ALM for bandwidth-intensive content
Tree-based ALM
Tree-based multicast

Balanced tree, fanout $f$, height $h$

Number of interior nodes: $(f^h - 1)/(f-1)$

Number of leaf nodes: $f^h$

Fraction of leaf nodes increases with $f$

**Example:**
- 90% of nodes are leaves in a tree with fanout=16.
- Forwarding load on 10% of the nodes
- The lower the fanout the more error prone
P2P ALM

- Tree-based protocols
  - Load unbalance: majority of nodes are leaves
  - Internal node failures

- Flooding-based protocols
  - Redundancy by default
  - Potentially high network traffic

- The drawbacks are even more important when it comes to intensive contents
  - Load balancing
  - Network load
Tree-based ALM: unbalance

\[ \text{IN: } n \text{ kb/sec} \]

\[ \text{OUT: } 2n \text{ kb/sec} \]
Striping contents

- Erasure code: k blocks out of n blocks to reconstruct a file
- MDC coding: increasing quality
SplitStream approach

Content divided in *stripes*  
Each stripe is distributed on an independent tree

- **Load balancing**  
  - Internal nodes in one tree are leaves in others

- **Reliability**  
  - Failure of one load leads to unavailability of x stripes if parents are independent and using appropriate coding protocols
The SplitStream forest
The SplitStream forest
The SplitStream forest
SplitStream forest

\[ N \text{ kb/sec} \]

\[ \frac{N}{2} \text{ kb/sec} \]

\[ \frac{N}{2} \text{ kb/sec} \]
System model

$N$ nodes
Each node $i$ wants to receive $I_i$ $(0 < I_i < k)$ distinct stripes
Forwarding capacity of $C_i$

Set of $S$ (in $N$) $(1 < |S| < k)$ source nodes (the forwarding capacity of a source $s$ needs to be at least equal to the number of stripes it originates)

**Definition**: Given a set of nodes $N$ and a set of sources $S \subseteq N$, forest construction is feasible if it is possible to connect the nodes such that each node $i \in N$ receives $I_i$ distinct stripes and has no more than $C_i$ children.
Capacity conditions

**Condition 1**: If forest construction is feasible, the sum of the desired indegrees cannot exceed the sum of the forwarding capacities

\[ \sum_{i \in N} I_i \leq \sum_{i \in N} C_i \]

**Condition 2**: A sufficient condition for the feasibility of forest construction is for Condition 1 to hold and for all nodes whose forwarding capacity exceeds their desired indegree to receive or originate all k stripes

\[ \forall i : C_i > I_i \Rightarrow I_i + T_i = k. \]
Forest feasibility

- Very high probability of building a forest if condition 1 and 2 hold
- Probability increases with the amount of spare capacity
- Avoid free riders by strengthening Condition 1 to require that the forwarding capacity of each node be greater than or equal to its desired indegree (i.e., $\forall i \in N : C_i \geq I_i$).
SplitStream

- Construction of one tree/group per data stripe

- A set of trees is said to be interior-node disjoint if each node is an interior node in at most one tree, and a leaf node in the other trees.
Interior node disjoint trees

- Each stripe identifier starts with a different digit (independence up to $2^b$ stripes)

- Assume $2^b = k$
SplitStream

Main issue: build and maintain multiple multicast tree in a fully decentralized and reliable way so that

- Each client receives the desired number of stripes
- Independent trees
- Control upon bandwidth allocation
- Reasonable latency and network load

Leverage Scribe/Pastry
- Pastry: P2P routing infrastructure (structured, efficient, reliable)
- Scribe: decentralized and efficient tree-based protocol
SplitStream: forest management

Constraints
- Limited out-degree potentially increases the tree depth
- Load balancing to ensure within trees and between trees
- Failure independence of trees

Solution: spare capacity tree
- Overloaded nodes push descendents down (Scribe)
- Underloaded nodes join the spare capacity tree
- Overloaded nodes give up descendents
- Orphans *anycast* to the spare capacity tree to discover new parents
Limiting the outdegree

- Push-down mechanism of Scribe: guarantee to finish if each node is required to take one child at least.
- Untrue in Splitstream

Overloaded nodes push descendents down (Push-down mechanism of Scribe): choice of orphans

- Children subscribing to stripes whose stripeIds do not share a prefix with the local node’s nodeId.
- Otherwise child whose nodeId has the shortest prefix match with that stripeId
Locating parent for a stripe: first step

- Orphan contacts random former siblings that share a prefix match with the stripeId for which it seeks a parent

- Same push-down mechanism used recursively
Locating parent: 2\textsuperscript{nd} step

\textbf{Solution: spare capacity tree}
- Underloaded nodes join the spare capacity tree
- Overloaded nodes give up descendents
- Orphans \textit{anycast} to the spare capacity tree to discover new parents
**Spare capacity tree**

E

Anycast
For stripe 6

{0,3,A}
Cap: 2

D

{1,..,F}
Cap: 4

A

C

Adopting
- Loop checking
- Descendants switching

G
Experiments

- Simulations (average on 10 runs)
  - Topologies GT, Mercator, MS Corp.
  - 40000 nodes
- Pastry (b=4, leafset = 16)
- SplitStream : 16 stripes
- Configurations in-degree x out-degree
  - Impact of spare capacity 16x16, 16x18, 16x32 and 16xNB
  - Impact of capacity/needs (Gnutella)
- Failure resilience
  - Path diversity
  - Catastrophic failures (25% of faulty nodes) in a 10,000 node system
- Results
  - Forest construction
  - Multicast performance
Forest construction: load on each node

![Graph showing cumulative proportion of nodes vs. node stress for different configurations: 16 x NB, 16 x 32, 16 x 18, 16 x 16.](chart.png)
Forest construction: load on each node

<table>
<thead>
<tr>
<th>Configuration</th>
<th>16x16</th>
<th>16x18</th>
<th>16x32</th>
<th>16xNB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>2971</td>
<td>1089</td>
<td>663</td>
<td>472</td>
</tr>
<tr>
<td>Mean</td>
<td>57.2</td>
<td>52.6</td>
<td>35.3</td>
<td>16.9</td>
</tr>
<tr>
<td>Med</td>
<td>49.9</td>
<td>47.4</td>
<td>30.9</td>
<td>12</td>
</tr>
</tbody>
</table>

Load decreases as the spare capacity increases
16xNB: no pushdown nor orphans
• 16x16: each node contacts the spare capacity tree for 8 stripes on average
• Nodes with id close to the spare capacity tree get the highest load
Forest construction: network load

Measured as the number of msg on physical links

<table>
<thead>
<tr>
<th>Configuration</th>
<th>16x16</th>
<th>16x18</th>
<th>16x32</th>
<th>16xNB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>5893</td>
<td>4285</td>
<td>2876</td>
<td>1804</td>
</tr>
<tr>
<td>Mean</td>
<td>74.1</td>
<td>65.2</td>
<td>43.6</td>
<td>21.2</td>
</tr>
<tr>
<td>Med</td>
<td>52.6</td>
<td>48.8</td>
<td>30.8</td>
<td>17</td>
</tr>
</tbody>
</table>

Load decreases as the spare capacity increases
Maximum approx. 7 fois < centralized system
Multicast: link stress

One message/stripes, no failure

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Centralized (0.43)</th>
<th>Scribble (0.47)</th>
<th>IP (0.43)</th>
<th>16x16 (0.98)</th>
<th>16x18</th>
<th>16x32</th>
<th>16xNB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>639984</td>
<td>3990</td>
<td>16</td>
<td>1411</td>
<td>1124</td>
<td>886</td>
<td>1616</td>
</tr>
<tr>
<td>Mean</td>
<td>128.9</td>
<td>39.6</td>
<td>16</td>
<td>20</td>
<td>19</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>Med</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

- 16xNB: absence of forwarding bounds causes contention on a small set of links
- Splitstream uses a larger fraction of links but load them less
Delay penalty during multicast
Path diversity

• Number of lost stripes (at most) on each node when the most significant ancestor is faulty (worst case scenario)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>16x16</th>
<th>16x32</th>
<th>16xNB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>6.8</td>
<td>6.6</td>
<td>1</td>
</tr>
<tr>
<td>Mean</td>
<td>2.1</td>
<td>1.7</td>
<td>1</td>
</tr>
<tr>
<td>Med</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Catastrophic failure (25% of 10,000 nodes are faulty): number of received stripes

- 14 stripes after 30 s
- Total repair after less than 3mn
Catastrophic failure (25% of 10,000 nodes are faulty): number of messages
Summary

- SplitStream: robust and efficient protocol for large-scale content streaming
  - Forest of independent trees / unique tree
  - Spare capacity tree for maintenance
  - Decentralized and scalable management relying on Scribe and Pastry
  - Robust in dynamic environments
References