Wrapping up BGP and the IP header

Spring 2024

Sylvia Ratnasamy

CS168.io

Outline

- Wrapping up BGP
  - Context
  - Goals
  - Approach
    - Wrap up Gao-Rexford
  - Protocol design
  - Limitations
- Designing the IP header

Recall: computing paths on the AS graph

Recall: computing paths on the AS graph

- Nodes are Autonomous Systems (AS)

Recall: computing paths on the AS graph

- Nodes are Autonomous Systems (AS)
- Edges reflect physical connections & biz relationships
  - Customers pay providers
  - Peers don't pay each other

Recall: computing paths on the AS graph

- Nodes are Autonomous Systems (AS)
- Edges reflect physical connections & biz relationships
  - Customers pay providers
  - Peers don’t pay each other
- Paths are selected based on policy

Recall: computing paths on the AS graph

- Nodes are Autonomous Systems (AS)
- Edges reflect physical connections & biz relationships
  - Customers pay providers
  - Peers don’t pay each other
- Paths are selected based on policy
- Policy reflects business goals (i.e., how money flows)
  - “Only carry traffic if you’re getting paid for it”
  - “Try and make/save money when sending traffic”

Recall: BGP

- Protocol that implements interdomain routing

Recall: BGP

- Protocol that implements interdomain routing
- Extends Distance-Vector

Recall: BGP

- Protocol that implements interdomain routing
- Extends Distance-Vector
- Basic idea
  - Destinations are prefixes
  - Each AS advertises its path to a prefix
  - Policy dictates which paths an AS selects (“import policy”) and which paths it advertises (“export policy”)

Recall: BGP

- Protocol that implements interdomain routing
- Extends Distance-Vector
- Basic idea
  - Destinations are prefixes
  - Each AS advertises its path to a prefix
  - Policy dictates which paths an AS selects (“import policy”) and which paths it advertises (“export policy”)
- Gao-Rexford rules tell us what import/export policies will achieve business goals

Gao-Rexford Rule: Import policy

- When importing (selecting) a route to a destination, pick route advertised by customer > peer > provider

Gao-Rexford Rule: Import policy

- When importing (selecting) a route to a destination, pick route advertised by customer > peer > provider

- In practice, ASes use additional rules to break ties

Gao-Rexford Rule: Import policy

- When importing (selecting) a route to a destination, pick route advertised by customer > peer > provider
- In practice, ASes use additional rules to break ties
- Typical order of priority:
  - First, make/save money (G-R rule)
  - Then, maximize performance
  - Then, minimize use of my network bandwidth
  - ....

Gao-Rexford Rule: Import policy

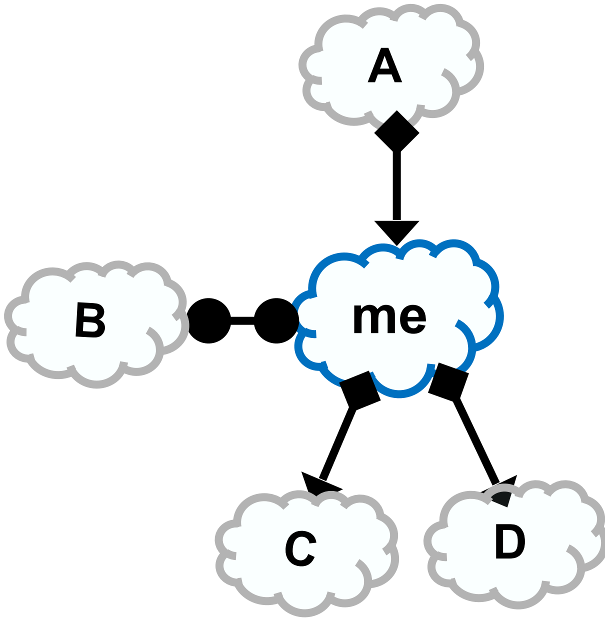
- When importing (selecting) a route to a destination, pick route advertised by customer > peer > provider

- In practice, ASes use additional rules to break ties

- Typical order of priority:
  - First, make/save money (G-R rule)
  - Then, maximize performance
  - Then, minimize use of my network bandwidth
  - ....

Gao-Rexford Rules: Export policy

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



Gao-Rexford Rules: Implication

- Under two assumptions about the AS graph (coming up), if all ASes follow Gao-Rexford, we can guarantee:
  - **Reachability**: any two ASes can communicate
  - **Convergence**: all routers agree on paths
- The above hold in steady state

Steady State and Convergence

- Steady state essentially means no changes
  - No addition/removal/failure of nodes, links, destinations
  - No change in policies, *etc.*

Steady State and Convergence

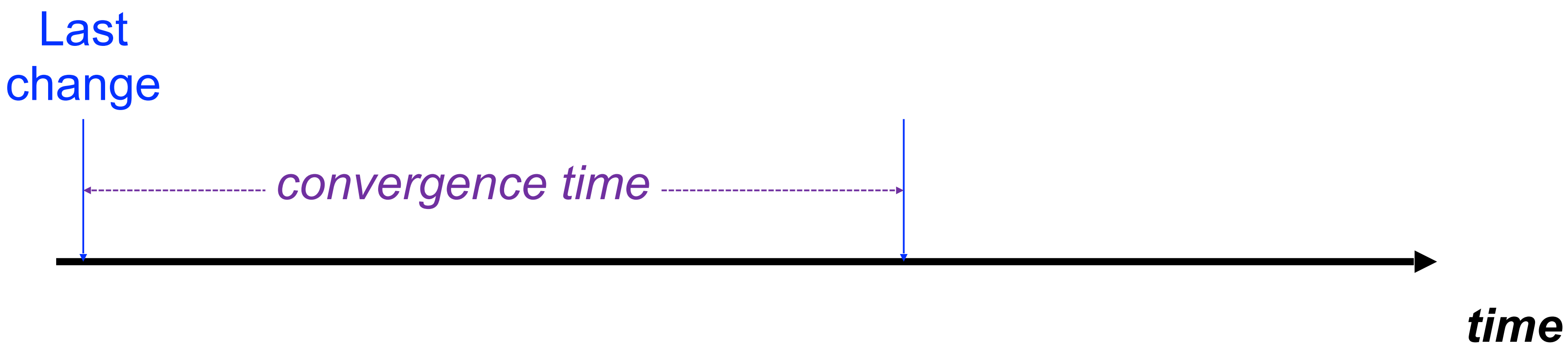
- Steady state essentially means no changes
  - No addition/removal/failure of nodes, links, destinations
  - No change in policies, *etc.*

Steady State and Convergence

- Steady state essentially means no changes
  - No addition/removal/failure of nodes, links, destinations
  - No change in policies, *etc.*

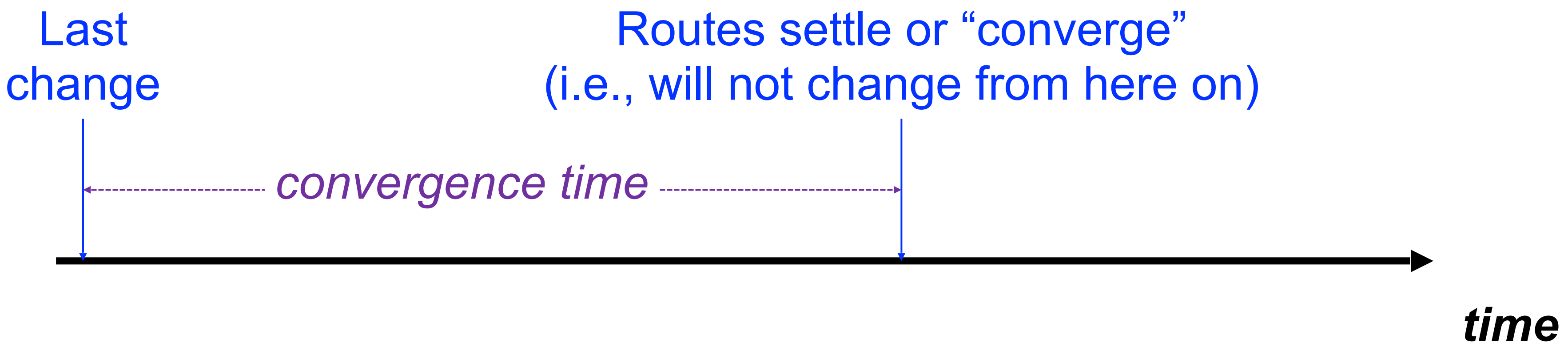
Steady State and Convergence

- Steady state essentially means no changes
  - No addition/removal/failure of nodes, links, destinations
  - No change in policies, *etc.*



Steady State and Convergence

- Steady state essentially means no changes
  - No addition/removal/failure of nodes, links, destinations
  - No change in policies, *etc.*



Two assumptions

Two assumptions

#1 The graph of customer-provider relationships is acyclic

- Cannot have A→B→...→C and then C→A (customer → provider)
- Means one can arrange providers in a hierarchy
- Note: OK if peering relationships are cyclic (A-B, B-C, C-A)

Gao-Rexford Rules: Implication

- Under two assumptions about the AS graph, if all ASes follow Gao-Rexford, we can guarantee:
  - **Reachability**: any two ASes can communicate
  - **Convergence**: all routers agree on paths
- The above hold in steady state

Gao-Rexford Rules: Implication

- Under two assumptions about the AS graph, if all ASes follow Gao-Rexford, we can guarantee:
  - **Reachability**: any two ASes can communicate
  - **Convergence**: all routers agree on paths
- The above hold in steady state
- The above are <u>not</u> guaranteed for general policies!
  - (You'll see an example of this in section)

Recap

Recap

- Policy is implemented by choosing which routes we import and which ones we export

Recap

- Policy is implemented by choosing which routes we import and which ones we export

- Gao-Rexford rules tell us which routes to import/export in order to make/save money

Recap

- Policy is implemented by choosing which routes we import and which ones we export

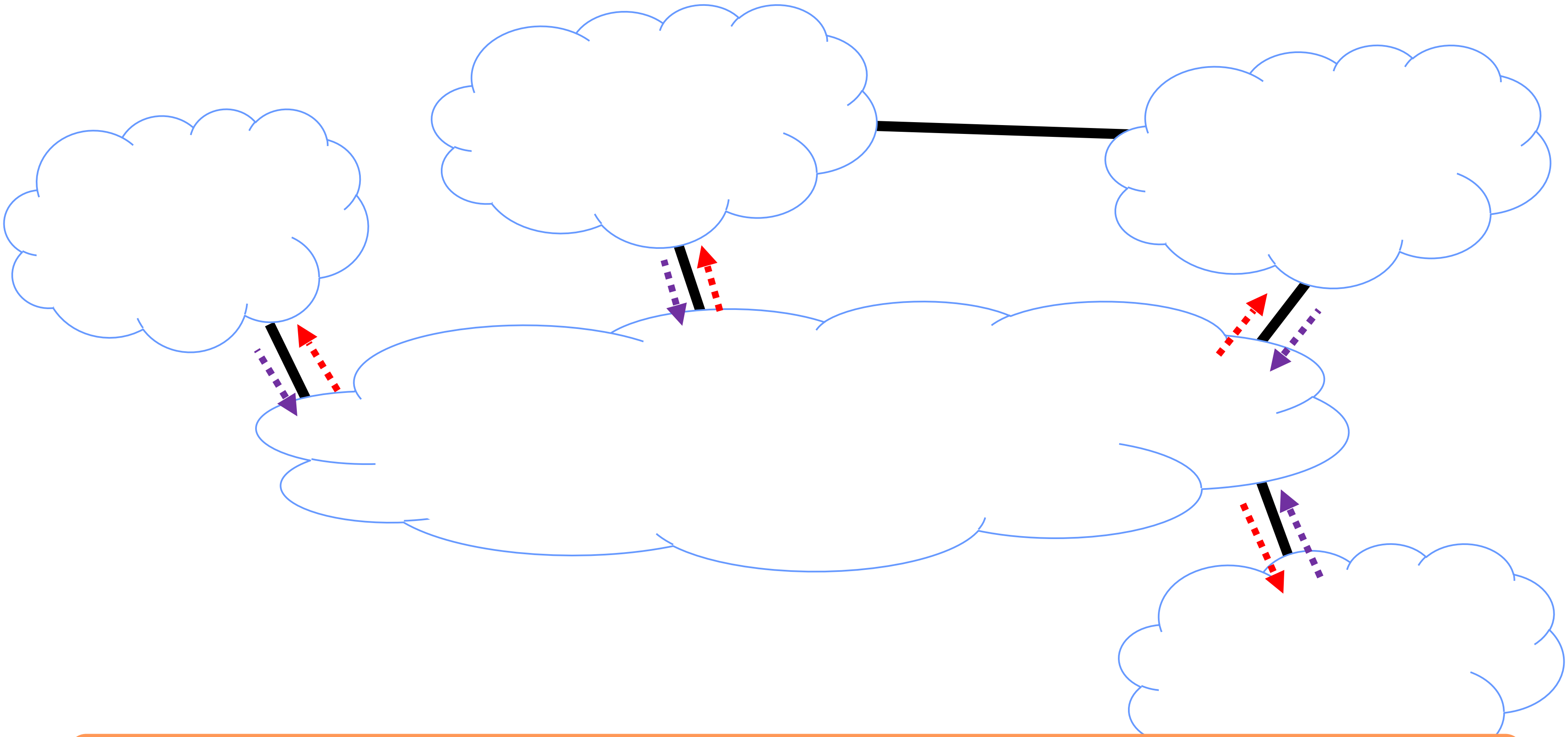
- Gao-Rexford rules tell us which routes to import/export in order to make/save money

- Good stuff happens when you follow G-R rules

Outline

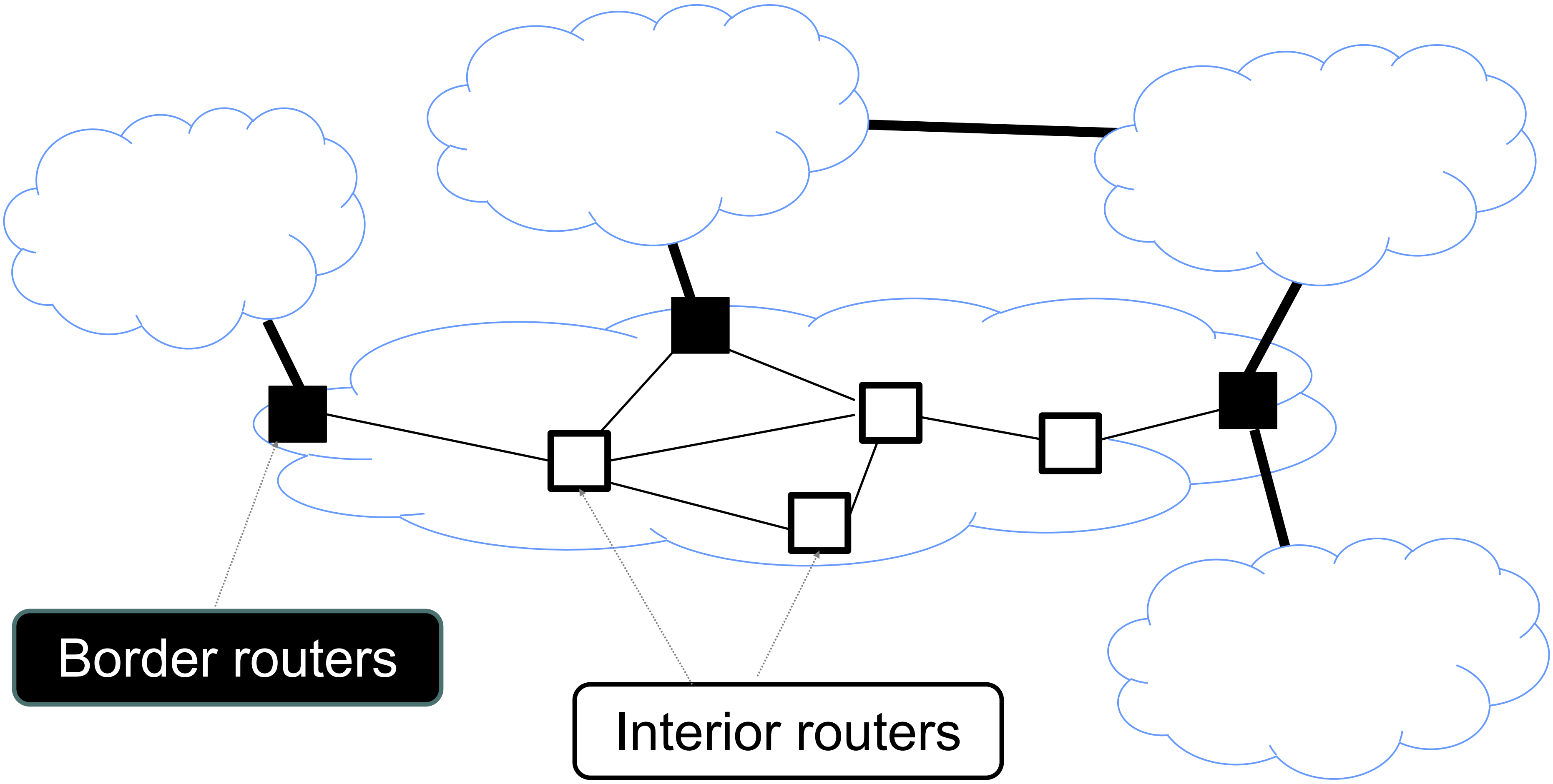
- Wrapping up BGP
  - Context
  - Goals
  - Approach
    - Wrap up Gao-Rexford
  - Protocol design
  - Limitations
- Designing the IP header

So far: our model of the AS graph



An AS advertises routes to its neighbor ASes

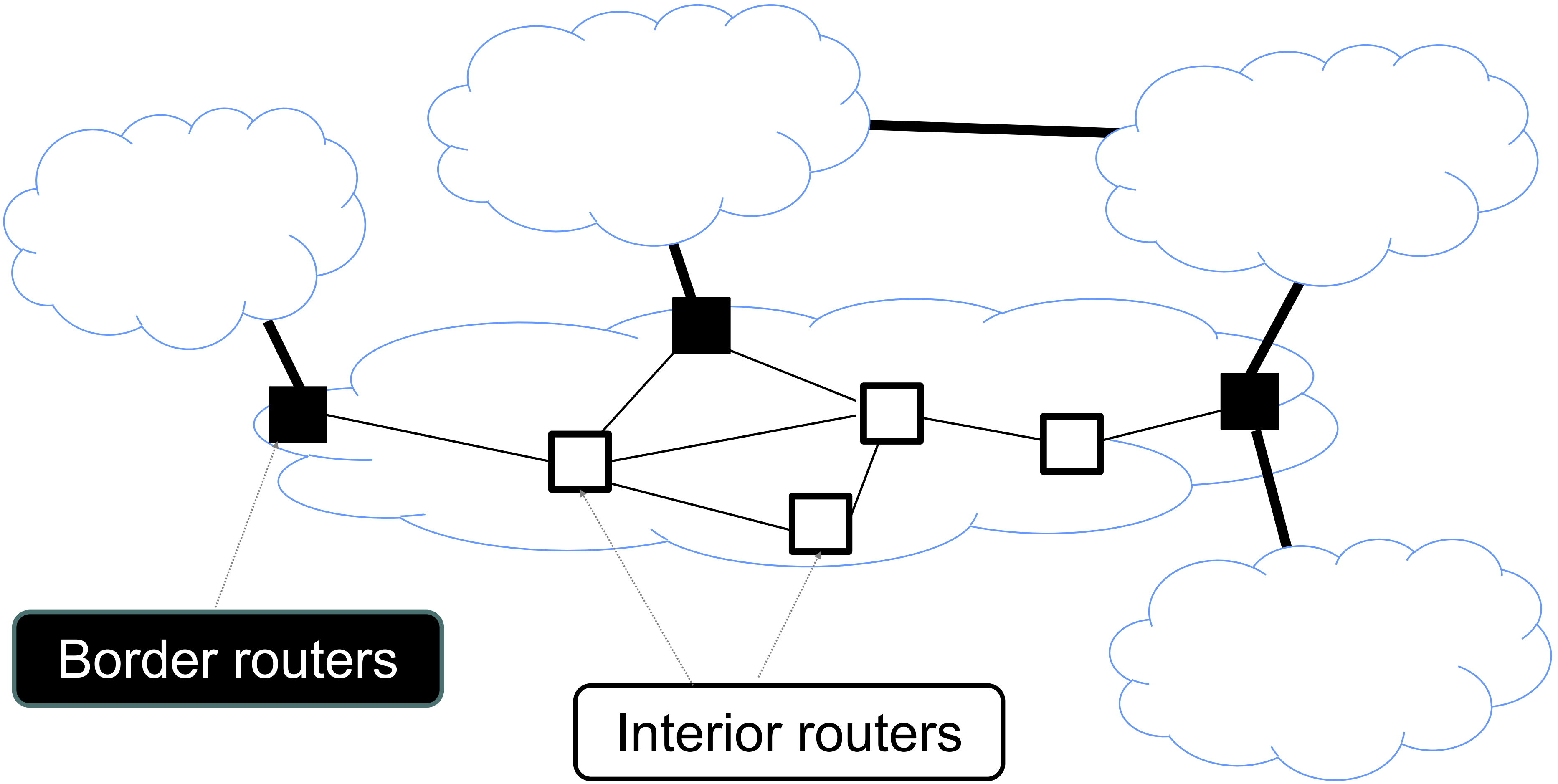
In reality...



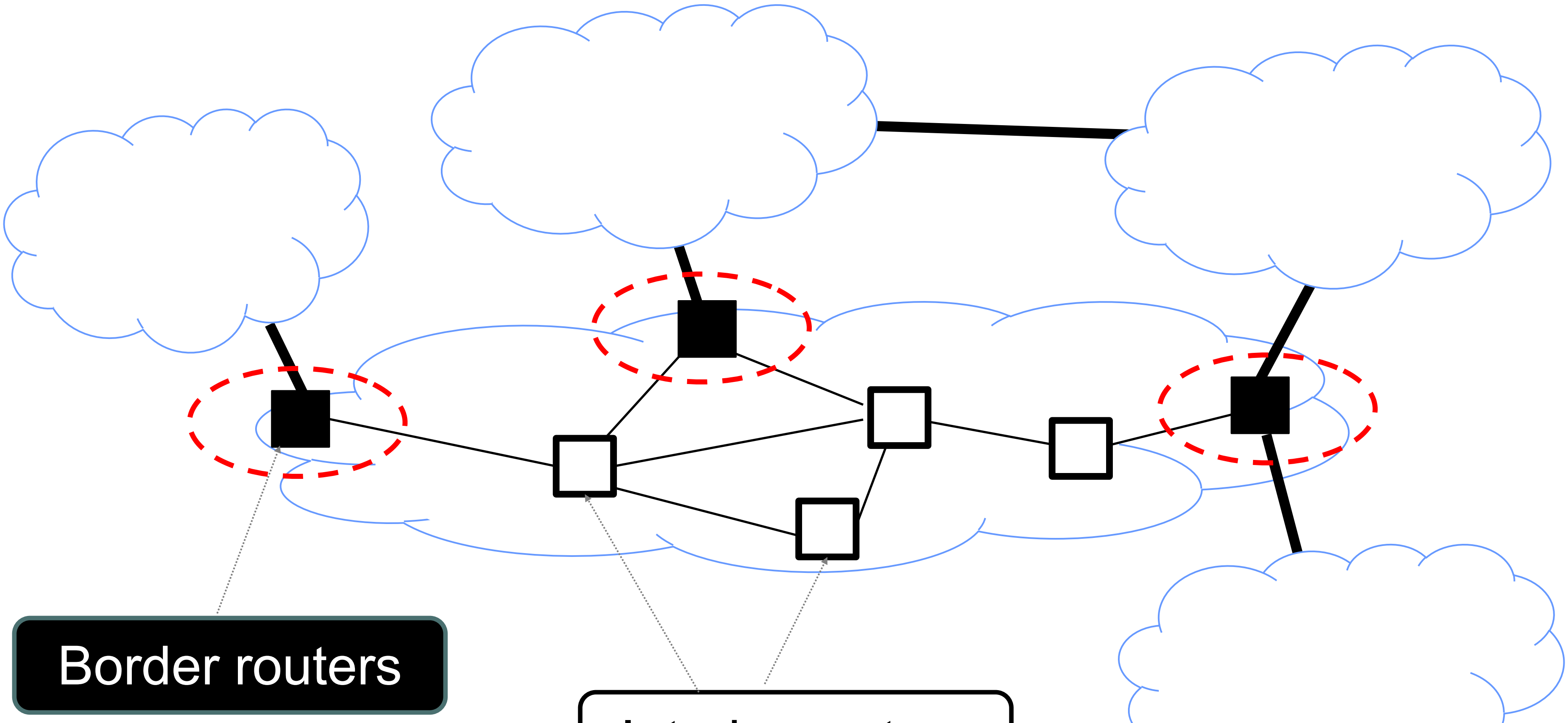
Many design questions....

- 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?



Who “speaks” BGP?



Border routers at an Autonomous System

What does “speak BGP” mean?

What does “speak BGP” mean?

- Advertise routes as specified by the BGP protocol standard
  - read more here: https://datatracker.ietf.org/doc/html/rfc4271

What does “speak BGP” mean?

- Advertise routes as specified by the BGP protocol standard
  - read more here: https://datatracker.ietf.org/doc/html/rfc4271
- Specifies what messages BGP “speakers” exchange
  - message types and syntax

What does “speak BGP” mean?

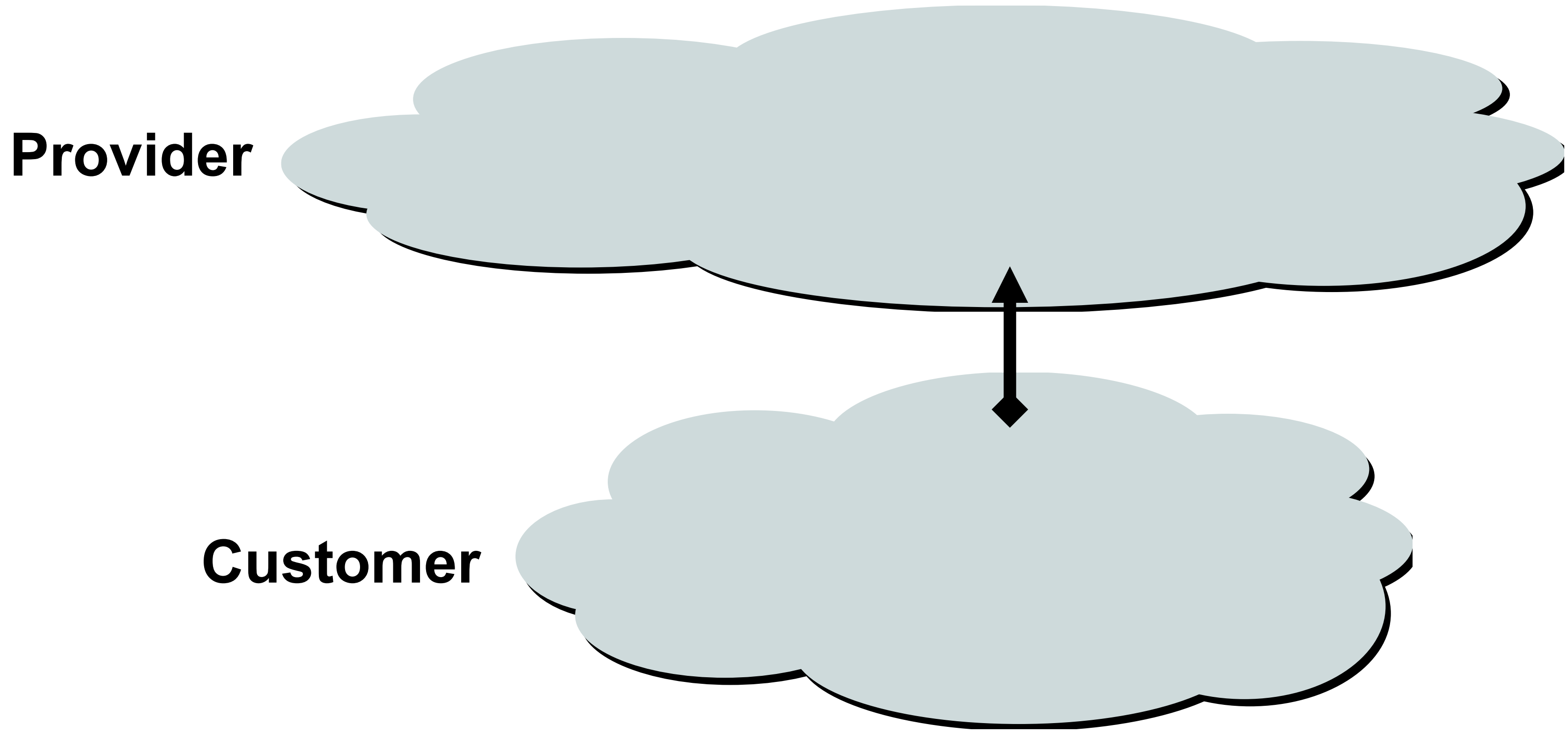
- Advertise routes as specified by the BGP protocol standard
  - read more here: https://datatracker.ietf.org/doc/html/rfc4271
- Specifies what messages BGP “speakers” exchange
  - message types and syntax
- And how to process these messages
  - e.g., *“when you receive a BGP update, do…. “*

Note: 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

Note: Some Border Routers Don't Need BGP

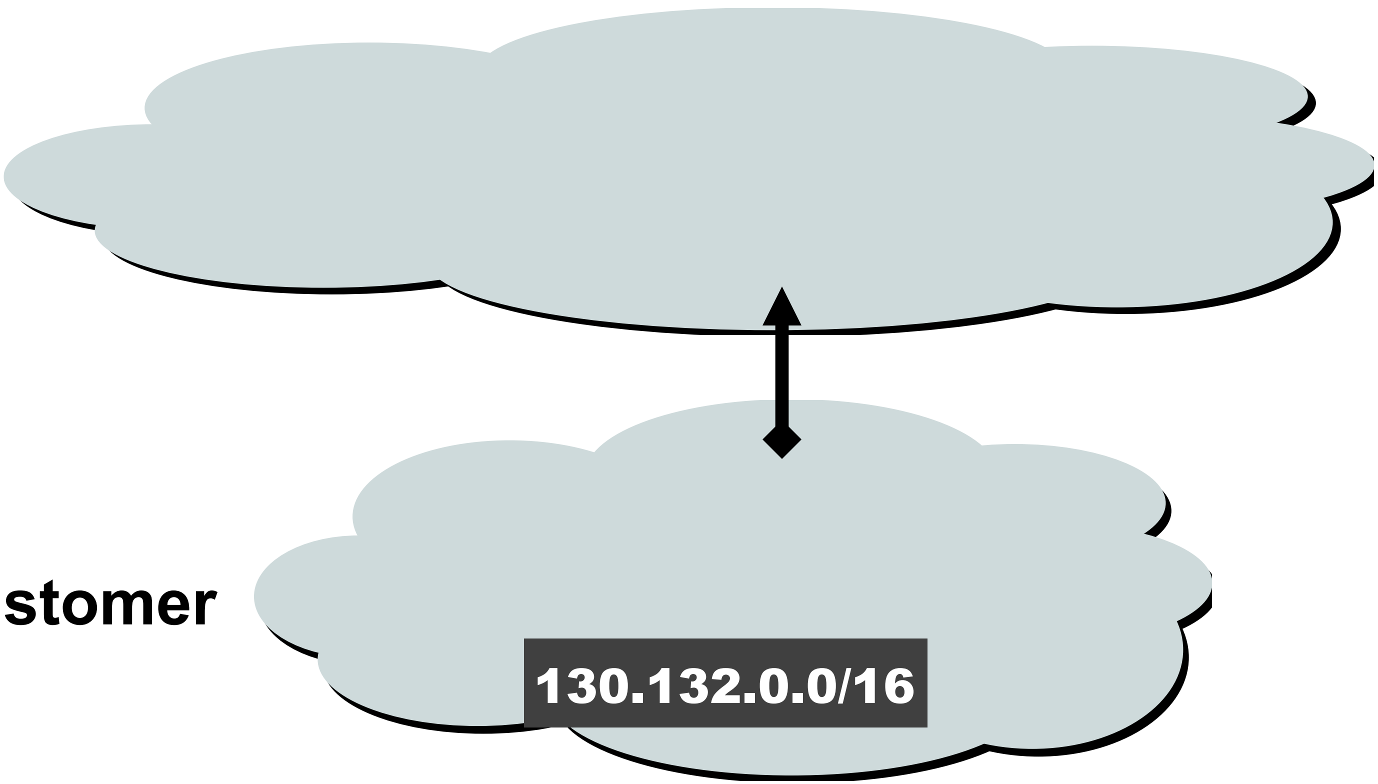
- Customer that connects to a single provider AS
  - Provider can advertise prefixes into BGP on behalf of customer
  - … and the customer can simply default-route to the AS



Note: 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

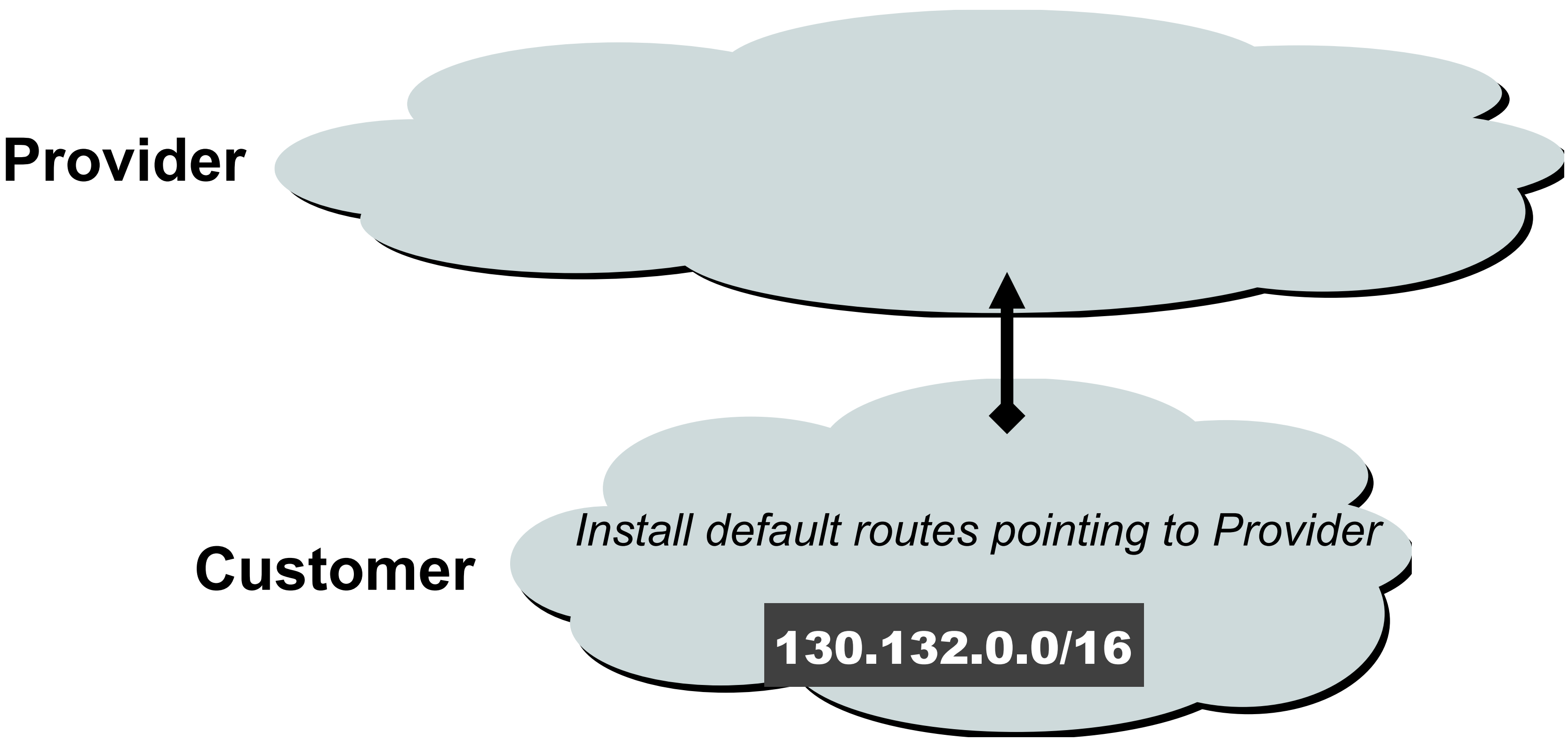
Provider



Customer

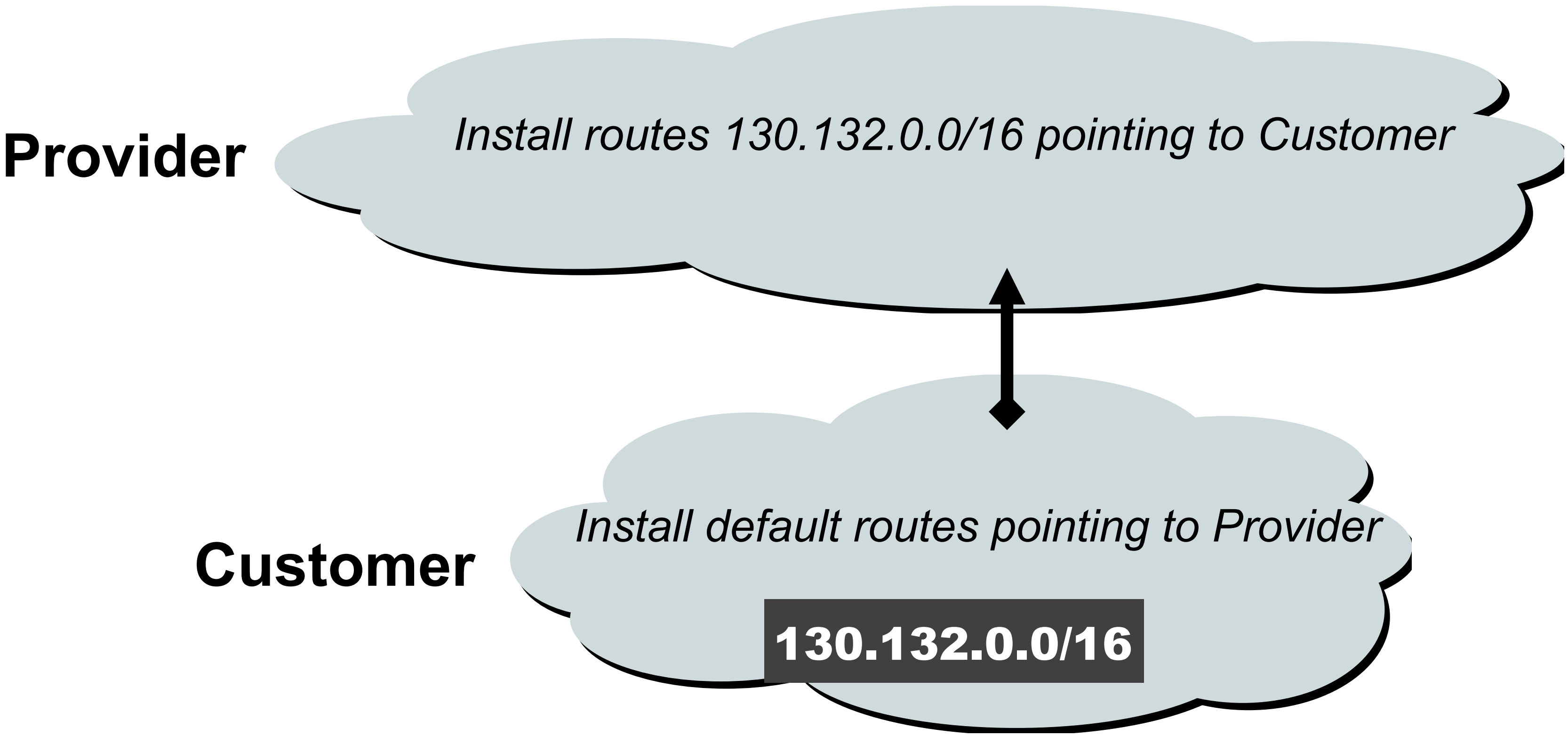
Note: Some Border Routers Don’t Need BGP

- 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

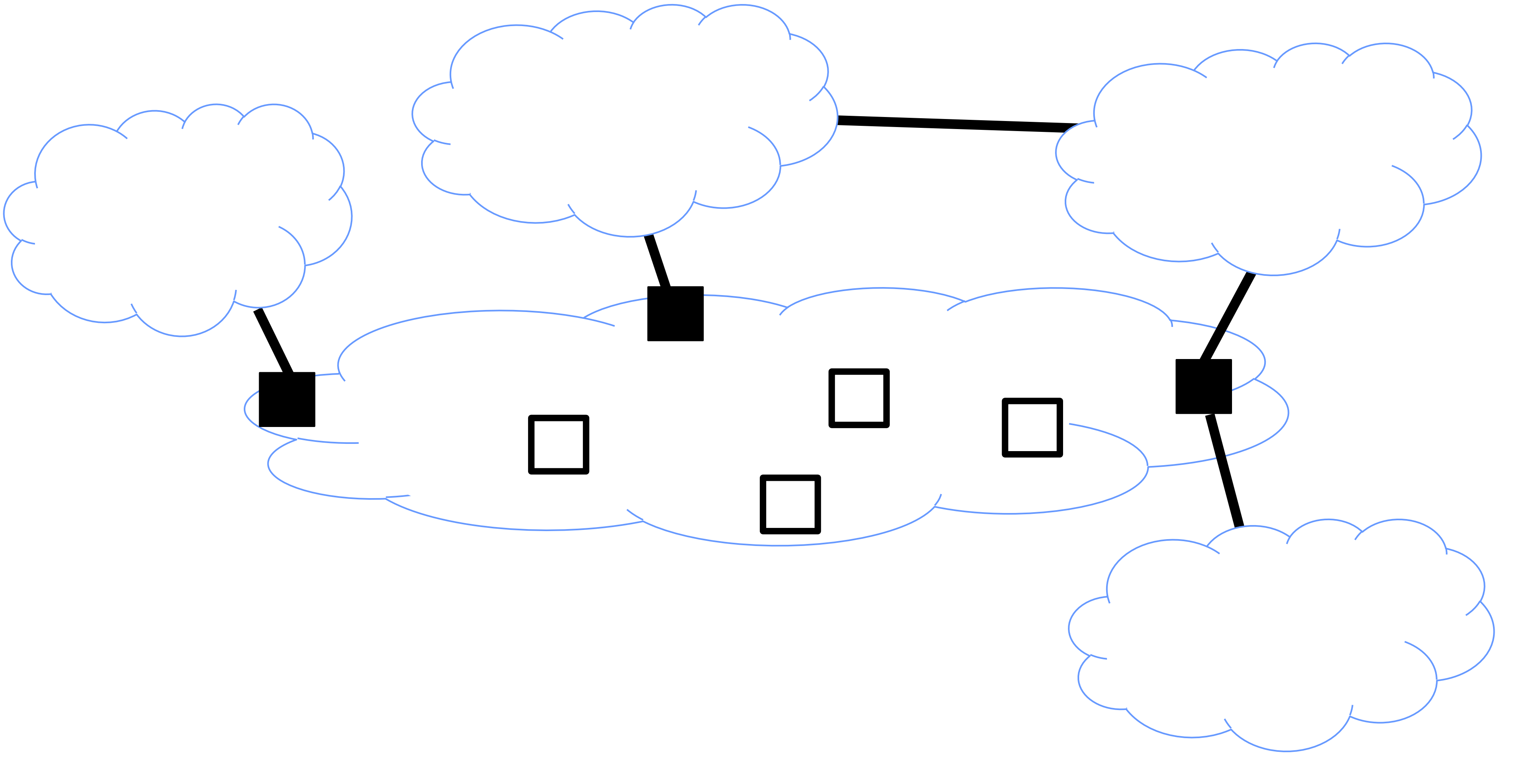


Note: Some Border Routers Don't Need BGP

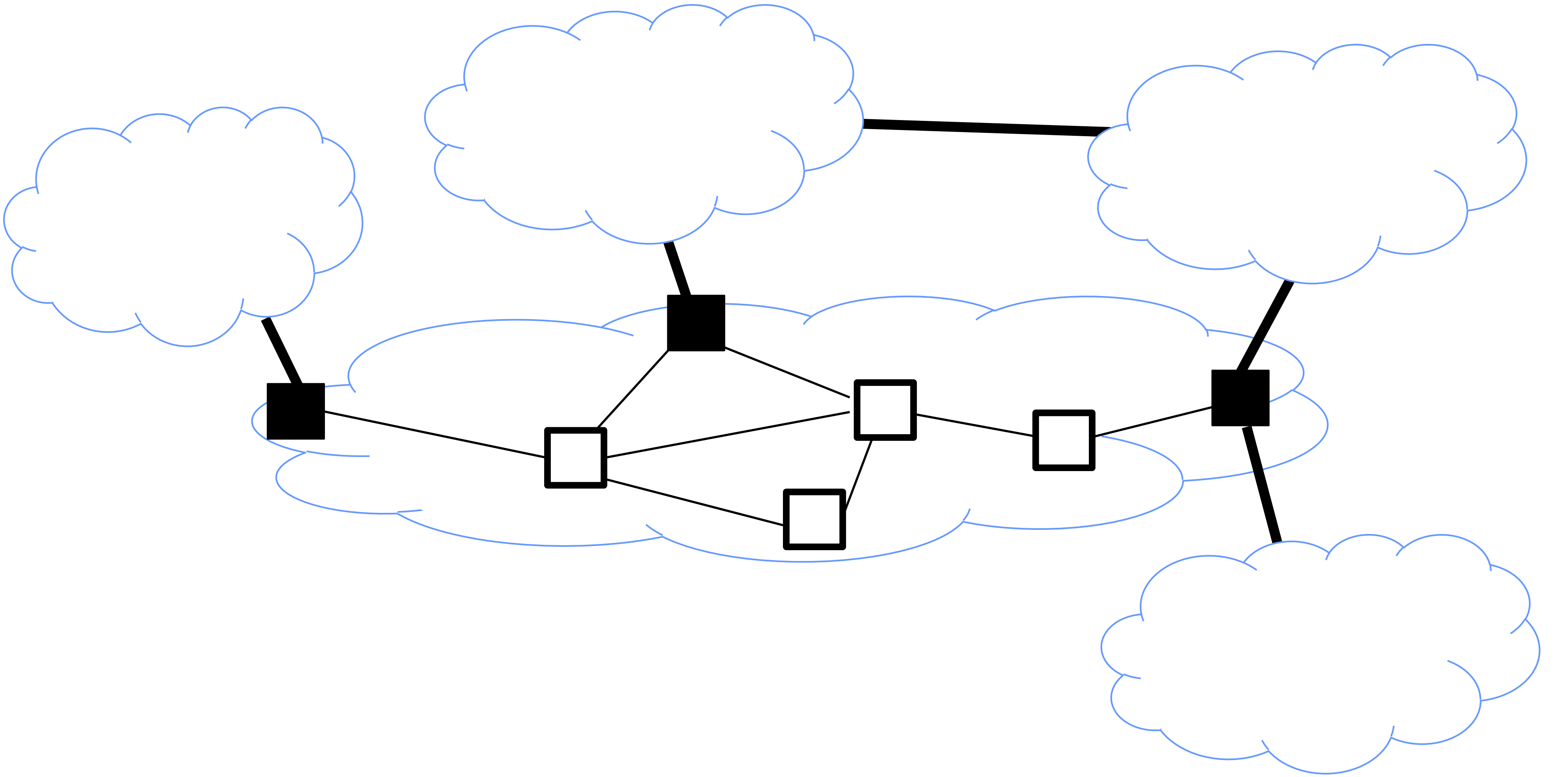
- Customer that connects to a single provider AS
  - Provider can advertise prefixes into BGP on behalf of customer
  - … and the customer can simply default-route to the AS



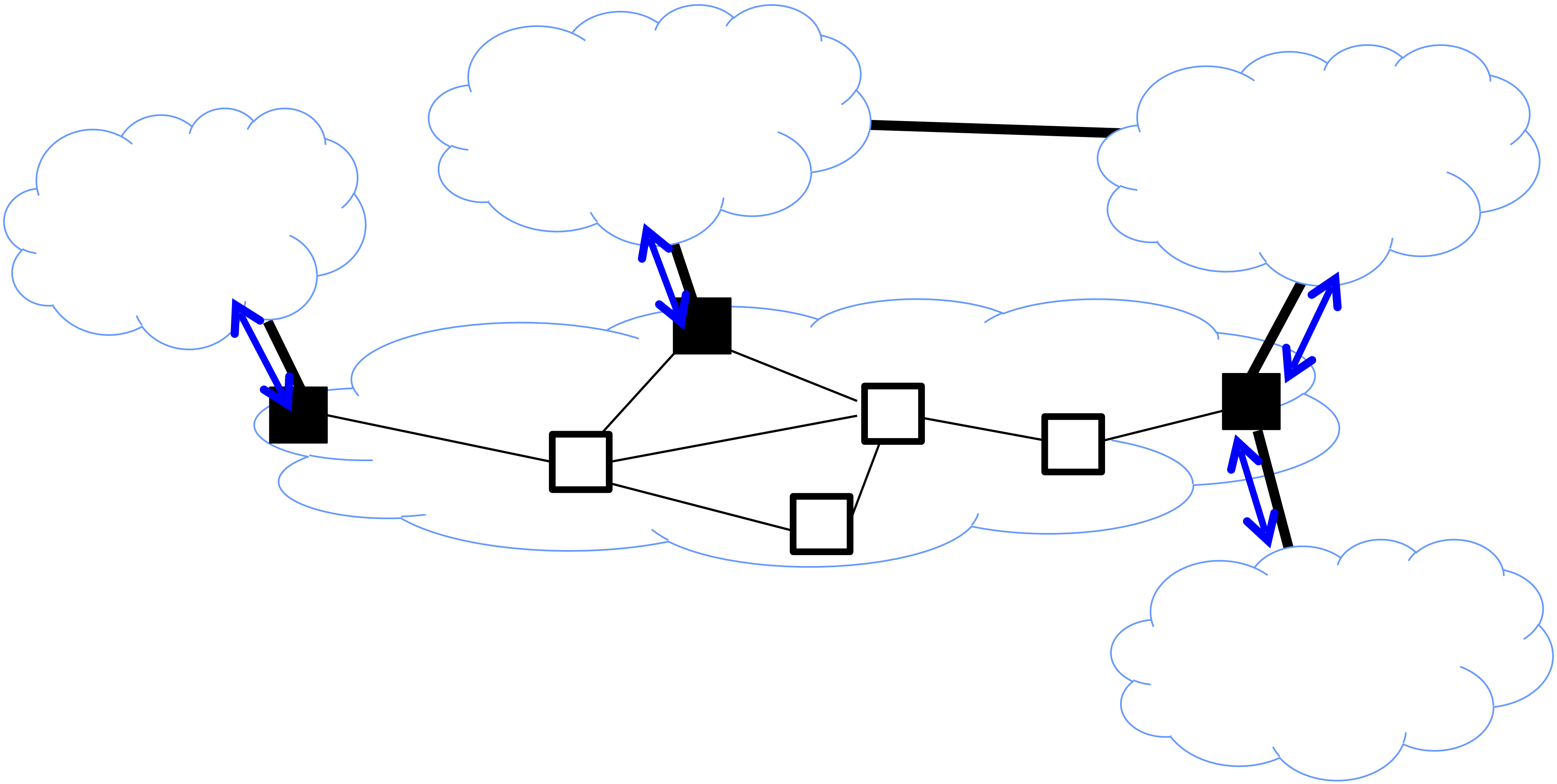
BGP “sessions”



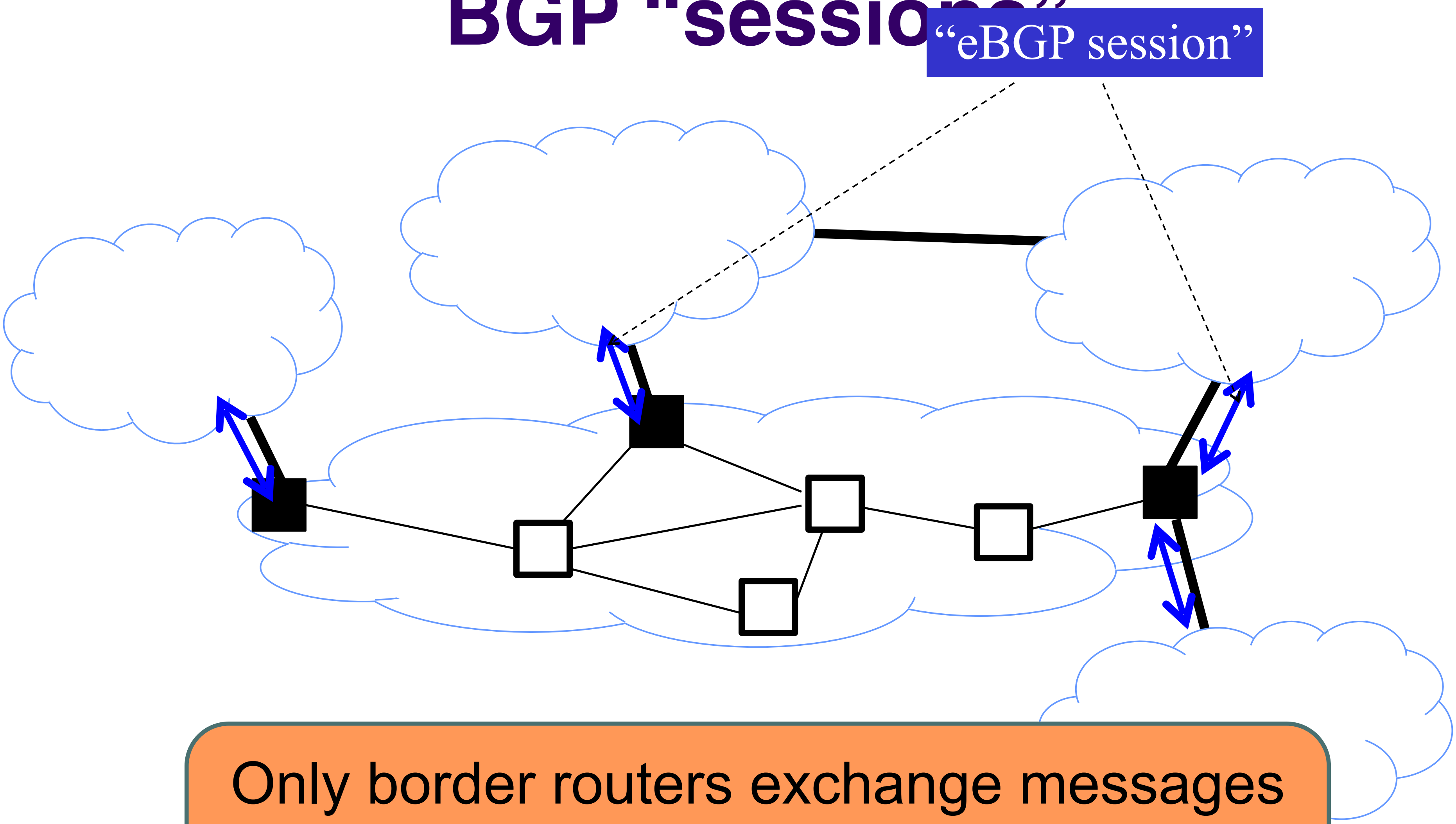
BGP “sessions”



BGP “sessions”

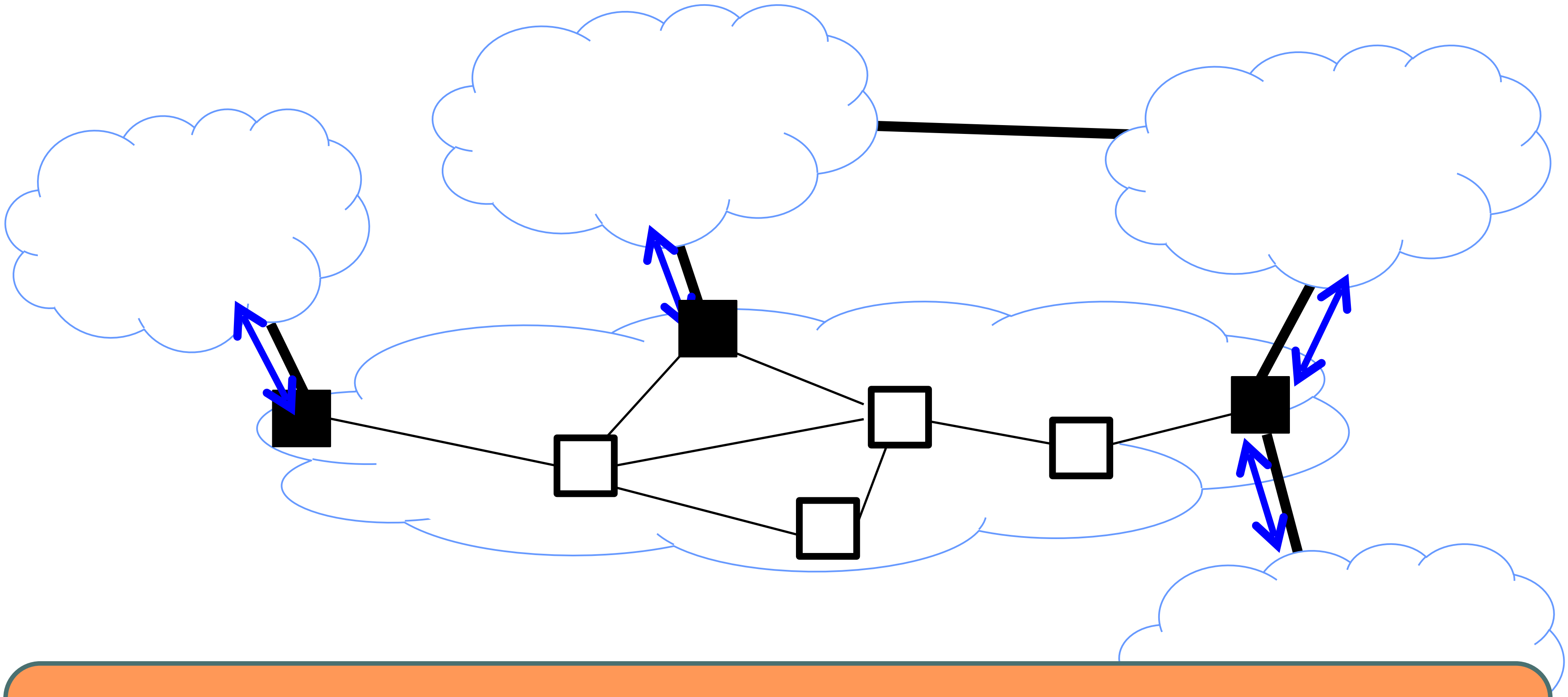


BGP "sessions"



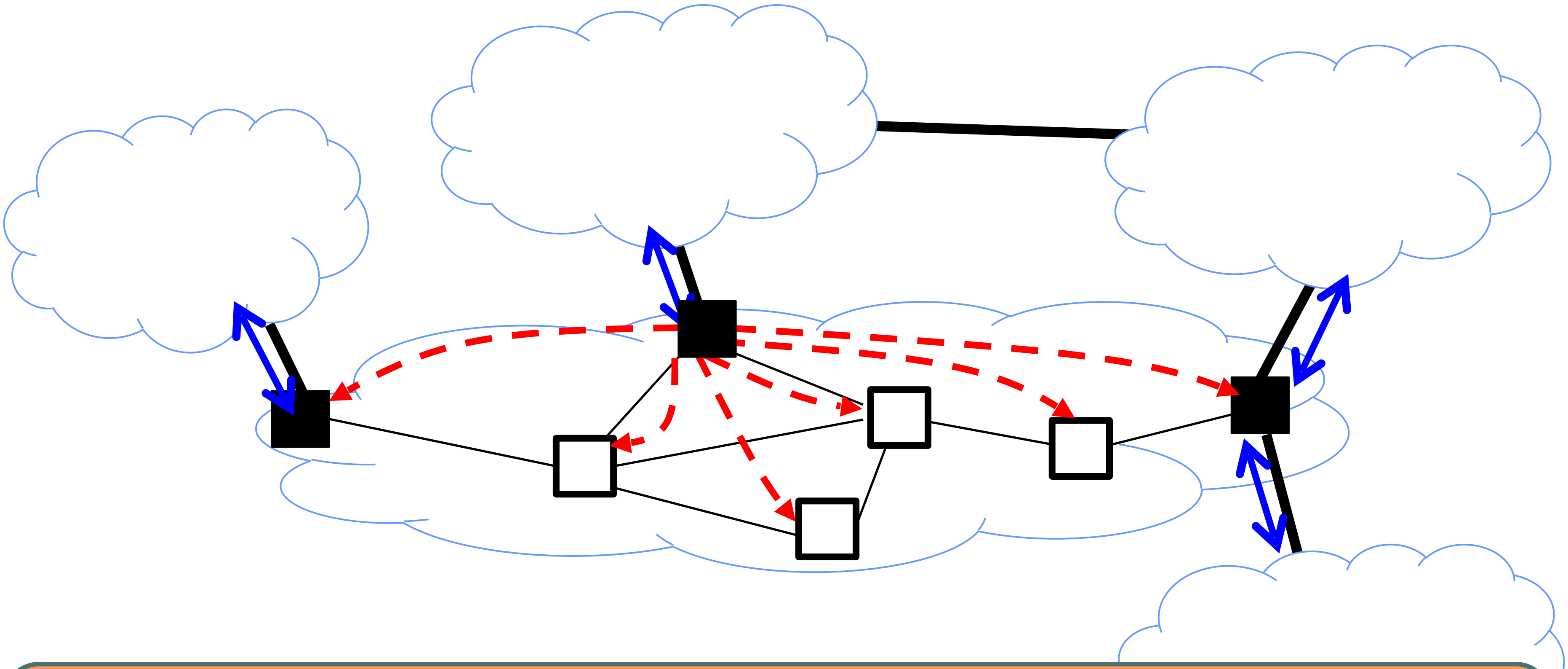
Only border routers exchange messages with routers in external domains (hence, *external BGP* or "eBGP")

BGP “sessions”



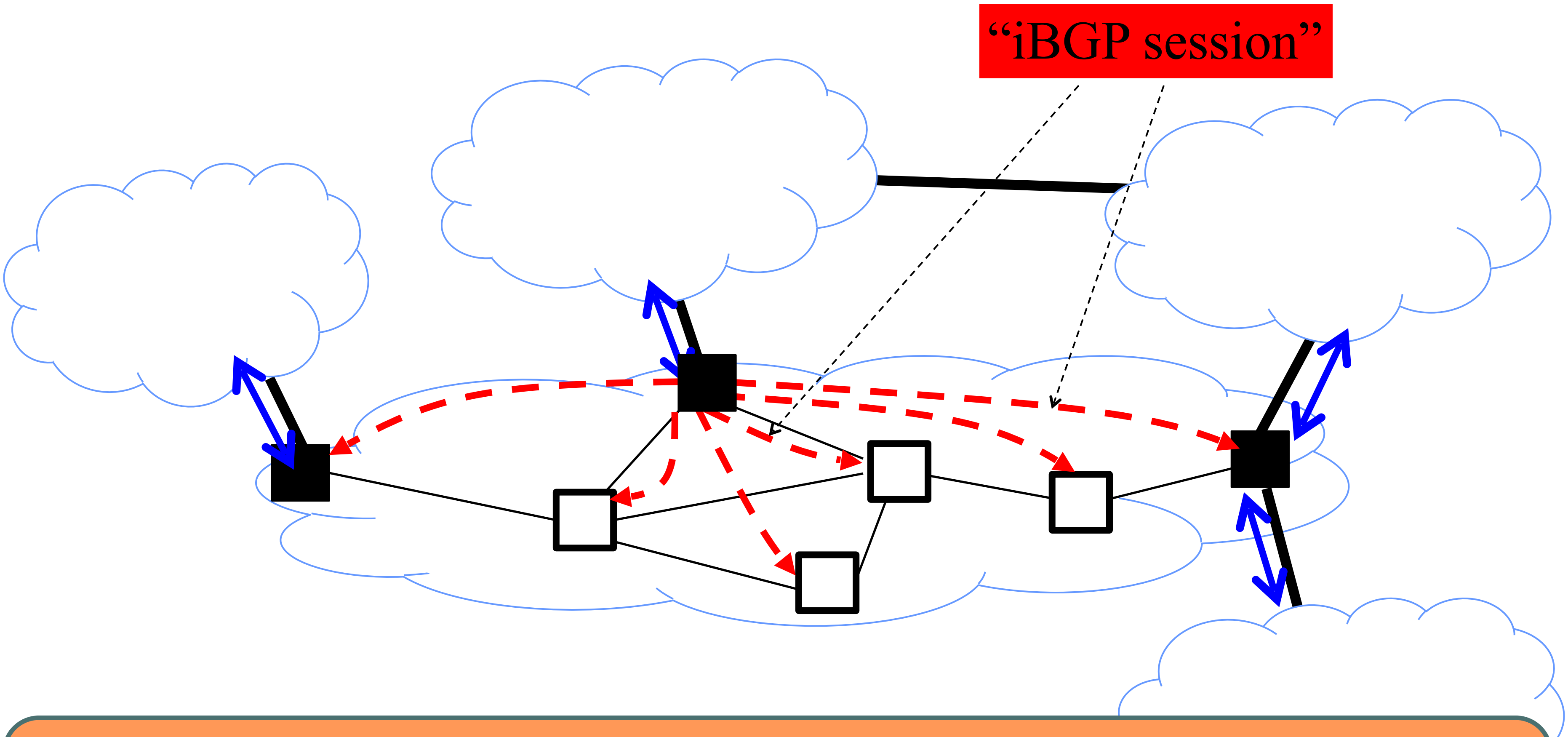
Border router speaks BGP with routers in its own AS
(hence, *internal* BGP, or “iBGP”)

BGP “sessions”



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

BGP "sessions"



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

eBGP, iBGP, IGP

eBGP, iBGP, IGP

- eBGP: BGP sessions between border routers in <u>different</u> ASes
  - exchange routes to different destination prefixes

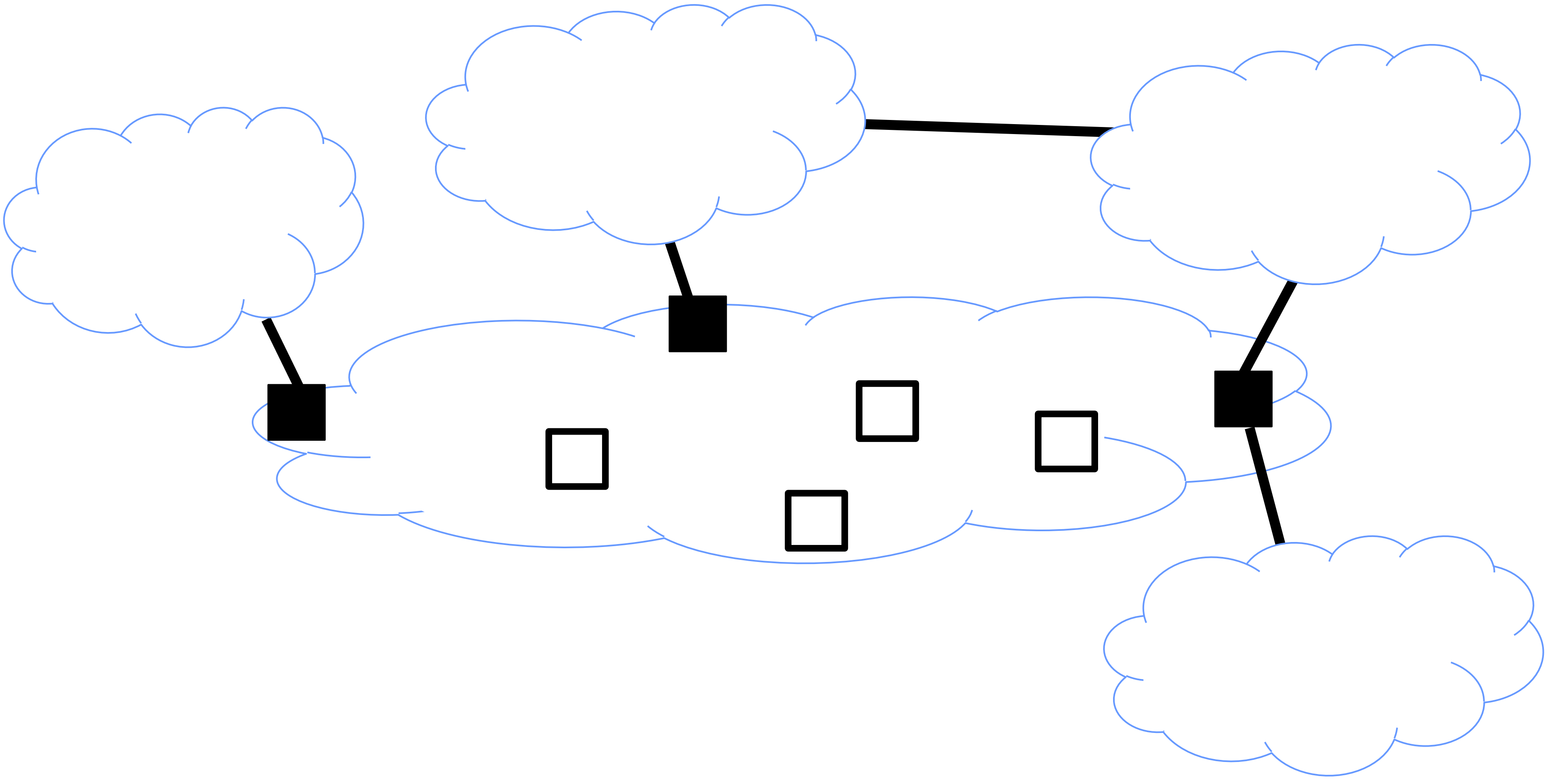
eBGP, iBGP, IGP

- eBGP: BGP sessions between border routers in <u>different</u> ASes
  - exchange routes to different destination prefixes
- iBGP: BGP sessions between border routers and other routers within the <u>same</u> AS
  - distribute externally learned routes internally

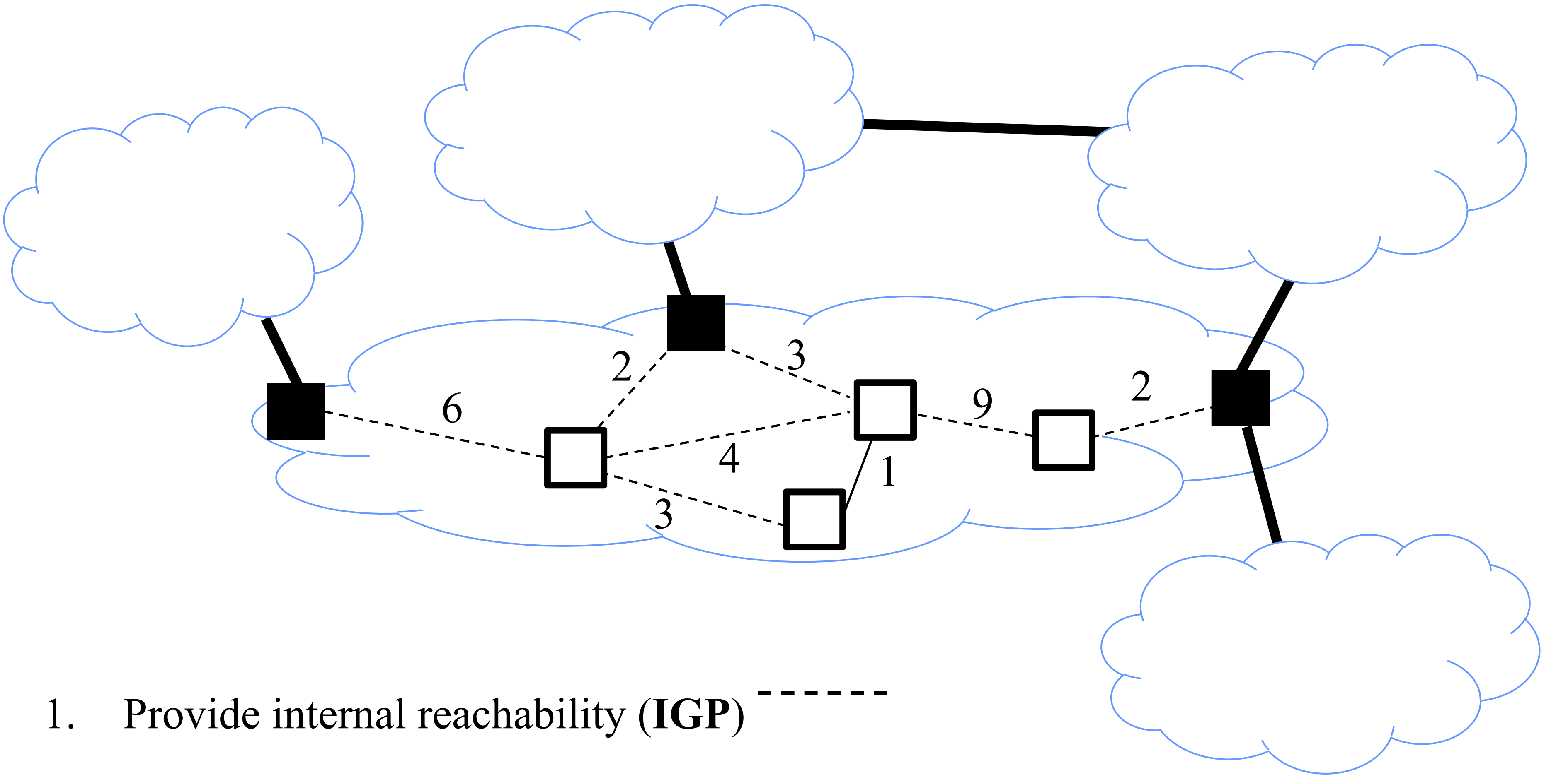
eBGP, iBGP, IGP

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

Putting the pieces together

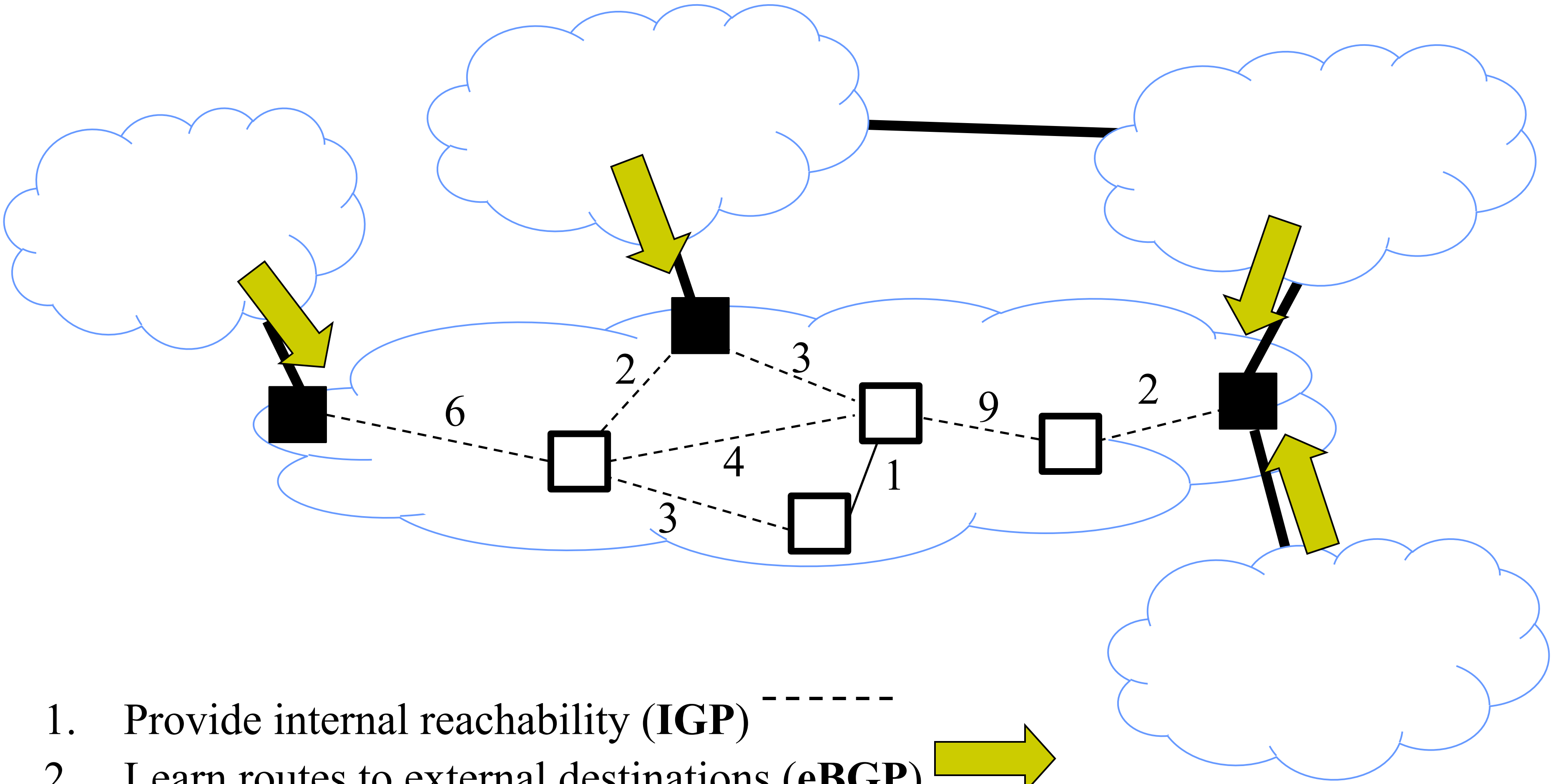


Putting the pieces together



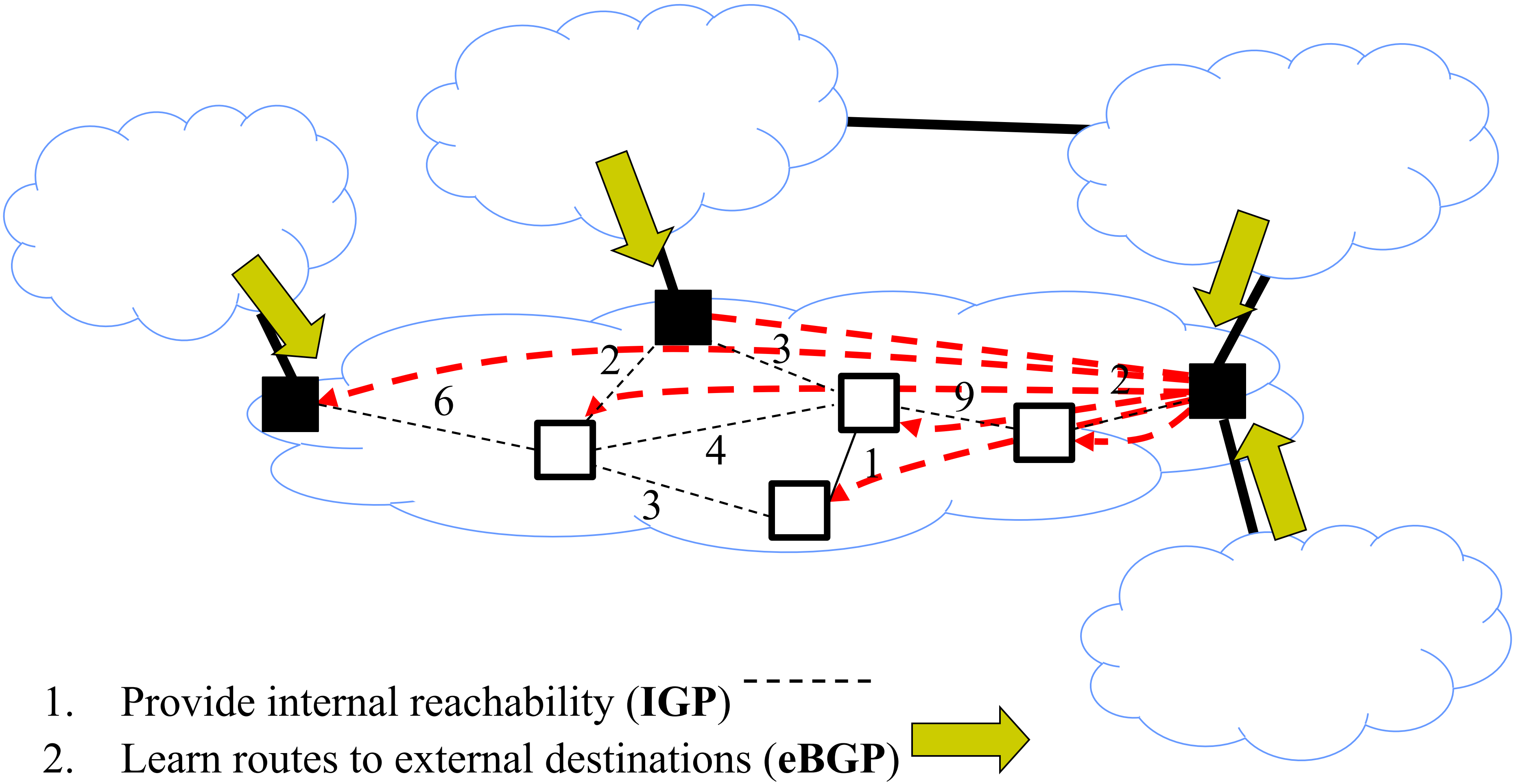
1. Provide internal reachability (**IGP**) - - - - - - -

Putting the pieces together



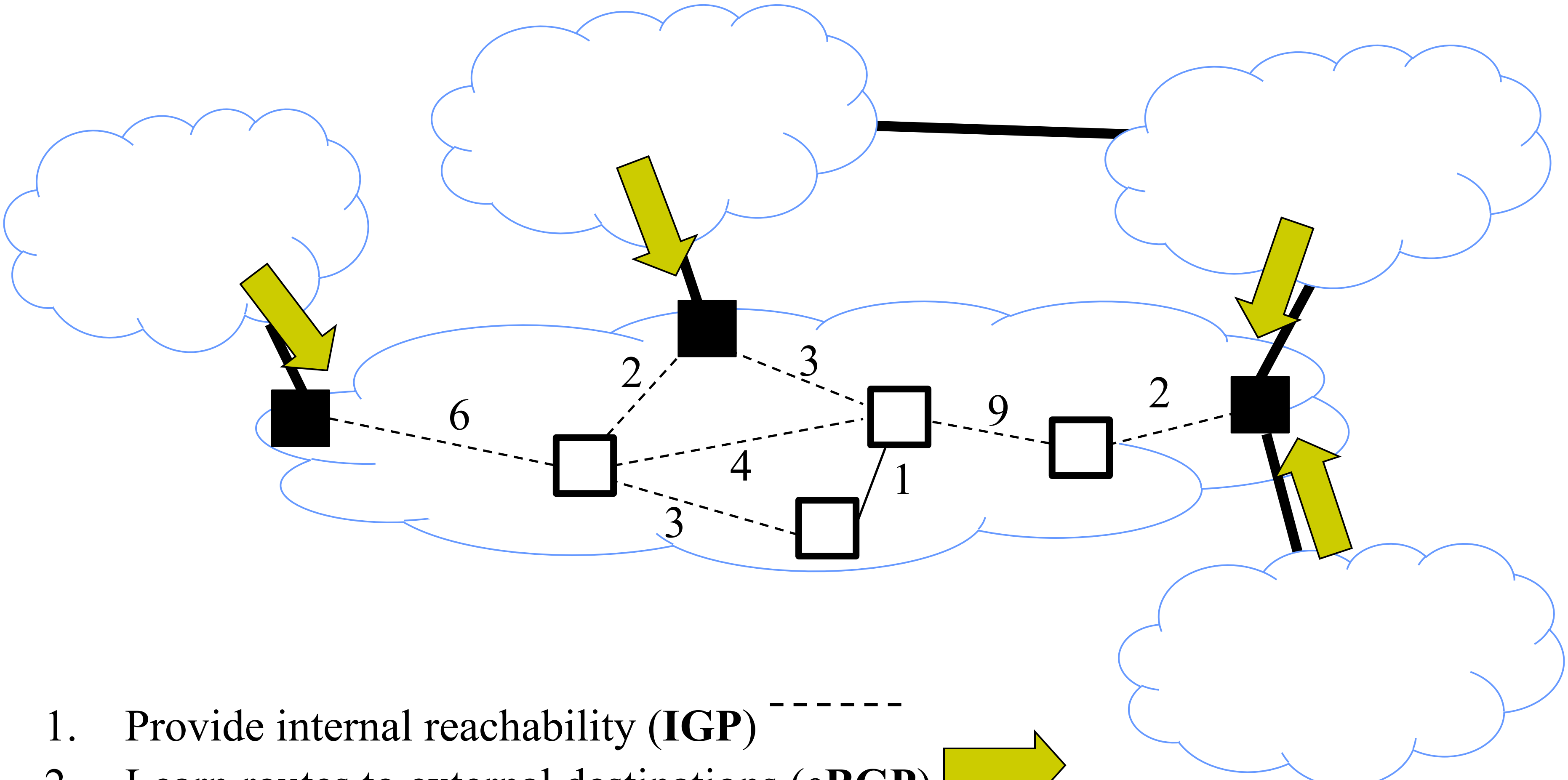
1. Provide internal reachability (**IGP**)
2. Learn routes to external destinations (**eBGP**)

Putting the pieces together



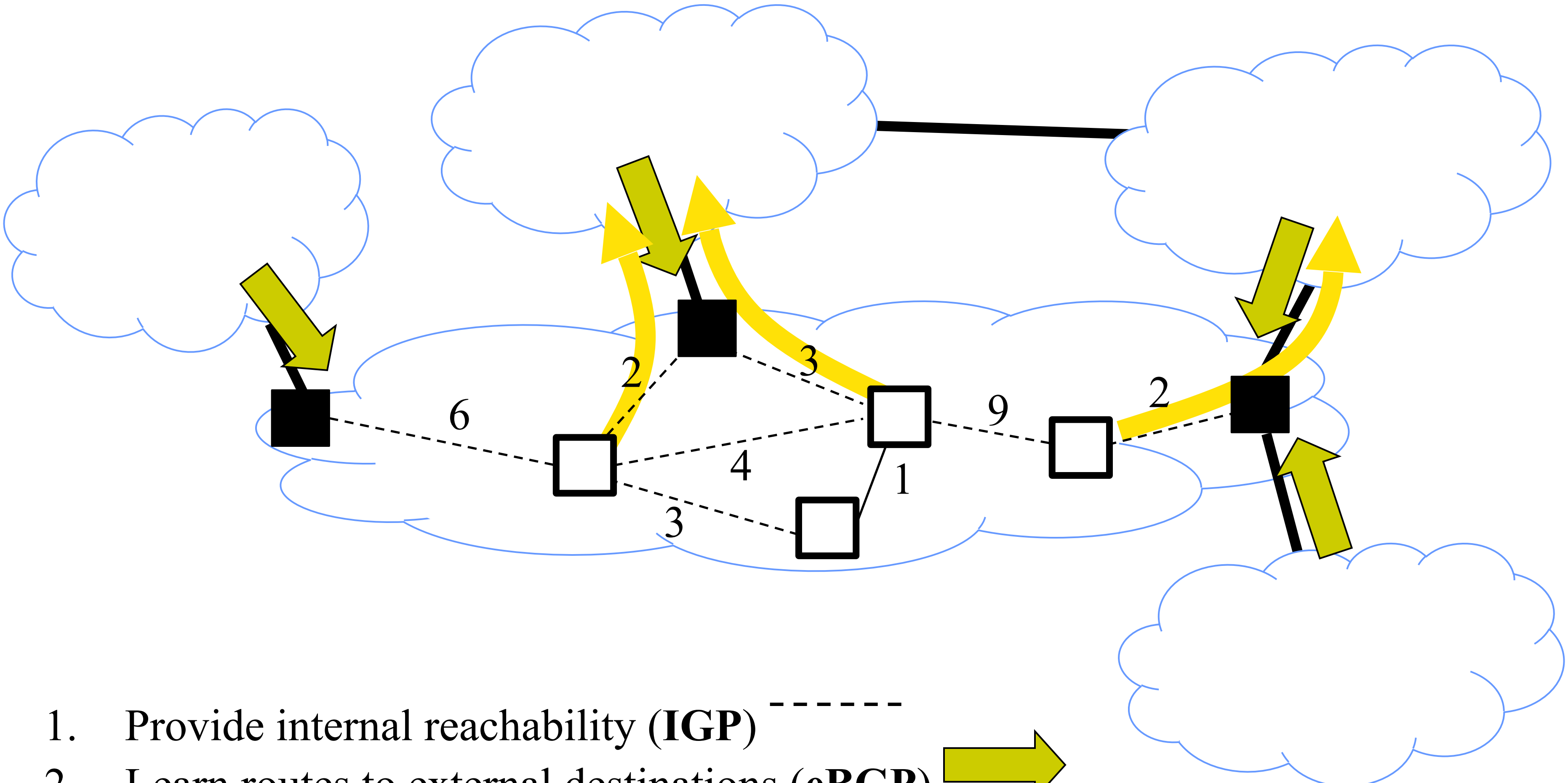
1. Provide internal reachability (**IGP**)
2. Learn routes to external destinations (**eBGP**)
3. Distribute externally learned routes internally (**iBGP**)

Putting the pieces together



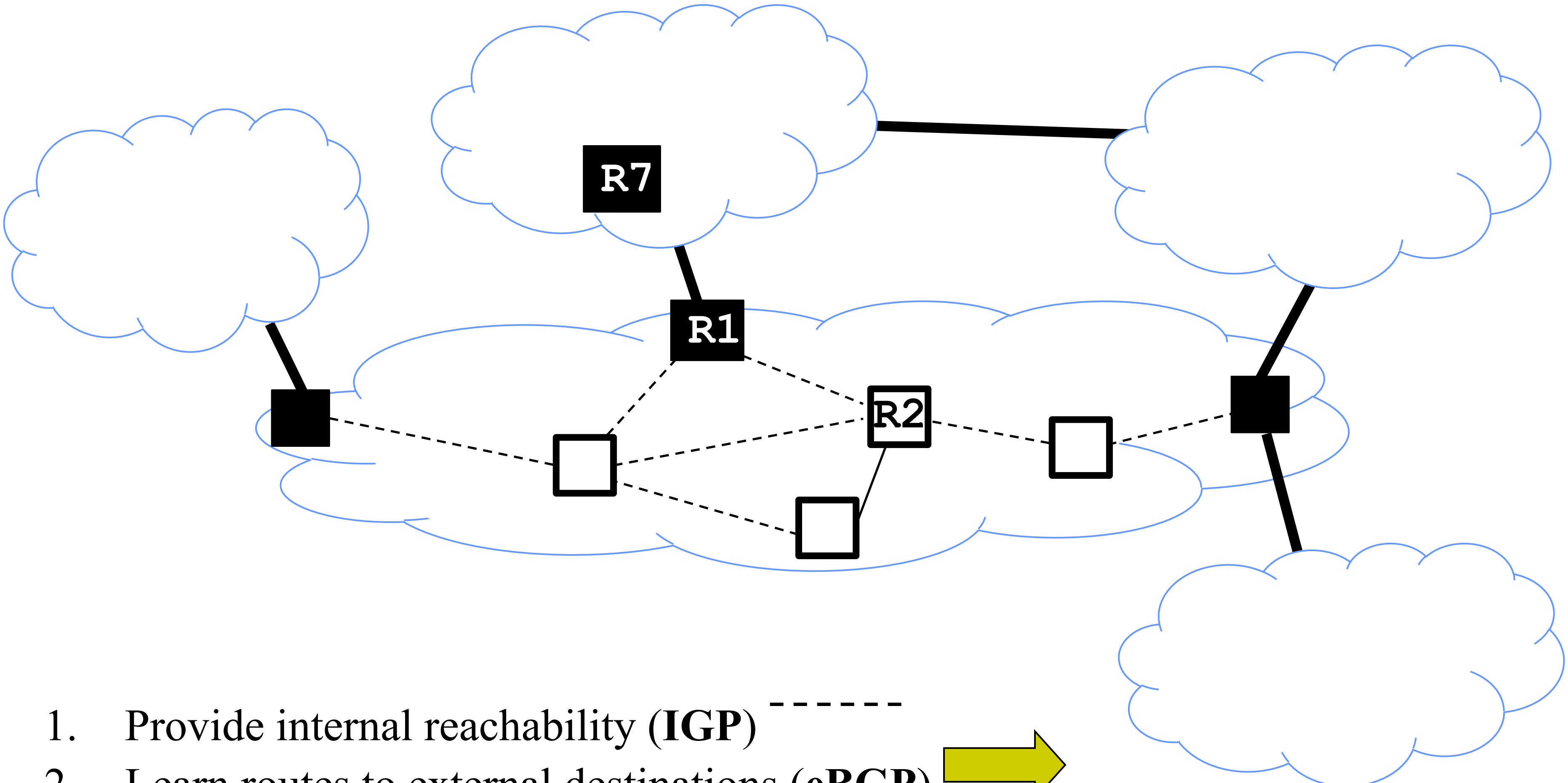
1. Provide internal reachability (**IGP**)
2. Learn routes to external destinations (**eBGP**)
3. Distribute externally learned routes internally (**iBGP**)

Putting the pieces together



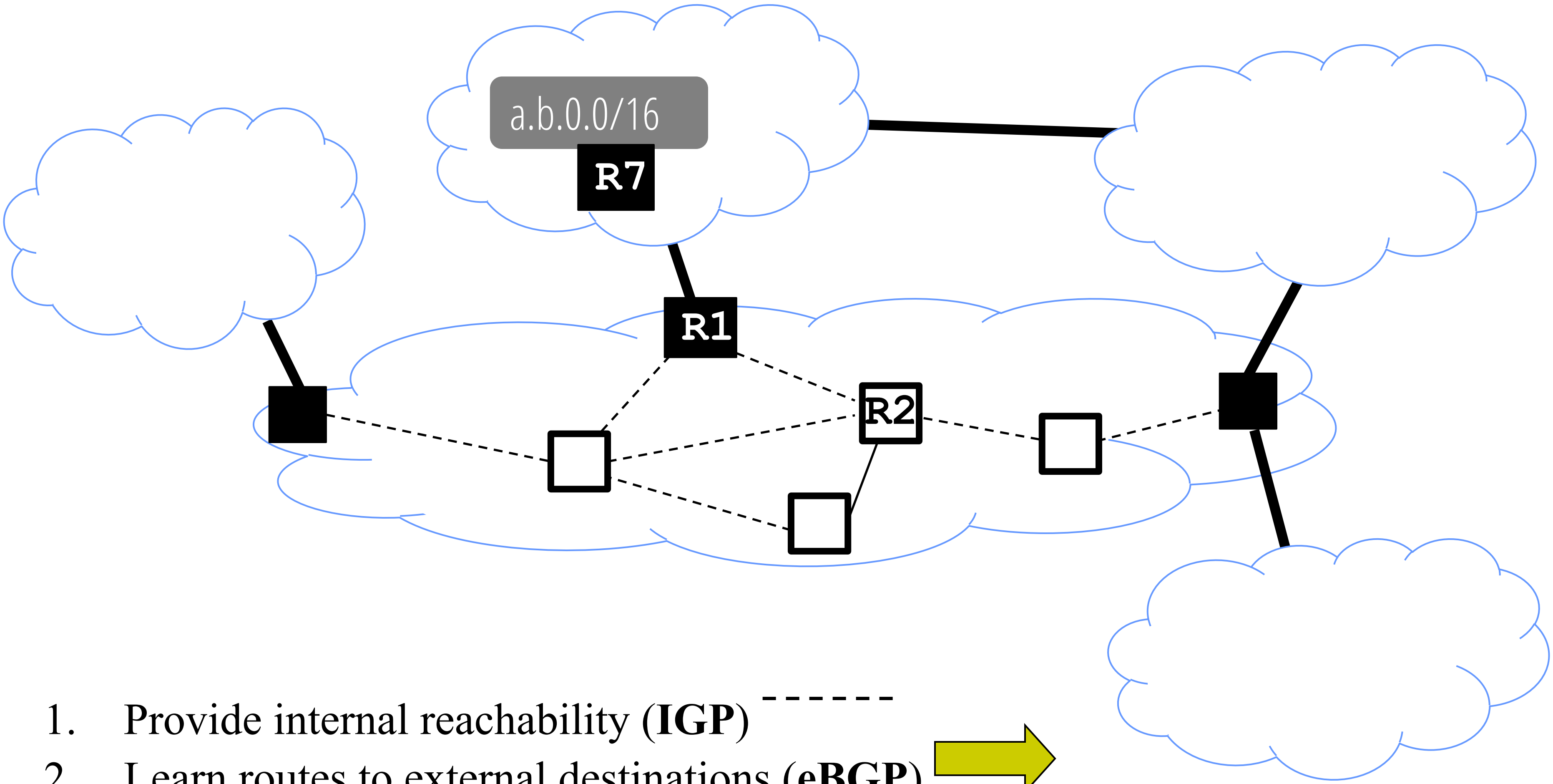
1. Provide internal reachability (**IGP**)
2. Learn routes to external destinations (**eBGP**)
3. Distribute externally learned routes internally (**iBGP**)
4. Travel shortest path to egress (IGP)

Putting the pieces together



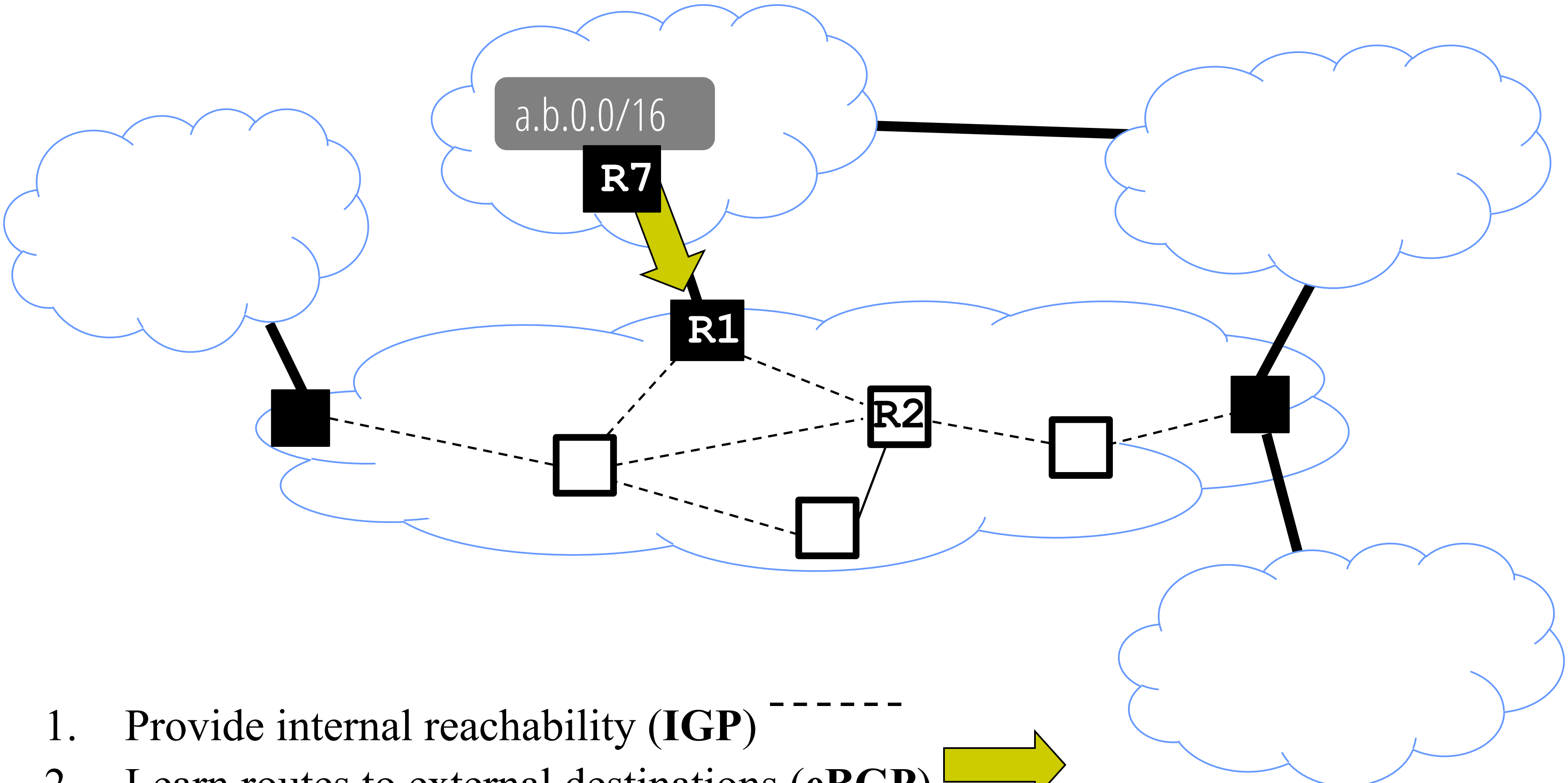
1. Provide internal reachability (**IGP**)
2. Learn routes to external destinations (**eBGP**)
3. Distribute externally learned routes internally (**iBGP**)
4. Travel shortest path to egress (IGP)

Putting the pieces together



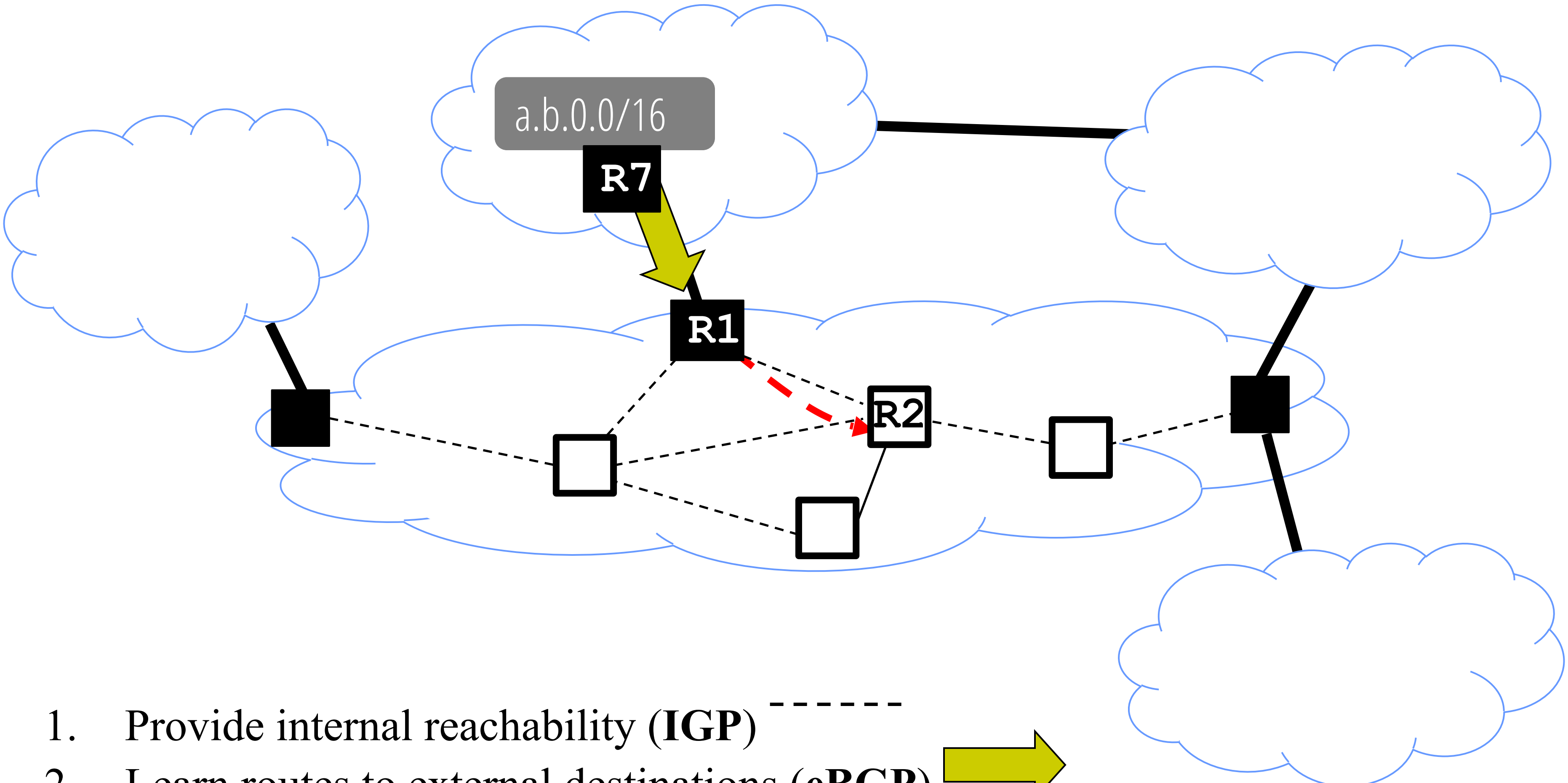
1. Provide internal reachability (**IGP**)
2. Learn routes to external destinations (**eBGP**)
3. Distribute externally learned routes internally (**iBGP**)
4. Travel shortest path to egress (IGP)

Putting the pieces together



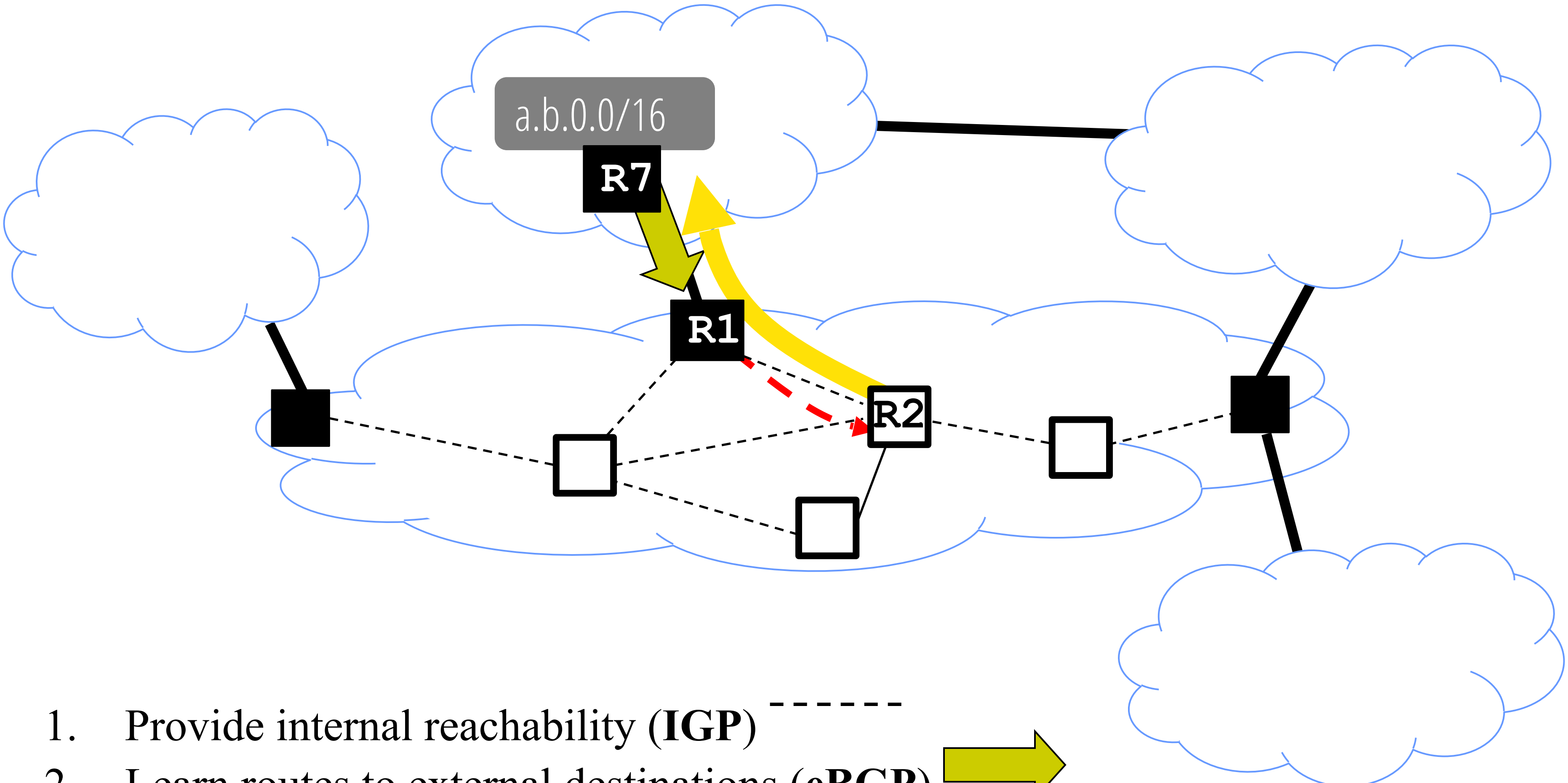
1. Provide internal reachability (**IGP**)
2. Learn routes to external destinations (**eBGP**)
3. Distribute externally learned routes internally (**iBGP**)
4. Travel shortest path to egress (IGP)

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

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

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>

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>

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

Note: In reality, there are a few different ways to integrate inter- and intra-domain routing

Note: In reality, there are a few different ways to integrate inter- and intra-domain routing

- Our option: run iBGP between all routers in domain
  - Requires NxB iBGP connections. Could be a scaling issue.
  - This is what we will assume

Many design questions....

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

- Many different 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

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

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

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

Route Attributes

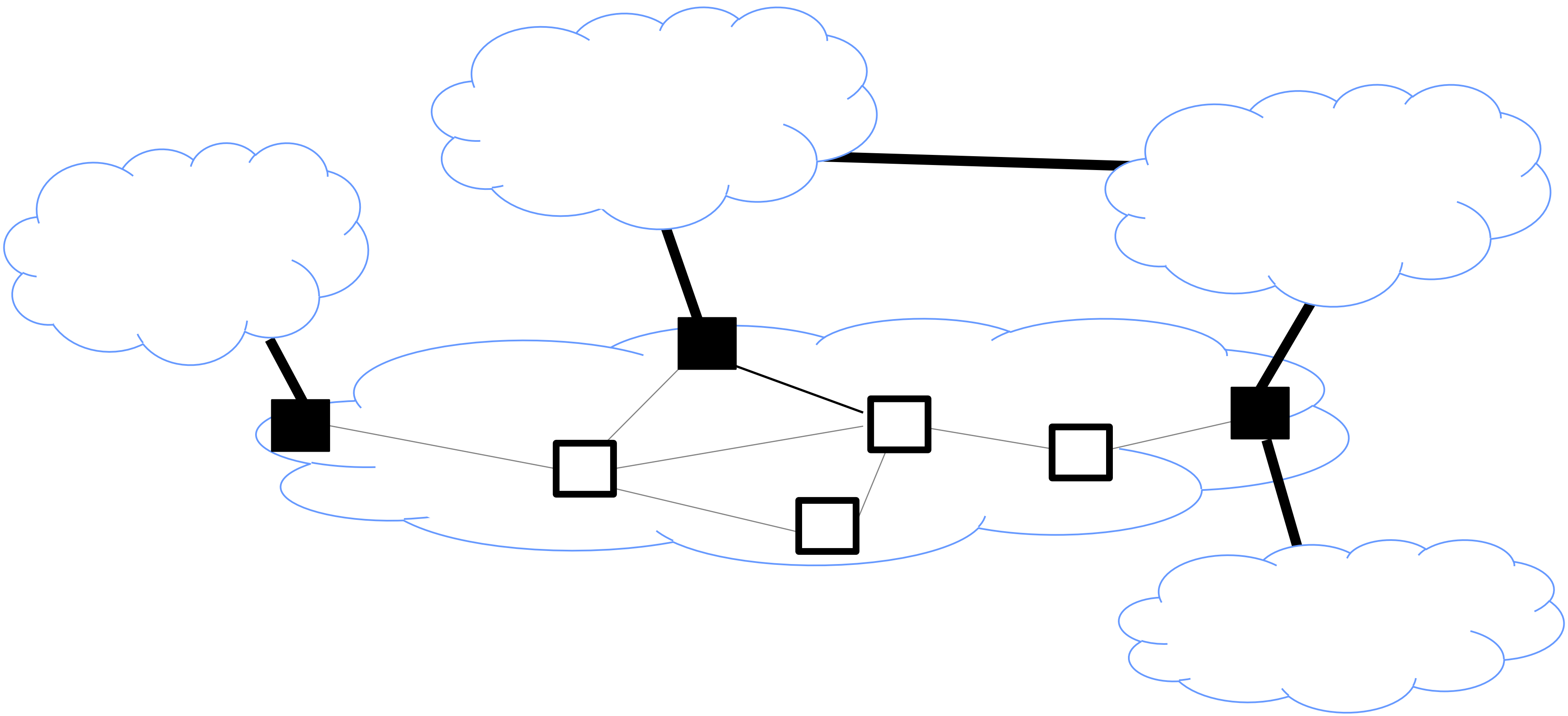
- General mechanism used to express properties about routes
  - Used in route selection/export decisions
- Some attributes are local to an AS
  - Not propagated in eBGP advertisements
- Others are propagated in eBGP route advertisements
- There are many standardized attributes in BGP
  - We will discuss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

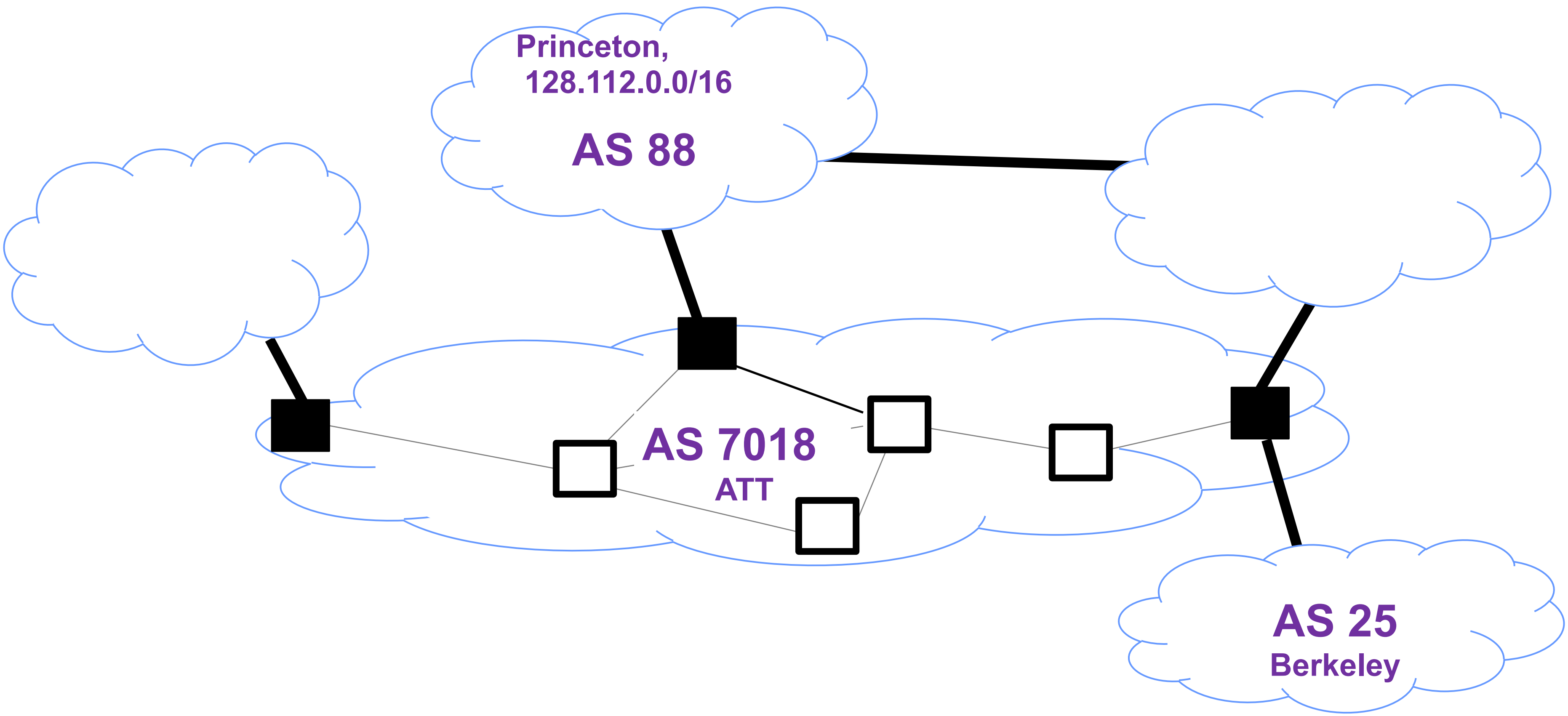
Attributes (1): ASPATH

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



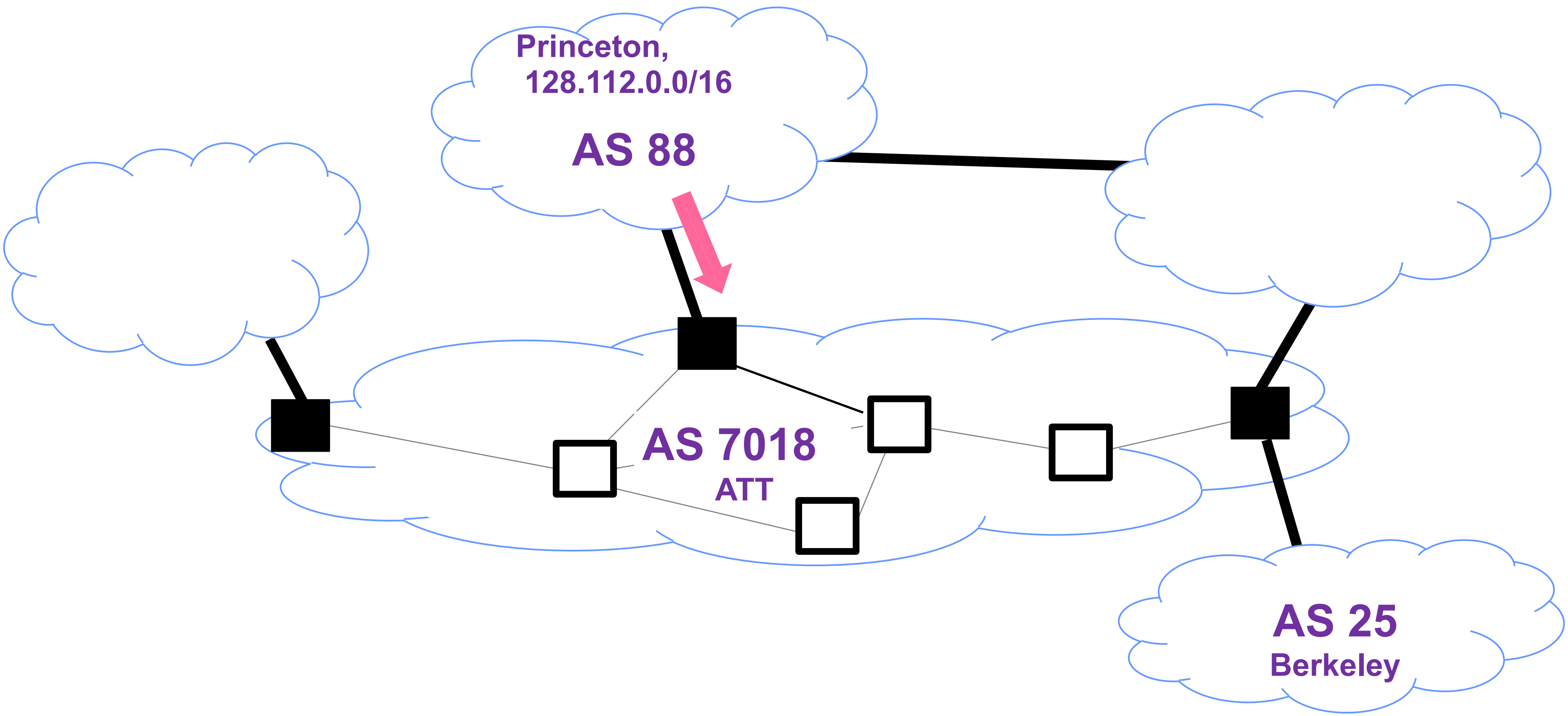
Attributes (1): ASPATH

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



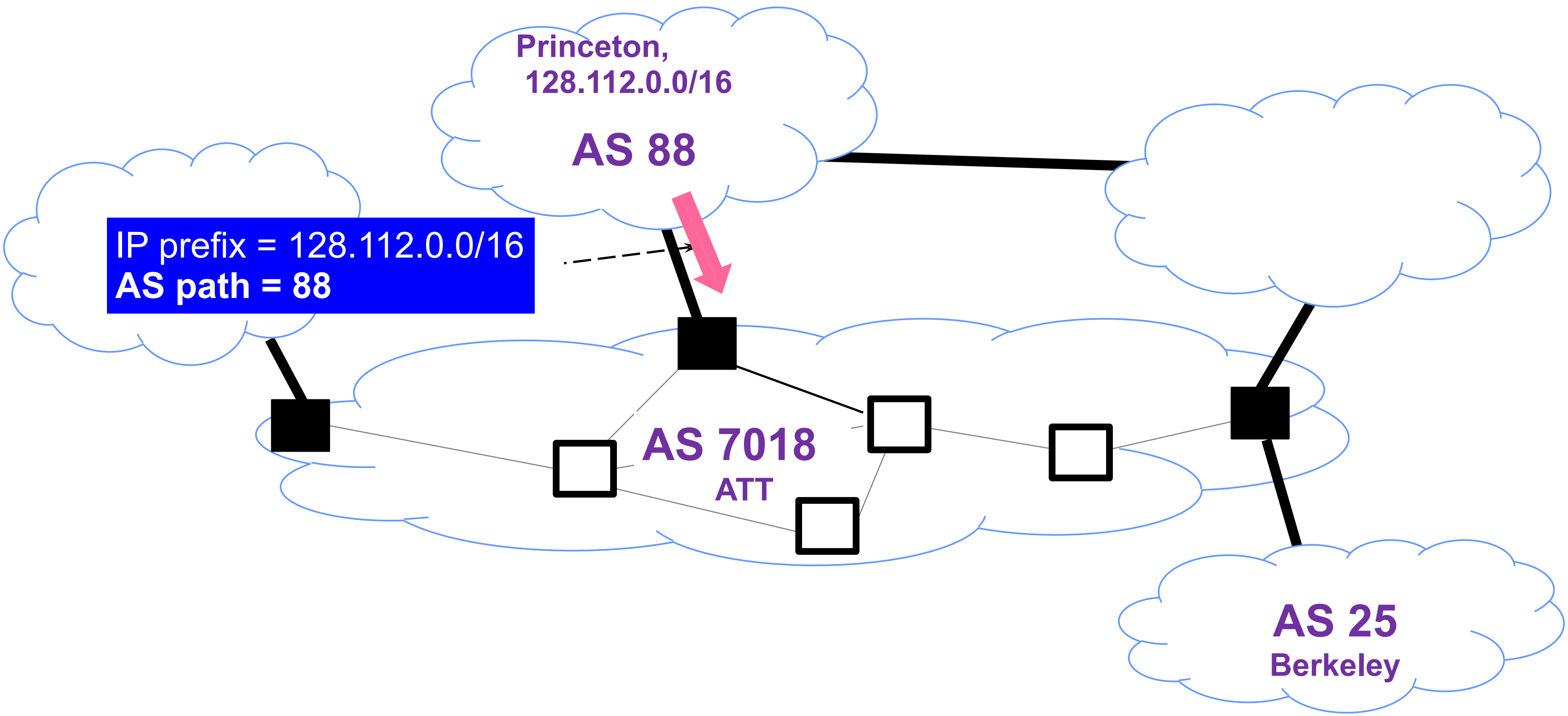
Attributes (1): ASPATH

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



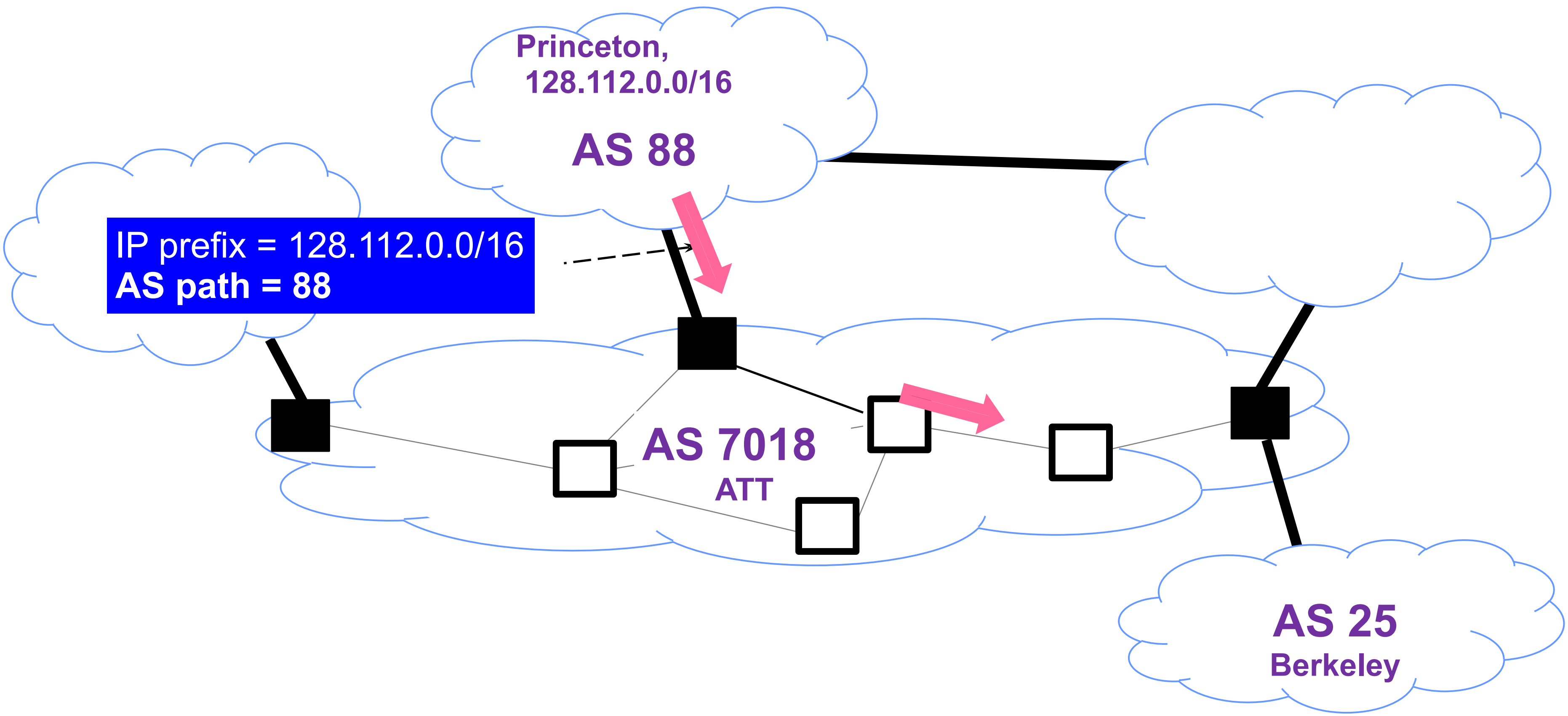
Attributes (1): ASPATH

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



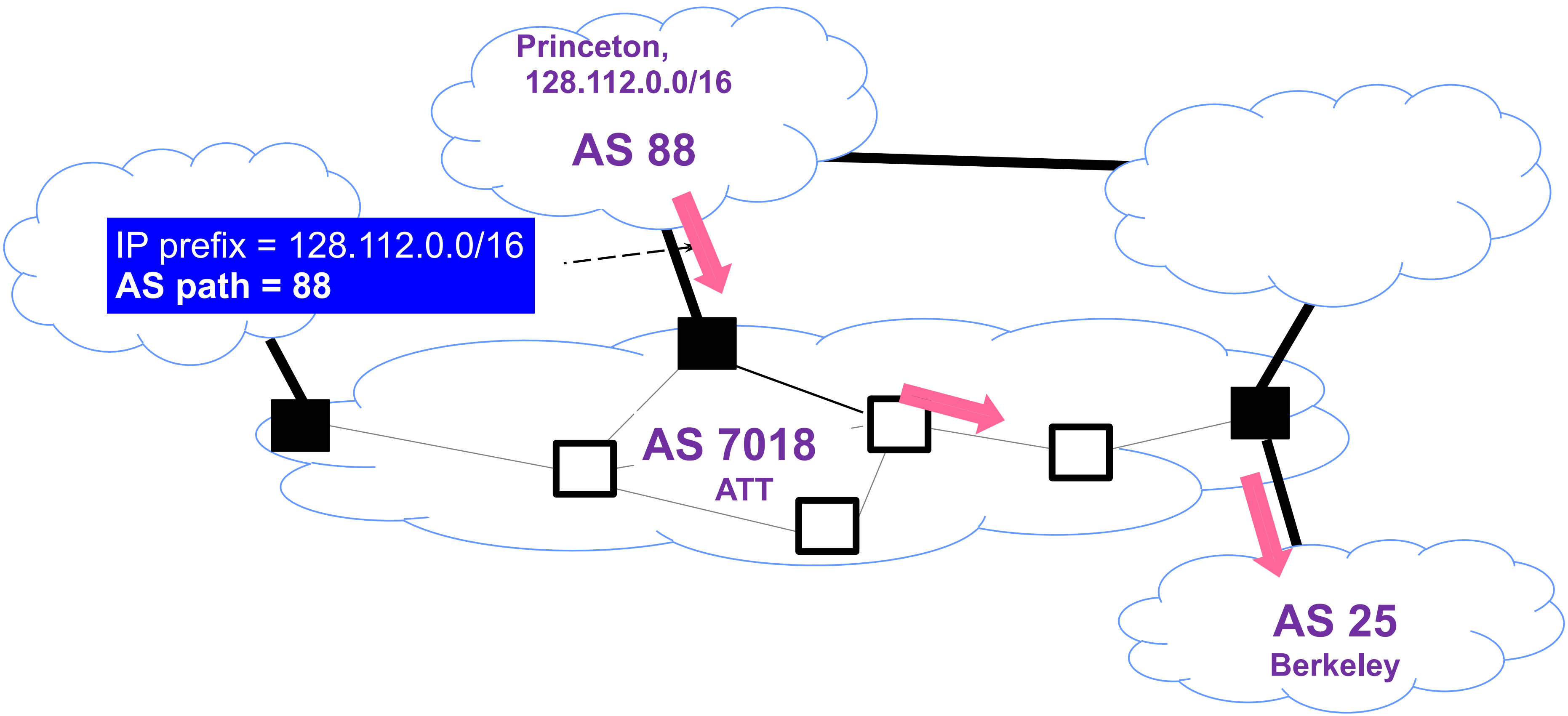
Attributes (1): ASPATH

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



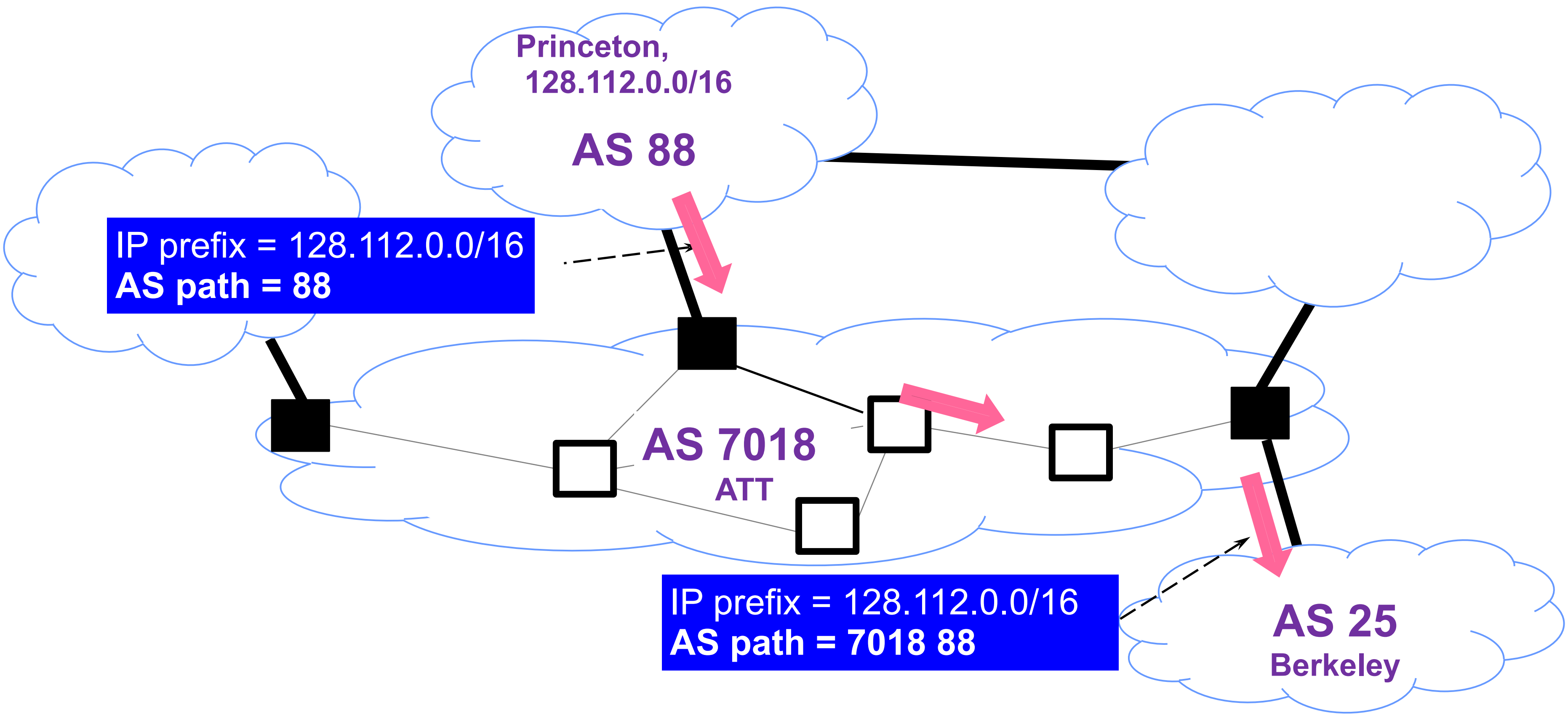
Attributes (1): ASPATH

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



Attributes (1): ASPATH

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

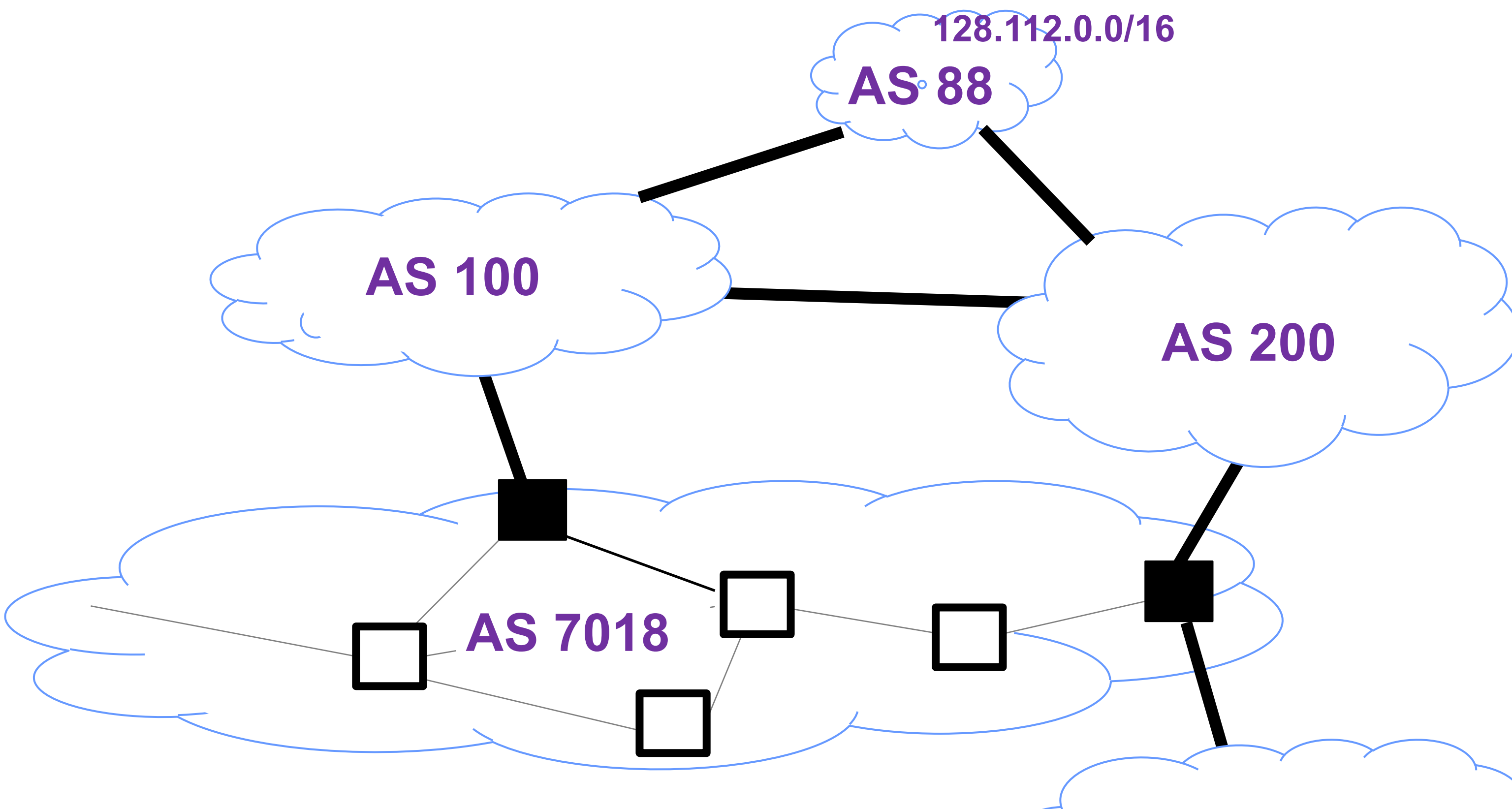


Attributes (2): LOCAL PREFERENCE

- 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

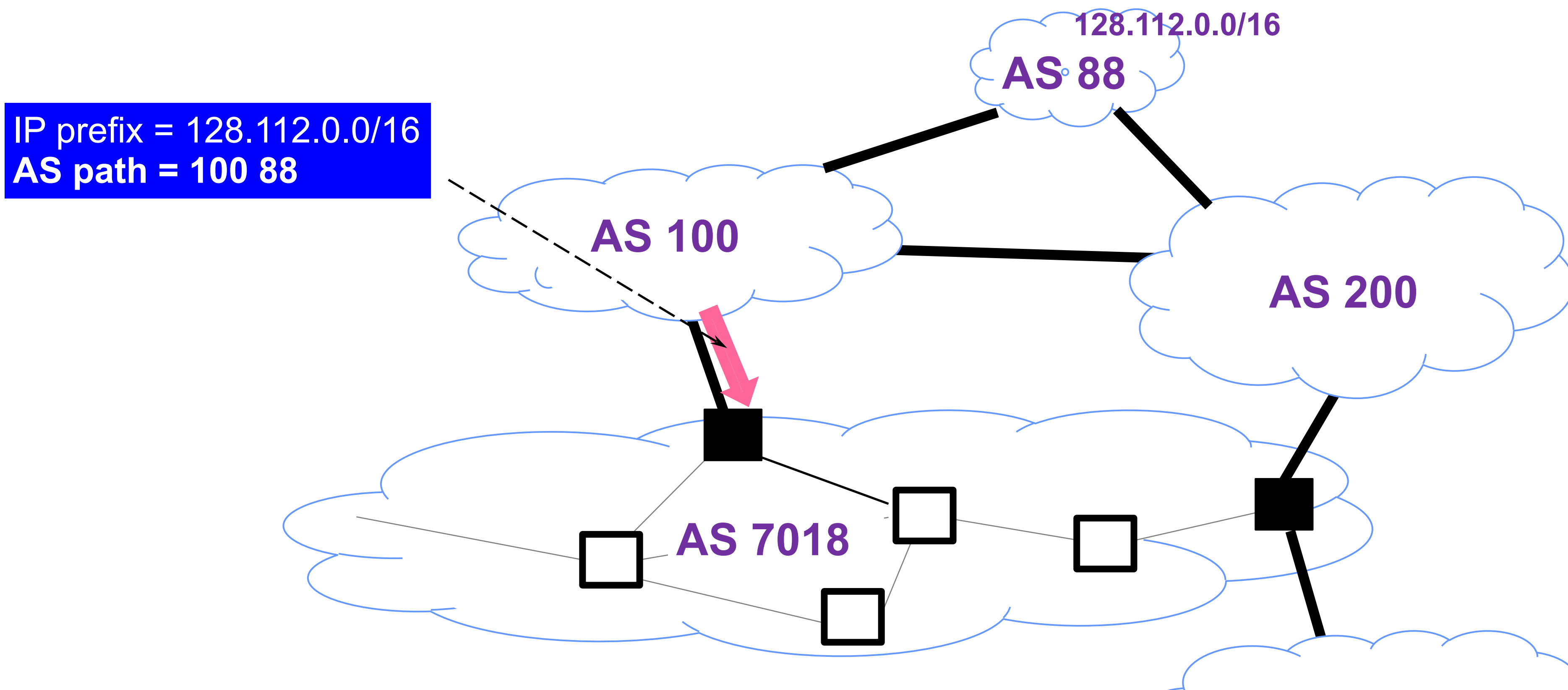
Attributes (2): LOCAL PREFERENCE

- 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



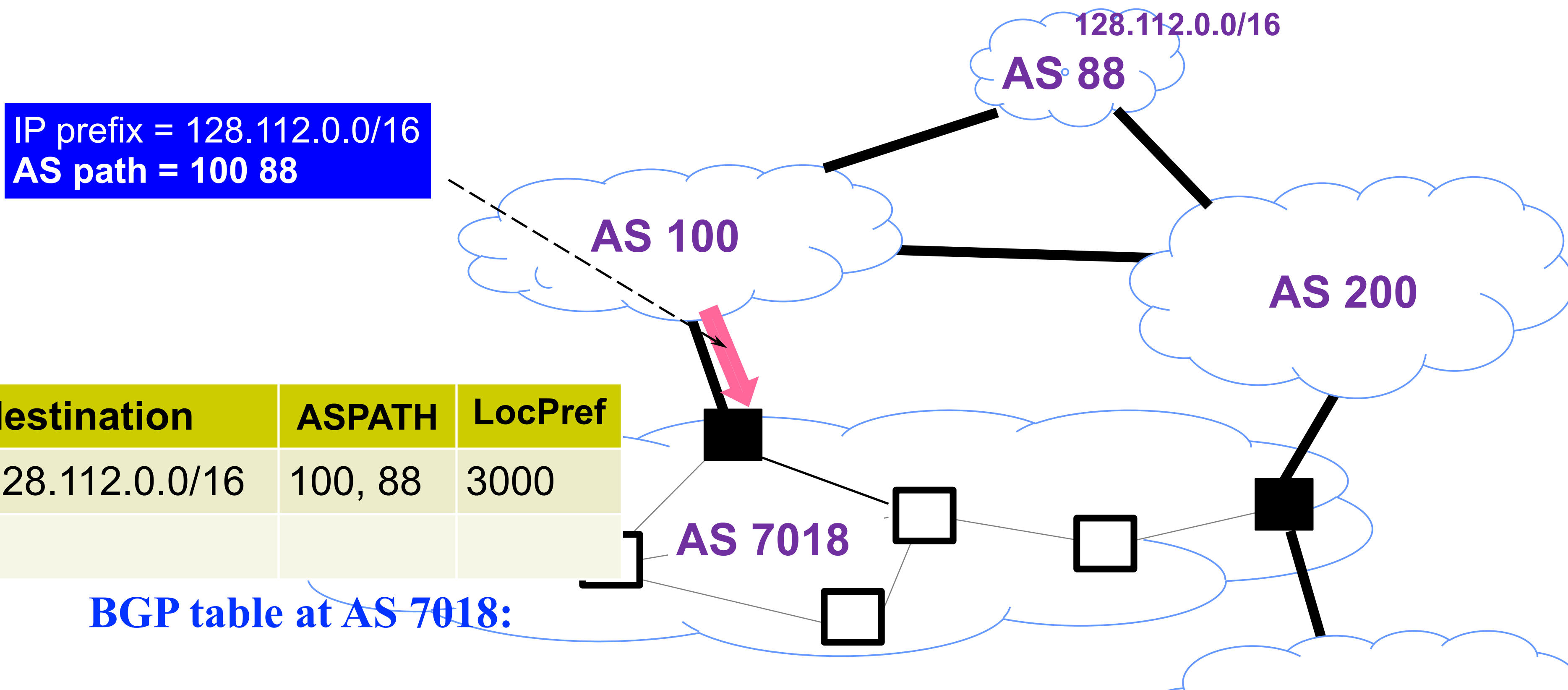
Attributes (2): LOCAL PREFERENCE

- 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



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

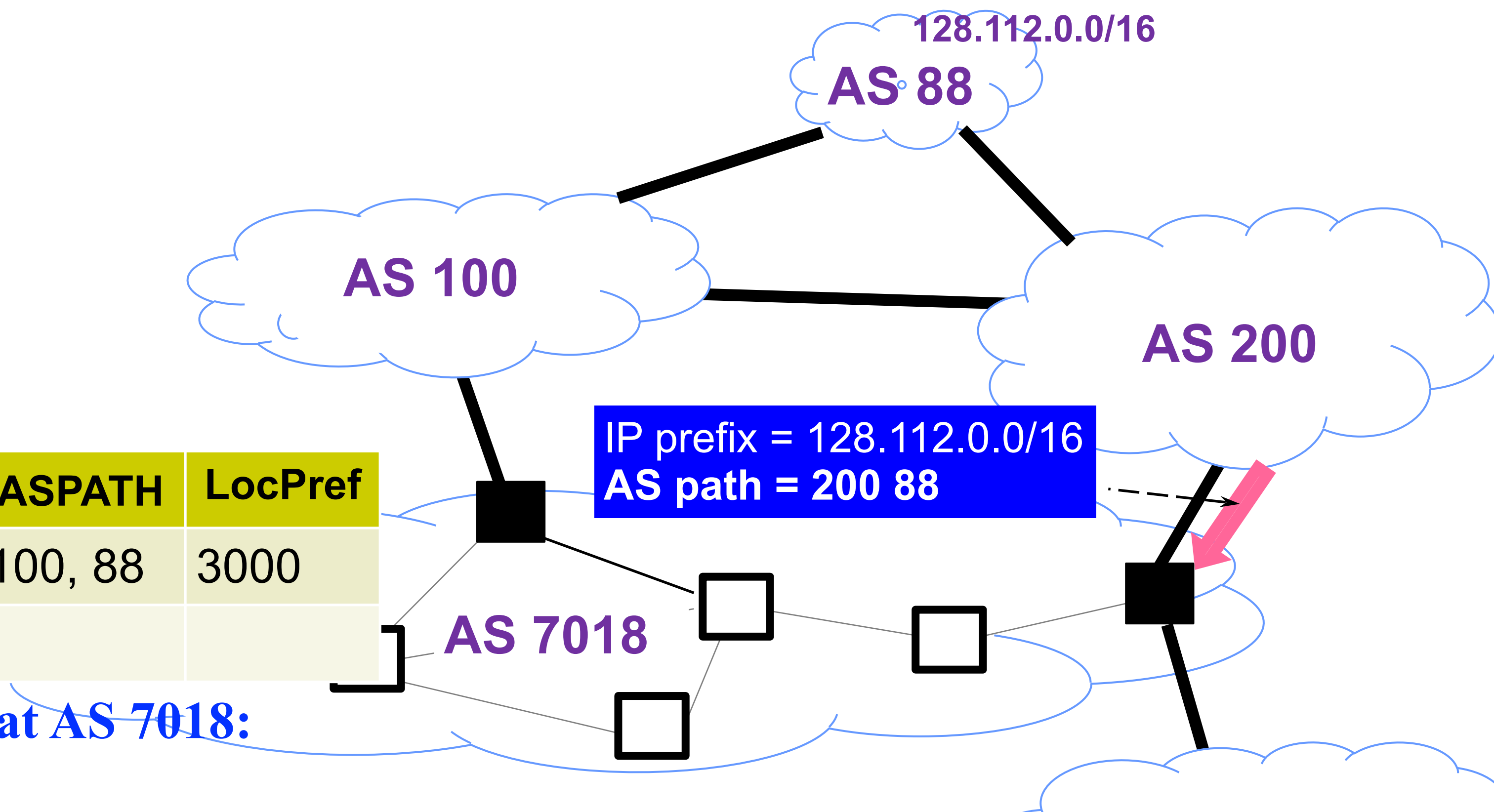


BGP table at AS 7018:

destination	ASPATH	LocPref
128.112.0.0/16	100, 88	3000

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

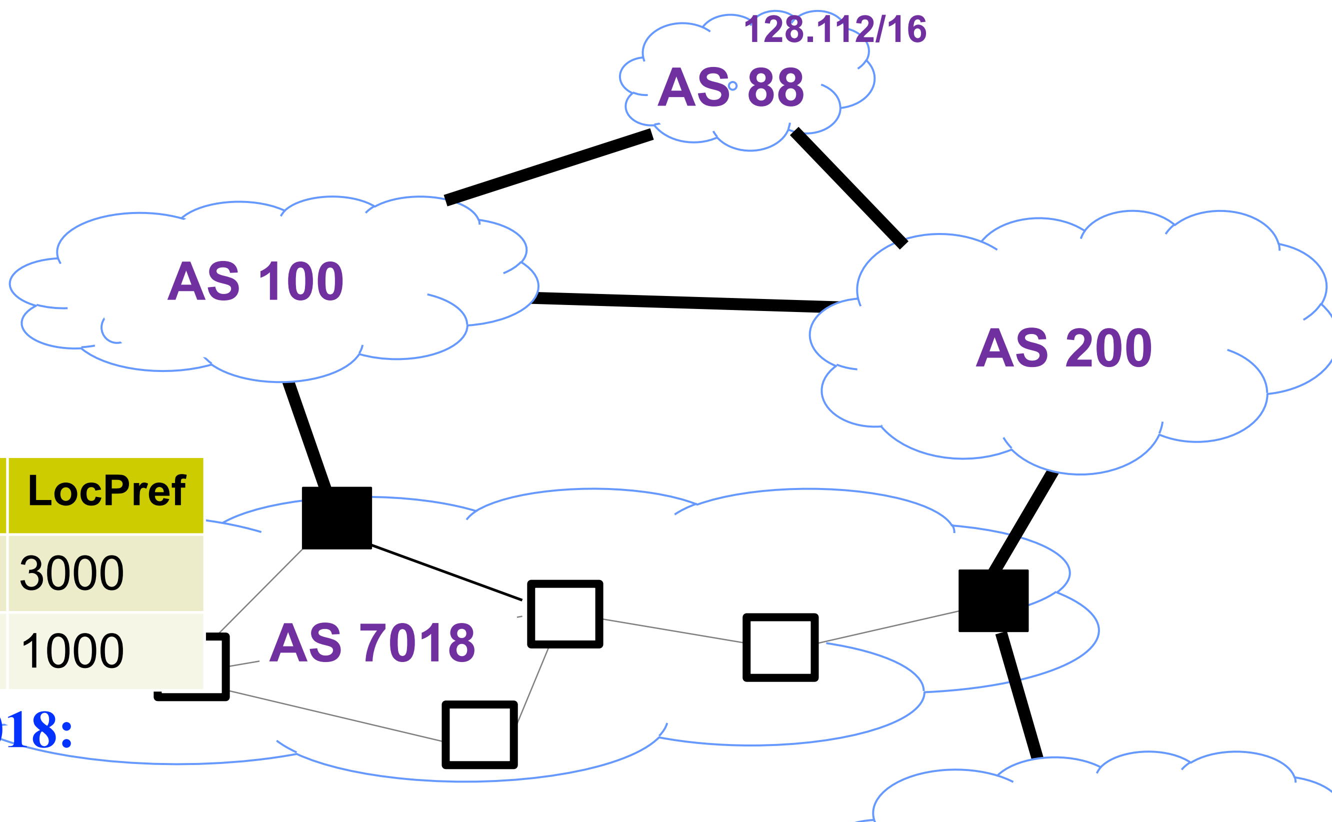


destination	ASPATH	LocPref
128.112.0.0/16	100, 88	3000

BGP table at AS 7018:

Attributes (2): LOCAL PREFERENCE

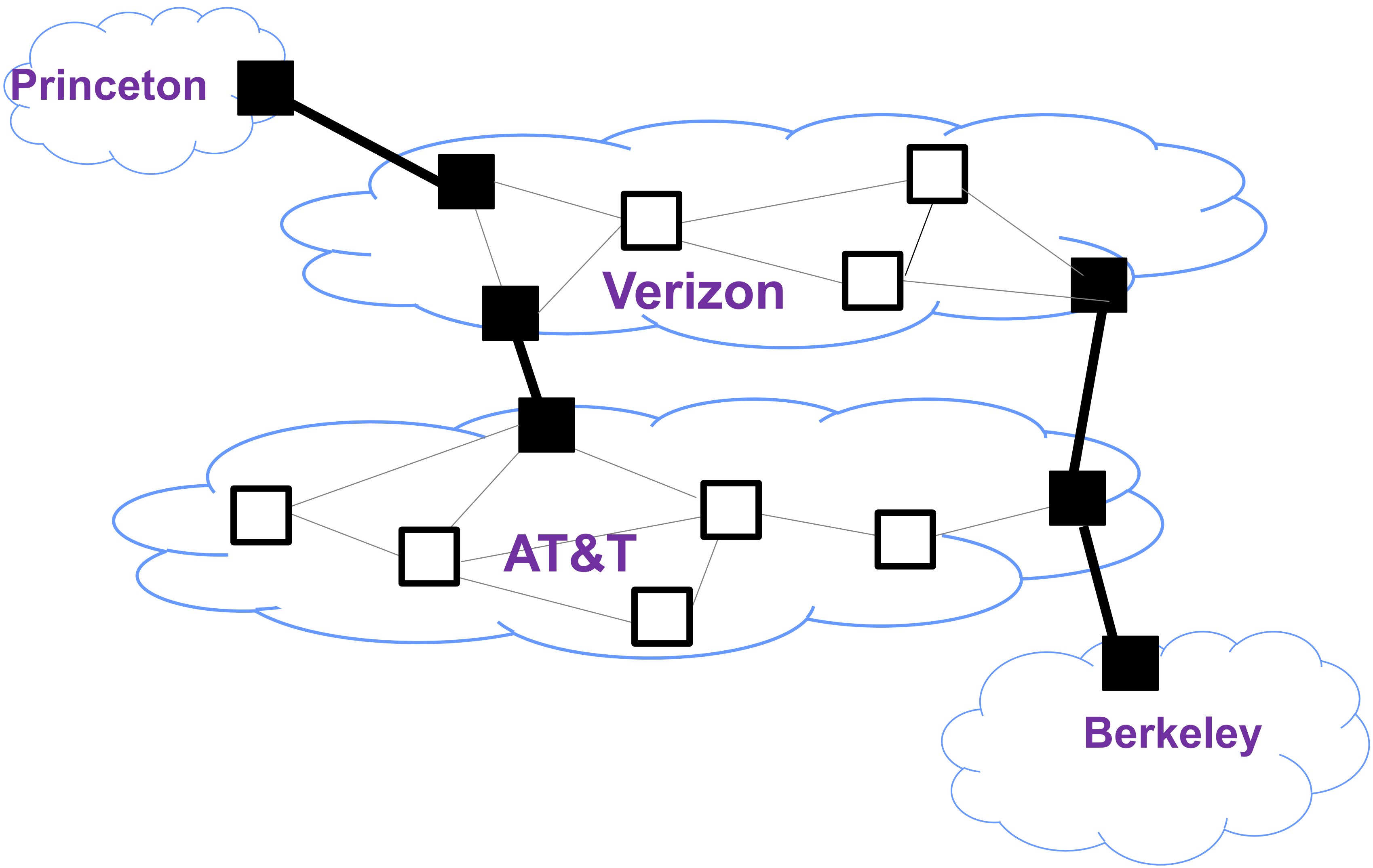
- Used to choose between different AS paths
- Local to an AS; carried only in iBGP messages
- The higher the value the more that route is preferred



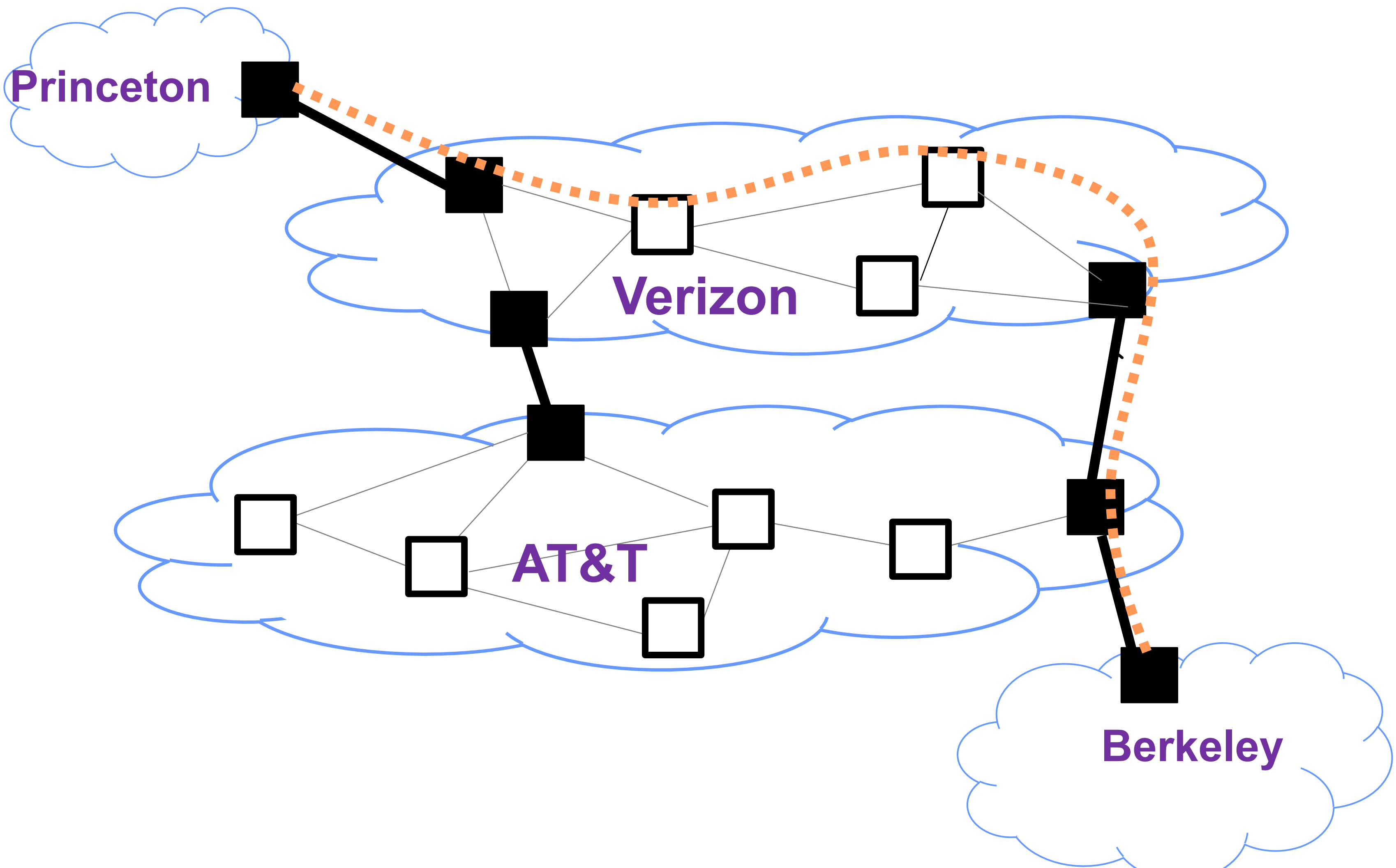
destination	ASPATH	LocPref
12.112.0.0/16	100, 88	3000
12.112.0.0/16	200, 88	1000

BGP table at AS 7018:

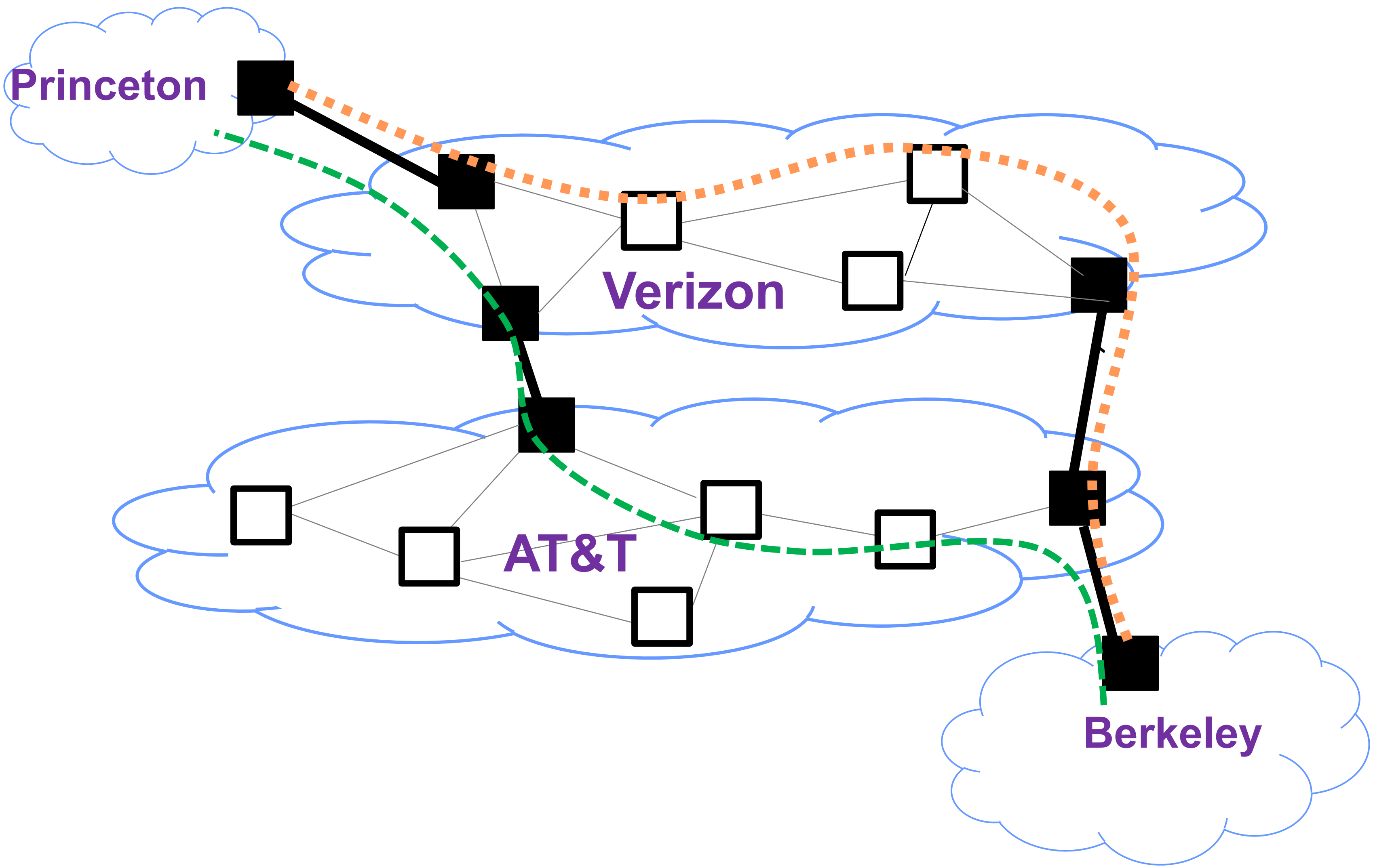
In reality...



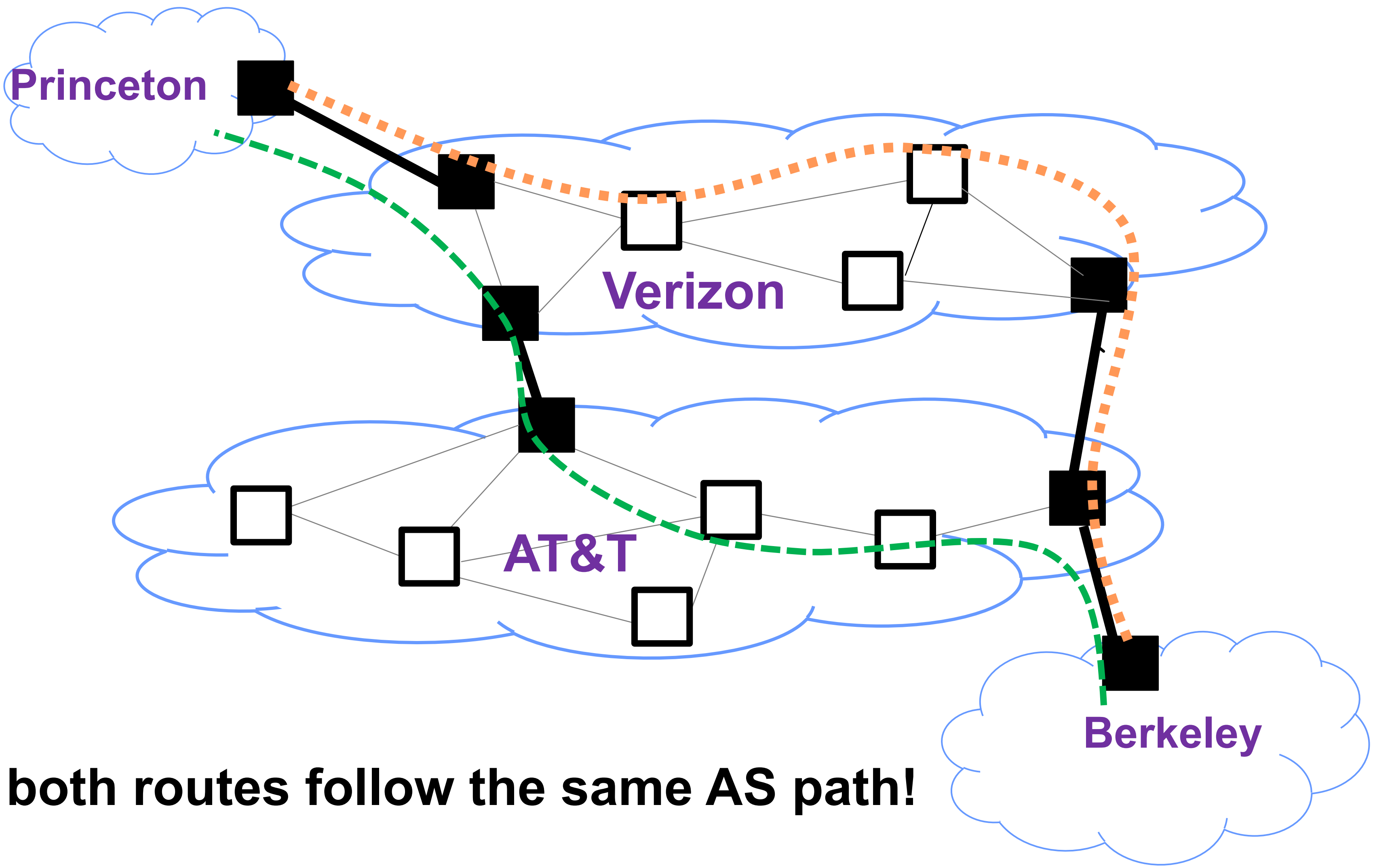
In reality...



In reality...

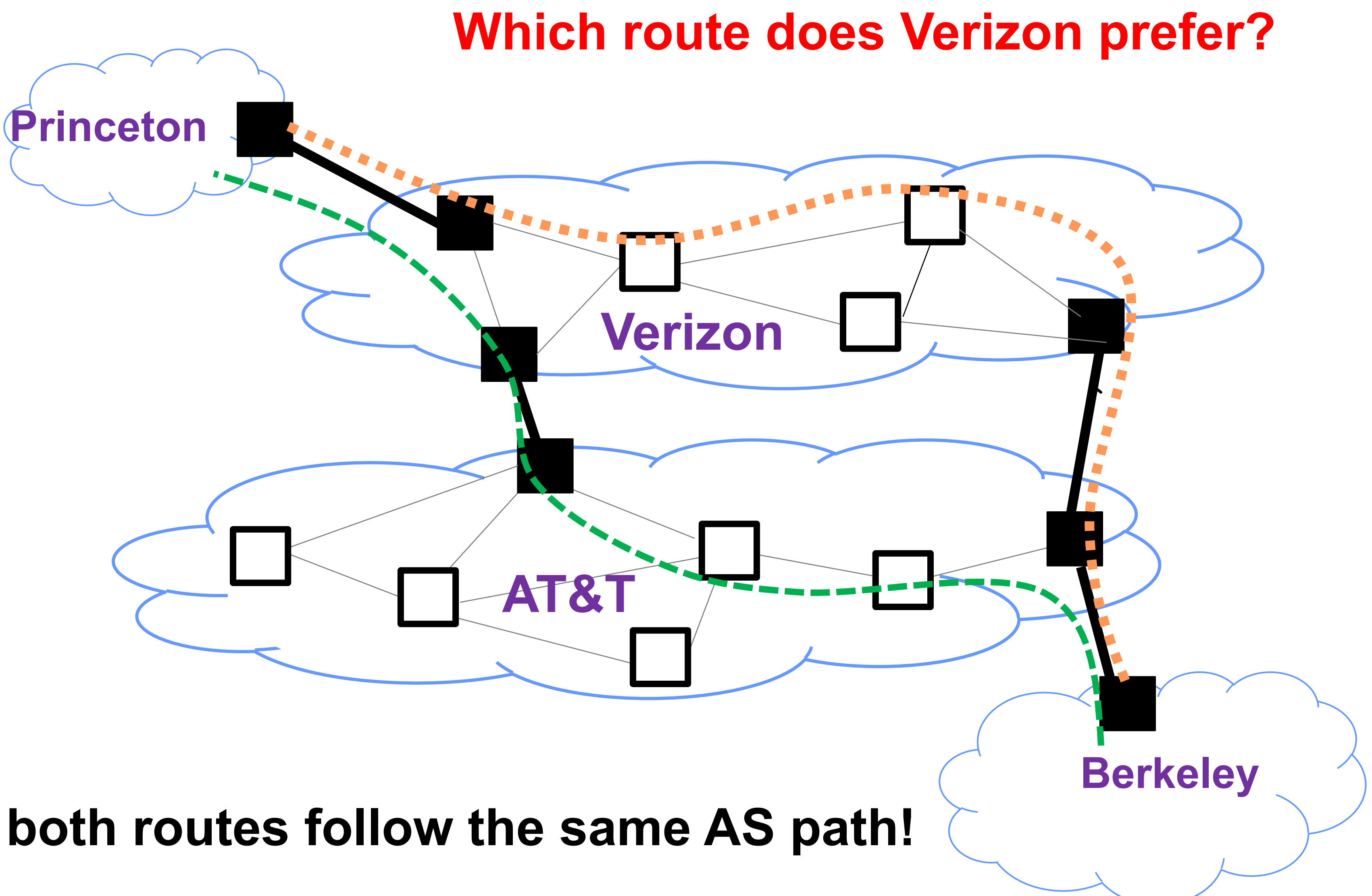


In reality...



Note: both routes follow the same AS path!

In reality...

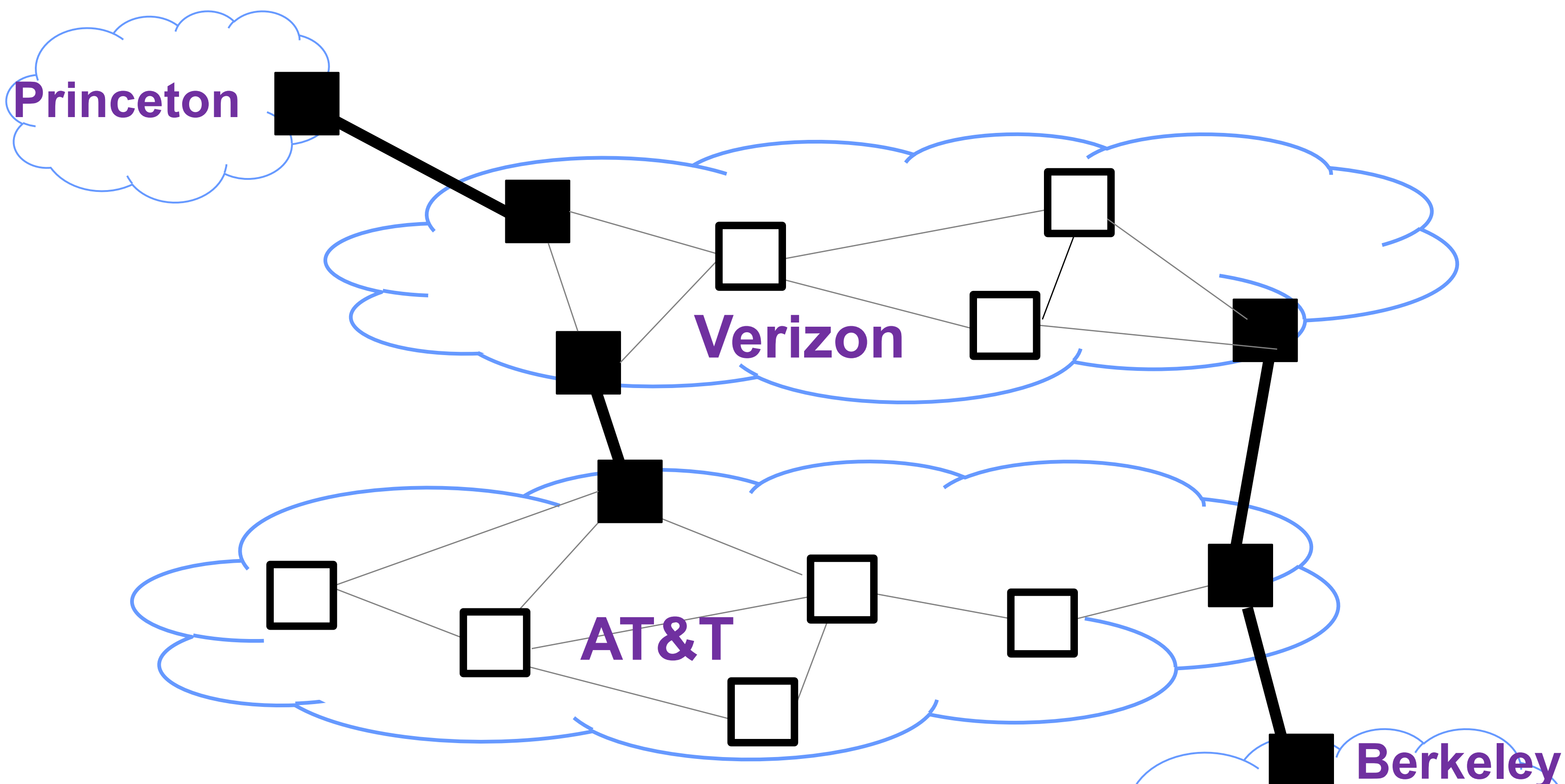


Note: both routes follow the same AS path!

Attributes (3) : `MED`

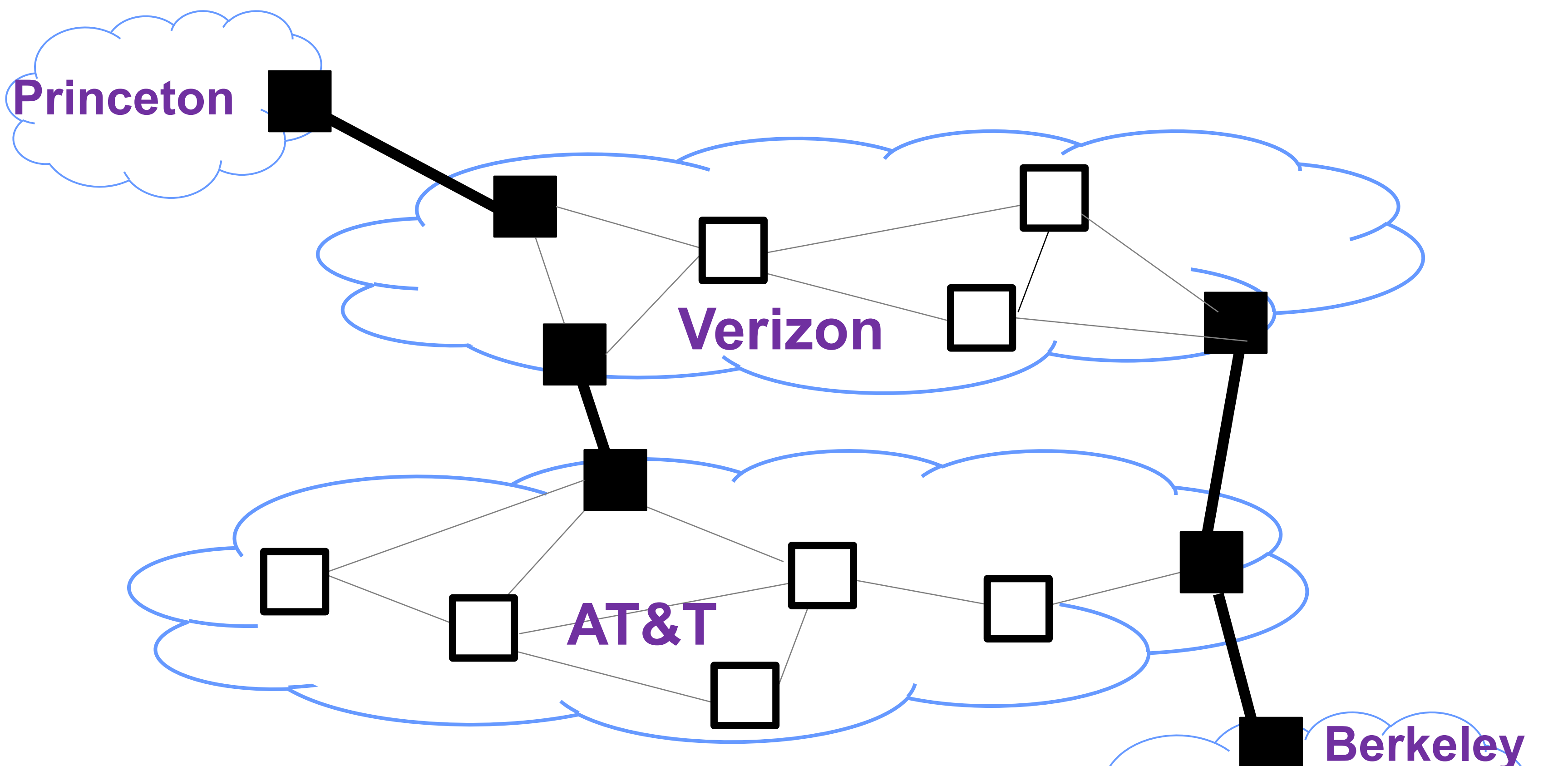
- MED = “Multi-Exit Discriminator”
- Used when ASes are interconnected via 2 or more links to specify how close a prefix is to the link it is announced on

Attributes (3) : MED



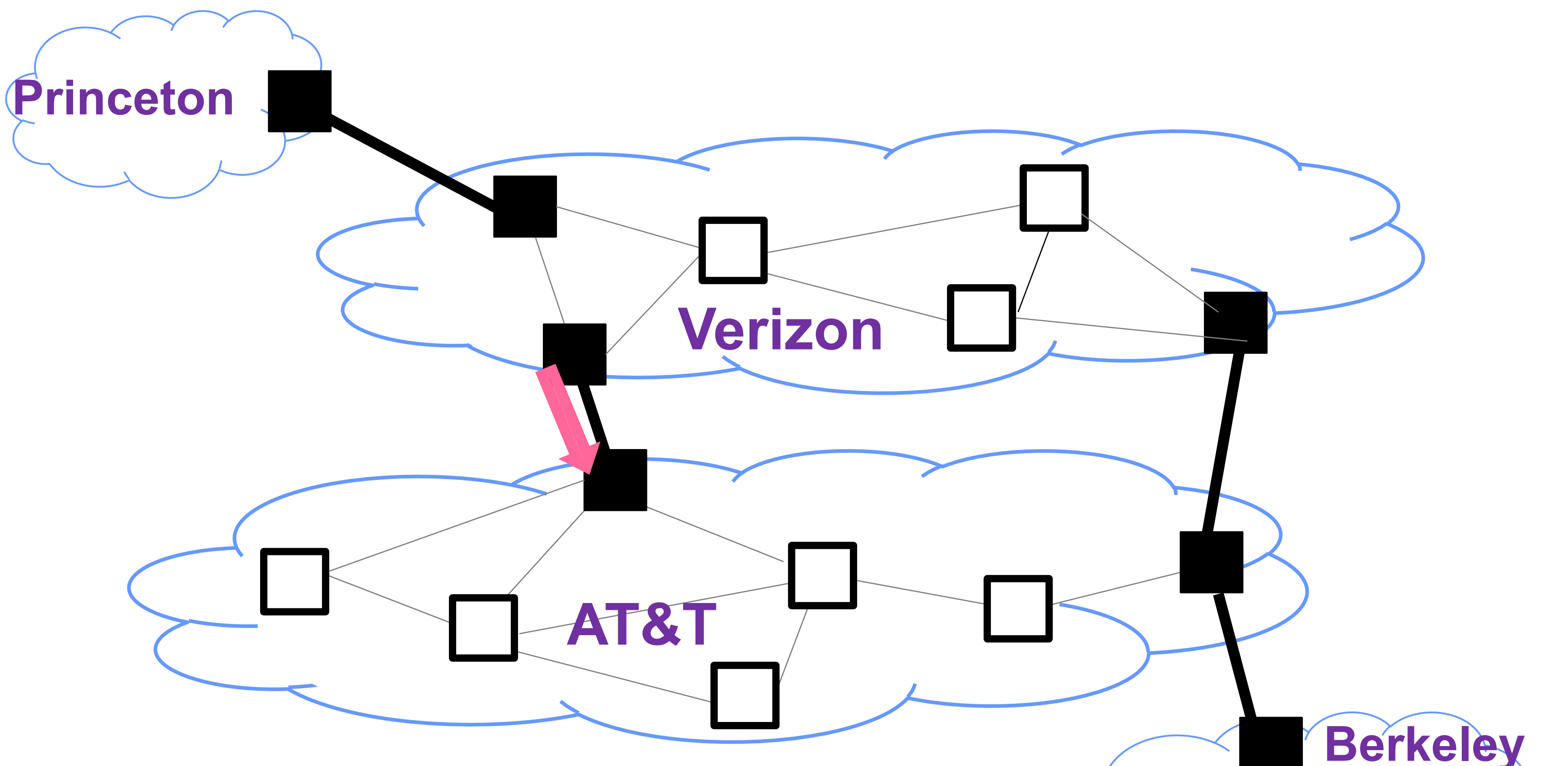
Attributes (3) : MED

- AS announcing prefix sets MED (lower is better)



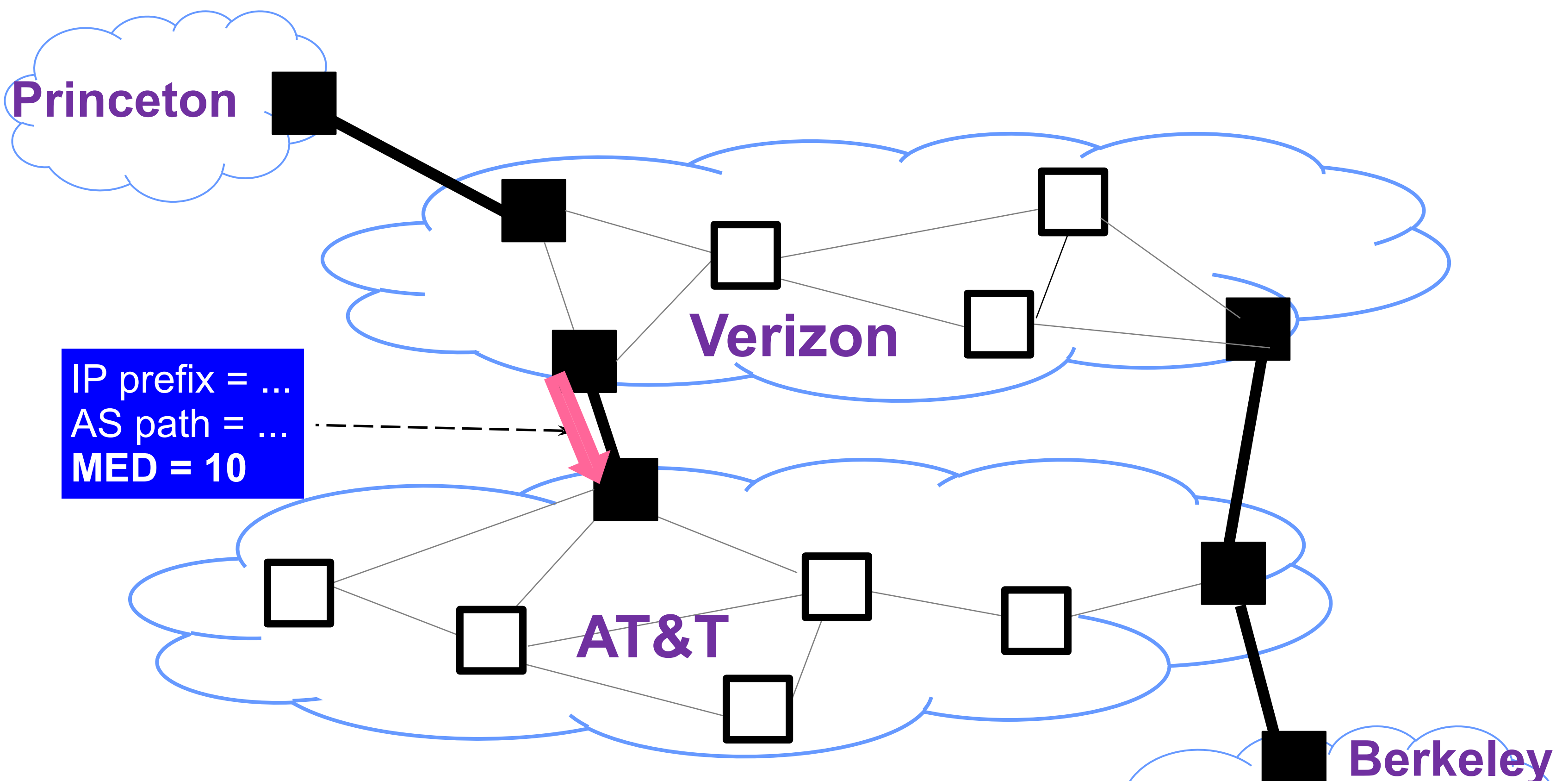
Attributes (3) : MED

- AS announcing prefix sets MED (lower is better)



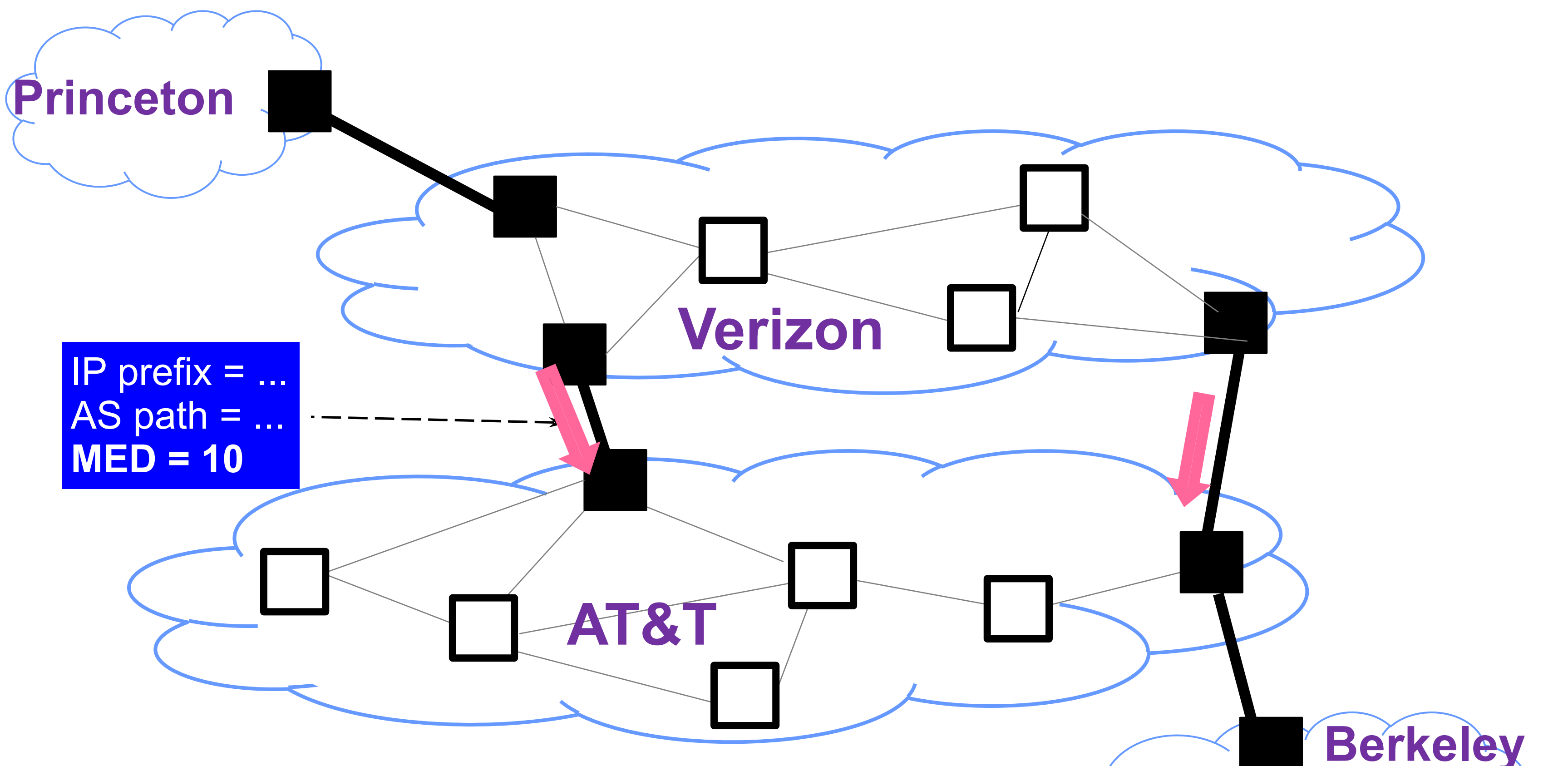
Attributes (3) : MED

- AS announcing prefix sets MED (lower is better)



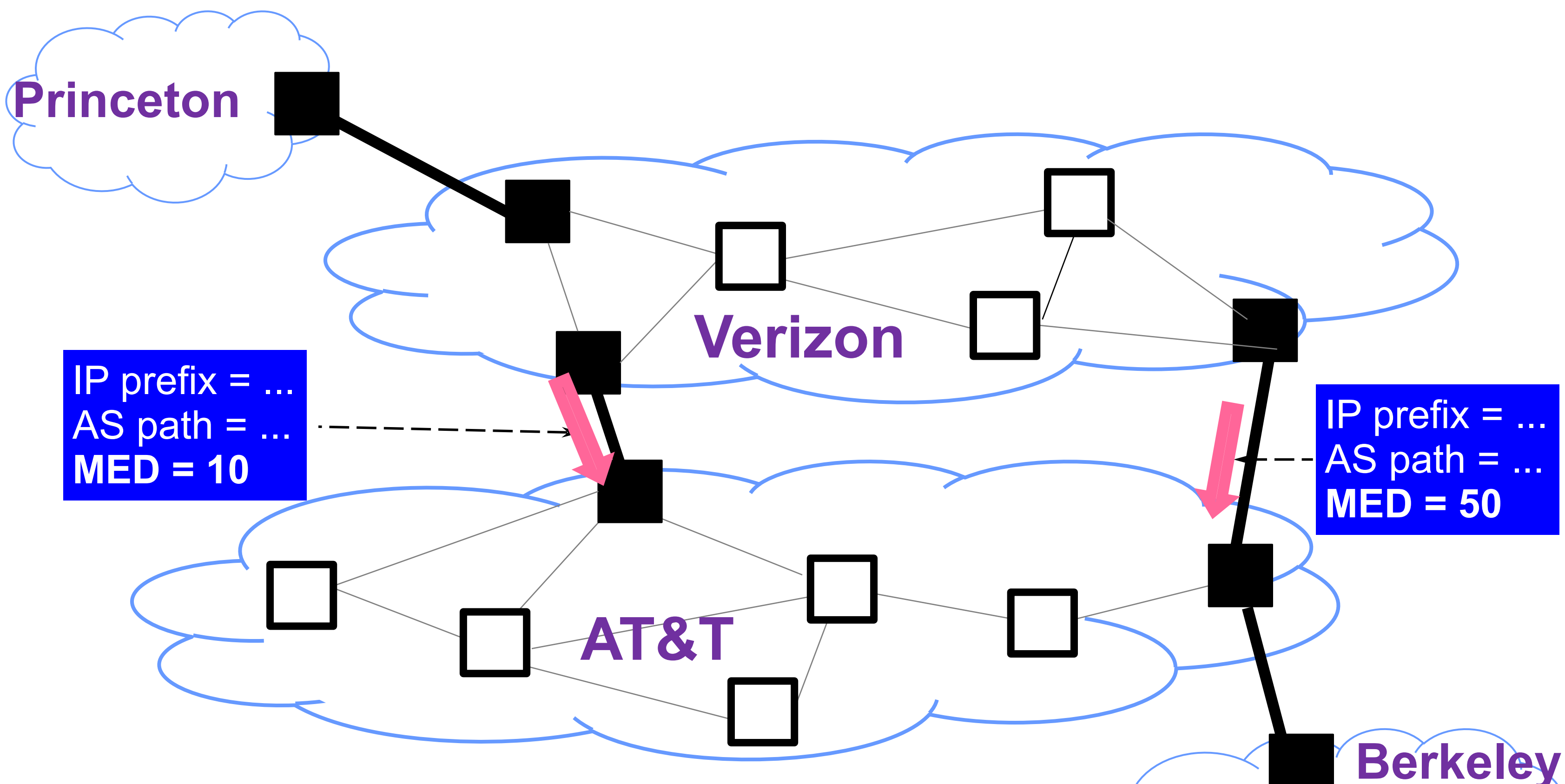
Attributes (3) : `MED`

- AS announcing prefix sets MED (lower is better)



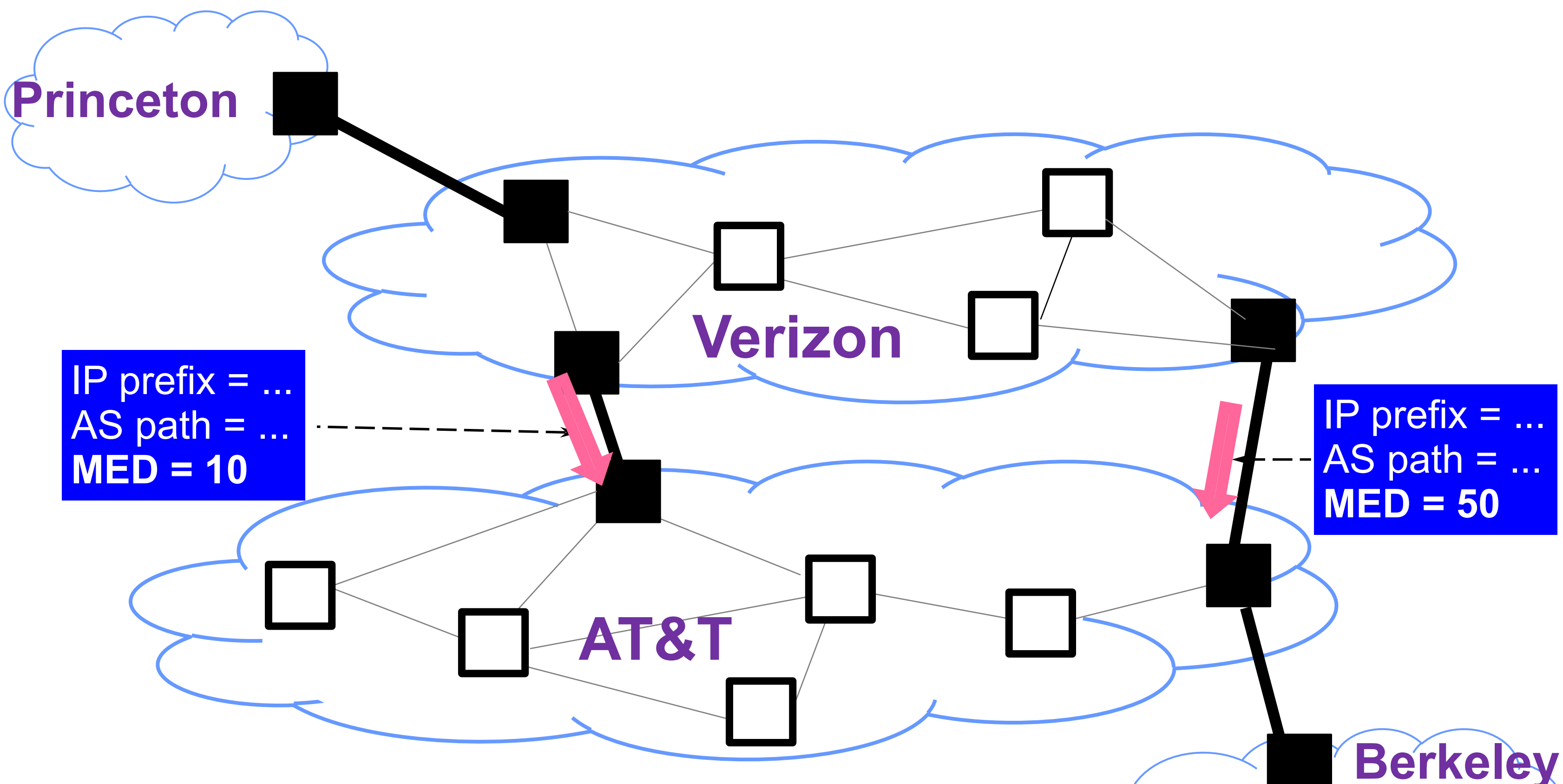
Attributes (3) : `MED`

- AS announcing prefix sets MED (lower is better)

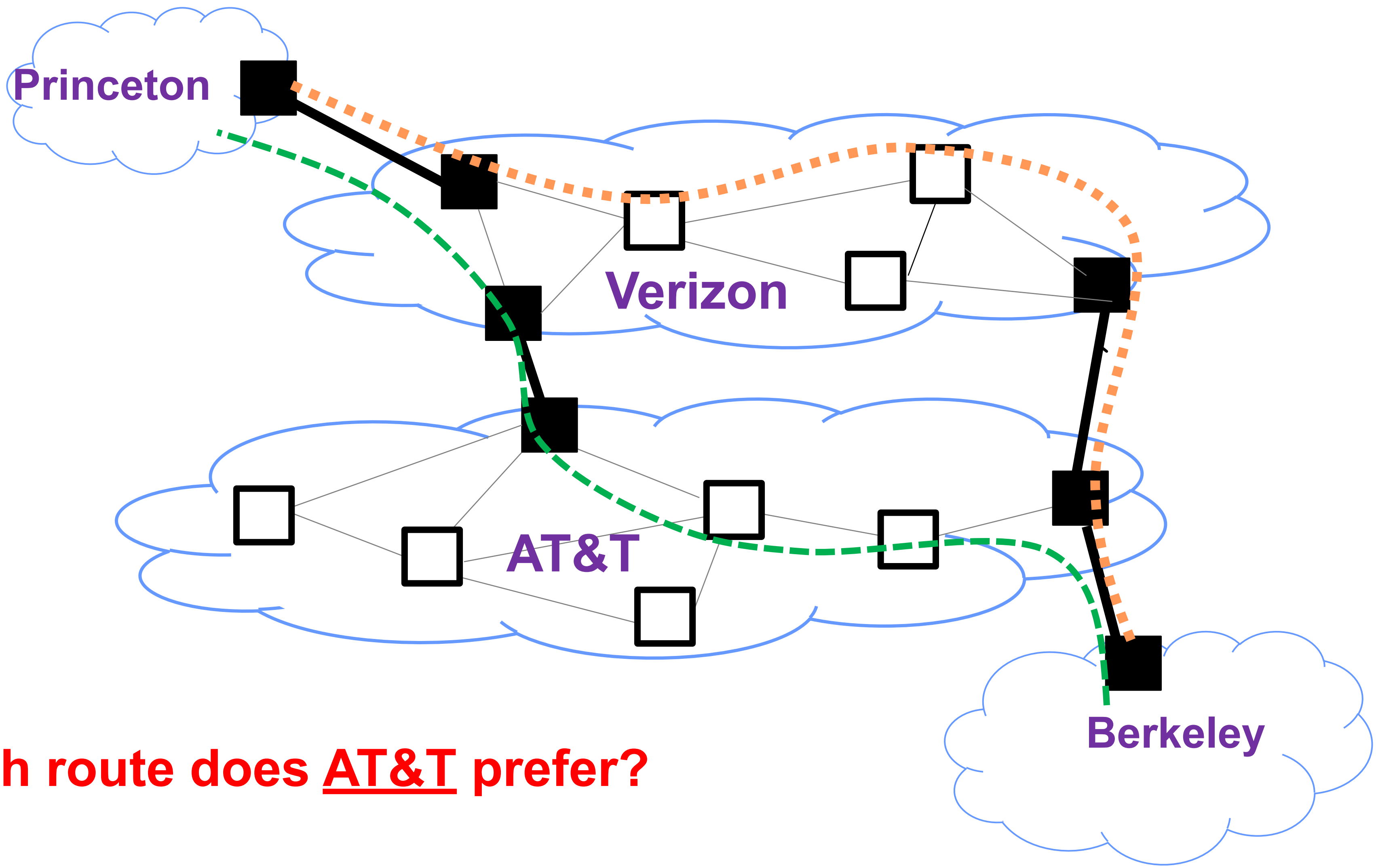


Attributes (3) : MED

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



More reality...



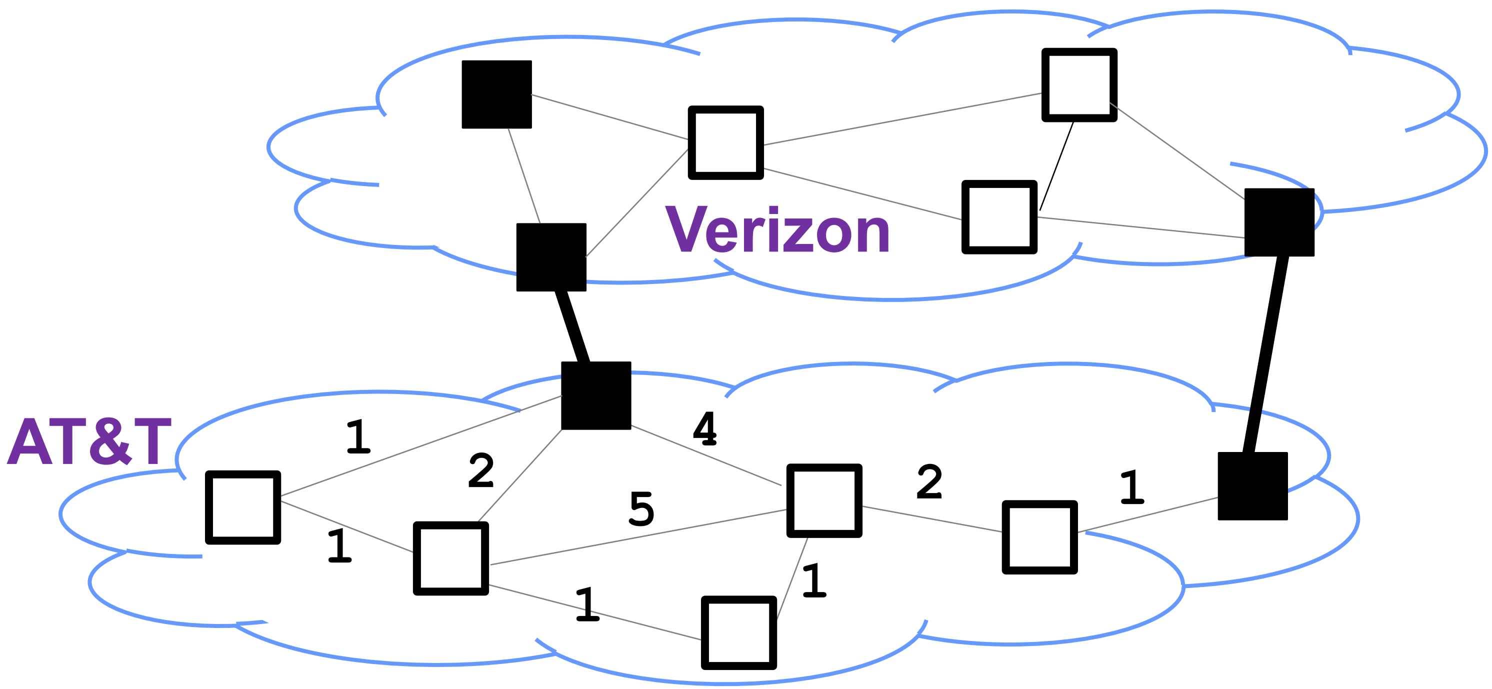
Which route does **AT&T** prefer?

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

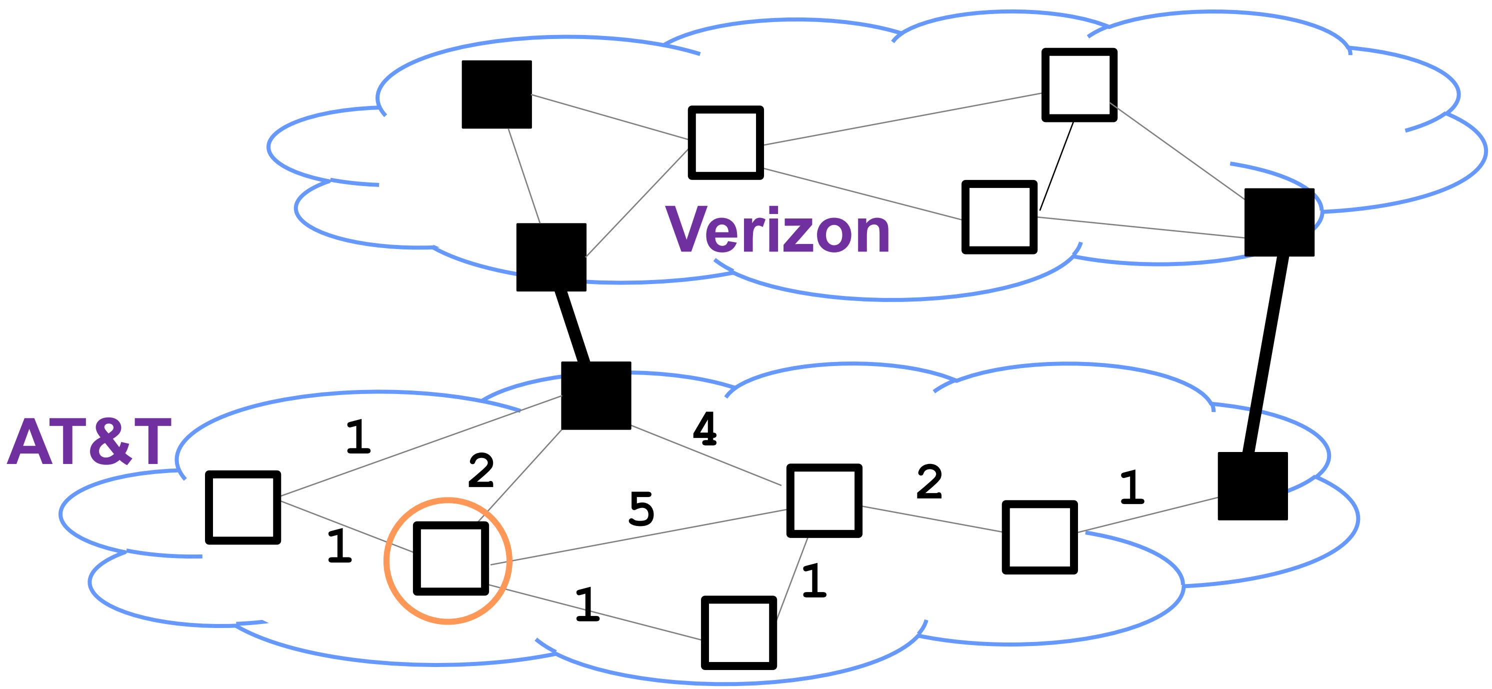
Attributes (4): IGP cost

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



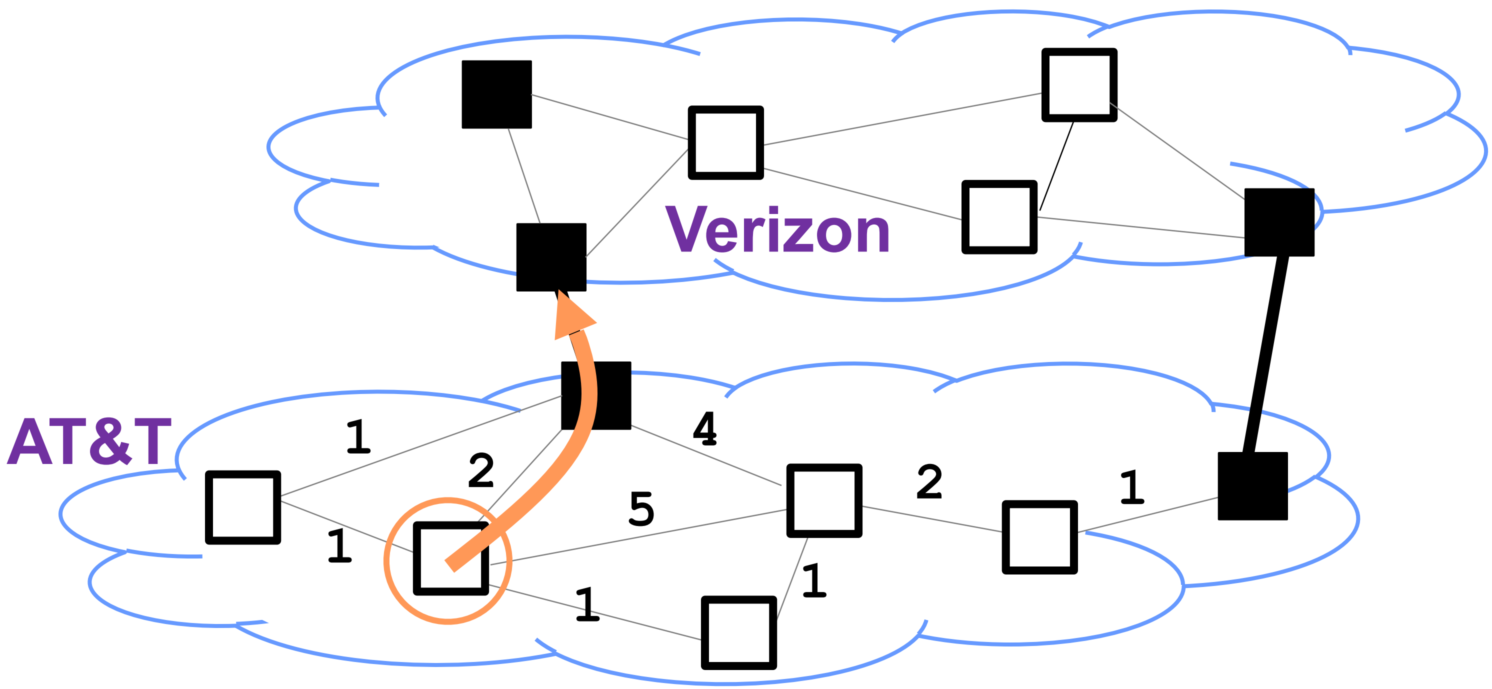
Attributes (4): IGP cost

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



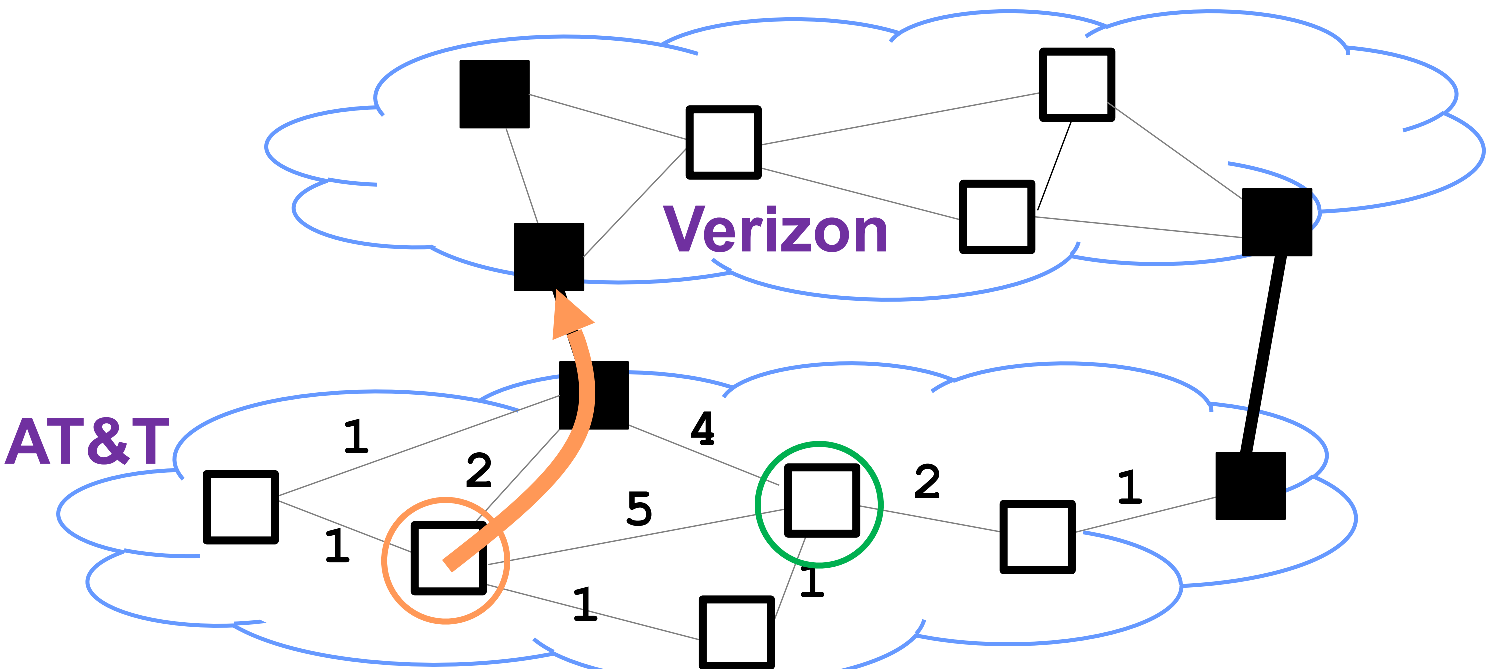
Attributes (4): IGP cost

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



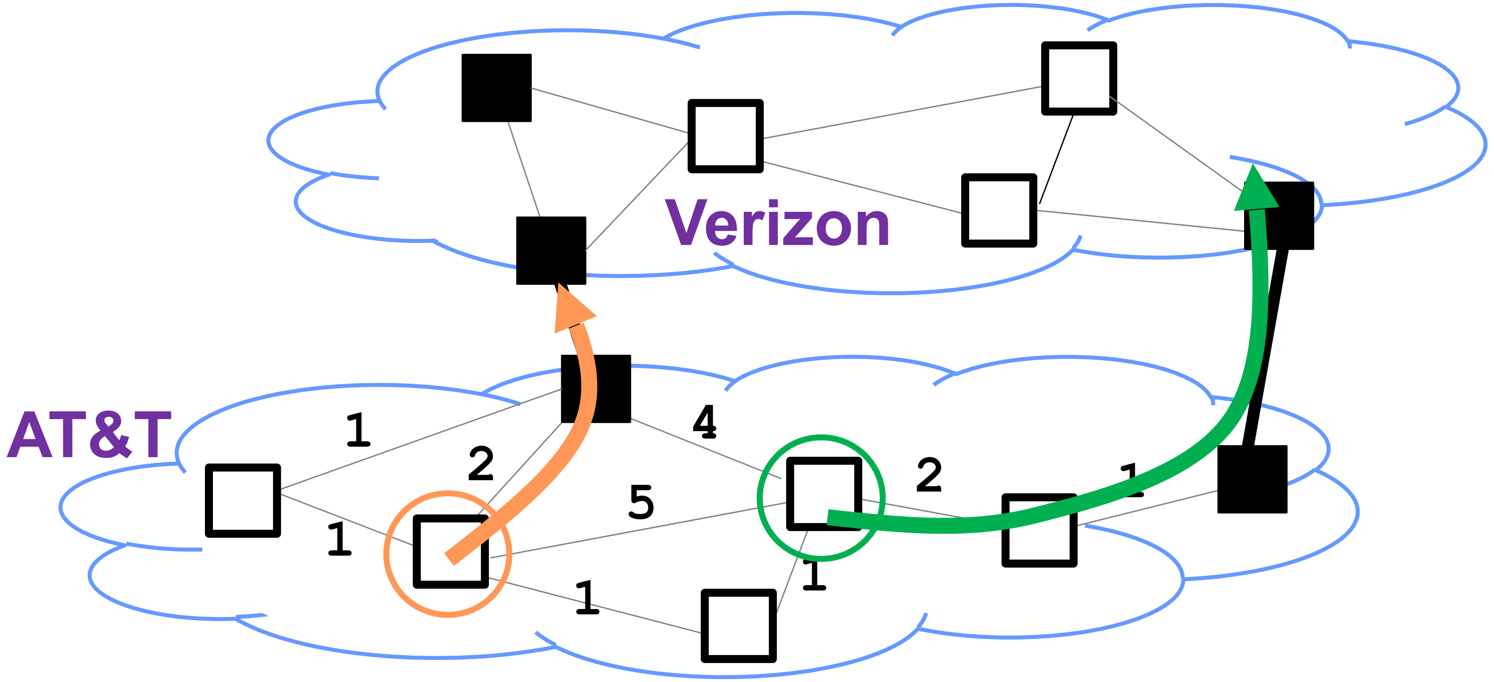
Attributes (4): IGP cost

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

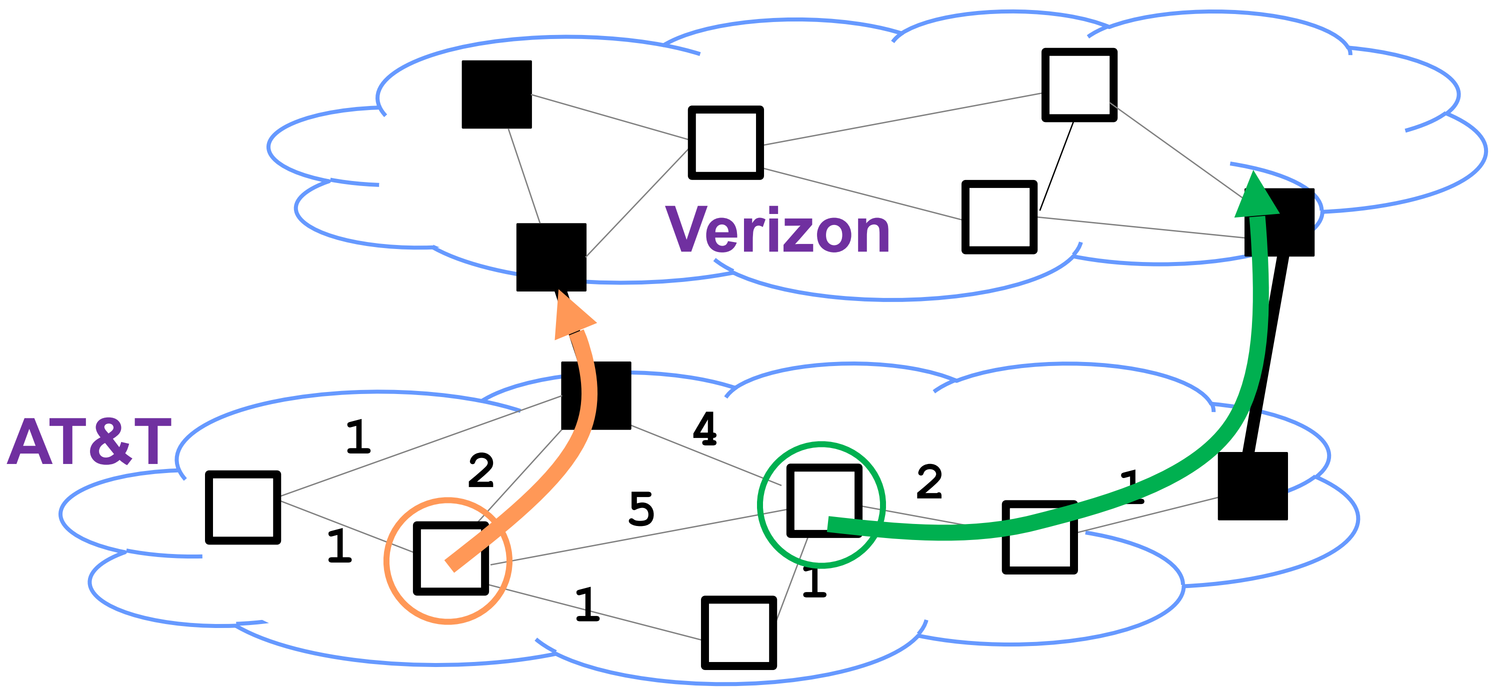


Attributes (4): IGP cost

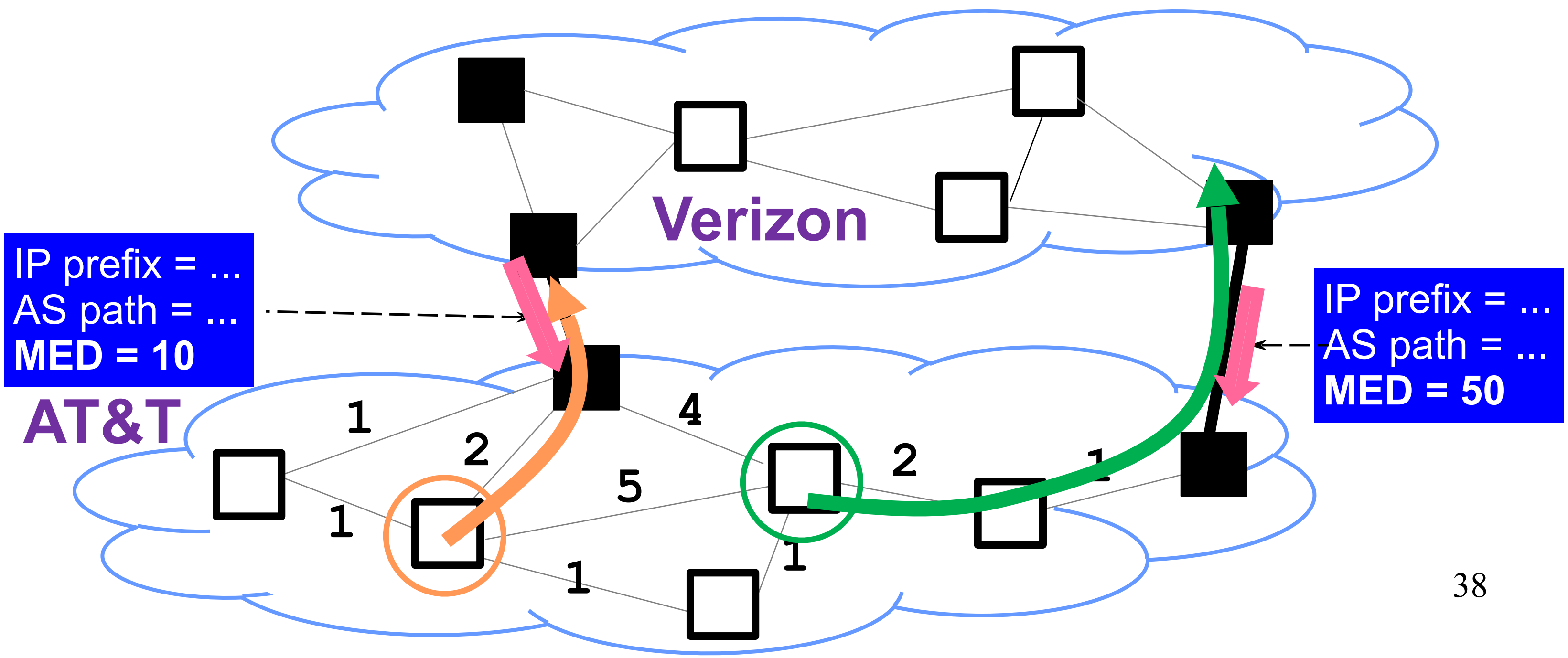
- Local to an AS
- Each router selects its closest border router
  - Closest based on IGP cost
  - a.k.a. "hot potato" routing



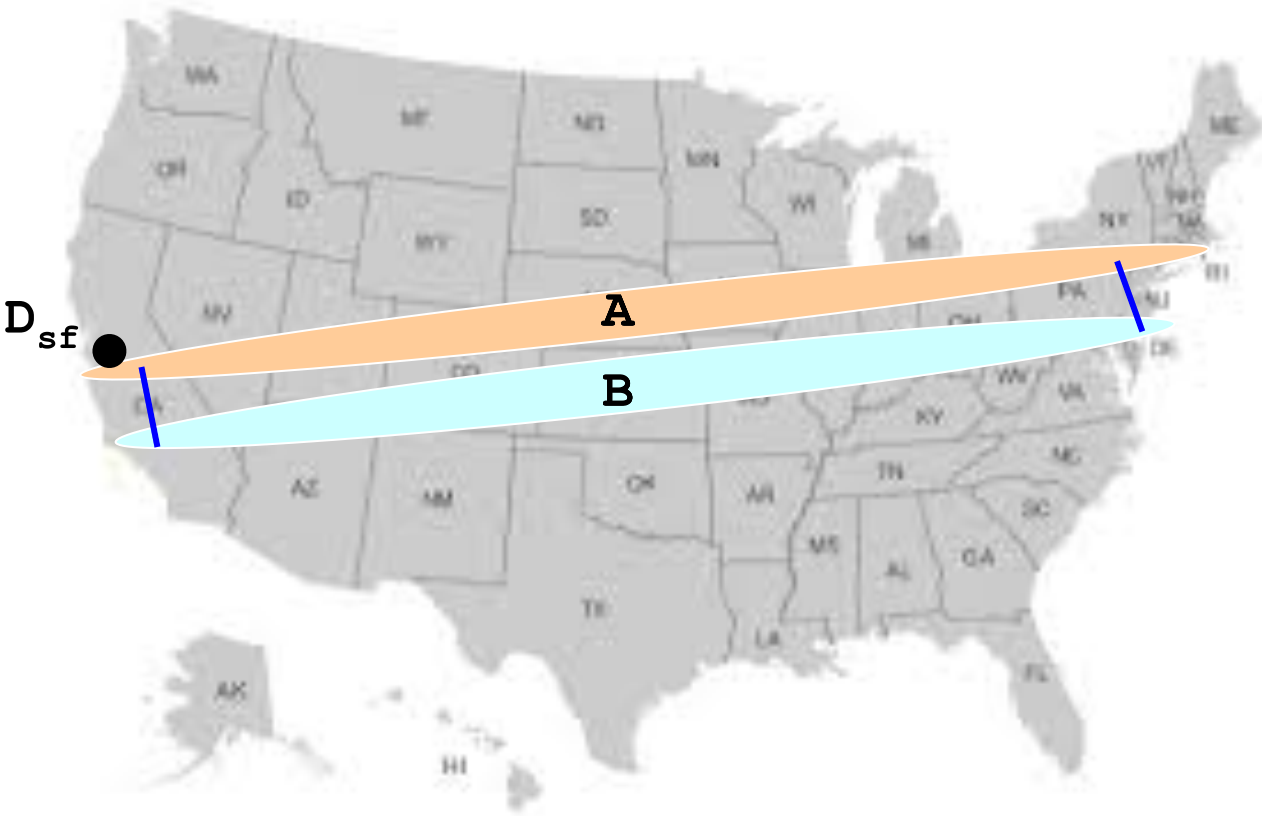
Note: IGP may conflict with MED



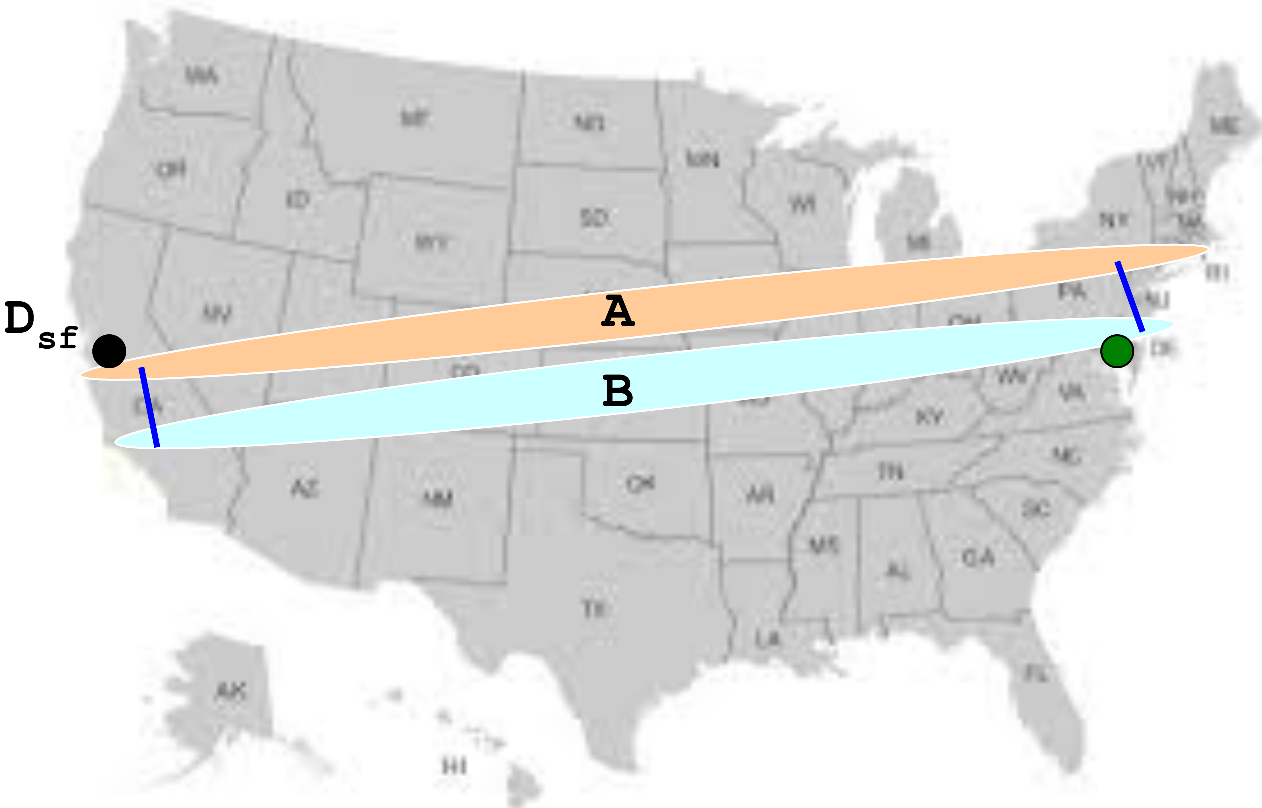
Note: IGP may conflict with MED



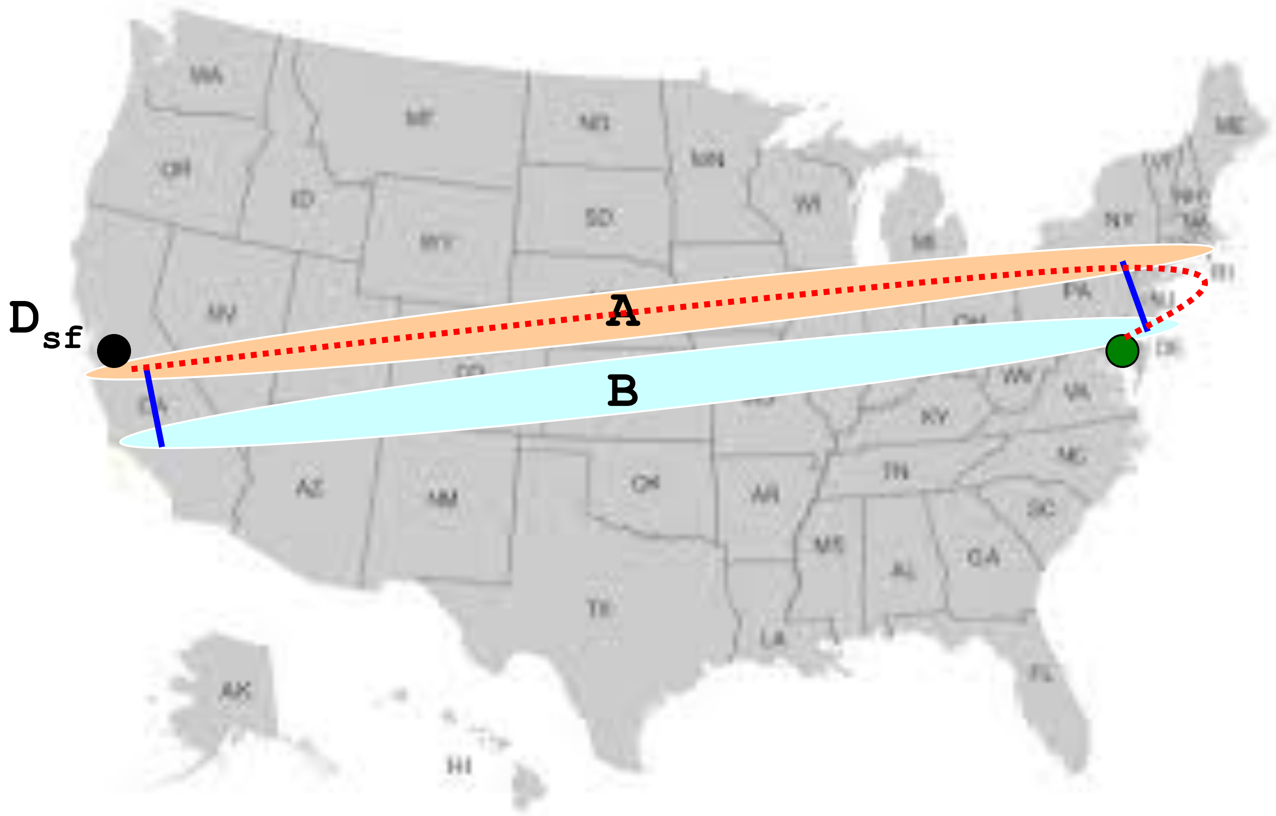
IGP-MED conflicts pretty common



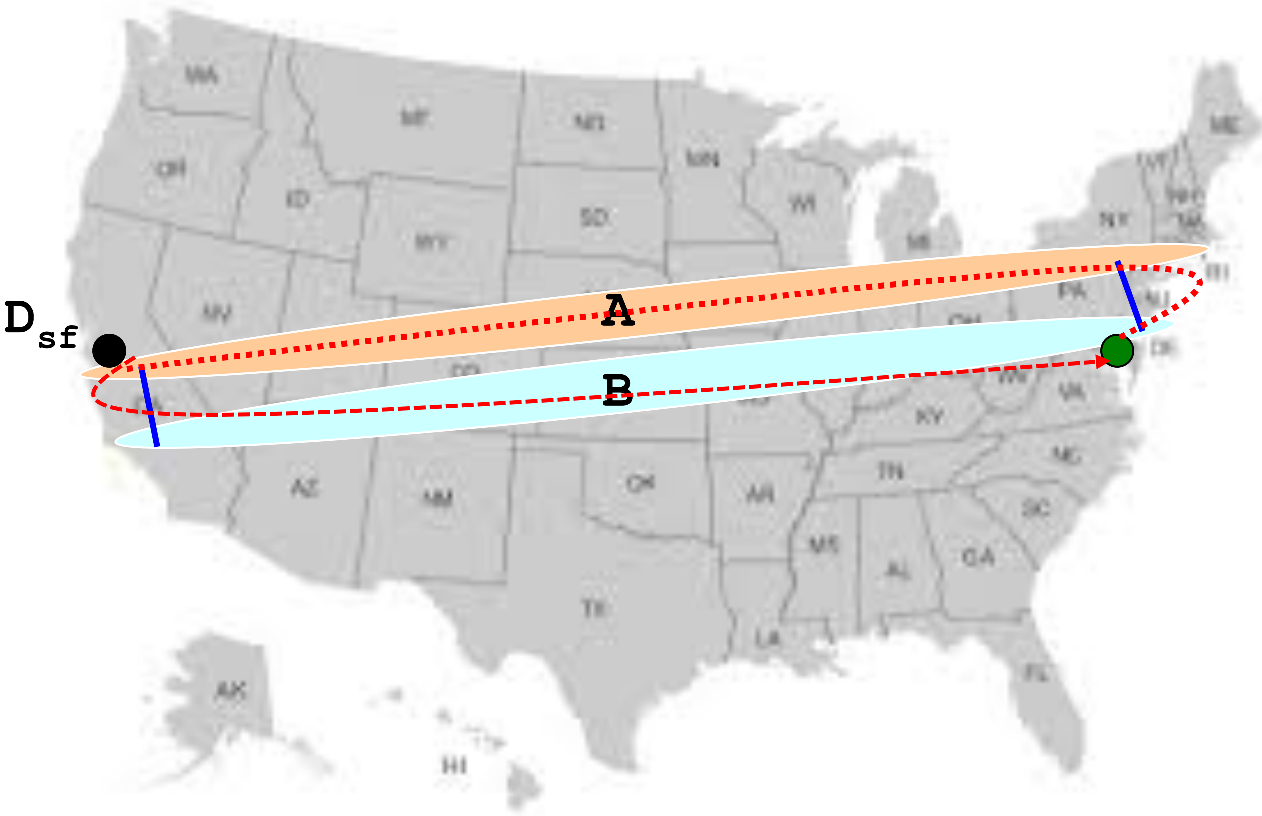
IGP-MED conflicts pretty common



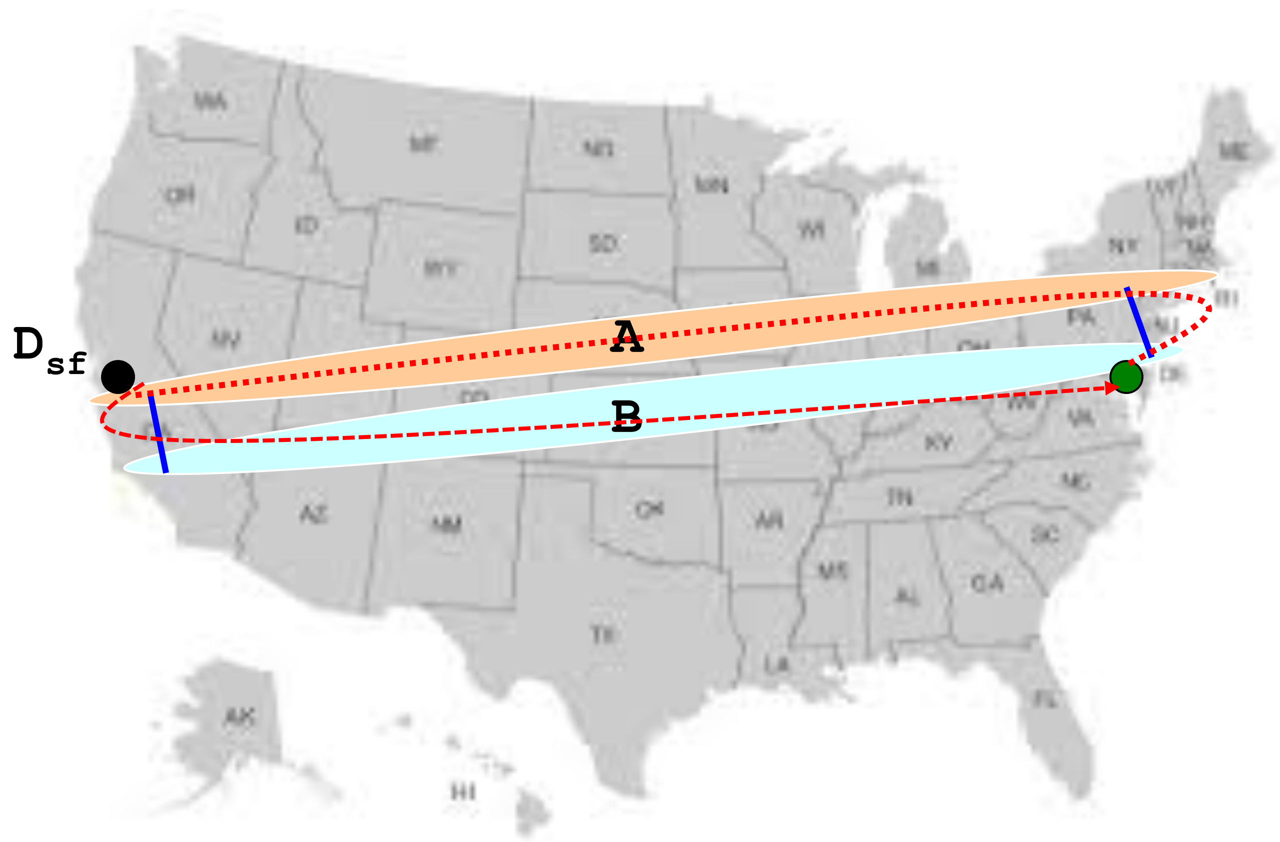
IGP-MED conflicts pretty common



IGP-MED conflicts pretty common



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

Issues with BGP

- Security

Issues with BGP

- Security
- Performance (non?)issues

Issues with BGP

- Security

- Performance (non?)issues

- Prone to misconfiguration

Issues with BGP

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

Issues with BGP

- Security
  - No guarantee that an AS owns the prefixes it advertises!
  - No guarantee that an AS will follow the path it advertises
- Performance (non?)issues

Issues with BGP

- Security
  - No guarantee that an AS owns the prefixes it advertises!
  - No guarantee that an AS will follow the path it advertises
- Performance (non?)issues
  - Policy-based paths not necessarily shortest/least-cost
  - AS path length can be misleading
- Prone to misconfiguration

Issues with BGP

- Security
  - No guarantee that an AS owns the prefixes it advertises!
  - No guarantee that an AS will follow the path it advertises
- Performance (non?)issues
  - Policy-based paths not necessarily shortest/least-cost
  - AS path length can be misleading
- Prone to misconfiguration
  - Many attributes; configuration often manual and ad-hoc
  - BGP misconfigurations a major source of Internet outages!
- Reachability and Convergence

Issues with BGP

- Security
  - No guarantee that an AS owns the prefixes it advertises!
  - No guarantee that an AS will follow the path it advertises
- Performance (non?)issues
  - Policy-based paths not necessarily shortest/least-cost
  - AS path length can be misleading
- Prone to misconfiguration
  - Many attributes; configuration often manual and ad-hoc
  - BGP misconfigurations a major source of Internet outages!
- Reachability and Convergence
  - Not guaranteed if Gao-Rexford doesn’t hold
  - Example of policy oscillations in discussion section

Questions?

Taking Stock: We've done...

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

- How L3 works
  - IP addressing and routers
  - Intra-domain routing
  - Inter-domain routing

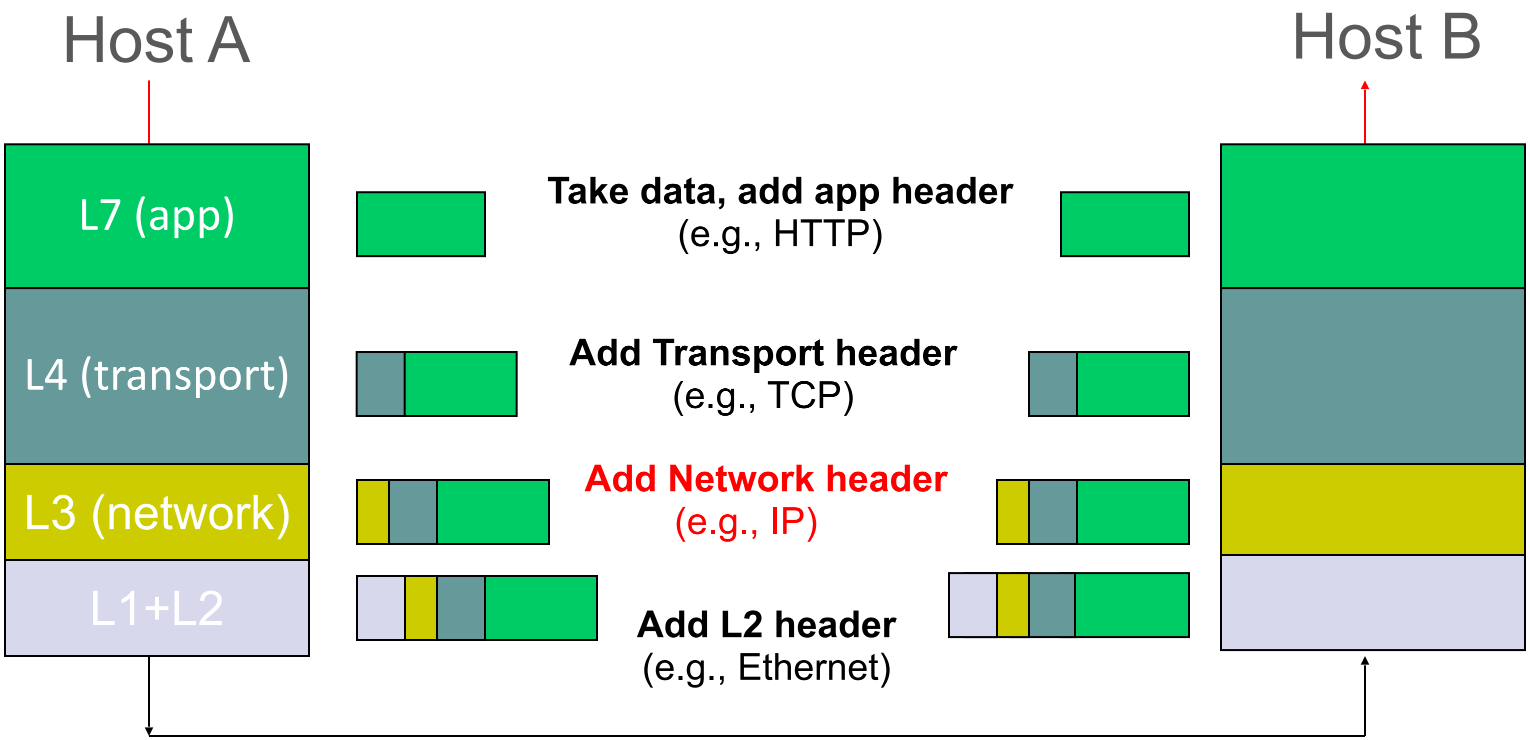
- **Last topic: the IP header**
  - At which point, you'll know how L3 works!

Let’s design the IP header

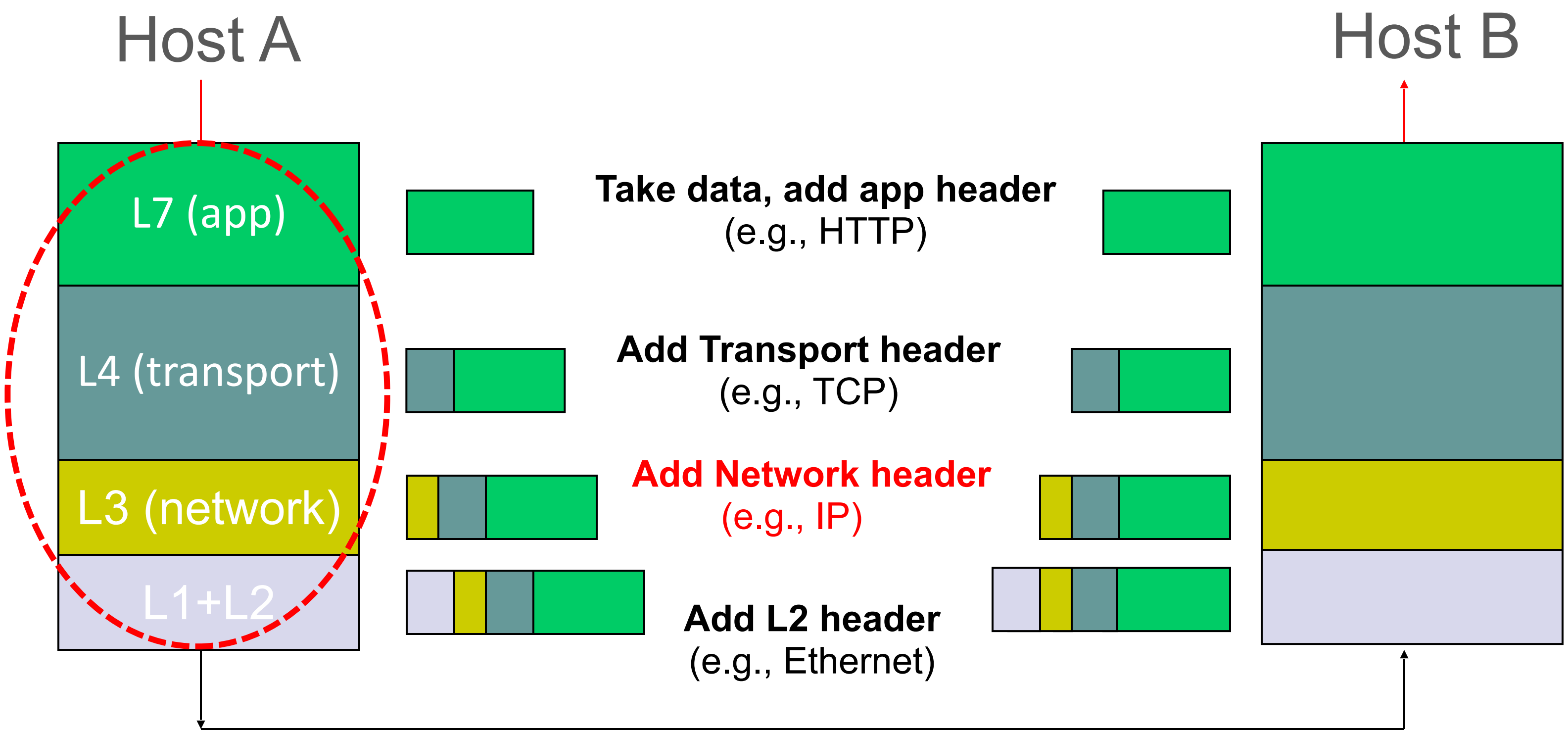
- **Syntax**: format of an IP packet
  - Nontrivial part: header
  - Rest is opaque payload

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

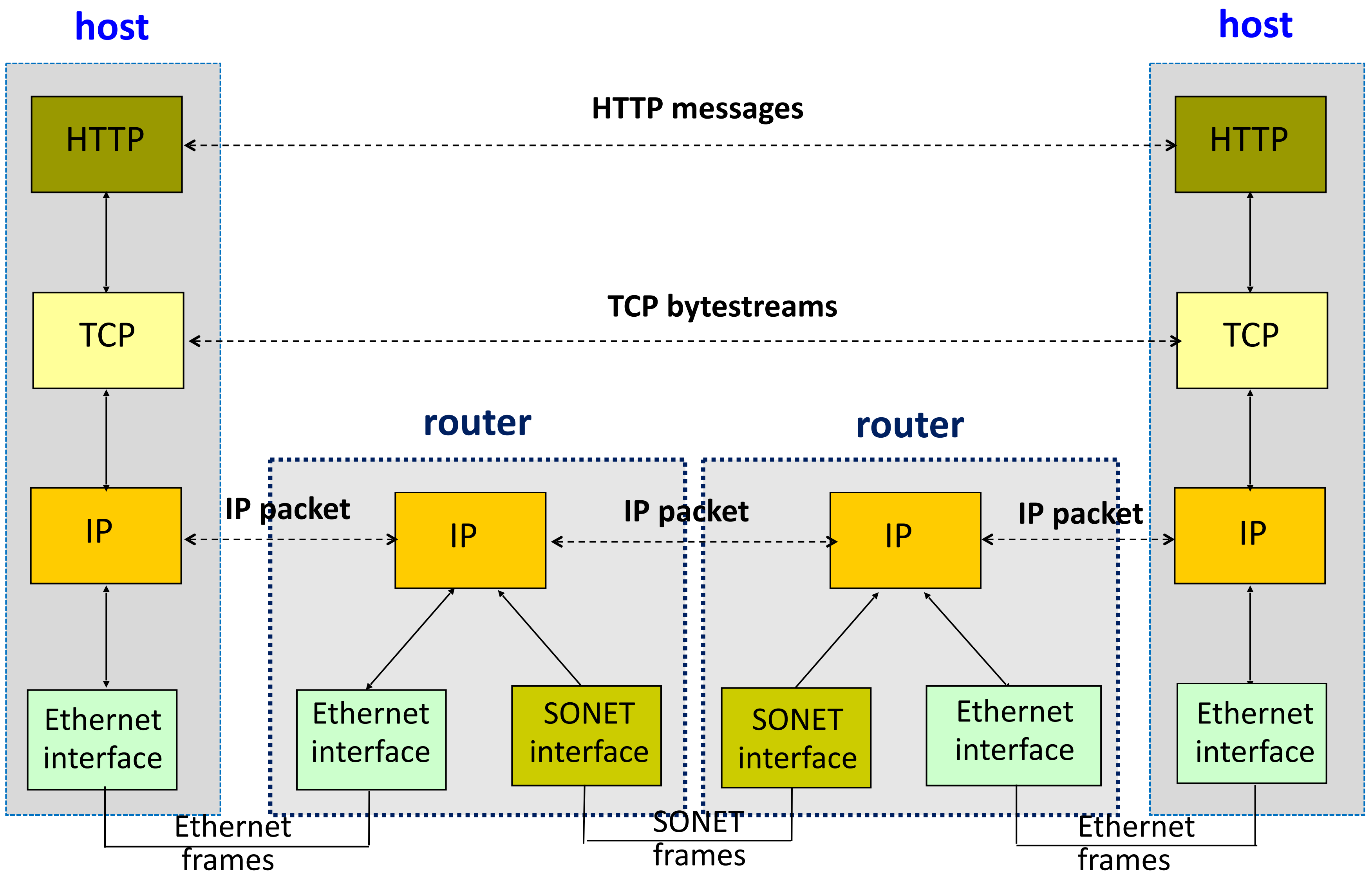
Recall: Layering



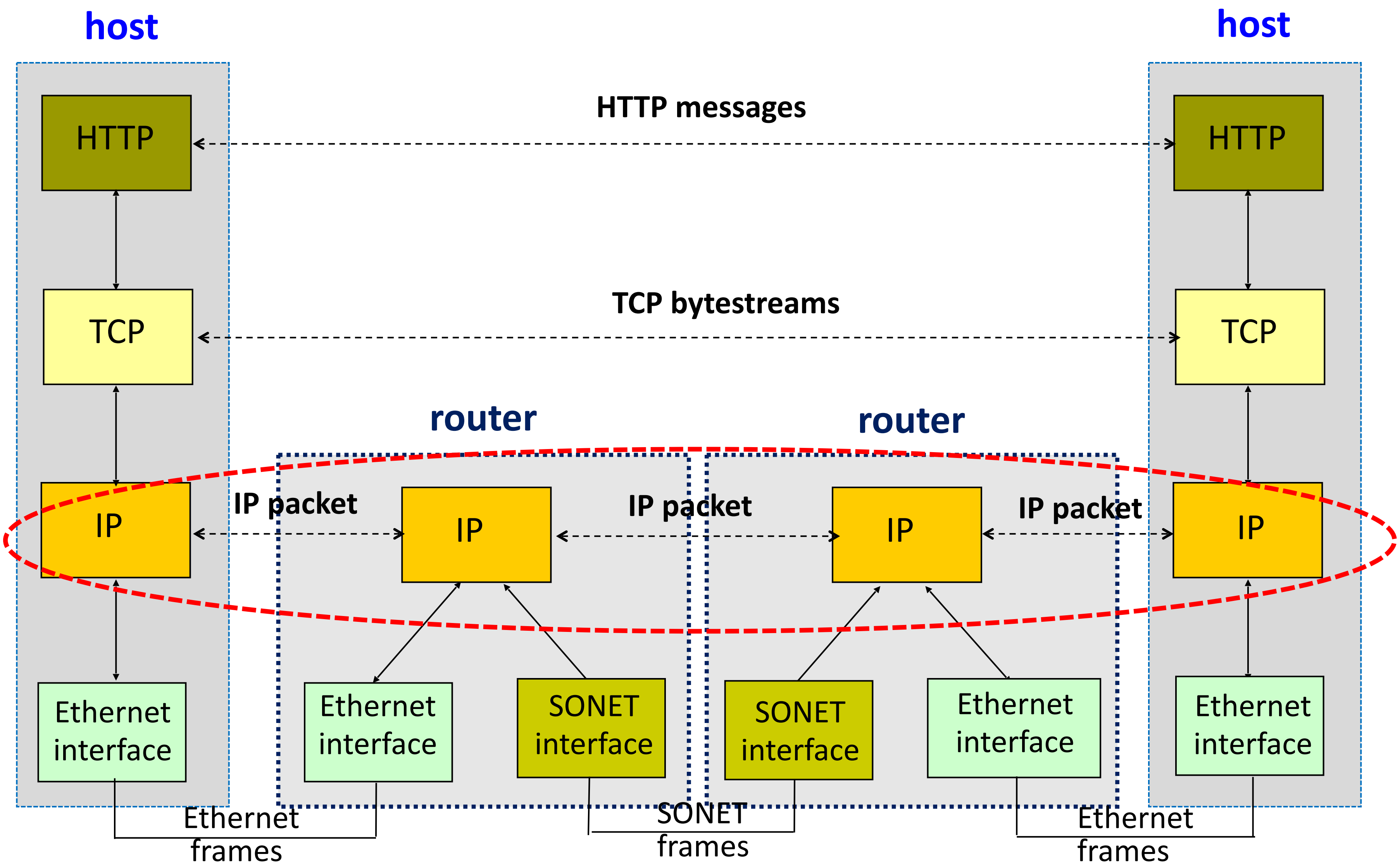
Recall: Layering



Recall: Hosts *vs.* Routers



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 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?

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)

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

- Parse packet *(router, dst host)*

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

- Parse packet *(router, dst host)*
- Forward packet to the L3 destination *(router)*

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)*

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)*

Next: what information do we need?

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)*

Next: what information do we need?

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)*

Next: what information do we need?

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

Parse Packet Correctly

- What version of IP?

Parse Packet Correctly

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

Parse Packet Correctly

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

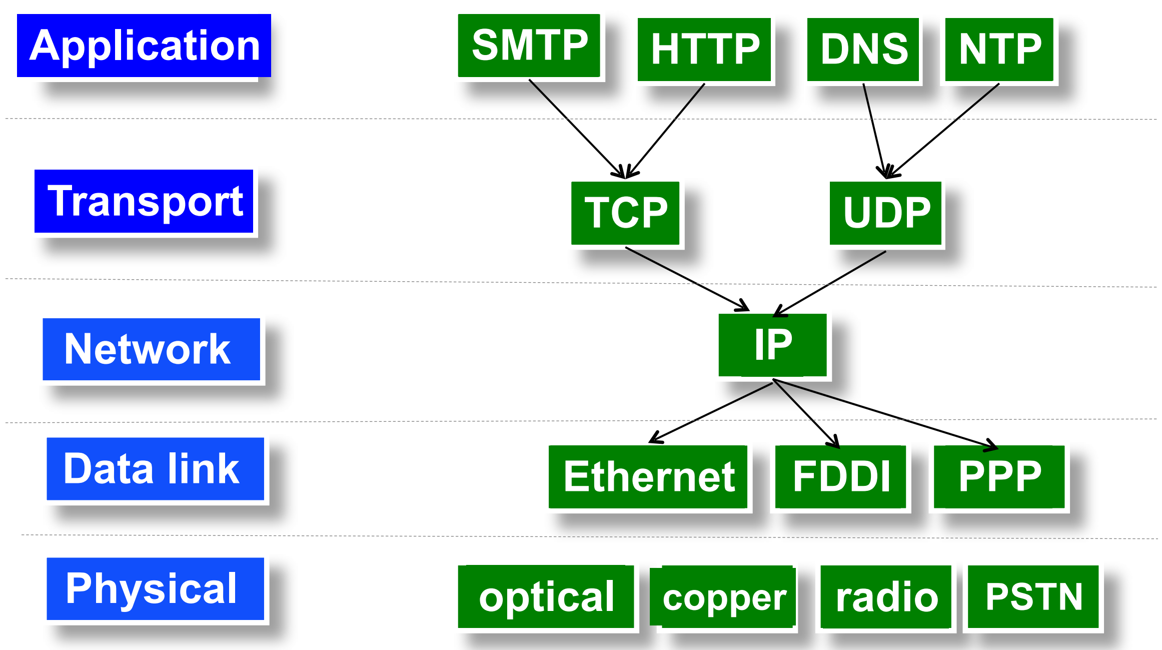
Deliver packet to the L3 destination

Deliver packet to the L3 destination

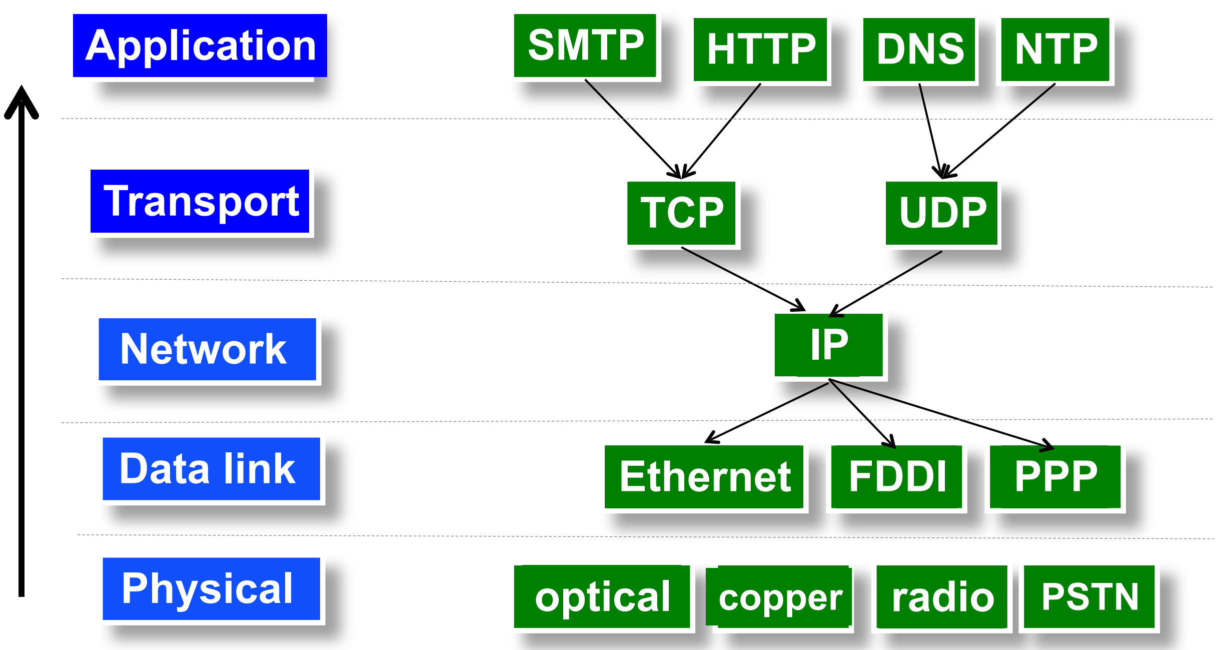
- Provide destination address (duh!)

Tell the destination how to handle packet

Tell the destination how to handle packet

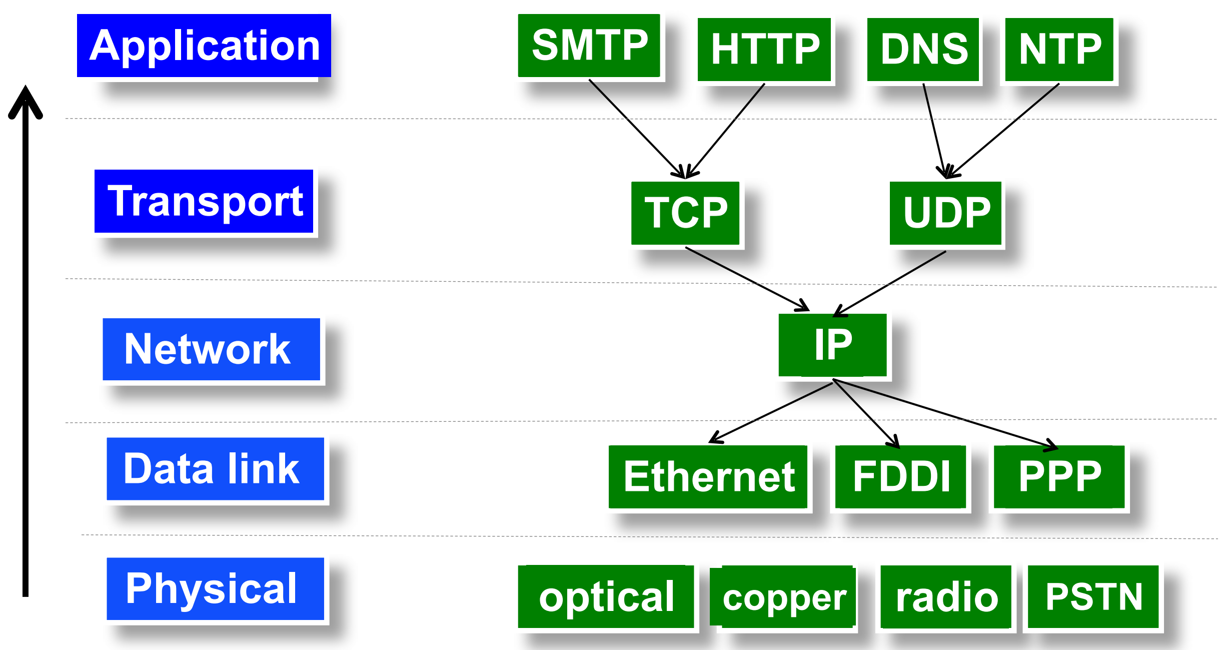


Tell the destination how to handle packet



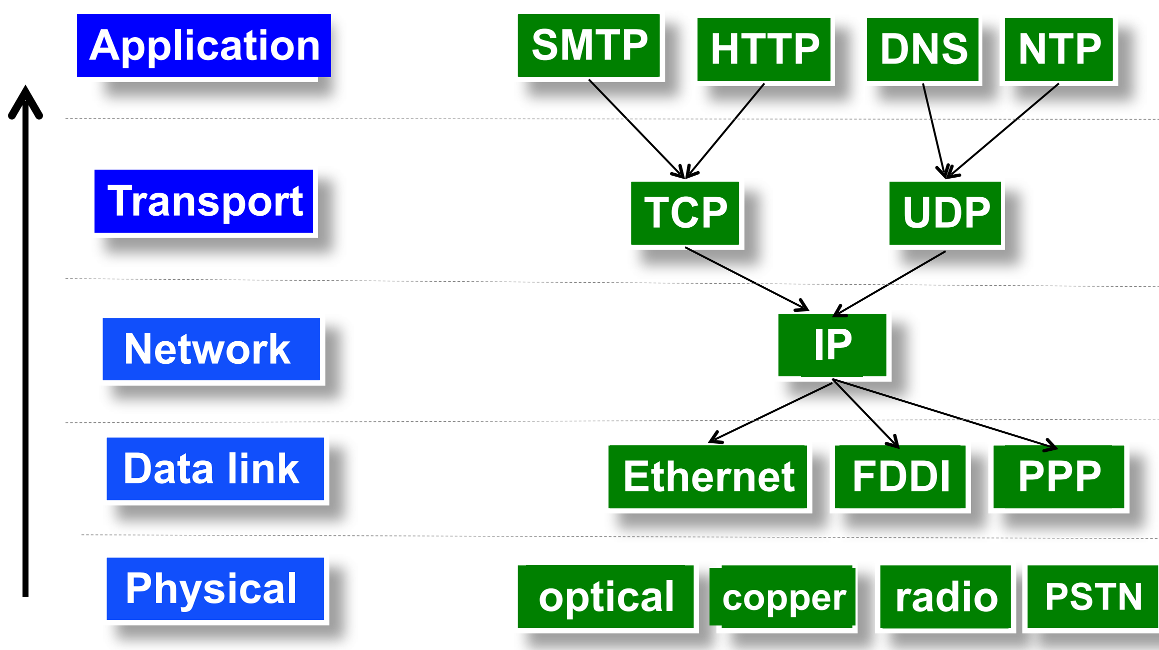
Tell the destination how to handle packet

- Indicate which protocol should handle packet next



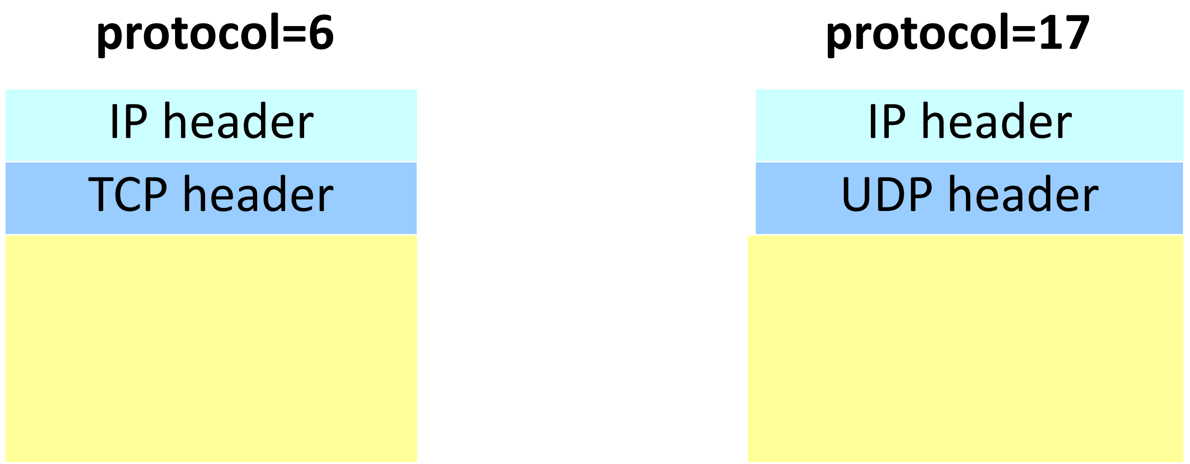
Tell the destination how to handle packet

- 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

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?

What problems?

- Loops

What problems?

- Loops
- Corruption

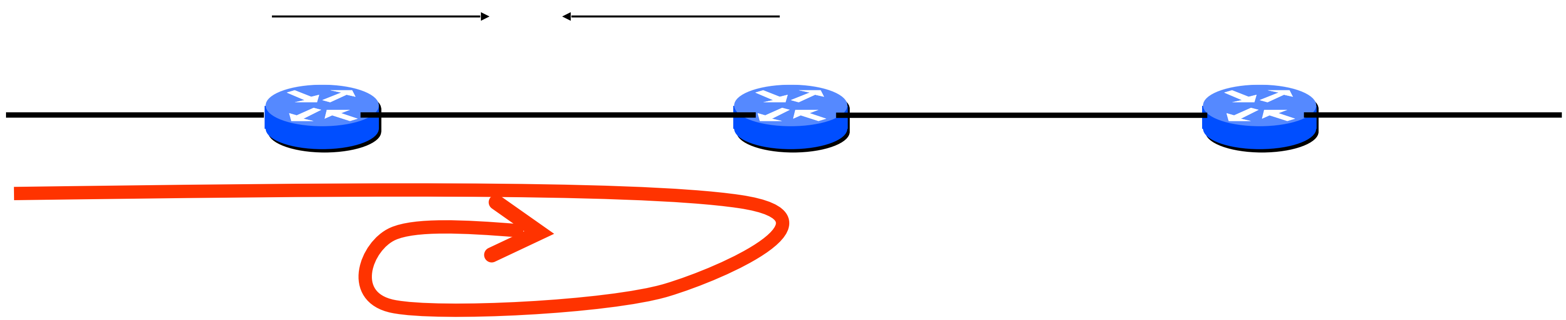
What problems?

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

Preventing Loops

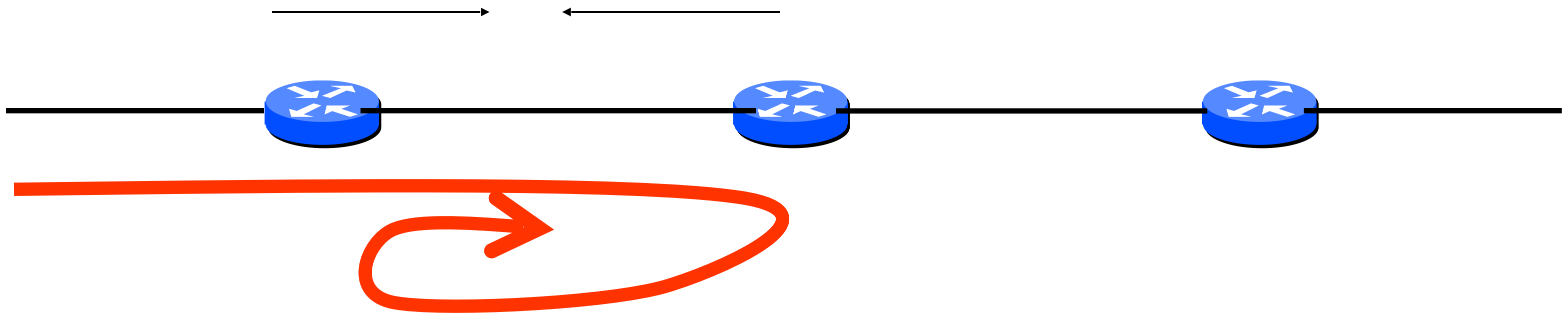
Preventing Loops

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



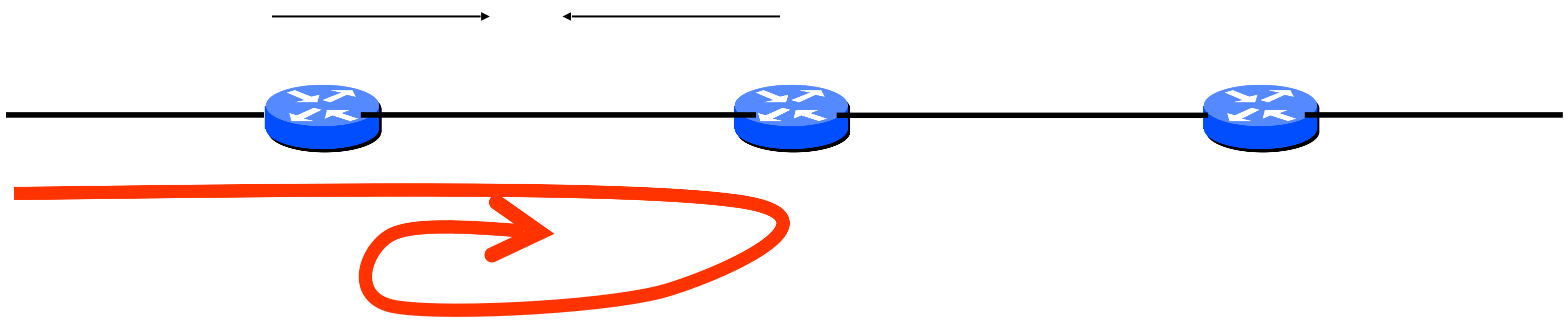
Preventing Loops

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



Preventing Loops

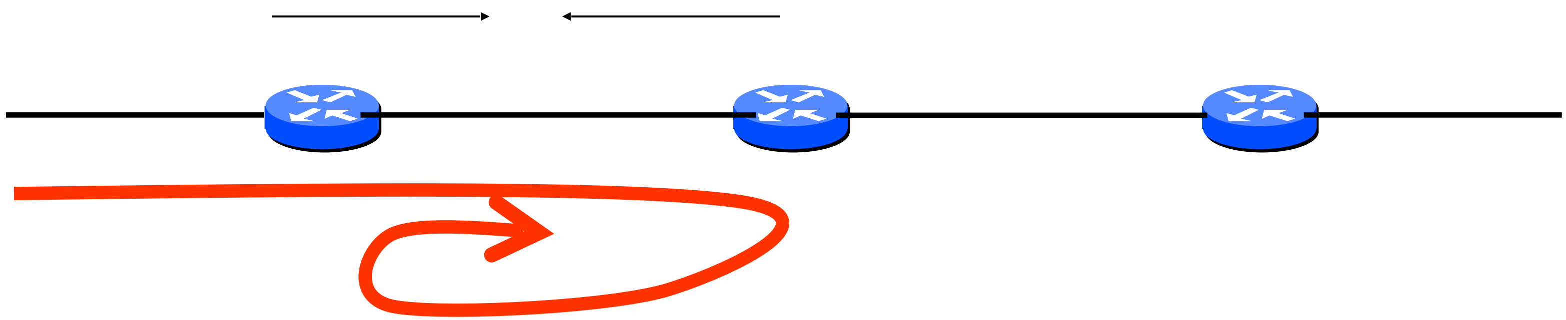
- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity



Means header must
include *source* IP address

Preventing Loops

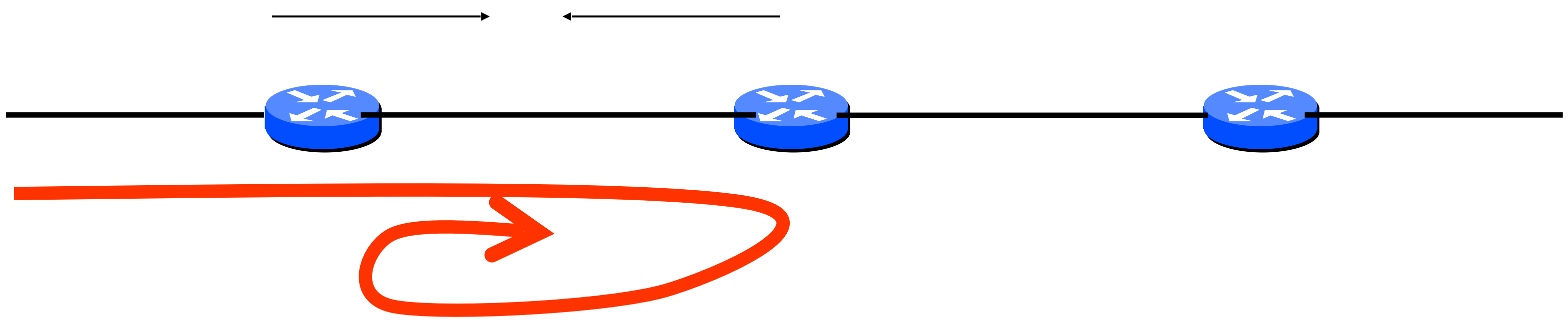
- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity



Means header must
include *source* IP address

Preventing Loops

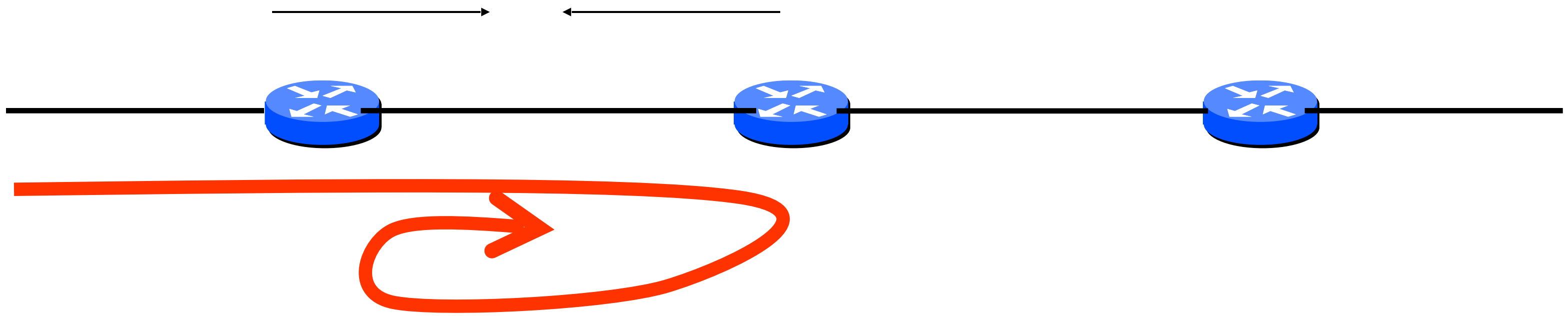
- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity



Means header must
include *source* IP address

Preventing Loops

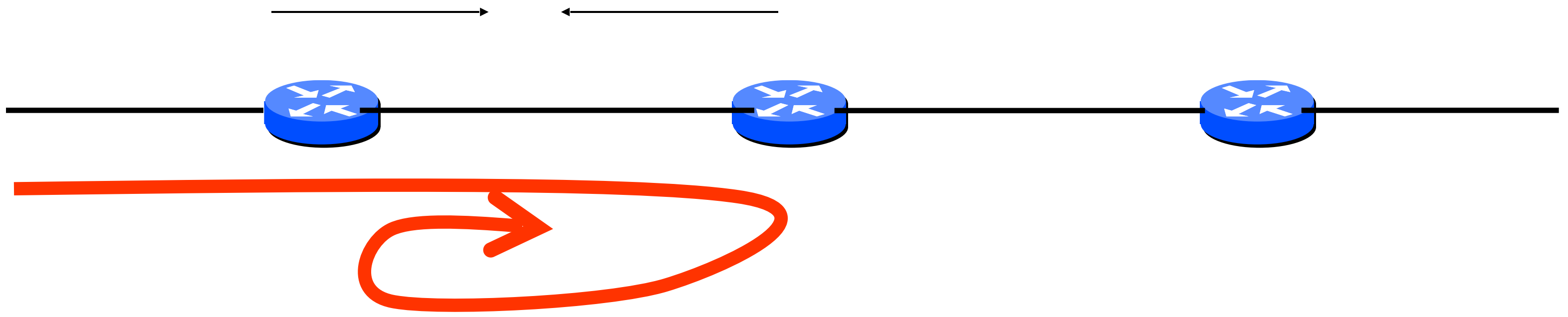
- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity



Means header must
include *source* IP address

Preventing Loops

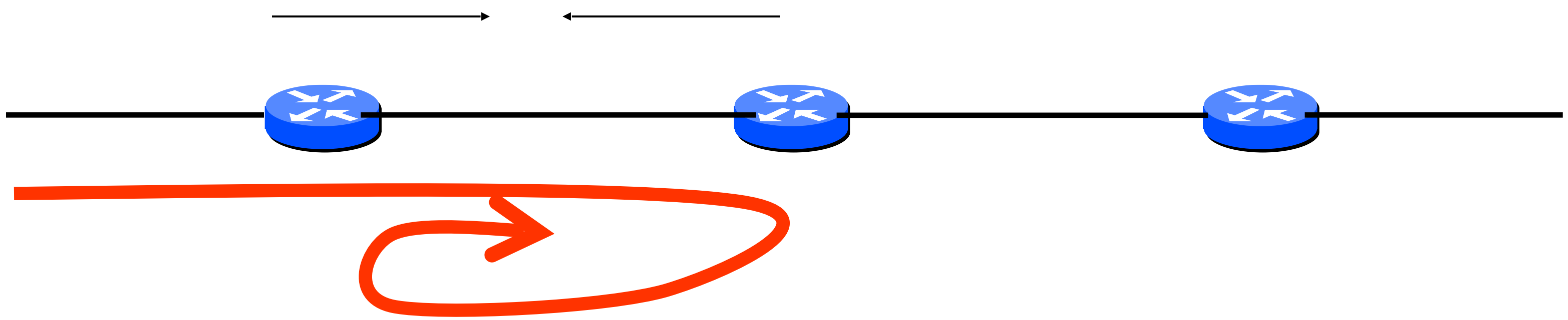
- Forwarding loops cause packets to cycle for a looong time
  - left unchecked would accumulate to consume all capacity



Means header must include *source* IP address

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

Header Corruption

- Checksum

Header Corruption

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

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

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

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?

Header Corruption

- Checksum
  - Small #bits used to check integrity of some data (e.g., hash)
  - Particular form of checksum over packet header
- If not correct, router/destination discards packets
  - So it doesn't act on bogus information
- Checksum updated at every router
  - Why?
  - Why include TTL?

Header Corruption

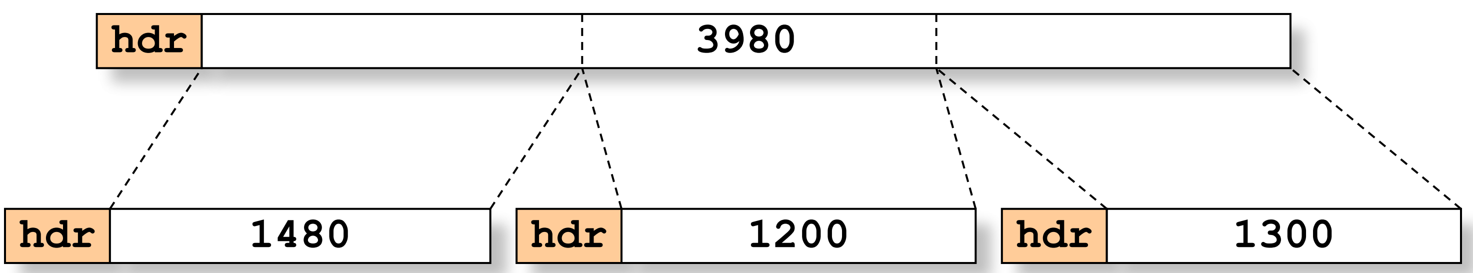
- Checksum
  - Small #bits used to check integrity of some data (e.g., hash)
  - Particular form of checksum <u>over packet header</u>
- If not correct, router/destination discards packets
  - So it doesn’t act on bogus information
- Checksum updated at every router
  - Why?
  - Why include TTL?
  - Why only header?

Fragmentation

- Every link has a "Maximum Transmission Unit" (MTU)
  - largest number of bits it can carry as one unit

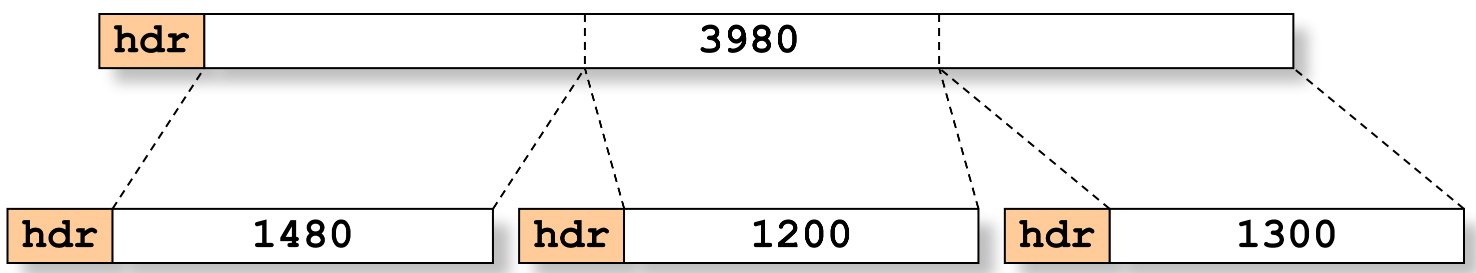
Fragmentation

- 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



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

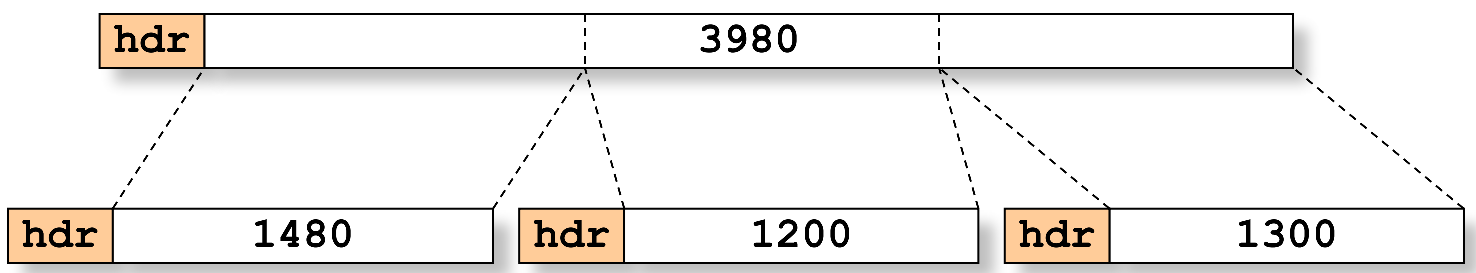


- Must reassemble to recover original packet

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?

What forms of special treatment?

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

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)

“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

“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

Options

- **Optional directives to the network**
- **Examples**
  - Record Route, Source Route, Timestamp, ...

Options

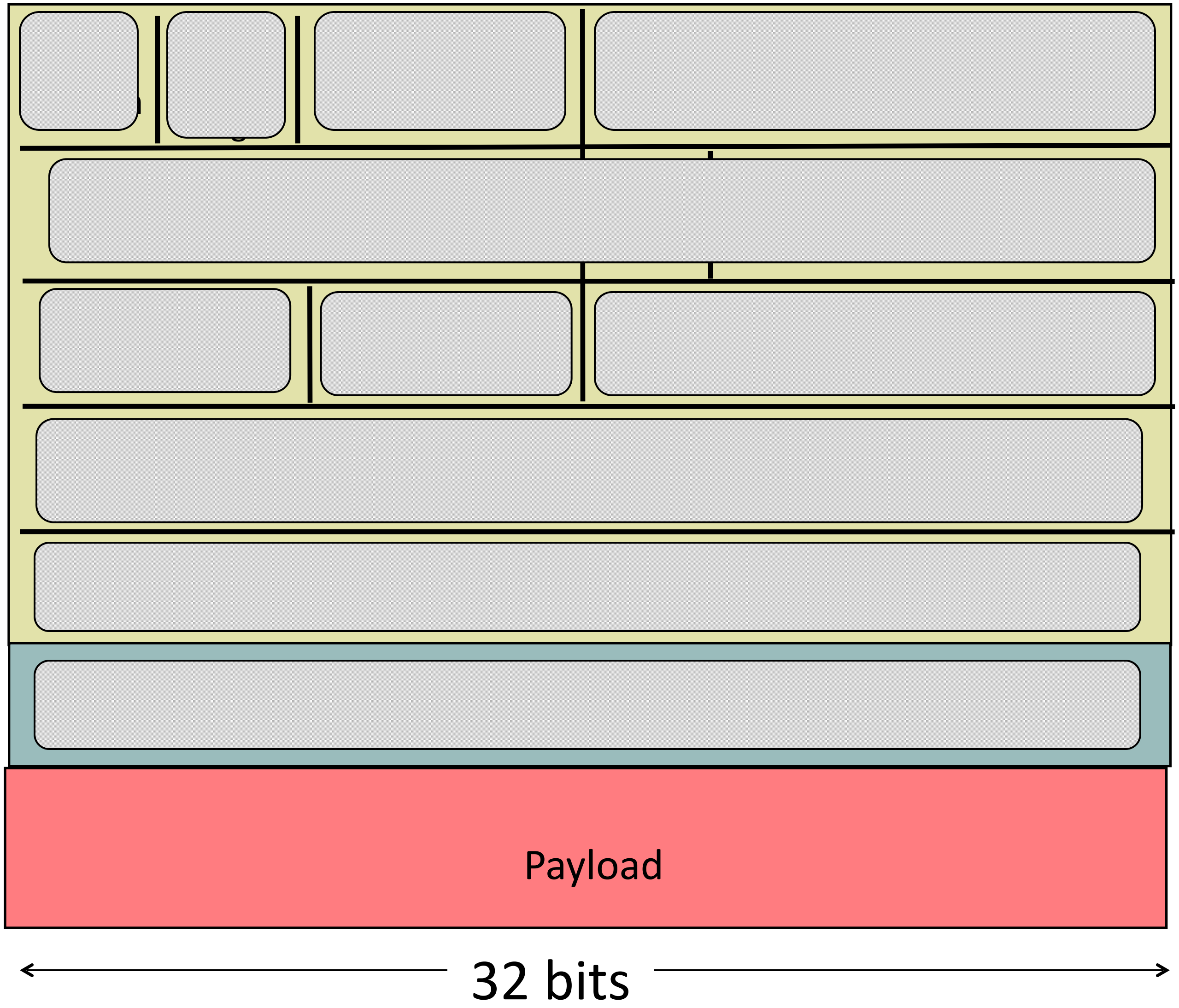
- **Optional directives to the network**
- **Examples**
  - Record Route, Source Route, Timestamp, ...
- More complex implementation
  - Leads to variable length headers
  - Often leads to higher processing overheads

Where are we …

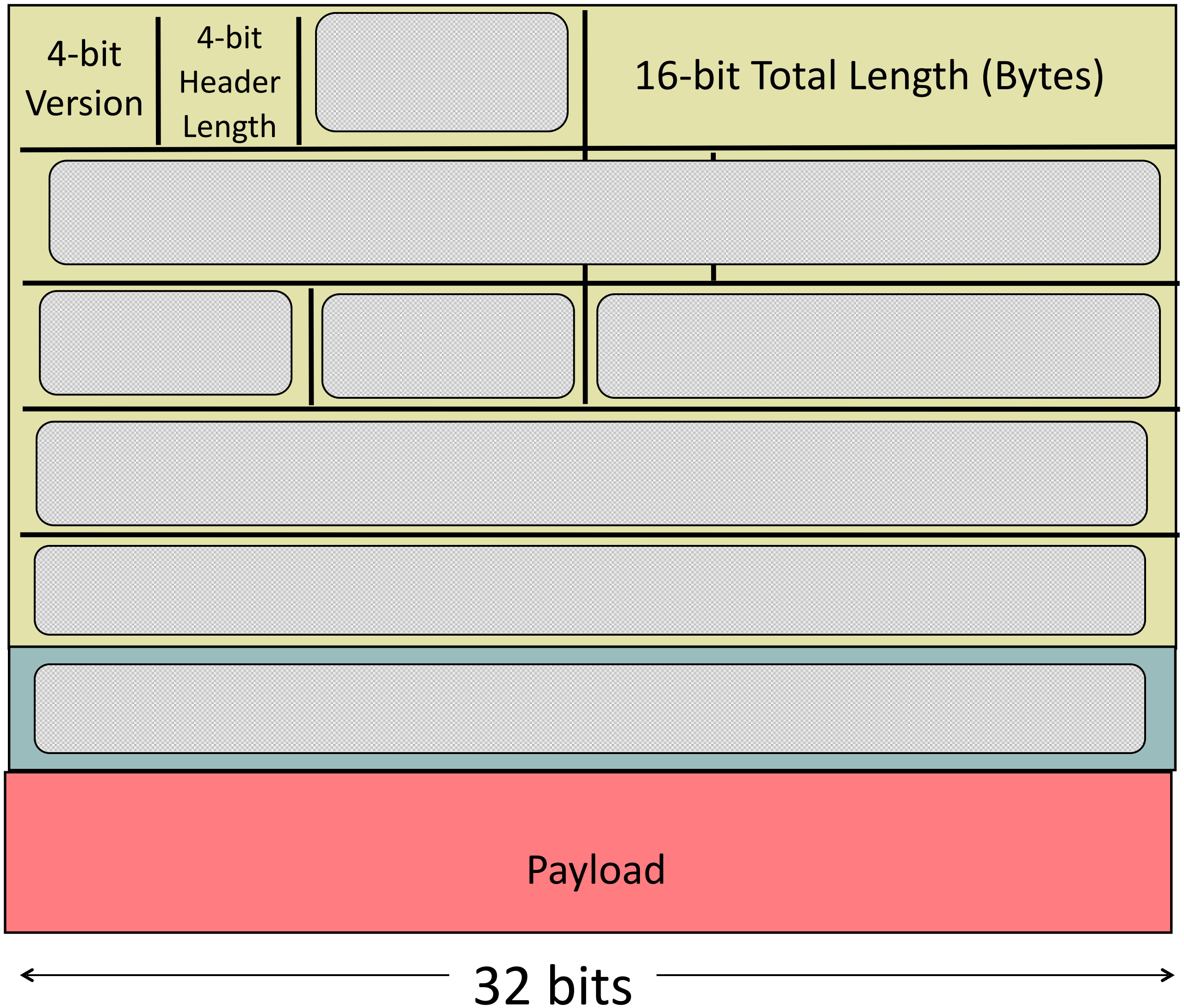
- Parse packet → *version, header length, packet length*
- Forward packet to the L3 dst → *destination address*
- Tell destination what to do next → *protocol field*
- Get responses back to the source → *source address*

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

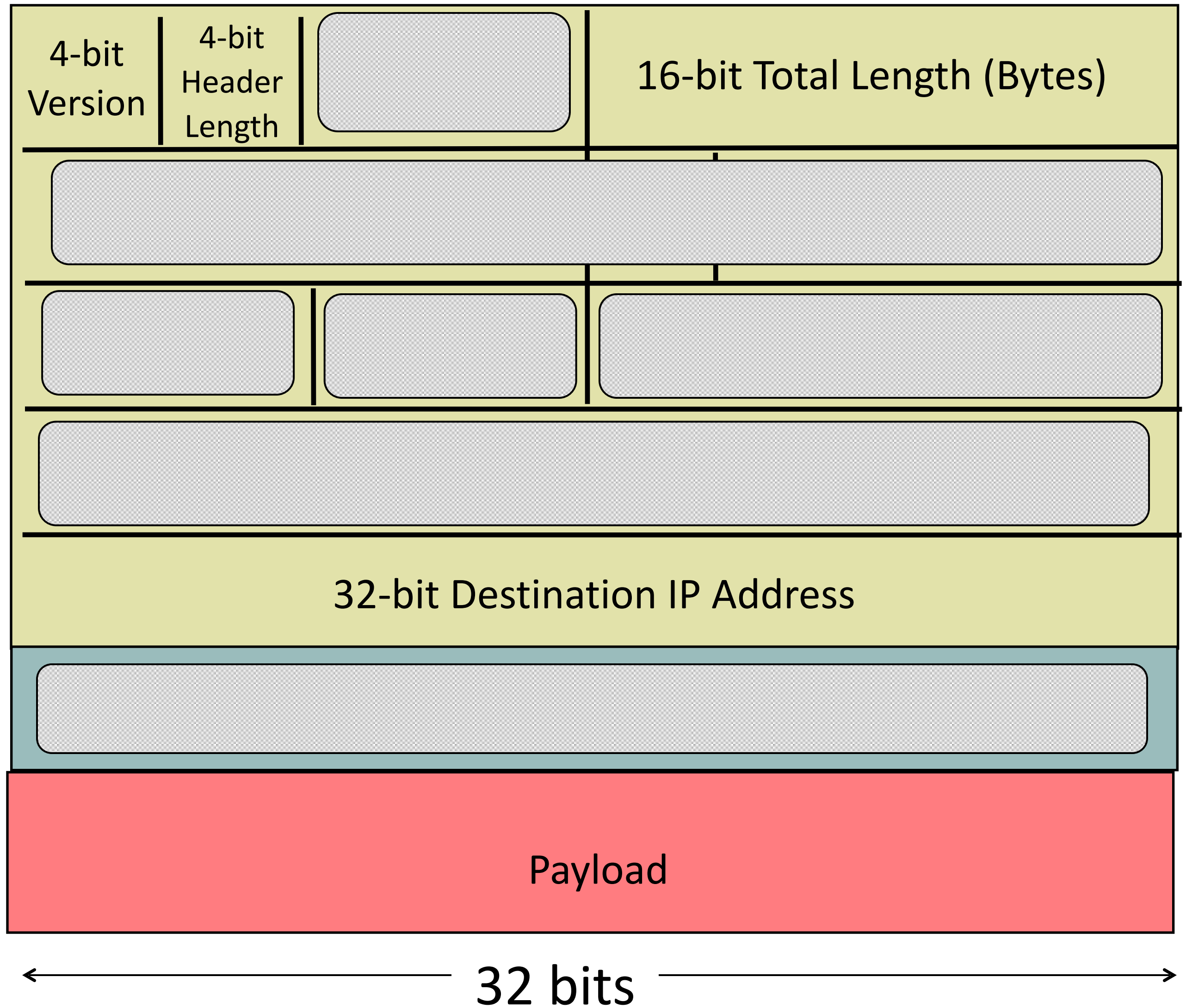
IP Packet Structure



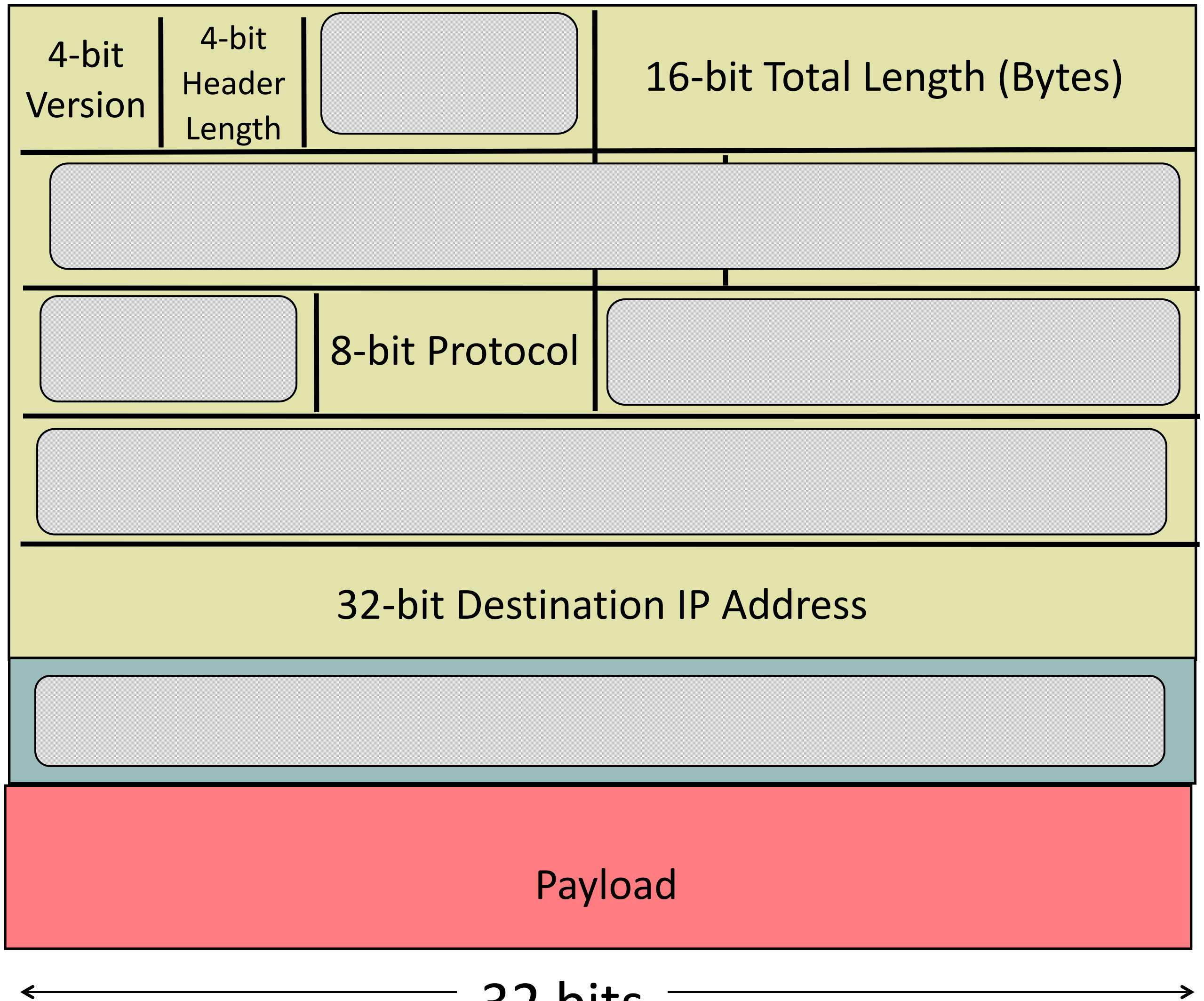
IP Packet Structure



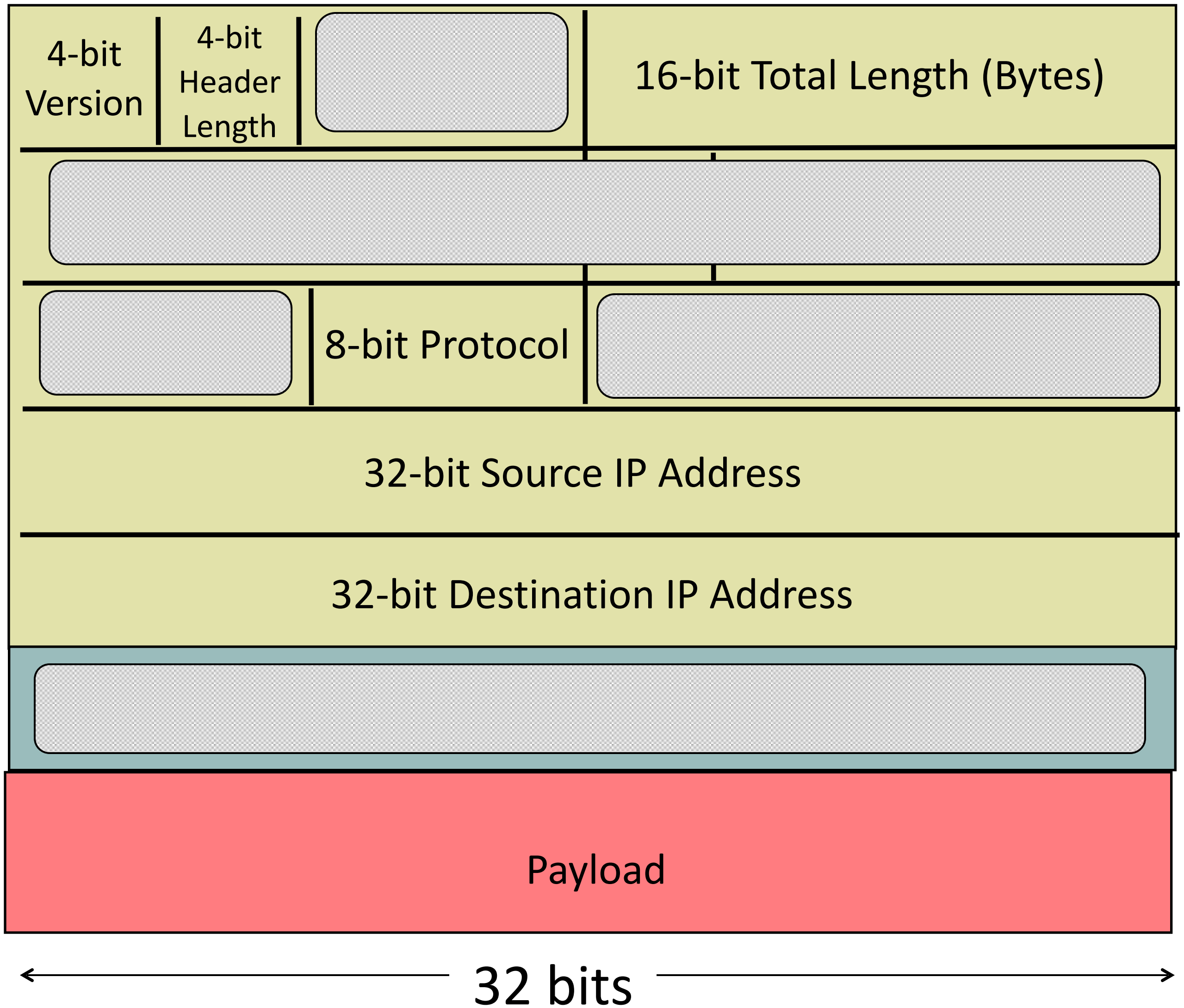
IP Packet Structure



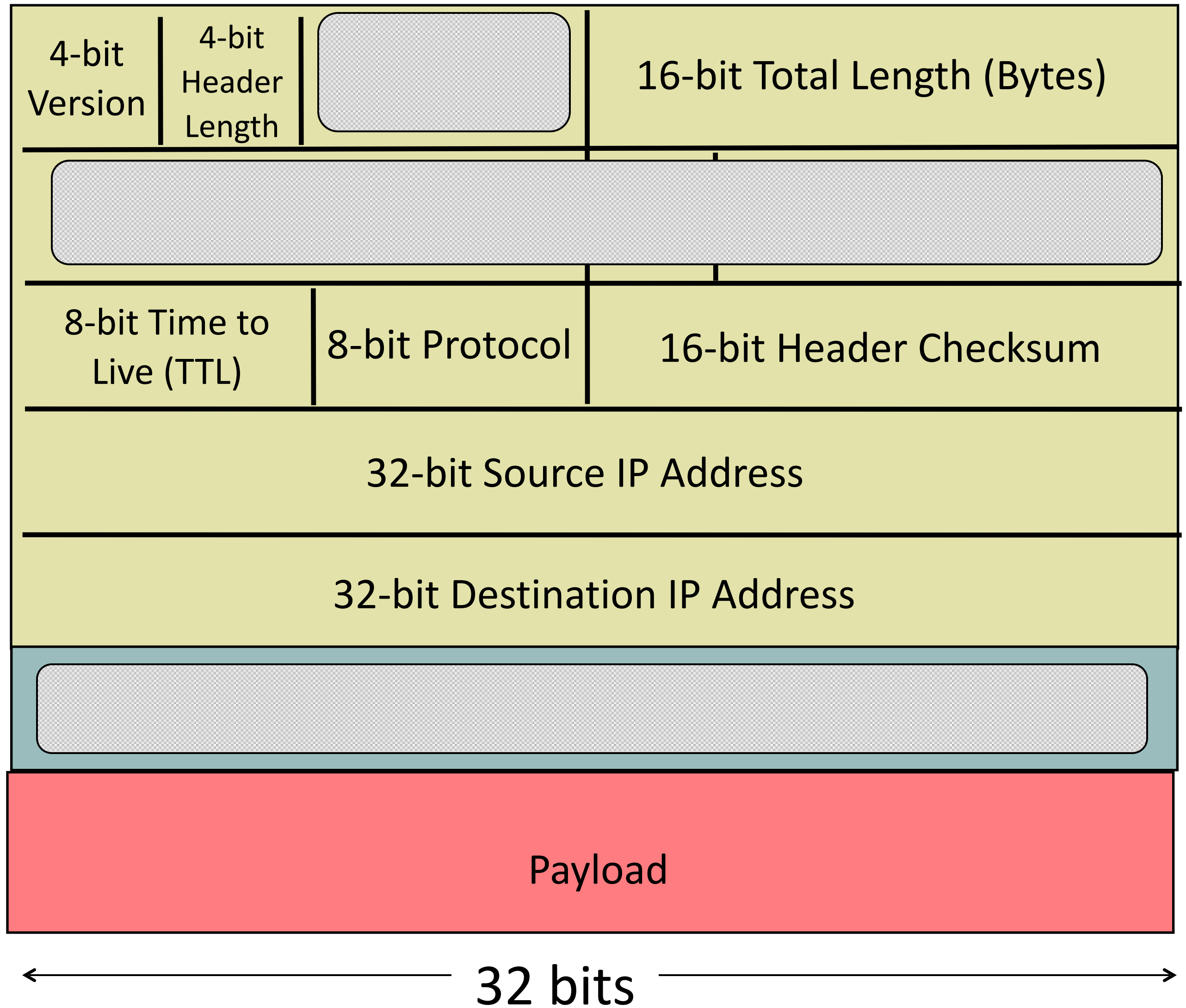
IP Packet Structure



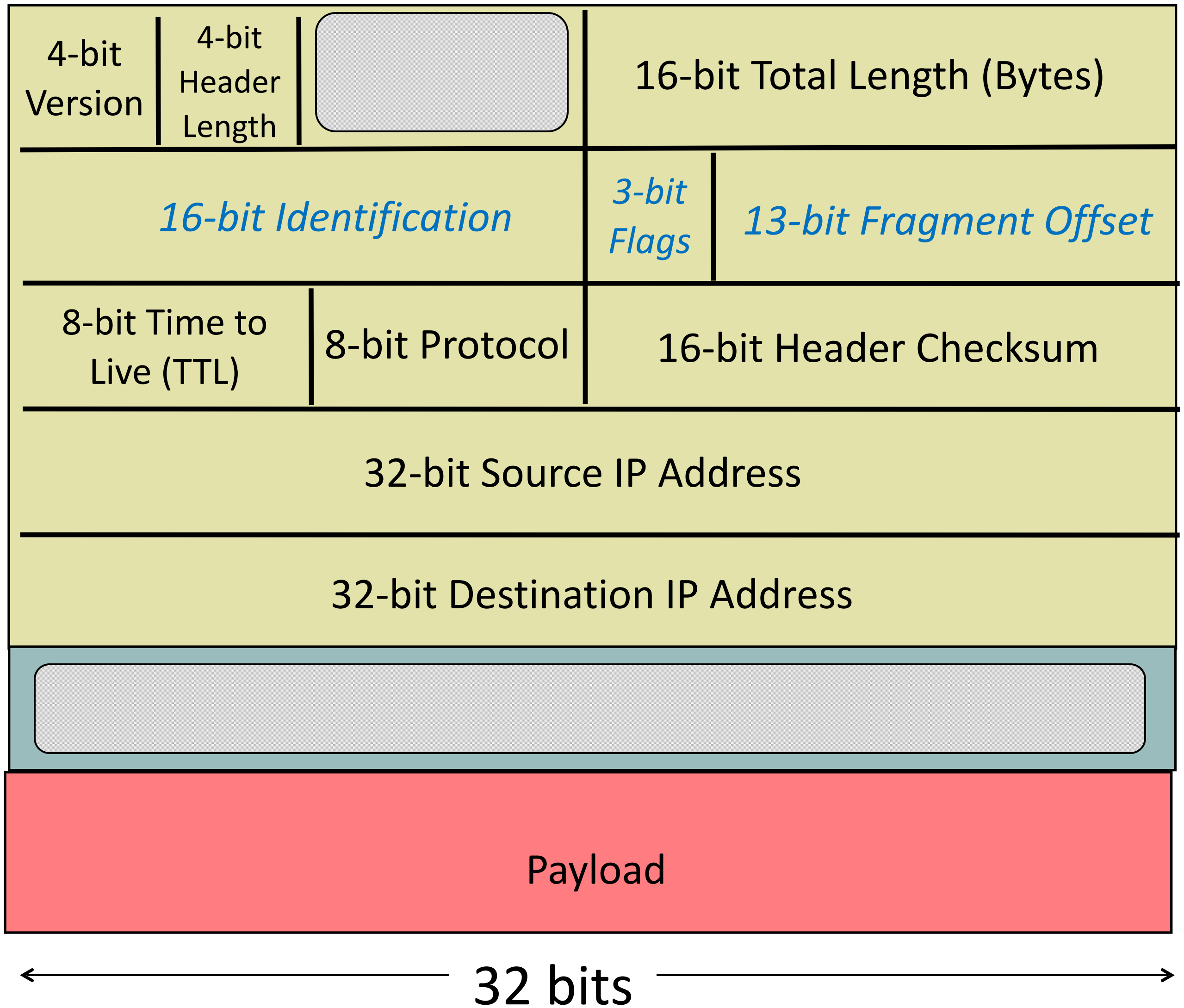
IP Packet Structure



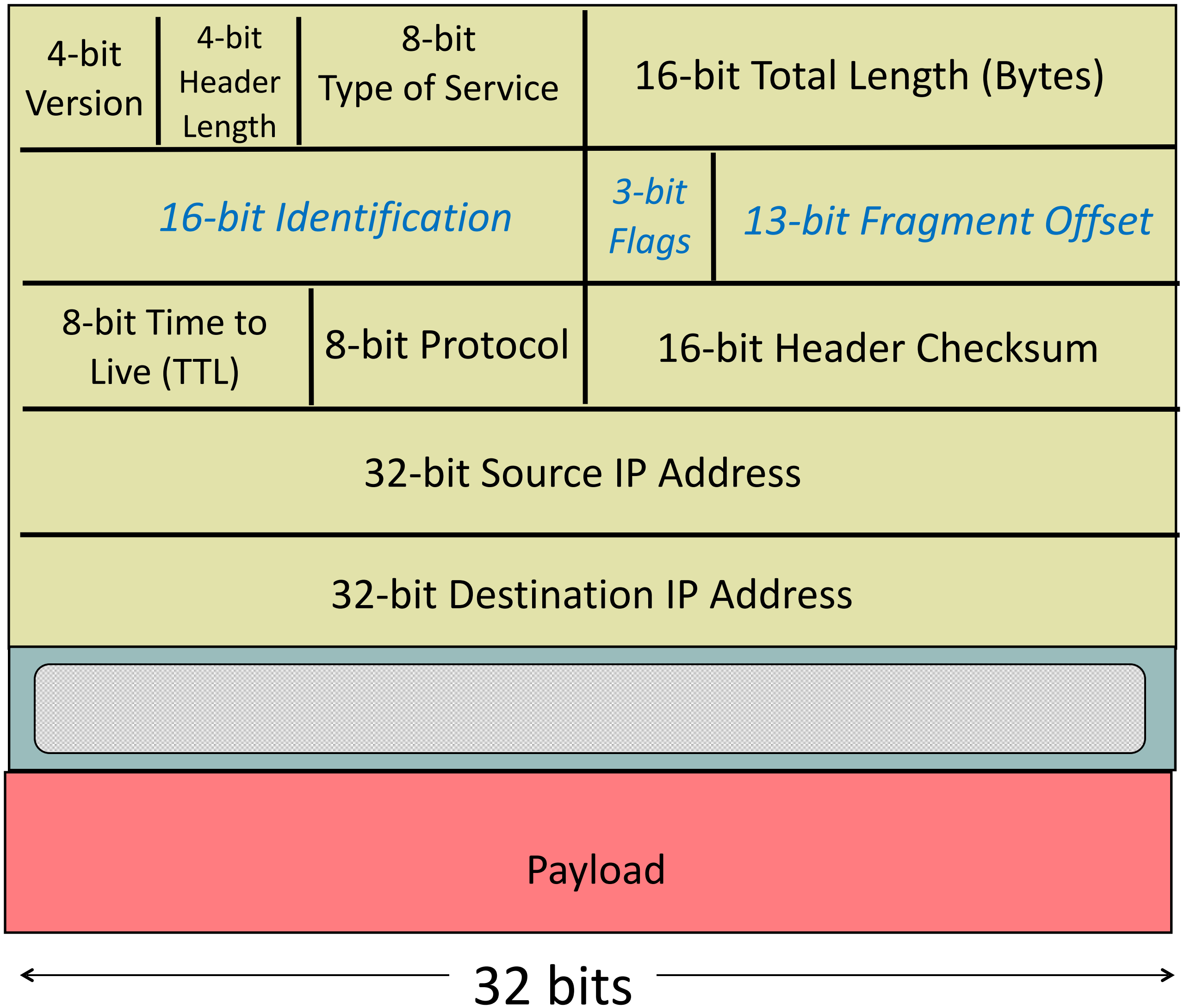
IP Packet Structure



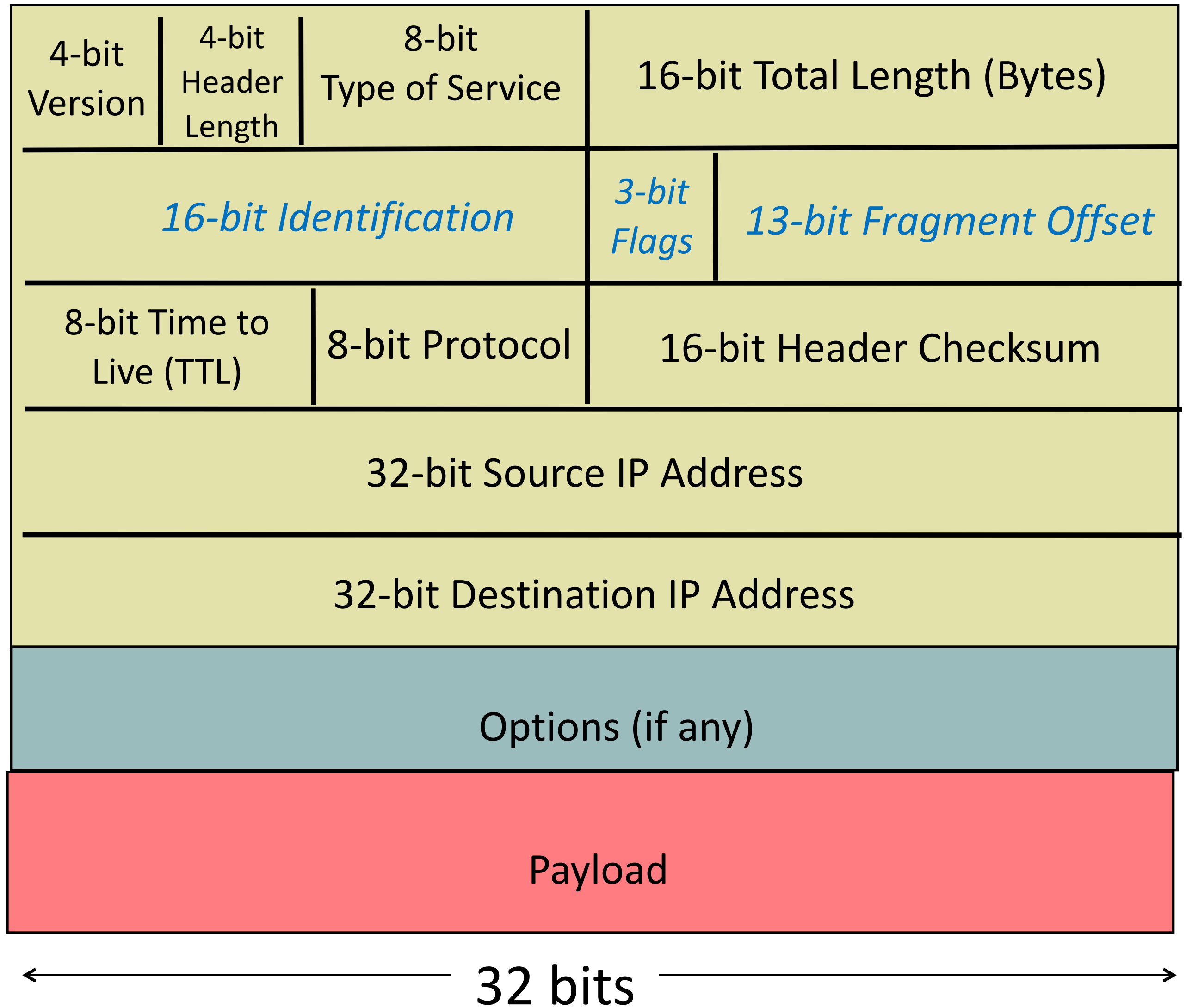
IP Packet Structure



IP Packet Structure



IP Packet Structure



Two remaining topics (next time)

- IPv4 → IPv6
- Security implications of the IP header

Questions?