CSE 461 – Routing

David Wetherall
djw@cs.washington.edu
Routing

• Focus:
  – How to find and set up paths through a network

• Distance-vector and link-state
• Shortest path routing
• Key properties of schemes
• Multicast

<table>
<thead>
<tr>
<th>Application</th>
<th>Transport</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Link</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Physical</td>
</tr>
</tbody>
</table>
Routing versus Forwarding

- Routing is the process by which all nodes exchange control messages to calculate the *routes* packets will follow
  - Distributed process with *global* goals; emphasis is *correctness*
  - Nodes build a routing table that models the global network

- Forwarding is the process by which a node examines packets and sends them along their *paths* through the network
  - Involves *local* decisions; emphasis is *efficiency*
  - Nodes distill a forwarding table from their routing table (keyed by packet attributes, e.g., address) that gives the *next hop*
Packet (Datagram) Forwarding
What is a “best” path anyhow?

• Ideally paths that:
  – Are as direct as possible (low latency)
  – Carry as much traffic as the network will fit (high bandwidth)
  – Carry traffic well for all of the nodes (fairness)

• This is a resource allocation problem with multiple constraints. Depends on topology and who sends how much traffic to who, which changes over time. Yikes!

• We want a simple, distributed solution
Lowest cost ("shortest path") routes

- Compute paths independently for different node pairs
  - Assign a cost or weight to each link
  - Find lowest total weight path between source/dest

- Typically costs are fixed
  - Does not take hotspots into account
  - Has simple subset optimality properties

- Costs usually set as a function of bandwidth and delay
  - More direct paths mean low latency and high bandwidth
  - Can tweak (traffic engineering) to match traffic to topology
Sink trees – union of routes to a node

- Example to reach B, assumes cost for all links is “1”.

Network

Sink Tree for B
Sink trees – work over example
Equal-cost multi-path (ECMP)

- Generalization for load balancing
  - Allow multiple paths if they have the same lowest cost

- Single path lowest cost routing produces a spanning tree
- ECMP produces a directed acyclic graph (DAG)
  - Still no possibility of loops
  - Simple for nodes: just keep a list of next hops

- Q: How to map traffic to the multiple paths?
Sink “tree” for ECMP case

- Cost for all links is still “1”. Tree becomes a DAG
To find trees – two routing methods

• Distance-vector and Link-state

• Scenario:
  – You’re driving from Seattle to Boston.
  – Gas station attendants in each city will tell you which way to go next to head towards your destination. But how do they know?

• Link-state method:
  – Every attendant shares their local cities with all others, makes their own map of the US, and consults it to direct you

• Distance-vector method:
  – Every attendant tells their neighbors the mileage to all cities and keeps the best directions to direct you
Distance Vector Algorithm

• Each router maintains a vector of costs to all destinations as well as routing table giving next hops
  – Initialize neighbors with known cost, others with infinity

• Periodically send copy of distance vector to neighbors

• On reception of a vector, if your neighbor’s path to a destination plus cost to that neighbor cost is better
  – Update the cost and next-hop in your outgoing vectors

• Assuming no changes, will converge to shortest paths
DV Example

- Consider the activity at node J
DV problem -- dynamics

- Consider knowledge of cost to reach A at other nodes

Desired convergence

“Count to infinity scenario”
DV problem -- dynamics

- Good news (better routes) propagate quickly
- Bad news (failures) propagate slowly
  - inferred by exploration
- Leads to “count to infinity” loops
  - Many heuristics (split horizon, poison reverse)
  - Takes ordered updates to eliminate (e.g., EGIRP uses diffusing computations) that are complicated and slow convergence
  - No great solutions

- No longer widely used except for resource constrained or legacy networks.
Routing Information Protocol (RIP)

- DV protocol with hop count as metric
  - Infinity value is 16 hops; limits network size
  - Includes split horizon with poison reverse
- Routers send vectors every 30 seconds
  - With triggered updates for link failures
  - Time-out in 180 seconds to detect failures

- RIPv1 specified in RFC1058
  - www.ietf.org/rfc/rfc1058.txt
- RIPv2 (adds authentication etc.) in RFC1388
  - www.ietf.org/rfc/rfc1388.txt
Link State Routing

• Same assumptions/goals, but different idea than DV:
  – Tell all routers the topology and have each compute best paths
  – Two phases:
    1. Topology dissemination (flooding)
    2. Shortest-path calculation (Dijkstra’s algorithm)

• Why?
  – In DV, routers hide their computation, making it difficult to decide what to use when there are changes
  – With LS, faster convergence and hopefully better stability
  – It is more complex though …
LS example database

• Q: what is the flooding rule to build the database?
• Q: how are shortest paths computed from the database?
Shortest Paths: Dijkstra’s Algorithm

• Graph algorithm for single-source shortest path (i.e., sink tree)

\[ \begin{align*}
S & \leftarrow \{\} \\
Q & \leftarrow \text{<all nodes keyed by distance>} \\
\text{While } Q \neq \{\} & \\
& \quad \text{u} \leftarrow \text{extract-min}(Q) \\
& \quad S \leftarrow S \text{ plus } \{u\} \\
& \quad \text{for each node } v \text{ adjacent to } u & \\
& \quad \quad \text{“relax” the cost of } v
\end{align*} \]

\( \Rightarrow \) \text{u is done, } \text{add to shortest paths}
Dijkstra Example – Step 1
Dijkstra Example – Step 2
Dijkstra Example – Step 3
Dijkstra Example – Step 4
Dijkstra Example – Step 5
Dijkstra Example – Done
Open Shortest Path First (OSPF)

- Widely-used Link State protocol today; see also ISIS

- Basic link state algorithms plus many features:
  - Load balancing: multiple equal cost routes
  - Extra hierarchy: partition into routing areas
  - Authentication of routing messages
Routing – desirable properties

- Correctness
- Network efficiency
- Network fairness

- Rapid convergence
  - To correct routes that are stable after changes, with minimal transient loss

- Scalability
  - Of messages and router state
  - Particularly an issue for large, mobile, or multicast networks
# Comparison

<table>
<thead>
<tr>
<th>Property</th>
<th>Distance Vector</th>
<th>Link State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>Yes - Distributed Bellman Ford</td>
<td>Yes - Replicated shortest path</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Approx - Least cost paths</td>
<td>Approx - Least cost paths</td>
</tr>
<tr>
<td>Fairness</td>
<td>Approx - Least cost paths</td>
<td>Approx - Least cost paths</td>
</tr>
<tr>
<td>Convergence</td>
<td>Slow – many exchanges</td>
<td>Fast – prop plus compute</td>
</tr>
<tr>
<td>Scalability</td>
<td>Good – O(1) per node/link</td>
<td>Moderate – at least O(edges)</td>
</tr>
</tbody>
</table>
Resource allocation timescales today

• From fast (very reactive) to slow (carefully planned)
  – Use of different timescales largely decouples mechanisms

• Congestion control
  – Adapts to packet loss; slows source

• Routing
  – Adapts to failures; finds paths with connectivity

• Traffic engineering
  – Route adjustments for cost/performance (e.g., weights)

• Provisioning
  – Build out network to match traffic workload
What didn’t work: early schemes to adapt weights to traffic load

- Early attempt (ARPAnet) to use routing for congestion control – but not stable
- Automatically adjusted weight (right) based on link properties and traffic load
- Capacity dominates at low load; we only try to move traffic if high load

What happens with traffic from A to B?
Delivery models

- **Unicast**
  - single sender to single receiver

- **Broadcast**
  - Single sender to all receivers

- **Multicast**
  - Single sender to multiple (but not all) receivers (in a group)

- **Anycast**
  - Single sender to nearest receiver in a set
Broadcast Routing

Broadcast sends a packet to all nodes

- RPF (Reverse Path Forwarding): send broadcast received on the link to the source out all remaining links
- Alternatively, can build and use sink trees at all nodes

Network

Sink tree for I is efficient broadcast

RPF from I is larger than sink tree
Anycast

-Simple extension for DV and LS algorithms
-Same destination “appears” at multiple places
  - Each router chooses the next hop with the lowest cost to the destination as before

-Used in the Internet for root nameservers
  - This is BGP routing across ISPs though, not within an ISP
Anycast example

Route to closest instance of “1”

Same thing viewed as a sink tree
Multicast

- A long and checkered history:
  - Multicast is simple on LANs (just broadcast) and useful for service discovery (“Oi! Who is the printer here?”)
  - Brilliant idea – let’s add it to the Interent
  - But it turned out to be complex, motivated by bandwidth efficiency, and lacking a killer application
  - Finally happening, given simpler schemes and apps like IPTV for an ISP and datacenter distribution
Multicast components

• Requires group membership management
  – To decide who is in the group of receivers
  – IGMP is used; hosts subscribe via routers

• Requires spanning trees to be computed
  – Key challenges are scalability and cross-ISP deployment
  – Handle dense and sparse cases separately
  – Dense: start with broadcast and prune a little
  – Sparse: make a tree just for nodes who need to know
Multicast Routing (1) – Dense Case

Multicast sends to a subset of the nodes called a group
- Uses a different tree for each group and source

Network with groups 1 & 2
Multicast tree from S to group 1
Multicast tree from S to group 2
Spanning tree from source S
Multicast Routing (2) – Sparse Case

CBT (Core-Based Tree) uses a single tree to multicast

- Tree is the sink tree from core node to group members
- Multicast heads to the core until it reaches the CBT

Sink tree from core to group 1
Multicast is send to the core then down when it reaches the sink tree