CS118 Discussion
Week 7
Outline

• Hints for project 2

• Lecture review: routing
About Course Project 2

• Please implement **byte-stream** reliable data transfer
  
  • Cwnd is in unit of bytes, not packets

• How to realize timeout?
  
  • **Option 1**: select( ) with countdown timer
  
  • **Option 2**: alarm( ) + signal ( )

• Should we implement RTT estimation?
  
  • No, if you do not plan to realize congestion control algorithms
Routing

- Intra-domain routing V.S. inter-domain routing
  - Performance V.S. policy
  - Scalability: hierarchical routing
- Distance-vector routing V.S. link-state routing
  - Fully-distributed algorithm V.S. decentralized algorithm
- Unicast V.S. multicast
Distance-vector Routing

• Foundation: Bellman-Ford algorithm

• Count-to-infinity problem
  • When will this problem happen?

  • Split Horizon: if B learns about D from C, B should not tell C that B can reach D

  • Poison Reverse: if A routes through C to reach D, A tells C that A’s distance to D is infinite

• Enhancement: BGP’s path-vector routing
  • All problems solved?
Link-state Routing

- Foundation: Dijkstra algorithm

- How to calculate A’s FIB?

- What if link failure happens?
  - Is forwarding loop completely avoided?
Multicast

- Reverse Path Forwarding
  - Why does it avoid loops?
  - For each node, how to forward A’s multicast packets?
MPLS: Multi-Protocol Label Switching
Routing VS. Switching

- Circuit-switch network: connection should be established before forwarding the data
  - At each hop, the circuit path is marked as a label
  - Data forwarding is based on label: $O(1)$ complexity
  - Vulnerable to link/node failures

- Packet-switched network: connectionless, packets are forwarded based on IP header
  - Longest prefix matching: $O(N)$ complexity
  - Robust to link/node failures

- Can we take advantage of both, while preventing any vulnerabilities?
Multi-Protocol Label Switching

- Idea: **switching technique** into **connectionless** network
- In IP routing table, each entry is associated with a label
- Neighboring routers exchange labels, and forms an index of next hop’s forwarding table
- When forwarding the packet, lookup the index only
  - Only the first hop performs longest prefix matching
Exchanging labels Between Routers

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.1</td>
<td>0</td>
</tr>
<tr>
<td>10.3.3</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.1</td>
<td>1</td>
</tr>
<tr>
<td>10.3.3</td>
<td>0</td>
</tr>
</tbody>
</table>

R1 0

R2

R3 10.1.1/24

R4 10.3.3/24
Exchanging labels Between Routers

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Next</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.1</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>10.3.3</td>
<td>0</td>
<td>16</td>
</tr>
</tbody>
</table>

Allocate label for each entry

Store remote labels

Exchange label between neighbors
Forwarding Packets

10.1.1.1

R1 0

R2

1 0

R3

10.1.1/24

R4

10.3.3/24

Prefix  Next  Label
10.1.1  0    15
10.3.3  0    16

Label  Prefix  Next
15     10.1.1  1
16     10.3.3  0

Longest Prefix Matching
Forwarding Packets

Label-based Switching

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Next</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.1</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>10.3.3</td>
<td>0</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Label</th>
<th>Prefix</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>10.1.1</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>10.3.3</td>
<td>0</td>
</tr>
</tbody>
</table>

10.1.1.1 (15)
Other Benefits

• In IP table, the IP addresses are aggregated based on prefixes

• In MPLS, the IP addresses can be aggregated in more flexible ways
  • How: assign same label for IP addresses in the same category
  • Benefit: better management (e.g., offer different levels of performance using different labels)
Lecture 13:
Distance-vector Routing

HW 3 due now
Lecture 13 Overview

- **Distance vector**
  - Assume each router knows its own address and cost to reach each of its directly connected neighbors

- **Bellman-Ford algorithm**
  - Distributed route computation using only neighbor’s info

- **Mitigating loops**
  - Split horizon and posion reverse
Bellman-Ford Algorithm

- Define distances at each node $X$
  - $d_x(y) =$ cost of least-cost path from $X$ to $Y$

- Update distances based on neighbors
  - $d_x(y) = \min \{c(x,v) + d_v(y)\}$ over all neighbors $V$

\[ d_u(z) = \min \{c(u,v) + d_v(z), c(u,w) + d_w(z)\} \]
Distance Vector Algorithm

Iterative, asynchronous: each local iteration caused by:
- Local link cost change
- Distance vector update message from neighbor

Distributed:
- Each node notifies neighbors only when its DV changes
- Neighbors then notify their neighbors if necessary

Each node:
- \textit{wait} for (change in local link cost or message from neighbor)
- \textit{recompute} estimates
- if distance to any destination has changed, \textit{notify} neighbors
Step-by-Step

- \( c(x,v) \) = cost for direct link from \( x \) to \( v \)
  - Node \( x \) maintains costs of direct links \( c(x,v) \)

- \( D_x(y) \) = estimate of least cost from \( x \) to \( y \)
  - Node \( x \) maintains distance vector \( D_x = [D_x(y) \colon y \in N] \)

- Node \( x \) maintains its neighbors’ distance vectors
  - For each neighbor \( v \), \( x \) maintains \( D_v = [D_v(y) \colon y \in N] \)

- Each node \( v \) periodically sends \( D_v \) to its neighbors
  - And neighbors update their own distance vectors
  - \( D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \) for each node \( y \in N \)
Example: Initial State

![Diagram of network nodes A, B, C, D, E with distances labeled]

<table>
<thead>
<tr>
<th>Info at node</th>
<th>Distance to Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 7 ∞ ∞ 1</td>
</tr>
<tr>
<td>B</td>
<td>7 0 1 ∞ 8</td>
</tr>
<tr>
<td>C</td>
<td>∞ 1 0 2 ∞</td>
</tr>
<tr>
<td>D</td>
<td>∞ ∞ 2 0 2</td>
</tr>
<tr>
<td>E</td>
<td>1 8 ∞ 2 0</td>
</tr>
</tbody>
</table>
**D sends vector to E**

- **I’m 2 from C, 0 from D and 2 from E**

<table>
<thead>
<tr>
<th>Info at node</th>
<th>Distance to Node</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
</tr>
</tbody>
</table>

- **D is 2 away, 2+2< ∞, so best path to C is 4**
**B sends vector to A**

I’m 7 from A, 0 from B, 1 from C & 8 from E

B is 7 away, 1+7<∞ so best path to C is 8

<table>
<thead>
<tr>
<th>Info at node</th>
<th>Distance to Node</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
</tr>
</tbody>
</table>
E sends vector to A

E is 1 away, 4+1<8
so C is 5 away, 1+2<
∞ so D is 3 away

I’m 1 from A, 8 from B, 4 from C, 2 from D & 0 from E

CSE 123 – Lecture 13: Distance-vector Routing
...until Convergence

CSE 123 – Lecture 13: Distance-vector Routing

<table>
<thead>
<tr>
<th>Info at node</th>
<th>Distance to Node</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
</tr>
</tbody>
</table>
Node B’s distance vectors

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>7</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
</tr>
<tr>
<td>E</td>
<td>8</td>
</tr>
</tbody>
</table>

CSE 123 – Lecture 13: Distance-vector Routing
Handling Link Failure

- A marks distance to E as $\infty$, and tells B.
- E marks distance to A as $\infty$, and tells B and D.
- B and D recompute routes and tell C, E and E.
- etc… until converge.

**Info at node**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>7</td>
<td>8</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>12</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Counting to Infinity

Distance to C

CSE 123 – Lecture 13: Distance-vector Routing
Why so High?

- Updates don’t contain enough information
- Can’t totally order bad news above good news
- $B$ accepts $A$’s path to $C$ that is *implicitly* through $B$!
- Aside: this also causes delays in convergence even when it doesn’t count to infinity
Mitigation Strategies

- **Hold downs**
  - As metric increases, delay propagating information
  - Limitation: Delays convergence

- **Loop avoidance**
  - Full path information in route advertisement
  - Explicit queries for loops (e.g. DUAL)

- **Split horizon**
  - Never advertise a destination through its next hop
    - A doesn’t advertise C to B
  - **Poison reverse**: Send negative information when advertising a destination through its next hop
    - A advertises C to B with a metric of ∞
    - Limitation: Only works for “loop”s of size 2
If $Z$ routes through $Y$ to get to $X$:

- $Z$ tells $Y$ its (Z’s) distance to $X$ is infinite (so $Y$ won’t route to $X$ via $Z$)

CSE 123 – Lecture 13: Distance-vector Routing
Split Horizon Limitations

- A tells B & C that D is unreachable
- B computes new route through C
  - Tells C that D is unreachable (poison reverse)
  - Tells A it has path of cost 3 (split horizon doesn't apply)
- A computes new route through B
  - A tells C that D is now reachable
- Etc…
Routing Information Protocol

- 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 vs. Distance-vector

Message complexity
- **LS**: with \( n \) nodes, \( E \) links, \( O(nE) \) messages sent
- **DV**: exchange between neighbors only

Speed of Convergence
- **LS**: relatively fast
- **DV**: convergence time varies
  - May be routing loops
  - Count-to-infinity problem

Robustness: what happens if router malfunctions?
- **LS**:
  - Node can advertise incorrect *link* cost
  - Each node computes only its *own* table
- **DV**:
  - Node can advertise incorrect *path* cost
  - Each node’s table used by others (error propagates)
Routing so far...

- Shortest-path routing
  - Metric-based, using link weights
  - Routers share a common view of path “goodness”
- As such, commonly used *inside* an organization
  - RIP and OSPF are mostly used as *intradomain* protocols
- But the Internet is a “network of networks”
  - How to stitch the many networks together?
  - When networks may not have common goals
  - ... and may not want to share information
For next time…

- Read Ch. 4.3.3-4 in P&D
- Keep moving on Project 2
Chapter 13

Routing Protocols (RIP, OSPF, BGP)

- INTERIOR AND EXTERIOR ROUTING
- RIP
- OSPF
- BGP
Introduction

- Packets may pass through several networks on their way to destination
- Each network carries a price tag, or a “metric”
- The metric of a network may be:
  - constant (i.e. each network costs one hop)
  - Service type-dependent (the cost of the network depends on what service the packet needs: e.g. throughput, delay, .. etc.)
  - Policy-dependent: a policy defines what paths should, or should not, be followed.
- The router uses a “routing table” to determine the path
  - Static vs. Dynamic routing tables.
13.1 Interior & Exterior Routing

**Autonomous system:**
a group of networks and routers under authority of a single administrator
Popular routing protocols

- Interior
  - RIP
  - OSPF
- Exterior
  - BGP
13.2 RIP: Routing Information Protocol

- **Distance Vector Routing**
  - Share the most you know about the entire autonomous system
  - Share with all your direct neighbors, and them only
  - Share periodically, e.g. every 30 seconds

<table>
<thead>
<tr>
<th>Destination</th>
<th>Hop Count</th>
<th>Next Hop</th>
<th>Other Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>163.5.0.0</td>
<td>7</td>
<td>172.6.23.4</td>
<td></td>
</tr>
<tr>
<td>197.5.13.0</td>
<td>5</td>
<td>176.3.6.17</td>
<td></td>
</tr>
<tr>
<td>189.45.0.0</td>
<td>4</td>
<td>200.5.1.6</td>
<td></td>
</tr>
</tbody>
</table>
RIP Updating Algorithm

Receive: a response RIP message

1. Add one to the hop count for each advertised destination
2. Repeat for each advertised destination
   - If (destination is not in my routing table)
     - Add the destination to my table
   - Else If (next-hop field is the same)
     - Replace existing entry with the new advertised one
   - Else if (advertised hop-count — after incrementing — is smaller)
     - Replace existing entry with the new advertised one
Example of updating a routing table

Receive: a response RIP message
1. Add one to the hop count for each advertised destination
2. Repeat for each advertised destination
   - If (destination is not in my routing table)
     - Add the destination to my table
   - Else if (next-hop field is the same)
     - Replace existing entry with the new advertised one
   - Else if (advertised hop-count –after incrementing- is smaller)
     - Replace existing entry with the new advertised one

Old routing table

<table>
<thead>
<tr>
<th>Net1</th>
<th>7</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net2</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>Net6</td>
<td>8</td>
<td>F</td>
</tr>
<tr>
<td>Net8</td>
<td>4</td>
<td>E</td>
</tr>
<tr>
<td>Net9</td>
<td>4</td>
<td>F</td>
</tr>
</tbody>
</table>

New routing table

<table>
<thead>
<tr>
<th>Net1</th>
<th>7</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net2</td>
<td>5</td>
<td>C</td>
</tr>
<tr>
<td>Net3</td>
<td>9</td>
<td>C</td>
</tr>
<tr>
<td>Net6</td>
<td>5</td>
<td>C</td>
</tr>
<tr>
<td>Net8</td>
<td>4</td>
<td>E</td>
</tr>
<tr>
<td>Net9</td>
<td>4</td>
<td>F</td>
</tr>
</tbody>
</table>

Net1: No news, do not change
Net2: Same next hop, replace
Net3: A new router, add
Net6: Different next hop, new hop count smaller, replace
Net8: Different next hop, new hop count the same, do not change
Net9: Different next hop, new hop count larger, do not change
Initial routing tables in a small autonomous system

- Configuration File
  - Directly attached networks
  - Hop-count = 1
Final routing tables for the previous autonomous system

- RIP messages are exchanged
- Routing tables are updated
RIP message format

1: Request
2: Response

Address Family Identifier
2: TCP/IP family

1 or 2

Command | Version | Reserved
---------|---------|---------
Family | All 0s |
Network address | All 0s |
Distance |

Hops from advertising router to dest. network

12 Bytes

up to 25 AFIs

256 Bytes


© Adapted for use at JMU by Mohamed Aboutabl, 2003
RIP Request Messages

- Sent by a router when booted, or when an entry times-out
- May request updates for ALL networks, or specific one(s)

<table>
<thead>
<tr>
<th>Com: 1</th>
<th>Version</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Family</td>
<td>All 0s</td>
</tr>
<tr>
<td></td>
<td>Network address</td>
<td>All 0s</td>
</tr>
<tr>
<td></td>
<td>All 0s</td>
<td>All 0s</td>
</tr>
<tr>
<td></td>
<td>All 0s</td>
<td>All 0s</td>
</tr>
<tr>
<td></td>
<td>All 0s</td>
<td>All 0s</td>
</tr>
</tbody>
</table>

- Request for some

<table>
<thead>
<tr>
<th>Com: 1</th>
<th>Version</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Family</td>
<td>All 0s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All 0s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All 0s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All 0s</td>
</tr>
</tbody>
</table>

- Request for all

RIP Response Messages

- Solicited responding to a previous request
- Unsolicited (sent periodically to all neighbors)
Example 1

What is the periodic response sent by router R1? Assume R1 knows about the whole autonomous system.

<table>
<thead>
<tr>
<th>RIP message</th>
<th>2</th>
<th>1</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>All 0s</td>
</tr>
<tr>
<td>144.2.7.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All 0s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>144.2.9.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All 0s</td>
<td>2</td>
<td></td>
<td>All 0s</td>
</tr>
<tr>
<td>144.2.12.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All 0s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RIP Timers

- **Periodic Timer (25 < random < 35):** controls advertising of update messages. There ONE such timer.

- **Expiration Timers:** governs route validity. Reset upon receipt of an update. If it ever expires, destination is considered unreachable.
  - Yet, entry is not removed from table, it continues to be advertised with hop count = 16 (i.e. infinity)

- **Garbage Collection Timers:** Reset to 120sec when a route is invalidated. If it expires, the route entry is completely removed from routing table.
Example 2

A routing table has 20 entries. It does not receive information about five routes for 200 seconds. How many timers are running at this time?

Solution

The timers are listed below:

- Periodic timer: 1
- Expiration timer: $20 - 5 = 15$
- Garbage collection timer: 5
RIP Problems: 1) Slow convergence

- Network topology changes propagate slowly (avg. 15 sec per hop)
- Solution: Limit the diameter of an autonomous system to 15 hops.

Total hop count should be less than 16
RIP Problems: 2) Instability

- Net1 is disconnected from Router A
- Router A updates its hop count to 16
- Router A waits for 30 seconds before sending it advertisement
- Router B advertises Net1 (with hop-count =2) to A before A has a chance to advertise that Net1 is disconnected
- A is fooled and sets its Hop-count to 2+1=3
Remedies for RIP Instability

- **Triggered Update:**
  - Send an immediate update (with hop count = 16) whenever a network becomes unreachable, otherwise send periodic updates.

- **Split Horizons:**
  - Never sent same information back to the interface it came from

![Diagram of RIP messages](image-url)
Remedies for RIP Instability: Poison reverse

- A variation of Split Horizon.
### RIP-v2 Format: Same length as in RIP-v1

<table>
<thead>
<tr>
<th>Command</th>
<th>Version</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family</td>
<td></td>
<td>Route tag</td>
</tr>
<tr>
<td>Network address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subnet mask</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Next-hop address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- RIP version 2 supports CIDR.
- RIP messages are encapsulated in a UDP datagram.
- RIP uses the services of UDP on well-known port 520.
Authentication

- Protect against unauthorized advertisement
- First entry (with family type = FFFF) is used for authentication

\[
\begin{array}{|c|c|c|}
\hline
\text{Command} & \text{Version} & \text{Reserved} \\
\hline
\text{FFFF} & & \text{Authentication type} \\
\hline
\text{Authentication data} \\
16 \text{ bytes} \\
\hline
\end{array}
\]
CS551
External v.s. Internal BGP
Bill Cheng

http://merlot.usc.edu/cs551-f12
EGP vs. IGP

- Exterior vs. Interior
- World vs. me
- Little control vs. complete *administrative control*
- **BGP** (and GGP, Hello, EGP) vs. (RIP, **OSPF**, IS-IS, IGRP, EIGRP)
BGP can be used by R3 and R4 to learn routes.

How do R1 and R2 learn routes? (How does R3 pass on the routes that it has learned to R1 and R2?)

Option 1: Inject routes in IGP (such as OSPF)
- only works for small routing tables

Option 2: Use I-BGP
Why BGP as an IGP?

- I-BGP has mechanisms to forward BGP policy directives across an AS
- Often use I-BGP with *some* other IGP (such as OSPF) that does internal routing
I-BGP

Upstream Provider A

AS100

AS 1

iBGP

eBGP

Upstream Provider B

AS200

AS 2

iBGP

eBGP
E-BGP vs. I-BGP

- E-BGP connects AS’s (external GP)
- I-BGP is intra-AS (internal GP)
- Differences in operation
  - direct vs. indirect connections
  - different failure modes
  - special attributes for internal use
Internal BGP (I-BGP)

- Same message types, attribute types, and state machine as E-BGP

- Different rules about re-advertising prefixes:
  - prefix learned from E-BGP can be advertised to I-BGP neighbor and vice-versa, but
  - prefix learned from one I-BGP neighbor cannot be advertised to another I-BGP neighbor
  - reason: no AS-PATH within the same AS and thus danger of looping
Internal BGP (I-BGP)

- R3 can tell R1 and R2 prefixes from R4
- R3 can tell R4 prefixes from R1 and R2
- R3 cannot tell R2 prefixes from R1
**Internal BGP (I-BGP)**

- R3 can tell R1 and R2 prefixes from R4
- R3 can tell R4 prefixes from R1 and R2
- R3 cannot tell R2 prefixes from R1

R2 can only find these prefixes through a direct connection to R1
Result: I-BGP routers must be *fully connected* (via TCP)!
- contrast with E-BGP sessions that map to physical links
BGP Example

R1 advertises routes within AS1 to R2
R2 advertises routes within AS2 and AS3 to R1
R2 learns AS3 routes from I-BGP with R4
R4 learns AS3 routes from E-BGP with R6
R4 advertises routes within AS2 and AS1 to R6
Link Failures

- Two types of link failures:
  - failure on an E-BGP link
  - failure on an I-BGP Link

- These failures are treated completely different in BGP

- Why?
Failure on an E-BGP Link

- Note that 138.39.1.1 and 138.39.1.2 are on the *same* network
- If the link R1-R2 goes down, then the TCP connection breaks and so does the E-BGP connection; BGP routes are removed
- This is the desired behavior
Failure on an I-BGP Link

If physical link R1-R2 goes down, the 138.39.1.0/30 network becomes unreachable, connection between R1 and R2 is lost.

R1 and R2 should, in theory, still be able to exchange traffic, i.e., the indirect path through R3 should be used.

- given the above configuration, it would not work!
- thus, E-BGP and I-BGP must use different conventions with respect to TCP endpoints

Note: I-BGP often does not go over a physical link
Virtual Interfaces (VIFs, a.k.a. Loop-back Interfaces)

Note that 138.39.128.1 and 138.39.128.5 are on different networks here!

- A VIF is not associated with a physical link or hardware interface
- How do routers learn of VIF addresses?
  - use IGP
Scaling the I-BGP Mesh

Two methods:

- **BGP confederations**
  - scale by adding hierarchy to AS (sub-AS)

- **Route reflectors**
  - scale by adding hierarchical IBGP route forwarding
AS Confederation

- Subdivide a single AS into multiple, internal sub-AS’s to reduce I-BGP mesh size
  - simple hierarchy
  - but only one level

- Still advertises a single AS to external peers
  - internally use *sub-AS’s*
An AS Confederation

- R2 does not see sub-AS 10-14, but sees AS1
Confederations

BGP sessions between sub-AS’s are like regular E-BGP but with some changes:

- *local-pref* attribute remains meaningful within confederation (E-BGP ignores it)
- *next-hop* attribute traverses sub-AS boundaries (assumes single IGP running - everyone has same route to *next-hop*)
- AS-PATH now includes AS-CONFED-SET and AS-CONFED-SEQUENCE to avoid loops
BGP Confederation

AS10

AS20

AS30

AS300
Route Reflectors

**Route Reflector (RR):** router whose BGP implementation allows re-advertisement of routes between I-BGP neighbors
- RR runs modified I-BGP

**Route Reflector Client (RRC):** router that depends on RR to re-advertise its routes to entire AS. It also depends on RR to learn routes from the rest of the network
- RRC runs normal I-BGP
With RR there are 7 I-BGP sessions instead of 21 \((=7 \times 6/2)\)
Rules for Route Reflectors

- Reflectors advertise routes learned from clients into the I-BGP mesh
  - RR1 advertises 138.39.0.0/16 learned from RRC2 into I-BGP
- Reflectors do not re-advertise routes between non-clients
  - RR1 will not re-advertise 128.4.0.0/16 learned from RR3 to RR2