CS168
How the Internet Works (contd.)

Rishabh Iyer

Spring 2026

Slide credits: Sylvia Ratnasamy, Rob Shakir, Peyrin Kao

Last week:
Building the Internet Bottom-Up

- Packets
  - Consisting of payloads and headers
  - Payloads are meaningful only to the endpoints
  - Headers are meaningful to the network *and* endpoint

- Links
  - Characterized by bandwidth, propagation delay and bandwidth-delay product
  - Packet transfer involves transmission delay and propagation delay
  - Circuit switching vs packet switching

Last week: Sharing Network Resources

- Statistical multiplexing, which can be implemented as:
  - Reservations (e.g., circuit switching)
  - Best-effort (packet switching)

- Detailed analysis of the trade-offs involved
  - As an abstraction to applications
  - Network efficiency
  - Handling failures (at scale)
  - Complexity of implementation (at scale)

Tradeoffs: Circuit *vs.* Packet Switching

Tradeoffs: Circuit *vs.* Packet Switching

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

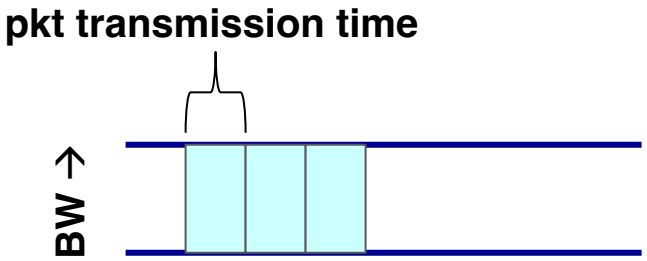
Tradeoffs: Circuit *vs.* Packet Switching

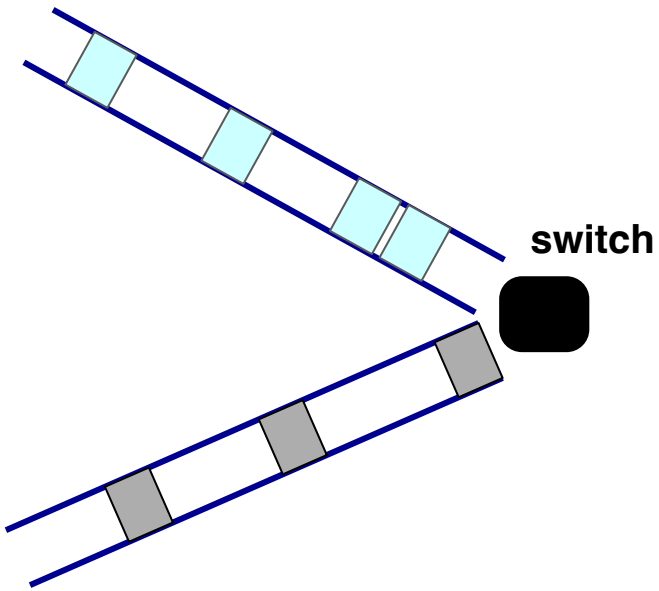
- **Pros for circuit switching**:
  - Better application performance (reserved bandwidth)
  - More predictable and understandable (w/o failures)

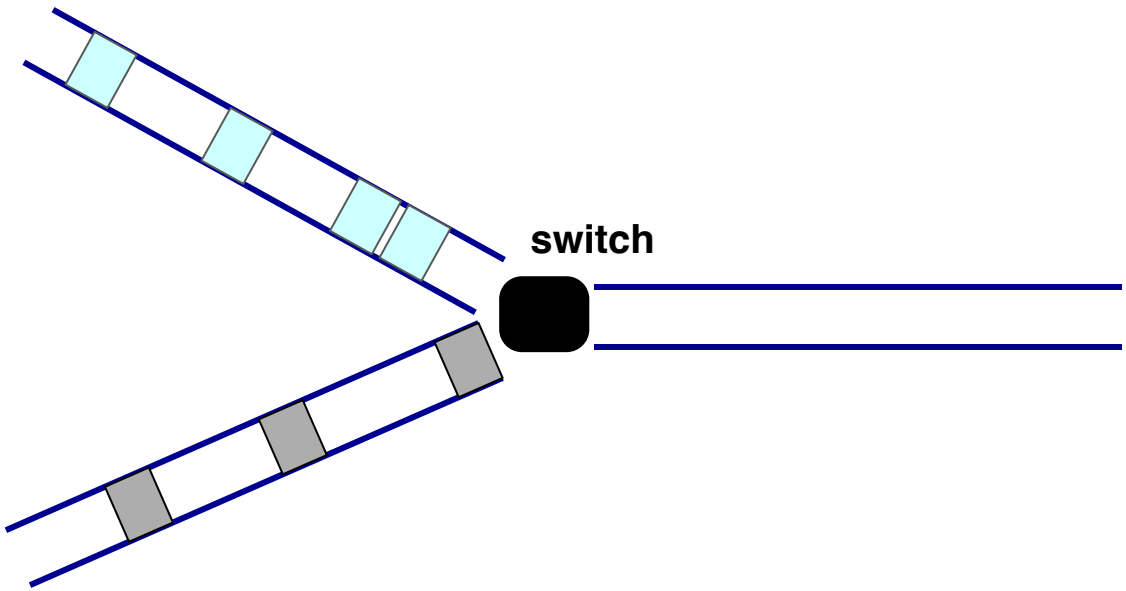
- **Pros for packet switching**:
  - Better efficiency
  - Faster startup to first packet delivered
  - Easier recovery from failure
  - Simpler implementation (avoids dynamic per-flow state management in switches)

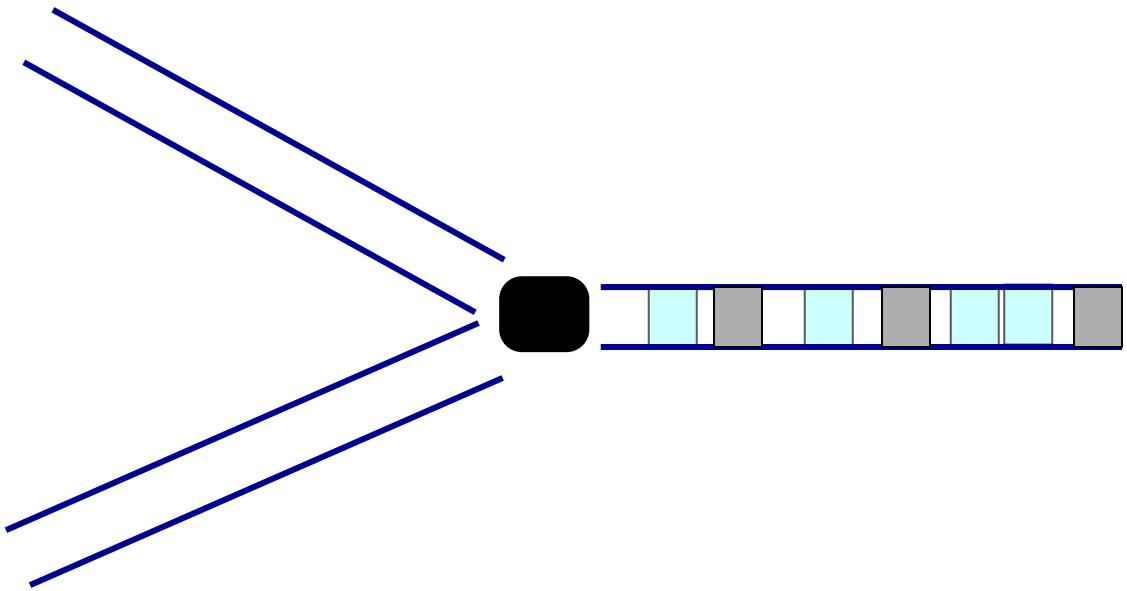
Let’s take a closer look at packet switching ….

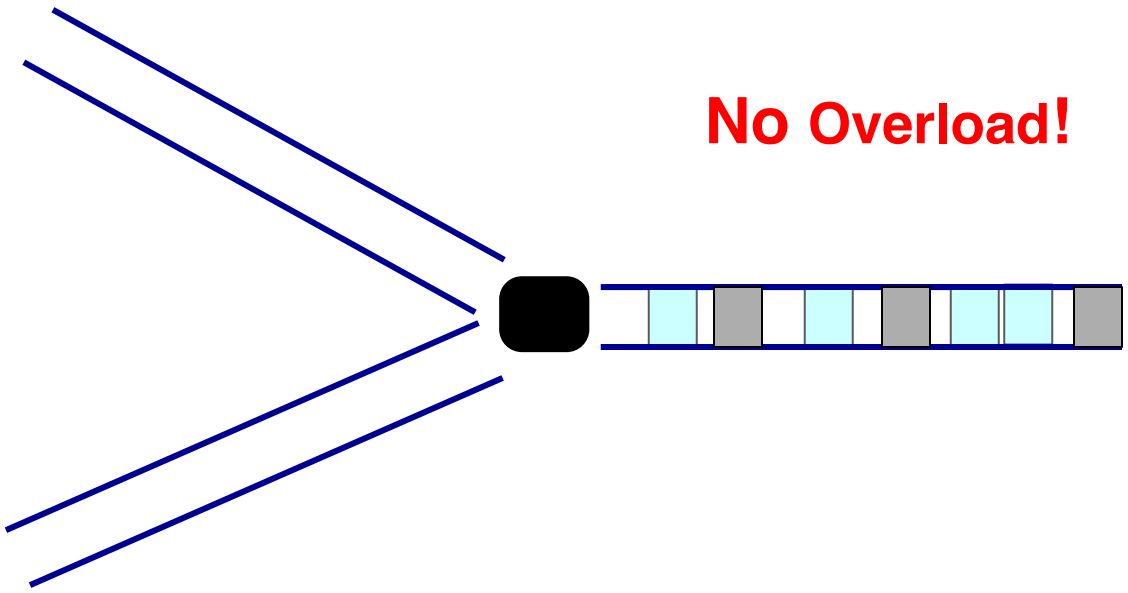
Recall, packets in flight: “pipe” view

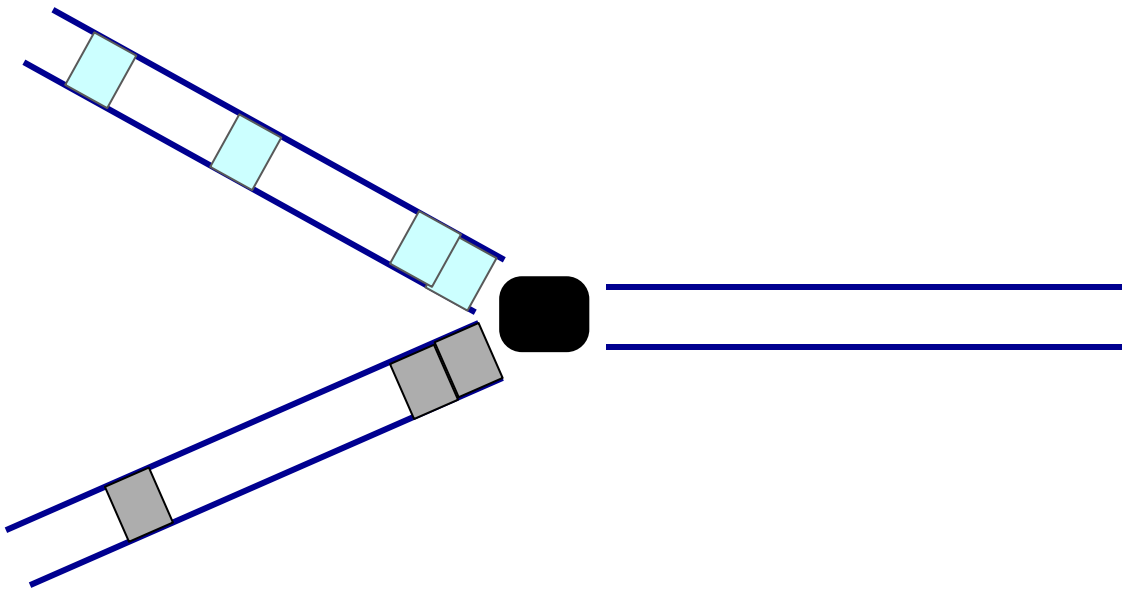


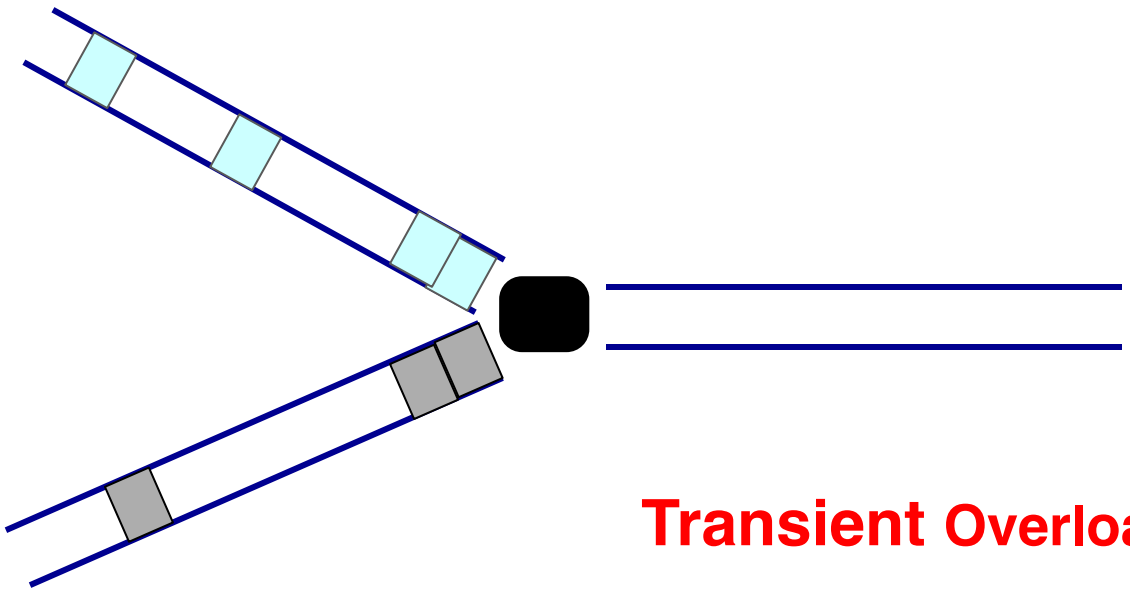

switch


switch

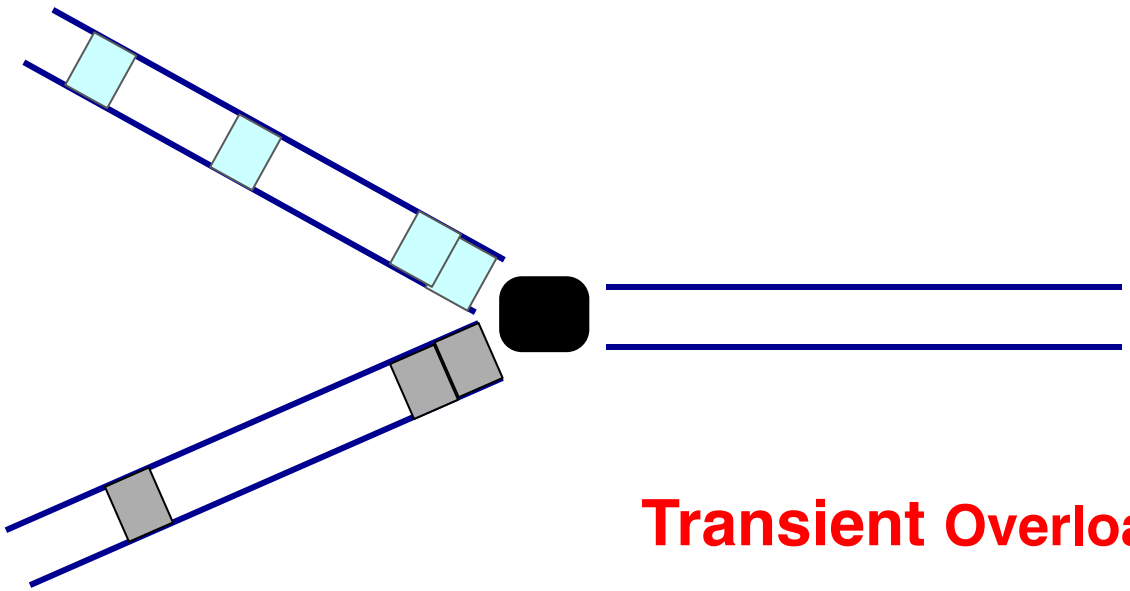



No Overload!



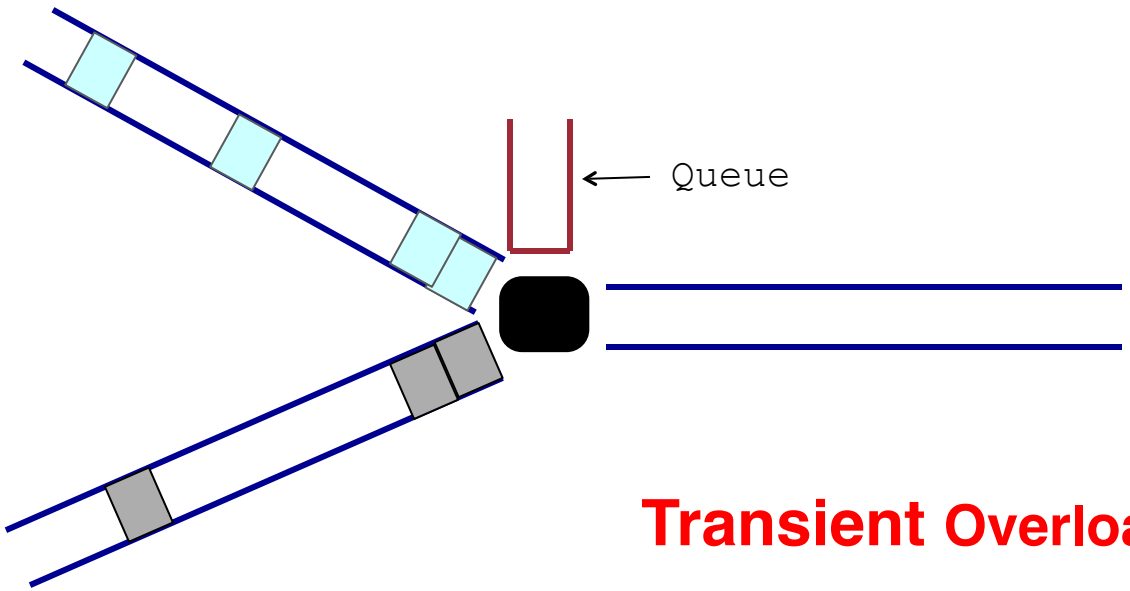


Transient Overload



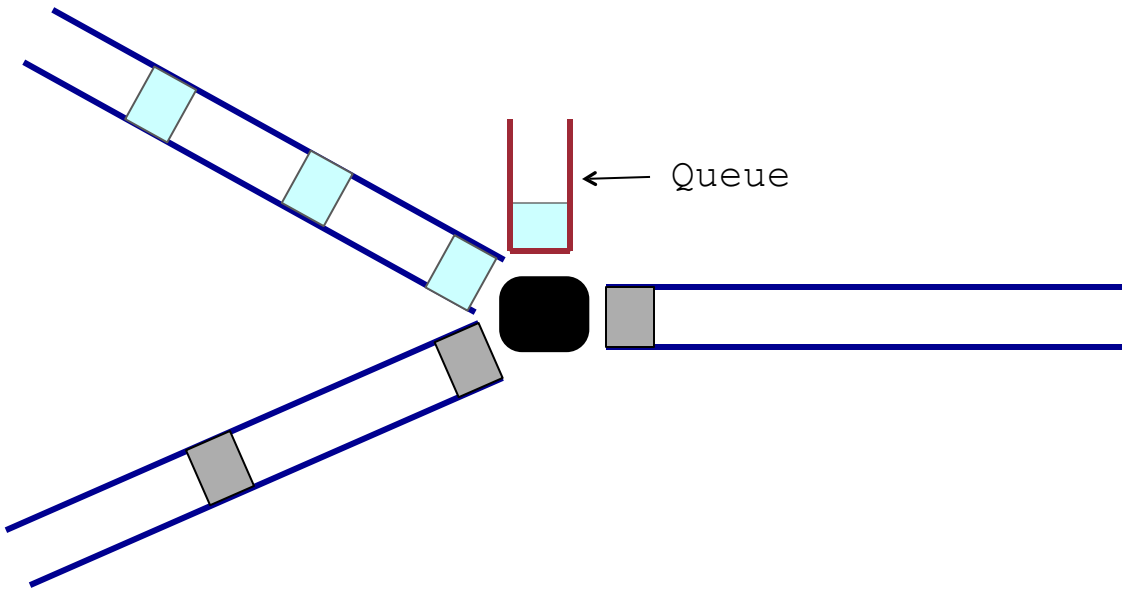
Transient Overload

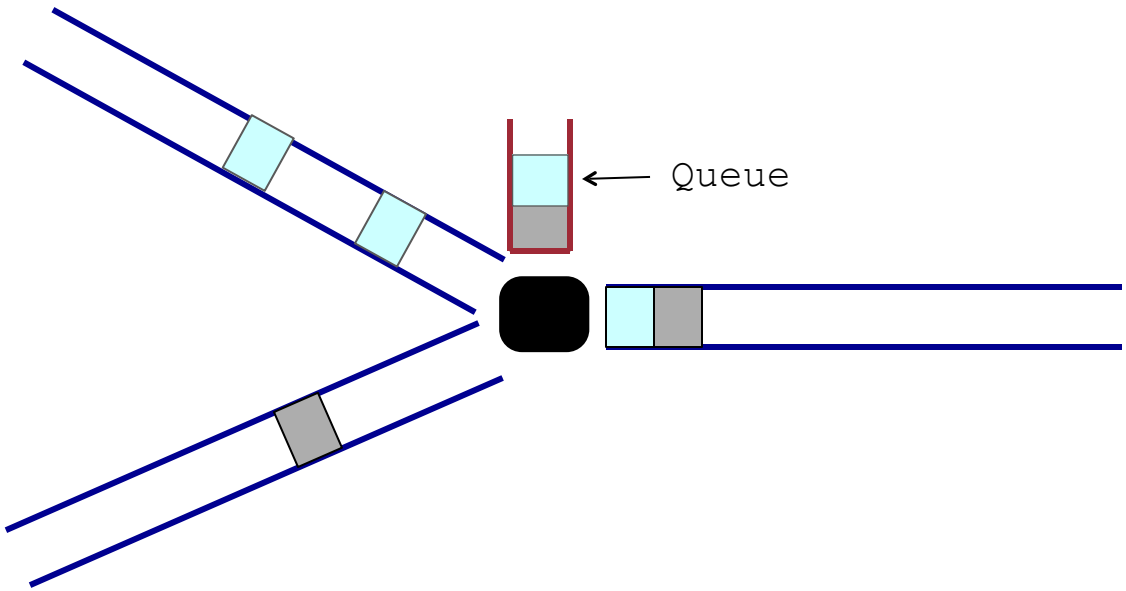
Not a rare event!

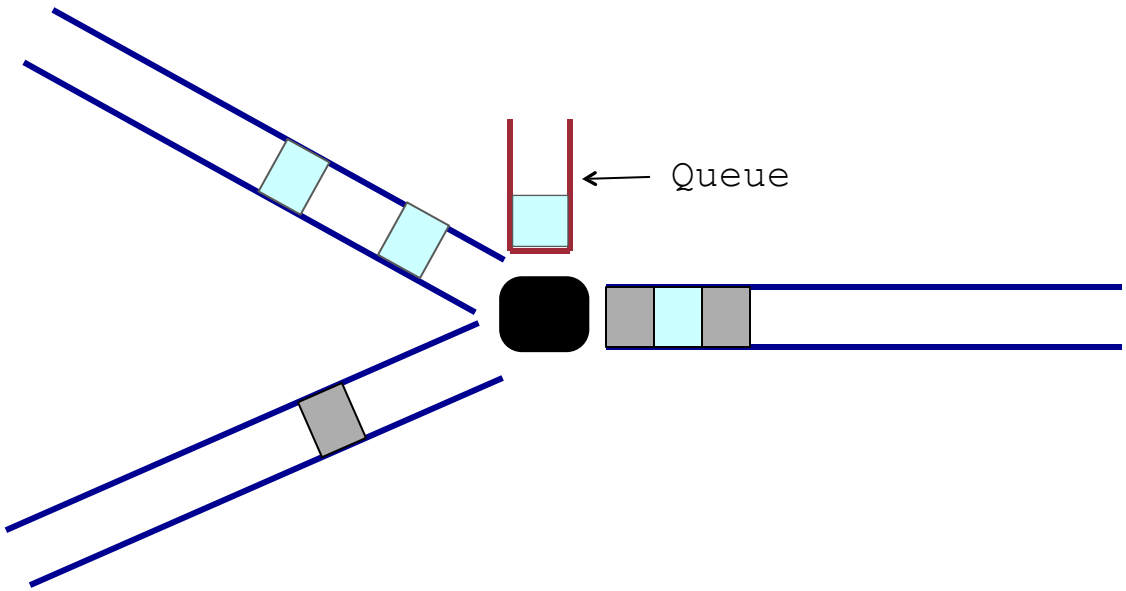


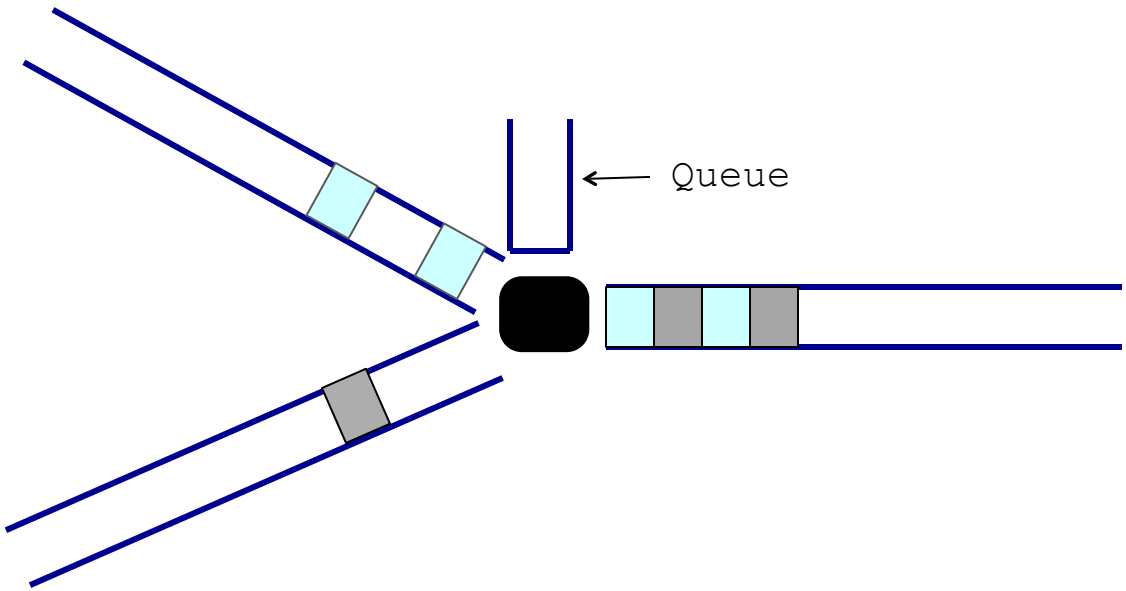
Transient Overload

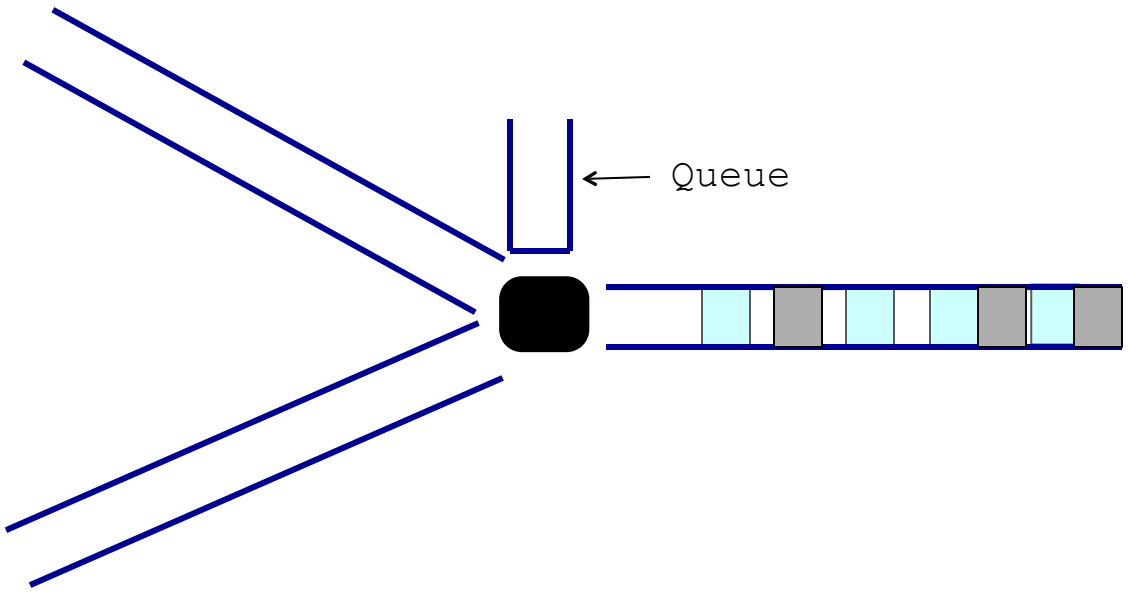
Not a rare event!

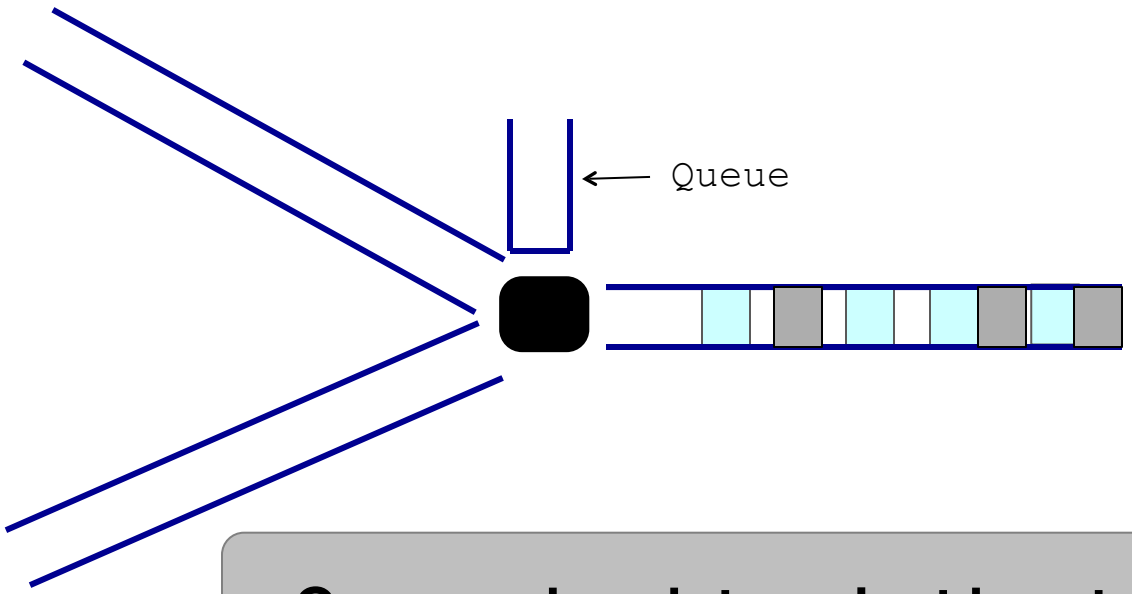

Queue


Queue

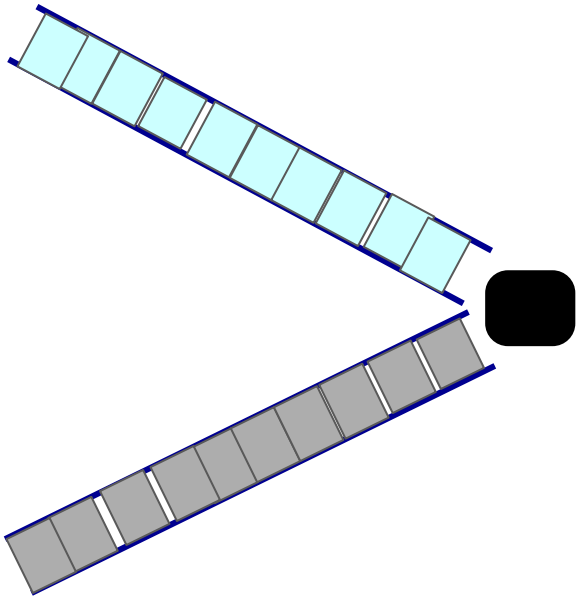

Queue

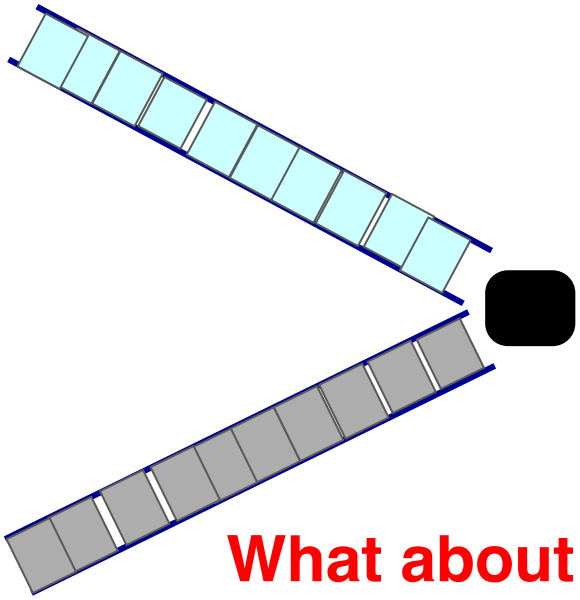

Queue


Queue

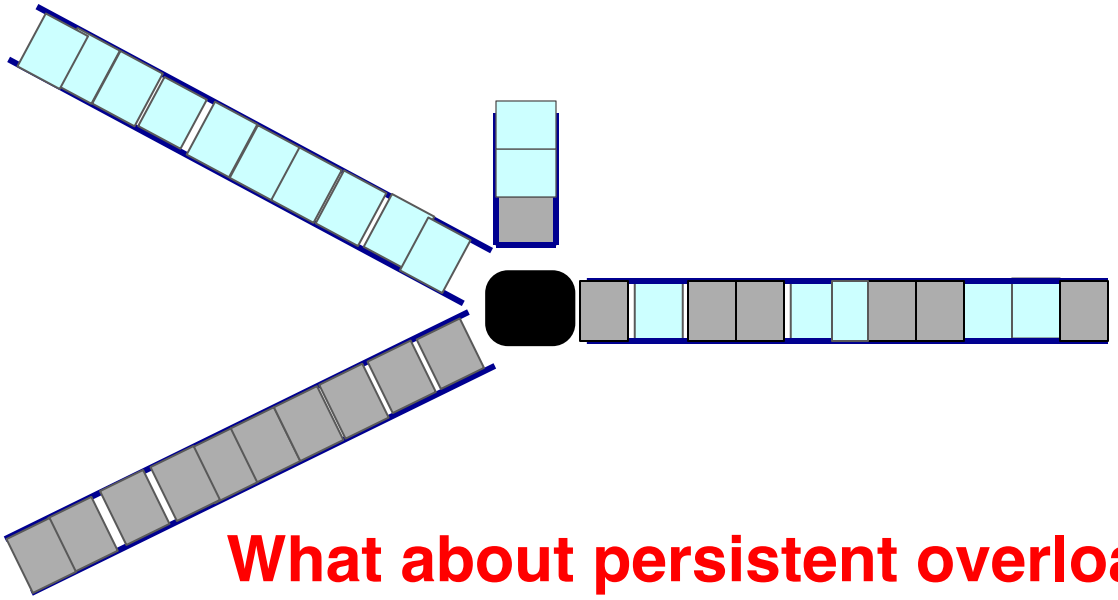


Queues absorb transient bursts!

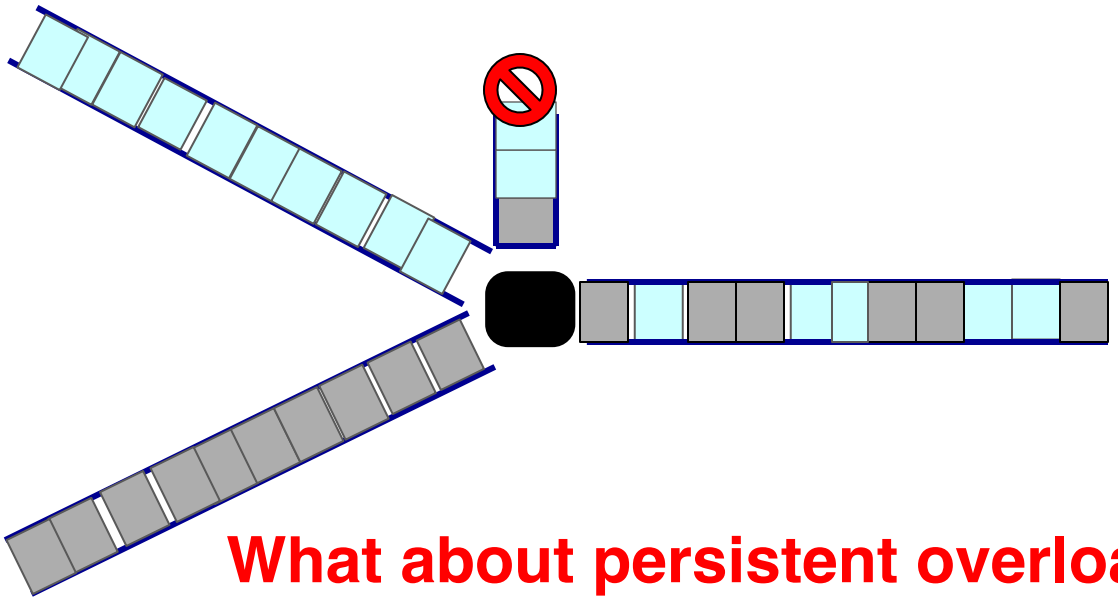




What about persistent overload?



What about persistent overload?



What about persistent overload?

Will eventually drop packets

Queues introduce queuing delays

- **Recall, packet delay = transmission delay + propagation delay**
- **With queues: packet delay = transmission delay + propagation delay + queueing delay**

Recall: life of a packet so far…

- Source has some data to send to a destination
- Chunks it up into packets: each packet has a payload and a header
- Packet travels along a link
- Arrives at a switch; switch forwards the packet to its next hop
- And the last step repeats until we reach the destination …

Recall: life of a packet so far… [updated]

- Source has some data to send to a destination
- Chunks it up into packets: each packet has a payload and a header
- Packet travels along a link
- Arrives at a switch; switch forwards the packet to its next hop
  - switch may buffer, or even drop, the packet
- And the last step repeats until we reach the destination …
  - or the packet is dropped

Challenge: Reliable packet delivery

Challenge: Reliable packet delivery

- **Packets can be dropped along the way**
  - Buffers in switch can overflow
  - Switch can crash while buffering packets
  - Links can garble/corrupt packets

Challenge: Reliable packet delivery

- **Packets can be dropped along the way**
  - Buffers in switch can overflow
  - Switch can crash while buffering packets
  - Links can garble/corrupt packets
- **Given an unreliable network, how do we make sure the destination receives its packets?**
  - Or at least know if they are delivered….

Challenge: Congestion control

Challenge: Congestion control

- Packet switching means network capacity is allocated on-demand

Challenge: Congestion control

- Packet switching means network capacity is allocated on-demand
- But endhosts independently decide at what rate they will send packets!

Challenge: Congestion control

- Packet switching means network capacity is allocated on-demand
- But endhosts independently decide at what rate they will send packets!
- This can be tricky!
  - How fast I send packets impacts whether *your* packets are dropped
  - What's a good rate at which I should send my packets?

Challenge: Congestion control

- Packet switching means network capacity is allocated on-demand
- But endhosts independently decide at what rate they will send packets!
- This can be tricky!
  - How fast I send packets impacts whether *your* packets are dropped
  - What's a good rate at which I should send my packets?
- Hence, congestion control:
  - How do we ensure that (endhosts') independent decisions lead to a good outcome?

Hence, our fundamental topics [updated]

- How do we name endhosts on the Internet? **(naming)**
- How do we address endhosts? **(addressing)**
- How do we map names to addresses? **(mapping names to addresses)**
- How do we compute forwarding tables? **(routing control plane → project 1)**
- How do we forward packets? **(routing data plane)**
- How do hosts communicate reliably? **(reliable packet delivery → project 3)**
- How do sources know at what rate they can send packets? **(congestion control)**

Hence, our fundamental topics [updated]

- How do we name endhosts on the Internet? **(naming)**
- How do we address endhosts? **(addressing)**
- How do we map names to addresses? **(mapping names to addresses)**
- How do we compute forwarding tables? **(routing control plane → project 1)**
- How do we forward packets? **(routing data plane)**
- How do hosts communicate reliably? **(reliable packet delivery → project 3)**
- How do sources know at what rate they can send packets? **(congestion control)**
- **How do our solutions change in modern environments (clouds, wifi, cellular)**

Recap: key takeaways from our bottom-up overview

- **What is a packet?**
- **Approaches to sharing the network – circuit vs. packet switching -- and their tradeoffs**
- **An overall sense of the life of a packet**
  - We'll continue to refine this picture over the course of the semester

Questions??

Changing Perspective

- Designing the Internet: a top-down approach
- In the process, discuss a few enduring ideas:
  - Layering
  - The end-to-end principle
  - Fate sharing

The Internet's problem definition

- Support the transfer of data between endhosts
- ... across multiple networks
  - The Internet

How do you solve a problem?

1. Decompose it (into tasks and abstractions)

How do you solve a problem?

1. Decompose it (into tasks and abstractions)

2. Assign tasks to entities (who does what)

Modularity

Modularity based on abstraction is the way things are done

– Barbara Liskov, Turing lecture

What is modularity?

What is modularity?

- Decomposing systems into smaller units
  - Providing a “separation of concerns”

What is modularity?

- Decomposing systems into smaller units
  - Providing a “separation of concerns”
- Plays a crucial role in computer science…

What is modularity?

- Decomposing systems into smaller units
  - Providing a “separation of concerns”
- Plays a crucial role in computer science…

What is modularity?

- Decomposing systems into smaller units
  - Providing a “separation of concerns”
- Plays a crucial role in computer science…
- The challenge is to find the *right* modularity

Network Modularity

Network Modularity

- The need for modularity still applies

Network Modularity

- The need for modularity still applies
  - **And is even more important!**

Network Modularity

- The need for modularity still applies
  - **And is even more important!**
- Normal modularity organizes code

Network Modularity

- The need for modularity still applies
  - **And is even more important!**
- Normal modularity organizes code
- But network implementations are not just distributed across many lines of code…

Network Modularity

- The need for modularity still applies
  - **And is even more important!**
- Normal modularity organizes code
- But network implementations are not just distributed across many lines of code…
  - Also distributed across many devices (hosts, routers)

Network Modularity

- The need for modularity still applies
  - **And is even more important!**
- Normal modularity organizes code
- But network implementations are not just distributed across many lines of code…
  - Also distributed across many devices (hosts, routers)
  - … *and* different players (clients, server, ISPs)

How do we decompose the job of transferring data between end-hosts?

Inspiration…

Inspiration…

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

Dear Sam,

Your days are numbered.

-- Sundar

Inspiration…

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

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**

The Path of the Letter

CEO

Aide

FedEx

CEO

Aide

FedEx

The Path of the Letter

The Path of the Letter

CEO

Aide

FedEx

CEO

Aide

FedEx

The Path of the Letter

The Path of the Letter

CEO

Aide

FedEx

CEO

Aide

FedEx

The Path of the Letter

CEO

Aide

FedEx

The Path of the Letter

CEO

Aide

FedEx

CEO

Aide

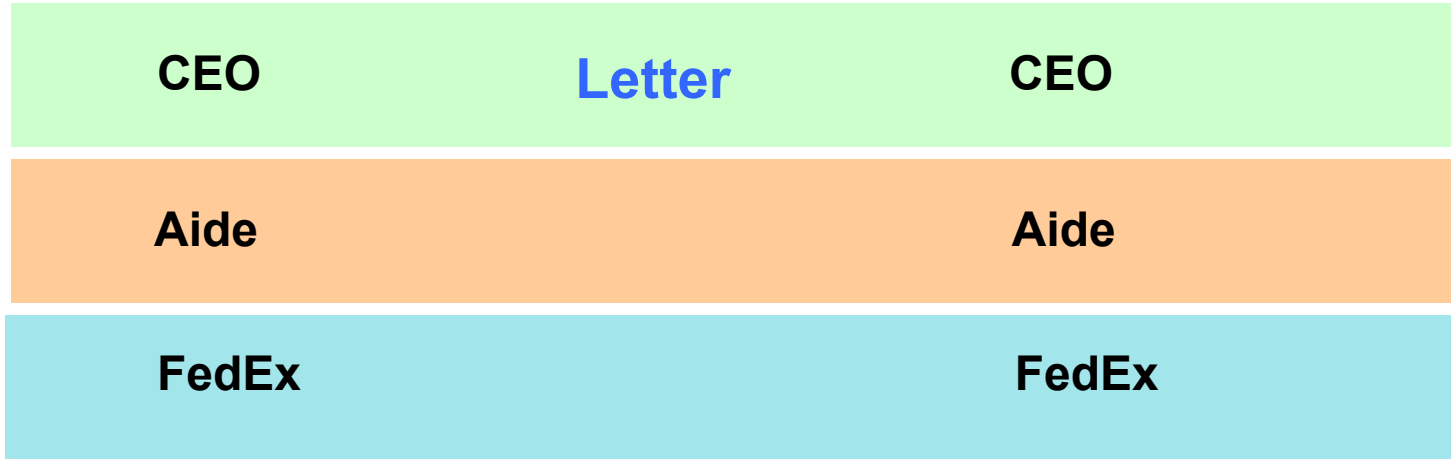
FedEx

The Path of the Letter

CEO	CEO
Aide	Aide
FedEx	FedEx

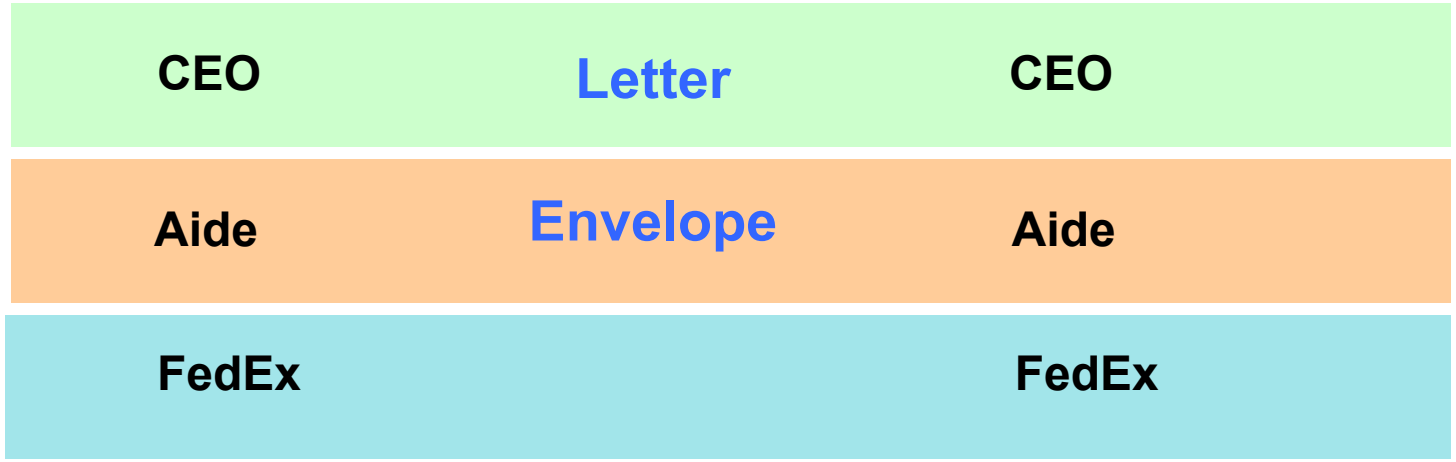
The Path of the Letter

- “Peers” understand the same things



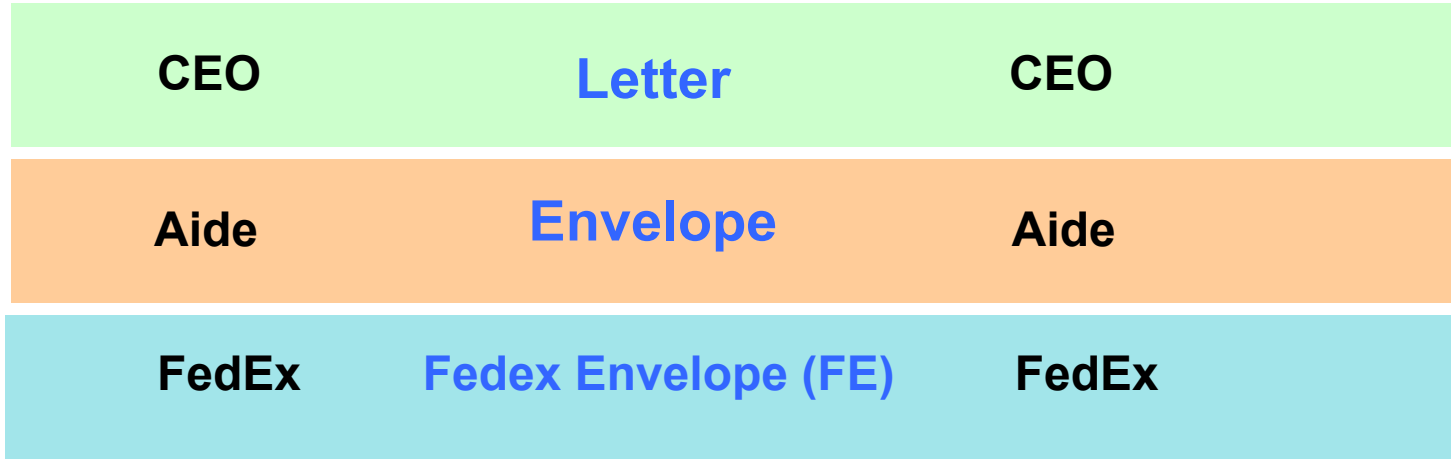
The Path of the Letter

- "Peers" understand the same things



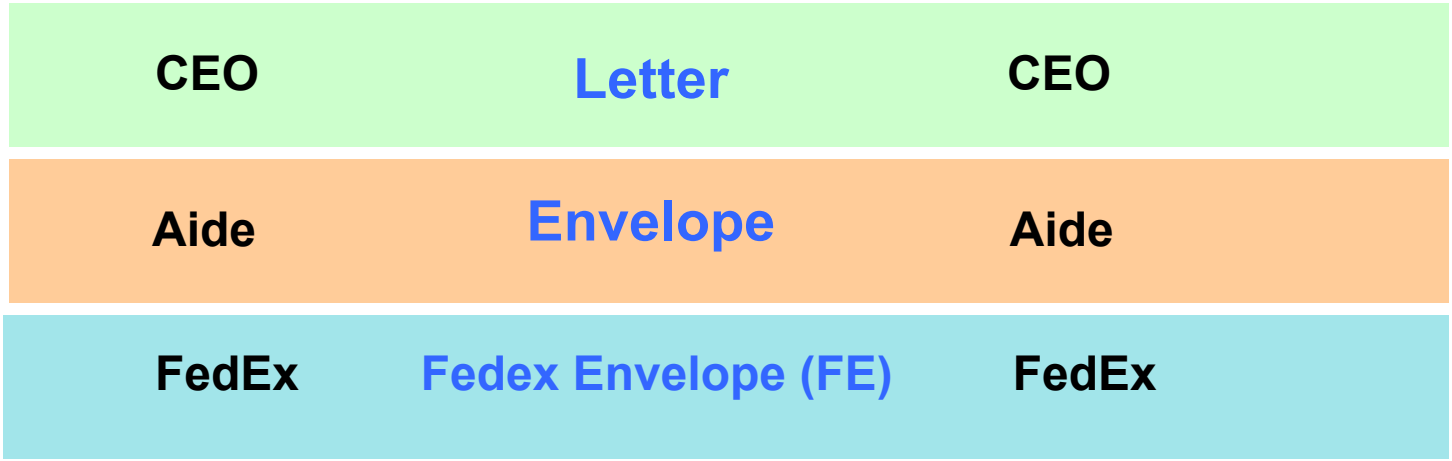
The Path of the Letter

- "Peers" understand the same things



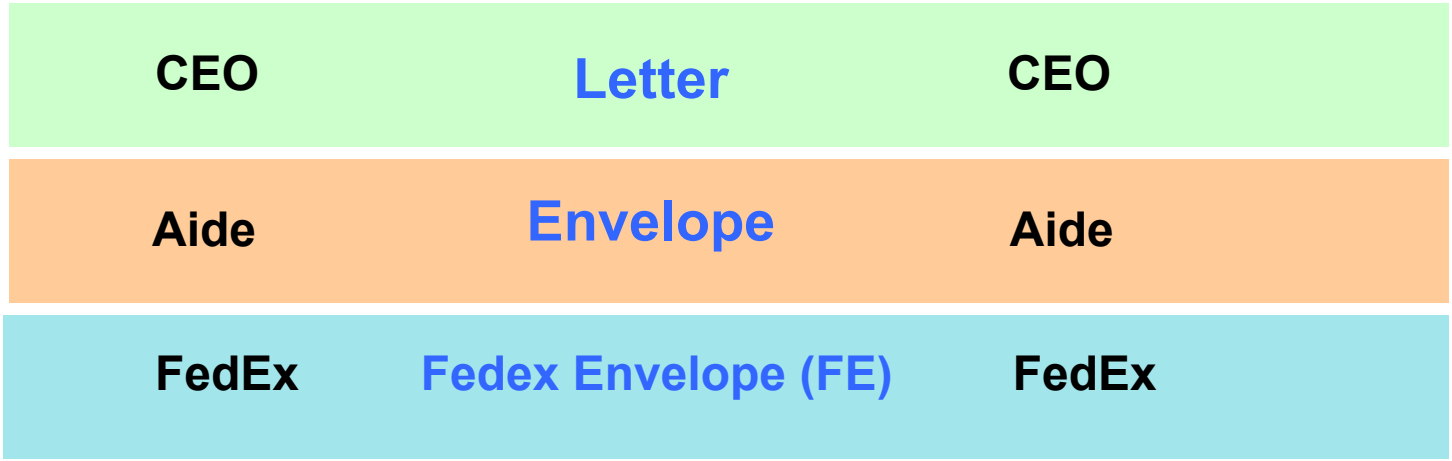
The Path of the Letter

- “Peers” understand the same things
- No one else needs to



The Path of the Letter

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



Thought Experiment

- How would ***you*** break the Internet into tasks?
- Just focus on what is needed to get packets between processes on different hosts….
- Do not consider application or control tasks
  - Naming, computing forwarding tables, etc.

Breakdown into Tasks

Breakdown into Tasks

- Bits across a link

Breakdown into Tasks

- Bits across a link
- Packets across a link

Breakdown into Tasks

- Bits across a link
- Packets across a link
- Deliver packets across a single network (“local” delivery)

Breakdown into Tasks

- Bits across a link
- Packets across a link
- Deliver packets across a single network (“local” delivery)
- Deliver packets across multiple networks (“global” delivery)

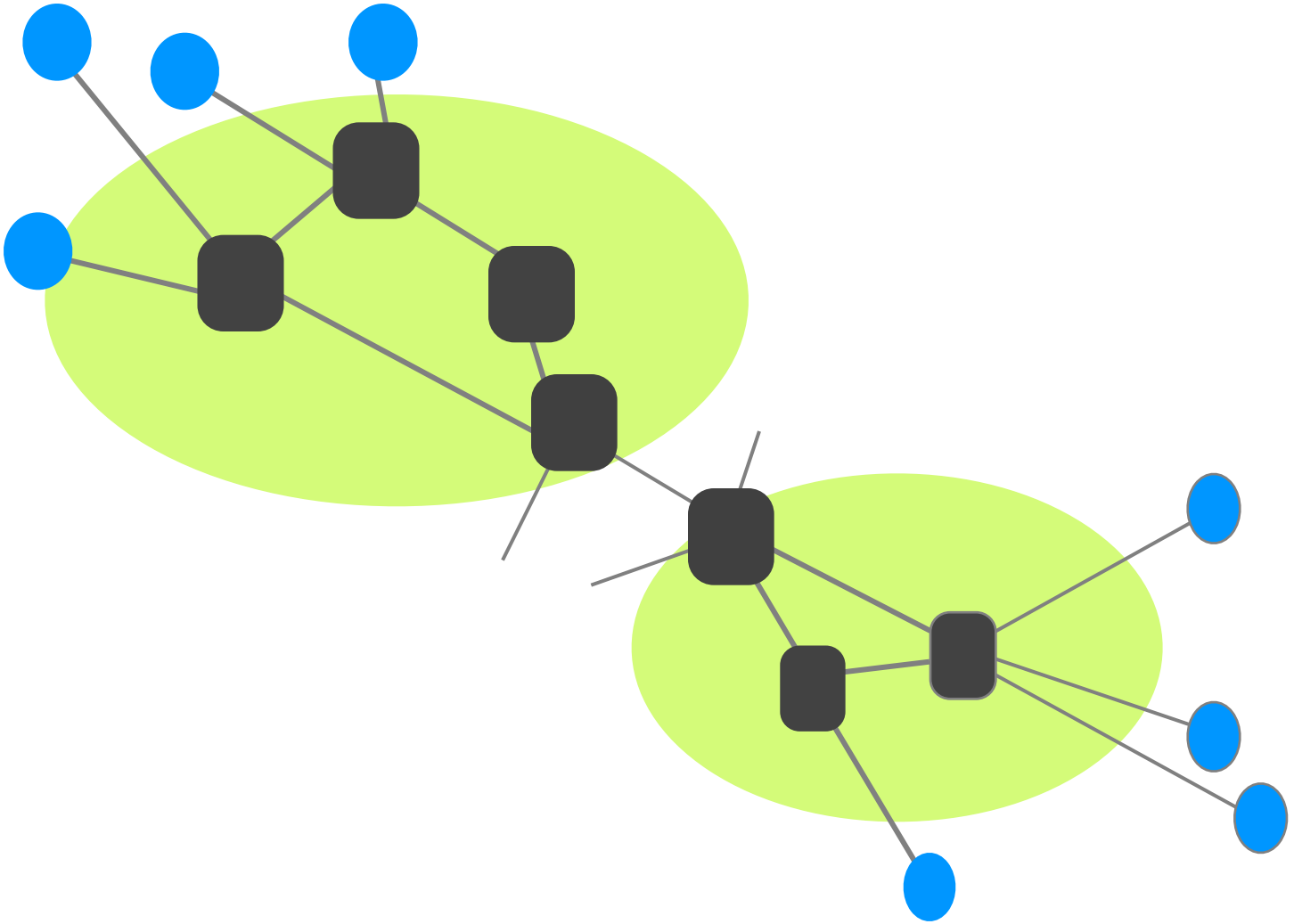
Breakdown into Tasks

- Bits across a link
- Packets across a link
- Deliver packets across a single network (“local” delivery)
- Deliver packets across multiple networks (“global” delivery)
- Deliver data reliably

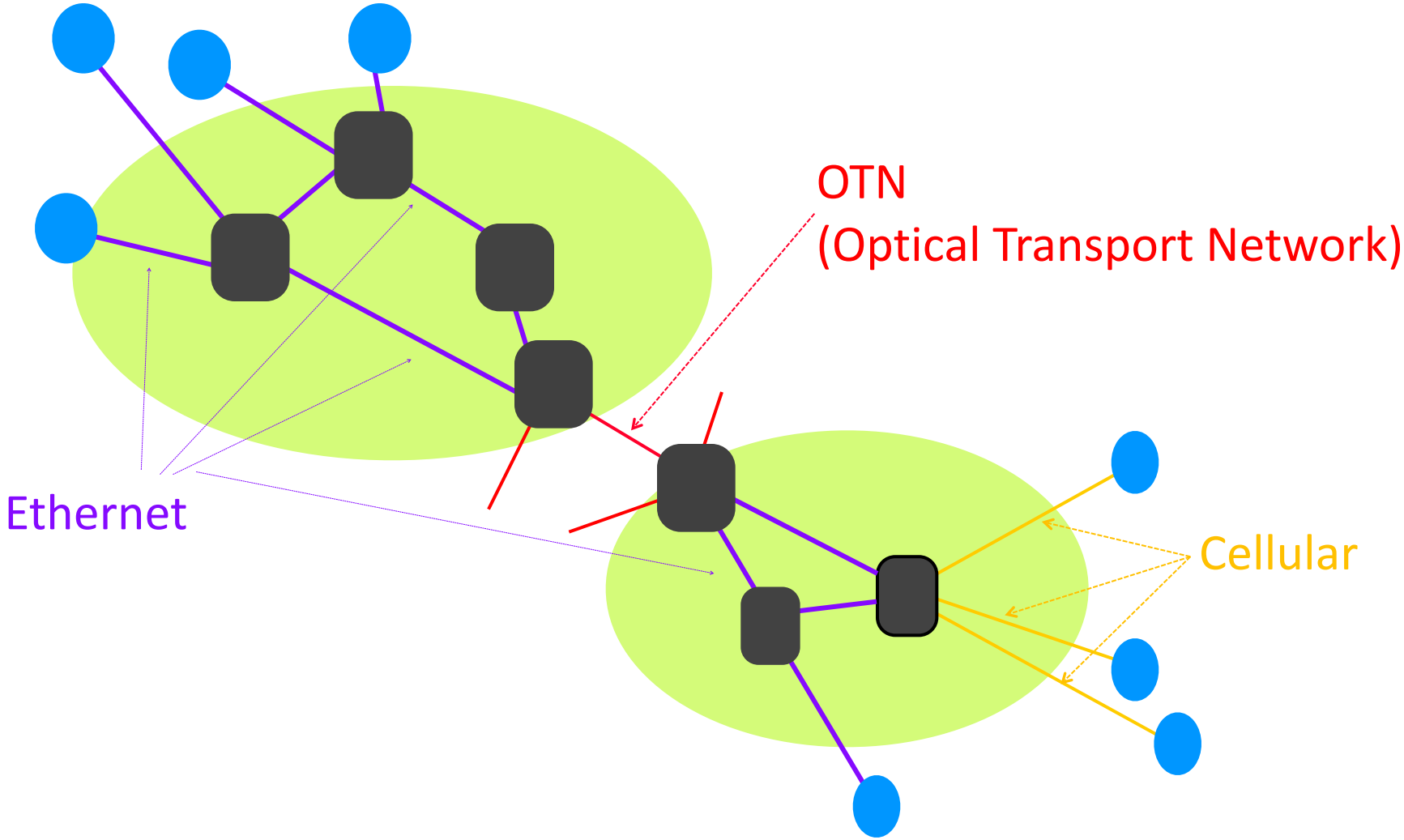
Breakdown into Tasks

- Bits across a link
- Packets across a link
- Deliver packets across a single network (“local” delivery)
- Deliver packets across multiple networks (“global” delivery)
- Deliver data reliably
- Do something with the data

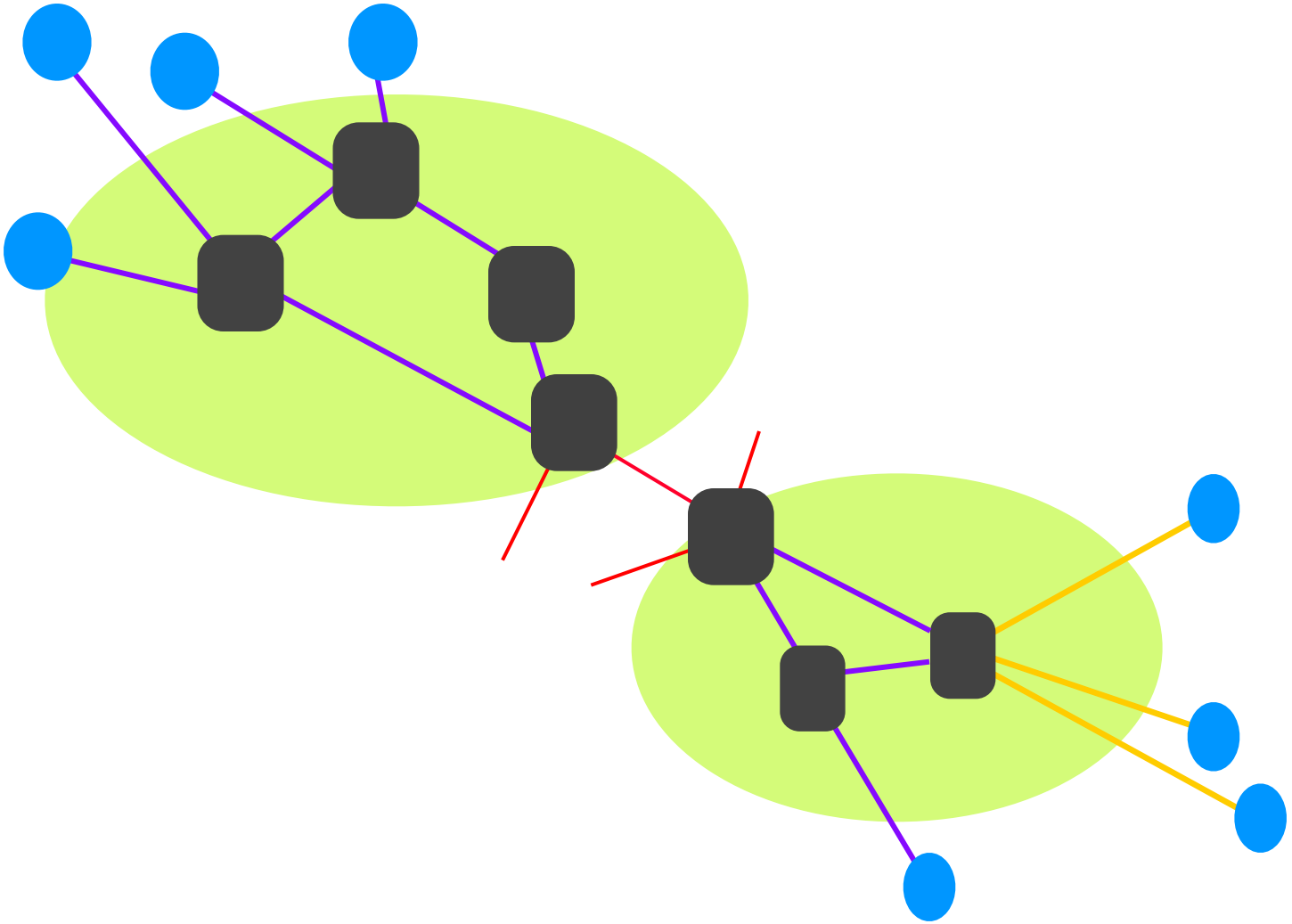
Local *vs.* Global Delivery



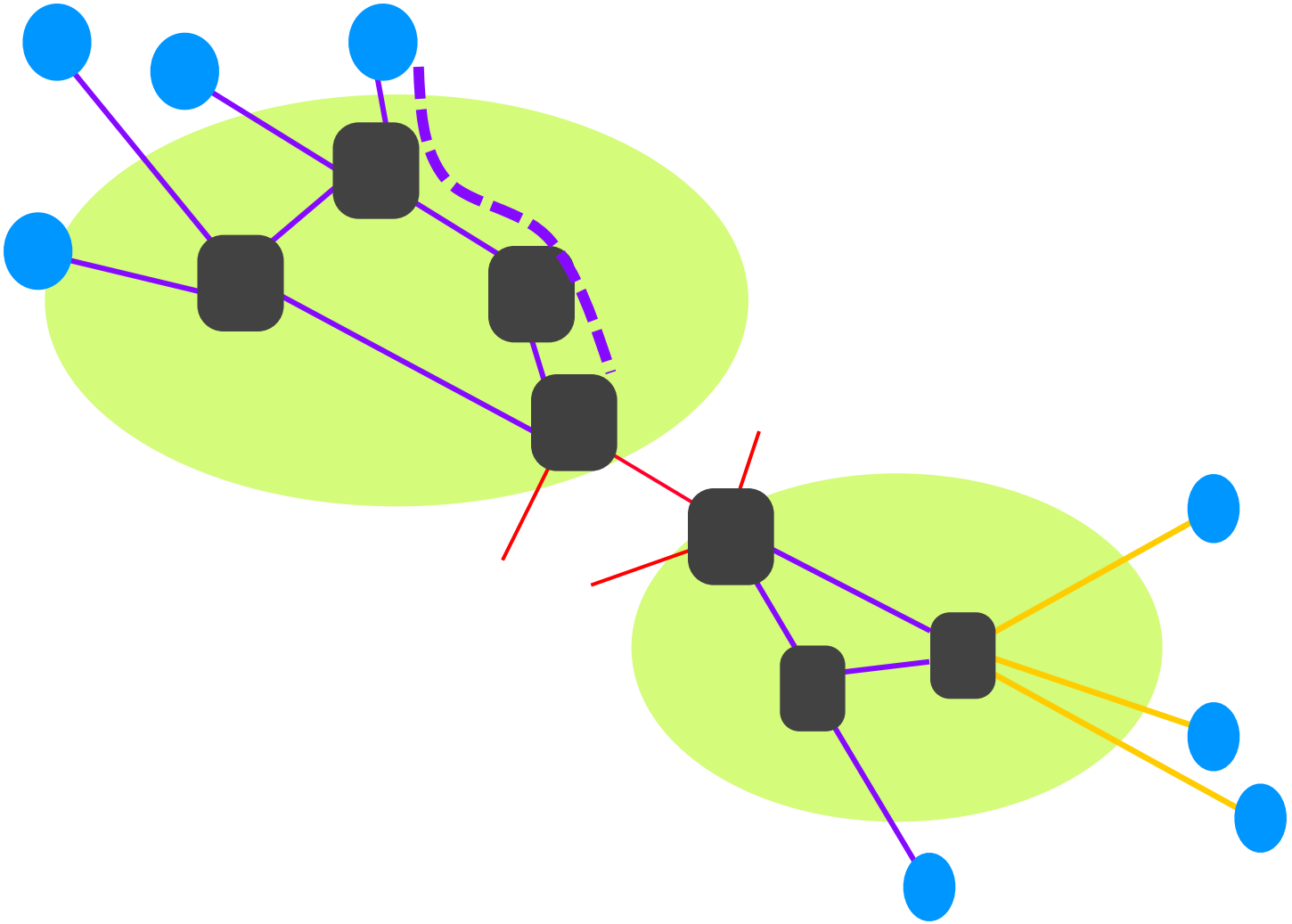
Local *vs.* Global Delivery



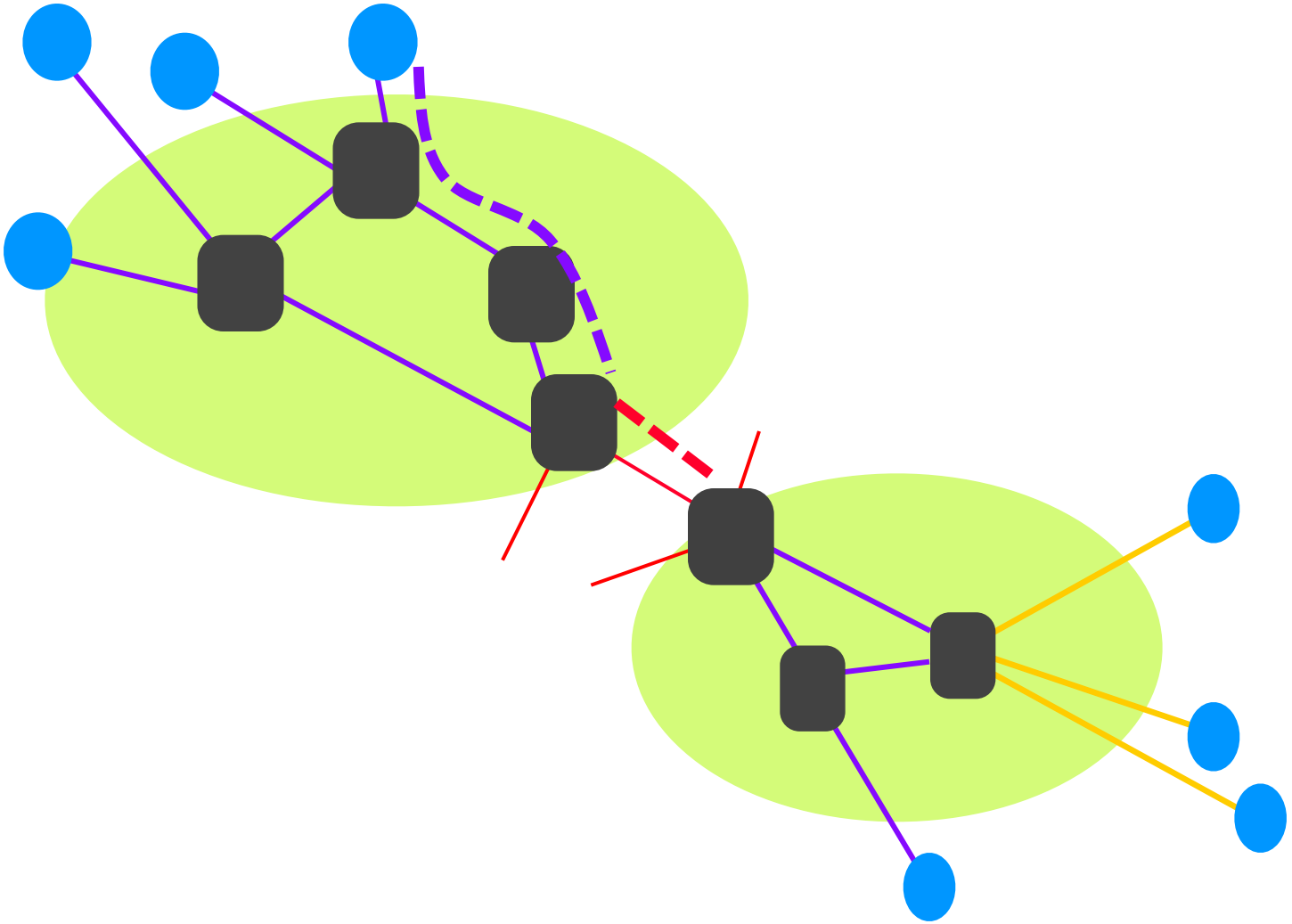
Local *vs.* Global Delivery



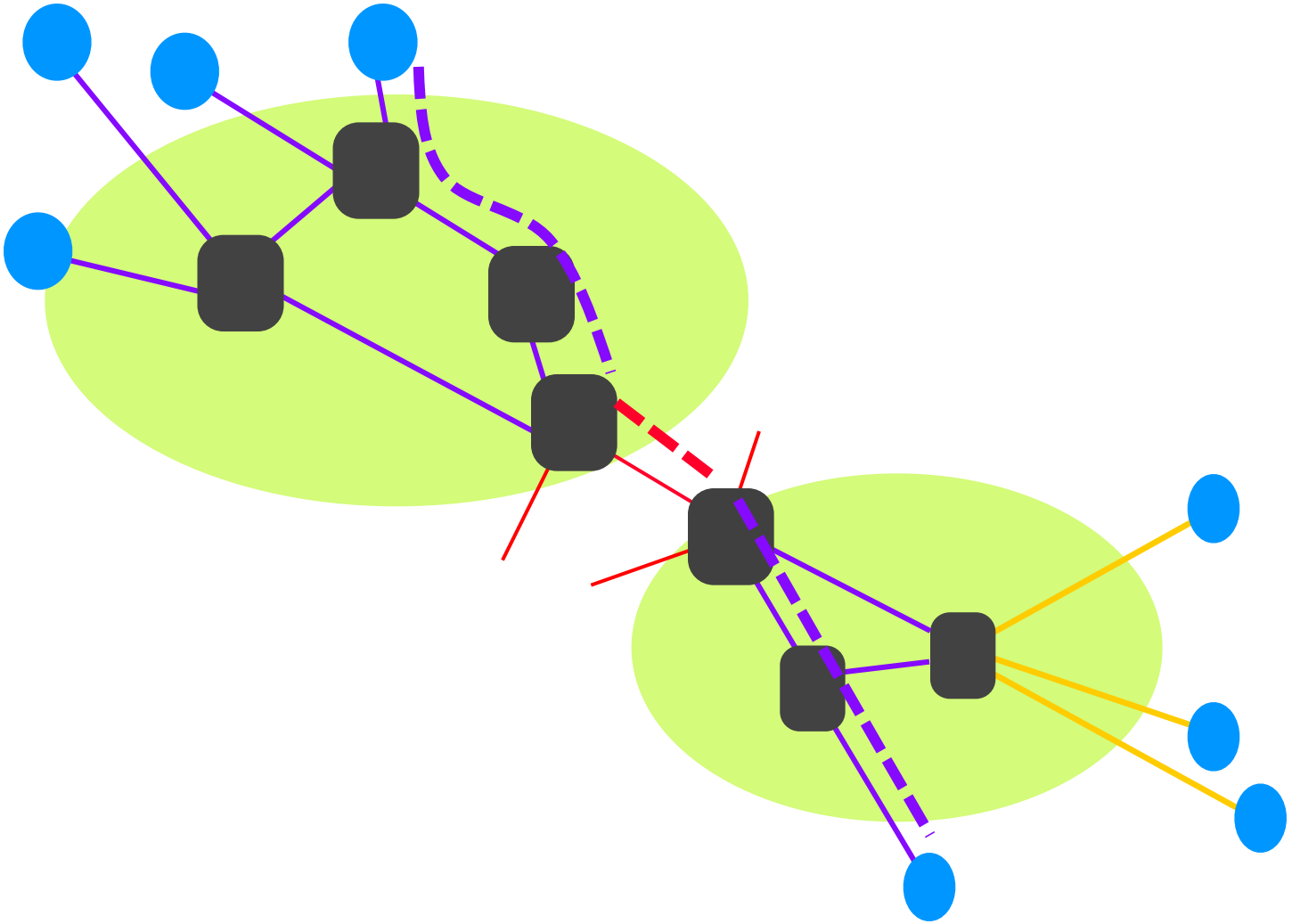
Local *vs.* Global Delivery



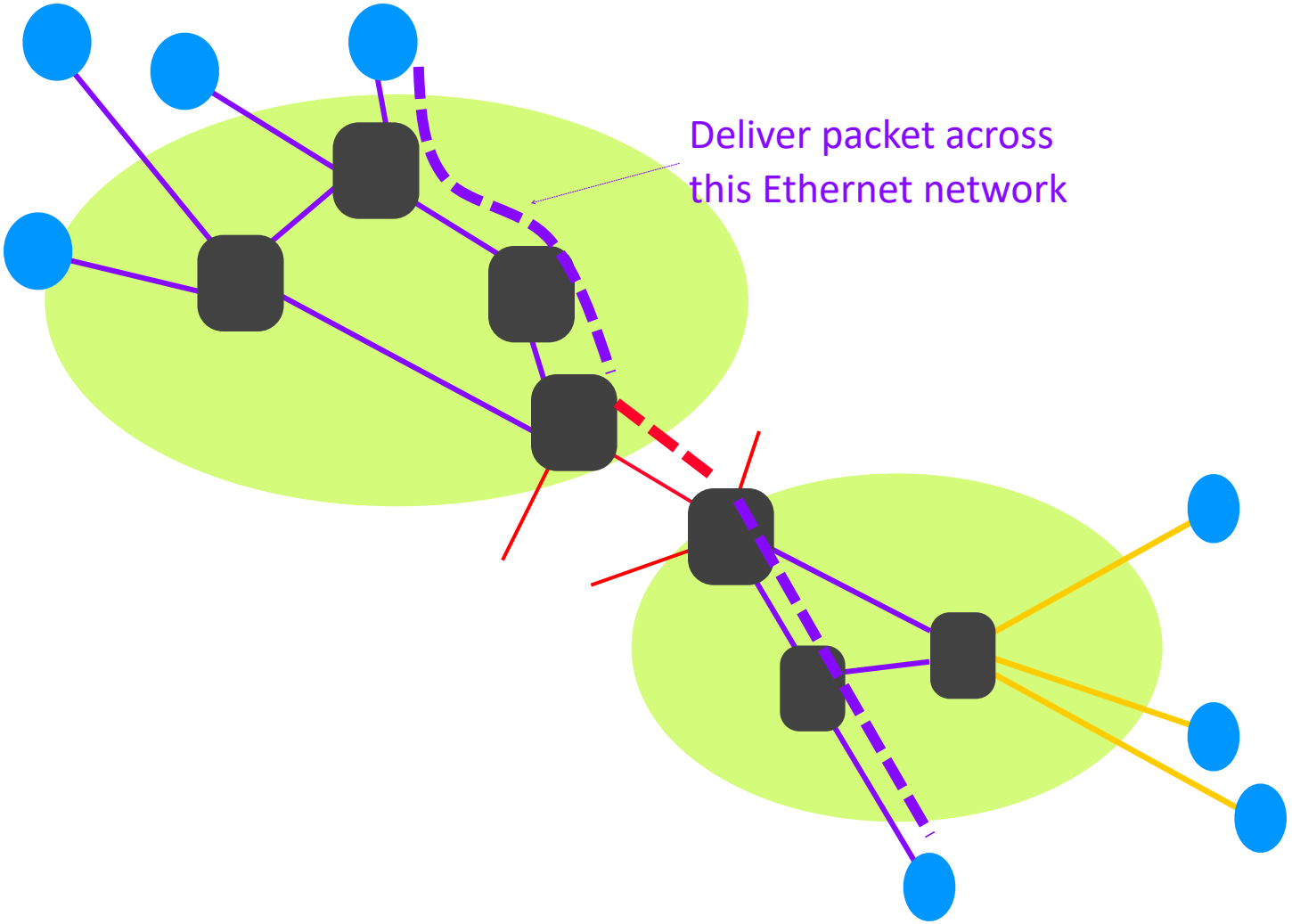
Local *vs.* Global Delivery



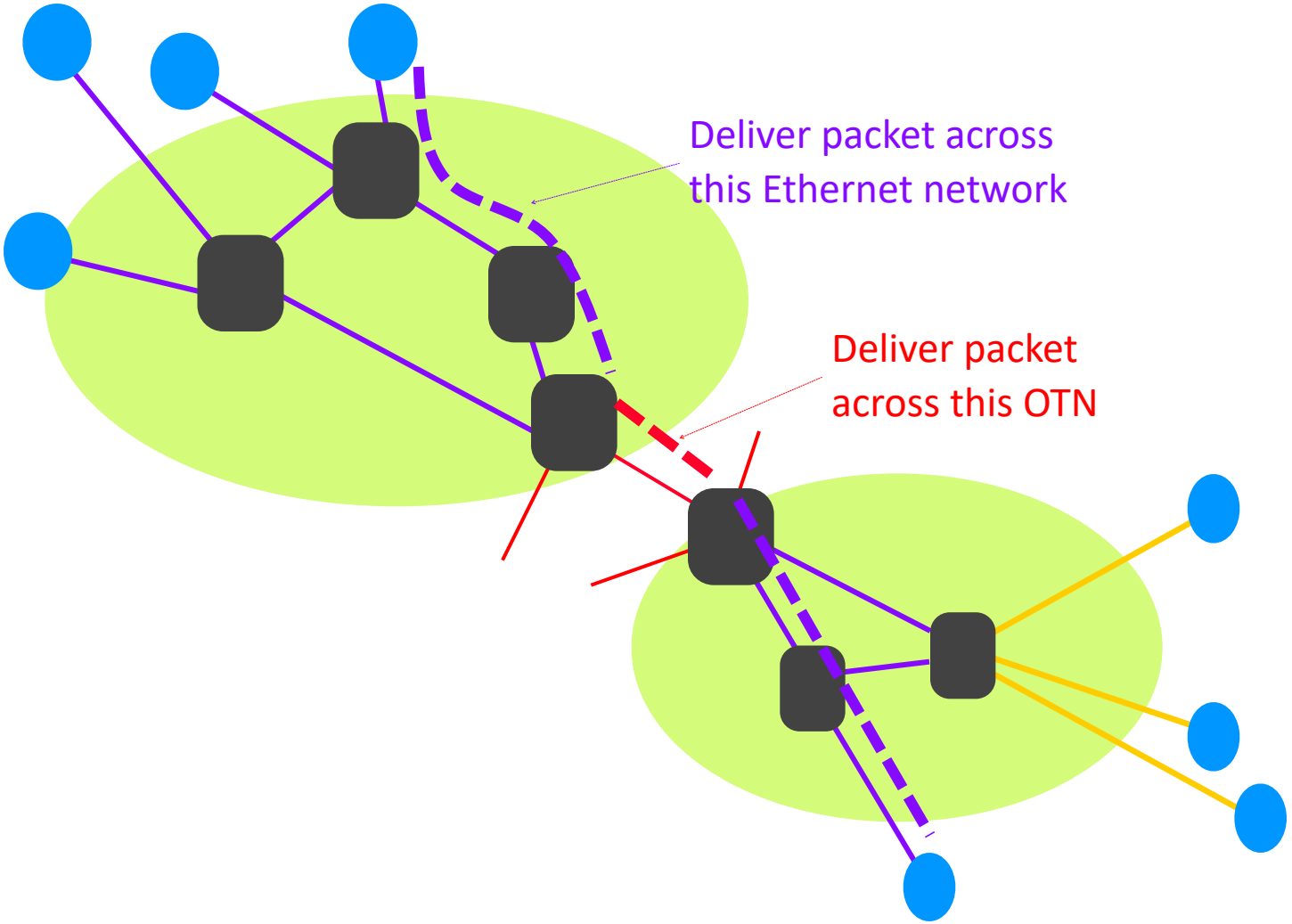
Local *vs.* Global Delivery



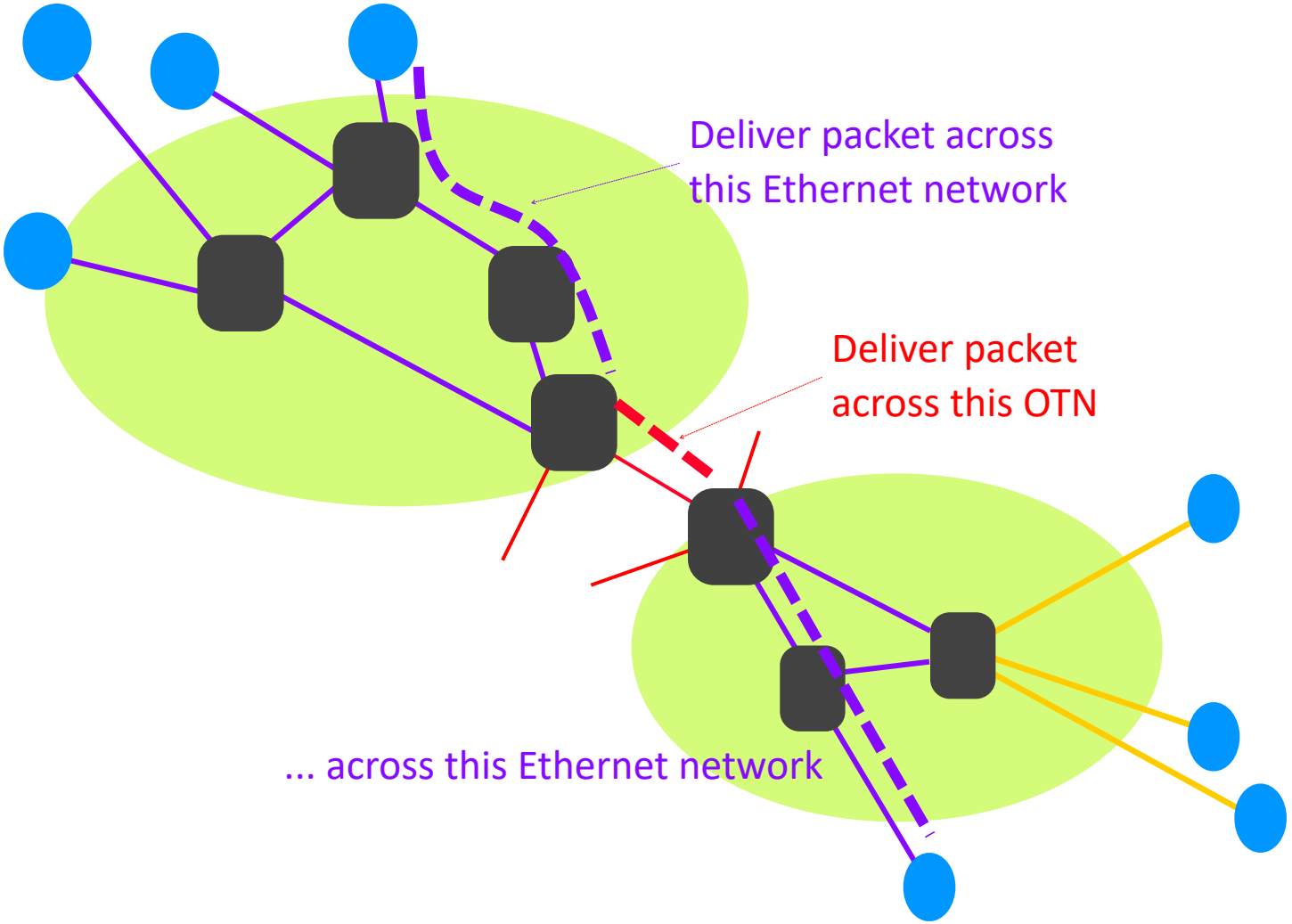
Local *vs.* Global Delivery



Local *vs.* Global Delivery



Local *vs.* Global Delivery



Breakdown into Tasks

- Bits across a link
- Packets across a link
- Deliver packets across a single network (“local” delivery)
- Deliver packets across multiple networks (“global” delivery)
- Deliver data reliably
- Do something with the data

Breakdown into Tasks

- Bits across a link
- Packets across a link
- Deliver packets across a single network (“local” delivery)
- Deliver packets across multiple networks (“global” delivery)
- Deliver data reliably
- Do something with the data

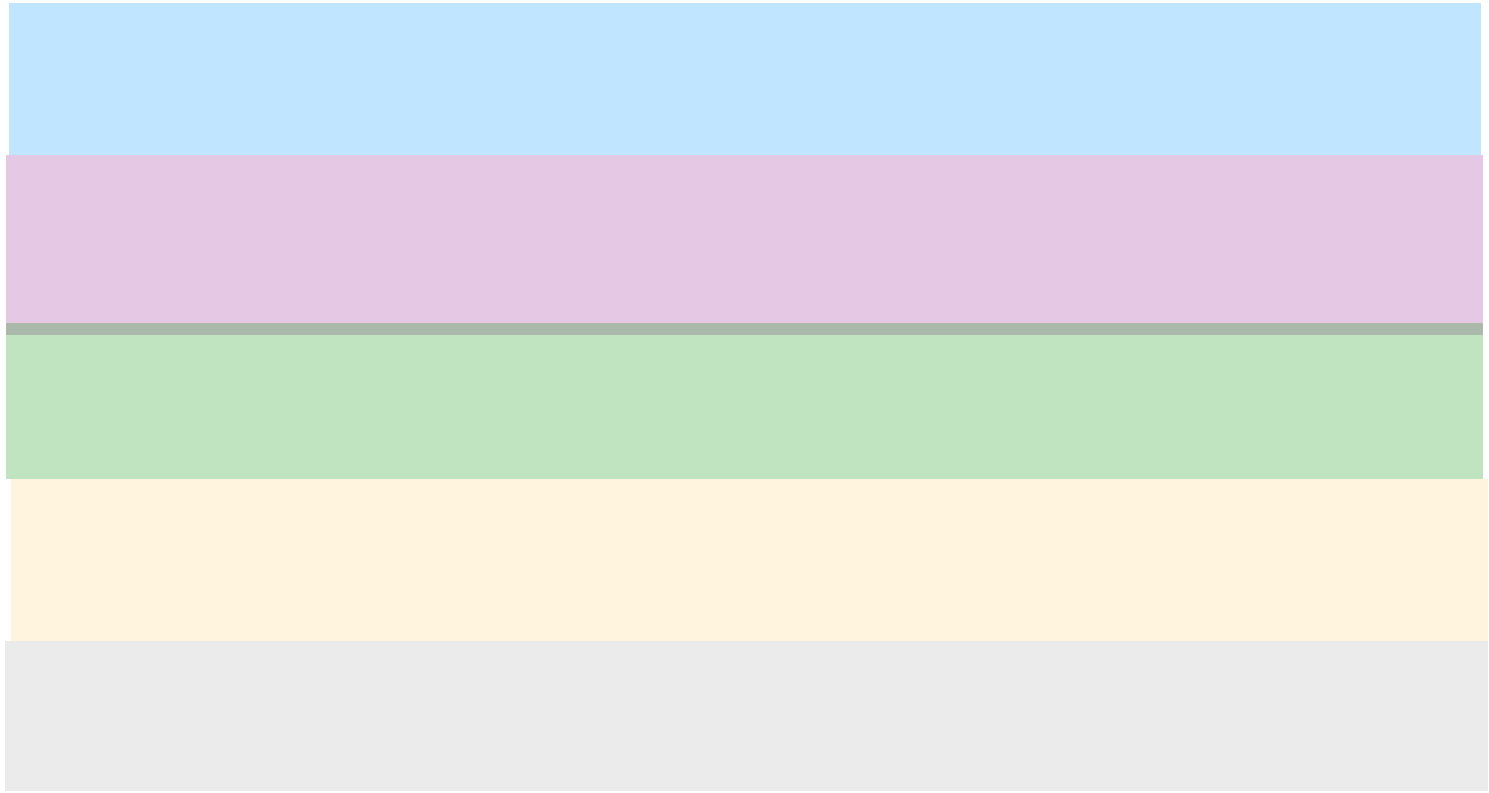
Breakdown into Tasks

- Bits across a link
- Packets across a single network (“local” delivery)
- Deliver packets across multiple networks (“global” delivery)
- Deliver data reliably
- Do something with the data

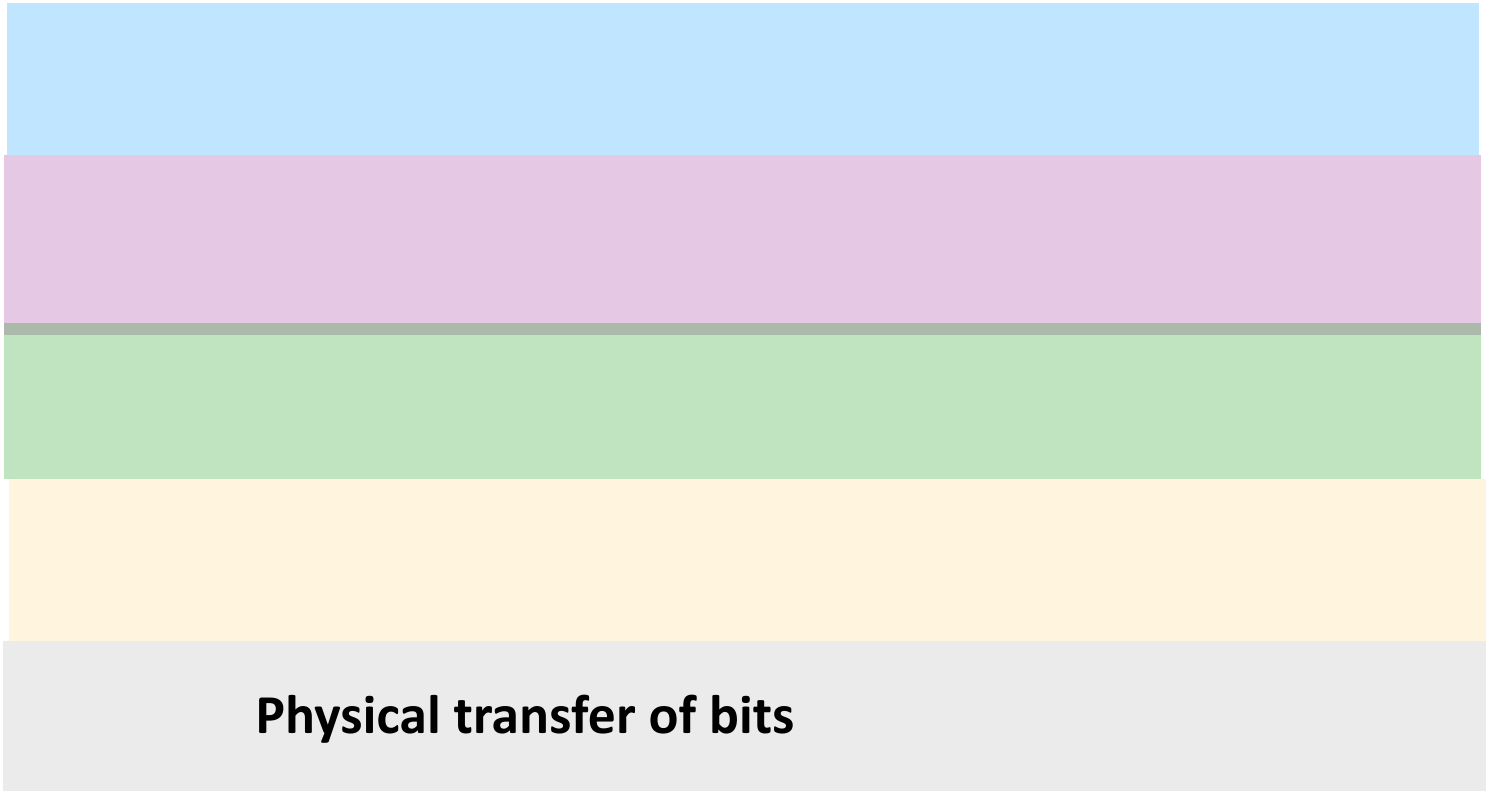
Breakdown into Tasks

- Bits across a link
- Packets across a single network (“local” delivery)
  - Local addresses
- Deliver packets across multiple networks (“global” delivery)
  - Global addresses
- Deliver data reliably
- Do something with the data

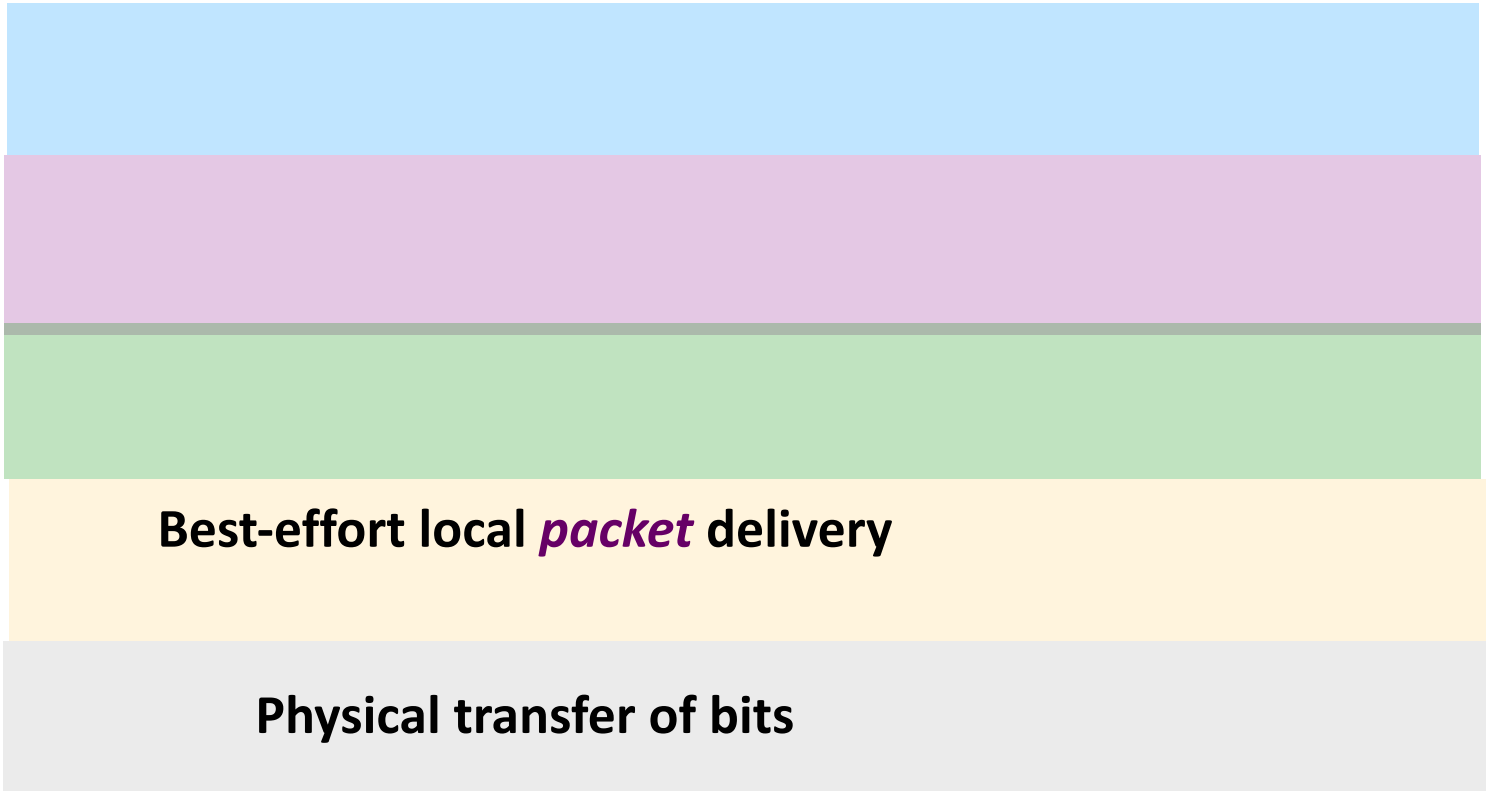
In the Internet: organization



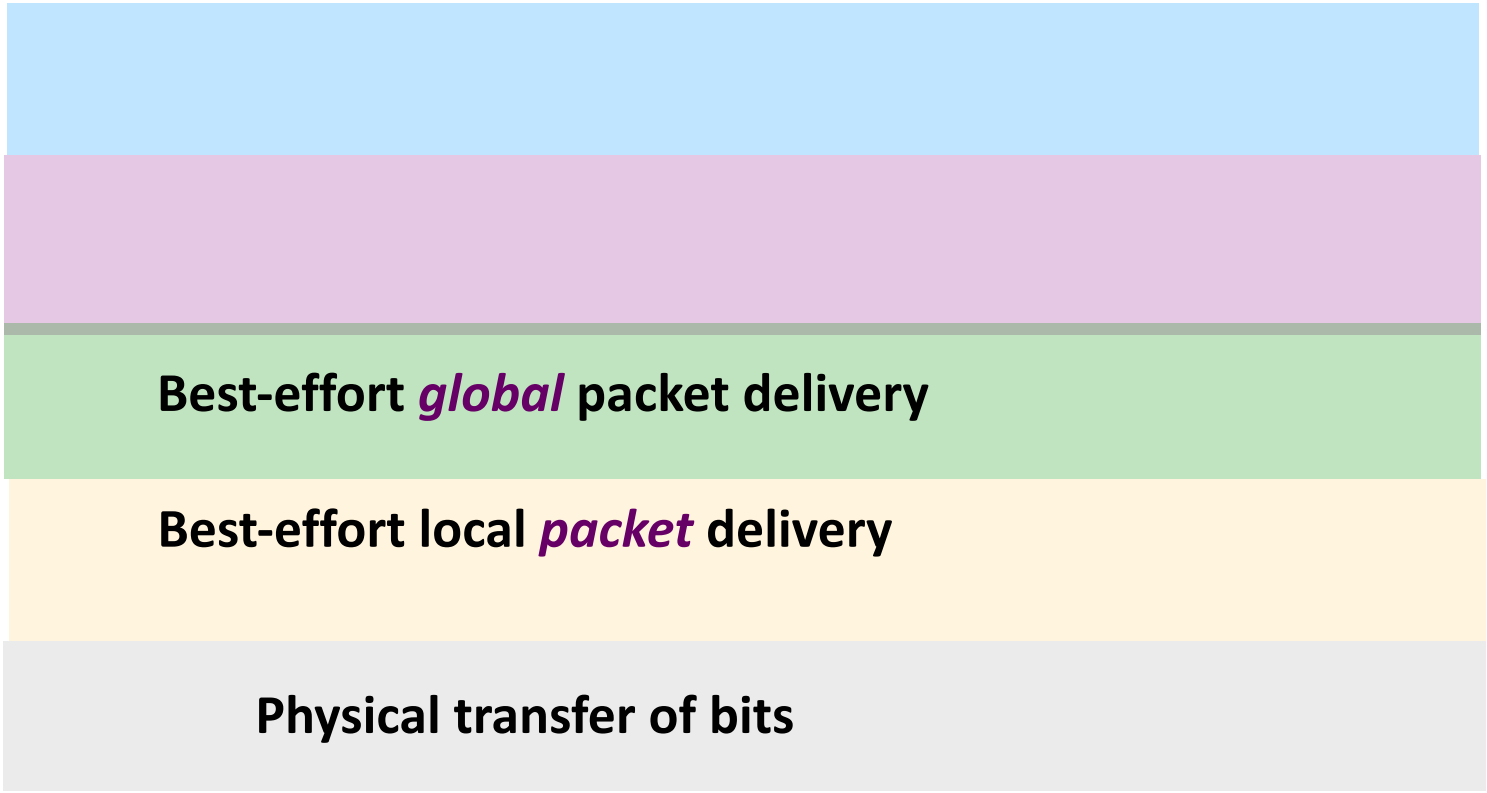
In the Internet: organization



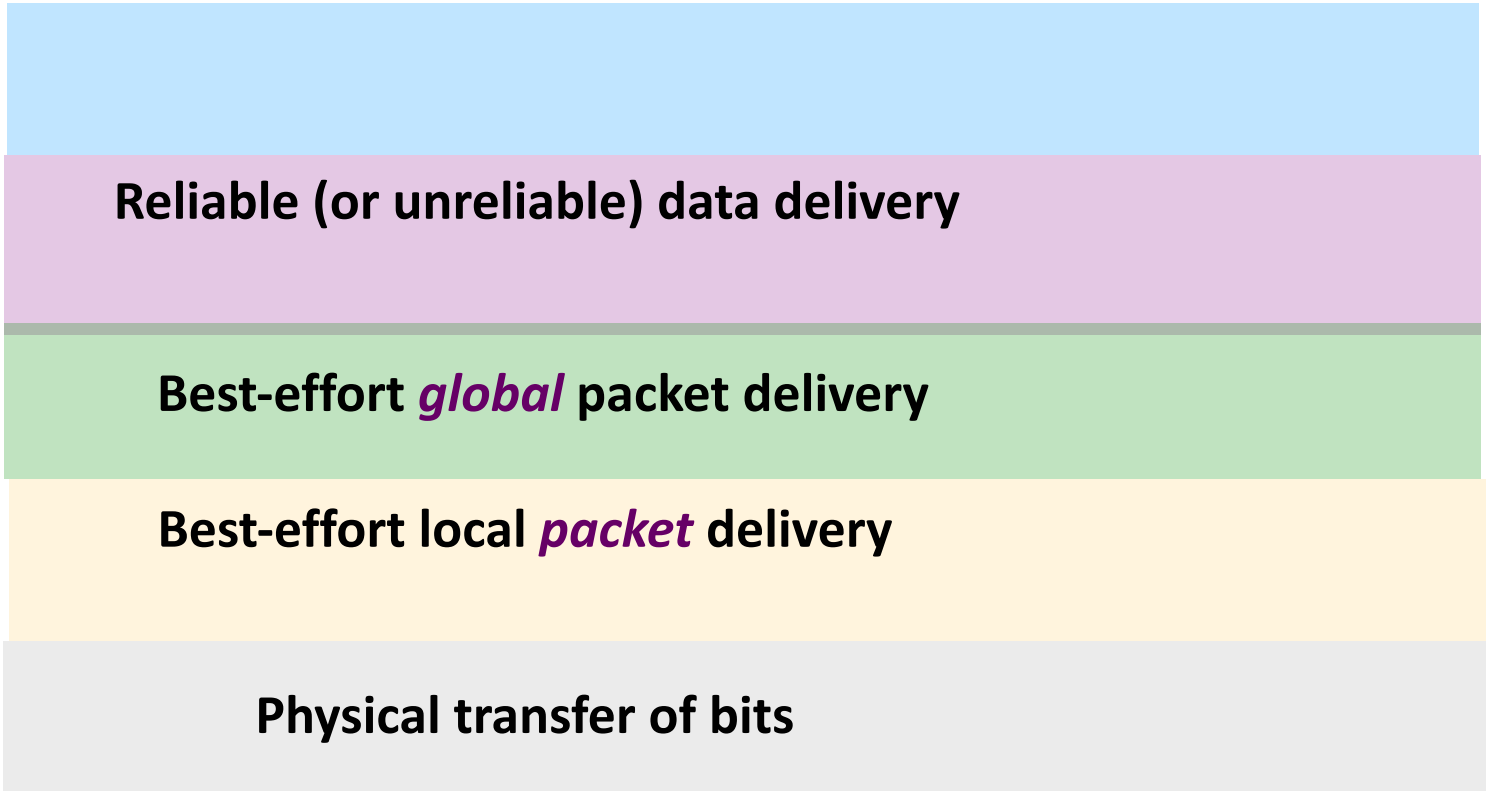
In the Internet: organization



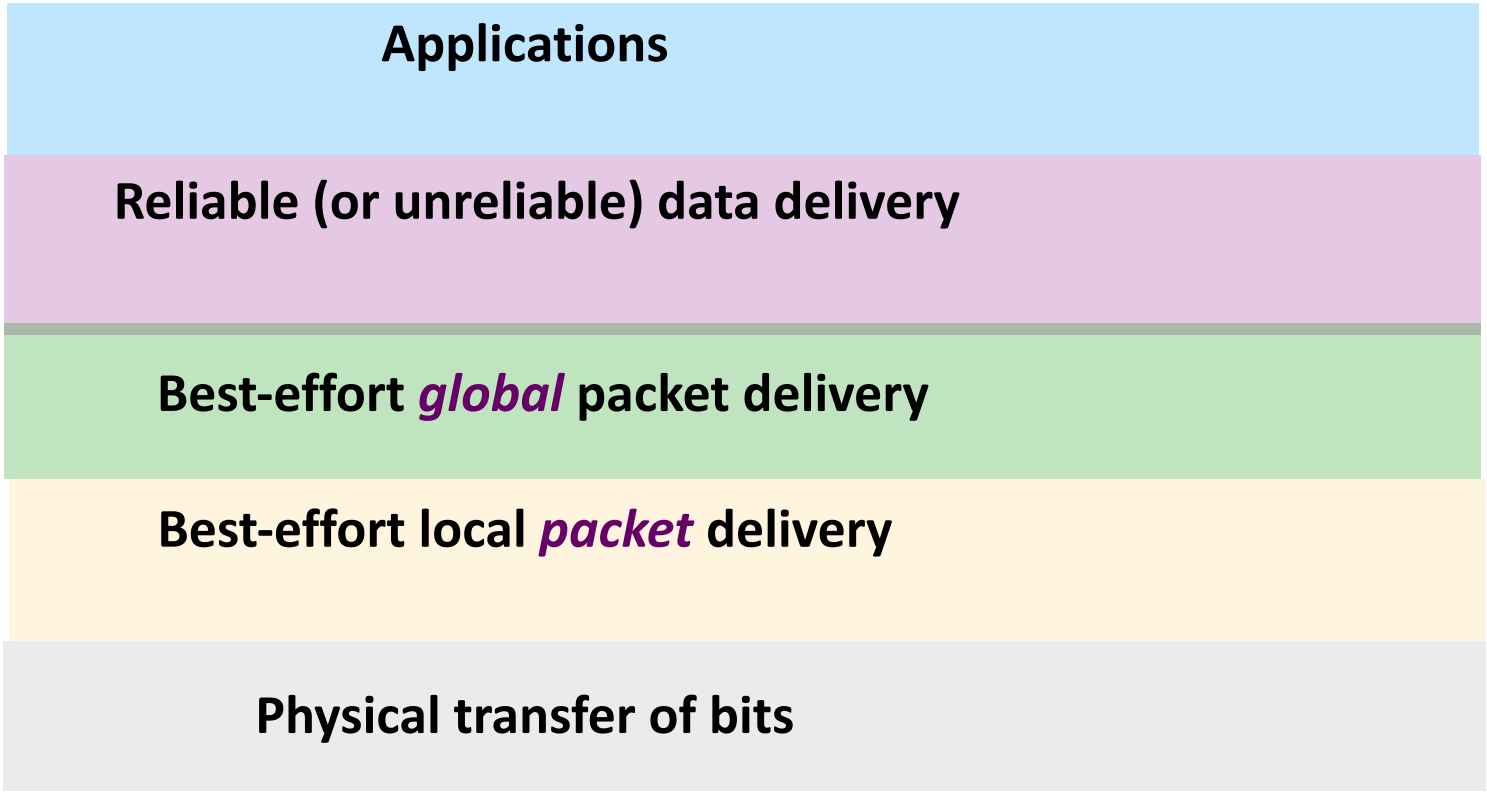
In the Internet: organization



In the Internet: organization



In the Internet: organization



A layered architecture

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

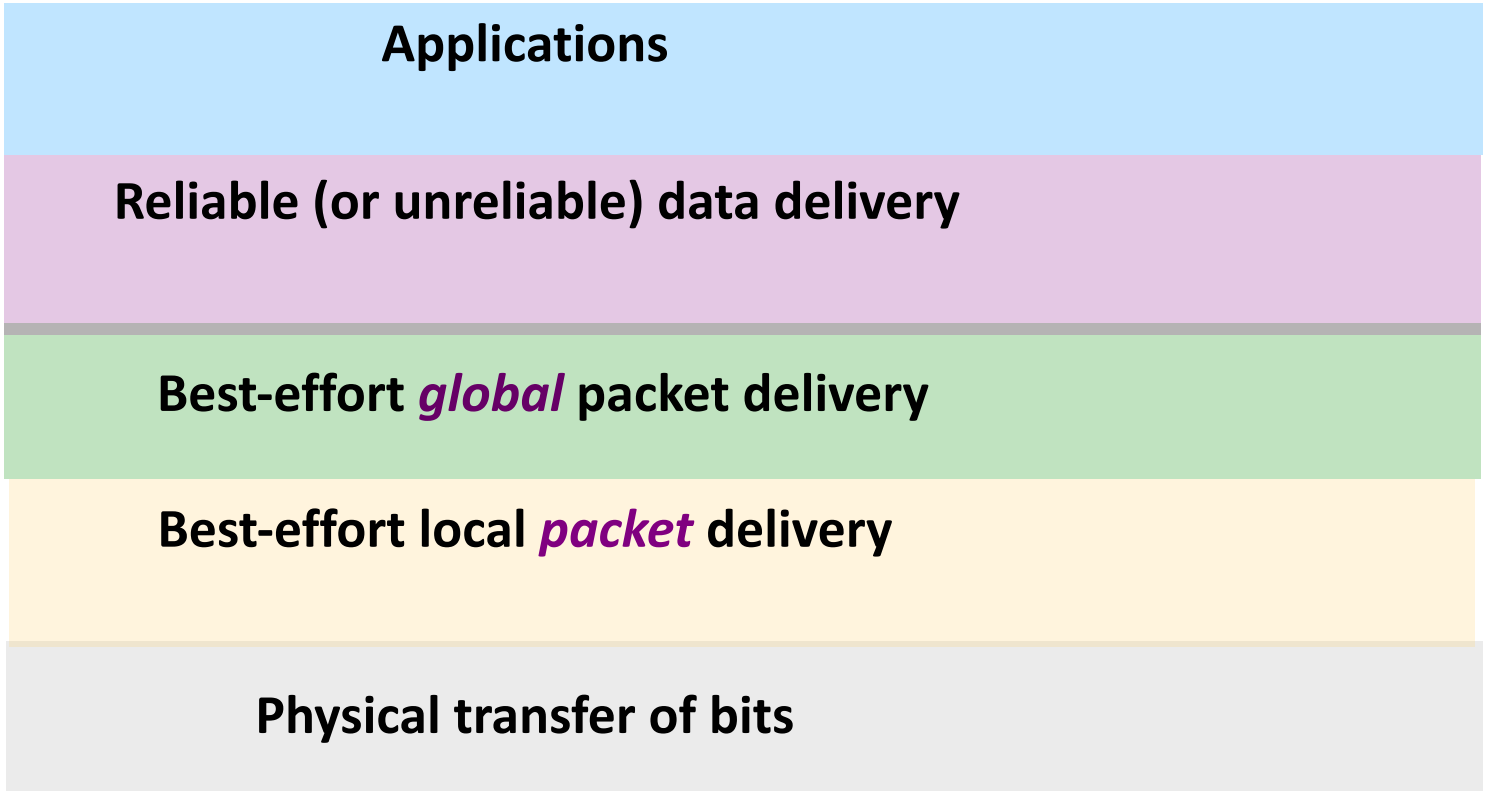
A layered architecture

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

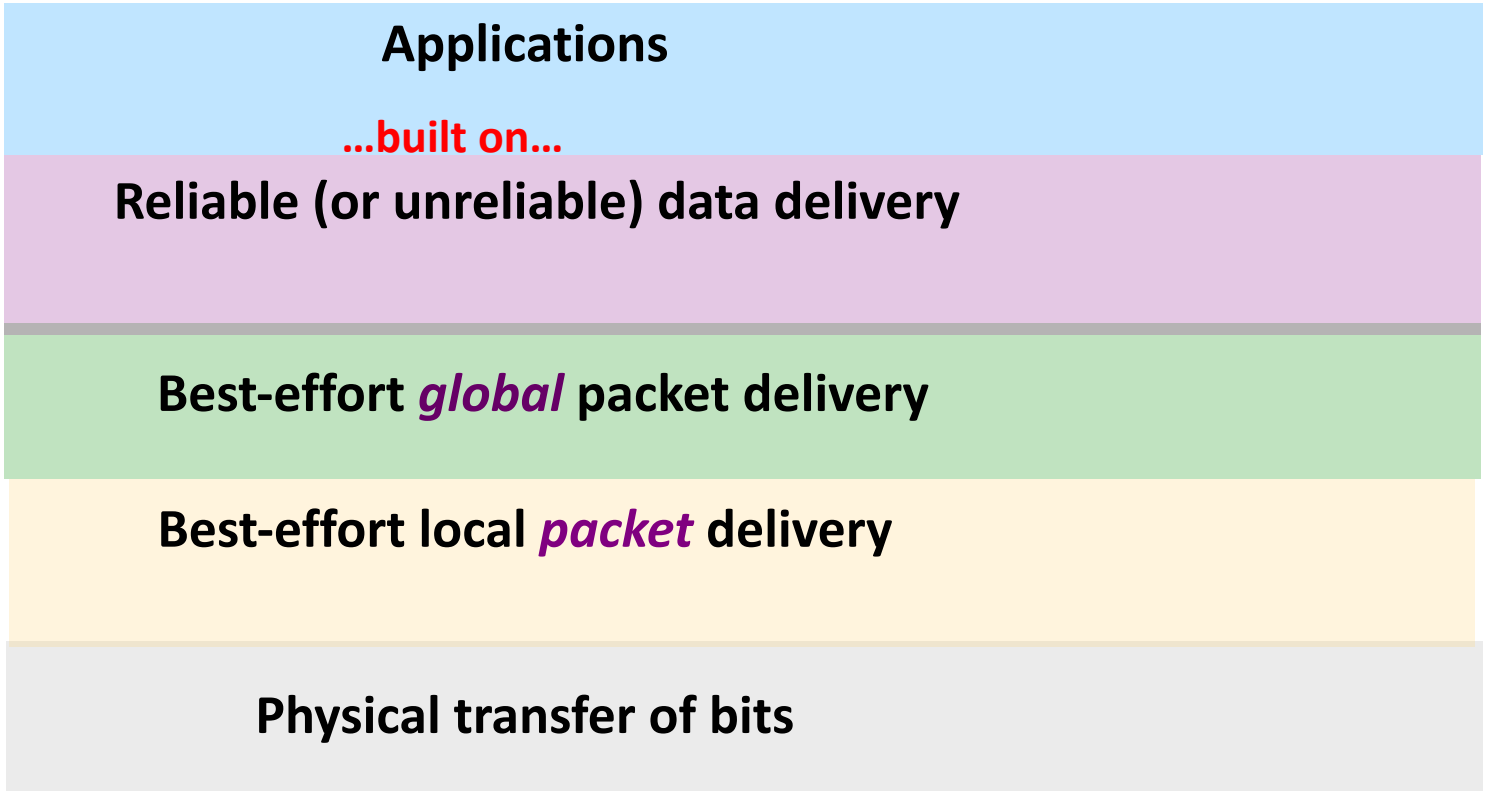
A layered architecture

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

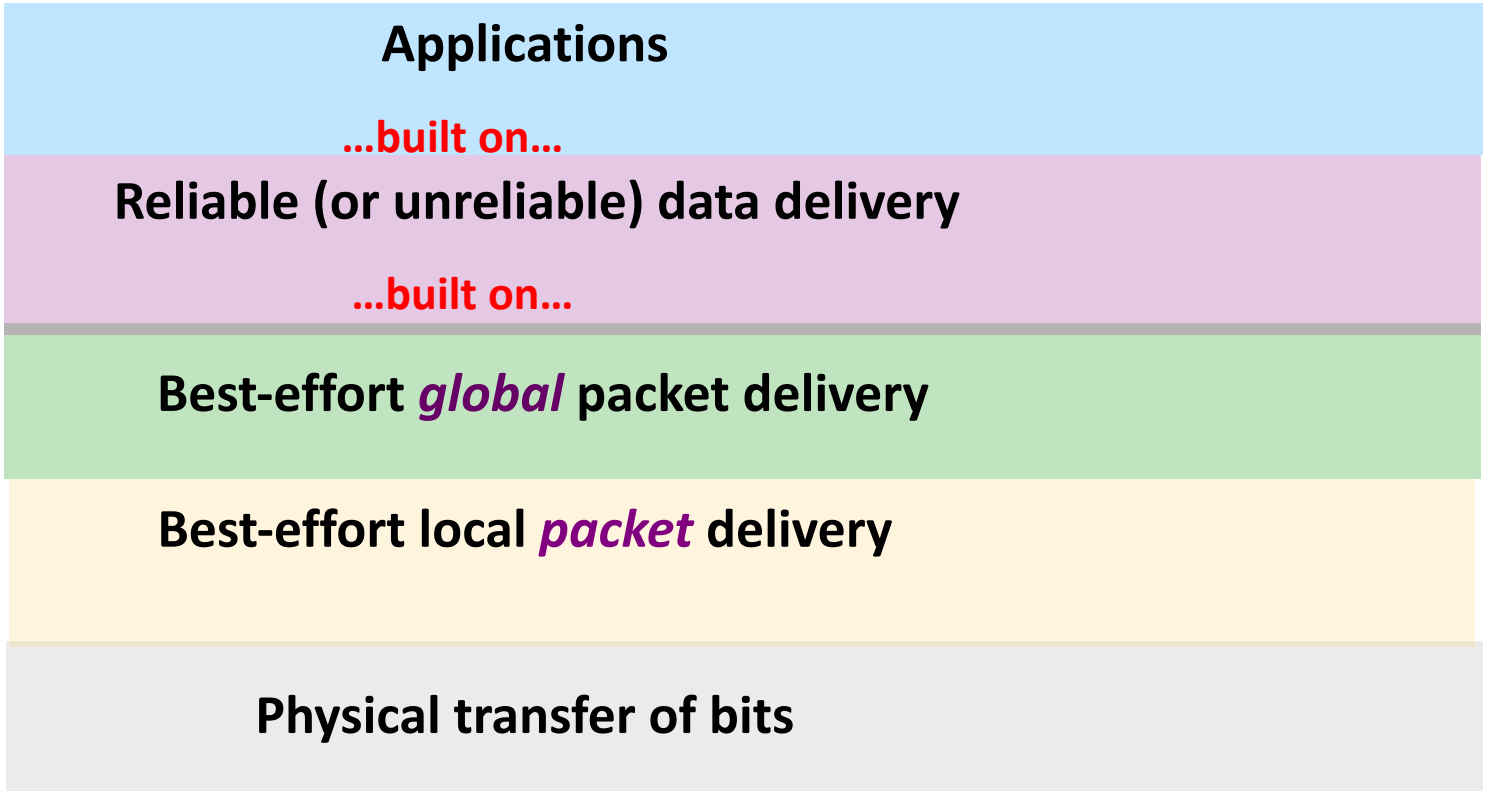
In the Internet: organization



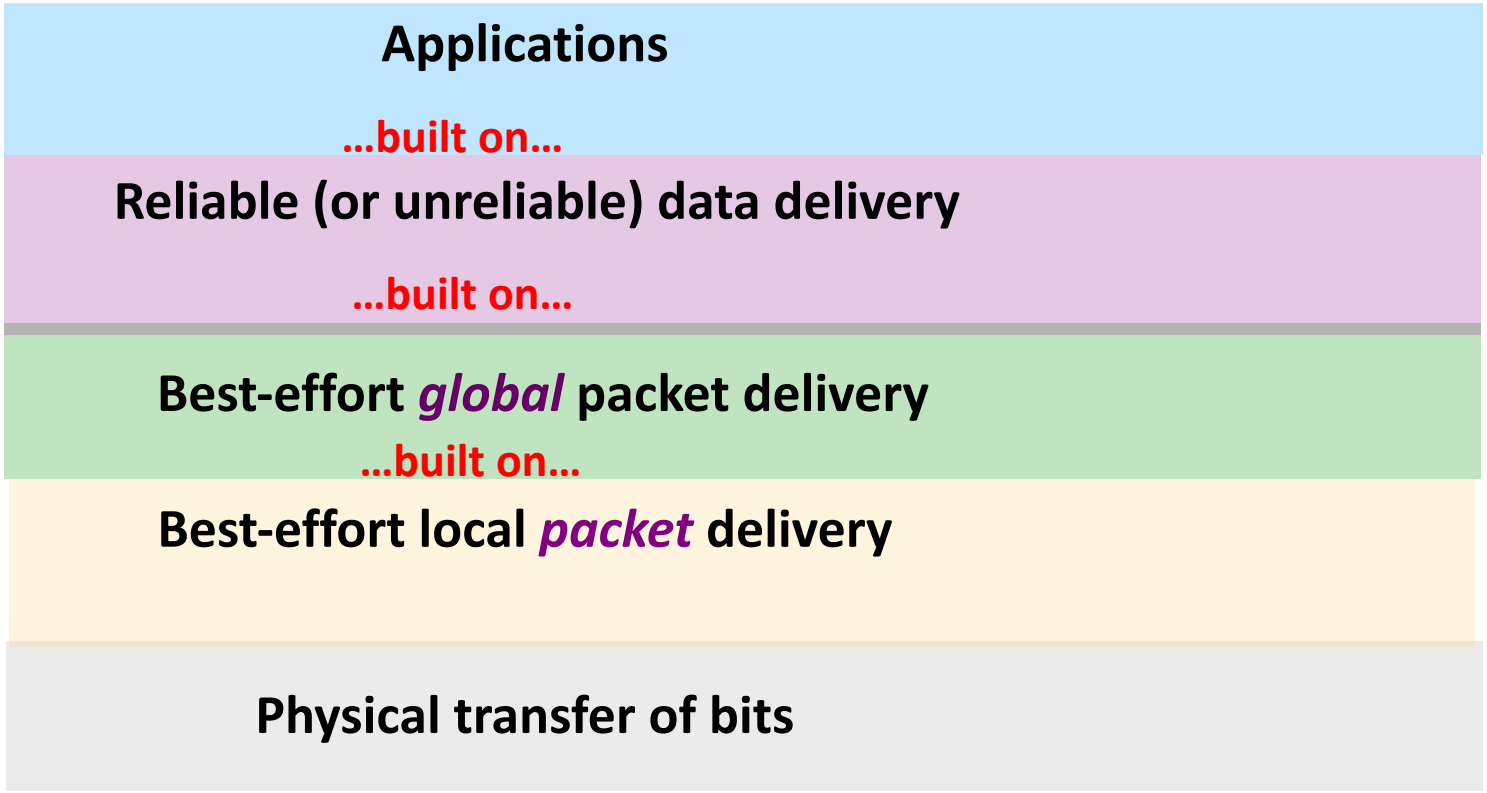
In the Internet: organization



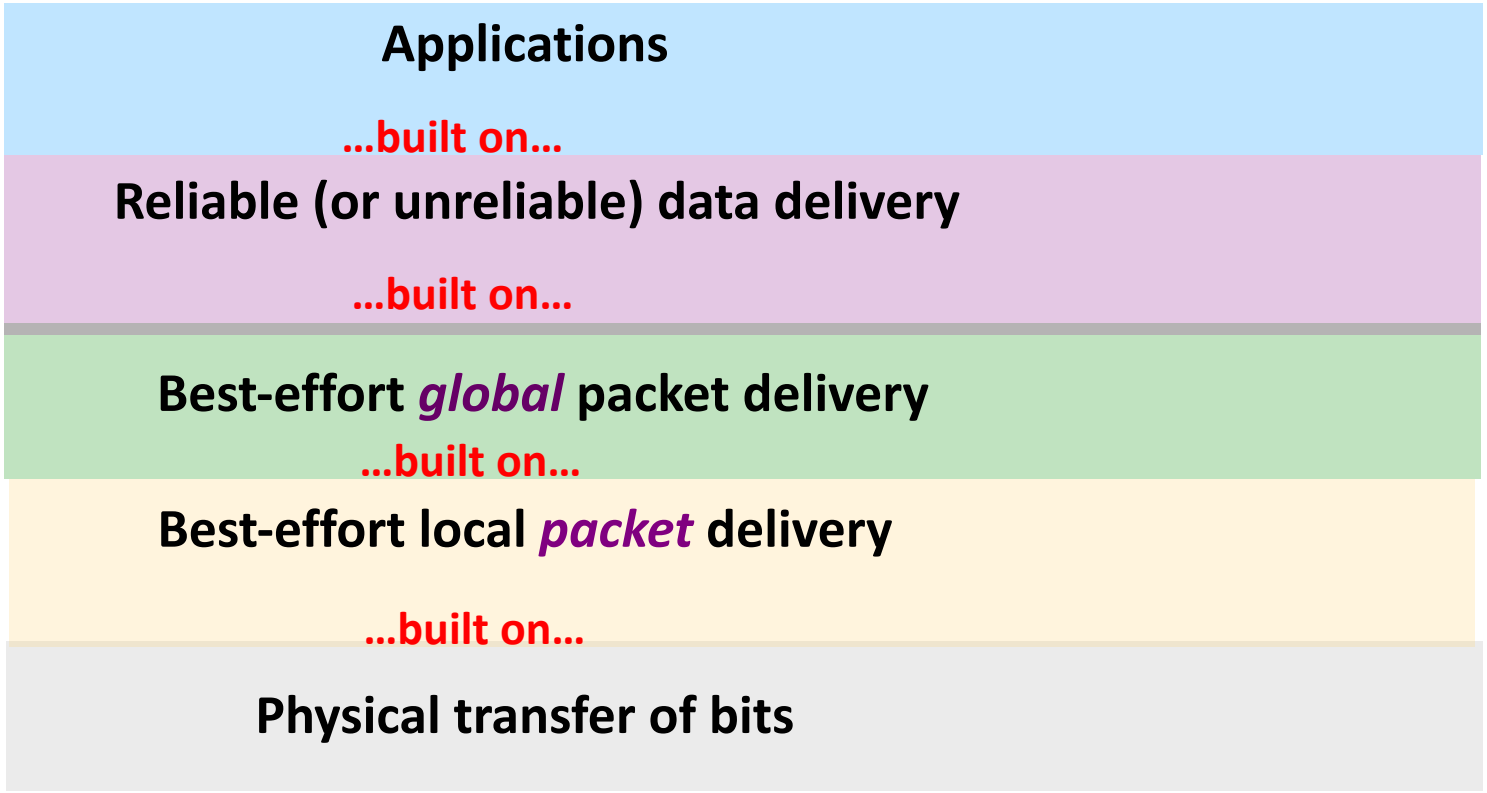
In the Internet: organization



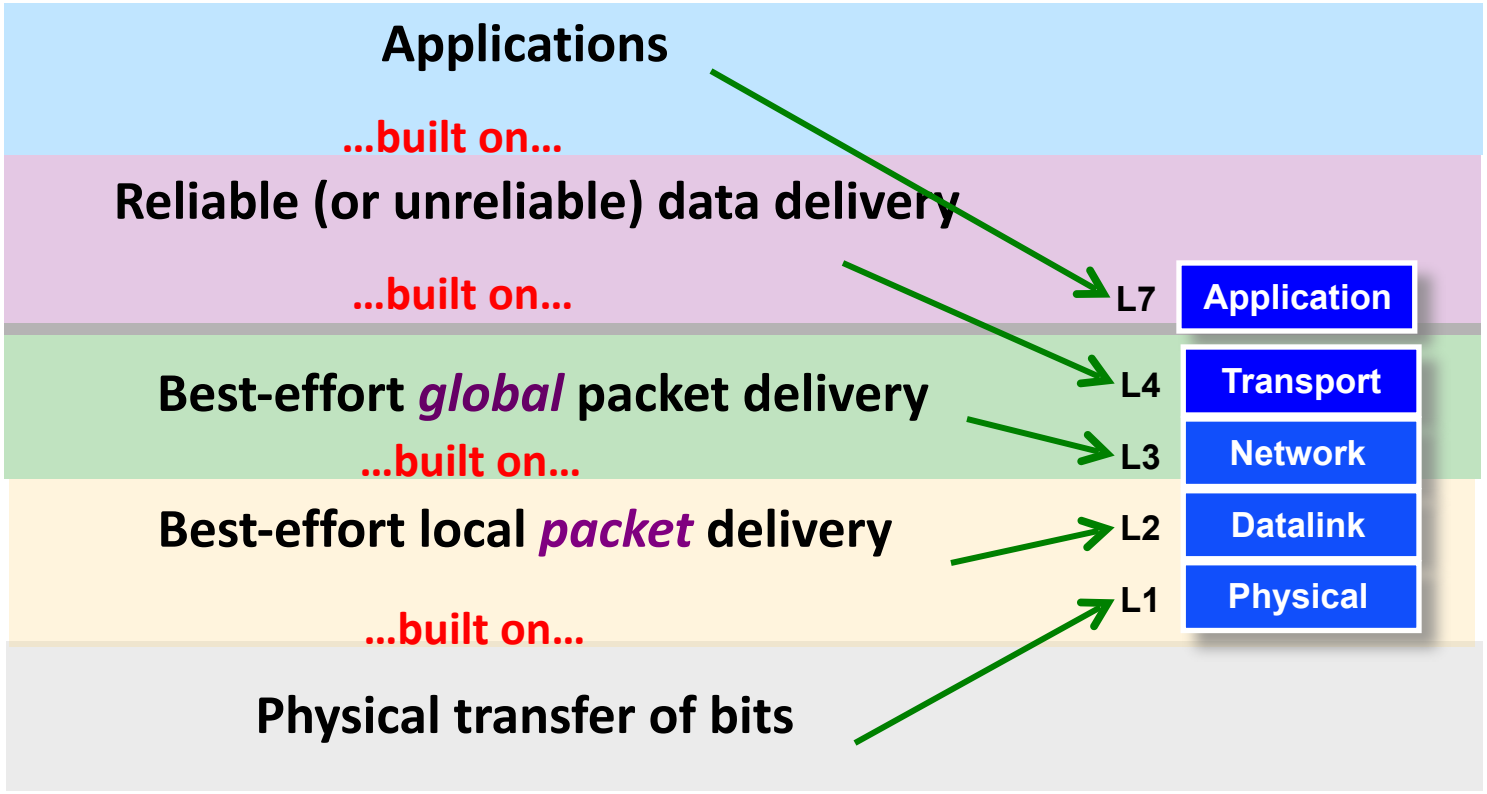
In the Internet: organization



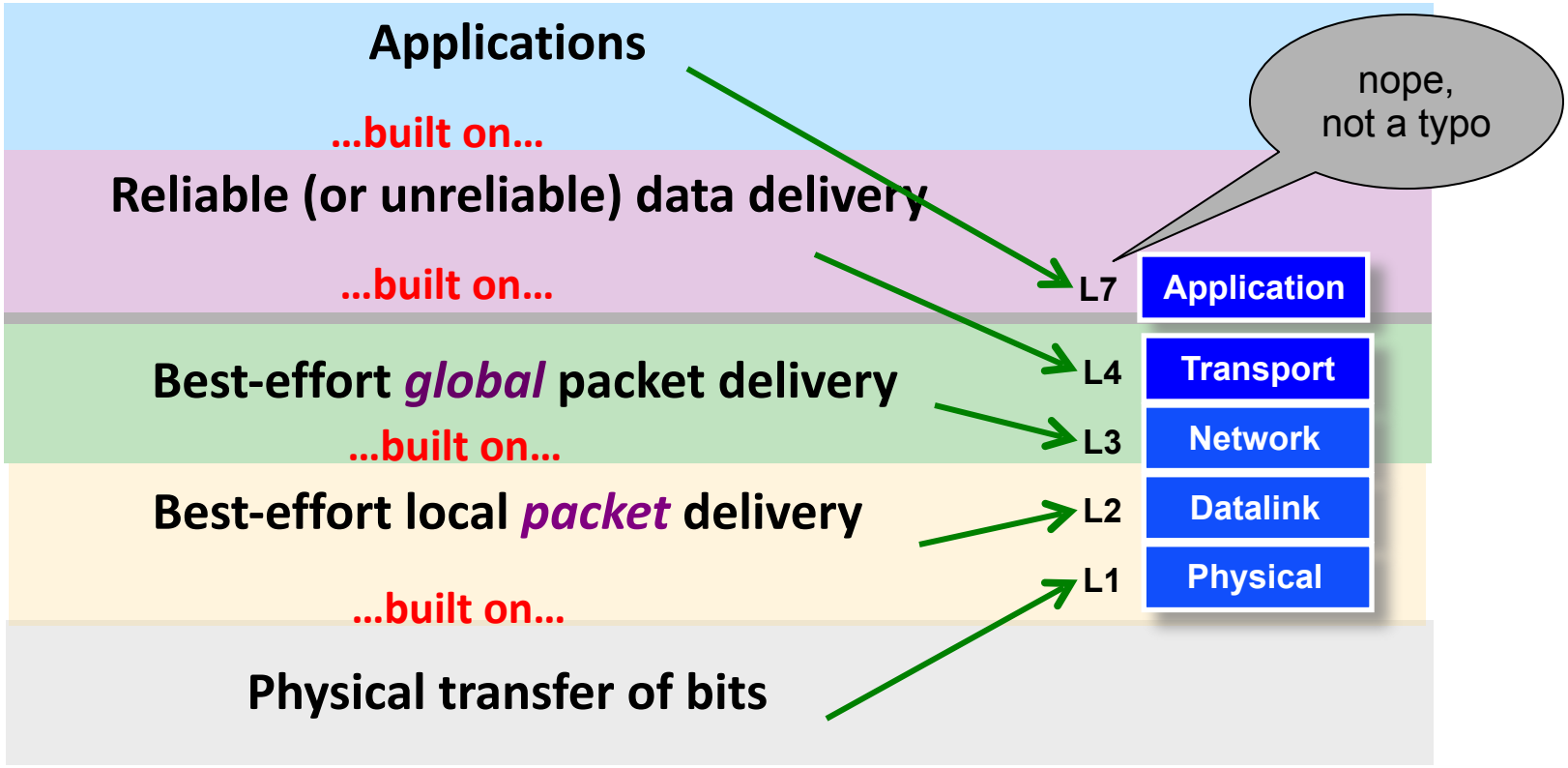
In the Internet: organization



In the Internet: organization

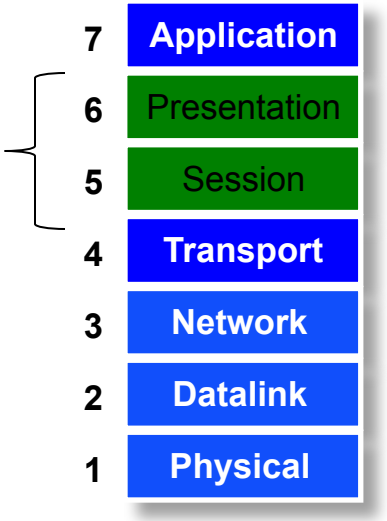


In the Internet: organization



Ancient history (late 1970s)

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

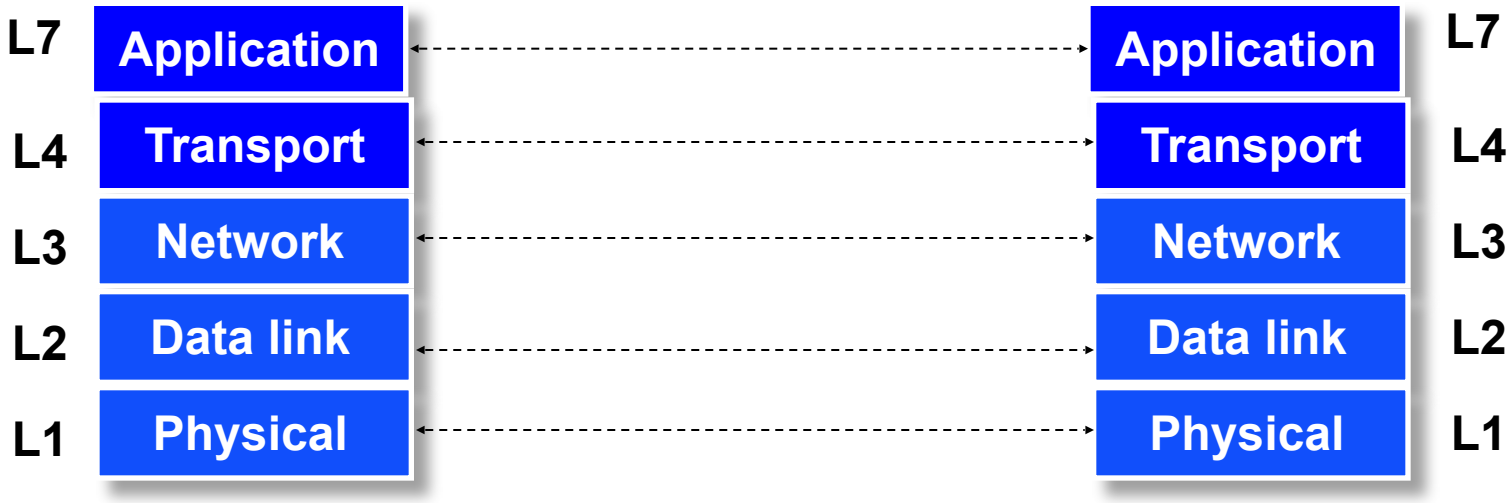


Questions?

Recall: peers understand the same things

CEO	Letter	CEO
Aide	Envelope	Aide
FedEx	Fedex Envelope (FE)	FedEx

Protocols and Layers



Communication between peer layers on different systems is defined by protocols

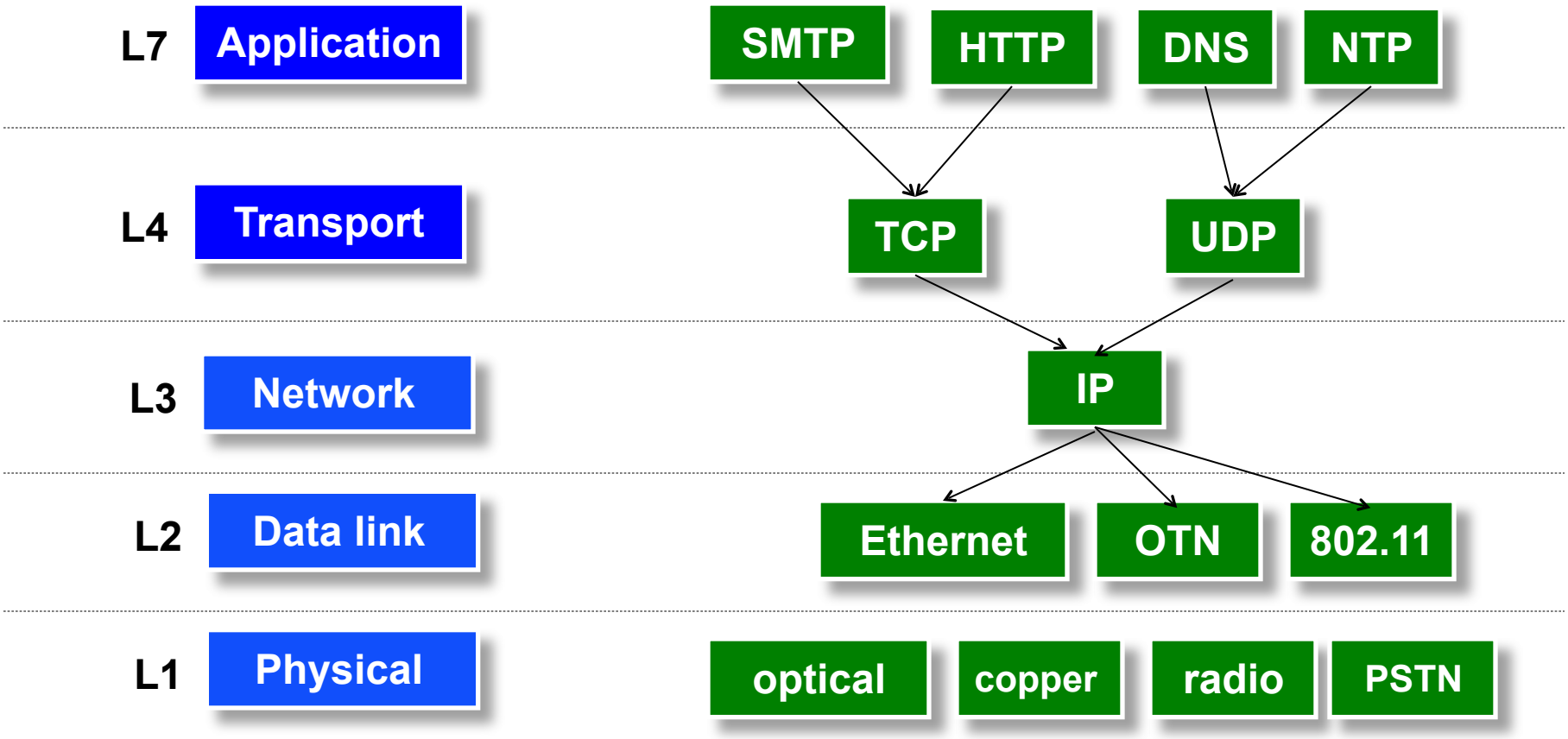
What is a Protocol?

- An agreement between parties on how to communicate
- Defines the syntax of communication
- And semantics
  - “first a hullo, then a request…”
  - essentially, a state machine
  - we’ll study many protocols later in the semester

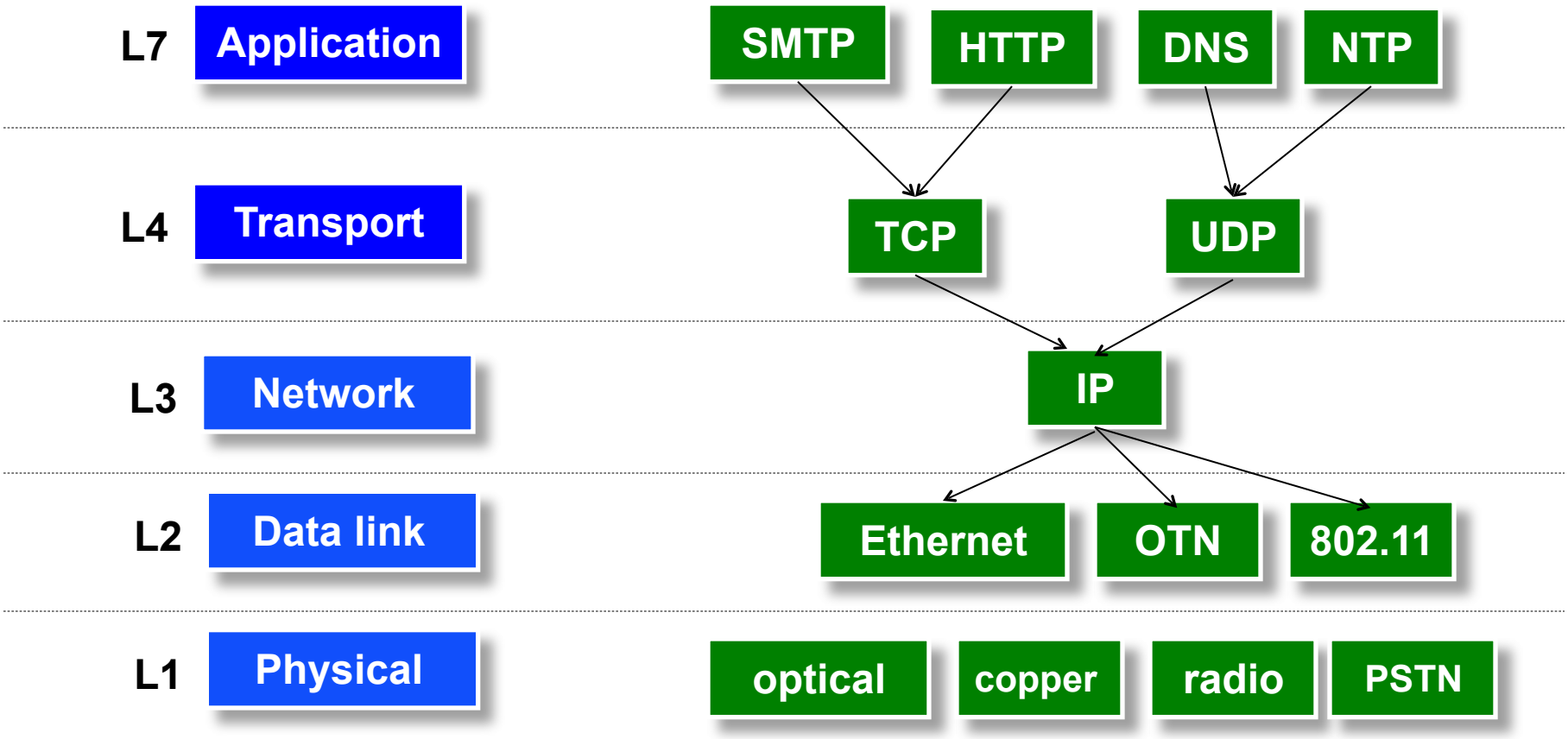
What is a Protocol?

- An agreement between parties on how to communicate
- Defines the syntax of communication
- And semantics
  - "first a hullo, then a request…"
  - essentially, a state machine
  - we'll study many protocols later in the semester
- Protocols exist at many layers
  - defined by a variety of standards bodies (IETF, IEEE, ITU)

Protocols at different layers

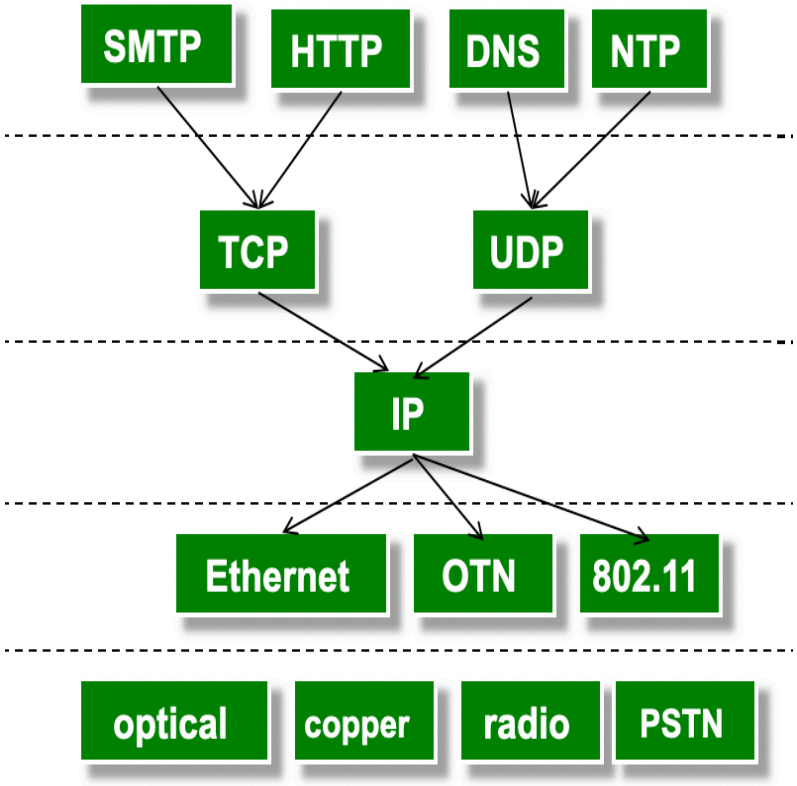


Protocols at different layers



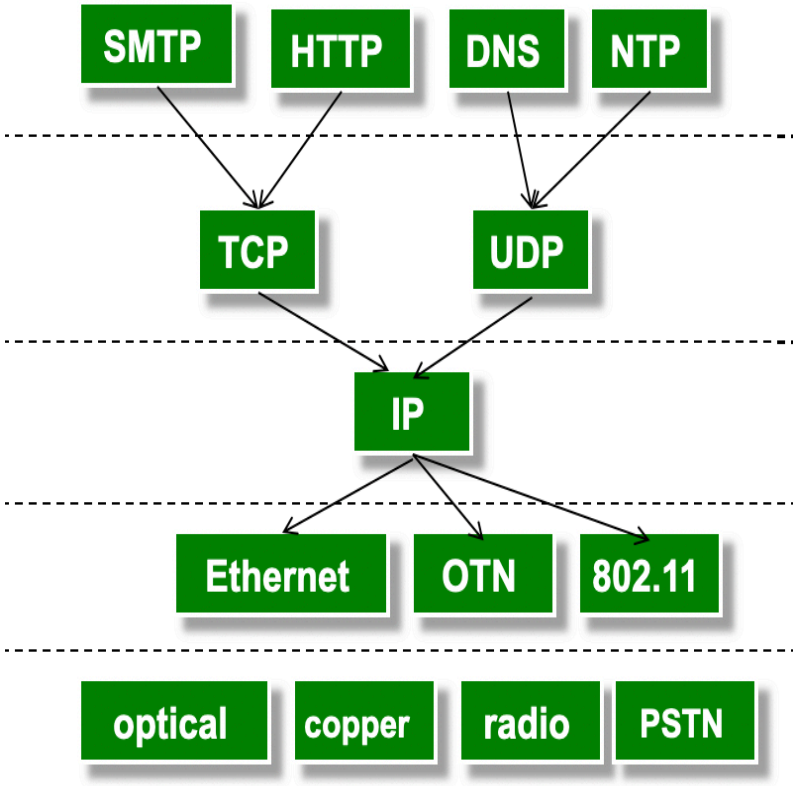
There is just one network-layer protocol!

Recap: Three important properties



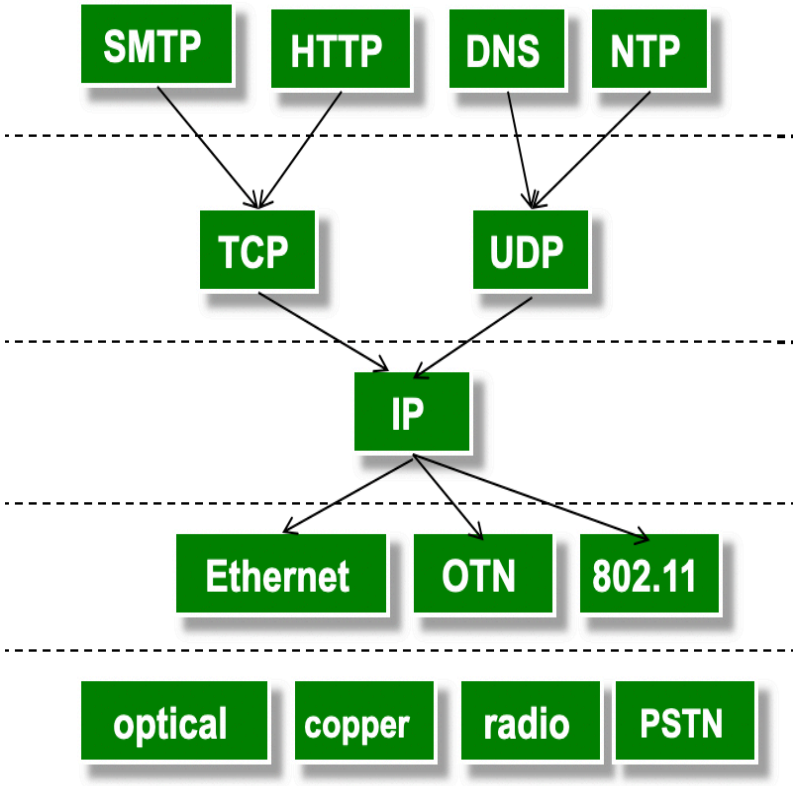
Recap: Three important properties

- Each layer:
  - Depends on layer below
  - Supports layer above
  - Independent of others



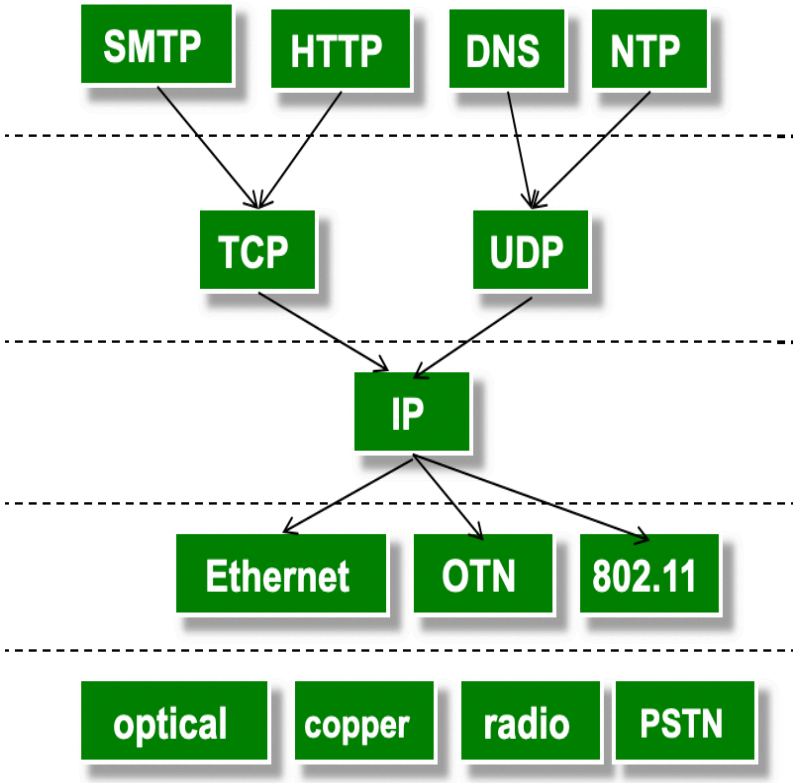
Recap: Three important properties

- Each layer:
  - Depends on layer below
  - Supports layer above
  - Independent of others
- Multiple versions in a layer
  - Interfaces differ somewhat
  - Components at one layer pick which lower-level protocol to use

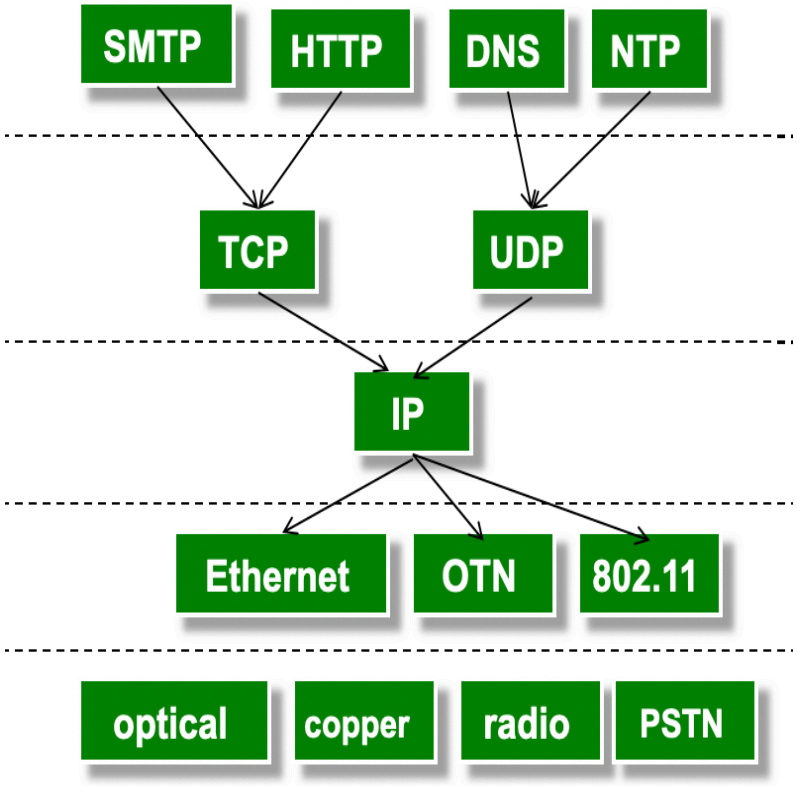


Recap: Three important properties

- Each layer:
  - Depends on layer below
  - Supports layer above
  - Independent of others
- Multiple versions in a layer
  - Interfaces differ somewhat
  - Components at one layer pick which lower-level protocol to use
- But only one IP layer
  - Unifying protocol

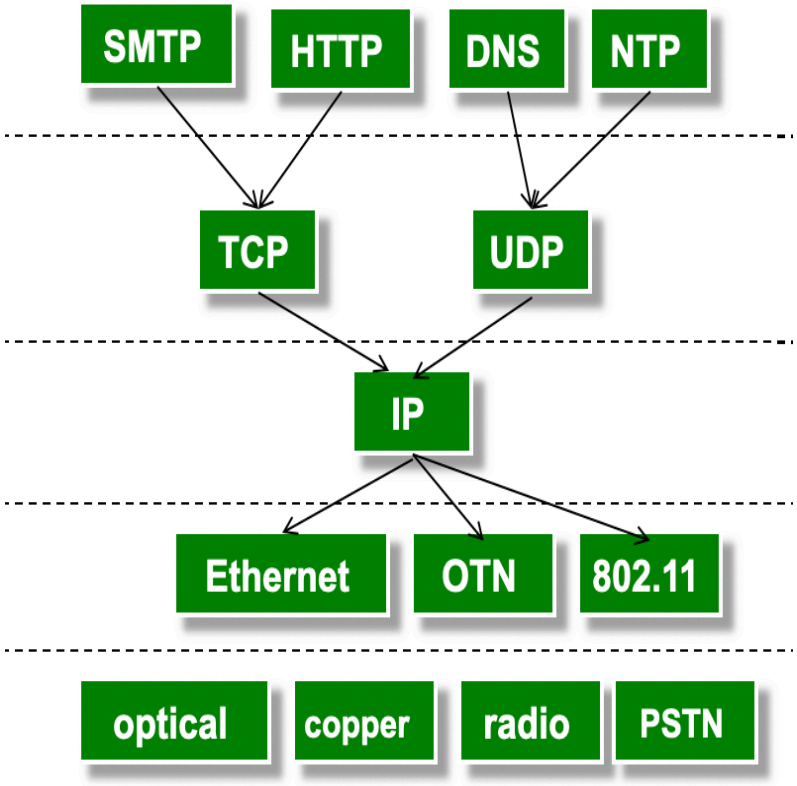


Why is layering important?



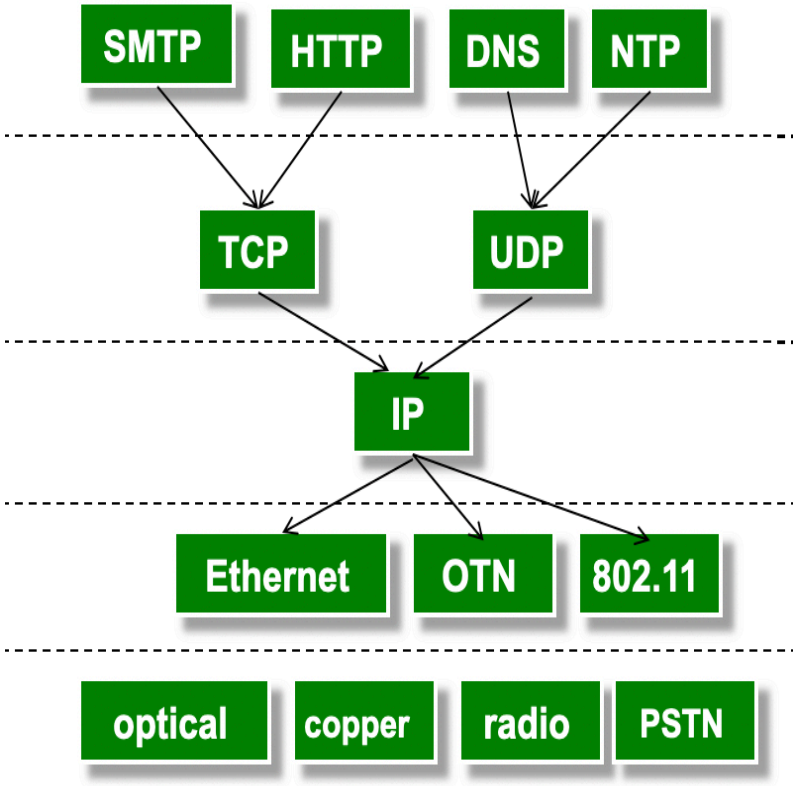
Why is layering important?

- Innovation can proceed largely in parallel!



Why is layering important?

- Innovation can proceed largely in parallel!
- Pursued by very different communities
  - App devs (L7), chip designers (L1/L2)



Questions?

How do you solve a problem?

1. Decompose it (into tasks and abstractions)

2. Assign tasks to entities (who does what)

Distributing Layers Across Network

Distributing Layers Across Network

- Layers are simple if only on a single machine
  - Just stack of modules interacting with those above/below

Distributing Layers Across Network

- Layers are simple if only on a single machine
  - Just stack of modules interacting with those above/below
- But we need to implement layers across:
  - Hosts
  - Routers (switches)

Distributing Layers Across Network

- Layers are simple if only on a single machine
  - Just stack of modules interacting with those above/below
- But we need to implement layers across:
  - Hosts
  - Routers (switches)
- What gets implemented where?

What gets implemented at the end host?

What gets implemented at the end host?

- Bits arrive on wire ... → must implement L1

What gets implemented at the end host?

- Bits arrive on wire ... → must implement L1
- ... must make it up to app → must implement L7

What gets implemented at the end host?

- Bits arrive on wire ... → must implement L1
- ... must make it up to app → must implement L7

- Therefore, all layers must exist at host!

What gets implemented in the network?

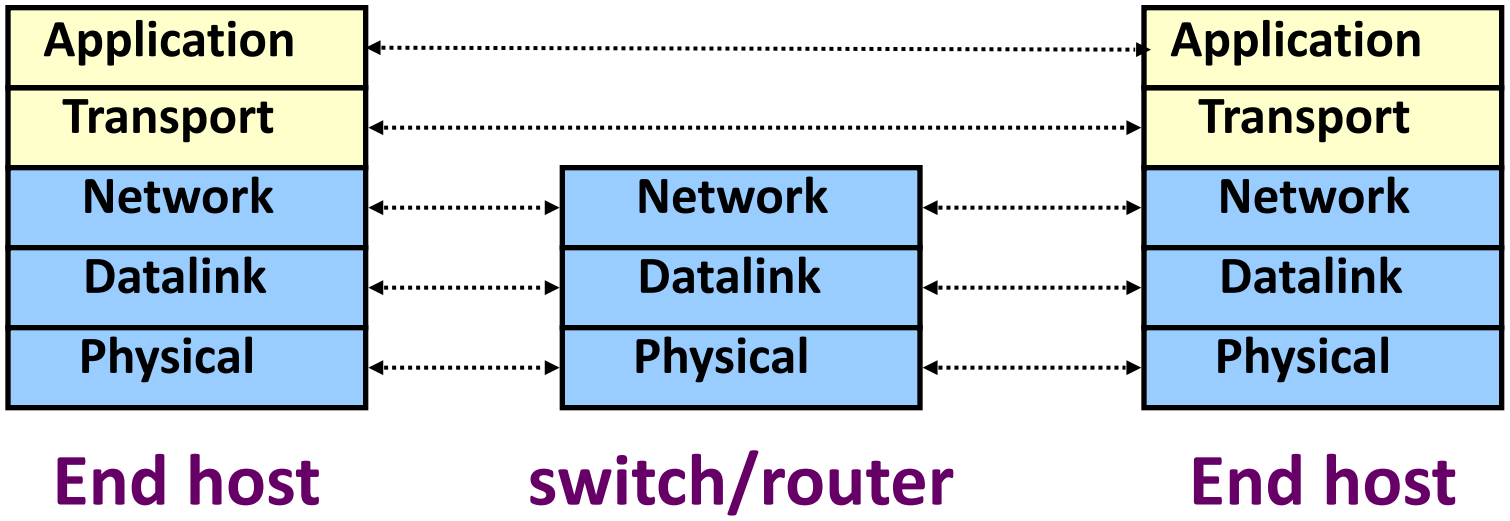
- Bits arrive on wire → physical layer (L1)
- Packets must be delivered across links and local networks → datalink layer (L2)
- Packets must be delivered between networks for global delivery → network layer (L3)

What gets implemented in the network?

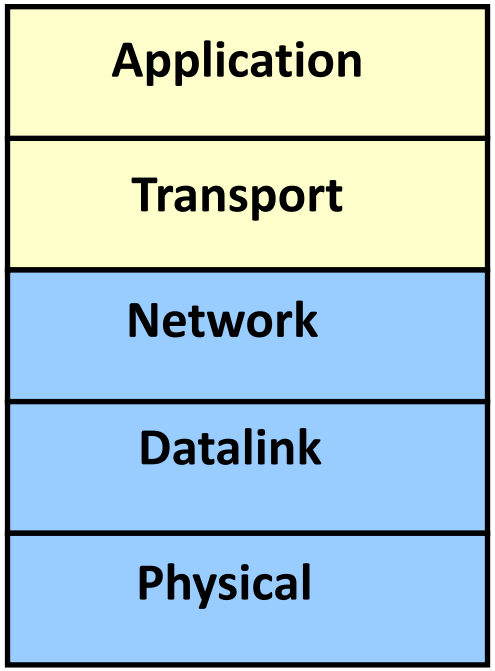
- Bits arrive on wire → physical layer (L1)
- Packets must be delivered across links and local networks → datalink layer (L2)
- Packets must be delivered between networks for global delivery → network layer (L3)
- The network does not support reliable delivery
  - Transport layer (and above) ***not***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

Simple Diagram

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

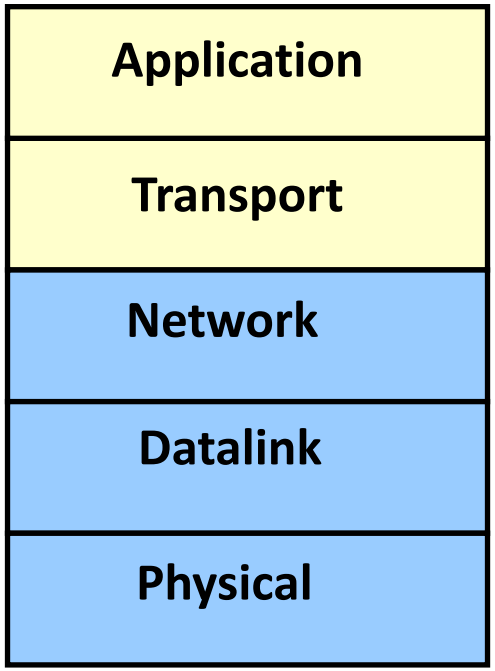


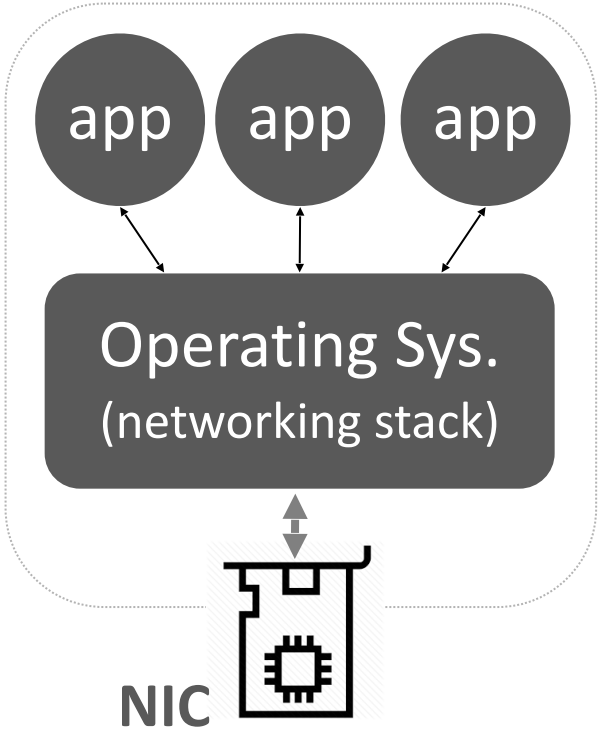
A closer look: end host



end host

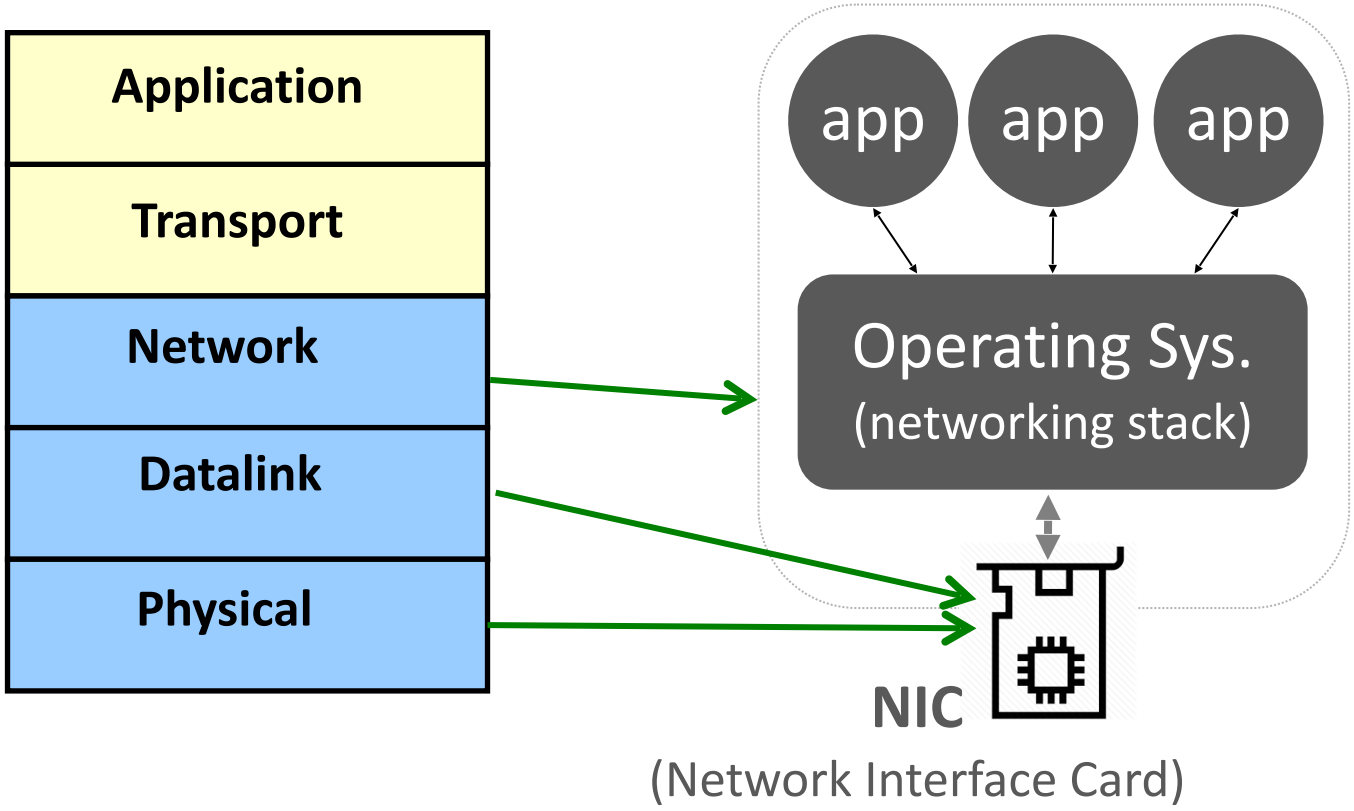
A closer look: end host



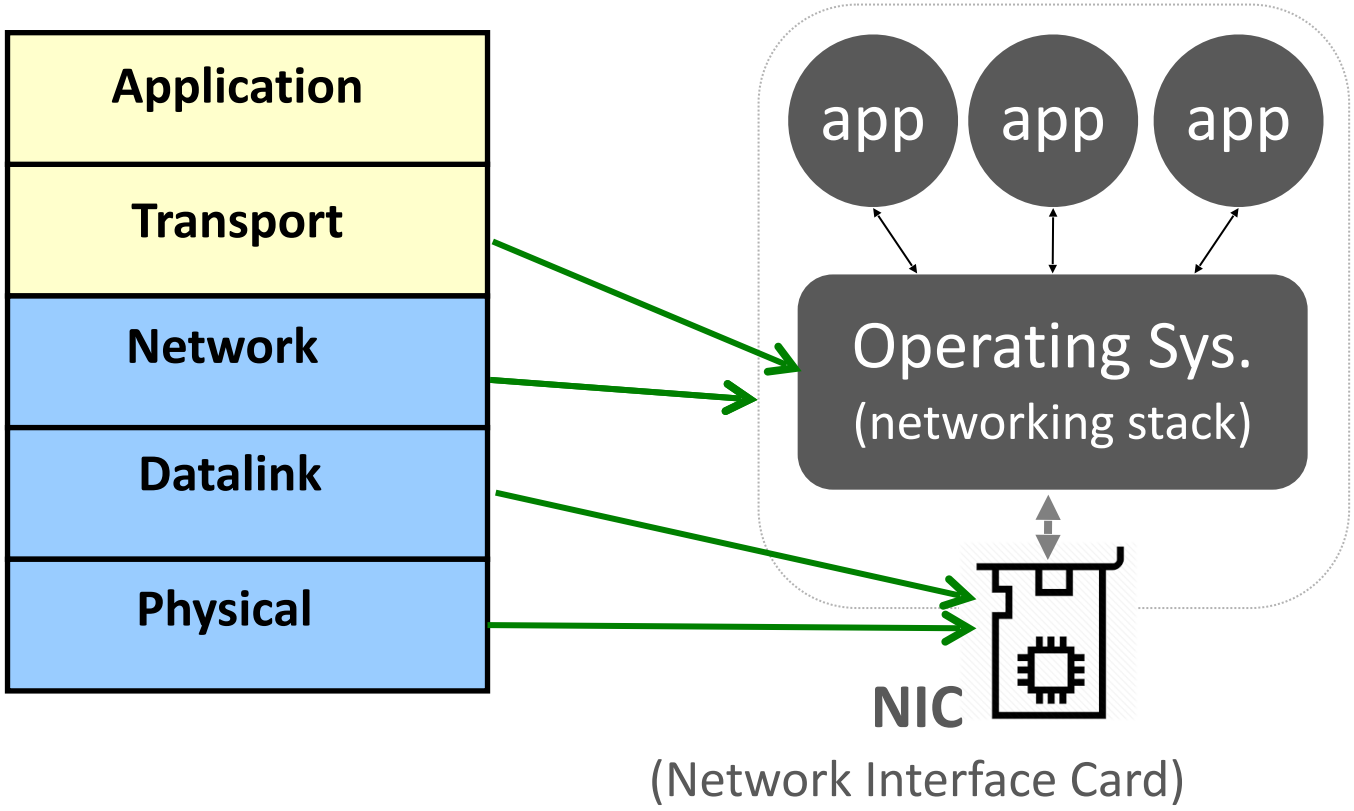


(Network Interface Card)

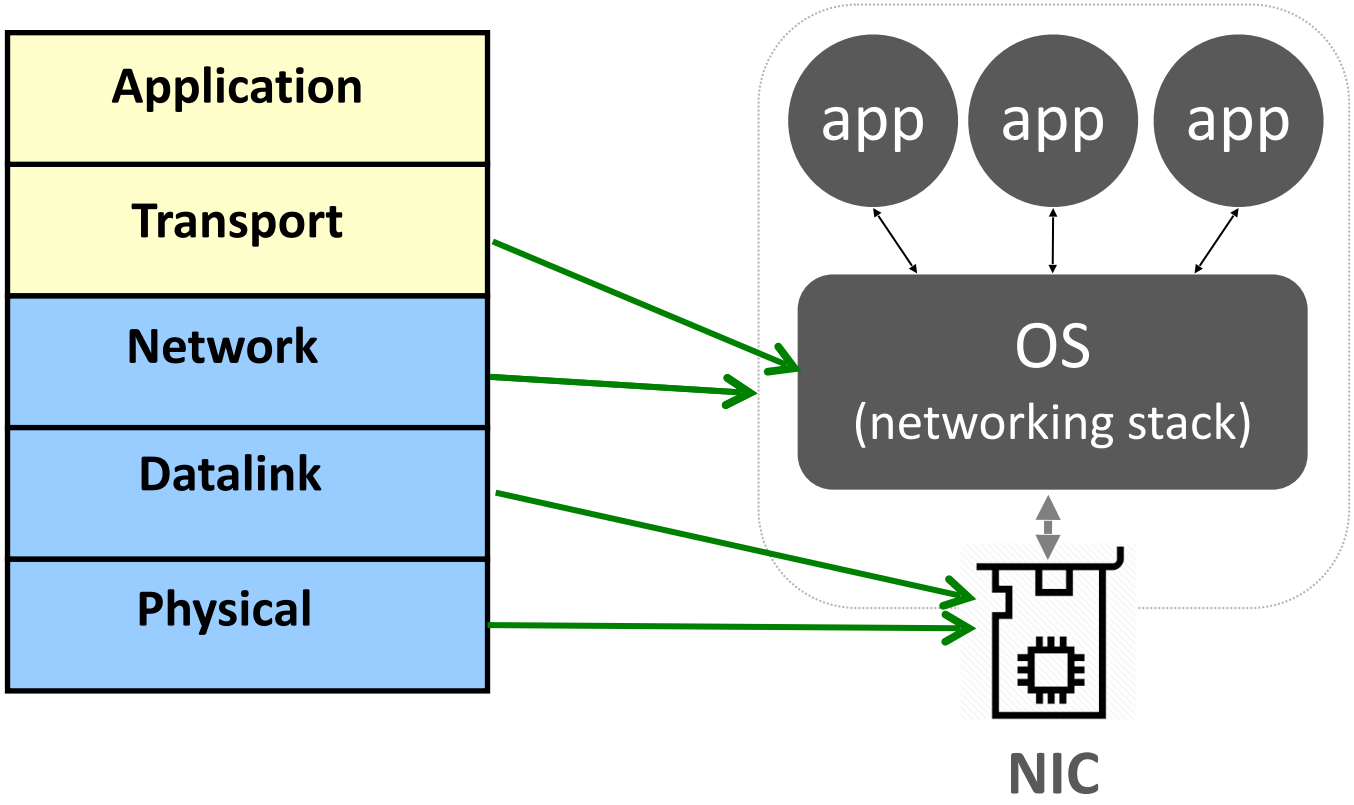
A closer look: end host



A closer look: end host

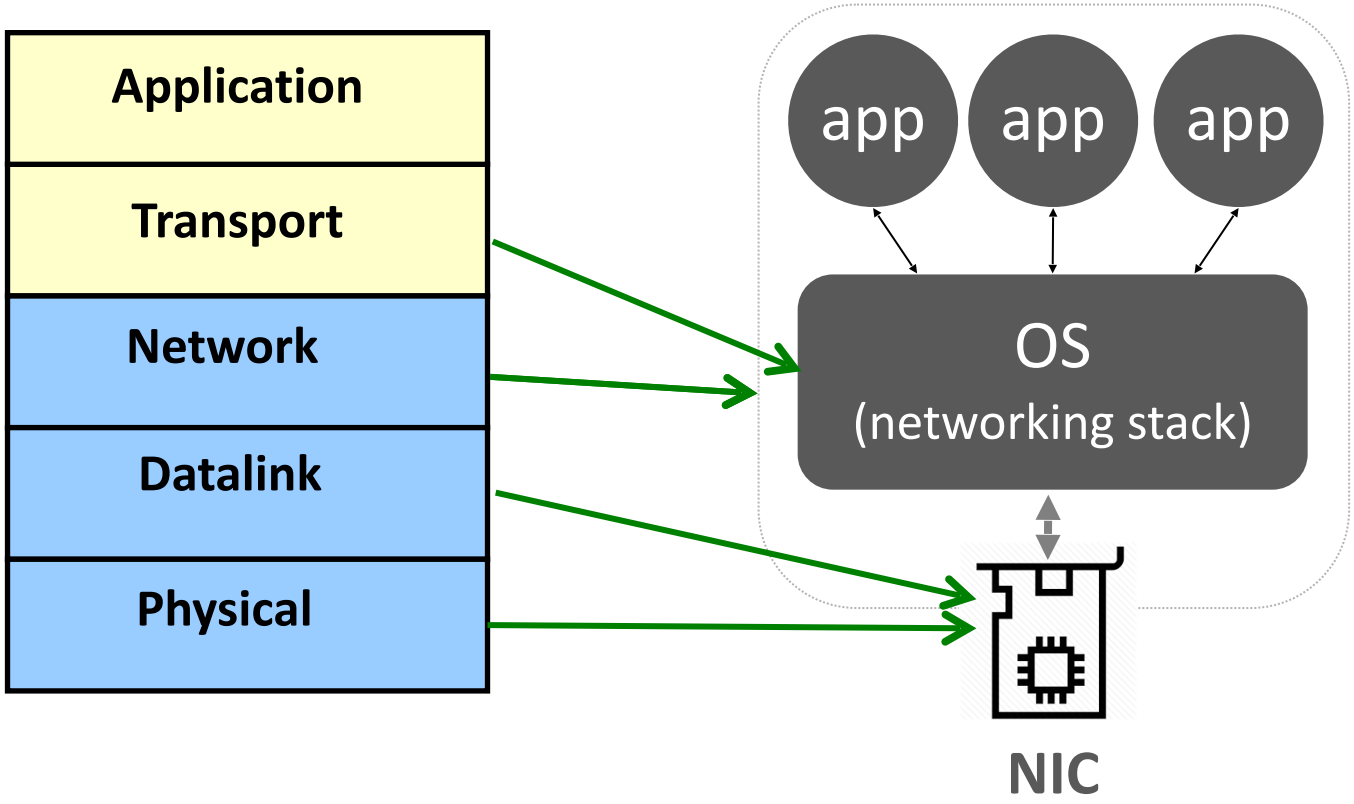


Note: addressing *within* the end host



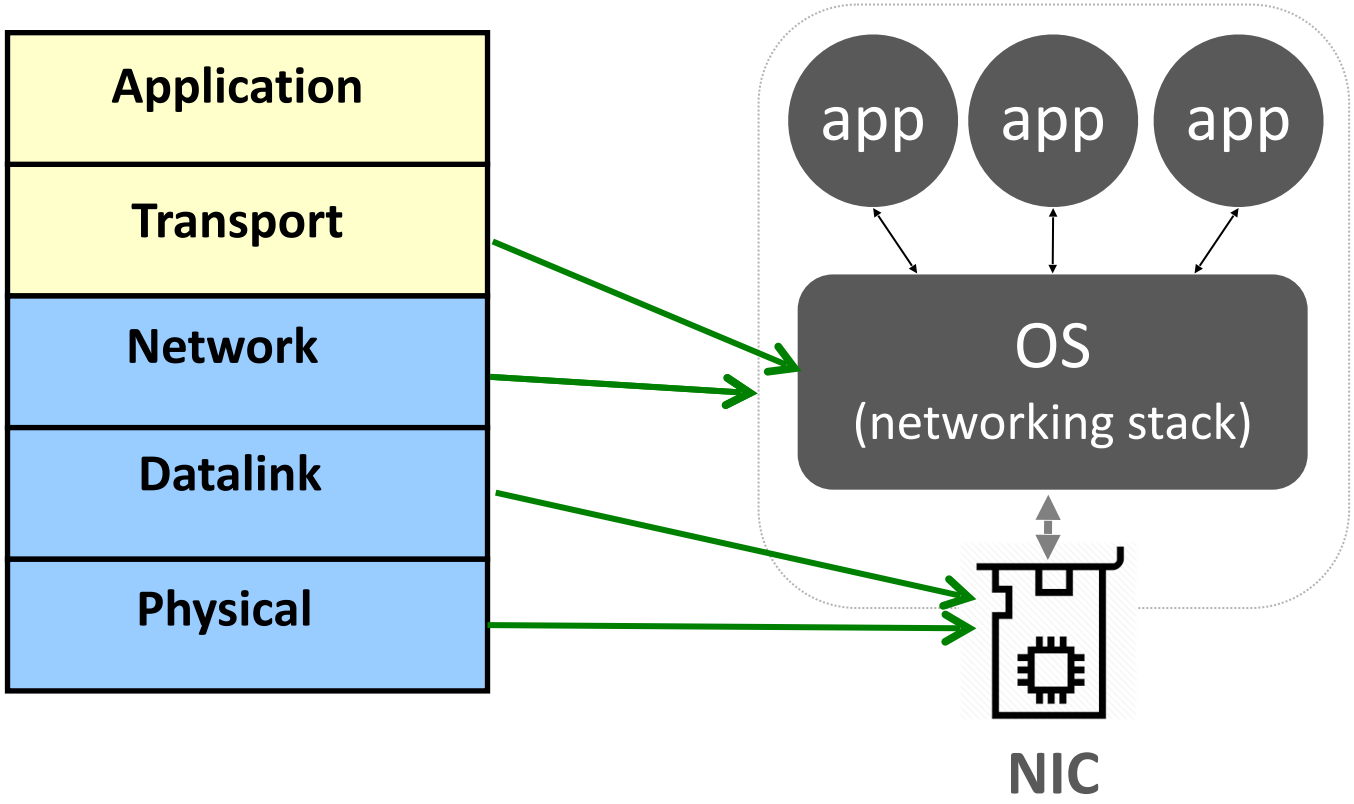
Note: addressing *within* the end host

Recall: packet contains the destination host's address



Note: addressing *within* the end host

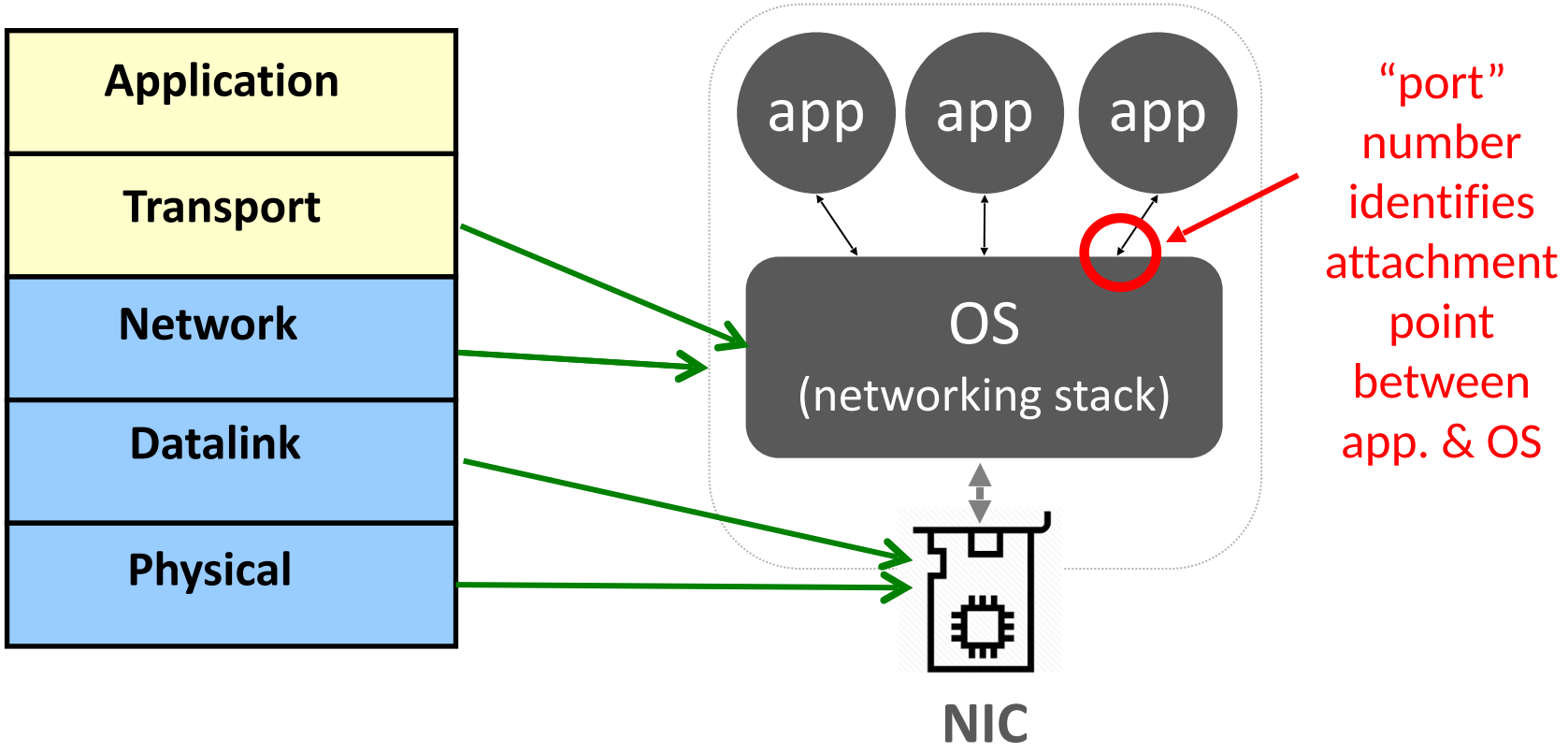
Recall: packet contains the destination host's address



When a packet arrives at the host, how does the OS know which app to send the packet to?

Note: addressing *within* the end host

Recall: packet contains the destination host's address

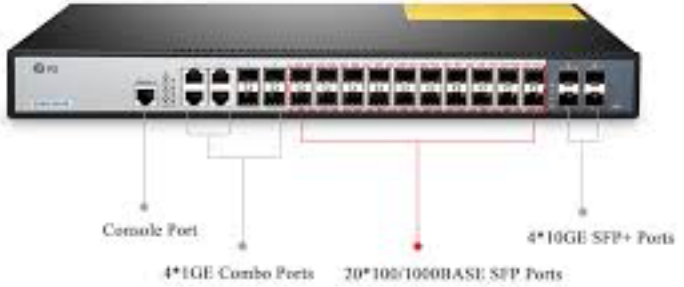


When a packet arrives at the host, how does the OS know which app to send the packet to?

Network “ports”: two types

Network “ports”: two types

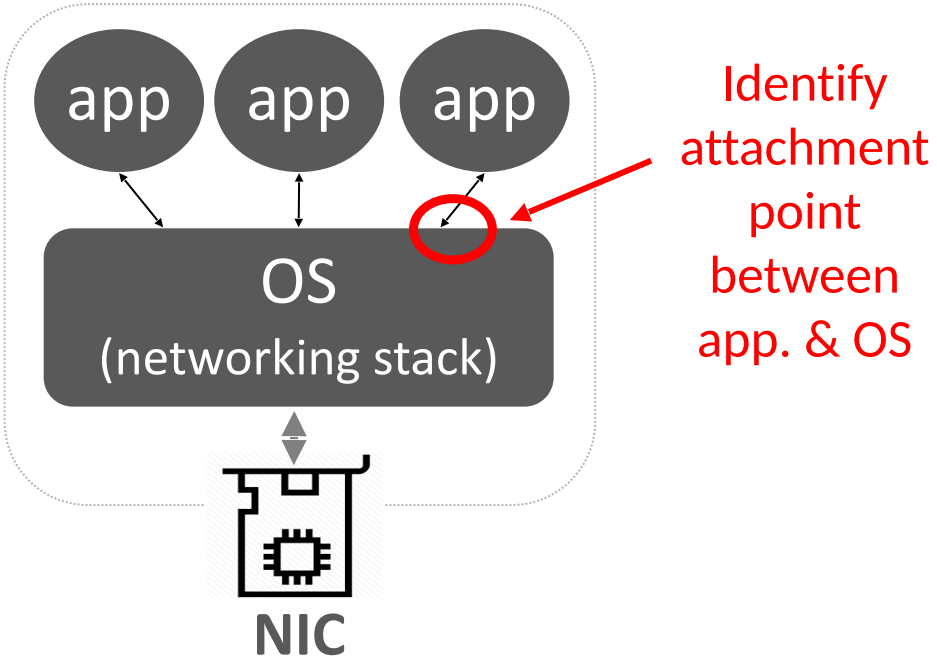
- Switches/routers have **physical ports**:
  - Places where links connect to switches



Network “ports”: two types

- Switches/routers have **physical ports**:
  - Places where links connect to switches

- The OS supports **logical ports**:
  - Place where app connects to OS network stack



Of Sockets and Ports

Section will cover sockets in detail

Of Sockets and Ports

- **Socket:** an OS mechanism that connects app processes to the networking stack

Section will cover sockets in detail

Of Sockets and Ports

- **Socket:** an OS mechanism that connects app processes to the networking stack
- When an app wants access to the network, it opens a **socket**, which is associated with a **port**
  - *This is not a physical port, just a logical one*

Section will cover sockets in detail

Of Sockets and Ports

- **Socket:** an OS mechanism that connects app processes to the networking stack
- When an app wants access to the network, it opens a **socket**, which is associated with a **port**
  - *This is not a physical port, just a logical one*
- The **port number** is used by the OS to direct incoming packets to its associated socket

Section will cover sockets in detail

Implications for Packet Header

- Packet header must include:
  - Destination host address (used by network to reach host)

Implications for Packet Header

- Packet header must include:
  - Destination host address (used by network to reach host)
  - Destination port (used by host OS to reach app) [new!]

Implications for Packet Header

- Packet header must include:
  - Destination host address (used by network to reach host)
  - Destination port (used by host OS to reach app) [new!]

- When a packet arrives at the destination end-host, it is delivered to the socket (process) associated with the packet's destination port

OS Network Stack Is An Intermediary

OS Network Stack Is An Intermediary

- Application has very clear task (w.r.t. network)
  - Thinks about data

OS Network Stack Is An Intermediary

- Application has very clear task (w.r.t. network)
  - Thinks about data
- NIC/driver has very clear task
  - Thinks about packets

OS Network Stack Is An Intermediary

- Application has very clear task (w.r.t. network)
  - Thinks about data
- NIC/driver has very clear task
  - Thinks about packets

OS Network Stack Is An Intermediary

- Application has very clear task (w.r.t. network)
  - Thinks about data
- NIC/driver has very clear task
  - Thinks about packets
- Network stack in the intermediary between them
  - Translates between their abstractions

Recap: layers at the end host

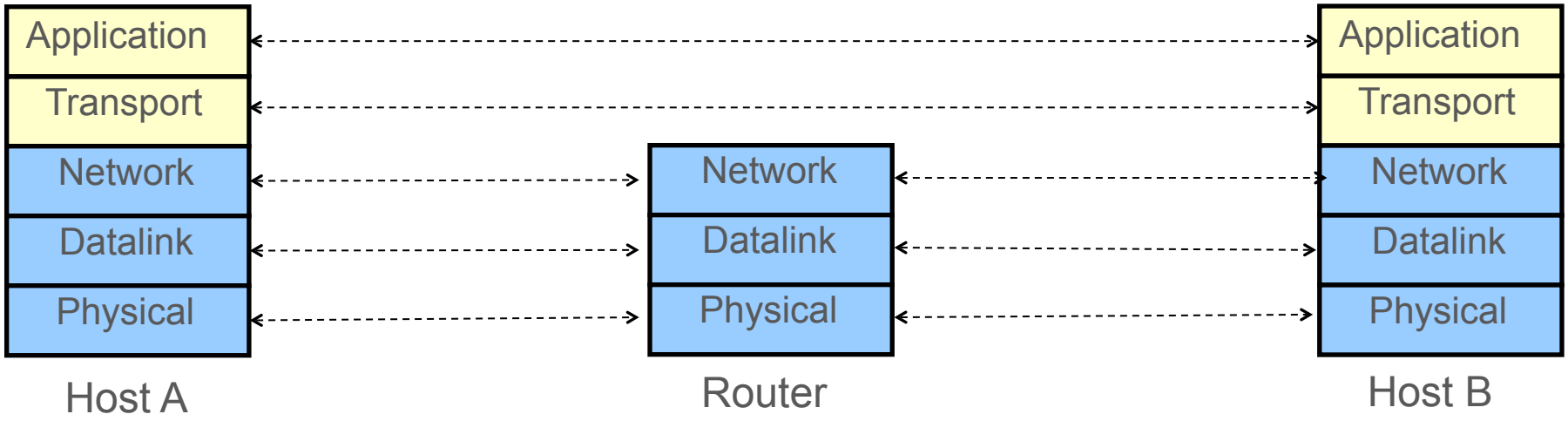
- **Application layer (L7)**
  - part of the app: browser, mail client,…
- **Transport and network layer (L3, L4)**
  - typically part of the OS
- **Datalink and physical layer (L1, L2)**
  - hardware/firmware/drivers

A closer look: network

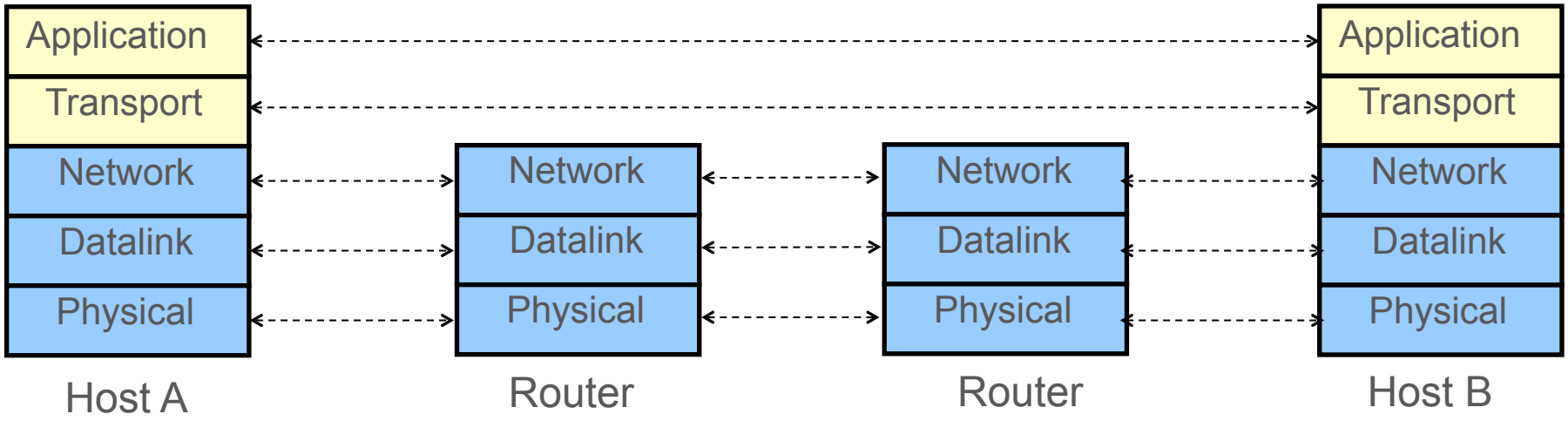
- Bits on wire → physical layer (L1)
- Local delivery of packets → datalink layer (L2)
- Global delivery of packets → network layer (L3)

Recall: Logical Communication

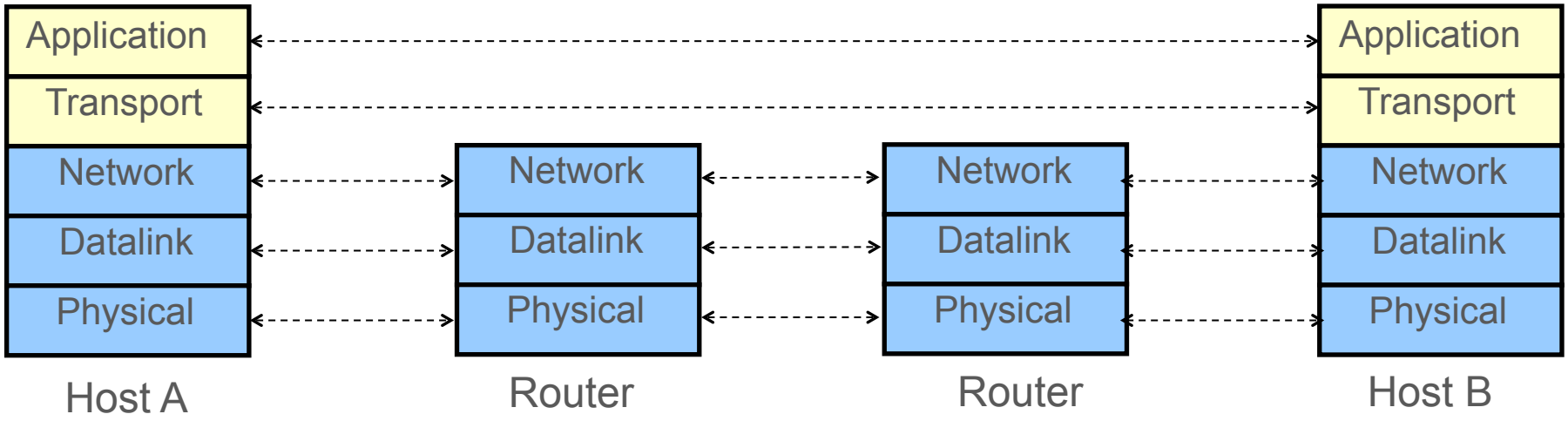
- Layers interact with peer's corresponding layer
- Lower three layers implemented everywhere
- Top two layers implemented only at hosts



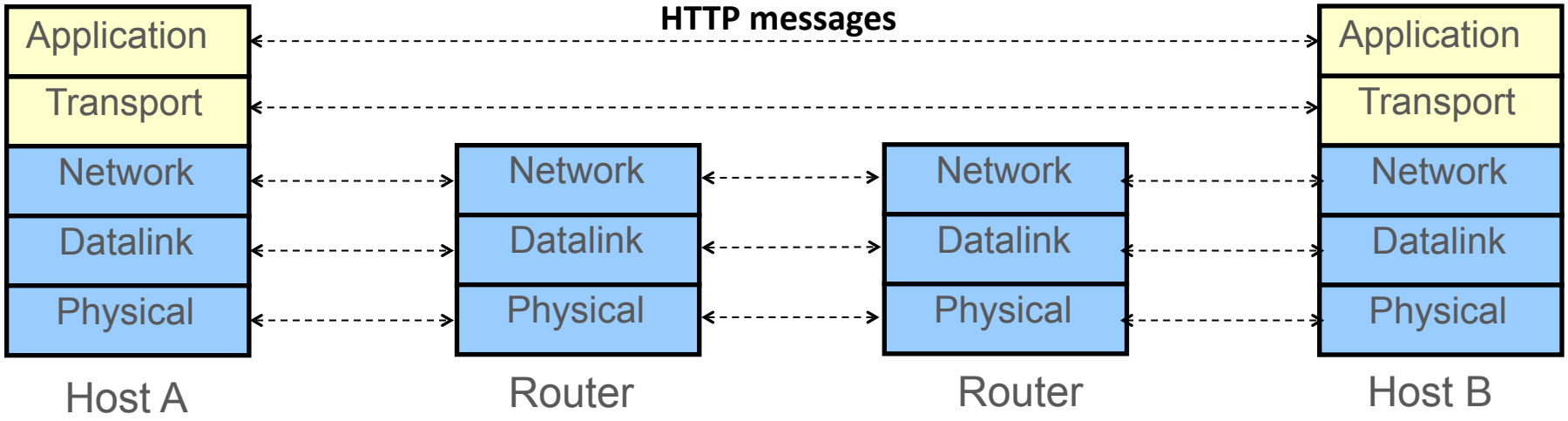
A closer look: network



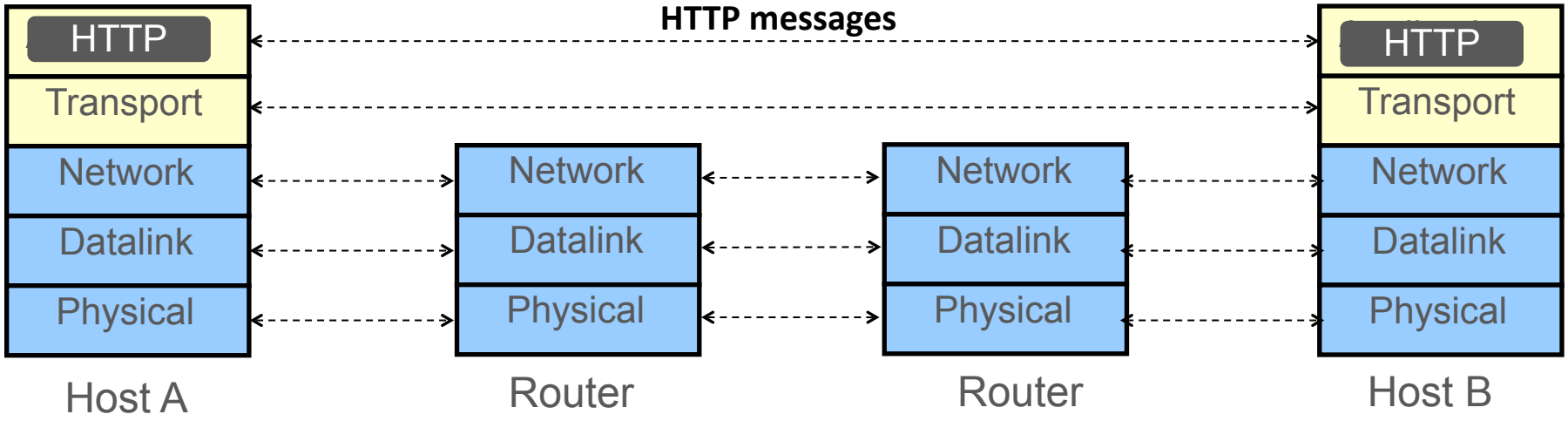
Example: simple protocol diagram



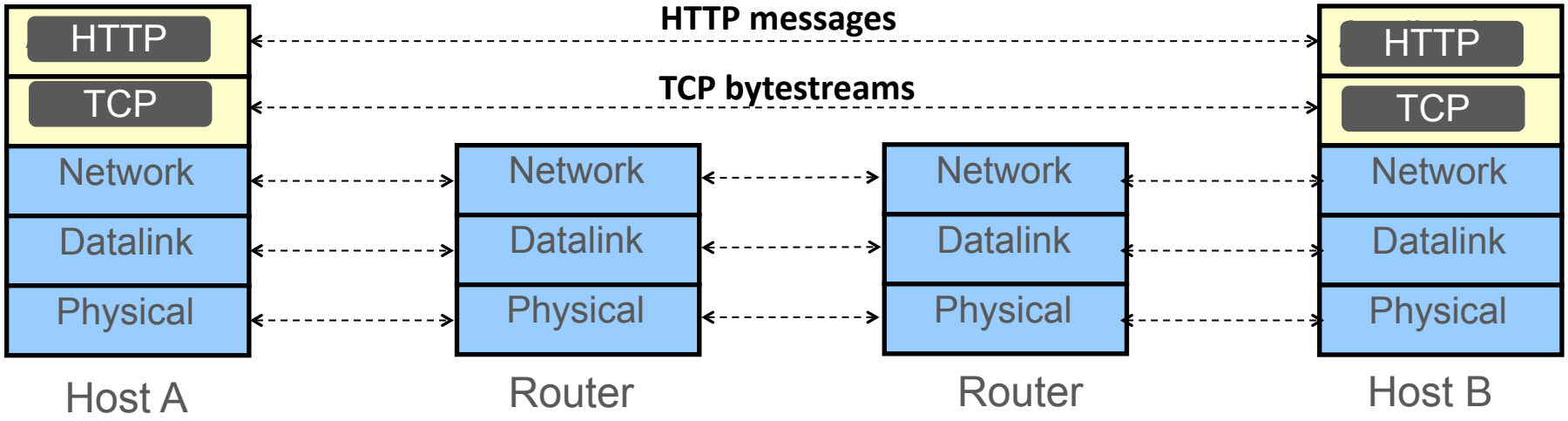
Example: simple protocol diagram



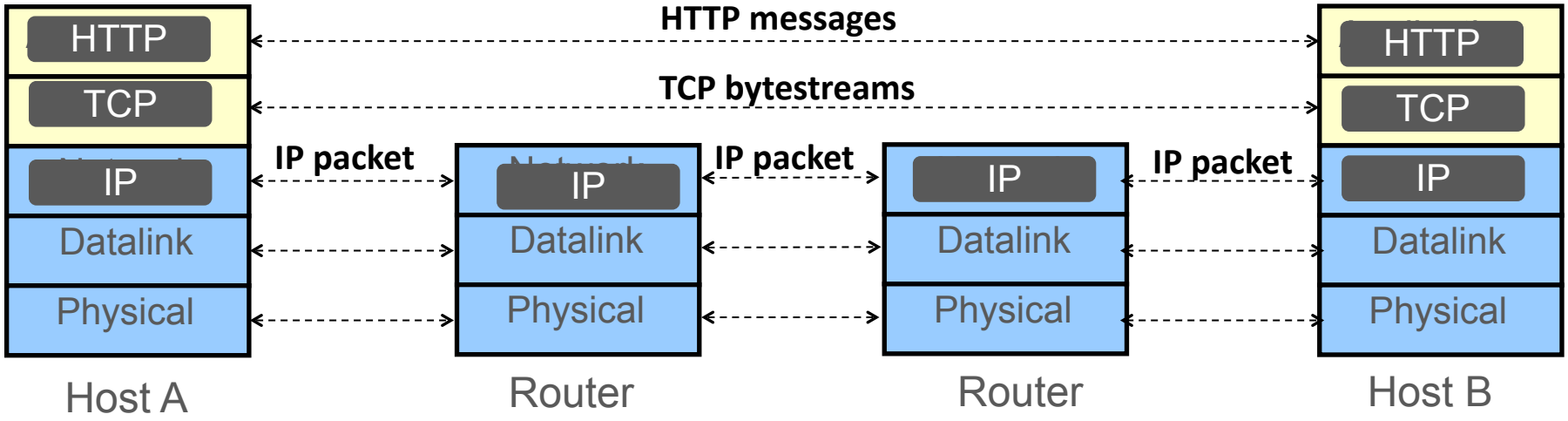
Example: simple protocol diagram



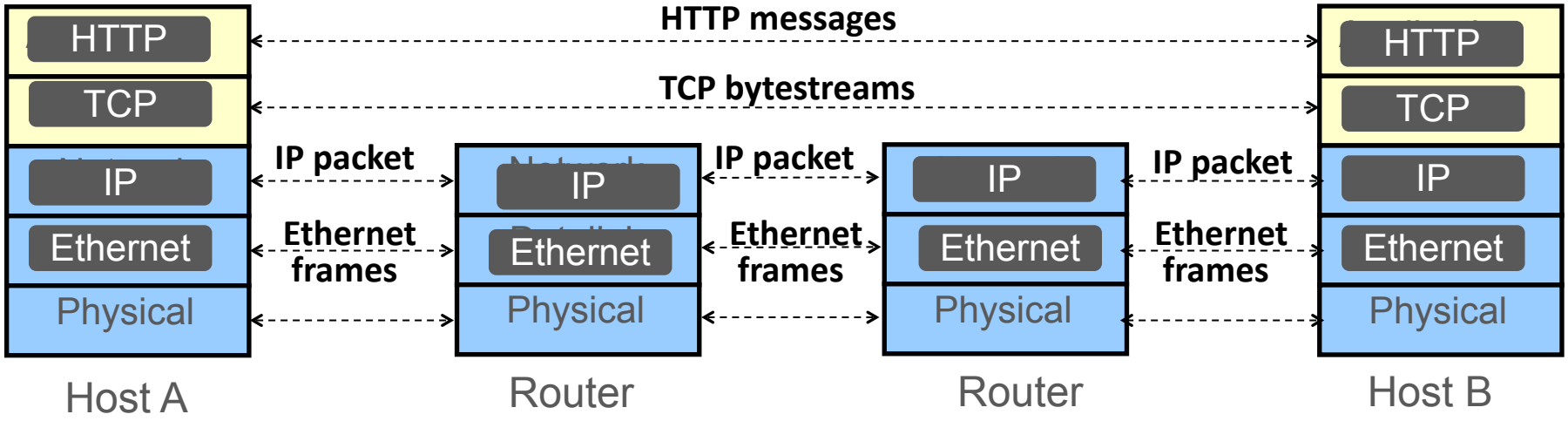
Example: simple protocol diagram



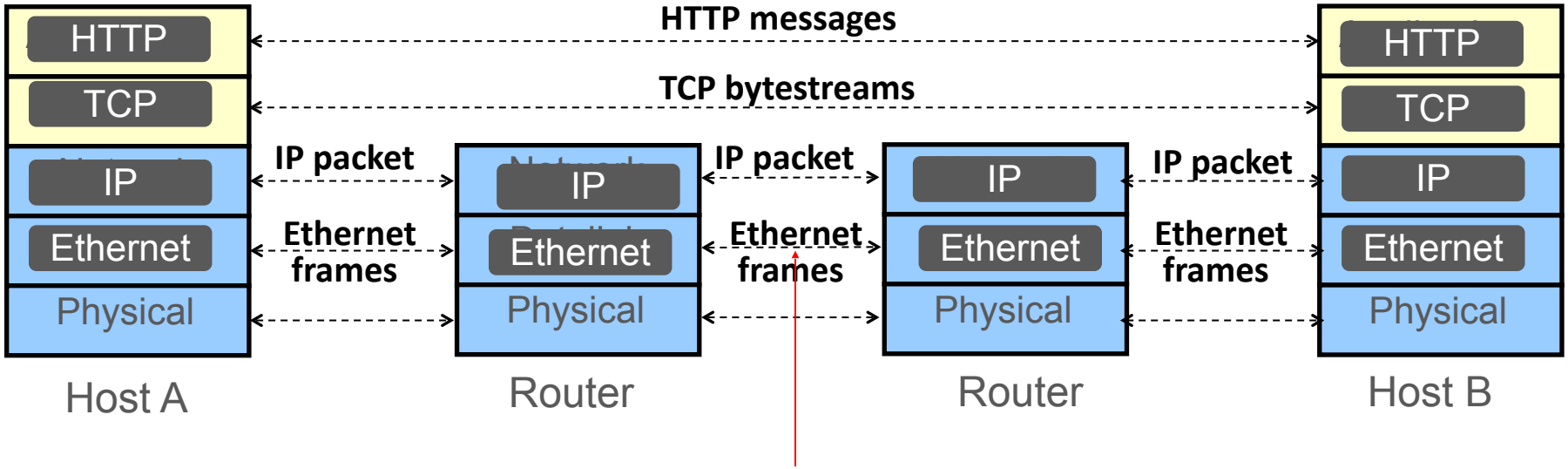
Example: simple protocol diagram



Example: simple protocol diagram

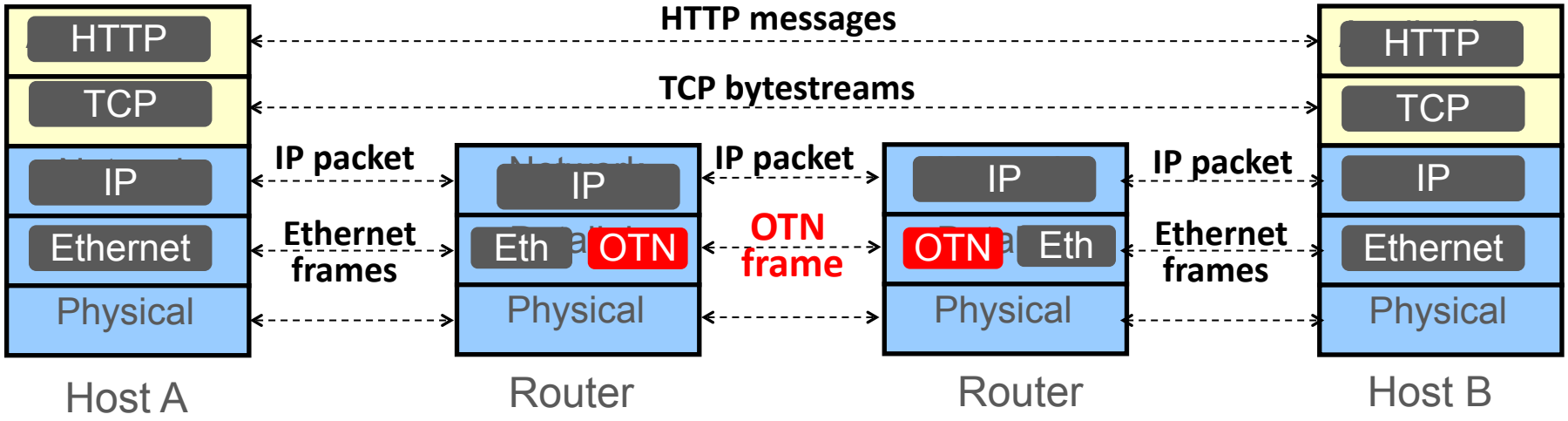


Example: simple protocol diagram



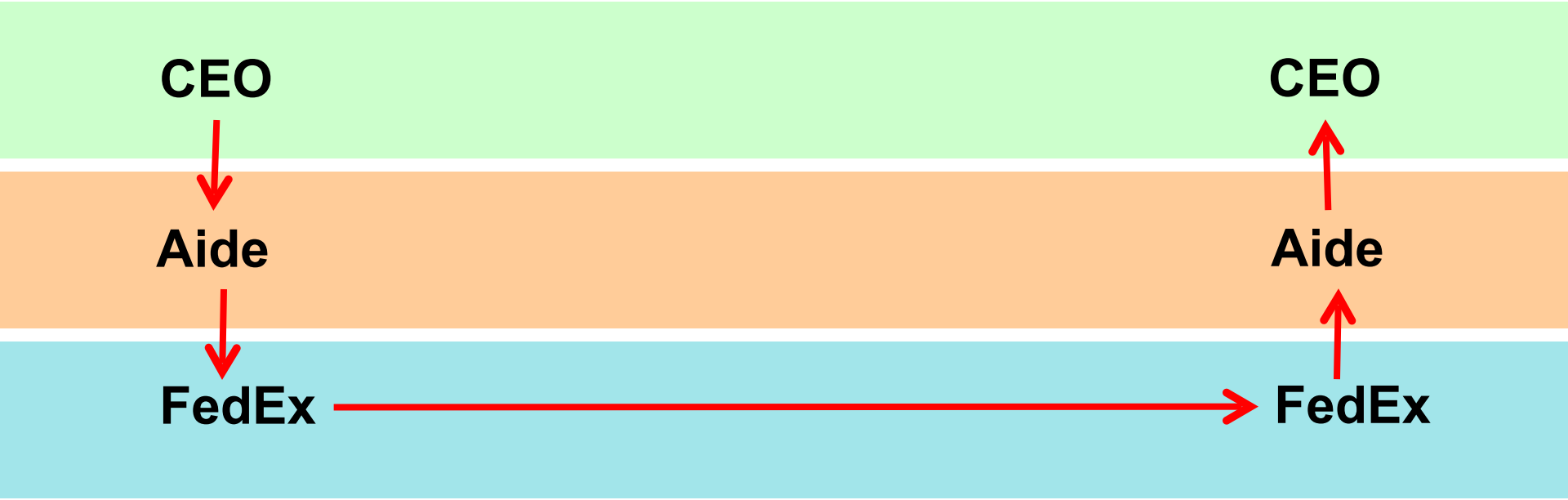
What if this was an OTN network?

Example: simple protocol diagram



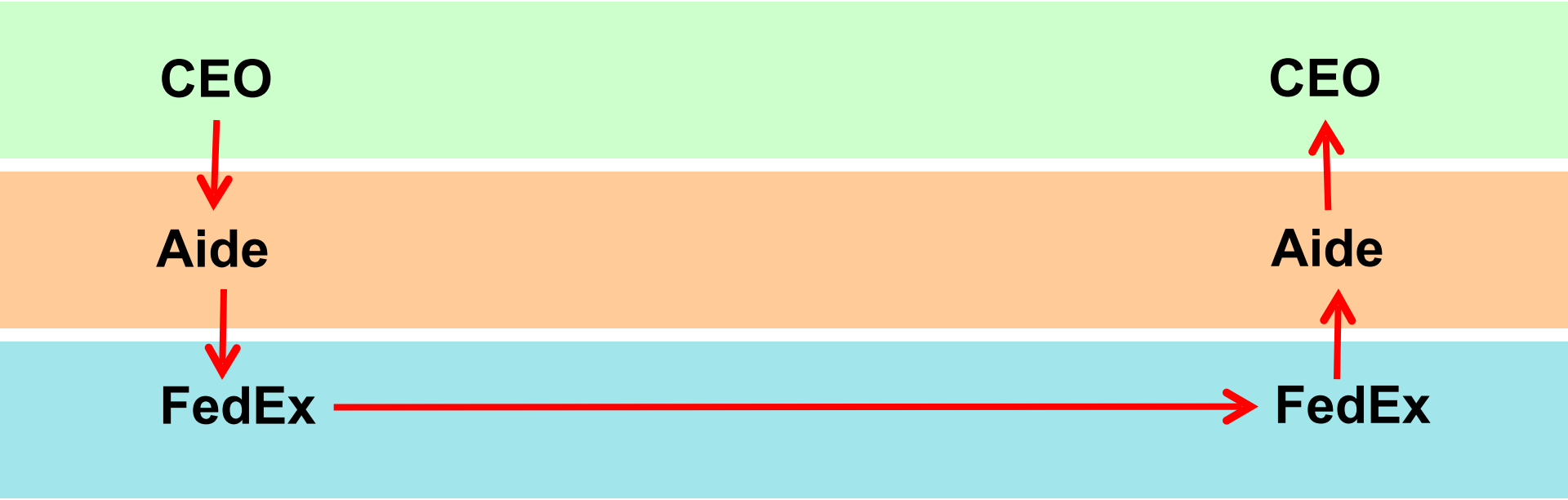
Recap: Physical Communication

- Communication goes down the layers
- Then up to relevant layer



Recall: the path of the letter

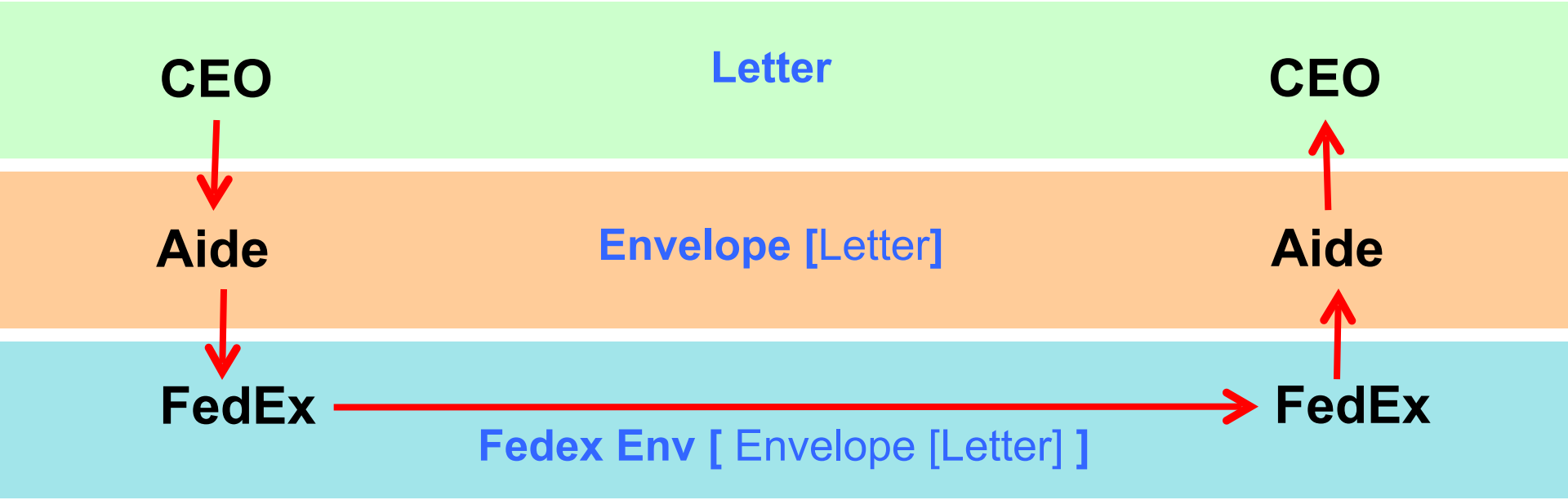
Recap: Physical Communication



Recall: the path of the letter

Recap: Physical Communication

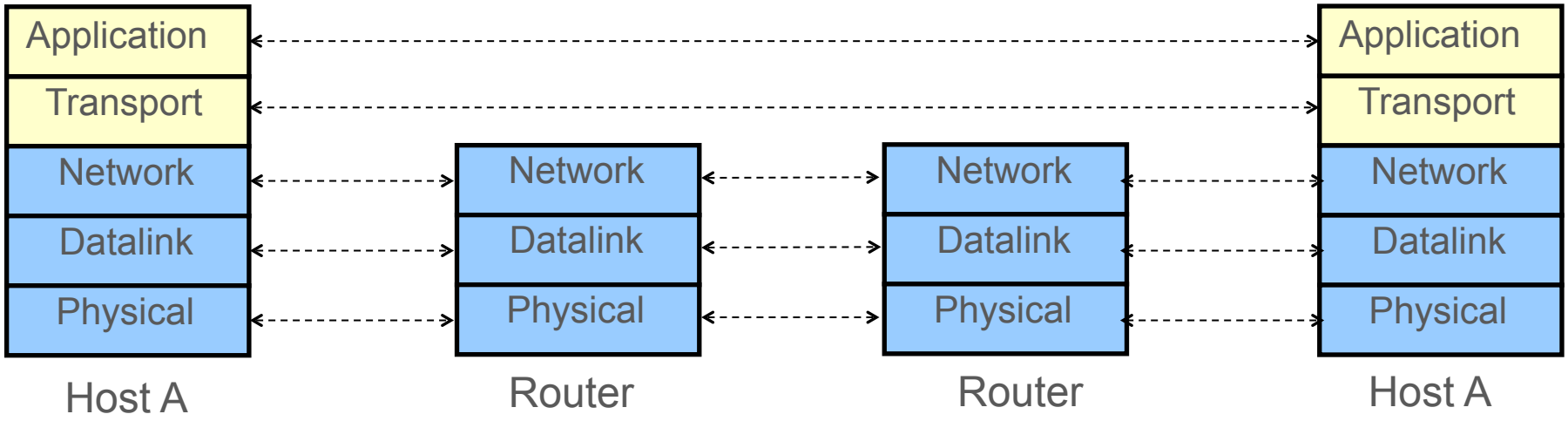
- Communication goes down the layers
- Then up to relevant layer
- Lowest layer has the most “packaging”



Recall: the path of the letter

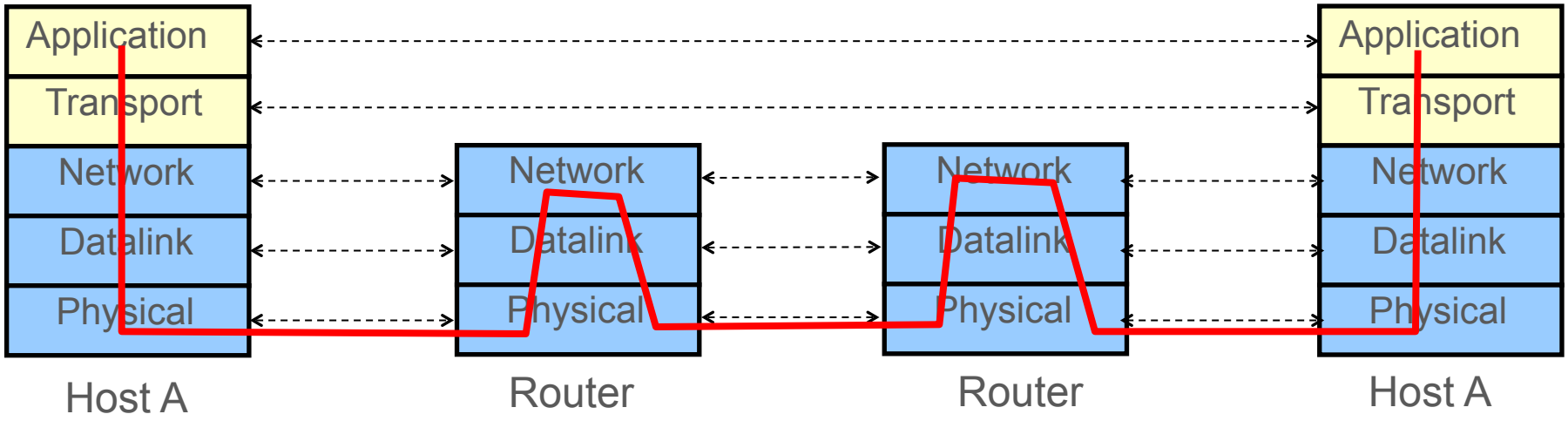
Recap: Physical Communication

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

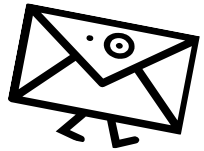


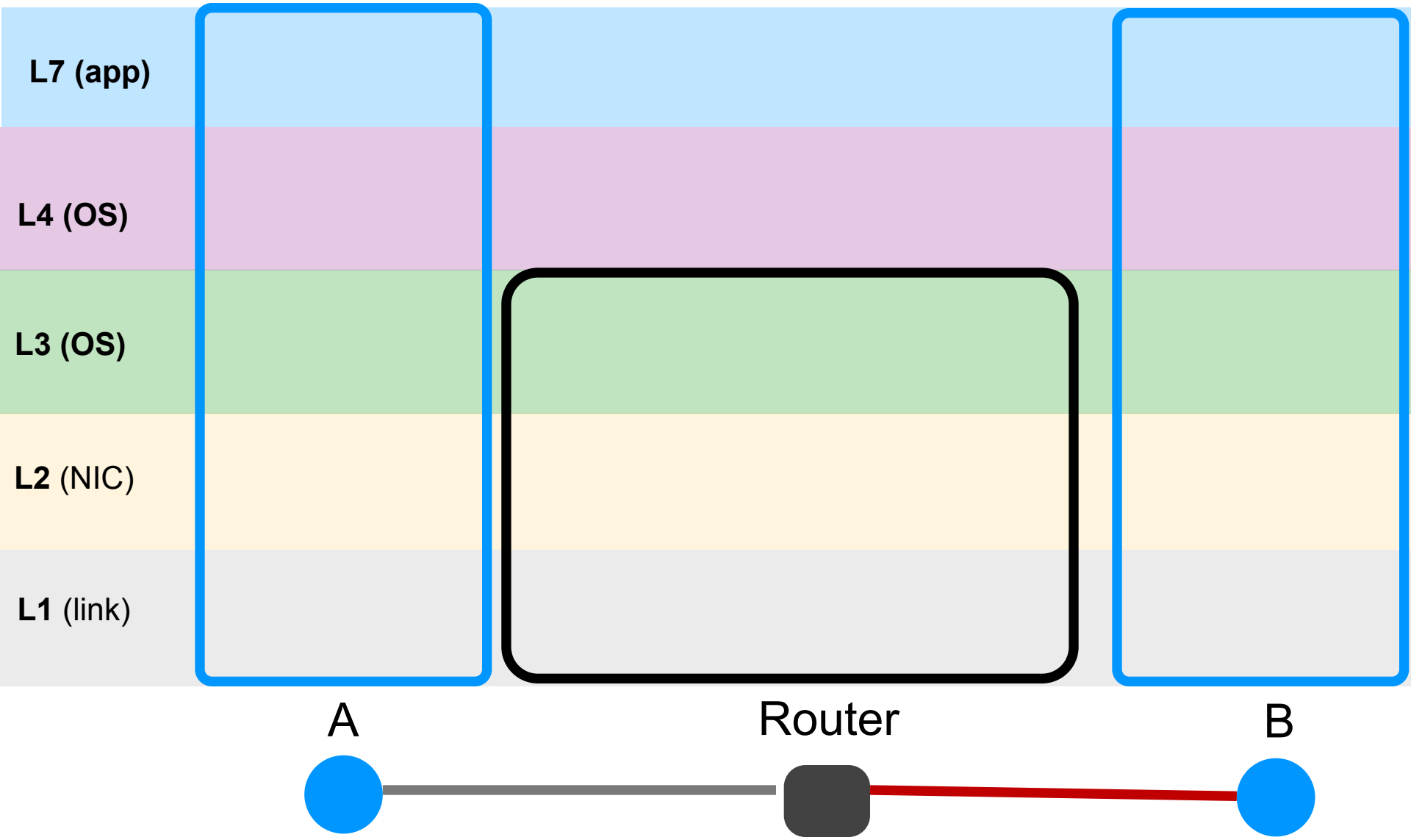
Recap: Physical Communication

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

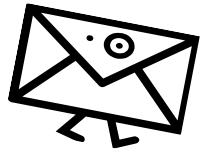


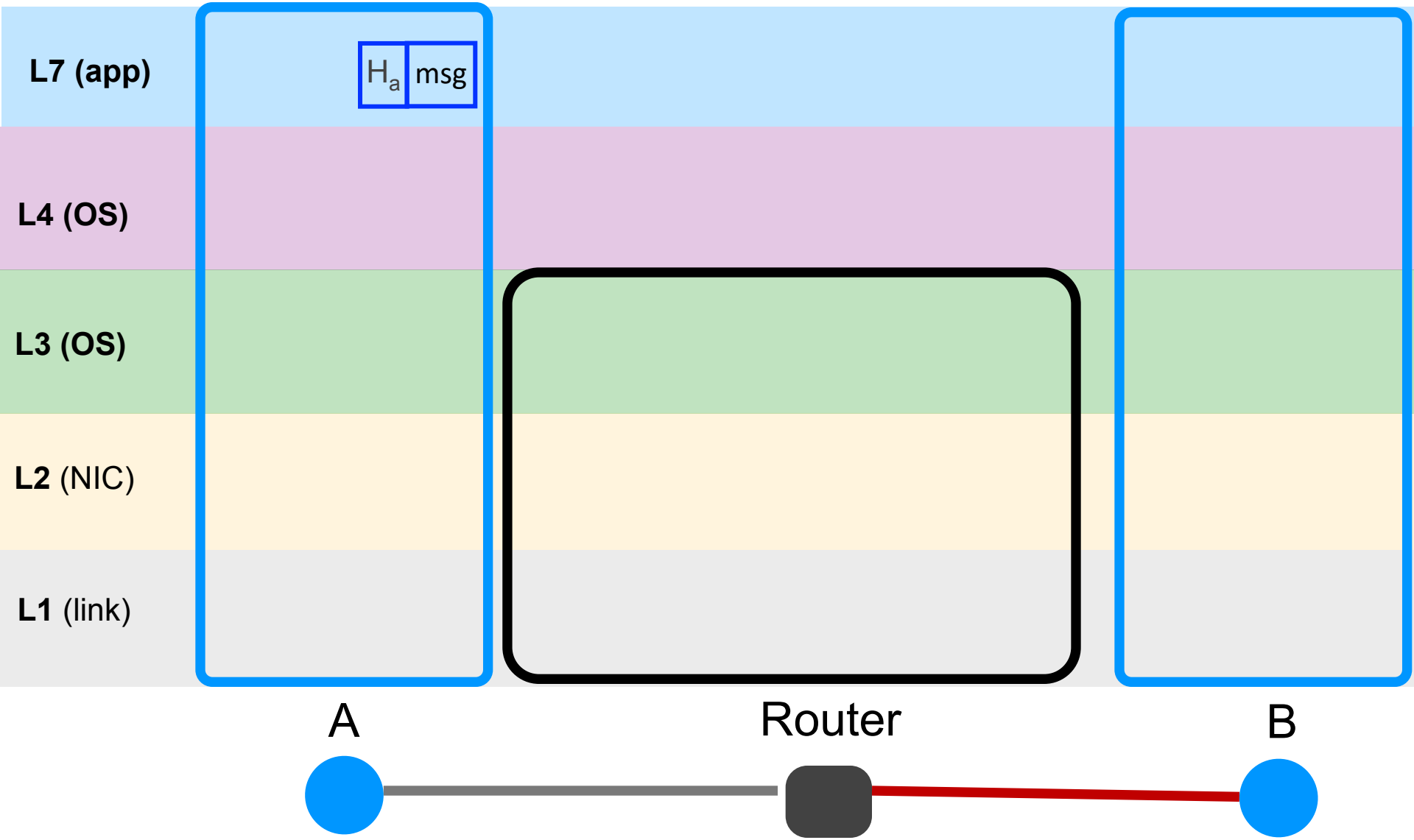
Layer Encapsulation



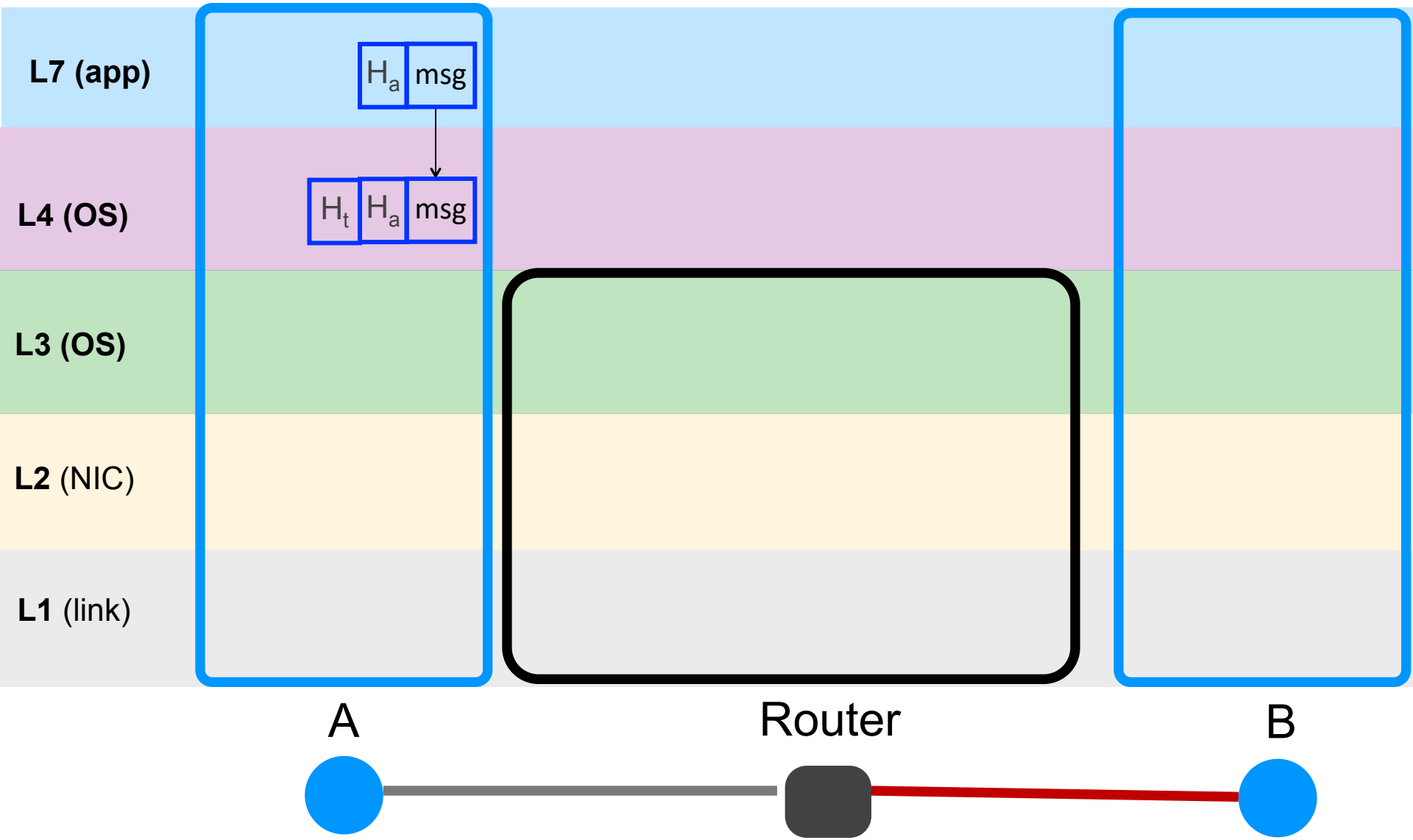


Layer Encapsulation

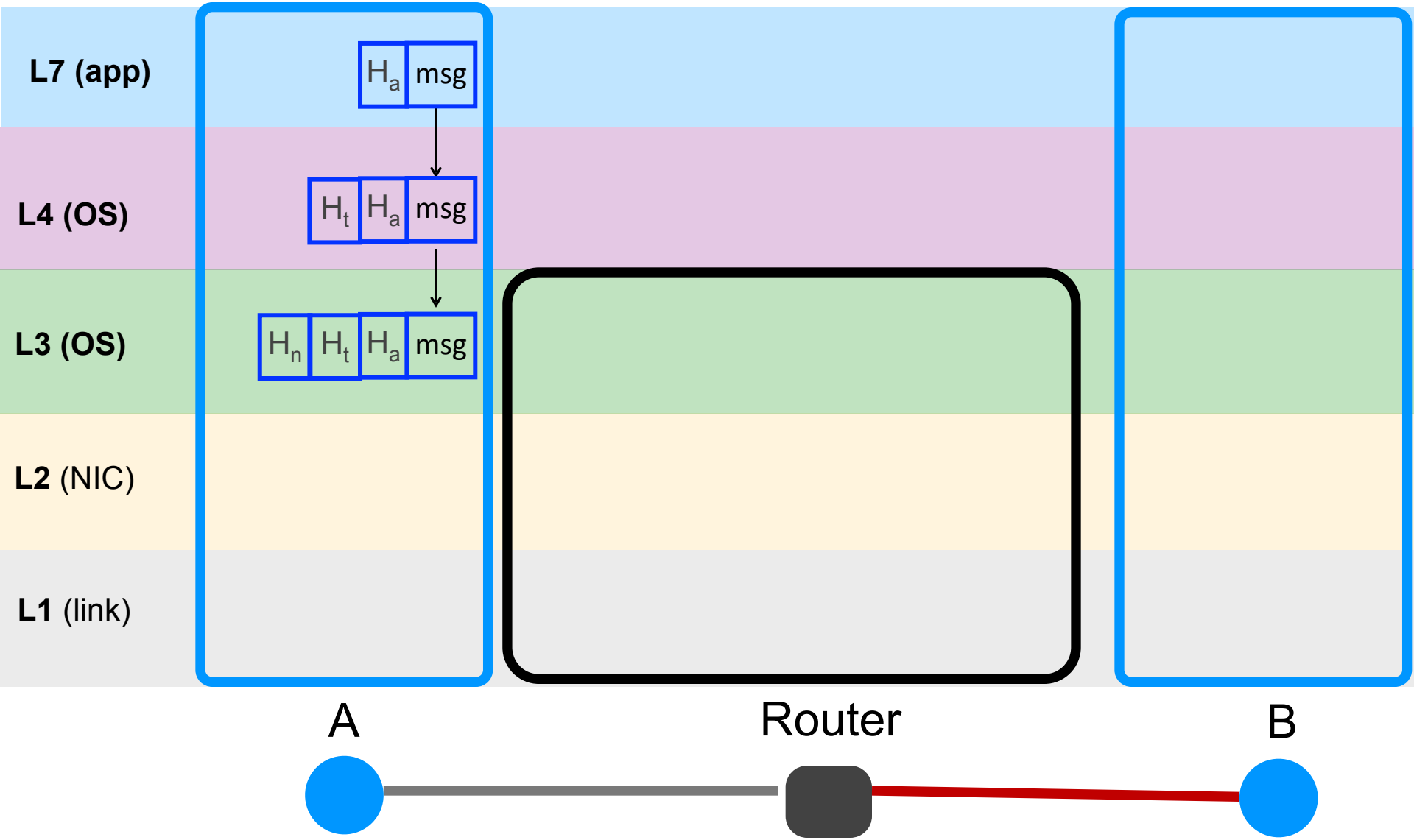




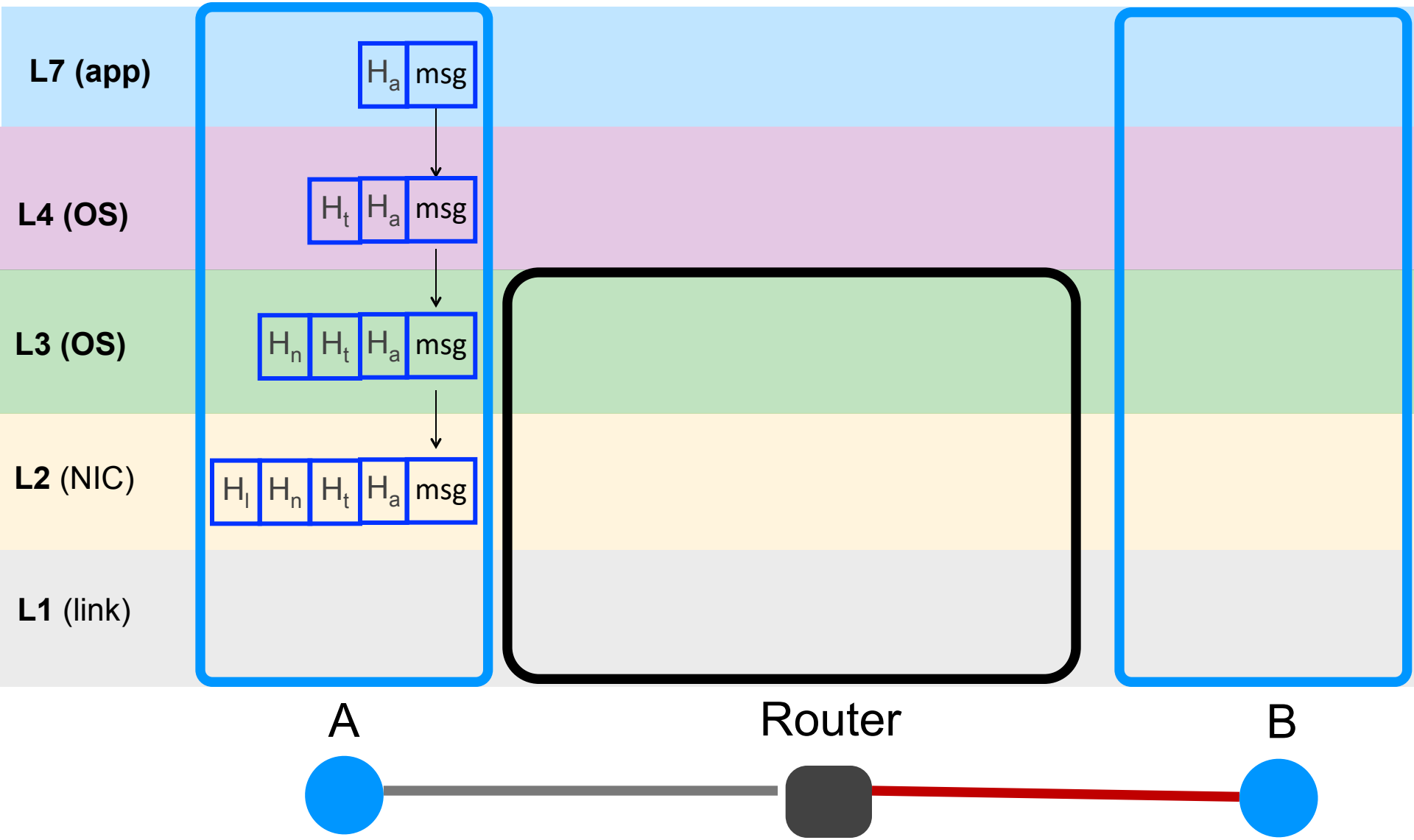
Layer Encapsulation



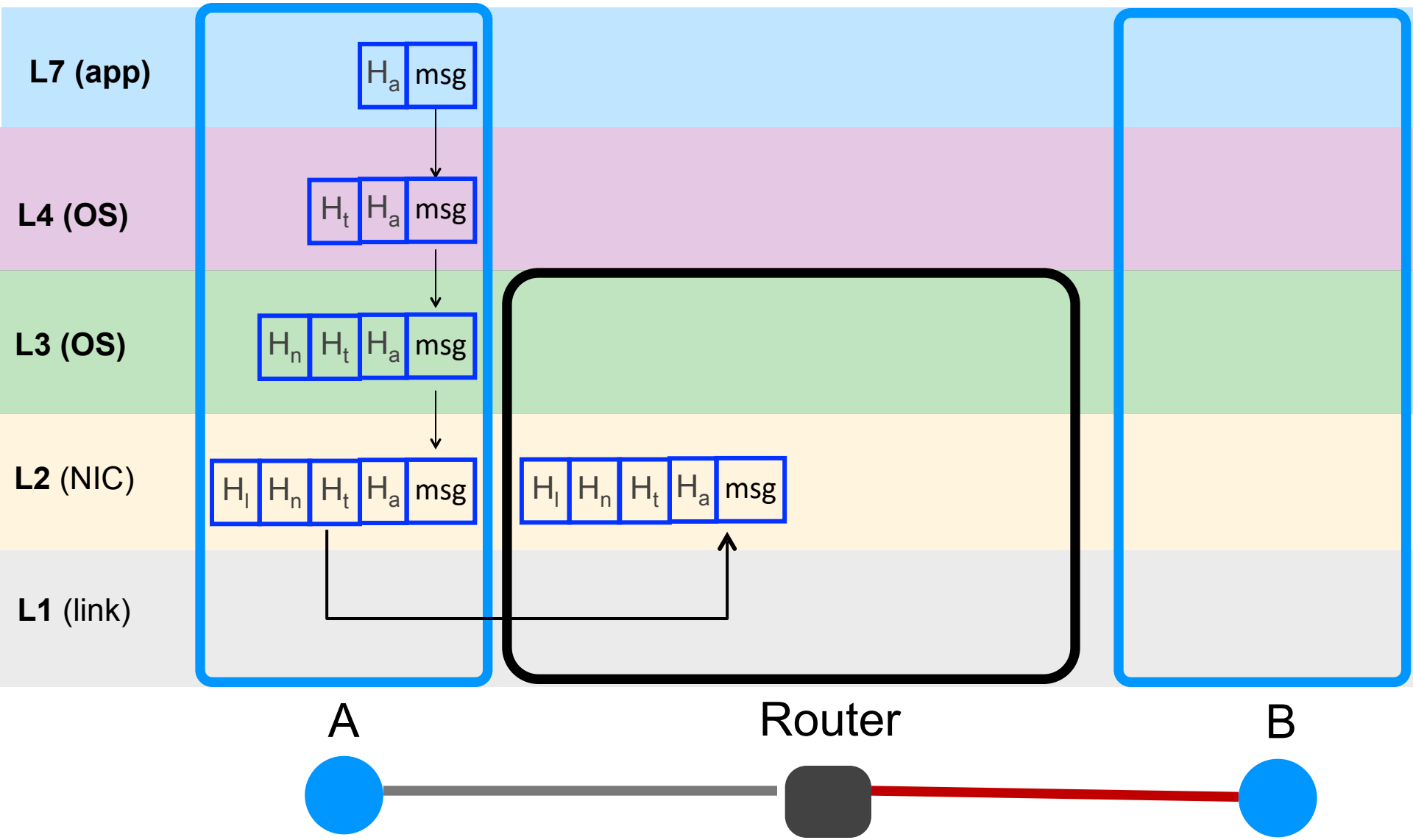
Layer Encapsulation



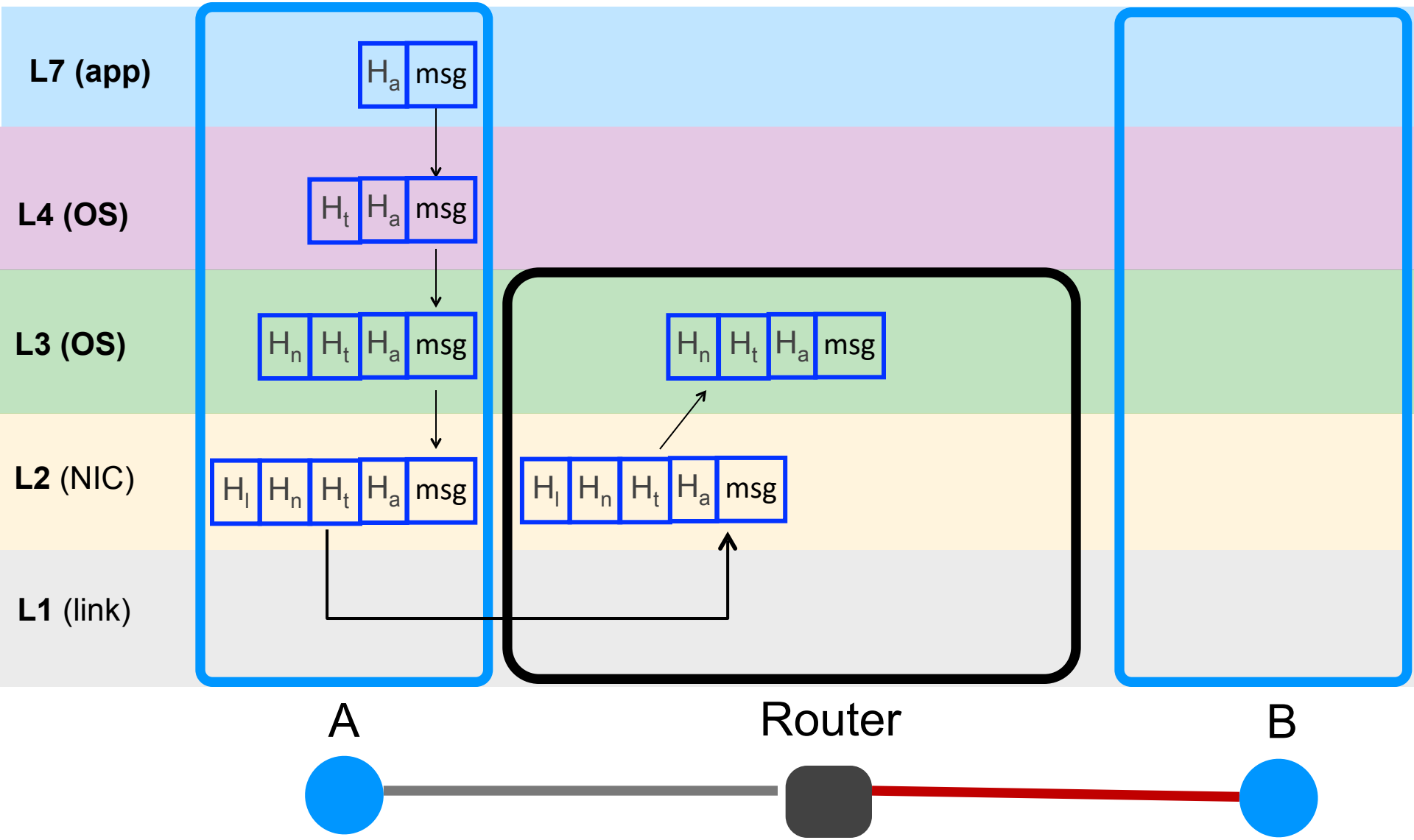
Layer Encapsulation



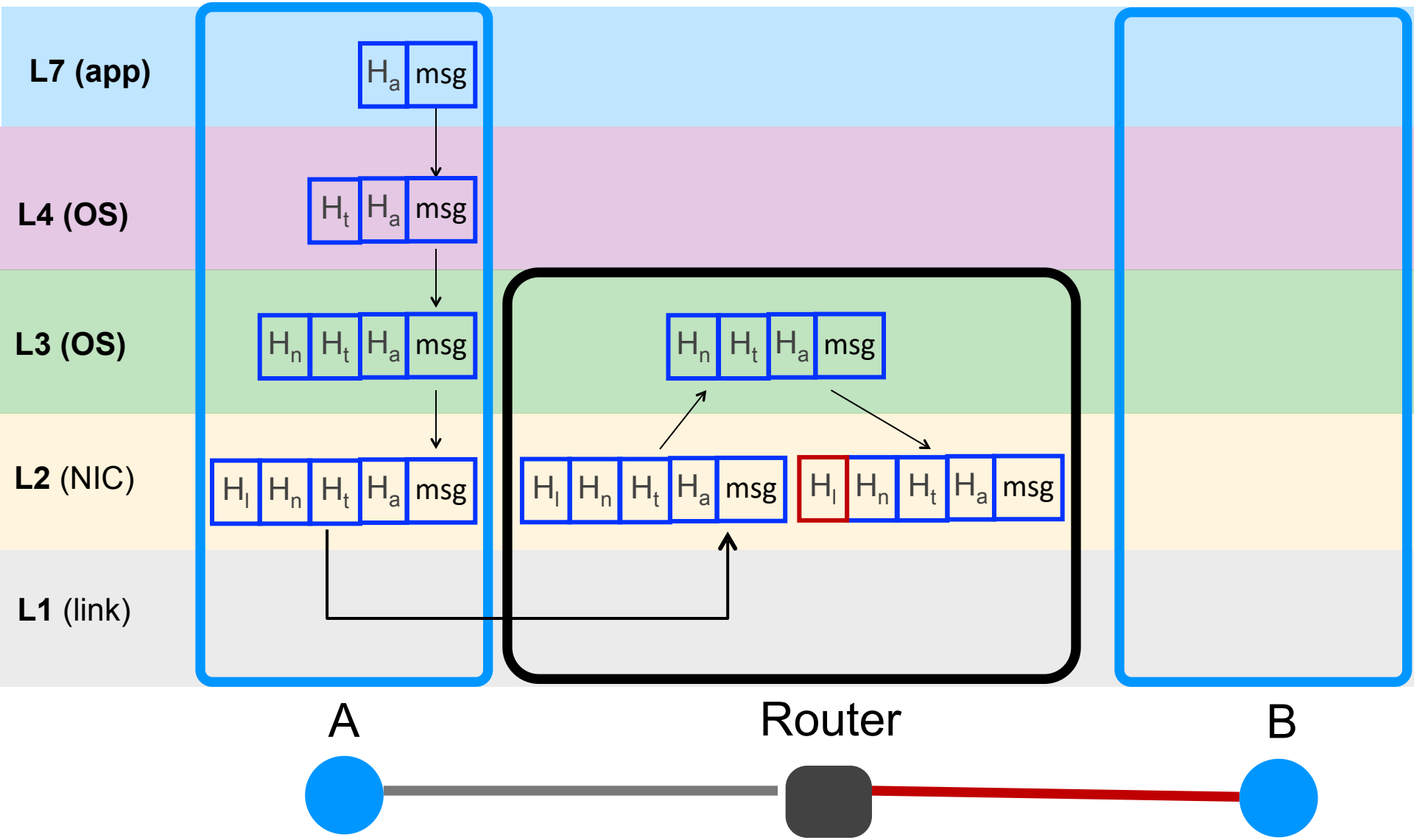
Layer Encapsulation



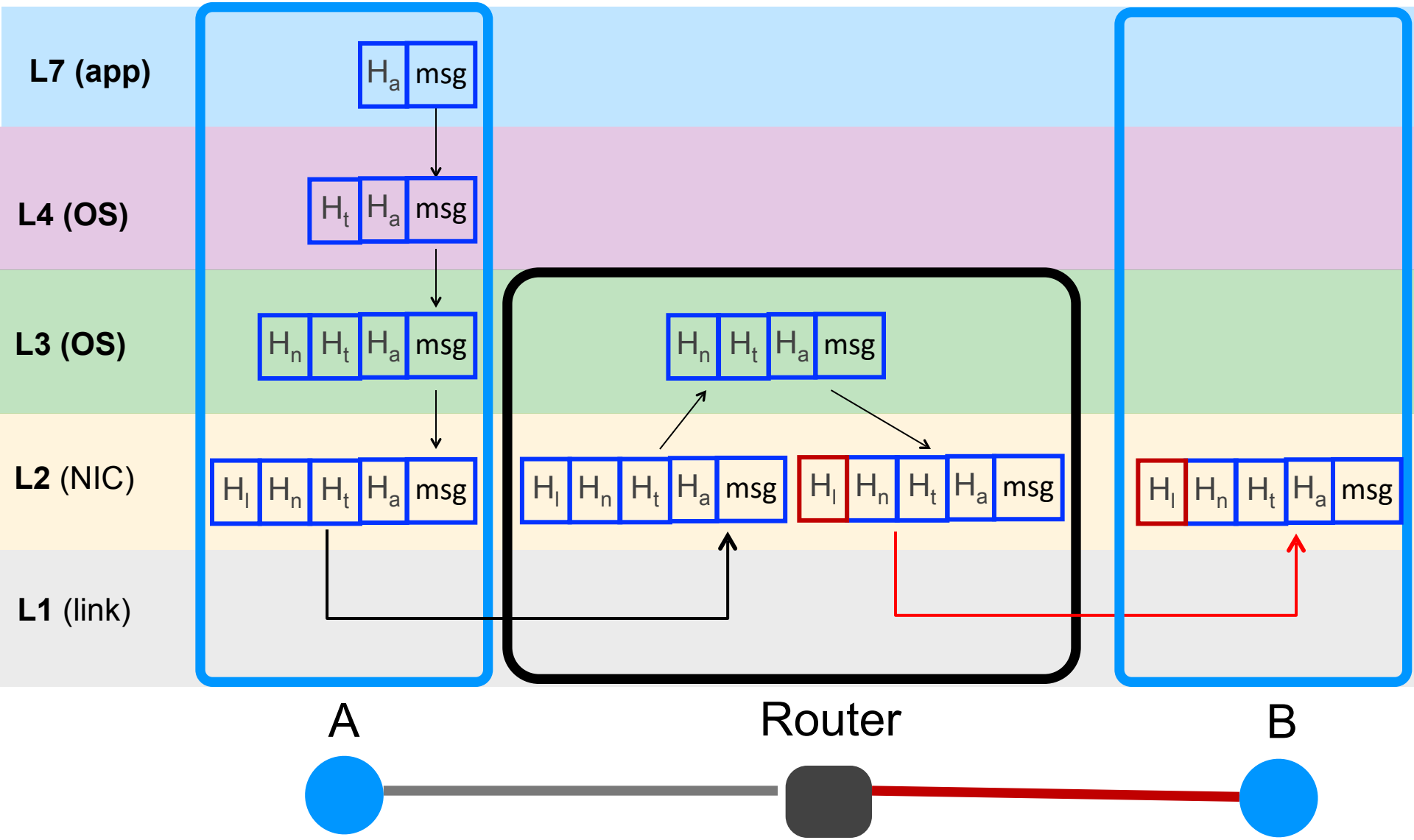
Layer Encapsulation



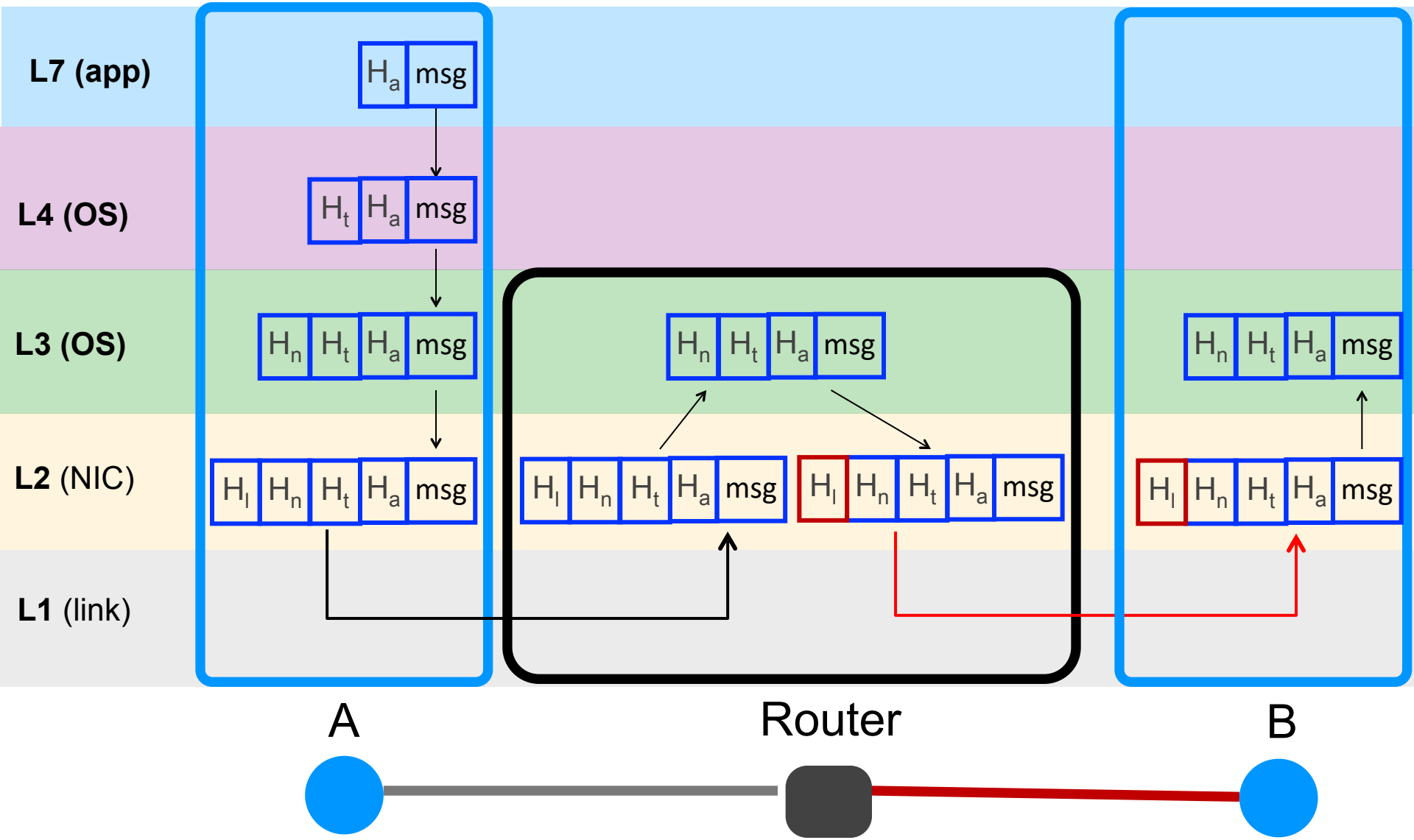
Layer Encapsulation



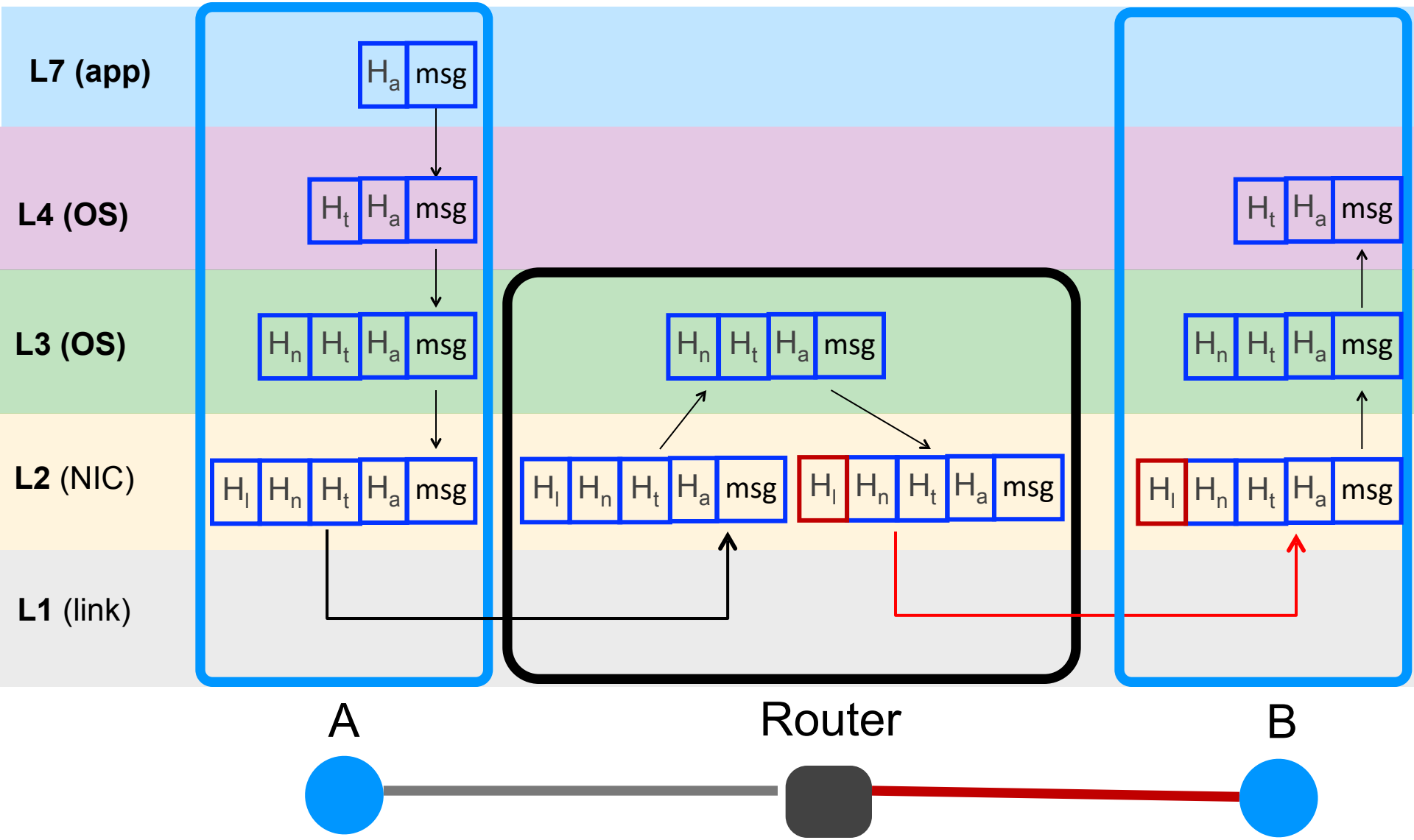
Layer Encapsulation



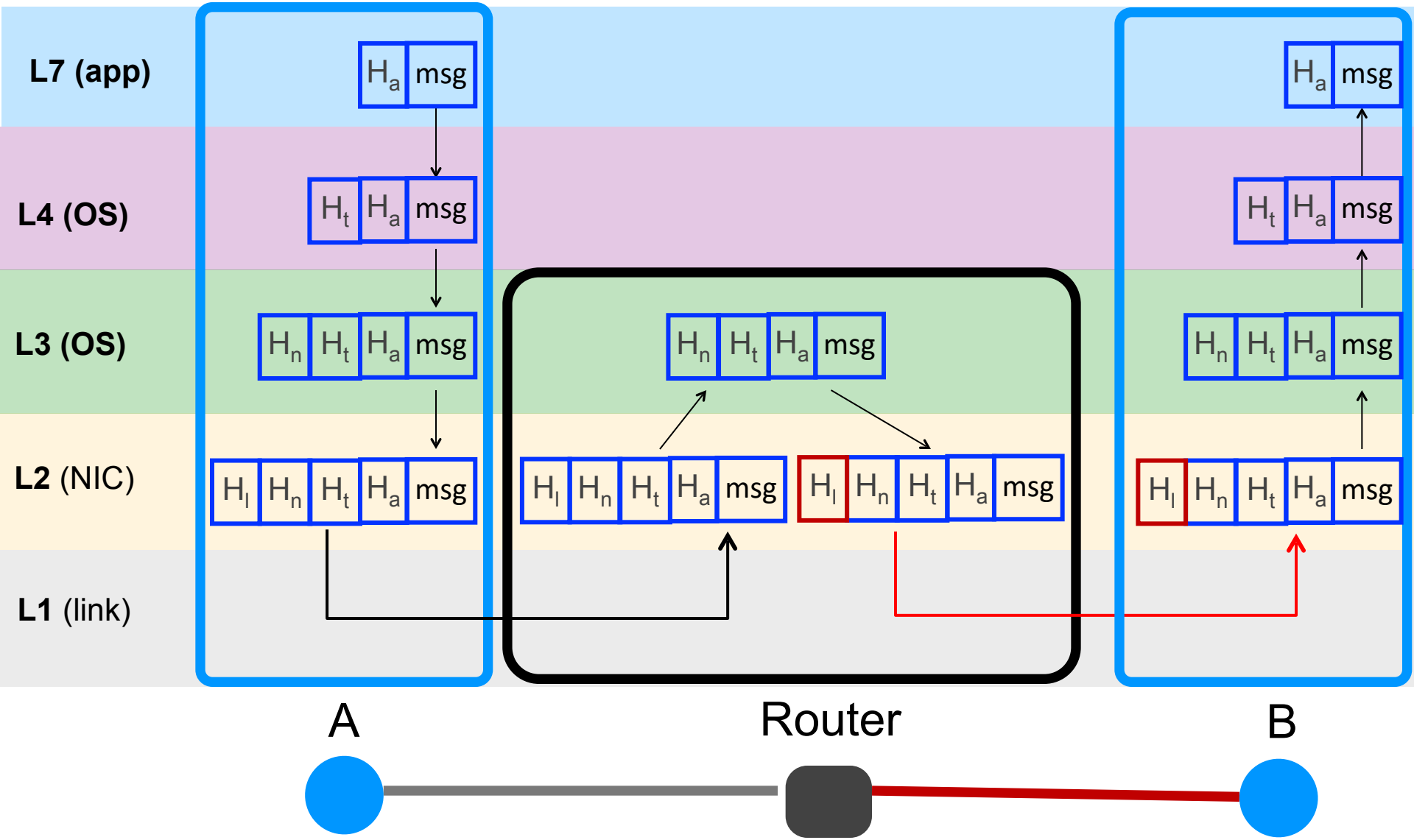
Layer Encapsulation



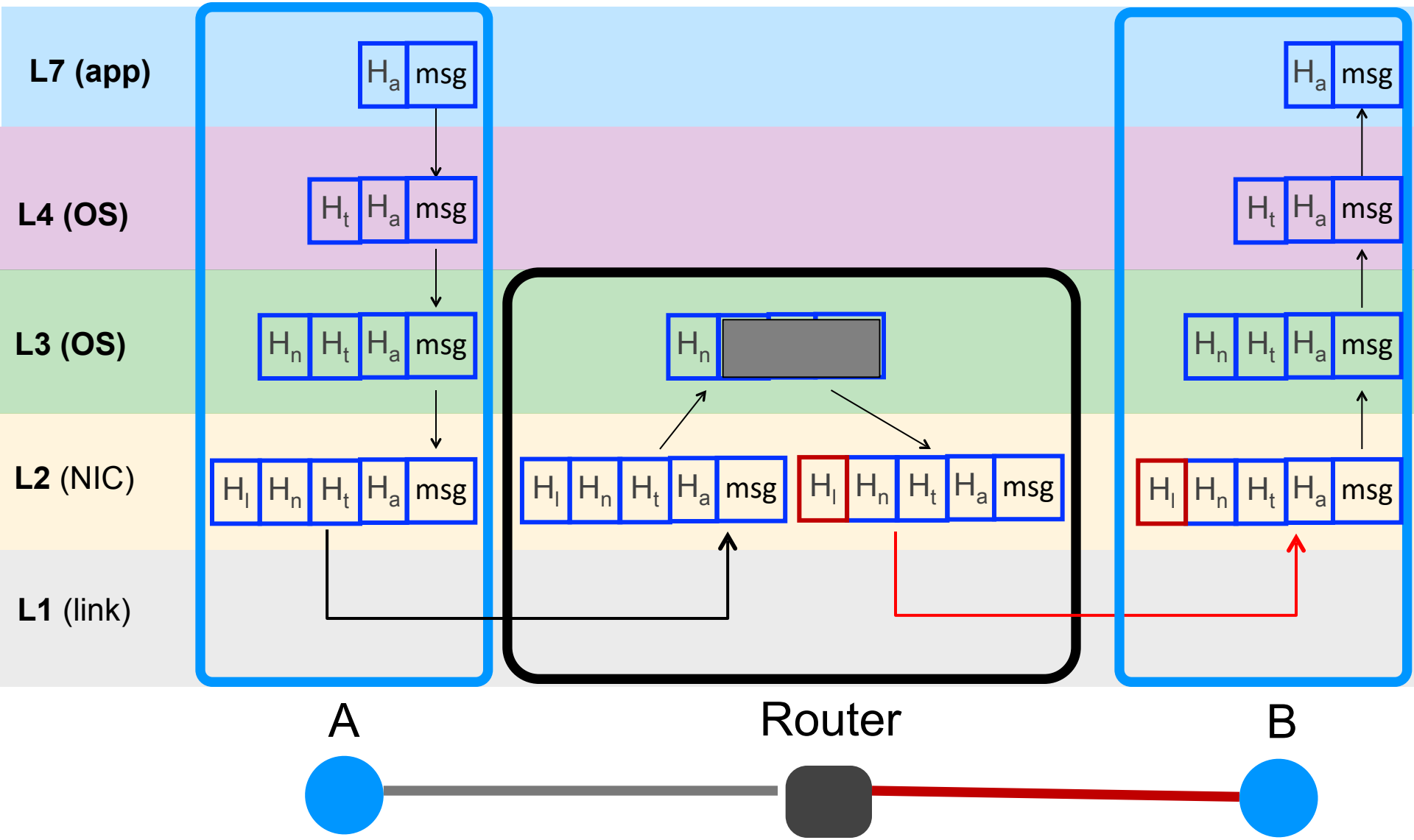
Layer Encapsulation



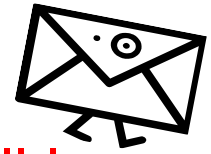
Layer Encapsulation



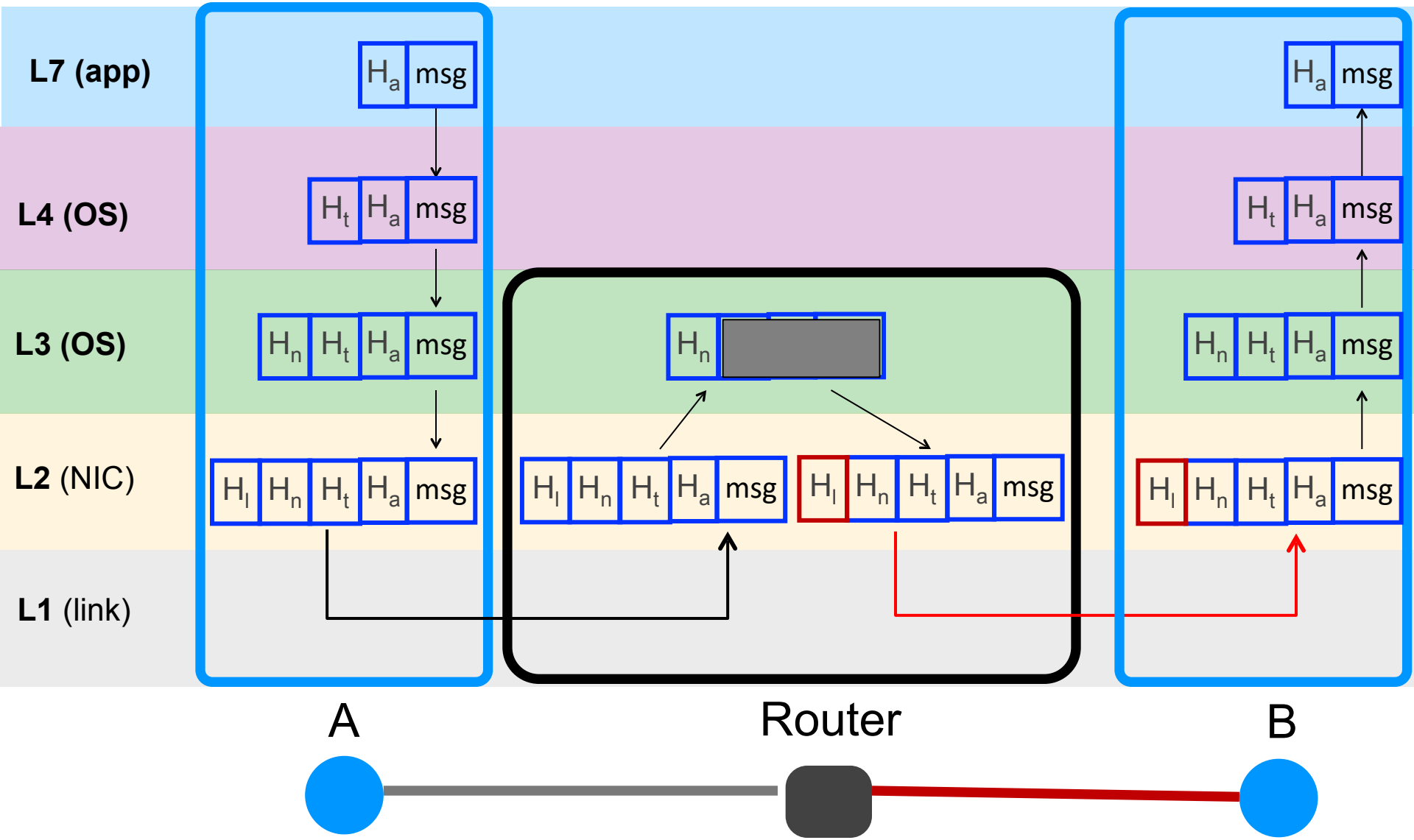
Layer Encapsulation



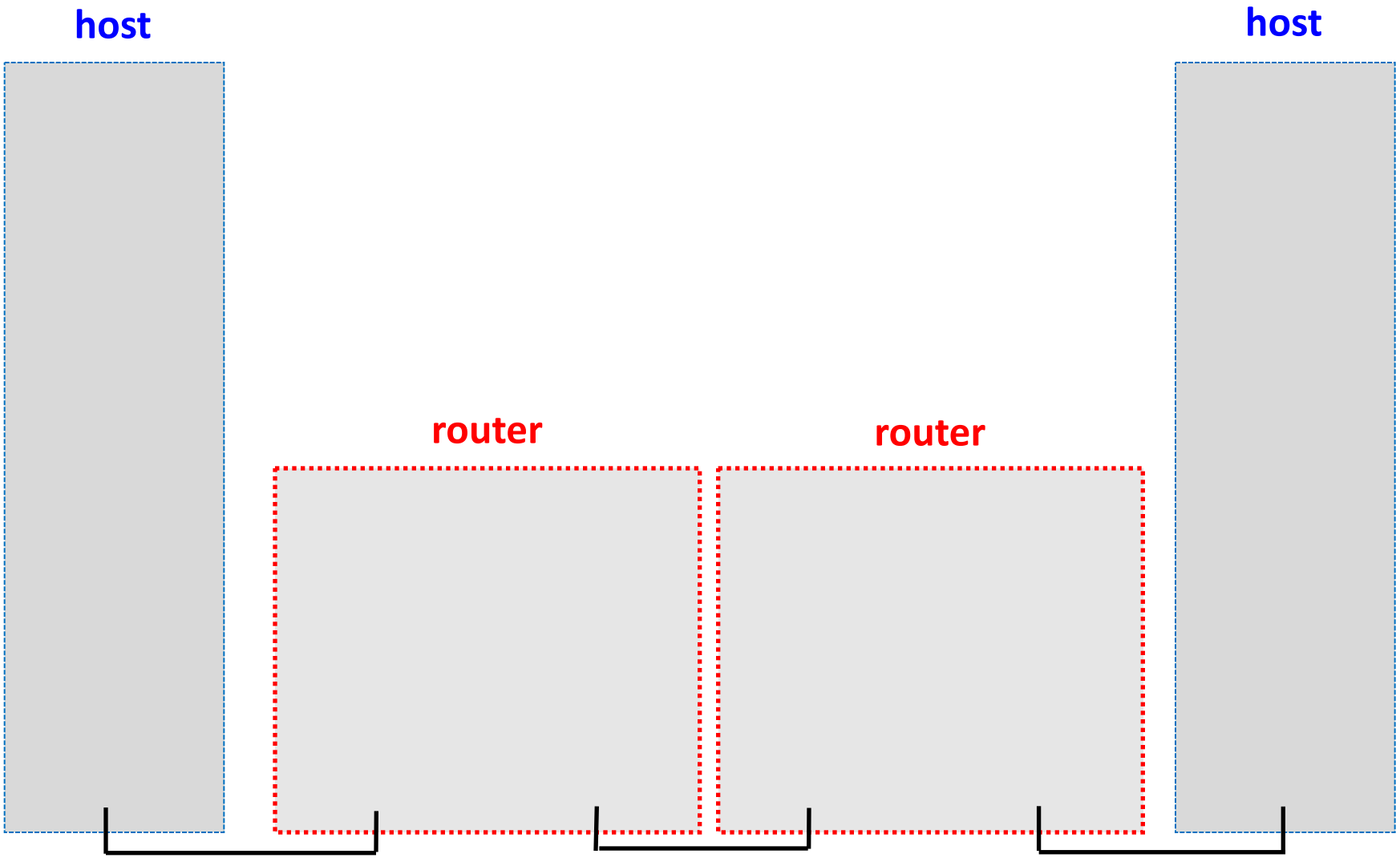
Layer Encapsulation



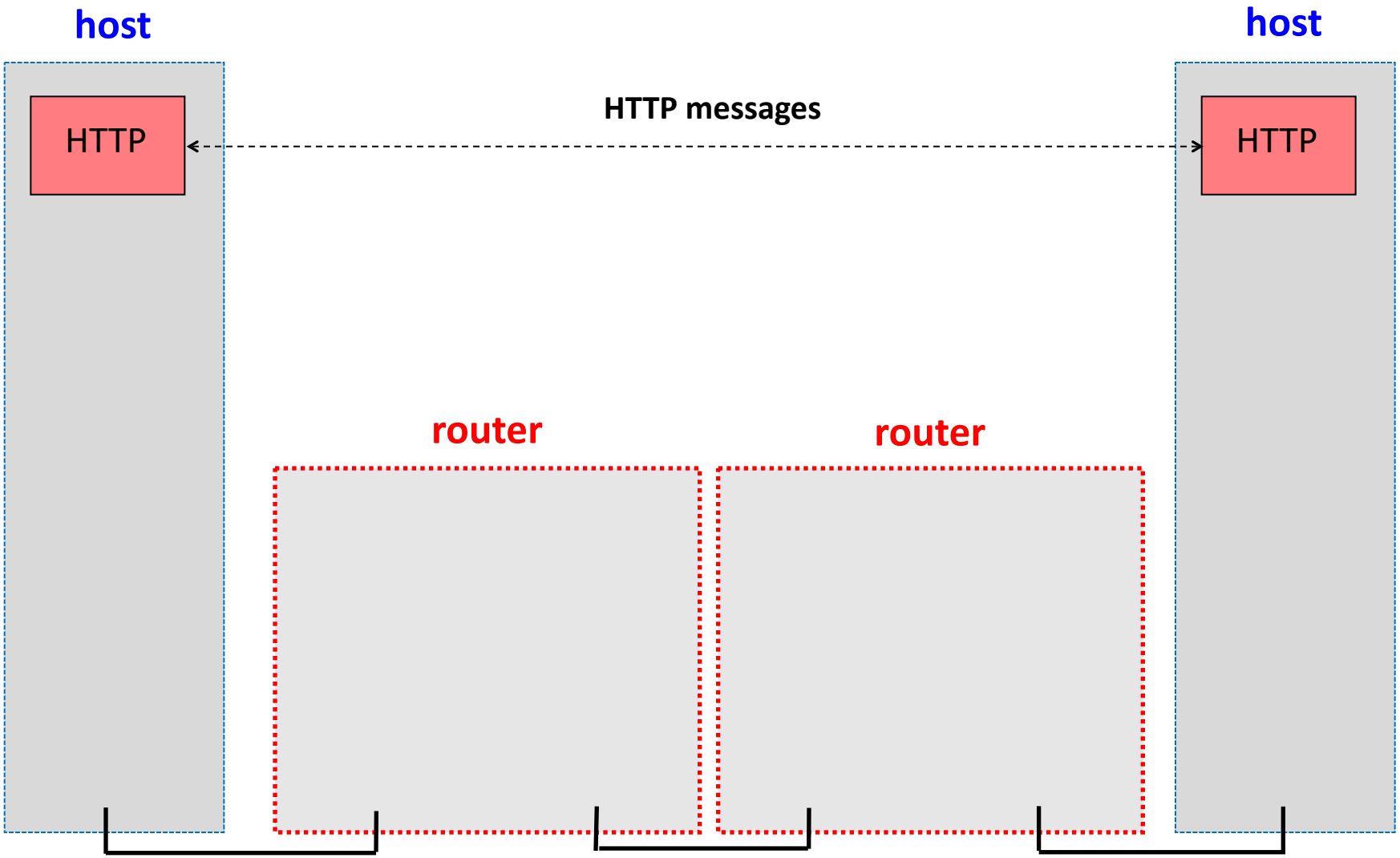
On the wire: packet has data + headers from all layers



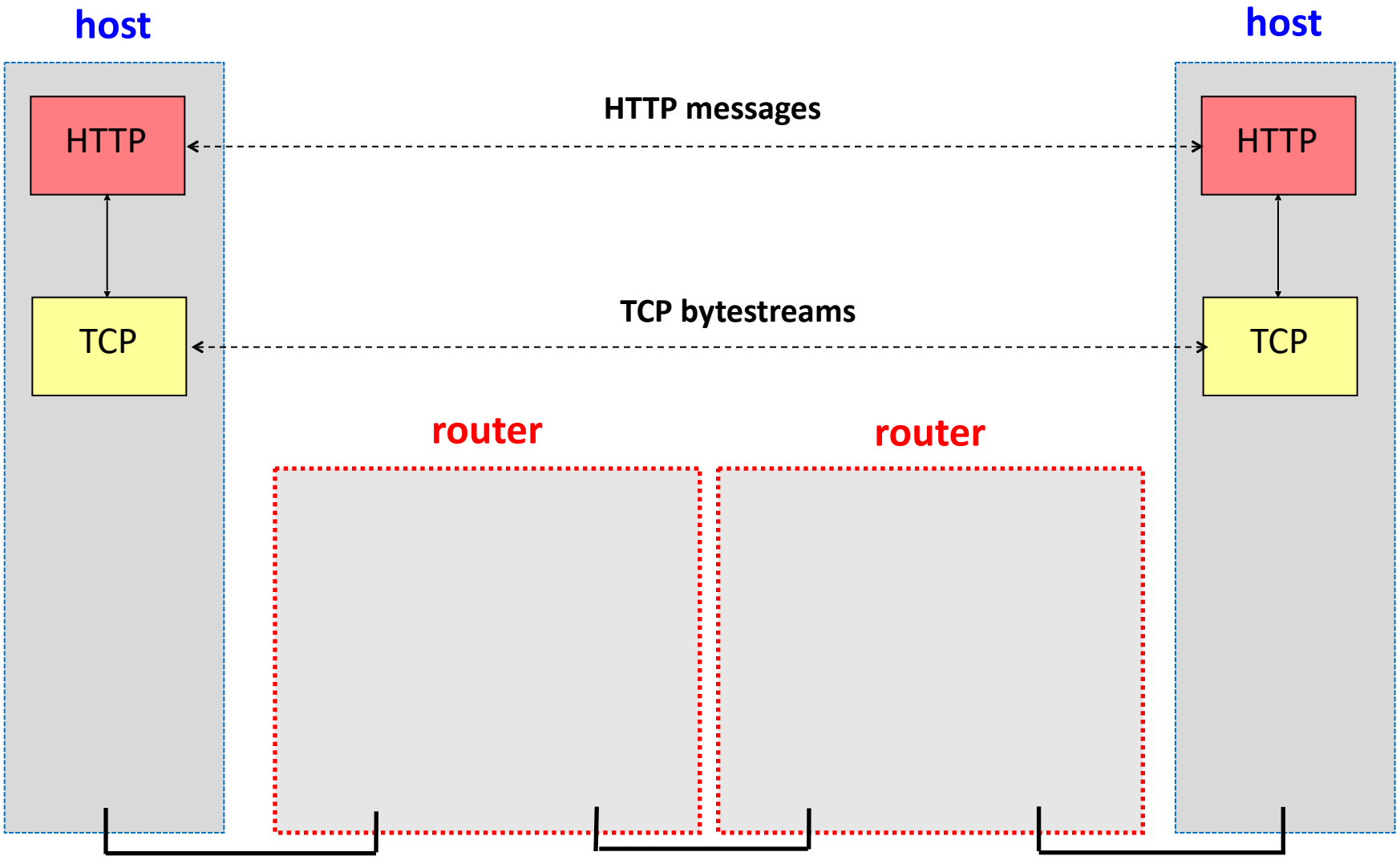
Complicated protocol diagram



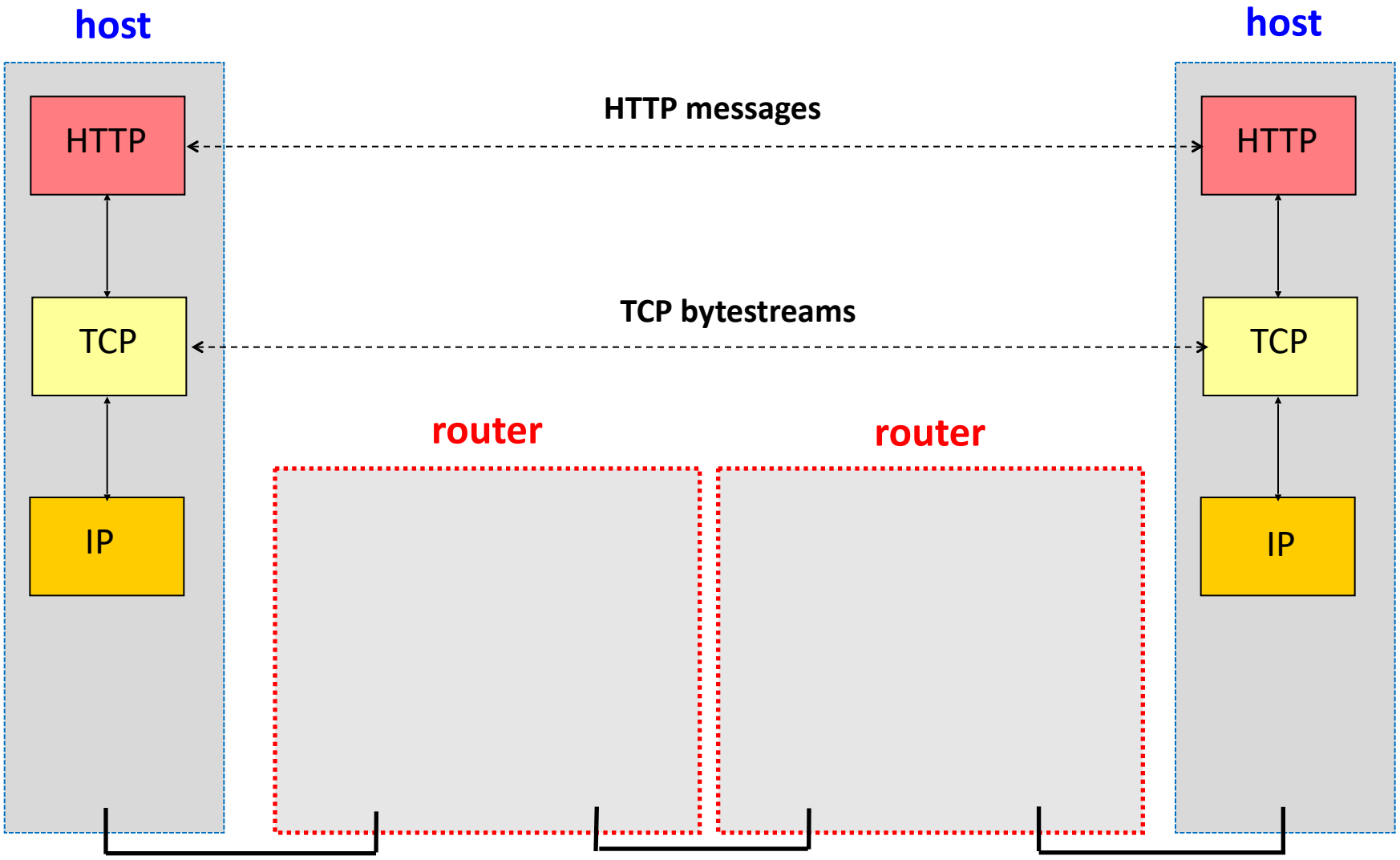
Complicated protocol diagram



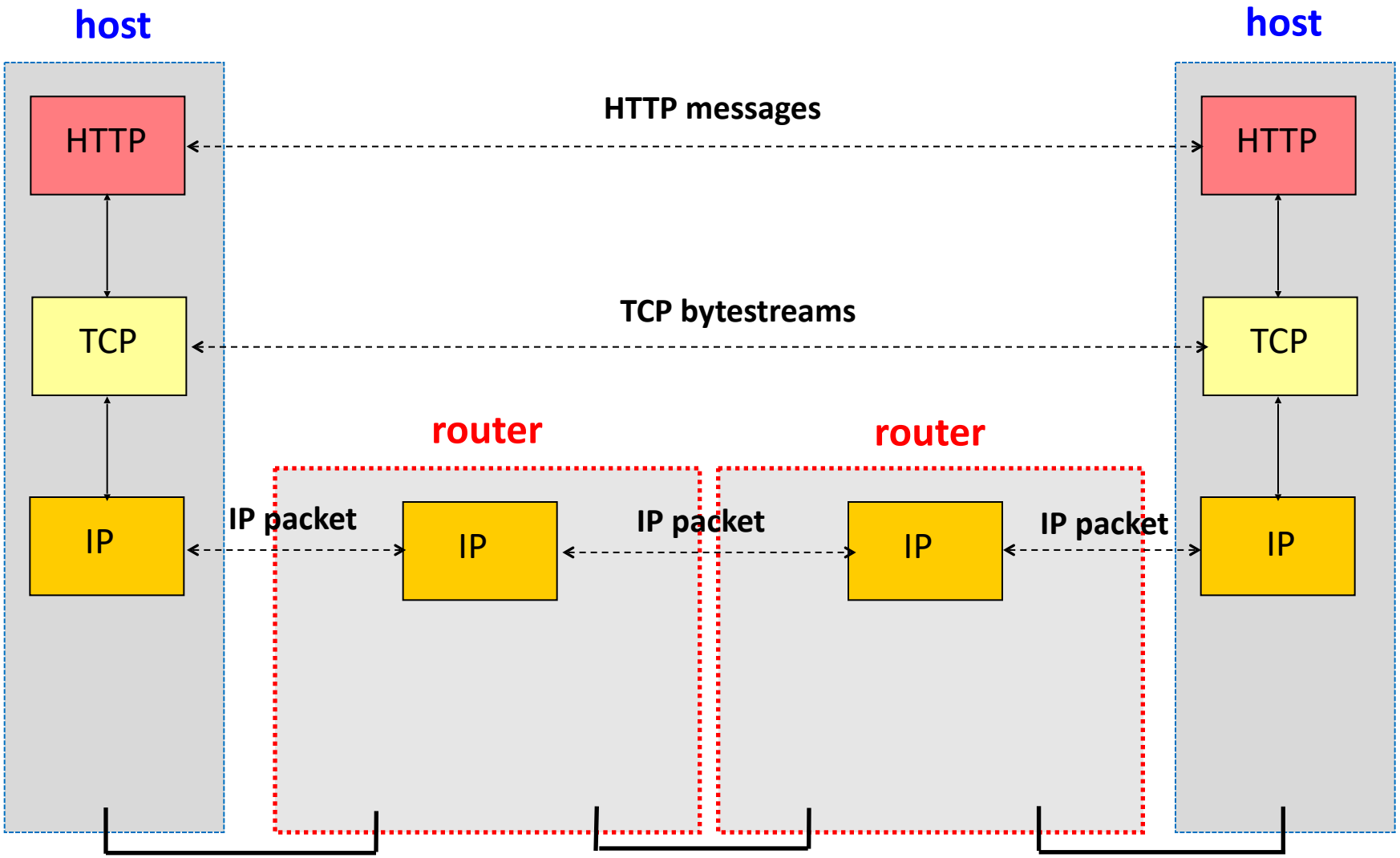
Complicated protocol diagram



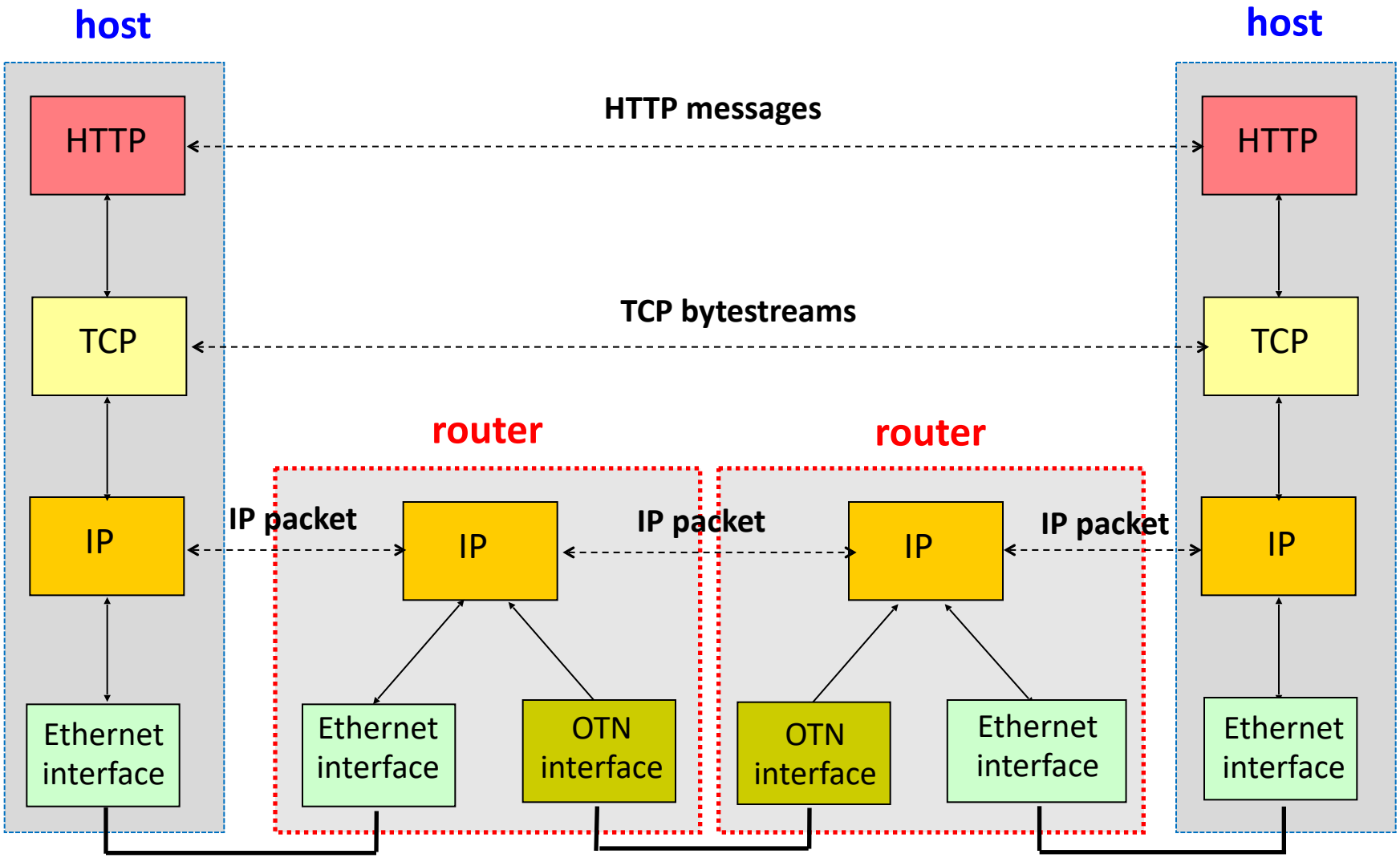
Complicated protocol diagram



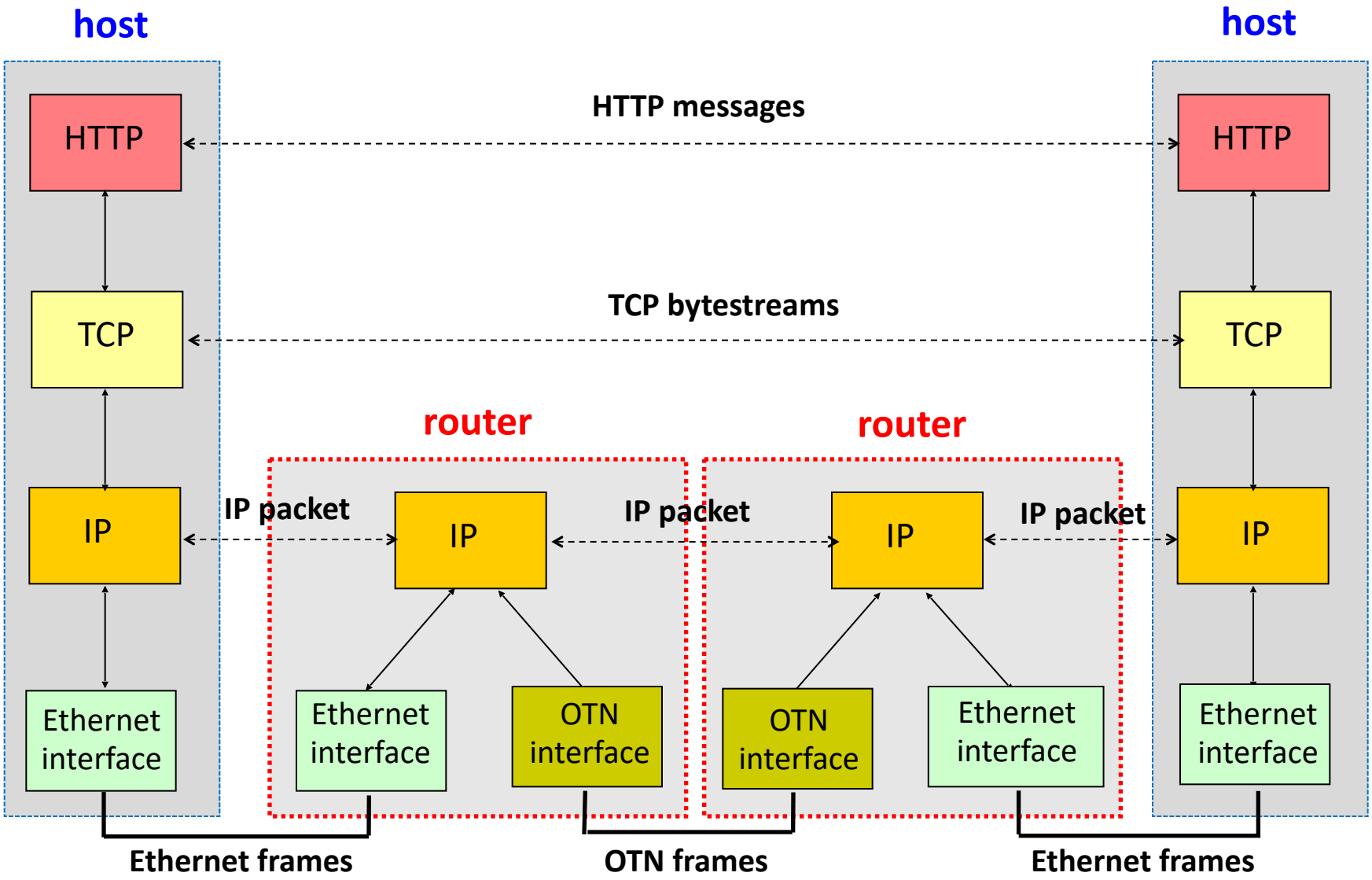
Complicated protocol diagram



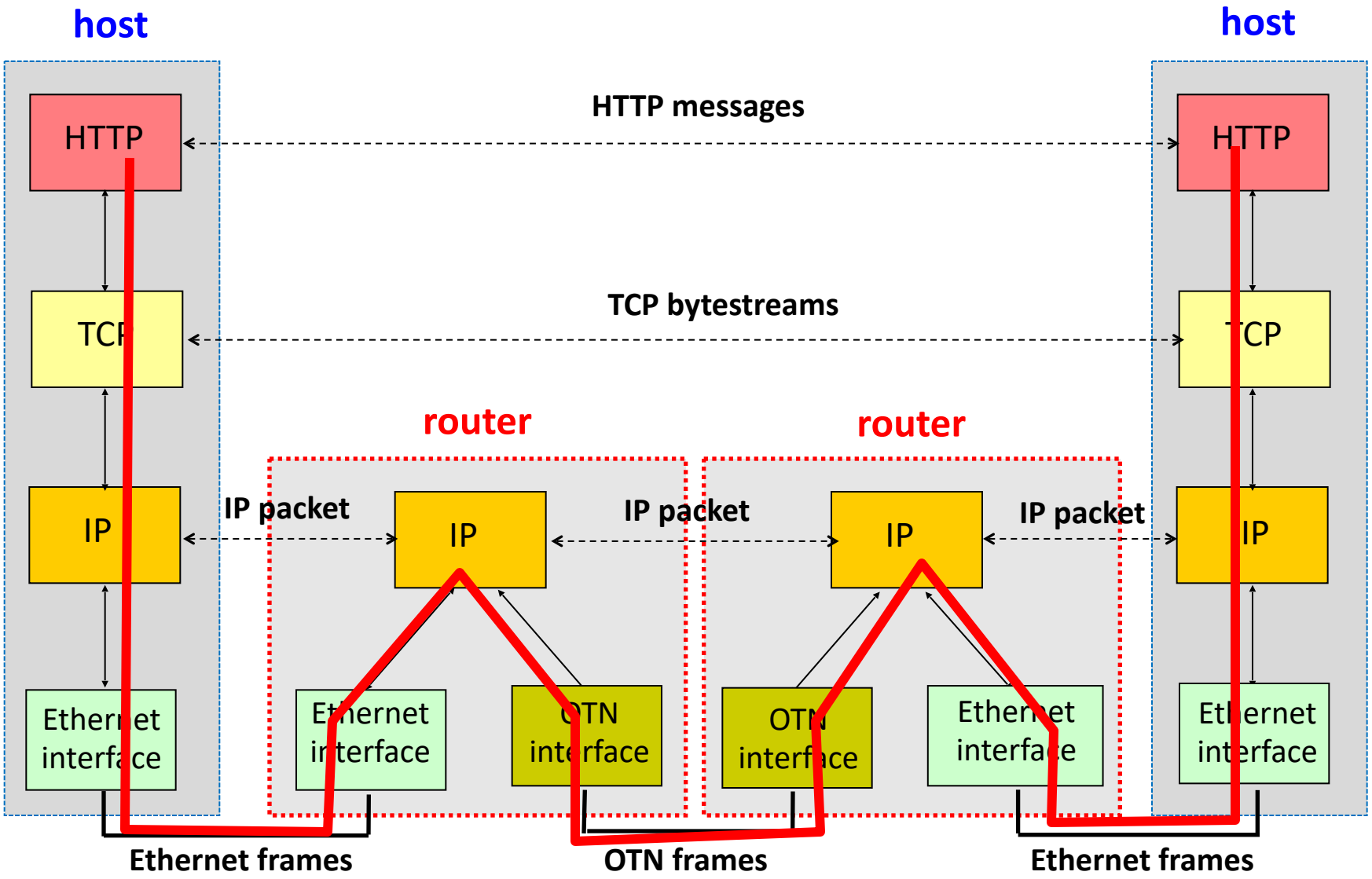
Complicated protocol diagram



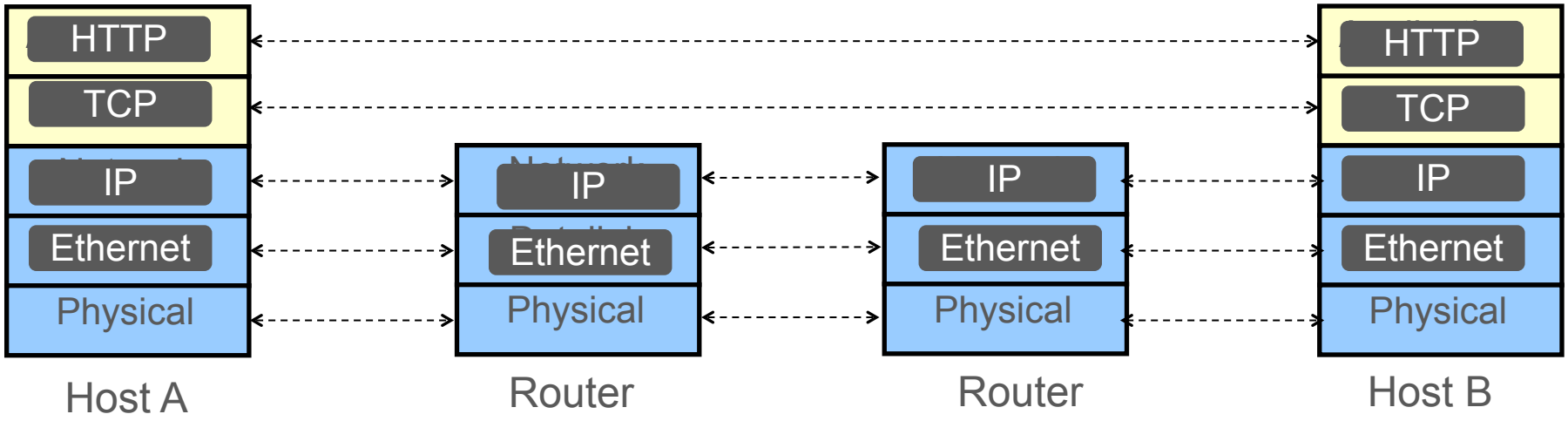
Complicated protocol diagram



Complicated protocol diagram

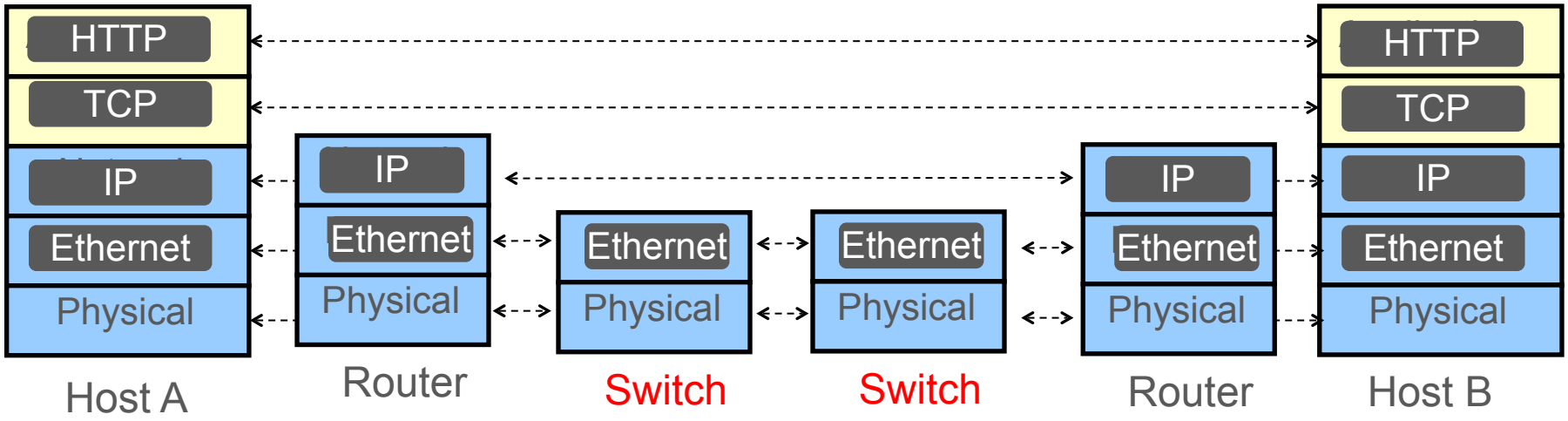


A side note: switches vs. routers

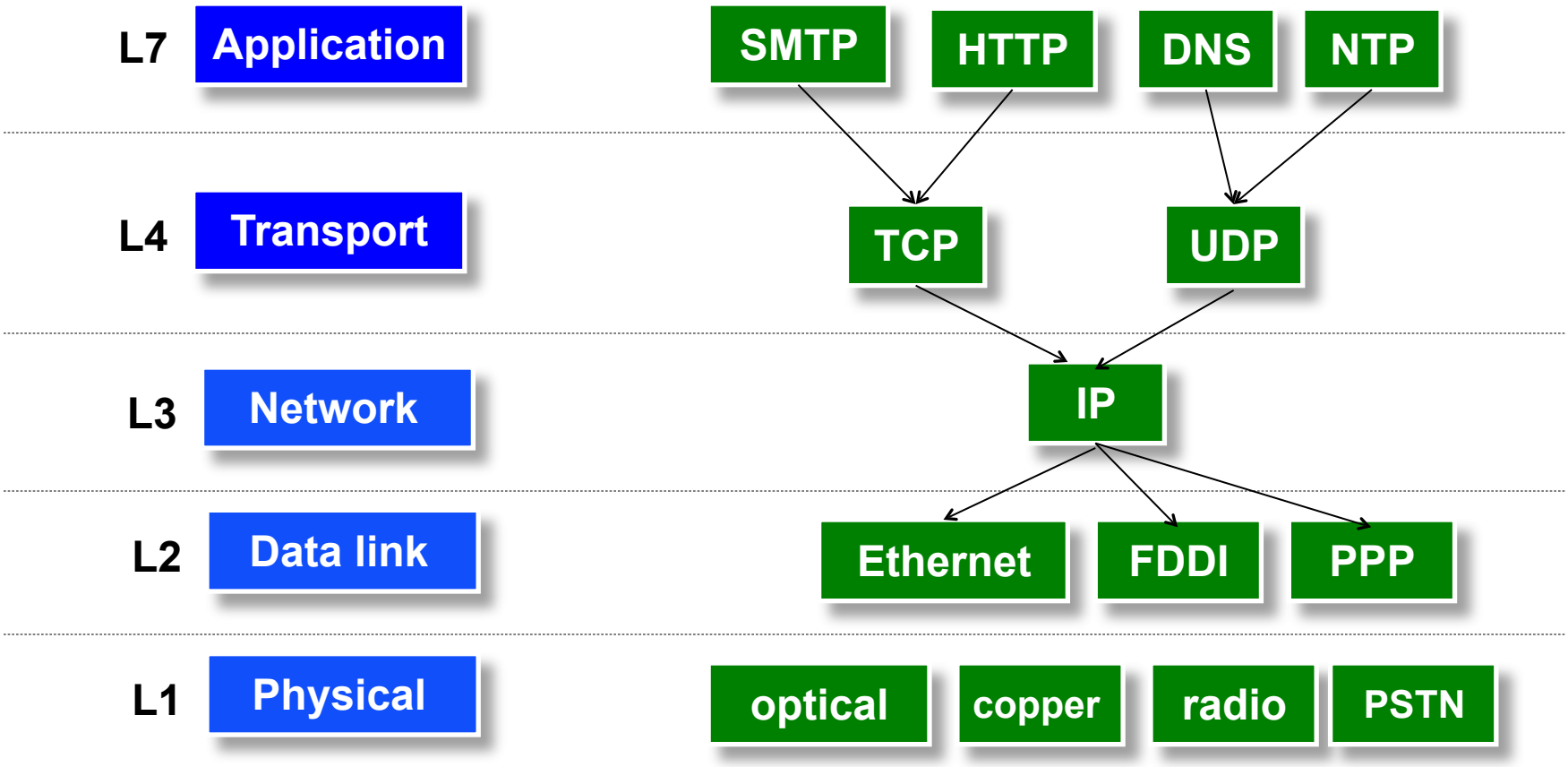


A side note: switches vs. routers

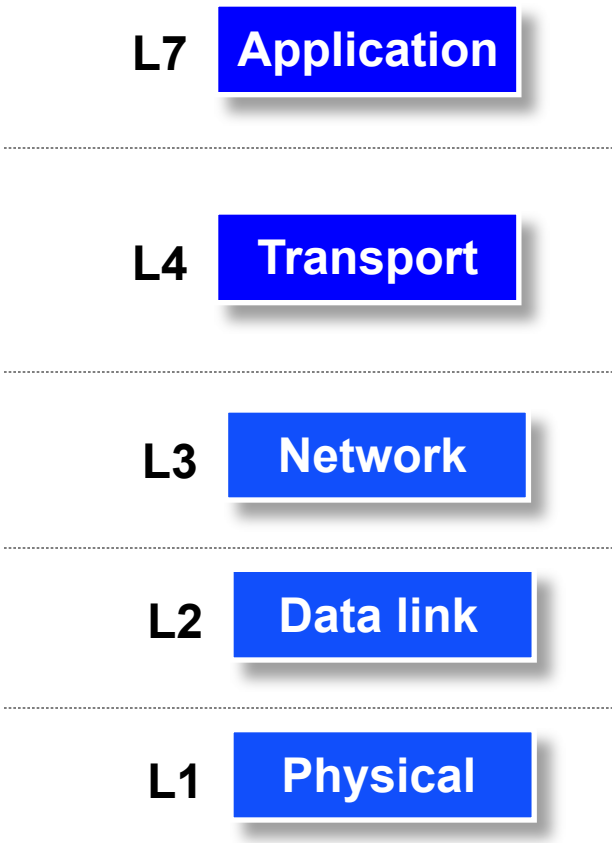
- Historically: switches implemented L1, L2 and routers L1, L2, L3
- These days, most switches also implement L3 hence we use the term switches and router interchangeably



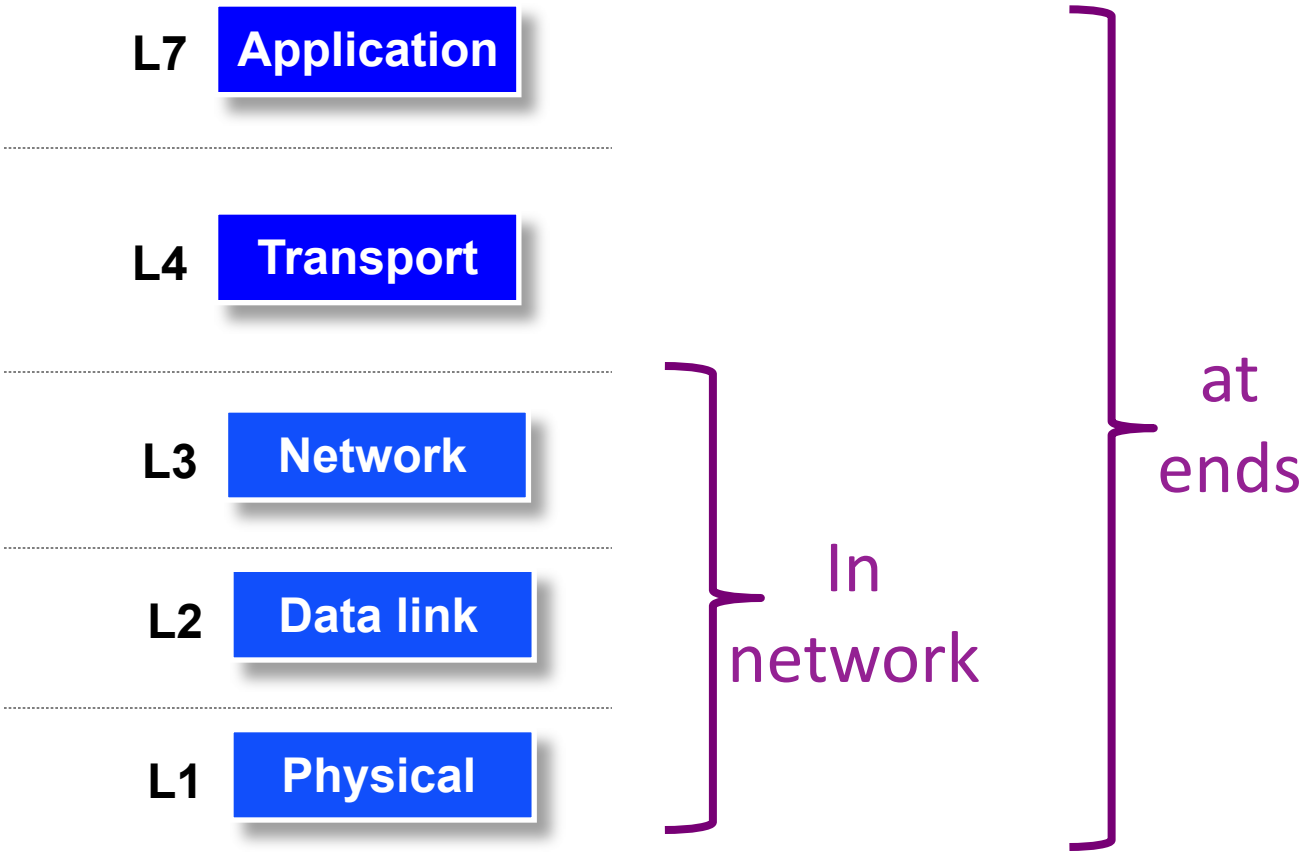
Review


L7
Application
L4
Transport
L3
Network
L2
Data link
L1
Physical
SMTP
HTTP
DNS
NTP
TCP
UDP
IP
Ethernet
FDDI
PPP
optical
copper
radio
PSTN

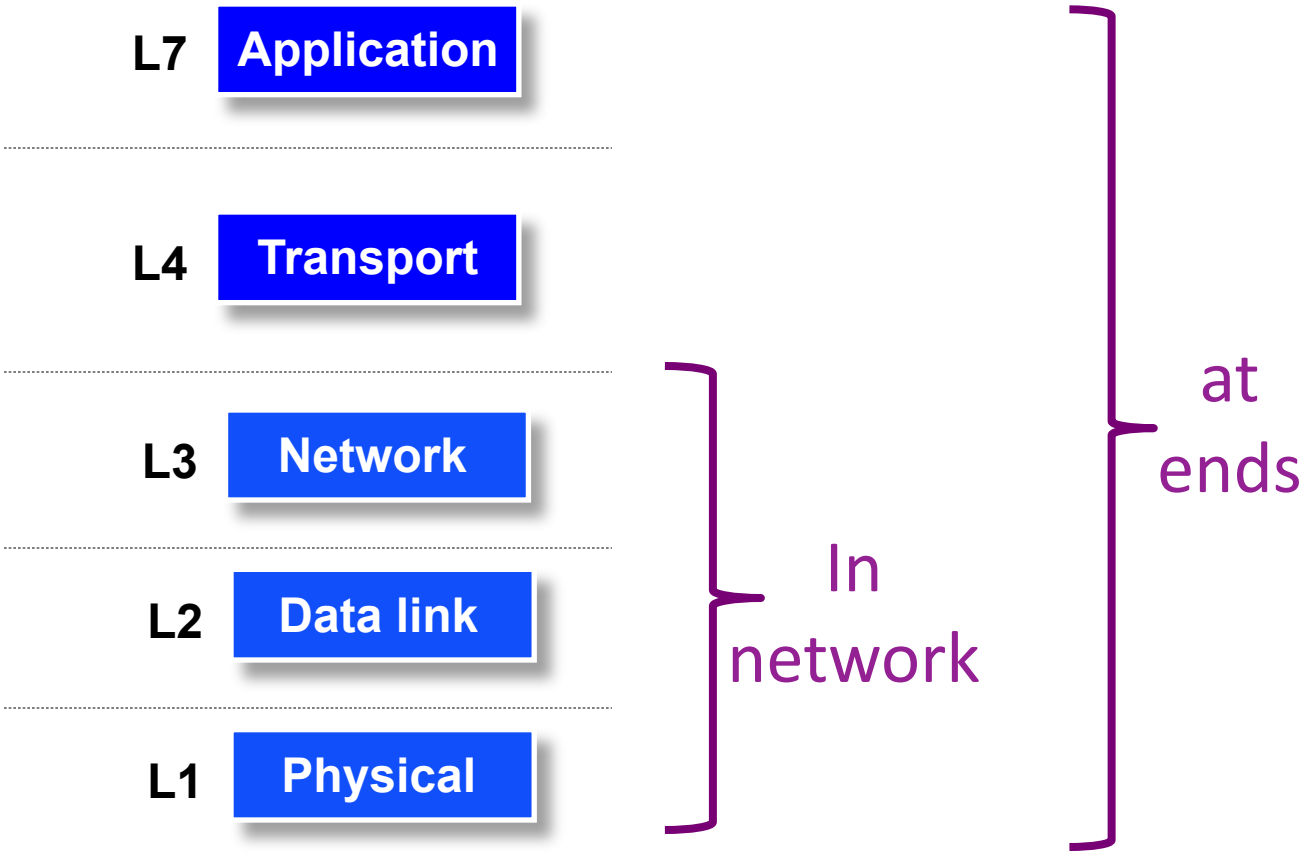
Review


L7
Application
L4
Transport
L3
Network
L2
Data link
L1
Physical

Review


L7
Application
L4
Transport
L3
Network
L2
Data link
L1
Physical
In network
at ends

Review



Next lecture: why is *this* a good assignment of tasks?