# CS4450

## Computer Networks: Architecture and Protocols

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

**Rachit Agarwal**

# Announcements

- **Problem set 3 solutions posted**
- **Project 2 posted**
- **Discussion on ongoing epidemic:**
    - **We will follow university policy word-by-word**
    - I understand that this may be tough:
        - Concerns about your health, about family, about friends …
        - I promise to go above and beyond: **anything** that you need …
- Moving target

# What have we covered so far

- **Sharing networks**
  - Circuit switching, packet switching, **why packets**
  - **Transmission delays, Propagation delays, queueing delays**

- **Architectural Principles and design goals**
  - **layering, end-to-end, fate sharing**
  - Internet design goals

- **Link Layer**
  - MAC names, broadcast medium (classical), sharing broadcast medium
  - Frames and framing frames
  - Random access on broadcast medium: **CSMA/CD**
  - Scalability Issues of broadcast medium: **why switched Ethernet**
  - **Spanning Tree Protocol**

- **Network Layer ….**

# What have we covered so far

- **…. Network Layer**
  - **Why do we need network layer?**
  - **The right way to think about routing tables and protocols**
  - Shortest-path routing protocols
    - Link state routing protocol
    - **Distance-vector routing protocol**
      - **Transient loops,** dead ends
      - ….. (just a bit more in next 2-3 lectures)

- **Sockets and Ports**
  - Creating connections

# Goals for Today's Lecture

- **Internet Addressing**

- Begin Inter-domain routing (Border-Gateway Protocol (BGP))

# 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 names
    - Unique identifiers hardcoded in hardware; no location information

- Local area networks route on these “flat” addresses
    - **Spanning Tree Protocol runs on switches**
    - Each switch stores a routing entry for each host connected to it
    - End-hosts store nothing

- Upon receiving a frame, an end-host:
    - Puts destination’s and its own MAC name in the header
    - Forwards it to the switch it is connected to

- **Switches forward the frame along edges in spanning tree**

- **Destination is able to recognize the frame is for them using address**

# If we were to use “Flat” Addressing on Internet …

- Ethernet routes frames via flooding on Spanning Tree
    - As we have discussed: cannot use it on entire Internet

- One possible way to use flat addresses on Internet
    - **Routing tables store one entry per end-host**
    - End-hosts store nothing

- Upon receiving a packet, an end-host:
    - Puts destination’s and its own flat address in the header
    - Forwards it to the switch it is connected to

- **Switches forward the packet using routing tables (no flooding!)**

- **Destination is able to recognize the frame 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 flat addresses (exact match is easy!)

- **Host must be able to recognize the packet is for them**
  - Easy

**Conclusion: L2 addressing does not enable scalable routing**

# How would you scale L2 addressing?

- Suppose we want to design a much larger network using flat addresses

- 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)

# Scalable L2 addressing: Towards Internet-scale addressing

- Assign each end-host an addresses of the form — **Switch.MAC**

- **Routing tables store 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**

**Better, but still not feasible**

**(Internet has billions of switches/routers)**

# Layer 3: Hierarchical addressing

- Define a collection of switches/routers to be a “Network”
- Use addresses of the form — **Network.MAC**
- Switches/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 MAC part of the address
    - When the packet reaches the right “network”
        - Packet forwarded using MAC 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 solution **Switch.MAC**: aggregate was **switch**
    - In our scalable solution **Network.MAC**: 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

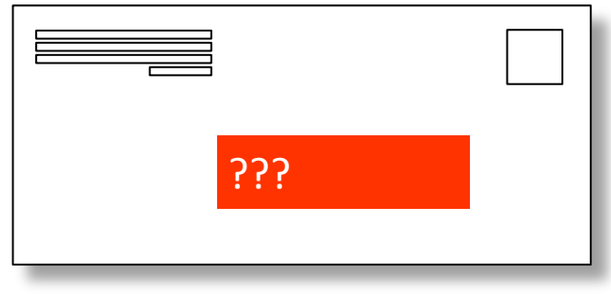
- The Internet is an “inter-network”
    - Used to connect networks together, not hosts

- 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
- How many networks can we support using 8 bits?
    - 256!
- 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 |

# 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 — **101.*.*.*, 128.*.*.*, ….**

- ARIN given AT&T one /8 — **128.*.*.*/8 (can support 2^24 hosts)**
    - **Network prefix: 10000000**

- AT&T gives Cornell one /16 — **128.84.*.*/16 (supports 2^16 hosts)**
    - **Network prefix: 10000000 01010100**

- Cornell gives CS one /24 — **128.84.139.*/24 (supports 2^8 hosts)**
    - **Network prefix: 10000000 01010100 10001011**

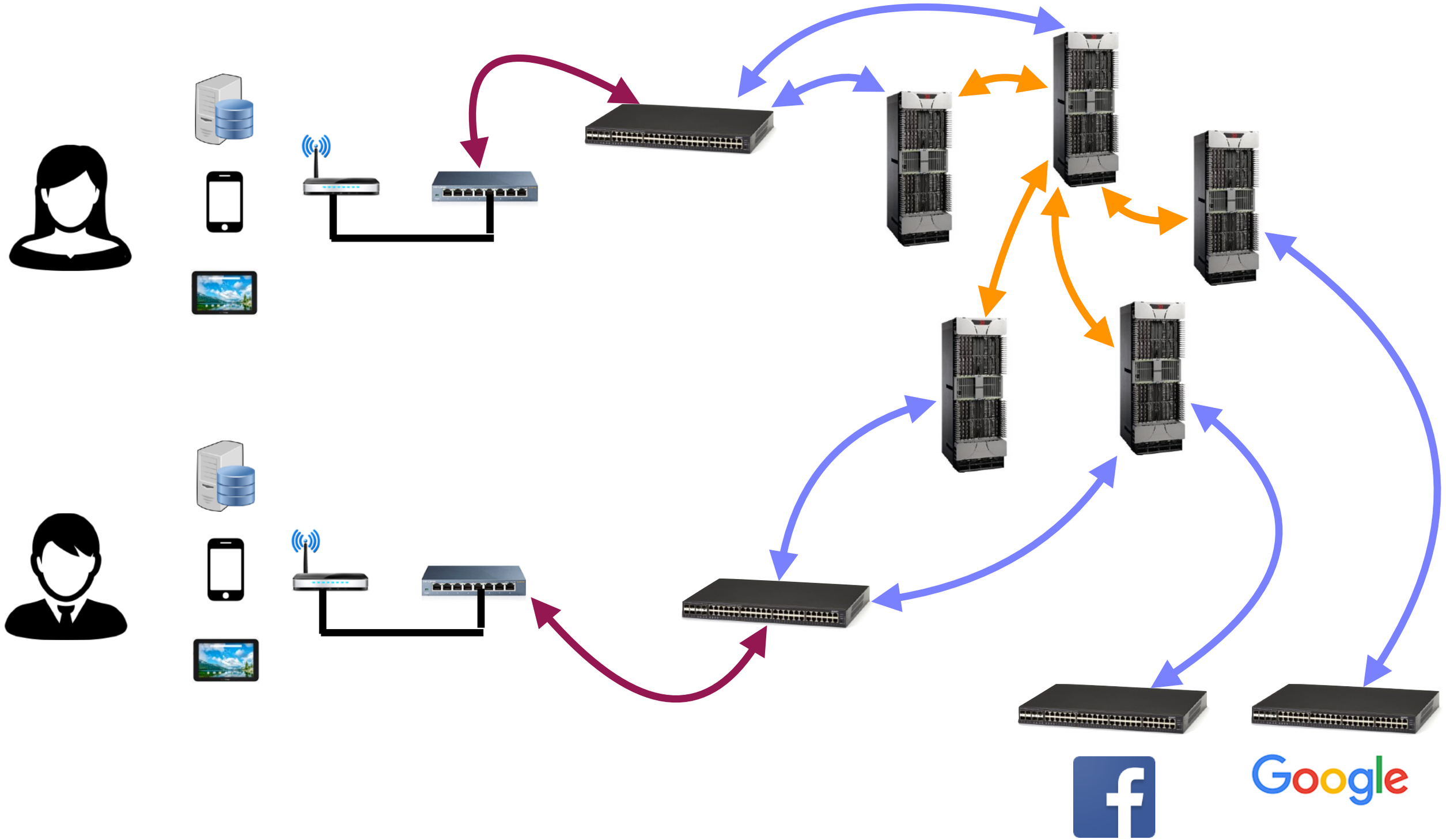
- CS given me a specific address — **128.84.139.5**
    - **IP address: 10000000 01010100 10001011 00000101**

# How does this meet our requirements?

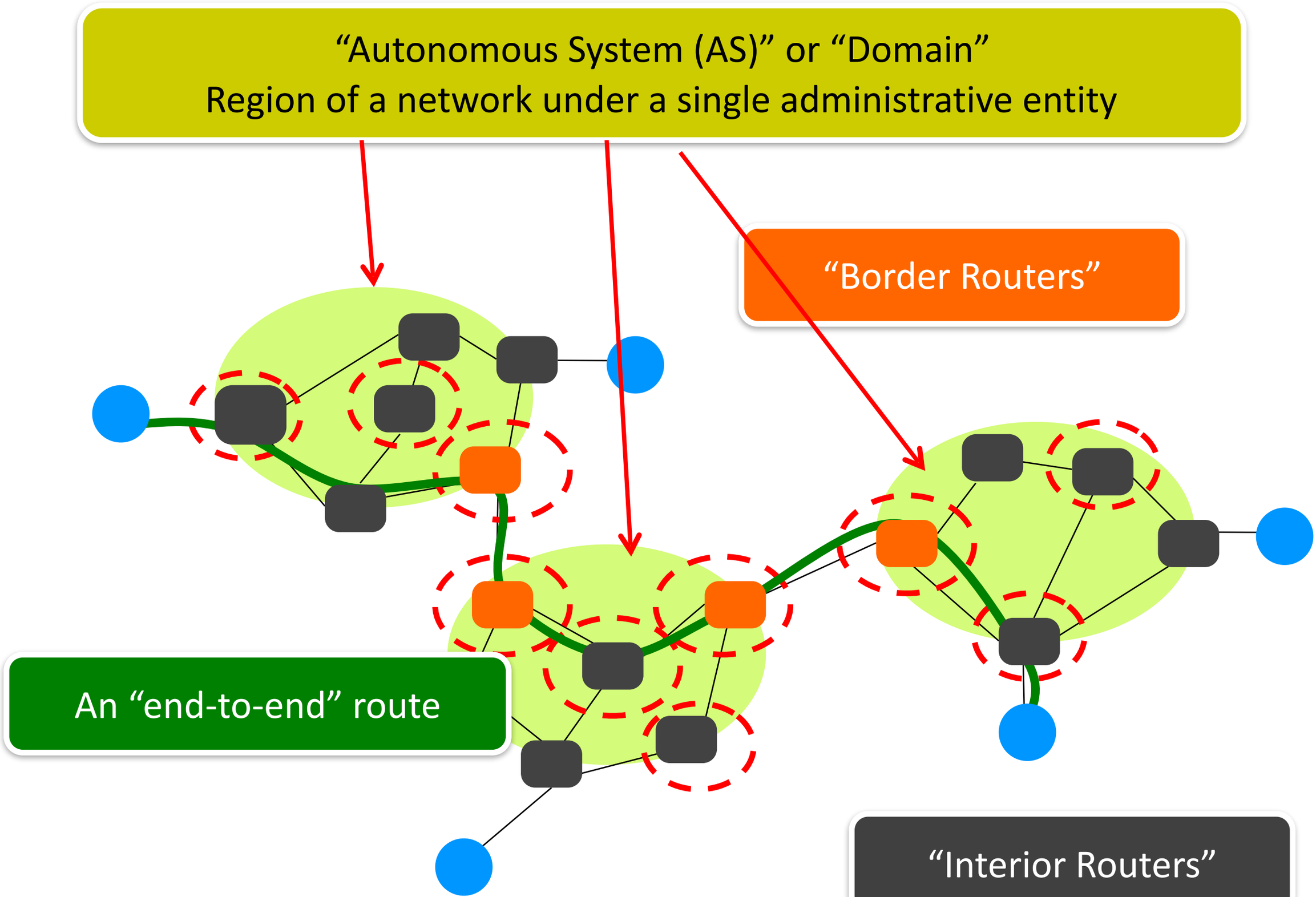
- To understand this, we need to understand the routing on the Internet
- And to understand that, we need to understand the Internet

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

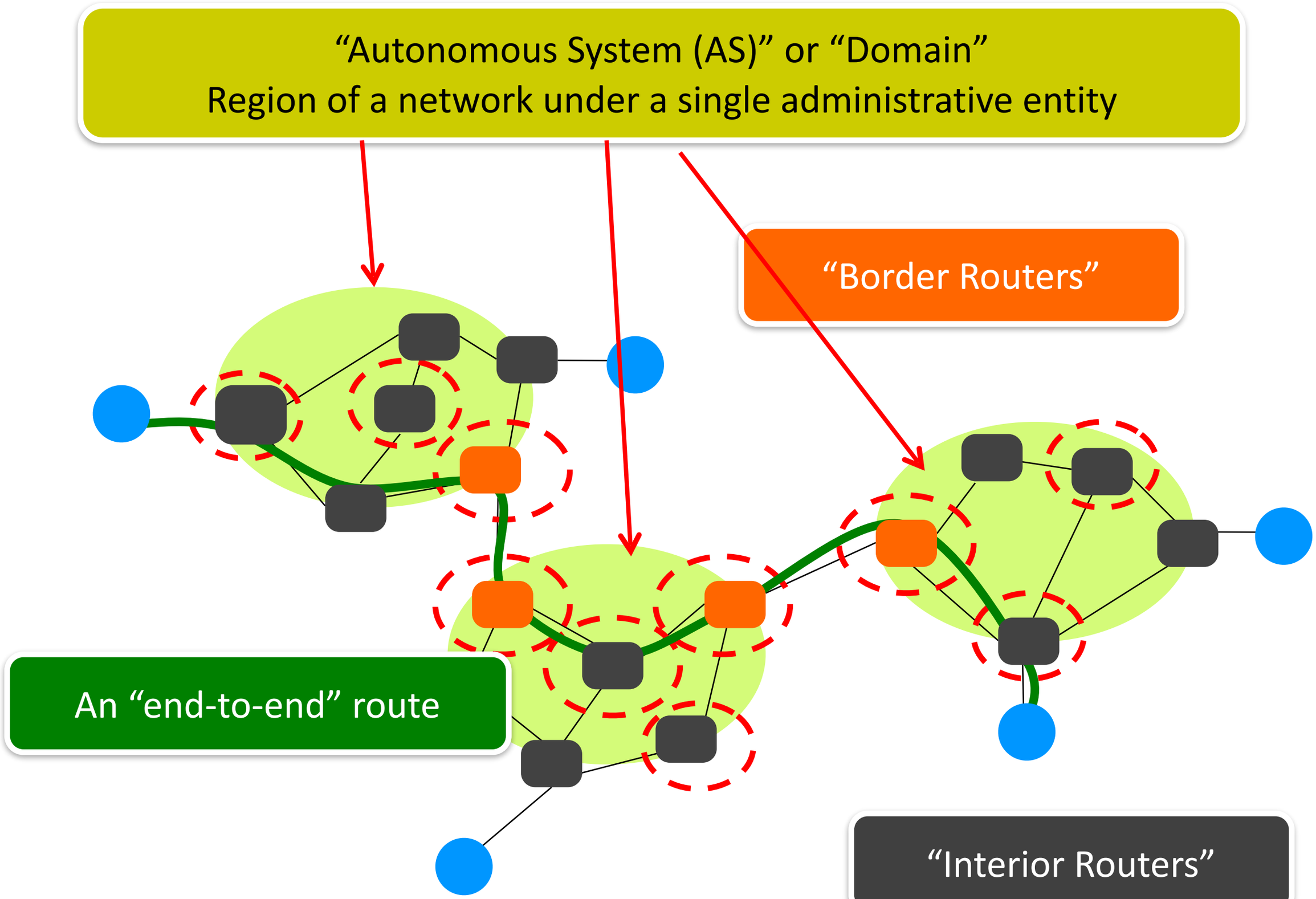
**A set of network elements connected together, that implement a set of protocols for the purpose of sharing resources at the end hosts**



# What does a computer network actually 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**