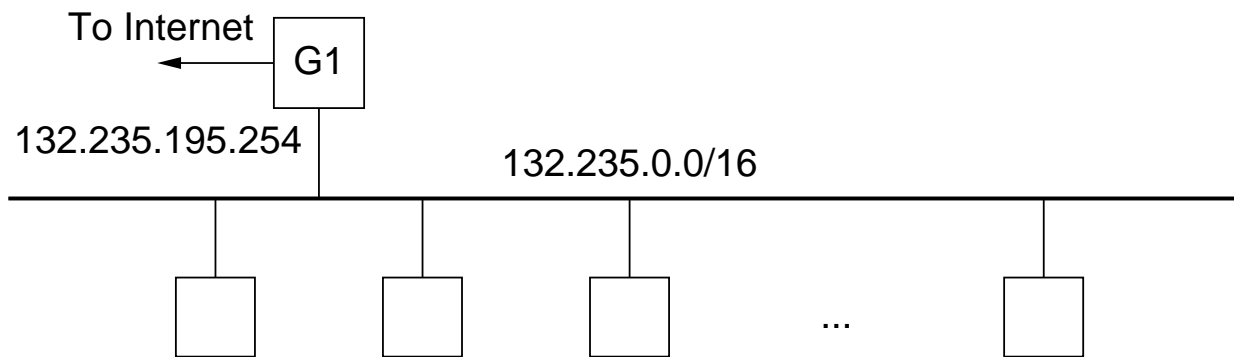


How/When Are IP Routing Tables Built?

- We already understand how routing tables are **used**
 - We've installed entries many times
- The next question is:
 - *How can we automate the process?*
- The process can generally be automated
 - Table initialized at boot time
 - What can we insert automatically?
 - Values inserted/updated by protocols that propagate route information

Hosts Review 1

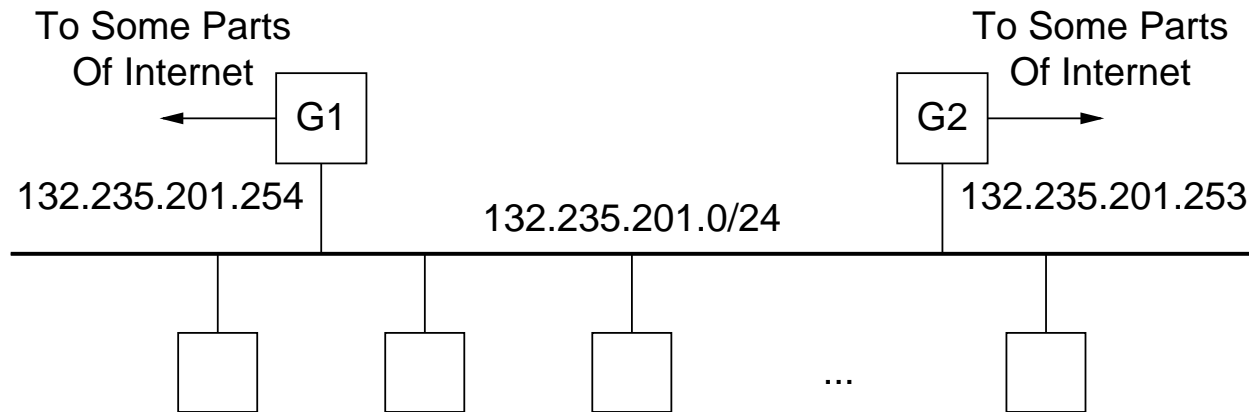
- Remember that hosts can generally be “stupid”
 - Very often, they only need a default route



Destination	Mask	Route
132.235.0.0	ffff0000	direct
default	00000000	132.235.195.254

Hosts Review 2

- Life for the hosts only really gets interesting when there are multiple gateways



Destination	Mask	Route
132.235.201.0	ffffff00	direct
default	00000000	132.235.201.254

Hosts ICMP redirects

- Host *H* generates datagram for 128.10.2.1 (which lies beyond G2)
- Host *H* routes datagram to G1
- G1 forwards the packet to G2
 - Also sends an *ICMP Redirect* back to *H*
- ICMP redirect from G1 updates table
- Recall that the *most specific* route (longest mask) wins

Destination	Mask	Route
132.235.201.0	ffffff00	direct
128.10.0.0	ffff0000	132.235.201.253
default	00000000	132.235.201.254

ICMP Redirect Details

- An important rule is that ICMP redirects can only be sent from a router to a directly-connected machine
 - Why???
- There are 4 kinds of messages:
 - code 0 - redirect datagrams for this net (obsolete)
 - code 1 - redirect datagrams for this host
 - code 2 - redirect datagrams for this type of service and net (obsolete)
 - code 3 - redirect datagrams for this type of service and host
- This isn't a terribly efficient way to populate host routing tables
 - What are the disadvantages?

Summary

- We know how hosts can “survive” failed routers
- We know how hosts can fix routing mistakes
 - Why is this important?
- We can't use this in general
 - Not efficient for hosts
 - Not sufficient in general for routers
 - internets are too complex
- We need an *algorithm!*

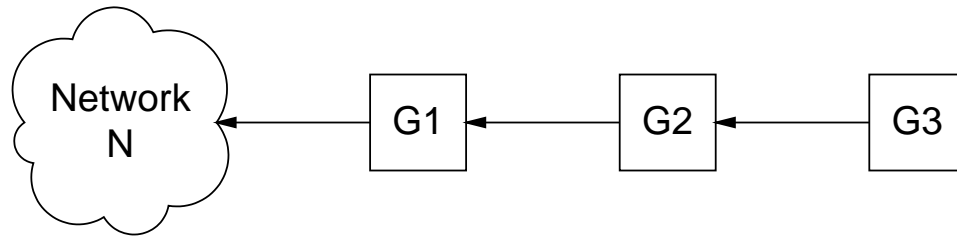
Vector Distance Algorithm

- Each router sends its routing table to all neighboring routers
- Table contains pairs of destination network and distance
- Receiver replaces entries in its table by routes to the sender if routing through the sender is less expensive than the current route
- Receiver propagates new routes next time it sends out an update
- Algorithm has several well-known shortcomings

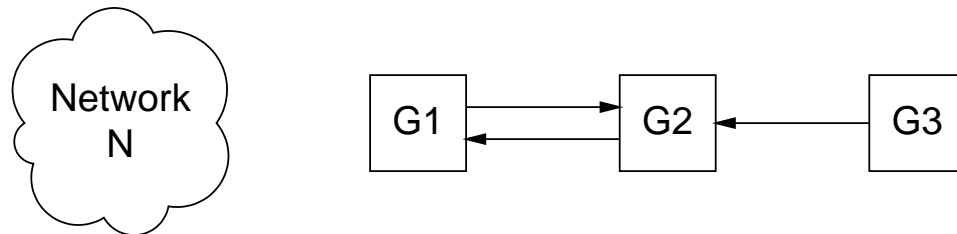
Routing Information Protocol (RIP)

- Vector-distance protocol
- Uses hop count metric for “distance”
- Implemented by 4BSD UNIX program *routed*
- Relies on IP broadcast
- Assumes low-delay local area network
- RIPv1 uses class-based network addresses
- RIPv2 uses CIDR network addresses

Slow Convergence Problem (Count To Infinity)



Gateways with routes to network N



G1 erroneously routes to G2 after failure

RIP2 Update Format

0	8	16	31
Command	Version	RESERVED (MBZ)	
Family OF Net 1		Route Tag for Net 1	
IP Address of Net 1			
Subnet Mask of Net 1			
Next Hop for Net 1			
Distance to Net 1			
Family OF Net 2		Route Tag for Net 2	
IP Address of Net 2			
Subnet Mask of Net 2			
Next Hop for Net 2			
Distance to Net 2			
...			

RIP2 Details

- Updates travel in UDP packets (port 520)
- Includes subnet mask information
- Supports multiple address families
 - Not just IP
- Next hop information helps prevent routing loops
- Route tags are *opaque*
 - Can be used to specify origin of information
- Uses split horizon and poison reverse techniques to solve inconsistencies

Using RIP in Unix

- The server *routed* listens for RIP updates
 - Uses *passive mode*
 - Updates normally sent every 30 seconds
- Routing table is updated for a route if one of the following is true
 - No routing table entry exists and the metric says *reachable*
 - Meaning not infinite (16)
 - The source is the same as the existing next hop
 - Metric might need to be updated
 - Existing entry is more than 90 seconds old and the new route is at least as good
 - The new route is shorter
- All RIP entries not updated in 3 minutes are set to *infinity* and marked for deletion

Solaris Routed

- Basically, just run

```
/usr/sbin/in.routed -q
```

- To see what it's doing:

```
/usr/sbin/in.routed -q -t
```

Configuring RIP on the Cisco Routers

- To enable/disable RIP processing
 - `router rip`
 - `no router rip`
- To specify which networks to use it on
 - `network IP`
- Example

```
router rip
network 128.99.0.0
network 192.31.7.0
```

Other Useful CISCO RIP Commands

- Commands that work under version 11.x

```
debug ip rip events
```

```
debug ip rip
```

```
show ip route
```

```
show ip protocols
```

References

- Comer Volume I - RIP
- http://www.cisco.com/warp/public/105/rip_index.shtml

Routing Algorithms Summary

- We've already talked about *Vector Distance* algorithms
 - Routers exchange tables of
 $nexthop(vector), metric(distance)$
- Vector distance algorithms have known problems
 - Information propogates slowly
 - Slow convergence problems
 - Routing loops
 - Large update messages
- The problems can “usually” be solved with various heuristics
 - Split horizon
 - Poison reverse
 - Hold down

Link State Algorithms

- The other commonly-used routing algorithm is the *link state* algorithm
 - Solves the problems of vector distance algorithms
- Sometimes known as *Shortest Path First* (SPF)
 - After Dijkstra's graph algorithm of the same name
- Assumes that each router already has full knowledge of network connectivity
- The algorithm only sends information about failed links

Link State Algorithm Concept

- Imagine a ring of 7 routers, labelled *A* through *G*
- Start with the *connectivity matrix*:

	A	B	C	D	E	F	G
A	-	1					1
B	1	-	1				
C		1	-	1			
D			1	-	1		
E				1	-	1	
F					1	-	1
G	1					1	-

Link State Algorithm Concept Step 1

- As a first conceptual step, propagate reachability to neighbors

	A	B	C	D	E	F	G
A	-	1	2			2	1
B	1	-	1	2			2
C	2	1	-	1	2		
D		2	1	-	1	2	
E			2	1	-	1	2
F	2			2	1	-	1
G	1	2			2	1	-

Link State Algorithm Concept

Step 2

- Continue until you know how to get everywhere

	A	B	C	D	E	F	G
A	-	1	2	3	3	2	1
B	1	-	1	2	3	3	2
C	2	1	-	1	2	3	3
D	3	2	1	-	1	2	3
E	3	3	2	1	-	1	2
F	2	3	3	2	1	-	1
G	1	2	3	3	2	1	-

Link State Algorithm Summary

- We talked about the general case
 - *All Source Shortest Path*
- Of course, each router only needs to compute its own row of the table
 - *Single Source Shortest Path*
 - Much faster to compute

SPF Advantages

- Guaranteed to converge
- Easier to debug
- Updates are small
- Scales to large system more easily

OSPF

- *Open Shortest Path First (OSPF)*
- The most common example of a routing protocol that uses the Link State Algorithm
- Like RIP, it's an *Interior Gateway Protocol*
 - Much more sophisticated
 - Very complex
 - Allows policy-based routing

Summary to Date

- We've looked at *Interior Gateway Protocols*
 - Used **inside** an organization
- We studied the IGP RIP
 - Fairly old
 - Quite usable on small networks
- We've also looked at the *Link State Algorithm*
 - OSPF is a common example
- Between organizations, we need something more powerful
 - Called an *Exterior Gateway Protocol*

The New Problem

- The Internet is no longer a centralized entity
- No single group is in charge
- Everything is based on cooperation
- Global consistency of routes isn't possible
- There are just too many routers and network now
 - Routing tables would get too big

Autonomous System

- An *Autonomous System* is an organization that makes its own decisions
 - A University (Ohio University)
 - An ISP (Frognets)
 - A backbone provider (Sprint)
- Each uses whatever IGP it chooses for internal connectivity
- Can choose its own metrics

BGP

- The *Border Gateway Protocol* (BGP) is designed to route traffic between autonomous systems
 - Neither pure vector distance nor pure link state
- Provides *reachability* information, not *routing* information
 - From which a particular machine can choose a route
 - Can't tell the cost of traversing a particular autonomous system

BGP Peering

- Two autonomous systems establish a relationship by *peering*
 - Agree to talk and exchange information
- Periodically exchange *Keepalive* messages
 - So a machine can verify that its peer is still alive

BGP Path Attributes

- BGP Updates consist of a list of destinations
 - Include next hop information
 - Also contain a list of *Path Attributes*
 - Origin of the path information
 - Source of the information
 - List of autonomous systems on path to destination
 - Preference used within an autonomous system
 - Indication that routes have been aggregated
 - ID of autonomous system that aggregated the routes

BGP Details

- Travels over TCP for extra reliability
- Updates are quite large
 - Highly compressed
- Has provisions for security
 - To prevent “bad guys” from inserting bad routing information
- There are no distance metrics, because they can't be directly compared
 - Each autonomous system uses its own metrics
 - Can only use the path information to determine which next hop to use

References

- Comer Volume I - Chapter 15
- RFC 1771 and followups (40 at present)