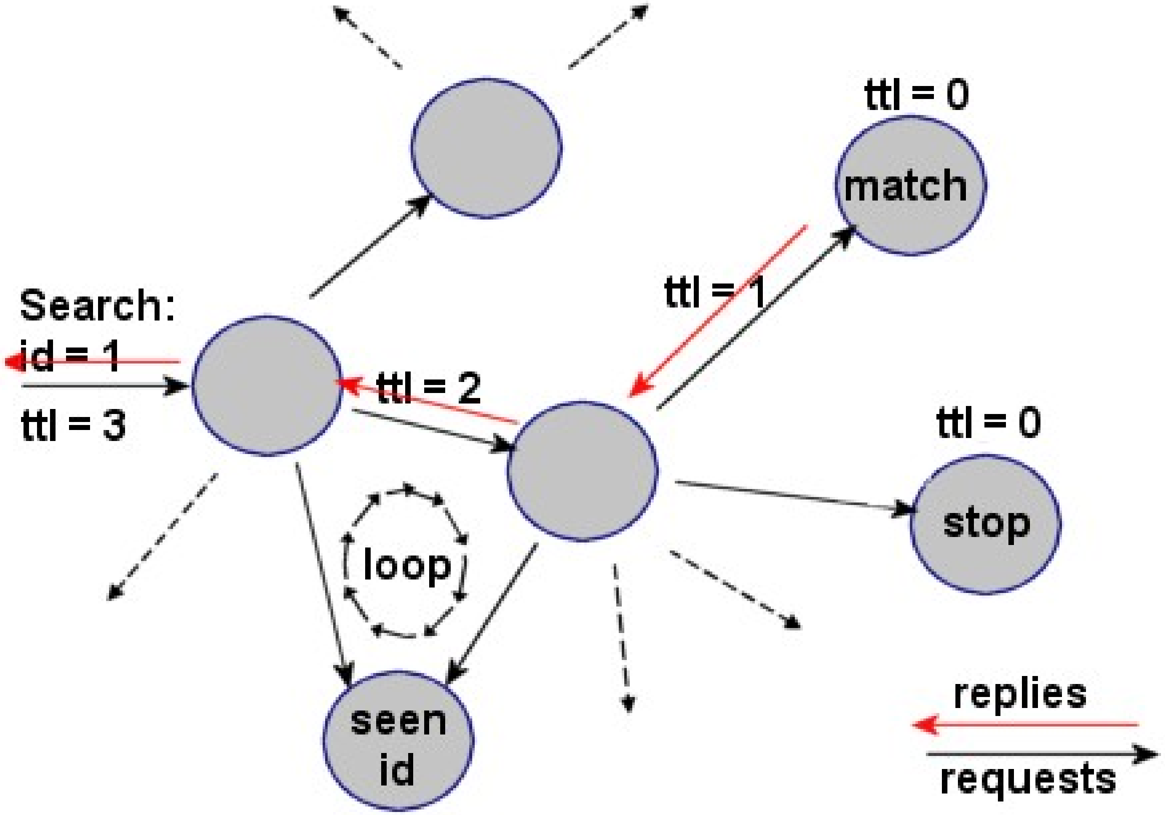
Hierarchical Overlay Structure



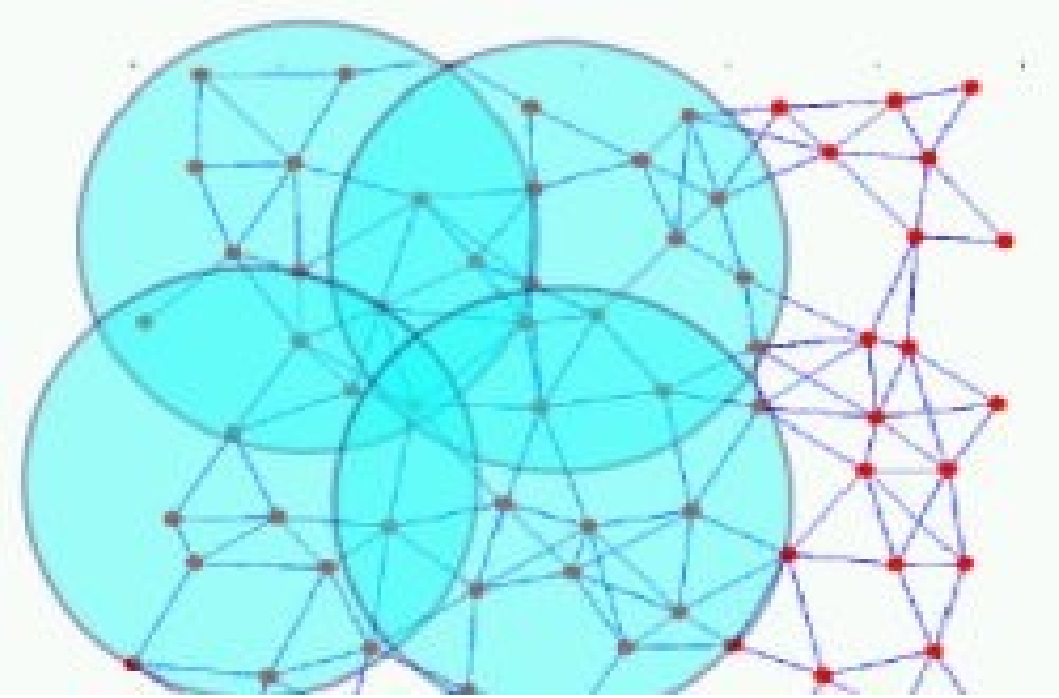
DHT-based Approach



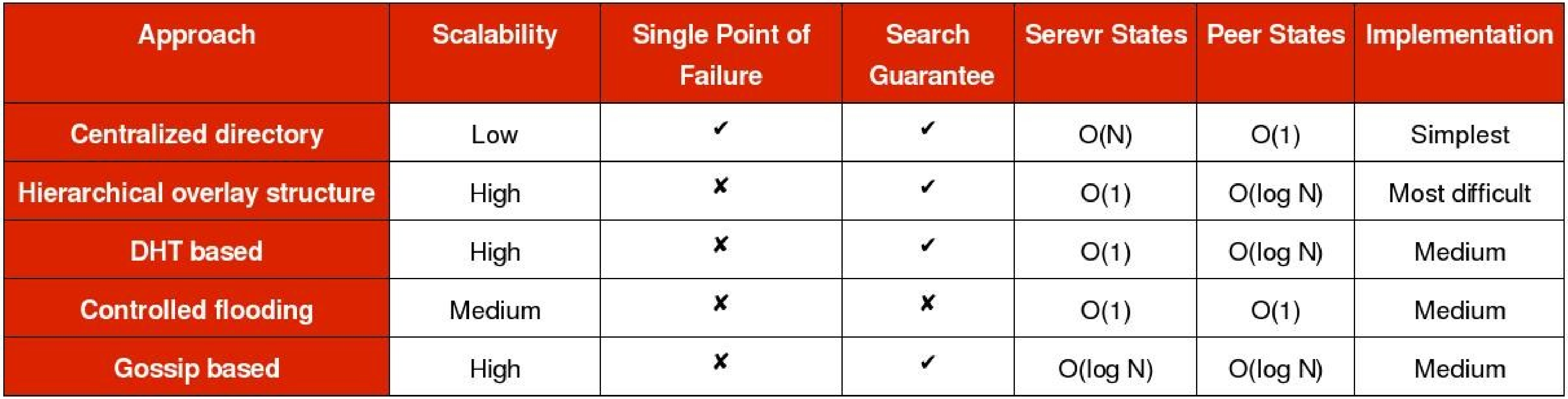
Controlled Flooding



Gossip-based Approach



Locating Supplying Peers (Comparison)



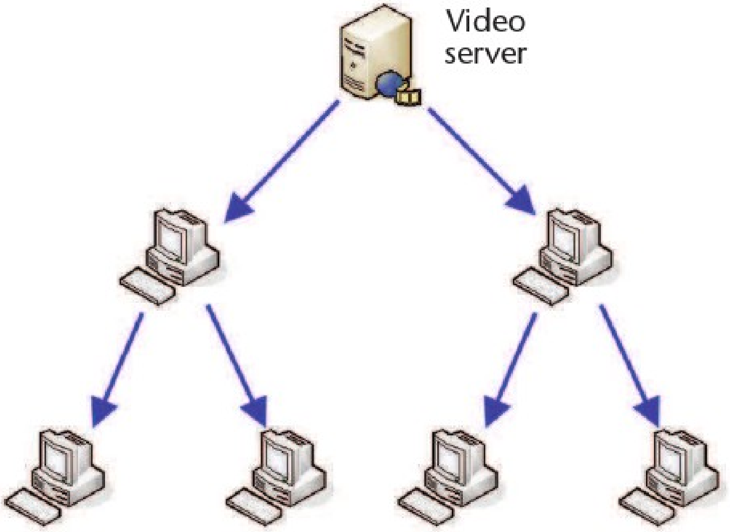
Approach	Scalability	Single Point of Failure	Search Guarantee	Serevr States	Peer States	Implementation
Centralized directory	Low	✓	✓	O(N)	O(1)	Simplest
Hierarchical overlay structure	High	✗	✓	O(1)	O(log N)	Most difficult
DHT based	High	✗	✓	O(1)	O(log N)	Medium
Controlled flooding	Medium	✗	✗	O(1)	O(1)	Medium
Gossip based	High	✗	✓	O(log N)	O(log N)	Medium

Maintaining Content Delivery Path

Maintaining Content Delivery Path

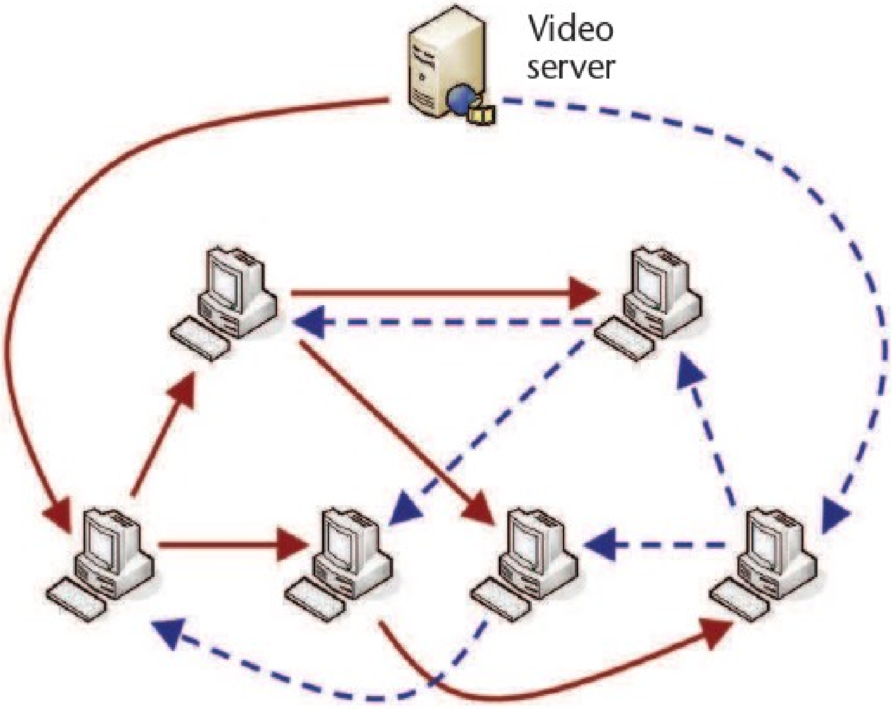
- Push based
  - Single tree
  - Multiple trees
- Pull based

Push-base

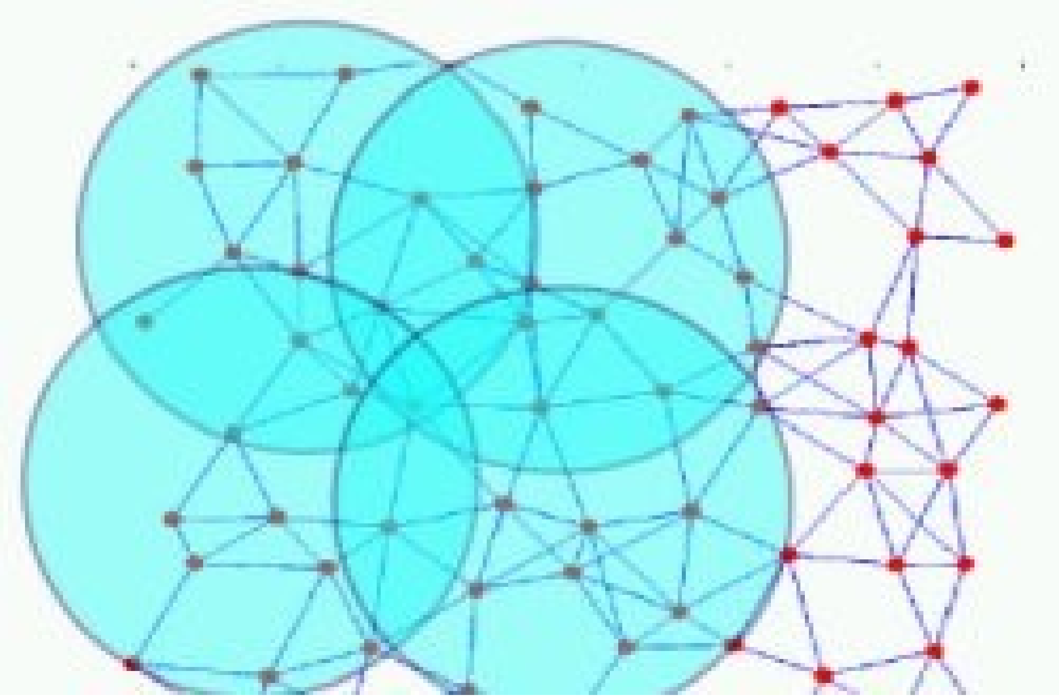


Single tree

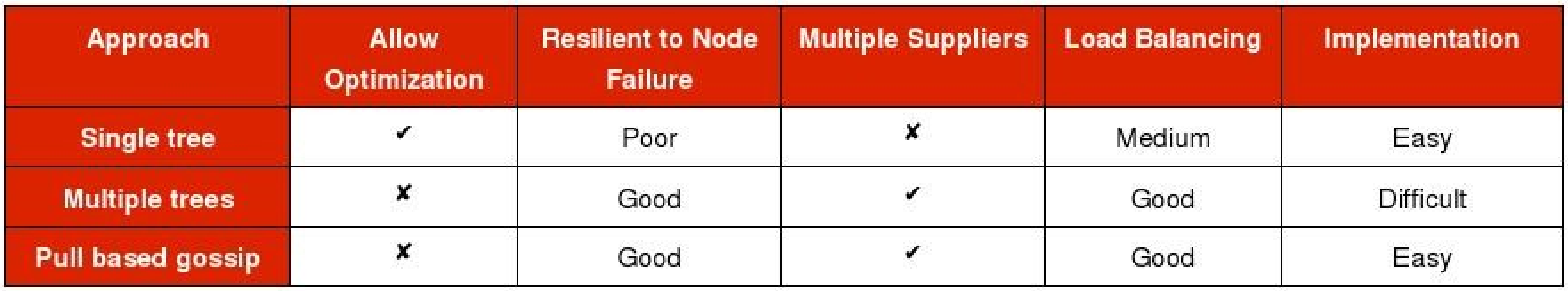
Multiple trees



Pull-based



Maintaining Content Delivery Path (Comparison)



Approach	Allow Optimization	Resilient to Node Failure	Multiple Suppliers	Load Balancing	Implementation
Single tree	✔	Poor	✘	Medium	Easy
Multiple trees	✘	Good	✔	Good	Difficult
Pull based gossip	✘	Good	✔	Good	Easy

Related Works

- SplitStream
  - DHT based
  - Push model (multiple tree)
- ZigZag
  - Hierarchical overlay structure
  - Push model (single tree)
- CoolStream
  - Gossip based
  - Pull model

- Pulsar
  - DHT based
  - Mixed (pull and push)
- Orchard
  - Gossip based
  - Push model (multiple tree)
- PULSE
  - Gossip based
  - Pull model

Outline

- ✓ Introduction
- ✓ Related Works
- ➡ ForestCast
- Simulation
- Summary

What is ForestCast?

- A solution to heuristically build multicast trees for live video streaming.

- What are we looking for?
  - Maximize the total utilization of upload bandwidth
  - Maximize continuity
  - Minimize latency
  - Minimize start up delay

The Answer of Two Questions

- Centralized directory
  - Locating supplying peers
- Push-based (Multiple trees)
  - Maintaining content delivery paths

Multicast Trees

- The stream is split into some stripes.
- One multicast tree for each stripe.
    - rooted at source.

How to Approach the Problem

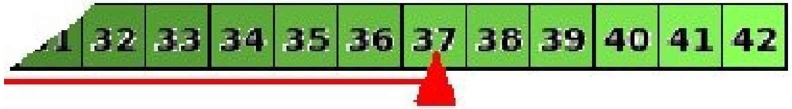
- What are the things that influence our goals?
  - Bandwidth of peers
  - Fair distribution of different stripes
  - Having distinct parents
  - Position of a peer in different trees
- How they affect the efficiency of system?
  - Needs appropriate heuristics
  - Evaluation

Peer roles

- Server
  - Central server that constructs the tree
- Source
  - The node which has the video to be streamed
- Peer
  - A node which downloads/uploads the stream

Definition of Some Terms

- Open node
  - A node which its available upload bandwidth is more than the stripe rate.
- Head of buffer
  - The largest segment number a node has in its buffer.
- Head to play latency
  - The difference between head of buffer of a node and its playback point.



Join Procedure

Join Procedure

- Server receives the join request from a peer.

- It decides from which node a joining peer should receive its live stream.

- The decision will be based on
  - The existing trees
  - The properties of the joining node
    - e.g. its available bandwidth

Join Procedure (Step 1)

- Collect a number of open nodes for each stripe.
  - How many open nodes should be selected?
  - Where to start picking the nodes?
    - Root?
    - Leaves?
    - ...
  - In what order?
    - BFS?
    - DFS?
    - ...

Join Procedure (Step 2)

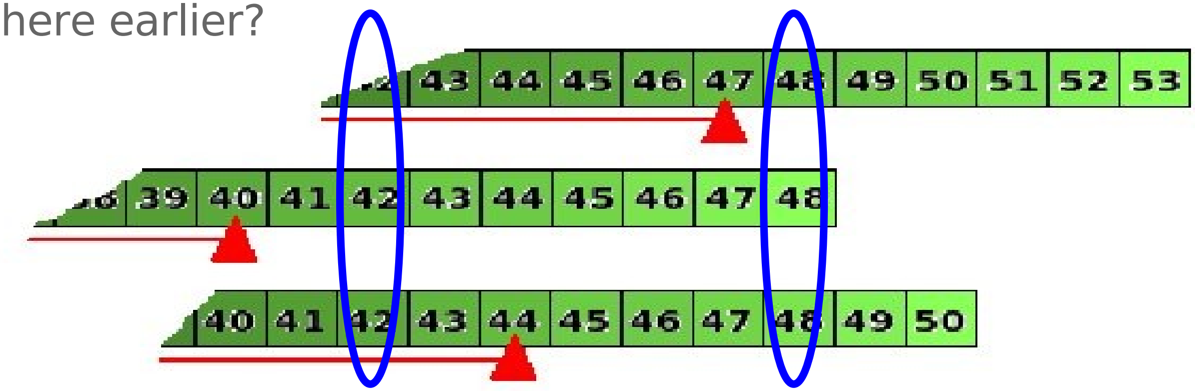
- Prioritize the collected nodes for being selected as a parent.
  - Number of existing children
  - Available upload bandwidth
  - Source-to-end latency
  - Any other parameter?

Join Procedure (Step 3)

- Select the best parents
  - Who is the best parent?
  - Distinct parents
    - To increase the resiliency to failure
  - Which stripe?
    - Rarest?
    - ...
  - Any other parameter?

Join Procedure (Step 4)

- Decide from which segment number the media should be forwarded to the joining node.
  - This segment number determines a specific time of the media that joining node will start to play.
  - Different parents have different segments at their head of buffer.
  - So from which segment?
    - Minimum of head of buffer of all parents?
    - Start somewhere earlier?



Join Procedure (Step 5)

- Decide about the playback time of the joining node.
  - There is a trade-off between playback latency and continuity.
- Asks the selected parents to forward data to the new node from the decided segment on.
- Asks the new node to start playback where its head-to-play is the decided length.

Each Peer After Joining

- Sends its profile periodically to server:
  - Latency to its parent for each stripe
  - First segment of each stripe
  - Last segment of each stripe
  - ...

Leave Procedure

Leave Procedure

- Server receives the leave request from a peer.
  - The request consists of the last segment number which the leaving node has sent to its children.

- It finds the substitute parents for the leaving peer.

- The decision will be based on
  - The existing trees
  - The properties of the orphan nodes
    - e.g. what is the last segment which they will receive from the leaving node

Leave Procedure (Step 1)

- Find substitute parents for the children of the leaving node.
  - The same as finding parent for joining node, but some more constraints
    - It should consider the last segment which the children have received.
    - The new parent should have from that segment on.

Leave Procedure (Step 2)

- In case of finding new parent
  - Server asks the leaving node to stop forwarding any more data to that child.
  - Server also sends order to the new parents to start forwarding data to their new children.

Leave Procedure (Step 3)

- In case of not finding new parent
  - The server repeats step 1 until it finds a new parent.
  - The leaving node continues sending stream to its children.

Leave Procedure (Step 4)

- After finding new parent for all children
  - The server asks the parent of leaving node to stop sending data to it.
  - Server grants the node to leave.
  - The node leaves the system upon receiving server’s message.

Failure Handling

Failure Handling

- Server receives the failure notification from the failed node's children.
  - The message consists of the last segment number which the children have in their buffer.
- It finds the substitute parents for the orphaned peers.
- The decision will be based on
  - The existing trees
  - The properties of the orphan nodes
    - e.g. what is the last segment which they will receive from the leaving node

Failure Handling (Step 1)

- Find substitute parents for the children of the failed node.
  - The same as finding parent for joining node, but some more constraints
    - It should consider the last segment which the children have received.
    - The new parent should have from that segment on.

Failure Handling (Step 2)

- In case of finding new parent,
  - Server sends order to the new parents to start forwarding data to their new children.
- Otherwise
  - Find a parent who causes the least disruption.

Outline

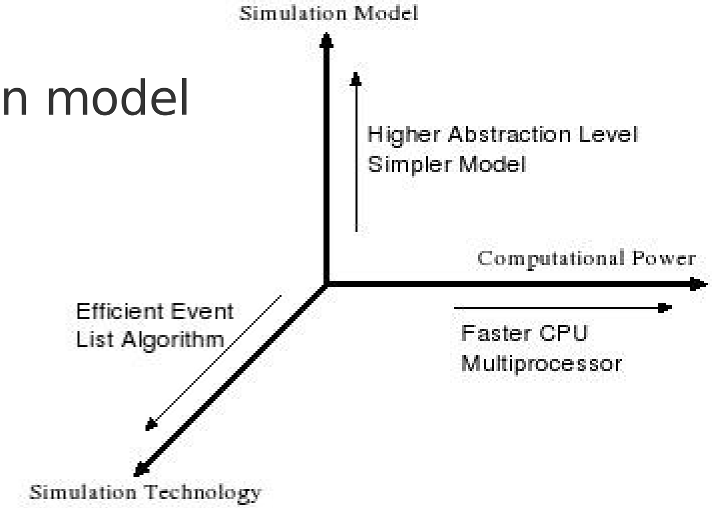
- ✓ Introduction
- ✓ Related Works
- ✓ ForestCast
- ➡ Simulation
- Summary

Simulation

- 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 an instant in time and makes a change of state in the system.
- Traditional modelling
  - Packet level modelling

Improving Scalability

- Improving computational power
  - Using faster and more powerful machines
- Improving simulation technology
  - Using better algorithms
- Changing simulation model
  - Using simpler and higher abstraction model

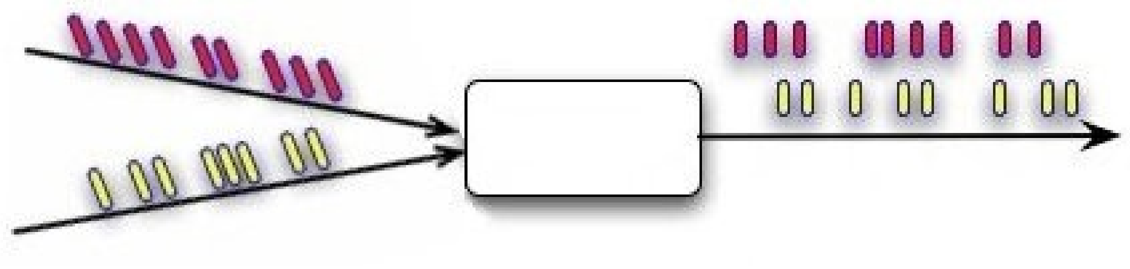


Packet-level vs. Fluid-level

- Packet level modelling
  - For each packet departures or arrivals one event will be generated.
- Fluid level modelling
  - The events are generated only when the rate of flows changes.

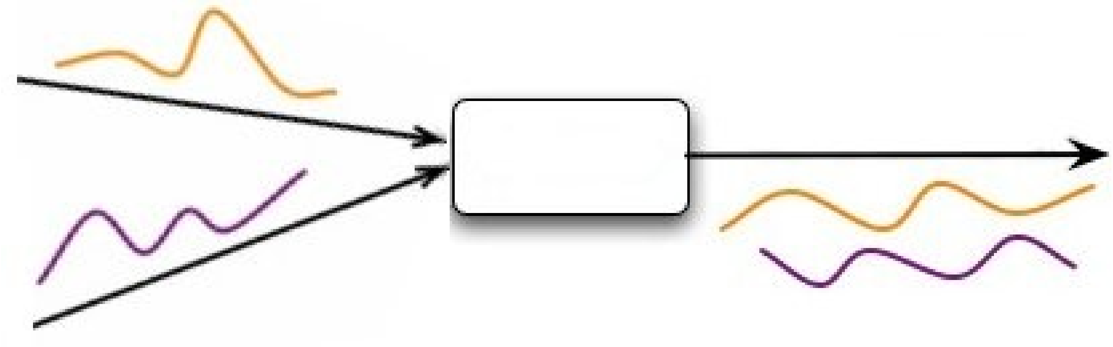
Packet-level modelling

- Advantages
  - High accuracy
    - Because of considering the detail information of individual packets
- Disadvantages
  - Low scalability
    - In case of growing the network size and links bandwidth
    - Huge number of events and cost of processing them



Fluid-level modelling

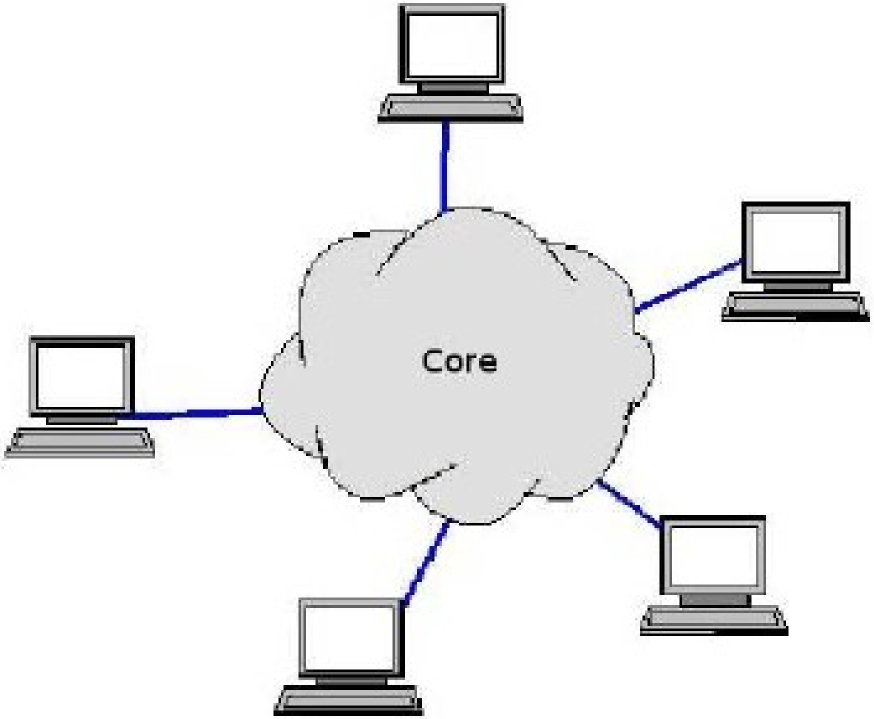
- Advantages
  - High scalability
  - High performance
    - In case of low changes in flow rates
- Disadvantages
  - Low accuracy
    - Because of ignoring the details of modelling
  - Ripple effect
    - Reduces the performance advantage of fluid-level modelling



Our modelling (SicsSim-II)

- We use fluid-based modelling.
  - To have a scalable simulator
- We partly consider the effects of underlying layer as well.
  - Network congestion

SicsSim-II Overall Structure



Internal Structure of SicsSim-II

- FEL (Future Event List)
  - A global queue which contains all the events of the system.
- Event scheduler
  - Lets the simulator to handle the control messages in a chronological order.
- The simulation loop proceeds by selecting the next event in queue, executing it and inserting new generated events in queue in simulation time order.

Control and Data Messages

- Control messages
  - The control messages are considered to have a very small size which would use 0 bandwidth.
  - To handle them we put them in future event list.

- Data messages
  - Data messages carry the real data.
  - We don’t transfer real data in the simulator.
  - We just assume there is a flow of data from nodes to nodes with a rate.

Outline

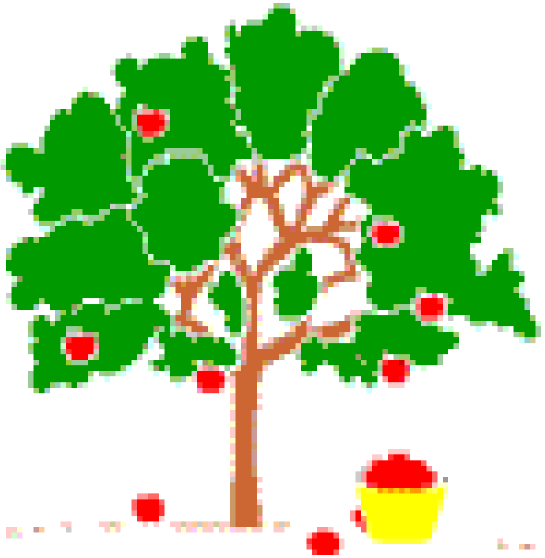
- ✓ Introduction
- ✓ Related Works
- ✓ ForestCast
- ✓ Simulation
- ➡ Summary

Future Work

- Evaluating the algorithm
- Improving the model
- Investigate the same approach for decentralized model

Summary

- P2P live streaming
- Different algorithms
  - Locating supplying peers
  - Maintaining content delivery path
- ForestCast
  - Centralized
  - Multiple trees
- SicsSim-II
  - Discrete event modelling
  - Fluid based



Questions?

&

Comments!