Multicast Protocols

IGMP - IP Group Membership Protocol
DVMRP - DV Multicast Routing Protocol
MOSPF - Multicast OSPF
(see notes pages for some slides!)

IGMPv2 - Internet Group Management Protocol implements Group Membership

All syst MC group 224.0.0.1

- IGMP runs directly on IP as protocol nr 2.
- TTL == 1 in all IGMP msgs
- Host will wait random[0...Max Resp Time] prior to response and will suppress its response if it sees another response to the same group
IGMPv3 adds selective reception from sources within a Group

Variants:
- General query: GA=0 and Nrof sources=0.
- Group specific query: GA=/=0, Nrof sources=0
- Group and source specific Query

Can exclude listed sources within a Group or include only listed sources within a Group

Experimental routing protocols have been developed for MBone - an overlay MC Internet

<table>
<thead>
<tr>
<th>Shared tree</th>
<th>Source based trees</th>
<th>Domainwide reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIM Sparse*</td>
<td>DVMRP</td>
<td>MOSPF</td>
</tr>
<tr>
<td>Core Based tree*</td>
<td>PIM Dense*</td>
<td></td>
</tr>
</tbody>
</table>

* Relies on Unicast routing protocol to locate MC-sources
Those that don’t, can route MC on routes separate from unicast routes.
Distance Vector Multicast Routing Protocol (DVMRP) is used for MC routing in the MBone

DVMRP header

- **Type=0x13**
- **Major vers =3**
- **Minor vers =0xff**
- **Reserved**
- **Checksum**

[IP protocol=2=IGMP, TTL=1, TOS=internwrk cntrl]

- **‘DVMRP Probe’ [Code=1]** for Neighbor discovery
- **‘DVMRP Report’ [Code=2]** for route exchange
- **‘DVMRP Prune’ [Code=7]** - cut a branch of MC tree
- **‘DVMRP Graft’ [Code=8]** (join) - MC tree expansion
- **‘DVMRP Graft Ack’ [Code=9]** - ack of graft reception

DVMRP is similar to RIP except that sources are like destinations in RIP

Probes are used for neighbor discovery

- **‘DVMRP Probe’** [list of neighbors on this i/f]
- Probes are exchanged on tunnel and phys. i/fs
- Mcasts are not exchanged until two-way neighbor relationship is established
- Routers see each others versions -> compatibility
- Keepalive --> fault detection, restart detection
- sent each 10s, timeout set at 35s
- If list of neighbors is empty - this is leaf ntwrk managed by IGMP

Unicast [whole DVMRP routing table]
Route reports are used to build the source based trees

Each DVMRP router periodically (60s) bcasts to its neighbors
- the list of pairs (source, metric)
- source aggregation according to CIDR may be used

- The receiving MC router calculates the previous hop on each sources Mcast path = the DVMRP router that reports shortest distance \textit{from the source}
- If equal distance --> choose smallest IP address

• ‘DVMRP Report’ \[ inf < \text{metric} < 2 \ast \text{inf} \]
cmp. poisonous reverse, inf=32.

Reports are processed:

\begin{itemize}
  \item Adjusted metric = metric + interface cost
  \item If Metric < inf & Adjust metric \geq \text{inf}
    \begin{itemize}
      \item Set adjusted metric to inf
    \end{itemize}
  \item If Route is new and Adj metric < \text{inf}
    \begin{itemize}
      \item Add route to RT
      \item Delete prune state of more general route
    \end{itemize}
  \item Elseif Route exists
    \begin{itemize}
      \item If Received metric < inf
        \begin{itemize}
          \item Check if Designated forwarder status for (S,G) changes
        \end{itemize}
      \item If Adjusted metric > existing metric
        \begin{itemize}
          \item From same neighbor: update metric, Sch flash update for route
        \end{itemize}
      \item Elseif Adj. metric < existing metric
        \begin{itemize}
          \item Update metric for the route
        \end{itemize}
      \item If sender was different, update RT, schedule flash updates
    \end{itemize}
    \begin{itemize}
      \item Elseif Adj. metric = existing metric: refresh route ...
      \item Elseif Received metric = \text{inf} ...
      \item Elseif Inf < Received Metric < 2 \ast \text{inf} ...
    \end{itemize}
\end{itemize}
Multicast algorithm

RPF - reverse path forwarding

Multicast [from=S, to=G]

If upstream interface = u

No

Yes

Mcast to list of i/f's

- At first mcast from RPF i/f a Forwarding Cache Entry [S,G]:(u,list...) is created using the DVMRP routing table
- List contains all downstream routers that have reported dependency on S
- Router is designated forwarder for downstream nodes
- If Designated forwarder becomes unreachable, Router assumes role of designated until it hears from a better candidate

List of dependent neighbors is used to minimise the MC tree

Router

Multicast [from=S, to=G]

Yes

Cache=[S,G]:(u,list)

- Initially list may contain all mc i/f's but the upstream i/f
- Downstream address is removed from list if
  - =leaf network and G ∉ IGMP DB for this phys. network
  - downstream node has selected another designated forw
  - Prune received from all dependent neighbors on this i/f

Yes

Empty list

Prune [S, G, lifetime]

Remove Cache Entry
On Probe timeout Caches are flushed

- All routes learned from A -> hold-down
- All downstream dependencies ON A are removed
- If A was designated forwarder, a new one is selected for each source, group pair
- Forw cache entries based on A are flushed
- Graft acks to A are flushed.
- Downstream dependencies are removed. If last, prune sent upstream --->

Route hold-down is a state prior to deleting the route

- Routes expire on Report timeout or when an infinite metric is received
- An alternate route (that in RIP caused temporary loops) may exist
- Routers continue to advertise the Route with inf metric for 2 report intervals - this is the hold-down period
- All Forw Cache entries for the Route are flushed
- During hold-down, the route may be taken back, if (<inf and = SAME) metric is received from SAME router
Prunes minimise the Mcast tree

If Known dependent neighbor
If mask and mask=sent mask with (S,G)
  Prune all sources in network (S, mask)
If prune is already active
  reset timeout to new value
If all dependent neighbors have sent prunes
  Remove u from all Forwarding Cache entries
If last u
Prune[S,(m), G]

Upstream Router

Multicasts ...

Dependent Router

Leaf network

If Mcasts keep arriving (3s)
  Resend Prune with exponential backoff =
  double interval each time
  Remove Cache Entry

Prune [S,(netmask),G, Lifetime]

Grafts are used to grow the tree when a new member joins the Group

Graft [S, G]

Graft is
- always acknowledged => if no MCast, nobody is sending
- if no Ack, is resent with exp. backoff retransmissions
- forwarded upstream if necessary

Upstream for [S,G] Router

Graft Ack

Router

Graft Ack
Multicast routing example

Source based trees for $G_1$

Tree for source $S_1$

Tree for source $S_2$

Tree for source $S_3$
Shared Multicast tree for G1

Mbone overlay is based on WSs running DVMRP

Tunneling is used to bypass unicast sections of the Internet
MOSPF (Multicast Extensions to OSPF)

- is an extension of OSPF, allowing Multicast to be introduced into an existing OSPF unicast routing domain
- unlike DVMRP, MOSPF is not susceptible to the normal convergence problems of Distance Vector algorithms.
- limits the extent of multicast traffic to group members, something e.g. DVRMP cannot always do. Restricting the extent of multicast datagrams is desirable for high-bandwidth multicast applications or limited-bandwidth network links (or both).

MOSPF can be deployed gracefully

- Introduces multicast routing by adding a new type of LSA to the OSPF link-state database and by adding calculations for the paths of multicast datagrams.
- The introduction of MOSPF to an OSPF routing can be gradual - MOSPF will automatically route IP multicast datagrams around those routers incapable of multicast routing, whereas unicast routing continues to function normally.
- MOSPF can be, and is in isolated places, deployed in the MBONE. A MOSPF domain can be attached to the edge of the MBONE, or can be used as a transit routing domain within the MBONE’s DVMRP routing system.
An MOSPF Routing Domain

E.g. G1 = 226.1.7.6
E.g. expanding ring search (TTL).
Group m-LSA created and flooded when e.g. host on 128.186.4.0 joins G1.

Group-membership-LSA is created and flooded when an IP user joins an MC-group using IGMP

<table>
<thead>
<tr>
<th>LS Age</th>
<th>Options</th>
<th>LS Type</th>
<th>Link State ID</th>
<th>Advertising Router</th>
<th>LS Sequence Number</th>
<th>LS Checksum</th>
<th>Length</th>
<th>Referenced Link State ID</th>
<th>Referenced LS Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>E-bit. LS Type 6 (group-membership-LSA)</td>
<td>226.1.7.6 (group G1)</td>
<td>128.186.4.1 (router E)</td>
<td>0x80000001</td>
<td>0x3da9</td>
<td>28 bytes</td>
<td>128.186.4.1 (128.186.4.0/24)</td>
<td>2 (network)</td>
</tr>
</tbody>
</table>
MOSPF calculates Shortest-path trees on demand

- Lines with label A are pruned when removing redundant shortest paths.
- Lines with label B are pruned when removing links that do not lead to G1

Result is stored in MC Forwarding Cache Entry
When network conditions change paths are recalculated
Hierarchy reduces the number of calculations

Forwarding Cache Entry stores MC path routing info

A Cache Entry may be deleted at any time -> will be recalculated on demand
Cache entries must be deleted, when changed LSAs are received
- Router-LSA, Network-LSA (on router or link failure or cost change) =>
delete all entries since can’t tell which are affected
- Group-m-LSA: delete entries of that group
- Hierarchy => the farther away the change is the fewer cache entries are deleted
On demand route calculations use Dijkstra’s SPF-algorithm

- Calculation is rooted on the source not the router as for unicast
- For a new mcast, every router performs the same calculation
- Stub networks do not appear in MOSPF calculation (e.g router F)
- Tiebreaks for equal cost routes - previous hop router that has highest address is chosen (e.g. G over E)

Two level hierarchy aggregates both sources and group addresses

- In aggregation some info is lost --> sometimes mcasts are sent needlessly: C->G:to G1
- Presence of sources is reported by summary-LSA with MC-bit set: F to H-> S3+S4 entry
- Area border router advertise Group-m-LSAs to bbone (B: G1, D,E,F:G1, C,D,E:G2) - no exact location
- Routers in non-bbone do not know location of group mmbrs

Wildcard mcast receiver receives all groups
### Summary of Multicast Protocols for the Internet

<table>
<thead>
<tr>
<th></th>
<th>Source based trees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bcast and Prune</td>
</tr>
<tr>
<td>PIM Sparse*</td>
<td></td>
</tr>
<tr>
<td>Core Based tree*</td>
<td></td>
</tr>
<tr>
<td>DVMRP</td>
<td>MOSPF</td>
</tr>
<tr>
<td>PIM Dense*</td>
<td></td>
</tr>
</tbody>
</table>

* Rely on Unicast routing protocol to locate MC-sources.

- Those that don’t, can route MC on routes separate from unicast routes.
- For Shared tree protocols an additional step of finding the Core or Rendesvouz Point must be performed.
- Directories are useful on service management level.