An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer Overlays

authors: M. Castro et. al.
Motivation

Overview of p2p multicast classification

• Lack of IP multicast deployment
• Difference in used overlay
  • Tree building
  • Flooding
• Difference in routing methods
  • Generalized hypercube routing (Pastry)
  • Cartesian hyper space (CAN)
• Paper structure..
P2P overlay networks

Different routing methods (CAN)

• Topology organization
  • d – dimensional cube
  • Each node owns “its space”

• Node’s addition
  • Selected region is split into 2 parts

• Tunable parameters
  • Node dimension, multiple node/zone
  • Net. Aware routing, uniform partitioning etc.
P2P overlay networks

Different routing methods (CAN)

2-D CAN topology & node addition

(before) (after)

1's coordinate neighbor set = \{2,3,4,5\}
7's coordinate neighbor set = \{\}

1's coordinate neighbor set = \{2,3,4,7\}
7's coordinate neighbor set = \{1,2,4,5\}
P2P overlay networks

Different routing methods (Pastry)

• Pastry namespace (128-bits)
  • $nodelag \& (message, \text{dest\_key})$
  • Next node $\rightarrow$ closest to the dest\_key

• Routing principle
  • $128/b$ levels and $2^b$ entries each
  • Next node – sharing max. bits with destination
  • At least $b$ bits “closer to destination”
### P2P overlay networks

#### Different routing methods (Pastry)

**Routing table**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>x</td>
<td>f</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

**Routing (65a1fc -> d467c4)**

- Route (d46a1c)
- d13da3
- d4213f
- d462ba
- d467c4
- d471f1
P2P overlay networks

Overlay based application level multicast

• Flooding
  • Main concept: broadcasting
  • Groups: smaller subsets
  • CAN flooding: "naive" & CAN Multicast
  • Pastry flooding: level tagging and forwarding

• Tree - based (Scribe approach)
  • Firstly target group roots
  • Decentralized approach is scalable
P2P overlay networks

Different overlays approaches

flooding

tree-build architectures
Evaluation

Experimental setup

• Experimental environment
  • measured -> number of packets
  • 5 network topologies
• 1st experimental set: one multicast group
• 2nd experimental set: multiple multicast groups
• Experimental phases
  • groups subscription
  • message is multicast to each group
Evaluation

Experimental setup

- Experimental criteria
- Relative (ratio app. level/IP multicast values)
  - Relative delay penalty
  - RMD (maximum ratio), RAD (average ratio)
- Link stress
  - Number of packets over the link
- Node stress
  - Routing table size (#nodes) & messages received
- Message duplication
CAN Results

• Parameters
  • Number of dimensions (d); nodes per zone (z)
  • policy: (distance, ratio, NDR), uniform part. (on/off)

![CAN flooding](chart1.png)

![CAN tree-build](chart2.png)
Pastry Results

- Parameters
  - \( b \) - number of "matched" dest. bits \((b=\{1,2,3,4\})\)
  - TART & TOP

Pastry flooding

Pastry tree-build
More results

• Link Stress

<table>
<thead>
<tr>
<th>Configuration</th>
<th>d=10</th>
<th>d=9</th>
<th>d=12</th>
<th>d=10</th>
<th>d=8</th>
</tr>
</thead>
<tbody>
<tr>
<td>State size</td>
<td>18</td>
<td>29</td>
<td>38</td>
<td>59</td>
<td>111</td>
</tr>
</tbody>
</table>

<p>| | Joining phase | Flooding phase |</p>
<table>
<thead>
<tr>
<th></th>
<th>Max</th>
<th>Average</th>
<th>Max</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>State size</td>
<td>18</td>
<td>29</td>
<td>38</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>154</td>
<td>149341</td>
<td>197977</td>
<td>309212</td>
</tr>
<tr>
<td></td>
<td>183</td>
<td>219</td>
<td>281</td>
<td>431</td>
</tr>
<tr>
<td></td>
<td>91615</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.49</td>
<td>2.93</td>
<td>2.73</td>
<td>2.69</td>
</tr>
</tbody>
</table>

flooding in CAN

CAN tree-based

<table>
<thead>
<tr>
<th></th>
<th>with TART</th>
<th>without TART</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RAND</td>
<td>TOP</td>
</tr>
<tr>
<td>Max</td>
<td>65.4</td>
<td>4.3</td>
</tr>
<tr>
<td>Average</td>
<td>4.3</td>
<td>1.4</td>
</tr>
</tbody>
</table>

Pastry flooding for b=4
Comparative Results
for multiple multicast groups (CDF functions)

RMD with localized members

RMD with distributed members

RMD with both localized and distributed members
Related Work

- Tapestry and Chord
  - Similar approaches
- Bayeux and Overcast
  - Different concepts
  - Prospective future work
- Non scalable
  - End System Multicast, RONs, ISIS
- IP Multicast
Summary

• First head2head p2p analysis (4 comb.)
  • Flooding & tree building
  • Hypercube & Cartesian metric space
• Tree-based is better than flooding
• Multicast trees-build
  • Pastry better than CAN
• Flooding overlay costs more..
• Related work (further considerations)
  • Overcast, Bayeux, IP Multicast
• Questions??

• Discussion..