Improving the scalability of an Application-Level Multicast Protocol

Ayman EL-SAYED

Vincent ROCA

[ayman.elsayed|vincent.roca]@inrialpes.fr

Planète project; INRIA Rhône-Alpes

February 25th, 2003
Introduction to application-level multicast

● motivations for application-level multicast
  ○ multicast routing is not available everywhere

● application-level multicast
  ○ shifts the multicast support from core routers to end-systems
  ○ automatic creation an overlay topology
    ○ uses unicast between two end-systems
    ○ the underlying physical topology is hidden
    ○ try to find an "optimal" overlay topology
      (e.g. a spanning tree with minimal global cost)
Introduction to ALM… (cont’)

- Example

With multicast routing

- Topology building algorithm can be:
  - Centralized (HBM, ALMI …)
  - Distributed (NARADA, Overcast, Nice, TBCP…)

With application-level multicast
Introduction to ALM... (cont’)

● requires a dynamic overlay topology update
  ○ because the networking conditions dynamically change
    ○ try to stay as close as possible to an optimal overlay topology
    ○ can be regarded as "static QoS routing"
  ○ because the group is dynamic, the topology becomes sub-optimal
    ○ after a node departure/failure, a quick and dirty local solution is found to avoid topology partitioning
    ○ when a node arrives, he joins the current topology as a leaf to create as little perturbation as possible

● we need to periodically update the whole topology!
Our HBM application-level multicast

- centralized approach:
  - everything is controlled by a Rendez-vous Point (RP)
  - the RP has a complete knowledge of group membership/communication costs
  - take into account several metrics (RTT, loss) when creating the virtual topology
  - data flows on the virtual topo. (no RP implication)

- each node periodically evaluates metrics between itself and other nodes and informs the RP (period: $T_{mu}$).

- the RP periodically refreshes the topology and informs all nodes (period: $T_{tu}$).
Our HBM proposal… (cont')

metric update msg (MU) topo. update msg (TU)

<table>
<thead>
<tr>
<th>Metric update</th>
<th>Topology update</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;metricupdate&gt;</code></td>
<td><code>&lt;topologyupdate&gt;</code></td>
</tr>
<tr>
<td><code>&lt;metric&gt;</code></td>
<td><code>&lt;metric&gt;</code></td>
</tr>
<tr>
<td><code>tecsid=2</code></td>
<td><code>tecsid=2</code></td>
</tr>
<tr>
<td><code>tecsid=3</code></td>
<td><code>tecsid=3</code></td>
</tr>
<tr>
<td><code>tecsid=4</code></td>
<td><code>tecsid=4</code></td>
</tr>
<tr>
<td><code>tecsid=5</code></td>
<td><code>tecsid=5</code></td>
</tr>
<tr>
<td><code>tecsid=6</code></td>
<td><code>tecsid=6</code></td>
</tr>
<tr>
<td><code>&lt;metric&gt;1.60, 0.010&lt;/metric&gt;</code></td>
<td><code>&lt;metric&gt;2.50, 0.001&lt;/metric&gt;</code></td>
</tr>
<tr>
<td><code>&lt;metric&gt;10.10, 0.12&lt;/metric&gt;</code></td>
<td><code>&lt;metric&gt;3.10, 0.012&lt;/metric&gt;</code></td>
</tr>
<tr>
<td><code>&lt;metric&gt;10.10, 0.12&lt;/metric&gt;</code></td>
<td><code>&lt;metric&gt;10.10, 0.12&lt;/metric&gt;</code></td>
</tr>
<tr>
<td><code>&lt;metric&gt;30.10, 0.195&lt;/metric&gt;</code></td>
<td><code>&lt;metric&gt;30.10, 0.195&lt;/metric&gt;</code></td>
</tr>
<tr>
<td><code>&lt;metric&gt;2.50, 0.001&lt;/metric&gt;</code></td>
<td><code>&lt;metric&gt;2.50, 0.001&lt;/metric&gt;</code></td>
</tr>
<tr>
<td><code>&lt;metric&gt;3.10, 0.012&lt;/metric&gt;</code></td>
<td><code>&lt;metric&gt;3.10, 0.012&lt;/metric&gt;</code></td>
</tr>
</tbody>
</table>

both follow an XML format (simplicity, flexibility)
Our HBM proposal… (cont’)

- centralized scalability problems limit the topology update frequency

- goal:
  how can we improve this scalability?

- solution: reduce the control traffic overhead
Improving the scalability

- Total control traffic rate: \( R_{\text{cntl}}(N) = R_{\text{tu}}(N) + R_{\text{mu}}(N) \)
- MU control rate: \( R_{\text{mu}}(N) \)
- TU control rate: \( R_{\text{tu}}(N) \)
- Number of Members: \( N \)
Improving the scalability… (cont’)

- message sizes
  - size of MU msg
    - size of MU msg = $S_{mu}(N)$
  - number of MU records per msg
    - number of MU records per msg = $n_{rmu}(N)$
  - mean size of a MU record
    - mean size of a MU record = $S_{rmu}$
  - size of a TU msg
    - size of a TU msg = $S_{all\_tu}(N)$

\[
S_{mu}(N) = S_{mu\_header} + n_{rmu}(N) \times S_{rmu}
\]

\[
S_{tu\_all} = N \times S_{tu\_h} + 2 \times h \times S_{rtu}
\]

\[
R_{ctrl}(N) = R_{mu}(N) + R_{tu}(N)
\]
Improving the scalability... (cont’)

● principle
  ○ keep the total control traffic rate inferior to a given percentage of the total session traffic
  ○ (same strategy as RTCP)

\[ R_{\text{ctrl}}(N) \leq \alpha \times (R_{\text{ctrl}}(N) + R_{\text{data}}) \]

○ in our experiments: \( \alpha = 5\% \), \( R_{\text{data}} = 128 \text{ kbps} \)

● play with the various protocol parameters
  ○ metric update refresh period: \( T_{\text{mu}} \)
  ○ number of records per MU message

\[ \} \quad \text{EH} \]

○ topology refresh period: \( T_{\text{tu}} \)

\[ \} \quad \text{RP} \]
Improving the scalability… (cont’)

we defined 4 strategies

- strategy 1 is the reference
  - does not include any optimization
- strategies 2, 3 and 4 differ on the way the $T_{mu}$, $N_{rmu}$, $T_{tu}$ parameters are managed

- they all achieve their goals!
Strategy 1: reference

- no control traffic bandwidth limitation!
Strategy 2

- achieves control traffic bandwidth limitation!
Strategy 3

- achieves control traffic bandwidth limitation!
Strategy 4

- achieves control traffic bandwidth limitation!
Improving the scalability... (cont')

- since all strategies achieve their goal, the question is now:

  “what is the best way to achieve this goal?”
$T_{tu}(N)$: outgoing control rate

- no difference between strategies 2, 3 and 4
$N_{rmu}(N)$: nb of records in a MU message

- $N_{rmu}(N)$ must be limited above a given threshold
\( T_{mu}(N) \): MU refresh period

- strategy 4 is bad because of high \( T_{mu}(N) \)

- now comparison is between strategies 2 & 3
Metric evaluation overhead

- Metric evaluation of strategy 3 is better than that of strategy 2

![Graph showing metric evaluation data rate vs. number of members (N)](image)
Representatativity ratio

- strategy 3 has a better representatativity than strategy 2

\[ R = \frac{n_{rmu}(N)}{N - 1} \]
Number of metrics refreshed per TU

- strategy 2 is the best (more metric records per topology update)
Conclusions

● a centralized application layer multicast
  ▪ everything is controlled by a RP
  ▪ simple and efficient

● to improve the scalability
  ▪ step 1: model of the protocol behavior
  ▪ step 2: define a target (max overhead)
  ▪ step 3: identify the best way to act on the model parameters (strategy 3)

● complementary solutions
  ▪ textual control message compression
  ▪ the number of effective users is larger since a EH can serve many local clients using local native multicast
The strategies

***** Limit the n_{rmu}(N) *****

● Strategy 2:
  ○ n_{rmu}(N) decreases, T_{mu}(N) remains constant
  ○ after a threshold, n_{rmu}(N) remains constant and T_{mu}(N) increases

● Strategy 3:
  ○ n_{rmu}(N) remains constant, T_{mu}(N) increases

● Strategy 4:
  ○ n_{rmu}(N) and T_{mu}(N) both increase