

Sharing & Evaluating Networks (contd.)

CPSC 433/533, Spring 2021
Anurag Khandelwal

Administrivia

Administrivia

- Homework#1 out tonight
 - You will have two weeks to wrap it up
 - The schedule for the rest of the homeworks/projects also available on Canvas

Administrivia

- Homework#1 out tonight
 - You will have two weeks to wrap it up
 - The schedule for the rest of the homeworks/projects also available on Canvas
- Do you feel like you have too much time on your hands?
 - Do you think of yourself as a **h4cker?**
 - **Email me:** I may have research projects for you!

Administrivia

- Homework#1 out tonight
 - You will have two weeks to wrap it up
 - The schedule for the rest of the homeworks/projects also available on Canvas
- Do you feel like you have too much time on your hands?
 - Do you think of yourself as a **h4cker?**
 - **Email me:** I may have research projects for you!
- TF introduction

Ramla Ijaz

- **PhD candidate** @Yale CS
- Advised by **Prof. Lin Zhong**
- Research interests: **OS Design, software correctness, heterogenous computer systems, networking, cloud-RAN**
- Office hours: **4:00pm - 5:00pm ET**
 - Zoom (see Canvas for link)
- **ramla.ijaz@yale.edu**



Recap...

- More details on what & how of networks
- Sharing a network
- Evaluating a network

Recap...

- More details on What & How of the Internet
- Sharing a network
- Evaluating a network

Performance Metrics

Performance Metrics

- **Delay:** How long does it take to send a packet from its source to destination?

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?

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?

Delay

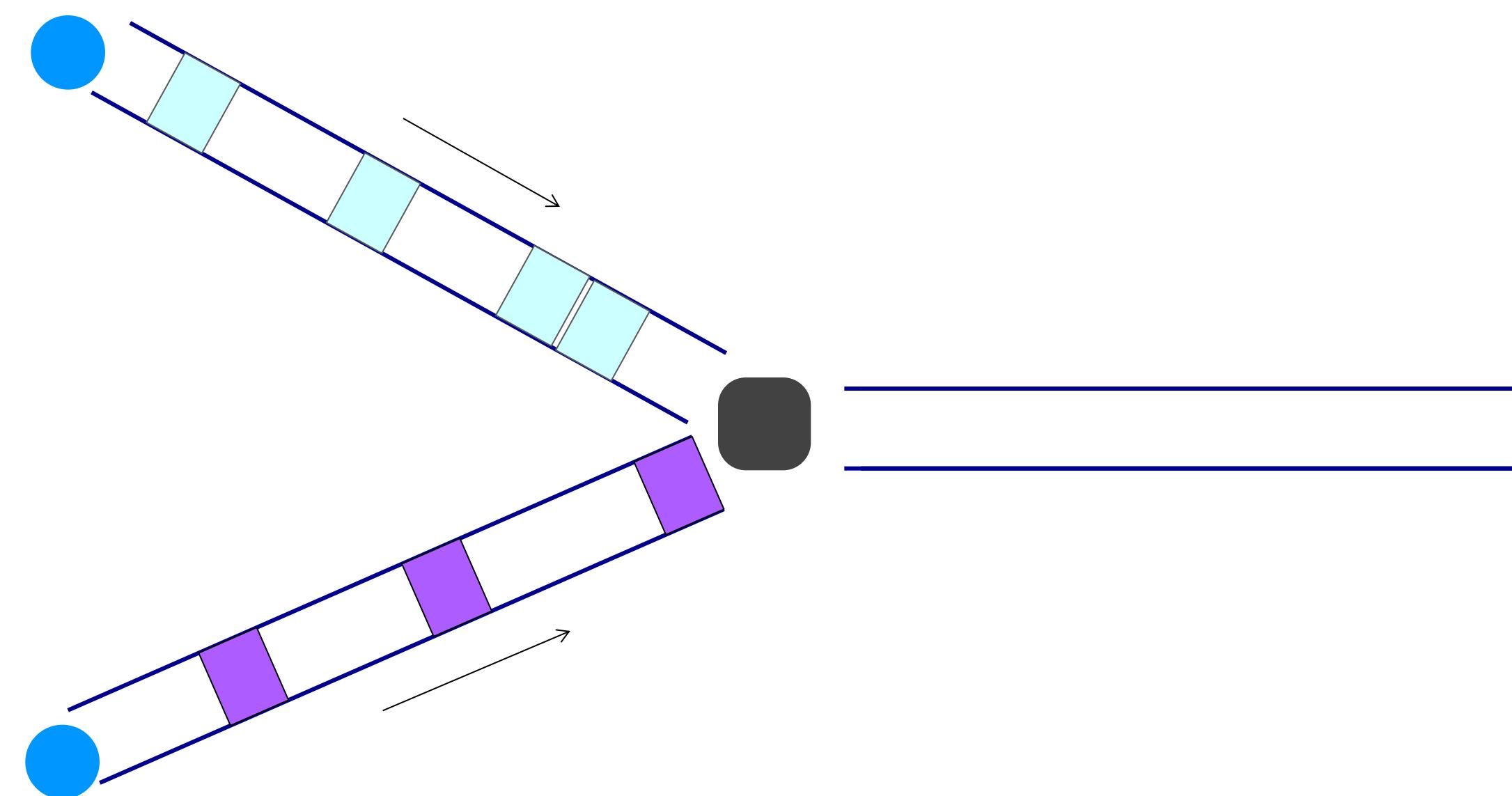
- Consists of four components
 - Transmission Delay
 - Propagation Delay
 - Queueing Delay
 - Processing Delay

Delay

- Consists of four components
 - Transmission Delay
 - Propagation Delay
 - Queueing Delay
 - Processing Delay
-
- The diagram illustrates the four components of delay. It uses brackets to group them into two main categories. The first category, 'Due to link properties', contains 'Transmission Delay' and 'Propagation Delay'. The second category, 'Due to traffic matrix and switch internals', contains 'Queueing Delay' and 'Processing Delay'. This visual grouping helps to distinguish between delays that occur at the physical link level and those that occur within network switches.
- Due to link properties
- Due to traffic matrix and switch internals

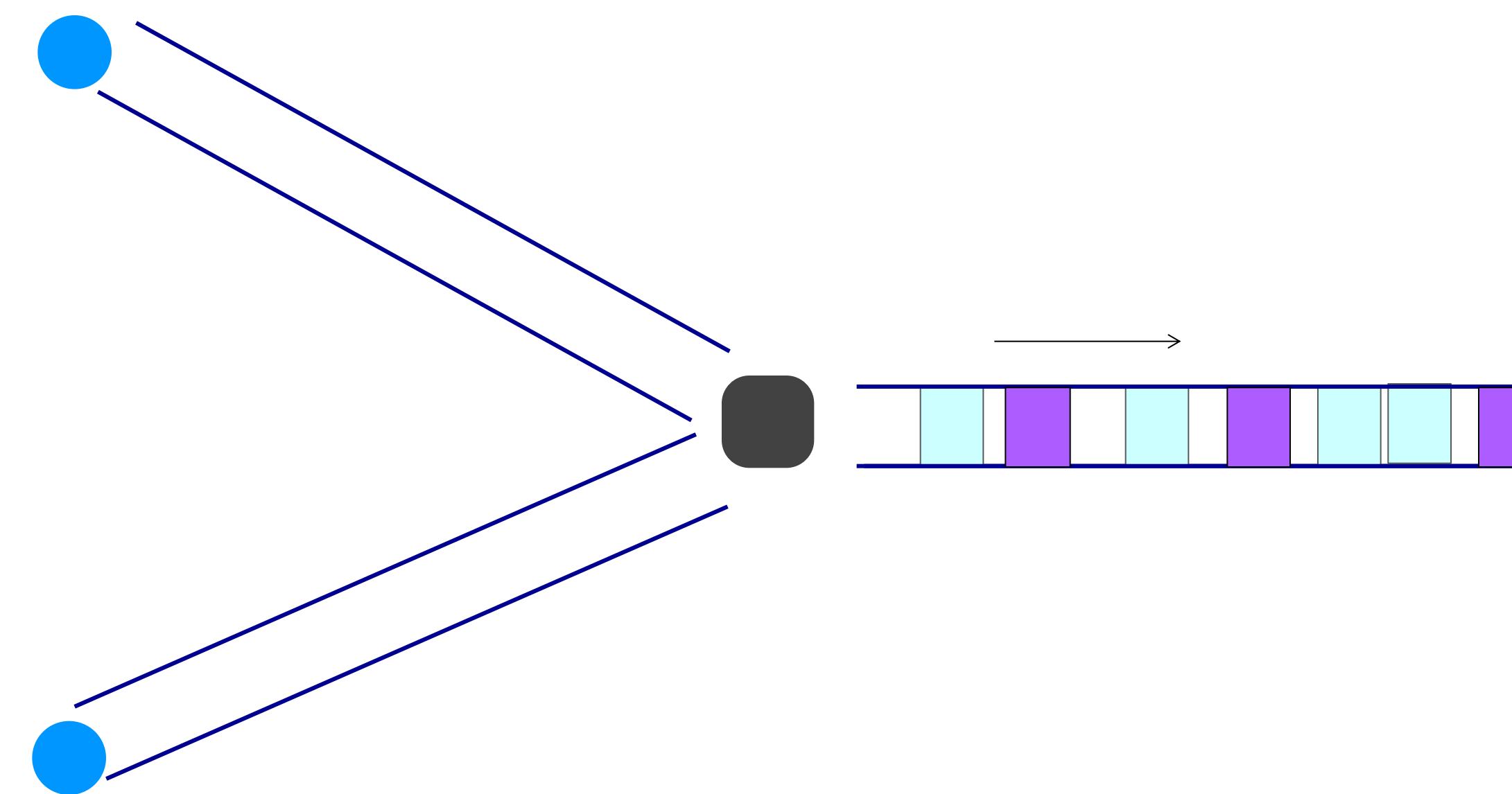
Queueing Delay: “Pipe” View

- How long does a packet have to sit in a buffer before it is processed



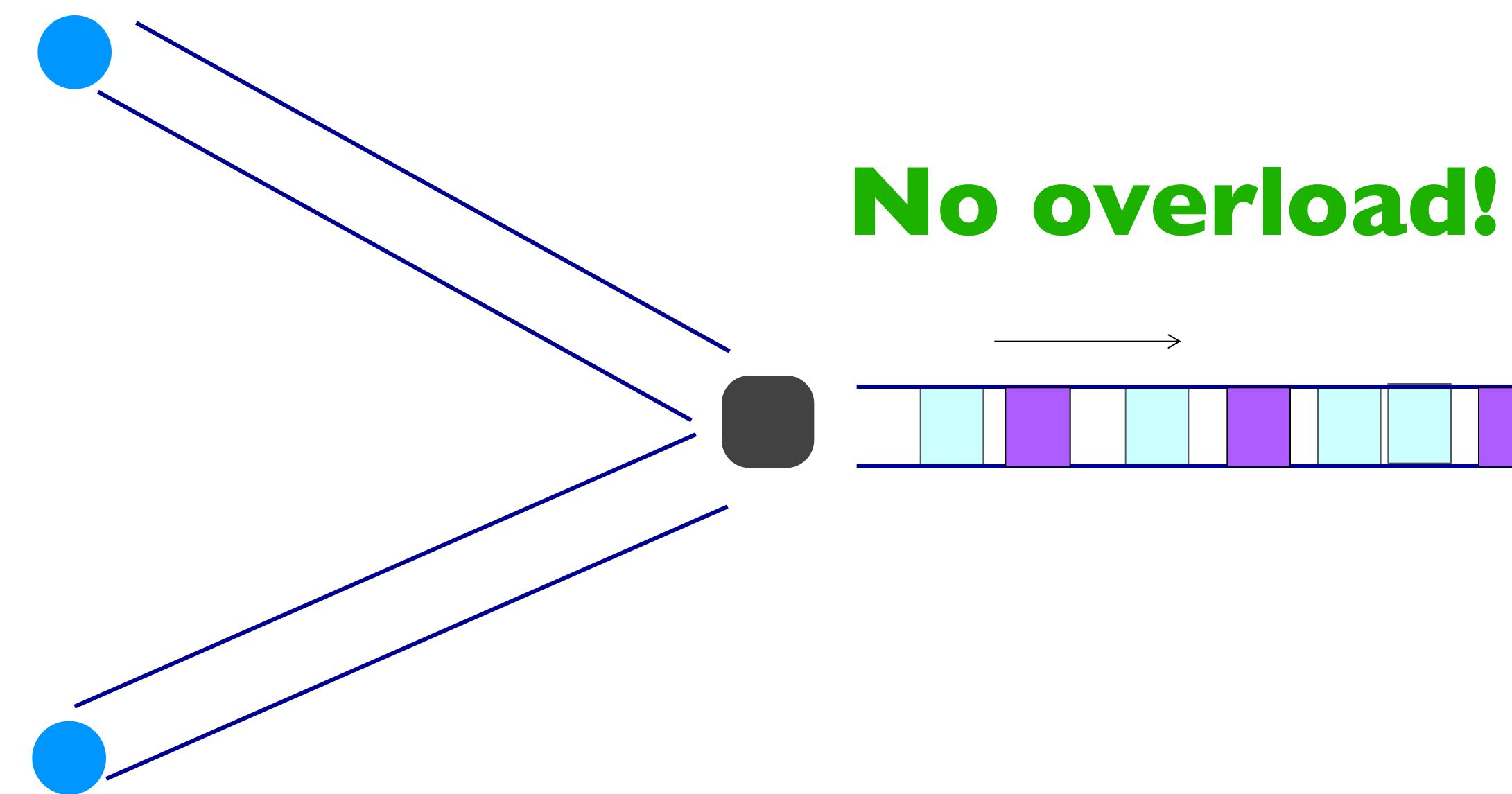
Queueing Delay: “Pipe” View

- How long does a packet have to sit in a buffer before it is processed



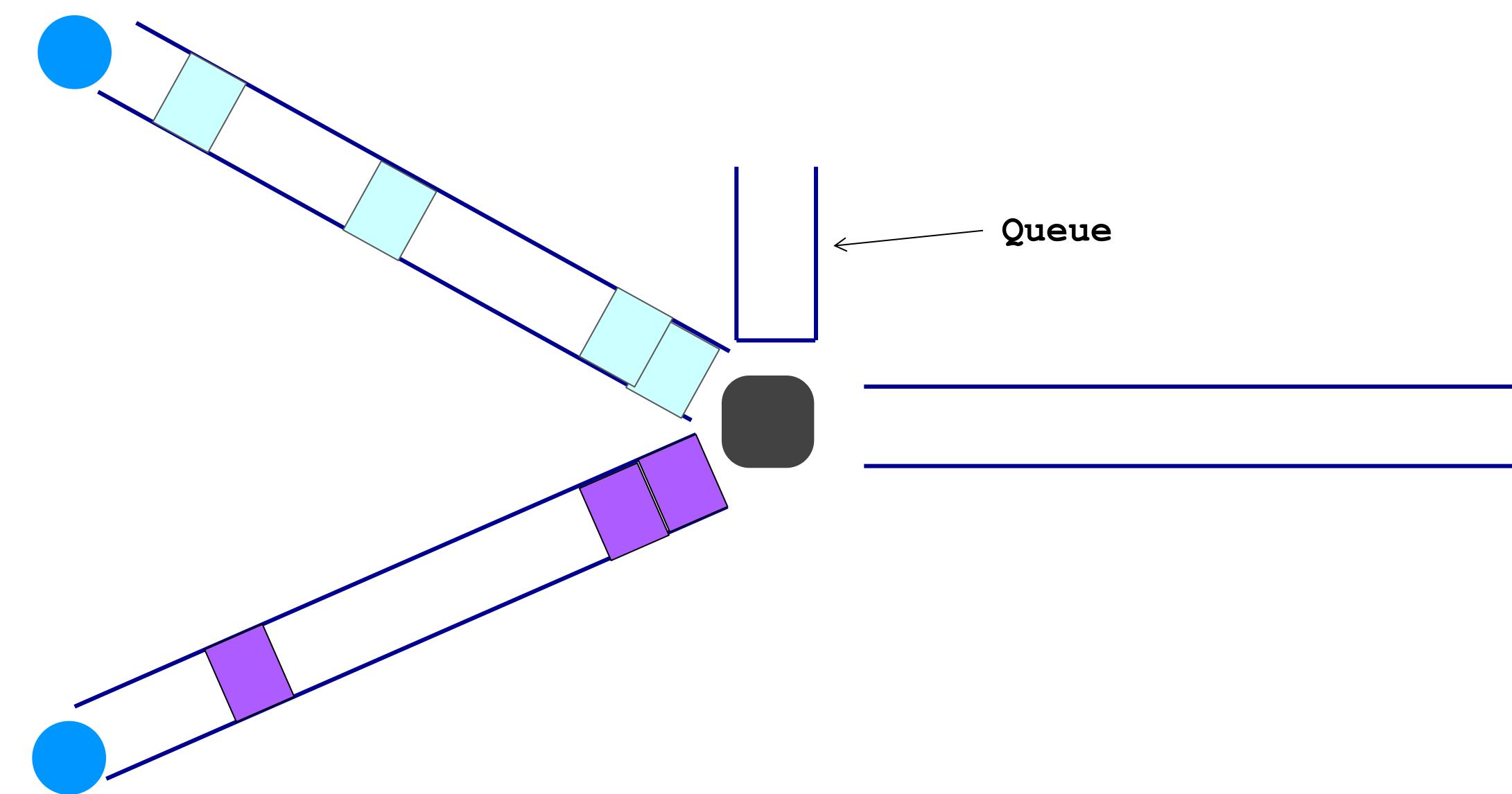
Queueing Delay: “Pipe” View

- How long does a packet have to sit in a buffer before it is processed



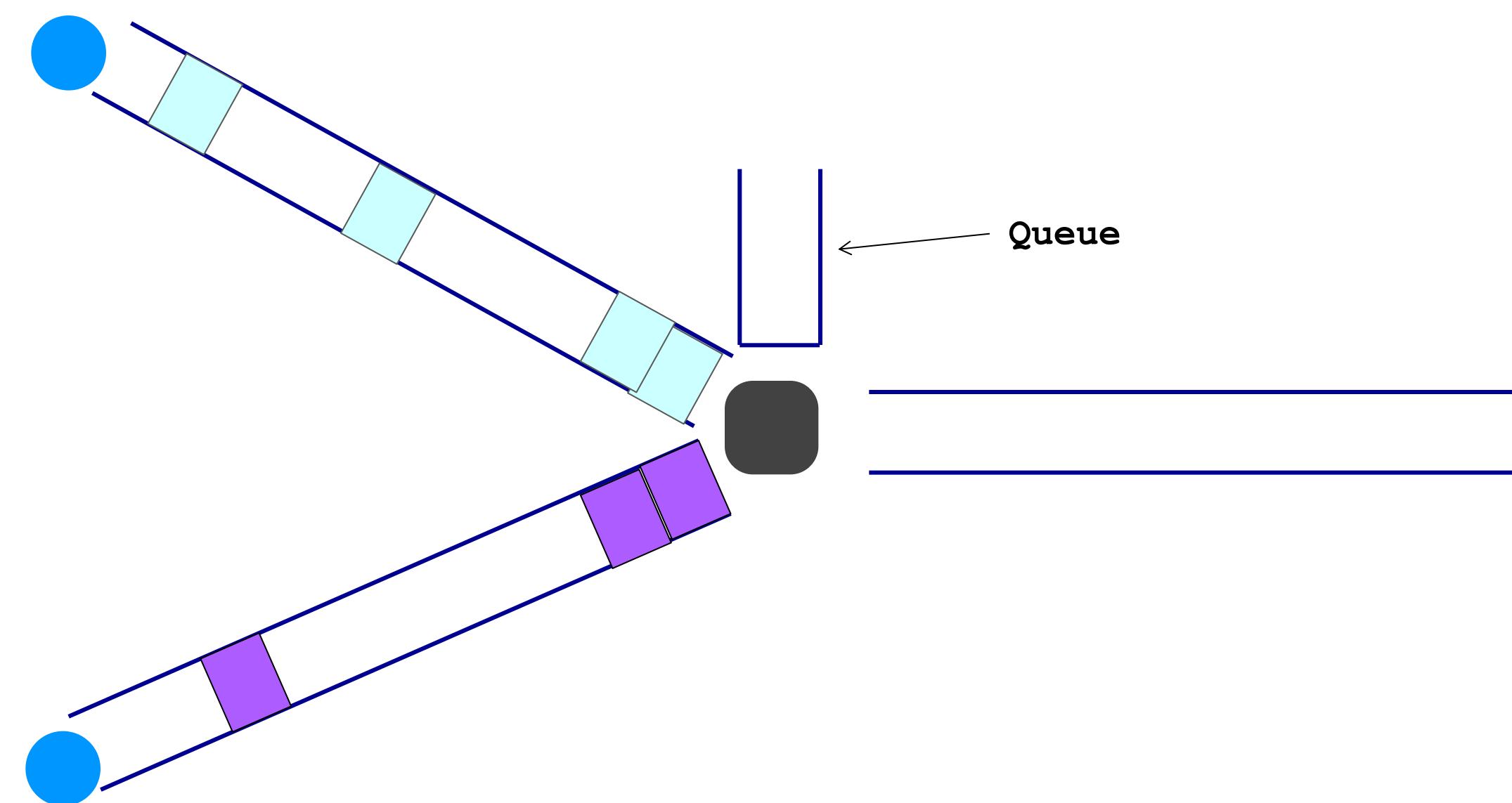
Queueing Delay: “Pipe” View

- How long does a packet have to sit in a buffer before it is processed



Queueing Delay: “Pipe” View

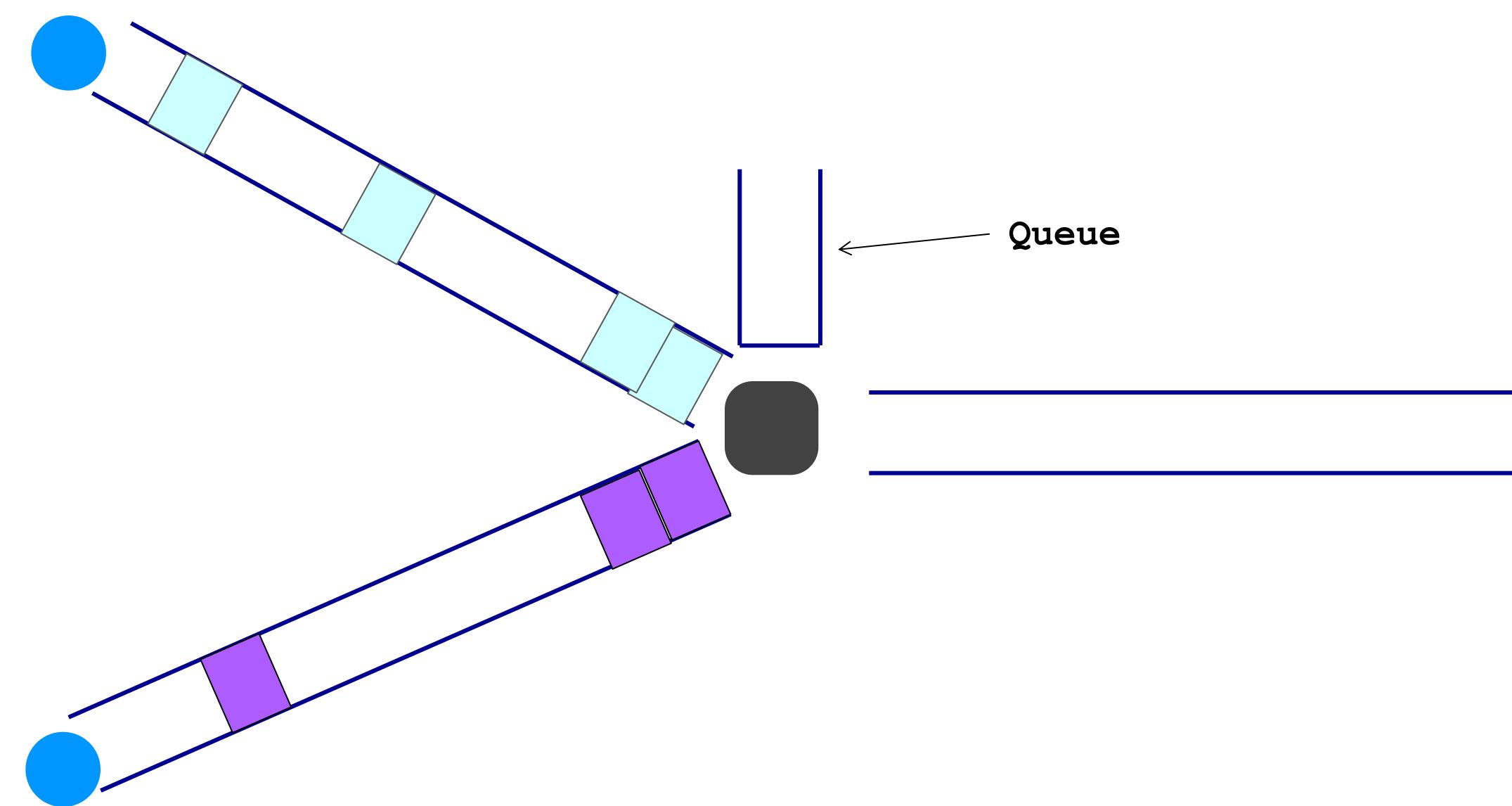
- How long does a packet have to sit in a buffer before it is processed



Transient overload!

Queueing Delay: “Pipe” View

- How long does a packet have to sit in a buffer before it is processed

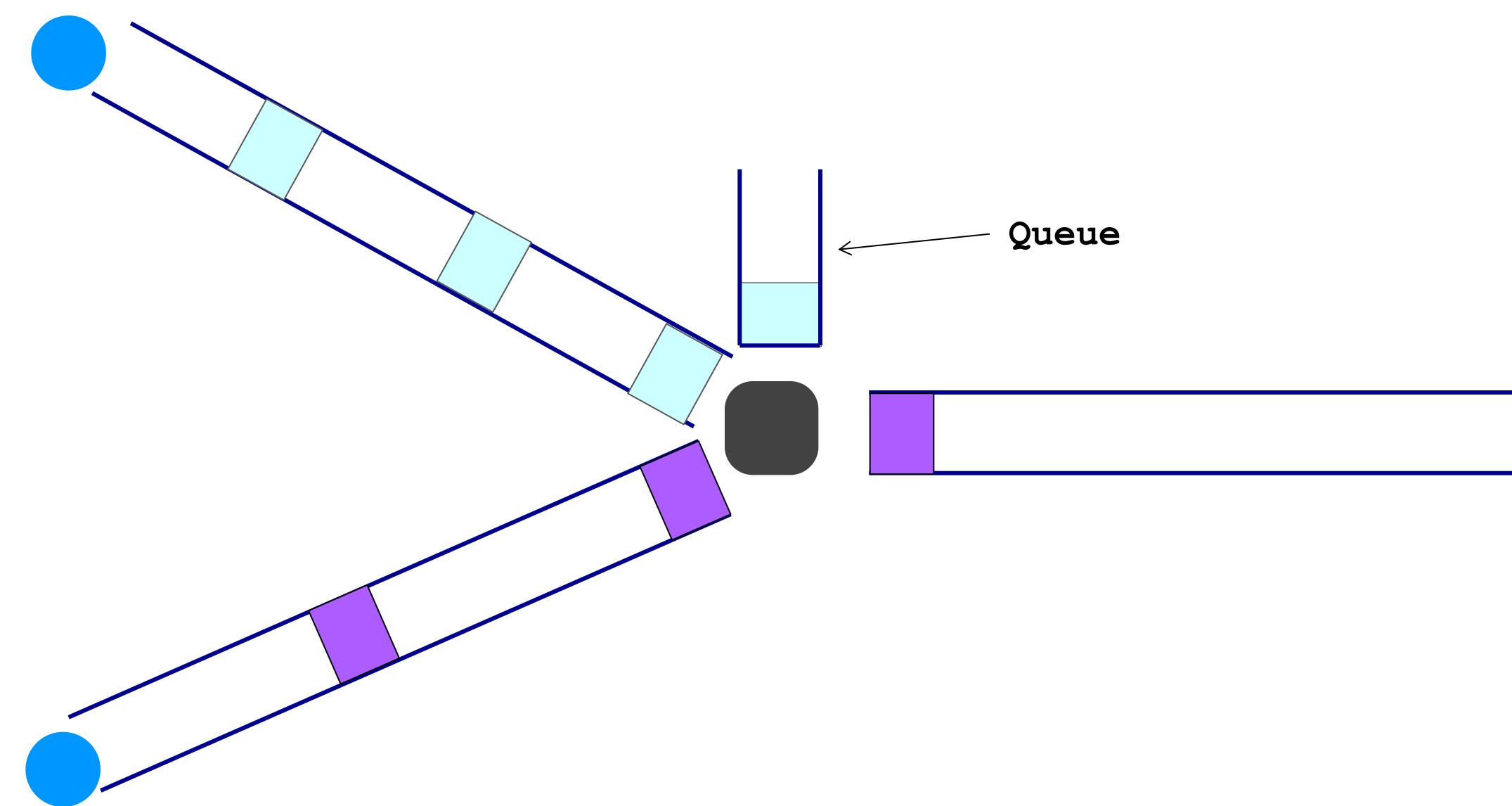


Transient overload!

Not a rare event.

Queueing Delay: “Pipe” View

- How long does a packet have to sit in a buffer before it is processed

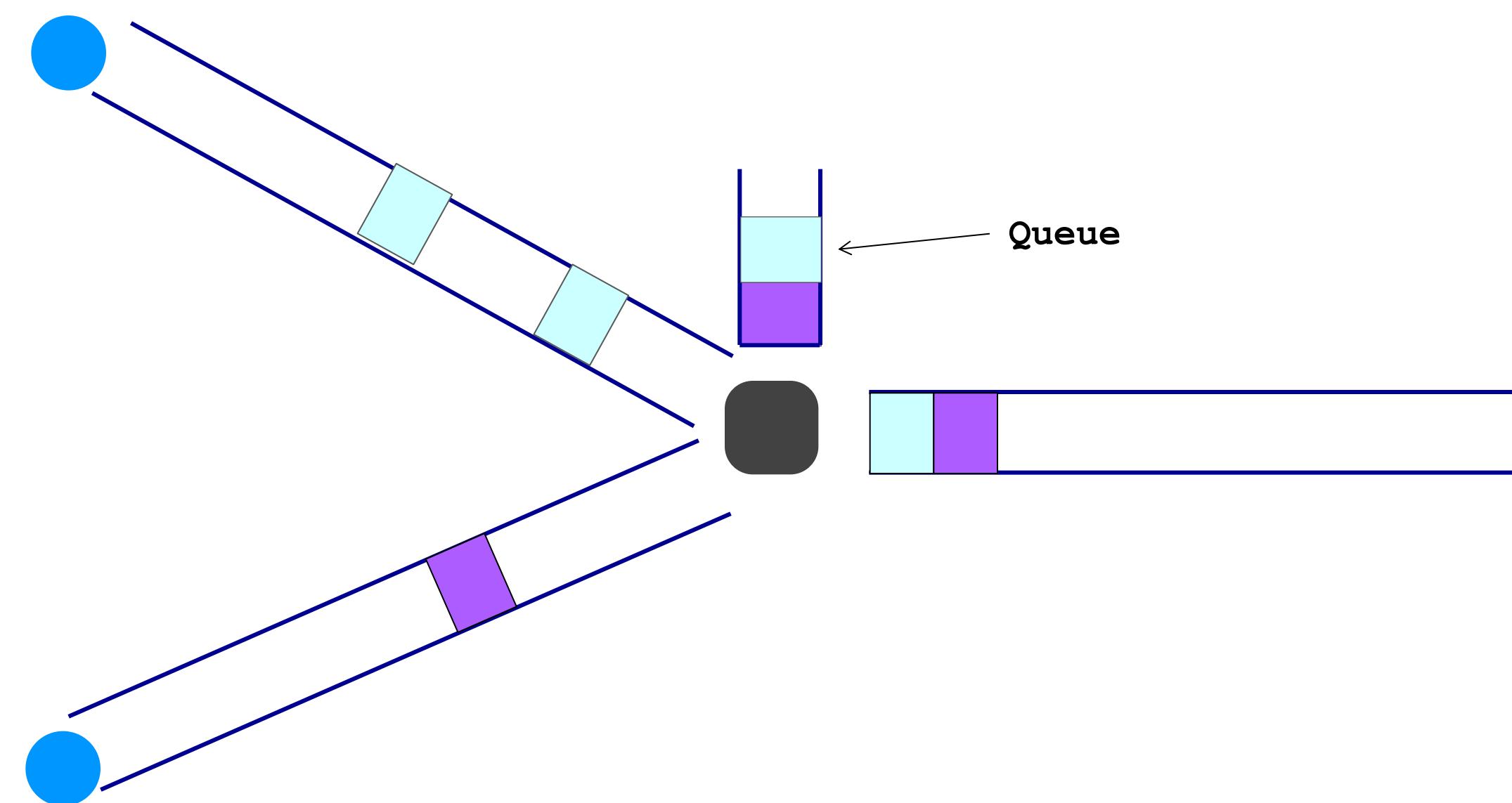


Transient overload!

Not a rare event.

Queueing Delay: “Pipe” View

- How long does a packet have to sit in a buffer before it is processed

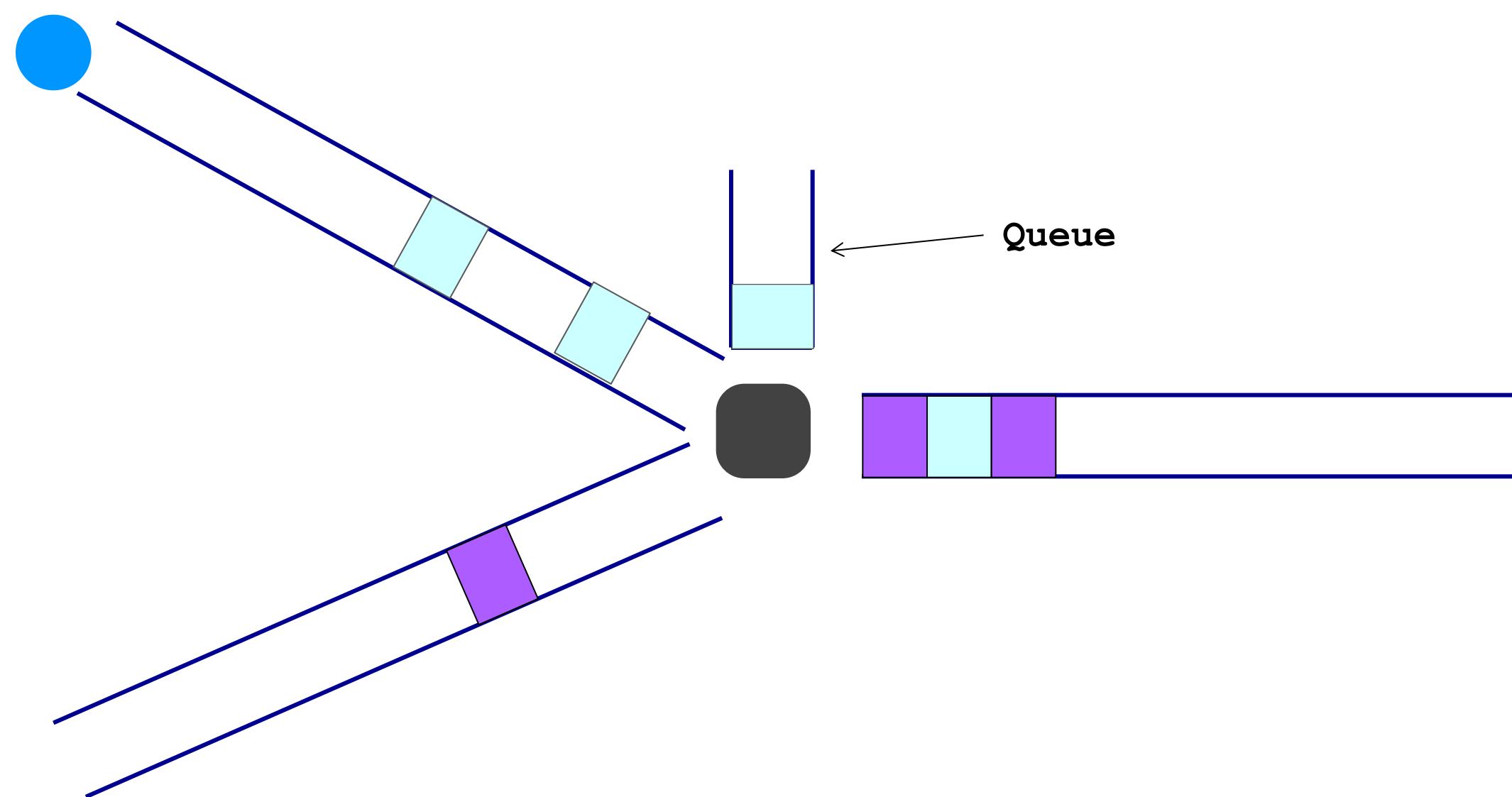


Transient overload!

Not a rare event.

Queueing Delay: “Pipe” View

- How long does a packet have to sit in a buffer before it is processed

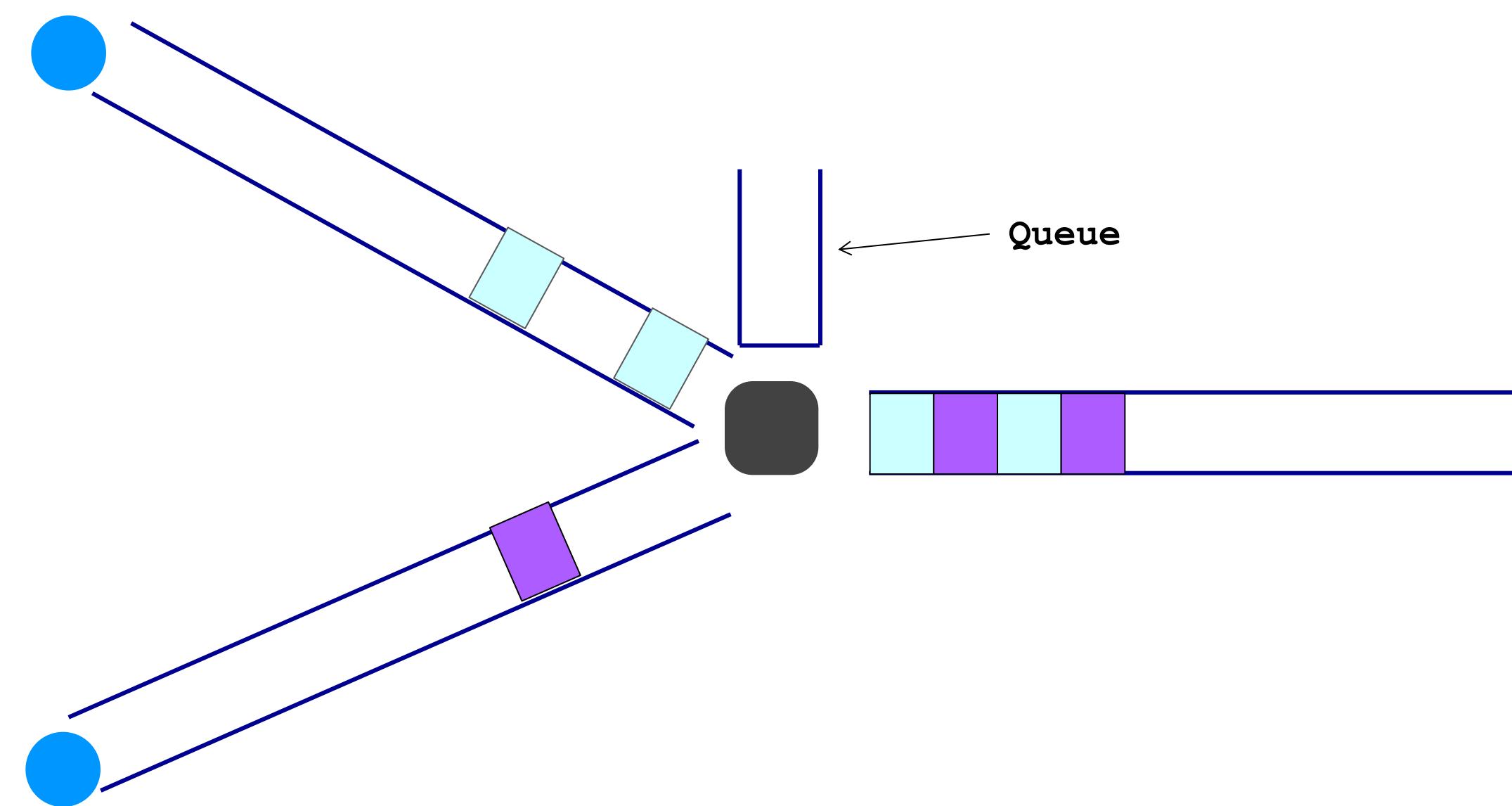


Transient overload!

Not a rare event.

Queueing Delay: “Pipe” View

- How long does a packet have to sit in a buffer before it is processed

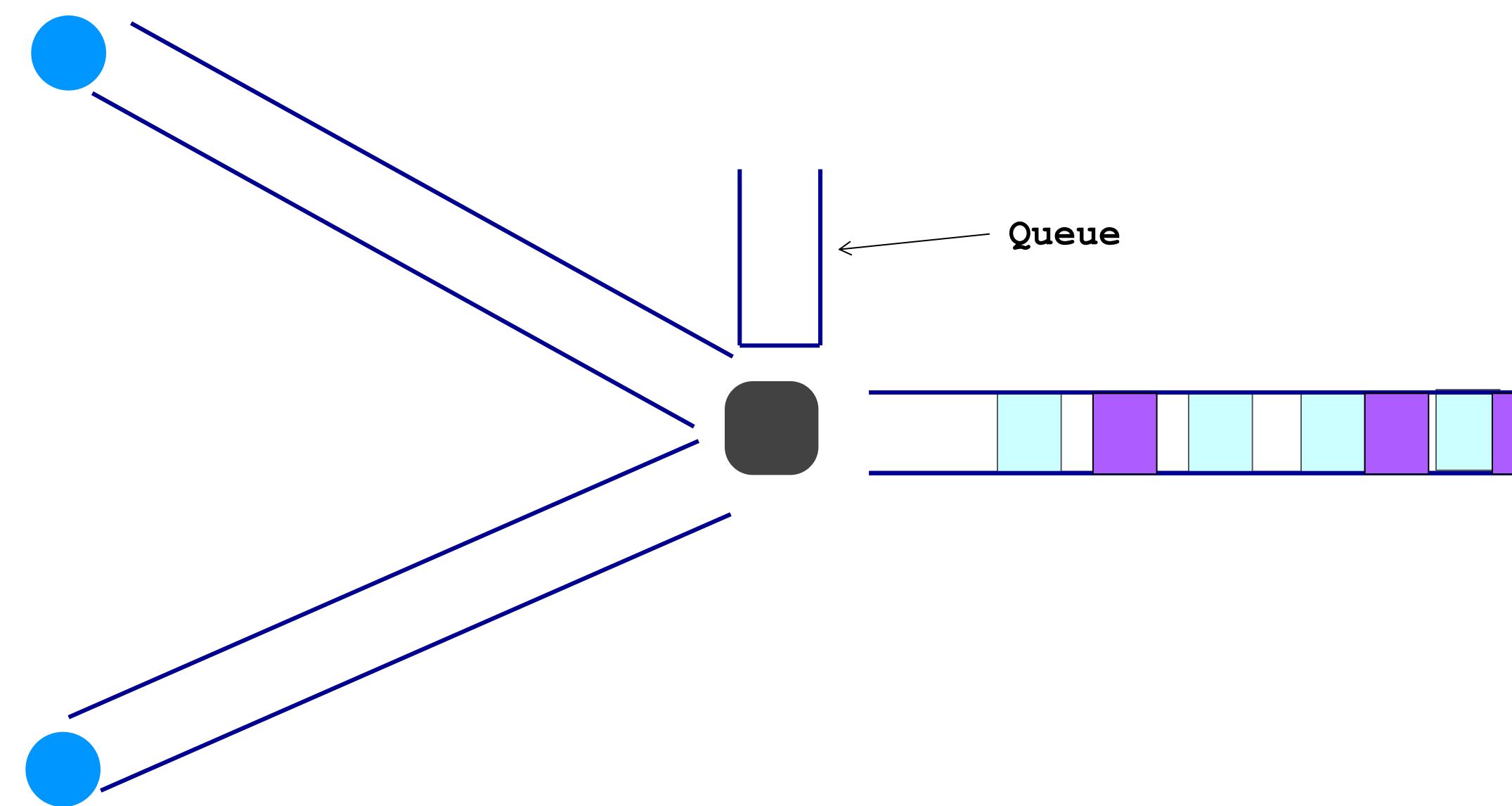


Transient overload!

Not a rare event.

Queueing Delay: “Pipe” View

- How long does a packet have to sit in a buffer before it is processed

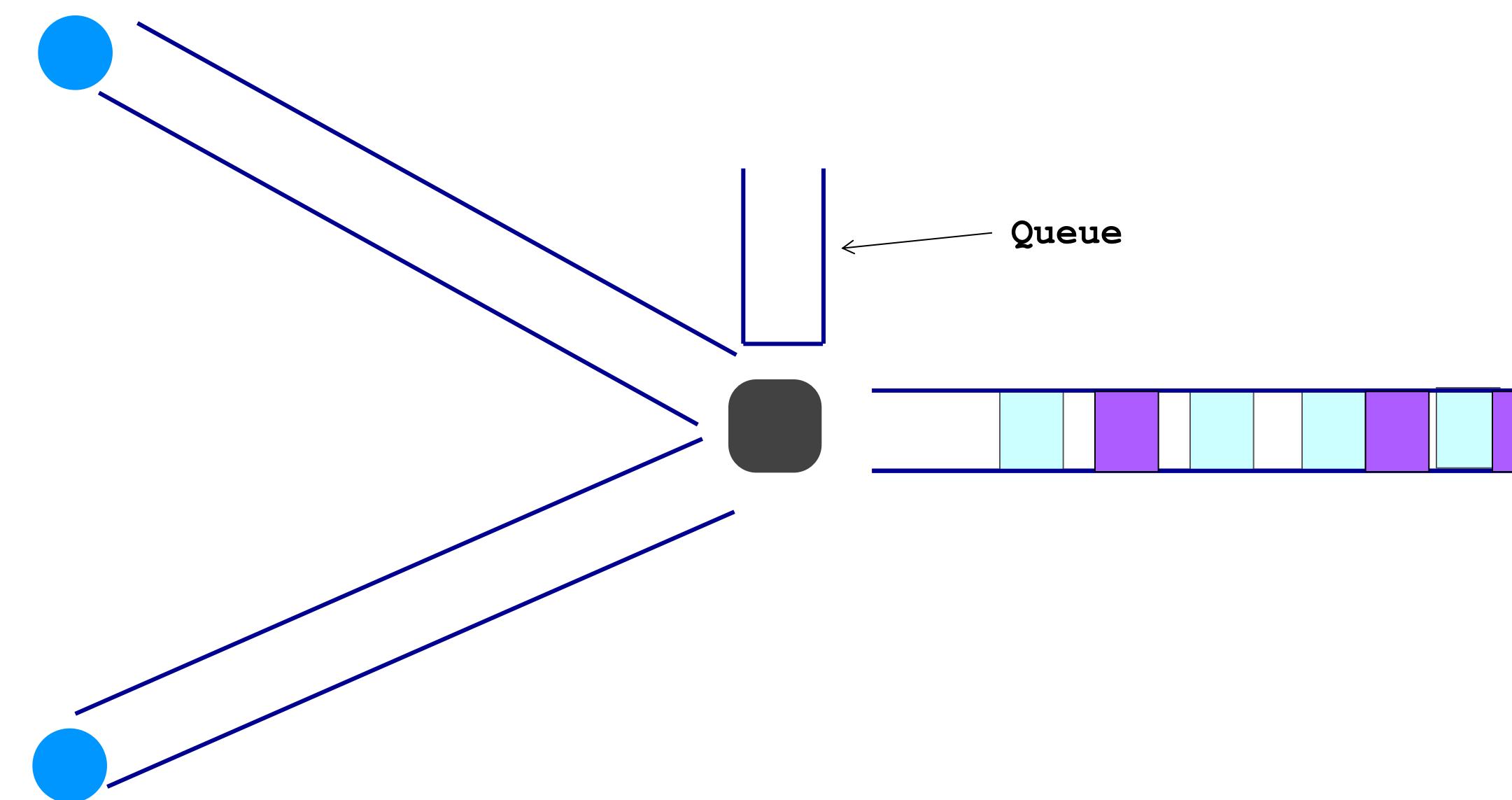


Transient overload!

Not a rare event.

Queueing Delay: “Pipe” View

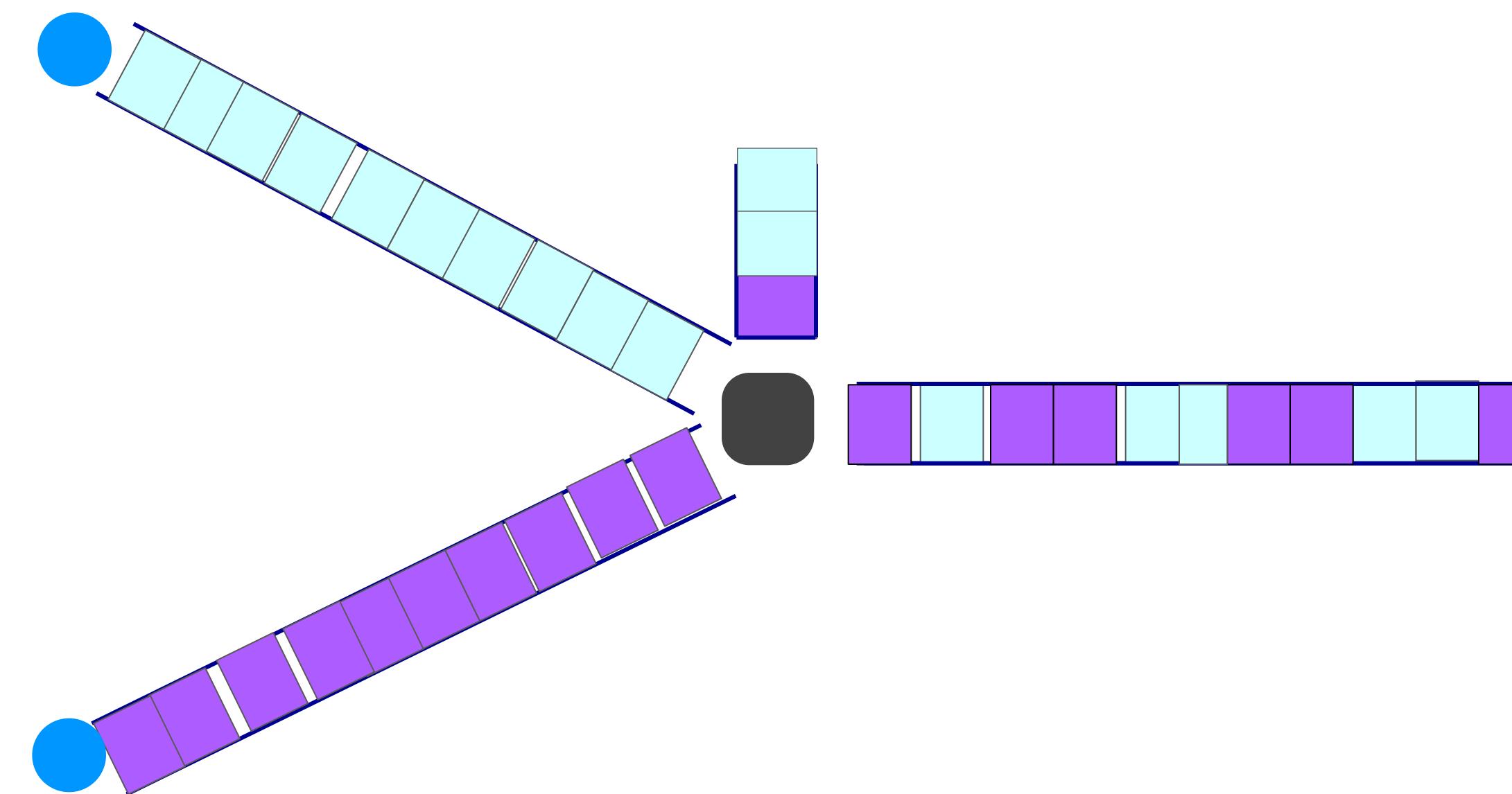
- How long does a packet have to sit in a buffer before it is processed



Queues absorb transient bursts but introduce queueing delay

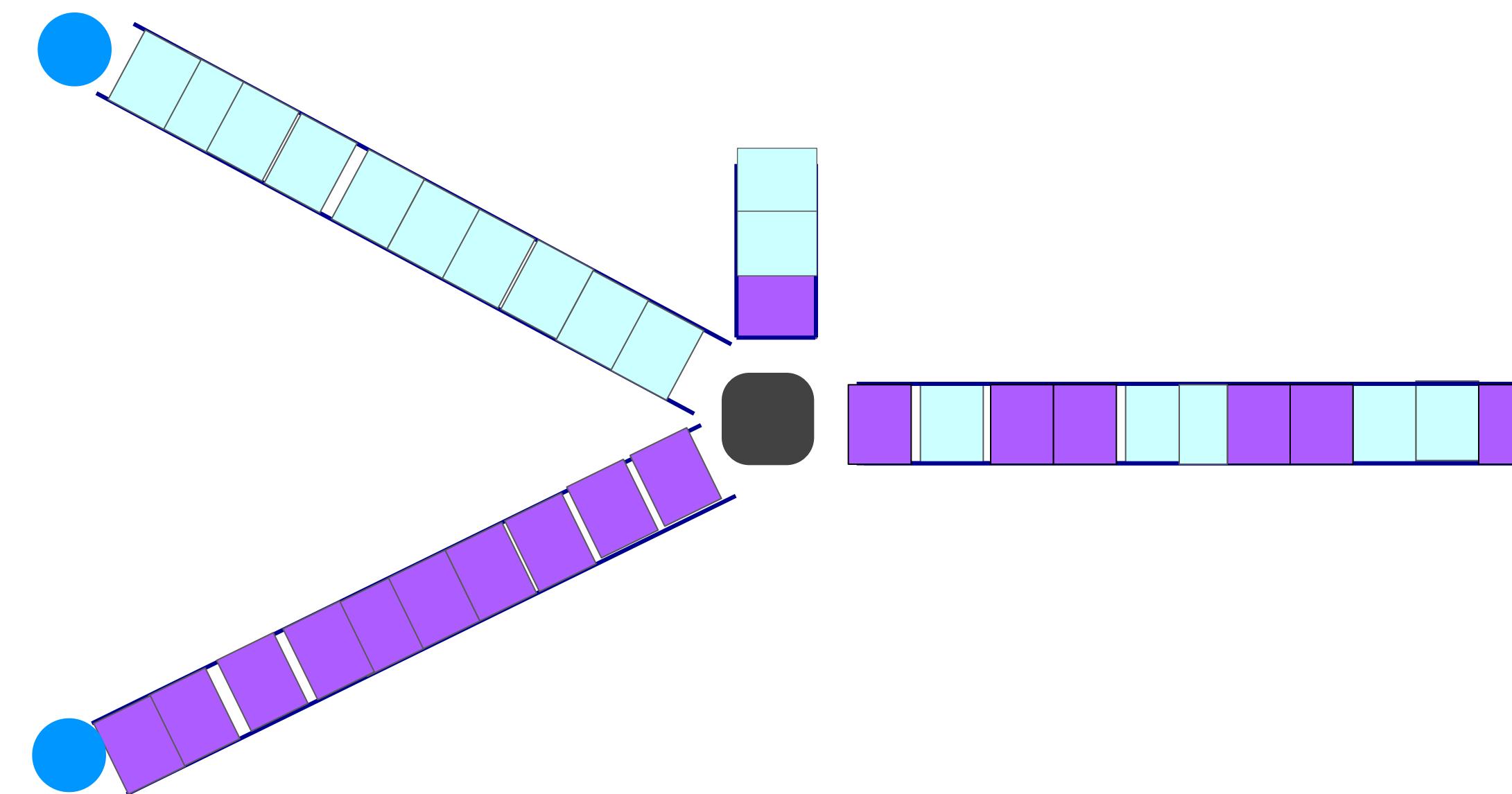
Queueing Delay: “Pipe” View

- How long does a packet have to sit in a buffer before it is processed



Queueing Delay: “Pipe” View

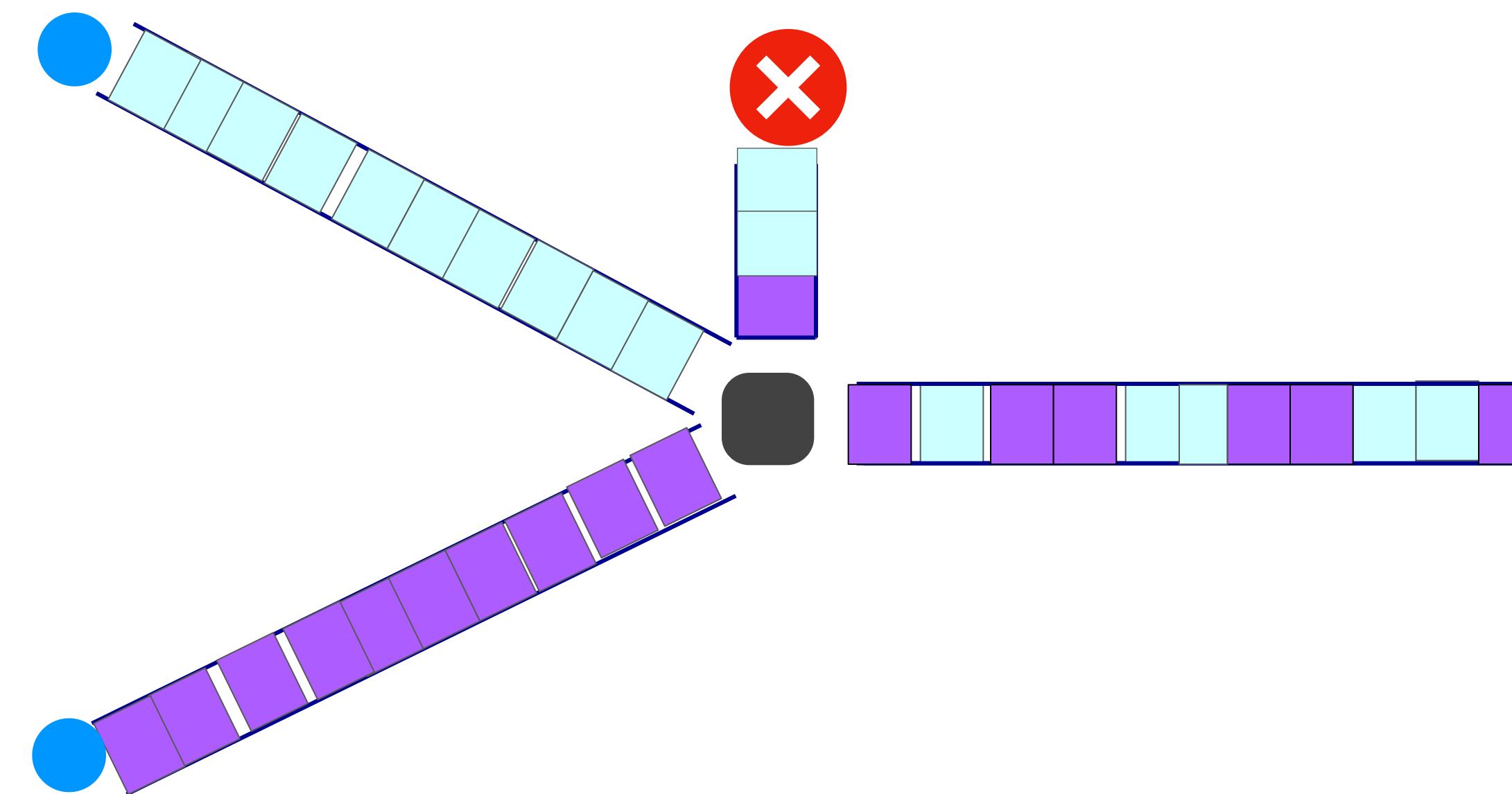
- How long does a packet have to sit in a buffer before it is processed



What about persistent overload?

Queueing Delay: “Pipe” View

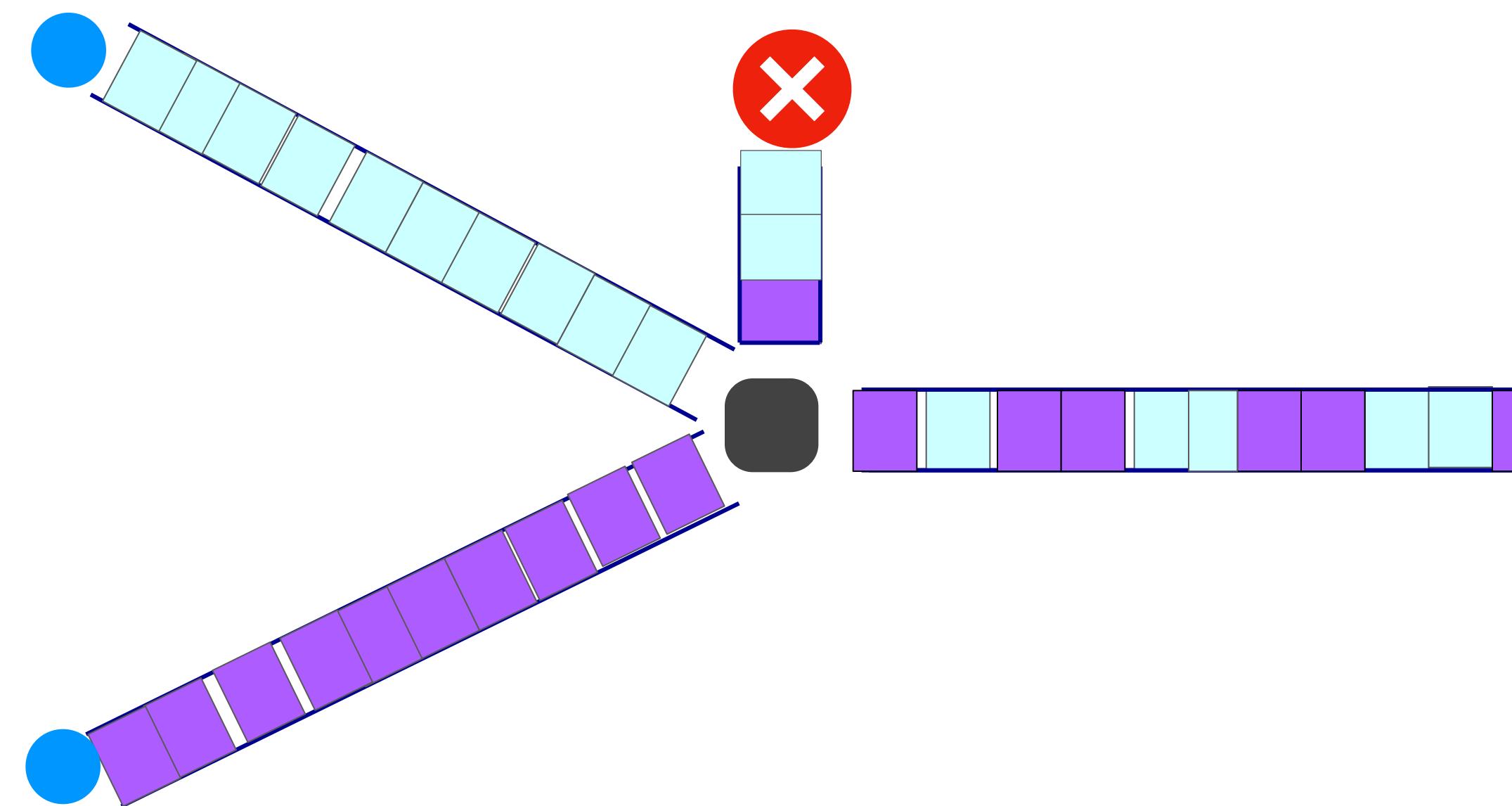
- How long does a packet have to sit in a buffer before it is processed



What about persistent overload?

Queueing Delay: “Pipe” View

- How long does a packet have to sit in a buffer before it is processed



What about persistent overload?

Leads to packet loss

Queueing Delay

Queueing Delay

- **If arrival rate > departure rate**
 - Approaches infinity (assuming an infinite buffer)
 - In practice, finite buffer = loss

Queueing Delay

- **If arrival rate > departure rate**
 - Approaches infinity (assuming an infinite buffer)
 - In practice, finite buffer = loss
- **If arrival rate < departure rate**
 - Depends on burst rate

Queueing Delay

- How long does a packet have to sit in a buffer before it is processed

Queueing Delay

- How long does a packet have to sit in a buffer before it is processed
- Depends on traffic pattern

Queueing Delay

- How long does a packet have to sit in a buffer before it is processed
- Depends on traffic pattern
- Characterized by statistical measures
 - Average queueing delay
 - Average arrival rate
 - Average departure rate

Basic Queueing Theory Terminology

Basic Queueing Theory Terminology

- **Arrival process:** how packets arrive
 - Characterized by average arrival rate **A**

Basic Queueing Theory Terminology

- **Arrival process:** how packets arrive
 - Characterized by average arrival rate **A**
- **W:** average time packets wait in the queue
 - W for “waiting time”

Basic Queueing Theory Terminology

- **Arrival process:** how packets arrive
 - Characterized by average arrival rate **A**
- **W:** average time packets wait in the queue
 - W for “waiting time”
- **L:** average number of packets waiting in the queue
 - L for “length of the queue”

Little's Law (1969)

A: avg. packet arrival rate (/s)

L: avg. # of packets waiting in queue

W: avg. time packets wait in queue

Little's Law (1969)

A: avg. packet arrival rate (/s)

L: avg. # of packets waiting in queue

W: avg. time packets wait in queue

$$L = A \times W$$

Little's Law (1969)

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,

$$W = L / A$$

Processing Delay

Processing Delay

- How long does a switch take to process this packet?
 - Typically assume this is negligible

Delay

- Consists of four components
 - Transmission Delay
 - Propagation Delay
 - Queueing Delay
 - Processing Delay
-
- The diagram illustrates the four components of delay. It starts with a bullet point 'Consists of four components' followed by a list of four items: 'Transmission Delay', 'Propagation Delay', 'Queueing Delay', and 'Processing Delay'. A large bracket on the left side groups the first two items ('Transmission Delay' and 'Propagation Delay') under the heading 'Due to link properties'. Another large bracket on the left side groups the last two items ('Queueing Delay' and 'Processing Delay') under the heading 'Due to traffic matrix and switch internals'.
- Due to link properties
- Due to traffic matrix and switch internals

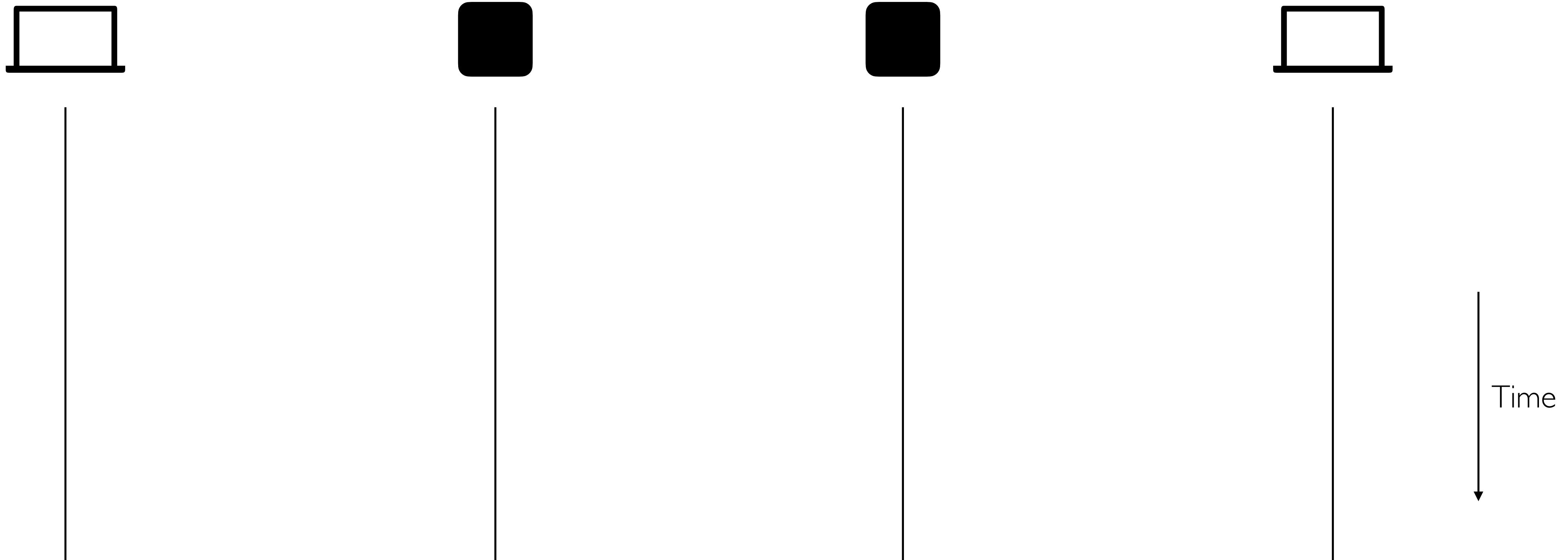
End-to-end Delay

T Transmission Delay

P Propagation Delay

Q Queuing Delay

Pr Processing Delay



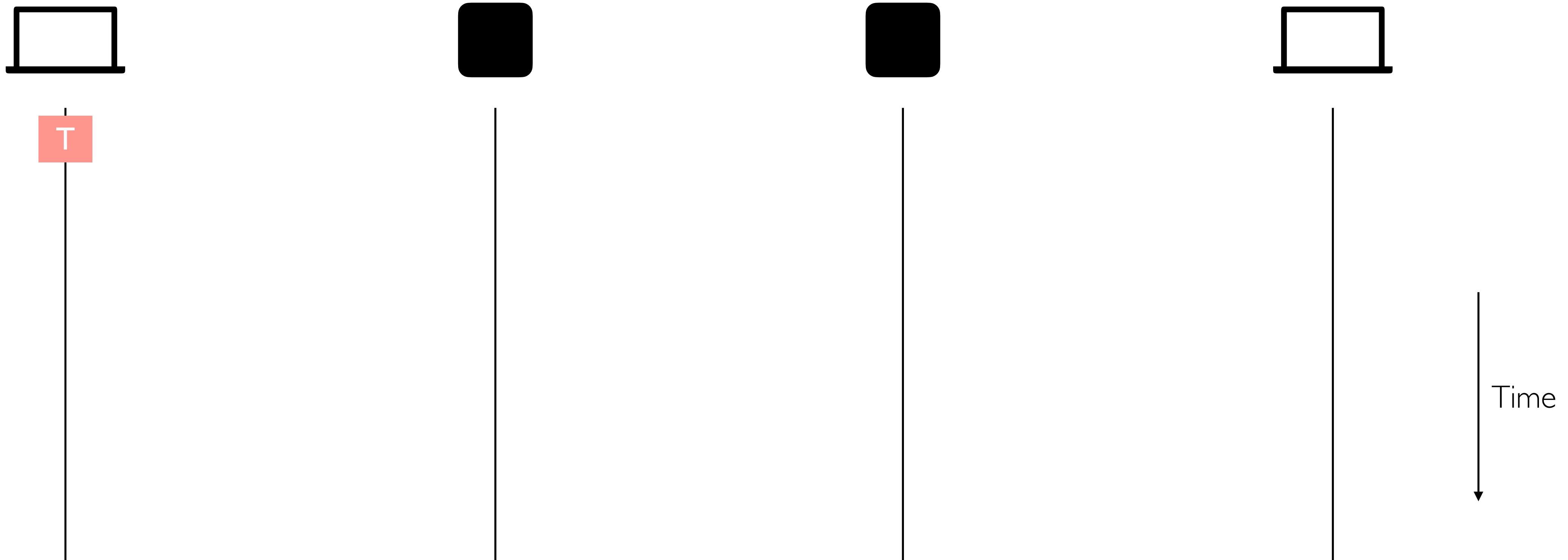
End-to-end Delay

T Transmission Delay

P Propagation Delay

Q Queuing Delay

Pr Processing Delay



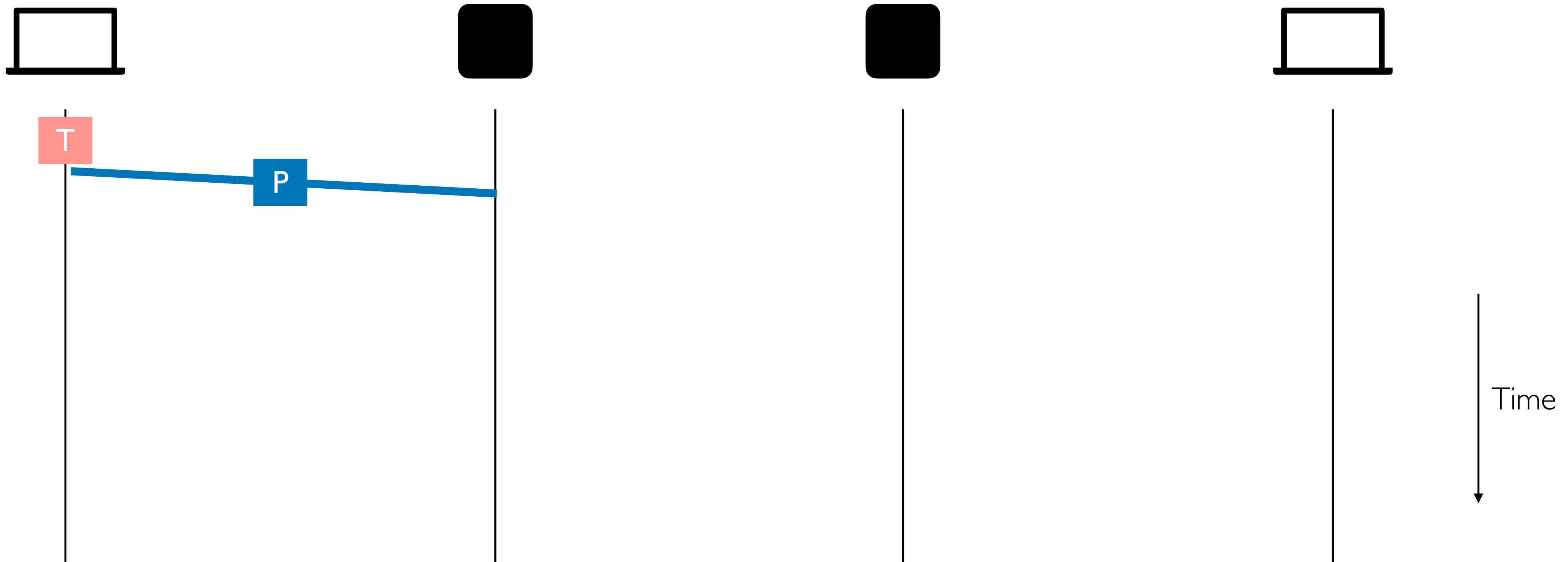
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



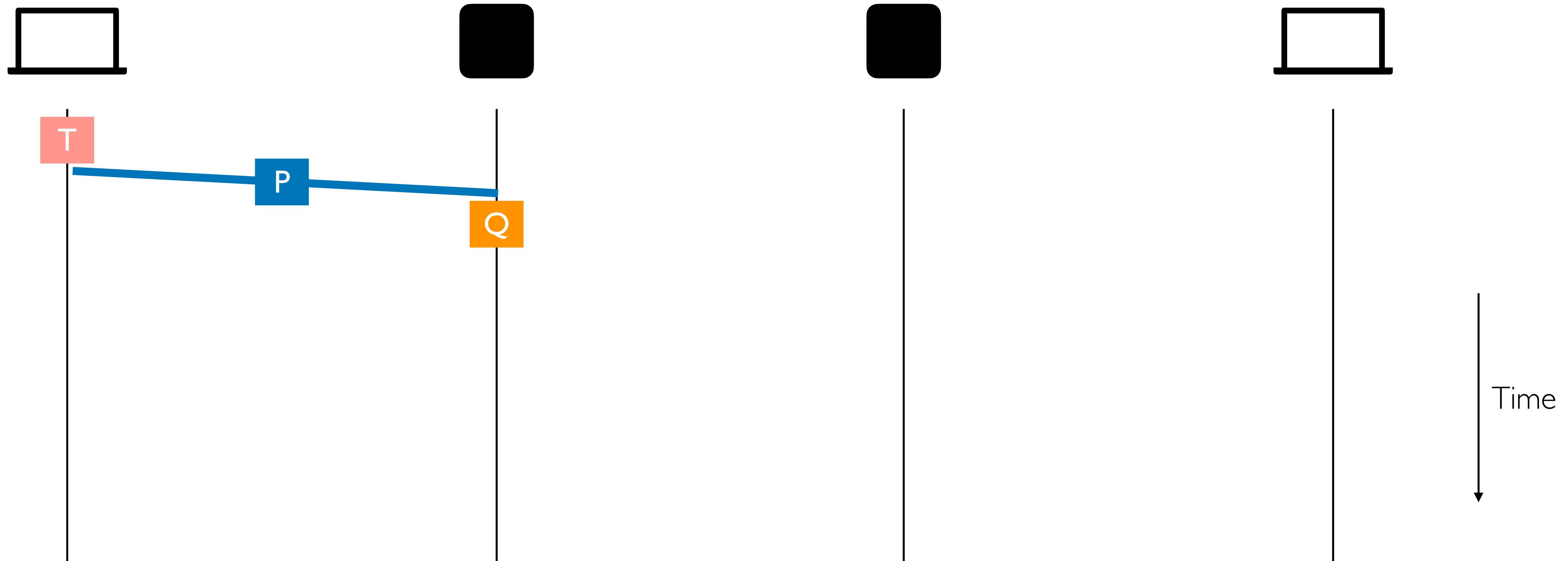
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



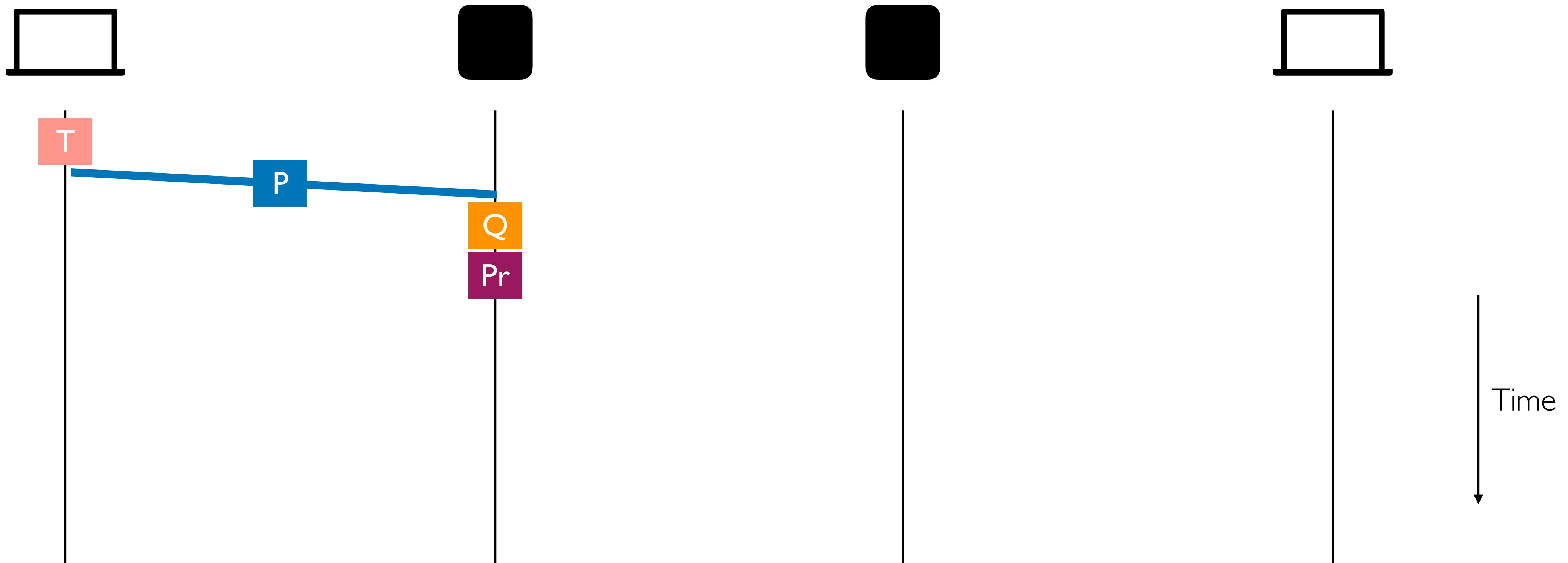
End-to-end Delay

T Transmission Delay

P Propagation Delay

Q Queuing Delay

Pr Processing Delay



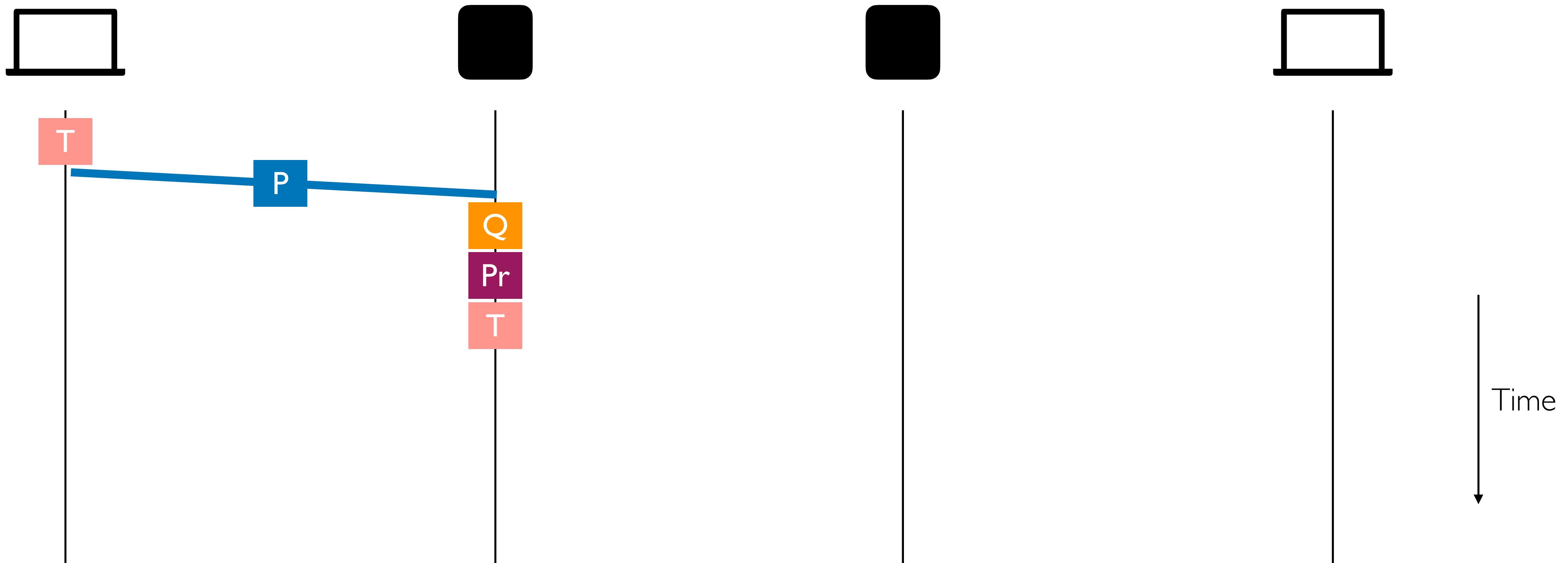
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



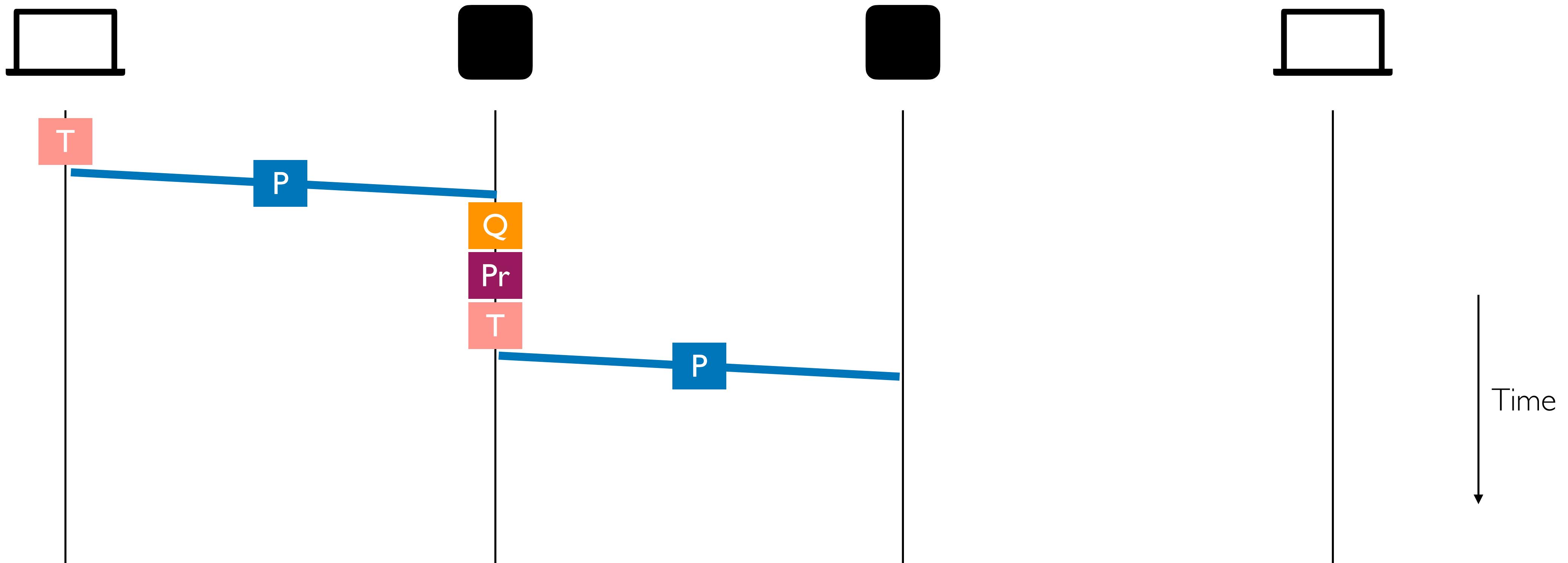
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



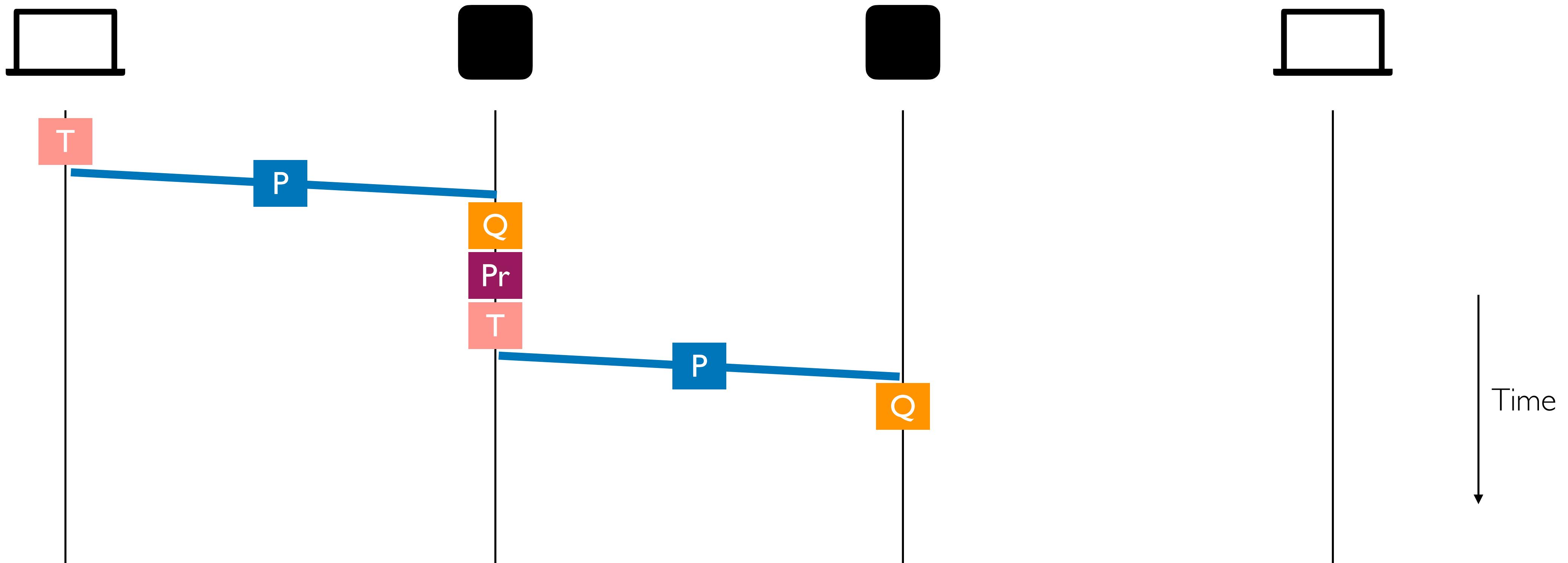
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



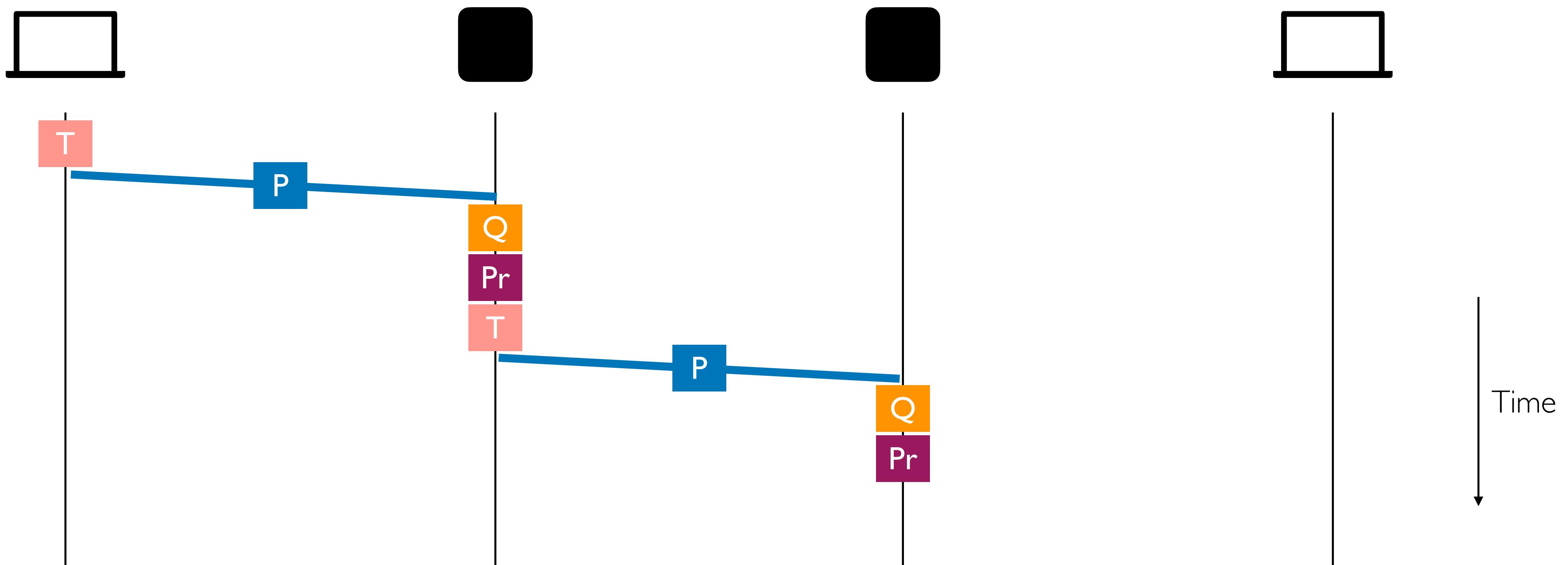
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



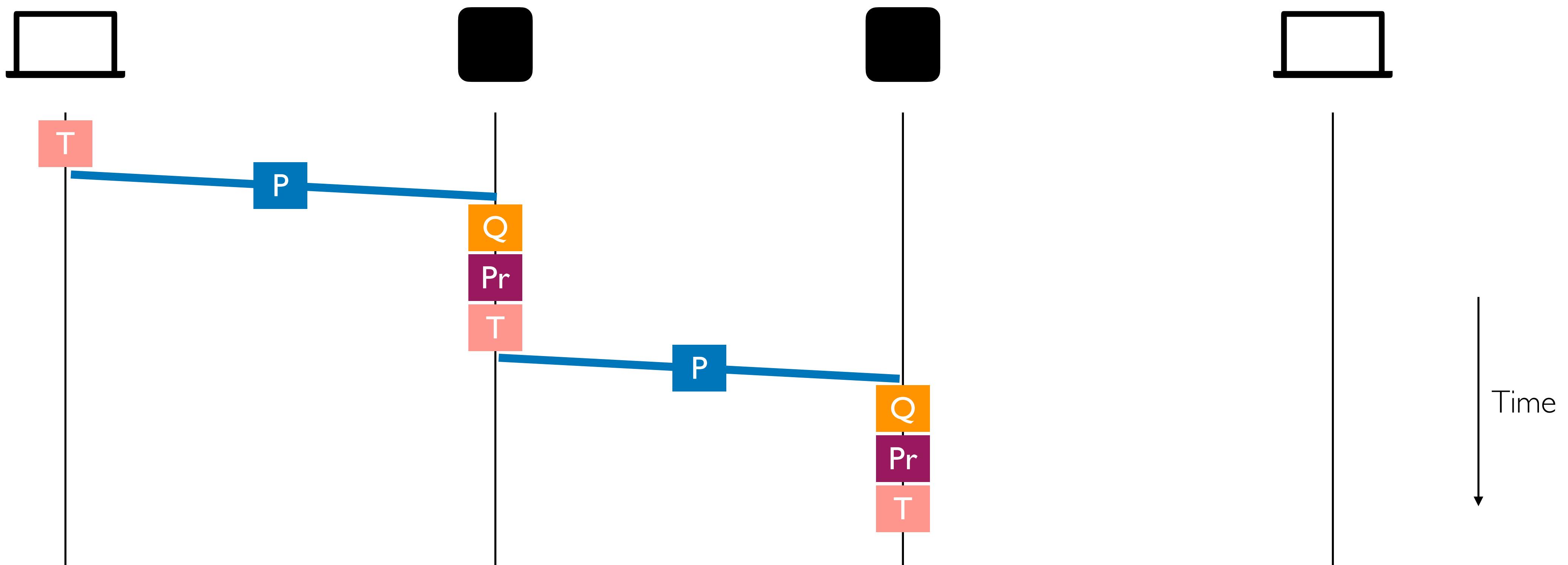
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



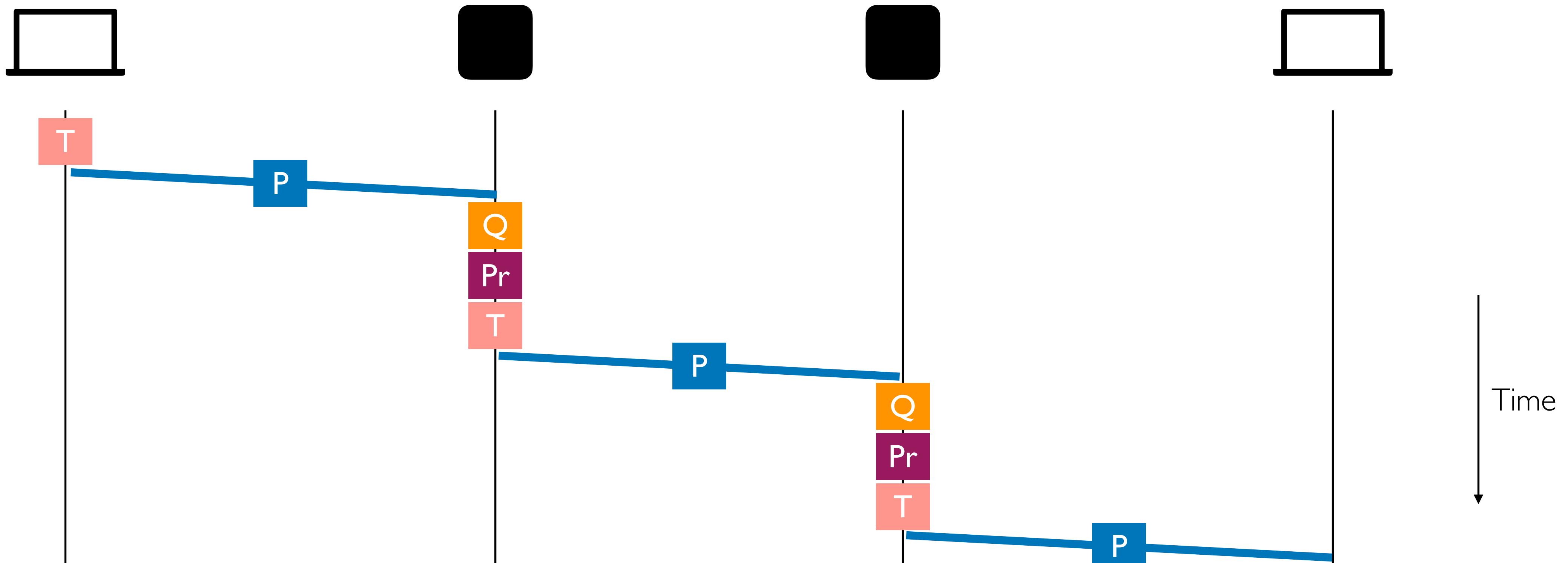
End-to-end Delay

T Transmission Delay

Q Queuing Delay

P Propagation Delay

Pr Processing Delay



Questions?

LOSS

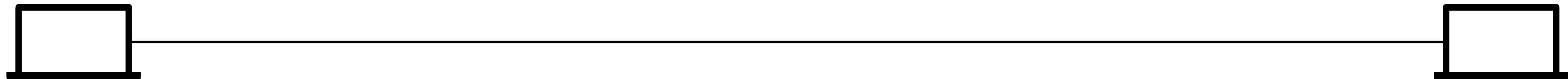
LOSS

- What fraction of the packets sent to a destination are dropped?

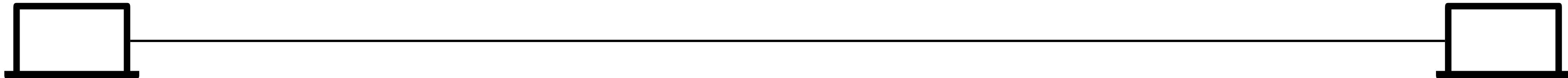
Throughput

- At what rate is the destination receiving data from the source?
 - **Data size / transfer time**

Throughput

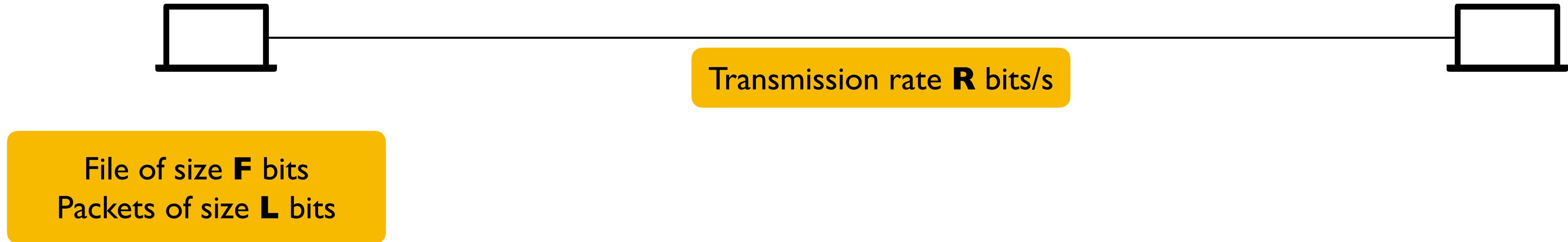


Throughput

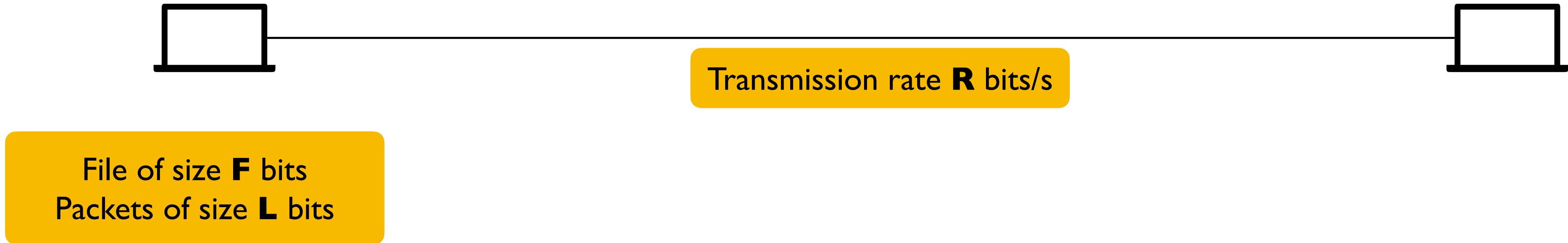


File of size **F** bits
Packets of size **L** bits

Throughput

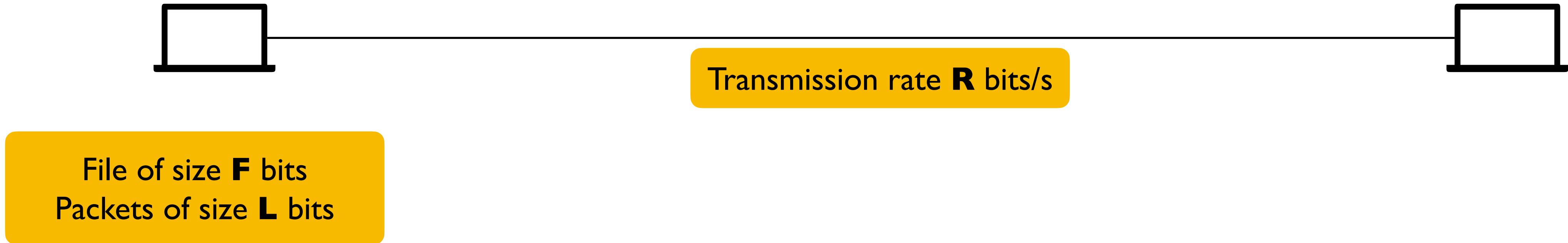


Throughput



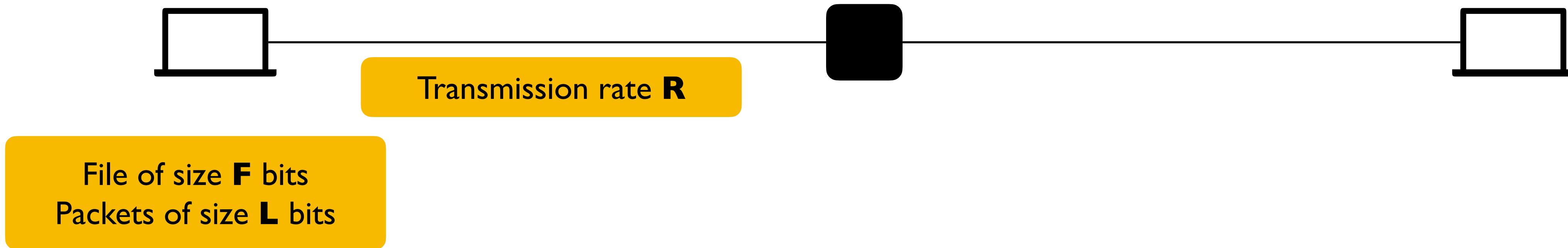
- Transfer time = **F/R** + propagation delay

Throughput

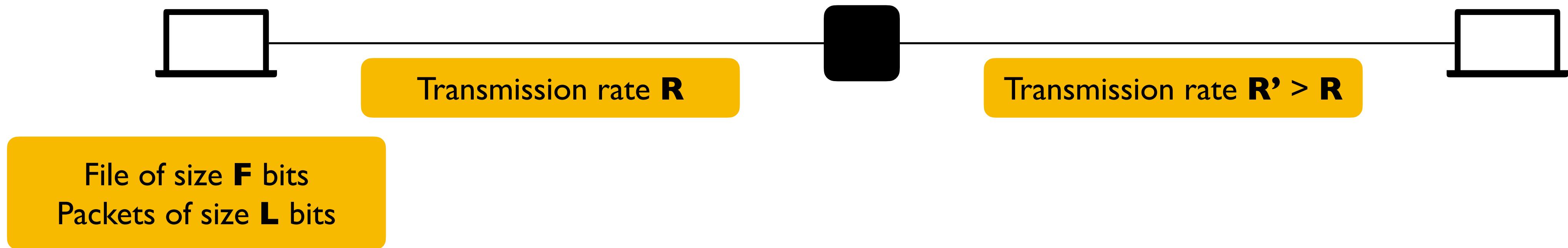


- Transfer time = **F/R** + propagation delay
- Average throughput = **R**

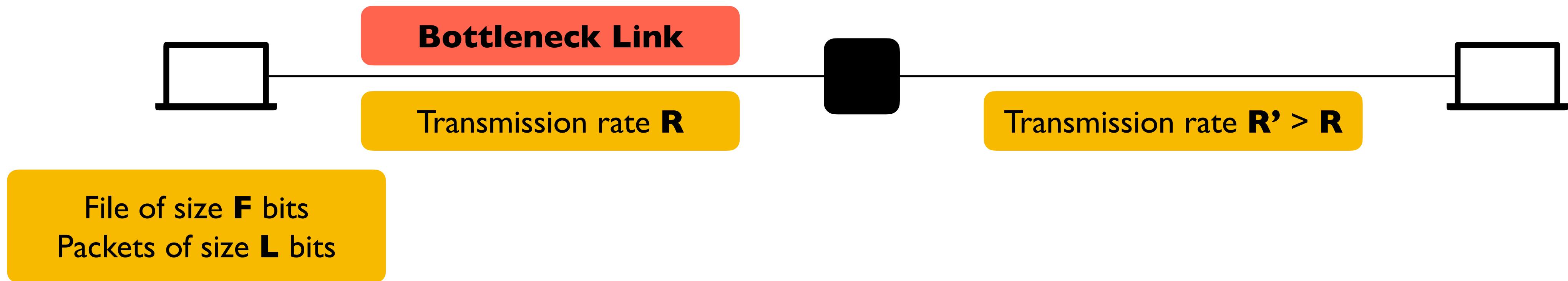
Throughput



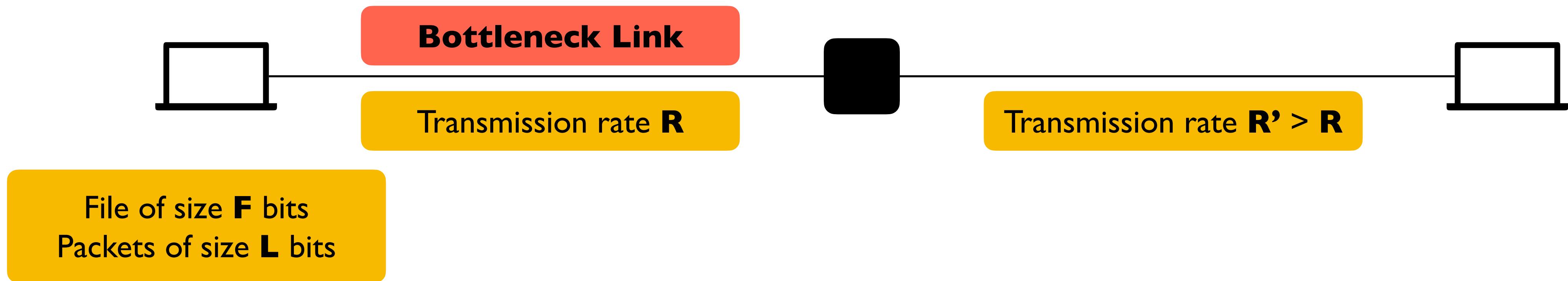
Throughput



Throughput



Throughput

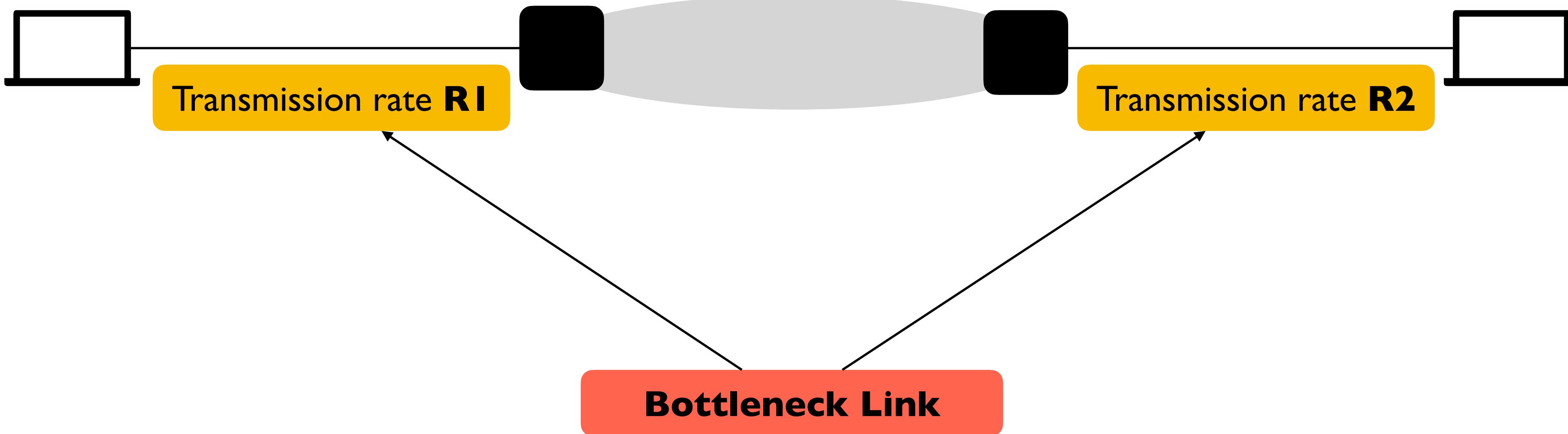


- Average throughput = $\min \{R, R'\} = R$

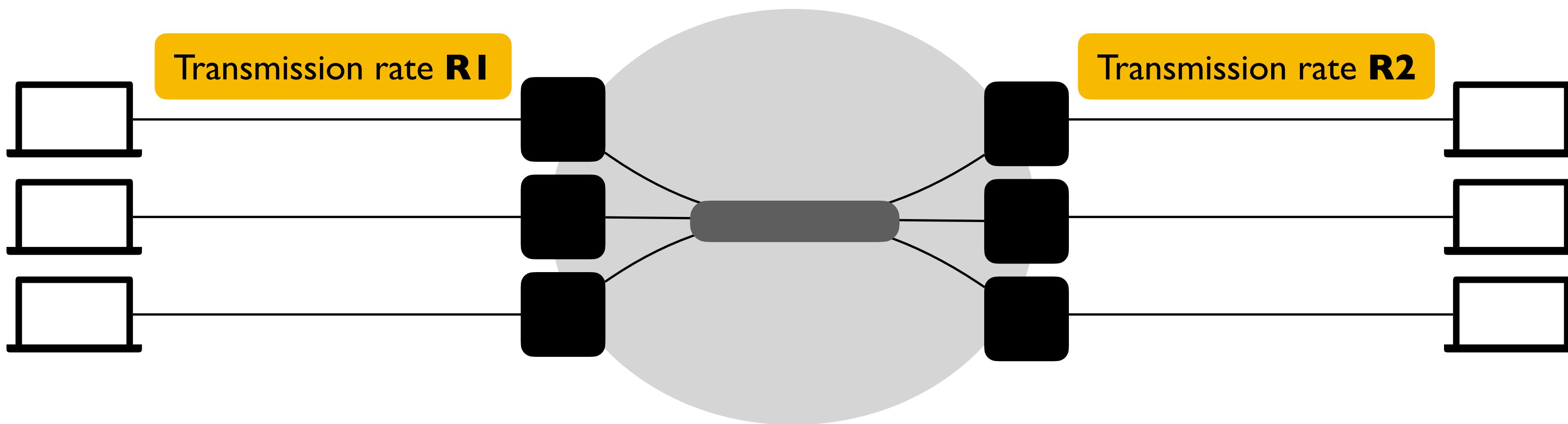
Throughput



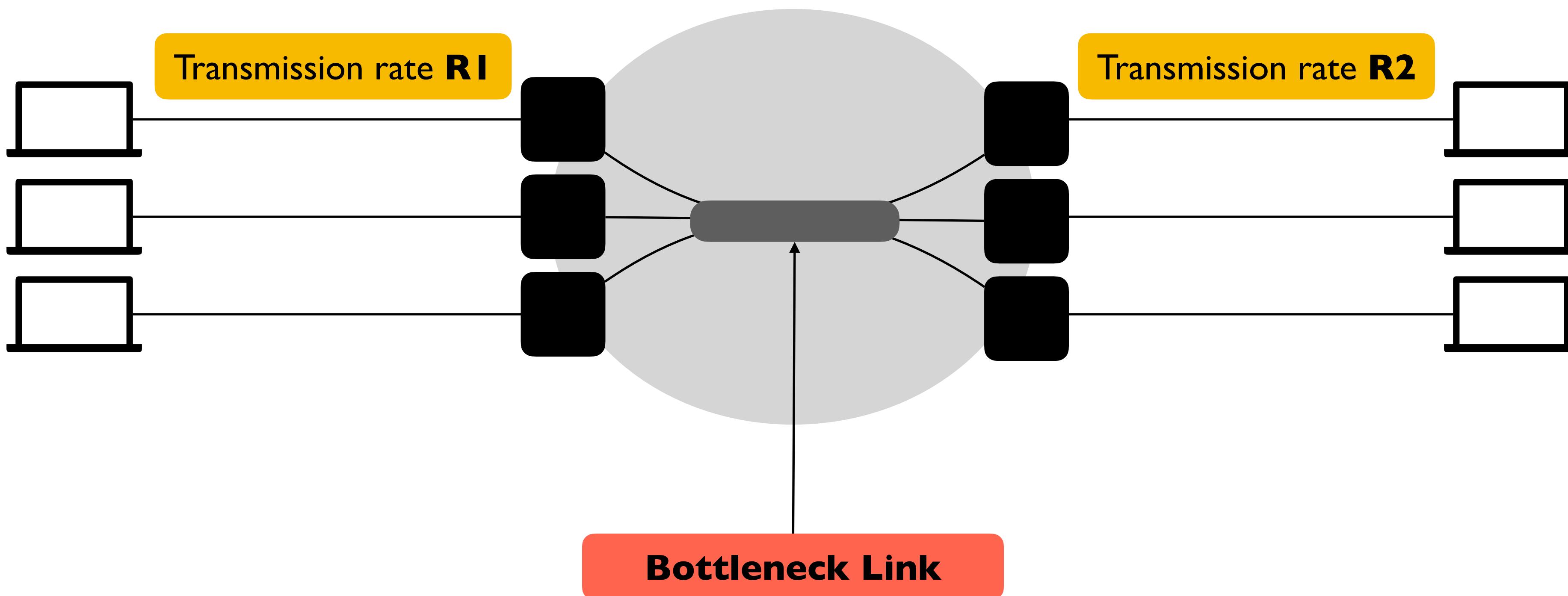
Throughput



Throughput



Throughput



Questions?

Taking Stock

Taking Stock

- What & how of the Internet
 - A slightly detailed look at physical infrastructure

Taking Stock

- **What & how of the Internet**
 - A slightly detailed look at physical infrastructure
- **Sharing networks**
 - Reserve or on-demand?

Taking Stock

- **What & how of the Internet**
 - A slightly detailed look at physical infrastructure
- **Sharing networks**
 - Reserve or on-demand?
- **Evaluating networks**
 - Delay (transmission, propagation, queueing, processing), loss and throughput

How is communication organized?

CPSC 433/533, Spring 2021

Anurag Khandelwal

Three steps

- **Decompose** the problem into tasks
- **Organize** these tasks
- **Assign** tasks to entities (who does what)

Inspiration...

Inspiration...

- CEO A writes letter to CEO B
 - Folds letter and hands it to administrative aide

Dear Pat,

Your days are numbered.

*- John
XOXO*

Inspiration...

- CEO A writes letter to CEO B
 - Folds letter and hands it to administrative aide
- Aide:
 - Puts letter in envelope with CEO B's full name
 - Takes to FedEx



Inspiration...

- CEO A writes letter to CEO B
 - Folds letter and hands it to administrative aide
- Aide:
 - Puts letter in envelope with CEO B's full name
 - Takes to FedEx
- FedEx Office
 - Puts letter in larger envelope
 - Puts name and street address on FedEx envelope
 - Puts package on FedEx delivery truck



Inspiration...

- CEO A writes letter to CEO B
 - Folds letter and hands it to administrative aide
- Aide:
 - Puts letter in envelope with CEO B's full name
 - Takes to FedEx
- FedEx Office
 - Puts letter in larger envelope
 - Puts name and street address on FedEx envelope
 - Puts package on FedEx delivery truck
- FedEx delivers to other company



Path of the Letter

CEO

Aide

FedEx

CEO

Aide

FedEx

Path of the Letter

CEO



Aide

FedEx

CEO

Aide

FedEx

Path of the Letter

CEO

Aide



FedEx

CEO

Aide

FedEx

Path of the Letter

CEO

Aide

FedEx

CEO

Aide

FedEx



Path of the Letter

CEO

Aide

FedEx

CEO

Aide



FedEx

Path of the Letter

CEO

Aide

FedEx

CEO



Aide

FedEx

Path of the Letter

CEO

CEO

Aide

Aide

FedEx

FedEx

Path of the Letter

CEO

Letter

CEO

Aide

Aide

FedEx

FedEx

Path of the Letter

CEO

Letter

CEO

Aide

Envelope

Aide

FedEx

FedEx

Path of the Letter

CEO

Letter

CEO

Aide

Envelope

Aide

FedEx

FedEx Envelope

FedEx

Path of the Letter

CEO

Semantic Content

CEO

Aide

Envelope

Aide

FedEx

FedEx Envelope

FedEx

Path of the Letter

CEO

Semantic Content

CEO

Aide

Identity

Aide

FedEx

FedEx Envelope

FedEx

Path of the Letter

CEO

Semantic Content

CEO

Aide

Identity

Aide

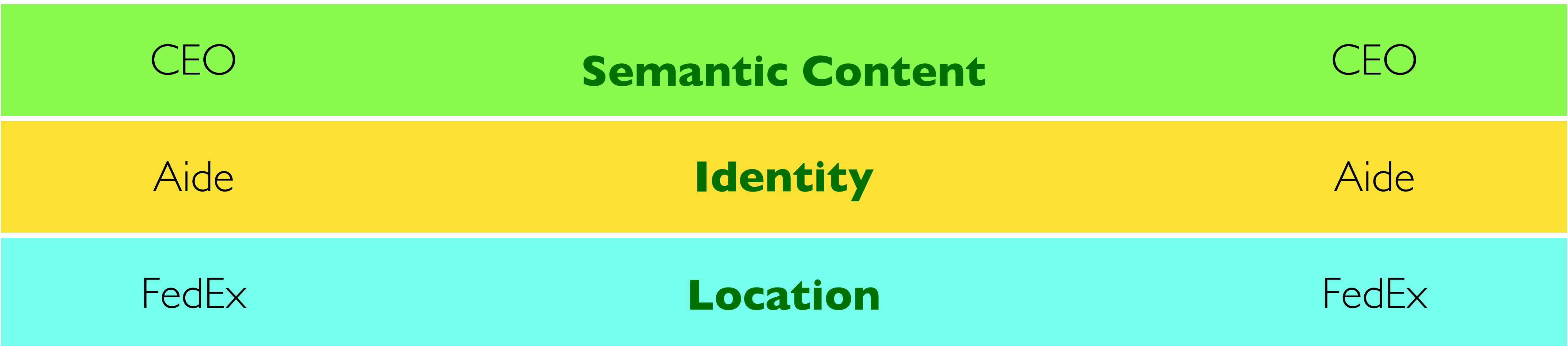
FedEx

Location

FedEx

Path of the Letter

- “Peers” in the same layer understand the same things
- No one else needs to
- Lowest level has most packaging



In the Internet: Decomposition

In the Internet: Decomposition

Physical transfer of bits

In the Internet: Decomposition

Best-effort local *packet* delivery

Physical transfer of bits

In the Internet: Decomposition

Best effort *global* packet delivery

Best-effort local *packet* delivery

Physical transfer of bits

In the Internet: Decomposition

Reliable (or unreliable) transport

Best effort *global* packet delivery

Best-effort local *packet* delivery

Physical transfer of bits

In the Internet: Decomposition

Applications

Reliable (or unreliable) transport

Best effort *global* packet delivery

Best-effort local *packet* delivery

Physical transfer of bits

In the Internet: Decomposition

Applications
...built on...

Reliable (or unreliable) transport

Best effort *global* packet delivery

Best-effort local *packet* delivery

Physical transfer of bits

In the Internet: Decomposition

Applications
...built on...

Reliable (or unreliable) transport
...built on...

Best effort *global* packet delivery

Best-effort local *packet* delivery

Physical transfer of bits

In the Internet: Decomposition

Applications
...built on...

Reliable (or unreliable) transport
...built on...

Best effort *global* packet delivery
...built on...

Best-effort local packet delivery

Physical transfer of bits

In the Internet: Decomposition

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

In the Internet: Decomposition

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

L7 Application

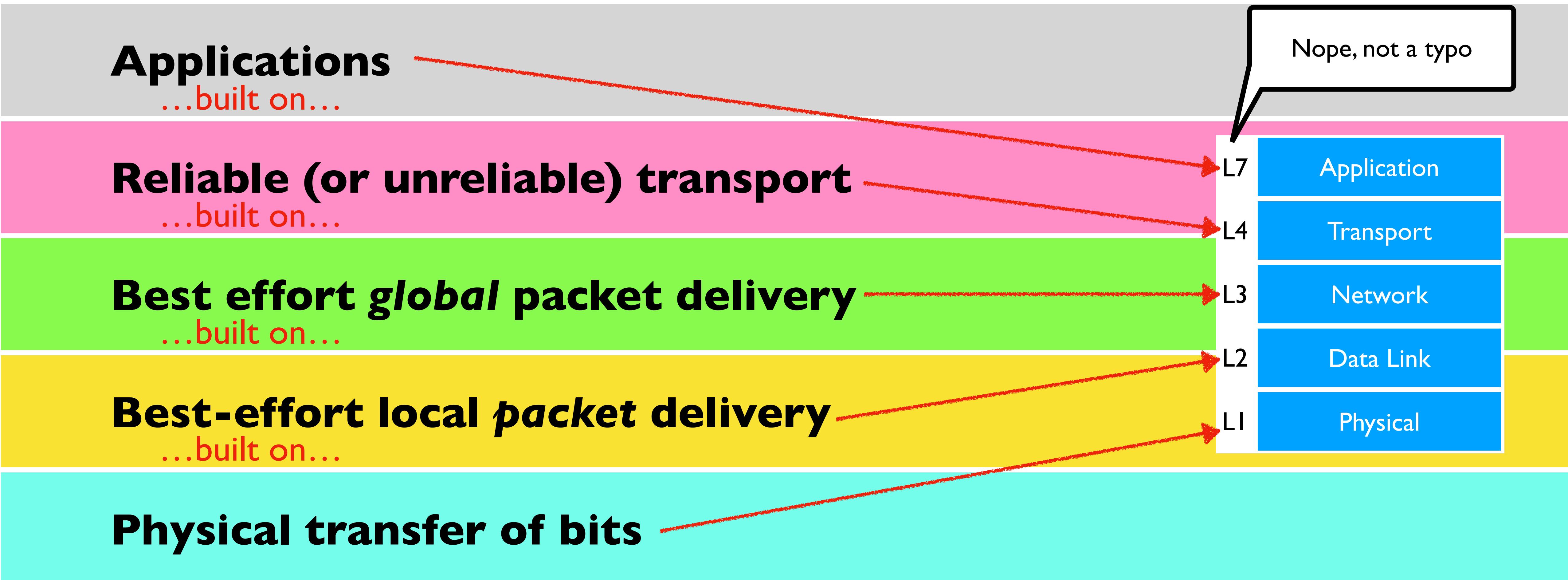
L4 Transport

L3 Network

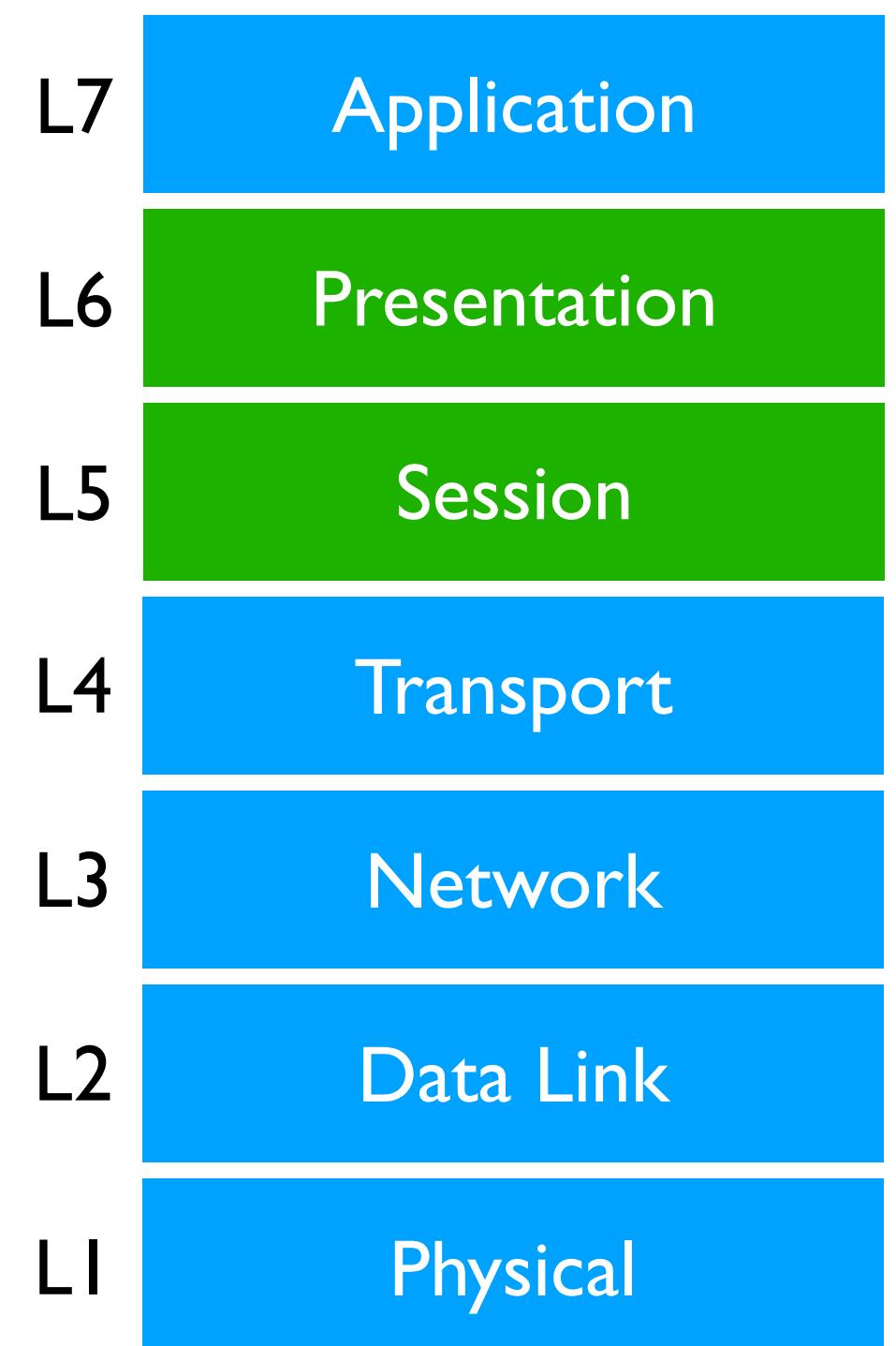
L2 Data Link

L1 Physical

In the Internet: Decomposition

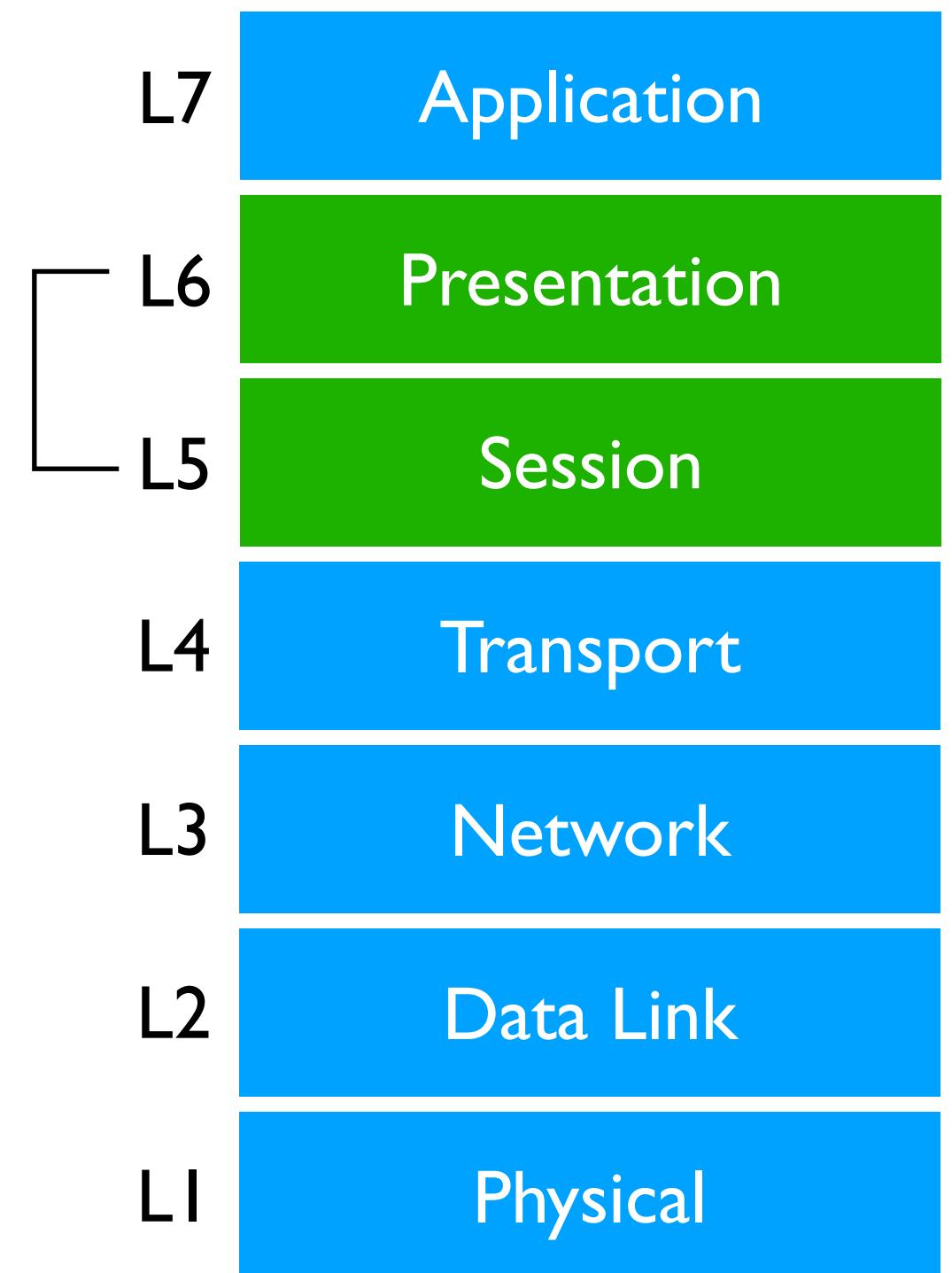


In the context of the Internet



In the context of the Internet

The Open Systems Interconnect (OSI) model developed by the ISO included two additional layers that are often implemented as a part of the application layer



Three steps

- **Decomposition**
- **Organization**
- **Assignment**

Three steps

- **Decomposition**
- **Organization**

- **Assignment**

Layering principle

Layering Principle

Layering Principle

- Layer = part of a system with well-defined interfaces to other parts

Layering Principle

- Layer = part of a system with well-defined interfaces to other parts
- A layer interacts only with layers above and below

Layering Principle

- Layer = part of a system with well-defined interfaces to other parts
- A layer interacts only with layers above and below
- Two layers interact only via the interface between them

Layering Principle

- Layer = part of a system with well-defined interfaces to other parts
- A layer interacts only with layers above and below
- Two layers interact only via the interface between them
- Each layer relies on functionality of layers below it
 - *E.g., reliable transport (L4) relies on best-effort delivery (L2, L3)*

Layers and Protocols



Layers and Protocols



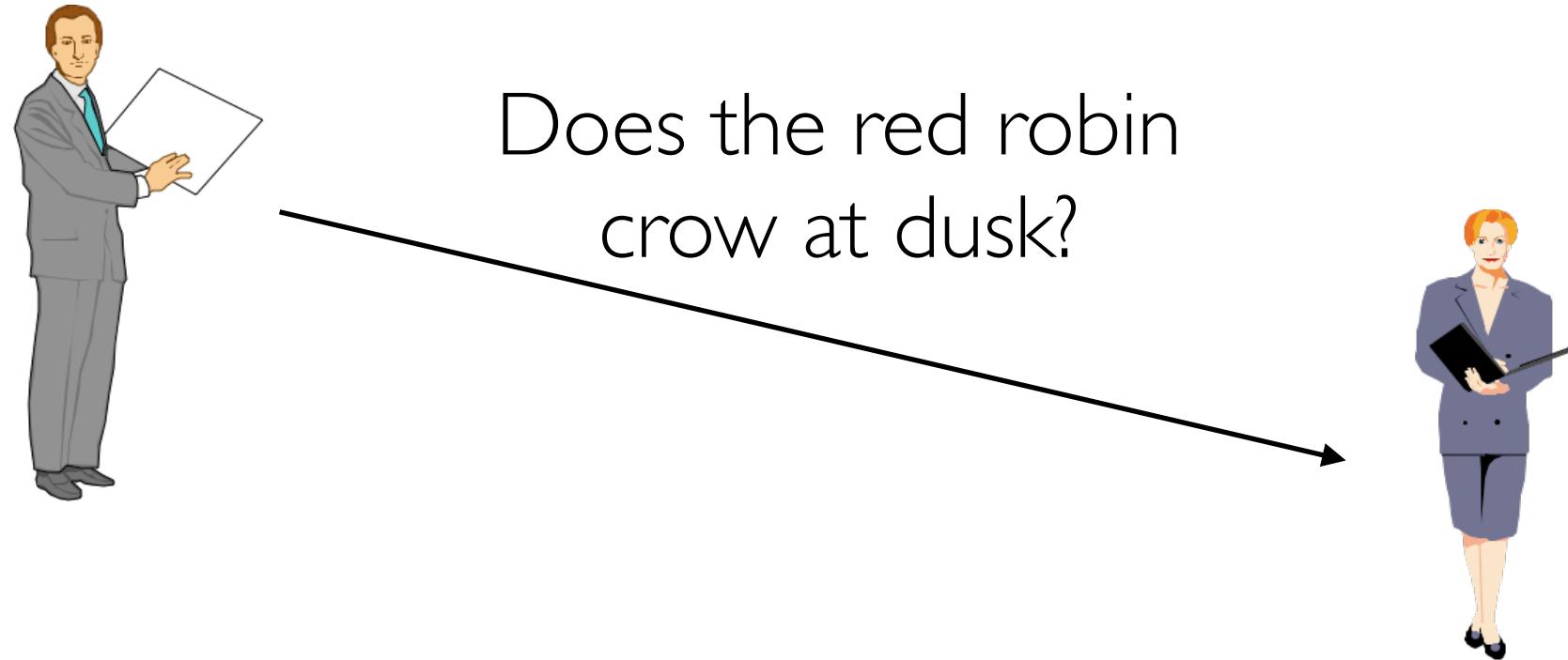
Communication between peer layers on different systems is defined by **protocols**

What is a Protocol?

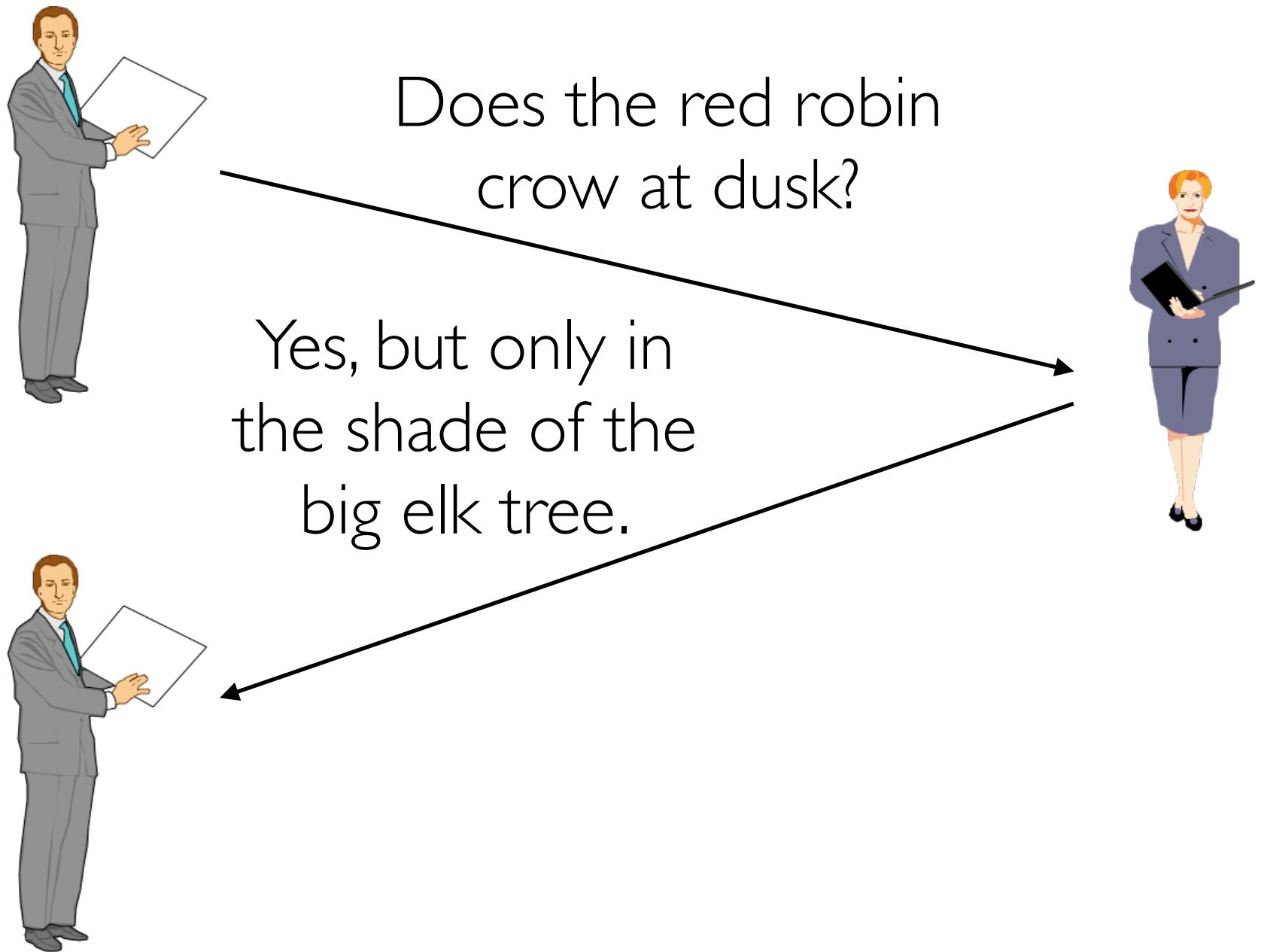
What is a Protocol?



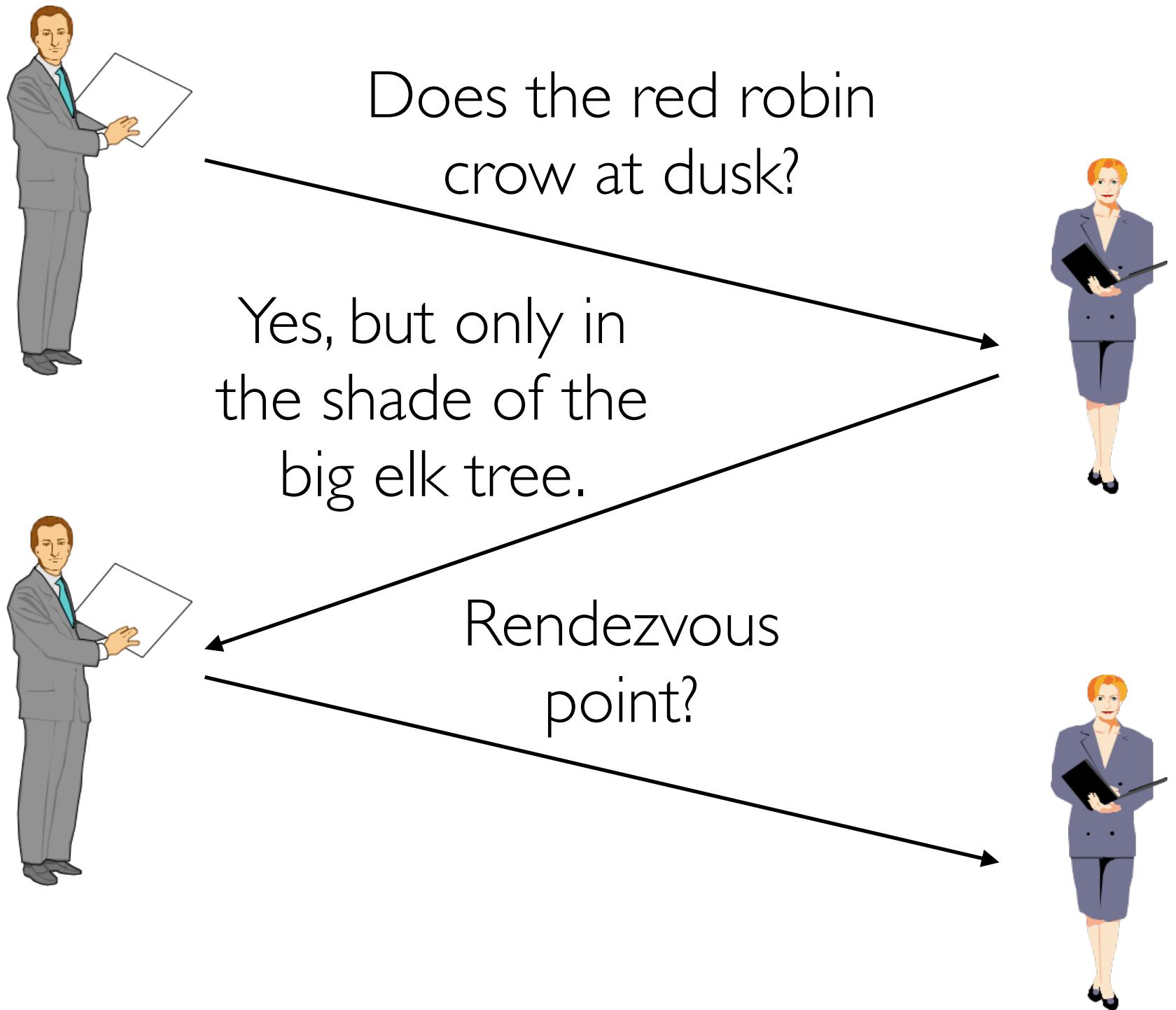
What is a Protocol?



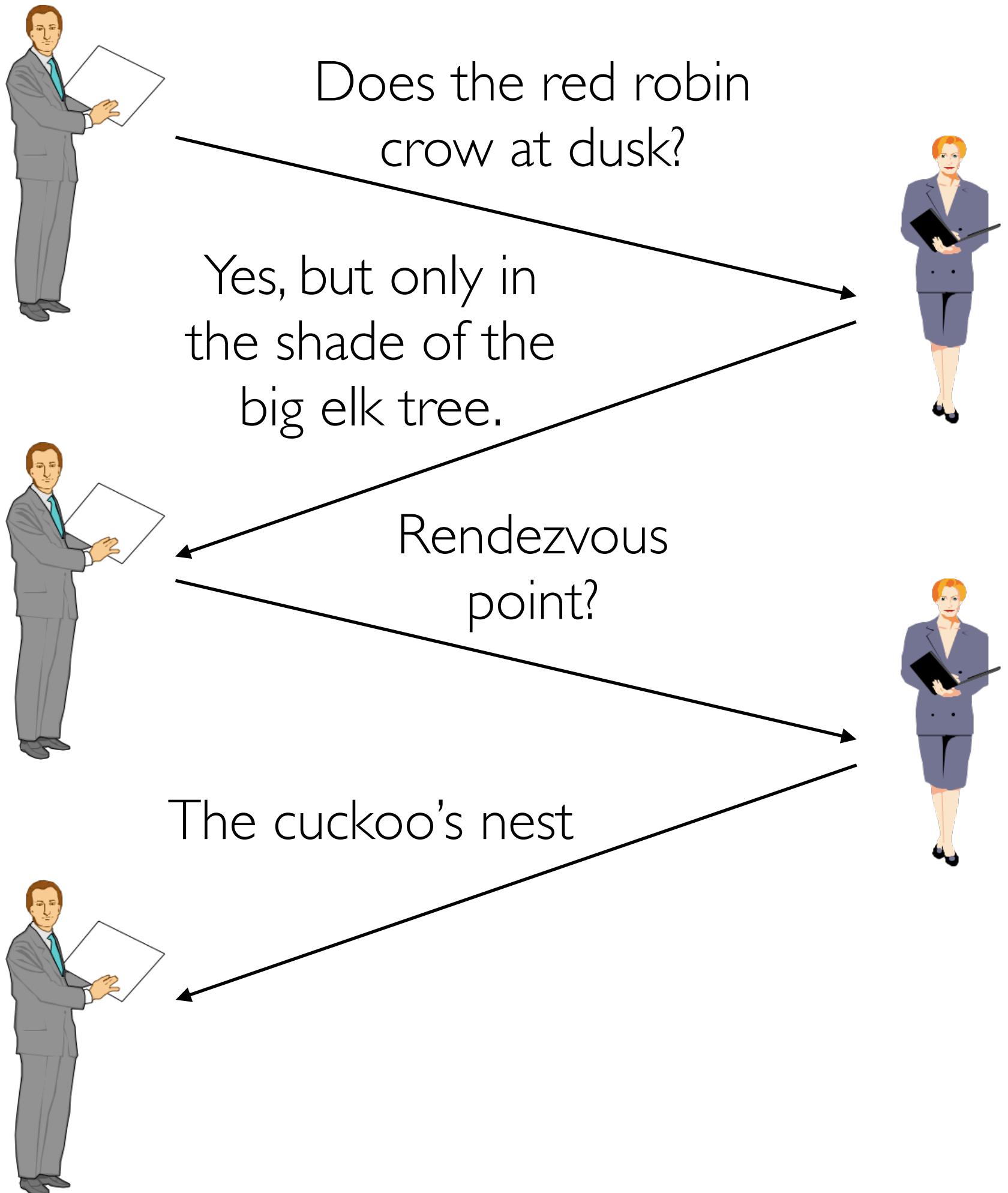
What is a Protocol?



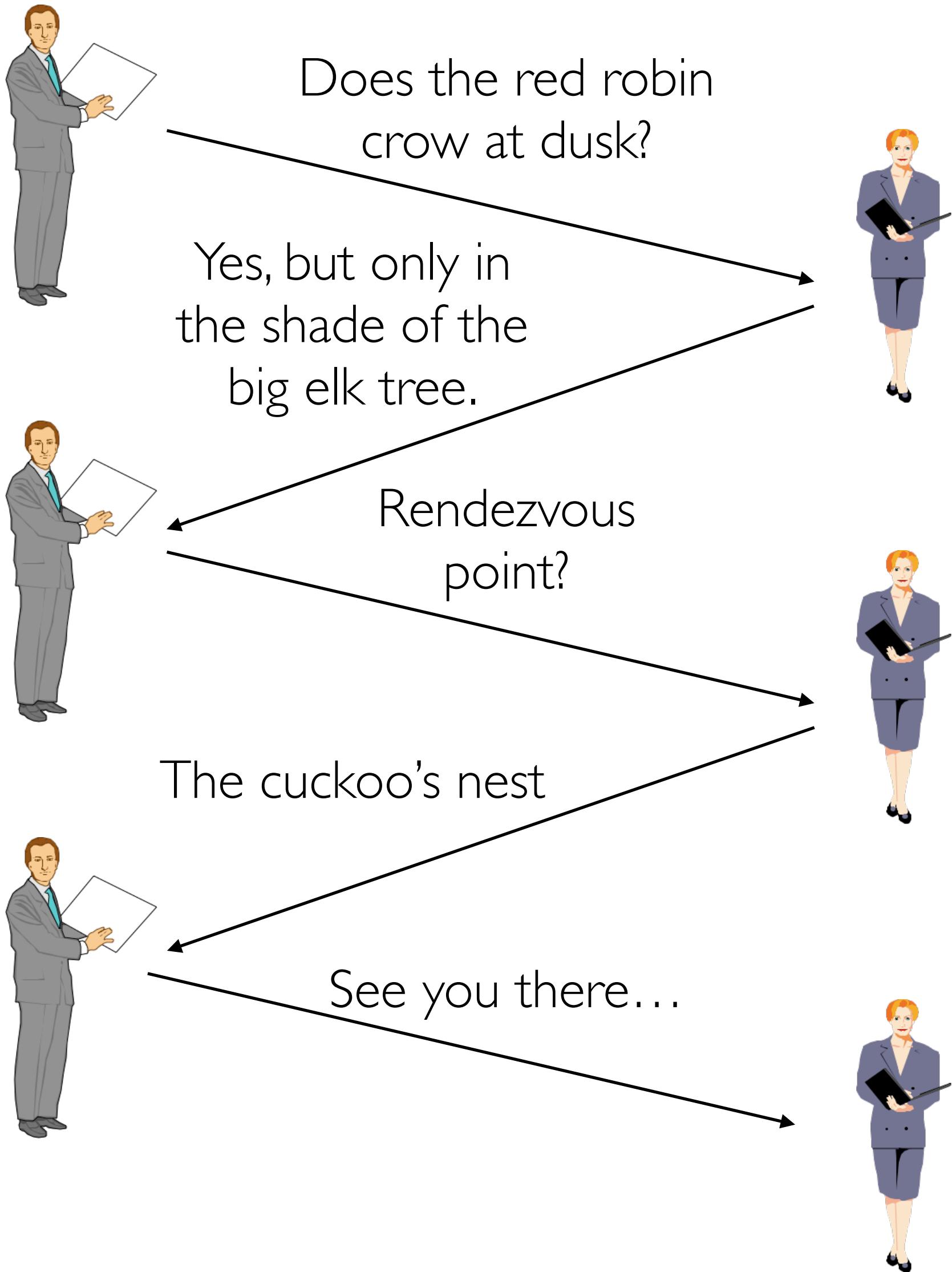
What is a Protocol?



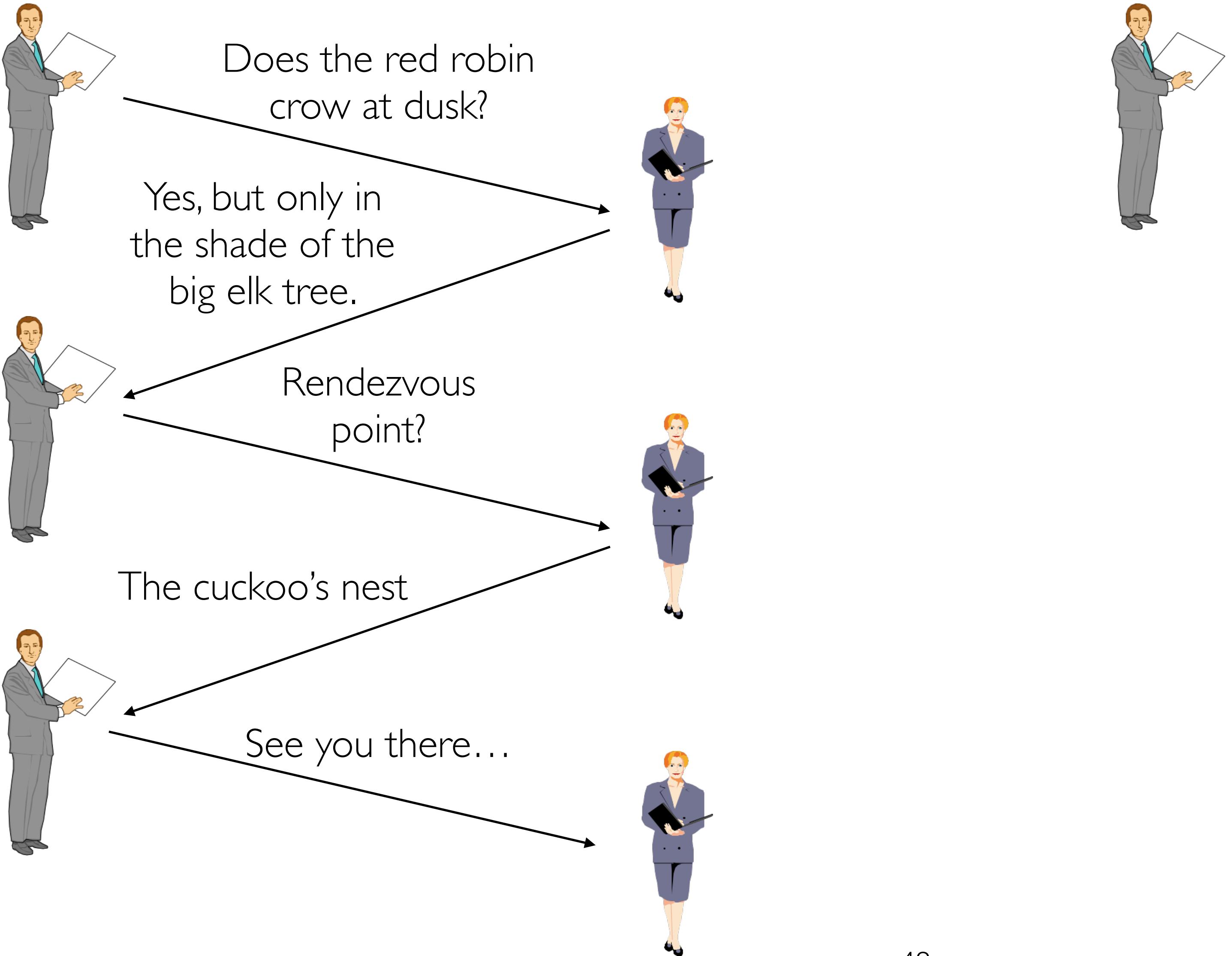
What is a Protocol?



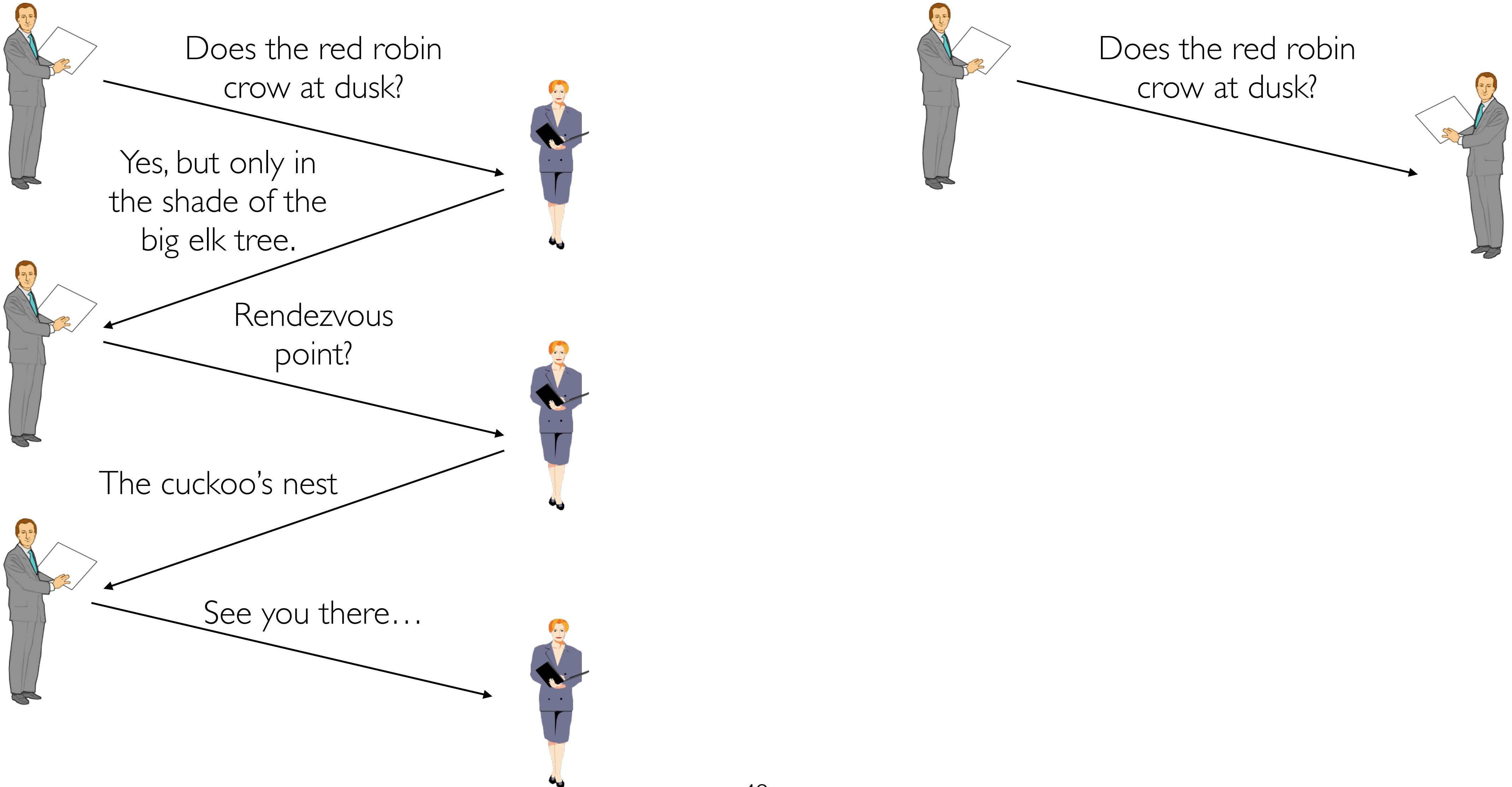
What is a Protocol?



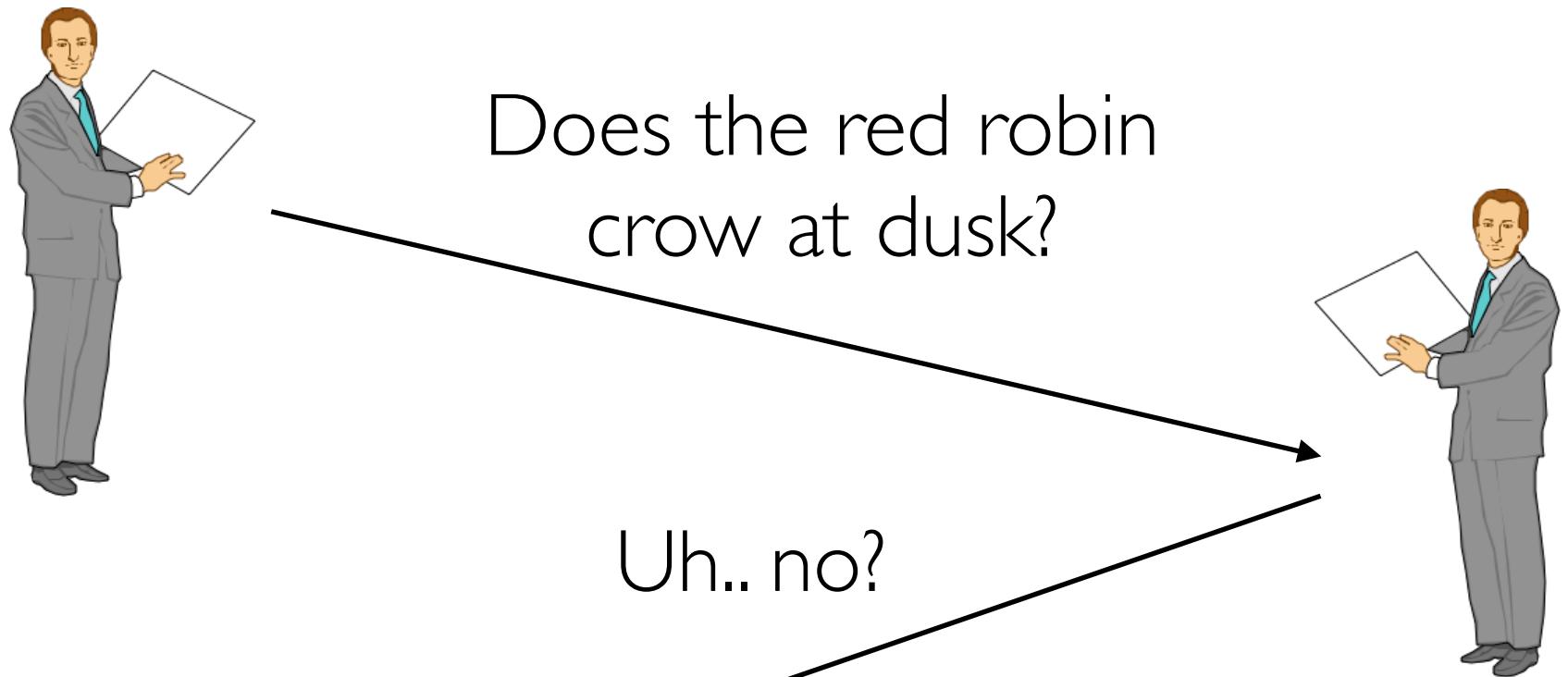
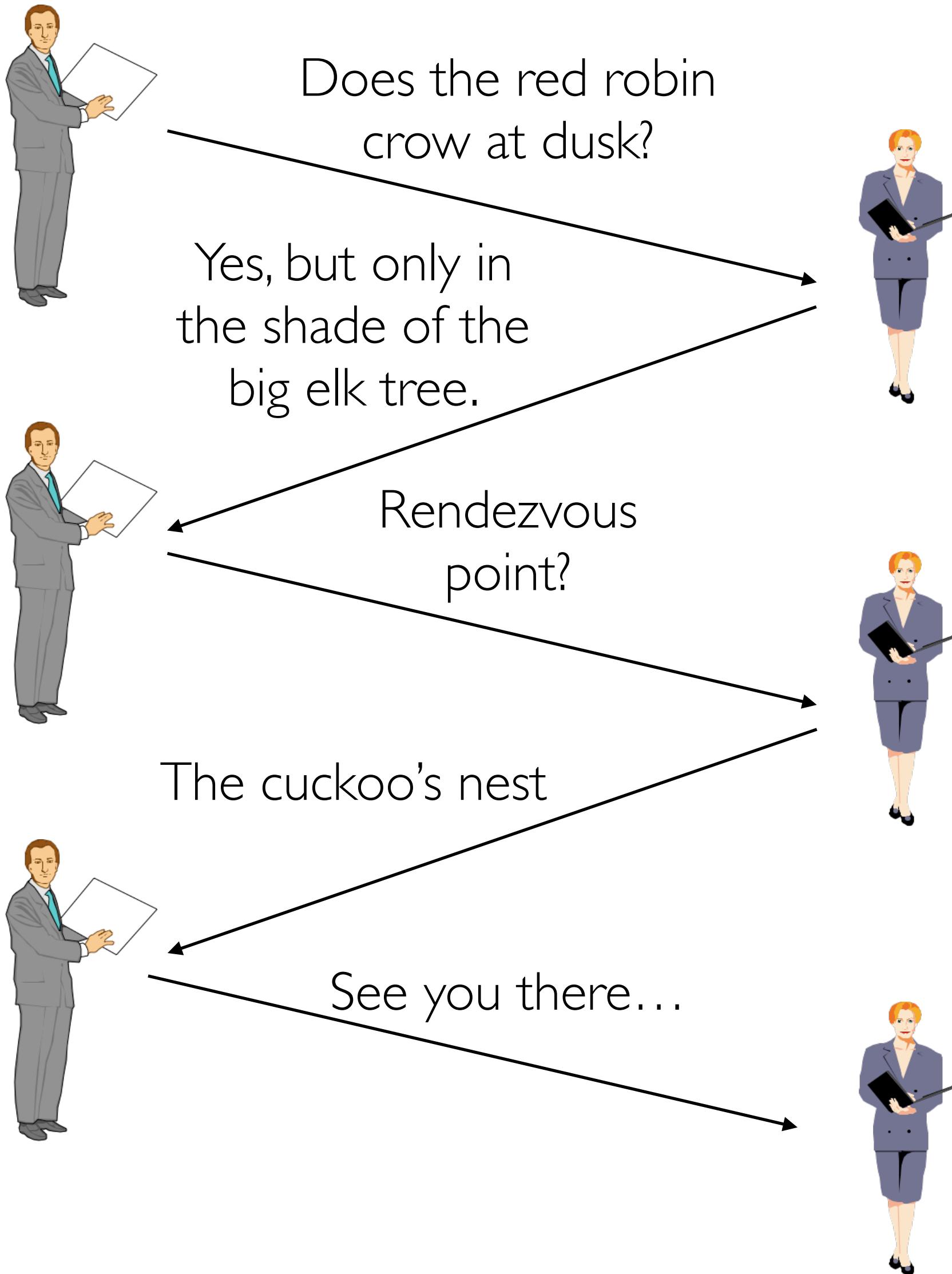
What is a Protocol?



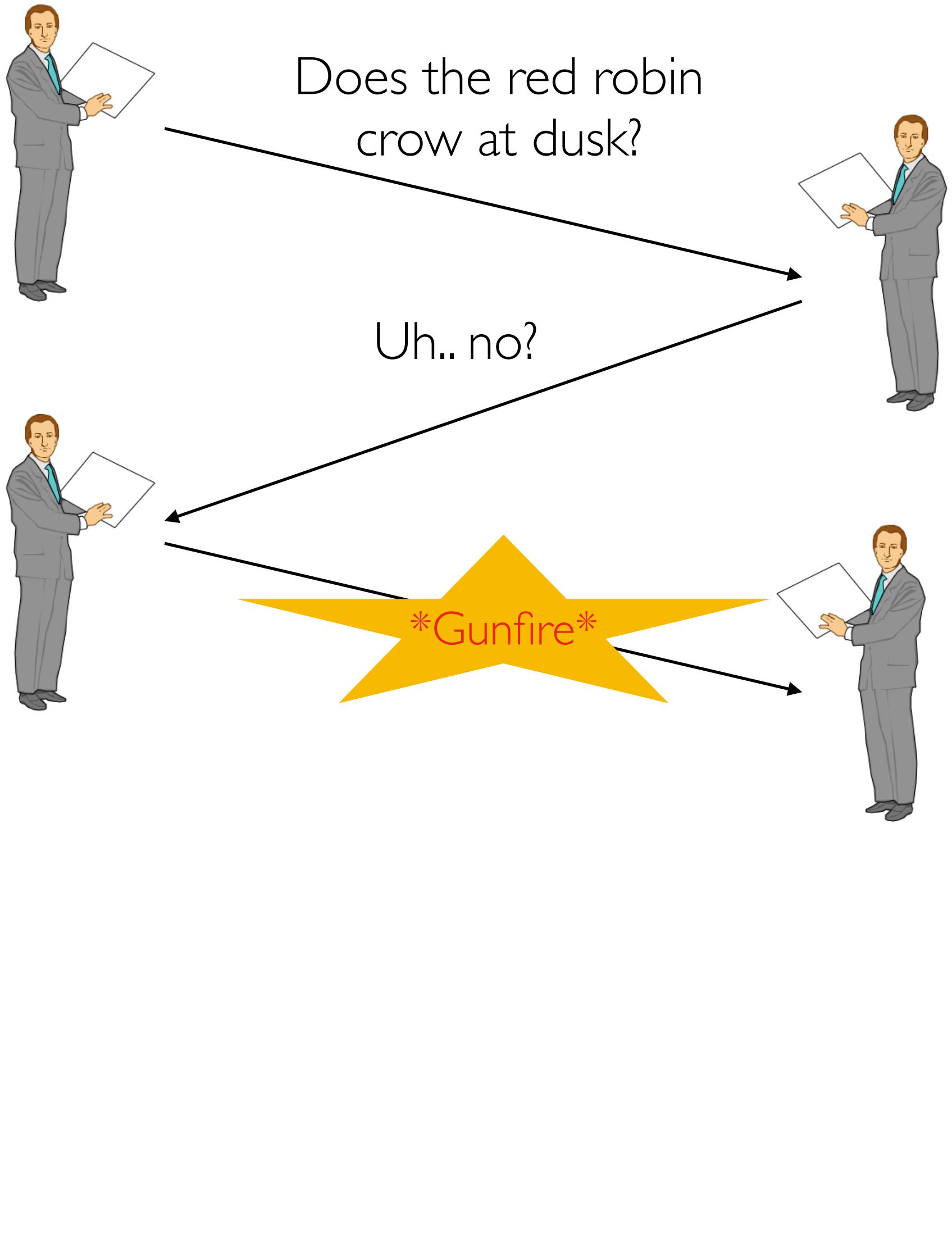
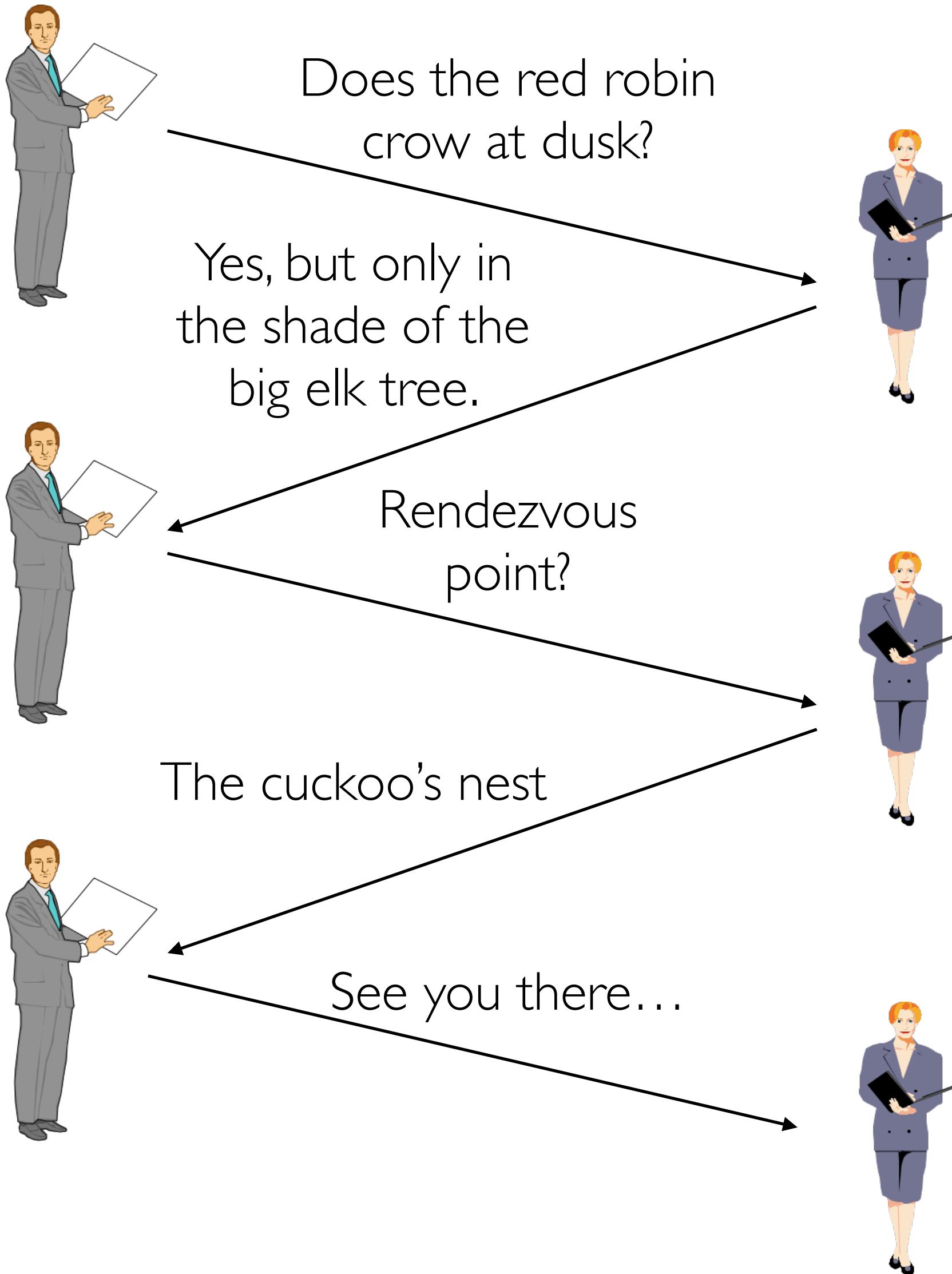
What is a Protocol?



What is a Protocol?



What is a Protocol?



What is a Protocol?

What is a Protocol?

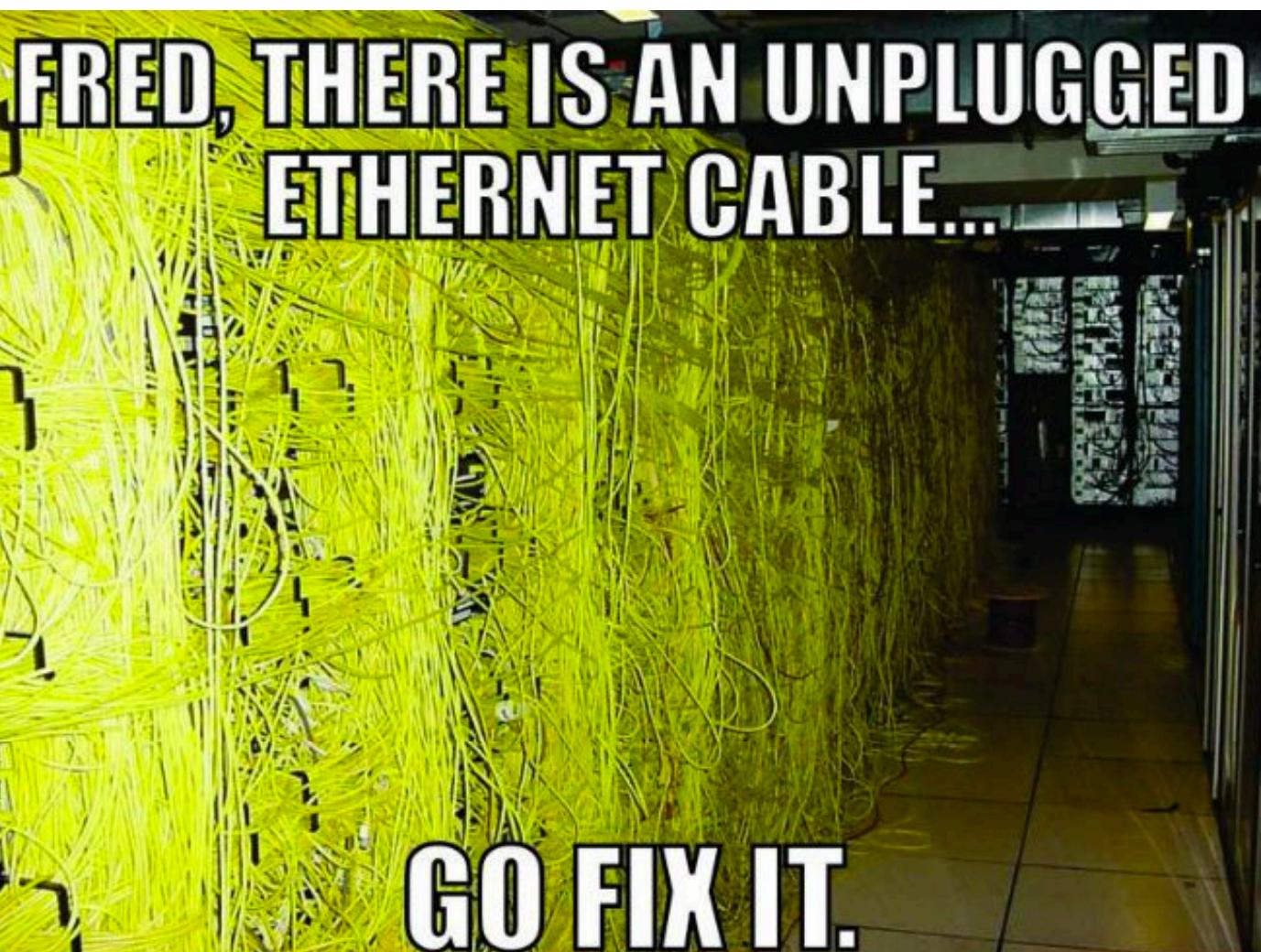
- An **agreement** between parties on **how to communicate**

What is a Protocol?

- An **agreement** between parties on **how to communicate**
- Defines the **syntax** of communication

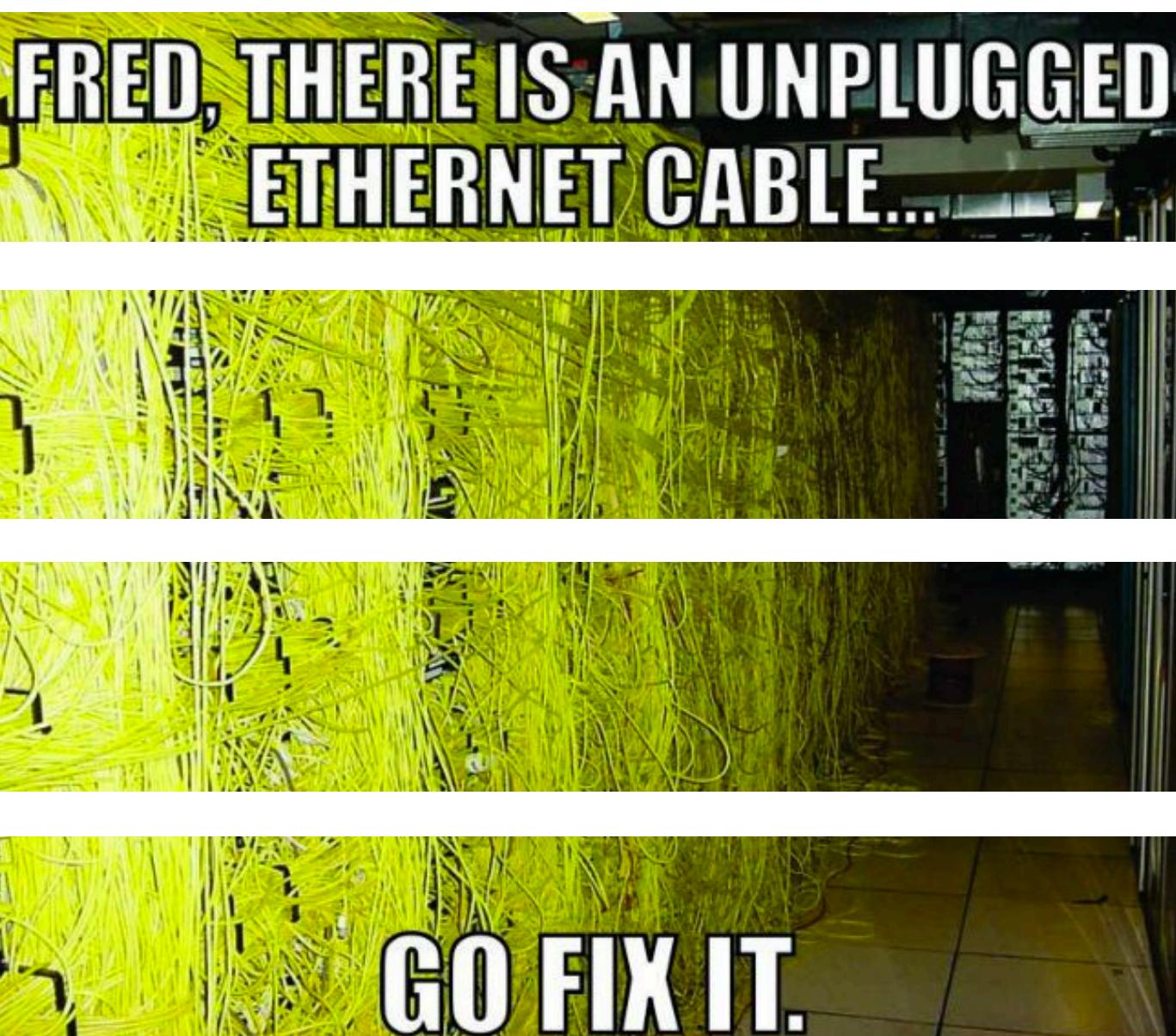
What is a Protocol?

- An **agreement** between parties on **how to communicate**
- Defines the **syntax** of communication



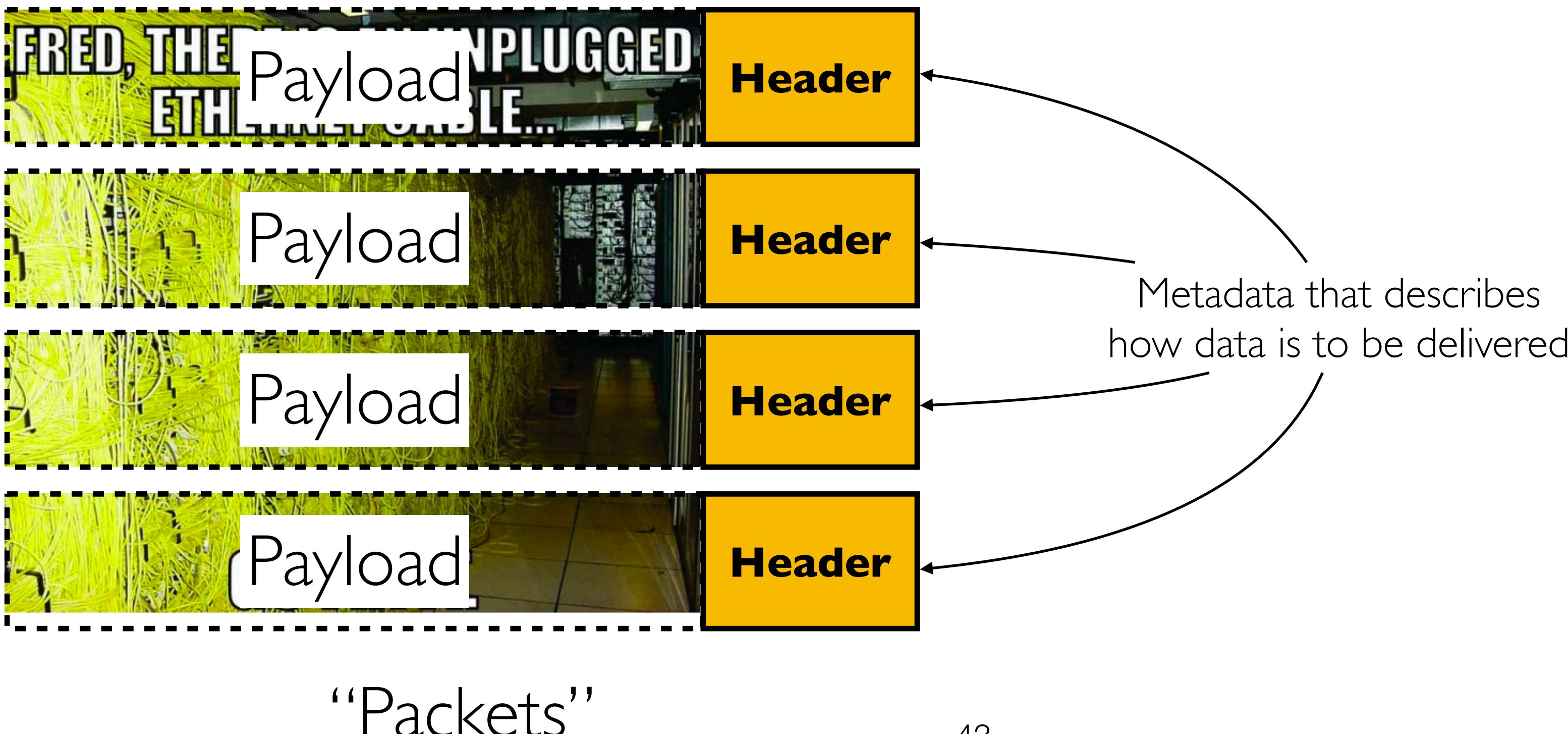
What is a Protocol?

- An **agreement** between parties on **how to communicate**
- Defines the **syntax** of communication



What is a Protocol?

- An **agreement** between parties on **how to communicate**
- Defines the **syntax** of communication



What is a Protocol?

- An **agreement** between parties on **how to communicate**
- Defines the **syntax** of communication
 - **Header:** instructions for how to process the payload



“Packets”

What is a Protocol?

- An **agreement** between parties on **how to communicate**
- Defines the **syntax** of communication
 - **Header:** instructions for how to process the payload
 - Each protocol defines the format of the packet **headers**



“Packets”

What is a Protocol?

- An **agreement** between parties on **how to communicate**
- Defines the **syntax** of communication
 - **Header:** instructions for how to process the payload
 - Each protocol defines the format of the packet **headers**
 - e.g., “the first 32 bits carry the destination address”



“Packets”

What is a Protocol?

- An **agreement** between parties on **how to communicate**
- Defines the **syntax** of communication
- And semantics

What is a Protocol?

- An **agreement** between parties on **how to communicate**
- Defines the **syntax** of communication
- And semantics
 - “First a hullo, then a request...”

What is a Protocol?

- An **agreement** between parties on **how to communicate**
- Defines the **syntax** of communication
- And semantics
 - “First a hullo, then a request...”
 - We’ll study many protocols later in the semester

What is a Protocol?

- An **agreement** between parties on **how to communicate**
- Defines the **syntax** of communication
- And semantics
 - “First a hullo, then a request...”
 - We’ll study many protocols later in the semester
- Protocols exist at many levels, hardware and software

What is a Protocol?

- An **agreement** between parties on **how to communicate**
- Defines the **syntax** of communication
- And semantics
 - “First a hullo, then a request...”
 - We’ll study many protocols later in the semester
- Protocols exist at many levels, hardware and software
 - Defined by a variety of standards (IETF, IEEE, ITU)

Protocols at different Layers



Protocols at different Layers



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

Protocols at different Layers



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

Implications of Hourglass

Implications of Hourglass

- Single network-layer protocol (IP)

Implications of Hourglass

- Single network-layer protocol (IP)
- Allows arbitrary networks to interoperate
 - Any network that supports IP can exchange packets

Implications of Hourglass

- Single network-layer protocol (IP)
- Allows arbitrary networks to interoperate
 - Any network that supports IP can exchange packets
- Decouples applications from low-level networking technologies
 - Applications function on all networks

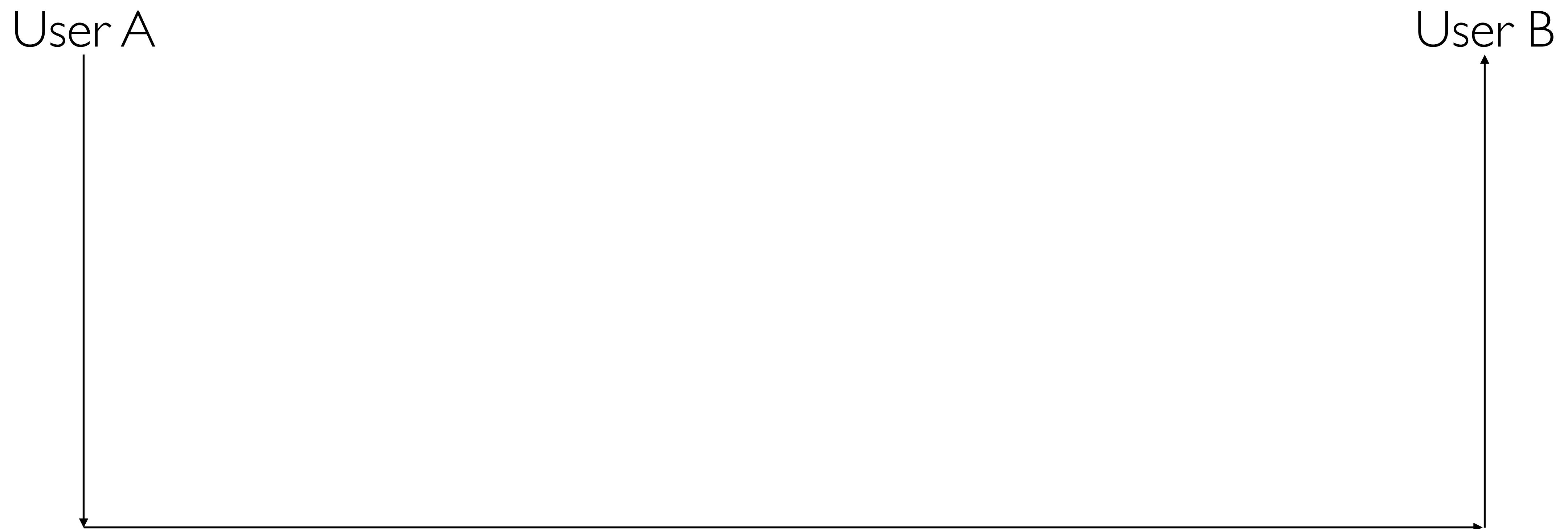
Implications of Hourglass

- Single network-layer protocol (IP)
- Allows arbitrary networks to interoperate
 - Any network that supports IP can exchange packets
- Decouples applications from low-level networking technologies
 - Applications function on all networks
- Supports simultaneous innovations above and below IP layer

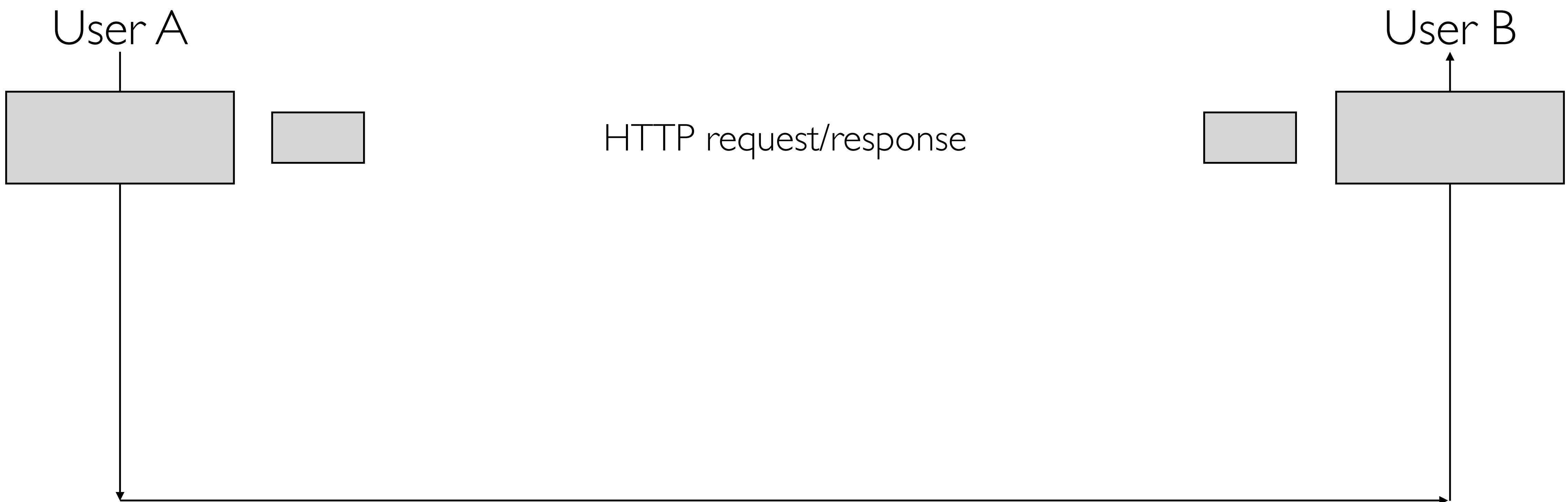
Implications of Hourglass

- Single network-layer protocol (IP)
- Allows arbitrary networks to interoperate
 - Any network that supports IP can exchange packets
- Decouples applications from low-level networking technologies
 - Applications function on all networks
- Supports simultaneous innovations above and below IP layer
- But changing IP itself is hard (e.g., IPv4 to IPv6)

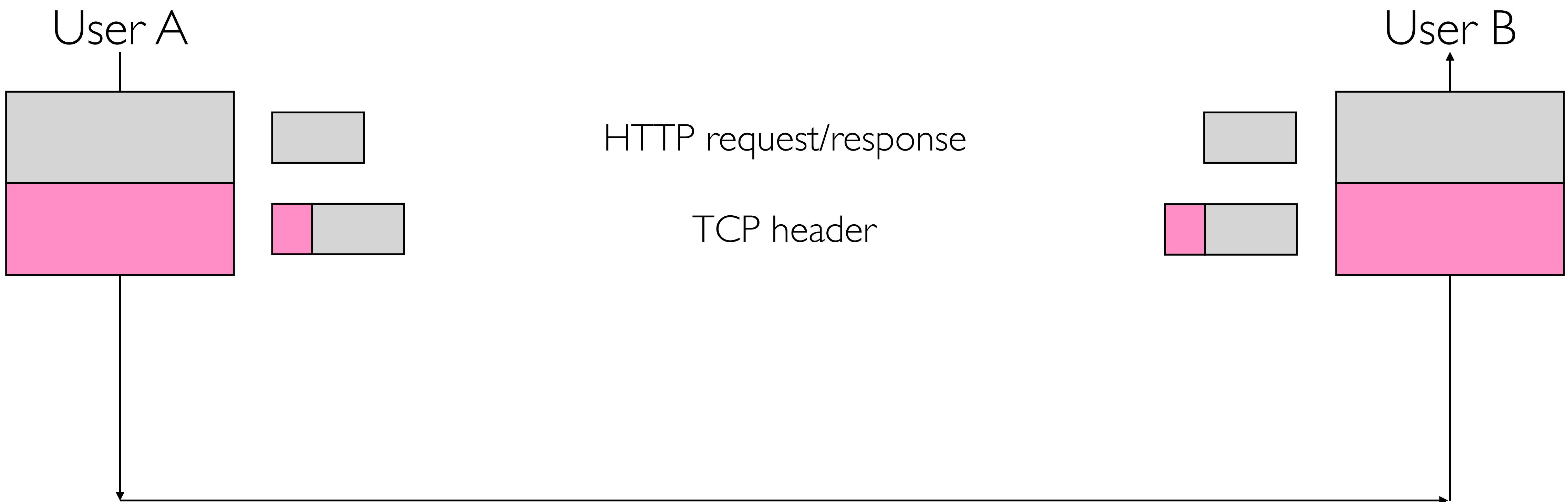
Layer Encapsulation: Protocol Headers



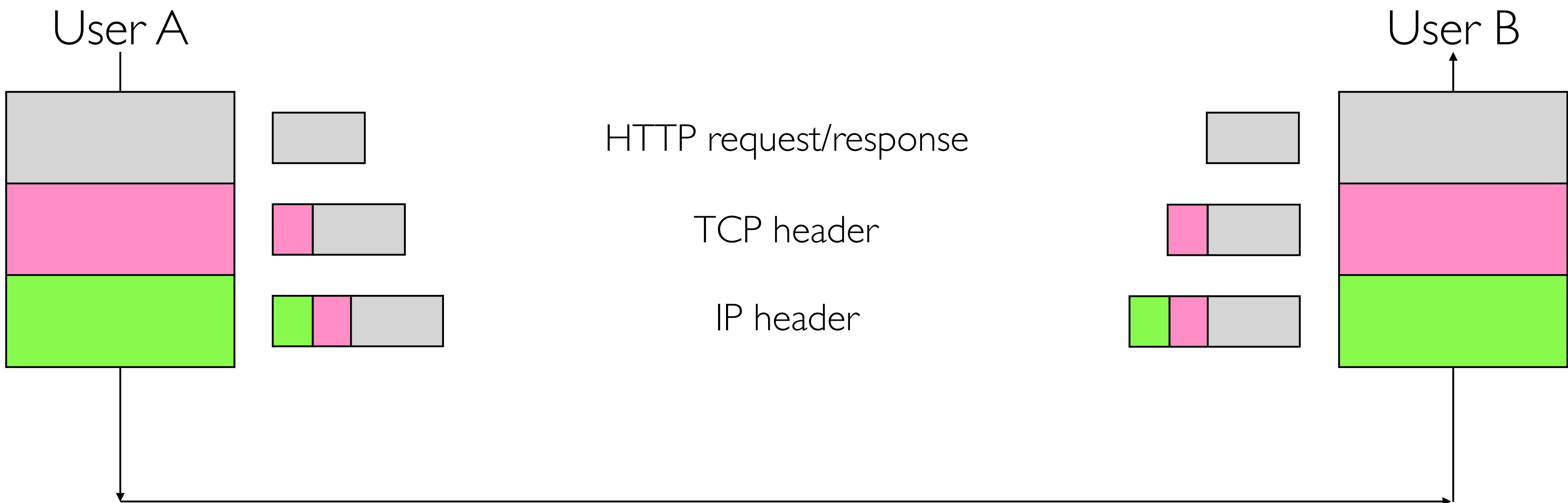
Layer Encapsulation: Protocol Headers



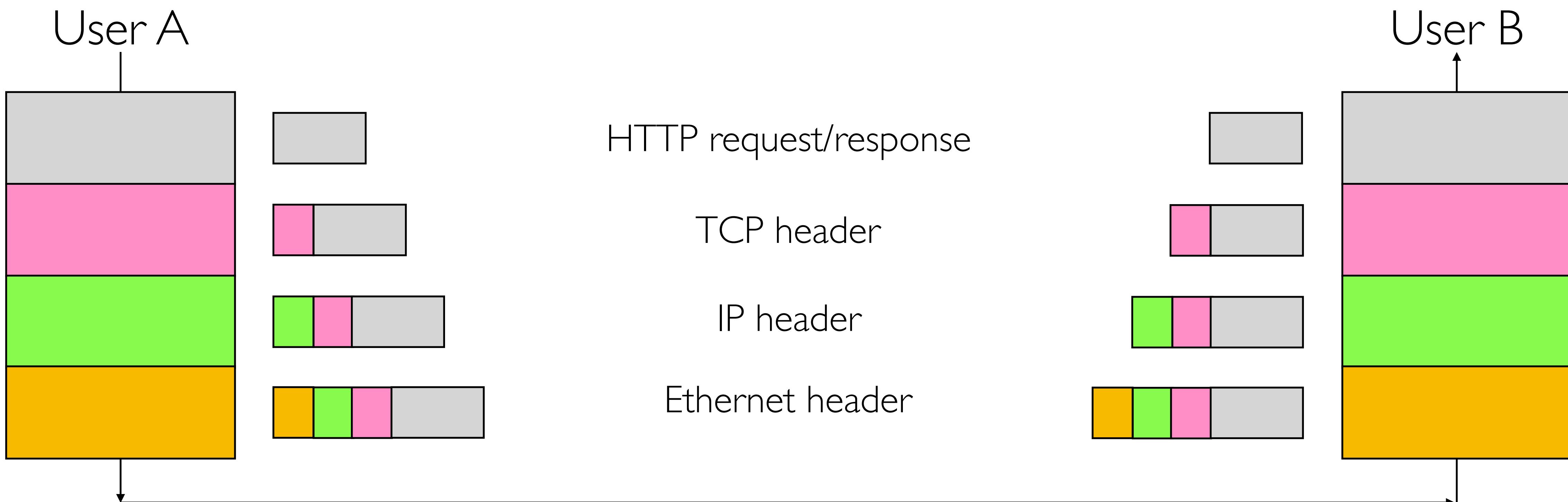
Layer Encapsulation: Protocol Headers



Layer Encapsulation: Protocol Headers



Layer Encapsulation: Protocol Headers



Why Layering?

Why Layering?

- Reduce complexity
 - A layer only needs to worry about its own function

Why Layering?

- Reduce complexity
 - A layer only needs to worry about its own function
- Improve flexibility
 - Or rather, **modularity**: change function of layer without affecting other layers

Why Not?

Why Not?

- Sub-optimal performance

Why Not?

- Sub-optimal performance
- May duplicate functionality
 - *E.g., error recovery at multiple layers*

Why Not?

- Sub-optimal performance
- May duplicate functionality
 - *E.g., error recovery at multiple layers*
- Cross-layer information often useful
 - Several “layer violations” in practice
 - *E.g., one layer may need timestamp information that is only present in another layer*

Questions?

Three steps

- **Decomposition**
- **Organization**
- **Assignment**

The Path through FedEx

FE = FedEx Envelope

Truck

Sorting Office

Airport

Truck

Sorting Office

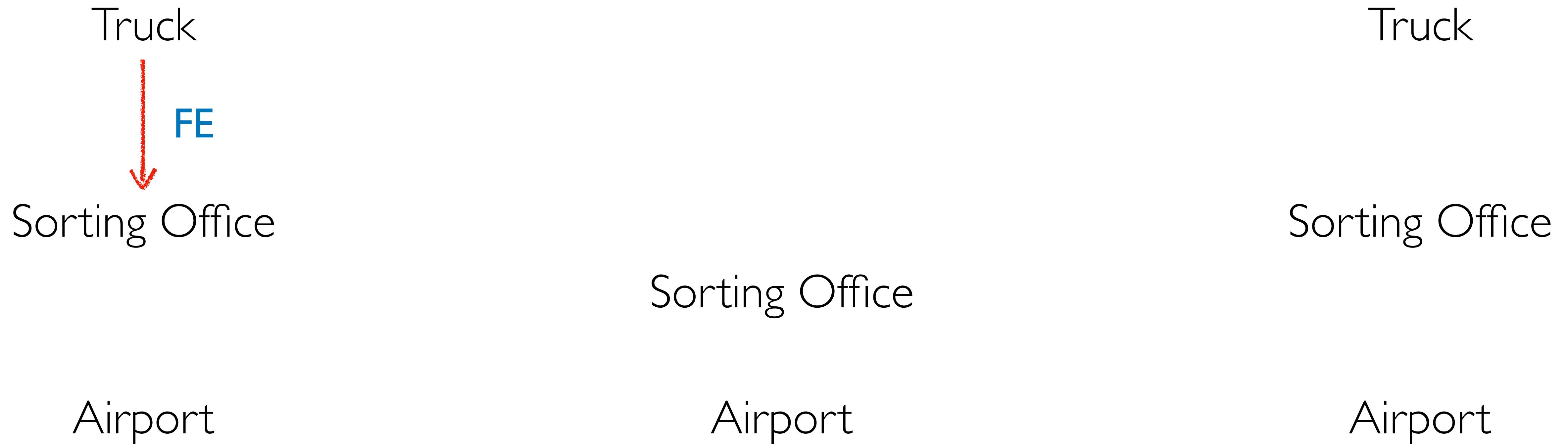
Sorting Office

Airport

Airport

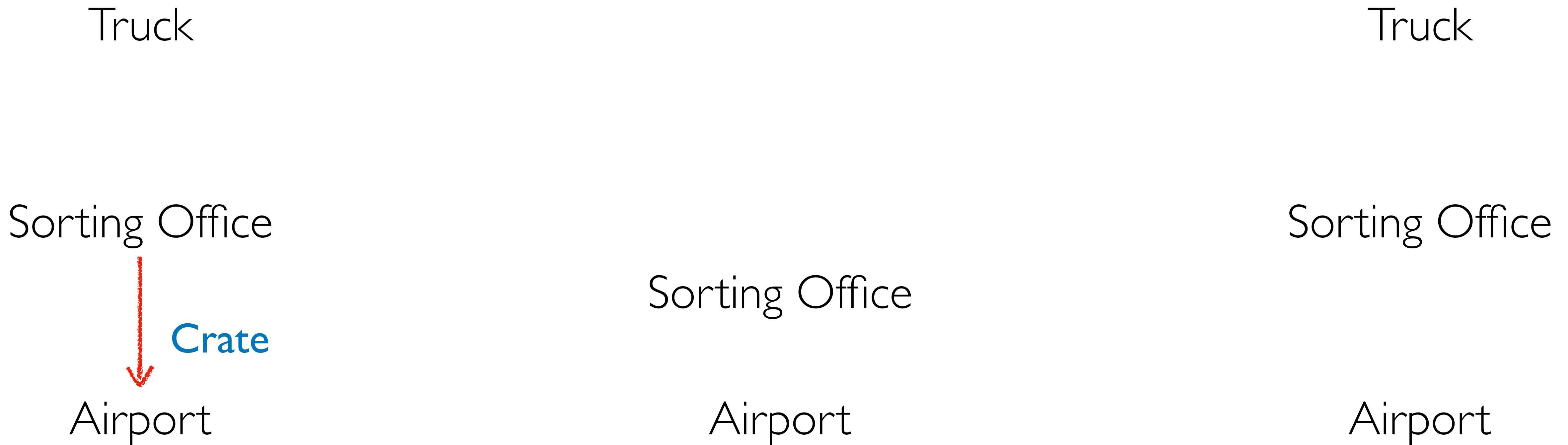
The Path through FedEx

FE = FedEx Envelope



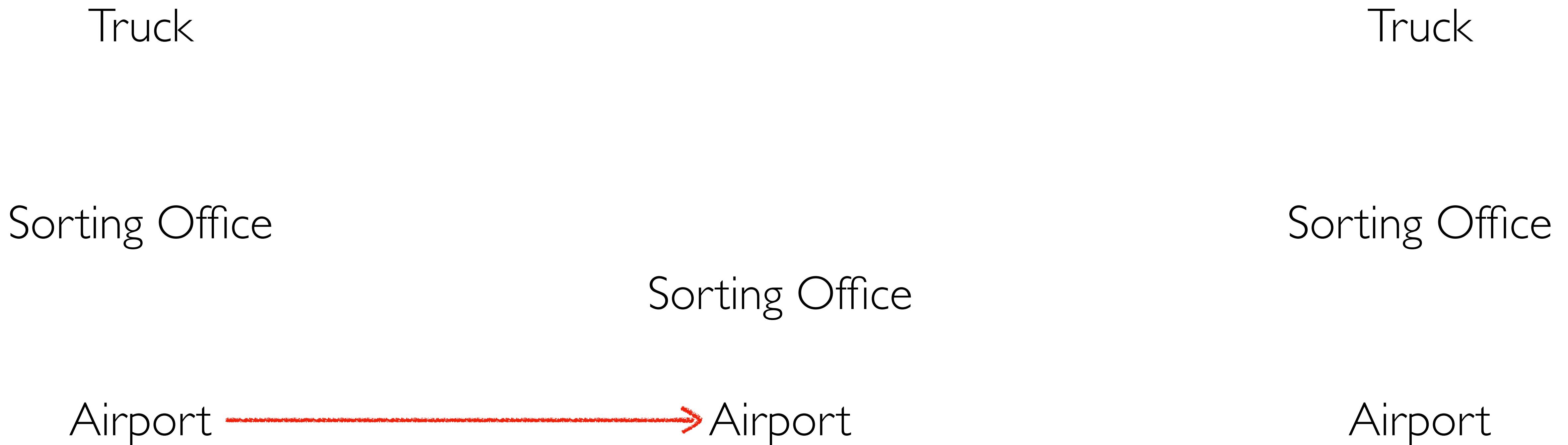
The Path through FedEx

FE = FedEx Envelope



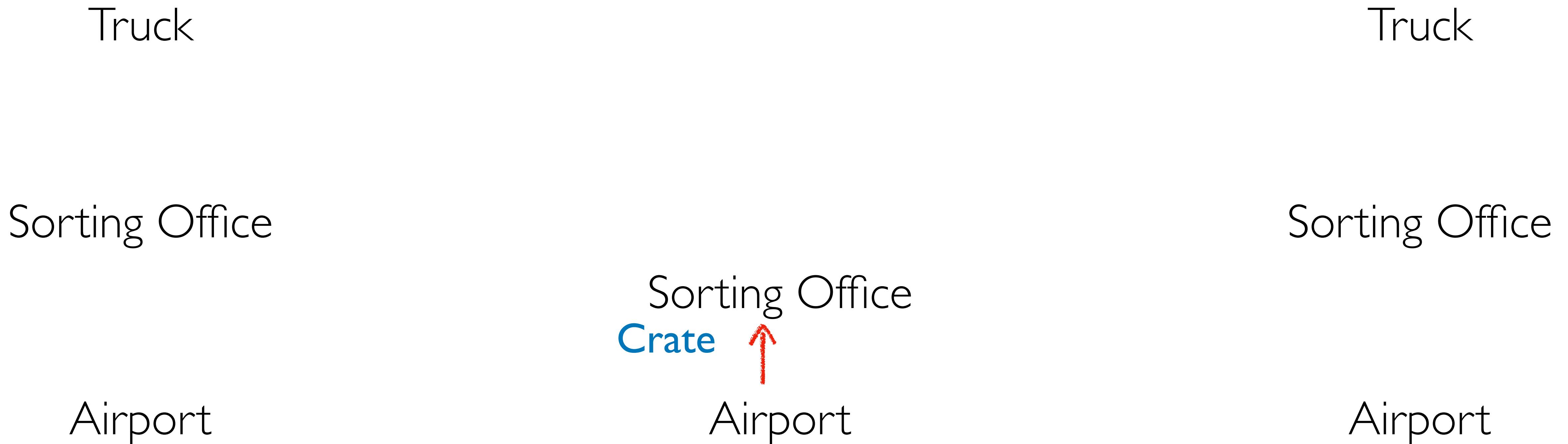
The Path through FedEx

FE = FedEx Envelope



The Path through FedEx

FE = FedEx Envelope



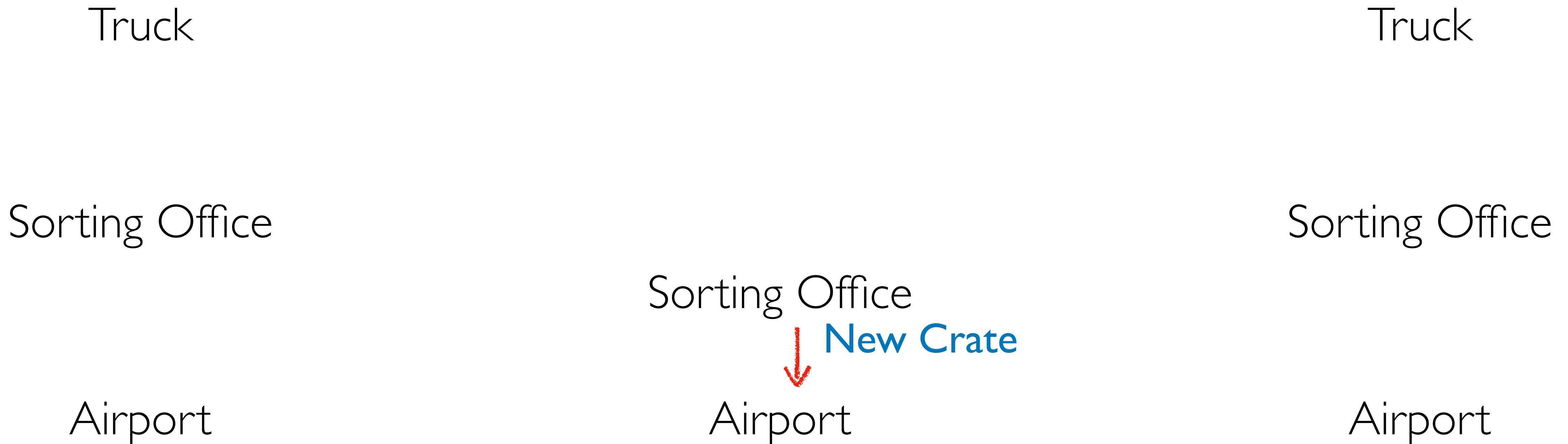
The Path through FedEx

FE = FedEx Envelope



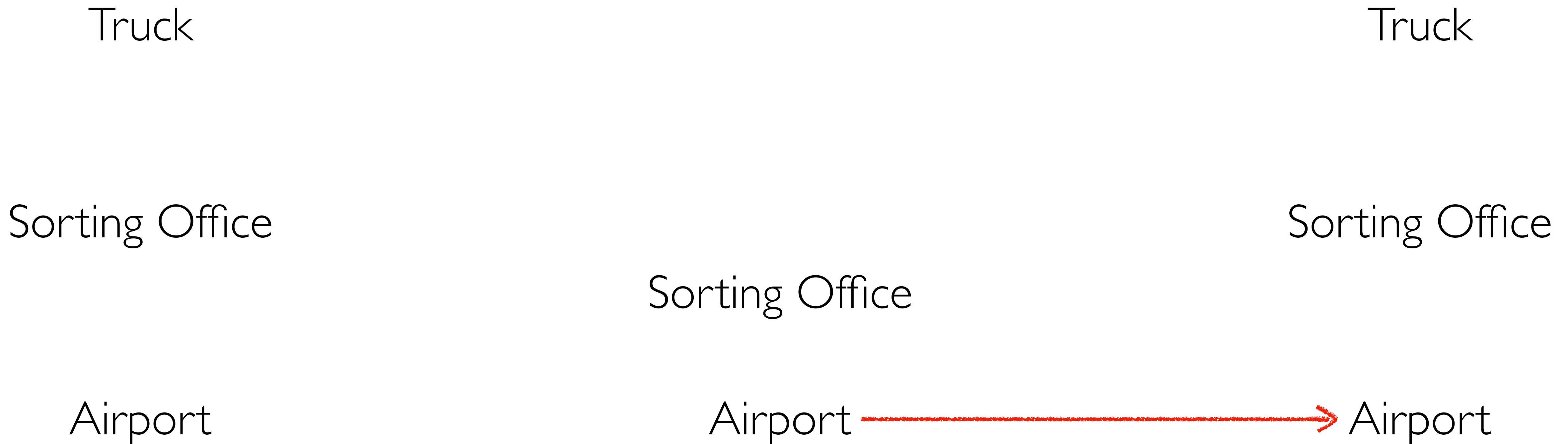
The Path through FedEx

FE = FedEx Envelope



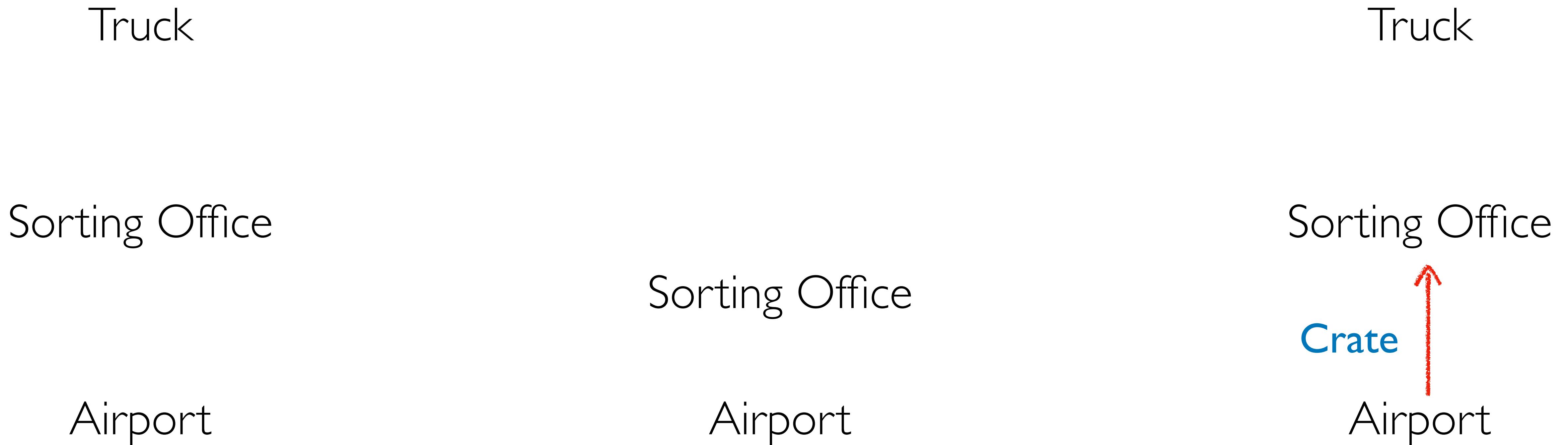
The Path through FedEx

FE = FedEx Envelope



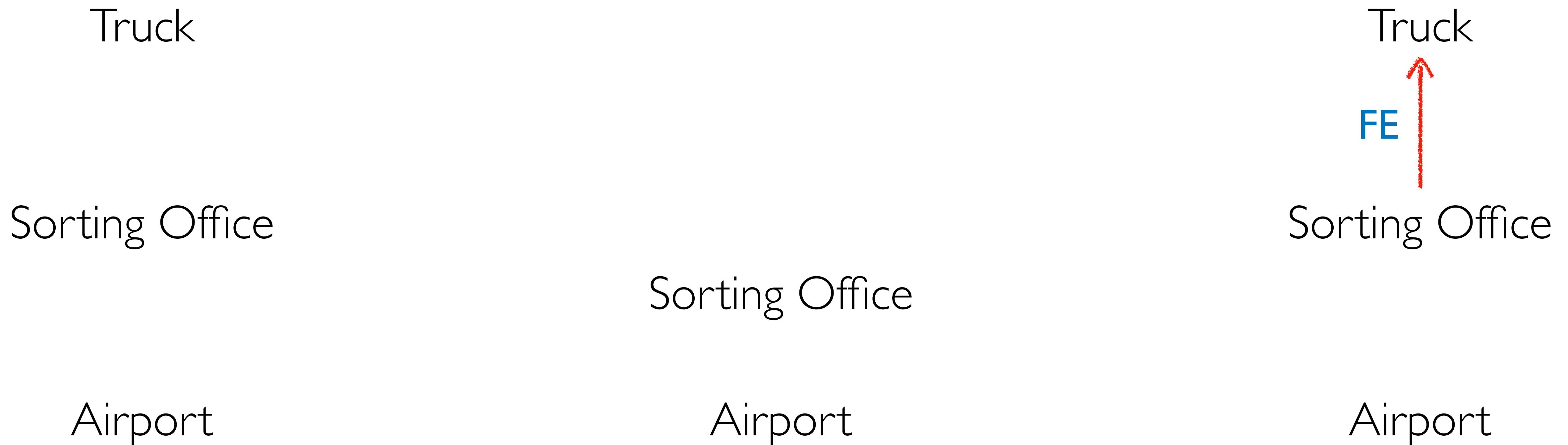
The Path through FedEx

FE = FedEx Envelope



The Path through FedEx

FE = FedEx Envelope



The Path through FedEx

FE = FedEx Envelope

Truck

Sorting Office

Airport

Truck

Sorting Office

Sorting Office

Airport

Airport

The Path through FedEx

FE = FedEx Envelope

Truck

Truck

Sorting Office

Sorting Office

Sorting Office

Airport

Airport

Airport

Deepest packaging (Envelope+FE+Crate) at Lowest level of transport

The Path through FedEx

Higher “Stack”
at Ends

Truck

Sorting Office

Airport

Deepest packaging (Envelope+FE+Crate) at Lowest level of transport

FE = FedEx Envelope

Truck

Sorting Office

Sorting Office

Airport

Airport

The Path through FedEx

Higher “Stack”
at Ends

Truck

Sorting Office

Airport

Deepest packaging (Envelope+FE+Crate) at Lowest level of transport

FE = FedEx Envelope

Partial “Stack”
during Transit

Sorting Office

Airport

Sorting Office

Airport

Truck

Path through Internet



Path through Internet



What gets implemented where?

What layers get implemented at the end-systems?

What layers get implemented at the end-systems?

- Bits arrive on wire, must make it up to application

What layers get implemented at the end-systems?

- Bits arrive on wire, must make it up to application
- Therefore, all layers must exist at host!

What layers get implemented in the network?

What layers get implemented in the network?

- Bits arrive on wire → physical layer (L1)

What layers get implemented in the network?

- Bits arrive on wire → physical layer (L1)
- Packets must be delivered across link and local networks → datalink layer (L2)

What layers get implemented in the network?

- Bits arrive on wire → physical layer (L1)
- Packets must be delivered across link and local networks → datalink layer (L2)
- Packets must be delivered b/w networks for global delivery → network layer (L3)

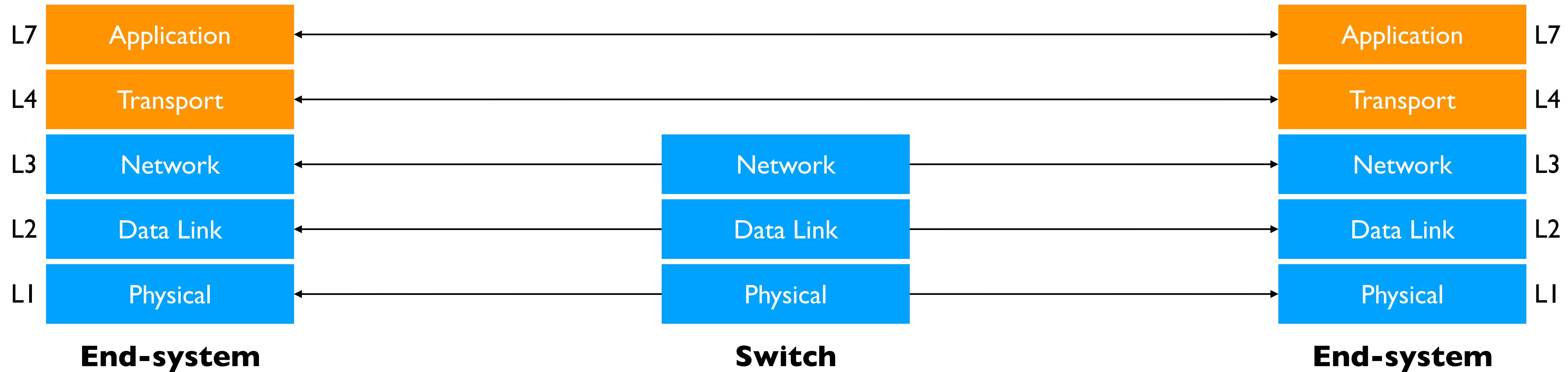
What layers get implemented in the network?

- Bits arrive on wire → physical layer (L1)
- Packets must be delivered across link and local networks → datalink layer (L2)
- Packets must be delivered b/w networks for global delivery → network layer (L3)
- The network does not support reliable delivery

What layers get implemented in the network?

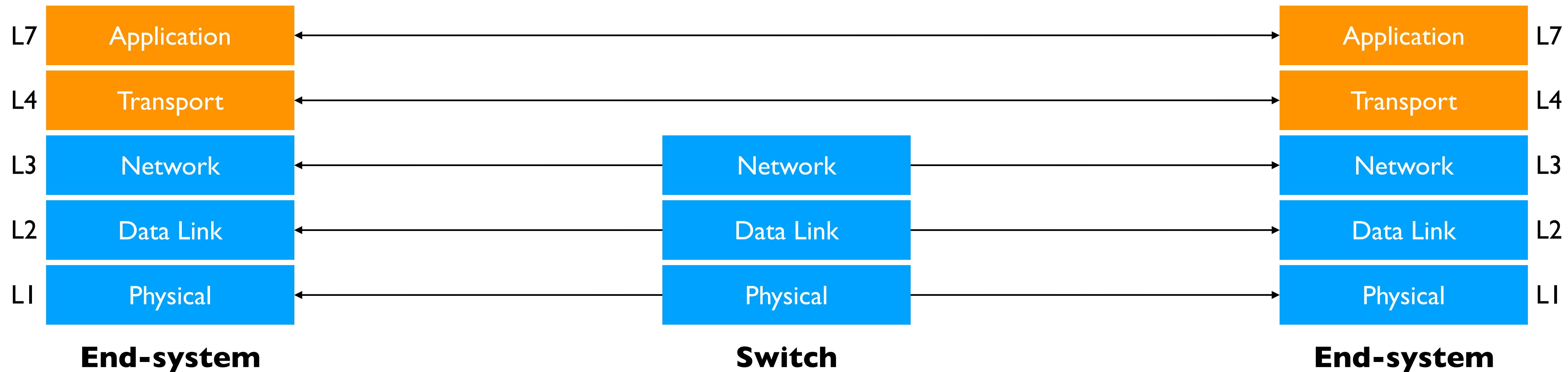
- Bits arrive on wire → physical layer (L1)
- Packets must be delivered across link and local networks → datalink layer (L2)
- Packets must be delivered b/w networks for global delivery → network layer (L3)
- The network does not support reliable delivery
 - *Transport layer (and above) not supported*

Path through Internet



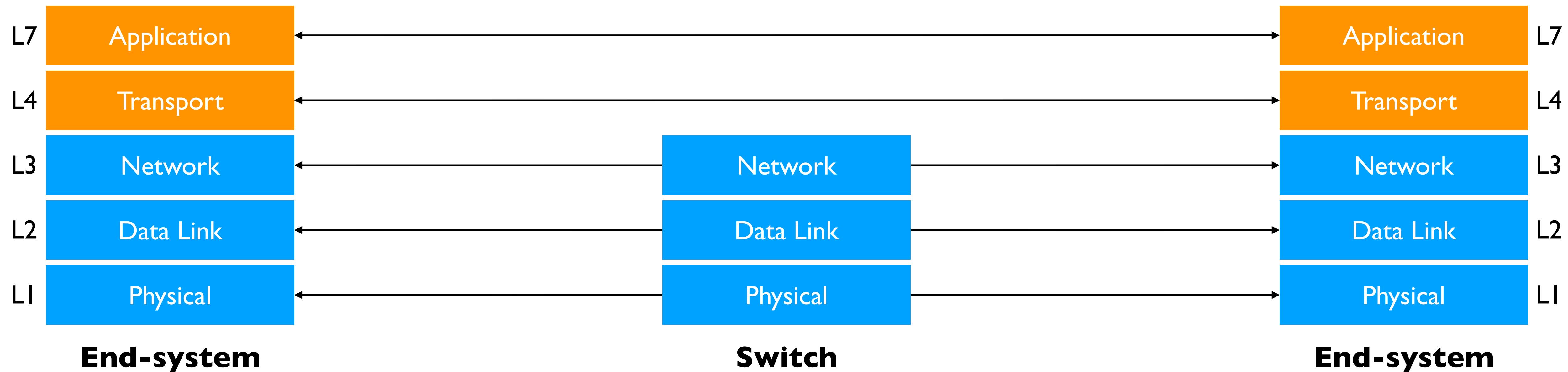
Path through Internet

- Lower three layers implemented everywhere



Path through Internet

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



A closer look: end-system

A closer look: end-system

- Application
 - *Web server, browser, mail, game*

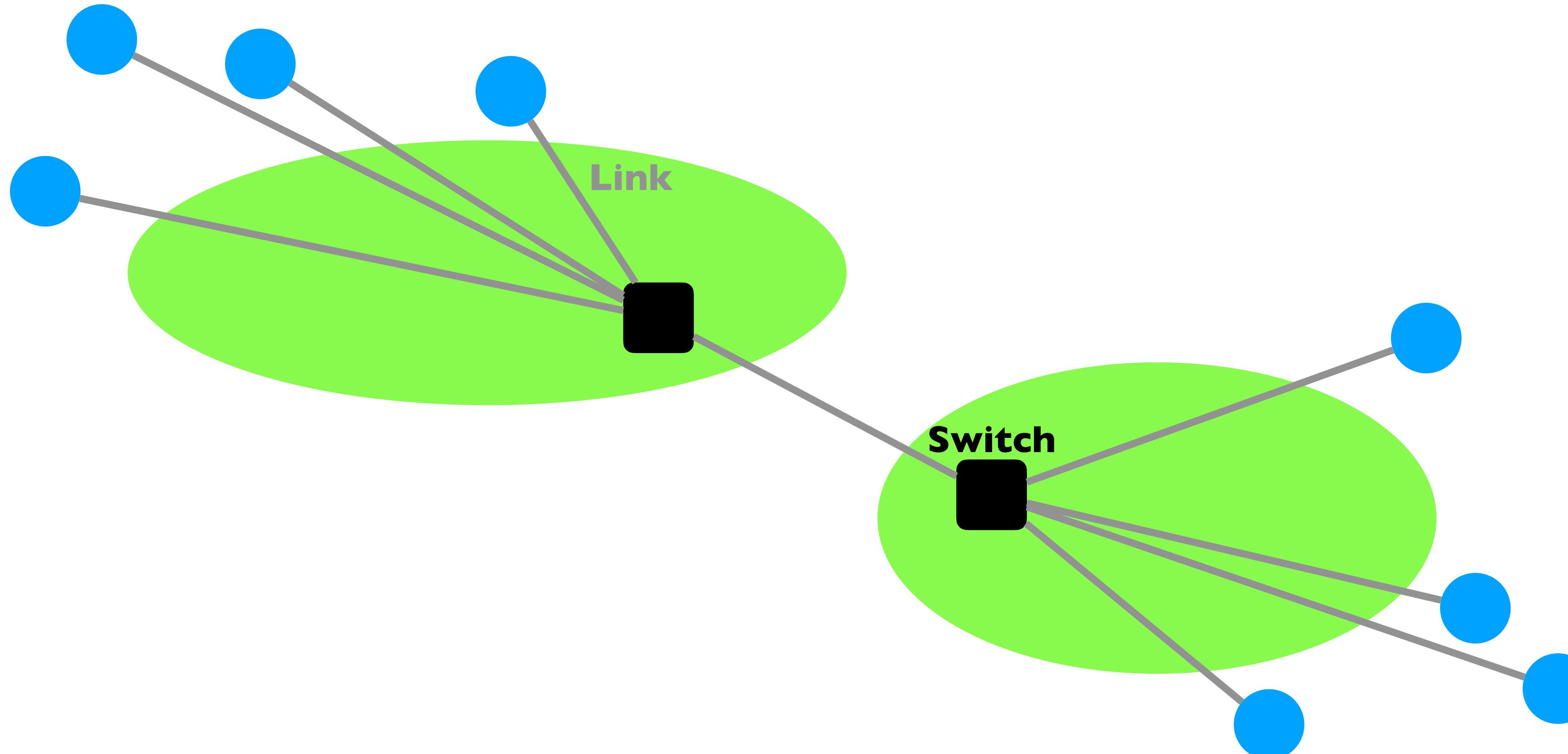
A closer look: end-system

- Application
 - *Web server, browser, mail, game*
- Transport and network layer
 - *Typically part of the operating system*

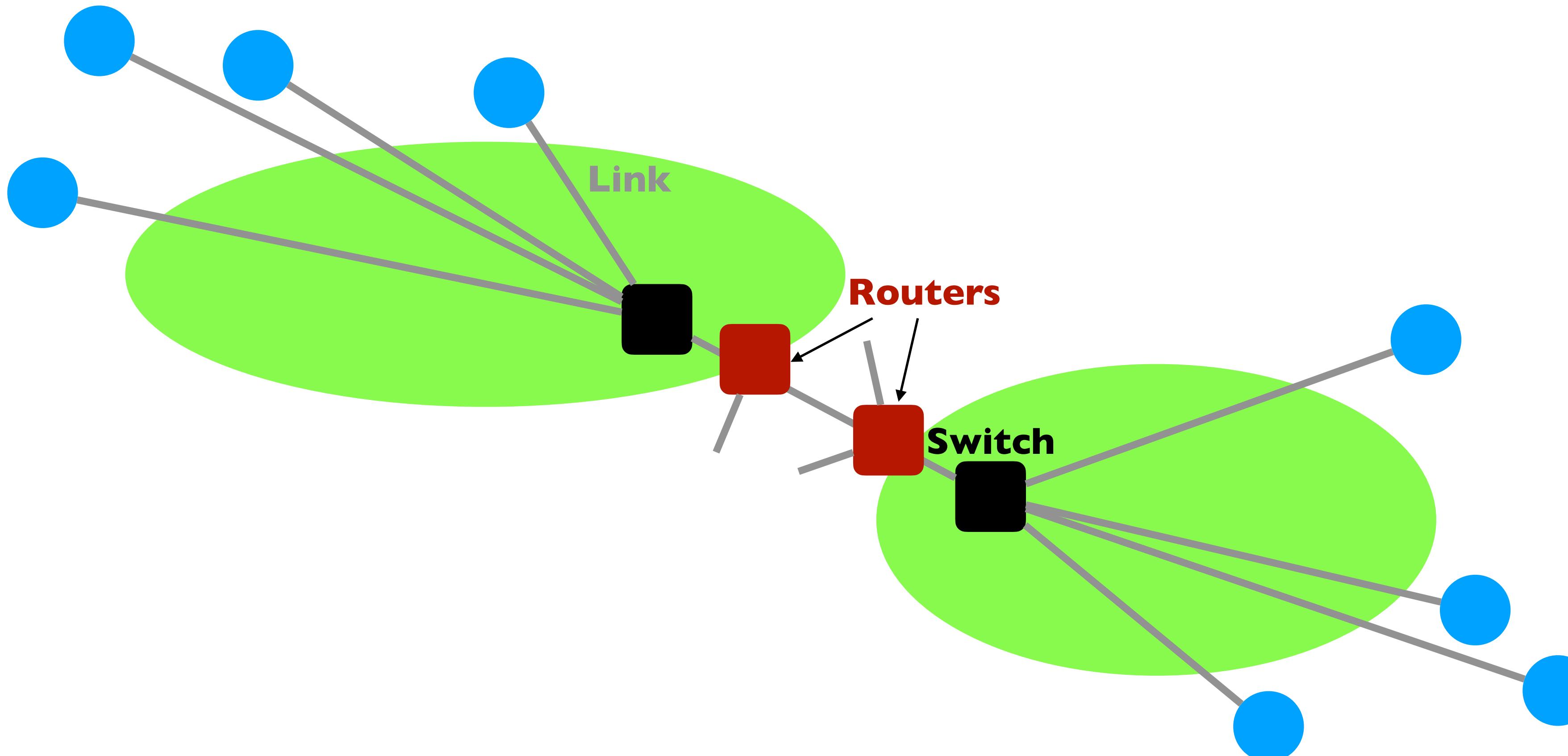
A closer look: end-system

- Application
 - *Web server, browser, mail, game*
- Transport and network layer
 - *Typically part of the operating system*
- Datalink and physical layer
 - *Hardware/firmware/drivers*

A closer look: Network



A closer look: Network



Switches vs. Routers

Switches vs. Routers

- Switches do what routers do, but don't participate in global delivery, just local delivery
 - *Switches only need to support L1, L2*
 - *Routers support L1-L3*

Switches vs. Routers

- Switches do what routers do, but don't participate in global delivery, just local delivery
 - *Switches only need to support L1, L2*
 - *Routers support L1-L3*
- Won't focus on the router/switch distinction
 - *When I say switch, I almost always mean router*
 - *Almost all boxes support network layer these days*

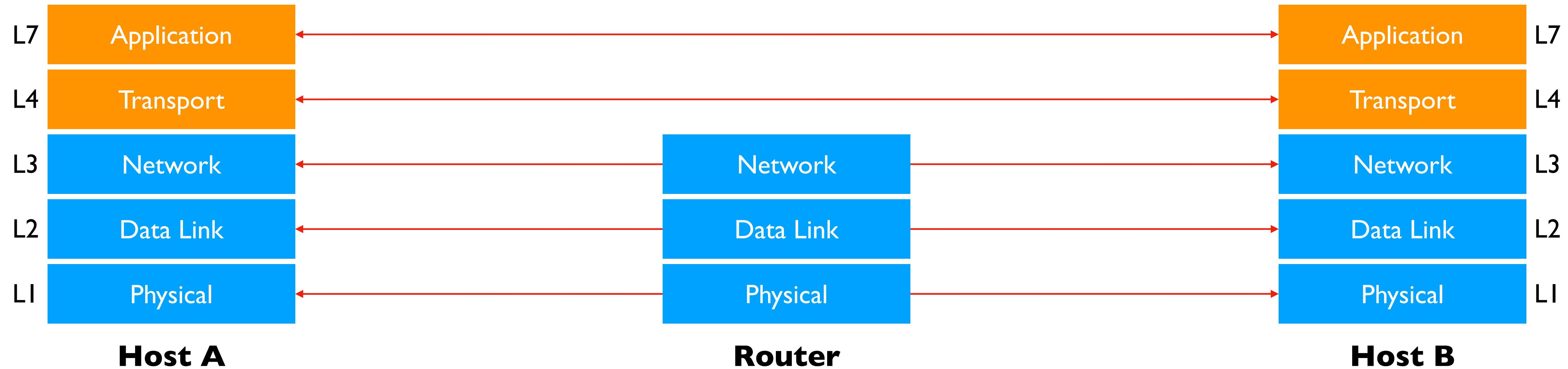
What layers get implemented in the network?

- Bits arrive on wire → physical layer (L1)
- Packets must be delivered across link and local networks → datalink layer (L2)
- Packets must be delivered b/w networks for global delivery → network layer (L3)

What layers get implemented in the network?

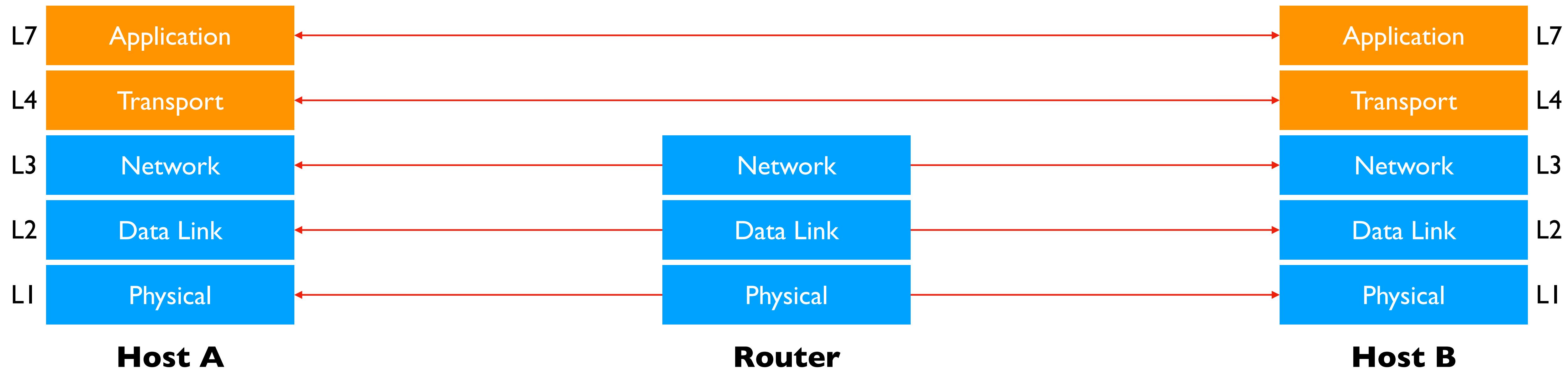
- Bits arrive on wire → physical layer (L1)
- Packets must be delivered across link and local networks → datalink layer (L2)
- Packets must be delivered b/w networks for global delivery → network layer (L3)
- Hence:
 - *Switches: implement physical and data link layers (L1, L2)*
 - *Routers: implement physical, datalink, network layers (L1, L2, L3)*

Logical Communication

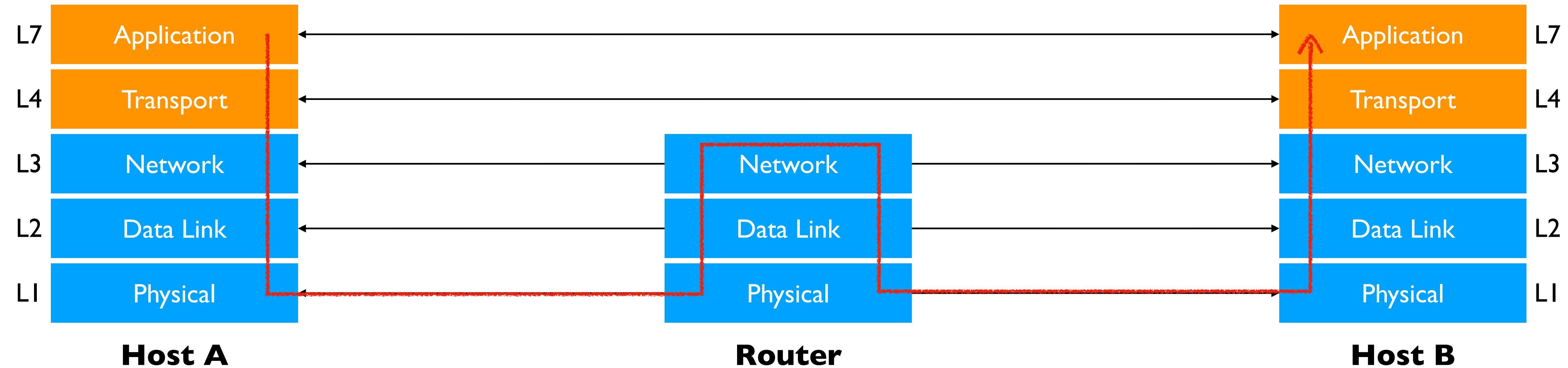


Logical Communication

- Layers interact with peer's corresponding layer

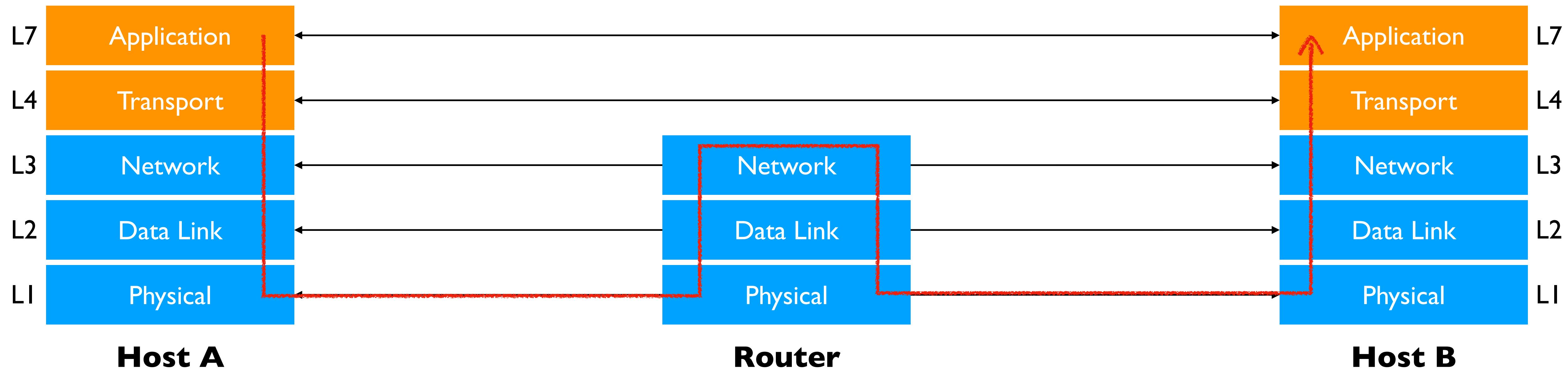


Physical Communication



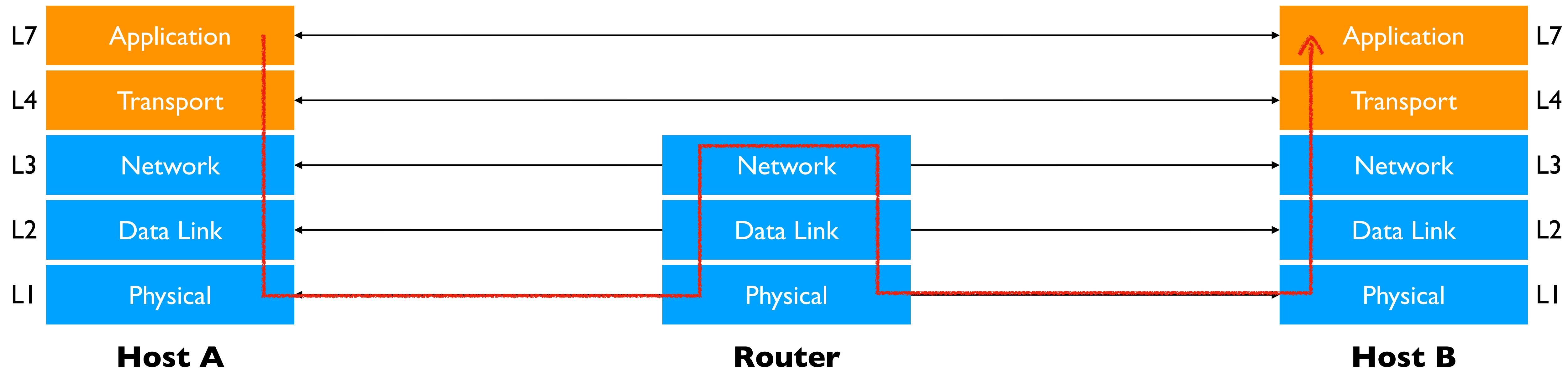
Physical Communication

- Communication goes down to the physical network

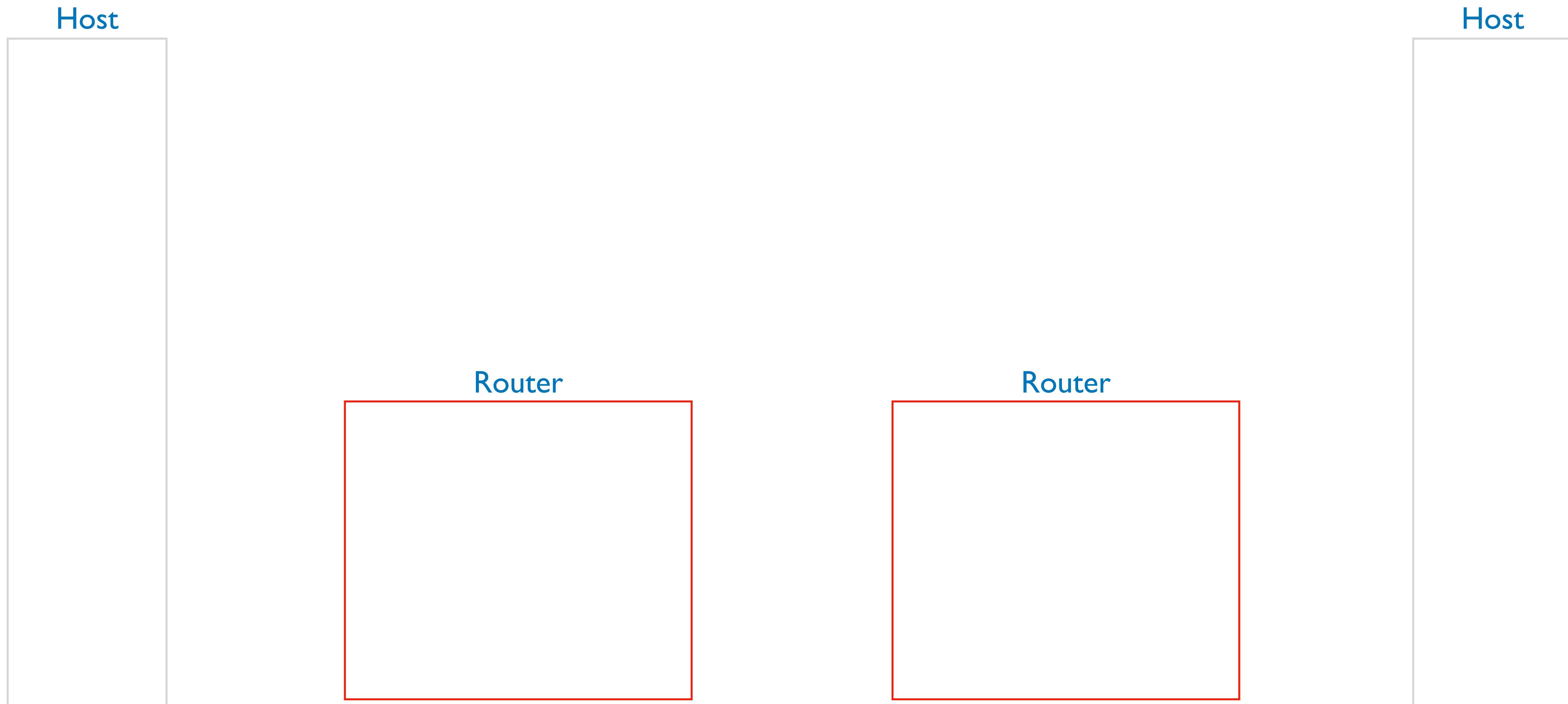


Physical Communication

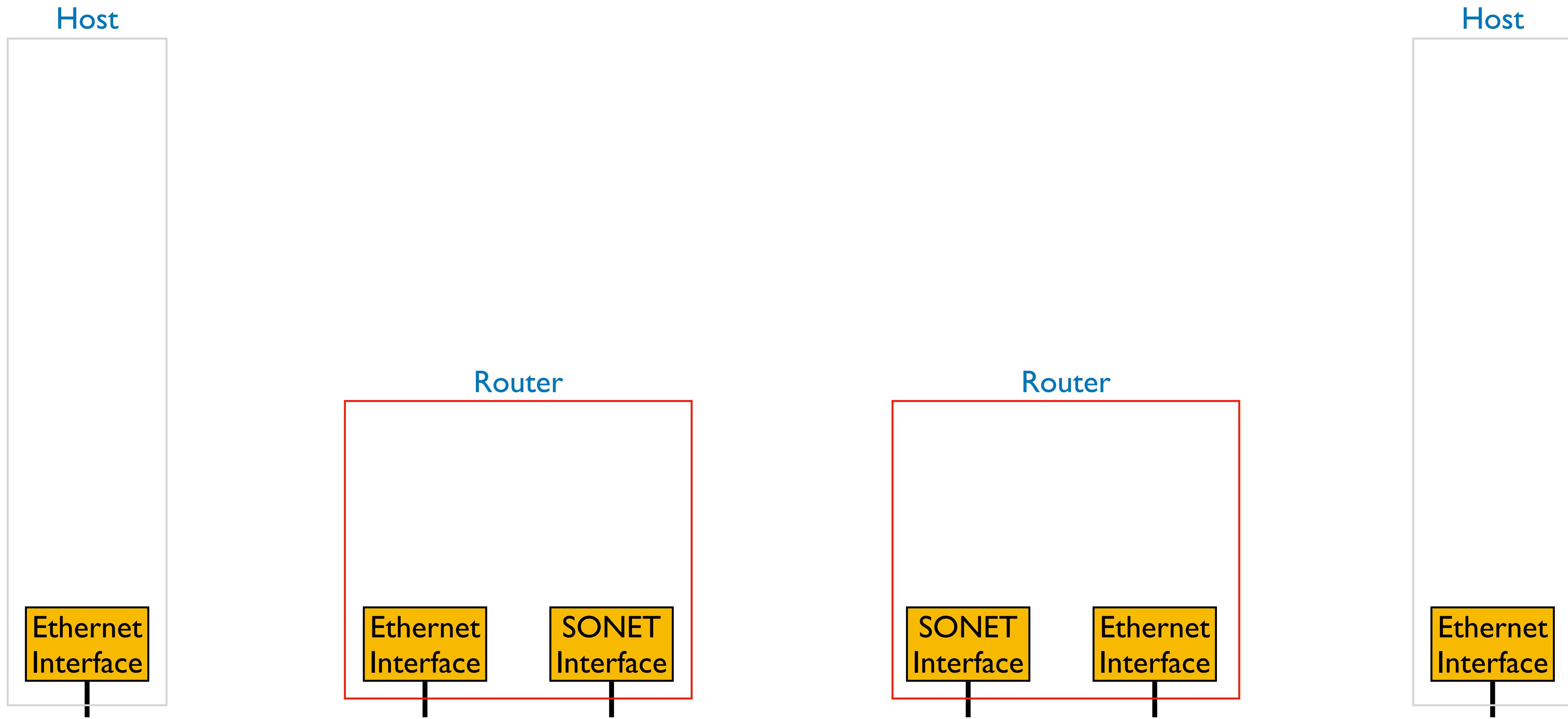
- Communication goes down to the physical network
- Then up to relevant layer



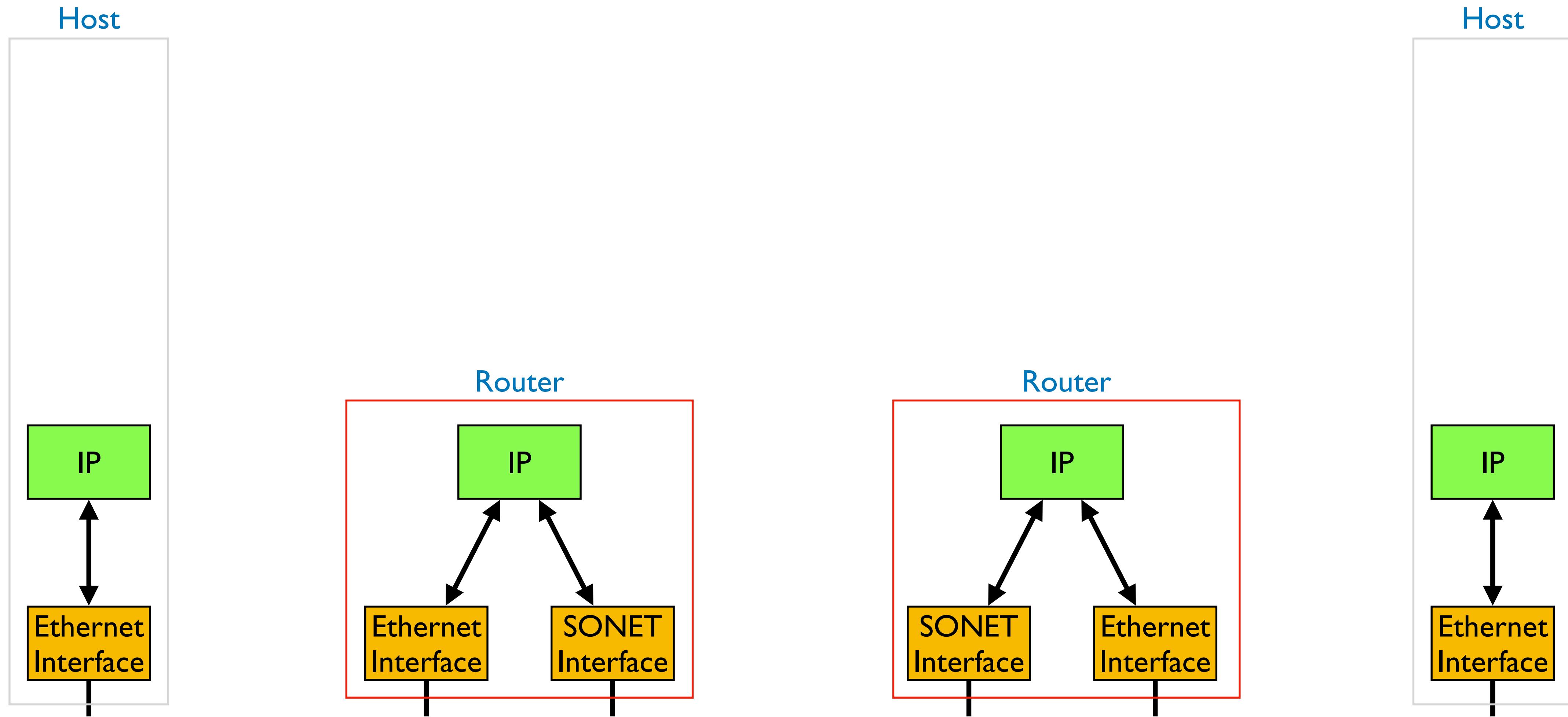
A Protocol-centric Diagram



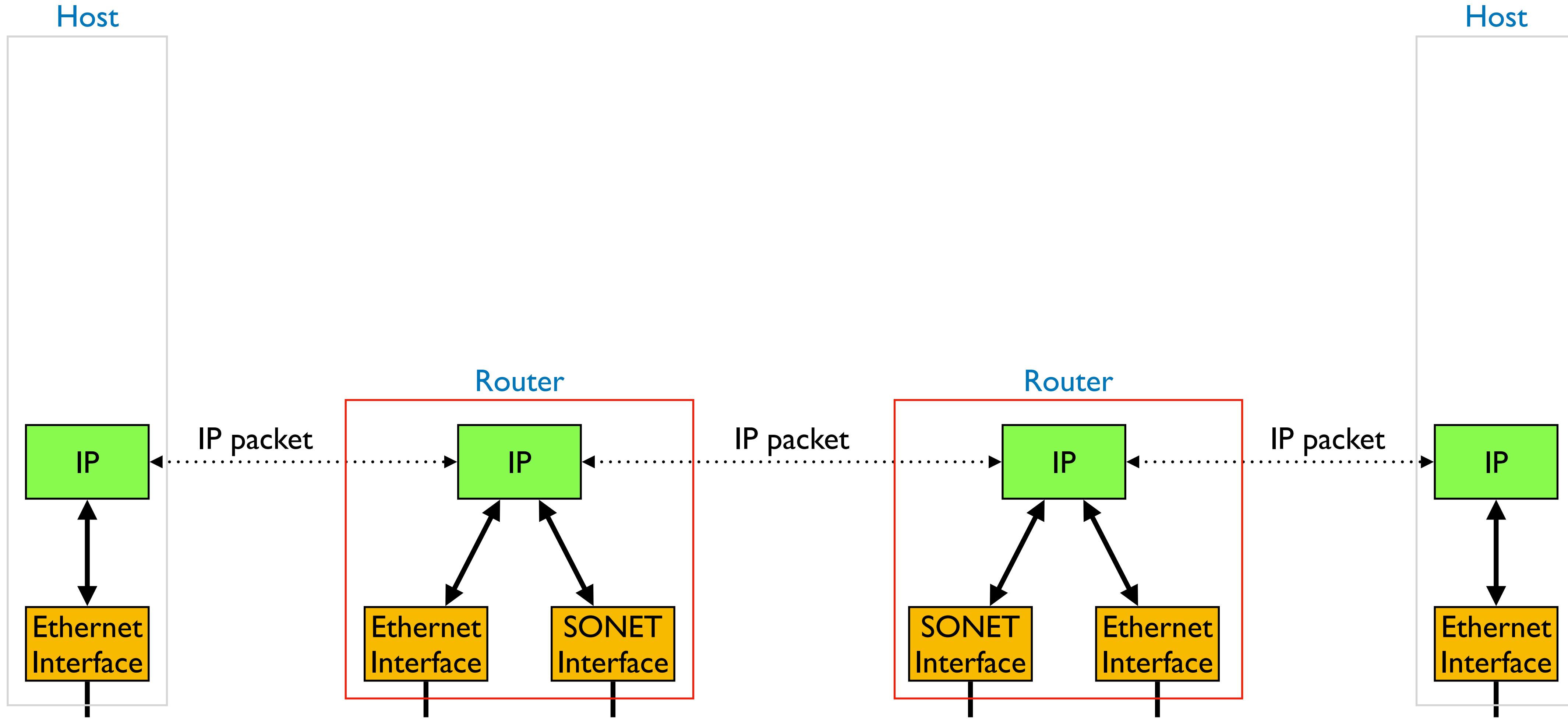
A Protocol-centric Diagram



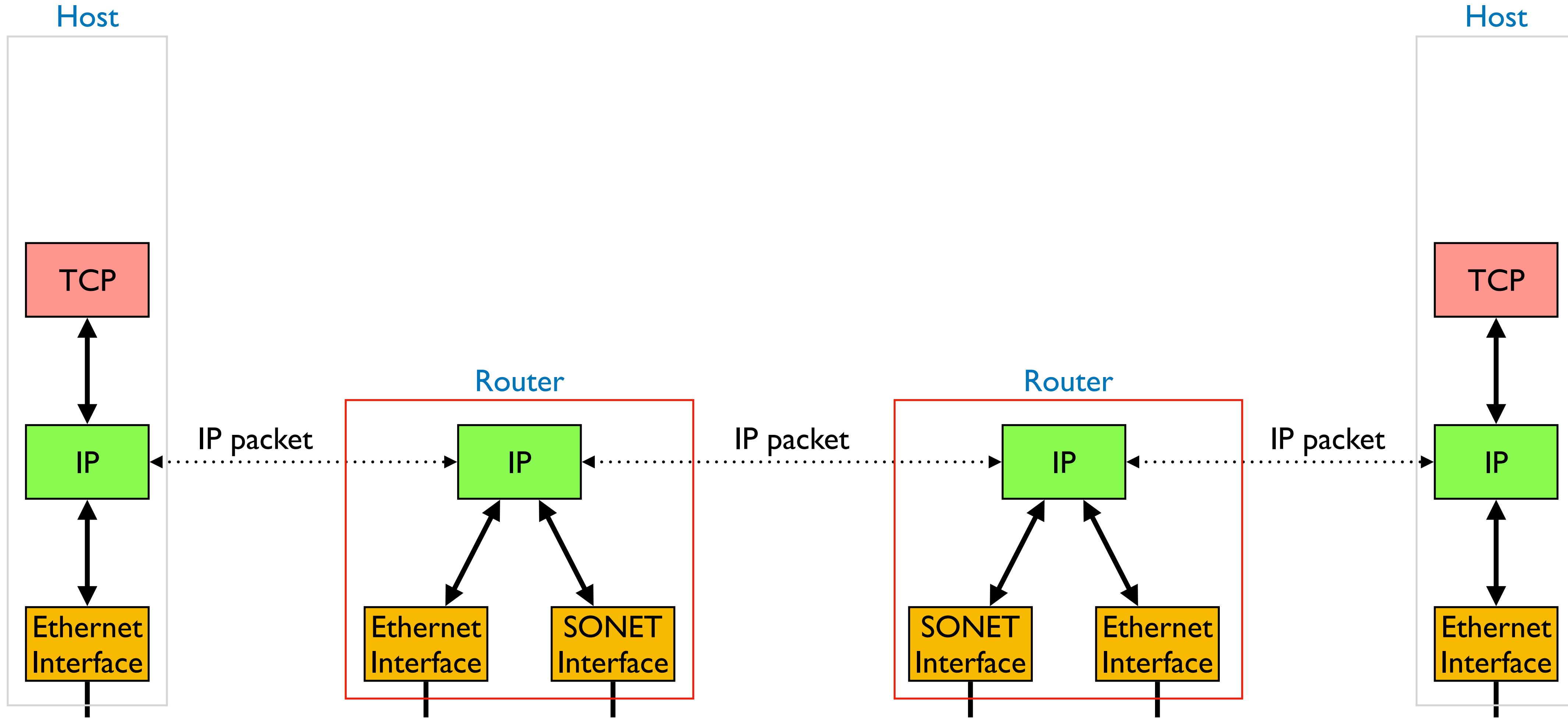
A Protocol-centric Diagram



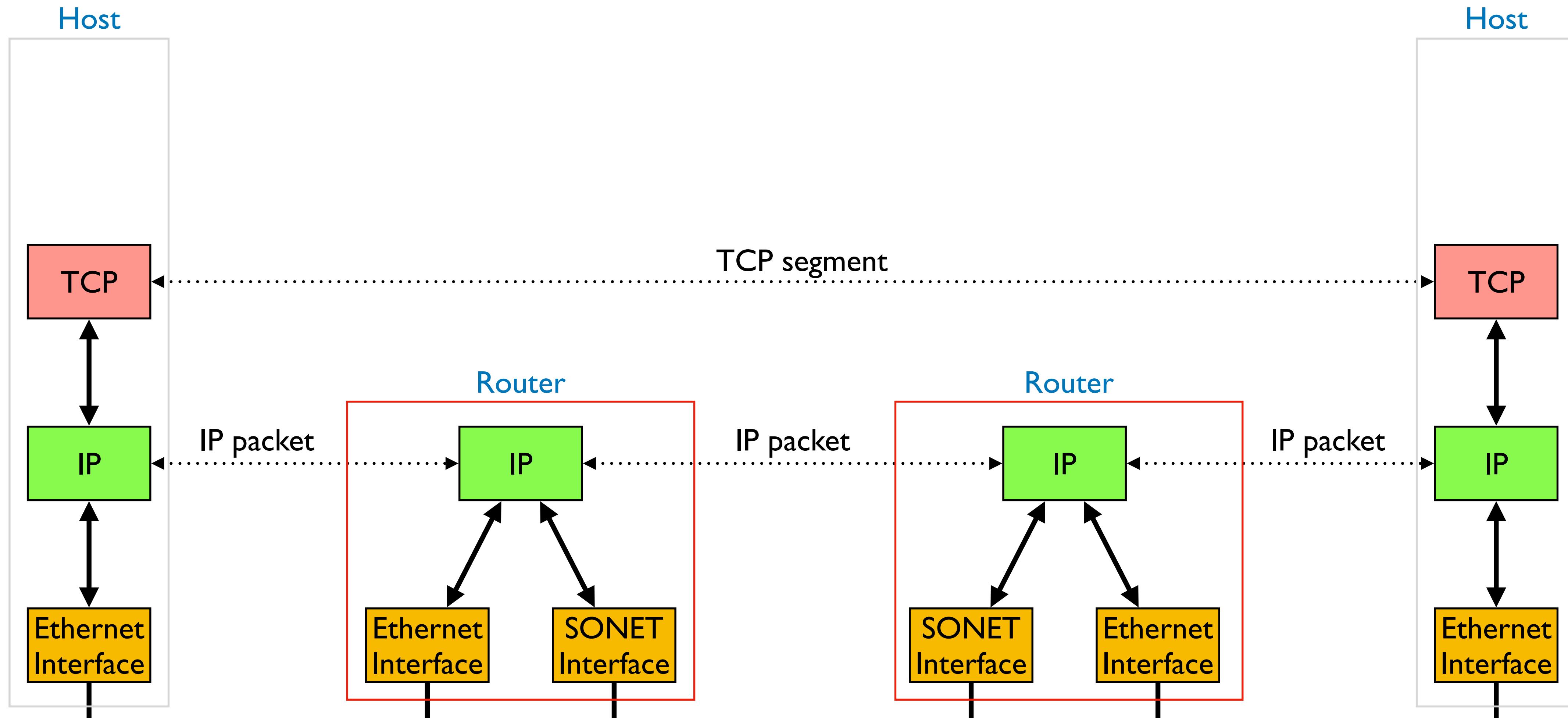
A Protocol-centric Diagram



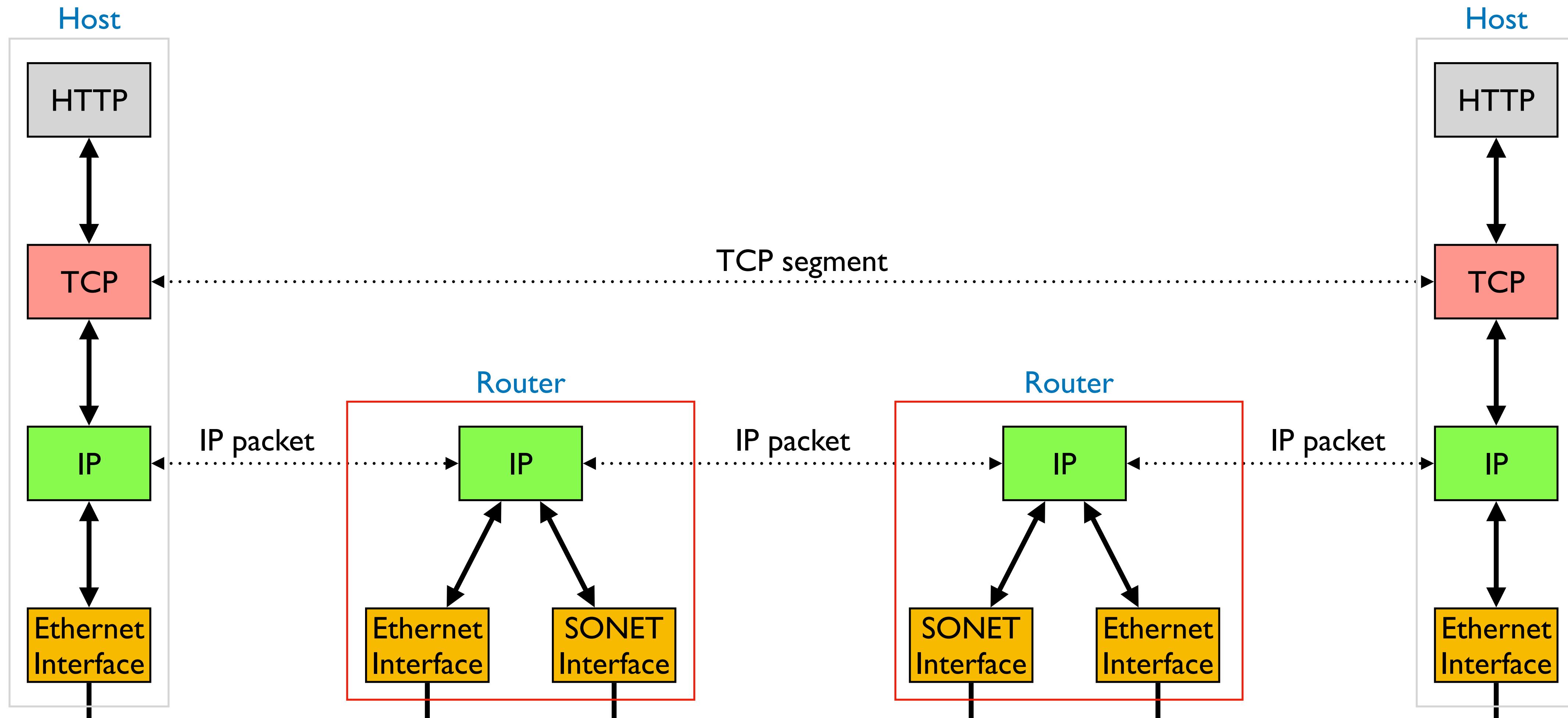
A Protocol-centric Diagram



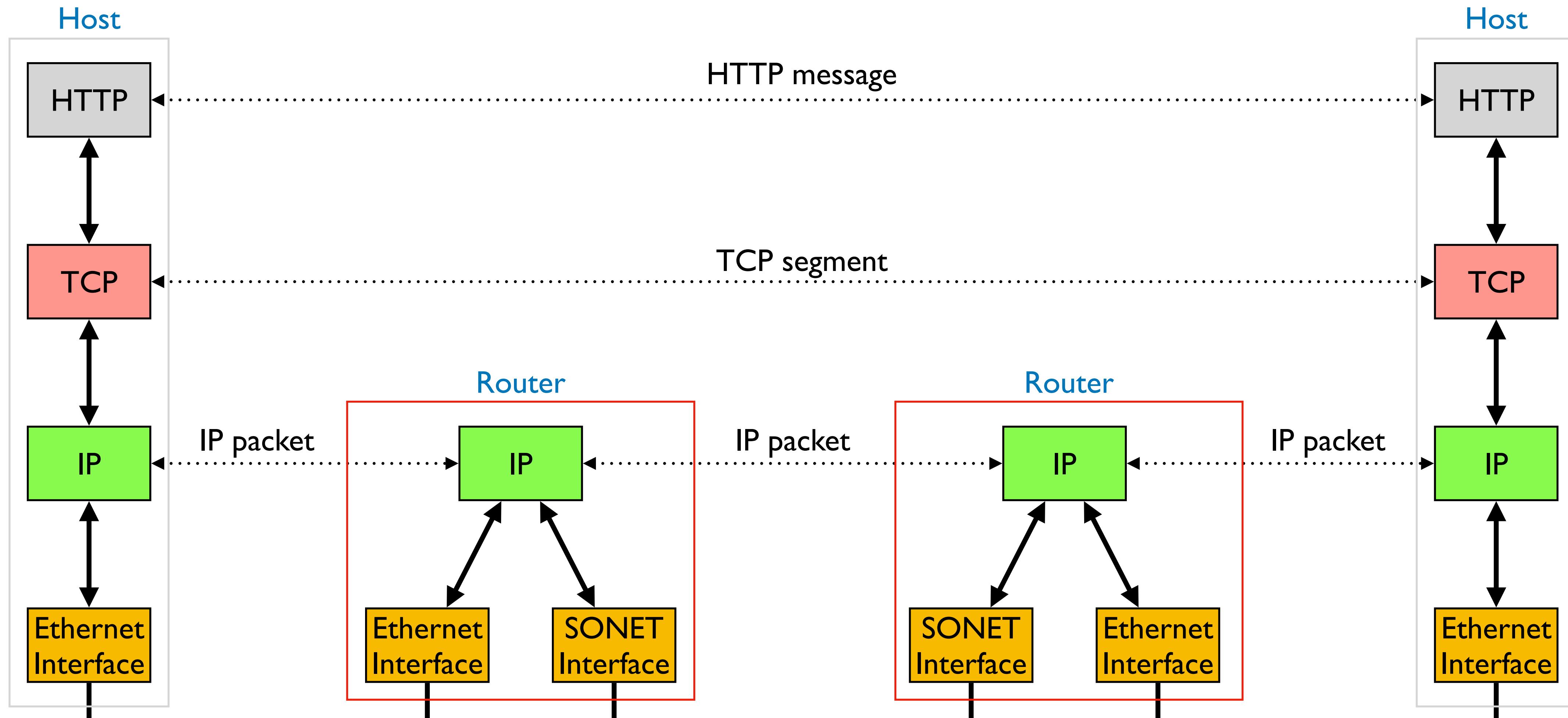
A Protocol-centric Diagram



A Protocol-centric Diagram



A Protocol-centric Diagram



Questions?

Three steps

- **Decomposition**
- **Organization**
- **Assignment**

Three steps

- **Decomposition**
- **Organization**

- **Assignment**

E2E principle

Placing Network Functionality

Placing Network Functionality

- Hugely influential paper: **“End-to-end Arguments in System Design” by Saltzer, Reed, and Clark ('84)**

Placing Network Functionality

- Hugely influential paper: **“End-to-end Arguments in System Design” by Saltzer, Reed, and Clark (’84)**
 - Articulated the “End-to-end Principle” (*E2E*)

Placing Network Functionality

- Hugely influential paper: **“End-to-end Arguments in System Design” by Saltzer, Reed, and Clark (’84)**
 - Articulated the “End-to-end Principle” (*E2E*)
 - Endless debate over what it means

Placing Network Functionality

- Hugely influential paper: **“End-to-end Arguments in System Design” by Saltzer, Reed, and Clark (’84)**
 - Articulated the “End-to-end Principle” (*E2E*)
 - Endless debate over what it means
 - Everyone cites it as supporting their position

The Statement

The Statement

- **If** a function **completely and correctly be implemented only with the knowledge and help of** the applications standing at the **end points** of the communication system,

The Statement

- **If** a function **completely and correctly be implemented only with the knowledge and help of** the applications standing at the **end points** of the communication system,
- **Then**, providing that function as a **feature of the communication system** itself is not possible.

The Statement

- **If** a function **completely and correctly be implemented only with the knowledge and help of** the applications standing at the **end points** of the communication system,
- **Then**, providing that function as a **feature of the communication system** itself is not possible.
- **Sometimes** an **incomplete version of the function** provided by the communication system **may be useful as a performance enhancement**

Deconstructing the Statement

Deconstructing the Statement

- Some application requirements can only be correctly implemented **at the end-system**

Deconstructing the Statement

- Some application requirements can only be correctly implemented **at the end-system**
 - *Reliability, security, etc.*

Deconstructing the Statement

- Some application requirements can only be correctly implemented **at the end-system**
 - *Reliability, security, etc.*
- Implementing these completely in the network may not only be hard, but impossible

Deconstructing the Statement

- Some application requirements can only be correctly implemented **at the end-system**
 - *Reliability, security, etc.*
- Implementing these completely in the network may not only be hard, but impossible
- End-systems:

Deconstructing the Statement

- Some application requirements can only be correctly implemented **at the end-system**
 - *Reliability, security, etc.*
 - Implementing these completely in the network may not only be hard, but impossible
 - End-systems:
 - **Can** satisfy the requirement without network's help

Deconstructing the Statement

- Some application requirements can only be correctly implemented **at the end-system**
 - *Reliability, security, etc.*
 - Implementing these completely in the network may not only be hard, but impossible
 - End-systems:
 - **Can** satisfy the requirement without network's help
 - **Will/must** do so, since they cannot rely on the network

Example: Reliable File Transfer



Example: Reliable File Transfer



- **Solution I:** Make each step reliable, and string them together to make the end-to-end process reliable

Example: Reliable File Transfer



- **Solution I:** Make each step reliable, and string them together to make the end-to-end process reliable

Example: Reliable File Transfer



- **Solution I:** Make each step reliable, and string them together to make the end-to-end process reliable

Example: Reliable File Transfer



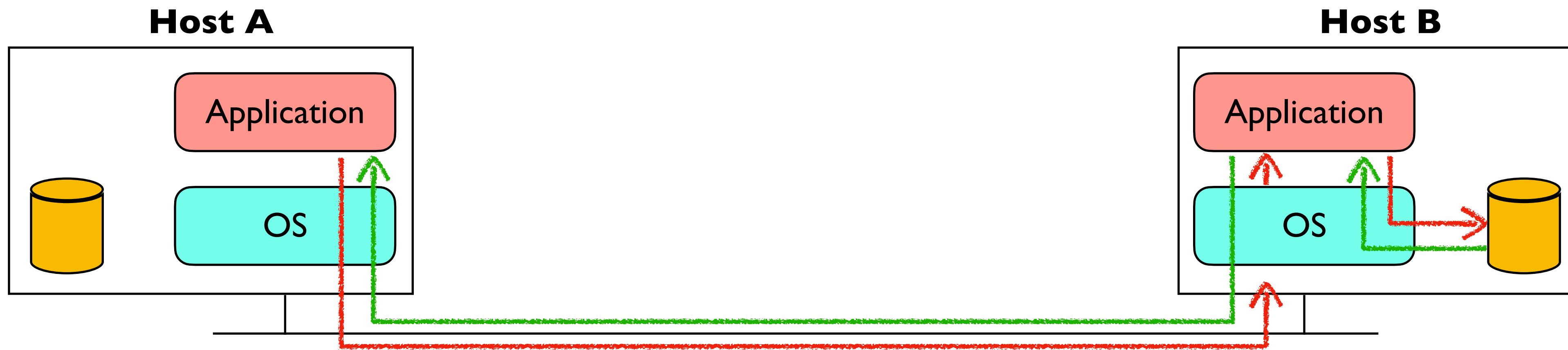
- **Solution I:** Make each step reliable, and string them together to make the end-to-end process reliable

Example: Reliable File Transfer



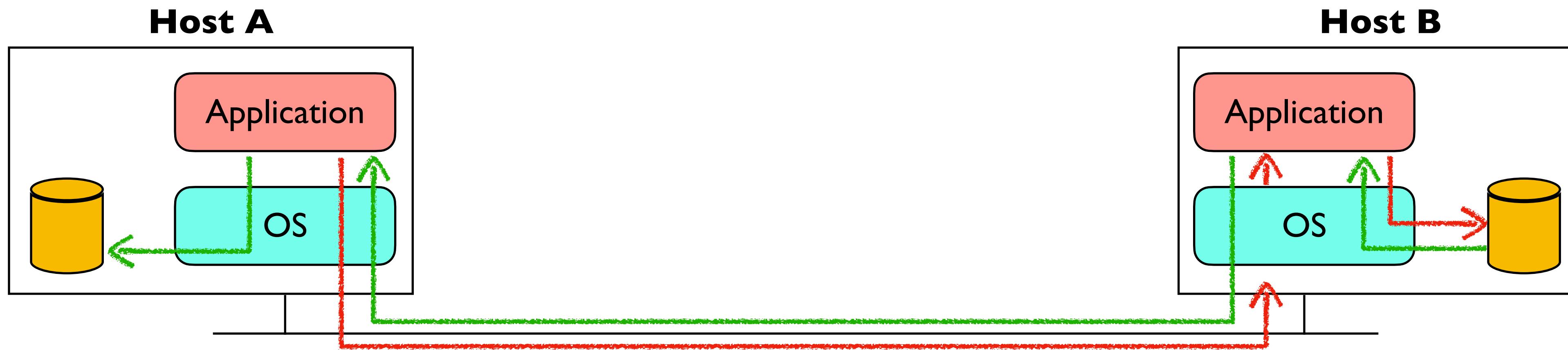
- **Solution I:** Make each step reliable, and string them together to make the end-to-end process reliable

Example: Reliable File Transfer



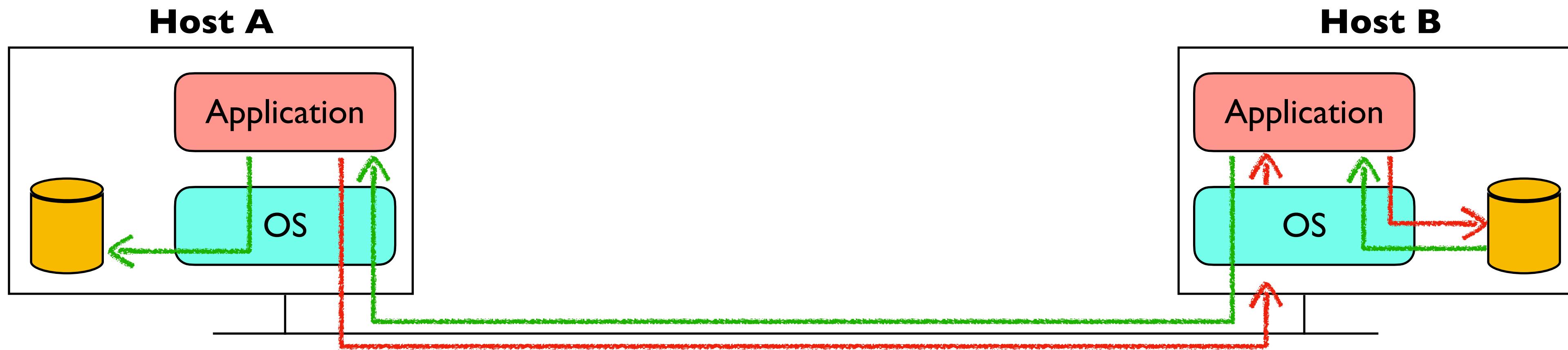
- **Solution I:** Make each step reliable, and string them together to make the end-to-end process reliable

Example: Reliable File Transfer



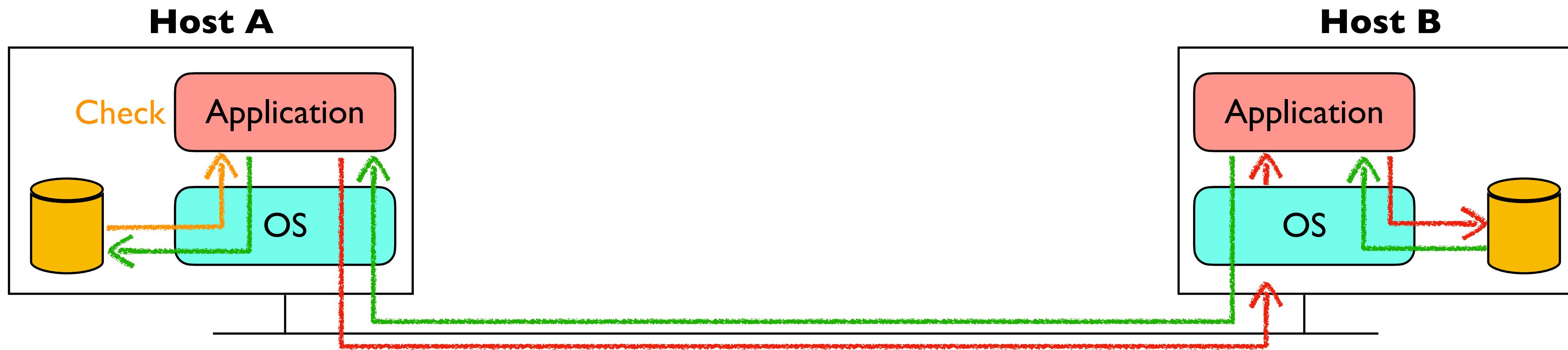
- **Solution I:** Make each step reliable, and string them together to make the end-to-end process reliable

Example: Reliable File Transfer



- **Solution 1:** Make each step reliable, and string them together to make the end-to-end process reliable
- **Solution 2:** End-to-end check and retry

Example: Reliable File Transfer



- **Solution 1:** Make each step reliable, and string them together to make the end-to-end process reliable
- **Solution 2:** End-to-end check and retry

Discussion

Discussion

- **Solution I** is incomplete

Discussion

- **Solution I** is incomplete
 - *What happens if any network element misbehaves?*

Discussion

- **Solution I** is incomplete
 - *What happens if any network element misbehaves?*
 - *Receiver has to do the check anyway!*

Discussion

- **Solution 1** is incomplete
 - *What happens if any network element misbehaves?*
 - *Receiver has to do the check anyway!*
- **Solution 2** is complete

Discussion

- **Solution 1** is incomplete
 - *What happens if any network element misbehaves?*
 - *Receiver has to do the check anyway!*
- **Solution 2** is complete
 - *Full functionality can be entirely implemented at application layer with no need for reliability from lower layers*

Discussion

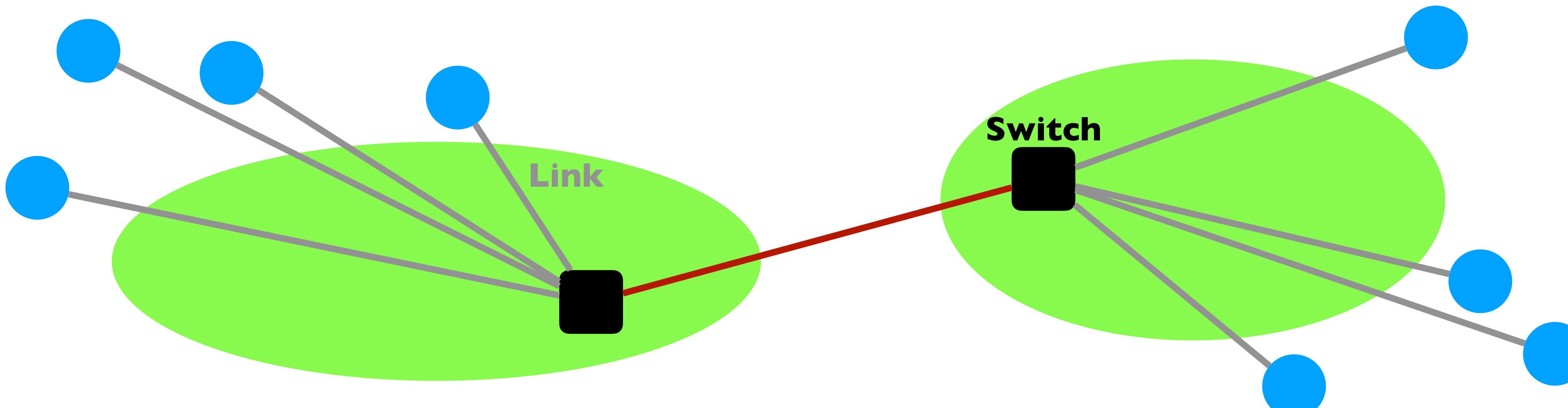
- **Solution 1** is incomplete
 - *What happens if any network element misbehaves?*
 - *Receiver has to do the check anyway!*
- **Solution 2** is complete
 - *Full functionality can be entirely implemented at application layer with no need for reliability from lower layers*
 - Is there any **need** to implement reliability at lower layers?

Except...

Sometimes an **incomplete version of the function** provided by the communication system may be useful as a **performance enhancement**

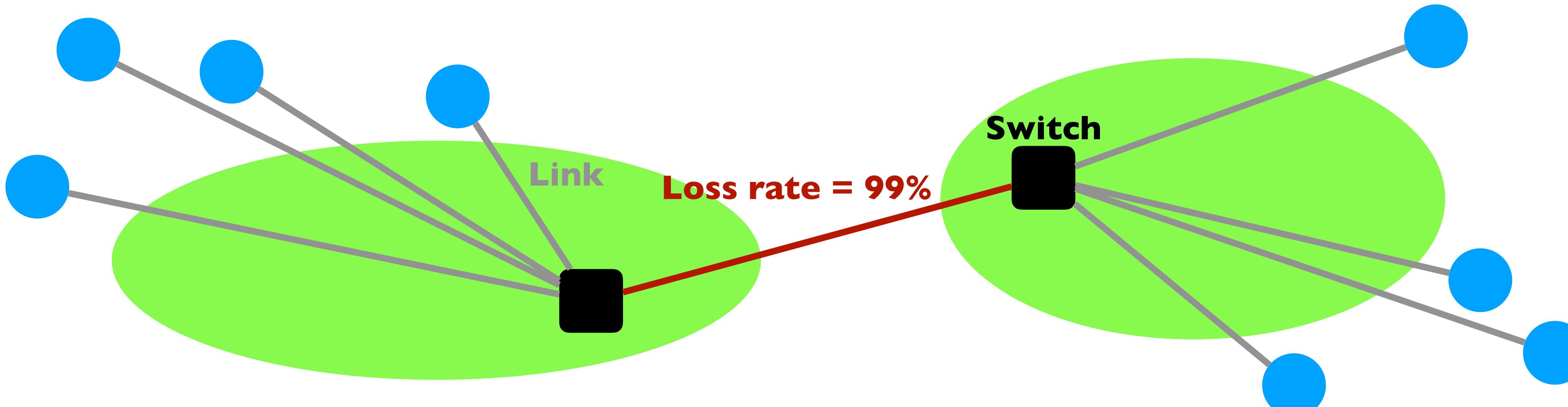
Except...

Sometimes an **incomplete version of the function** provided by the communication system may be useful as a **performance enhancement**



Except...

Sometimes an **incomplete version of the function** provided by the communication system may be useful as a **performance enhancement**



Summary of the E2E principle

Summary of the E2E principle

- Implementing functionality (e.g., reliability) in the network

Summary of the E2E principle

- Implementing functionality (e.g., reliability) in the network
 - Doesn't reduce host *implementation complexity*

Summary of the E2E principle

- Implementing functionality (e.g., reliability) in the network
 - Doesn't reduce host implementation complexity
 - Does increase network complexity

Summary of the E2E principle

- Implementing functionality (e.g., reliability) in the network
 - Doesn't reduce host implementation complexity
 - Does increase network complexity
 - Probably increases delay and overhead on all applications even if they don't need the functionality (e.g., VoIP)

Summary of the E2E principle

- Implementing functionality (e.g., reliability) in the network
 - Doesn't reduce host implementation complexity
 - Does increase network complexity
 - Probably increases delay and overhead on all applications even if they don't need the functionality (e.g., VoIP)
- However, implementing in the network can improve performance in some cases

Summary of the E2E principle

- Implementing functionality (e.g., reliability) in the network
 - Doesn't reduce host implementation complexity
 - Does increase network complexity
 - Probably increases delay and overhead on all applications even if they don't need the functionality (e.g., VoIP)
- However, implementing in the network can improve performance in some cases
 - E.g., consider a very lossy link

Commonly used examples

Commonly used examples

- Error handling in file transfer

Commonly used examples

- Error handling in file transfer
- End-to-end versus in-network encryption

Commonly used examples

- Error handling in file transfer
- End-to-end versus in-network encryption
- The partition of work between TCP and IP for reliable packet delivery

Commonly used examples

- Error handling in file transfer
- End-to-end versus in-network encryption
- The partition of work between TCP and IP for reliable packet delivery
- What about Quality-of-Service (QoS)?

Commonly used examples

- Error handling in file transfer
- End-to-end versus in-network encryption
- The partition of work between TCP and IP for reliable packet delivery
- What about Quality-of-Service (QoS)?
 - *Communication throughput or delay guarantee*

Some consequences

Some consequences

- In layered design, the E2E principle provides guidance on which layers are implemented where

Some consequences

- In layered design, the E2E principle provides guidance on which layers are implemented where
- “Dumb” network and “smart” end-systems

Some consequences

- In layered design, the E2E principle provides guidance on which layers are implemented where
- “Dumb” network and “smart” end-systems
 - *Often credited as a key to the Internet’s success!*

Some consequences

- In layered design, the E2E principle provides guidance on which layers are implemented where
- “Dumb” network and “smart” end-systems
 - *Often credited as a key to the Internet’s success!*
- “Fate-sharing”

Some consequences

- In layered design, the E2E principle provides guidance on which layers are implemented where
- “Dumb” network and “smart” end-systems
 - *Often credited as a key to the Internet’s success!*
- “Fate-sharing”
 - *Store state at the system entities that rely on the state*

Some consequences

- In layered design, the E2E principle provides guidance on which layers are implemented where
- “Dumb” network and “smart” end-systems
 - *Often credited as a key to the Internet’s success!*
- “Fate-sharing”
 - *Store state at the system entities that rely on the state*
 - *Often translated to keep end-host state out of routers*

Cracks in the E2E argument?

Cracks in the E2E argument?

- Ignores incentives of different stakeholders

Cracks in the E2E argument?

- Ignores incentives of different stakeholders
 - e.g., ISP looking for new revenue-generating services

Cracks in the E2E argument?

- Ignores incentives of different stakeholders
 - e.g., ISP looking for new revenue-generating services
 - e.g., users looking for a performance boost

Cracks in the E2E argument?

- Ignores incentives of different stakeholders
 - e.g., ISP looking for new revenue-generating services
 - e.g., users looking for a performance boost
- Sometimes we don't trust end-systems to do the job

Cracks in the E2E argument?

- Ignores incentives of different stakeholders
 - e.g., ISP looking for new revenue-generating services
 - e.g., users looking for a performance boost
- Sometimes we don't trust end-systems to do the job
 - e.g., firewalls, intrusion detection systems

Questions?

Asking the right architectural questions...

Asking the right architectural questions...

- How to break system into modules?

Asking the right architectural questions...

- How to break system into modules?
 - *Classic decomposition into tasks*

Asking the right architectural questions...

- How to break system into modules?
 - *Classic decomposition into tasks*
 - Where are modules implemented?

Asking the right architectural questions...

- How to break system into modules?
 - *Classic decomposition into tasks*
- Where are modules implemented?
 - *Hosts? Routers? Both?*

Asking the right architectural questions...

- How to break system into modules?
 - *Classic decomposition into tasks*
- Where are modules implemented?
 - Hosts? Routers? Both?
- Where is state stored?

Asking the right architectural questions...

- How to break system into modules?
 - *Classic decomposition into tasks*
- Where are modules implemented?
 - *Hosts? Routers? Both?*
- Where is state stored?
 - *Hosts? Routers? Both?*

... leads to three design principles

- How to break system into modules?
 - *Layering*
- Where are modules implemented?
 - *End-to-end principle*
- Where is state stored?
 - *Fate-sharing*

Taking Stock

Taking Stock

- Layering is a good way to organize networks
 - *Unified IP layer decouples applications from networks*

Taking Stock

- Layering is a good way to organize networks
 - *Unified IP layer decouples applications from networks*
- E2E argument encourages us to keep IP simple

Taking Stock

- Layering is a good way to organize networks
 - *Unified IP layer decouples applications from networks*
- E2E argument encourages us to keep IP simple
- Fate-sharing principle keeps state out of routers