Wrapping up BGP
&
Designing IP

Spring 2022

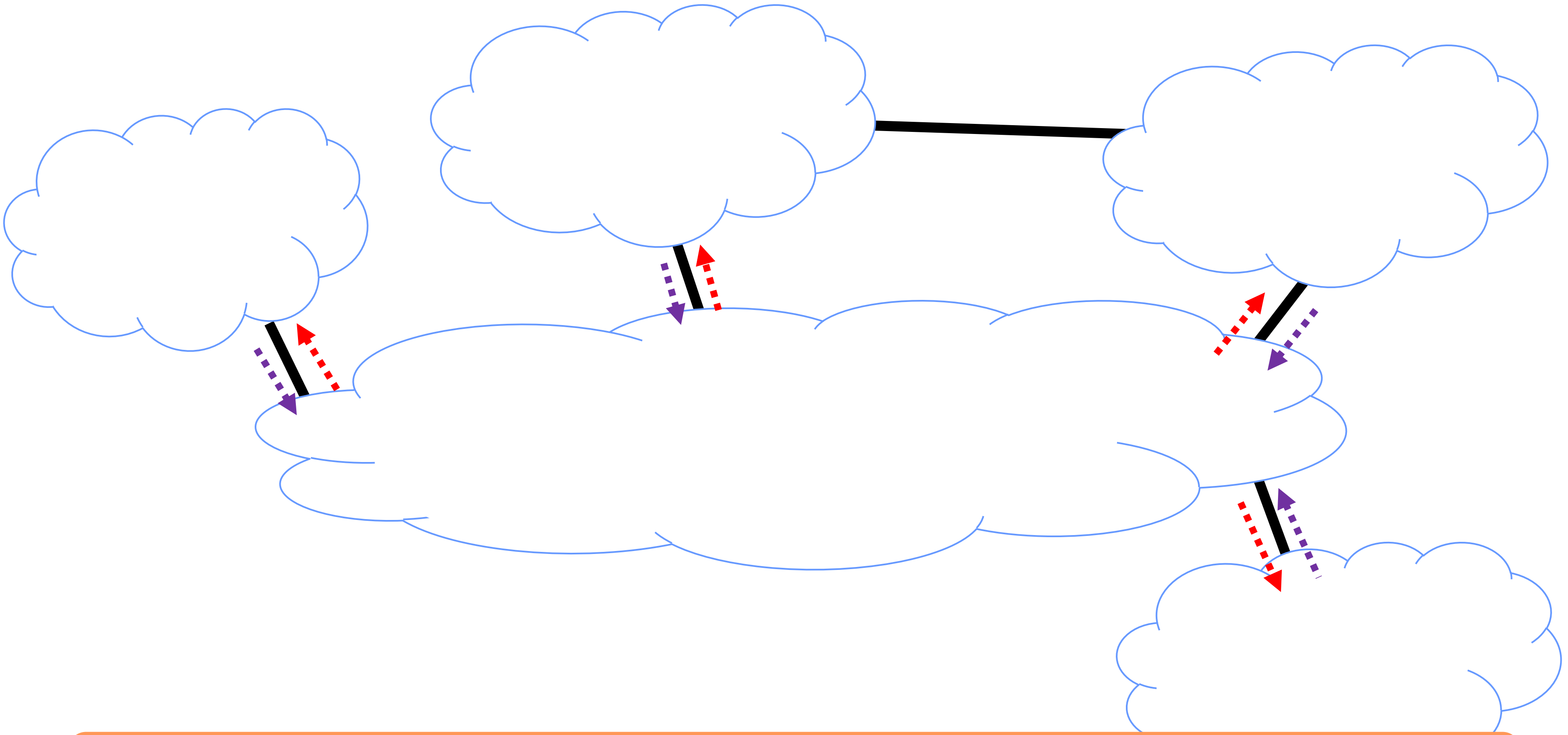
Sylvia Ratnasamy

CS168.io

Outline

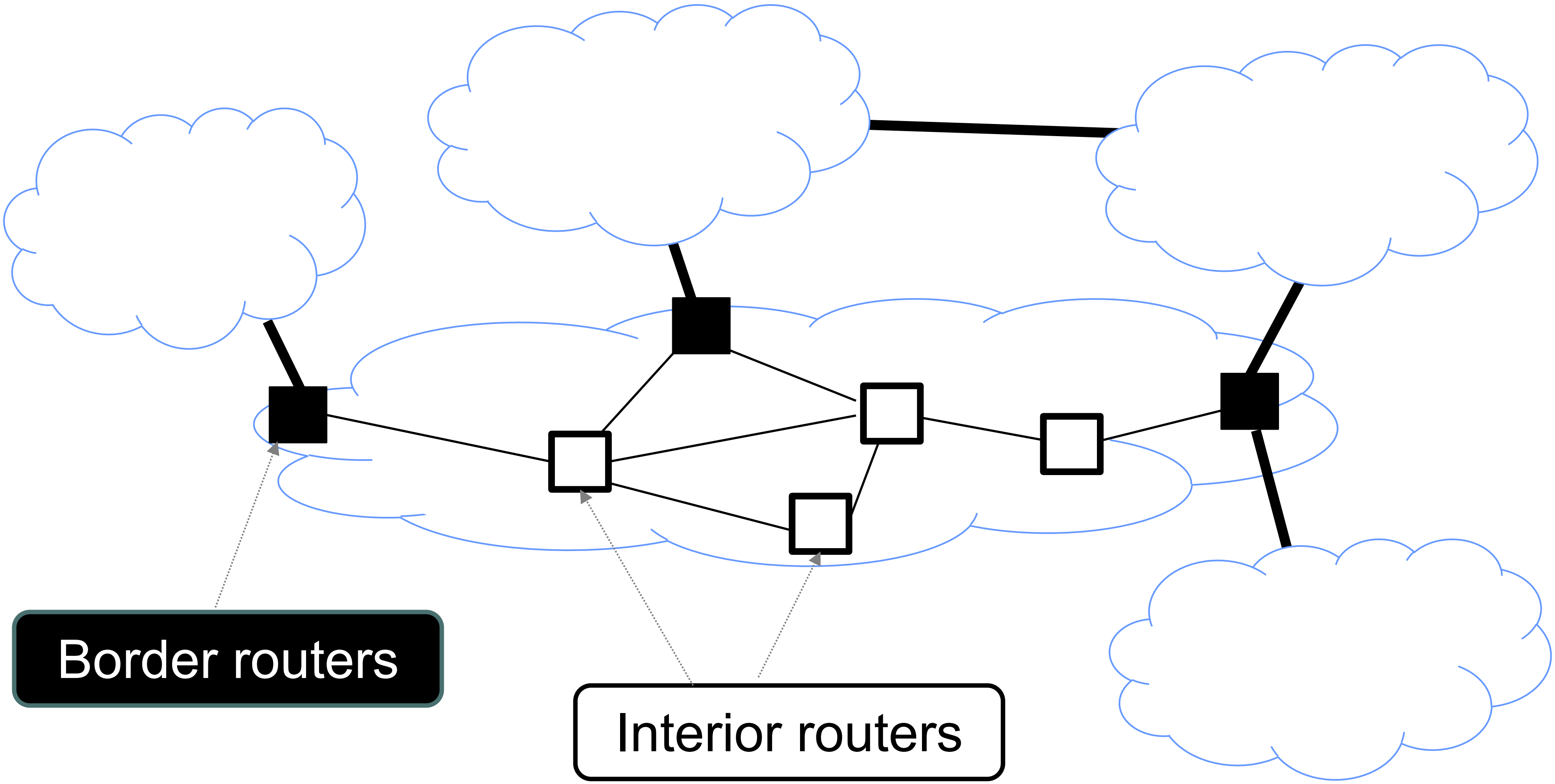
- Wrapping up BGP
  - Context
  - Goals
  - Approach
  - Protocol design
  - Limitations
- Designing IP

So far: our model of the AS graph



An AS advertises routes to its neighbor ASes

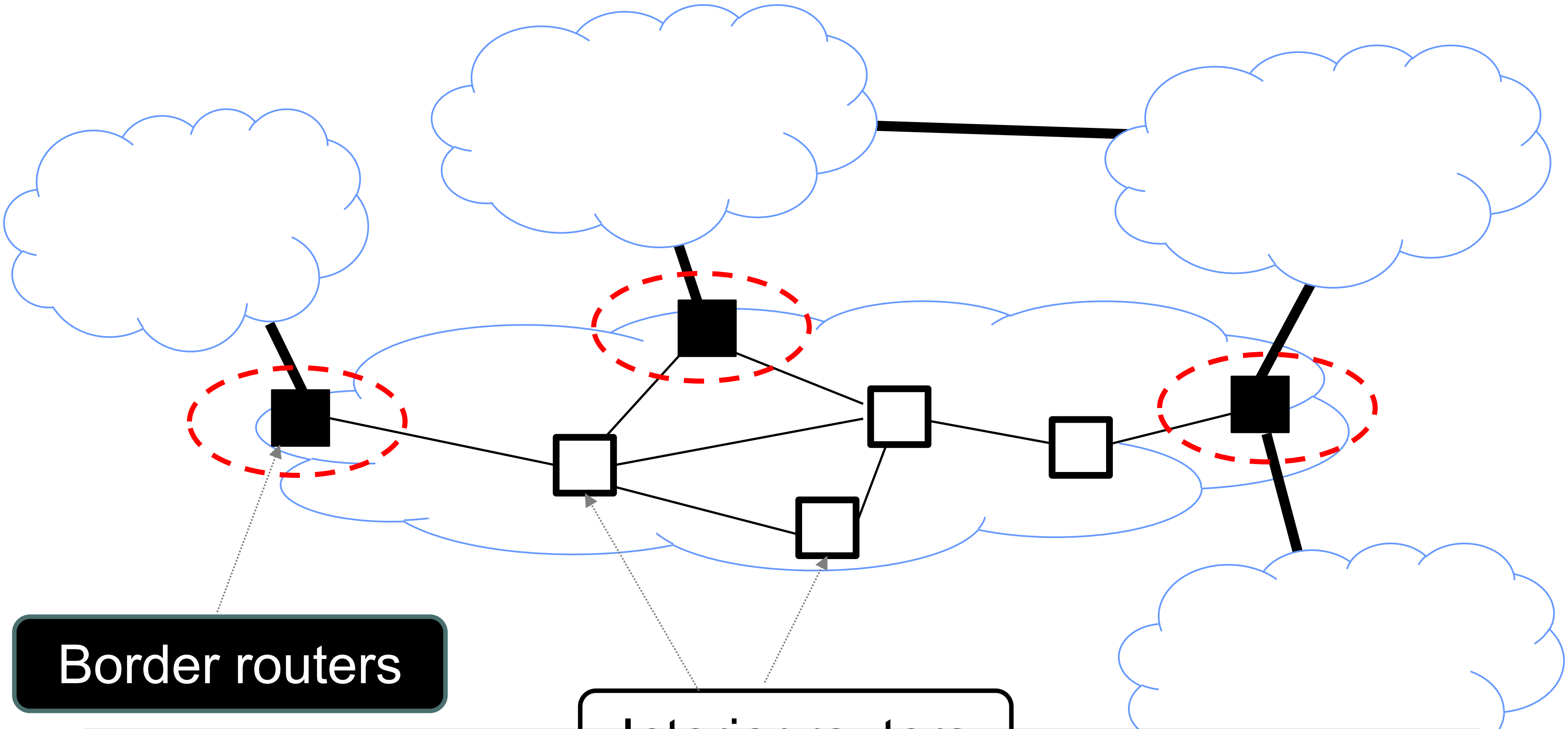
In reality...



Many design questions....

- How do we ensure the routers "act as one"?
  - The role of border *vs.* interior routers?
  - Interaction between BGP and IGP?
  - How does BGP implement all this?

Who speaks BGP?



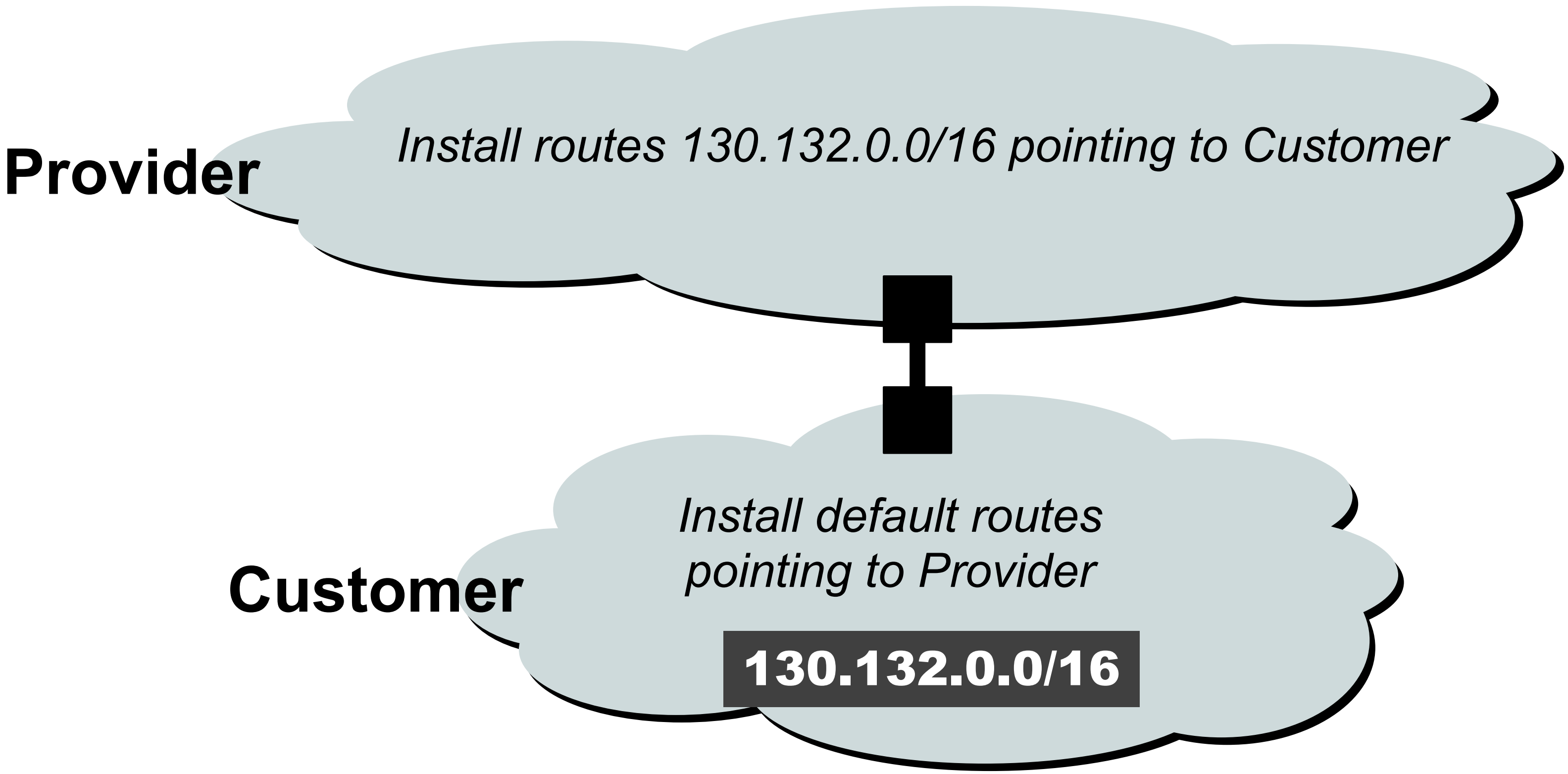
Border routers at an Autonomous System

What does “speak BGP” mean?

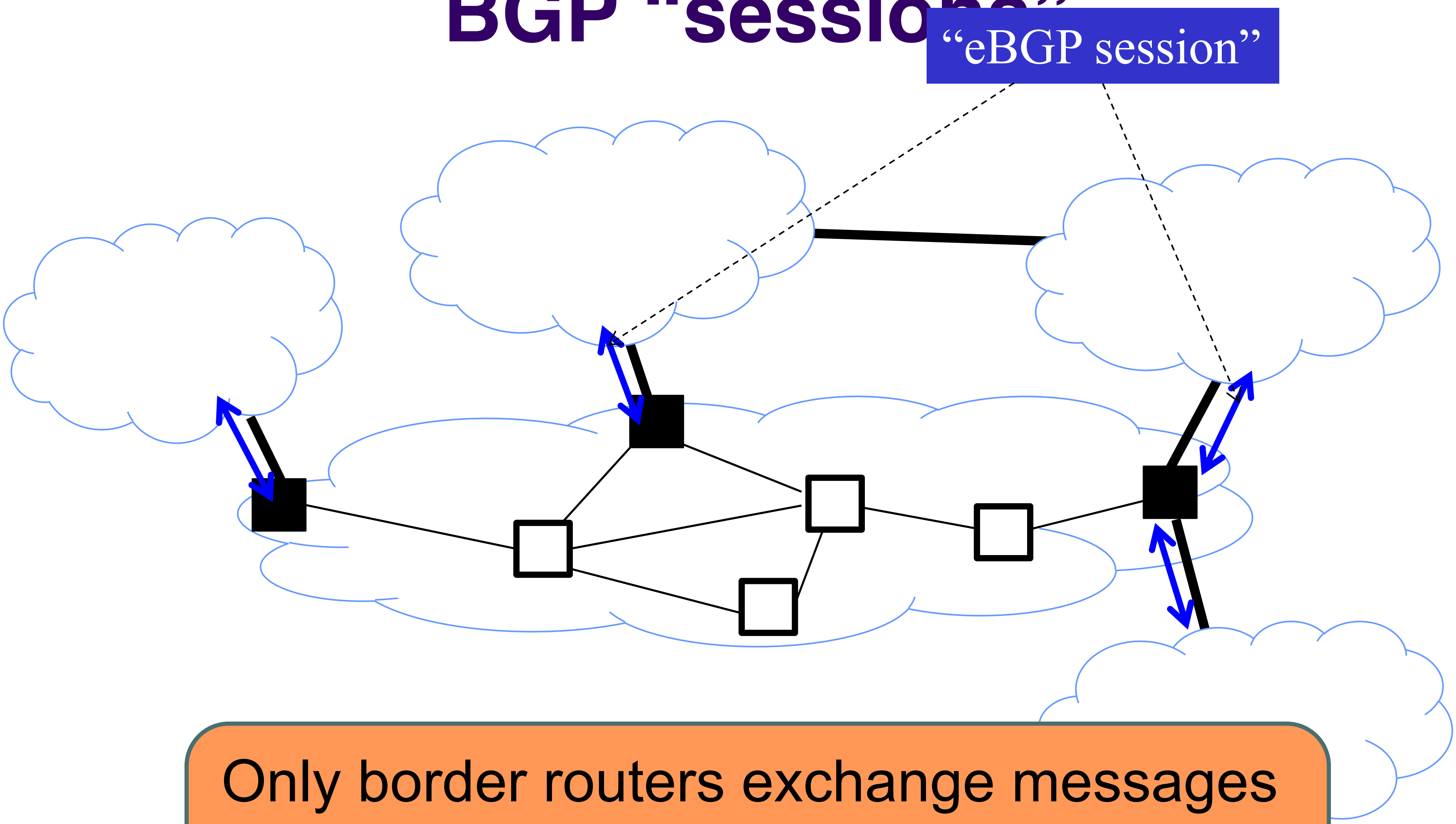
- Advertise routes as specified by the BGP protocol standard
  - read more here: http://tools.ietf.org/html/rfc4271
- Specifies what messages BGP “speakers” exchange
  - message types and syntax
- And how to process these messages
  - e.g., *“when you receive a BGP update, do…. “*

Some Border Routers Don't Need BGP

- Customer that connects to a single provider AS
  - Provider can advertise prefixes into BGP on behalf of customer
  - … and the customer can simply default-route to the AS

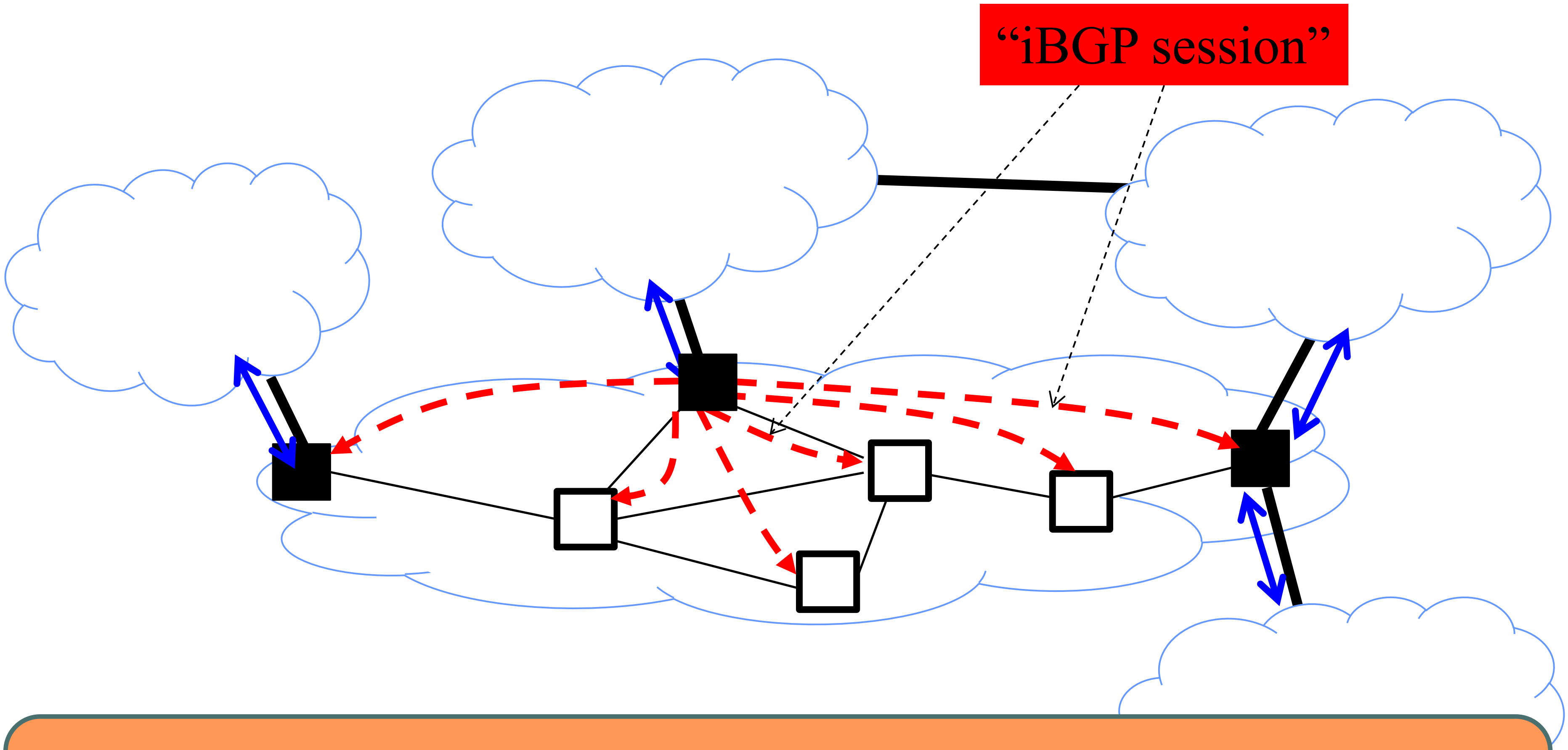


BGP “sessions”



Only border routers exchange messages with routers in external domains (hence, *external BGP* or “eBGP”)

BGP "sessions"

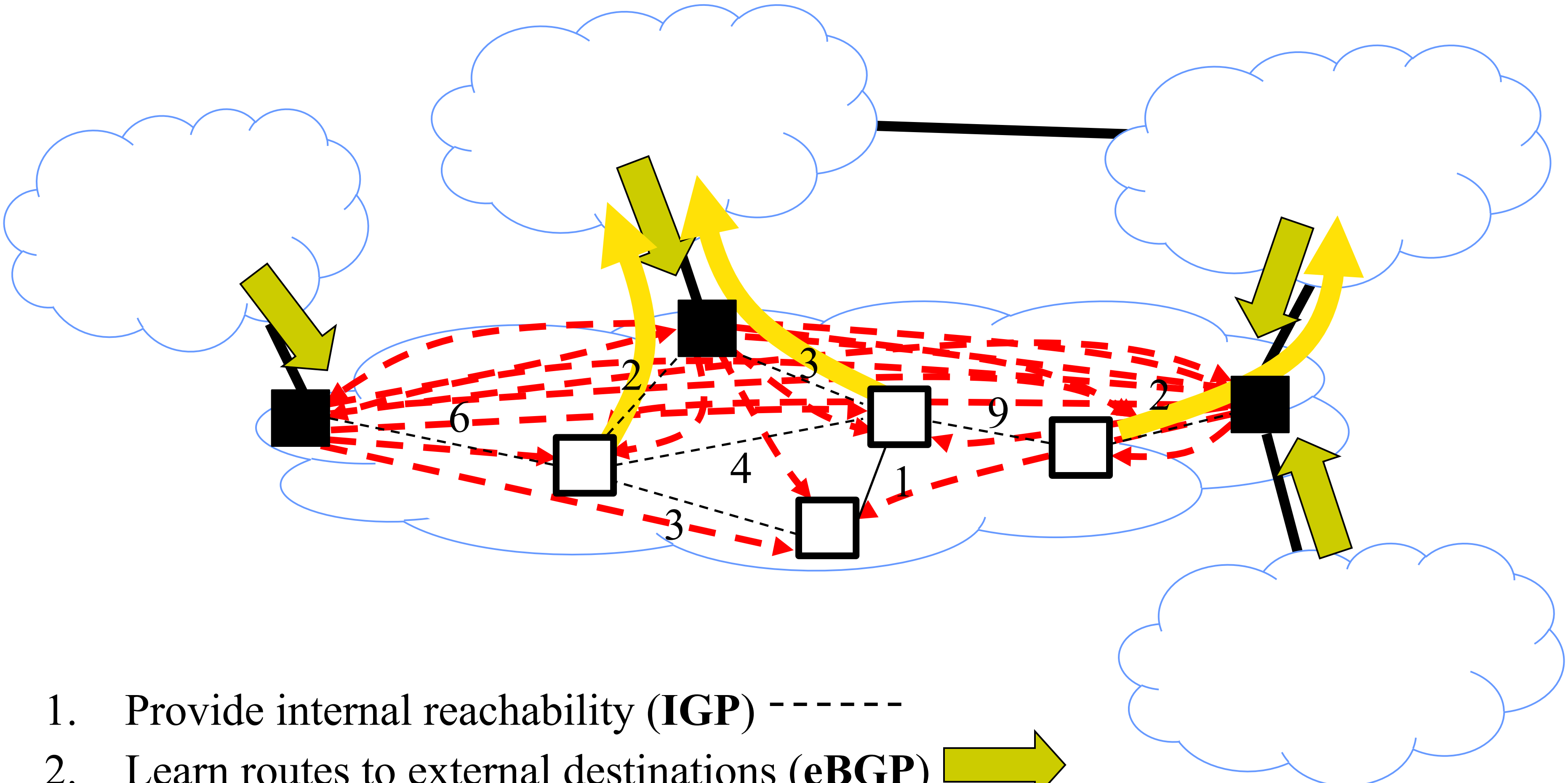


Border router speaks BGP with routers in its own AS (hence, *internal* BGP, or "iBGP")

eBGP, iBGP, IGP

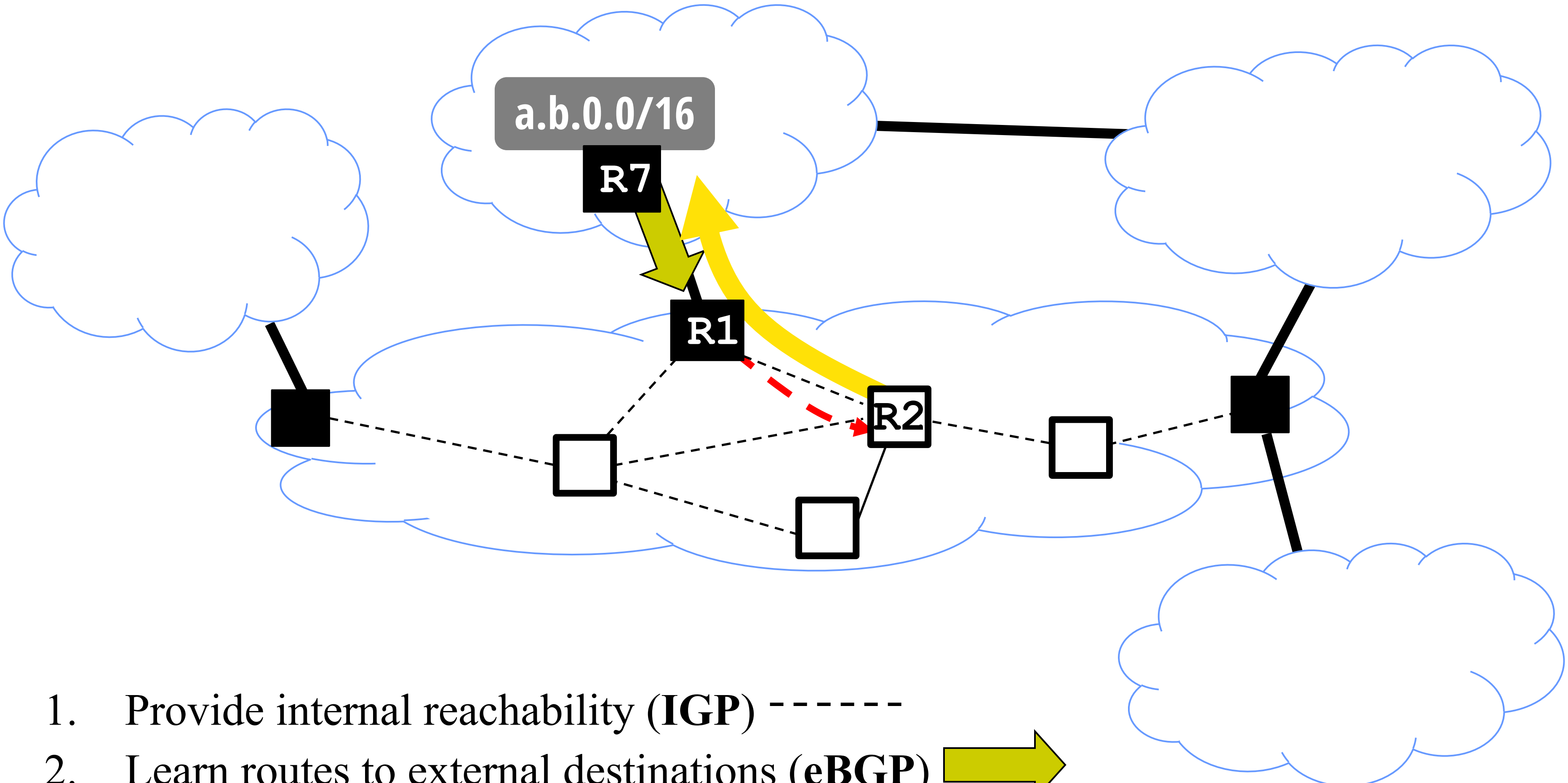
- eBGP: BGP sessions between border routers in different ASes
  - exchange routes to different destination prefixes
- iBGP: BGP sessions between border routers and other routers within the same AS
  - distribute externally learned routes internally
- IGP: "Interior Gateway Protocol" = Intradomain routing protocol
  - provide internal reachability
  - e.g., OSPF, RIP

Putting the pieces together



1. Provide internal reachability (**IGP**) - - - - - -
2. Learn routes to external destinations (**eBGP**)
3. Distribute externally learned routes internally (**iBGP**)
4. Travel shortest path to egress (IGP)

Putting the pieces together



1. Provide internal reachability (**IGP**) - - - - - -
2. Learn routes to external destinations (**eBGP**)
3. Distribute externally learned routes internally (**iBGP**)
4. Travel shortest path to egress (IGP)

Short Summary

- Every router in AS has two routing tables:
  - From IGP: next hop router to all *internal* destinations
  - From iBGP: egress router to all *external* destinations
- For internal addresses, just use IGP
  - Entry <internal destination, internal next hop>
- For external locations: use iBGP to find egress
  - Use IGP to find next hop to egress router

In Reality….

- ***Many different ways to configure a domain***
- Option #1: run iBGP between all routers in domain
  - Requires NxB iBGP connections. Could be a scaling issue.
  - This is what we will assume
- Option #2: only run iBGP between border routers
  - Inject external routes into IGP
- Option #3: Run a “route reflector” for iBGP
  - N rather than NxB connections

Many design questions....

- How do we ensure the routers in an AS “act as one”?
  - The role of border vs. interior routers?
  - Interaction between BGP and IGP
  - How is all this implemented?
    - Route updates and attributes

BGP protocol message types

- Open
- Keepalive
- Notification
- ...
- Update
  - Inform neighbor of new routes
  - Inform neighbor of updates to old routes
  - “Withdraw” a route that’s now inactive

Route Updates

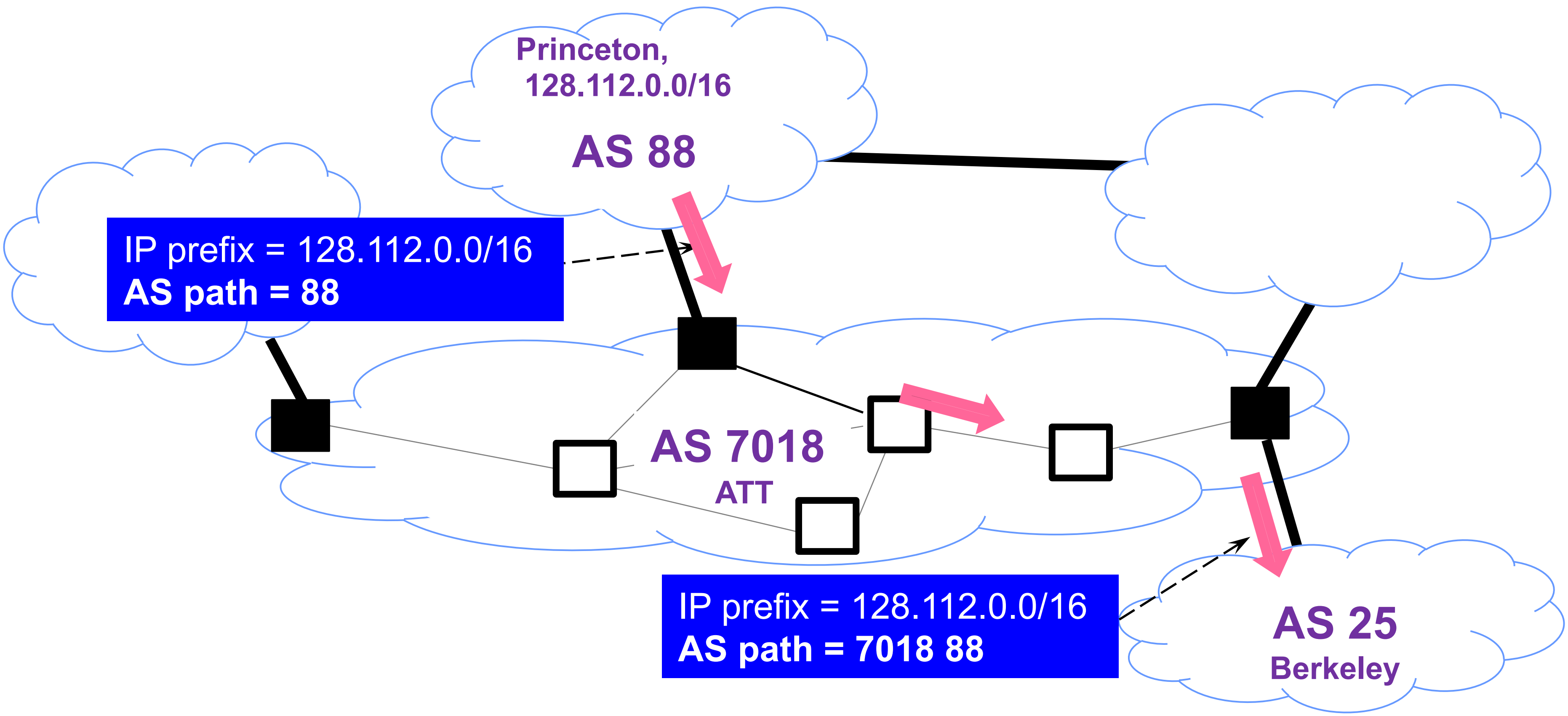
- Format *<IP prefix: route attributes>*
  - attributes describe properties of the route

Route Attributes

- General mechanism used to express properties about routes
  - Used in route selection/export decisions
- Some attributes are local to an AS
  - Not propagated in eBGP advertisements
- Others are propagated in eBGP route advertisements
- There are many standardized attributes in BGP
  - We will discuss four important ones

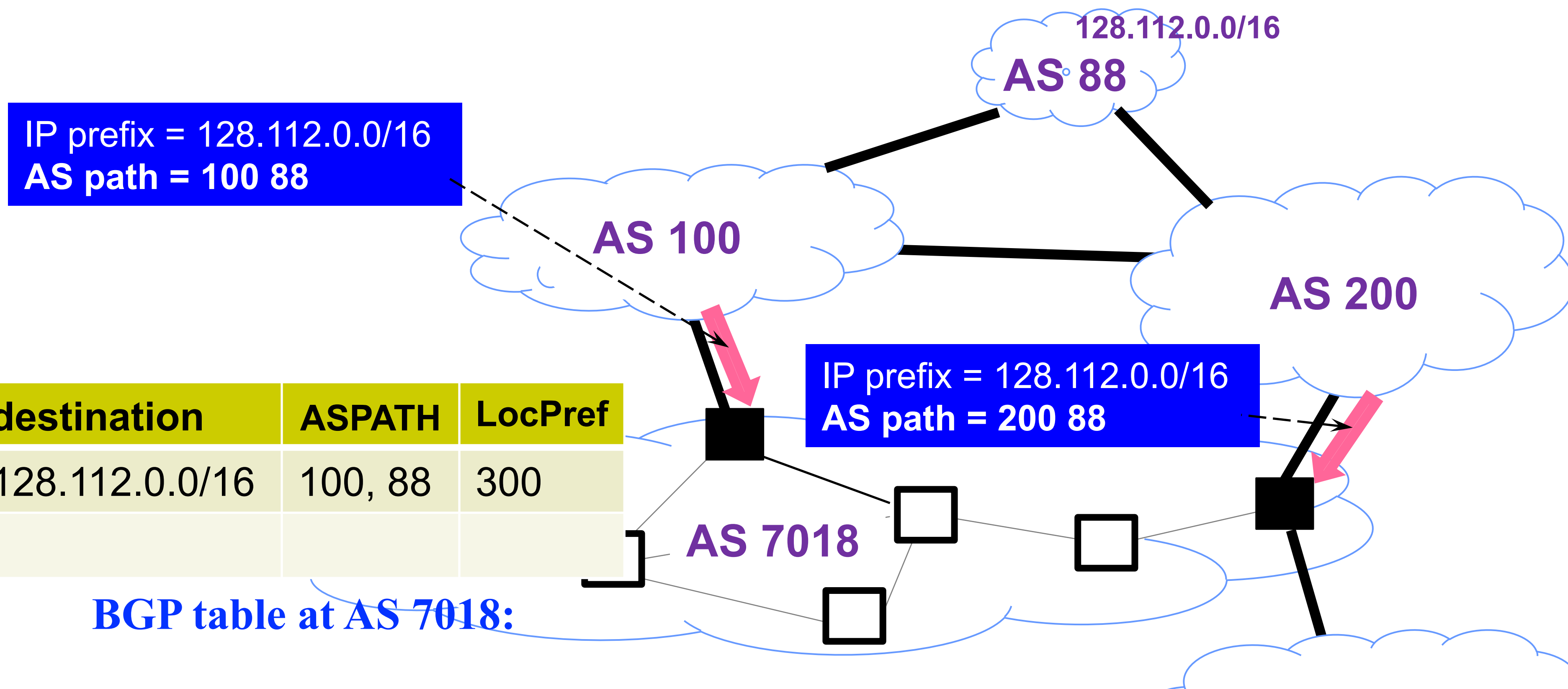
Attributes (1): ASPATH

- Path vector that lists all the ASes a route advertisement has traversed (in reverse order)
- Carried in route announcements



Attributes (2): LOCAL PREFERENCE

- Used to choose between different AS paths
- Local to an AS; carried only in iBGP messages
- The higher the value the more that route is preferred

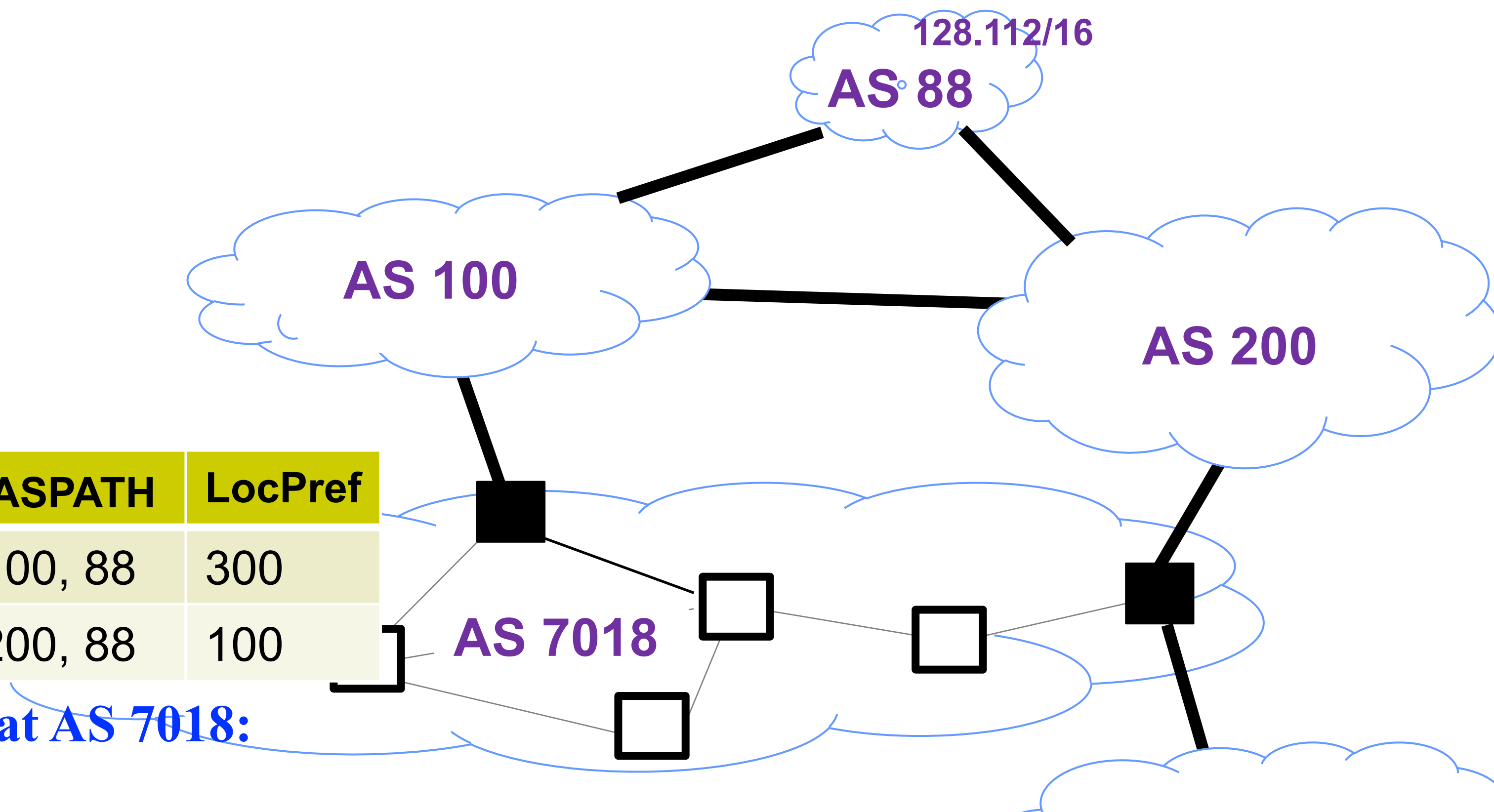


destination	ASPATH	LocPref
128.112.0.0/16	100, 88	300

BGP table at AS 7018:

Attributes (2): LOCAL PREFERENCE

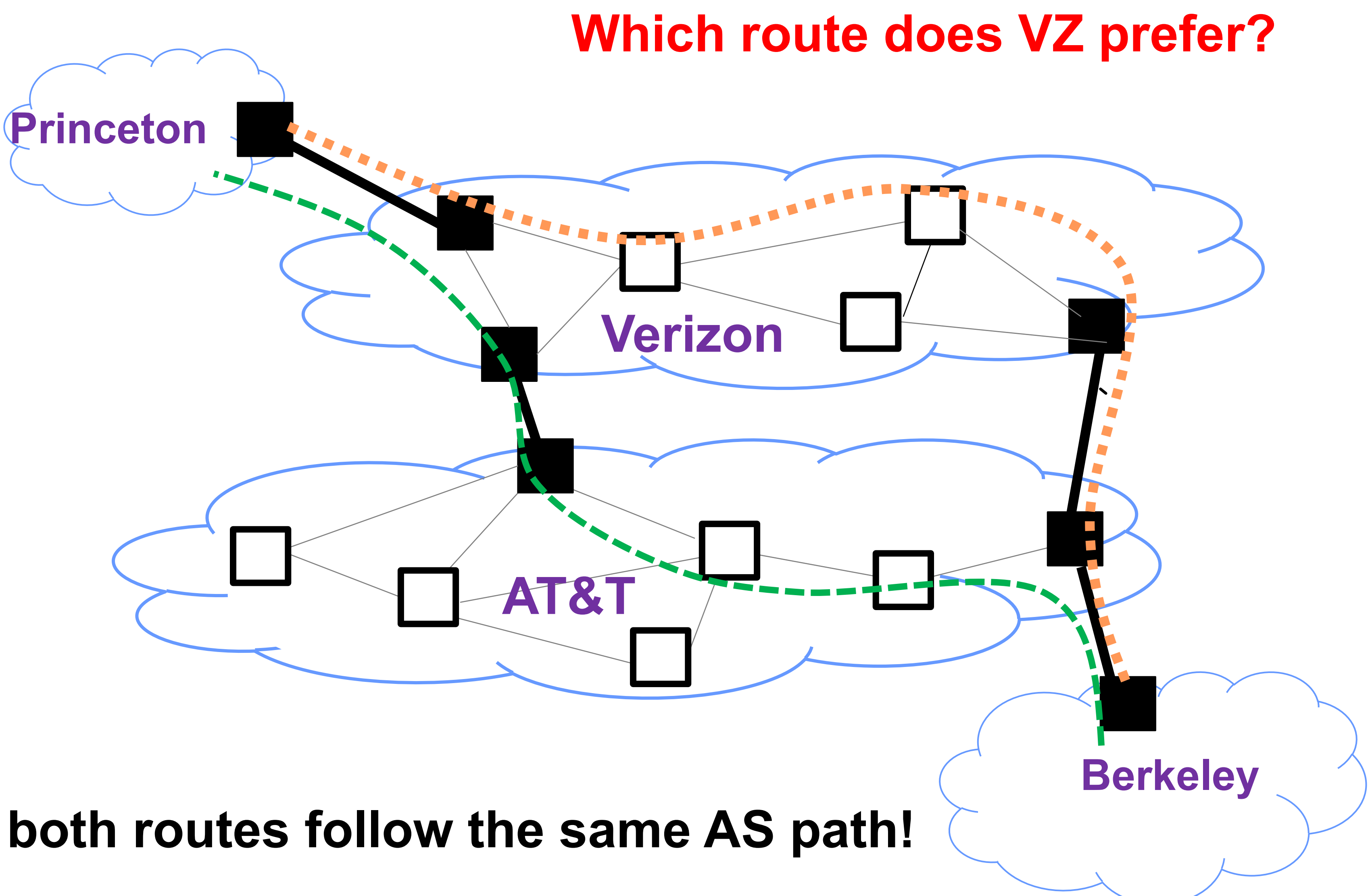
- Used to choose between different AS paths
- Local to an AS; carried only in iBGP messages
- The higher the value the more that route is preferred



destination	ASPATH	LocPref
12.112.0.0/16	100, 88	300
12.112.0.0/16	200, 88	100

BGP table at AS 7018:

In reality...



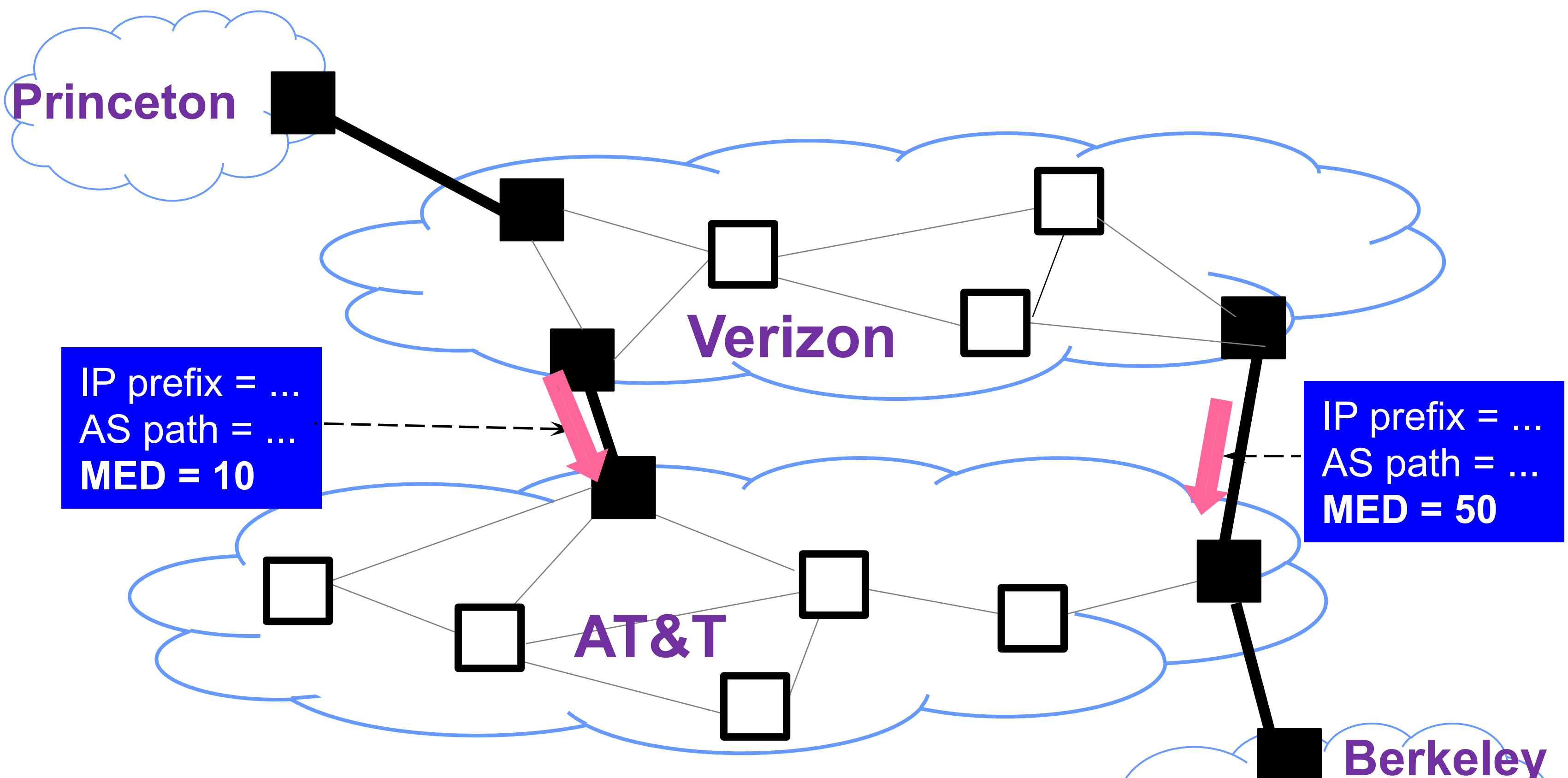
Note: both routes follow the same AS path!

Attributes (3) : `MED`

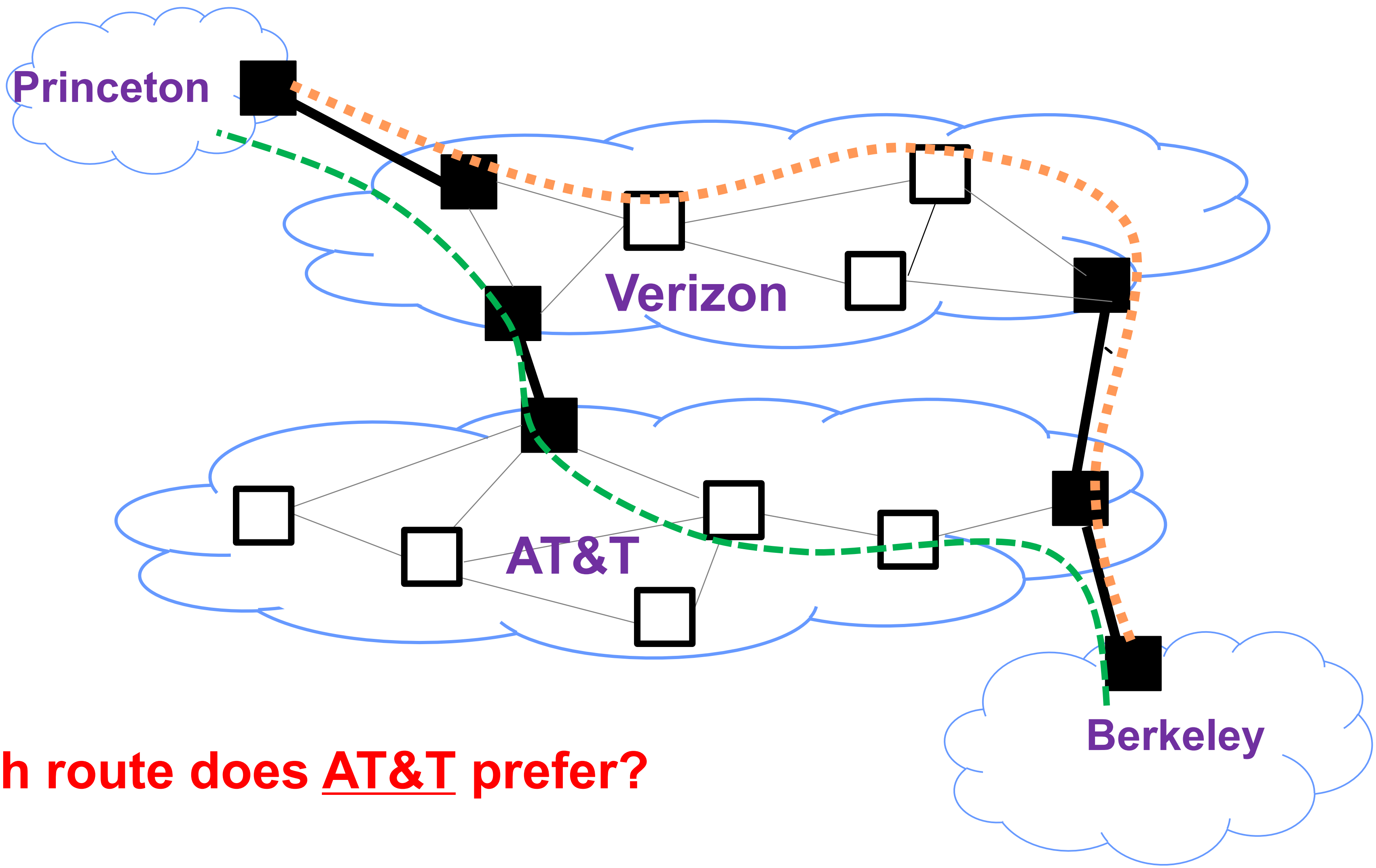
- MED = “Multi-Exit Discriminator”
- Used when ASes are interconnected via 2 or more links to specify how close a prefix is to the link it is announced on

Attributes (3) : MED

- AS announcing prefix sets MED (lower is better)
- AS receiving prefix (optionally!) uses MED to select link



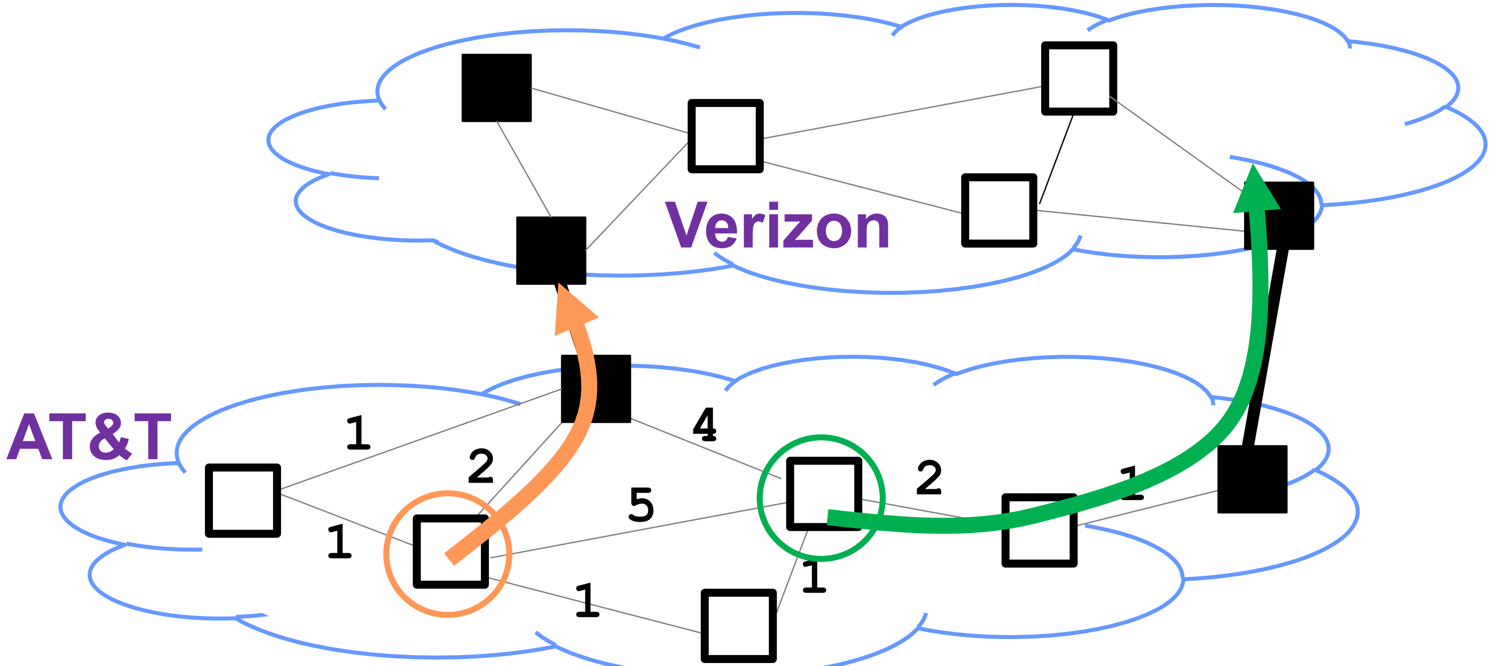
More reality...



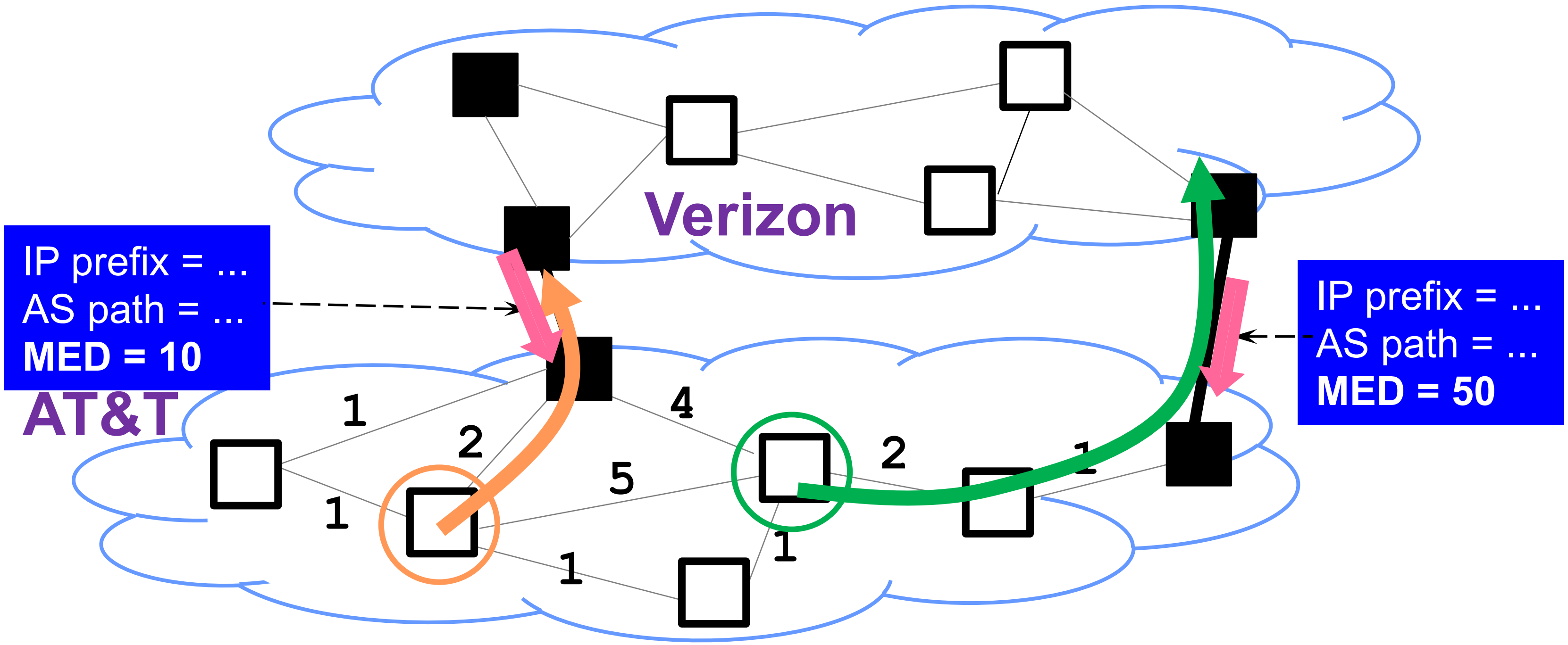
Which route does AT&T prefer?

Attributes (4): IGP cost

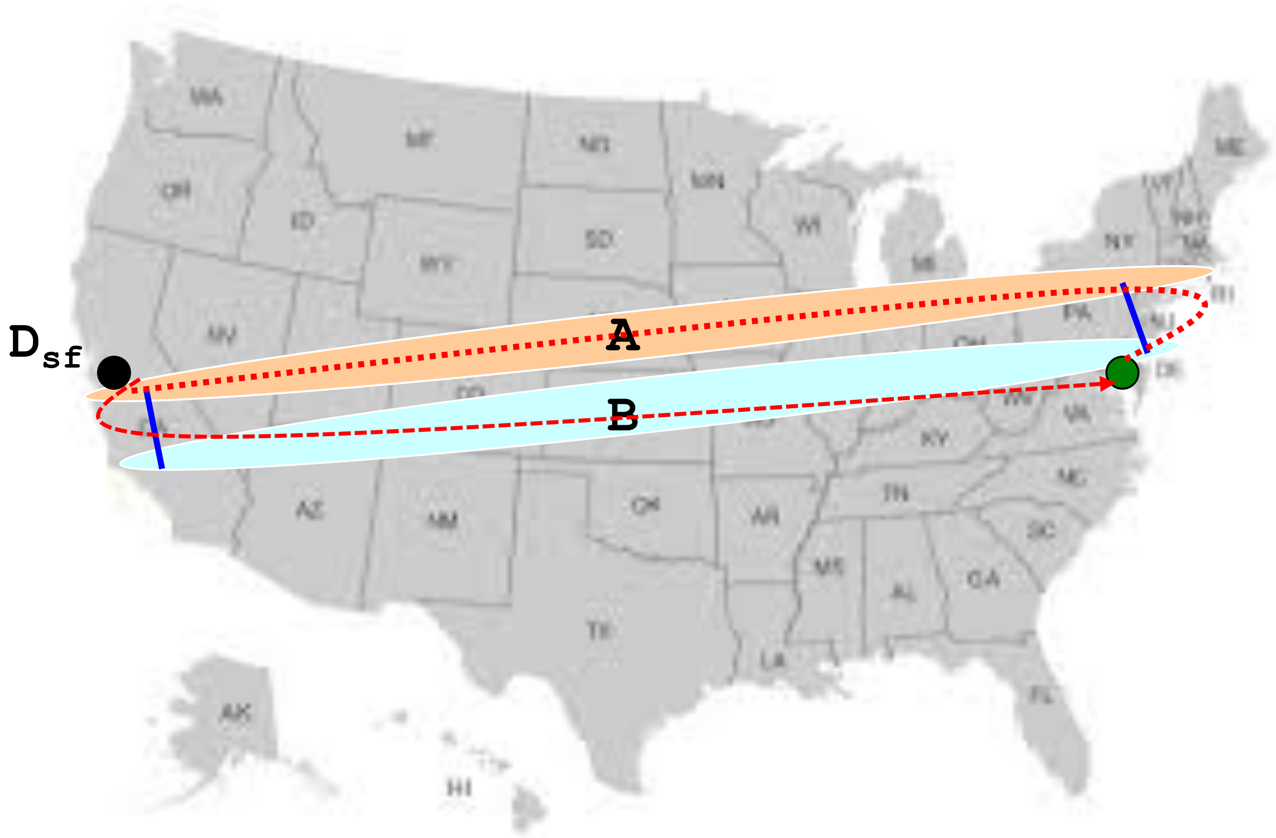
- Local to an AS
- Each router selects its closest border router
  - Closest based on IGP cost
  - a.k.a. "hot potato" routing



Note: IGP may conflict with MED



IGP-MED conflicts pretty common



Can lead to asymmetric paths!

Closing the loop...
Typical Selection Policy

- In decreasing order of priority
  - make/save money
  - maximize performance
  - minimize use of my network bandwidth
  - …
  - …

Closing the loop... Typical Selection Policy

- In decreasing order of priority
  - make/save money: LOCAL PREF (cust > peer > provider)
  - maximize performance: length of ASPATH
  - minimize use of my network bandwidth: “hot potato”, MED
  - …
  - …

Using Attributes

- Rules for route selection in priority order

Priority	Rule	Remarks
1	LOCAL PREF	Pick highest LOCAL PREF
2	ASPATH	Pick shortest ASPATH length
3	IGP path	Lowest IGP cost to next hop (egress router)
4	MED	MED preferred
5	Router ID	Smallest next-hop router's IP address as tie-breaker

Questions?

Outline

- Context
- Goals
- Approach
- Detailed design
- Limitations

Issues with BGP

- Security
- Performance (non?)issues
- Prone to misconfiguration
- Reachability and Convergence

Questions?

Taking Stock: We've done...

- An end-to-end overview of the Internet arch.

- A deep dive on how routing works (intra/inter)
  - Fundamental part of a network's control plane

- This week: back to the network data plane
  - **Today: what data packets look like at the IP layer**
  - Thursday: how routers forward these IP packets

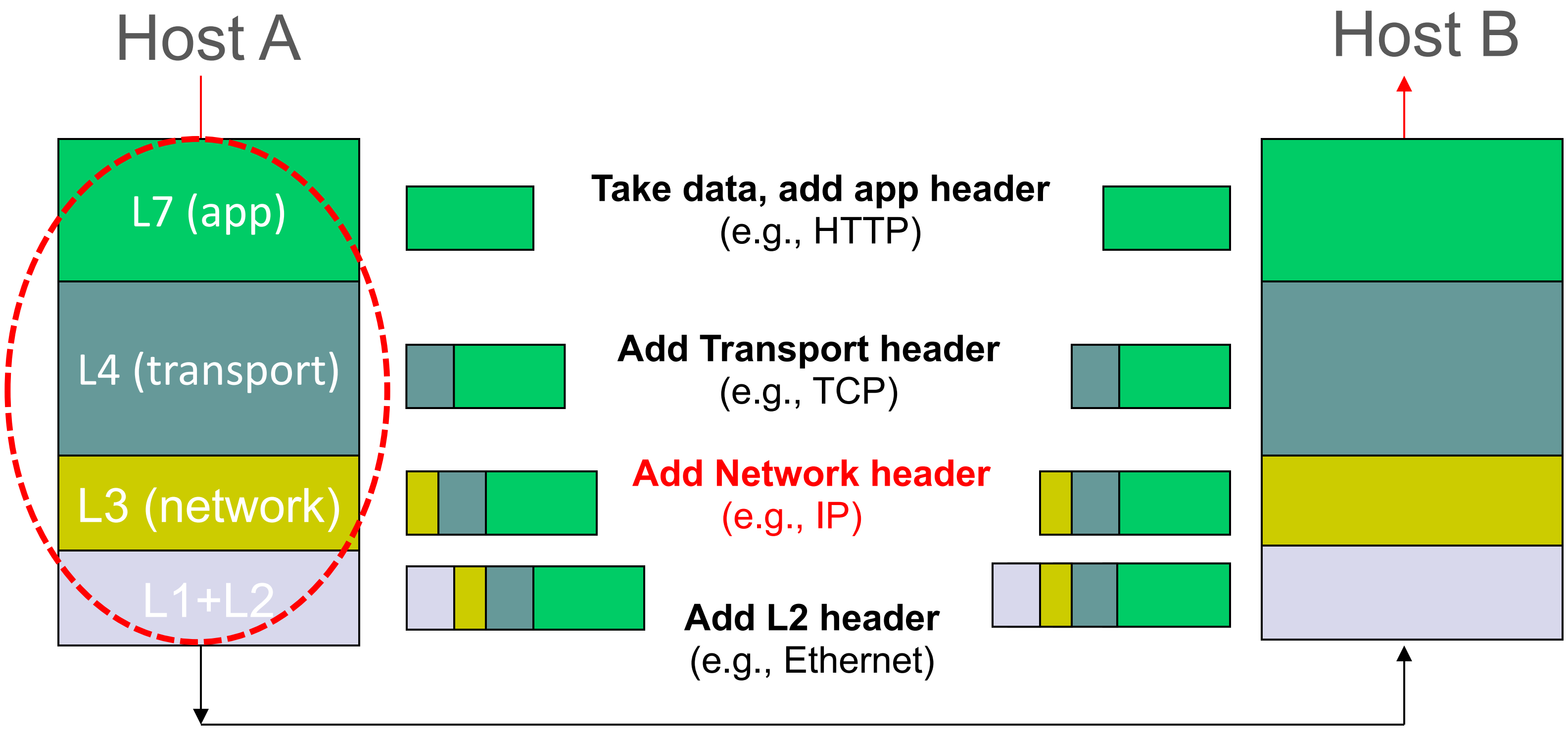
- At which point, you'll know how L3 works!

Let's design the IP header

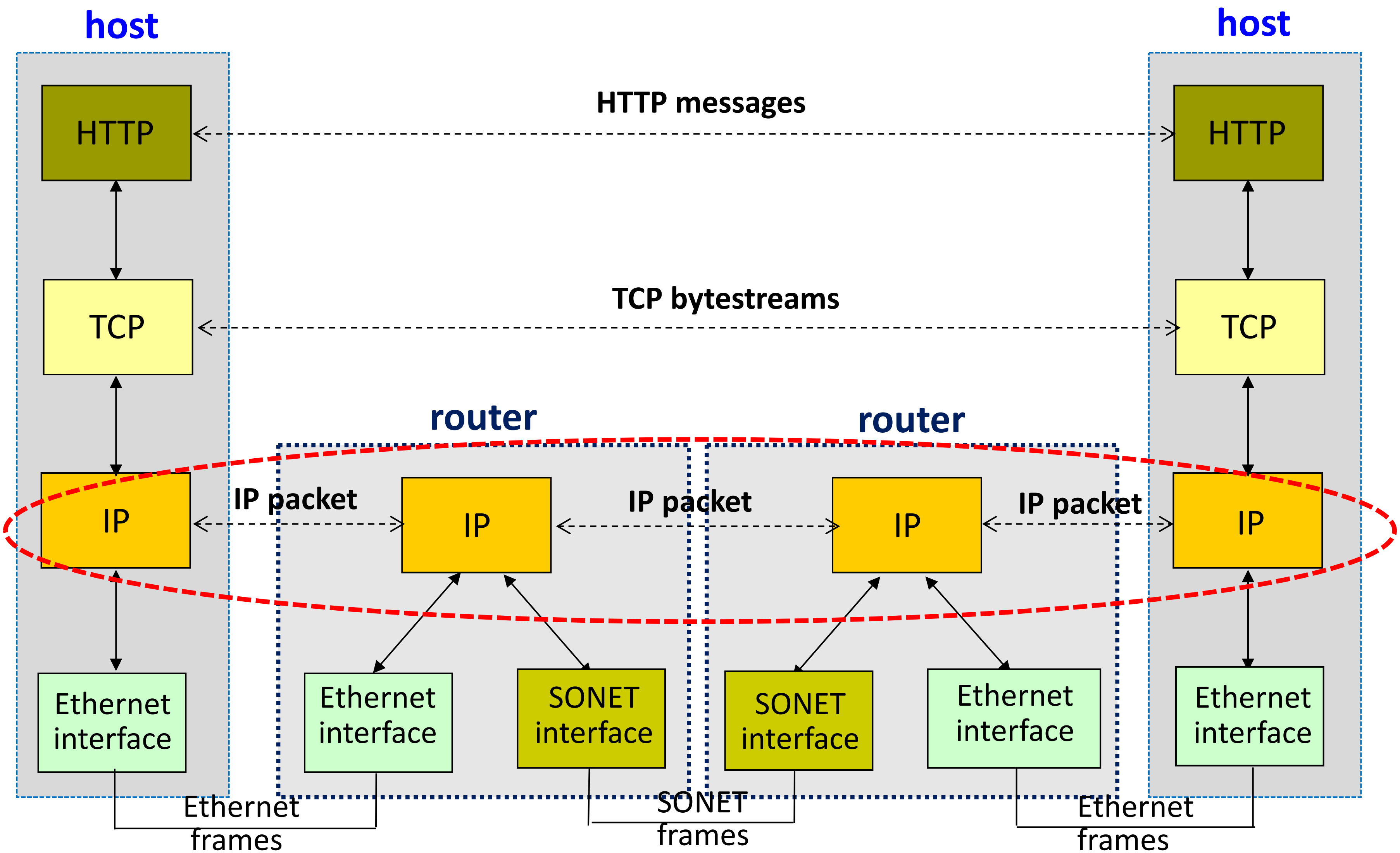
- **Syntax**: format of an IP packet
  - Nontrivial part: header
  - Rest is opaque payload

- **Semantics**: meaning of IP header fields
  - How they're processed

Recall: Layering



Recall: Hosts *vs.* Routers



Designing the IP header

- Think of the IP header as an interface
  - between the source and network (routers)
  - between the source and destination endhosts
- Designing an interface
  - what task(s) are we trying to accomplish?
  - what information is needed to do it?
- Header reflects information needed for basic tasks

What are these tasks?
(at a router, at the destination host)

- Parse packet *(router, dst host)*
- Forward packet to the L3 destination *(router)*
- Tell destination what to do next *(dst host)*
- Get responses back to the source *(dst host, router)*

- Deal with problems along the way *(router, dst host)*
- Specify any special handling *(router, dst host)*

Next: what information do we need?

Parse Packet Correctly

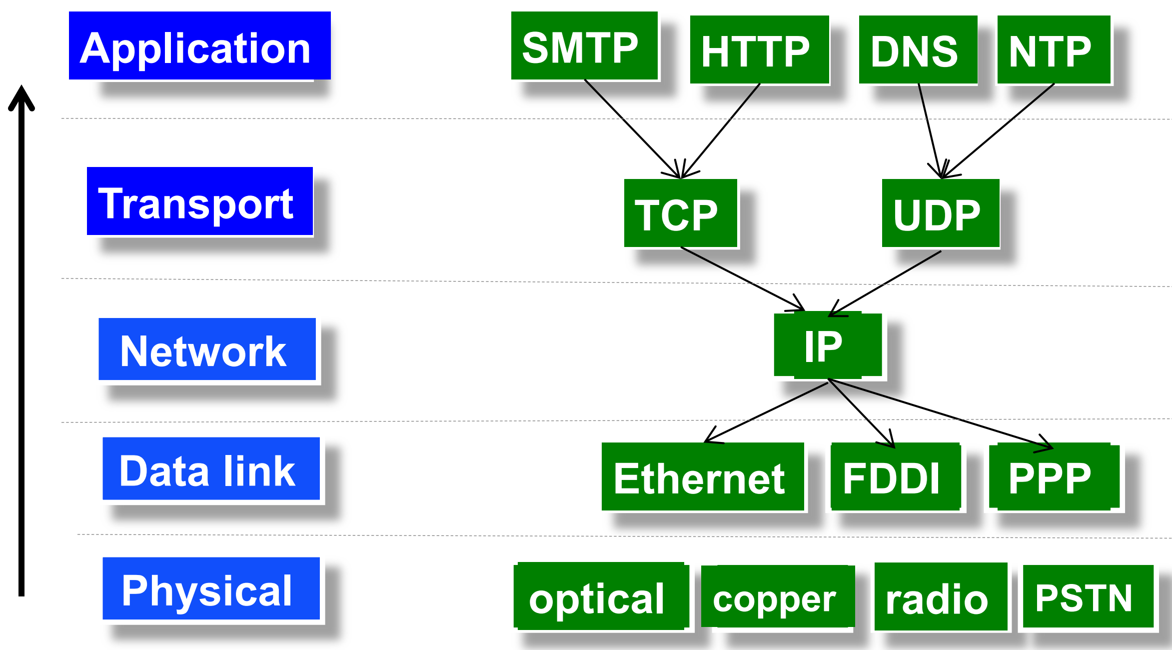
- What version of IP?
- Where does header end?
- Where does packet end?

Deliver packet to the L3 destination

- Provide destination address (duh!)

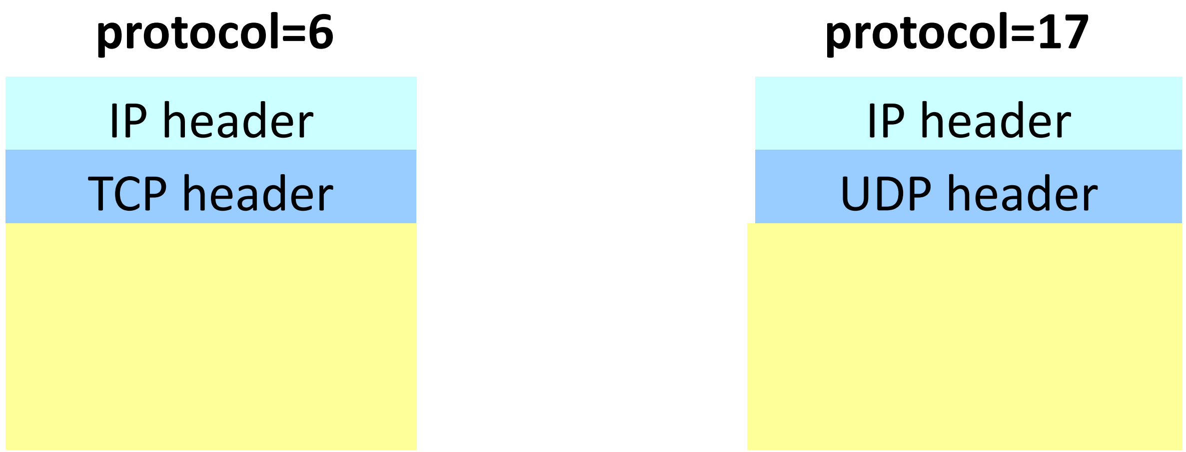
Tell the destination how to handle packet

- Indicate which protocol should handle packet next
- **Protocol** field: identifies the higher-level protocol
  - Important for de-multiplexing at receiving host



Tell the destination how to handle packet

- Protocol field that identifies the L4 protocol for this packet
- Common examples
  - "6" for the Transmission Control Protocol (TCP)
  - "17" for the User Datagram Protocol (UDP)



Get responses back to the source

- *Source IP address*

Where are we …

- Parse packet → *version, header length, packet length*
- Forward packet to the L3 dst → *destination address*
- Tell destination what to do next → *protocol field*
- Get responses back to the source → *source address*

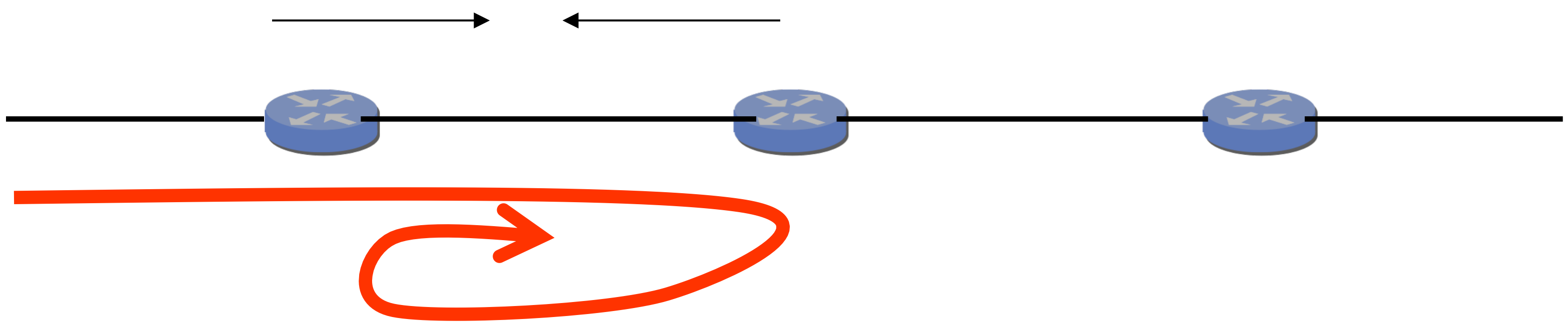
- Deal with problems along the way
- Specify any special handling

What problems?

- Loops
- Corruption
- Packet too large (> MTU)

Preventing Loops

- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity



- Time-to-Live (TTL) field
  - decremented at each hop, packet discarded if reaches 0
  - …and "time exceeded" message is sent to the source

Means header must include *source* IP address

Header Corruption

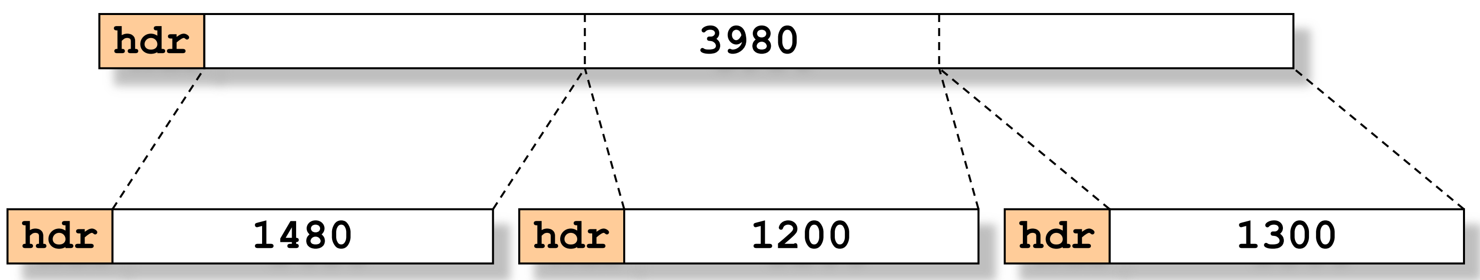
- Checksum
  - Small #bits used to check integrity of some data (e.g., hash)
  - Particular form of checksum <u>over packet header</u>

- If not correct, router/destination discards packets
  - So it doesn’t act on bogus information

- Checksum updated at every router
  - Why?
  - Why include TTL?
  - Why only header?

Fragmentation

- Every link has a "Maximum Transmission Unit" (MTU)
  - largest number of bits it can carry as one unit
- A router can split a packet into multiple "fragments" if the packet size exceeds the link's MTU



- Must reassemble to recover original packet

Details of fragmentation will be covered in section

Where are we …

- Parse packet → *version, header length, packet length*
- Forward packet to the L3 dst → *destination address*
- Tell destination what to do next → *protocol field*
- Get responses back to the source → *source address*

- Deal with problems along the way
  → *TTL, source address, checksum, frag. fields (TBD)*
- Specify any special handling

What forms of special treatment?

- Don’t treat all packets the same (“Type of Service”)
  - Idea: treat packets based on app/customer needs

- “Options”
  - Request advanced functionality for this packet

“Type of Service” (ToS)

- Originally: multiple bits used to request different forms of packet delivery
  - Based on priority, delay, throughput, reliability, or cost
  - Frequently redefined, never fully deployed
  - Only notion of priorities remained

- Today:
  - Differentiated Services Code Point (DSCP): traffic “classes”
  - Explicit Congestion Notification (ECN): a later lecture

Options

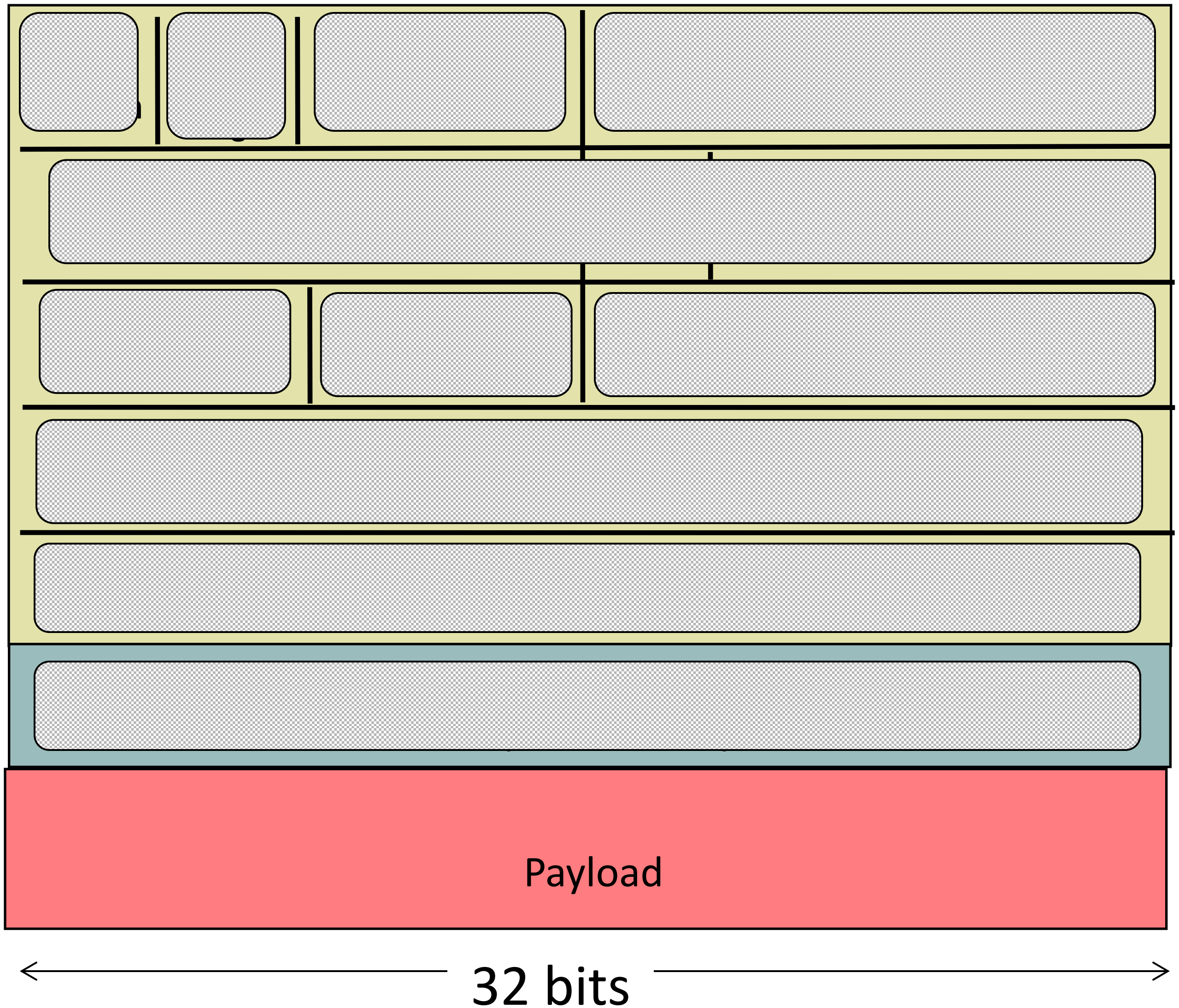
- **Optional directives to the network**
- **Examples**
  - Record Route, Source Route, Timestamp, ...
- More complex implementation
  - Leads to variable length headers
  - Often leads to higher processing overheads

Where are we …

- Parse packet → *version, header length, packet length*
- Forward packet to the L3 dst → *destination address*
- Tell destination what to do next → *protocol field*
- Get responses back to the source → *source address*

- Deal with problems along the way
  → *TTL, source address, checksum, frag. fields (TBD)*
- Specify any special handling → *ToS, options*

IP Packet Structure



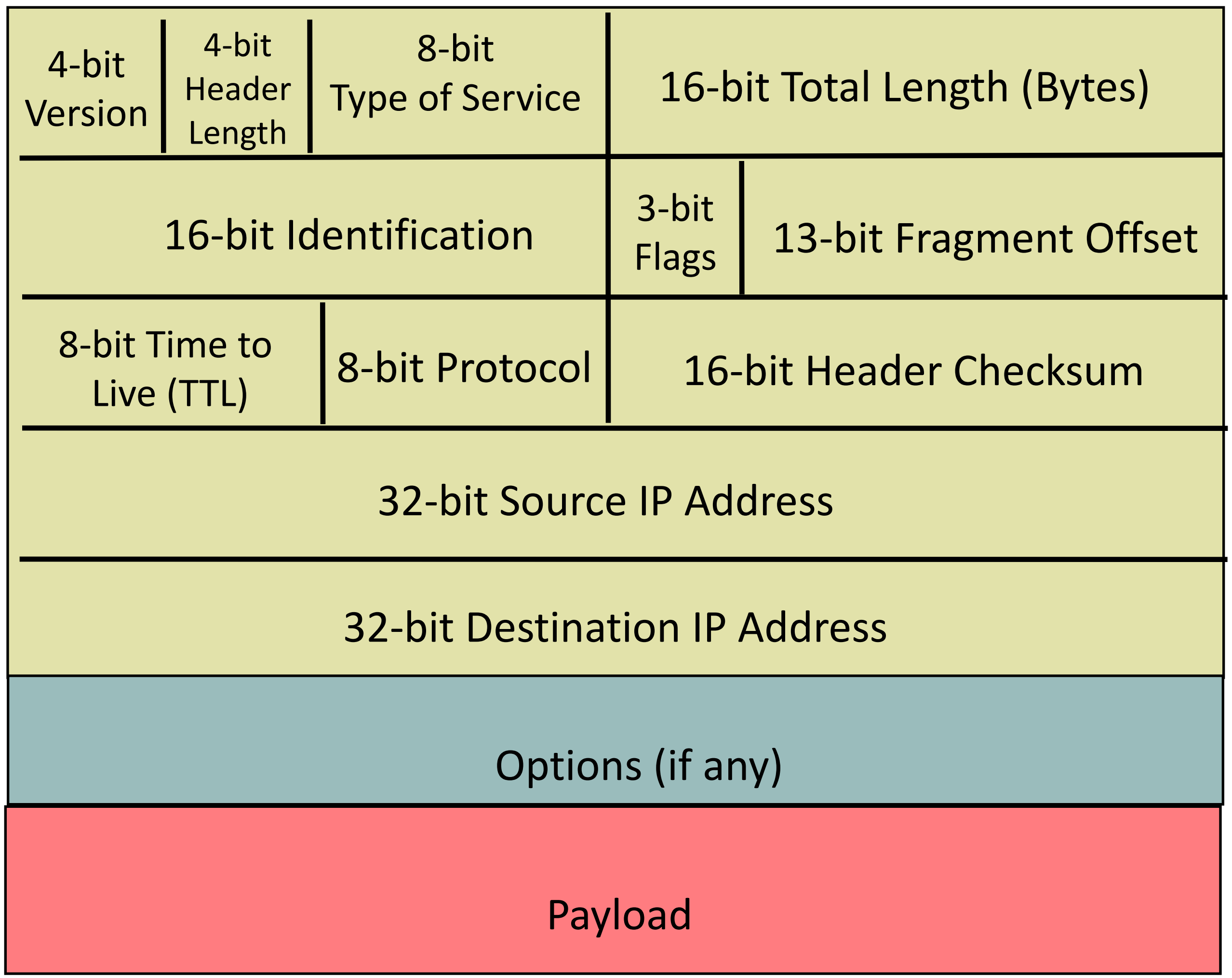
Two remaining topics

- IPv4 → IPv6
- Security implications of the IP header

IPv6

- Motivated (prematurely) by address exhaustion
  - Addresses ***four*** times as big
- Took to the opportunity to do some “spring cleaning”
  - Got rid of all fields that were not absolutely necessary
- Result is an elegant, if unambitious, protocol

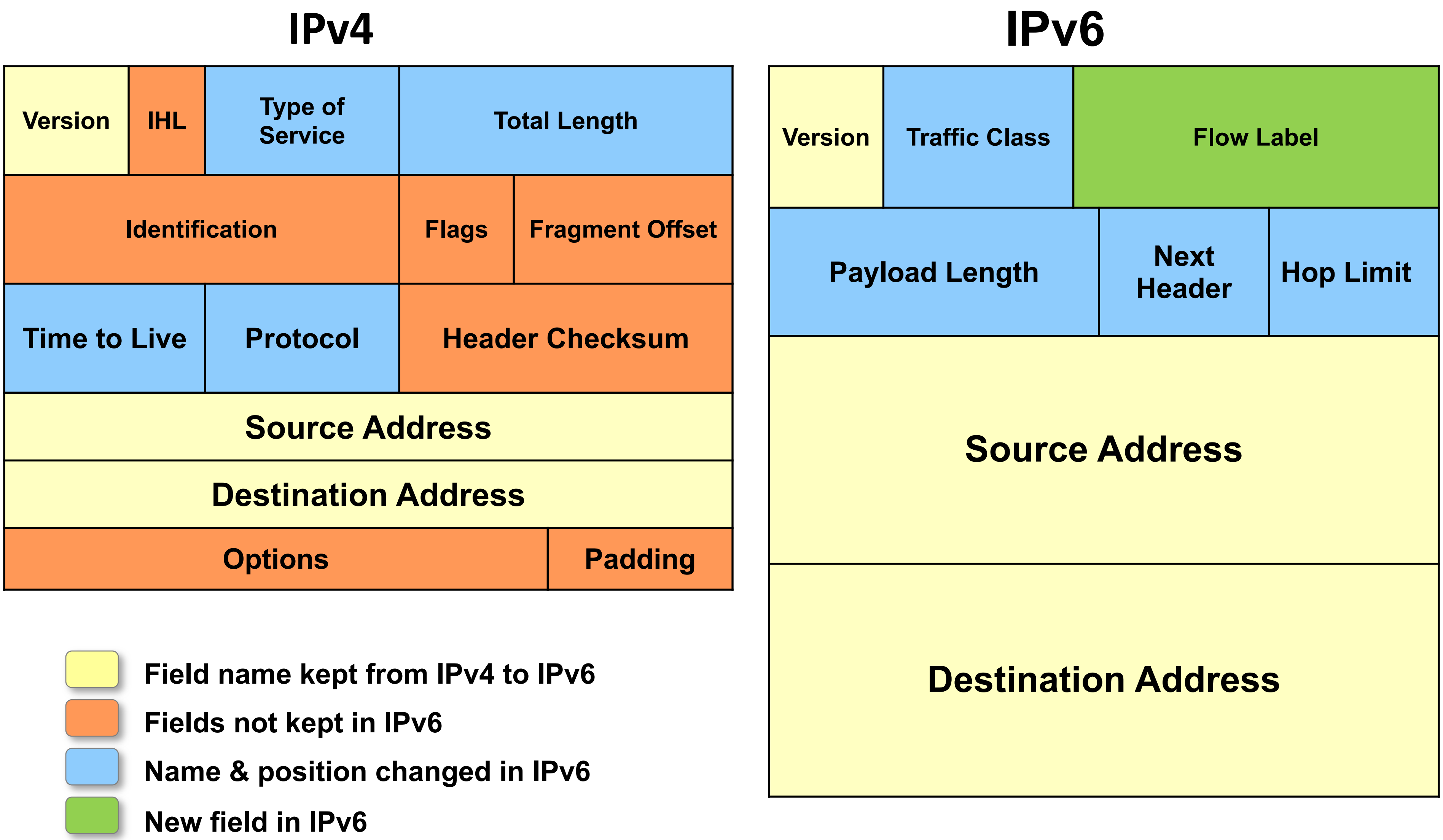
What “clean up” would you do?



Summary of Changes

- Expanded addresses
- Eliminated checksum ***(why?)***
- Eliminated fragmentation ***(why?)***
- New options mechanism → "next header"
- Eliminated header length ***(why?)***
- Added Flow Label
  - *Explicit* mechanism to denote related streams of packets

IPv4 and IPv6 Header Comparison



Philosophy of Changes

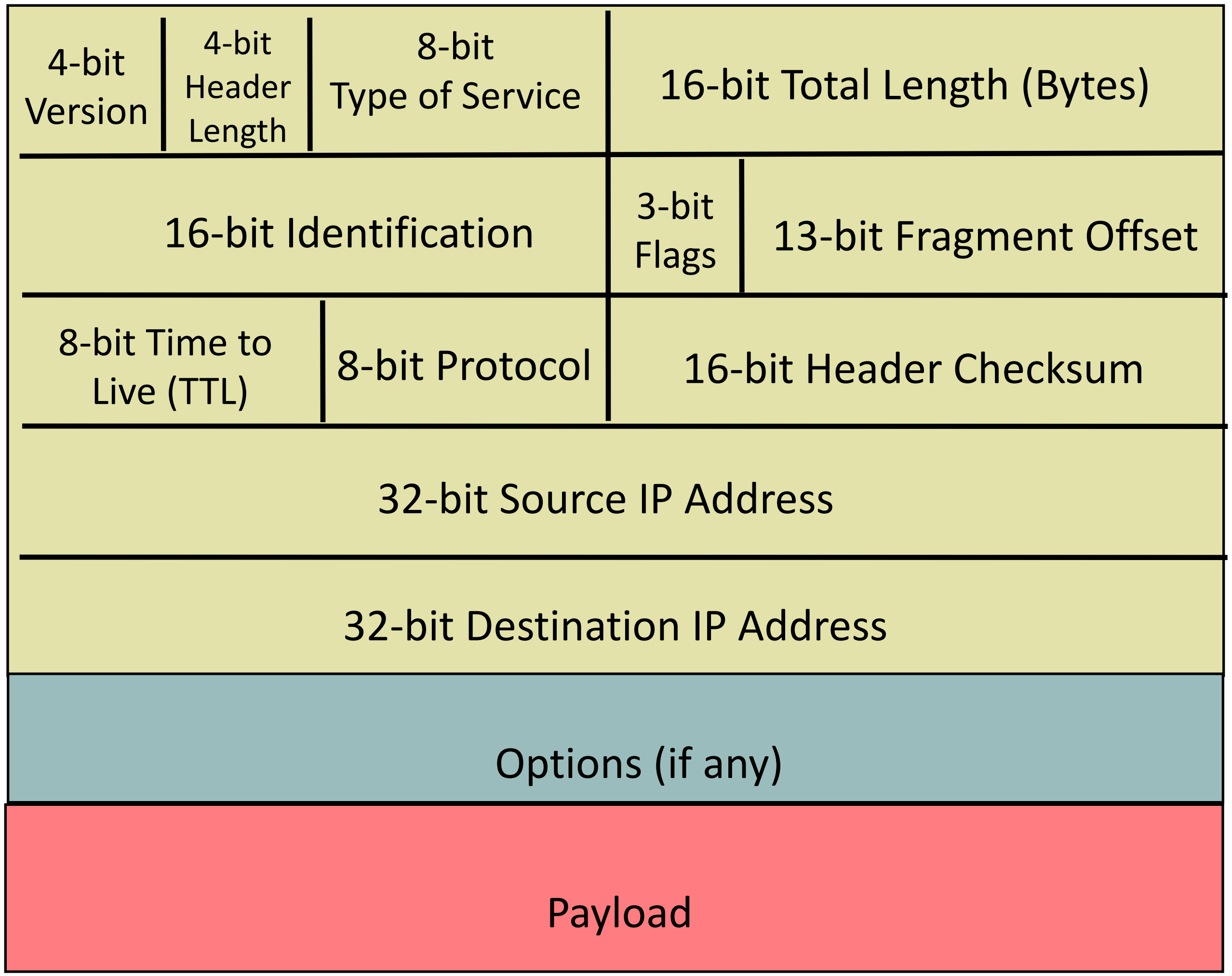
- Don't deal with problems: leave to ends
  - Eliminated fragmentation
  - Eliminated checksum
  - Why retain TTL?
- Simplify:
  - Got rid of options
  - Got rid of IP header length
- While still allowing extensibility
  - general next-header approach
  - general flow label for packet

Quick Security Analysis of IP Header

Focus on Sender Attacks

- Vulnerabilities a sender can exploit
- Note: not a comprehensive view of potential attacks!
  - For example, we'll ignore attackers other than the sender

IP Packet Structure



IP Address Integrity

- Source address should be the sending host
  - But who’s checking?
  - You could send packets with any source you want

Implications of IP Address Integrity

- Why would someone use a bogus source address?

- Attack the destination
  - Send excessive packets, overload network path to destination
  - But: victim can identify/filter you by the source address
  - Hence, evade detection by putting different source addresses in the packets you send ("spoofing")

- Or: as a way to bother the spoofed host
  - Spoofed host is wrongly blamed
  - Spoofed host may receive return traffic from the receiver(s)

Security Implications of TOS?

- Attacker sets TOS priority for their traffic?
  - Network *prefers* attack traffic

- What if the network charges for TOS traffic …
  - … and attacker spoofs the victim's source address?

- Today, mostly set/used by operators, not end-hosts

Security Implications of Fragmentation?

- Send packets larger than MTU → make routers do extra work
  - Can lead to resource exhaustion

More Security Implications

- IP options
  - Misuse: e.g., **Source Route** lets sender control the path taken through network - say, to sidestep a firewall
  - Processing IP options often processed in router's slow path → attacker can try to overload routers (coming up)
  - Routers sometimes configured to **drop** packets with options

Security Implications of TTL? (8 bits)

- Allows discovery of topology (a la *traceroute*)

- Initial value is somewhat distinctive to sender's operating systems. This plus other such initializations allow OS fingerprinting …
  - Which allow attacker to infer its likely vulnerabilities

Other Security Implications?

- No apparent problems with protocol field (8 bits)
  - It's just a de-muxing handle
  - If set incorrectly, next layer will find packet ill-formed
- Bad IP checksum field (16 bits) will cause packet to be discarded by the network
  - Not an effective attack…

Recap: IP header design

- More nuanced than it first seems!
- Must juggle multiple goals
  - Efficient implementation
  - Security
  - Future needs

Questions?