Midterm Review

CPSC 433/533, Spring 2021

Anurag Khandelwal

Logistics

Logistics

- **Test will be during class hours, starting at 1.00pm, ending at 1:15pm (a total of 75 minutes)**
  - Will be Zoom proctored: join via this zoom link, and keep your videos on!
  - We will keep track of Zoom participants and compare to submissions.
  - Please aim to join the Zoom meeting 5 minutes early!

Logistics

- **Test will be during class hours, starting at 1.00pm, ending at 1:15pm (a total of 75 minutes)**
  - Will be Zoom proctored: join via this zoom link, and keep your videos on!
  - We will keep track of Zoom participants and compare to submissions.
  - Please aim to join the Zoom meeting 5 minutes early!
- **Closed book, closed notes, etc.**

Logistics

- **Test will be during class hours, starting at 1.00pm, ending at 1:15pm (a total of 75 minutes)**
  - Will be Zoom proctored: join via this zoom link, and keep your videos on!
  - We will keep track of Zoom participants and compare to submissions.
  - Please aim to join the Zoom meeting 5 minutes early!
- **Closed book, closed notes, etc.**
- **You can prepare a 1 page cheat-sheet, handwritten notes only**
  - No calculators (test does not require any complicated calculations)

General Guidelines (1)

General Guidelines (1)

- **Test only assumes material covered in lecture**
  - Read text to clarify details and context for the above

General Guidelines (1)

- **Test only assumes material covered in lecture**
  - Read text to clarify details and context for the above

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

General Guidelines (1)

- **Test only assumes material covered in lecture**
  - Read text to clarify details and context for the above

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

- **You do need to understand *how* and *why* things work!**
  - Understand pros/cons, when a solution is applicable/useful/useless, etc.

General Guidelines (1)

- **Test only assumes material covered in lecture**
  - Read text to clarify details and context for the above

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

- **You do need to understand how and why things work!**
  - Understand pros/cons, when a solution is applicable/useful/useless, etc.

- **You do need to understand enough to design new protocols**
  - Use the expertise you have gained with my incessant questioning during class!

General Guidelines (2)

General Guidelines (2)

- **Exam format:**
  - Two parts: 20 points + 40 points

General Guidelines (2)

- **Exam format:**
  - Two parts: 20 points + 40 points

- **First part: Multiple choice questions**
  - Mixed bag, most are easy, some trickier than you think!

General Guidelines (2)

- **Exam format:**
  - Two parts: 20 points + 40 points
- **First part: Multiple choice questions**
  - Mixed bag, most are easy, some trickier than you think!
- **Second Part: Longer questions (how & why things work)**
  - "I was having a nice day, minding my own business, and then BGP went nuts…"

General Guidelines (2)

- **Exam format:**
  - Two parts: 20 points + 40 points
- **First part: Multiple choice questions**
  - Mixed bag, most are easy, some trickier than you think!
- **Second Part: Longer questions (how & why things work)**
  - "I was having a nice day, minding my own business, and then BGP went nuts…"
- Cheating: Don't do it
  - You will be required to sign a honor code at the beginning of the exam
  - Questions are conceptual — require thinking, will take time
  - Don't waste it trying to cheat; we have ways of finding out

This Review

This Review

- **Walk through what you are expected to know**
  - Key topics, important aspects of each

This Review

- **Walk through what you are expected to know**
  - Key topics, important aspects of each

- **Just because I didn't cover it in review doesn't mean you don't need to know it!**
  - But if I covered it today, you definitely need to know it

This Review

- **Walk through what you are expected to know**
  - Key topics, important aspects of each

- **Just because I didn't cover it in review doesn't mean you don't need to know it!**
  - But if I covered it today, you definitely need to know it

- **My plan: summarize, not explain**
  - Stop me when you want to discuss something further!

Topics

- **Basic concepts (Lecture 1, 2)**
- **Architecture & Principles (Lecture 3, 4)**
- **Network Layer (Lectures 4-10)**
  - Concepts: valid routing state, convergence, least-cost paths
  - Overall context (inter- and intra-domain routing)
  - Computing least-cost routes (DV, LS)
  - IP addressing
  - Inter-domain routing
  - Router architecture
- **Transport (Lectures 10-14)**
  - Role of the transport layer
  - UDP vs. TCP
  - TCP basics: reliability
  - TCP flow control
  - TCP congestion control
  - TCP critiques & router assistance

Basic Concepts

- **You should know:**
  - Statistical multiplexing
  - Packet vs. circuit switching
  - Link characteristics
  - Packet delays

How are network resources shared?

Two approaches

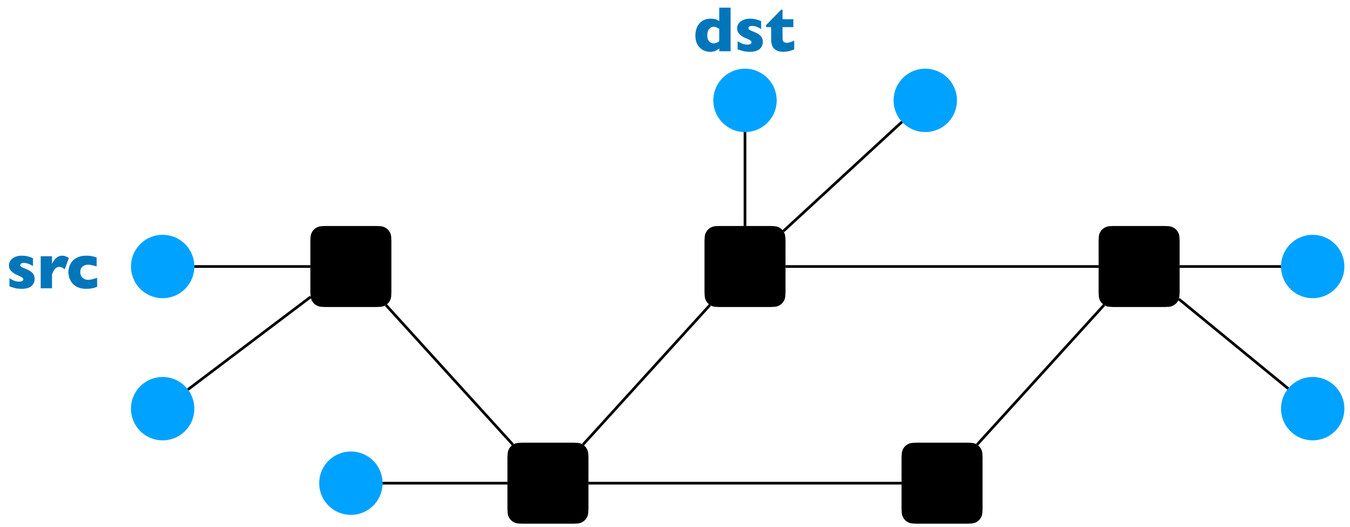
- Reservations → circuit switching
- On demand → packet switching

Two approaches to sharing

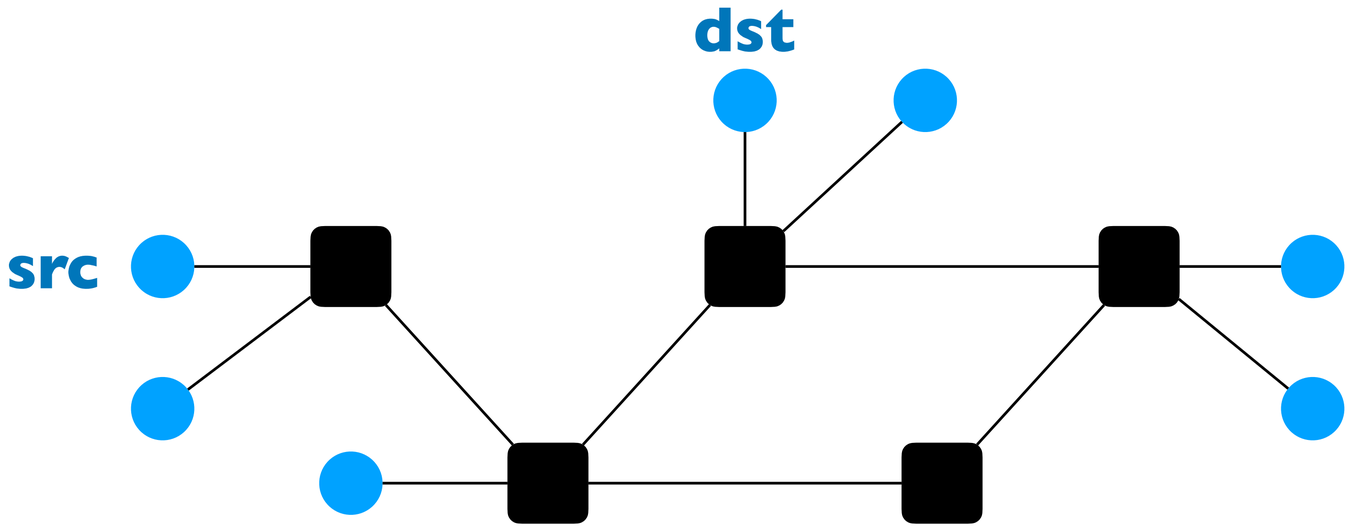
- **Packet switching**
  - Network resources consumed on demand per-packet
  - "Admission control": per packet

- **Circuit switching**
  - Network resources reserved a priori at "connection" initiation
  - "Admission control": per connection

Circuit Switching

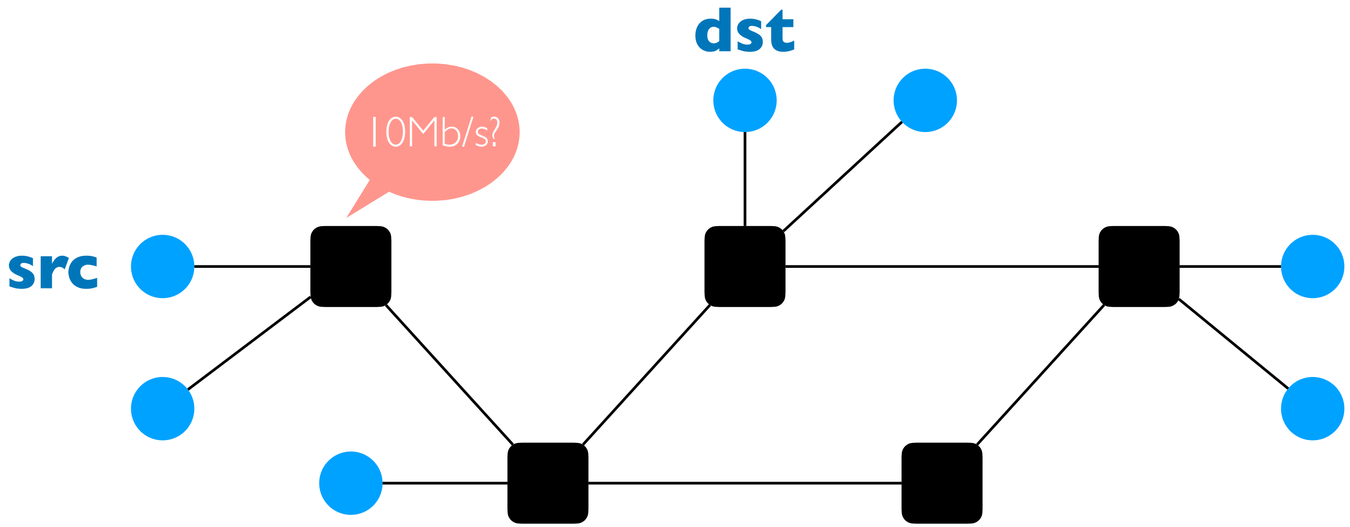


Circuit Switching



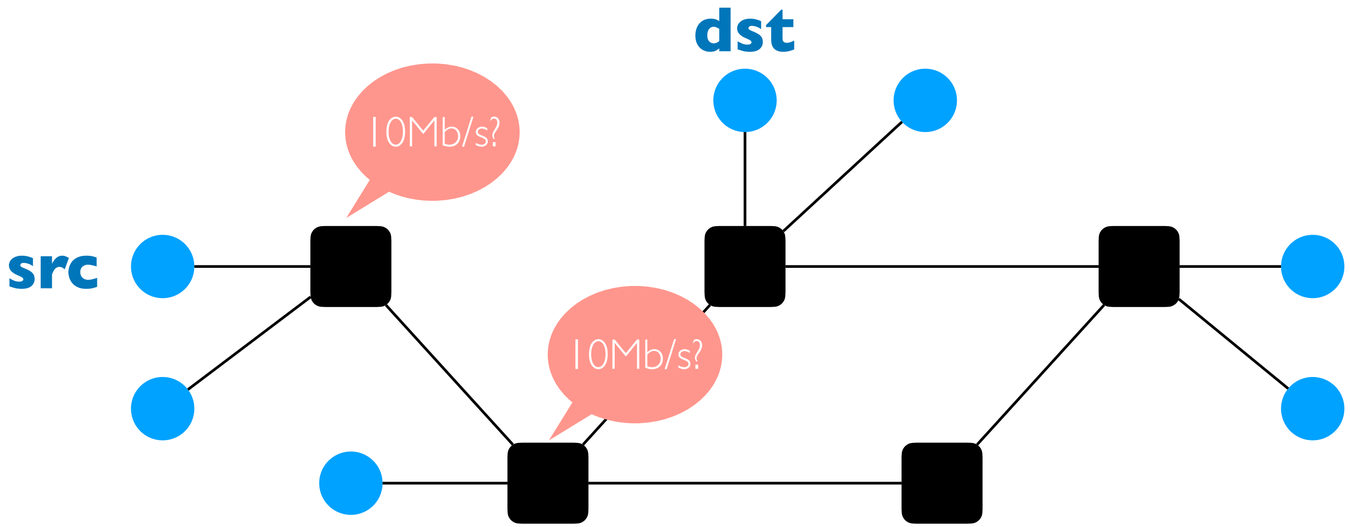
1. The **src** sends a reservation request to **dst**

Circuit Switching



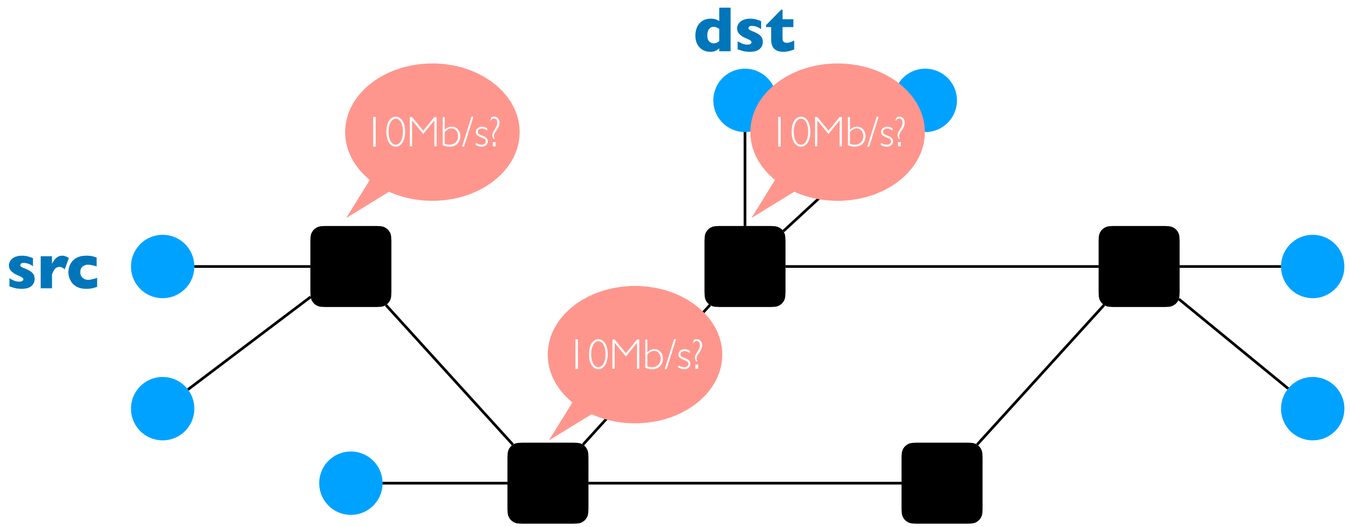
1. The **src** sends a reservation request to **dst**

Circuit Switching



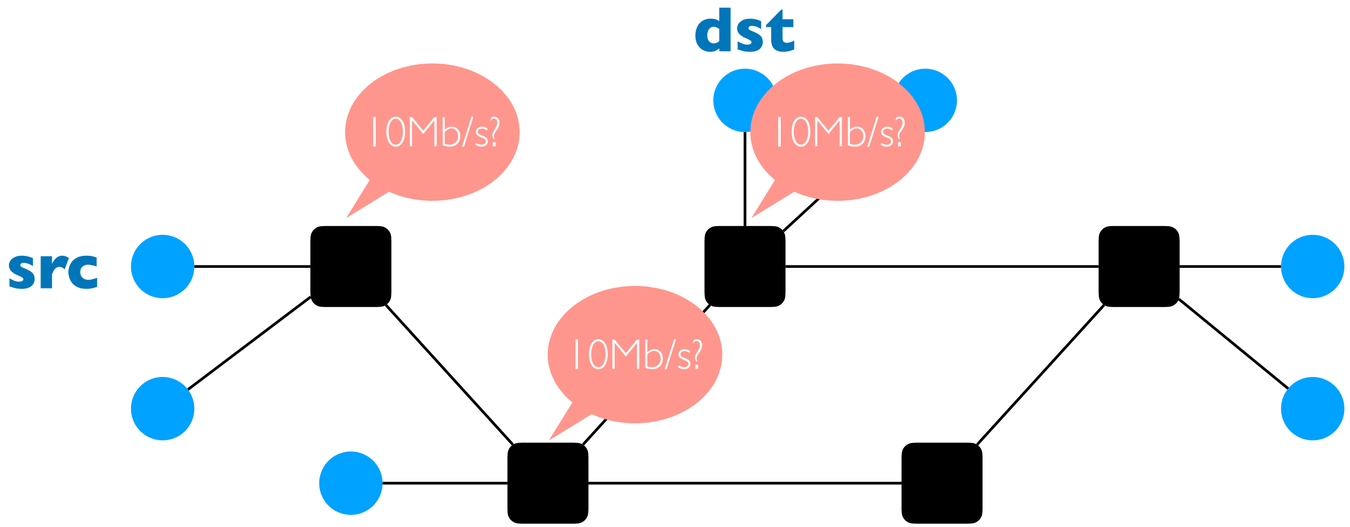
1. The **src** sends a reservation request to **dst**

Circuit Switching



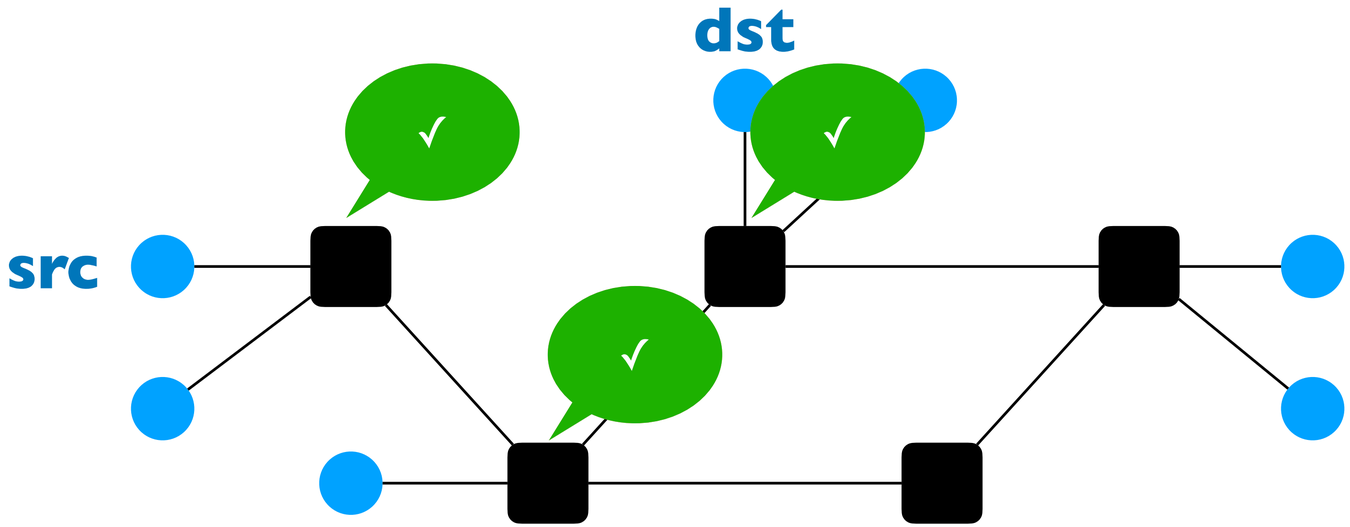
1. The **src** sends a reservation request to **dst**

Circuit Switching



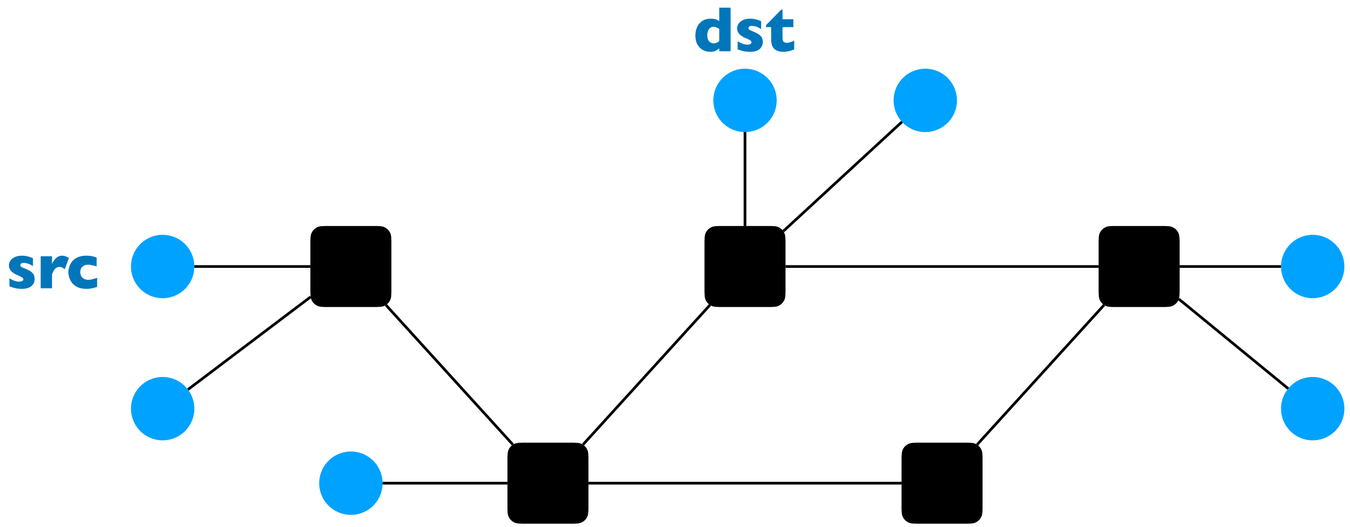
1. The **src** sends a reservation request to **dst**
2. Switches "establish a circuit"

Circuit Switching



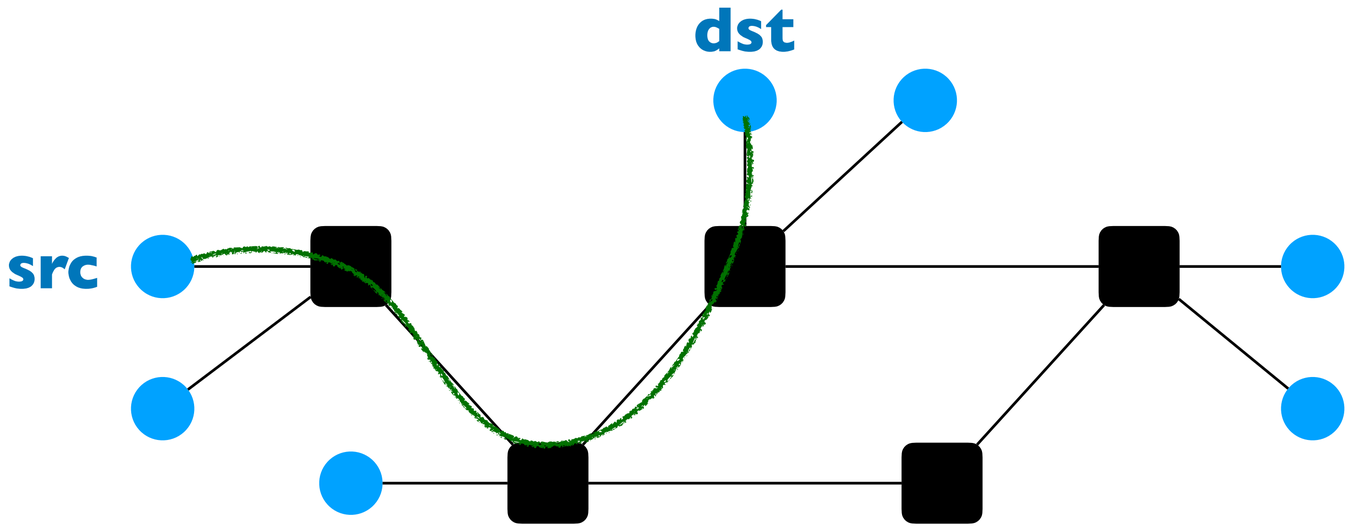
1. The **src** sends a reservation request to **dst**
2. Switches "establish a circuit"

Circuit Switching



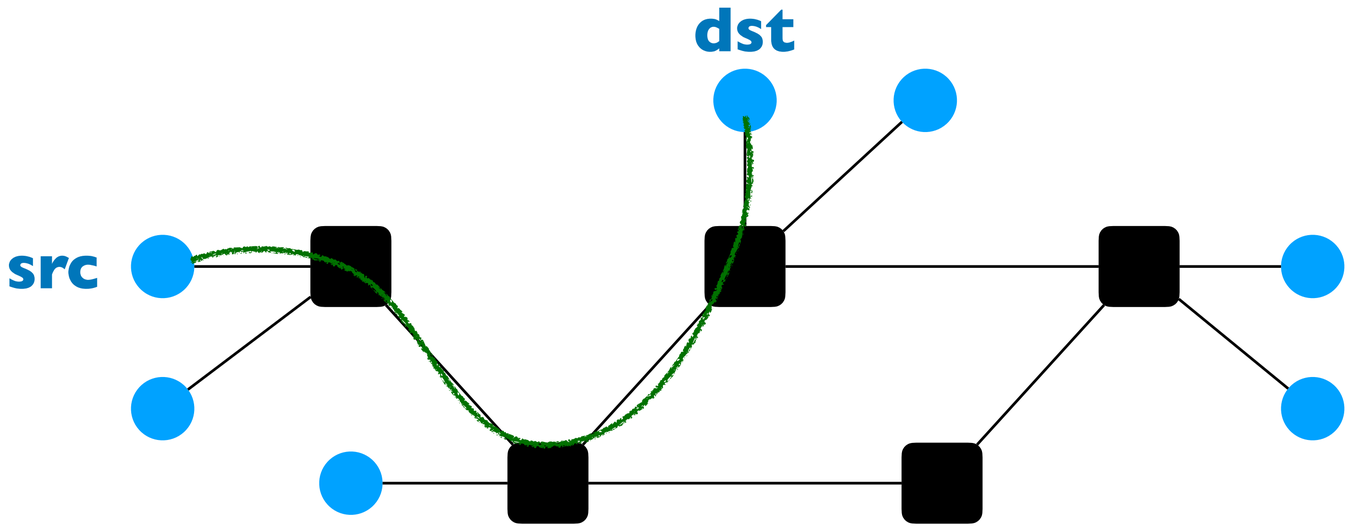
1. The **src** sends a reservation request to **dst**
2. Switches "establish a circuit"

Circuit Switching



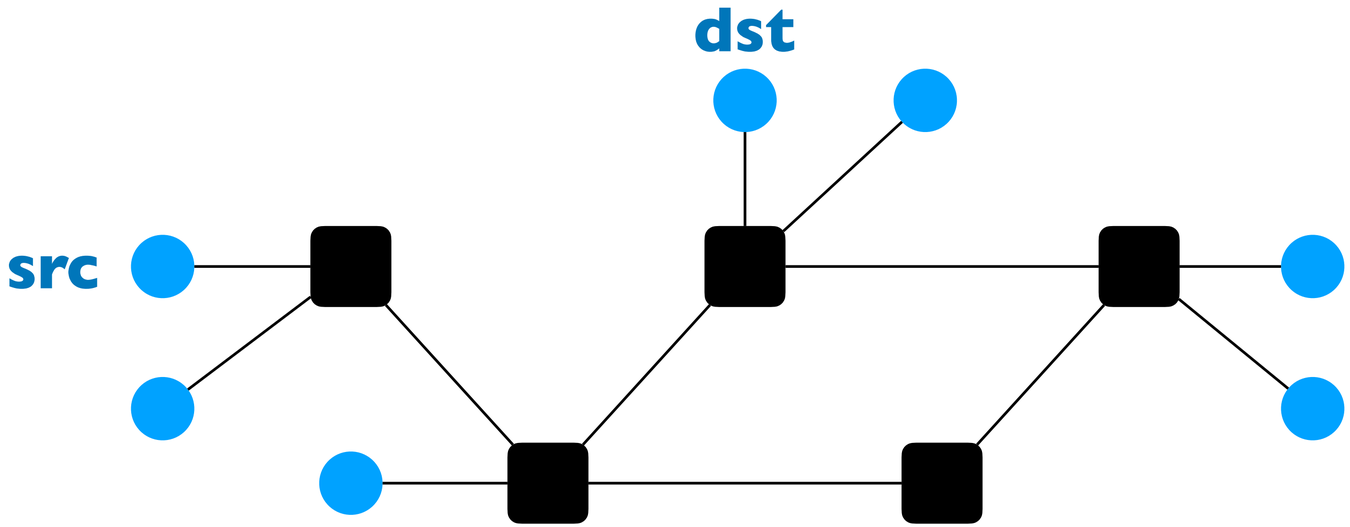
1. The **src** sends a reservation request to **dst**
2. Switches "establish a circuit"

Circuit Switching



1. The **src** sends a reservation request to **dst**
2. Switches "establish a circuit"
3. The **src** starts sending data

Circuit Switching



1. The **src** sends a reservation request to **dst**
2. Switches "establish a circuit"
3. The **src** starts sending data
4. The **src** sends a "teardown circuit" message

Packet Switching

Packet Switching

- Data is sent as chunks of formatted bits (packets)

Packet Switching

- Data is sent as chunks of formatted bits (packets)
- Packets consist of a “header” and “payload”

Packet Switching

- Data is sent as chunks of formatted bits (packets)
- Packets consist of a “header” and “payload”
- Switches “forward” packets based on their headers

Packet Switching

- Data is sent as chunks of formatted bits (packets)
- Packets consist of a “header” and “payload”
- Switches “forward” packets based on their headers
- Each packet travels independently

Packet Switching

- Data is sent as chunks of formatted bits (packets)
- Packets consist of a “header” and “payload”
- Switches “forward” packets based on their headers
- Each packet travels independently
- No link resources are reserved in advance

Packet Switching

- Data is sent as chunks of formatted bits (packets)
- Packets consist of a "header" and "payload"
- Switches "forward" packets based on their headers
- Each packet travels independently
- No link resources are reserved in advance
- Packet switching exploits statistical multiplexing better than circuit switching
  - Sharing using the statistics of demand
  - Good for bursty traffic (average << demand)

Circuit Switching

- **What's good?**
  - *Predictable performance*
  - *Simple/fast switching (once circuit is established)*
- **What's not-so-good?**
  - *Complexity of circuit setup/teardown*
  - *Inefficient when traffic is bursty*
  - *Circuit setup adds delay*
  - *Switch fails -> its circuits fail*

Packet Switching

- **What's good?**
  - *Efficient use of network resources*
  - *Simpler to implement*
  - *Robust: can "route around trouble"*
- **What's not-so-good?**
  - *Unpredictable performance*
  - *Requires buffer management and congestion control*

Performance Metrics

- **Delay:** How long does it take to send a packet from its source to destination?
- **Loss:** What fraction of the packets sent to the destination are dropped?
- **Throughput:** At what rate is the destination receiving data from the source?

A Network Link

- **Link bandwidth**
  - Number of bits sent/received per unit time
- **Propagation delay**
  - Time for one bit to move through the link (seconds)
- **Bandwidth-Delay Product (BDP)**
  - Number of bits "in flight" at any time
  - BDP = Bandwidth x Propagation delay

Delay

- Consists of four components
  - Transmission Delay
  - Propagation Delay
  - Queueing Delay
  - Processing Delay

Delay

- Consists of four components
  - Transmission Delay
  - Propagation Delay
  - Queueing Delay
  - Processing Delay

Transmission Delay, Propagation Delay: Due to link properties

Queueing Delay, Processing Delay: Due to traffic matrix and switch internals

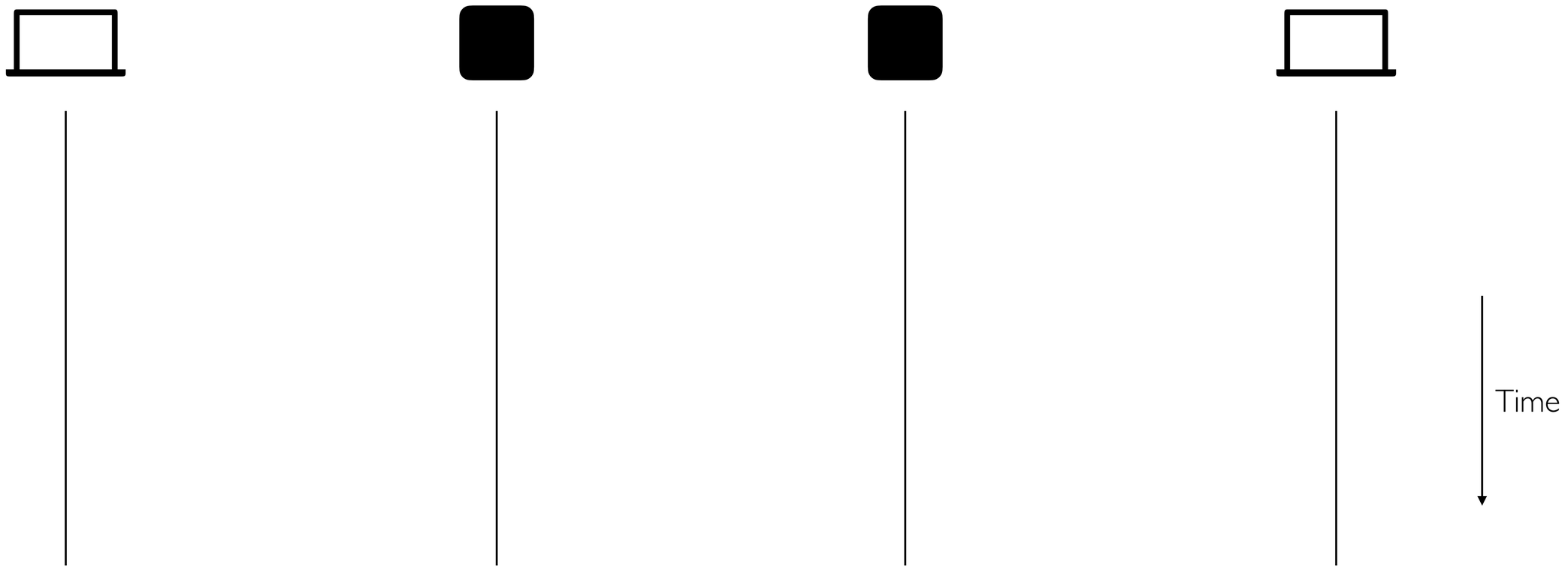
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



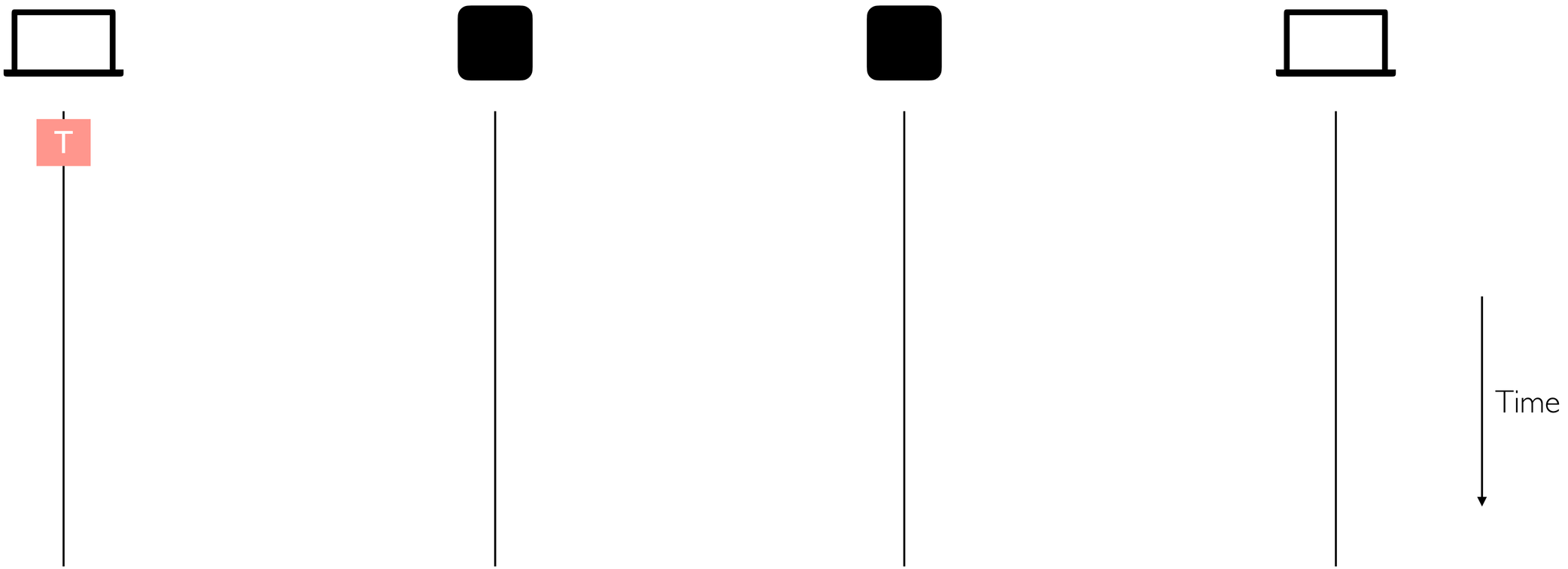
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



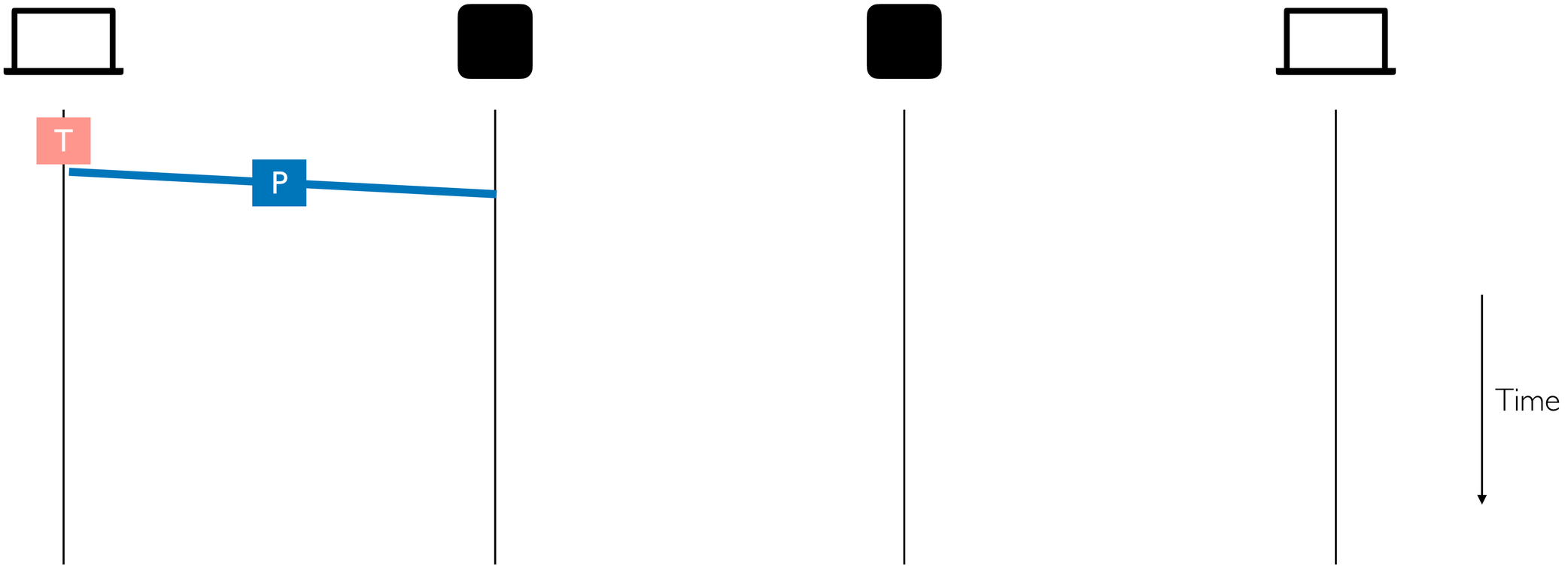
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



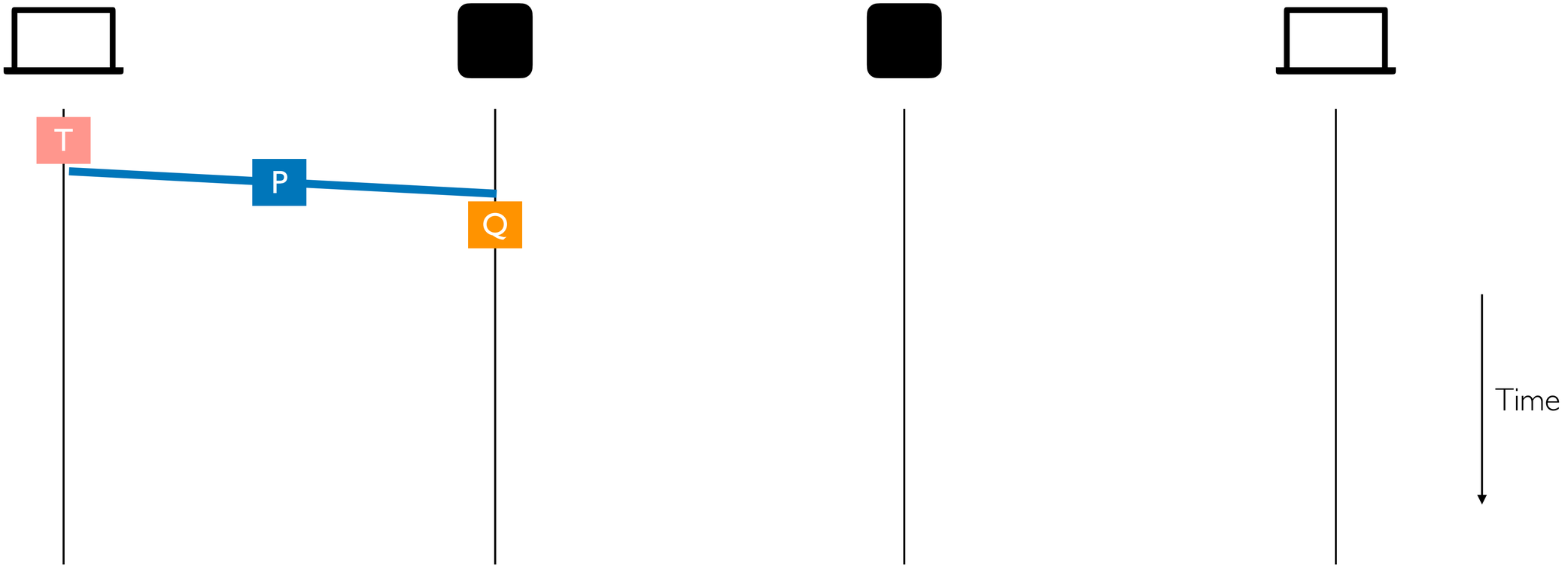
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



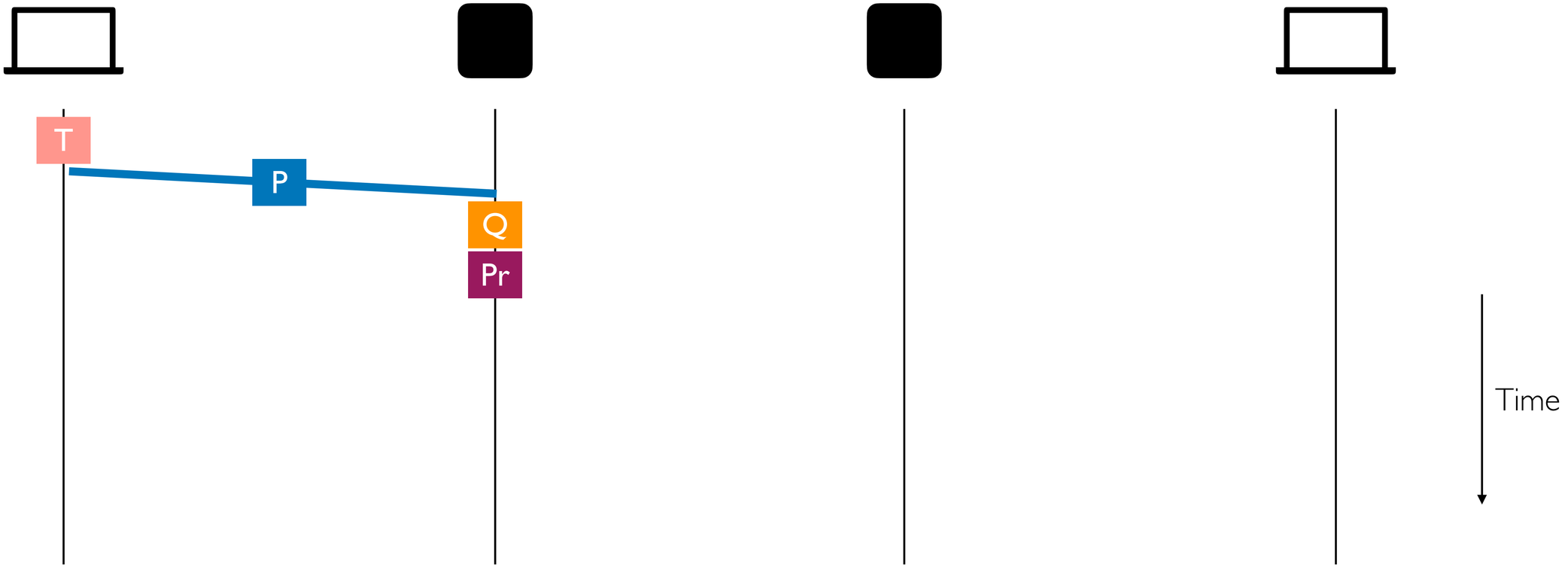
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



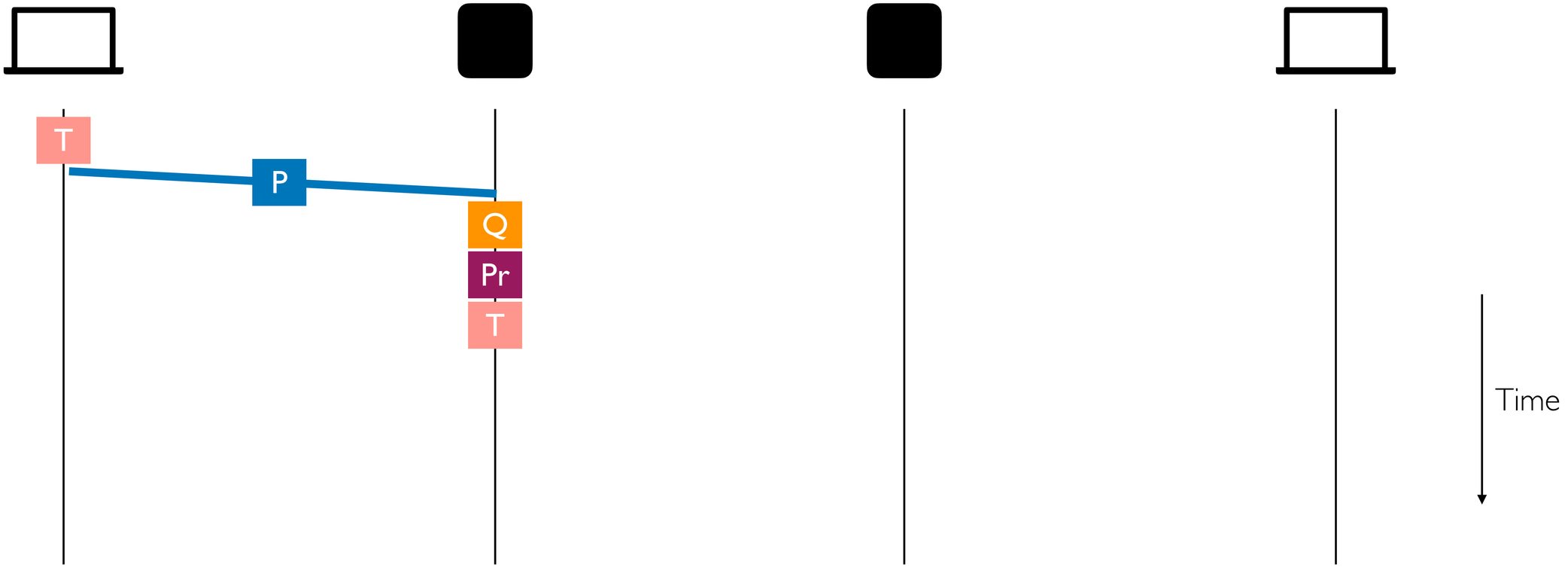
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



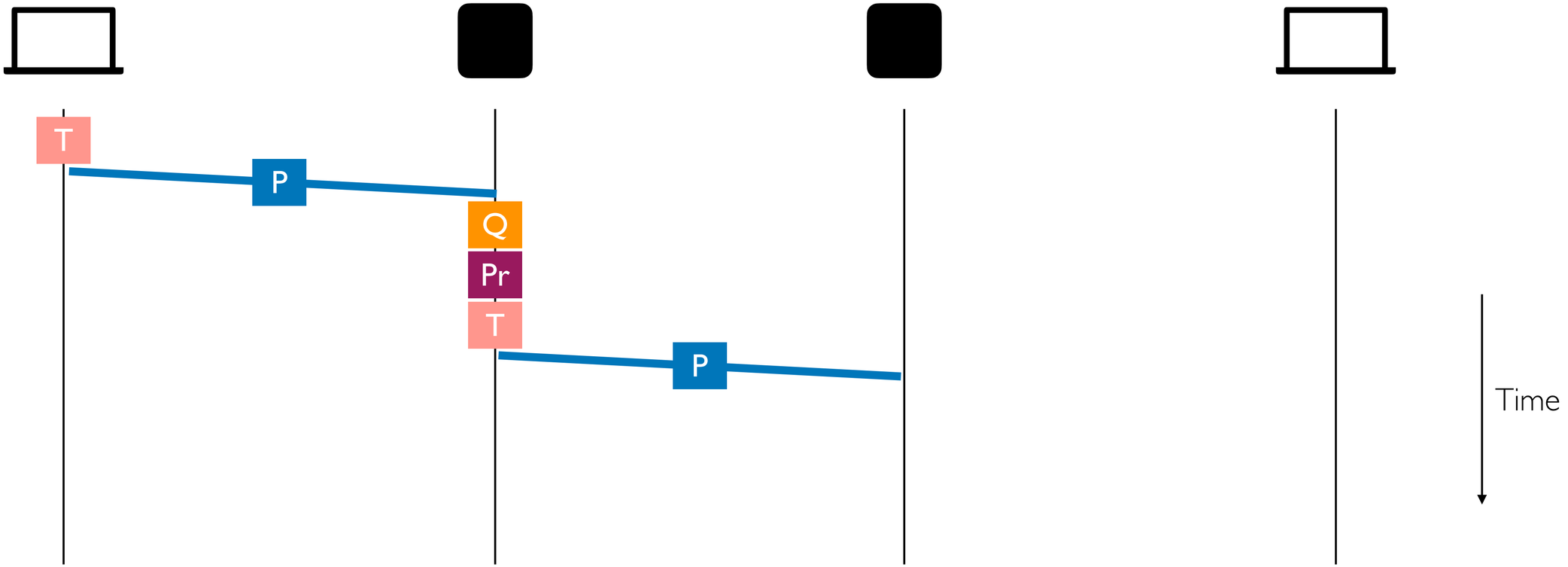
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



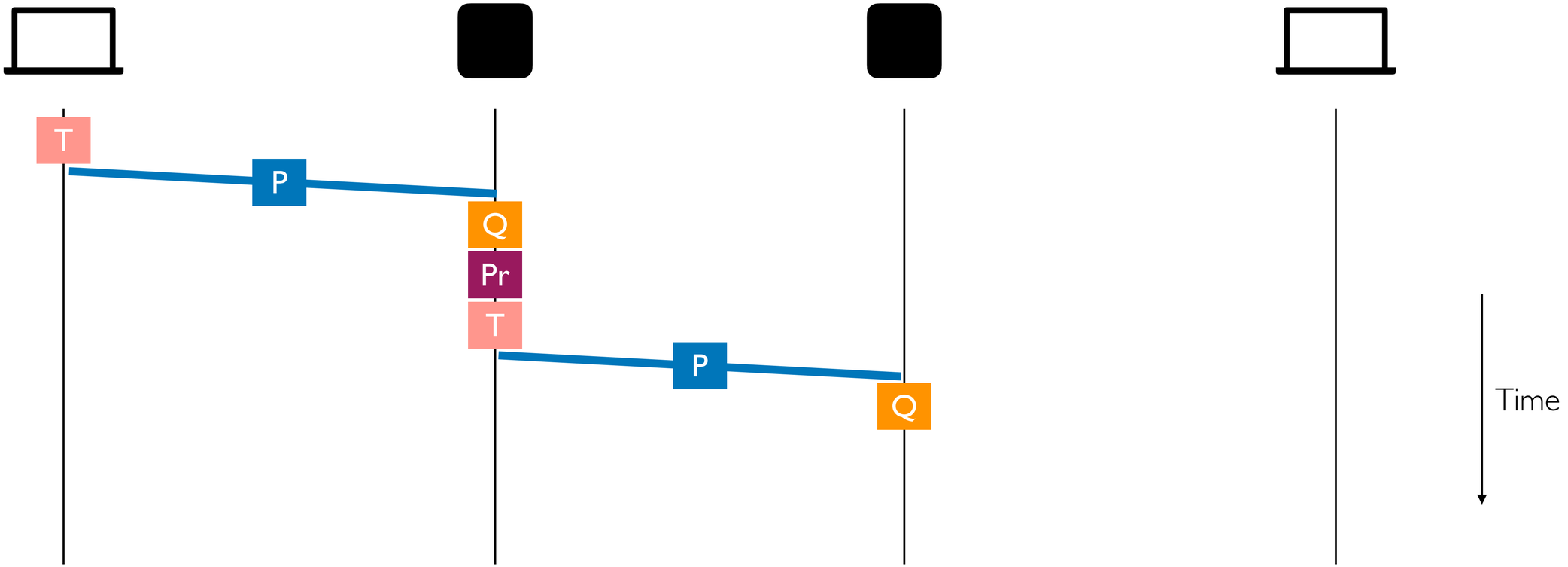
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



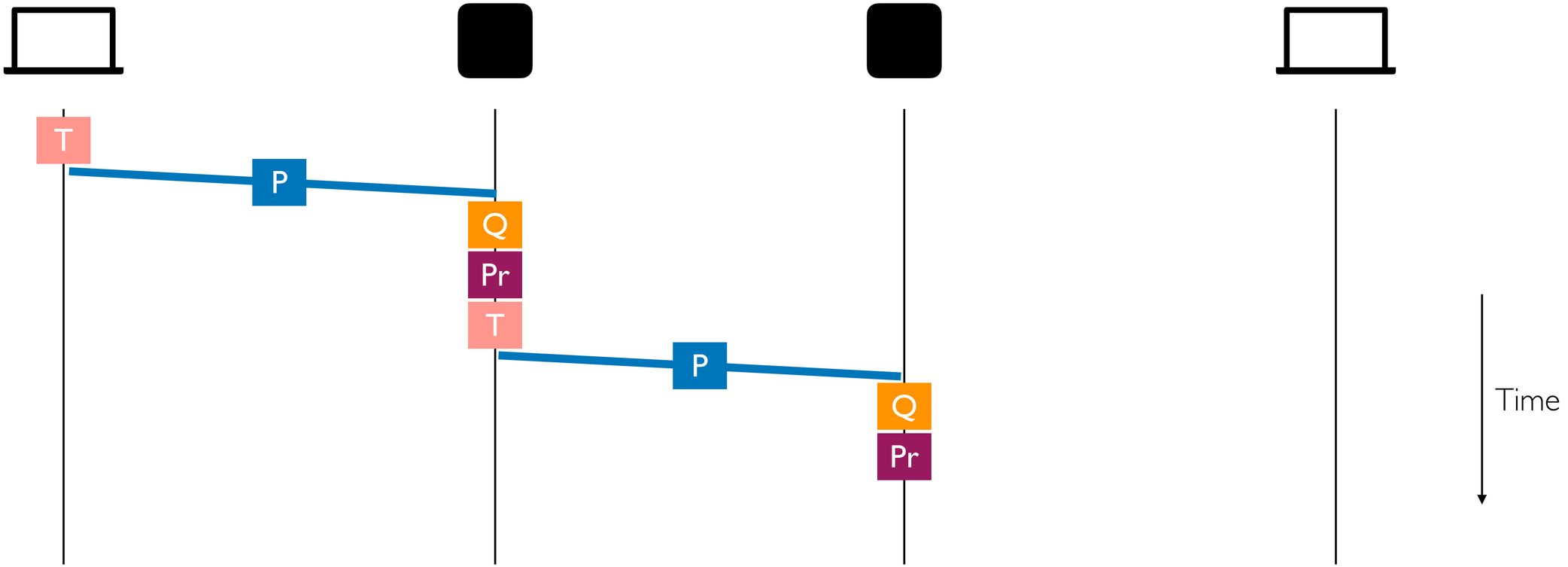
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



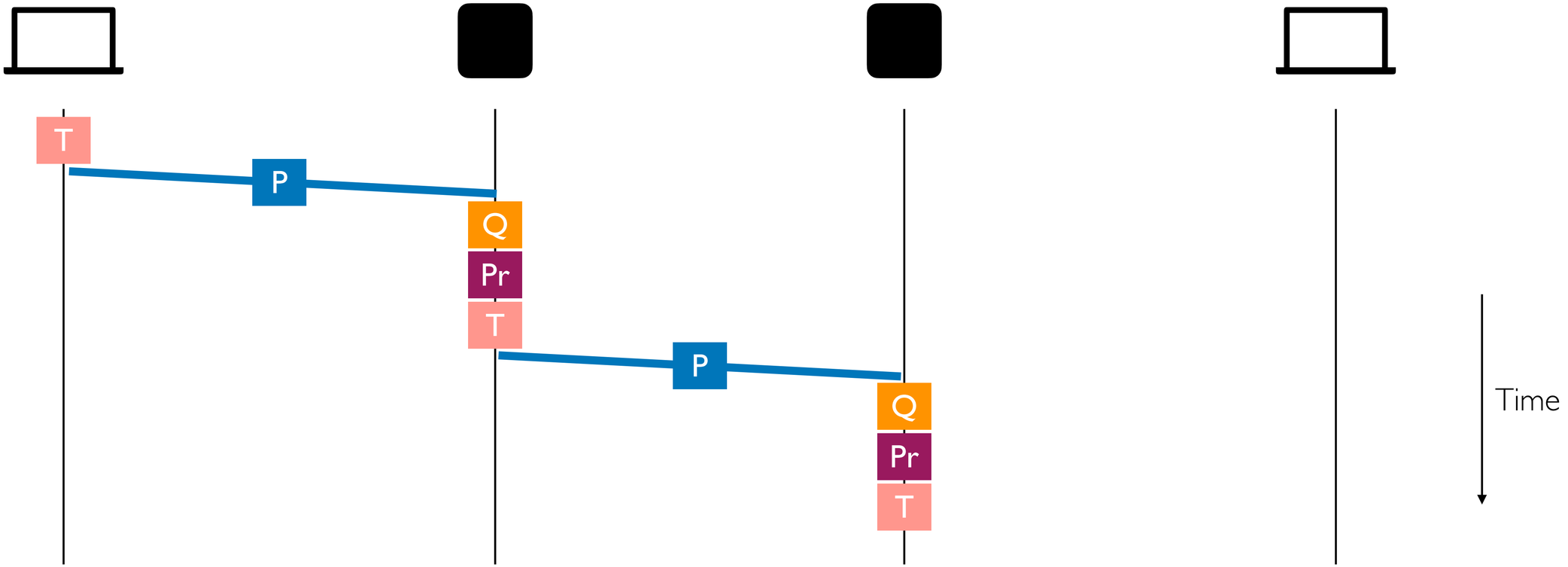
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



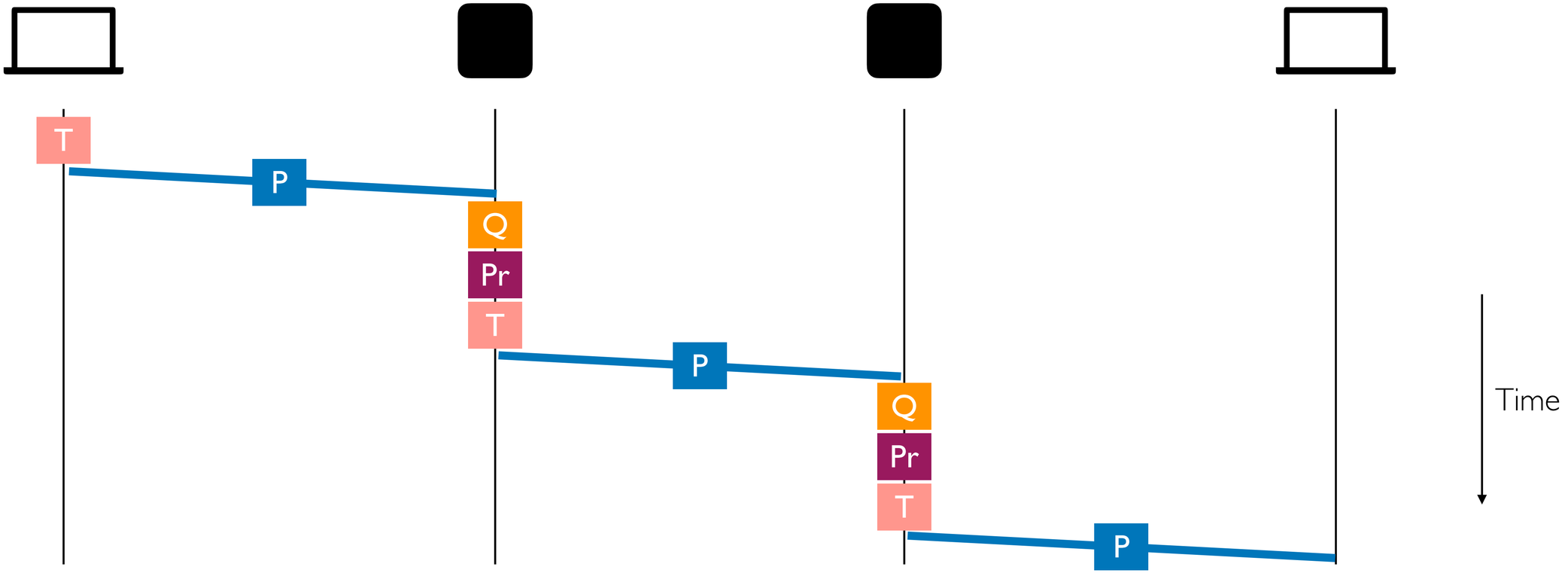
End-to-end Delay

T Transmission Delay

Q Queuing Delay

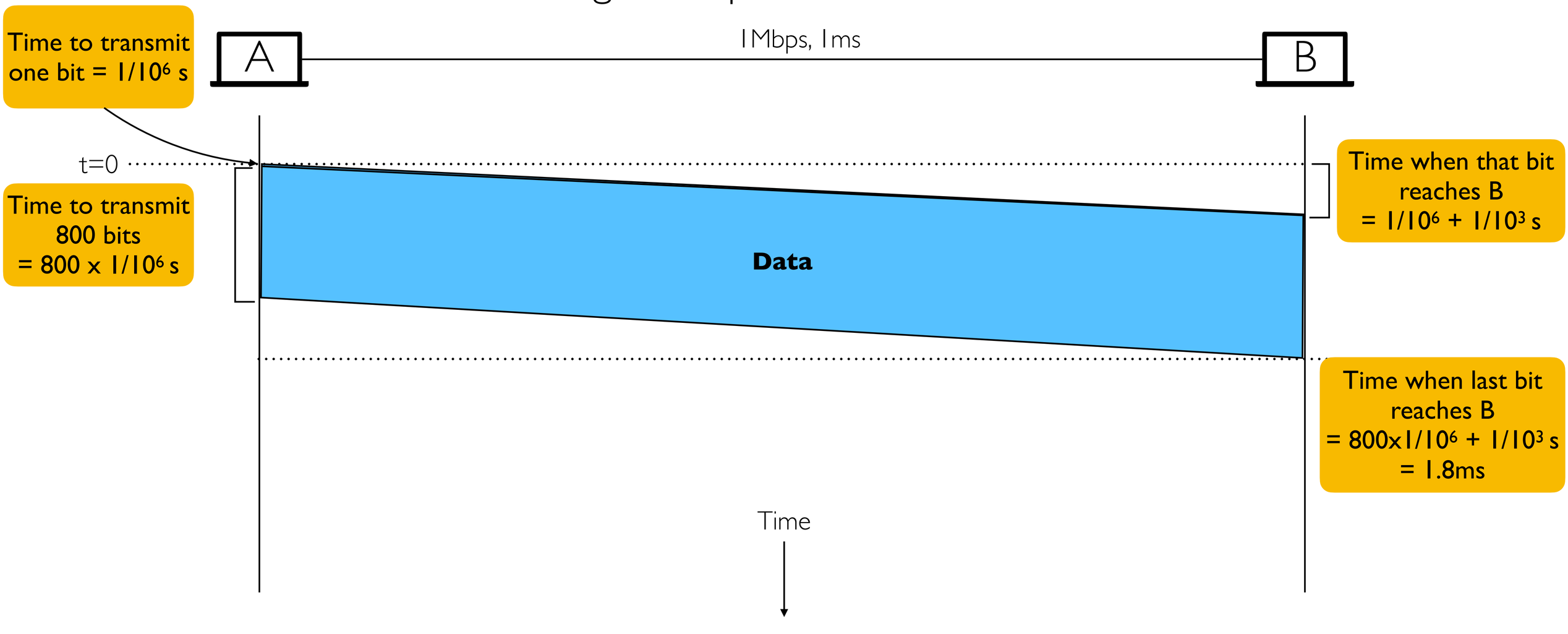
P Propagation Delay

Pr Processing Delay



Packet Delay

Sending 100B packets from A to B?



Queueing Delay: Little's Law

A: avg. packet arrival rate (/s)

L: avg. # of packets waiting in queue

W: avg. time packets wait in queue

$$L = A \times W$$

or,

$$W = L / A$$

Architecture

You should know:

- Layering: what/where/why
- Protocols: what/where/why
- Principles: layering, e2e argument, fate-sharing, "narrow waist"
- Benefits and weaknesses/consequences of principles/choices
  - *E.g., layering is good because… but has hurt…*

Layering

- Layering is a form of modularization
- System is broken into a **vertical hierarchy** of logically distinct entities (layers)
- Service provided by one layer is based **solely** on the service provided by layer below

Internet Layers

Applications

…built on…

Reliable (or unreliable) transport

…built on…

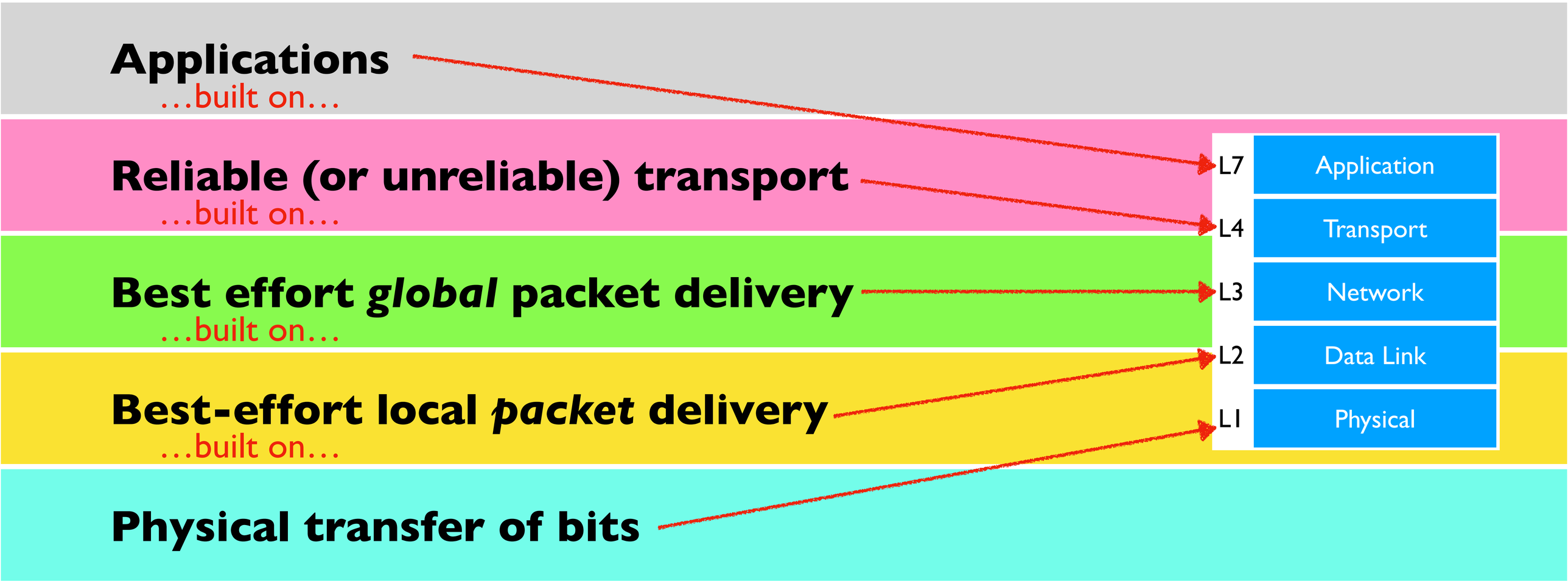
Best effort *global* packet delivery

…built on…

Best-effort local *packet* delivery

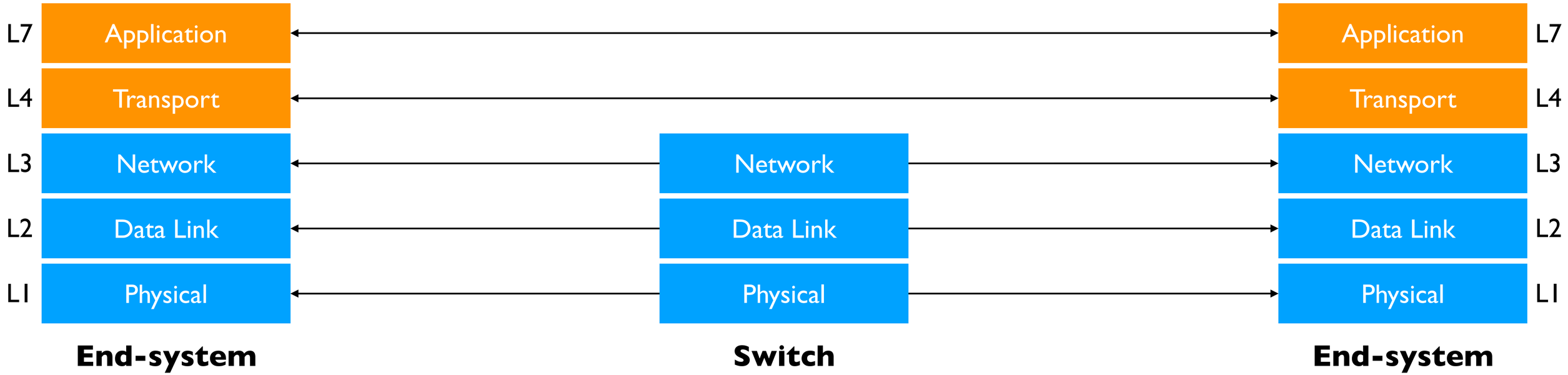
…built on…

Physical transfer of bits



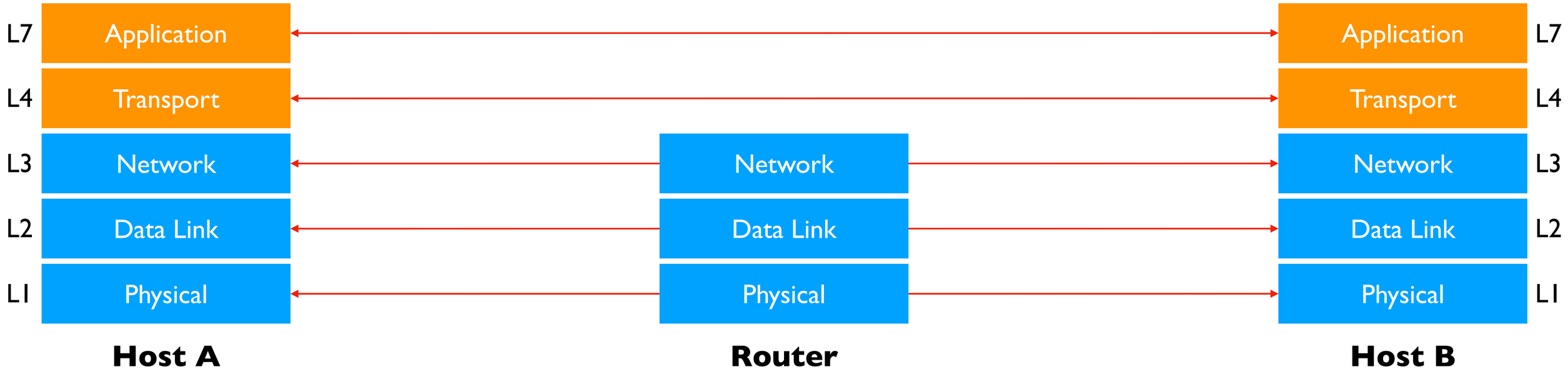
What gets implemented where?

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



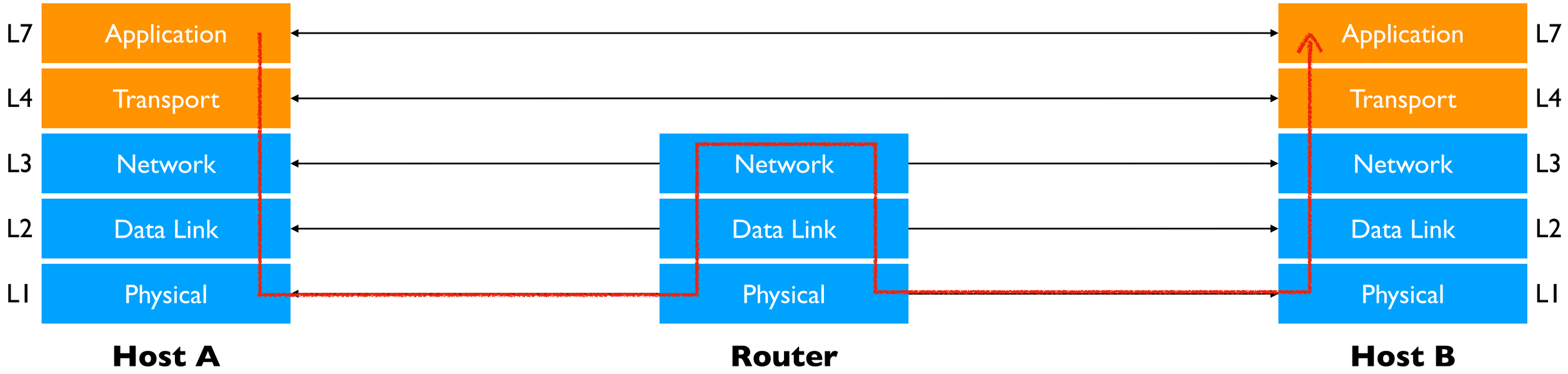
Logical Communication

- Layers interact with peer's corresponding layer



Physical Communication

- Communication goes down to the physical network
- Then up to relevant layer



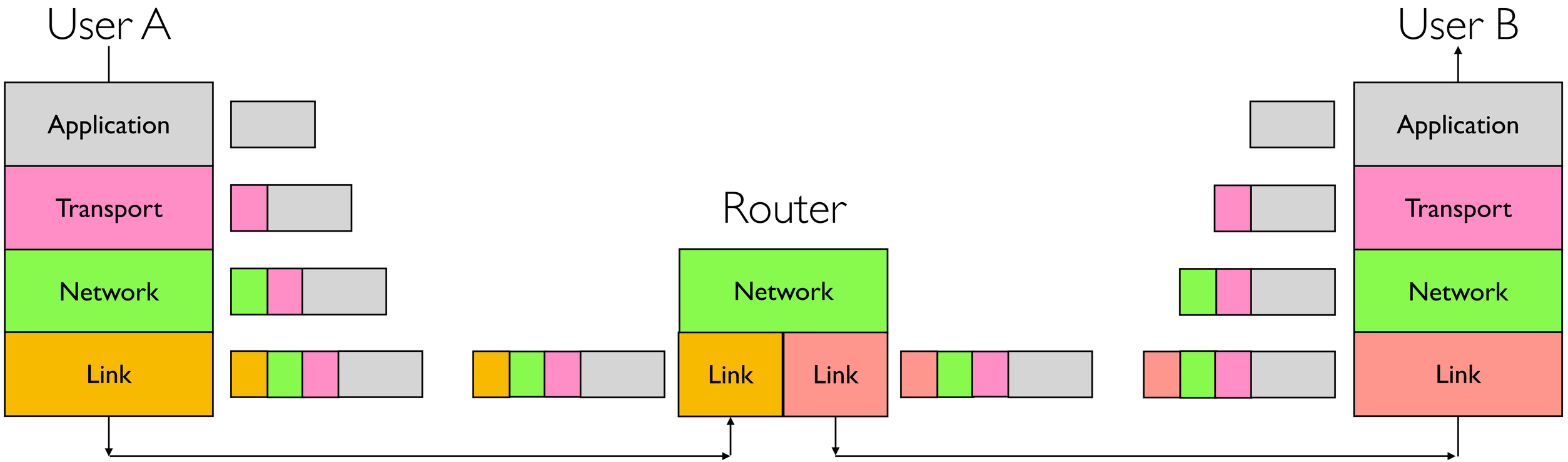
Layers and Protocols

Communication between peer layers on different systems is defined by **protocols**

Protocols at different Layers

There's only ***one*** network layer protocol!

Layer Encapsulation



Layers: Pros & Cons

- **Why Layer?**
  - Reduce complexity
  - Improve flexibility
    - Each layer can evolve independently
- **Why not layer?**
  - Sub-optimal performance
  - Cross-layer information often useful

End-to-end argument: Intuition

- Some application requirements can only be correctly implemented **at the end-system**
  - *Reliability, security, etc.*
- End-systems:
  - ***Can*** *satisfy the requirement without network's help*
  - ***Will/must*** *do so, since they cannot rely on the network*
- Put this functionality at end-system, unless…
  - Performance

Implications of the E2E argument

- In layered design, the E2E principle provides guidance on which layers are implemented where
- Key argument for why IP offers only "best effort" delivery (leading to "dumb network / smart ends")
  - Reliability implemented at the end-system (TCP)
  - Often credited as key to the Internet's success

Architectural Wisdom

- **Layering**
  - Reduce complexity, increase flexibility
- **IP as the "narrow waist"**
  - Eases interoperability
- **"Smart ends, dumb network" (E2E argument)**
  - No application knowledge in network → more general
  - **Fate sharing:** No/minimal state in network → more robust to failure

Network Layer

Forwarding vs. Routing

- **Forwarding:** "data plane"
  - Directing one data packet
  - Each router using local routing state
- **Routing:** "control plane"
  - Computing the forwarding tables that guide packets
  - Jointly computed by routers using a distributed algorithm

Routing: Basic concepts

- Valid routing state
- Convergence
- Least-cost paths

“Valid” Routing State

- Global state is **“valid”** if it produces forwarding decisions that always deliver packets to their destinations

- Global routing state is “valid” ***if and only if:***

  - There are no dead-ends (other than the destination)

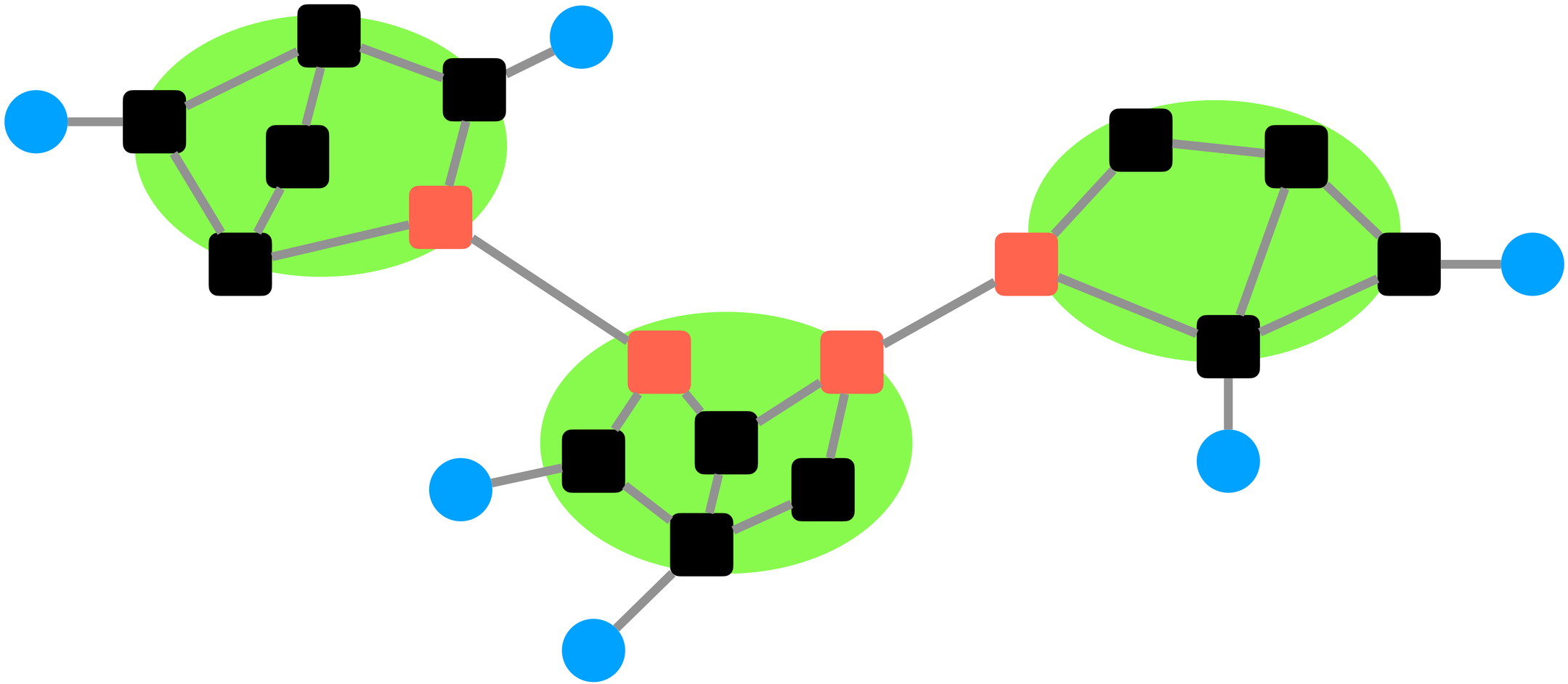
  - There are no loops

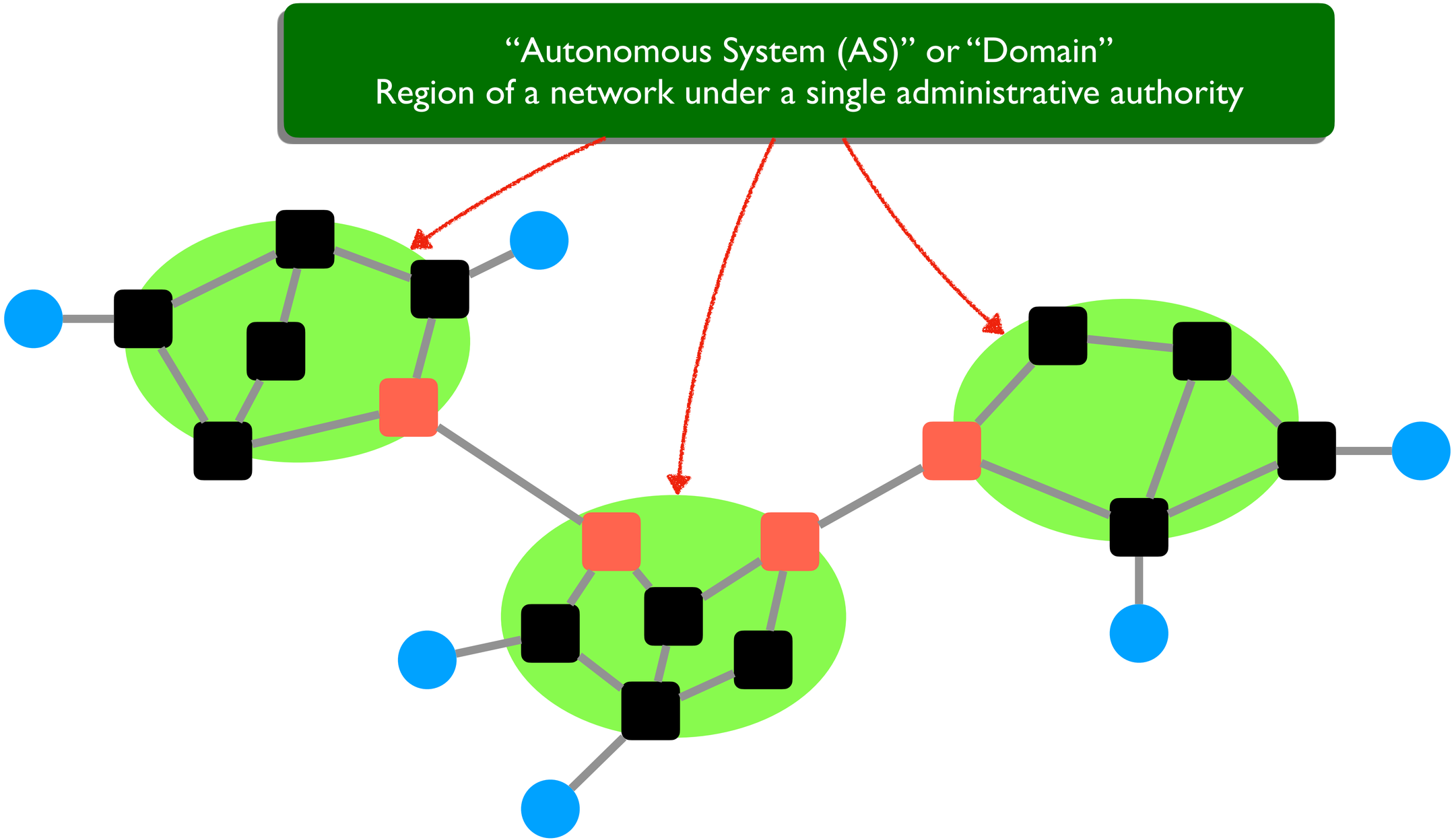
Convergence Delay

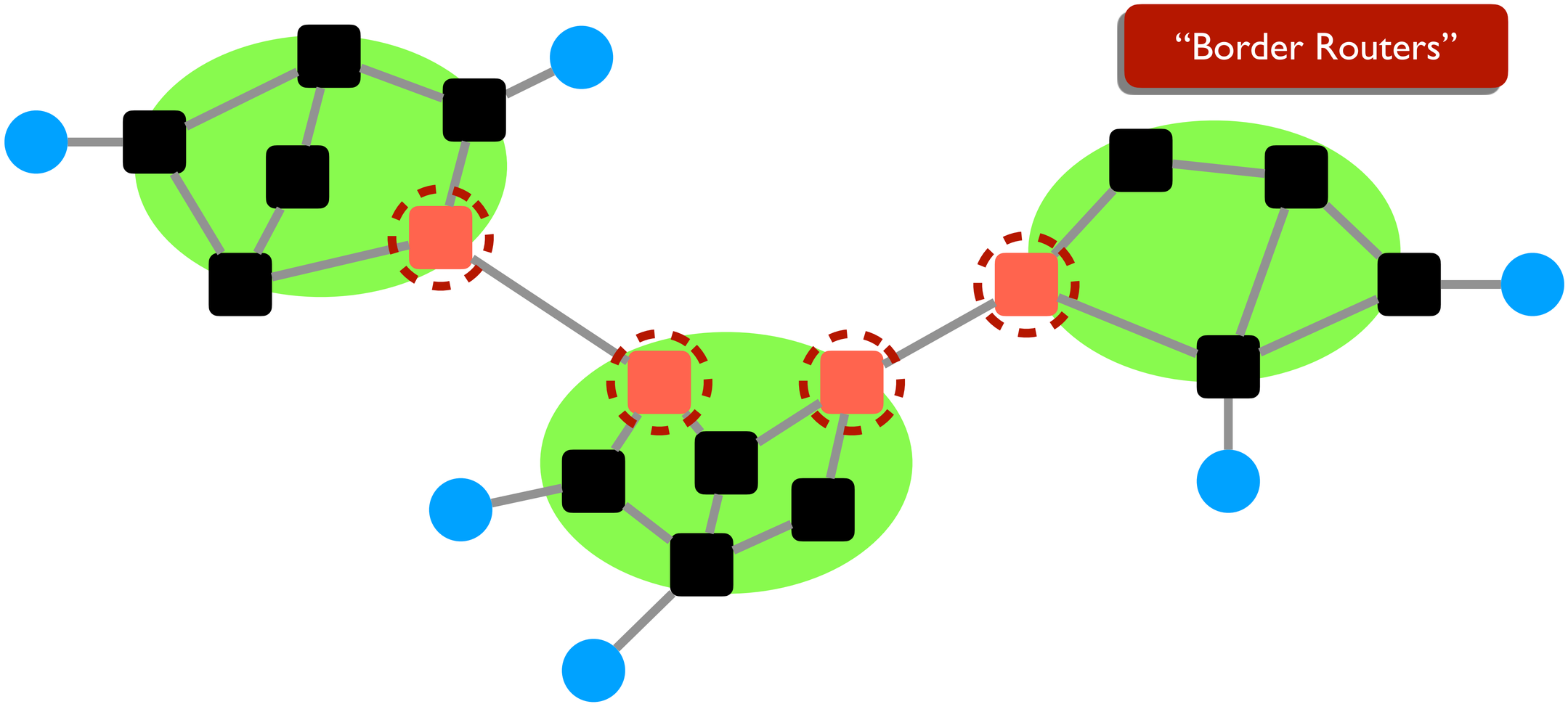
- **Time to achieve convergence**
  - E.g., all nodes have the same link-state database
- **Sources of convergence delay?**
  - Time to detect failure
  - Time to flood link-state information
  - Time to recompute forwarding tables
- **Performance during convergence period**
  - Lost packets due to black-holes
  - Looping packets
  - Out-of-order packets reaching the destination

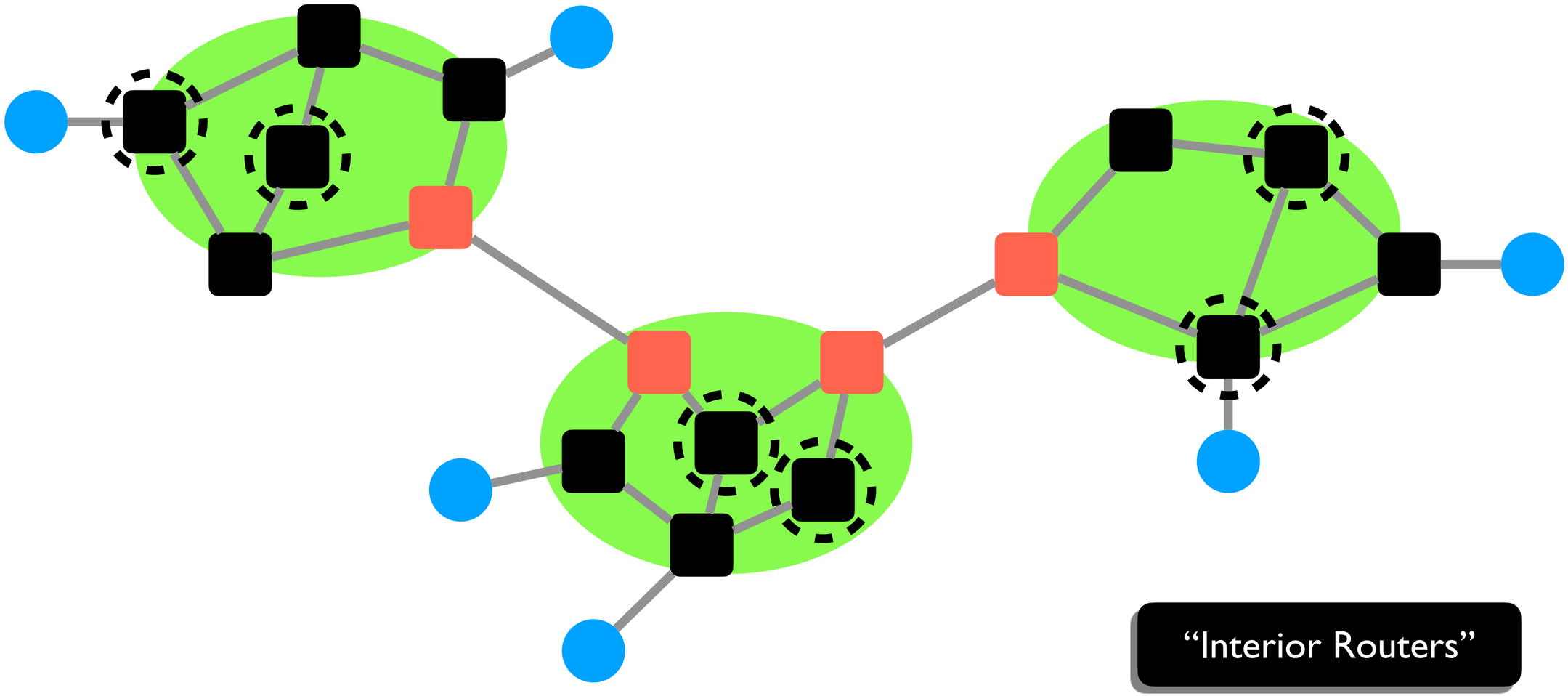
Least-cost path routing

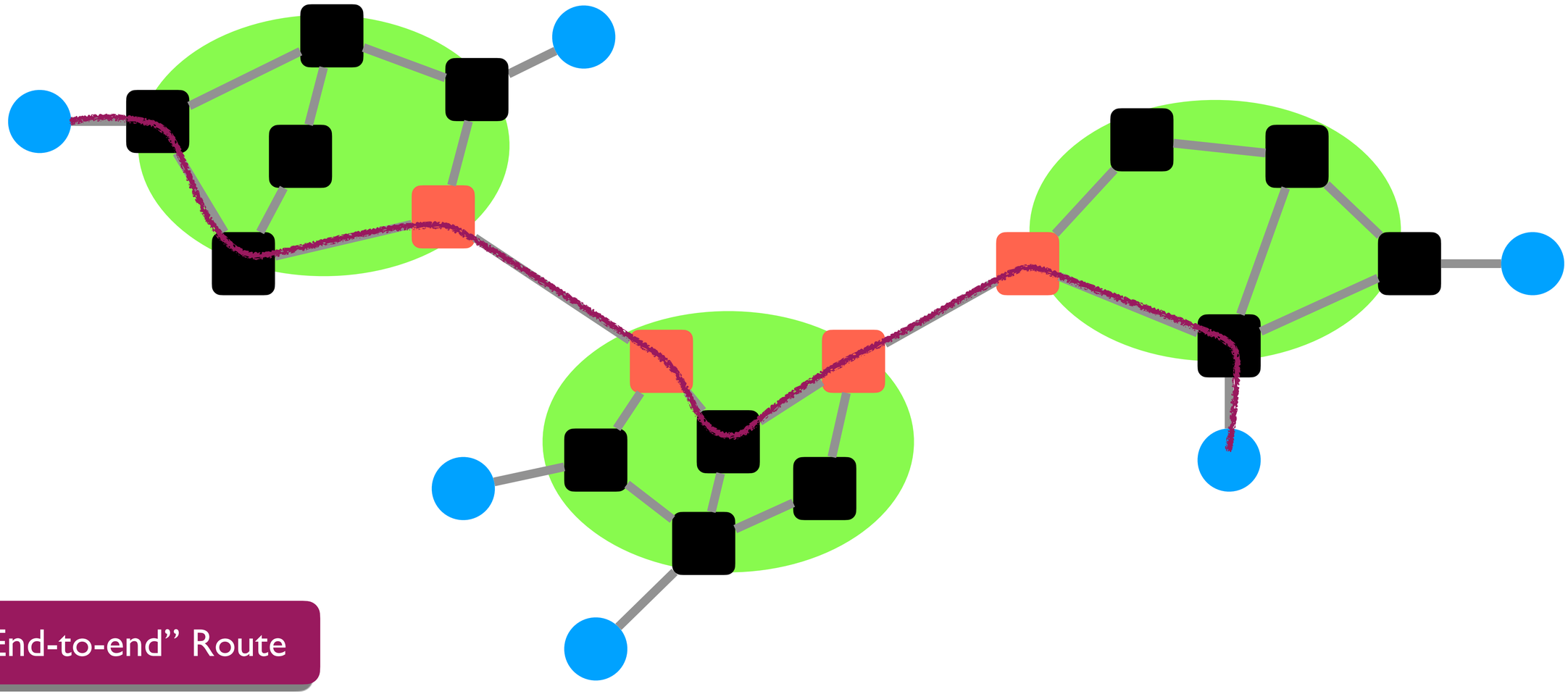
- **Given:** Router graph & link costs
- **Goal:** find least-cost path
  from each source router
  to each destination router
- **"Least cost" routes are an easy way to avoid loops**
  - *No **sensible** cost metric is minimized by traversing a loop*
- **Least cost routes are also "destination-based"**
  - *i.e., they do not depend on the source*
- **Least cost paths form a spanning tree**




"Autonomous System (AS)" or "Domain"
Region of a network under a single administrative authority


"Border Routers"


"Interior Routers"



"End-to-end" Route

Internet Routing

- Internet routing works at two levels

- Each AS runs an **intra-domain** routing protocol that establishes routes within its domain
  - Intra-domain routes are "least cost"
  - E.g., Link State (OSPF) and Distance Vector (RIP)

- ASes participate in an **inter-domain** routing protocol that establishes routes between domains
  - Inter-domain routes determined by policy (need not be least-cost)
  - E.g., Path Vector (BGP)

Link State Routing

Link State Routing

- Every router knows its local "link state"

Link State Routing

- Every router knows its local “link state”
- A router floods its link state to all other routers

Link State Routing

- Every router knows its local "link state"
- A router floods its link state to all other routers
- Hence, every router learns the entire network graph

Link State Routing

- Every router knows its local "link state"
- A router floods its link state to all other routers
- Hence, every router learns the entire network graph
- Every router locally runs Dijkstra's to compute its forwarding table

Distance-vector Routing

Distance-vector Routing

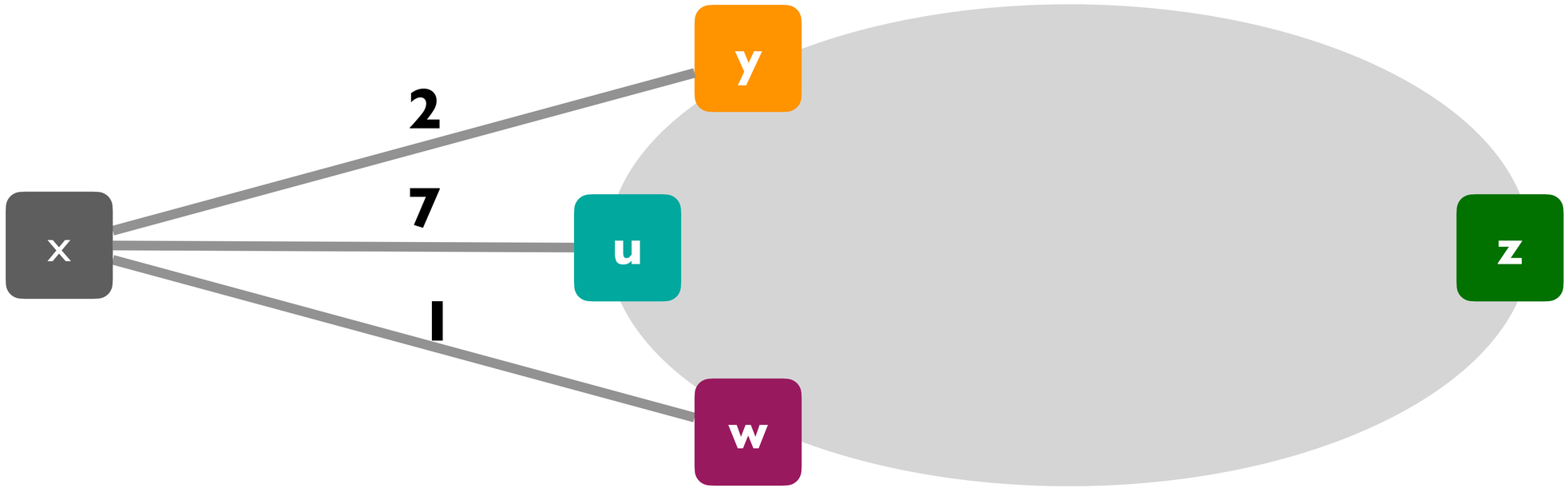
- Distributed algorithm

Distance-vector Routing

- Distributed algorithm
- All routers run it “together"
  - *Each router runs its own instance*
  - *Neighbors exchange and react to each other’s messages*

Distance-vector Routing

- Each router knows the links to its neighbors
- Each router has provisional "least cost" estimate to every other router — its distance vector (DV)
  - *E.g., Router A: "A can get to B with cost 1 1"*
- Routers exchange this DV with their neighbors
- Routers look over the set of options offered by their neighbors and select the best one
- Iterative process converges to set of shortest paths



$$d_x(z) = \min_n \{ \; cost(x,n) + d_n(z) \; \}$$

For all neighbors n

Bellman-Ford Equation

DV Routing: Problems & Solutions

- **Problem: Count-to-infinity**
  - Cause:
    - z routes through y, y routes through x (to reach dst)
    - y loses connectivity to x
    - y decides to route through z (to reach dst)
  - Can take a very long time to resolve
- **Solution: Split Horizon with Poisoned Reverse**
  - How:
    - If z routes to dst through y, z advertises to y that its cost to dst is infinite
    - y never dictates to route to dst through z
  - Often avoids the count-to-infinity problem

Addressing Goal: **Scalable** Routing

- **State:** Small forwarding tables at routers
  - Much less than the number of hosts
- **Churn:** Limited rate of change in routing tables

Ability to aggregate addresses is crucial for both
(One entry to *summarize* many addresses)

IP Addresses (IPv4)

- 32 bits are partitioned into a prefix and suffix components

IP Addresses (IPv4)

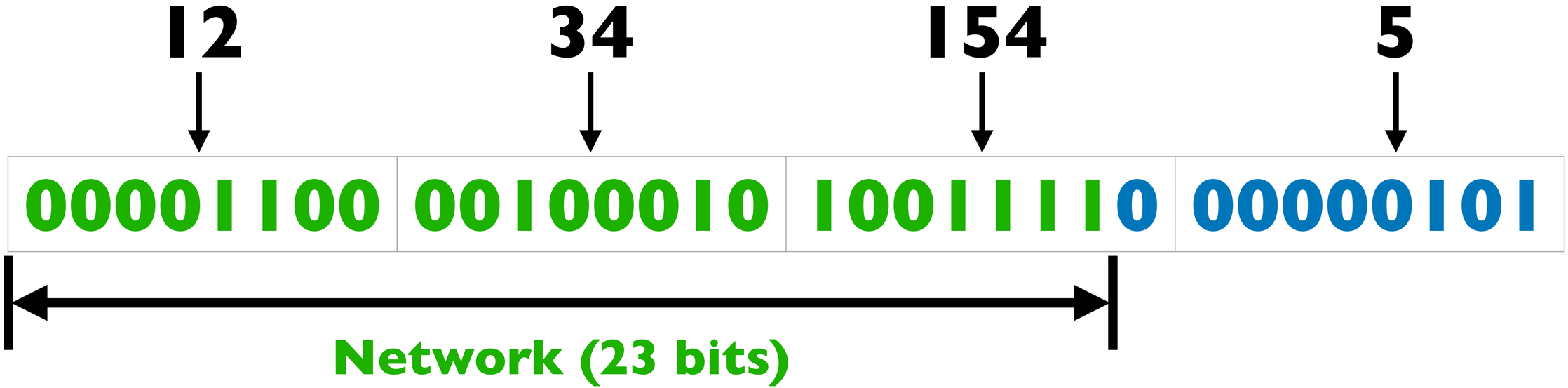
- 32 bits are partitioned into a prefix and suffix components
- Prefix if the **network component**; suffix is the **host component**

IP Addresses (IPv4)

- 32 bits are partitioned into a prefix and suffix components
- Prefix if the **network component**; suffix is the **host component**

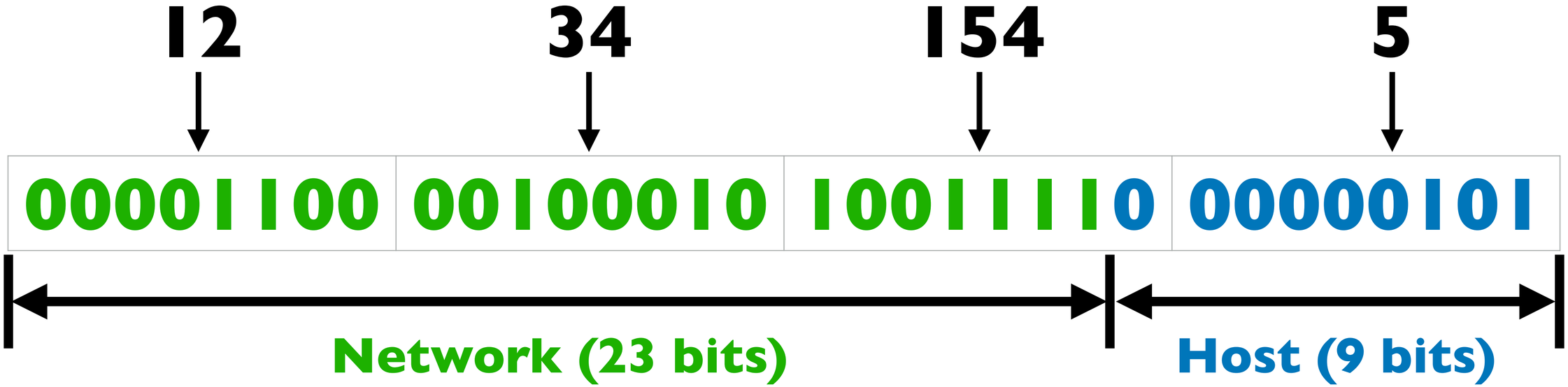
IP Addresses (IPv4)

- 32 bits are partitioned into a prefix and suffix components
- Prefix if the **network component**; suffix is the **host component**



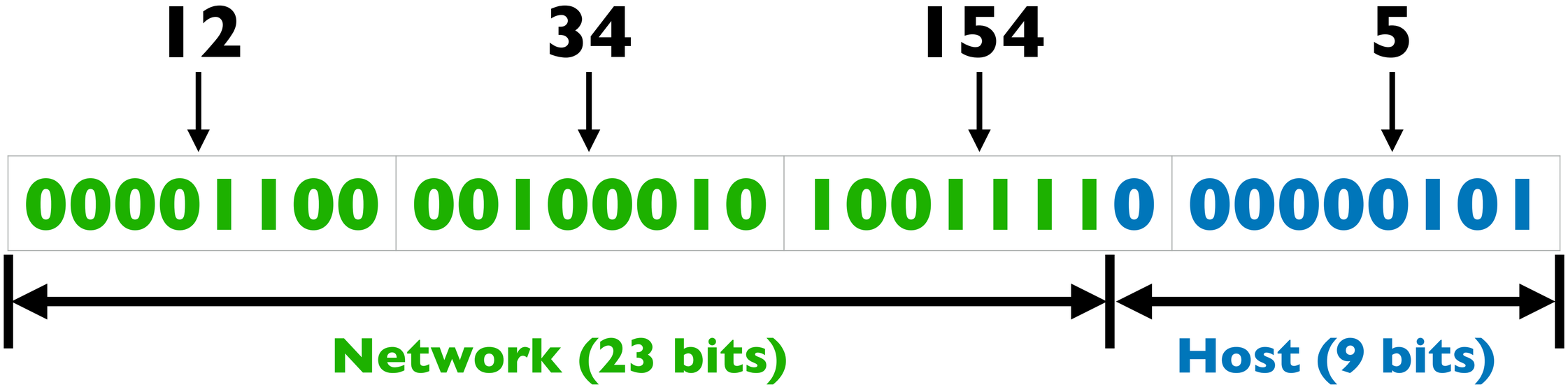
IP Addresses (IPv4)

- 32 bits are partitioned into a prefix and suffix components
- Prefix if the **network component**; suffix is the **host component**



IP Addresses (IPv4)

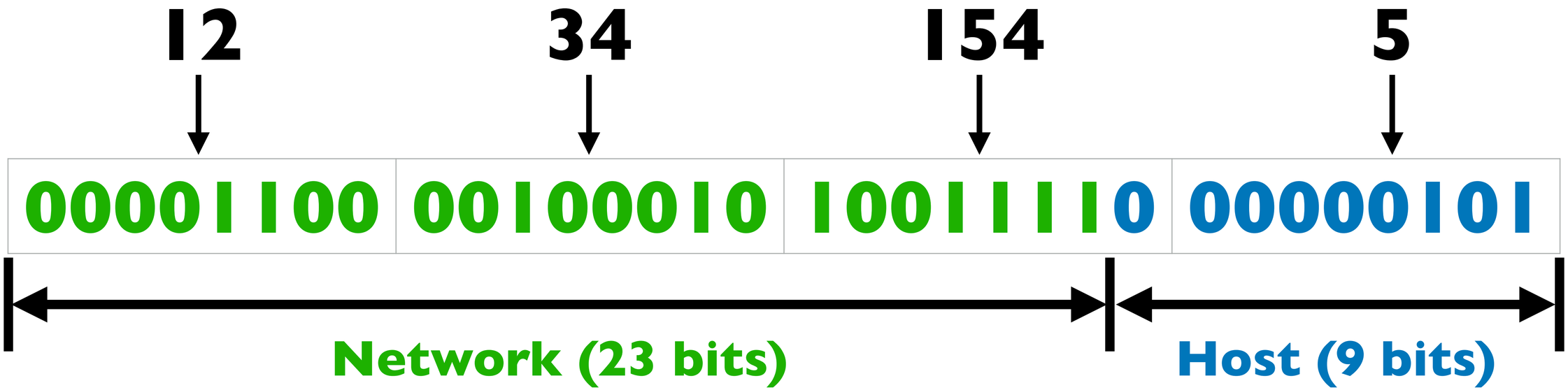
- 32 bits are partitioned into a prefix and suffix components
- Prefix if the **network component**; suffix is the **host component**



- Interdomain routing operates on the **network prefix**

IP Addresses (IPv4)

- 32 bits are partitioned into a prefix and suffix components
- Prefix if the **network component**; suffix is the **host component**



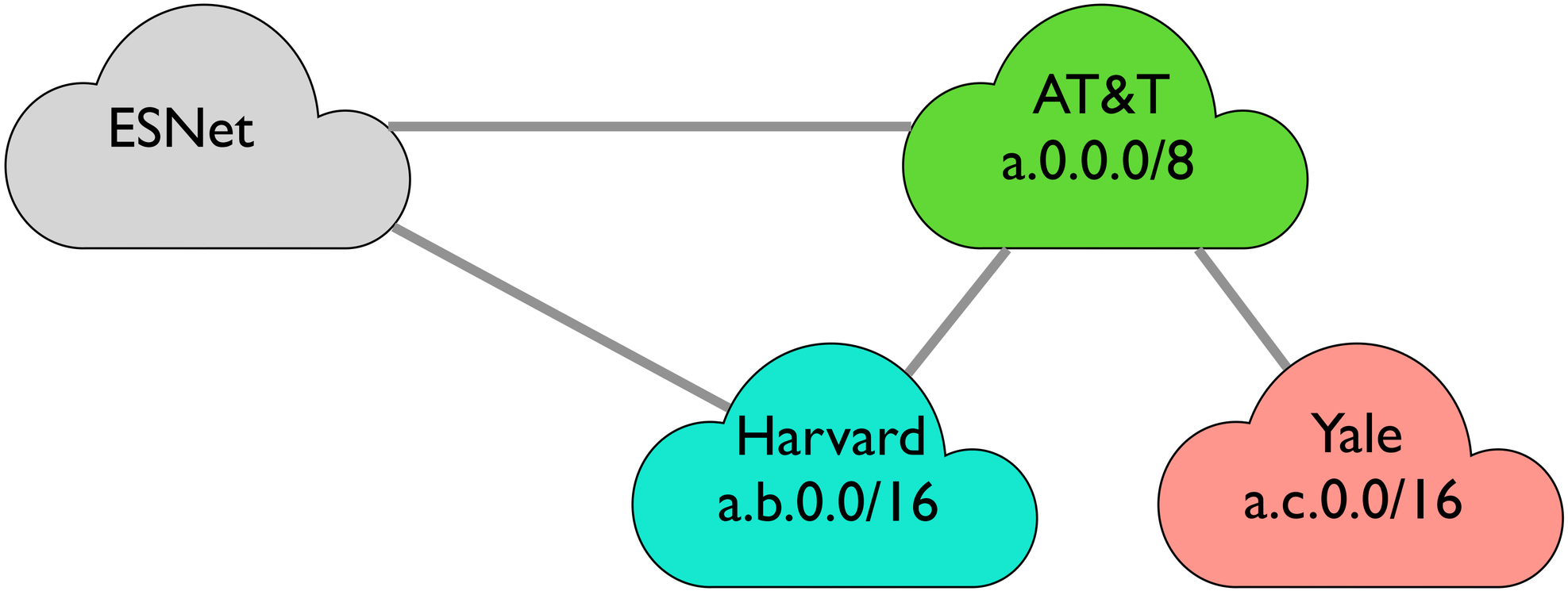
- Interdomain routing operates on the **network prefix**
- "Slash" notation: 12.34.158.0/23 → network with a 23 bit prefixes and 2^9 host addresses

IP Addressing → Scalable Routing?

Hierarchical address allocation only helps routing scalability if allocation matches topological hierarchy

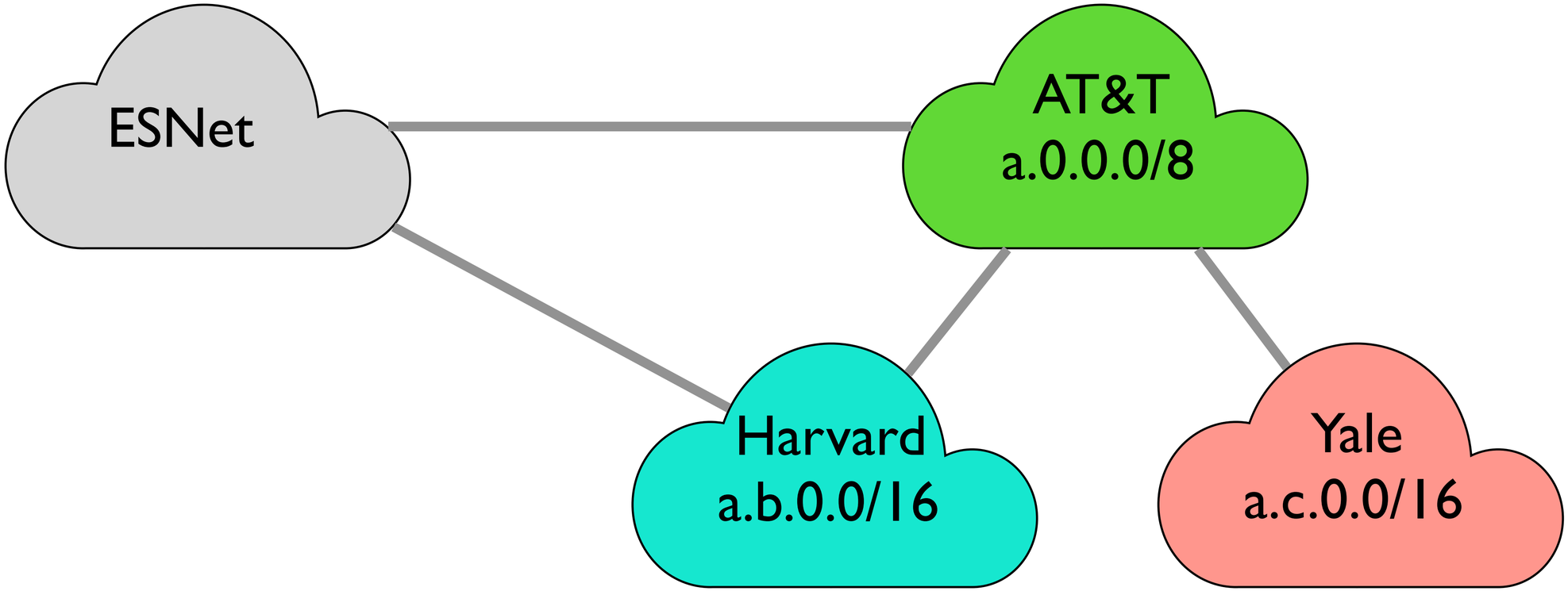
- **Problem:** May not be able to aggregate addresses for "multi-homed" networks

IP Addressing → Scalable Routing?



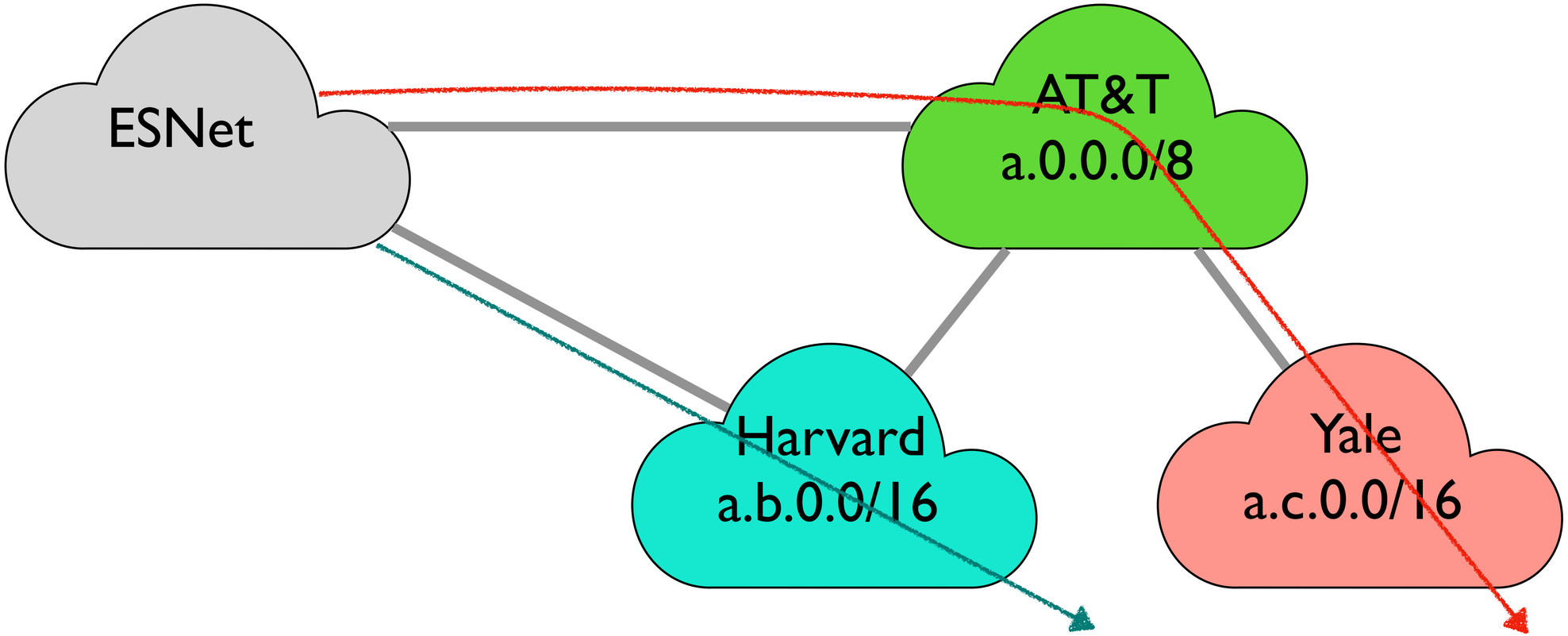
IP Addressing → Scalable Routing?

Harvard is "multi-homed" to AT&T and ESNet (fake example)
Multi-homed domain → domain has 2 (or more) providers



IP Addressing → Scalable Routing?

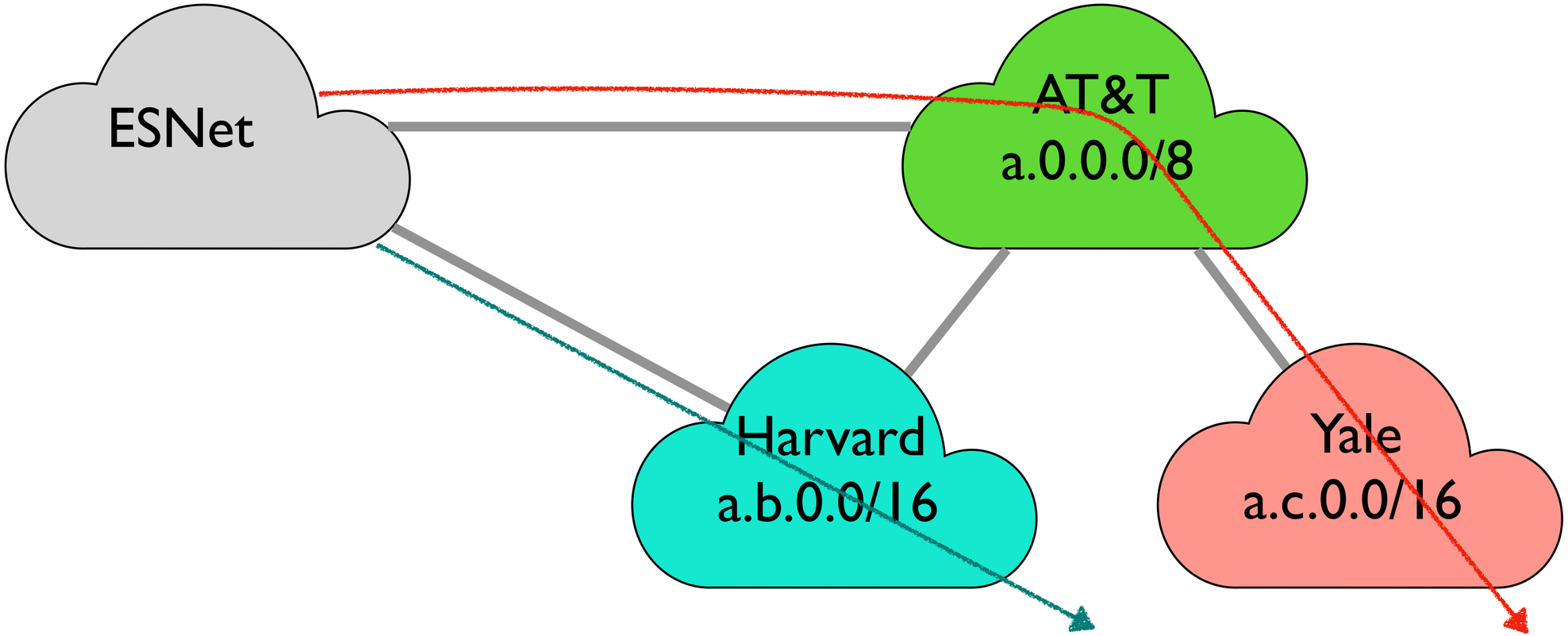
Harvard is "multi-homed" to AT&T and ESNet (fake example)
Multi-homed domain → domain has 2 (or more) providers



IP Addressing → Scalable Routing?

Harvard is "multi-homed" to AT&T and ESNet (fake example)
Multi-homed domain → domain has 2 (or more) providers

ESNet must maintain routing entries for a.*.*.* and a.b.*.*



IP Addressing → Scalable Routing?

Hierarchical address allocation only helps routing scalability if allocation matches topological hierarchy

- **Problem:** May not be able to aggregate addresses for "multi-homed" networks
- Two competing forces in scalable routing
  - Aggregation reduces the number of routing entries
  - Multi-homing increases number of entries

BGP & Interdomain Routing

- Destinations are IP prefixes (12.0.0.0/8)
- Nodes are Autonomous Systems (ASes)
- Links represent both physical links and business relationships
  - Customer-provider or peer-to-peer
- Border Gateway Protocol (BGP) is the Interdomain routing protocol
  - Implemented by AS border routers

Topology and Policy is shaped by business relationships between ASes

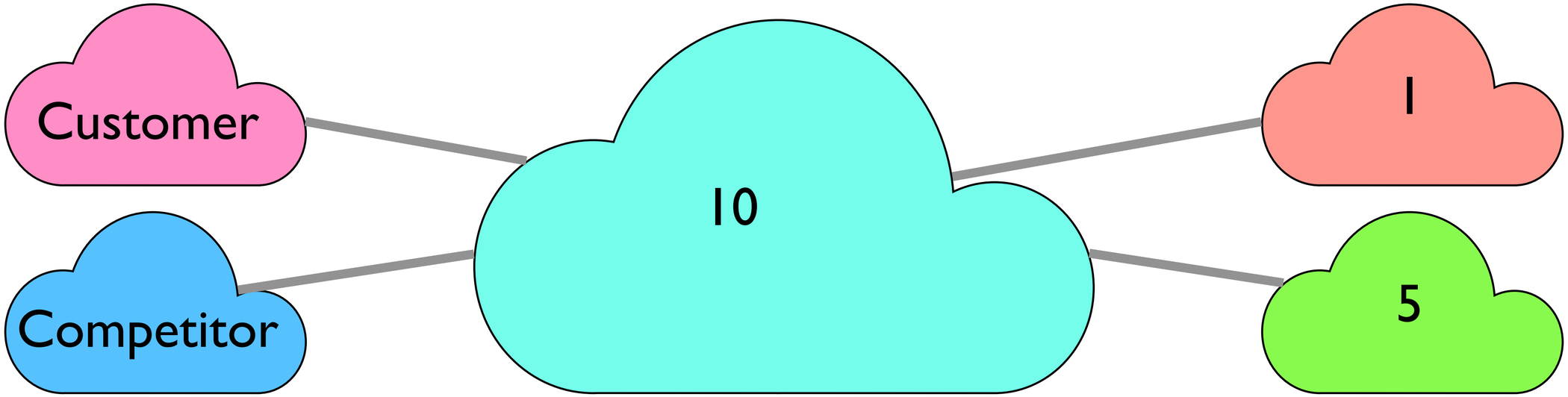
- **Three basic kinds of relationships between ASes**
  - AS A can be AS B's *customer*
  - AS A can be AS B's *provider*
  - AS A can be AS B's *peer*
- **Business implications**
  - Customer pays provider
  - Peers don't pay each other

BGP extends DV

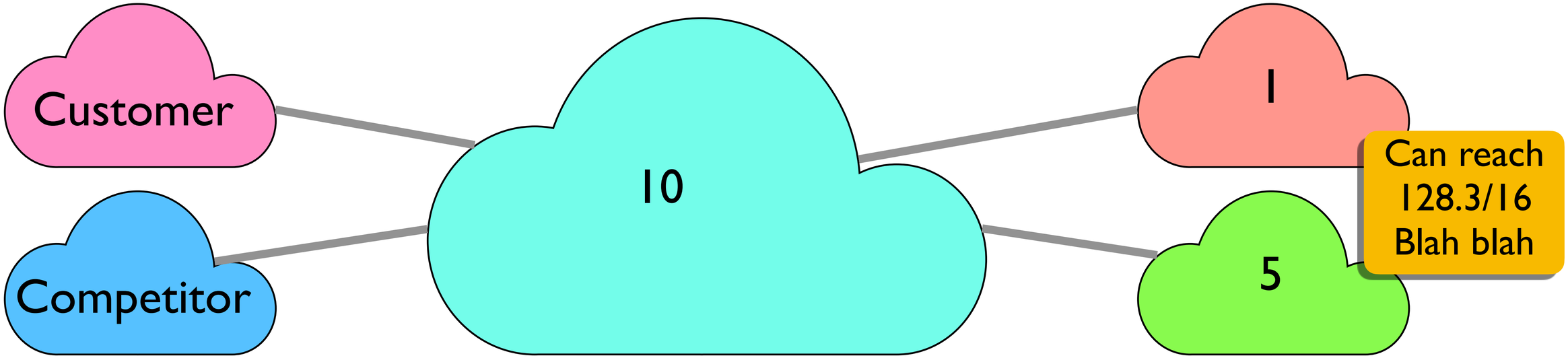
- **With some important differences**
  - Routes selected based on policy, not just shorted path
  - Path vector (useful to avoid loops)
  - Selective route advertisement
  - May aggregate routes (aggregating prefixes)

Policy imposed in how routes are
selected and exported

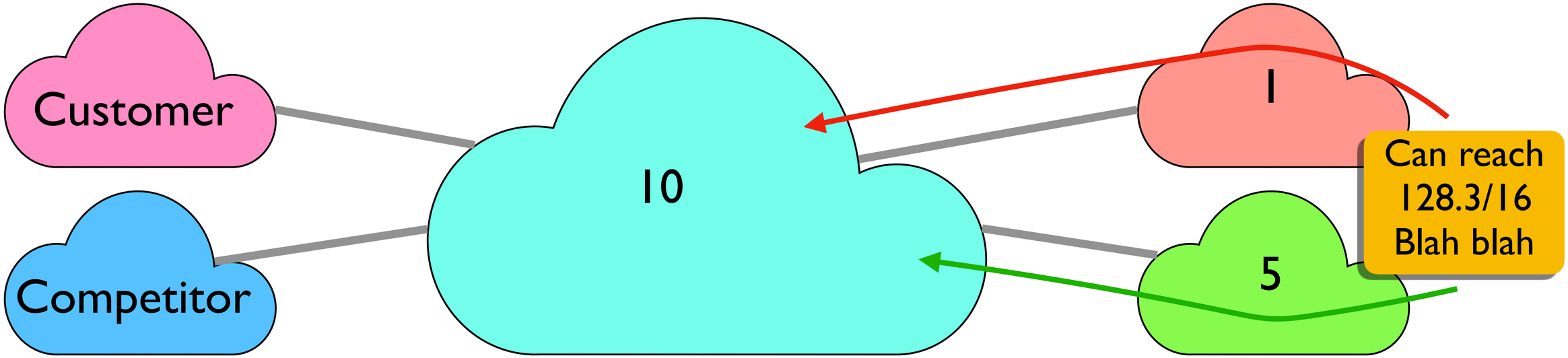
Policy imposed in how routes are **selected and exported**



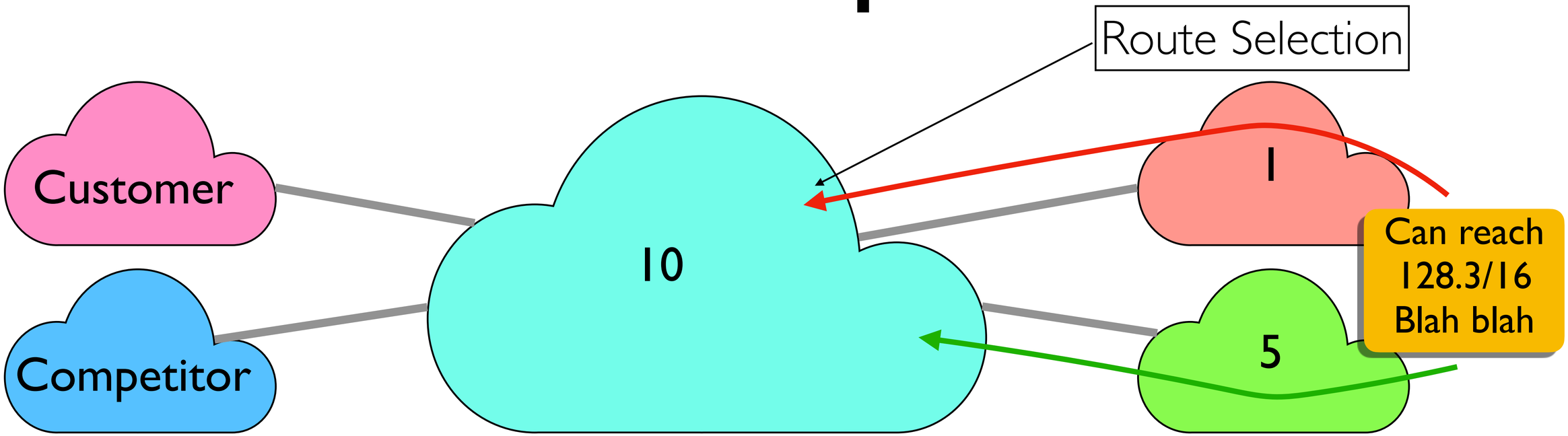
Policy imposed in how routes are **selected and exported**



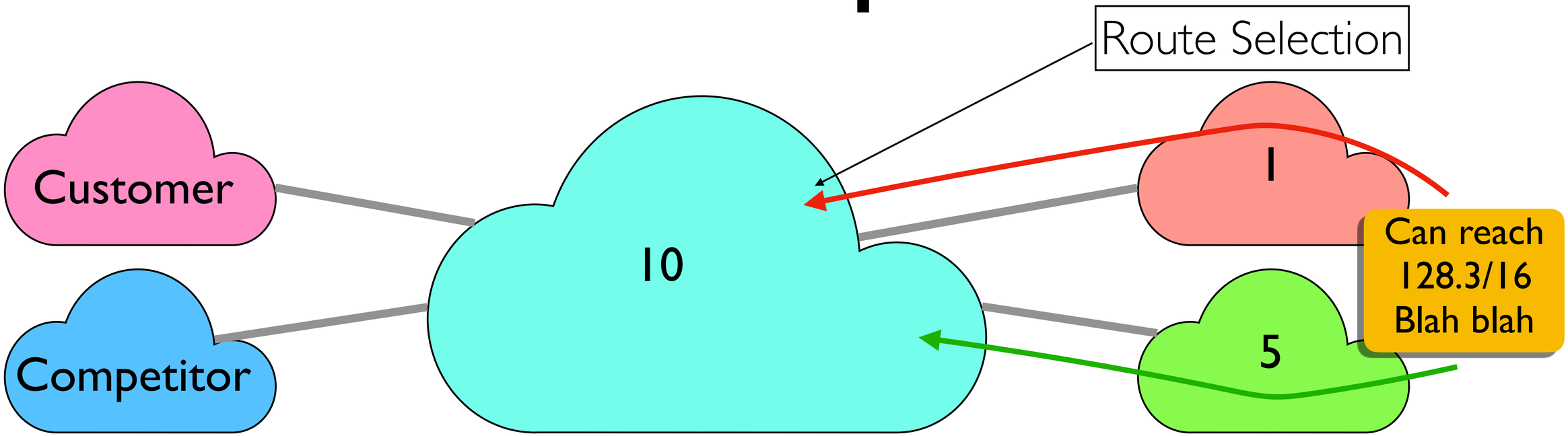
Policy imposed in how routes are **selected and exported**



Policy imposed in how routes are **selected and exported**

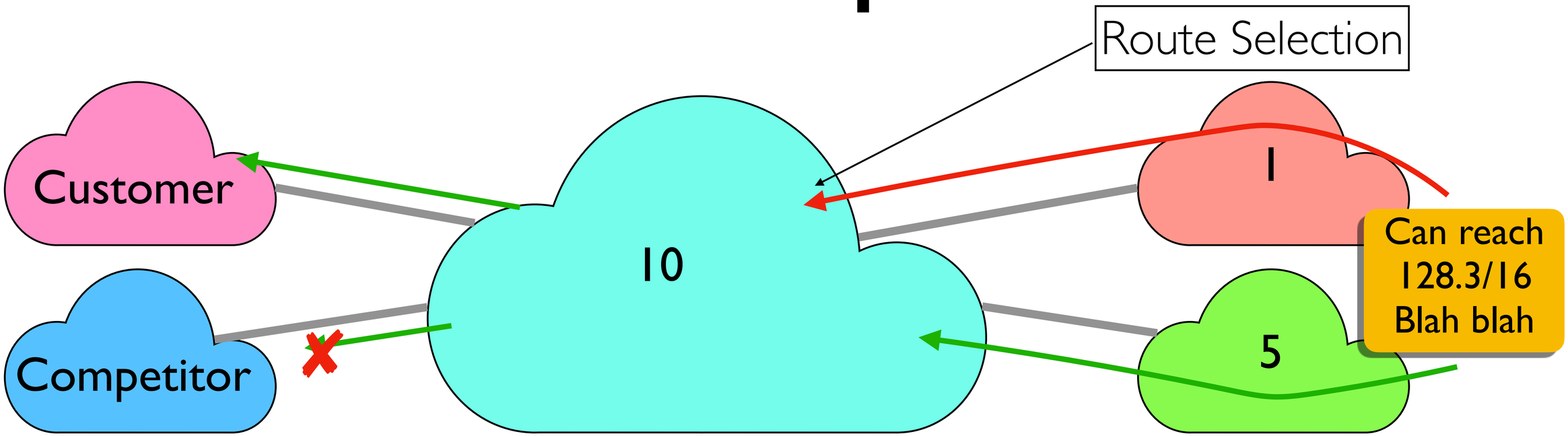


Policy imposed in how routes are **selected and exported**



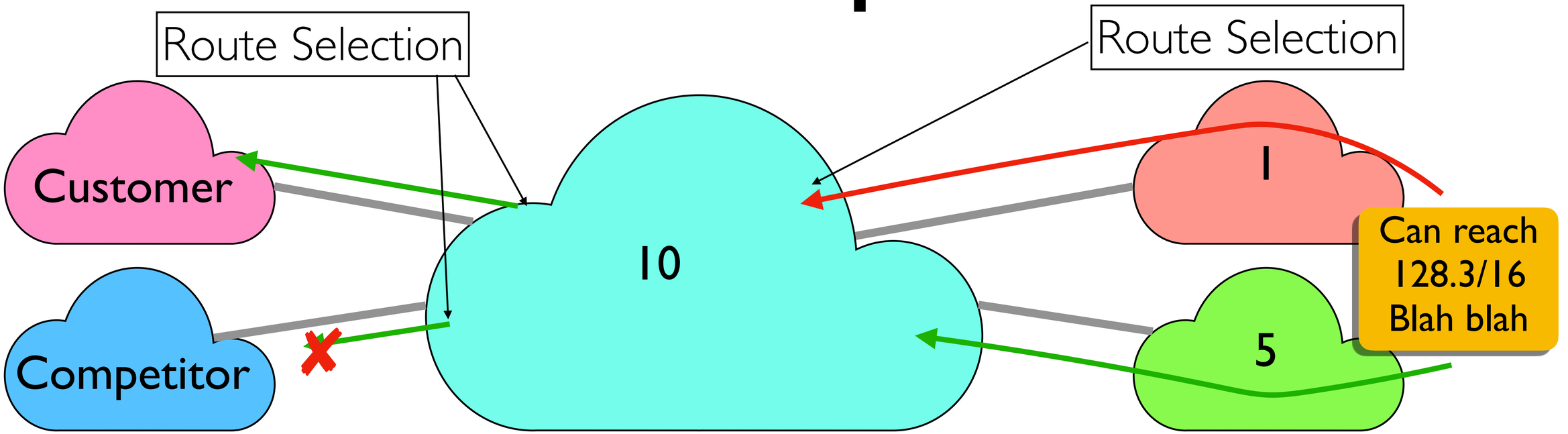
- **Selection:** Which path to use?
  - Controls whether/how traffic **leaves** the network

Policy imposed in how routes are **selected and exported**



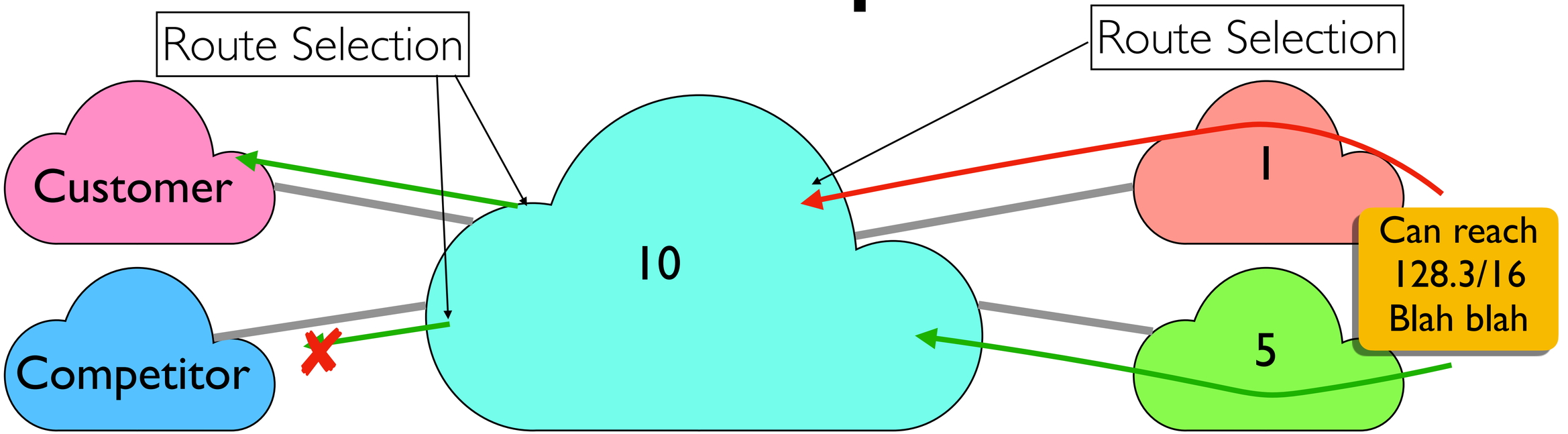
- **Selection:** Which path to use?
  - Controls whether/how traffic **leaves** the network

Policy imposed in how routes are **selected and exported**



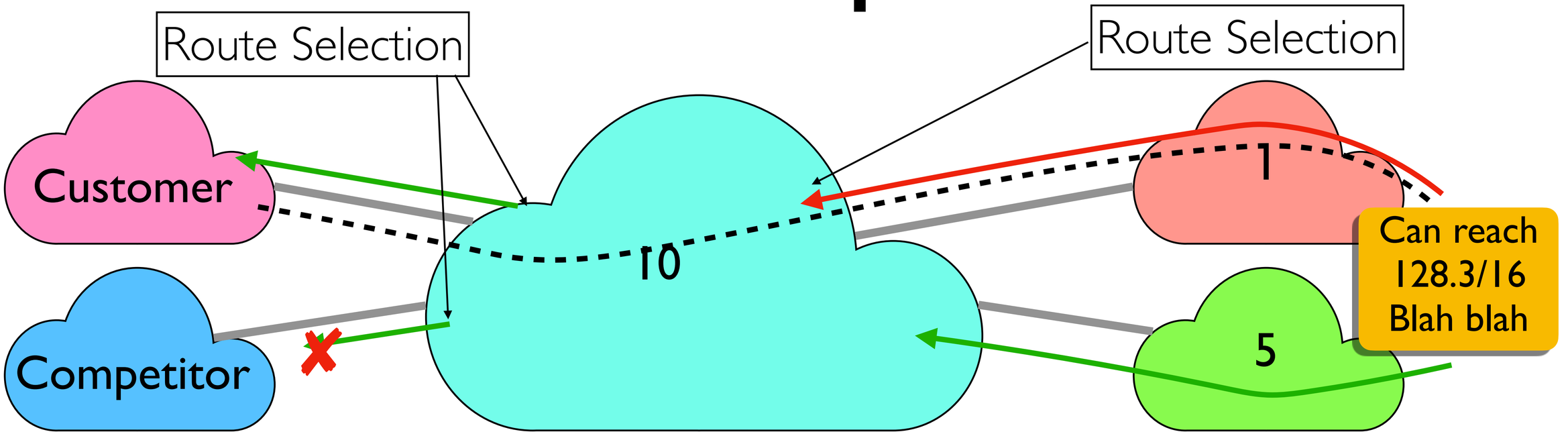
- **Selection:** Which path to use?
  - Controls whether/how traffic **leaves** the network

Policy imposed in how routes are **selected and exported**



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

Policy imposed in how routes are **selected and exported**



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

Typical Export Policy

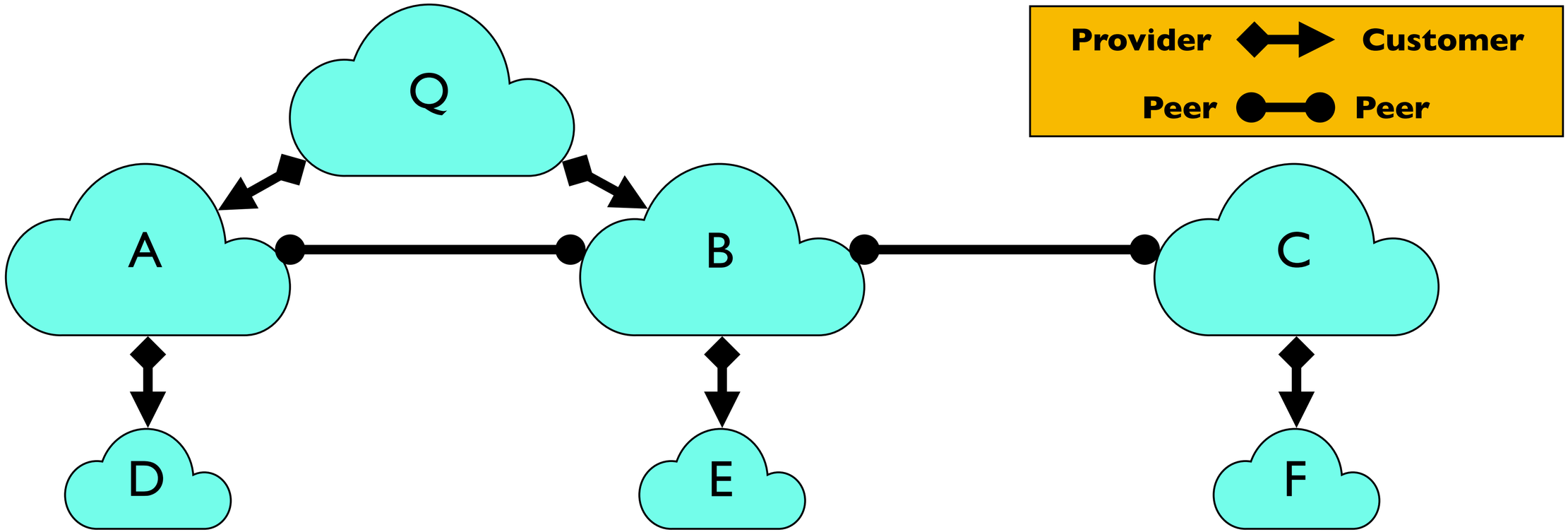
Destination prefix advertised by…	Export route to…
Customer	Everyone (providers, peers, other customers)
Peer	Customers
Provider	Customers

We'll refer to these as the "Gao-Rexford" rules
(Capture common — but not required! — practice)

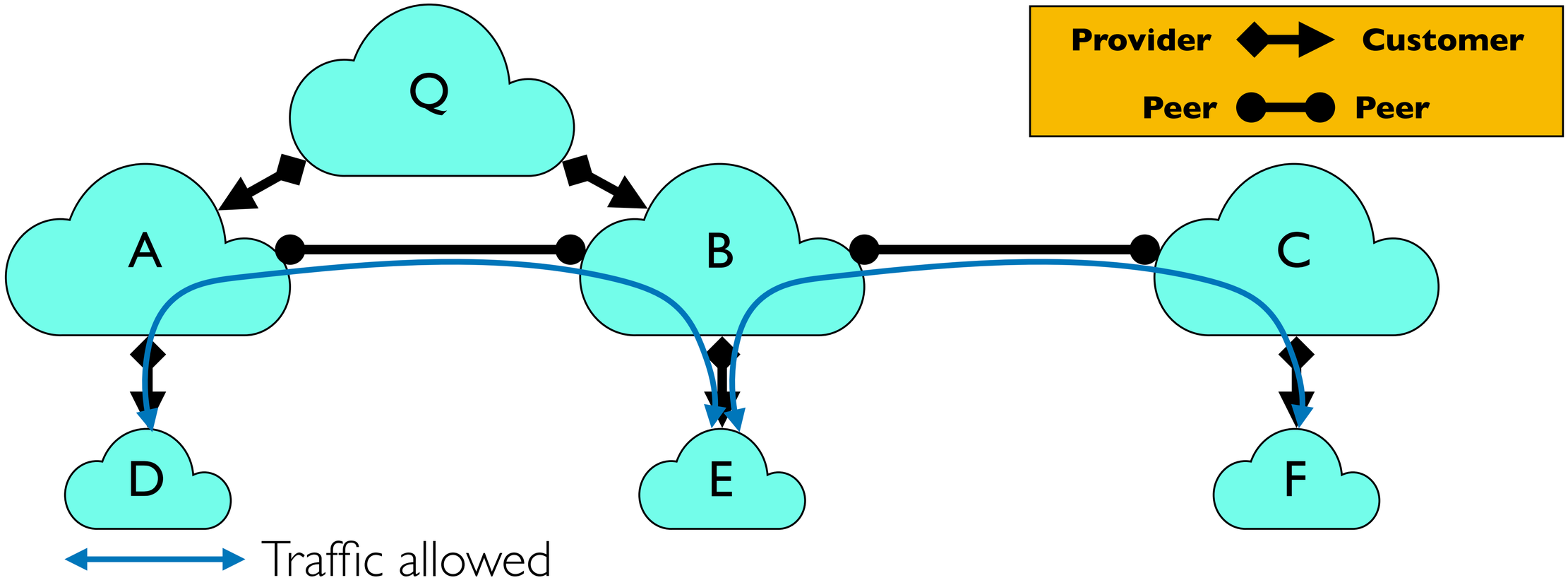
Typical Selection Policy

- **In decreasing order of priority**
  - Make/save money (send to customer > peer > provider)
  - Maximize performance (small AS path length)
  - Minimize use of my network bandwidth ("hot potato")
  - …
- **BGP uses route attributes to implement the above**
  - ASPATH, LOCAL_PATH, MED, …

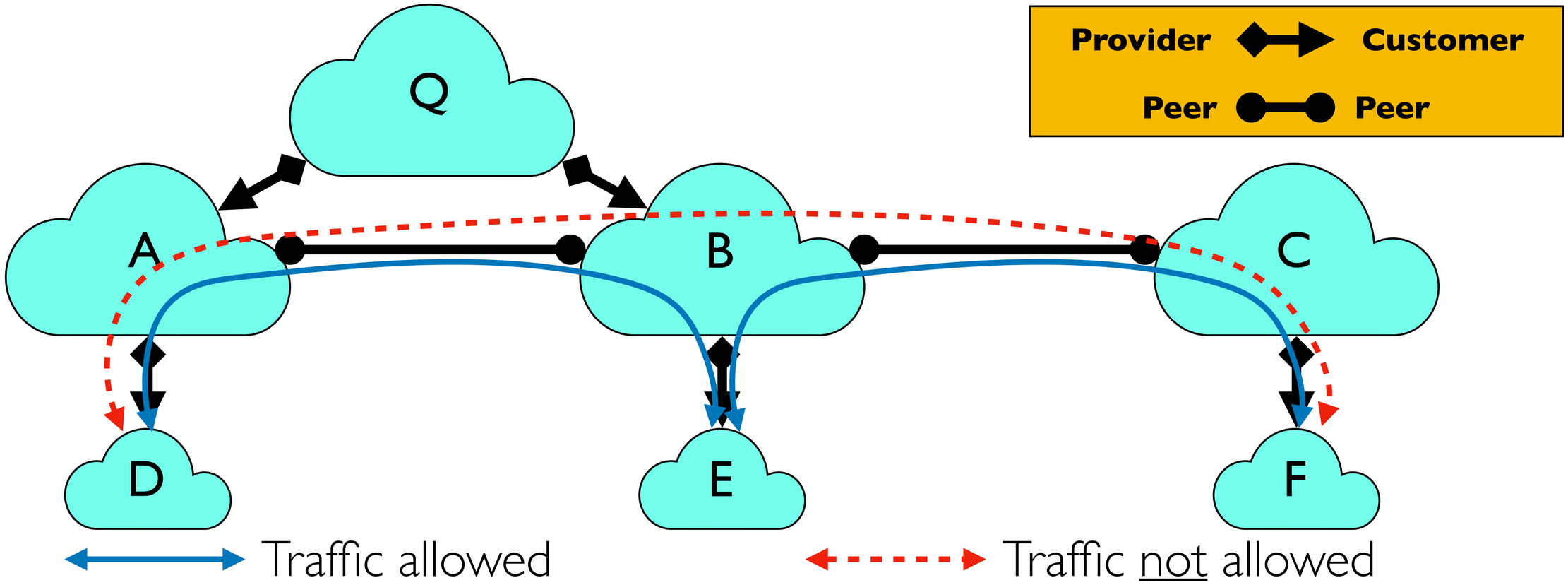
Policy Dictates Route Selection



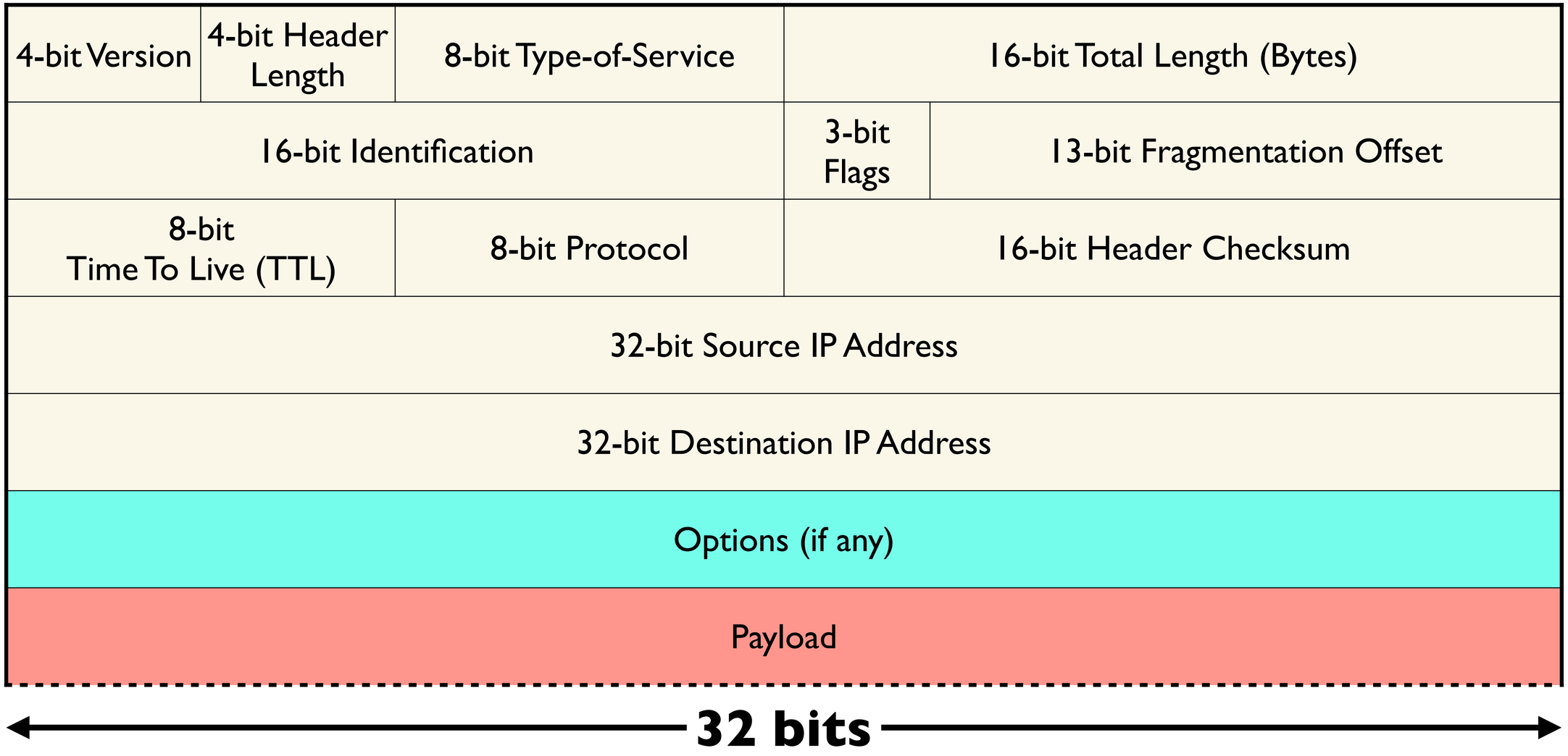
Policy Dictates Route Selection



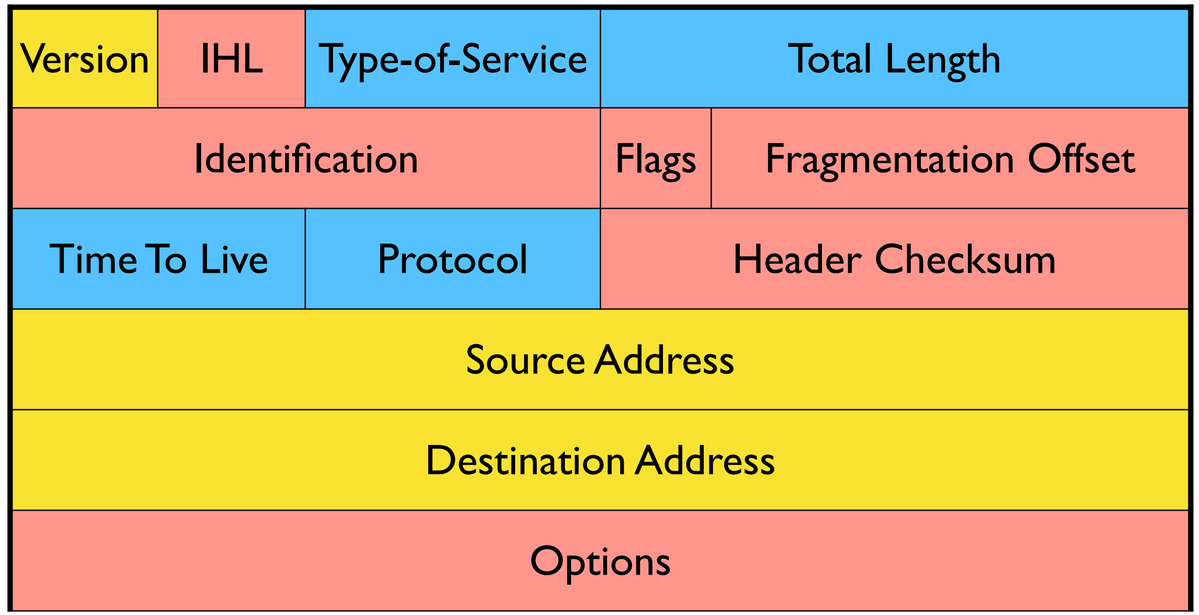
Policy Dictates Route Selection

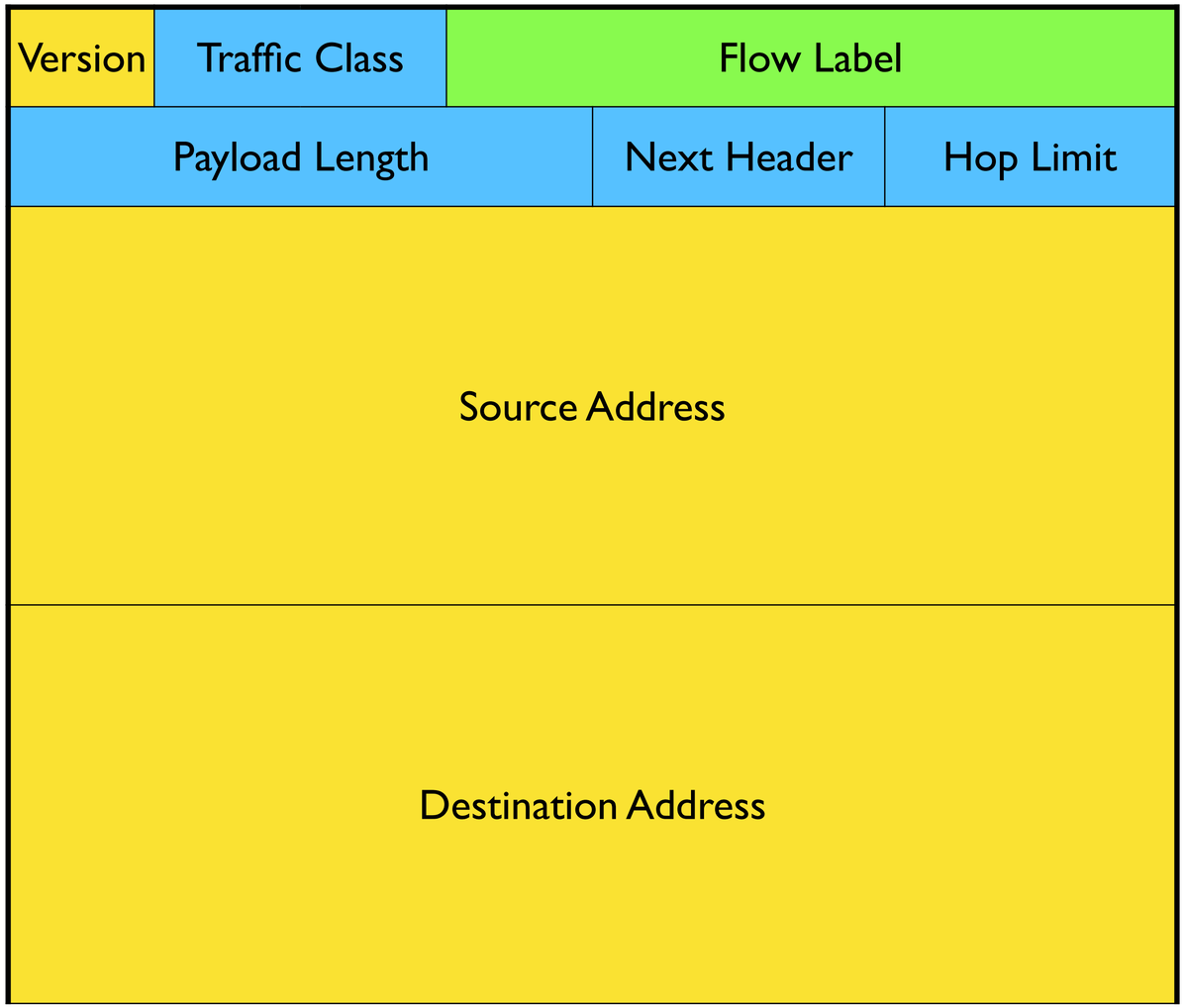


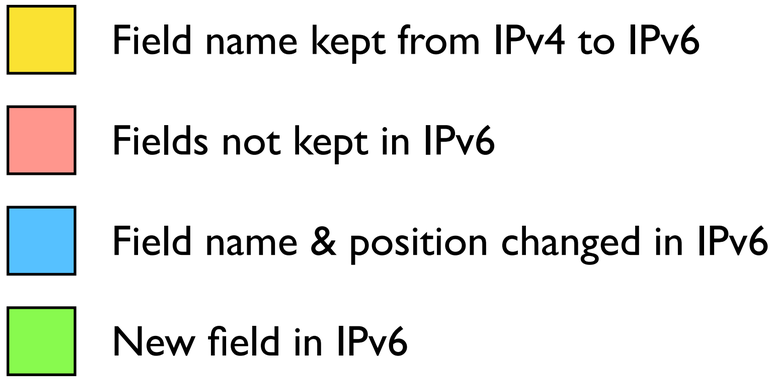
IP Packet Structure



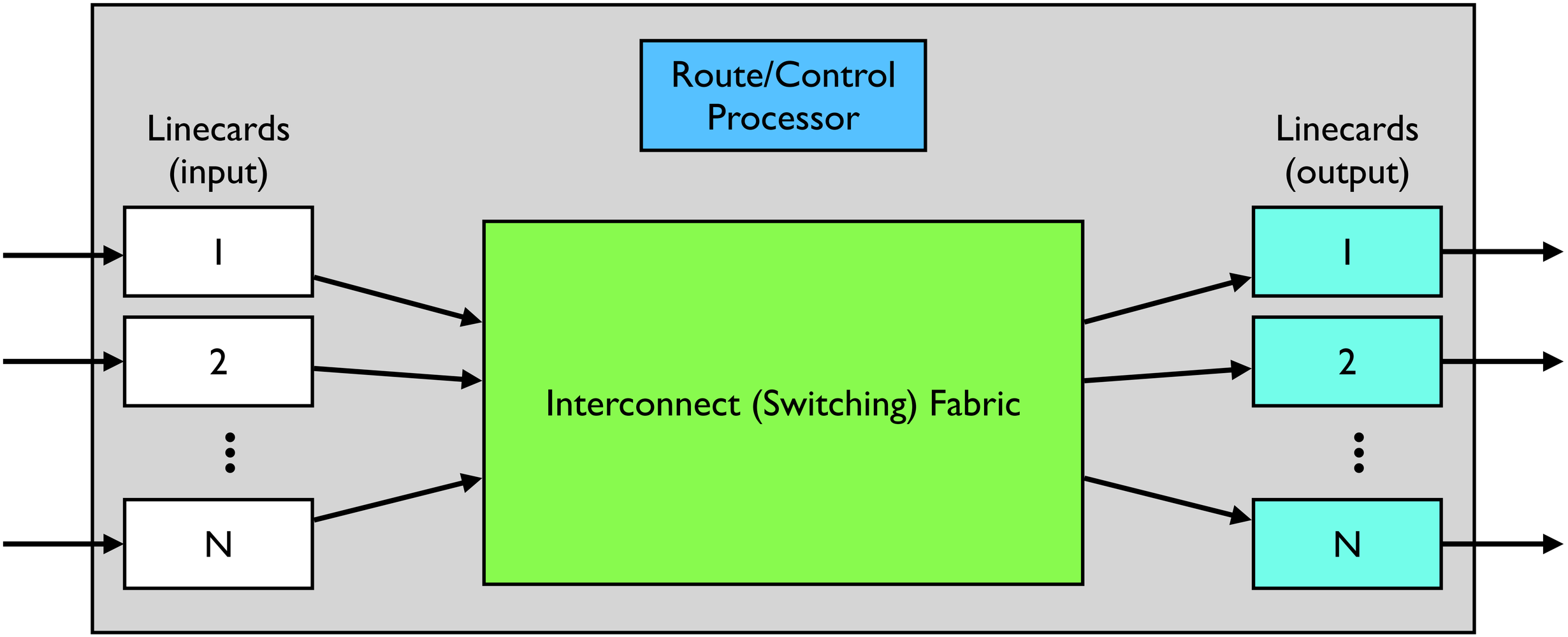
IPv4 and IPv6 Header Comparison



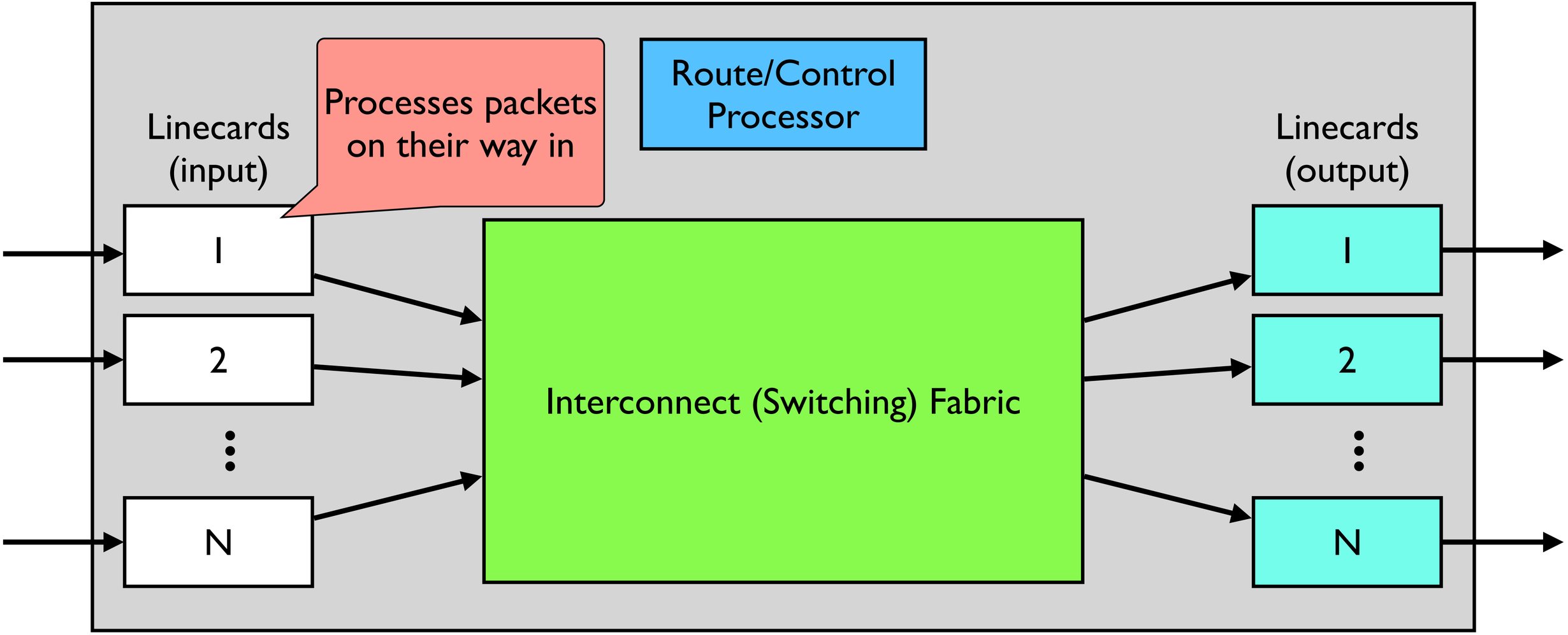




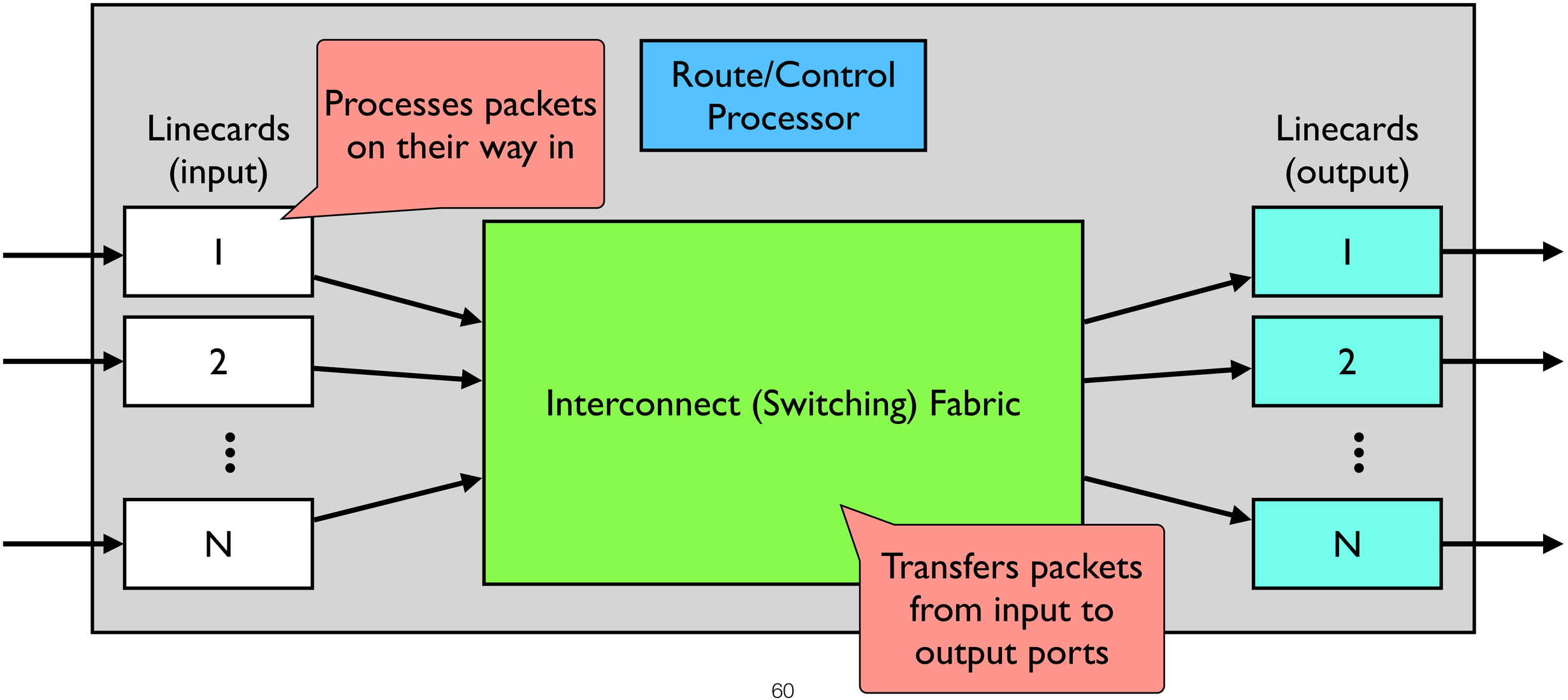
What's inside a router?



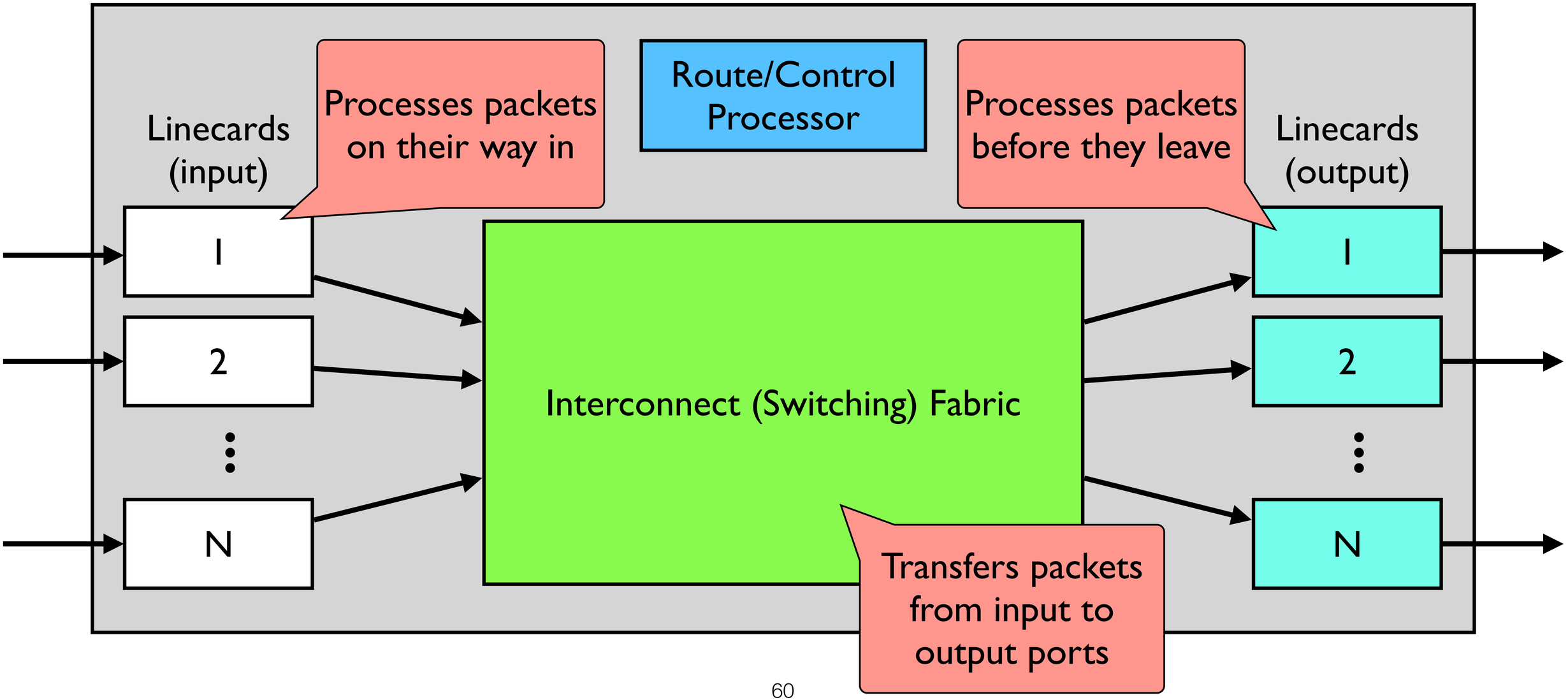
What's inside a router?



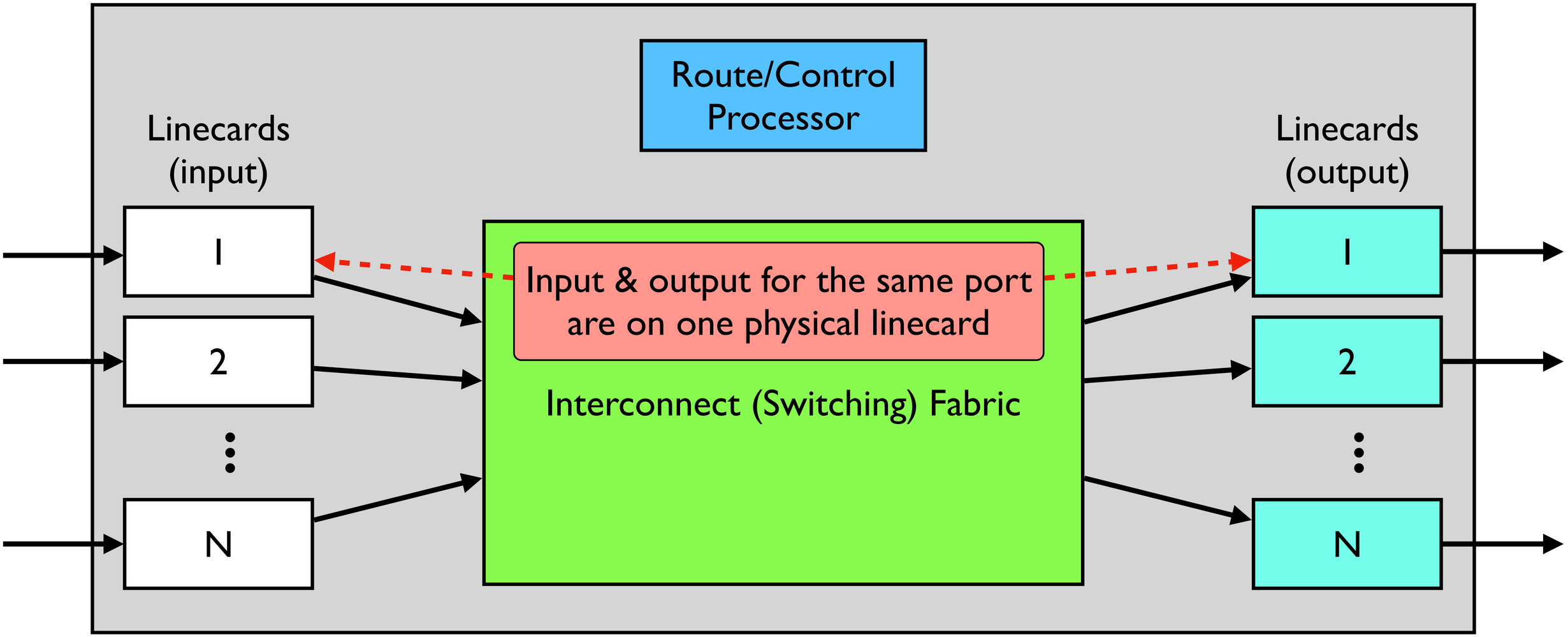
What's inside a router?



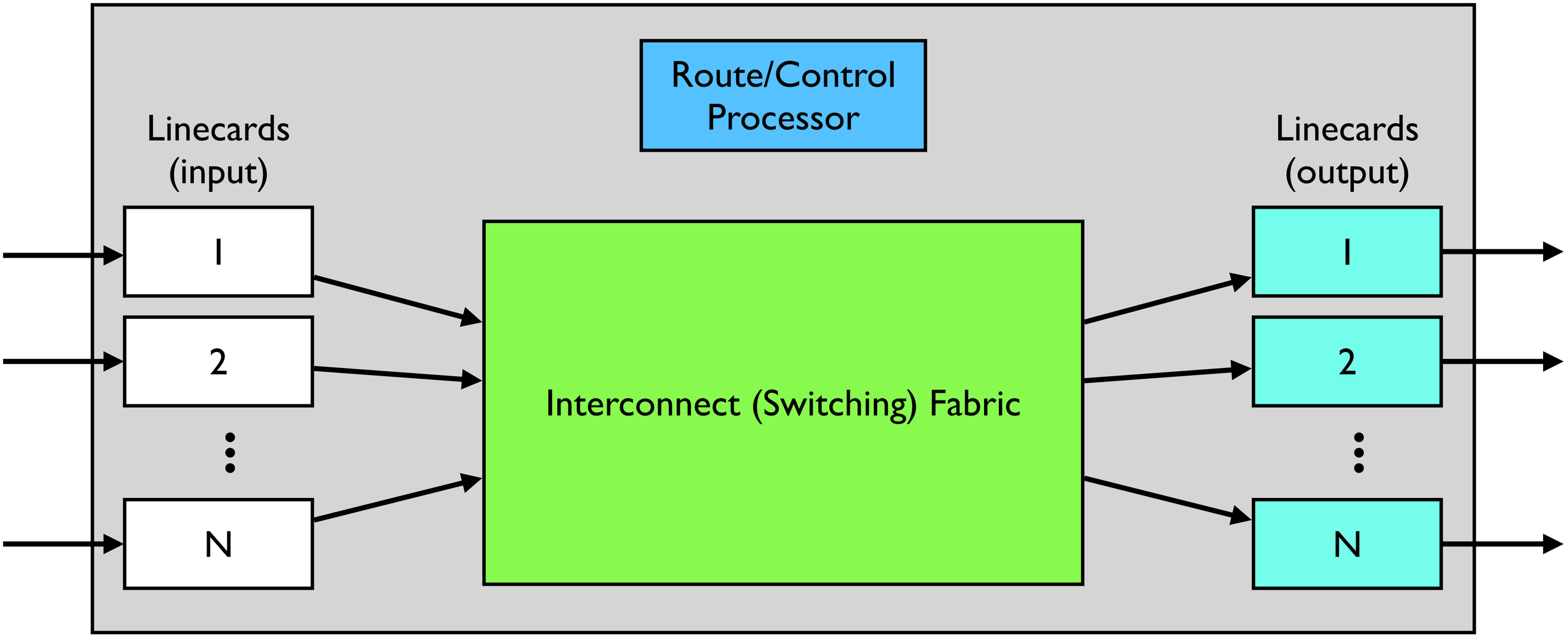
What's inside a router?



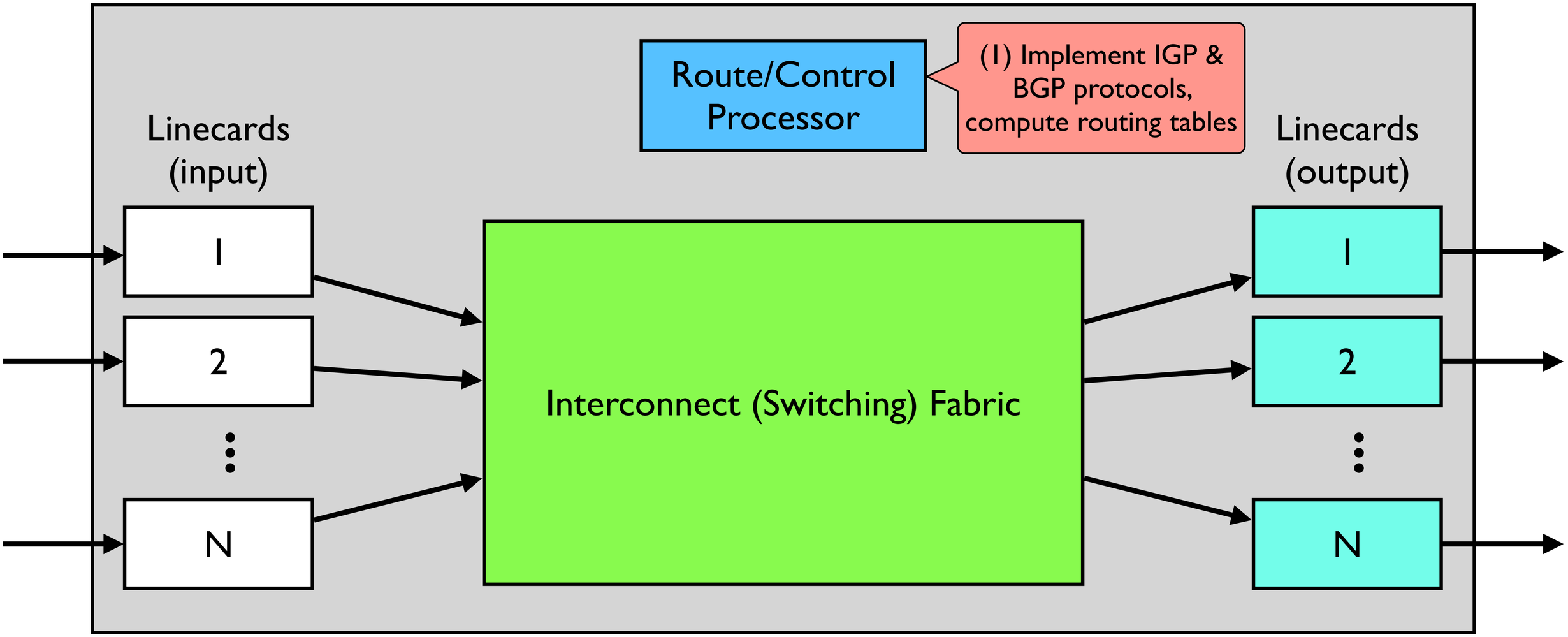
What's inside a router?



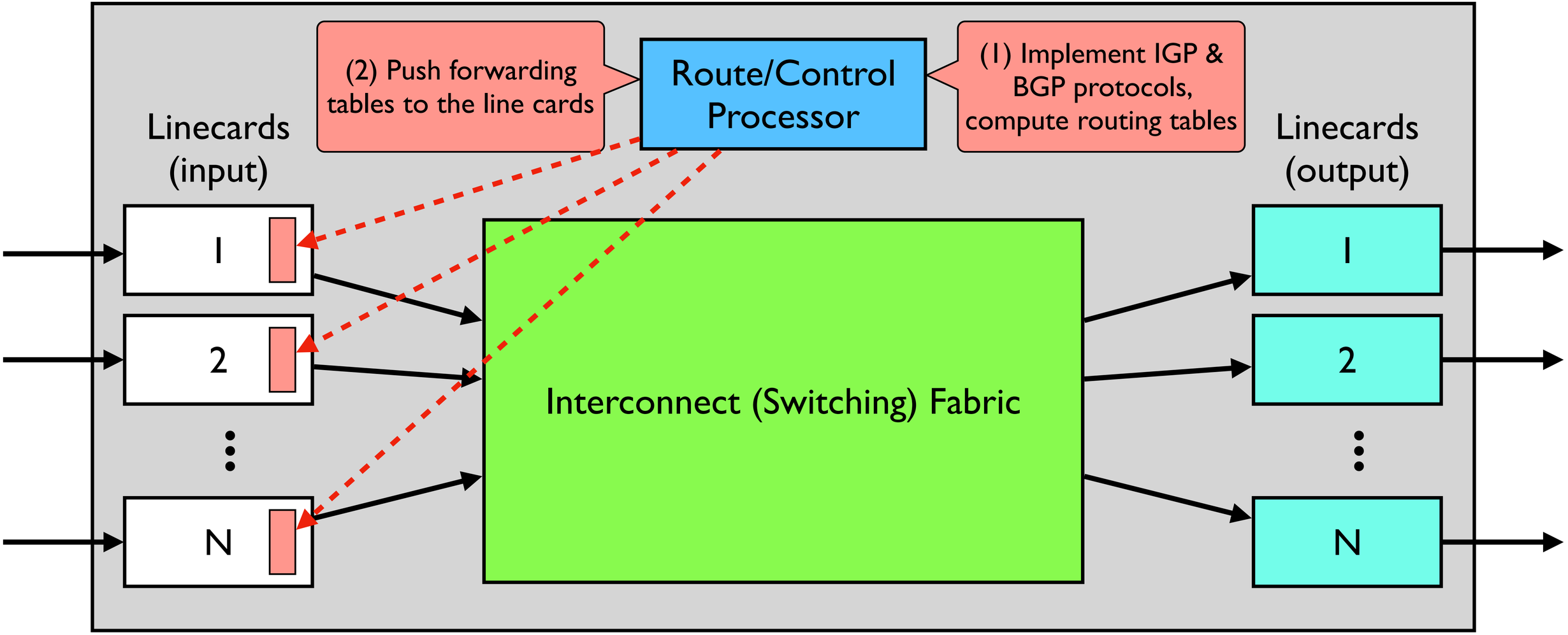
What's inside a router?



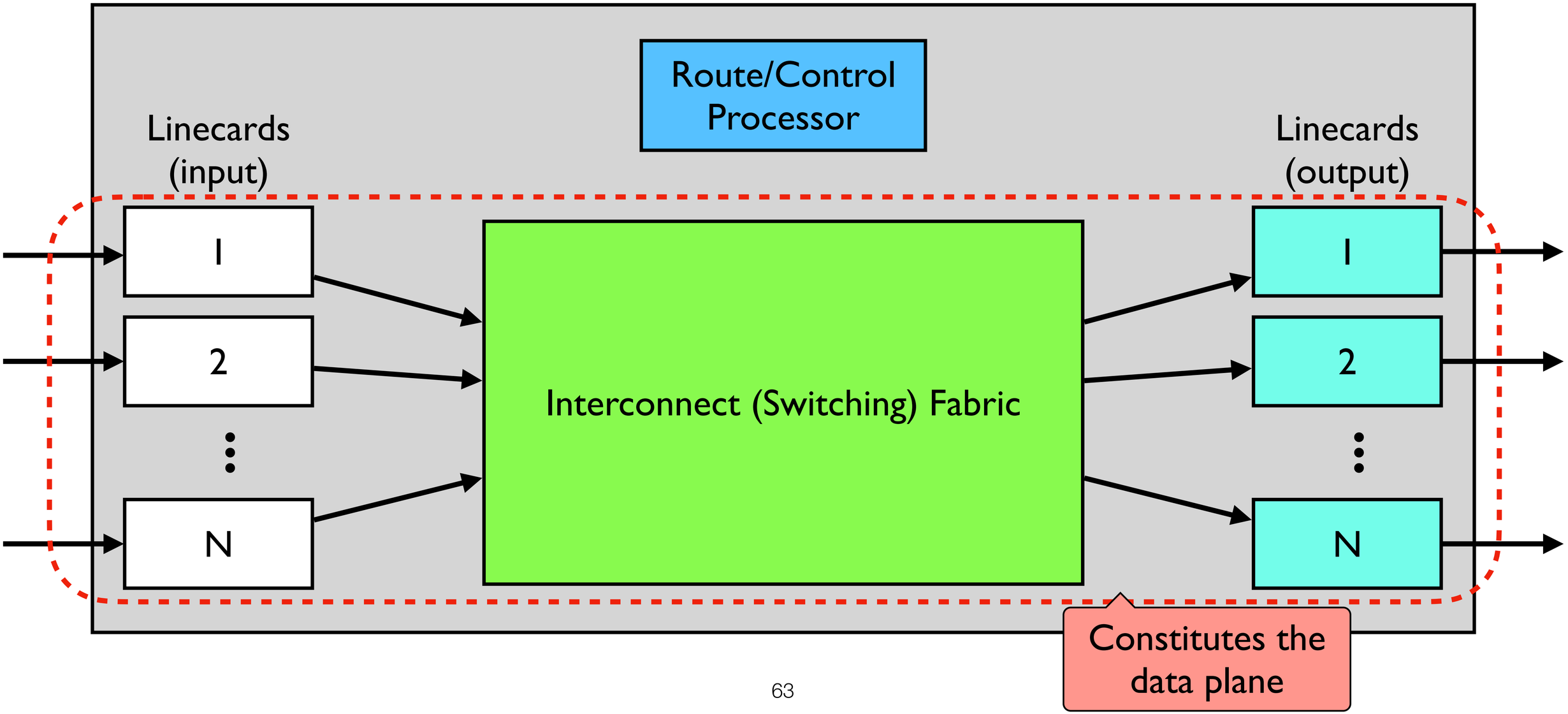
What's inside a router?



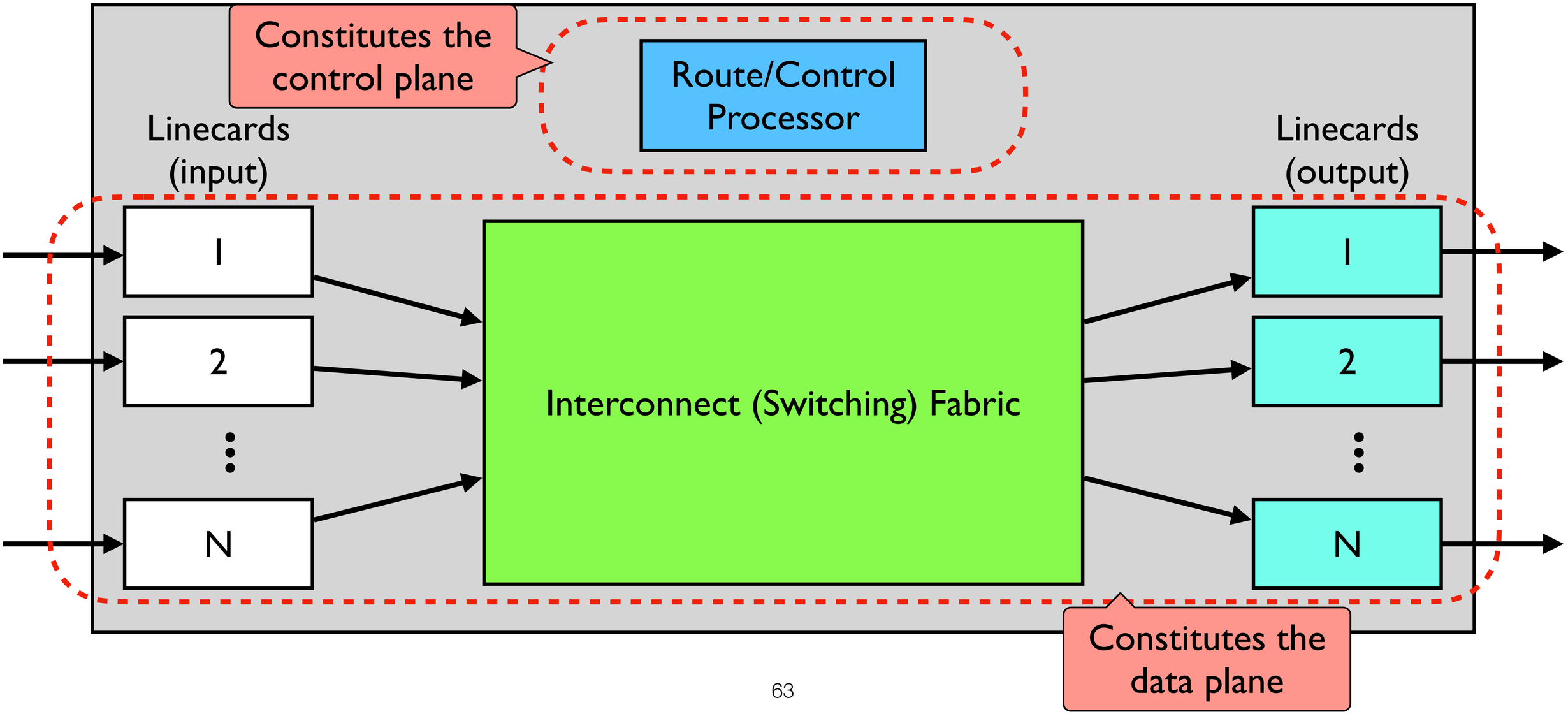
What's inside a router?



What's inside a router?



What's inside a router?



Challenges in Router Design

- **@ Line cards: Destination lookups at high speed**
  - E.g., find the longest prefix match (LPM) in the table that matches the packet destination address
- **@ Switch fabric: head-of-line blocking, scheduling the switch fabric at high speed**
- **@ Route processor: Complexity/correctness more a problem than performance**

Transport Layer

Role of the Transport Layer

- **Communication between application processes**
  - Multiplex and demultiplex from/to application processes
  - Implemented using ***ports*** (not the same as router ports!)
- **Provide common end-to-end services for application layer**
  - Reliable, in-order data delivery
  - Well-placed data delivery

UDP vs. TCP

- **Both UDP and TCP provide multiplexing/demultiplexing via ports**

	UDP	TCP
Data Abstraction	Packet (datagrams)	Byte-stream of arbitrary length
Service	Best-effort (same as IP)	Reliability, In-order delivery, Flow control, Congestion control
Applications	Video, audio streaming	File transfer, chat

Reliable Transport: General Concepts

- Checksums (for error detection)
- Timers (for loss detection)
- Acknowledgements (feedback from receiver)
  - Cumulative: "Received everything up to X"
  - Selective: "Received X"
- Sequence numbers (detect duplicates, accounting)
- Sliding windows (for efficiency)

Reliable Transport: General Concepts

- Checksums (for error detection)
- Timers (for loss detection)
- Acknowledgements (feedback from receiver)
  - Cumulative: "Received everything up to X"
  - Selective: "Received X"
- Sequence numbers (detect duplicates, accounting)
- Sliding windows (for efficiency)

You should know:
- What these concepts are
- Why they exist
- How TCP uses them

Things to know about TCP

- How TCP achieves reliability
- RTT estimation
- Connection establishment/teardown
- Flow Control
- Congestion Control
- Critiques on Congestion Control, Router-based Approaches

- For each, know the *how* the functionality is implemented, and *why* it is needed

E.g., RTT Estimation

E.g., RTT Estimation

- Why? TCP uses timeouts to retransmit packets

E.g., RTT Estimation

- **Why? TCP uses timeouts to retransmit packets**
- **But RTT may vary (significantly!) for different reasons and different timescales**
  - Due to temporary congestion
  - Due to long-lived congestion
  - Due to change in routing paths

E.g., RTT Estimation

- Why? TCP uses timeouts to retransmit packets
- But RTT may vary (significantly!) for different reasons and different timescales
  - Due to temporary congestion
  - Due to long-lived congestion
  - Due to change in routing paths
- An incorrect RTT estimate might introduce spurious retransmissions or overly long delays

E.g., RTT Estimation

- **Why? TCP uses timeouts to retransmit packets**
- **But RTT may vary (significantly!) for different reasons and different timescales**
  - Due to temporary congestion
  - Due to long-lived congestion
  - Due to change in routing paths
- **An incorrect RTT estimate might introduce spurious retransmissions or overly long delays**
- **RTT estimators should react to change, but not too quickly**
  - Proposed solutions used EWMA, incorporate deviations

E.g., Reliability

E.g., Reliability

- **Why? IP is best-effort but many applications need reliable delivery**
  - Having TCP take care of it simplifies application development!

E.g., Reliability

- **Why? IP is best-effort but many applications need reliable delivery**
  - Having TCP take care of it simplifies application development!
- **How?**
  - Checksums & timers (for error and loss detection)
  - Fast retransmit (for faster-than-timeout loss detection)
  - Cumulative ACKs (feedback from receiver — what's lost/what's not)
  - Sliding windows (for efficiency)
  - Buffers at sender (to hold packets while waiting for ACKs)
  - Buffers at receiver (to reorder packets before delivery to application)

E.g., Connection Establishment

E.g., Connection Establishment

- **Why?**
  - TCP is a stateful protocol (CWND, buffer-space, ISN, etc.)
  - Need to initialize connection state at both ends
  - Exchange initial sequence numbers

E.g., Connection Establishment

- **Why?**
  - TCP is a stateful protocol (CWND, buffer-space, ISN, etc.)
  - Need to initialize connection state at both ends
  - Exchange initial sequence numbers
- **How? Three-way handshake**
  - Host A sends a SYN to host B
  - Host B returns a SYN acknowledgement (SYN ACK)
  - Host A sends an ACK (+data) to acknowledge the SYN ACK
  - Hosts exchange proposed Initial Sequence Numbers at each step

E.g., Flow Control

E.g., Flow Control

- **Why?**
  - TCP offers a reliable in-order byte-stream abstraction
  - Hence, TCP at the receiver must buffer a packet until all packets before it (in byte-order) have arrived and the receiving application has consumed available bytes
  - Hence, receiver advances its window when the receiving application consumes data
  - But, sender advances its window when new data is ACKed
  - Hence, risk the sender might overrun the receiver's buffers

E.g., Flow Control

- **Why?**
  - TCP offers a reliable in-order byte-stream abstraction
  - Hence, TCP at the receiver must buffer a packet until all packets before it (in byte-order) have arrived and the receiving application has consumed available bytes
  - Hence, receiver advances its window when the receiving application consumes data
  - But, sender advances its window when new data is ACKed
  - Hence, risk the sender might overrun the receiver's buffers
- **How? "Advertised Window" field in TCP header**
  - Receiver *advertises* the "right hand edge" of its window to sender
  - Sender agrees not to exceed the amount

E.g., Congestion Control

E.g., Congestion Control

- **Why?**
  - Because a sender shouldn't overload the network itself
  - But, should make efficient use of available network capacity
  - While sharing the available capacity fairly with other flows
  - And adapting to changes in available capacity

E.g., Congestion Control

- **Why?**
  - Because a sender shouldn't overload the network itself
  - But, should make efficient use of available network capacity
  - While sharing the available capacity fairly with other flows
  - And adapting to changes in available capacity
- **How?**
  - Quickly find current available capacity (slow start)
  - Then adapt to changes (congestion avoidance, AIMD)
  - With optimizations (fast recovery)
  - *Study the TCP state machine diagram from the text!*

Advanced CC and Fairness

Advanced CC and Fairness

- **TCP Throughput Equation**
  - Know the equation
  - Understand its implications

$$\text{Throughput} = \sqrt{\frac{3}{2}} \cdot \frac{1}{RTT \cdot \sqrt{p}}$$

Advanced CC and Fairness

- **TCP Throughput Equation**
  - Know the equation
  - Understand its implications

$$\text{Throughput} = \sqrt{\frac{3}{2}} \cdot \frac{1}{RTT \cdot \sqrt{p}}$$

- **Max-min fairness / fair queuing**
  - Definition, know how to calculate fair-rate
  - Know pros and cons of FQ (isolation, router complexity)

Advanced CC and Fairness

- **TCP Throughput Equation**
  - Know the equation
  - Understand its implications

$$\text{Throughput} = \sqrt{\frac{3}{2}} \cdot \frac{1}{RTT \cdot \sqrt{p}}$$

- **Max-min fairness / fair queuing**
  - Definition, know how to calculate fair-rate
  - Know pros and cons of FQ (isolation, router complexity)
- **Router assisted congestion control**
  - Pros (better info.) and cons (complexity in routers)
  - E.g. explicit rate allocation (RCP) and explicit congestion notification (ECN)

Final Questions?

- Good luck!