Midterm Review
CS168

Sylvia Ratnasamy

Fall 2022

Coverage

- Everything up to and including Lecture 11 & up to slide#21 from Lecture 12
  - i.e., everything up to Reliable Transport

- Test only assumes material covered in lecture & sections
  - Text: only to clarify details and context for the above
  - If the text disagrees with lecture, go with what we said in lecture

- The test doesn't require you to do complicated calculations
  - Use this as a hint to whether you are on the right track

General Guidelines (1)

Be prepared to contemplate new designs we haven't talked about

- e.g., *here's a new BGP policy ...*

Be prepared to analyze how the designs we've discussed behave in new scenarios

- e.g., *here's a strange topology, what happens when STP ...*

We're testing that you understand the material, not just remember it...

Don't let this daunt you! Reason from what you know about the concepts we did study

General Guidelines (2)

Exam format

- Start with a set (~25) of “quick questions”
- Q2+ : questions that dive deeper on specific topics
- If you’re struggling with a particular question, move on and return to it later

Attitude

- Do not panic if you don't know something
  - We don't expect anyone is going to get 100% (none of us did!)
- Some questions may appear surprising:
  - You know all the relevant pieces
  - But haven't put them all together
- Stay calm, and just reason yourself through it

This Review

- **Walk through what we expect you to know**
  - Concepts and details
- **Just because I didn't cover it, doesn't mean you don't need to know it**
  - But if I covered it today, you should know it
  - Use this slide deck as an all-in-one check list
- **My plan: summarize, not explain**
  - Stop me when you want to discuss something in more detail!

Start with links

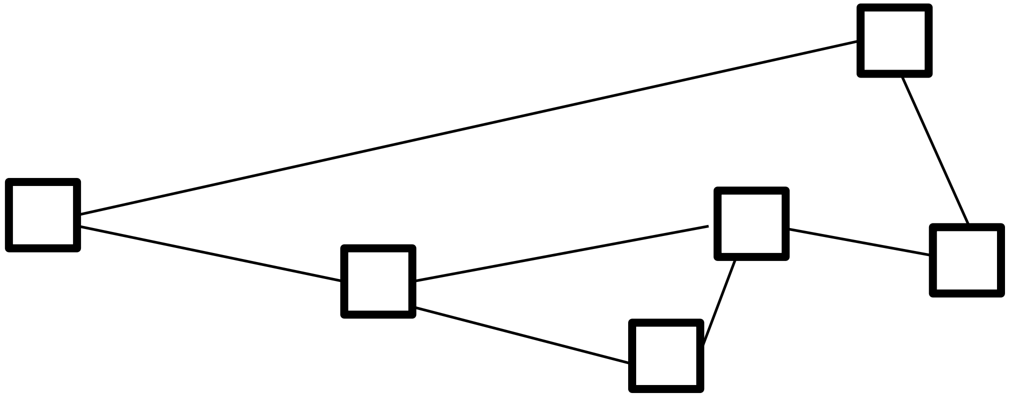
You should know about the key properties that characterize a link:

- Bandwidth
- Propagation delay
- Bandwidth-delay product (BDP)

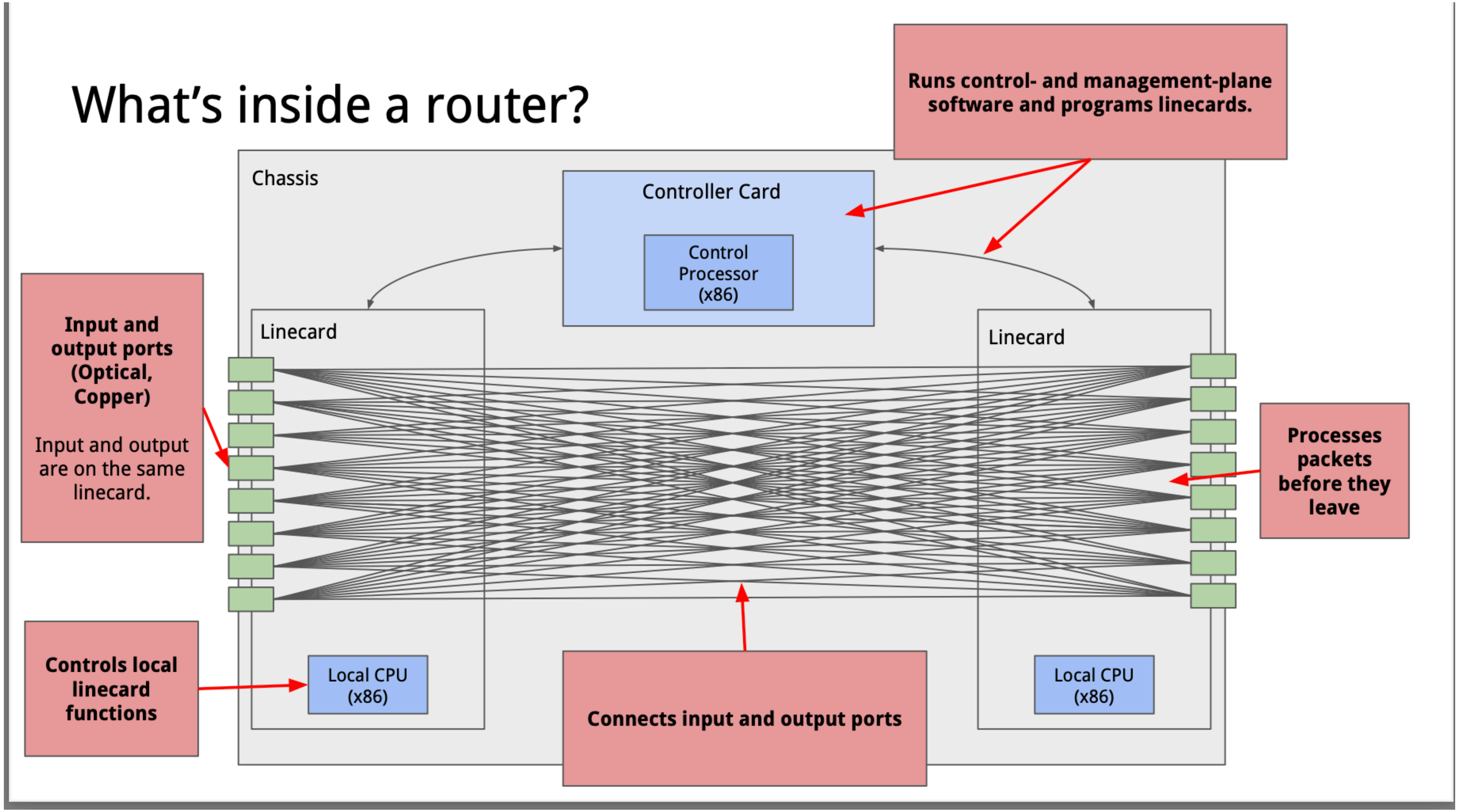
And, from these, how to compute the time it takes to transfer X bytes from A to B

- Transmission time, propagation time, queueing delay, …

Links and routers ...



Routers



What's inside a router?
Runs control- and management-plane software and programs linecards.
Chassis
Controller Card
Control Processor (x86)
Linecard
Linecard
Input and output ports (Optical, Copper)
Input and output are on the same linecard.
Processes packets before they leave
Controls local linecard functions
Local CPU (x86)
Connects input and output ports
Local CPU (x86)

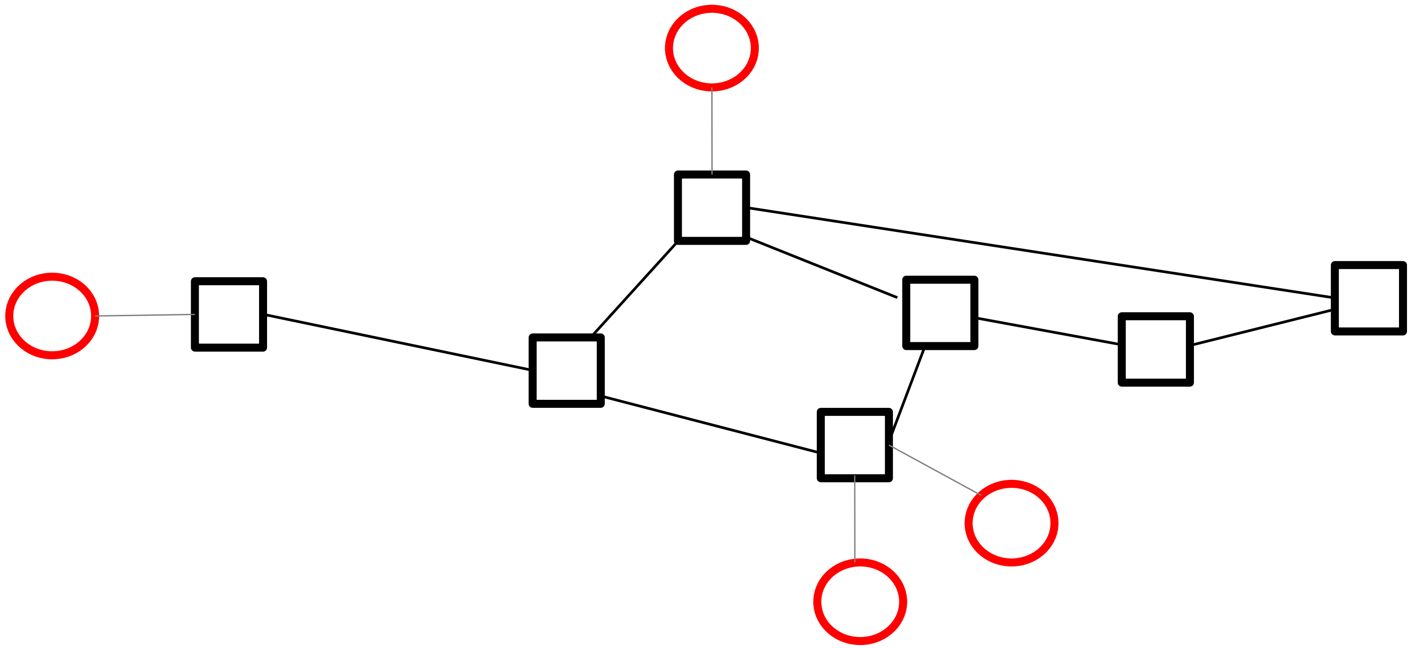
Routers

You should know the purpose (and difference between):

- Route processor vs. linecards
- Control vs. data plane
- Fast path vs. slow path

Know that the key challenge in building a router dataplane is achieving high performance

Links and routers and endhosts → topology



Fundamental challenge: routing

Routing

Basic concepts that you should know:

- Valid routes
  - No deadends, no loops
- Link costs and least-cost paths
- Route convergence

Know that sometimes, we achieve forwarding using:

- Default routes
- Static routes

Otherwise, we use a routing algorithm / protocol

Routing algorithms (within a domain)

We studied three different appraoches

- Distance-Vector (DV)
- Link State (LS)
- Spanning Tree Protocol (STP)

You should be very familiar with how these work

- be ready to apply them in different scenarios/topologies

Distance-Vector (1)

- Basic idea: I tell my neighbors about my least-cost distance to a destination; they update their least-cost distances and next-hop choices accordingly

- Components of a solution (know in detail)
  - Where we advertise routes
  - Logic/rules for when to update a route
  - Periodic vs. triggered updates

- Challenge: convergence when things change

Distance-Vector (2)

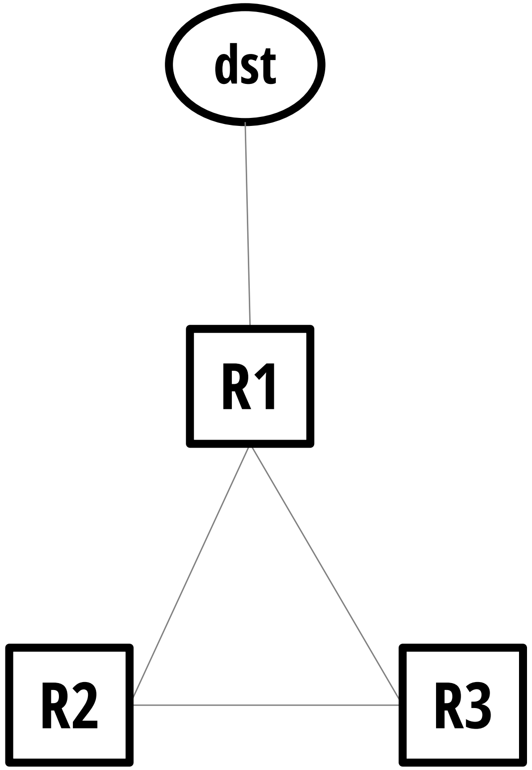
Should understand what happens when things go wrong

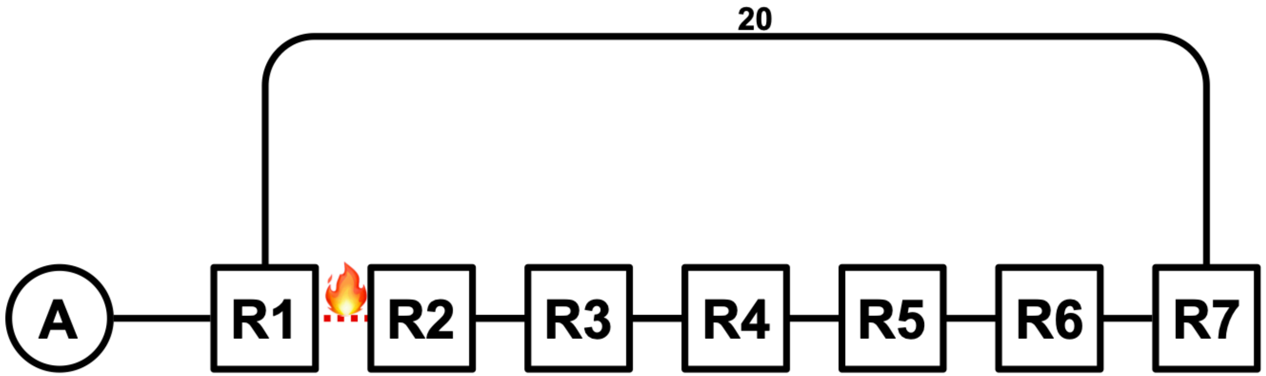
- E.g., when routers fail, links fail, advertisements are dropped
- The "counting to infinity" problem
- How TTLs and advertisement rules can lead to long convergence times

And the various rules that influence how/when convergence occurs

- Split horizon ("don't tell your next-hop anything")
- Poison reverse ("tell your next-hop infinity")
- Poisoning a route ("tell all your neighbors infinity")

Work through various scenarios under various events





Link-State: Overview

- Every router:
  - Gets the state of all links and location of all destinations
  - Uses that global information to build full graph
  - Finds paths from itself to every destination on graph
  - Uses the second hop in those paths to populate its forwarding table

Simple conceptually but w/ subtle details that you should be aware of

- E.g., how flooding works, problems during convergence, *etc.*

Understand: **Distance-Vector vs. Link-State**

- **Distance-Vector**
  - Global computation (it's distributed across all nodes)
  - .. using local data (from just itself and its neighbors)

- **Link-State**
  - Local computation
  - .. using global data (from all parts of the network)

Learning Switches and STP

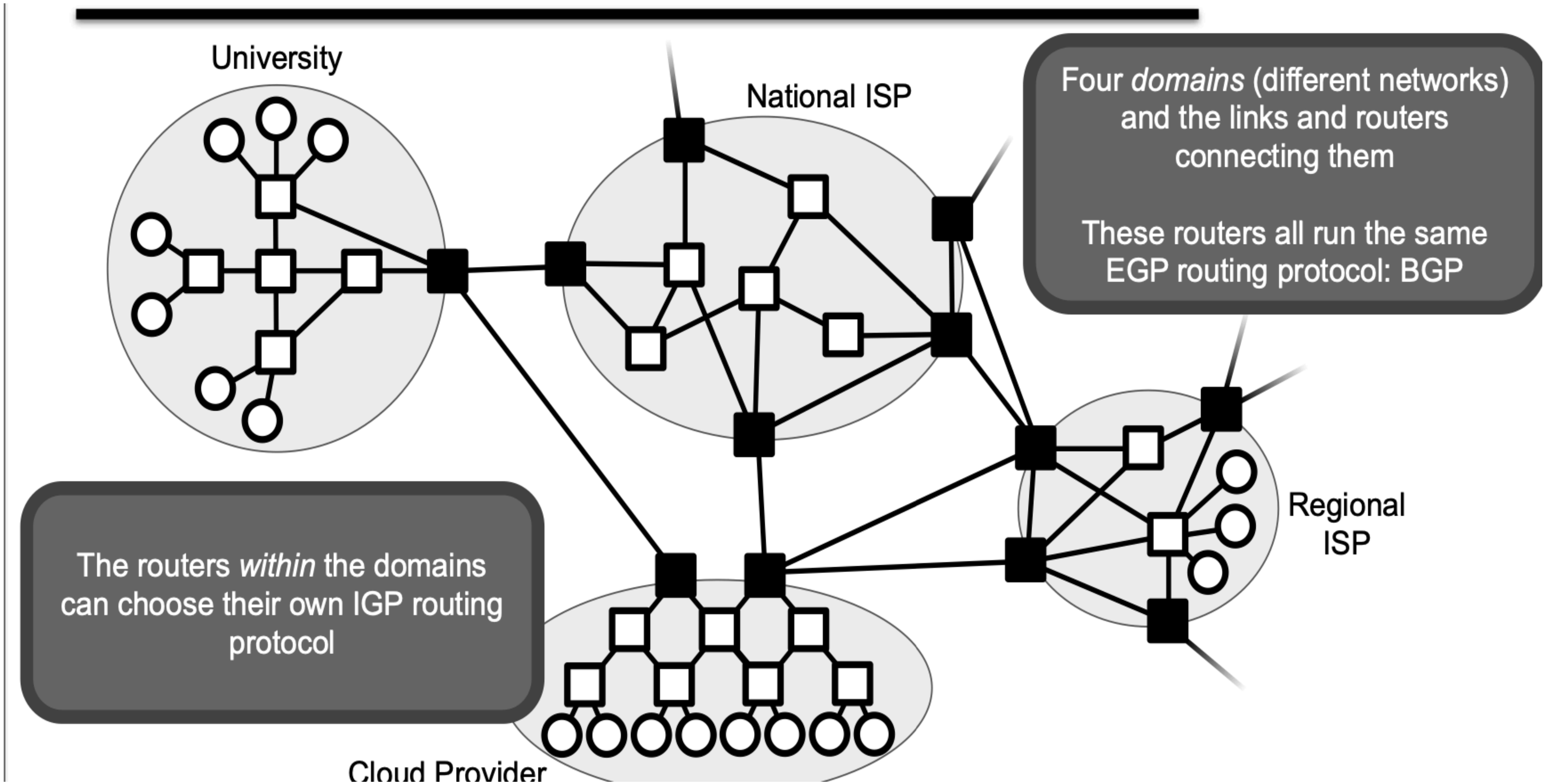
- **Basic idea:** learn routes from watching where data packets come from
  - And if you don't have a route, just flood
  - Flooding → need a loop-free topology → STP
- **STP:** Know how the protocol works
  - How we discover the root and the next-hop to the root (DV-like, with twist)
  - How we disable links not on the path to the root
- Know the pros and cons:
  - Enables "plug and play" hosts
  - Disabling links is wasteful

Where are we.. .

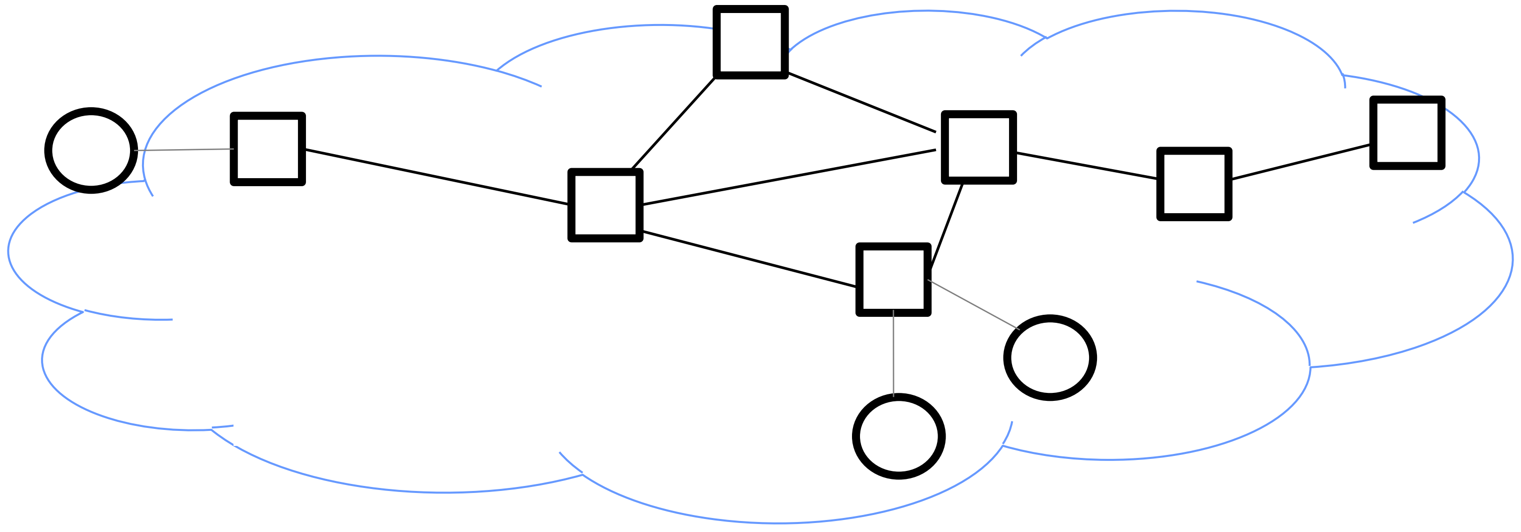
- Links → Routers → Topology → Routing (basics) → Routing protocols

Have all the components to talk about **domains** and **inter-domain routing**

Intra- & Inter-domain routing



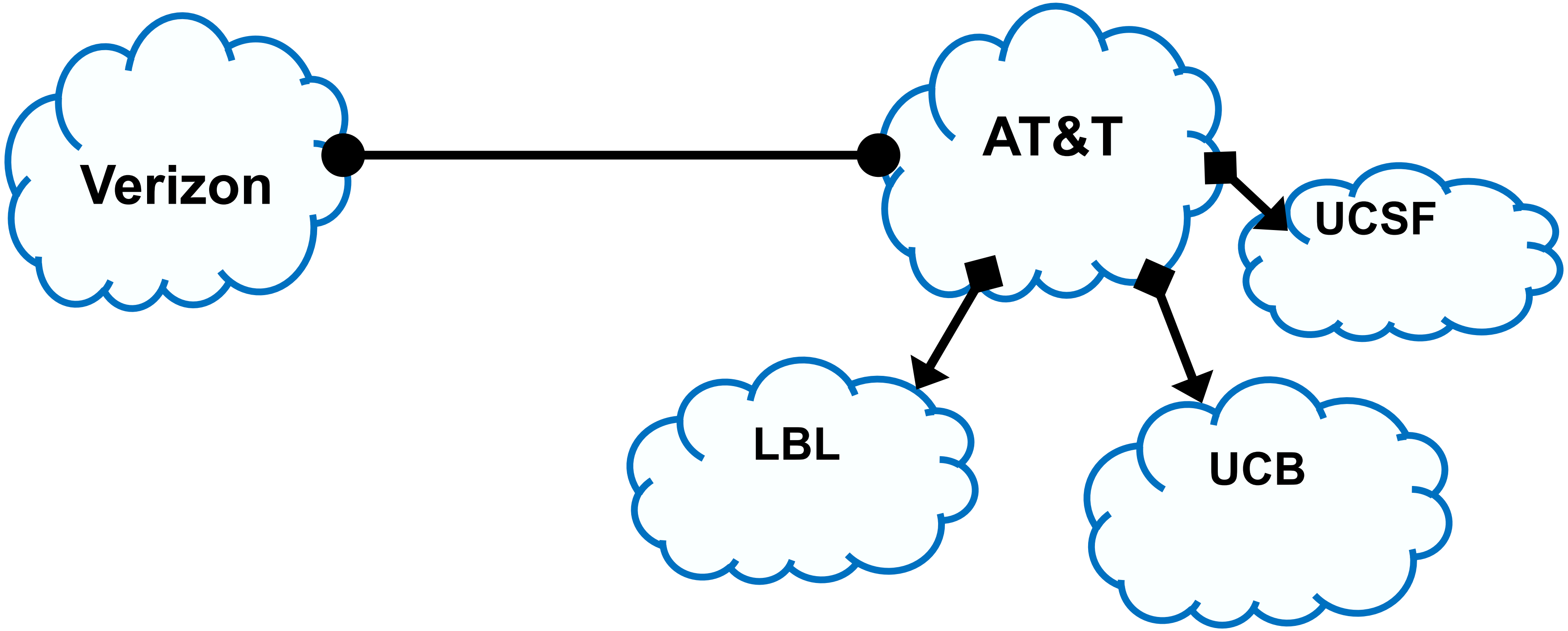
Domain: network under a single administrative control



Domain (Autonomous Systems)

- Understand the types of ASes
  - Transit vs. Stub
  - Tier-1 Transit providers
- Understand the business relationships between ASes & their implications
  - Customer-provider vs. peer-to-peer
  - Customers pay providers; peers don’t pay each other
- **Challenges in inter-domain routing:** scalability and policy

Understand how hierarchical addressing → scalability



Addressing: hierarchical

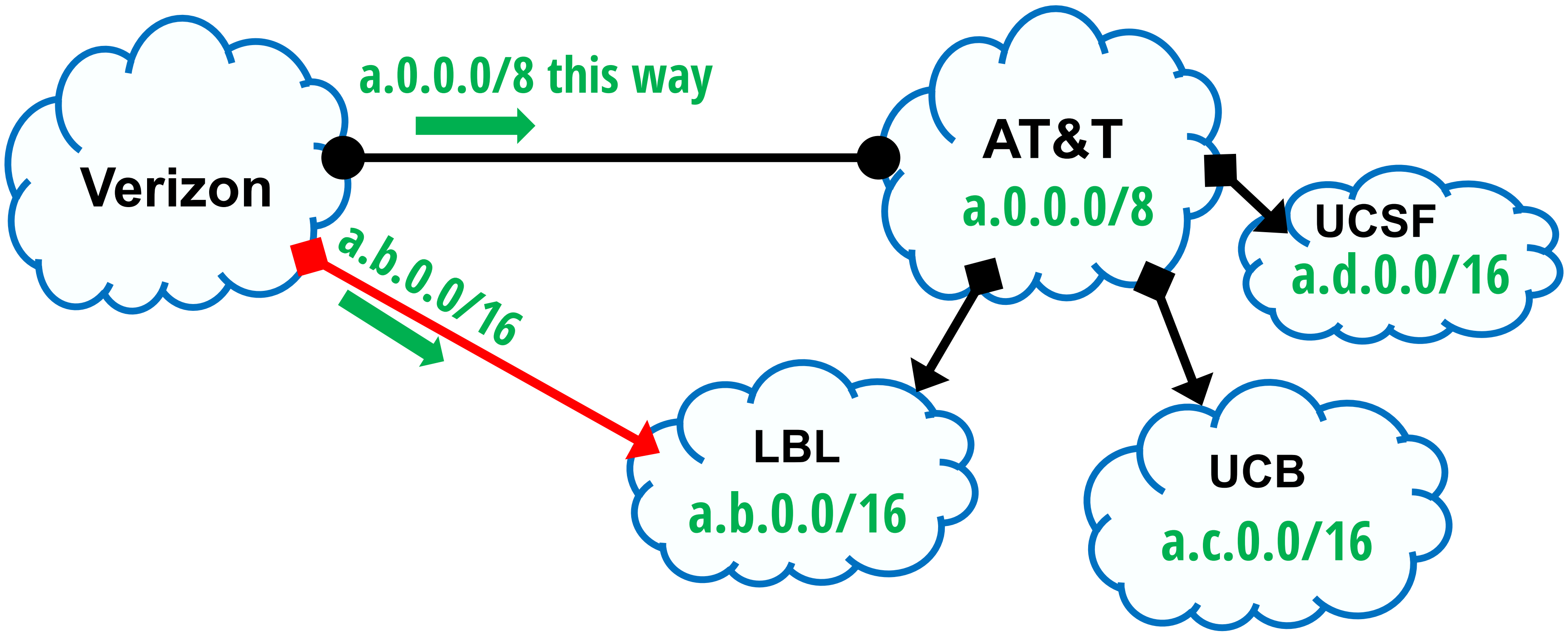
- Addresses structured as a network prefix and host suffix
  - With CIDR: length of the network prefix is flexible (recall: “slash” notation)

- Remember that addr. allocation is hierarchical: ICANN → RIR → AT&T → UCB

- And that inter-domain routing works on prefixes (vs. individual host destinations)

- And these prefixes can be aggregated (within limits)

Understand: what is multi-homing & why it limits aggregation



Verizon needs routing entries for both a.0.0.0/8 and a.b.0.0/16

CIDR and aggregation → forwarding needs LPM lookups

Address	11001001	10001111	00000111	11010010

Routing table

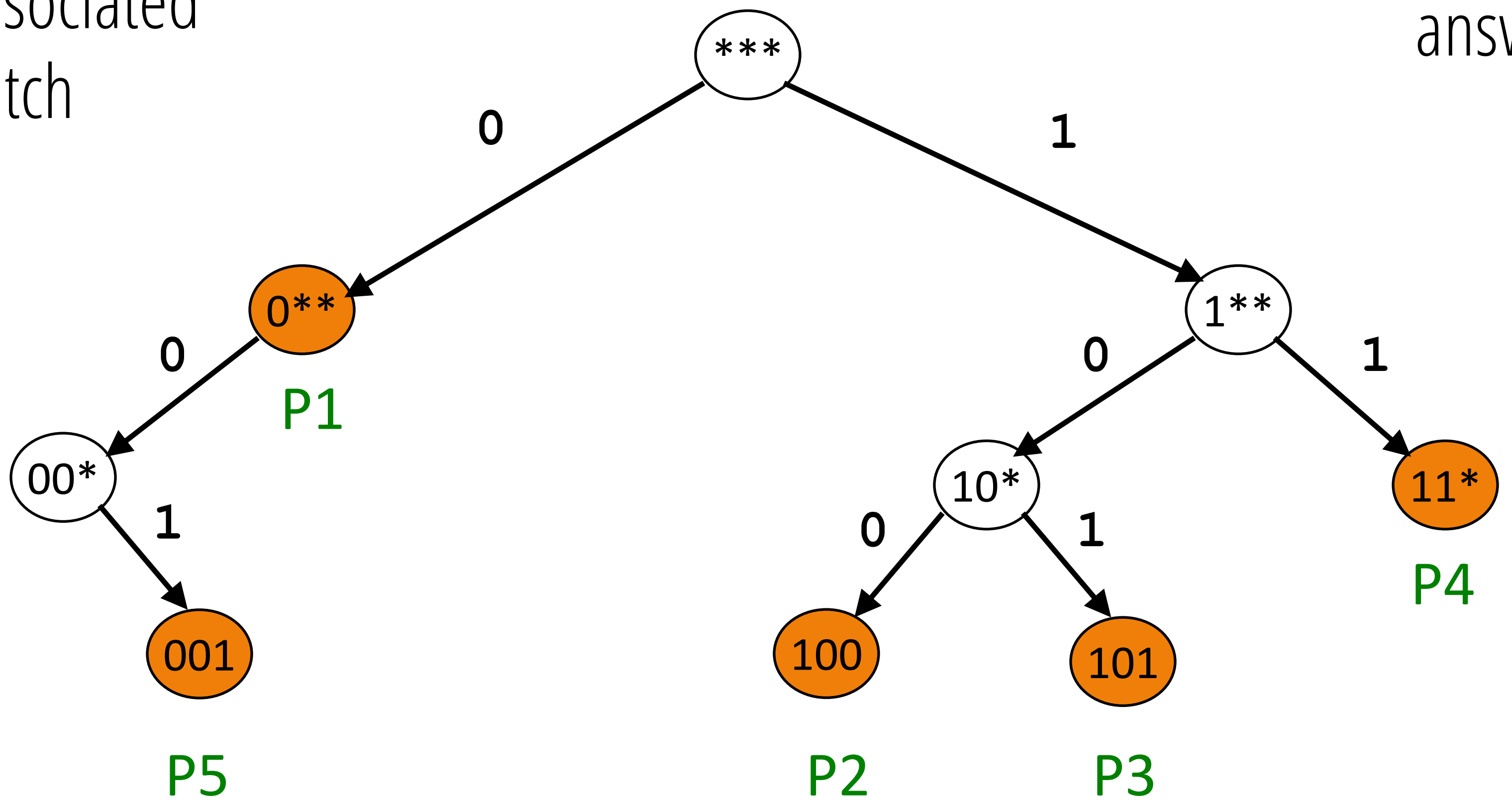
Prefix				
201.143.0.0/22	11001001	10001111	000000--	--------
201.143.4.0/24	11001001	10001111	00000100	--------
201.143.7.0/25	11001001	10001111	00000111	0-------
201.143.6.0/23	11001001	10001111	0000011-	--------

Check an address against all destination prefixes and select the longest prefix it matches with

LPM: Efficient Implementations

Walk down the tree; record port associated with latest match

If you ever leave path, you are done, last matched prefix is answer



Taking stock: we now know that…

An AS connects to other ASes as a customer/provider/peer

An AS obtains a prefix used to represent all the hosts within an AS

Next: how do we establish paths between these prefixes?

Remember, we said: routing between domains is based on policy

What do we mean by policy and what are typical policies?

- **Goal** is to let ASes to pick routes based on **policy**
  - While preserving an AS's **autonomy** and **privacy**

- Typical policy: make/save money ("routing follows the money")

Policy-based routing: the how

- **Approach:** AS <u>exports</u> and <u>selects</u> routes to a prefix
  - similar to DV, with a few differences

- **Gao-Rexford (G-R):** capture common practice for export/selection rules

Gao-Rexford

- **Selection:** customer > peer > provider

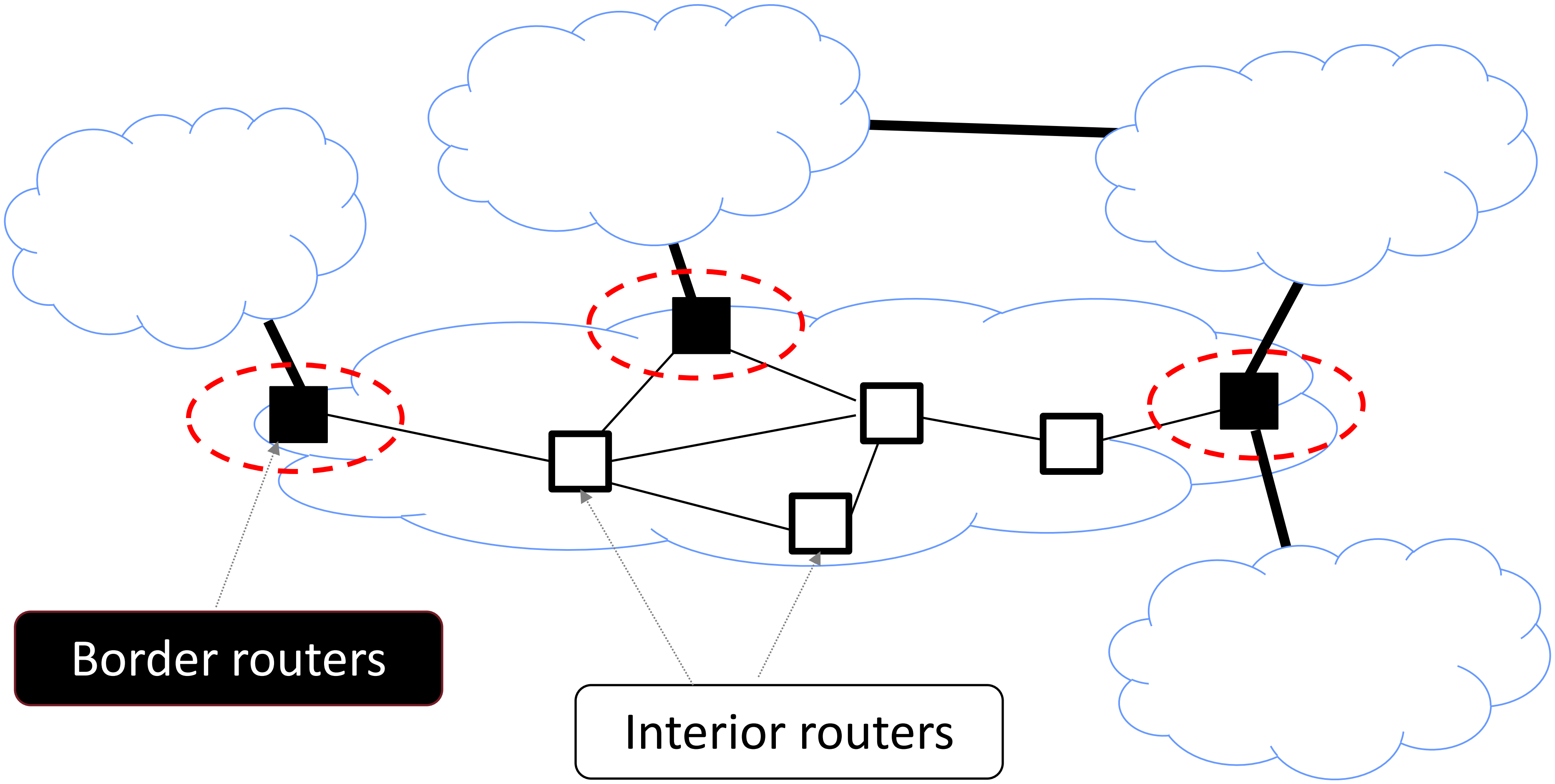
- Export

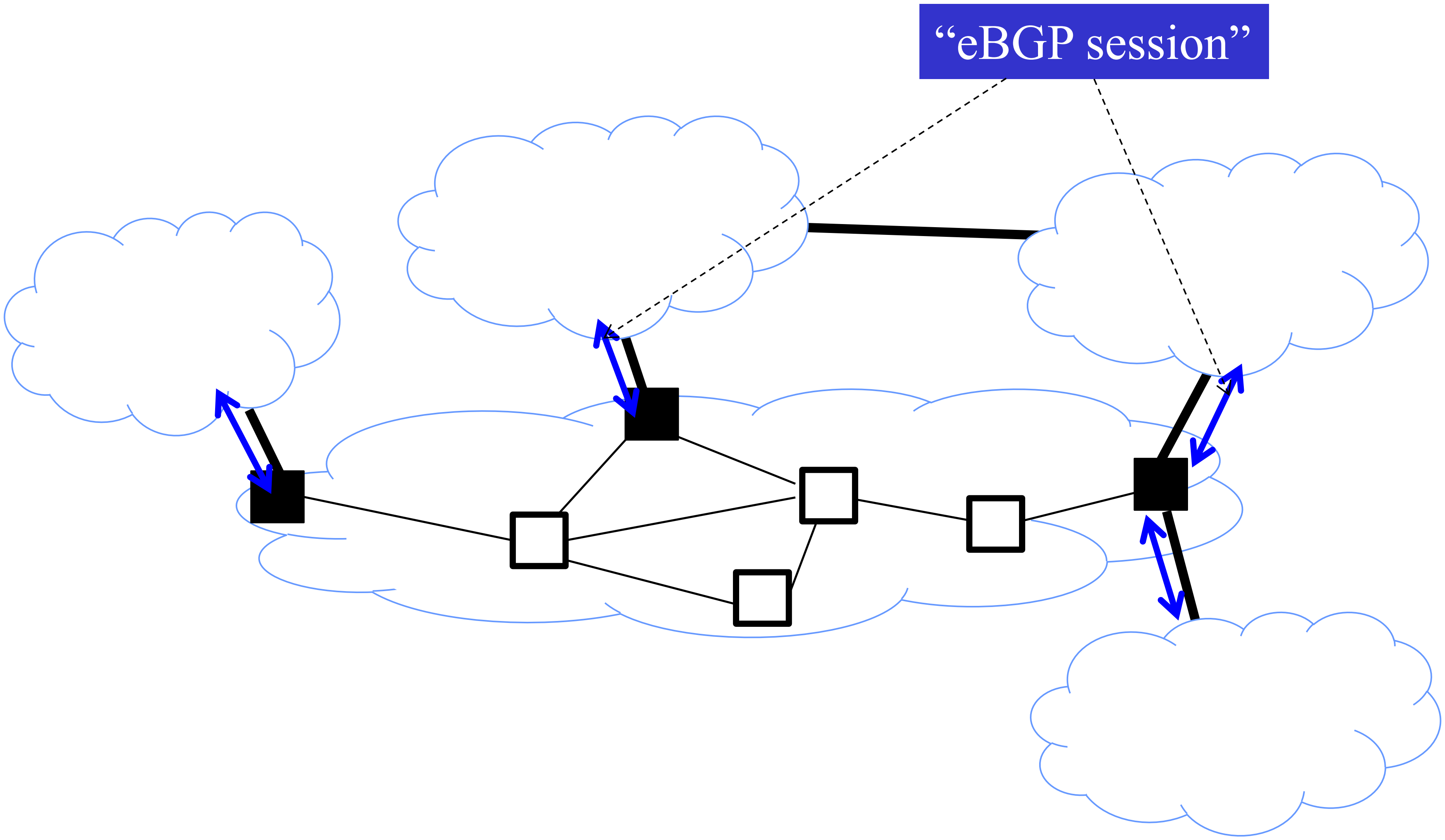
Destination prefix advertised by...	Export route to...
Customer (C)	Everyone
Peer (B)	Customers (C)
Provider (A)	Customers (C)

- Under certain assumptions, guarantees reachability and convergence

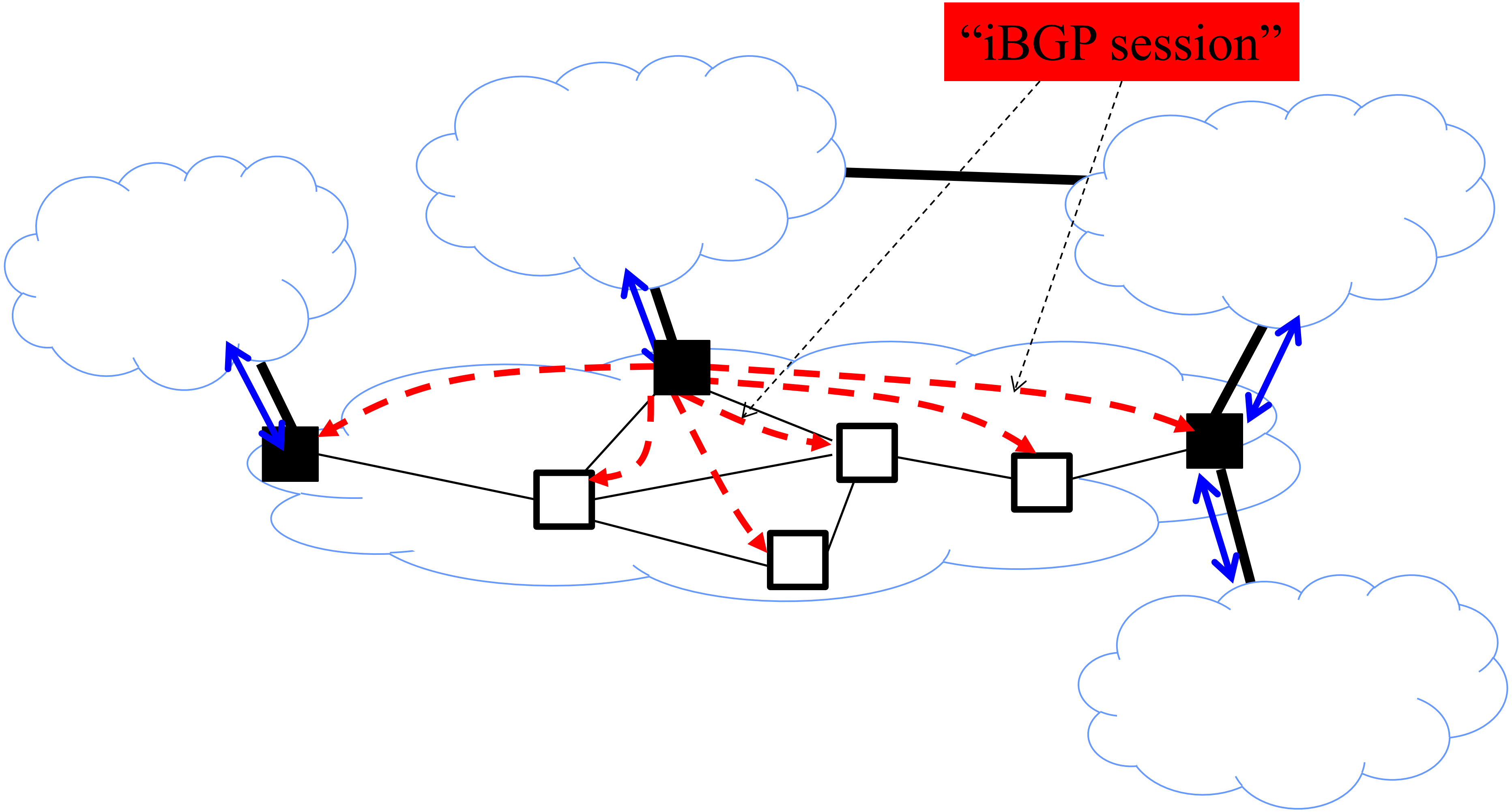
How is BGP implemented?

- You should understand the role of border vs. interior routers
- You should understand the role of: eBGP vs. iBGP vs. IGP


Border routers
Interior routers


"eBGP session"

BGP “sessions”



How is BGP implemented?

- You should understand the role of border vs. interior routers

- You should understand the role of: eBGP vs. iBGP vs. IGP

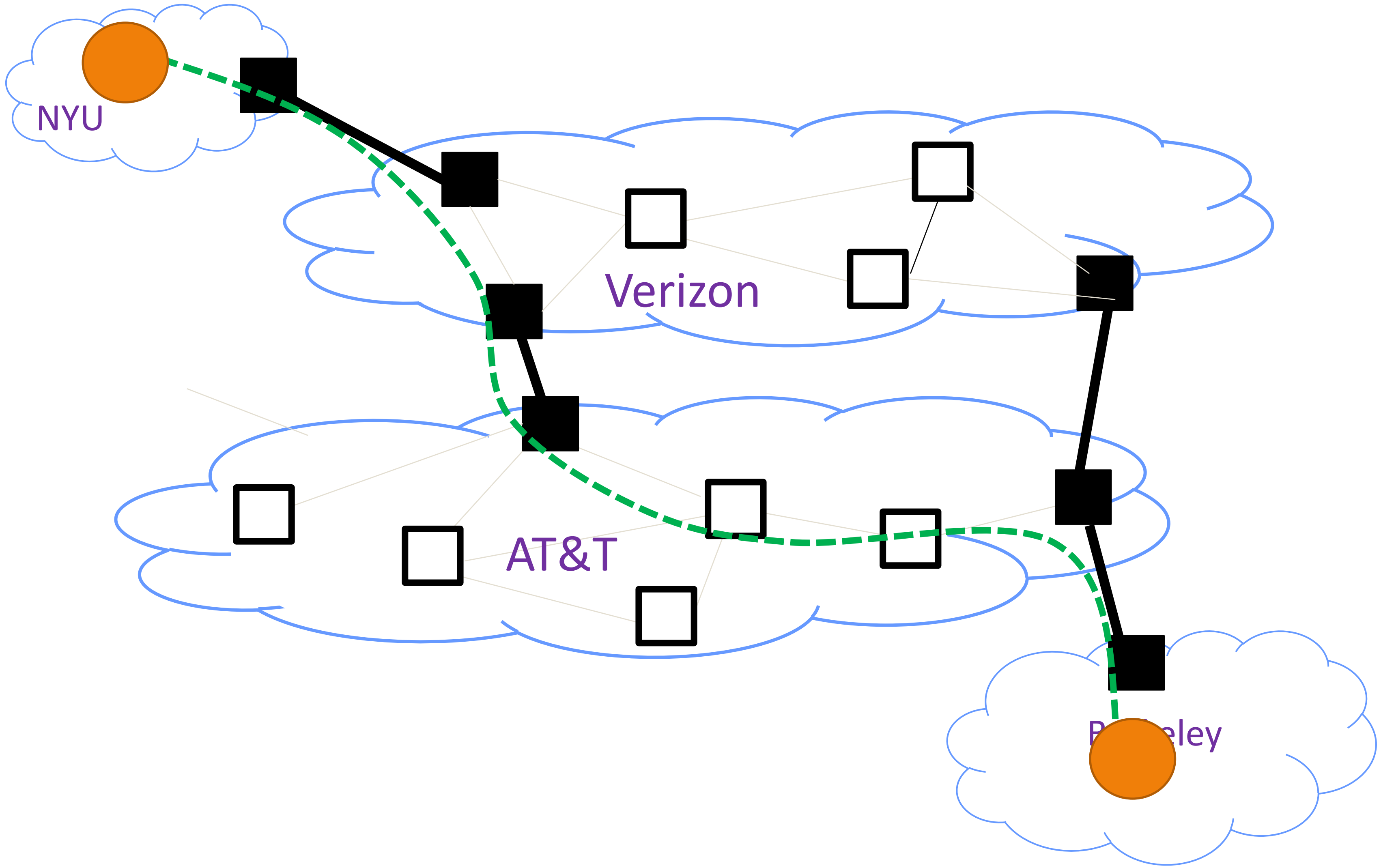
- Recall that route advertisements look like: <prefix, **attributes**>
    - Attributes associated with a route capture info needed to implement policy
    - Know these four types: ASPATH, Local pref, MED, IGP costs

- **Understand BGP's limitations/challenges** (but only at a high level)
    - Security, misconfigurations, oscillations, etc

Where are we.. .

- Links → Routers → Topology → Routing (basics) → Routing protocols
- Domains → Global (L3) addressing → Global (L3) reachability

All the pieces we need for any two end-hosts on the Internet to communicate!


NYU
Verizon
AT&T

Backing up …

- Goal of the Internet is to allow hosts to communicate across multiple networks

- The Internet reflects some fundamental choices on how to do this, that you should understand ("what and how") and appreciate ("why"):
  - Choose a best-effort service model (vs. reservations)

You should know ....

Two canonical approaches to sharing

- **Reservations**: end-hosts explicitly reserve BW
- **Best-effort**: just send data packets when you have them and hope for the best ...

Two canonical designs to implementing these approaches

- Reservations via **circuit switching**
- Best-effort via **packet switching**

Understand tradeoffs: Circuit vs. Packet Switching

- Pros for circuit switching:
  - Better application performance (reserved bandwidth)
  - More predictable and understandable (w/o failures)

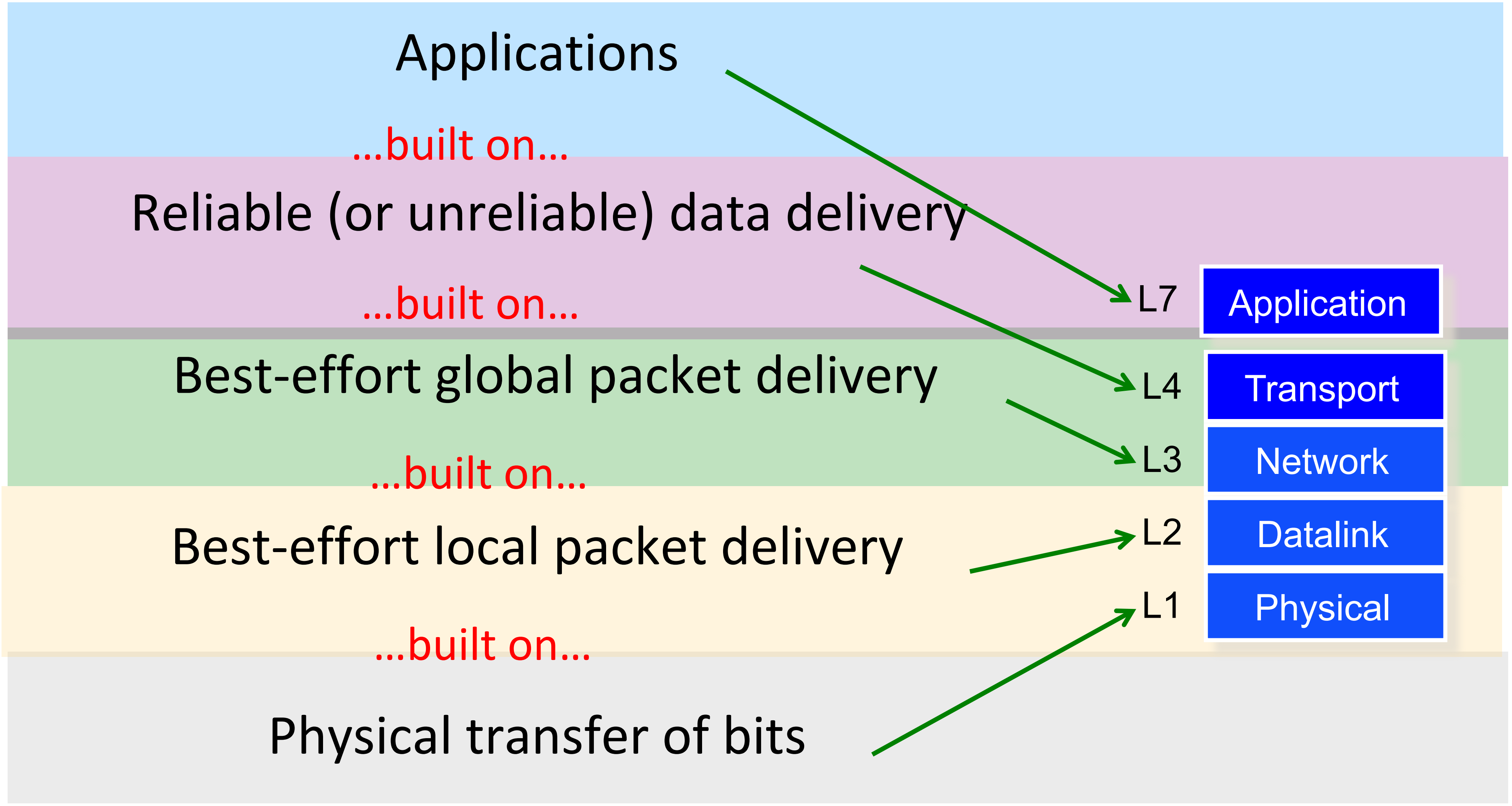
- Pros for packet switching:
  - Better efficiency
  - Faster startup to first packet delivered
  - Easier recovery from failure
  - Simpler implementation (avoids dynamic per-flow state management in switches)

Backing up …

- Goal of the Internet is to transfer data between end hosts

- The Internet reflects some fundamental choices on how to do this, that you should understand (“what and how”) and appreciate (“why”):
  - Choose a **best-effort** service model (vs. reservations)
  - Modularity through layering

Understand: Layered organization

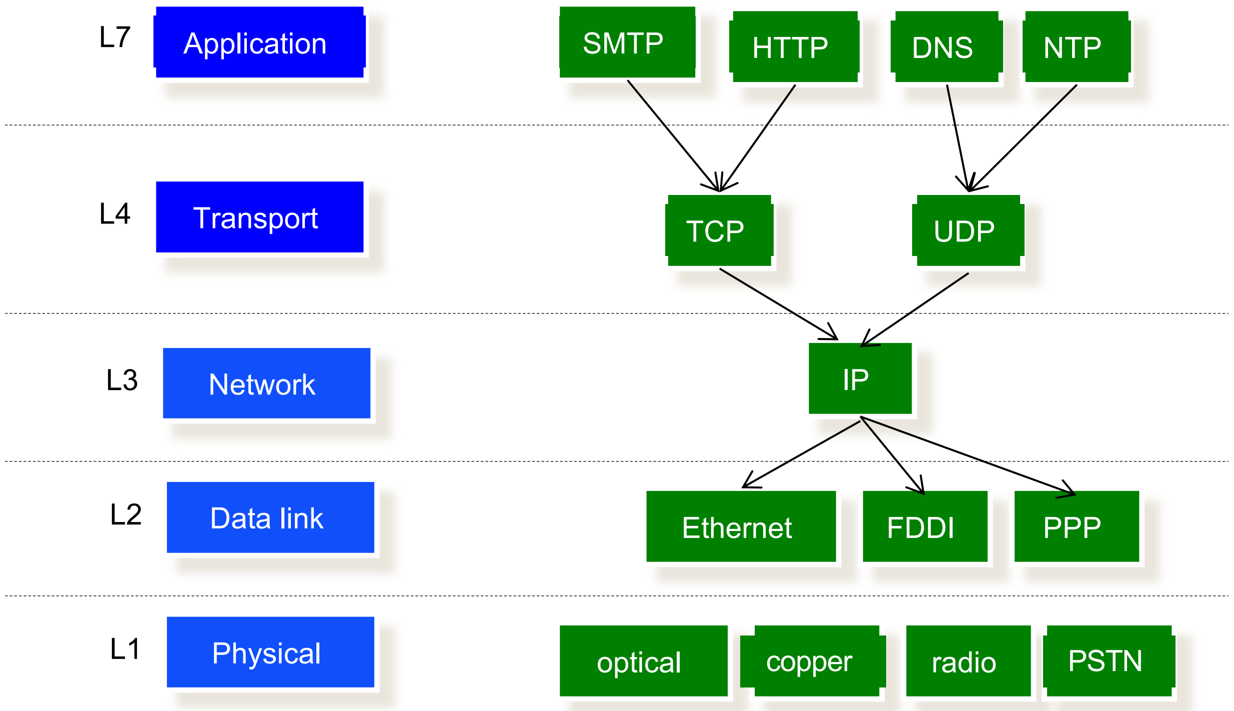


Protocols and Layers

Communication between peer layers on different systems is defined by protocols

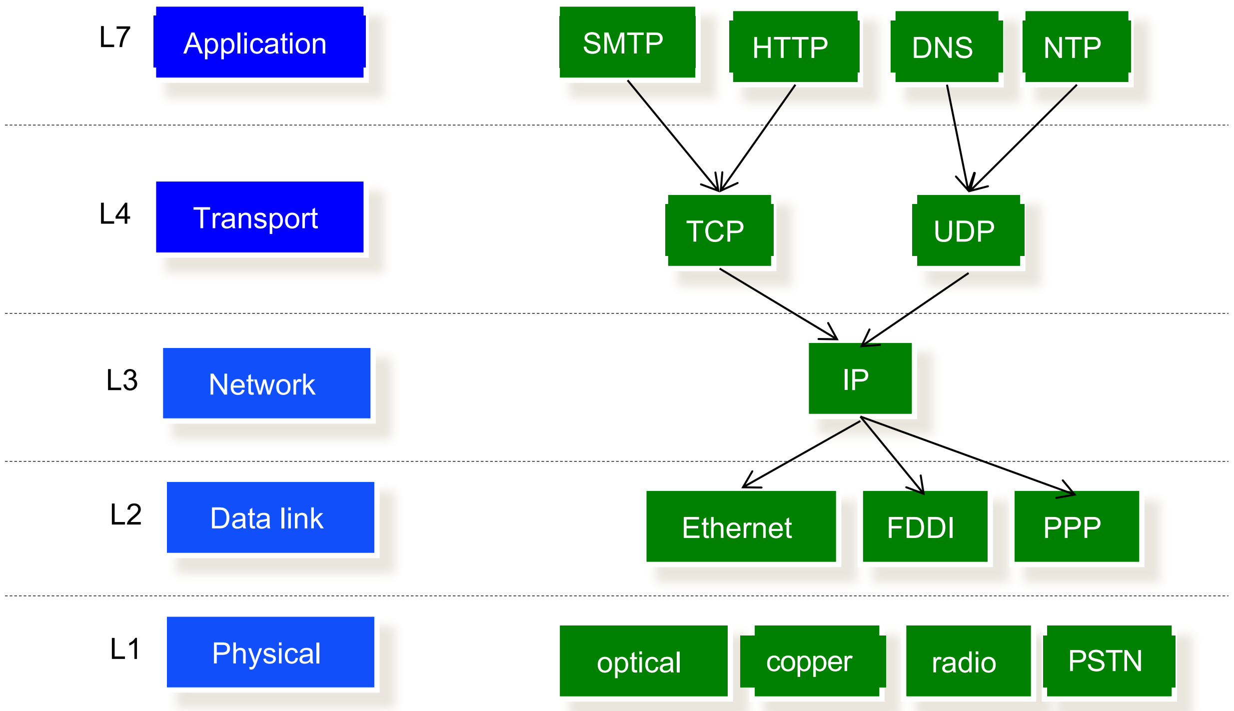
Definitely know!: three important properties

- Each layer:
  - Depends on layer below
  - Supports layer above
  - Independent of others
- Multiple versions in a layer
- But only one IP layer
  - Unifying protocol enables interoperability



Definitely understand: Why was layering important?

- Innovation proceeded largely in parallel
  - Payoff of modularity!



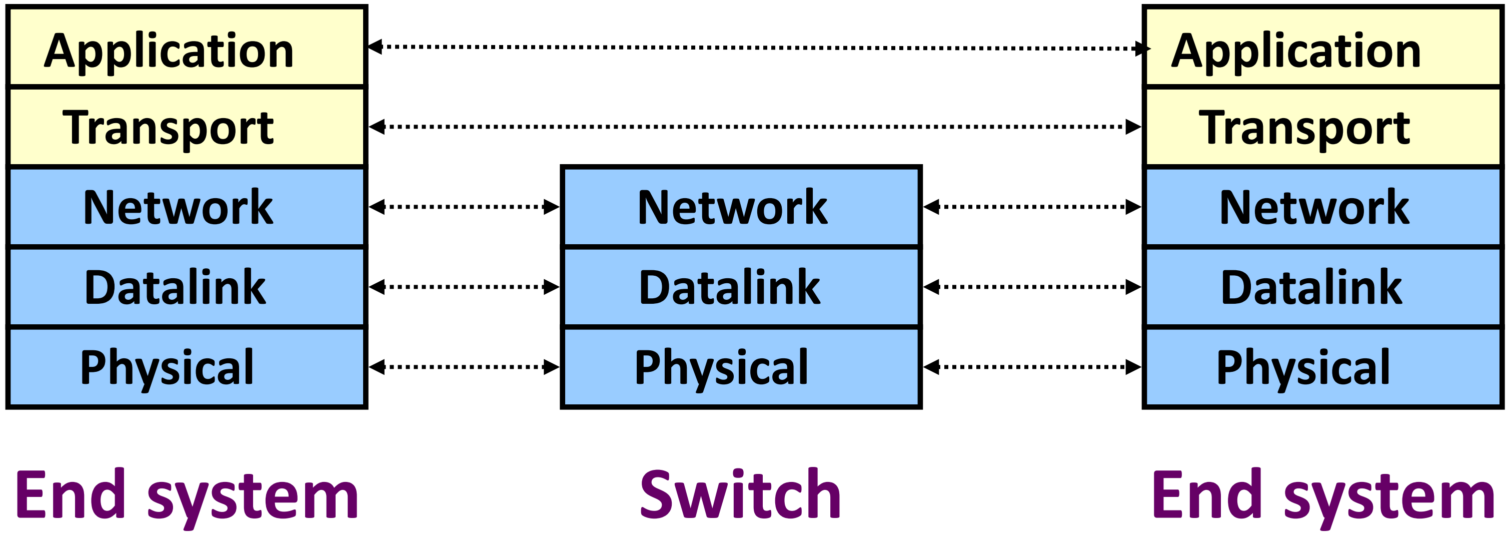
Backing up ...

- Goal of the Internet is to transfer data between end hosts

- The Internet reflects some fundamental choices on how to do this, that you should understand ("what") and appreciate ("why"):
  - Choose a **best-effort** service model (vs. reservations)
  - Modularity through **layering**
  - Placement of functionality (and state) guided by the e2e principle (and fate-sharing)

Know: what layers, where

- Lower three layers implemented everywhere
- Top two layers implemented only at hosts



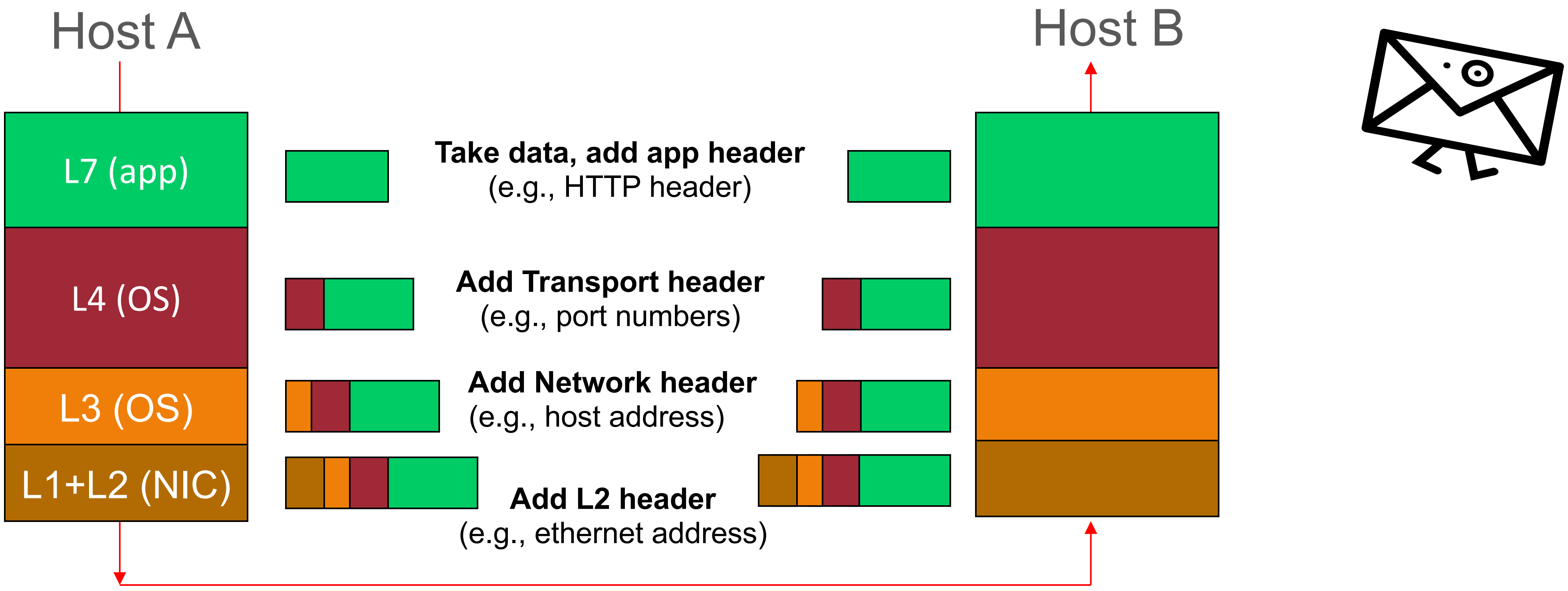
Know: *why* we layers are placed in this way

"The function in question can completely and correctly be implemented only with the knowledge and help of the application at the end points. Therefore, providing that function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.) "

- the end-to-end argument

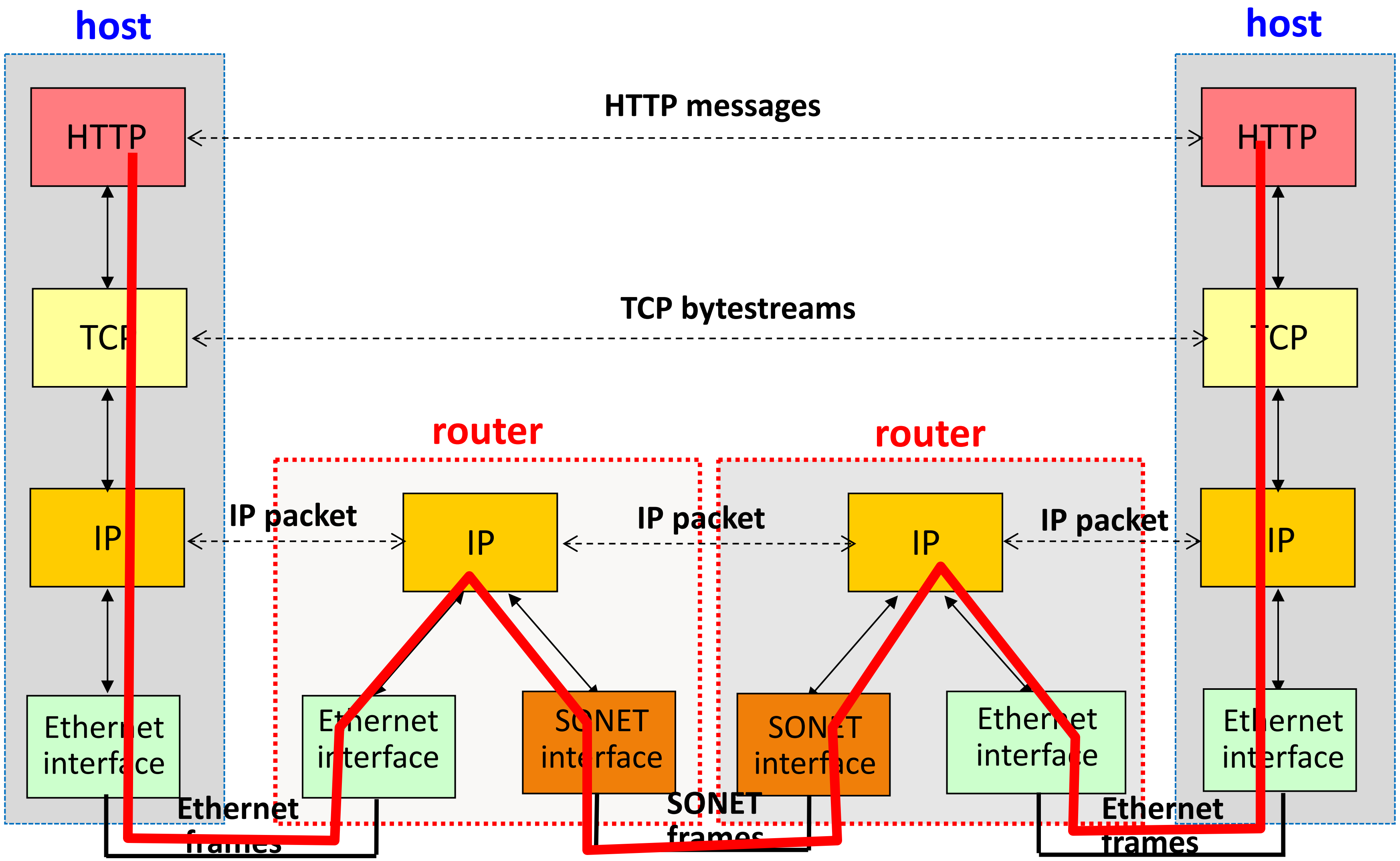
Understand what the argument is saying and be ready to analyze design choices through the lens of this argument

Understand: how data travels in a layered architecture



Packets, headers, header encap/decap at different points

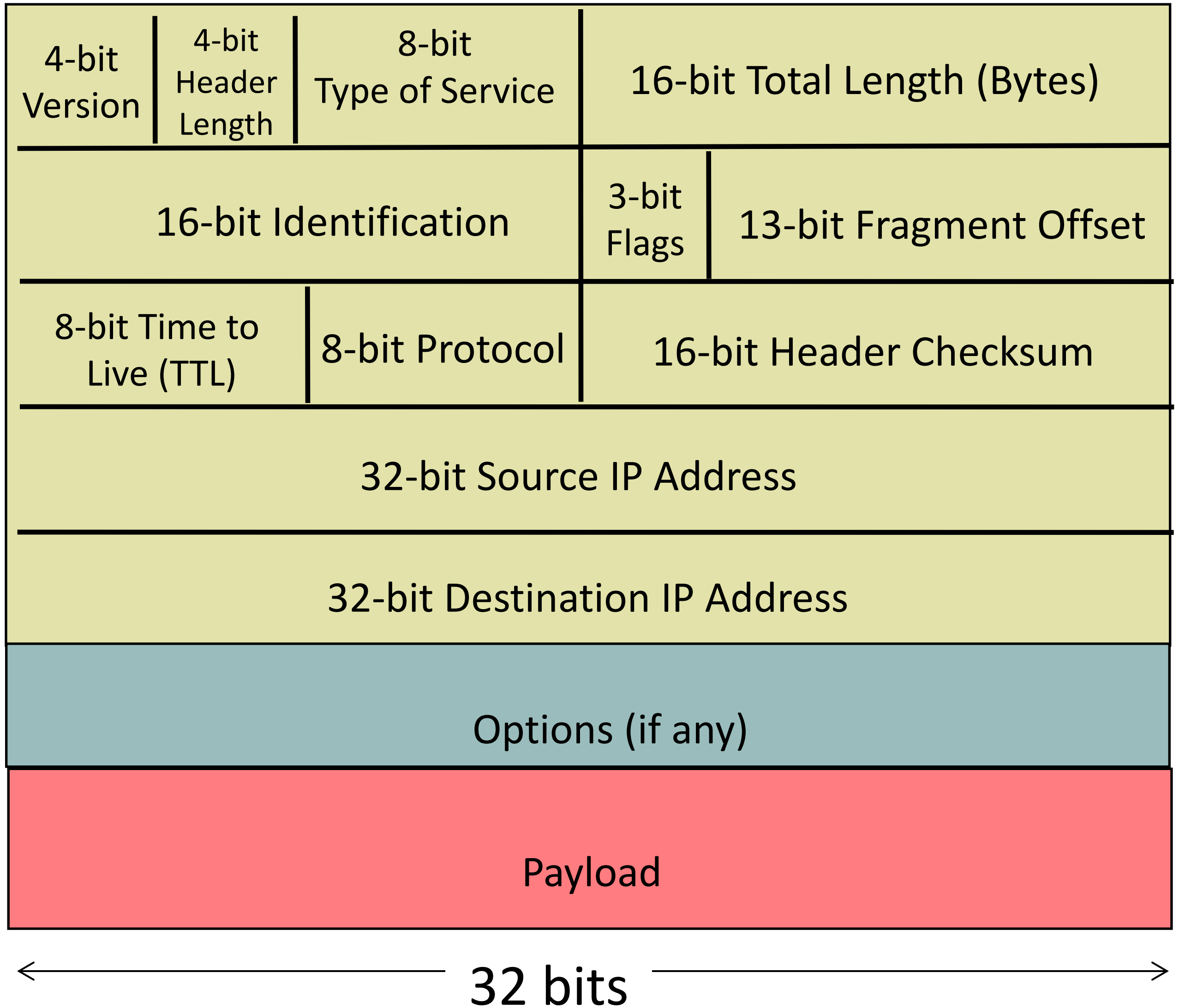
Understand: which parts of network/host process what layers/headers



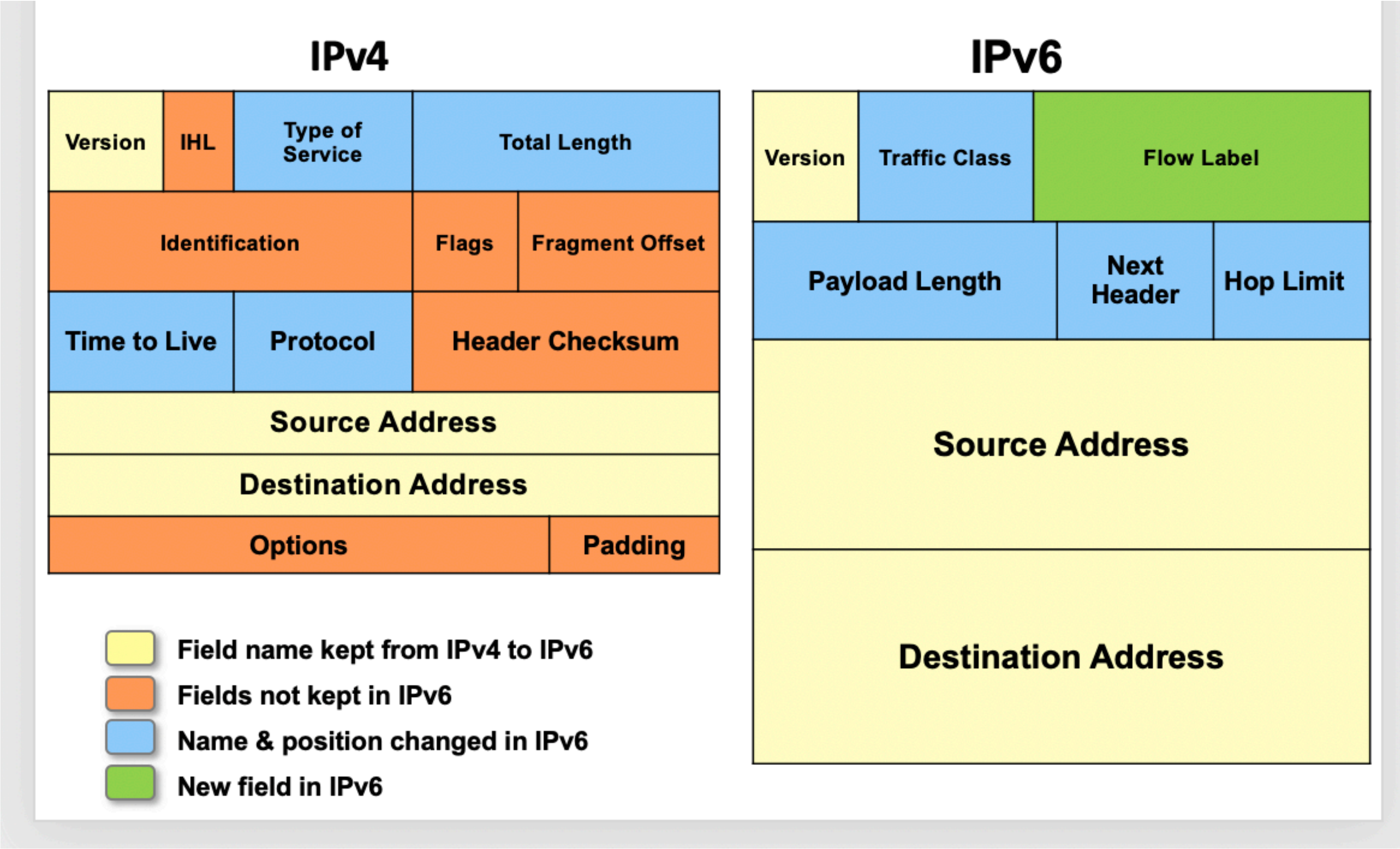
Taking stock

- Looked at how we do things (control plane)
  - Links, routers, domains, intra-domain routing, inter-domain routing, addressing
- To why we do things this way
  - Best-effort, layering, e2e arguments, fate-sharing, interoperability
- Back to how (data plane)
  - One remaining topic: design of IP

IP: why each field, what's well done vs. not



Understand why: IPv4 and IPv6 Header Comparison



That’s it!

- If you get all the concepts in this slide deck, you’re in good shape
- During the exam, remember to stay calm and reason through a problem
  - You can do it!
- All the best!