Introduction to Routing

CSE 561 Lecture 4, Spring 2002.
David Wetherall

Looking back: concepts you should be comfortable with …

- Protocols and layering, the end-to-end argument
- Circuit / packet switching; virtual circuits and datagrams
- Internetworking and packet fragmentation
- Timeouts and retransmissions
- Sliding windows and flow control
- Connection establishment
- Checksums and Forward Error Correction

The essence of routing

How do I get
there from **here**?

Overview

- Taxonomy of kinds of forwarding / routing
- Intra-domain routing (Review)
  - Distance Vector
  - Link State
- Inter-domain routing
  - Start on BGP; defer Policy until next time

What does a router do?

- **Forwarding**
  - Move packet from input link to the appropriate output link
    - “Next hop” only path to ultimate destination
  - Purely local computation
  - Must go be very fast (executed for ever packet)
- **Routing**
  - Doing work so you’re sure that the “next hop” actually leads to the destination
  - Distributed computation and communication
  - Can go slower (only important when topology changes)

Kinds of forwarding

- Source routing
  - Complete path in packet
- Virtual circuits
  - Set up path out-of-band and store path identifier in routers
  - Local path identifier in packet
- **Global address lookup**
  - Router looks up address in forwarding table
  - Forwarding table contains (address, next-hop) tuples

Source routing

- Packet contains complete ordered path information
  - I.e. node A then D then X then J…
- Host computes path
- Router looks up next hop in packet header
- Strips next hop and forwards remaining packet

Source routing evaluation

- Strengths
  - Very fast to lookup next hop
  - Very flexible (every packet can take different path)
- Weaknesses
  - Host must know global topology and detect failures
  - Variable packet header up to max path
- In practice
  - Ad hoc networks (DSR)
  - Multicomputer (Paragon) and SAN networks (Myrinet)
  - Minimal Internet support (LSR, SSR)

Virtual circuits

- Setup path out-of-band
  - Enter (input id, output id, next hop) entry into each router on path
  - Provide initial local input id to sending host as path id
- Forwarding
  - Send packet with path id
  - Router looks up input, swaps for output, forwards on next hop
  - Repeat until reach destination

Virtual circuit evaluation

- Strengths
  - Table lookup for forwarding (why faster than IP lookup?)
  - Flexible (one path per flow)
- Weaknesses
  - Requires connection setup before can send
  - Complicated to deal with failures
- In practice
  - ATM: fixed VC identifiers and separate signaling code
  - MPLS: ATM meets the IP world (why? *traffic engineering*)

Global address lookup

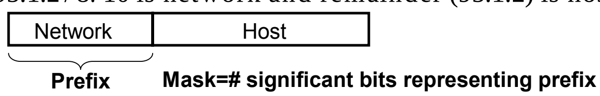
- All addresses are globally known
- Host sends packet with destination address in header
- Router maintains forwarding table
  - (Address, next-hop) tuple
  - Lookup address, send packet to next-hop link
- Distributed routing protocol used to populate tables

Global address evaluation

- Strengths
  - Handles failures well; No path state, so any router can forward any packet
  - No connection setup required
- Weaknesses
  - Inflexible
    - Usually all packets to destination follow same path
  - More state
    - Must store information on all destinations even if never used
  - Forwarding lookup more expensive
    - This is a whole lecture in itself; not now
- In practice
  - IP routing

Recap: Classless IP addressing

- Routes represented by tuple (network prefix / mask)
  - Allows arbitrary allocation between network and host address
  - e.g. 10.95.1.2 / 8: 10 is network and remainder (95.1.2) is host



- Route lookup: longest prefix match
  - For a given destination, find entry in route table that matches the most number of bits (i.e. with the largest mask)
  - Example: 128.95.4.1
    - One route for 128.95.0.0 / 16 (CMU)
    - One route for 128.95.4.0 / 24 (CMU SCS)

Intra-domain routing

- Routing **within** a network / organization
- A **single** administrative domain

- Overall goals
  - Adapt quickly to failures or topology changes
  - Optimize use of network resources
  - Provide intra-network connectivity
  - Scale to large networks

- Problem statement
  - Network is a directed graph G=(V,E)
  - Routers are vertices, links are edges labeled with some metric
    - For simplicity ignore hosts, they are part of each V
  - For each V, find the shortest path to every other V

Quick aside: host routing

- Generally, hosts are *single-homed*
  - Connected to a single network
- Don't need to understand topology
- Can simply have a *default* route
  - All non-local traffic sent to default next hop (a router)
  - Router maintains "default-free" forwarding table (or knows how to get to a router that does)

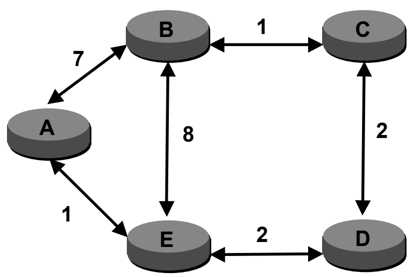
Three approaches

- **Static**
  - Type in the right answers and hope they are always true
- **Distance vector**
  - Tell your neighbors when you know about everyone
- **Link state**
  - Tell everyone what you know about your neighbors

Distance Vector routing (Review)

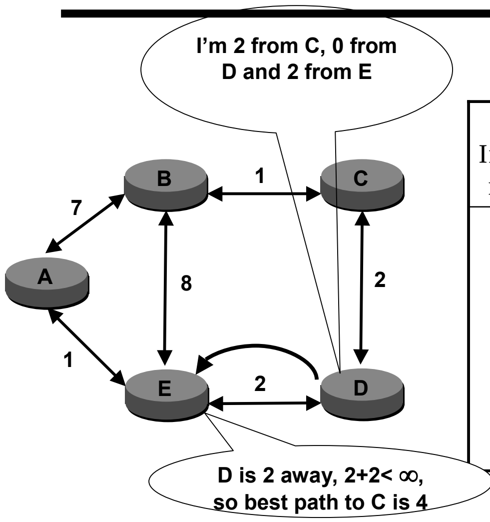
- Assume
  - Each router knows own address & cost to reach neighbors
- Goal
  - Calculate routing table containing next-hop information for every destination at each router
- **Distributed Bellman-Ford algorithm**
  - Each router maintains a vector of costs to all destinations
    - Initialize neighbors with known cost, others with infinity
  - Periodically send copy of distance vector to neighbors
  - On reception of a vector
    - If neighbor's path to a destination is shorter, switch to it

Initial conditions



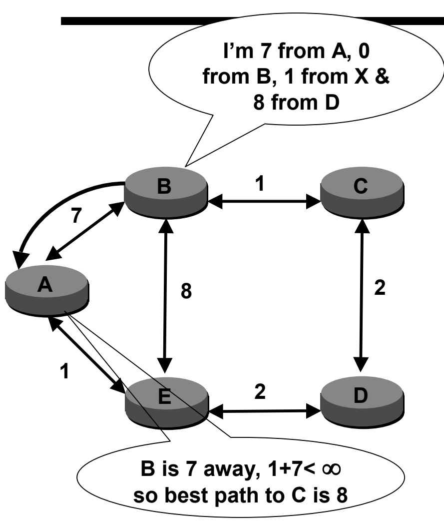
Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	∞	2	0

E receives D's vector



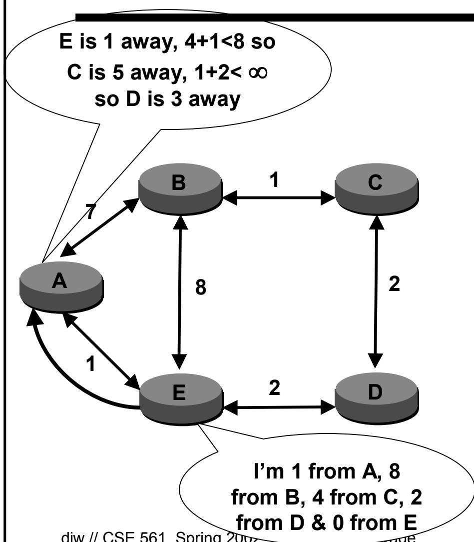
Info at node	Distance to Node A	B	C	D	E
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	**4**	2	0

A receives B's vector



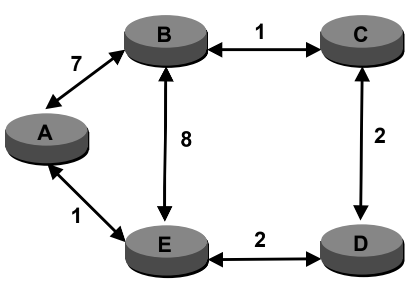
Info at node	Distance to Node A	B	C	D	E
A	0	7	**8**	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

A receives E's vector



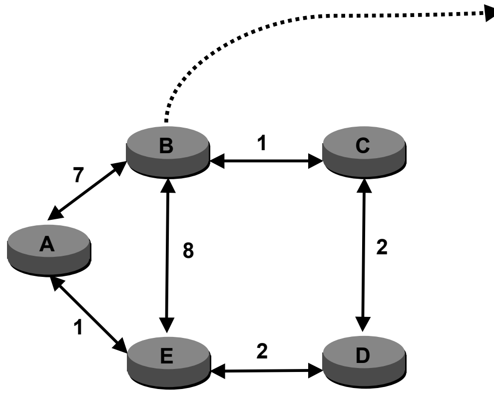
Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	7	**5**	**3**	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

Final state



Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	6	5	3	1
B	6	0	1	3	5
C	5	1	0	2	4
D	3	3	2	0	2
E	1	5	4	2	0

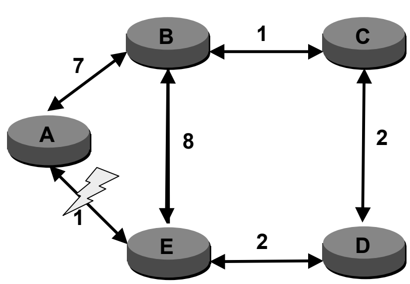
View from a node (B)



Dest	Next hop			
	A	**A**	**E**	**C**
A	0	7	9	**6**
B	6	**0**	13	2
C	5	12	12	**1**
D	3	10	10	**3**
E	1	8	8	**5**

Link failure

- **A marks distance to E as x, and tells B**
- **E marks distance to A as x, and tells B and D**
- **B and D recompute routes and tell C, E and E**
- **etc… until converge**



Info at node	Distance to Node				
	A	**B**	**C**	**D**	**E**
A	0	7	8	10	12
B	7	0	1	3	5
C	8	1	0	2	4
D	3	3	2	0	2
E	12	5	4	10	0

Link State routing (Review)

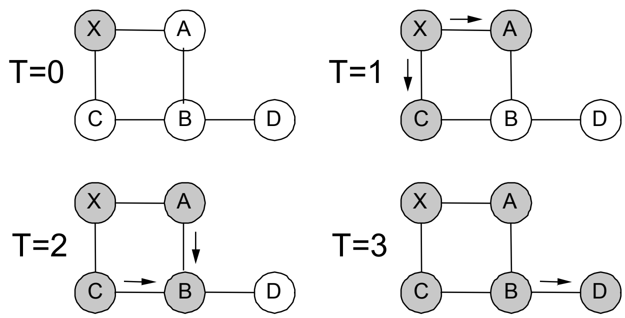
- Same goal, different approach
- Two phases:
  - **Reliable flooding**
    - Tell **all** routers what you know about your **local** topology
  - **Path calculation** (Dijkstra's algorithm)
    - Each router computes best path over **complete** network

Reliable flooding

- Goal: tell everyone what you know about local topology
- Periodically send link state packets (LSPs) on **all** links
  - LSP contains [node, neighbors, costs, sequence number]
- If node X receives an LSP from node Y over link Q
  - If it is the "newest" LSP from Y that X has seen then save it in local link state database & forward LSP on all links **except** Q
  - Otherwise drop LSP
- Use explicit ACKs and retransmits to make flooding reliable
- Each LSP will travel at most once over each link

Flooding example

- LSP generated by X at T=0
- Nodes become orange as they receive it



djw // CSE 561, Spring 2002, with credit to savage L4.27

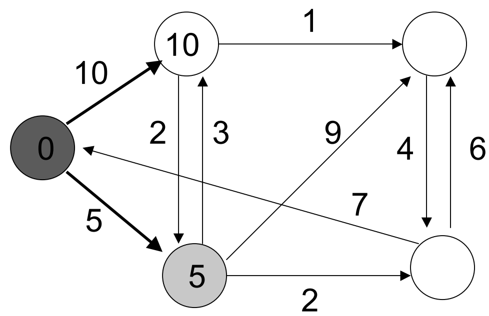
Dijkstra's Shortest Path Tree (SPT) algorithm

- Graph algorithm for single-source shortest path tree

```
S ← {}
Q ← <all nodes keyed by distance>
While Q != {}
        u ← extract-min(Q)
        S ← S plus {u}                    ← u is done
        for each node v adjacent to u
                "relax" the cost of v
```

djw // CSE 561, Spring 2002, with credit to savage L4.28

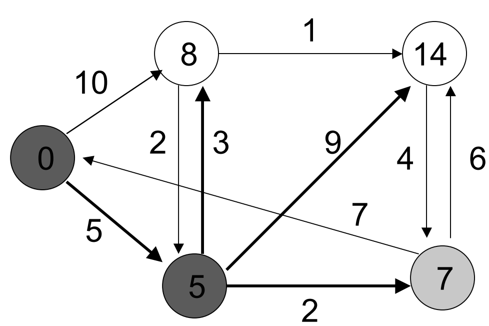
Example – Step 2



djw // CSE 561, Spring 2002, with credit to savage

L4.29

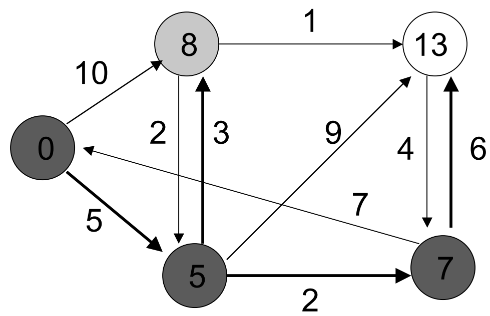
Example – Step 3



djw // CSE 561, Spring 2002, with credit to savage

L4.30

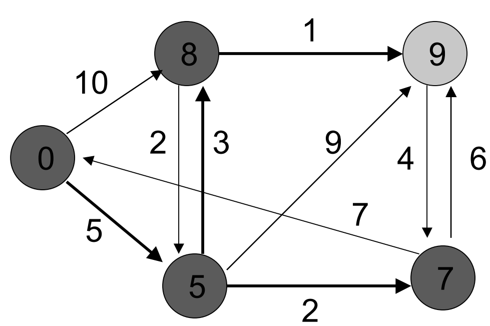
Example – Step 4



djw // CSE 561, Spring 2002, with credit to savage

L4.31

Example – Step 5



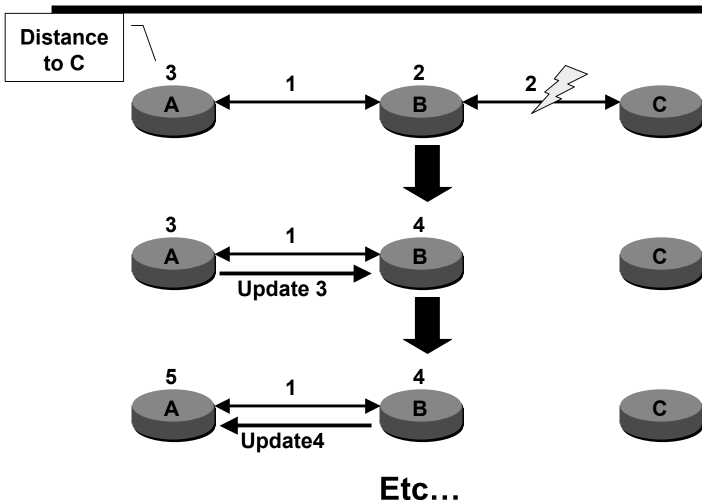
djw // CSE 561, Spring 2002, with credit to savage

L4.32

DV and LS comparison

- DV is simple, but convergence can be slow as each node only has local information.
- LS offers faster convergence and better stability (hopefully), but it's more complex.
- Arpanet switch from DV to LS because of this, and today ISPs use LS protocols (OSPF, IS-IS).

DV Problems: *Count to Infinity*



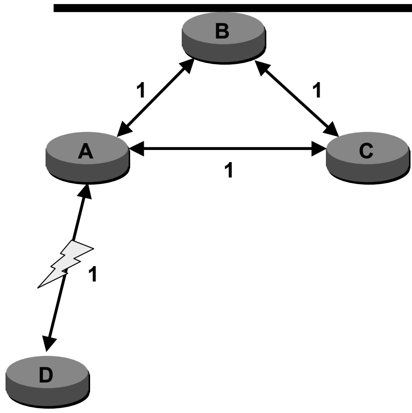
Why?

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

Many potential solutions

- Hold downs
  - As metric increases, delay propagating information
  - Limitation: ?
- 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: ?
- Loop avoidance
  - Full path information in route advertisement (e.g., BGP)
  - Explicit queries for loops (e.g. DUAL)
  - Limitation: ?

How split horizon/pv fails



- A tells B & C that D is unreachable
- B tells C that D is unreachable
- B tells A that D is reachable with cost=3 (since route is through C, split horizon doesn't apply)
- A tells C that D is reachable through A (cost=4)
- Etc…

DV: Other issues

- When to send route updates?
- Periodically
  - Limits granularity of failure recovery
  - Global synchronization can cause packet loss
- Jittered
  - Random offset from periodic deals with synchronization problem
- Triggered
  - Send updates immediately when metric changes
  - Converges more quickly, but causes flood of packets

Distance Vector in practice

- RIP
  - Small infinity (RIPv1, inf=16)
  - Split horizon / poison reverse
  - Jittered 30 second periodic updates
  - Triggered updates on failure
  - Metric is hop count
- EIGRP (Cisco proprietary)
  - Uses DUAL algorithm to avoid loops at all times
  - Keeps track of alternate loop-free next hops; explicit queries for loop-free paths otherwise
- BGP
  - Full path information to avoid loops

Reliable flooding challenges

- When link / router fails need to remove old data…how?
  - LSPs carry sequence numbers to distinguish new from old
  - Send a new LSP with cost infinity to signal a link down
- What happens when a router fails and restarts?
  - What sequence # should it use? Don't want data ignored
  - One option: Age LSPs and send with cost 0 to purge
  - Router can listen at startup to learn right sequence #
- What happens if the network is partitioned and heals?
  - Different LS databases must be synchronized
  - Use version #s

Link State in practice

- OSPF (Open Shortest Path First) and IS-IS
  - Most widely used IGPs
  - Run by almost all ISPs and many large organizations
- Basic link state algorithm plus many features:
  - Authentication of routing messages
  - Extra hierarchy: Partition into routing areas
  - Load balancing: Multiple equal cost routes

Discussion

- How to pick metrics?
- How can you do load balancing?
- How does congestion impact routing?
- What if a router lies?
- What are the biggest scalability issues?

Inter-domain routing: historic context

- Original ARPAnet had single routing protocol
  - Dynamic DV scheme, replaced with static metric LS algorithm
- New networks came on the scene
  - NSFnet, CSnet, DDN, etc…
  - With their own routing protocols (RIP, Hello, ISIS)
  - And their own rules (e.g. NSF AUP)
- Problem: how to deal with routing heterogeneity?

What to do?

- Some problems
  - **Consistency**: Network A uses hop count as a metric, Network B uses measured delay, Network C uses link capacity
  - **Policy**: Network A connects to Networks B and C. Network B is only allowed to carry network C's traffic?
- How would you resolve these problems?

One solution: Inter-domain routing

- Exterior Gateway Protocols (EGPs)
  - Only exchange **reachability** information (no metrics)
  - Decide what to do based on local policy
- Autonomous Systems (ASs)
  - Unit of abstraction in interdomain routing
  - Roughly, a network with common administrative control, a coherent internal routing policy, and presenting a **consistent** external view of connectivity
  - Represented by a 16-bit number
    - Example: UUnet (701), Sprint (1239), UCSD (7377)
  - Run IGPs within an AS, EGPs between ASs

First attempt

- Protocol called EGP (can be confusing)
  - Connected NSFnet Backbone to regional networks, DDN / Milnet, etc..
  - EGP only provided reachability information (no metrics)
  - Assumed spanning tree topology based on single backbone
    - No loops
- In 1995 NSFnet got out of the backbone business
  - Many backbones (MCI, Sprint, AT&T…)
  - Multiconnected regional networks
  - Meshed topology, loops…
- Need a new protocol

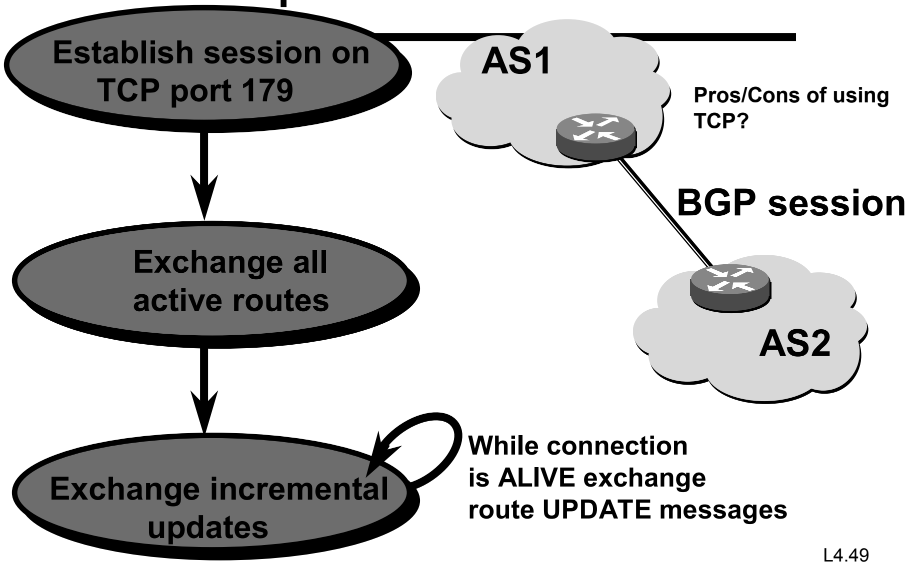
What kind of protocol?

- Link state?
  - Relies on global metric & policy
- Distance vector?
  - May not converge; loops
- Solution: path vector
  - Reachability protocol, no metrics
  - Route advertisements carry list of ASs
    - "I can reach 128.95/16 through path: AS73, AS703, AS1"
    - Automatic loop detection? How?

Border Gateway Protocol (BGP-4)

- Principal protocol used for routing across the Internet
  - Relatively simple protocol, complex usage
- Path vector protocol
  - Explicitly announce or withdraw routes
  - Routes include **attributes** in addition to path vector
  - Incremental updates (stateful)
- Policy is not part of protocol, but is built on top by filtering/mapping on attributes
  - Which routes do you listen to?
  - Which routes do you put in forwarding table?
  - Which routes do you advertise?

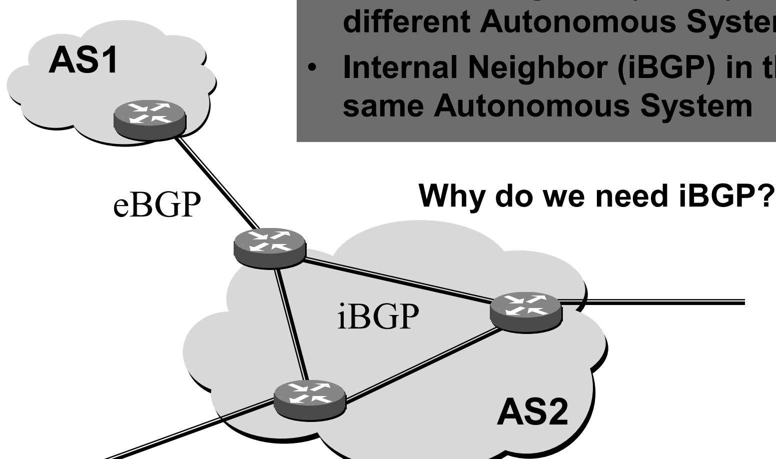
How BGP operates between nodes



L4.49

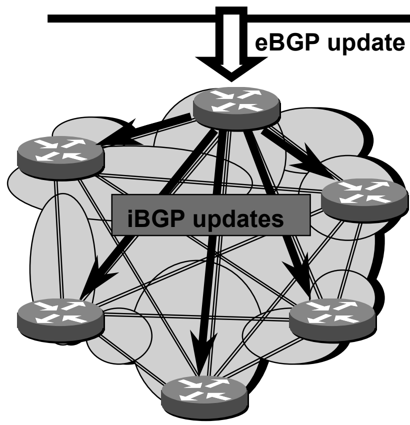
The Interior / Exterior split

- External Neighbor (eBGP) in a different Autonomous Systems
- Internal Neighbor (iBGP) in the same Autonomous System



L4.50

iBGP keeps eBGP consistent



iBGP neighbors do not announce routes received via iBGP to other iBGP neighbors.

- **iBGP is needed to avoid routing loops within an AS**
- **Injecting external routes into IGP does not scale and causes BGP policy information to be lost**

L4.51

Next time

- More on BGP: policy and mechanism
- Traffic engineering

L4.52