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
- ...
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

<table>
<thead>
<tr>
<th>Finding supplying peers</th>
<th>Data Delivery</th>
<th>Push method (Single tree)</th>
<th>Push method (Multiple trees)</th>
<th>Pull method</th>
<th>Push-Pull method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mTreeBone (2007)</td>
</tr>
<tr>
<td>Controlled flooding method</td>
<td></td>
<td></td>
<td></td>
<td>GnuStream (2003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ChainSaw (2005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PPLive (2004)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

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

<table>
<thead>
<tr>
<th>Finding supplying peers</th>
<th>Data Delivery</th>
<th>Push method (Single tree)</th>
<th>Push method (Multiple trees)</th>
<th>Pull method</th>
<th>Push-Pull method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mTreeBone (2007)</td>
</tr>
<tr>
<td>Controlled flooding method</td>
<td></td>
<td></td>
<td></td>
<td>GnuStream (2003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ChainSaw (2005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PPLive (2004)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
  - \( \text{Strength}(A_i) = \text{age}_A \cdot (\text{fanout}_{A_i} + \text{freeBw}_A) \)
- 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
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.

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

Playback latency vs. percentage of received quality for different node collection policies.

- BFS-DFS policy (low churn)
- BFS policy (low churn)
- BFS-DFS policy (high churn)
- BFS policy (high churn)
Parent Selection (Priority Function)

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

• 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

Playback latency

Fraction of nodes

Percentage of received quality

Quality

Fraction of nodes

Head-Segment policy (low churn)
Mid-Segment policy (low churn)
Head-Segment policy (high churn)
Mid-Segment policy (high churn)
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

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

Avg of playback latency

- Join only
- Low churn 20%
- High churn 50%

Number of nodes vs Playback latency (ms)
Received Quality

![Graph showing received quality over number of nodes with different churn rates.]
Disrupted Nodes

![Graph showing the average number of disrupted nodes versus the number of nodes. The graph includes lines for 'join only', 'low churn 20%', and 'high churn 50%'.]
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 :)

ForestCast for P2P Live Streaming, 6th March 2008