CompSci 356: Computer Network Architectures

Lecture 12: Dynamic routing protocols: Link State
Chapter 3.3.3

Xiaowei Yang
xwy@cs.duke.edu
Today

- Routing Information Protocol
- Link-state routing
  - Algorithm
  - Protocol: Open shortest path first (OSPF)
RIP - Routing Information Protocol

• A simple intra-domain protocol

• Straightforward implementation of Distance Vector Routing

• Each router advertises its distance vector every 30 seconds (or whenever its routing table changes) to all of its neighbors

• RIP always uses 1 as link metric

• Maximum hop count is 15, with “16” equal to “∞”

• Routes are timeout (set to 16) after 3 minutes if they are not updated
RIP - History

• Late 1960s: Distance Vector protocols were used in the ARPANET

• Mid-1970s: XNS (Xerox Network system) routing protocol is the ancestor of RIP in IP (and Novell’s IPX RIP and Apple’s routing protocol)

• 1982 Release of routed for BSD Unix

• 1988 RIPv1 (RFC 1058)
  - classful routing

• 1993 RIPv2 (RFC 1388)
  - adds subnet masks with each route entry
  - allows classless routing

• 1998 Current version of RIPv2 (RFC 2453)
RIPv1 Packet Format

- **IP header**
- **UDP header**

**RIP Message**

- Command
- Version: Set to 00...0
- address family: Set to 00.00
- 32-bit address
- Unused (Set to 00...0)
- Unused (Set to 00...0)
- metric (1-16)

- One RIP message can have up to 25 route entries

- Up to 24 more routes (each 20 bytes)

1: request
2: response
2: for IP

Address of destination
Cost (measured in hops)
RIPv2

• RIPv2 extends RIPv1:
  – Subnet masks are carried in the route information
  – Authentication of routing messages
  – Route information carries next-hop address
  – Uses IP multicasting to send routing messages

• Extensions of RIPv2 are carried in unused fields of RIPv1 messages
RIPv2 Packet Format

- IP header
- UDP header
- RIP Message

Command | Version | Set to 00...0
--- | --- | ---
address family | Set to 00.00

- 32-bit address
- Unused (Set to 00...0)
- Unused (Set to 00...0)
- metric (1-16)

One route entry (20 bytes)

Up to 24 more routes (each 20 bytes)

32 bits

One RIP message can have up to 25 route entries

1: request
2: response
2: for IP

Address of destination
Cost (measured in hops)

2: RIPv2
**RIPv2 Packet Format**

<table>
<thead>
<tr>
<th>Command</th>
<th>Version</th>
<th>Set to 00.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>address family</td>
<td>route tag</td>
<td></td>
</tr>
<tr>
<td>IP address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subnet Mask</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Next-Hop IP address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>metric (1-16)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Up to 24 more routes (each 20 bytes)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **IP header**
- **UDP header**
- **RIPv2 Message**

- Used to provide a method of separating "internal" RIPv2 routes (routes for networks within the RIP routing domain) from "external" RIPv2 routes.
- Subnet mask for IP address.
- Identifies a better next-hop address on the same subnet than the advertising router, if one exists (otherwise 0….0).

**Notes:**
- Command: The command field specifies the type of message (e.g., 2: RIPv2).
- Version: The version field is set to 00.00.
- Address family: Indicates the address family (IPv4 or IPv6).
- Route tag: A 32-bit value used to identify different route sets within the same routing domain.
- IP address: The IP address of the destination or next-hop router.
- Subnet Mask: The subnet mask for the IP address.
- Next-Hop IP address: The IP address of the next-hop router.
- Metric: A 16-bit value indicating the number of hops to reach the destination.
- Additional routes: Up to 24 more routes (each 20 bytes).

**Diagram:**
- 32 bits
- One route entry (20 bytes)
- Up to 24 more routes (each 20 bytes)
RIP Messages

- This is the operation of RIP in *routed*. Dedicated port for RIP is UDP port 520.

- Two types of messages:
  - **Request messages**
    - used to ask neighboring nodes for an update
  - **Response messages**
    - contains an update
Routing with RIP

- **Initialization:** Send a request packet (command = 1, address family=0..0) on all interfaces:
  - RIPv1 uses broadcast if possible,
  - RIPv2 uses multicast address 224.0.0.9, if possible
requesting routing tables from neighboring routers

- **Request received:** Routers that receive above request send their entire routing table

- **Response received:** Update the routing table

- **Regular routing updates:** Every 30 seconds, send all or part of the routing tables to every neighbor in an response message

- **Triggered Updates:** Whenever the metric for a route change, send the entire routing table.
RIP Security

- Issue: Sending bogus routing updates to a router
- RIPv1: No protection
- RIPv2: Simple authentication scheme

![Diagram of RIPv2 Message structure]

<table>
<thead>
<tr>
<th>Command</th>
<th>Version</th>
<th>Set to 00.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xffff</td>
<td></td>
<td>Authentication Type</td>
</tr>
<tr>
<td>Password (Bytes 0 - 3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Password (Bytes 4 - 7)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Password (Bytes 8 - 11)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Password (Bytes 12 - 15)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Up to 24 more routes (each 20 bytes)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RIP Problems

• RIP takes a long time to stabilize
  – Even for a small network, it takes several minutes until the routing tables have settled after a change

• RIP has all the problems of distance vector algorithms, e.g., count-to-Infinity
  » RIP uses split horizon to avoid count-to-infinity

• The maximum path in RIP is 15 hops
Today

- RIP
- Link-state routing
  - Algorithm
  - Protocol: Open shortest path first (OSPF)
Distance Vector vs. Link State Routing

- **DV only sees next hop “direction”**
  - Node A: to reach F go to B
  - Node B: to reach F go to D
  - Node D: to reach F go to E
  - Node E: go directly to F

- **Wrong directions lead to wrong routes**
  - Count to infinity
Distance Vector vs. Link State Routing

• In link state routing, each node has a complete map of the topology

• If a node fails, each node can calculate the new route

• **Challenge:** All nodes need to have a consistent view of the network
Link State Routing: Basic operations

1. Each router establishes *link adjacency*

2. Each router generates a *link state advertisement (LSA)*

3. Each router maintains a database of all received LSAs (*topological database* or *link state database*)

4. Each router runs the Dijkstra’s algorithm
Link state routing: graphical illustration

Collecting all pieces yield a complete view of the network!
Operation of a Link State Routing protocol

Received LSPs → Link State Database

LSPs are flooded to other interfaces

Dijkstra’s Algorithm → IP Routing Table
Reliable flooding

• We’ve learned a flooding algorithm used by Ethernet switches

• Question: why is it insufficient for link-state routing?
  – Lost LSAs may result in inconsistent topologies at different routers
  – Inconsistent topologies may lead to routing loops
Reliable flooding

• LSPs are transmitted reliably between adjacent routers
  – ACK and retransmission

• For a node $x$, if it receives an LSA sent by $y$
  – Stores $LSA(y)$ if it does not have a copy
  – Otherwise, compares SeqNo. If newer, store; otherwise discard
  – If a new $LSA(y)$, floods $LSA(y)$ to all neighbors except the incoming neighbor
An example of reliable flooding

(a) X → A → B → D

(b) X → A

(c) X → A → B → D

(d) X → A
When to flood an LSP

• Triggered if a link’s state has changed
  – Detecting failure
    • Neighbors exchange hello messages
    • If not receiving hello, assume dead

• Periodic generating a new LSA
  – Fault tolerance (what if LSA in memory is corrupted?)
Path computation

Dijkstra’s Shortest Path Algorithm for a Graph

**Input:** Graph \((N,E)\) with
- \(N\) the set of nodes and \(E\) the set of edges
- \(c_{vw}\) link cost \((c_{vw} = \infty\) if \((v,w) \notin E, c_{vv} = 0)\)
- \(s\) source node.

**Output:** \(D_n\) cost of the least-cost path from node \(s\) to node \(n\)

\[
M = \{s\};
\]

for each \(n \notin M\)
\[
D_n = c_{sn};
\]

while (\(M \neq \text{all nodes}\)) do

Find \(w \notin M\) for which \(D_w = \min\{D_j ; j \notin M\}\);
Add \(w\) to \(M\);

for each neighbor \(n\) of \(w\) and \(n \notin M\)
\[
D_n = \min[ D_n, D_w + c_{wn} ];
\]
Update route;

enddo
Practical Implementation: forward search algorithm

- More efficient: extracting min from a smaller set rather than the entire graph
- Two lists: Tentative and Confirmed
- Each entry: (destination, cost, nextHop)

1. Confirmed = {(s,0,s)}
2. Let Next = Confirmed.last
3. For each Nbr of Next
   - Cost = my→Next + Next → Nbr
     - If Neighbor not in Confirmed or Tentative
       - Add (Nbr, Cost, my.NextHop(Next)) to Tentative
     - If Nbr is in Tentative, and Cost is less than Nbr.Cost, update Nbr.Cost to Cost
4. If Tentative not empty, pick the entry with smallest cost in Tentative and move it to Confirmed, and return to Step 2
   - Pick the smallest cost from a smaller list Tentative, rather than the rest of the graph
<table>
<thead>
<tr>
<th>Step</th>
<th>Confirmed</th>
<th>Tentative</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(D,0,-)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Confirmed</td>
<td>Tentative</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>1</td>
<td>(D,0,-)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(D,0,-)</td>
<td>(B,11,B), (C,2,C)</td>
</tr>
<tr>
<td>3</td>
<td>(D,0,-), (C,2,C)</td>
<td>(B,11,B)</td>
</tr>
<tr>
<td>4</td>
<td>(D,0,-), (C,2,C)</td>
<td>(B,5,C) (A,12,C)</td>
</tr>
<tr>
<td>5</td>
<td>(D,0,-), (C,2,C), (B,5,C)</td>
<td>(A,12,C)</td>
</tr>
<tr>
<td>6</td>
<td>(D,0,-), (C,2,C), (B,5,C)</td>
<td>(A,10,C)</td>
</tr>
<tr>
<td>7</td>
<td>(D,0,-), (C,2,C), (B,5,C), (A,10,C)</td>
<td></td>
</tr>
</tbody>
</table>
OSPF

• **OSPF = Open Shortest Path First**
  – Open stands for open, non-proprietary

• A link state routing protocol

• The complexity of OSPF is significant
  – RIP (RFC 2453 ~ 40 pages)
  – OSPF (RFC 2328 ~ 250 pages)

• History:
  – 1989: RFC 1131  OSPF Version 1
Features of OSPF

• Provides authentication of routing messages
  – Similar to RIP 2

• Allows hierarchical routing
  – Divide a domain into sub-areas

• Enables load balancing by allowing traffic to be split evenly across routes with equal cost
OSPF Packet Format

OSPF packets are not carried as UDP payload! OSPF has its own IP protocol number: 89

TTL: set to 1 (in most cases)

Destination IP: neighbor’s IP address or 224.0.0.5 (ALLSPFRouters) or 224.0.0.6 (AllIDRouters)

Link state advertisement

OSPF packets are not carried as UDP payload! OSPF has its own IP protocol number: 89

TTL: set to 1 (in most cases)

Destination IP: neighbor’s IP address or 224.0.0.5 (ALLSPFRouters) or 224.0.0.6 (AllIDRouters)
OSPF Common header

- **Version**: 2: current version is OSPF V2

**Message types:**
1: Hello (tests reachability)
2: Database description
3: Link Status request
4: Link state update
5: Link state acknowledgement

**Standard IP checksum** taken over entire packet

**Authentication passwd = 1**: 64 cleartext password
**Authentication passwd = 2**: 0x0000 (16 bits)
KeyID (8 bits)
Length of MD5 checksum (8 bits)
Nondecreasing sequence number (32 bits)

**Prevents replay attacks**
OSPF LSA Format

LSAs
- Type 1: cost of links between routers
- Type 2: networks to which the router connects
- Others: hierarchical routing
Type 1 LSA

- Link state ID and Advertising router are the same, 32-bit router ID
- Link ID: router ID at the other end of the link
- Link Data: identify parallel links
- Metric: cost of the link
- Type: types of the link e.g., point-to-point
Open question

- How to set link metrics?

- Design choice 1: all to 1

- Design choice 2: based on load
  - Problems?

- In practice: static
Hierarchical OSPF
Hierarchical OSPF

- Two-level hierarchy: local area, backbone.
  - Link-state advertisements only in area
  - Each node has detailed area topology; only know direction (shortest path) to nets in other areas.

- **Area border routers**: “summarize” distances to nets in own area, advertise to other Area Border routers.

- **Backbone routers**: run OSPF routing limited to backbone.
Scalability and Optimal Routing

• A frequent tradeoff in network design
• Hierarchy introduces information hiding
OSPF summary

• A link-state routing protocol
• Each node has a map of the network and uses Dijkstra to compute shortest paths
• Nodes use reliable flooding to keep an identical copy of the network map
Summary

• Routing information protocol (RIP)
• Link-state routing
  – Algorithm
  – Protocol: Open shortest path first (OSPF)