# CS4450

Computer Networks:
Architecture and Protocols

## Lecture 13
## Distance-vector, Internet, Addressing, Path-Vector (BGP)

**Rachit Agarwal**

# Announcements

- **Prelim: 28th March, In-class (Confirmed)**
    - **Nobody should be in conflict**

- During my first lecture, I promised you:
    - **I care about you(r learning)!**
    - **If you stick to the contract, I'll bring my A game in every lecture!**

- **You have been great so far!**

- I will stick to my promise
    - We are almost half-way through
    - If you think I am not bringing my A-game in the course
        - I want to know and improve!!!
    - **Please fill out the mid-term evaluation (this weekend)**
        - **Completely anonymized; only for my eyes; max 5 min**

# Goals for Today's Lecture

- **Finish Distance-Vector Protocol**
- **Internet Addressing**
- Begin Border-Gateway Protocol (BGP)

# Recap from last lecture

# Recap: Three flavors of protocols for producing valid routing state

- **Create Tree, route on tree**
  - E.g., Spanning tree protocol (switched Ethernet)
  - **Good:** easy, no (persistent) loops, no dead ends
  - **Not-so-good:** unnecessary processing, high latency, low bandwidth

- **Obtain a global view:**
  - Link state
  - **Good:** conceptually simple, no (persistent) loops, no dead ends
  - **Not-so-good:** flooding of link state to every node

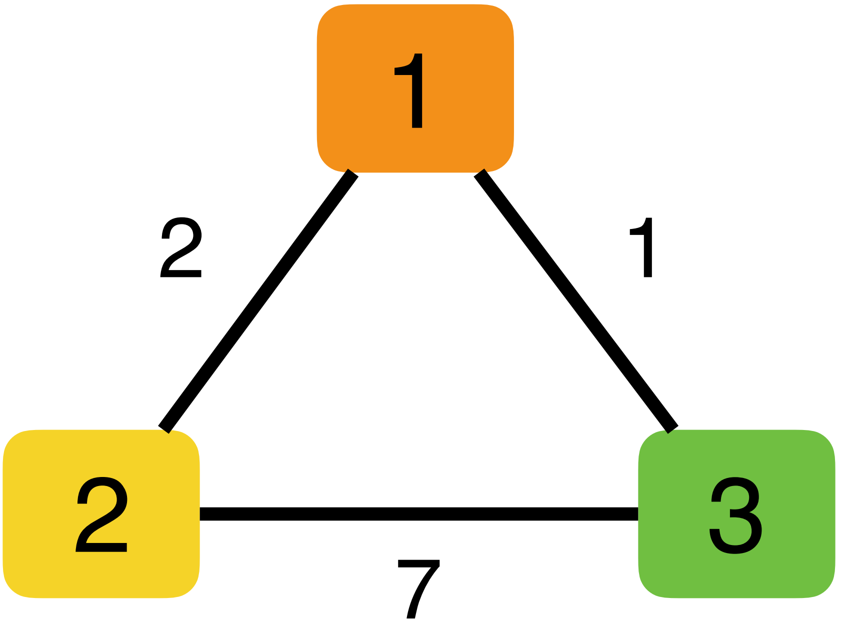
- **Distributed route computation:**
  - Distance-vector protocol

# Recap: Distance Vector Protocol

- **Messages (Y,d,X): For root Y; From node X; advertising a distance d to Y**
- Initially each switch X initializes its routing table to (X,0,-) and distance infinity to all other destinations
- Switches announce their entire distance vectors (routing table w/0 next hops)
- Upon receiving a routing table from a node (say X), each node does:
    - For each destination Y in the announcement (distance(X, Y) = d):
        - If current_distance_to_Y > d + cost of link to X:
            - update current_distance_to_Y = d
            - update next_hop_to_destination = X
- If shortest distance to any destination changed, send all neighbors your distance vectors

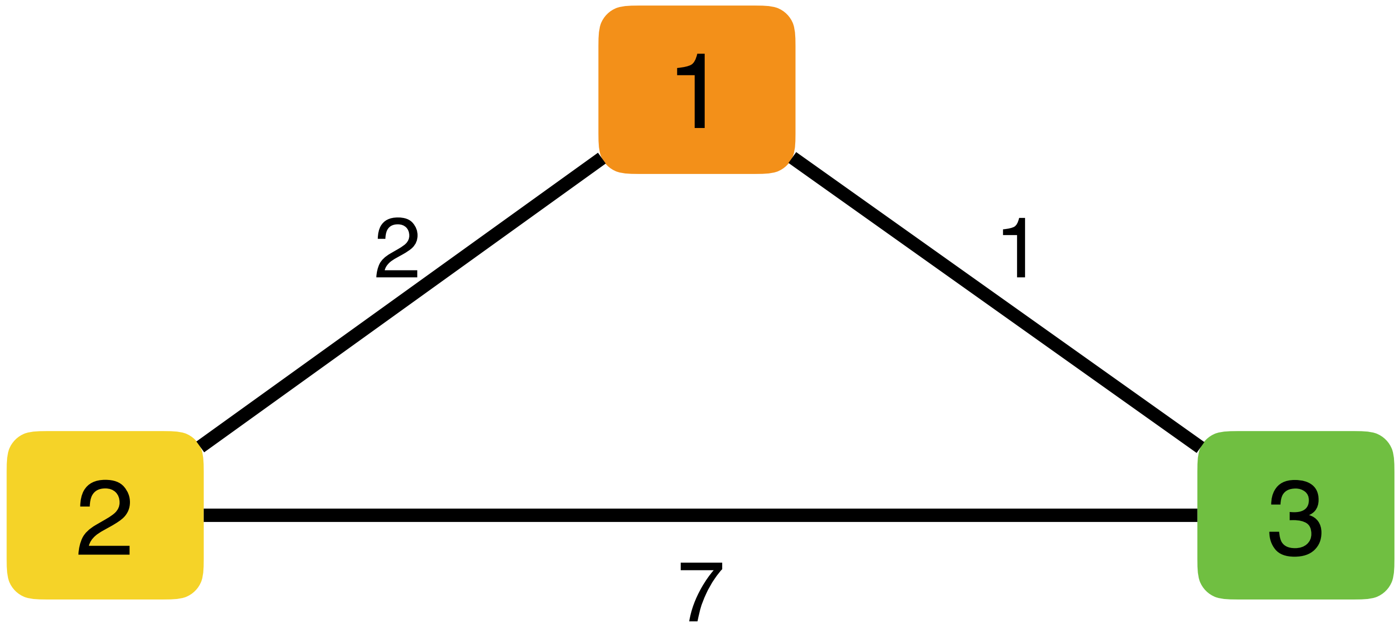
# Recap: Lets run the Protocol again on this example

# (with distance vectors)



# Round 1

| | distance | next-hop |
|---|---|---|
| 1 | 0 | - |
| 2 | infinity | |
| 3 | infinity | |

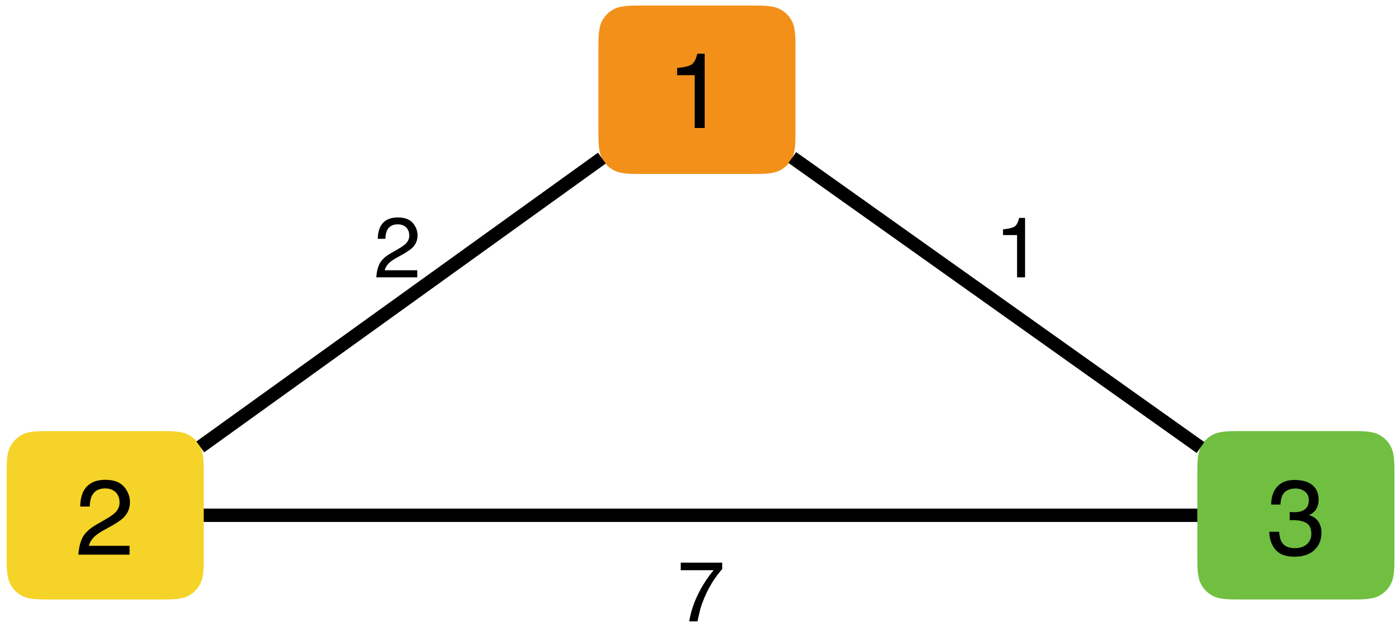



| | distance | next-hop |
|---|---|---|
| 1 | infinity | |
| 2 | 0 | - |
| 3 | infinity | |

| | distance | next-hop |
|---|---|---|
| 1 | infinity | |
| 2 | infinity | |
| 3 | 0 | - |

# Round 2

| | distance | next-hop |
|---|---|---|
| 1 | 0 | - |
| 2 | 2 | 2 |
| 3 | 1 | 3 |

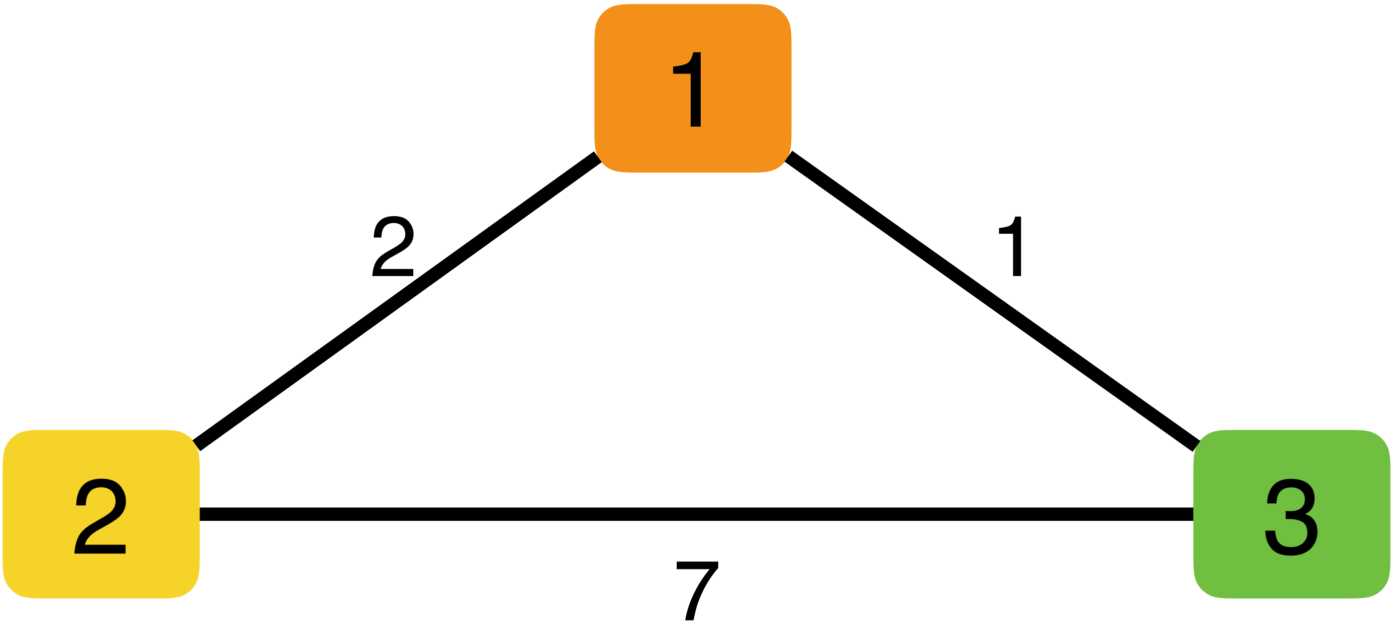



| | distance | next-hop |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 0 | - |
| 3 | 7 | 3 |

| | distance | next-hop |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 7 | 2 |
| 3 | 0 | - |

# Round 3

| | distance | next-hop |
|---|---|---|
| 1 | 0 | - |
| 2 | 2 | 2 |
| 3 | 1 | 3 |

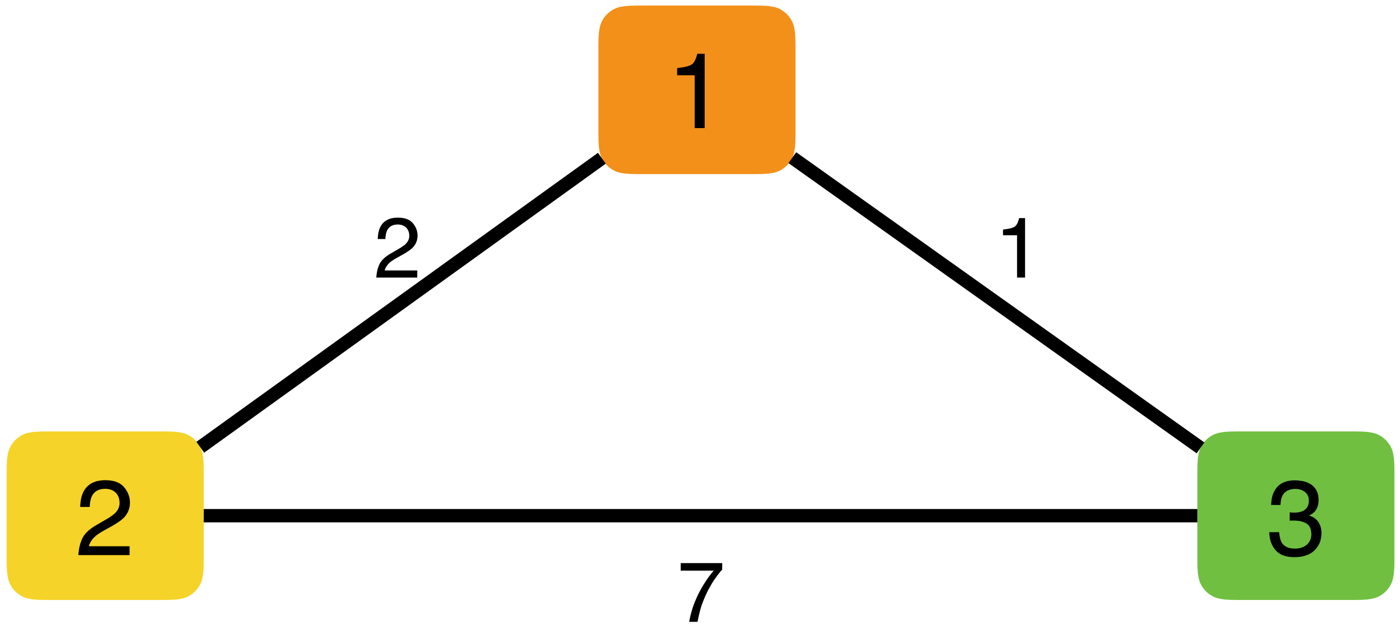



| | distance | next-hop |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 0 | - |
| 3 | **3** | **1** |

| | distance | next-hop |
|---|---|---|
| 1 | 1 | 1 |
| 2 | **3** | **1** |
| 3 | 0 | - |

# Round 4

| | distance | next-hop |
|---|---|---|
| 1 | 0 | - |
| 2 | 2 | 2 |
| 3 | 1 | 3 |




| | distance | next-hop |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 0 | - |
| 3 | 3 | 1 |

| | distance | next-hop |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 3 | 1 |
| 3 | 0 | - |

# From Algorithm to Protocol

- Algorithm:
  - Nodes use Bellman-Ford to compute distances
- Protocol
  - Nodes exchange distance vectors
  - Update their own routing tables
  - And exchange again…
  - Details: when to exchange, what to exchange, etc….

# Other Aspects of Protocol

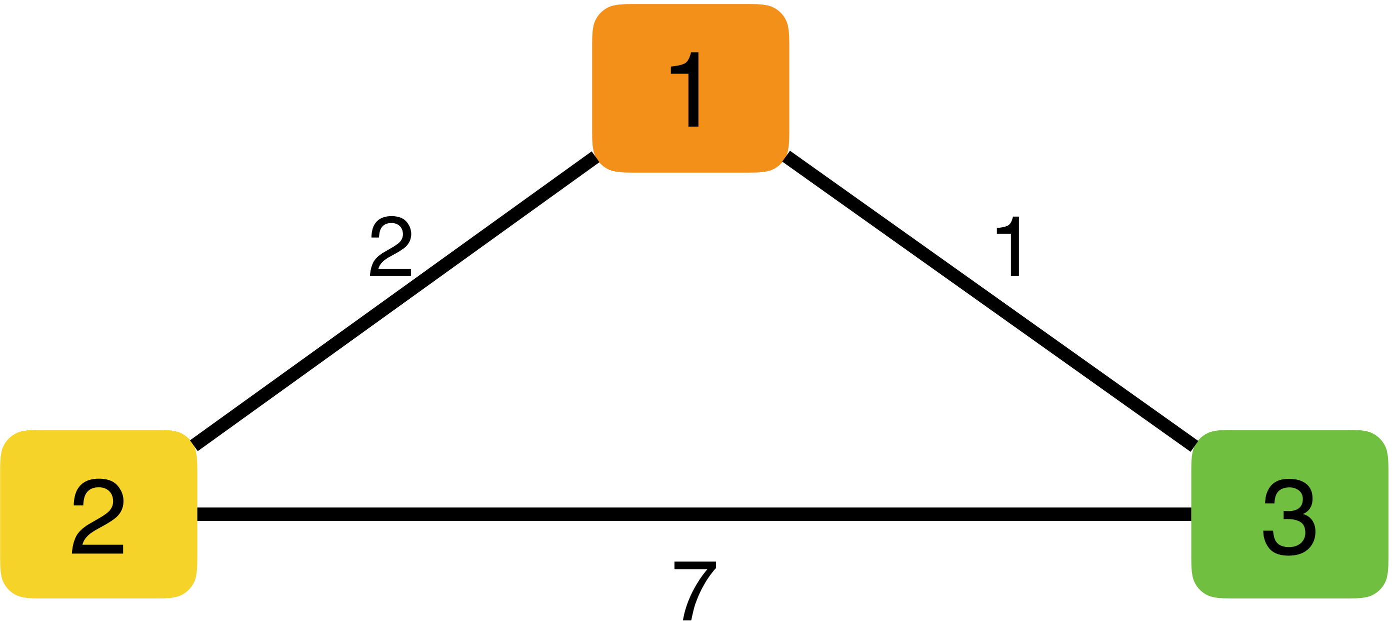
- When do you send messages?
    - When any of your distances d(u,v) change
        - What about when c(u,v) changes?
    - Periodically, to ensure consistency between neighbors
- What information do you send?
    - Could send entire vector
    - Or just updated entries
- Do you send everyone the same information
    - Consider the following slides

# One detail about Distance Vector:

# Handling Count-to-Infinity Problem

# Three node network

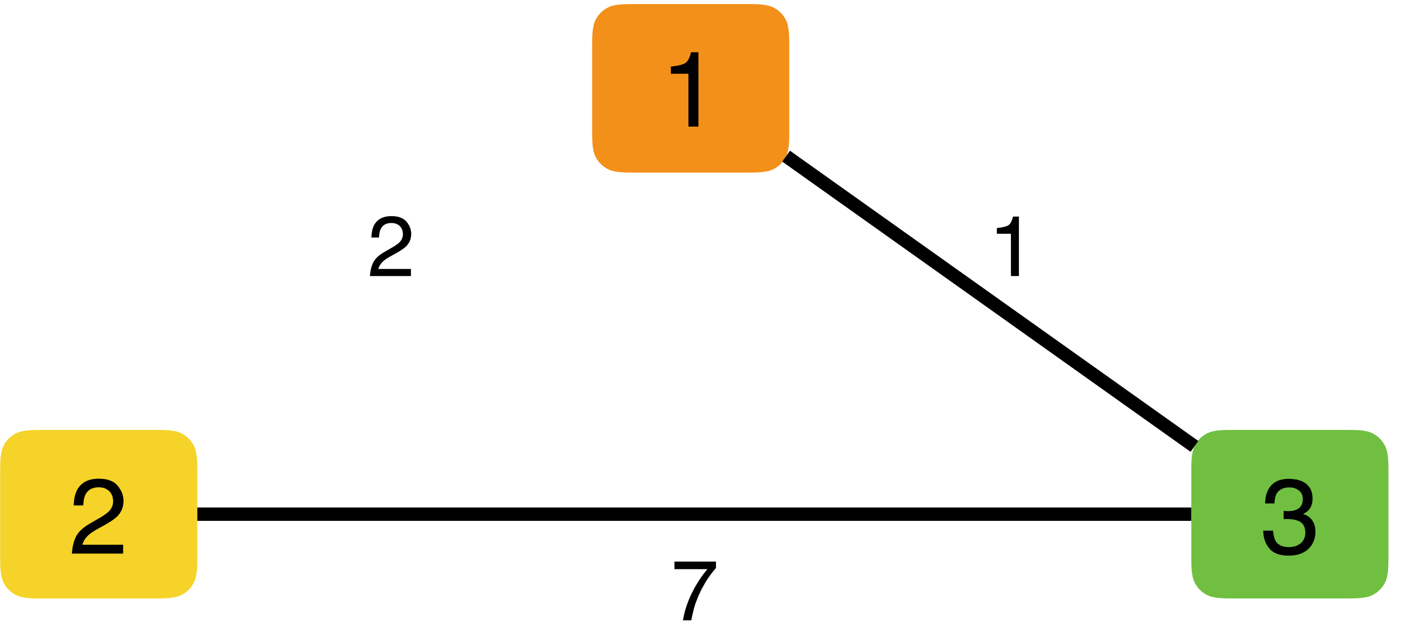
| | distance | next-hop |
|---|---|---|
| 1 | 0 | - |
| 2 | 2 | 2 |
| 3 | 1 | 3 |




| | distance | next-hop |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 3 | 1 |
| 3 | 0 | - |

# Three node network

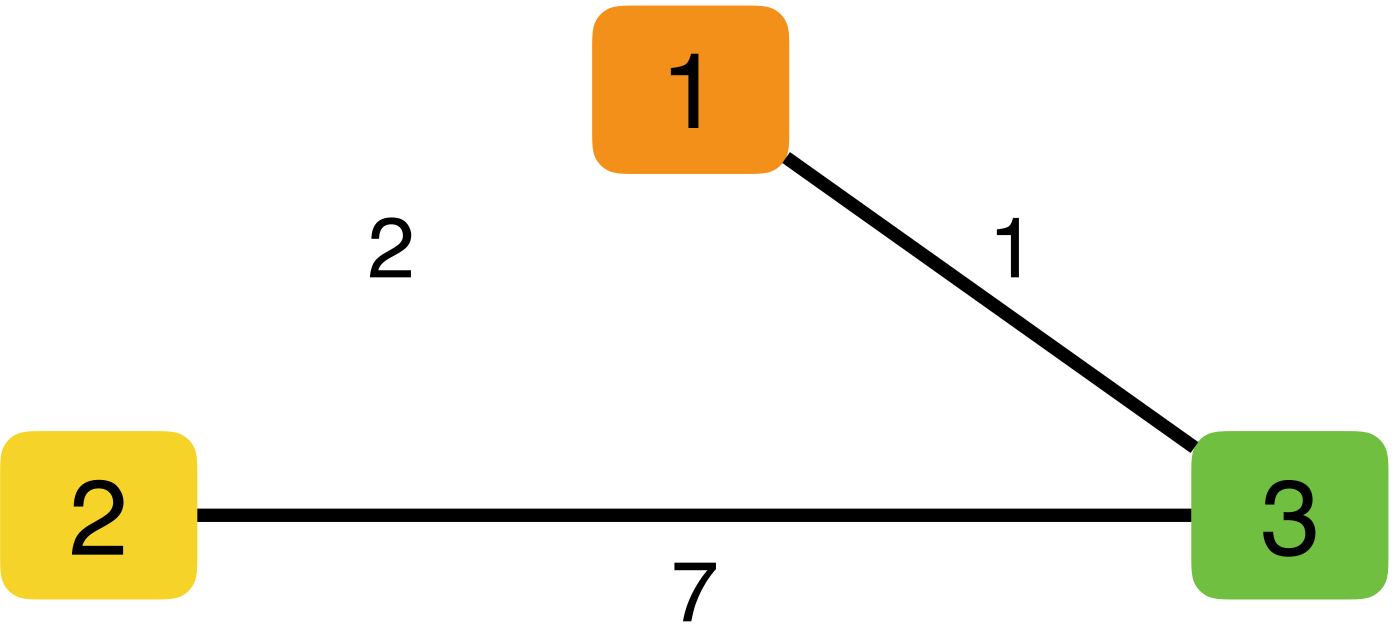
| | distance | next-hop |
|---|---|---|
| 1 | 0 | - |
| 2 | infinity | |
| 3 | 1 | 3 |




| | distance | next-hop |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 3 | 1 |
| 3 | 0 | - |

# Round 1

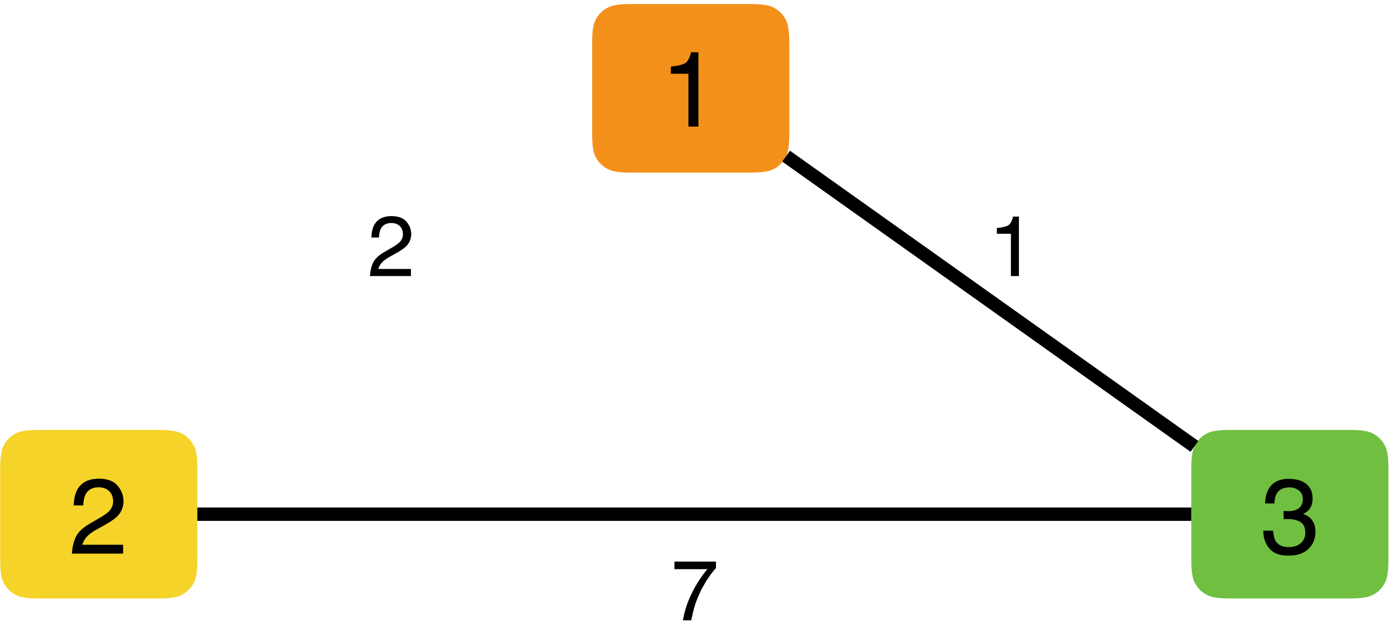
| | distance | next-hop |
|---|---|---|
| 1 | 0 | - |
| 2 | **4** | **3** |
| 3 | 1 | 3 |




| | distance | next-hop |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 3 | 1 |
| 3 | 0 | - |

# Round 2

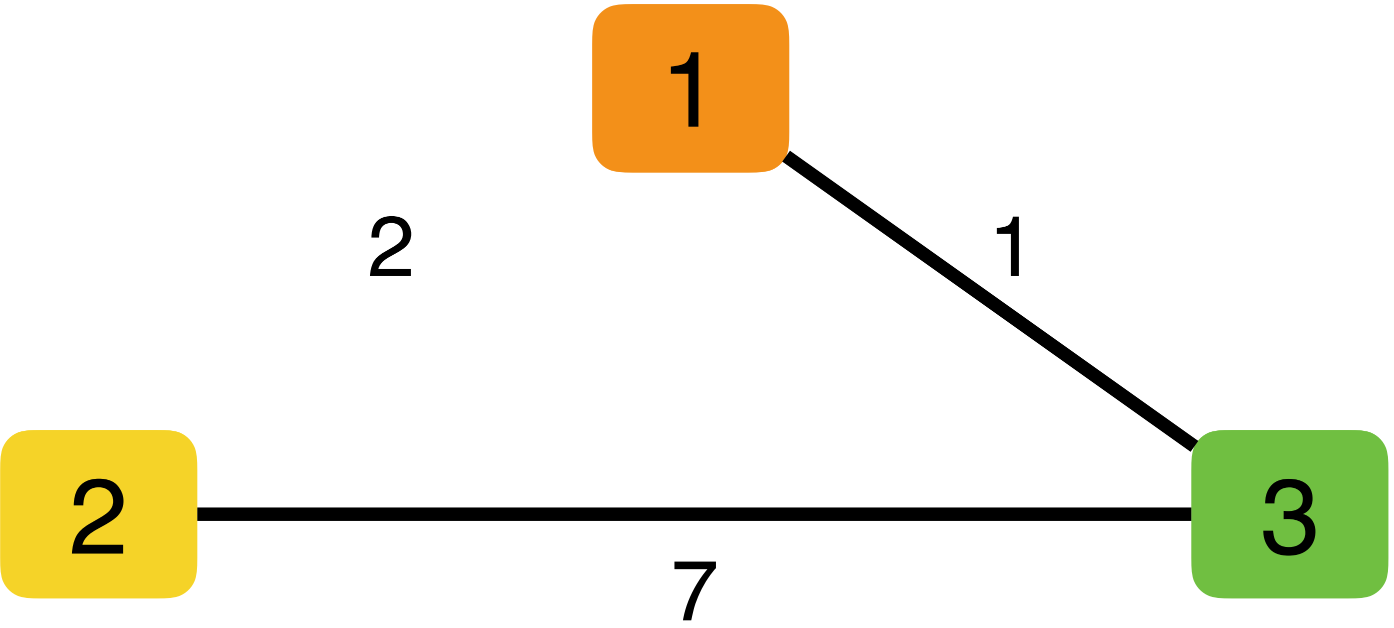
| | distance | next-hop |
|---|---|---|
| 1 | 0 | - |
| 2 | 4 | 3 |
| 3 | 1 | 3 |




| | distance | next-hop |
|---|---|---|
| 1 | 1 | 1 |
| 2 | **5** | **1** |
| 3 | 0 | - |

# Round 3

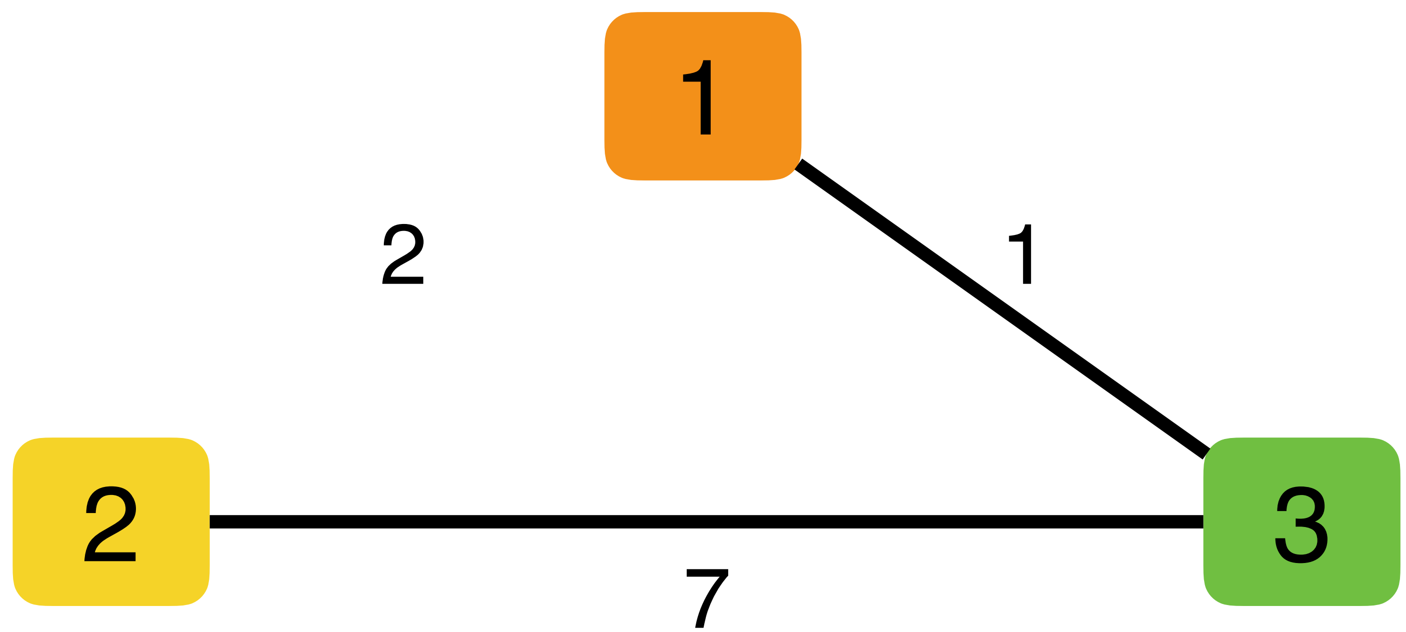
| | distance | next-hop |
|---|---|---|
| 1 | 0 | - |
| 2 | 6 | 3 |
| 3 | 1 | 3 |




| | distance | next-hop |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 5 | 1 |
| 3 | 0 | - |

# Round 4

| | distance | next-hop |
|---|---|---|
| 1 | 0 | - |
| 2 | 6 | 3 |
| 3 | 1 | 3 |

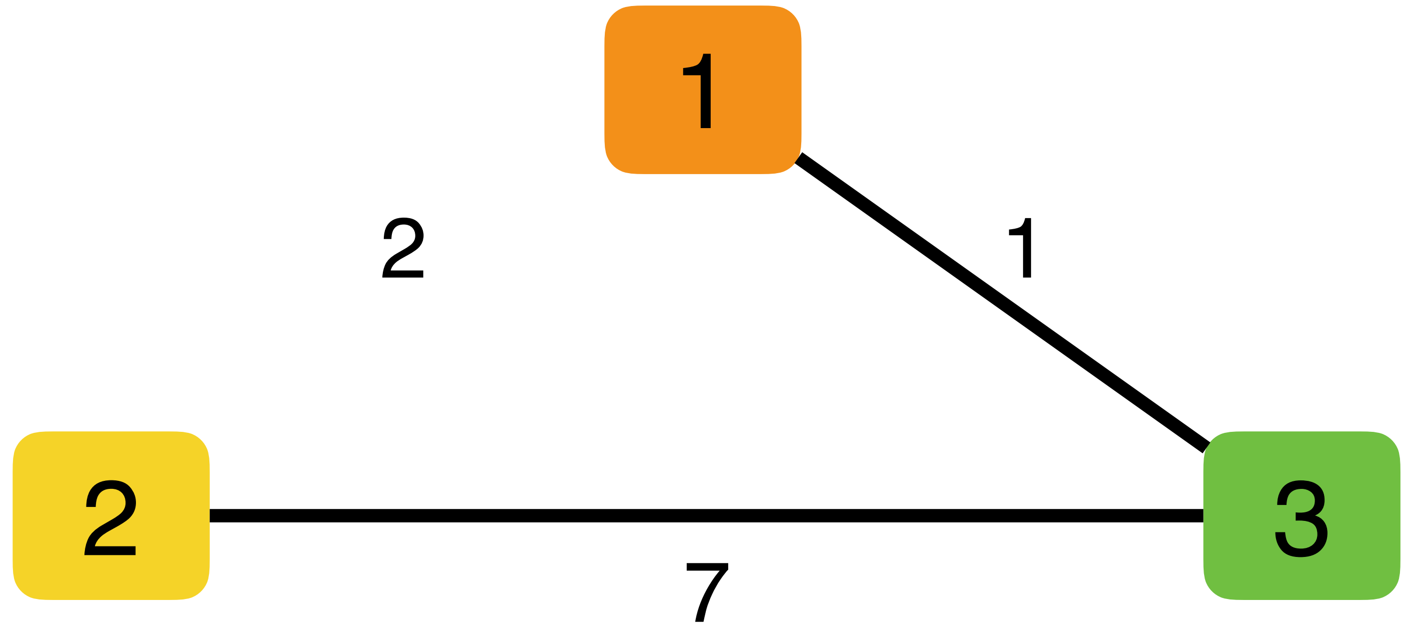



| | distance | next-hop |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 7 | 1 |
| 3 | 0 | - |

**COUNT-TO-INFINITY problem!!!!**

# Count-to-infinity problem

| | distance | next-hop |
|---|---|---|
| 1 | 0 | - |
| 2 | 6 | 3 |
| 3 | 1 | 3 |




**Not just due to failures:**
**Can happen with changes in cost!**

| | distance | next-hop |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 7 | 1 |
| 3 | 0 | - |

# How Can You Fix This?

- **Do not advertise a path back to the node that is the next hop on the path**
  - Called **"split horizon"**
  - Telling them about your entry going through them
    - Doesn't tell them anything new
    - Perhaps misleads them that you have an independent path
- **Another solution: if you are using a next-hop's path, then**:
  - Tell them not to use your path (by telling them cost of infinity)
  - Called "**poisoned reverse**"
- **More in Problem Set 3**

# Convergence

- Distance vector protocols can converge slowly
  - While these corner cases are rare
  - The resulting convergence delays can be significant

# Comparison of Scalability

- **Link-State:**
  - Global flood: each router's link-state (#ports)
  - Send it once per link event, or periodically
- **Distance Vector:**
  - Send longer vector (#dest) just to neighbors
    - But might end up triggering their updates
  - Send it every time DV changes (which can be often)
- **Tradeoff:**
  - LS: Send it everywhere and be done in predictable time
  - DV: Send locally, and perhaps iterate until convergence

**End of Distance-vector Routing**

**Now you know just as much as my PhD students :-)**

# Internet Addressing

# Addressing so far

- Each node has a "name"
    - We have so far worked only with names
    - Assumed that forwarding/routing etc. done on names
- Today:
    - Why do we need addresses?
    - Why do we assign addresses the way we assign addresses?

# Three requirements for addressing

- **Scalable routing**
  - How must state must be stored to forward packets?
  - How much state needs to be updated upon host arrival/departure?
- **Efficient forwarding**
  - How quickly can one locate items in routing table?
- **Host must be able to recognize packet is for them**

# Layer 2 (link layer): “Flat” Addressing

- Uses MAC address
  - “Names”, remember? Used as identifier
- Unique identifiers hardcoded in the hardware
  - No location information
- Local area networks route on these “flat” addresses
  - **Spanning Tree Protocol runs on switches and hosts**
  - Each switch stores a separate routing entry **for each host**
  - End-hosts store nothing
- Upon receiving a packet, an end-host:
  - Puts destination’s and its own MAC address in the header
  - Forwards it to the switch it is connected to
- **Destination is able to recognize the packet is for them using address**

# How does this meet our requirements?

- **Scalable routing**
    - How much state to forward packets?
        - One entry per host per switch
    - How much state updated for each arrival/departure?
        - One entry per host per switch

- **Efficient forwarding**
    - Exact match lookup on MAC addresses (exact match is easy!)

- **Host must be able to recognize the packet is for them**
    - MAC address does this perfectly

## Conclusion: L2 addressing does not enable scalable routing

# How would you scale L2?

- Suppose we want to design a much larger L2 network
- Must use MAC address as part of the address
    - Only way host knows that the packet is for them
- **But how would you enable scalable routing?**
    - Small #routing entries (less than one entry per host per switch)
    - Small #updates (less than one update per switch per host change)

# One possible Solution: Towards Internet-scale addressing

- Assign each end-host an addresses of the form — Switch:MAC
- Spanning Tree Protocol runs only on switches
    - So, each switch has one entry per switch (rather than per host)
- Upon receiving a packet, an end-host:
    - Puts destination's and its own Switch:MAC address in the header
    - Forwards it to the switch it is connected to
- **Switches forward the packet using first part of the address**
- **Destination is able to recognize the packet is for them using second part of the address**

# Layer 3: Hierarchical addressing

- Routing tables cannot have entry for each switch in the Internet
- Use addresses of the form — Network:Host
- Routers know how to reach all networks in the world
  - Routing algorithms only announce "Network" part of the addresses
  - Routing tables now store a next-hop for each "network"
- Forwarding:
  - Routers ignore host part of the address
  - When the packet reaches the right network
    - Packet forwarded using Host part of the address
    - Using Layer 2
- **This was the original IP addressing scheme**

# What do I mean by "network"

- In the original IP addressing scheme …
  - Network meant an L2 network
  - Often referred to as a "subnet"
  - There are too many of them now to scale

# Aggregation

- **Aggregation:** single forwarding entry used for many individual hosts

- Example:
    - In our scalable L2 solution: aggregate was switch
    - In our scalable L3 solution: aggregate was network

- Advantages:
    - Fewer entries and more stable
    - Change of hosts do not change tables
        - Don't need to keep state on individual hosts

# Hierarchical Structure

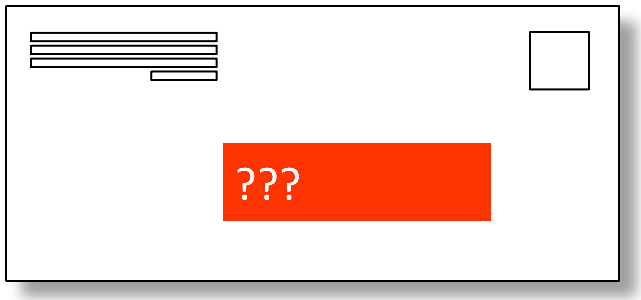
- The Internet is an “inter-network”
  - Used to connect networks together, not hosts

- Forms a natural two-way hierarchy
  - Wide Area Network (WAN) delivers to the right “network”
  - Local Area Network (LAN) delivers to the right host

# Hierarchical Addressing

- Can you think of an example?

- Addressing in the US mail
  - Country
  - City, Zip code
  - Street
  - House Number
  - Occupant “Name”



# IP addresses

- Unique 32 bit numbers associated with a host
- Use dotted-quad notation, e.g., 128.84.139.5

| Country | City, State | Street, Number | Occupant |
|---|---|---|---|
| (8 bits) | (8 bits) | (8 bits) | (8 bits) |
| 10000000 | 0-1010100 | 10001011 | 00000-101 |
| 128 | 84 | 139 | 5 |

# Original Addressing mechanism

- First eight bits: network address (/8)
  - Slash notation indicates network address
- Last 24 bits: host address
- Assumed 256 networks were more than enough!!!
  - Now we have millions!

# Suppose we want to accommodate more networks

- We can allocate more bits to network address

- Problem?
    - Fewer bits for host names
    - What if some networks need more hosts?

# Today’s Addressing: CIDR

- Classless Inter-domain Routing
- Idea: Flexible division between network and host addresses
- Prefix is **network address**
- Suffix is **host address**
- **Example:**
    - **128.84.139.5/23 is a 23 bit prefix with:**
    - First 23 bits for network address
    - Next 9 bits for host addresses: maximum 2^9 hosts
- **Terminology: “Slash 23”**

# Example for CIDR Addressing

- **128.84.139.5/23 is a 23 bit prefix with 2^9 host addresses**

| | | | |
|---|---|---|---|
| 10000000 | 0-1010100 | 10001011 | 00000-101 |
| 128 | 84 | 139 | 5 |

Network (23 bits) — Host (9 bits)

# Allocating addresses

- Internet Corporation for Assigned Names and Numbers (ICANN) …
- Allocates large blocks of addresses to Regional Internet Registries
  - E.g., American Registry for Internet Names (ARIN) …
- That allocates blocks of addresses to Large Internet Service Providers (ISP)
- That allocate addresses to individuals and smaller institutions
- Fake example:
  - ICANN -> ARIN -> AT&T -> Cornell -> CS -> Me

# Allocating addresses: Fake example

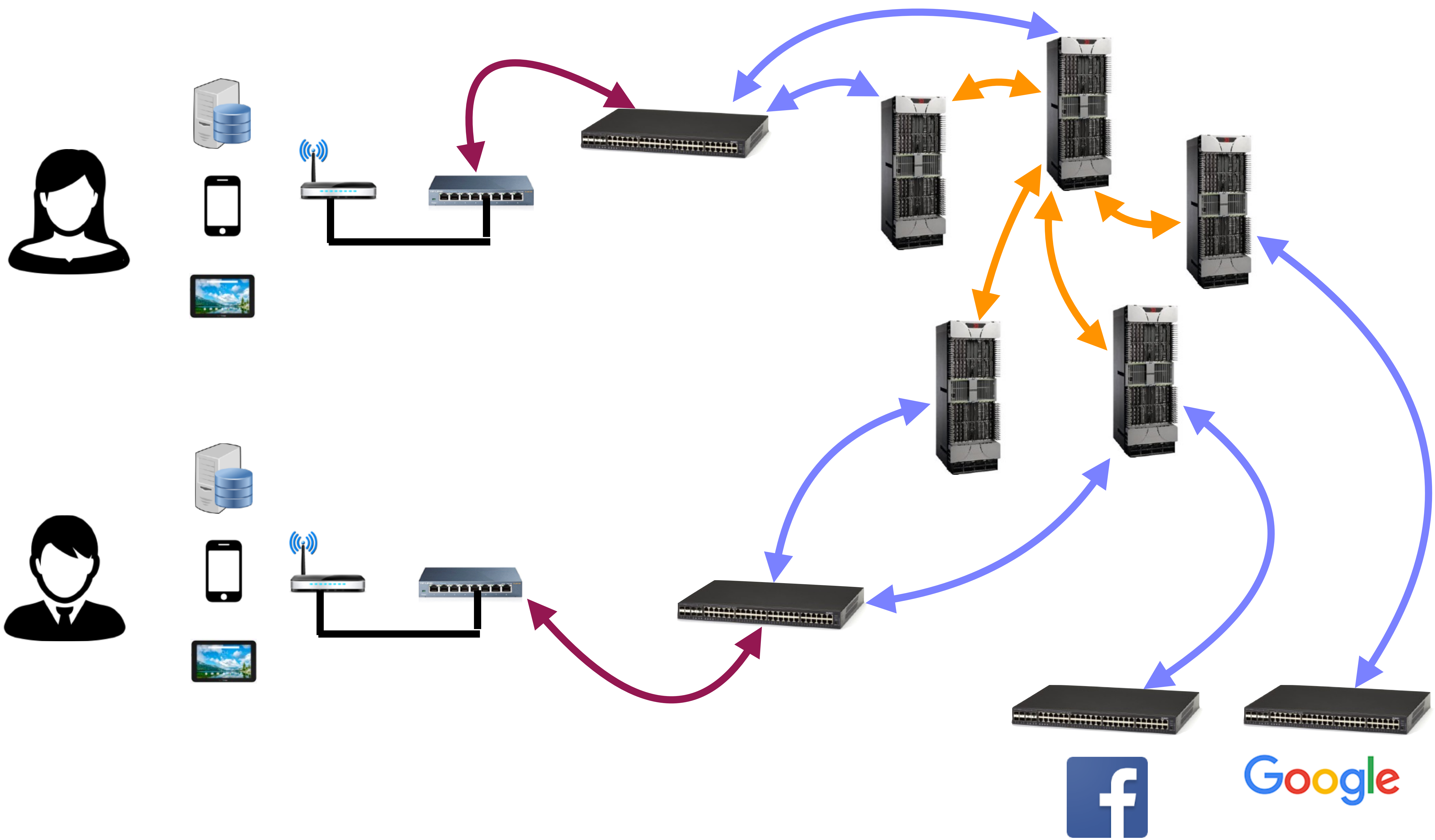
- ICANN gives ARIN several /8s
- ARIN given AT&T one /8, **128.0/8**
  - **Network prefix: 10000000**
- AT&T gives Cornell one /16, **128.84/16**
  - **Network prefix: 10000000 01010100**
- Cornell gives CS one /24, **128.84.139/24**
  - **Network prefix: 10000000 01010100 10001011**
- CS given me a specific address **128.84.139.5**
  - **Network prefix: 10000000 01010100 10001011 00000101**

# How does this meet our requirements?

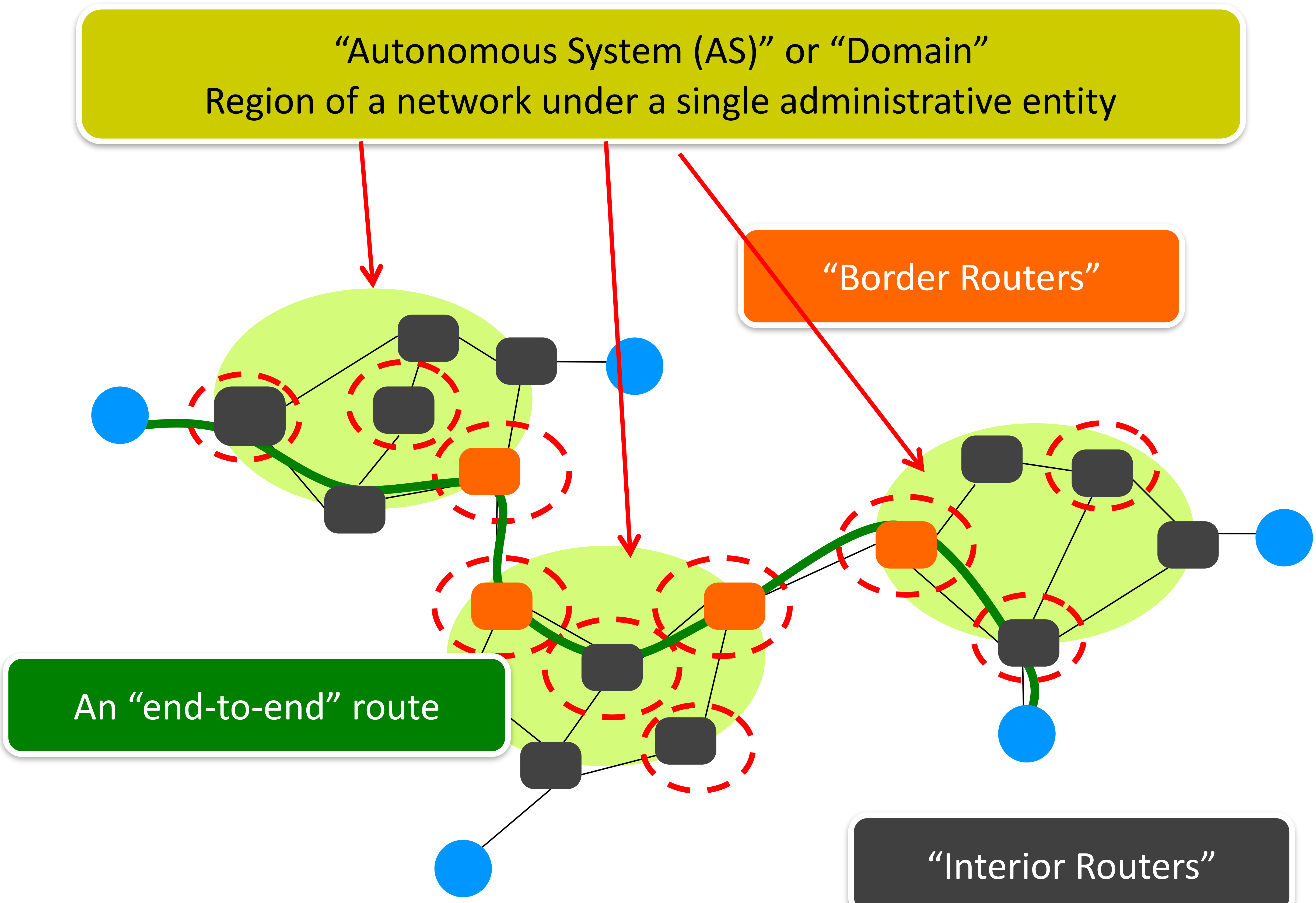
- To understand this, we need to understand the routing on the Internet
- And to understand that, we need to understand the Internet

# Back to the basics: what is a computer network?

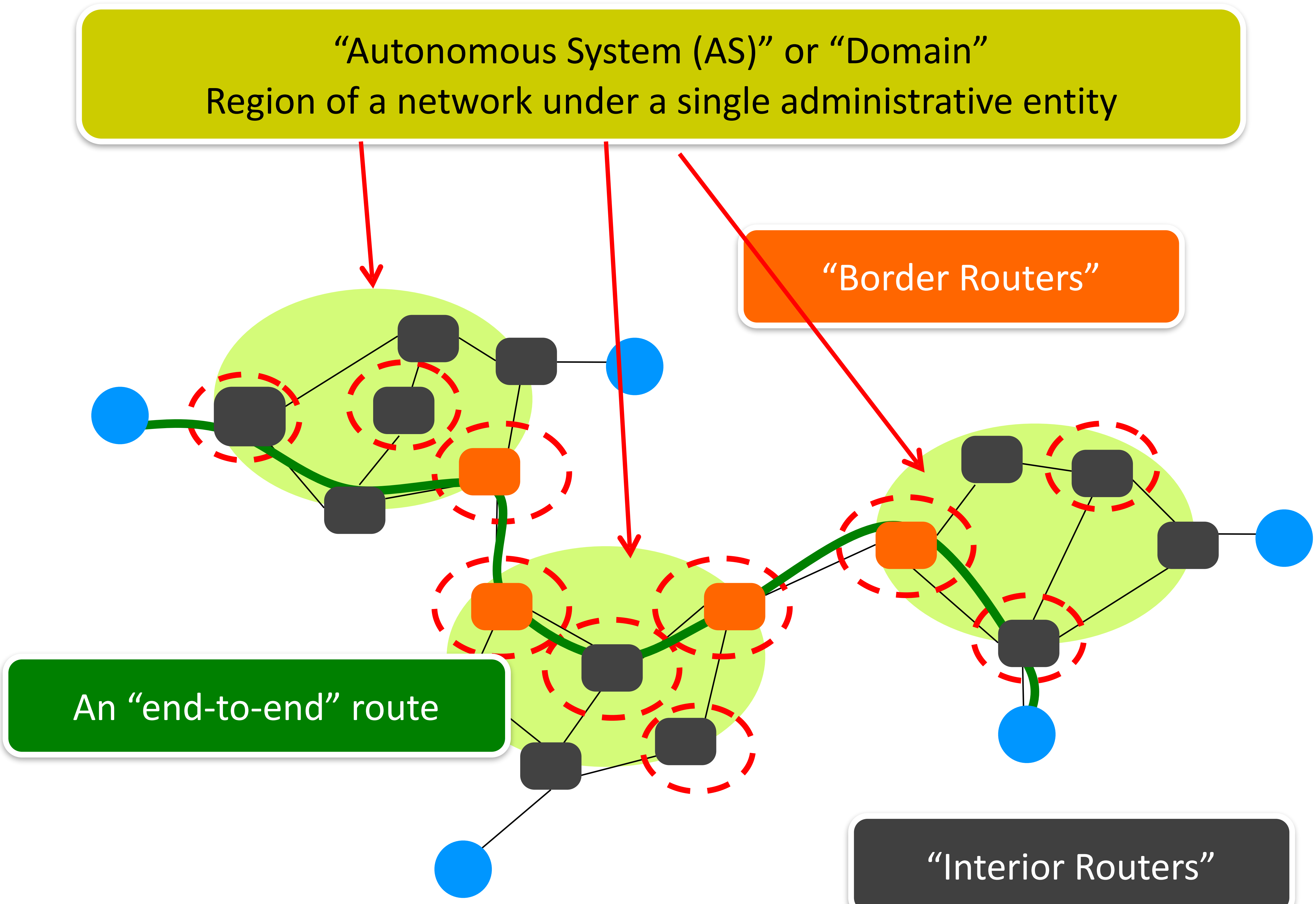
**A set of network elements connected together, that implement a set of protocols for the purpose of sharing resources at the end hosts**



# What does a computer network look like?



# What does a computer network look like?



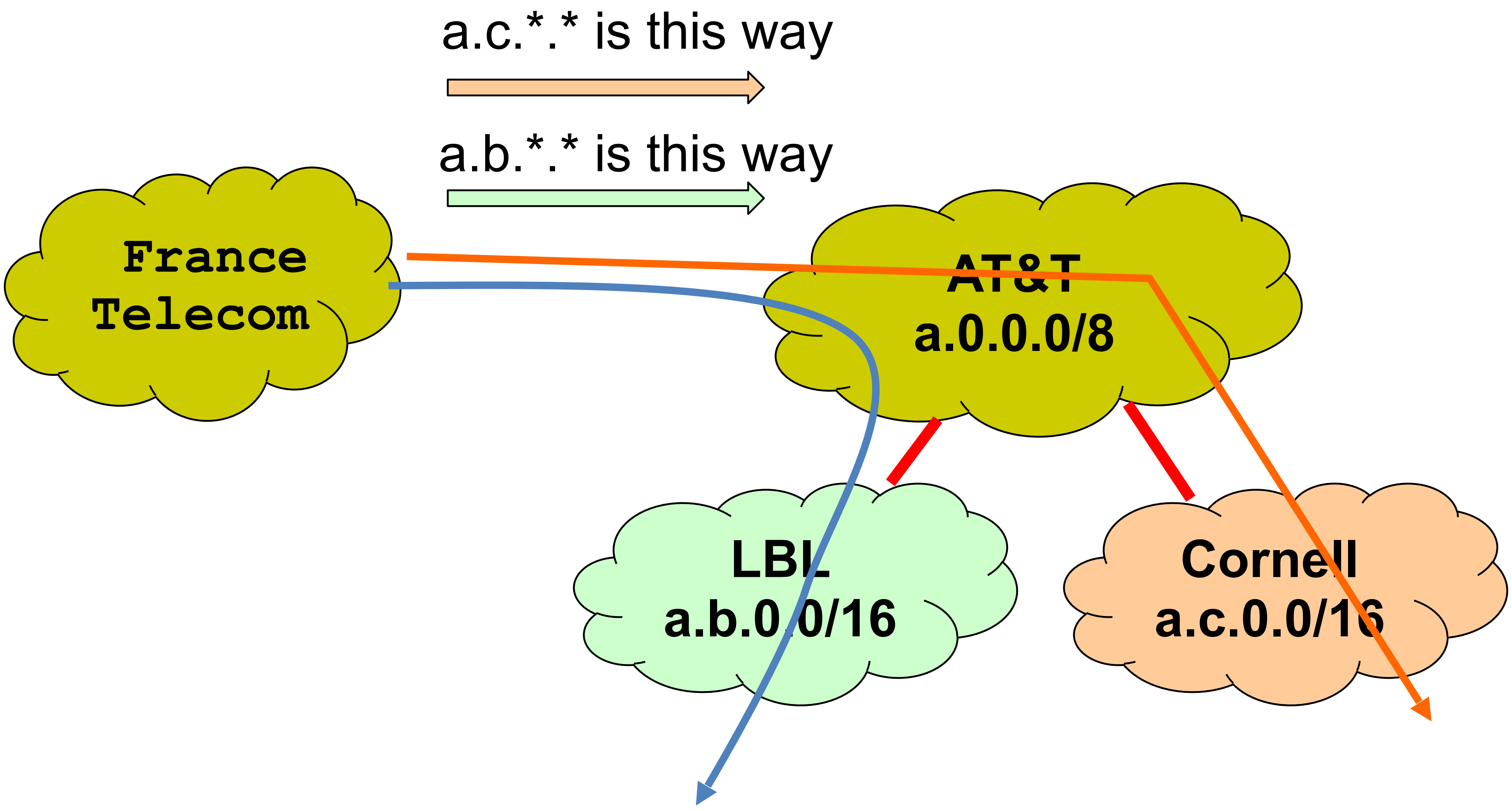
# Autonomous Systems (AS)

- An AS is a network under a single administrative control
  - Currently over 30,000
  - **Example: AT&T, France Telecom, Cornell, IBM, etc.**
  - A collection of routers interconnecting multiple switched Ethernets
  - And interconnections to neighboring ASes

- Sometimes called “Domains”

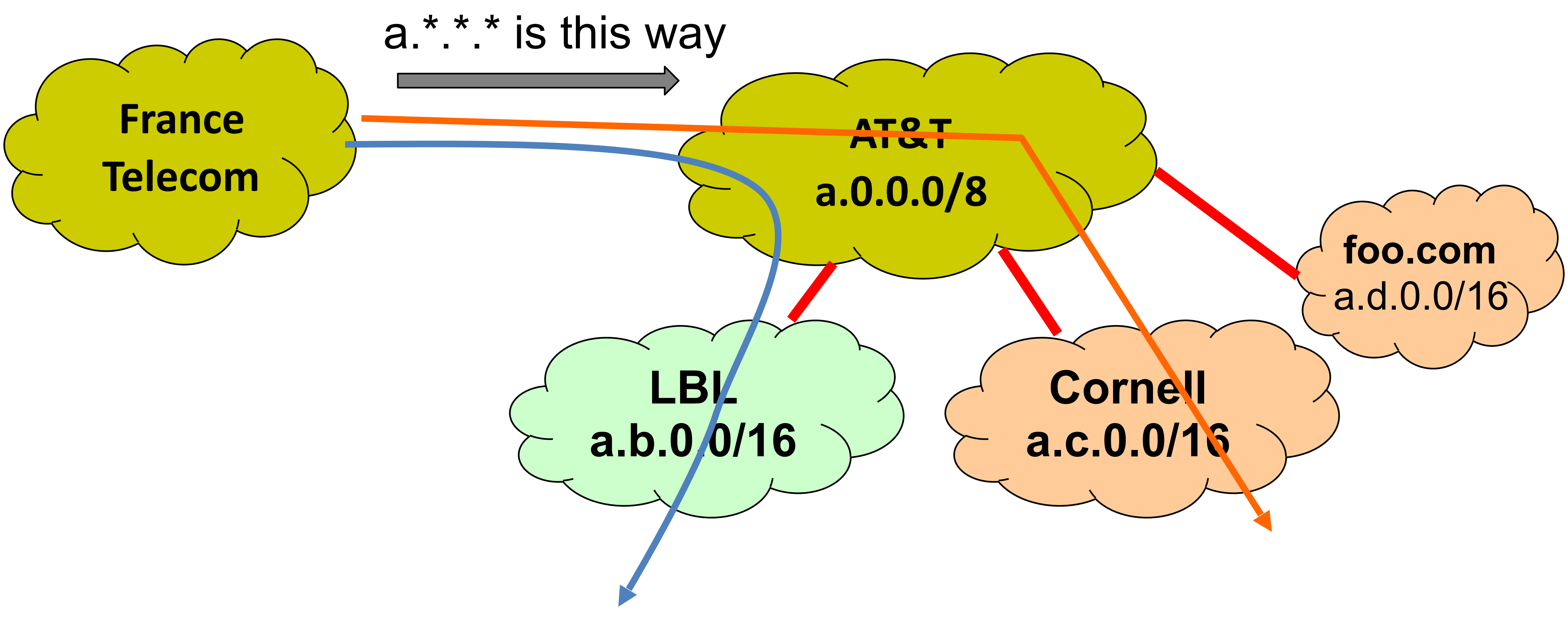
- Each AS assigned a unique identifier
  - **16 bit AS number**

# IP addressing -> Scalable Routing?



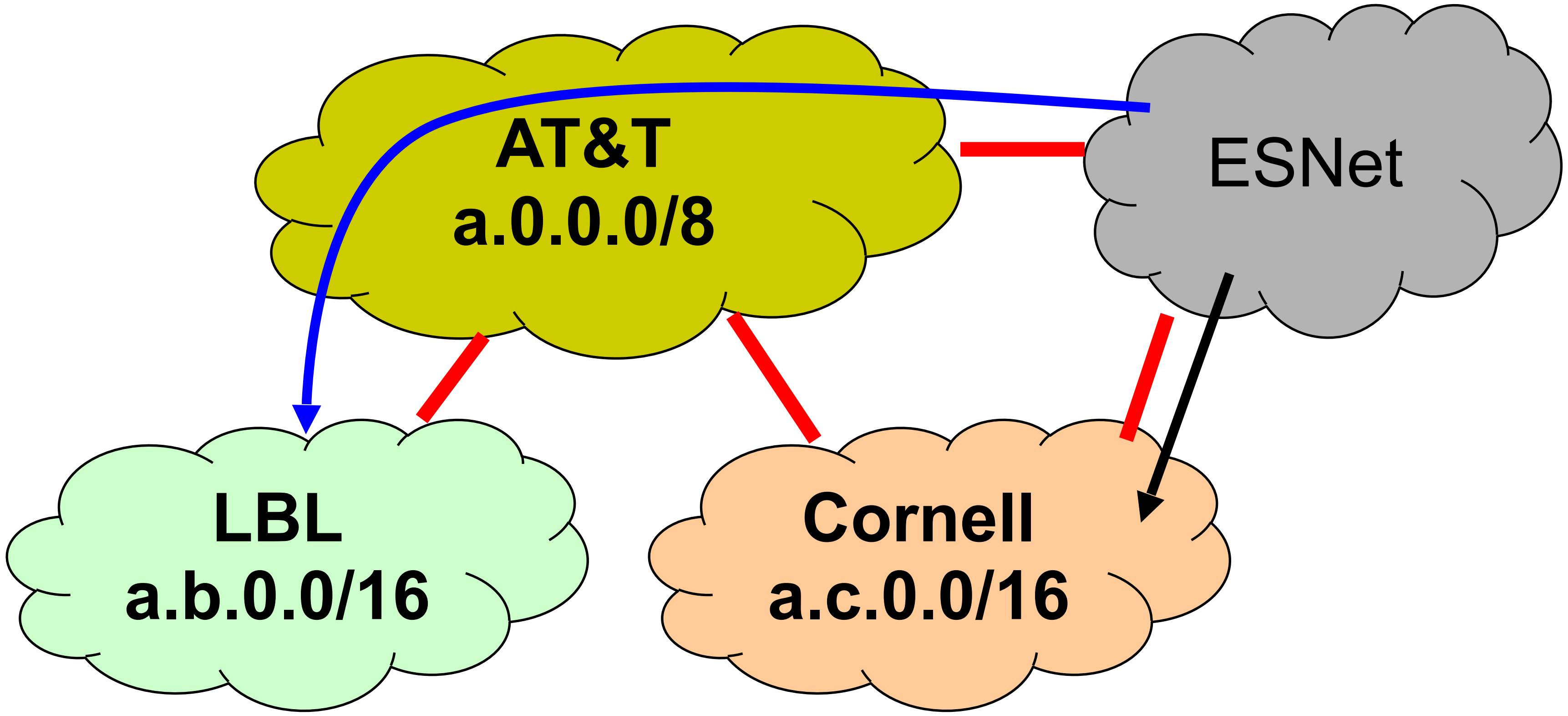
# IP addressing -> Scalable Routing?

Can add new hosts/networks without updating the routing entries at France Telecom



# IP addressing -> Scalable Routing?

ESNet must maintain routing entries for both a.*.*.* and a.c.*.*



# Administrative Structure Shapes Inter-domain Routing

- ASes want freedom to pick routes based on policy
  - *“My traffic can’t be carried over my competitor’s network!”*
  - *“I don’t want to carry A’s traffic through my network!”*
  - Cannot be expressed as Internet-wide “least cost”

- ASes want autonomy
  - Want to choose their own internal routing protocol
  - Want to choose their own policy

- ASes want privacy
  - Choice of network topology, routing policies, etc.

# Choice of Routing Algorithm

- Link State (LS) vs. Distance Vector (DV)

- LS offers no privacy — broadcasts all network information
- LS limits autonomy — need agreement on metric, algorithm

- DV is a decent starting point
  - Per-destination updates by intermediate nodes give us a hook
  - But, wasn't designed to implement policy
  - … and is vulnerable to loops if shortest paths not taken

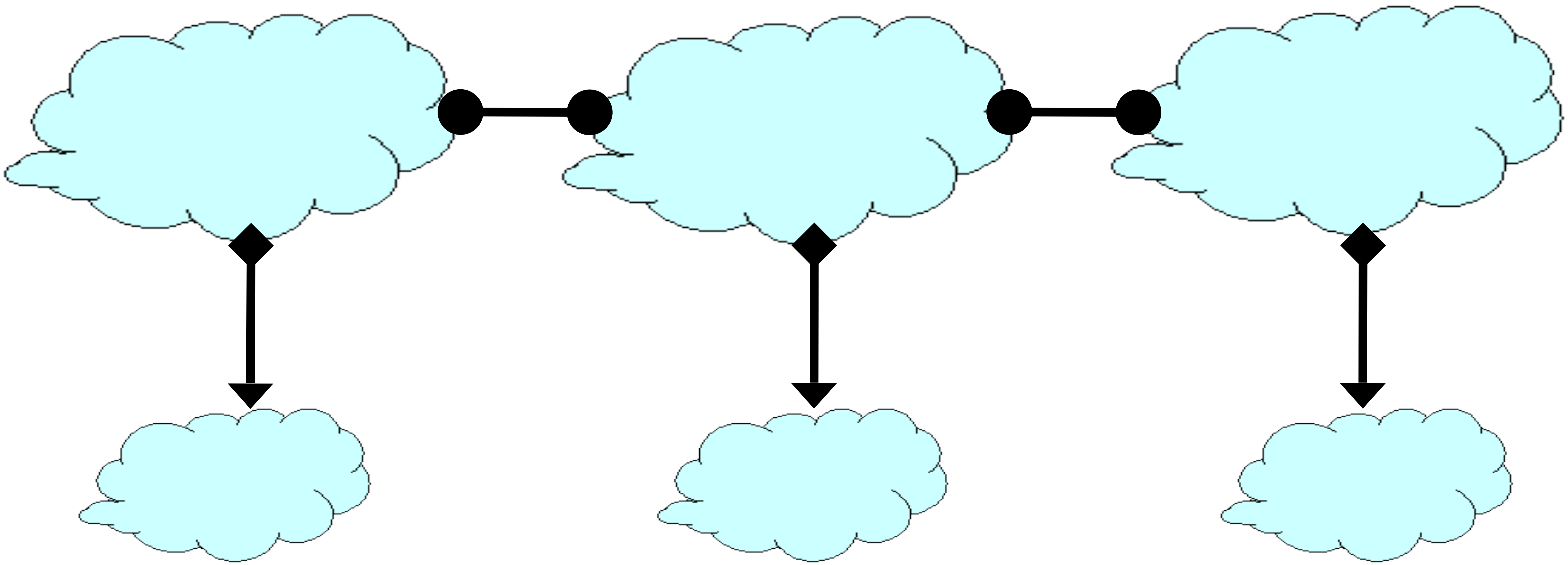
**The "Border Gateway Protocol" (BGP) extends Distance-Vector ideas to accomodate policy**

# Business Relationships Shape Topology and Policy

- Three basic kinds of relationships between ASes
  - AS A can be AS B's *customer*
  - AS A can be AS B's *provider*
  - AS A can be AS B's *peer*

- Business implications
  - Customer pays provider
  - Peers don't pay each other
    - Exchange roughly equal traffic

# Business Relationships



*Relations between ASes*

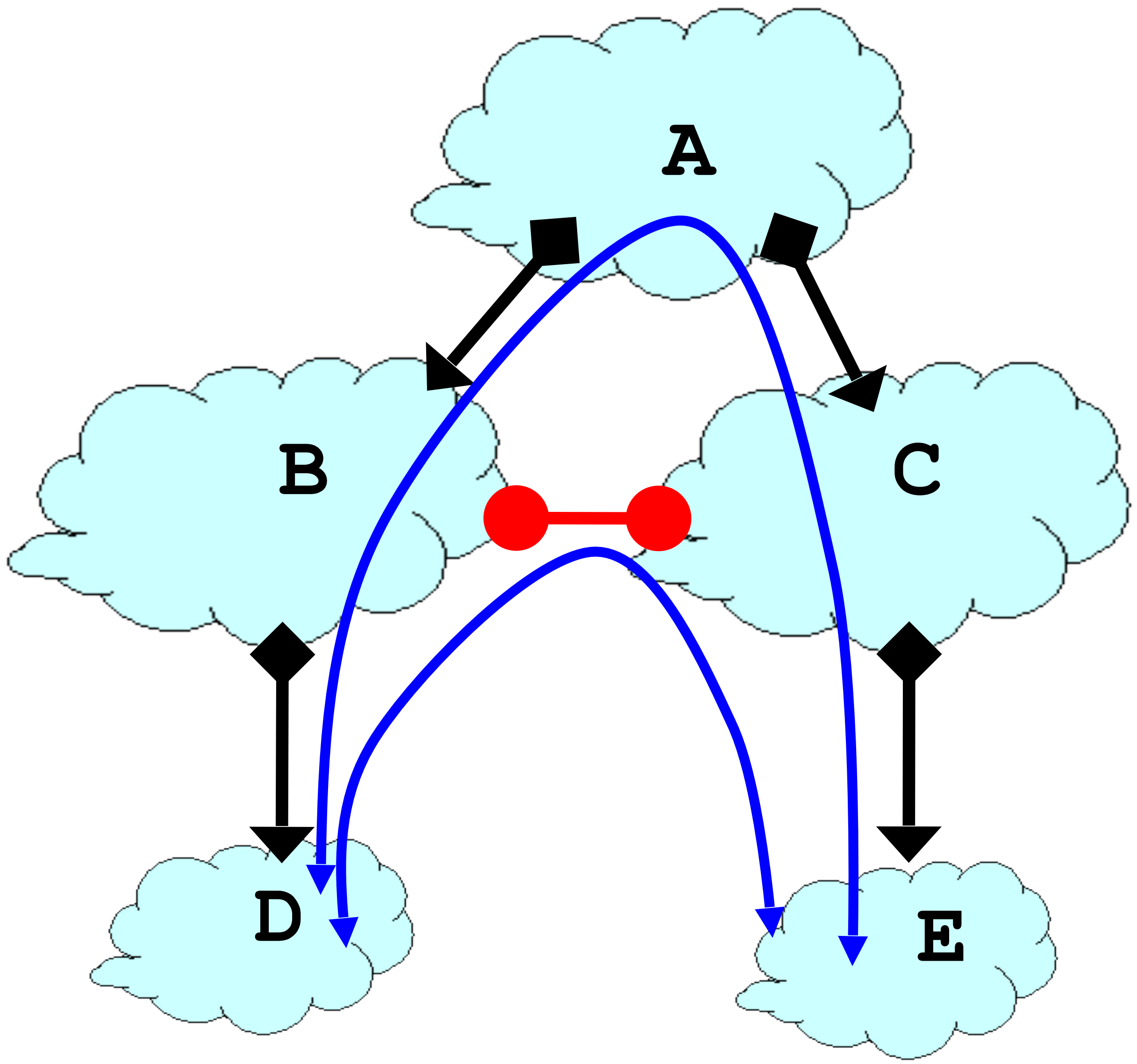
provider ◆———▶ customer

peer ●———● peer

*Business Implications*

- **Customers pay provider**
- **Peers don't pay each other**

# Why Peer?




**E.g., D and E talk a lot**

**Peering saves B *and* C money**

*Relations between ASes*

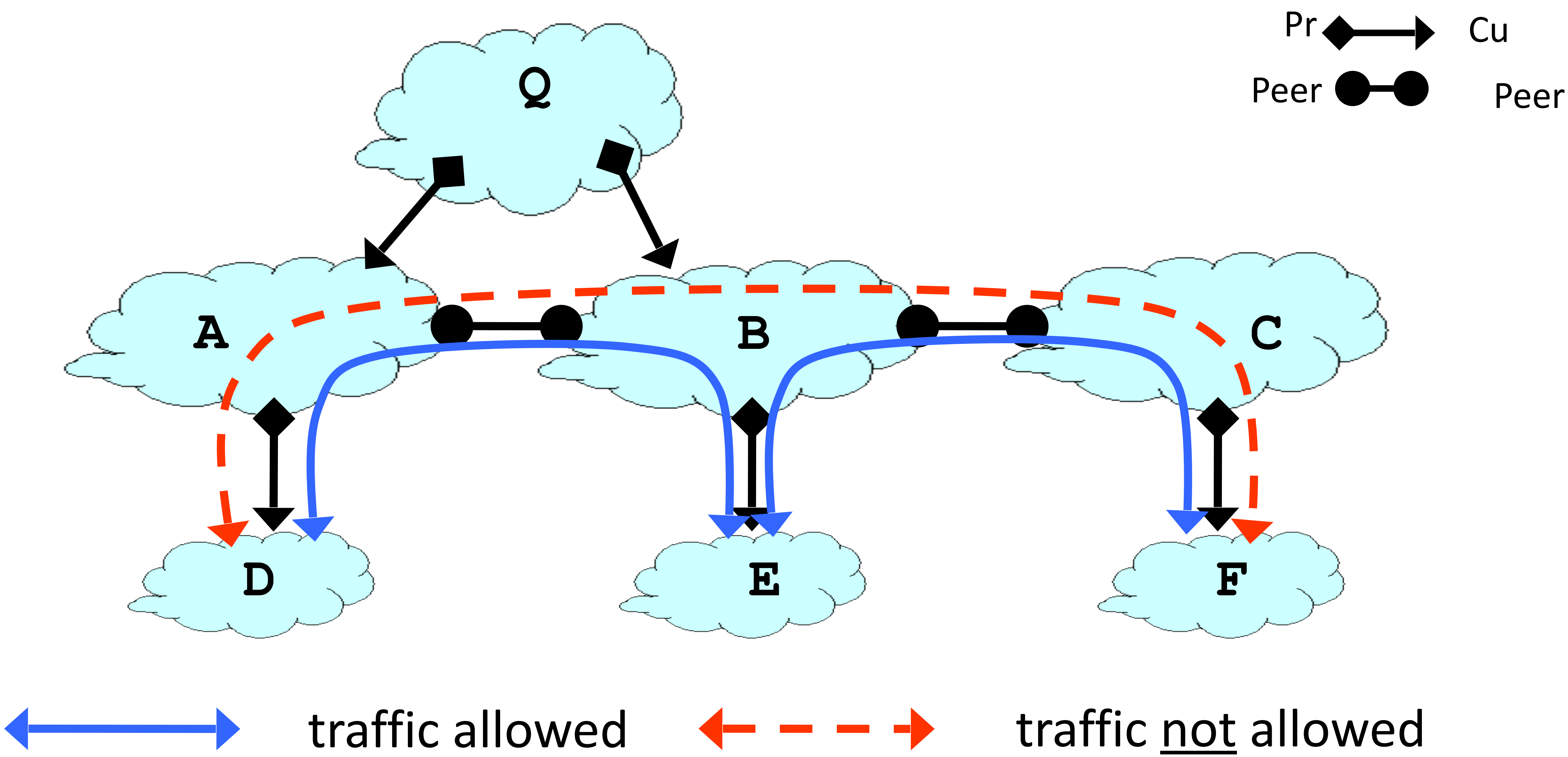
**provider → customer**

**peer — peer**

*Business Implications*

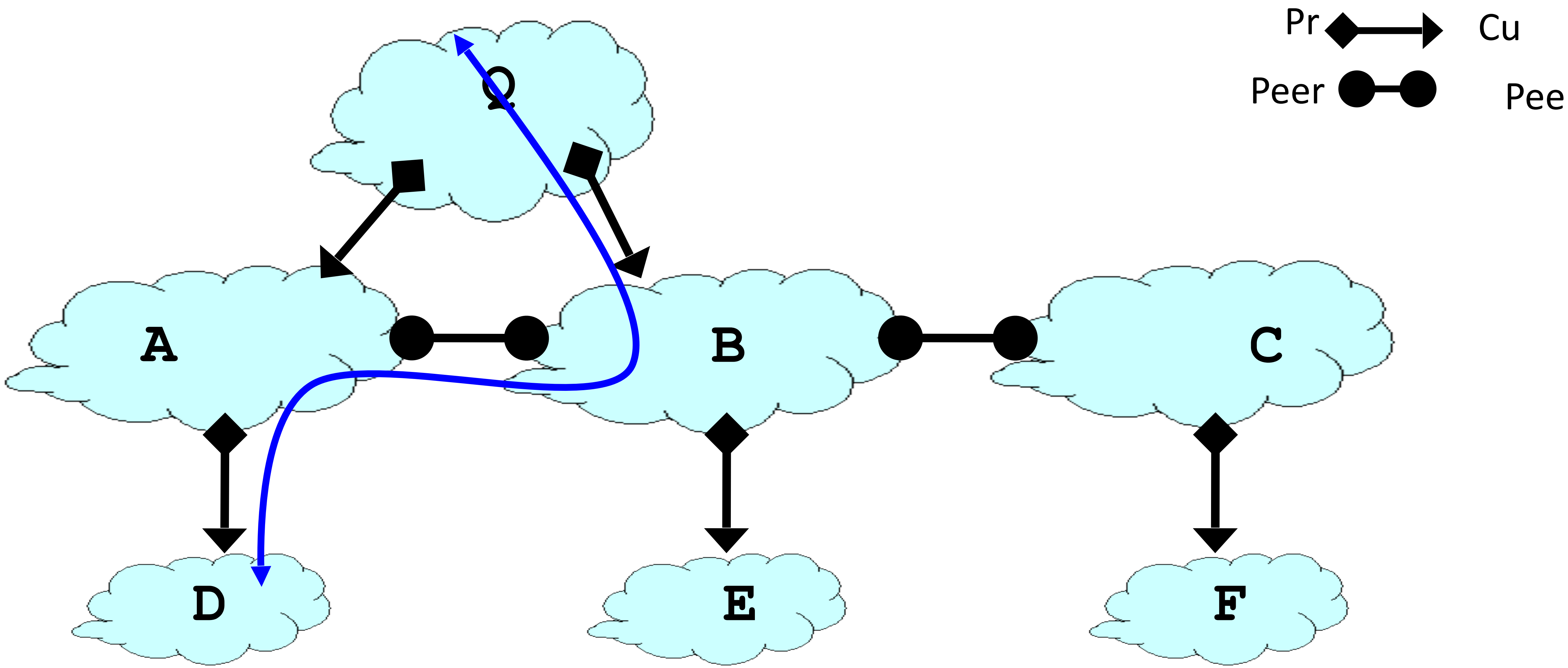
- **Customers pay provider**
- **Peers don't pay each other**

# Routing Follows the Money




- ASes provide "transit" between their customers
- Peers do not provide transit between other peers

# Routing Follows the Money




- An AS only carries traffic to/from its own customers over a peering link

# Inter-domain Routing: Setup

- Destinations are IP prefixes (12.0.0.0/8)

- Nodes are Autonomous Systems (ASes)
  - Internals of each AS are hidden

- Links represent both physical links and business relationships

- BGP (Border Gateway Protocol) is the Interdomain routing protocol
  - Implemented by AS border routers

# BGP

An AS advertises
its best routes
to one or more IP prefixes

Each AS selects the
“best” route it hears advertised for
a prefix

**Sound familiar?**

# BGP Inspired by Distance Vector

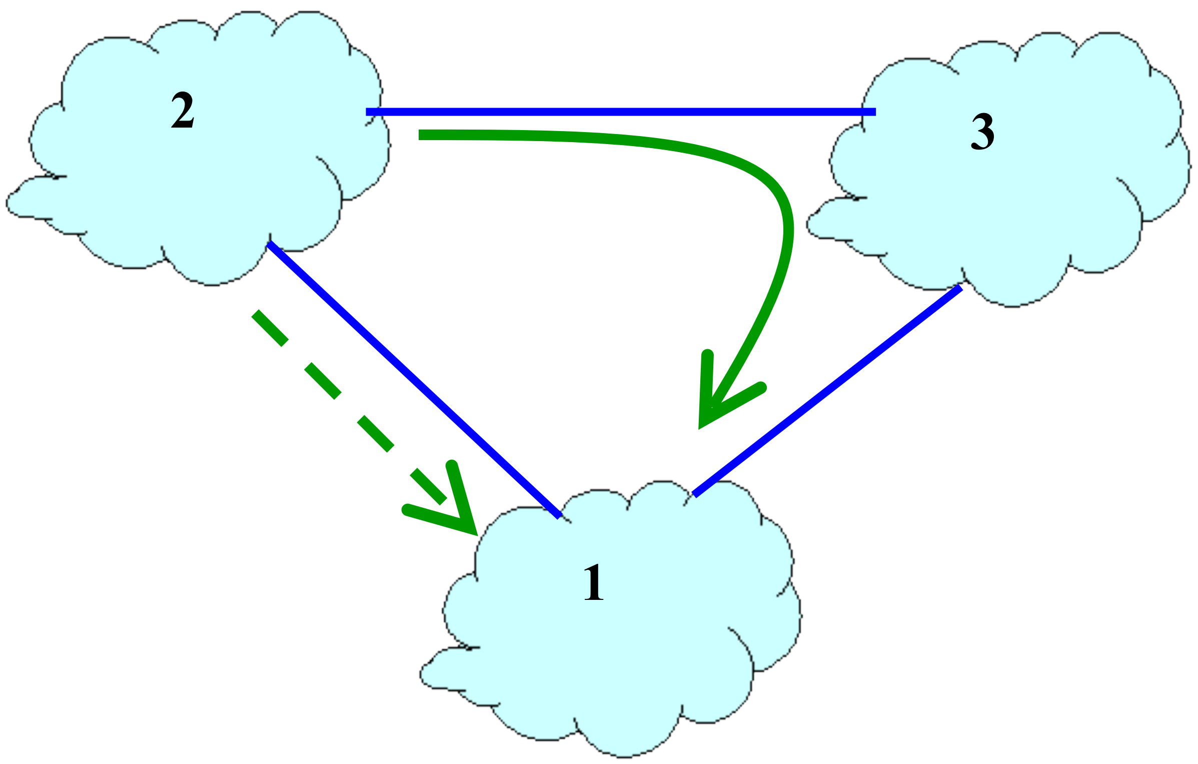
- Per-destination route advertisements
- No global sharing of network topology
- Iterative and distributed convergence on paths
- But, four key differences

# BGP vs. DV

## (1) BGP does not pick the shortest path routes!

- BGP selects route based on policy, not shortest distance/least cost

Node 2 may prefer 2, 3, 1
over 2, 1

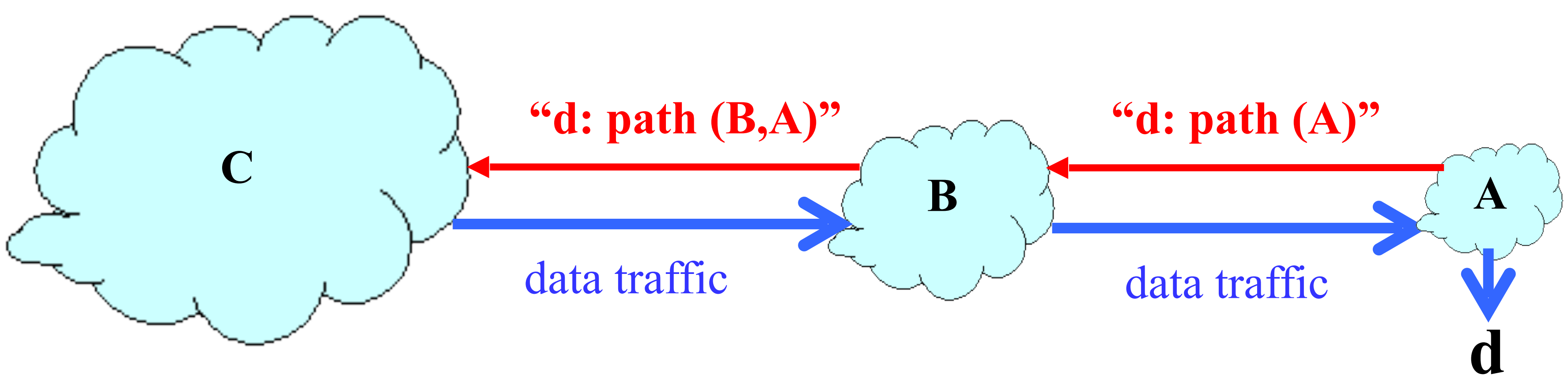



- How do we avoid loops?

# BGP vs. DV

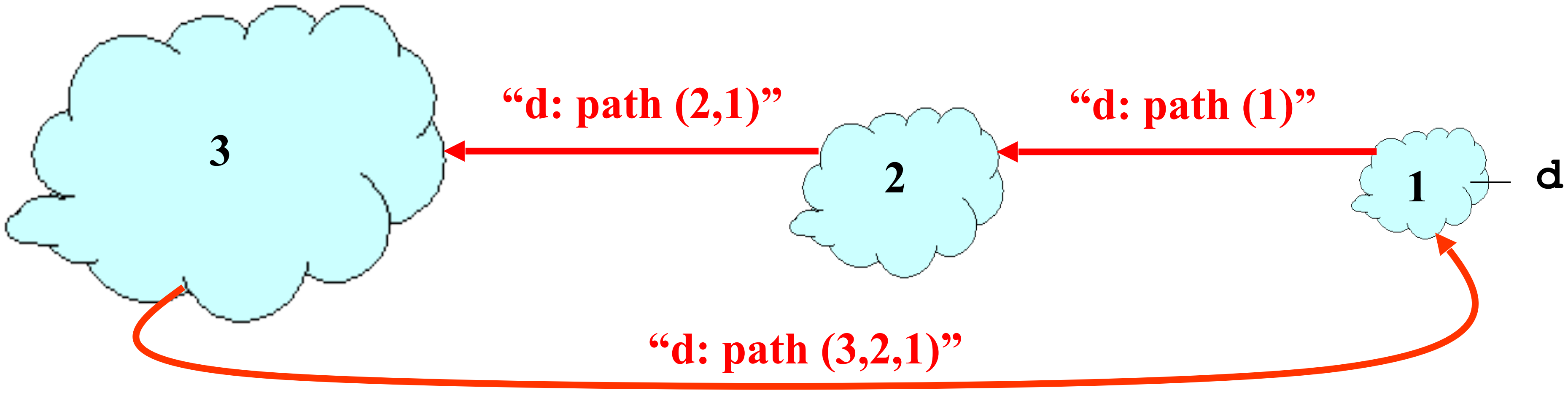
## (2) Path-vector Routing

- Idea: advertise the entire path
  - Distance vector: send *distance metric* per dest. d
  - Path vector: send the *entire path* for each dest. d



# Loop Detection with Path-Vector

- Node can easily detect a loop
  - Look for its own node identifier in the path
- Node can simply discard paths with loops
  - e.g. node 1 sees itself in the path 3, 2, 1



# BGP vs. DV

## (2) Path-vector Routing

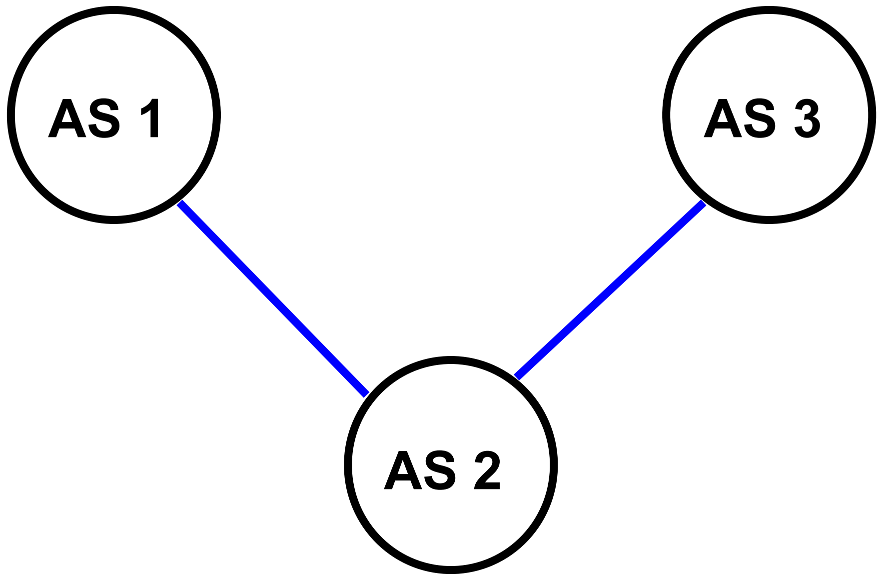
- Idea: advertise the entire path
  - Distance vector: send *distance metric* per dest. d
  - Path vector: send the *entire path* for each dest. d

- Benefits
  - Loop avoidance is easy
  - Flexible policies based on entire path

# BGP vs. DV

## (3) Selective Route Advertisement

- For policy reasons, an AS may choose not to advertise a route to a destination
- As a result, reachability is not guaranteed even if the graph is connected

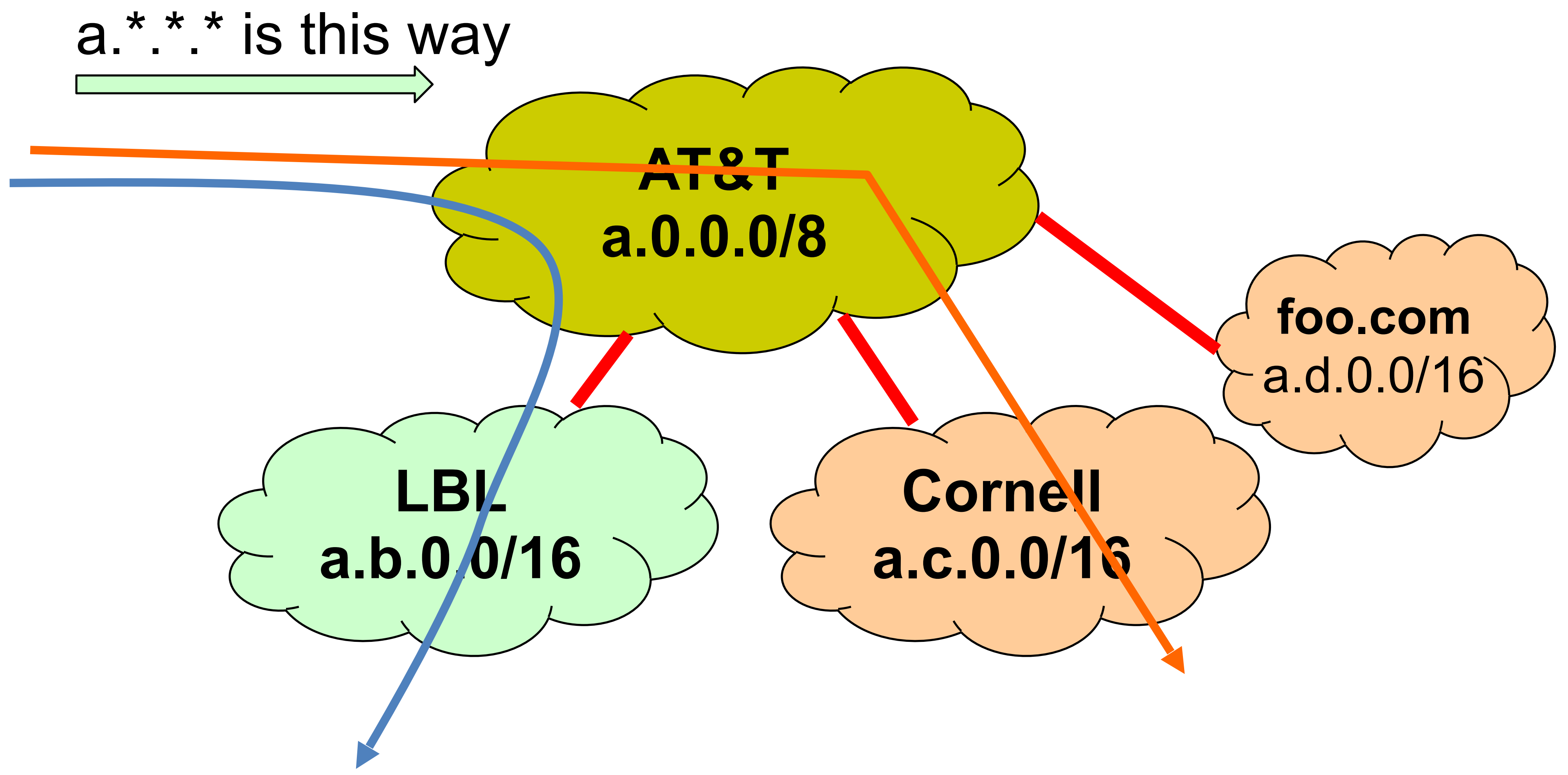



Example: AS#2 does not want to carry traffic between AS#1 and AS#3

# BGP vs. DV

## (4) BGP may aggregate routes

- For scalability, BGP may aggregate routes for different prefixes

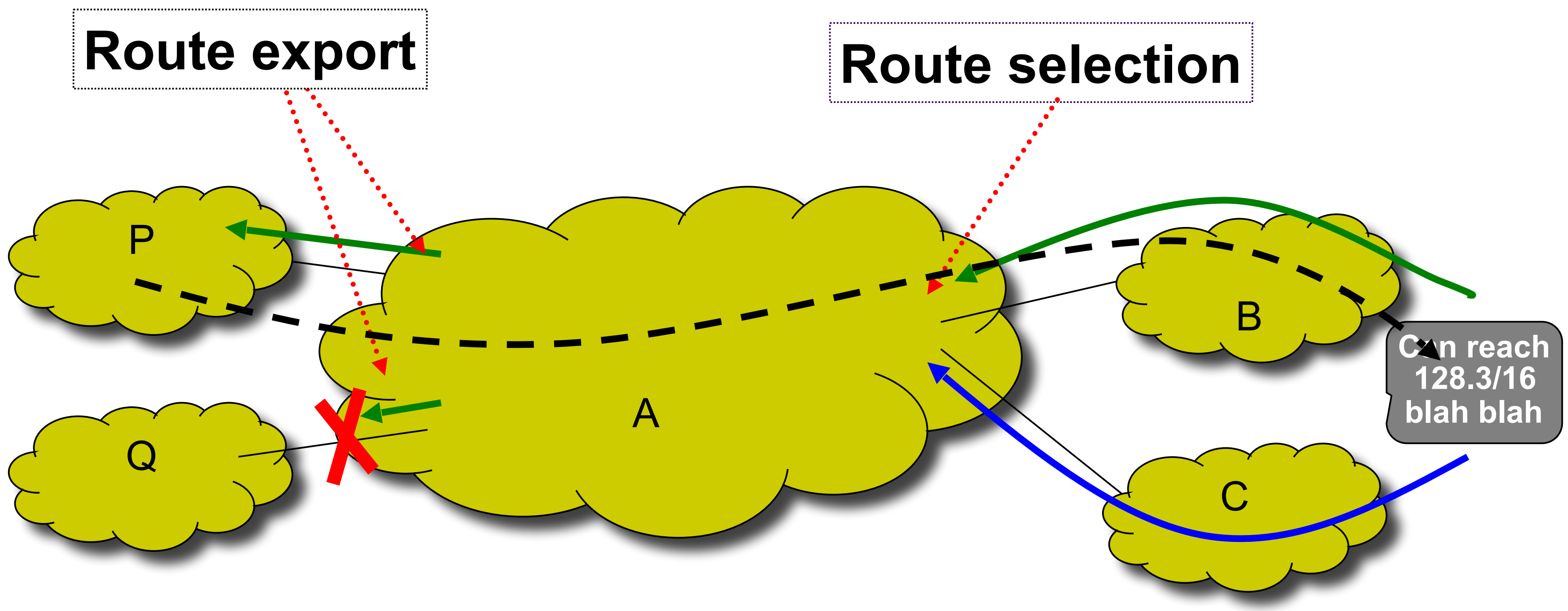


# BGP Outline

- BGP Policy
  - Typical policies and implementation
- BGP protocol details
- Issues with BGP

# Policy:

## Imposed in how routes are selected and exported




- **Selection**: Which path to use
  - Controls whether / how traffic leaves the network
- **Export**: Which path to advertise
  - Controls whether / how traffic enters the network

# Typical Selection Policy

- In decreasing order of priority:
  1. Make or save money (send to customer > peer > provider)
  2. Maximize performance (smallest AS path length)
  3. Minimize use of my network bandwidth (“hot potato”)
  4. …

# Typical Export Policy

| Destination prefix advertised by… | Export route to… |
|---|---|
| Customer | Everyone (providers, peers, other customers) |
| Peer | Customers |
| Provider | Customers |

Known as the "Gao-Rexford" rules
Capture common (but not required!) practice