HyperCast -
Toward Super-Scalable Multicast

Jörg Liebeherr
University of Virginia
jorg@cs.virginia.edu

Outline

- Motivation and Background
- Hypercube Control Topology
- HyperCast Protocol
- Evaluation of Protocol
- Conclusions
**Multicast Applications**

<table>
<thead>
<tr>
<th>Games</th>
<th>Collaboration Tools</th>
<th>&quot;Pointcast&quot; Software Distribution</th>
<th>Distributed Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Senders</td>
<td>10</td>
<td>1,000,000</td>
<td>1,000,000</td>
</tr>
</tbody>
</table>

**What is the problem?**

- **ACK Implosion**: Sender is overwhelmed with state
Multicast Framework

Introducing the Hypercube

- An n-dimensional hypercube has $N=2^n$ nodes, where
  - Each node is labeled $k_n k_{n-1} \ldots k_1$ ($k_i = 0, 1$)
  - Two nodes are connected by an edge only if their labels differ in one position.
Hypercube as Control Topology

- **Idea:**
  - Arrange the members of a multicast group as nodes of a hypercube
  - Embed trees into the hypercube
  - Use trees to disseminate control information

- **Goals:**
  - Trees should have small height
  - Hypercube should be as compact as possible
We Use Gray Ordering for Nodes

- A Gray code, denoted by $G(.)$, is defined by
  - $G(i)$ and $G(i+1)$ differ in exactly one bit.
  - $G(2^d - 1)$ and $G(0)$ differ in only one bit.

<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>
</tr>
</thead>
<tbody>
<tr>
<td>“normal”</td>
<td>000</td>
<td>001</td>
<td>010</td>
<td>011</td>
<td>100</td>
<td>101</td>
<td>110</td>
<td>111</td>
</tr>
<tr>
<td>Gray ordering</td>
<td>000</td>
<td>001</td>
<td>011</td>
<td>010</td>
<td>110</td>
<td>111</td>
<td>101</td>
<td>100</td>
</tr>
</tbody>
</table>

Nodes are added in a Gray order

![Diagram showing nodes added in a Gray order]
Tree Embedding Algorithm

Input: $G(i) := I = I_n \ldots I_2 I_1$, $G(r) := R = R_n \ldots R_2 R_1$
Output: Parent of node $I$ in the embedded tree rooted at $R$.

Procedure Parent $(I, R)$
If $(G^{-1}(I) < G^{-1}(R))$
    Parent := $I_{n-1} \ldots I_{k+1} (1 - I_k) I_{k-1} \ldots I_2 I_1$
    with $k = \min_i (I_i \neq R_i)$.
Else
    Parent := $I_{n-1} \ldots I_{k+1} (1 - I_k) I_{k-1} \ldots I_2 I_1$
    with $k = \max_i (I_i \neq R_i)$.
Endif

Tree Embedding

• Node 000 is root:
Another Tree Embedding

- Node 111 is root:

Properties of Algorithm

- All trees generated by the algorithm satisfy:
  - **Completely Contained Trees**
    A spanning tree can be created even in an incomplete hypercube
  - **Local Computation of Trees**
    Given the label of the root, every node can compute its parent simply using its own label
Compare Hypercube with Tree Approach

- $T_l$ is a control tree with root $l$
- $w_k(T_l):$ The number of children at a node $k$ in tree $T_l$
- $v_k(T_l):$ The number of descendants in the sub-tree below a node $k$ in tree $T_l$
- $p_k(T_l):$ The path length from a node $k$ in tree $T_l$ to the root of the tree

\[
\begin{align*}
    w_k &:= \frac{1}{N} \sum_{i=1}^{N} w_k(T_l) \\
    v_k &:= \frac{1}{N} \sum_{i=1}^{N} v_k(T_l) \\
    w_{\text{avg}} &:= \frac{1}{N} \sum_{k=1}^{N} w_k \\
    v_{\text{avg}} &:= \frac{1}{N} \sum_{k=1}^{N} v_k \\
    w_{\text{max}} &:= \max_k w_k \\
    v_{\text{max}} &:= \max_k v_k \\
    p_{\text{avg}} &:= \frac{1}{N} \sum_{i=1}^{N} p_k \\
    p_{\text{max}} &:= \max_k p_k
\end{align*}
\]

Average number of descendants in a tree
HyperCast Protocol

• **Goal**: The goal of the protocol is to organize members of an IP multicast group in a logical hypercube.

• **Design criteria** for scalability:
  – Soft-State (state is not permanent)
  – Decentralized (every node is aware only of its neighbors in the cube)
  – Must handle dynamic group membership efficiently

HyperCast Protocol

• The HyperCast protocol maintains a stable hypercube:
  
  • **Consistent**: No two nodes have the same logical address
  
  • **Compact**: The dimension of the hypercube is as small as possible
  
  • **Connected**: Each node knows the physical address of each of its neighbors in the hypercube
HyperCast Protocol

Each node in a hypercube:

- has a **physical address** (= IP address/Port number)
- has a **logical address** (= position in hypercube)
- knows the physical address of its neighbors
- listens to a multicast address (*Control channel*)

HyperCast Messages

- HyperCast protocol only uses four message formats:
  
  - **Beacon**: Multicast message sent to all members of the group
  - **Ping**: Short unicast message periodically sent between neighbors
  - **Leave**: Unicast message sent when a node wishes to exit the hypercube
  - **Kill**: Unicast message used to resolve address conflicts by forcing a node to exit
Hypercast Protocol

1. A new node joins

2. A node fails

The node with the highest logical address (HRoot) sends out beacons.
The beacon message is received by all nodes.
Other nodes use the beacon to determine if they are missing neighbors.
Each node sends ping messages to its neighbors periodically.

A node that wants to join the hypercube will send a beacon message announcing its presence.
The HRoot receives the beacon and responds with a ping, containing the new logical address.

The joining node takes on the logical address given to it and adds “110” as its neighbor.
The new node responds with a ping. The new node is now the new HRoot, and so it will beacon.
Upon receiving the beacon, some nodes realize that they have a new neighbor and ping in response.

The new node responds to a ping from each new neighbor.
The join operation is now complete, and the hypercube is once again in a stable state.

Hypercast Protocol

1. A new node joins

2. A node fails
A node may fail unexpectedly

Holes in the hypercube fabric are discovered via lack of response to pings
The HRoot and nodes which are missing neighbors send beacons.

Nodes which are missing neighbors can move the HRoot to a new (lower) logical address with a ping message.
The HRoot sends leave messages as it leaves the 111 logical address.

The HRoot takes on the new logical address and pings back.
The "old" 111 is now node 001, and takes its place in the cube.
The new HRoot and the nodes which are missing neighbors will beacon.

Upon receiving the beacons, the neighboring nodes ping each other to finish the repair operation.
The repair operation is now complete, and the hypercube is once again stable.
HyperCast Verification via SPIN

- State machine of HyperCast was verified with the SPIN tool (from Bell Labs)
- HyperCast was implemented in the Process Meta-Language (PROMELA) for use by Spin
  - Over 1300 lines of code

- Verification showed (for up to six nodes):
  - HyperCast always correctly forms a stable hypercube from a set of nodes who want to join
  - HyperCast always successfully repairs a hypercube when nodes fail

Implementation of HyperCast

- HyperCast was implemented in Java
  - multithreaded design
  - over 5000 lines of code

- A cluster of PCs (on Centurion testbed at UVA) was used to run a hypercube with up to 1024 nodes:
  - We ran HyperCast on up to 32 machines
  - Latest upgrade of Centurion has > 200 machines
Performance Measurements

- So far, results from three experiments are analyzed:
  - **Experiment 1**: Many nodes join simultaneously
  - **Experiment 2**: Many nodes fail simultaneously failing
  - **Experiment 3**: Constant number of nodes with varying packet loss rate

- **Performance measures**:
  - Time until hypercube reaches a stable state
  - Rate of unicast transmissions
  - Rate of multicast transmissions
Experiment 1:
Unicast Traffic for Join Operation vs. Size of Hypercube
and Number of Joining Nodes

Experiment 1:
Multicast Traffic for Join Operation vs. Size of Hypercube
and Number of Joining Nodes
**Summary**

- The hypercube approach structure has great potential to provide scalable multicasting for networked computer applications

**Future work:**
- Add protocols and applications on top of HyperCast
- Exploit fault-tolerance to node failures
- Use in non-multicast networks and mobile ad-hoc networks
- Operate in network without IP multicast support
- Support of network topologies