Routing

- Shortest-Path Unicast Routing
  - **Goal**: determine a least-cost path between src and dest
  - **Implementation**: distributed or centralized algorithms
- Classes of (distributed) Algorithms:
  - Distance Vector
  - Link State
- Multicast Routing
  - **Goal**: determine a least-cost path between a src and a subset of all dests
  - **Implementation**: enhanced DV or LS

Distance-Vector Routing

- Deployed for over 2 decades, including original ARPANET
- The idea looks like a dynamic programming solution:

$$D_{i,i} = 0, \forall i$$

$$D_{i,j} = \min_k (d_{i,k} + D_{k,j}), i \neq j$$

- The Bellman-Ford Algorithm solves this iteratively on h:

$$D_{i,i}(h+1) = 0, \forall i$$

$$D_{i,j}(h+1) = \min_k (d_{i,k} + D_{k,j}(h)), i \neq j$$

with initial conditions

$$D_{i,i}(0) = 0, \forall i$$

$$D_{i,j}(0) = \infty, for\ i \neq j$$

- Computational complexity of finding path from one src to all dests is $O(N^3)$, where N is the number of nodes in the network.
- Worst case, the algorithm requires $h \leq N-1$ iterations for each of $N-1$ destinations nodes, and for each node it must consider $N-1$ alternatives in choosing a minimum-cost path.

Distributed DV

- The Distributed Bellman-Ford Algorithm (outline):
  - each node i estimates the cost to each dest j
  - for each neighbor k of i, i collects k's estimate of the cost of the route to j then adds $d_{i,k}$ to it
  - i then compares all newly computed estimates and selects the smallest
- The Algorithm

$$D_{i,i}(t) = 0, \forall i$$

$$D_{i,j}(t) = \min_k (d_{i,k} + D^i_{k,j}(t)), \forall i \neq j$$

  where $D_{i,j}(t)$ is i's estimate of the cost to j available to node i at time t and $D^i_{k,j}$ is k's estimate of the cost to j available to node i at time t.
- Implications and constraints:
  - will converge to the correct minimum-cost routes in finite time, provided no link, costs change after a given time t_0
  - nodes never stop recomputing or sending cost estimates
  - all cost estimates made at time t_0 by i are non-negative, and all cost estimates exchanged with neighbors after t_0 are non-negative
  - old cost estimates do not remain indefinitely in the network
  - no time synchronization of updates is required
  - messages scale with number of destinations

DV Routing Example

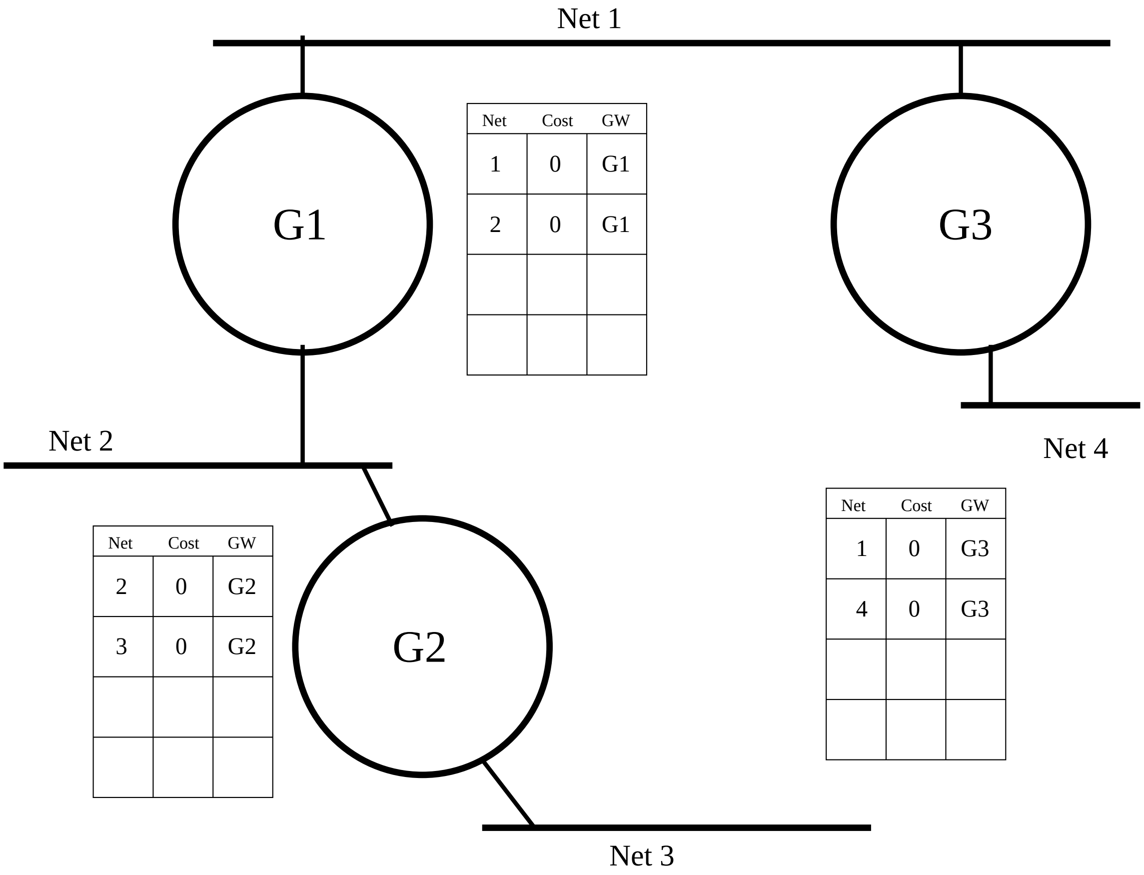


Figure 1: Sample Network and Initial Routing Tables

- **Phase 0** – all nodes know only about their immediately-attached nets
- Cost 0 is means directly attached

DV Routing Example

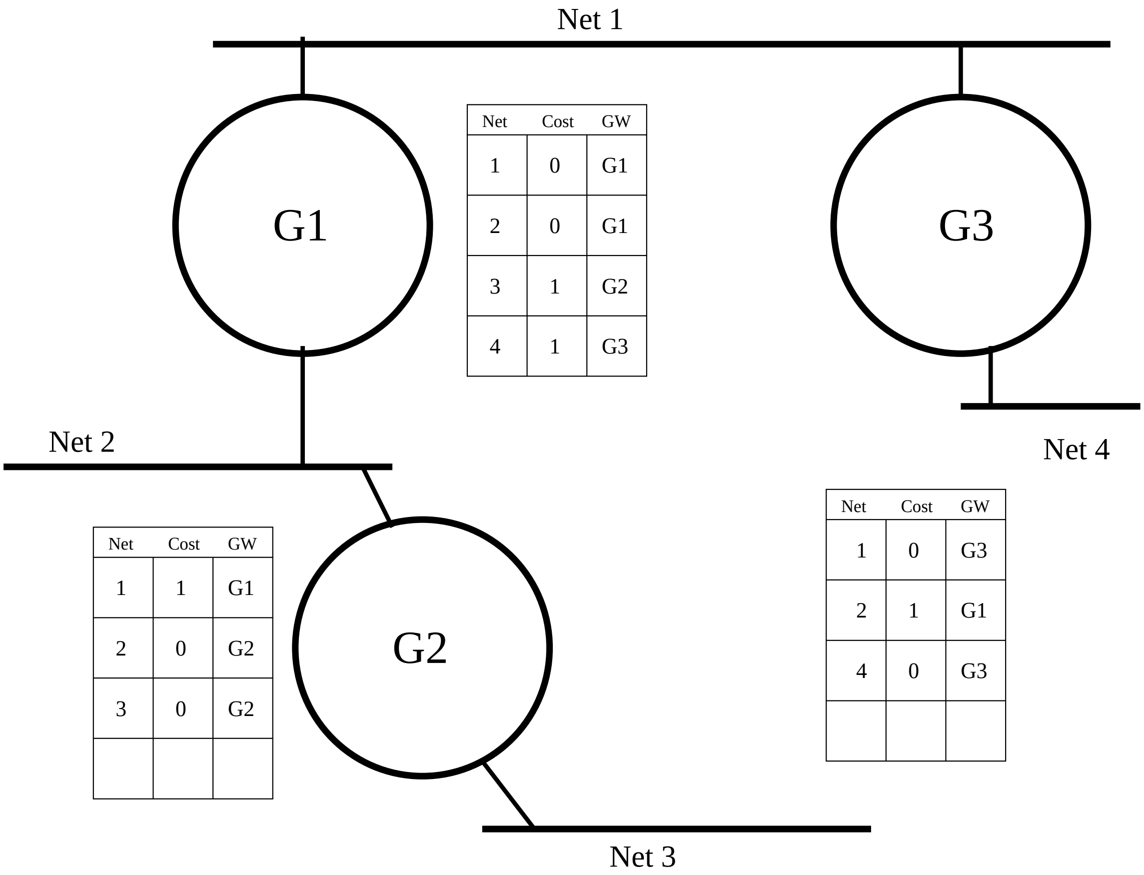
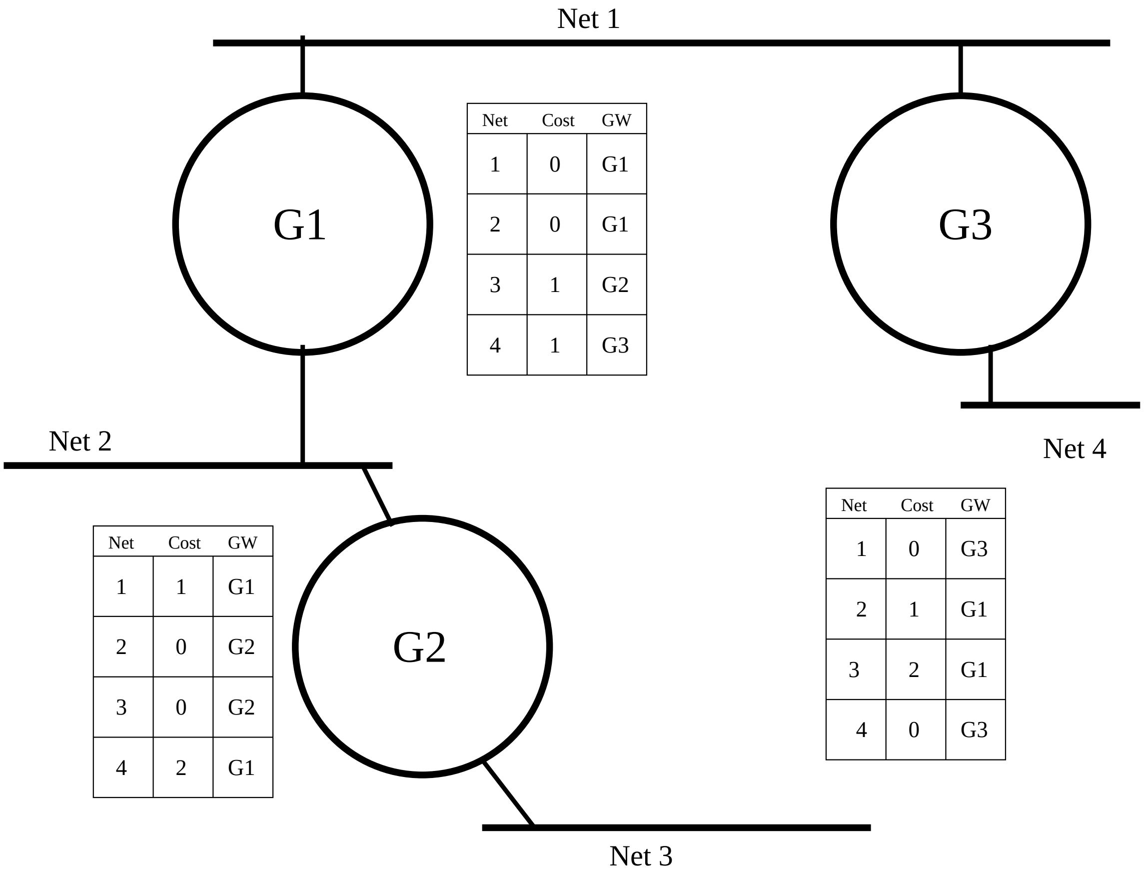


Figure 2: Sample Network and Initial Routing Tables

- **Phase 1** – nodes have broadcast tables once to adjacent nets
- Cost 1 is means one GW must be traversed

DV Routing Example



Net	Cost	GW
1	0	G1
2	0	G1
3	1	G2
4	1	G3

Net	Cost	GW
1	1	G1
2	0	G2
3	0	G2
4	2	G1

Net	Cost	GW
1	0	G3
2	1	G1
3	2	G1
4	0	G3

Figure 3: Sample Network and Initial Routing Tables

- **Phase 2** – nodes have broadcast tables twice to adjacent nets
- Network now has complete tology info

RIP: Routing Information Protocol

- One of the most widely used IGPs
- Widely available as *routed* on UNIX machines
- Widely deployed several years before a spec. appeared
- Specification appeared as RFC 1058 in 1988
- Machines are *active* or *passive*
- Active machines supply RIP to everyone
- Passive machines listen but do not advertise
- Hop count 1 indicates gateway on connected network
- Hop count 0 indicates local machine
- Hop counts metric doesn't always produce optimal routes (consider a longer path over a faster network)
- advertisements done by broadcast

RIP Features

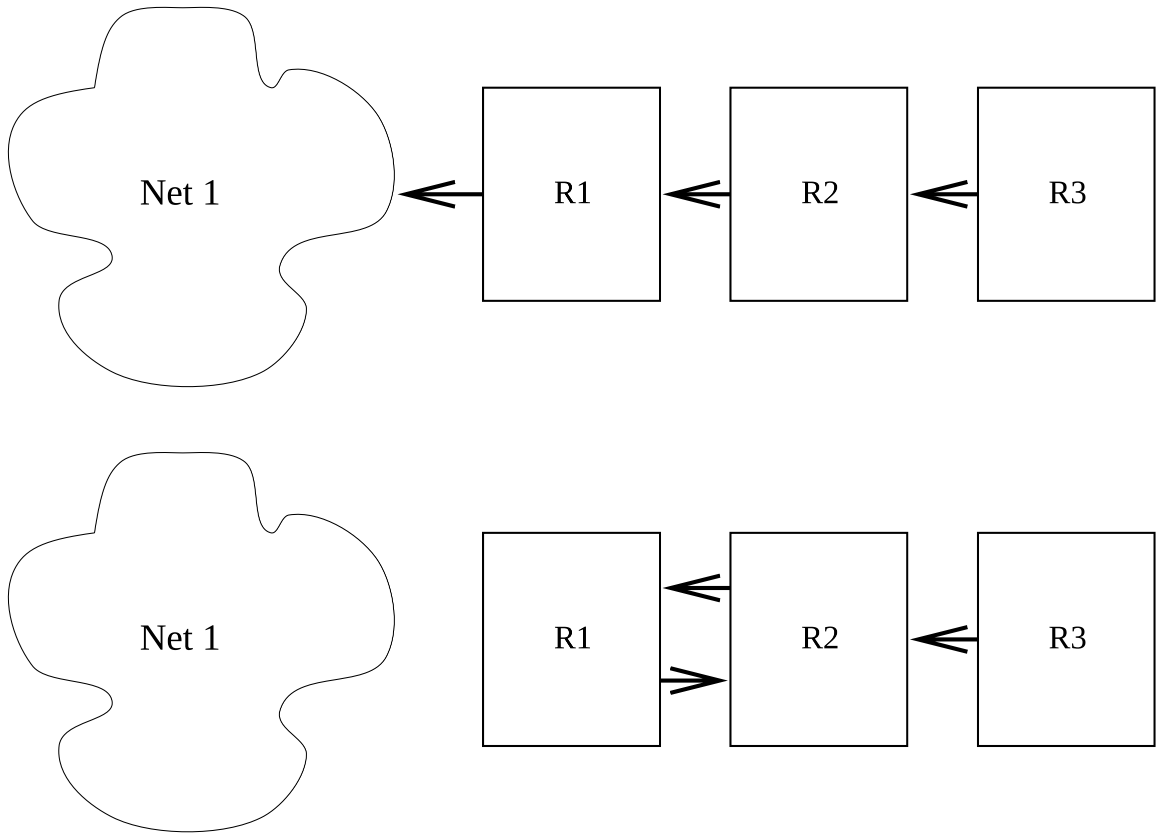
- RIP uses UDP for transport (max size 512 data bytes)
- longest path is 15 hops
- supports both host and network routes
- will converge if topology doesn't change and messages aren't infinitely delayed
- no central clock required; random time added to avoid synchronization
- active RIP senders broadcast every 30 secs
- routes time out if not refreshed in 180 secs
- subnets only supported for local networks
- supports other address families (but nobody uses this)
- supports a *default* address

Special Rules for RIP

- Existing routes should only be replaced by strictly shorter routes
  - prevents oscillations between two equal routes
- Route timeouts:
  - routes become invalid if not seen again in 180 seconds
- Problems:
  1. must avoid routing loops
  2. must use low value for maximum distance (currently 16 hops)
  3. try to avoid *count to infinity* (slow convergence) problem

Slow Convergence

- Basic DV scheme is guaranteed to converge given no changes in link costs after a certain amount of time
- In practice, link costs can change due to failures (sw or hw)
- DV schemes can fail to converge entirely and result in *routing loops*
- Simple example of the *count to infinity* problem:



Split Horizon Techniques

- Observations:
  - when a gateway stops advertising a route, timeouts must expire
  - when a new network or better metric is advertised, it's immediately incorporated into gateways' tables
  - Thus:

    good news travels quickly,
    bad news travels slowly

- Split Horizon is used in DV routing (incl RIP) to reduce slow convergence
- *Idea:* exclude a destination from advertisement to a neighbor if the next hop to that destination is that neighbor
- Poisoned Reverse includes the advertisements, but with an infinity metric, making bad news look like good news
- So, Split Horizon with Poisoned Reverse can improve the slow convergence problem, but the infinity metrics take up extra room in advertisements
- *Problem:* does not cover all topologies

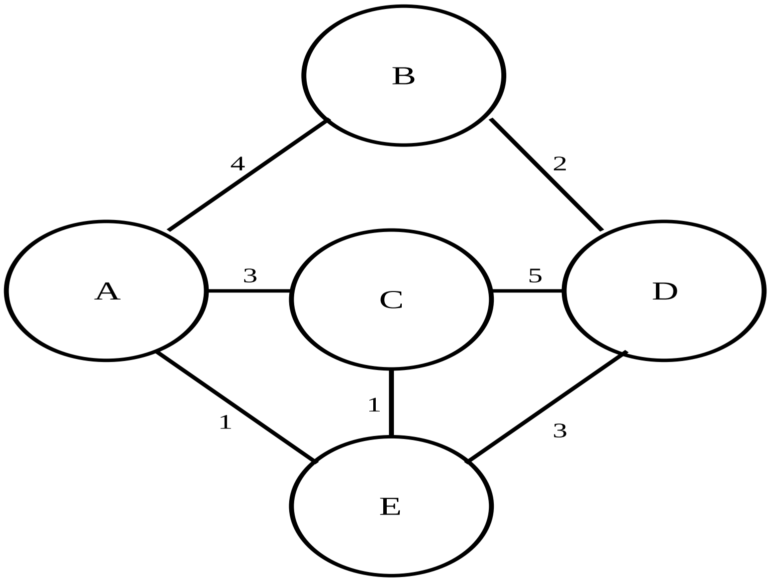
Triggered Updates

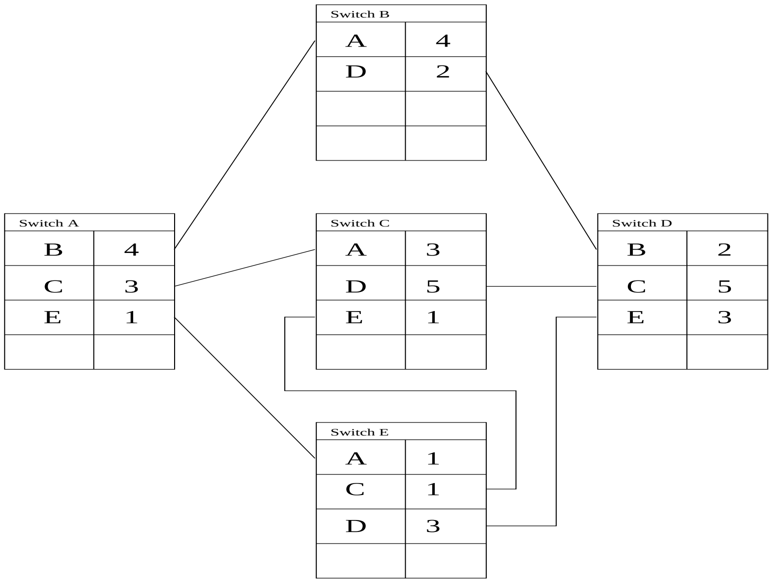
- Normally, routing updates are issued based on timer
- Triggered Updates modify this behavior:
  - distribute an update immediately after discovering a change in a cost metric to a destination
  - with a short delay, may be combined with other triggered updates to form smaller subsequent updates
  - will cause a cascade of updates to all routers providing routing to the affected destination
- Holddowns:
  - *Issue:* some routers may provide a periodic update while another is providing a triggered update. When this happens it is possible to have an old route reinstated.
  - With *hold down*, when a router removes a route to a particular destination, it will disable acceptance of new routes to this destination for a specified hold down period
  - does cause a slowing down for routers to learn correct new routes

Link-State Routing

- A (comparatively) new alternative to Distance-Vector Routing
- First developed as the "new" routing algorithm in ARPANET (1979)
- Components:
  - each router keeps a distributed, replicated database of network topology
  - each router advertises its operating interfaces in a (*link-state advertisement*)
  - LSA's are propagated using *flooding* (or multicasting)
  - routers compute routes to all destinations, often using the *Dijkstra algorithm*
- Flooding:
  - each switch propagates a packet on all interfaces except the one on which it was received
  - must be reliable and quick for efficient LSA distribution
- The Dijkstra Algorithm
  - computes a shortest-path to all possible destinations from local router
  - complexity is $O(n \log(n))$

Link-State Example





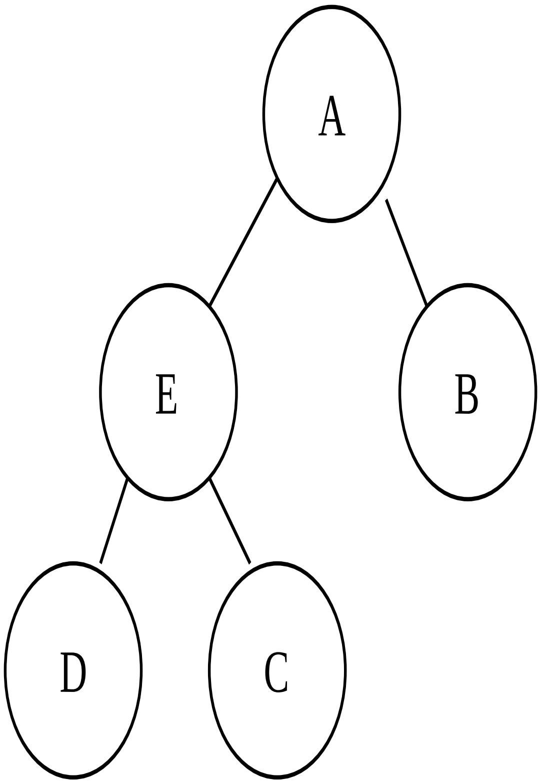
SPF Calculation

- Start with two sets: S and C (shortest path and candidate)
- Let a be the switch performing the calculation
- Let $N(X)$ be the neighbors of node X
- Let $D(X, Y)$ be the minimum distance from node X to node Y
- To start, let $cur \leftarrow S \leftarrow a$ and $C \leftarrow \emptyset$.
  1. $C \leftarrow C \cup (N(cur) - S)$
  2. if $C = \emptyset$, done
  3. $cur \leftarrow \{z | \min_{z \in C} D(a, z)\}$ (the $O(\log n)$ step)
  4. $S = S \cup cur$, $C = C - cur$
  5. Go to 1
- In the example given, this would mean:

  Initialization $S = \{a\}, C = \emptyset$
  Iteration 1 $S = \{a\}, C = \{b, c, e\}$
  Iteration 1 $S = \{a, e\}, C = \{b, c\}$
  Iteration 2 $S = \{a, e\}, C = \{b, c, d\}$
  Iteration 2 $S = \{a, e, c\}, C = \{b, d\}$
  Iteration 3 $S = \{a, e, c, b\}, C = \{d\}$
  Iteration 4 $S = \{a, e, c, b, d\}, C = \emptyset$

Example SPF Results

- The resultant shortest-path tree and routing table are:



Dest	Nxt Hop	Cost
A	-	-
B	B	4
C	E	2
D	E	4
E	E	1

New ARPANET Routing

- Algorithm calculates least-delay paths
- control traffic under 1% of link BW
- routing calculations less than 2% of switch CPU
- Particulars:
  - each switch originated a single LSA listing all the switch's neighbors and corresponding packet delays
  - packet delays expressed as 10s average
  - thus, it dynamically responds to congestion
  - LSA's were on the average short (176 bits)
  - specialized flooding algorithm
- Flooding
  - delay of 60s to advertise a new link
  - flooding still occurs over these links
  - switches thus receive a full LSA database before forwarding user traffic
  - old LSA's from same switch would time-out
- Specialized *incremental* Dijkstra calculation

Network Layer Multicast

- Single Link-Layer multicast already existing
- Desire to extend to Network Layer
- Benefits
  - efficient multidestination delivery
  - location-independent addressing
- Groups, Membership and Scope
  - group of hosts that should receive a multicast packet (in IP multicast a class D address)
  - sender need not know membership
  - sender need not be a member itself
  - can be *scope* limited, possibly at admin boundaries

RPF and RPB

- Reverse Path Flooding (RPF)
  - a router forwards a broadcast originating at S iff it arrives via the shortest path from the router back to S (the "reverse path")
  - packet is forwarded onto all links except the one on which it arrived
- When sending multicast on multicast capable, multiaccess LANs, the link-layer multicast is used
- *Major Drawback (as a broadcast mechanism):* any single broadcast packet may be transmitted more than once across any link, up to the number of routers that share that link
- Reverse Path Broadcasting (RPB)
  - eliminates the duplicates of RPF
  - on each link a "parent" router is identified, relative to each possible source S
  - parents are the routers with minimum distance to S (or the one with lower address in the case of a tie)
- These techniques can be used to broadcast a packet to all destinations

TRPB

- Truncated Reverse Path Broadcasting (TRPB)
  - want to avoid broadcasting all traffic then filtering it out
  - so, want to *prune* RPB tree back to active participants
- *Leaf Links:*child links not used by any other router to reach S
- Routers must now be able to:
  1. identify leaves
  2. detect group membership
- Identification is done by hosts periodically reporting their interest in groups (currently, with IGMP)
- The forwarding algorith is now:

  If a multicast packet with source address S and destination group G arrives from the *next-hop-link* for S, forward a copy of the packet on all *child* links for S, except leaf links that have no members of G

RPM

- Reverse Path Multicasting (RPM)
  - use pruning, as in TRPB
  - only construct multicast trees that are actually being used
- When a new source *first* sends to a group, delivery occurs to all links on the SP broadcast tree except nonmember leaves
- routers receiving mcast packets having only leaves as children with no members generate a *non-membership report* (NMR)
- NMR's indicate non-membership for a $(source, group)$
- NMR propagate back up the distribution tree
- if a given upstream router receives NMR's from all its child routers and its own child links have no members, it can generate its own NMR
- NMRs will time-out, leading to automatic grafting
- *graft* messages can be used to un-do a previous NMR