Multicast and Scribe
Jeff Chase
Duke University
(Thanks to Adolfo Rodriguez and Ben Zhao)

Multicast Trees
The basic idea

Applications that need multicast
- One way, single sender: "one-to-many"
  - TV - streaming apps (NCAA games)
  - Non-interactive learning
  - Database update
  - Information dissemination
- Two way, interactive, multiple sender: "many-to-many"
  - Teleconference
  - Interactive learning

Multicast Routing
- Naïve approach: flooding (controlled broadcast)
- Better: form a spanning tree with the sender at the root, spanning all the members of a multicast group.

Multicast Trees
- e.g. a teleconference
- Multiple source trees
**Multicast Forwarding is Sender-specific**

<table>
<thead>
<tr>
<th>Group Address</th>
<th>Src Address</th>
<th>Src Interface</th>
<th>Dst Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>S2</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

**Distance-vector Multicast**

**RPB: Reverse-Path Broadcast**
- Uses existing unicast shortest path routing table.
- If packet arrived through interface that is the shortest path to the packet’s SA, then forward packet to all interfaces.
- Else drop packet.

**Distance-vector Multicast**

**RPB: Reverse-Path Broadcast**

- **Sender/Speaker**
  - Multicast Group (S1, G)

- **LAN**

- **Designated Parent Router:** One parent router picked per LAN (one “closest” to source).

**Distance-vector Multicast**

**RPM: Reverse-Path Multicast**
- **RPM = RPB + Prune**
- **RPB used when a source starts to send to a new group address.**
- **Routers that are not interested in a group send prune messages up the tree towards source.**
- **Prunes sent implicitly by not indicating interest in a group.**
- **DVMRP works this way.**

**IP Multicast: Trees and Addressing**
- **All members of the group share the same “Class D” Group Address.**
- **An end-station “joins” a multicast group by (periodically) telling its nearest router that it wishes to join (uses IGMP - Internet Group Management Protocol).**
- **An end station may join multiple groups.**
- **Routers maintain “soft state” indicating which end-stations have subscribed to which groups.**
- **IGMP itself does not deal with the multicast routing problem.**
- **DVMRP, PIM**
Link State Multicast
- MOSPF (Multicast OSPF)
- Use IGMP to determine LAN members
- Flood topology/group changes
- Each router gets complete topology, group membership
  - Compute shortest path spanning tree
  - Recompute tree every time topology changes
  - Add/delete links if membership changes
- Scalability concerns similar to OSPF
  - Overhead of flooding

Protocol Independent Multicast
- PIM-DM (Dense Mode) uses RPM.
- PIM-SM (Sparse Mode) designed to be more efficient than DVMRP.
  - Routers explicitly join multicast tree by sending unicast Join and Prune messages.
  - Routers join a multicast tree via a RP (rendezvous point) for each group.
  - Several RPs per domain (picked in a complex way).
  - Provides either:
    - Shared tree for all senders (default).
    - Source-specific tree.

Multicast: Issues
- How to make multicast reliable?
- What service model, e.g., delivery ordering?
  - Much work in group communication (CATOCS)
- How to implement flow control?
- How to support different rates for different end users?
- How to secure a multicast conversation?
- What does end-to-end mean here?
- Will IP multicast become widespread?

The End-to-end Challenge
- Keep the network simple & robust
- Rely upon end-to-end adaptation
- Layer reliability on top of IP multicast...or not
- Unlike TCP, RM has to cope with
  - Scale
  - Heterogeneity among receivers
- Been trying for a decade
  - This is a HARD problem

Application-Layer Multicast
- IP multicast is not enough.
  - Inter-domain multicast routing not widely deployed.
  - Topology-aware, but not reliable.
  - No success in deploying Reliable Internet Multicast
- Interest in overlay multicast began with Hui Zhang@CMU, and a few others, in late 1990s.
  - Conference telecasts, etc.
- Now dozens of papers
- Several deployed systems and broadcast/multicast services offered by CDNs.
  - Single-source, multi-source, meshes, speed differences, reliability, resource management, etc.
- How to structure the overlay?

Scribe
- Scribe is a scalable application-level multicast infrastructure built on top of Pastry
- Provides topic-based publish-subscribe service.
  - Provides best-effort delivery of multicast messages
  - Fully decentralized
  - Supports large number of groups
  - Supports groups with a wide range of size
  - High rate of membership turnover (churn?)
API’s for Scribe

**Pastry’s API**
- Pastry exports
  - Route(msg, key)
  - Send(msg, IPAddr)
- Application’s build on Pastry must exports
  - Deliver(msg, key)
  - Forward(msg, key, nextId)

**Scribe’s API**
- Create(credentials, topicId)
- Subscribe(credentials, topicId, evtHandler)
- Unsubscribe(credentials, topicId)
- Publish(credentials, topicId, event)

---

Scribe API

- create (credentials, group-id)
- join (credentials, group-id, message-handler)
- leave (credentials, group-id)
- multicast (credentials, group-id, message)

---

The Pastry API

- Operations exported by Pastry
  - nodeId = pastryInit(Credentials, Application)
  - route(msg, key)
- Operations exported by the application working above Pastry
  - deliver(msg, key)
  - forward(msg, key, nextId)
  - newLeaves(leafSet)

---

Scribe on Pastry

- Use Pastry to manage topic/group creation, subscription, and to build a per-topic multicast tree used to disseminate the events published in the topic.

---

Pastry

- Routes based on ‘digits’
- Similar to Chord, CAN, and Tapestry
- Each hop takes you one digit closer to your destination
- Improves on locality by finding the ‘closest’ node to you with the same prefix
- Number of nodes from which decreases exponentially as you get closer to the destination

---

Pastry: Properties

- NodeId randomly assigned from (0, 2^{128})
- \(b, |L|\) are configuration parameters

Under normal conditions:
1. A pastry node can route to the numerically closest node to a given key in less than \(\log_b N\) steps
2. Despite concurrent node failures, delivery is guaranteed unless more than \(|L|/2\) nodes with adjacent NodeIds fail simultaneously
3. Each node join triggers \(O(\log_b N)\) messages
Pastry Node State

Set of nodes with \(|L|/2\) smaller and \(|L|/2\) larger numerically closest NodeIds

| Neighboring set | Prefix-based routing entries | \(|M|\) "physically" closest nodes |
|-----------------|-----------------------------|-------------------------------|
| NodeId          |                           |                               |
| 10000000        | 5 10 20 40 80 160 320     |                               |
| 10010000        | 5 10 20 40 80 160 320     |                               |

Pastry: Routing Table

- NodeIds are in base \(2^b\)
- Several rows - one for each prefix of local NodeId \((\log_2 N) \text{ populated on average}\)
- \(2^b - 1\) columns - one for each possible digit in the NodeId representation

\[ b \text{ defines the tradeoff: } (\log_2 N) \times (2^b - 1) \text{ entries Vs. } \log_2 N \text{ routing hops} \]

Pastry Proximity

- Application provides the "distance" function
- Invariant: "All routing table entries refer to a node that is near the present node, according to the proximity metric, among all live nodes with an appropriate prefix"
- Invariant maintained on self-organization

Messaging Distance

\[ b=4; \ |L|=16; \ |M|=32; \ 200,000 \text{ lookups; Random end points} \]

Quality of Routing Tables

\[ b=4; \ |L|=16; \ |M|=32; \ 5000 \text{ New Nodes} \]

Scribe Node

A Scribe node
- May create a group
- May join a group
- May be the root of a multicast tree
- May act as a multicast source

B. Zhao
Scribe messages
- Scribe messages
  - CREATE
    - create a group
  - JOIN
    - join a group
  - LEAVE
    - leave a group
  - MULTICAST
    - publish a message to the group

Scribe Group
- A Scribe group
  - Has a unique group-id
  - Has a multicast tree associated with it for dissemination of messages
  - Has a rendezvous point which is the root of the multicast tree
  - May have multiple sources of multicast messages

Scribe Multicast Tree
- Scribe creates a per-group multicast tree rooted at the rendezvous point for message dissemination
- Nodes in a multicast tree can be
  - Forwarders
    - Non-members that forward messages
    - Maintain a children table for a group which contains IP address and corresponding node-id of children
  - Members
    - They act as forwarders and are also members of the group

Create Group
- Create Group
  - Scribe node sends a CREATE message with the group-id as the key
  - Pastry delivers the message to the node with node-id numerically closest to group-id, using deliver method
  - This node becomes the rendezvous point
  - deliver method checks and stores credentials and also updates the list of groups

GroupID
- Is the hash of the group’s textual name concatenated with its creator’s name
- Making creator the Rendez-Vous point
  - Pastry nodeID be the hash of the textual name of the node and a groupID can be the concatenation of the nodeID of the creator and the hash of the textual name of the group
- They claim this improves performance with good choice of creator

Join Group
- Join Group
  - Scribe node sends a JOIN message with the group-id as the key
  - Pastry routes this message to the rendezvous point using forward method
  - If an intermediate node is already a forwarder
    - adds the node as a child
  - If an intermediate node is not a forwarder
    - creates a child table for the group, and adds the node
    - sends a JOIN towards the rendezvous point
  - terminates the JOIN message from the child

B. Zhao
Join group

Leave Group

- Leave Group
  - Scribe node records locally that it left the group
  - If the node has no children in its table, it sends a LEAVE message to its parent
    - The message travels recursively up the multicast tree
    - The message stops at a node which has children after removing the departing node

Multicast Message

- Multicast a message to the group
  - Scribe node sends MULTICAST message to the rendezvous point
  - A node caches the IP address of the rendezvous point so that it does not need Pastry for subsequent messages
  - Multiple multicast trees for each group
  - Access control for a message is performed at the rendezvous point

Multicast Tree Repair I

- Broken link detection and repair
  - Non-leaf nodes send heartbeat message to children
  - Multicast messages serve as implicit heartbeat
  - If child does not receive heartbeat message
    - Assumes that the parent has failed
    - Finds a new route by sending a JOIN message to the group-id, thus finding a new parent and repairing the multicast tree
Multicast Tree Repair

Non-leaf nodes in the tree send HeartBeat (HB) msgs to its children.

If a node fails to receive HB msgs, it routes a (SUBSCRIBE, topicId) msg and attach to a new parent.

Avoid root failure by replicating the topicId across k closest nodes to the root node in the nodeid space.

Children table entries are discarded unless refresh msgs received from children periodically.

Scribe provides best-effort service, events may be out of order. Reliable services can be built on top of Scribe.

Reliability

- Non-leaf nodes in the tree send HeartBeat (HB) msgs to its children.
- If a node fails to receive HB msgs, it routes a (SUBSCRIBE, topicId) msg and attach to a new parent.
- Avoid root failure by replicating the topicId across k closest nodes to the root node in the nodeid space.
- Children table entries are discarded unless refresh msgs received from children periodically.
- Scribe provides best-effort service, events may be out of order. Reliable services can be built on top of Scribe.

Multicast Tree Repair II

- Rendezvous point failure
  - The state associated with a rendezvous point is replicated across k closest nodes
  - When the root fails, the children detect the failure and send a JOIN message which gets routed to a new node-id numerically closest to the group-id

- Fault detection and recovery is local and accomplished by sending minimal messages

Stronger Reliability

- Scribe provides reliable, ordered delivery only if there are no faults in the multicast tree
- Scribe provides a mechanism to implement stronger reliability
  - Applications built on top of Scribe should provide implementation of certain upcall methods to implement stronger reliability...

Reliability API

- forwardHandler(msg)
  - invoked by Scribe before the node forwards a multicast message to its children
- joinHandler(JOINmsg)
  - invoked by Scribe after a new child has been added to one of the node’s children tables
- faultHandler(JOINmsg)
  - invoked by Scribe when a node suspects that its parent is faulty

The messages can be modified or buffered in these handlers to implement reliability.

Example, Reliable delivery

- forwardHandler
  - Root assigns a sequence number to each message, such that messages are buffered by root and nodes in multicast tree
- faultHandler
  - Adds the last sequence number, n, delivered by the node to the JOIN message
- joinHandler
  - Retransmits buffered messages with sequence numbers above n to new child

Messages must be buffered for an amount of time that exceeds the maximal time to repair the multicast tree after a TCP connection breaks.
Scribe Results

- Experiments
  - Compare the delay, node and link load with IP multicast
  - Scalability test with large number of small groups
- Setup
  - Network topology with 5050 routers (GaTech random graph generator using transit-stub model)
  - Number of scribe nodes: 100,000
  - Number of groups: 1500
  - Group Size: minimum 11 maximum 100,000

Methodological Issues

- Simulation via their own packet-level simulator
- Only considers propagation delay
- Does not take into account queuing delay or packet losses!
- 100,000 nodes!
- Created 1,500 with very varied group sizes

Delay Penalty

- Delay Penalty
  - Measured the distribution of delays to deliver a message to each member of a group using both Scribe and IP multicast
  - Measure Ratio of Average Delay (RAD)
    - 50% groups 1.68
    - max: 2
  - Measure Ratio of Maximum Delay (RMD)
    - 50% of groups: 1.69
    - Max: 4.26
- The message delivery delay is more in Scribe compared to IP Multicast
  - Only in 2.2% of groups it is lower

Node Stress

- Node Stress
  - Measure the number of groups with non-empty children tables for each node
  - Measure the number of entries in the children table in each node
  - The mean number of non-empty children tables per node is only 2.4 although there are 1500 groups, median is 2
  - Results indicate Scribe does a good job of partitioning and distributing the load. This is one of the factors that ensures scalability.
Node Stress II

- Number of table entries per Scribe node (average standard deviation was 3.2)

Link Stress

- **Link Stress**
  - Measure the number of packets that are sent over each link when a message is multicast to each of the 1500 groups
  - Measured mean number of messages per link
    - Scribe: 2.4
    - IP Multicast: 0.7
  - Maximum link stress
    - Scribe: 4031
    - IP multicast: 950
  - Scribe Link stress = 4 x IP Multicast Stress

Link Stress

- Maximum link stress
  - Scribe: 4031
  - IP multicast: 950

Bottleneck Remover

- All nodes may not have equal capacity in terms of computational power and bandwidth
- Under high load conditions, the lower capacity nodes become bottlenecks
- Solution: Offload children to other nodes
  - Choose the group that uses the most resources
  - Choose a child of this group that is farthest away
  - Ask the child to join its sibling which is closest in terms of delay
- This gives an improved performance
- Increases link stress for joining

Bottleneck Remover

- Number of children table entries per Scribe node with the bottleneck remover (average standard deviation was 57)

Scalability Test

- Scalability test with many small groups
  - 30000 groups with 11 members
  - 50000 groups with 11 members
- Scribe Multicast Trees are not efficient for small groups because it creates trees with long paths with no branching
- Scribe Collapse algorithm
  - Collapsed paths by removing nodes
  - Not members of the group
  - Only have one entry in the group’s children table
  - Reduce average link stress from 6.1 to 3.3, average number of children per node from 21.2 to 8.5