History

- Multicast and QoS dominated research literature in the 90's
- Both failed in their attempt to become pervasively available
  - Both now available in enterprises, but not in public Internet
- Both now scorned as research topics

Irony

- The biggest critics of QoS were the multicast partisans
  - And the QoS advocates envied the hipness of mcast...
- They complained about QoS being unscalable
  - Among other complaints....
- Irony #1: multicast is no more scalable than QoS
- Irony #2: scaling did not cause either of their downfalls
- Many now think economics was the problem
  - Revenue model did not fit delivery model

Motivation

- Often want to send data to many machines at once
  - Video distribution (TV broadcast)
  - Teleconferences, etc.
  - News updates
- Using unicast to reach each individual is hard and wasteful
  - Sender state: $O(n)$ and highly dynamic
  - Total load: $\sim O(nd)$ where $d$ is net diameter
  - Hotspot load: load $\sim O(n)$ on host and first link
- Multicast:
  - Sender state: $O(1)$, total load $O(d \log n)$, hotspot load $O(1)$
### Multicast Service Model

- Send to logical group address
  - Location-independent
- Delivery limited by specified scope
  - Can reach “nearby” members
- Best effort delivery

### Target Environment

- LANs connected in arbitrary topology
- LANs support local multicast
- Host network cards filter multicast traffic

### Open Membership Model

- Anyone, anywhere, can join
- Dynamic membership
  - join and leave at will
- Anyone can send at any time
  - Even nonmembers

### Division of Responsibilities

- Host’s responsibility to register interest with networks
  - IGMP
- Network’s responsibility to deliver packets to host
  - Multicast routing protocol
- Left unspecified:
  - Address assignment (random, MASC, etc.)
  - Application-to-group mapping (session directory, etc.)

### Multicast Routing Algorithms

### Routing Performance Goals

- Roughly equivalent to unicast best-effort service in terms of drops/delays
  - Efficient tree
  - No complicated forwarding machinery, etc.
- Low join/leave latency
Two Basic Routing Approaches

- Source-based trees: (e.g., DVMRP, PIM-DM)
  - A tree from each source to group
  - State: $O(GS)$
  - Good for dense groups (all routers involved)

- Shared trees: (e.g., CBT, PIM-SM)
  - A single tree for group, shared by sources
  - State: $O(G)$
  - Better for sparse groups (only routers on path involved)

DVMRP

- Developed as a sequence of protocols:
  - Reverse Path Flooding (RPF)
  - Reverse Path Broadcast (RPB)
  - Truncated Reverse Path Broadcasting (TRPB)
  - Reverse Path Multicast (RPM)

- General Philosophy: multicast = pruned broadcast
  - Don’t construct new tree, merely prune old one

- Observation:
  - Unicast routing state tells router shortest path to S
  - Reversing direction sends packets from S without forming loops

Basic Forwarding Rule

- Routing state:
  - To reach S, send along link L

- Flooding Rule:
  - If a packet from S is received along link L, forward on all other links

  This works fine for symmetric links
  - Ignore asymmetry today

  This works fine for point-to-point links
  - Can result in multiple packets sent on LANs

Example

- Flooding can cause a given packet to be sent multiple times over the same link

Broadcasting Extension

- For each link, and each source S, define parent and child
  - Parent: shortest path to S (ties broken arbitrarily)
  - All other routers on link are children

- Broadcasting rule: only parent forwards packet to L

- Problem fixed

  But this is still broadcast, not multicast!

Multicast = Pruned Broadcast

- Start with full broadcast (RPB)

  If leaf has no members, prune state
  - Send non-membership report (NMR)

  If all children of a router R prune, then router R sends NMR to parent

  New joins send graft to undo pruning
### Problems with Approach

- Starting with broadcast means that all first packets go everywhere
- If group has members on most networks, this is ok
- But if group is sparse, this is lots of wasted traffic
- What about a different approach:
  - Source-specific tree vs shared tree
  - Pruned broadcast vs explicitly constructed tree

### Disadvantages

- Sub-optimal delay
- Small, local groups with non-local core
  - Need good core selection
  - Optimal choice (computing topological center) is NP complete

### Core Based Trees (CBT)

- Ballardie, Francis, and Crowcroft,
  - “Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing”, SIGCOMM 93
- Similar to Deering’s Single-Spanning Tree
- Unicast packet to core, but forwarded to multicast group
- Tree construction by receiver-based “grafts”
  - One tree per group, only nodes on tree involved
- Reduce routing table state from $O(S \times G)$ to $O(G)$

### Why Isn’t Multicast Pervasive?

- Sound technology
- Implemented in most routers
- Used by many enterprises
- But not available on public Internet

### Example

- Group members: M1, M2, M3
- M1 sends data

![Diagram of tree construction by receiver-based "grafts"

- One tree per group, only nodes on tree involved
- Reduce routing table state from $O(S \times G)$ to $O(G)$

### Possible Explanation

[Holbrook & Cheriton ’99]

- Violates ISP input-rate-based billing model
  - No incentive for ISPs to enable multicast!
- No indication of group size (needed for billing)
- Hard to implement sender control
  - Any mcast app can be subject to simple DoS attack!!
- Multicast address scarcity
  - Global allocation required
- Awkward interdomain issues with “cores”
Solution: Single-Source Multicast

- Each group has only one source
- Use both source and destination IP fields to define a group
  - Each source can allocate 16 millions “channels”
  - Use RPM algorithm
- Add a counting mechanism
  - Use a recursive CountQuery message
- Use app-level relays to for multiple sources

How to Make Multicast Reliable?

- FEC can help, but isn’t perfect
- Must have retransmissions
- But sender can’t keep state about each receiver
  - Has to be told when someone needs a packet

Discussion

- Does multicast belong in the network layer?
  - Why not implemented by end hosts?
- How important is economic analysis in protocol design?
  - Should the design drive economics, or the other way around?
- Multicast addresses are “flat”
  - Doesn’t that make it hard for routers to scale?
  - Address allocation and aggregation?
- Should everything be multicast?
- What other delivery models are needed?

SRM Design Approach

- Let receivers detect lost packets
  - By holes in sequence numbers
- They send NACK when loss is detected
- Any node can respond to NACK
- NACK/Response implosion averted through suppression
  - Send NACKs at random times
  - If hear NACK for same data, reset NACK timer
  - If node has data, it resends it, using similar randomized algorithm

Reliable Multicast

- Chosen from the uniform distribution on
  \[ 2(C_{d_{max}}(C_i + C_j)d_{max}) \]
  - \( A \): node that lost the packet
  - \( S \): source
  - \( C_i, C_j \): algorithm parameters
  - \( d_{max} \): latency between \( S \) and \( A \)
  - \( i \): iteration of repair request tries seen
- Algorithm
  - Detect loss → set timer
  - Receive request for same data → cancel timer, set new timer
  - Timer expires → send repair request
Timer Randomization

- Repair timer similar
  - Every node that receives repair request sets repair timer
  - Latency estimate is between node and node requesting repair
- Timer properties – minimize probability of duplicate packets
  - Reduce likelihood of implosion (duplicates still possible)
    - Poor timer, randomized granularity
    - High latency between nodes
  - Reduce delay to repair
    - Nodes with low latency to sender will send repair request more quickly
    - Nodes with low latency to requester will send repair more quickly
    - When is this sub-optimal?

Bounded Degree Tree

- Use both
  - Deterministic suppression (chain topology)
  - Probabilistic suppression (star topology)
- Large $C_2/C_1$ $\Rightarrow$ fewer duplicate requests, but larger repair time
- Large $C_1$ $\Rightarrow$ fewer duplicate requests
- Small $C_1$ $\Rightarrow$ smaller repair time

Chain Topology

- $C_1 = D_1 = 1$, $C_2 = D_2 = 0$
- All link distances are 1

Star Topology

- $C_1 = D_1 = 0$,
- Tradeoff between (1) number of requests and (2) time to receive the repair
  - $C_2 \leq 1$
    - $E(\#\text{ of requests}) = g - 1$
  - $C_2 > 1$
    - $E(\#\text{ of requests}) = 1 + (g-2)/C_2$
    - $E(\text{time until first timer expires}) = 2C_2/g$
  - $C_2 = \sqrt{2}$
    - $E(\#\text{ of requests}) = 4/\sqrt{2}$
    - $E(\text{time until first timer expires}) = 1/\sqrt{2}$

Local Recovery

- Some groups are very large with low loss correlation between nodes
  - Multicasting requests and repairs to entire group wastes bandwidth
- Separate recovery multicast groups
  - e.g. hash sequence number to multicast group address
  - only nodes experiencing loss join group
  - recovery delay sensitive to join latency
- TTL-based scoping
  - send request/repair with a limited TTL
  - how to set TTL to get to a host that can retransmit
  - how to make sure retransmission reaches every host that heard request
Suppression

- Two kinds:
  - Deterministic suppression
  - Randomized suppression
- Subject of extensive but incomplete scaling analysis

Local Recovery

- Large groups with low loss correlation
  - Multicasting requests and repairs to entire group wastes bandwidth
- Separate recovery multicast groups
  - e.g. hash sequence number to multicast group address
  - only nodes experiencing loss join group
  - recovery delay sensitive to join latency
- TTL-based scoping
  - send request/repair with a limited TTL
  - how to set TTL to get to a host that can retransmit?
  - how to make sure retransmission reaches every host that heard request?

Application Layer Framing (ALF)

- Application should define Application Data Unit (ADU)
- ADU is unit of error recovery
  - app can recover from whole ADU loss
  - app treats partial ADU loss/corruption as whole loss
- App can process ADUs out of order

Multicast’s True Legacy

- Efficient delivery to multiple hosts (initial focus)
  - Addressed by SSM and other simple mechanisms
- Logical addressing (pleasant byproduct)
  - Provides layer of indirection
  - Now focus of much architecture research
  - Provided by DHTs and other kinds of name resolution mechanisms