Dynamic Routing Protocols I
RIP

- Autonomous Systems
- Distance Vector Routing and RIP

Routing

- **Recall**: There are two parts to routing IP packets:
  1. How to pass a packet from an input interface to the output interface of a router (packet forwarding)?
  2. How to find and setup a route?

- We already discussed the packet forwarding part

- There are two approaches for calculating the routing tables:
  - Static Routing
  - **Dynamic Routing**: Routes are calculated by a routing protocol
Autonomous Systems

- An **autonomous system** is a region of the Internet that is administered by a single entity.
- Examples of autonomous regions are:
  - UVA’s campus network
  - MCI’s backbone network
  - Regional Internet Service Provider

- Routing is done differently within an autonomous system (**intradomain routing**) and between autonomous system (**interdomain routing**).
Interdomain and Intradomain Routing

**Intradomain Routing**
- Routing within an AS
- Ignores the Internet outside the AS
- Protocols for Intradomain routing are also called Interior Gateway Protocols or IGP’s.
- Popular protocols are
  - RIP (simple, old)
  - OSPF (better)

**Interdomain Routing**
- Routing between AS’s
- Assumes that the Internet consists of a collection of interconnected AS’s
- Normally, there is one dedicated router in each AS that handles interdomain traffic.
- Protocols for interdomain routing are also called Exterior Gateway Protocols or EGP’s.
- Routing protocols:
  - EGP
  - BGP (more recent)
Components of a Routing Algorithm

- A procedure for sending and receiving reachability information about network to other routers
- A procedure for calculating optimal routes
  - Routes are calculated using a shortest path algorithm:
    - **Goal**: Given a network were each link is assigned a cost. Find the path with the least cost between two networks with minimum cost.
- A procedures for reacting to and advertising topology changes

Approaches to Shortest Path Routing

- There are two basic routing algorithms found on the Internet.
  1. **Distance Vector Routing**
     - Each node knows the distance (=cost) to its directly connected neighbors
     - A node sends periodically a list of routing updates to its neighbors.
     - If all nodes update their distances, the routing tables eventually converge
     - New nodes advertise themselves to their neighbors
  2. **Link State Routing**
     - Each node knows the distance to its neighbors
     - The distance information (=link state) is broadcast to all nodes in the network
     - Each node calculates the routing tables independently
Routing Algorithms in the Internet

Distance Vector
- Routing Information Protocol (RIP)
- Gateway-to-Gateway Protocol (GGP)
- Exterior Gateway Protocol (EGP)
- Interior Gateway Routing Protocol (IGRP)

Link State
- Intermediate System - Intermediate System (IS-IS)
- Open Shortest Path First (OSPF)

Dynamic IP Routing Protocols

- In Unix systems, the dynamic setting of routing tables is done by the routed or gated daemons
- The routing daemons execute the following intradomain and interdomain routing protocols

<table>
<thead>
<tr>
<th>Daemon</th>
<th>Hello</th>
<th>RIP</th>
<th>OSPF</th>
<th>EGP</th>
<th>BGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>routed</td>
<td></td>
<td>V1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gated</td>
<td>Yes</td>
<td>V1</td>
<td>V2</td>
<td>Yes</td>
<td>V2, V3</td>
</tr>
<tr>
<td>(Version 3)</td>
<td></td>
<td>V2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Remark: CS 457 vs. CS551

- **In CS457**, networks were represented as a graph, where:
  - each link has a cost
  - destination are nodes

- **In CS551**, we will use a different interpretation of a network graph:
  - nodes are connected by networks
    - network can be a link or a LAN
    - network has a cost
  - networks are destinations
  - \( \text{Net}(v,w) \) is an IP address of a network

- For ease of notation, we will replace the clouds between nodes by simple links.

---

### Distance Vector Algorithm: Routing Table

<table>
<thead>
<tr>
<th>Dest</th>
<th>via (next hop)</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net</td>
<td>n</td>
<td>( D(v,\text{Net}) )</td>
</tr>
</tbody>
</table>
Distance Vector Algorithm: Messages

RoutingTable of node \(v\)

<table>
<thead>
<tr>
<th>Dest</th>
<th>via (next hop)</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net</td>
<td>n</td>
<td>(D(v,Net))</td>
</tr>
</tbody>
</table>

- Nodes send messages to their neighbors which contain routing table entries

\[ [Net, D(v,Net)] \]

- A message has the format: \([Net, D(v,Net)]\) means "My cost to go to Net is \(D(v, d)\)"

Distance Vector Algorithm: Sending Updates

RoutingTable of node \(v\)

<table>
<thead>
<tr>
<th>Dest</th>
<th>via (next hop)</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net_1</td>
<td>m</td>
<td>(D(v,Net_1))</td>
</tr>
<tr>
<td>Net_2</td>
<td>n</td>
<td>(D(v,Net_2))</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net_w</td>
<td>w</td>
<td>(D(v,Net_w))</td>
</tr>
</tbody>
</table>

Periodically, each node \(v\) sends the content of its routing table to its neighbors:

\[ [Net_1, D(v,Net_1)] \]
\[ [Net_2, D(v,Net_2)] \]
\[ [Net_w, D(v,Net_w)] \]

© Jörg Liebeherr, 2000
Initiating Routing Table I

- Suppose a new node v becomes active.
- The cost to access directly connected networks is zero:
  - $D(v, Net(v,m)) = 0$
  - $D(v, Net(v,w)) = 0$
  - $D(v, Net(v,n)) = 0$

Initiating Routing Table II

- New node v sends the routing table entry to all its neighbors:

  ![Diagram showing the routing table entry sent to neighbors](image-url)
### Initiating Routing Table III

- Node v receives the routing tables from other nodes and builds up its routing table

\[
[\text{Net}_v, D(m, \text{Net}_v)] \rightarrow \quad \text{[Net}_v, D(w, \text{Net}_v)]
\]

\[
[\text{Net}_m, D(m, \text{Net}_m)] \rightarrow \quad \text{[Net}_m, D(w, \text{Net}_m)]
\]

\[
\text{[Net}_n, D(n, \text{Net}_n)]
\]

\[
\text{[Net}_w, D(n, \text{Net}_w)]
\]

### Updating Routing Tables I

- Suppose node v receives a message from node m: \([\text{Net}, D(m, \text{Net})]\)

Node v updates its routing table and sends out further messages if the message reduces the cost of a route:

\[
\text{if } (D(m, \text{Net}) + c(v,m) < D(v,\text{Net})) \{ \\
D^{\text{new}}(v,\text{Net}) := D(m,\text{Net}) + c(v,m); \\
\text{Update routing table;} \\
\text{send message [Net, } D^{\text{new}}(v,\text{Net}) \text{] to all neighbors}
\}
Updating Routing Tables II

- Before receiving the message:

\[ [\text{Net}, \text{D}(m, \text{Net})] \rightarrow \]

\[ \text{c}(v, m) \]

\[ \text{Net}(v, m) \]

\[ \text{n} \]

\[ v \]

\[ w \]

- Suppose \( \text{D}(m, \text{Net}) + \text{c}(v, m) < \text{D}(v, \text{Net}) \):

\[ [\text{Net}, \text{D}^{\text{new}}(v, \text{Net})] \rightarrow \]

\[ \text{c}(v, m) \]

\[ \text{Net}(v, m) \]

\[ \text{n} \]

\[ v \]

\[ w \]

\[ [\text{Net}, \text{D}^{\text{new}}(v, \text{Net})] \]

Example

Assume:
- link cost is 1, i.e., \( \text{c}(v, w) = 1 \)
- all updates, updates occur simultaneously
- Initially, each router only knows the cost of connected interfaces

<table>
<thead>
<tr>
<th>Net</th>
<th>via</th>
<th>1st Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.1.0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>10.0.2.0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>10.0.1.0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>10.0.2.0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>10.0.3.0</td>
<td>10.0.2.2</td>
<td>1</td>
</tr>
<tr>
<td>10.0.1.0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>10.0.2.0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>10.0.3.0</td>
<td>10.0.2.2</td>
<td>1</td>
</tr>
<tr>
<td>10.0.4.0</td>
<td>10.0.2.2</td>
<td>2</td>
</tr>
<tr>
<td>10.0.1.0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>10.0.2.0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>10.0.3.0</td>
<td>10.0.2.2</td>
<td>1</td>
</tr>
<tr>
<td>10.0.4.0</td>
<td>10.0.2.2</td>
<td>2</td>
</tr>
<tr>
<td>10.0.1.0</td>
<td>10.0.2.1</td>
<td>1</td>
</tr>
<tr>
<td>10.0.2.0</td>
<td>10.0.3.1</td>
<td>1</td>
</tr>
<tr>
<td>10.0.3.0</td>
<td>10.0.4.1</td>
<td>1</td>
</tr>
<tr>
<td>10.0.4.0</td>
<td>10.0.5.1</td>
<td>1</td>
</tr>
<tr>
<td>10.0.1.0</td>
<td>10.0.2.1</td>
<td>1</td>
</tr>
<tr>
<td>10.0.2.0</td>
<td>10.0.3.1</td>
<td>1</td>
</tr>
<tr>
<td>10.0.3.0</td>
<td>10.0.4.1</td>
<td>1</td>
</tr>
<tr>
<td>10.0.4.0</td>
<td>10.0.5.1</td>
<td>1</td>
</tr>
</tbody>
</table>
Characteristics of Distance Vector Routing

- **Periodic Updates**: Updates to the routing tables are sent at the end of a certain time period. A typical value is 90 seconds.

- **Triggered Updates**: If a metric changes on a link, a router immediately sends out an update without waiting for the end of the update period.

- **Full Routing Table Update**: Most distance vector routing protocols send their neighbors the entire routing table (not only entries which change).

- **Route invalidation timers**: Routing table entries are invalid if they are not refreshed. A typical value is to invalidate an entry if no update is received after 3-6 update periods.
The Count-to-Infinity Problem

A's Routing Table

<table>
<thead>
<tr>
<th>to</th>
<th>via (next hop)</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
<td>2</td>
</tr>
</tbody>
</table>

B's Routing Table

<table>
<thead>
<tr>
<th>to</th>
<th>via (next hop)</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
<td>1</td>
</tr>
</tbody>
</table>

now link B-C goes down

<table>
<thead>
<tr>
<th>to</th>
<th>via (next hop)</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>to</th>
<th>via (next hop)</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>-</td>
<td>∞</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>to</th>
<th>via (next hop)</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>to</th>
<th>via (next hop)</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>-</td>
<td>∞</td>
</tr>
</tbody>
</table>

Count-to-Infinity

- The reason for the count-to-infinity problem is that each node only has a “next-hop-view”
- For example, in the first step, A did not realize that its route (with cost 2) to C went through node B
- How can the Count-to-Infinity problem be solved?
Count-to-Infinity

• The reason for the count-to-infinity problem is that each node only has a “next-hop-view”
• For example, in the first step, A did not realize that its route (with cost 2) to C went through node B

• How can the Count-to-Infinity problem be solved?
  • Solution 1: Always advertise the entire path in an update message (Path vectors)
    – If routing tables are large, the routing messages require substantial bandwidth
    – BGP uses this solution

Count-to-Infinity

• The reason for the count-to-infinity problem is that each node only has a “next-hop-view”
• For example, in the first step, A did not realize that its route (with cost 2) to C went through node B

• How can the Count-to-Infinity problem be solved?
  • Solution 2: Never advertise the cost to a neighbor if this neighbor is the next hop on the current path (Split Horizon)
    – Example: A would not send the first routing update to B, since B is the next hop on A’s current route to C
    – Split Horizon does not solve count-to-infinity in all cases!
RIP - Routing Information Protocol

- A simple intradomain 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 precursor 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

<table>
<thead>
<tr>
<th>Command</th>
<th>Version</th>
<th>Set to 00...0</th>
</tr>
</thead>
<tbody>
<tr>
<td>address family</td>
<td>Set to 00.00</td>
<td></td>
</tr>
<tr>
<td>32-bit address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unused (Set to 00...0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unused (Set to 00...0)</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>

One RIP message can have up to 25 route entries

RIPv2

- RIPv2 is an extends RIPv1:
  - Subnet masks are carried in the route information
  - Authentication of routing messages
  - Route information carries next-hop address
  - Exploites IP multicasting

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

**IP header** | **UDP header** | **RIPv2 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)**: 

#### Address of destination

**Up to 24 more routes (each 20 bytes)**

**One RIP message can have up to 25 route entries**

---

### RIPv2 Packet Format

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

#### Command

- **Version**: Set to 00.00
- **Address family**: 
- **Route tag**: 
- **IP address**: 
- **Subnet Mask**: 
- **Next-Hop IP address**: 
- **Metric (1-16)**: 

#### One route entry

**Up to 24 more routes (each 20 bytes)**

**Used to carry information from other routing protocols (e.g., autonomous system number)**

**Identifies a better next-hop address on the same subnet than the advertising router, if one exists (otherwise 0…0)**
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 entire routing table.
RIP Security

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

![RIPv2 Message Diagram]

- Command
- Version
- Set to 00.00
- Authentication Type
- Password (Bytes 0 - 3)
- Password (Bytes 4 - 7)
- Password (Bytes 8 - 11)
- Password (Bytes 12 - 15)
- Up to 24 more routes (each 20 bytes)

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