Multicast – Recall

- Multicast to anonymous group; users join/leave the group
- Multicast Performance Metrics
  - Stress
  - Stretch
  - Join delay
  - Router overhead (computation/state)
  - Other...
- Multicast on a LAN segment (directly connected nodes)
- Using IGMP to track membership on a LAN segment
- Generalizing to a switched network

Multicast on a Switched Network

- How to implement multicast over a large switched network?
- Dumb solution: flood (broadcast) to everyone; ignore if you don’t belong to the group
  - Why dumb?
- Good solution must only send it to appropriate parts of the network
  - Network as a graph – how should a multicast packet be forwarded ideally?

Reverse Path Broadcast (RPB)

- When a multicast packet comes in:
  - If it came from the NextHop for the source of the multicast, send it out on all links
    - Better than a flood?
  - Broadcasts the packet away from the source, but does not loop back to it

- How does this approach perform in terms of our metrics?
Problems

- Floods the network, even if part of it does not include a member in the group
- Message forwarded twice over a LAN with two routers connected to it
- What extra capability is needed?

Solutions

- Elect one of the routers as multicast agent
- Possible solution - inform upstream routers which destinations they are your next hop for

Improvement – Truncated Reverse Path Broadcast (TRPB)

- Improve RPB by pruning leaf networks that do not contain a group member
- How to determine if a network is a leaf?
  - A router can tell from the routing protocol messages (if it receives an update from a router on the same network)
  - Can be done through a modification to RIP
  - If a leaf, how do you figure out if it has members in the group?
    - IGMP – Good enough?

Reverse Path Multicasting (RPM)

- Idea: prune full subtrees, not just leaf networks
- Extend TRPB to prune subtrees
  - "on demand" pruning: done only on receipt of message to group
    - If a packet arrives at a router none of whose children have group members, a non membership report (NMR) packet is sent one hop towards the root
    - if the one-hop-back router receives such messages from all its child routers and none of its incident networks have group members, it sends an NMR packet to its predecessor, etc.
    - NMRs are stored with a finite time to live (TTL) – when TTL expires, multicast transmission resumes
  - Essentially, this is Distance Vector Multicast Routing Protocol (DVMRP) – RFC 1075

Link State Multicast

- Recall: link-state – exchange information with all routers about immediate neighbors
- Supporting Multicast –
  - Idea: information about what multicast groups that hosts on your network belong to becomes part of the state – how?
  - The "expanded" state information gets exchanged in the update
    - Need to update when groups appear/disappear on a link
  - Each router can figure out the shortest multicast tree depending on the source and forward packets accordingly (a tree per source per group)
- Protocol is known as Multicast OSPF (MOSPF); RFC 1584
Multicast LS Overhead

- Computing and Storing Multicast trees
  - Expensive: need to compute/store for all senders on all groups
  - Optimization: Use a cache of computed multicast trees
  - Recompute on demand (e.g., when the first packet is seen); LRU to expire cache
  - Tradeoff space (cache) for router cpu

- Problem: group volatility triggers flooding
  - Aggregate changes by delaying updates?

Multicast – Discussion

- Would like optimal tree for each source/group
  - Expensive, does not scale gracefully, especially at intermediate routers
  - Optimization: use a multicast route cache for active groups, construct trees as needed. Still need to keep and exchange information

- Dense vs. Sparse Multicast
  - Dense-mode multicast
    - Most networks have members in the group
    - DVMRP and MOSPF handle dense mode well
  - Sparse-mode multicast
    - Few networks have members
    - DVMRP and MOSPF don’t perform well
    - Would like:
      - Routers not involved do nothing
      - Distribution trees do not require many resources
      - Scalable solutions

Core Based Trees (CBT)

- Idea, a single distribution tree per multicast group; also called “shared tree” approach

- A multicast message is picked up by a router
  - If it is part of the tree, it floods it along the tree
  - If not, it unicasts to the root of the tree (the core) which floods the tree

- Joining the tree
  - Node informs its router using IGMP of interest in joining the group
    - If the router is not already part of the tree, it unicasts a join request to the “core”
    - As each intermediate router receives the request, it sets up a transient join state
      - If router part of the CBT, it sends a Join ACK; otherwise
      - It forwards the join request on to the core

CBT Discussion

- Overhead is good, for G groups with S sources each, need only G trees, instead of S * G
- Performance possibly bad
  - Data does not follow shortest route (average delay up)
  - Traffic concentration: may congest the tree while alternative paths are idle
  - Recall learning bridges — any similarities?
Protocol Independent Multicast (PIM)

- Multicast most expensive when the group is sparse
- PIM distinguishes between sparse and dense multicast
- In the sparse mode
  - Similar to CBT – a shared distribution tree
  - Hosts join/leave groups explicitly using join/prune protocol messages
  - Where to send this message?
    - Every group is assigned a rendezvous point (RP) using a distributed algorithm
    - The RP collects information about members in the group and builds a shared multicast tree
    - It may elect later (“if traffic becomes heavy”) to build a source specific tree that is more optimal
- RFC 2362

PIM Discussion

- Choice of RP important
  - Single point of failure
  - can cause suboptimal operation
- Problem: both receivers and senders need to know addresses of RPs for every group
  - Configured or can extend IGMP to provide this information
- How do we make it real? Multicast across different AS’s
- Multicast requires router state
  - Breaks the E2E principle
  - Adds significant complexity and serious scaling issues

IP Multicast – Summary so far

- Source specific trees most efficient but require too much overhead
  - Flood and Prune (RPM)
    - IGMP to keep track of membership on leaves
    - NMR to prune if no members interested
  - Link layer multicast – efficient trees but not scalable either
- Sparse mode multicast
  - Construct one shared tree
  - Join by linking up to the tree
  - Tree not optimal, but significantly less overhead
MBone (Multicast Backbone)

- Large scale experiment in supporting multicasting in the internet
- Collection of Islands supporting multicasting (overlay network)
- Each Island has a multicast router (e.g., a host running mrouted)
- Routers connected via tunnels (IP-in-IP)
- DVMRP has been the routing protocol being replaced by PIM

Sample applications: video conferencing (vic); shared whiteboard (wb)

Highlights the difficulty of adding functionality to the internet

- Qbone, 6bone

Book on Mbone: http://www.savetz.com/mbone/

Problems with M-bone

- Scalability:
  - large flat network unscalable – especially in the absence of route aggregation
  - Mbone had 10,000 routes at its peak, most were /28 and /32
  - Problem: How to apply aggregation to multicast?

- Manageability
  - No central management – site-by-site basis
  - No control of how sites attach to the backbone
  - Inefficient operation results
A new source for a group becomes active, registers with domain RP.

**Potential Long Term Solutions**

- Grand Unified Multicast (GUM aka BGMP)
  - Associate multicast addresses with domains
  - Single core tree model per group; single root
  - If a domain owns an address, it will be involved in the multicast for it (root will reside there)
  - Requires strict address allocation (paper discusses some schemes)

**Deployment**

- Within Internet, co-locate MBone with true multicast
  - Assign an AS for MBone, to connect it to inter-domain MCast
  - As AS’s transition to true multicast support, they unsubscribe from MBone
  - Paper mentions that number of MBone routes decreasing dramatically while number of MBGP routes increasing dramatically (it is happening?)
- Deployment in Internet 2 (a research backbone connecting universities)
  - Support for native multicast is a design requirement
  - Painful realization that dense mode is unscalable
  - MBGP/PIM-SM/MSDP solution deployed and working reasonably

**Scalability Issues**

- Big join latency
  - SA’s are expensive and delayed
  - Problem for bursty sources
    - Lots of packets then Long silence time
    - Packets lost because “forwarding state” hasn’t been established yet
    - Forwarding state times out by the time the next burst occurs
- RPB for among RPs – RPB not scalable
- Generally believed to be a short term solution
Adding Functionality to Multicast

- Receiver driven Layered Multicast (RLM)
  - How to adapt transmission rates to match heterogeneous link bandwidths
- Scalable Reliable Multicast (SRM)
  - How do you recover from losses
- Unicast – Sender does all the hard work (remember sliding window)
- Multicast
  - Cannot have sender adapt effectively (and scale)

Discussion

- Naming: How to obtain globally unique multicast addresses
  - Some global form of dhcp?
  - How do we do this in a scalable and distributed fashion?
  - Use DNS?
- Security Nightmare?
- Reliability? congestion control?
- User Level multicast (surprisingly, only recently studied)
- Will discuss the papers next time
- Multicast Still an open problem – partial solutions deployed
Determining Capacity

- Who decides what the path capacity is, and how?
- Receiver-driven approach
  - Receiver determines the path bottleneck capacity
    - Using drop rates, for example
  - Receiver decides how the flow gets degraded
    - By choosing the layer it should listen to
  - No changes to routers beyond IP multicast

Rate Determination and Adaptation

- One possibility: Use a fixed rate
  - Lowest common denominator?
  - Send as high as the best link?
- Allow sender to adapt to congestion
  - Solves congestion, but converges to lowest common denominator
- Solution – Adapt to congestion in a heterogeneous way
  - Sender transmits at highest rate
  - Degrade flows as they go down more constrained links (how?)
  - Everyone receives the best possible quality for them

How it works

- Sender sends in multiple layers, each in a separate multicast group
- Receiver listens to as many layers as its bandwidth can take
  - This is its “level of subscription”
  - Routers implicitly configured to add or drop a layer based on receivers joining/leaving the group
  - Multicast tree automatically extended when needed
- Adaptation
  - Congestion – drop a layer
  - How to go back up if there is available capacity?
- We will only skim the implementation details

Key Idea: Layered Compression

- A layered approach to compressing video data
  - Receiving at a low layer, provides bad quality (but viewable video)
  - Successive layers refine the stream
  - Sender transmits all layers
  - Degradation is possible by dropping to the layer matching the available bandwidth
- Alternative is “simulcast”: send different streams concurrently at different quality levels
  - Is this better or worse?
Shared Learning

- Receivers multicast experiment notifications to the group for the layer
  - Receivers interested in layer watch for congestion
    + One experiment serves many receivers
  - Receivers decide based on failed experiments only
    + An experiment can succeed for some receivers but fail for others
  - Should the learning multicast be group wide??

Measuring Capacity

- Receivers probe link capacity by doing “join experiments”
  - Join the group for the next layer; back off if congestion
  - Increase if no congestion
  - Repeat periodically to adapt to changes in the network
- Need to control the frequency of join-experiments
  - Solution used: a join-timer with exponential backoff

Discussion

- Many parameters/magic numbers involved. Not clear how to determine what the best set is, or whether there is one set that fits all
- Relies on cooperate of all sources (one bad source can congest all)
- Is shared learning an optimization or a critical component
  - With enough sources, there will always be one probing a higher layer

Scalability

- What happens if receivers do join-experiments independently?
- Ideally, join experiment rate is independent of group size
  - But we would like the adaptation rate not to suffer
- Solution: shared learning
  - Receivers of a layer learn from experiments to join that layer
  - How?