Chapter 27 (and a little of 13)
Routing

Routing Scalability

- Routing to millions of hosts can cause scalability problems.
- We must be able to route to every possible destination (or do we?).
- We need to forward packets quickly
- We need to exchange routing information without putting a lot of load on the network.

Example WAN Topology

- Routers connect to Routers to form the interconnect
- Hosts (sometimes LANs) connect to edge Routers

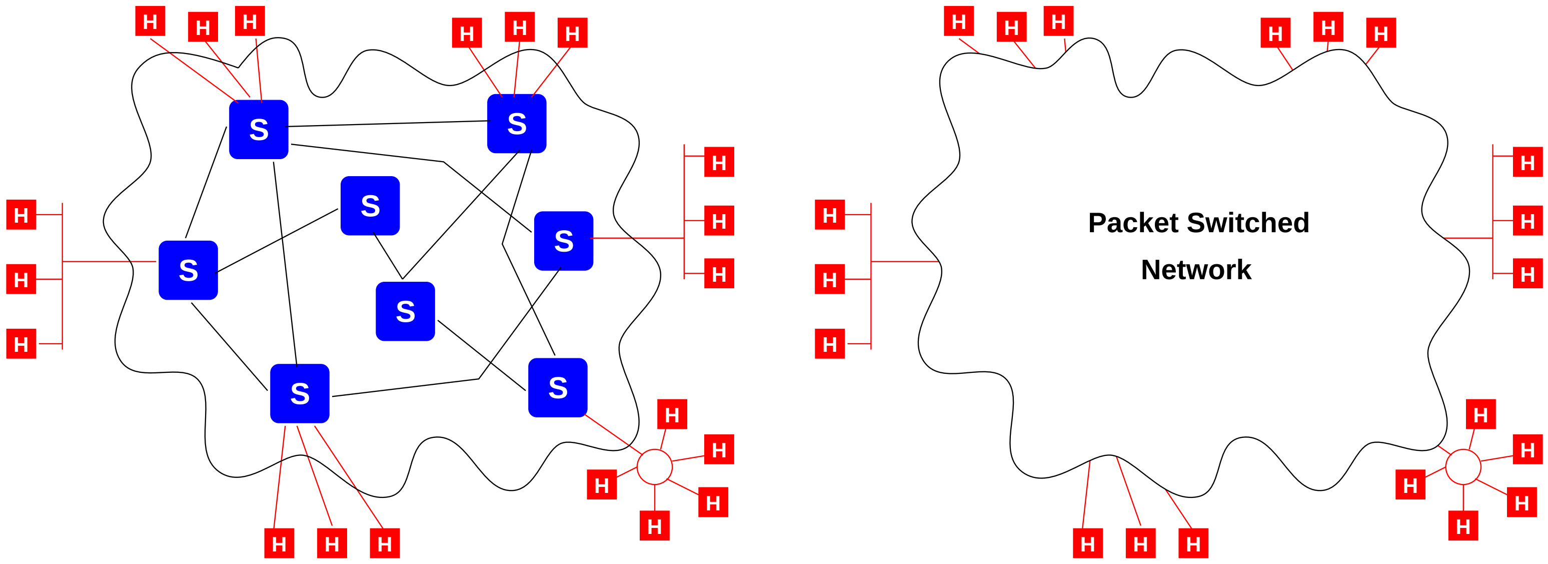


Figure 1: (a) Detailed WAN topology, (b) Logical WAN topology

Flat Addressing

- Flat Addressing refers to the fact that each machine has a unique id as its address
- Any id can be assigned to any machine as long as it is unique.

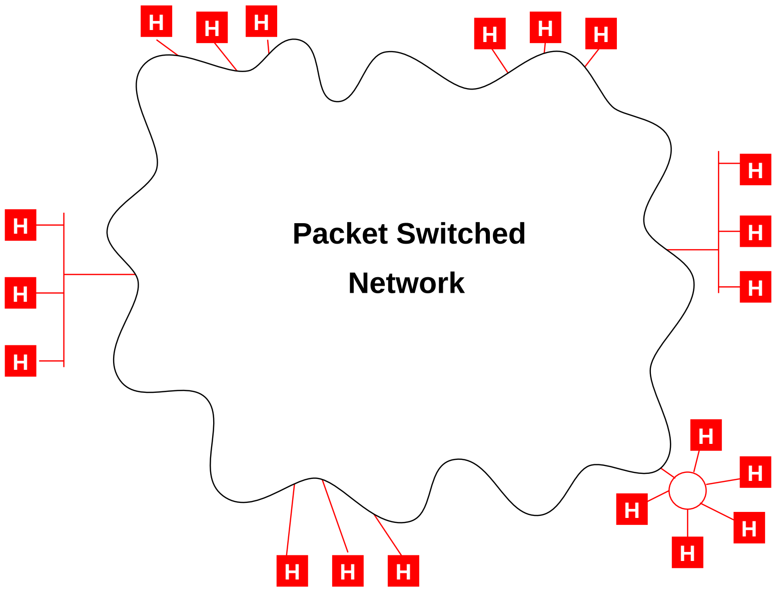


Figure 2: Each machine need a unique id

Hierarchical Addressing

- Consider the following topology

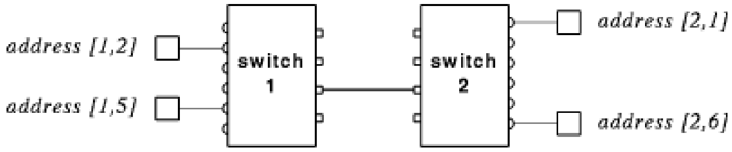


Figure 3: Hierarchical Addressing

- Each host can be uniquely addressed using a hierarchical address consisting of:

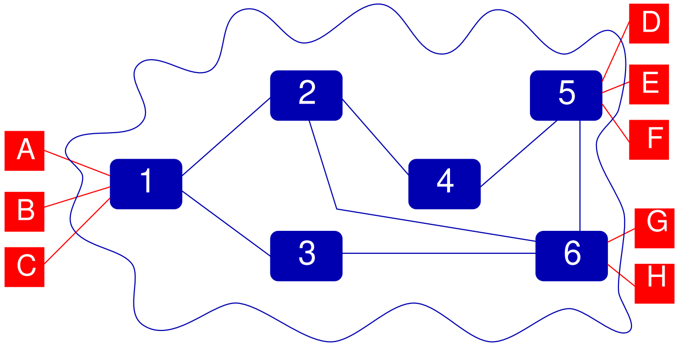
- Hierarchical addresses are very helpful to routing (as we will soon see).

Routing Tables

- The Routing Table stored in each switch defines the route (or path) that a packet should take to get to the destination
- What goes in a routing table depends on several factors including the addressing mechanism (e.g., are addresses hierarchical?), the routing protocols (e.g., do we know complete paths?), the forwarding mechanism (e.g., how are lookups done?), etc.
- The size of the routing tables is an important issue both because of the amount of memory needed to store it, but also because of the time required to lookup a route in the table.
- We will look at several ways to store routing information.

(Worst Case) Routing Table Size

- Routing info maintained at switches and hosts .
- Each node records the (1) complete route to (2) every destination

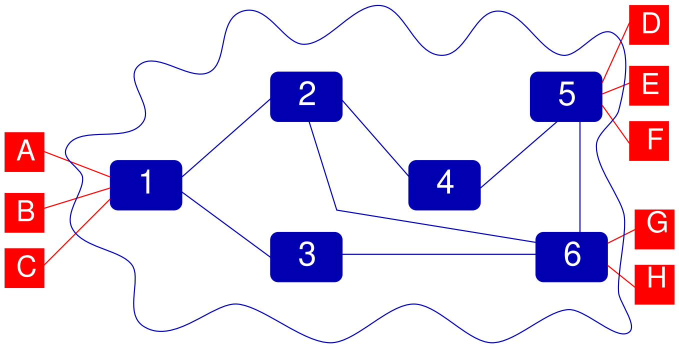


Host A	
Dst	Route

Switch 2	
Dst	Route

Nexthop Routing

- Nexthop routing reduces table size by forgetting all but the next hop
- For example, consider the following network



Host A	
Dst	Next Hop

Switch 2	
Dst	Next Hop

Hierarchical Routing

- Hierarchical Routing reduces the number of entries in the table
- For example, consider the following network

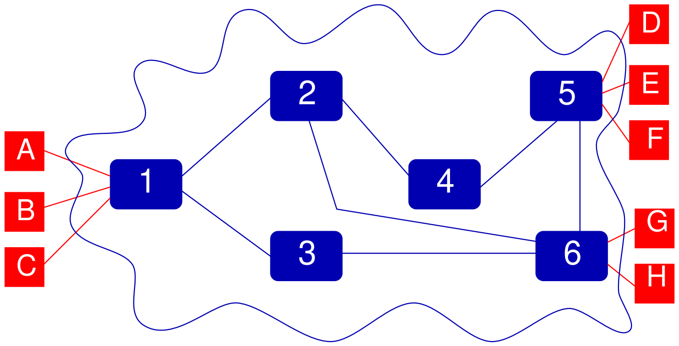


Figure 4: An Example Network Topology

Host A	
Dst Switch	Next Hop

Switch 2	
Dst Switch	Next Hop

Default Routes

- Default Routes further reduce table size
- Particularly useful for

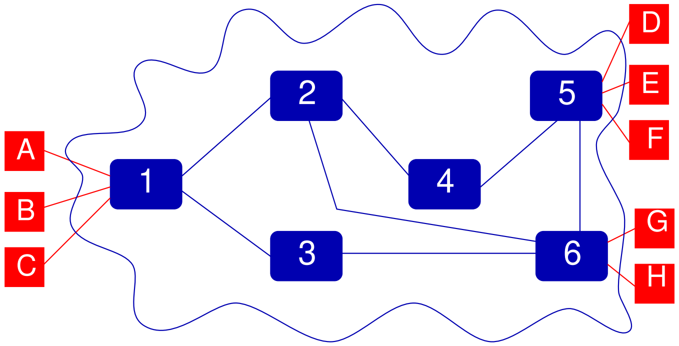


Figure 5: An Example Network Topology

Host A	
Dst Switch	Next Hop

Switch 2	
Dst Switch	Next Hop

Src Dependent vs. Src Independent

- Using Source Dependent routing, the next hop to the destination depends on the source of the packet
- Source Dependent routing allow packets from different machines to travel different routes

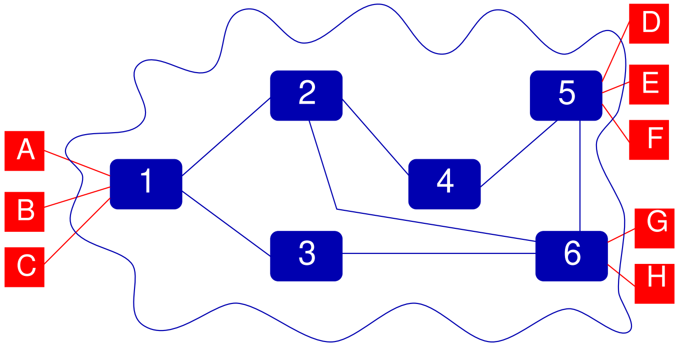


Figure 6: An Example Network Topology

- Using Source Independent routing, the next hop to the destination does not depend on the source of the packet
- Source Independent routing helps reduce table size

What if the Topology Changes?

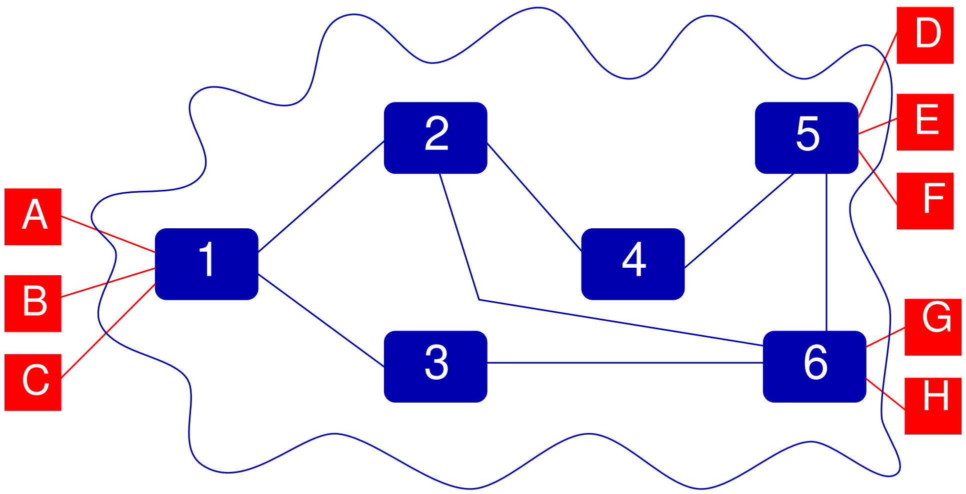


Figure 7: An Example Network Topology

- If each node records the complete path to every destination,

- With nexthop routing,

Building Routing Tables

- How to enter information into routing tables:
  - Manual entry - initialization file
  - Automatic - computed (via routing protocol)
- How to compute routing table information:
  - Static routing -
  - Dynamic routing -
- Static is simpler, but doesn't accommodate changes to network topology
- Dynamic requires additional (complex) protocol(s), but can work around network failures
- The above assumes we already know or can discover the best routes and then enter them into the tables

Static Routing

- Routing table constructed manually or from a file at boot time
- Not good if the network topology changes frequently

- Cannot change a route based on link loads
- Most frequently used on

Dynamic Routing

- want to create routing tables automatically
- want to react to topology changes
- want to be able to choose best route based on dynamic characteristics like link loads, link error rates, ...
- two popular approaches:
  1. Distance Vector Routing
  2. Link State Routing

Distance Vector Routing

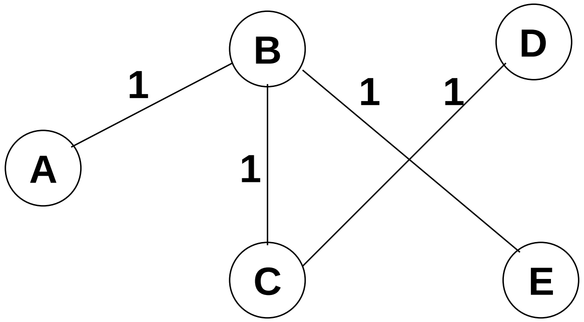
- Why the name Distance Vector ?
  - Distance:
  - Vector:

- Basic Idea:
  - Each switch remembers the shortest known route (next hop) to every possible destination.
  - If a switch learns of a shorter route, replace the old information with the new.
  - Periodically tell your neighbors the routes you know about.

Distance Vector Routing Tables

- Each host maintains a routing table of the form:

- Initially each switch starts off with a routing table filled with
- The distance/cost to unknown nodes starts as



- The routing Table at A starts out as:

Distance Vector Example

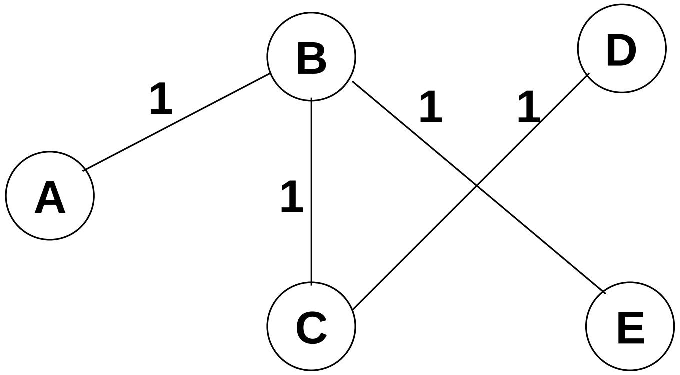
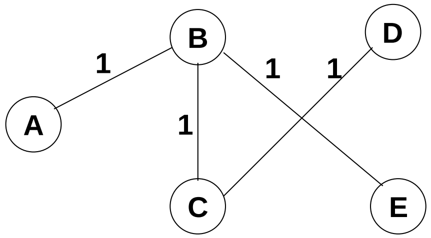


Figure 8: Example network

- Initially each node only knows directly connected neighbors
- Each node then sends its table to its neighbors
- When the neighbor receives a routing table, it updates its routes to use the best know routes.

Distance Vector Example: (continued)



- Initial routing tables (each row = a table)

	A	B	C	D	E
A					
B					
C					
D					
E					

- After the first exchange of update messages:

	A	B	C	D	E
A					
B					
C					
D					
E					

- How long will it take before all nodes know all routes?

Remaining Problems?

- How often should a node send updates?

- What if a route becomes longer? Or a link Fails?

- Will packets get where they are going at all times?

- What about graphs with loops?

Algorithm Outline

a. Wait for next update message
b. Iterate through entries in message
c. If entry has shorter path to destination:

   1. Insert source as next hop to destination
   2. Record distance as distance from next hop to destination PLUS distance from this switch to next hop
   3. trigger an update with the new information

Link State Routing

- Is a 2-part algorithm:

- Note that DV routing has no idea what the topology looks like

Determining The Topology

- Each machine finds out who its directly connected neighbors are. How?
- Broadcast a packet that tells who your neighbors are.
- Eventually all nodes will know who is connected to who

Determining the Best Route

- We know the entire topology (from the link-state messages)
- Can apply any graph algorithm to find the shortest path from any source to any destination
- Typically use Djikstra's Shortest Path Algorithm (SPF)

DV vs. LS

Distance Vector Routing	Link State Routing
Send packets only to neighbors, May trigger other xmissions	Send by flooding to everyone
Message size depends on the number of hosts in the network Thus, msg size does not scale	Message size depends on the number of neighbors Thus msg does scale well
Depends on intermediate nodes computing routes correctly	Does not depend on routes calculated by other machines
Routing msgs change as they propagate across the net	Link status msgs propagate unchanged
Loops can arise that must be handled	Since each node computes routes independently, loops are not a problem
May take a while to stablize after a link goes down	Stabalizes quickly after a link goes down

Interior Gateway Protocols

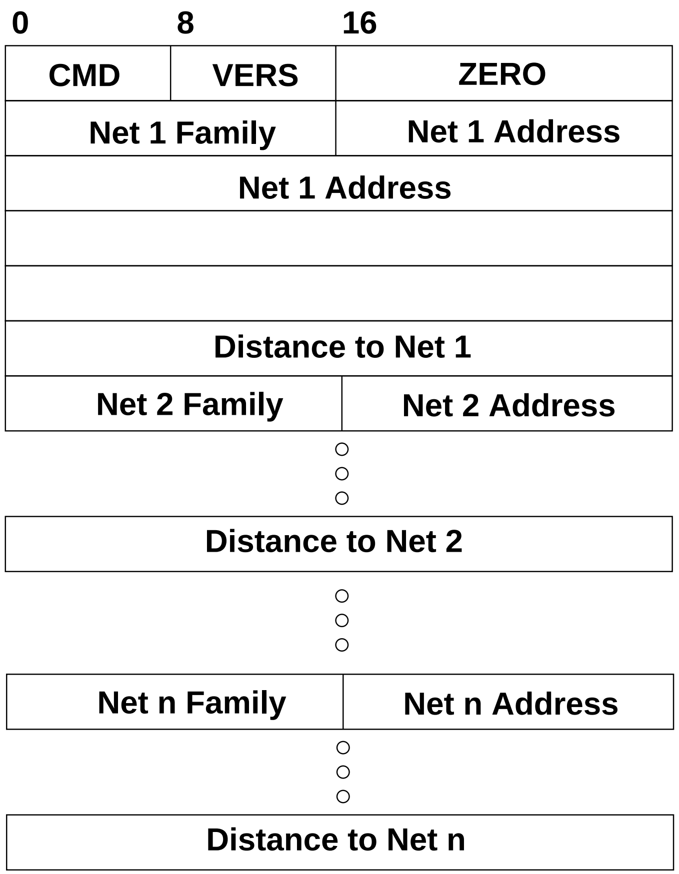
- Interior Gateway Protocols (IGP) are routing protocols used within AS (Autonoumous System)
- Each AS can use its own IGP
- Gateways on the border with other ASes run multiple routing protocols
- Example IGP Protocols:
  - RIP
  - HELLO
  - OSPF

Routing Information Protocol (RIP)

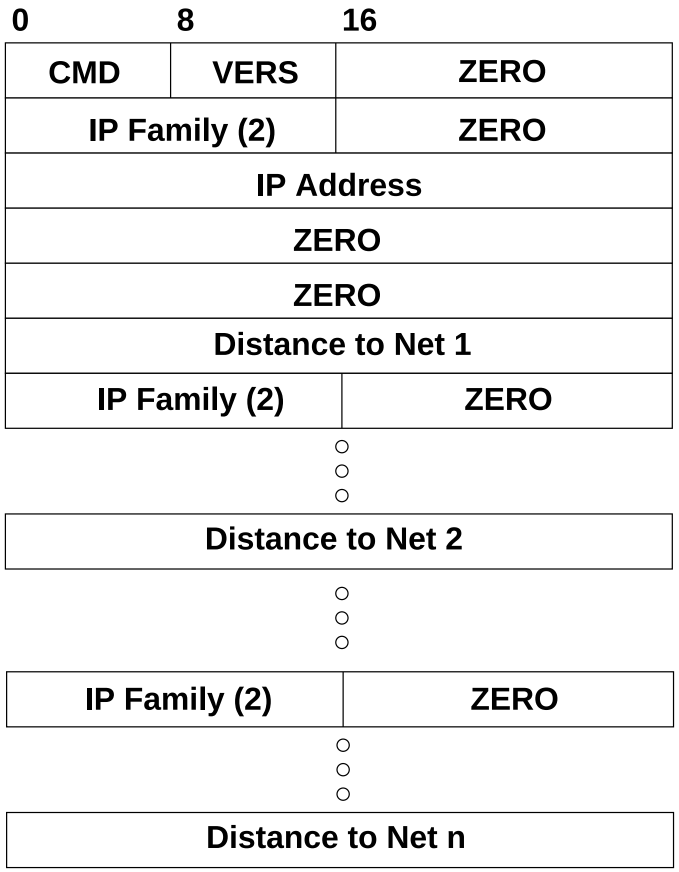
- One of the most popular IGPs because it was in routed in BSD UNIX.
- It can also be used for protocols other than IP
- It employs a distance vector algorithm
- Advertises the cost of reaching other networks
- The metric used = number of hops
- It solves the routing loops problem by limiting infinity
- It used both triggered and periodic updates

- RIP daemons can be either passive or active
  - passive = just listen to route advertisements
  - active = listen and send out advertisements

RIP Packet Format



RIP Packet Format (IP Addrs)



HELLO

- Like RIP, HELLO employs a distance vector algorithm
- Unlike RIP, it uses   as the metric
- Assumes synchronized clocks

Open SPF (OSPF)

- Open means the standard is widely available
- Is a link-state SPF algorithm
- Is a reasonably complex protocol
- Has several “enhancements” over standard link-state routing
  - Type of Service Routes
  - Load Balancing
  - Authenticated Routing Message Exchange

– Designated Router

– Virtual Links

– External Routing Info

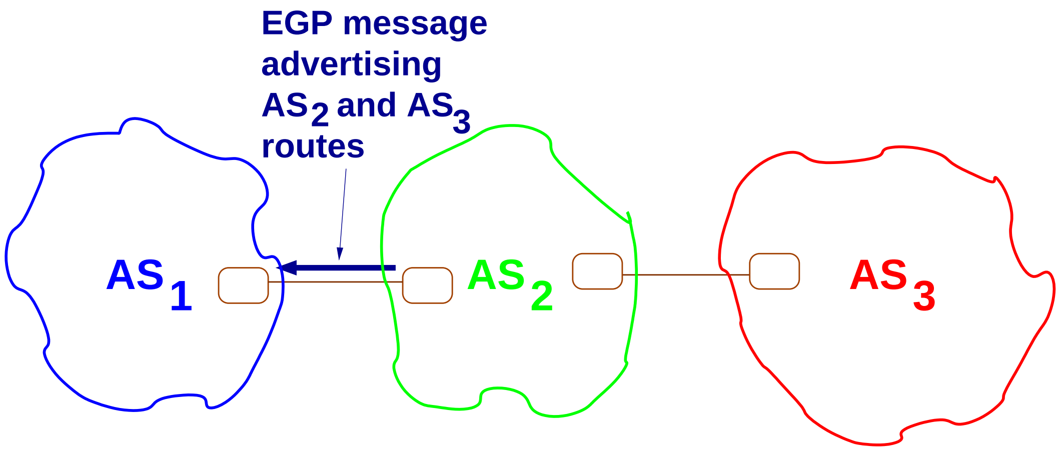
– Areas

Border/Exterior Gateway Protocols

- Border/Exterior Gateway Protocols are routing protocols used to communicate between ASes.
- Only the border routers speak an exterior gateway protocol
- If two border routers in different ASes are connected, they must speak the same border gateway protocol
- Example Border Gateway Protocols are:
  - EGP
  - BGP

Exterior Gateway Protocol (EGP)

- One of the earliest border gateway protocols
- Was used in the early Internet
- It requires a strict hierarchy of ASes
- Basic Idea: Advertise all routes in an AS (regardless of the IGP protocol used to find them) to the neighboring AS.
- However, EGP also allows a border router to advertise routes to "external routers" (routers not in the router's AS).



EGP Features

1. Performs neighbor acquisition

2. Continously checks connectivity

3. Exchanges routing update messages

EGP Problems

1. It only advertises “reachability”
2. ASes must be arranged in a hierarchy
3. No load sharing
4. Can produce non-optimal routes
5. It is difficult (or impossible) to switch to an alternate path if a link fails

Border Gateway Protocol (BGP)

- Was a follow-on to the EGP protocol
- Is probably the most popular external gateway protocol in use today.
- Is a pretty complicated protocol

BGP Features

- Allows and arbitrary collection of ASes

- Defines 3 types of ASes

- Allows multiple “Border Gateways”

- Like EGP, it just finds “a path”

- But it can make some routing choices

BGP Basics

- BGP is different than EGP in that the speaker advertises “complete paths”
- Routes are recorded as a list of ASes, not a list of networks
- You can issue an explicit “withdraw message” to remove routes