Layering

CS 438: Spring 2014

Instructor: Matthew Caesar

http://www.cs.illinois.edu/~caesar/cs438

Outline

Last time: low-level plumbing

Today: top-down architecting of the Internet

- Goals
- Layering
- Protocols
- The end-to-end principle

Recall from Lecture #1

- Architecture is not the implementation itself
- Architecture is how we structure implementations
  - what functions?, where?, what interfaces?
- Architecture is the modular design of the network

How would you go about designing the Internet?

Sit down and…

- List your goals
- Prioritize them
- Hence define the service you will offer
- Architect a solution that implements the service

Of course, the original designers of the Internet didn’t do anything of the sort…

Reality

- The lessons accrued over time; many contributors
  - 1961: packet switching (Baran and Kleinrock)
  - 1967: vision of a robust network (ARPANET)
  - 1972: “best effort inter-networking” proposed (Kahn)
  - 1974: TCP/IP paper (Cerf/Kahn)

Reality

- The lessons accrued over time; many contributors
- Many of the lessons were learnt “on the job”
  - E.g., TCP’s congestion control algorithms were developed in response to the Internet meltdowns of the early 1980s

Reality

- The lessons accrued over time; many contributors
- Many of the lessons were learnt "on the job"
- Consensus didn't come easy
  - 1961: packet switching is proposed
  - 1972: best-effort communication is advocated
  - 1980: IP adopted as the defense standard
  - 1985: NSFnet picks IP
  - 199x: Circuit switching rises (and falls) in the form of ATM
  - 199x: `Quality of Service' (QoS) rises and falls

Reality

- The lessons accrued over time; many contributors
- Many of the lessons were learnt “on the job”
- Consensus didn’t come easy
- And progress was ad-hoc
    - *“rough consensus and running code.”*

Reality

- The lessons accrued over time; many contributors
- Many of the lessons were learnt “on the job”
- Consensus didn’t come easy
- And progress was ad-hoc
- Yet, there was also
  - constant dialogue
  - constant introspection
  - constant experimentation, leading to…
- A strong consistency of vision emerging by the ‘80s, driven by D. Clark, chair of the Internet Arch. Board

Internet Design Goals
(from Clark's SIGCOMM 1988 paper)

- Connect existing networks
- Robust in face of failures
- Support multiple types of delivery services
- Accommodate a variety of networks
- Allow distributed management
- Cost effective
- Easy host attachment
- Allow resource accountability

Connect Existing Networks

- Wanted a single unifying interface that could be used to connect any pair of (existing) networks

- Interface to be compatible with existing networks
  - couldn't demand performance capabilities not supported by existing networks
  - had to support existing packet switched networks

- Led to focus on an inter-networking service based on the best-effort delivery of packets

How would you go about designing the Internet?

Sit down and…

- List your goals
- Prioritize them
- Hence define the service you will offer
- Architect a solution that implements the service

Three steps

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

Decomposition

What does it take to send packets across the globe?

- Bits on wire
- Packets on wire
- Delivery packets within a single physical network
- Deliver packets across multiple networks
- Ensure the destination received the data
- Do something with the data

This is decomposition…

Now, how do we organize these tasks?

Inspiration…

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

Dear John,

- Aide:
  - Puts letter in envelope with CEO B's full name
  - Takes to FedEx

Your days are numbered.

- FedEx Office
  - Puts letter in larger envelope

--Pat

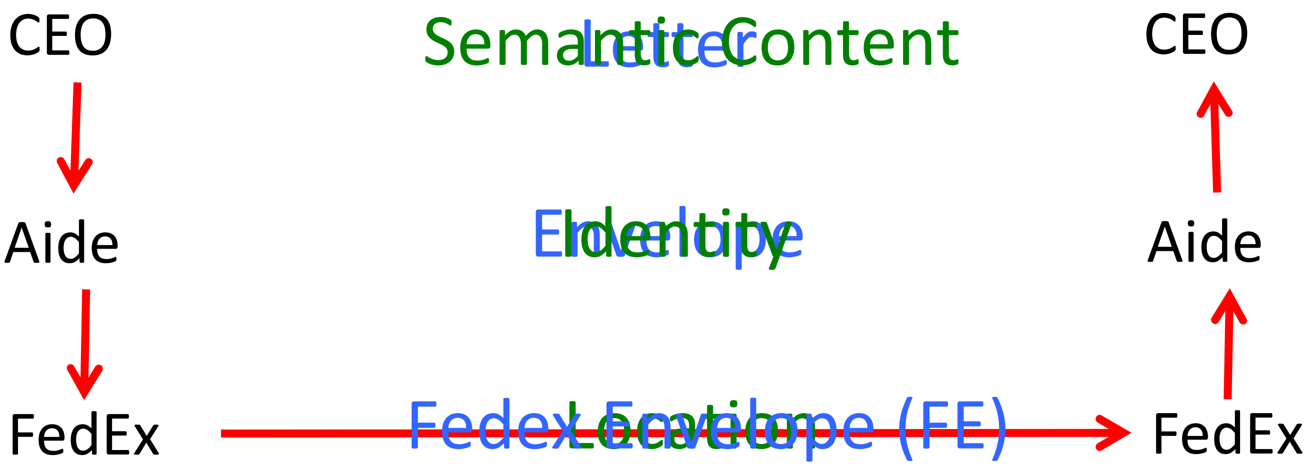
  - Puts name and street address on FedEx envelope
  - Puts package on FedEx delivery truck
- FedEx delivers to other company

The Path of the Letter

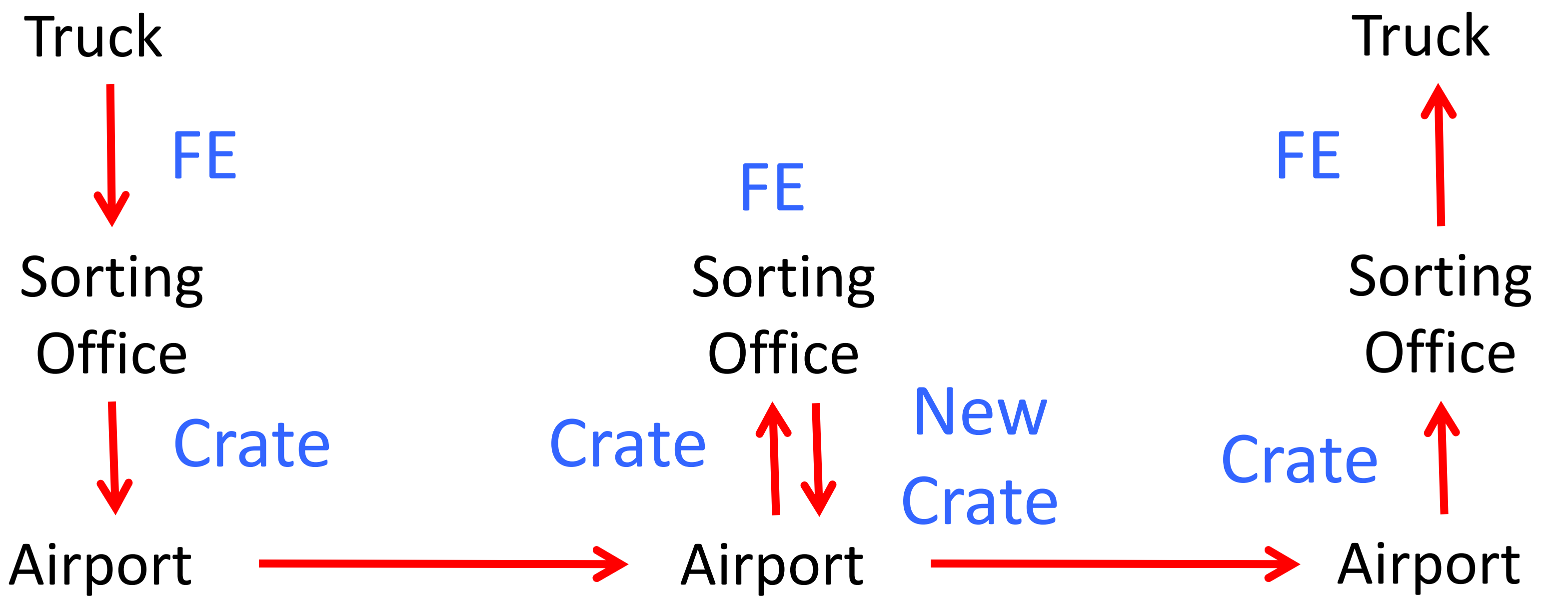
"Peers" on each side understand the same things

No one else needs to

Lowest level has most packaging



The Path Through FedEx



Deepest Packaging (Envelope+FE+Crate)
at the Lowest Level of Transport

The Path Through FedEx

Higher “Stack”
at Ends

Highest Level of
“Transit Stack”
is Routing

Partial “Stack”
During Transit

Truck

FE

Sorting
Office

Crate

Airport

FE

Sorting
Office

Crate

New
Crate

Airport

Truck

FE

Sorting
Office

Crate

Airport

Deepest Packaging (Envelope+FE+Crate)
at the Lowest Level of Transport

In the context of the Internet

Applications

...built on...

Reliable (or unreliable) transport

...built on...

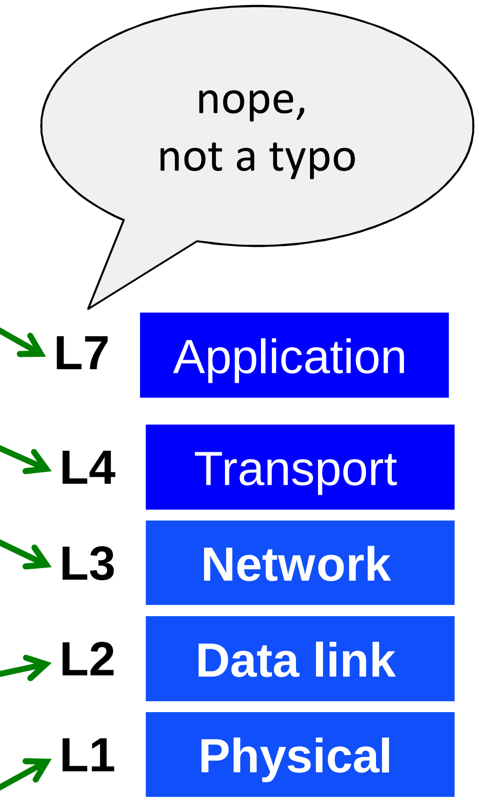
Best-effort global packet delivery

...built on...

Best-effort local packet delivery

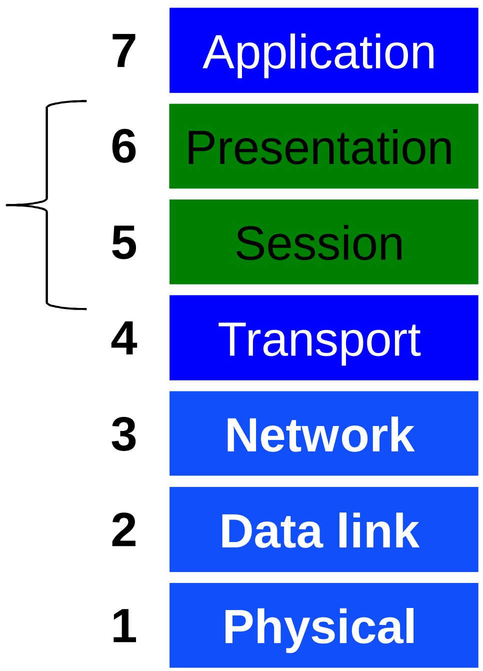
...built on...

Physical transfer of bits



In the context of the Internet

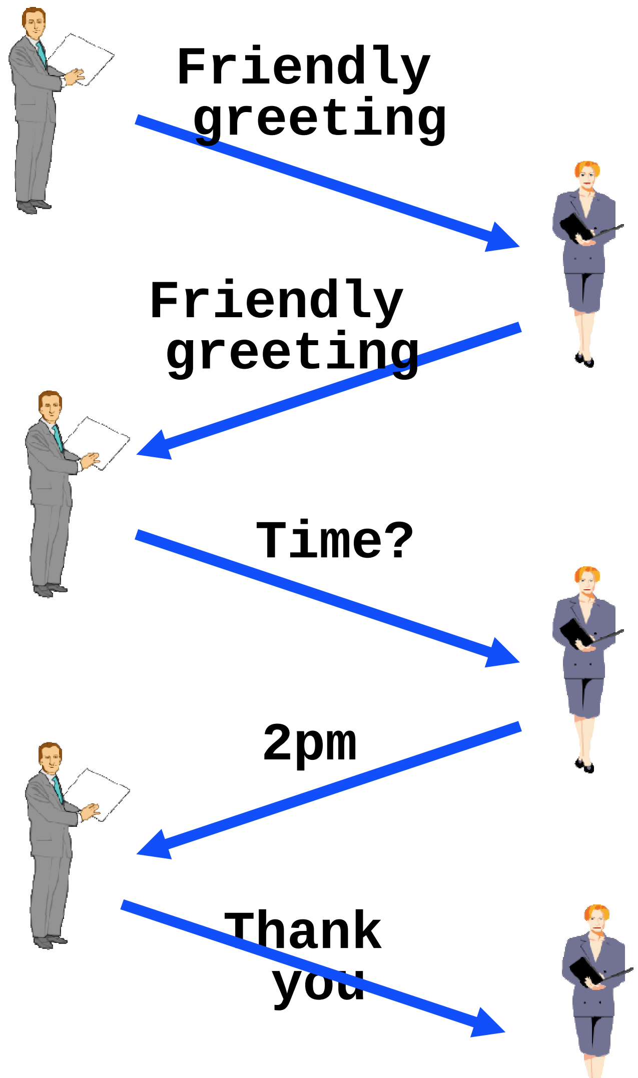
The Open Systems Interconnect (OSI) model developed by the ISO included two additional layers that are often implemented as part of the application



Protocols and Layers

Communication between peer layers on different systems is defined by protocols

What is a Protocol?



What is a Protocol?

- E.g., the destination address is in the 1st four bytes of the packet

- When A sends B a packet of type X…

  …B should return a packet of type Y to A

  … then A should respond with Z

  ….

What is a Protocol?

- An agreement between parties on how to communicate

- Include syntax and semantics
  - how a communication is specified and structured
  - what a communication means

- Protocols exist at many hardware, software, *all* levels!

- Defined by a variety of standards bodies
  - IETF (ietf.org), IEEE, ITU, …

So we have decomposition and organization

Next: what gets implemented where?

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 machines
  - Hosts
  - Routers (switches)
- What gets implemented where?

What gets implemented at the end host

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

- Therefore, all layers must exist at host!

What gets implemented in the network?

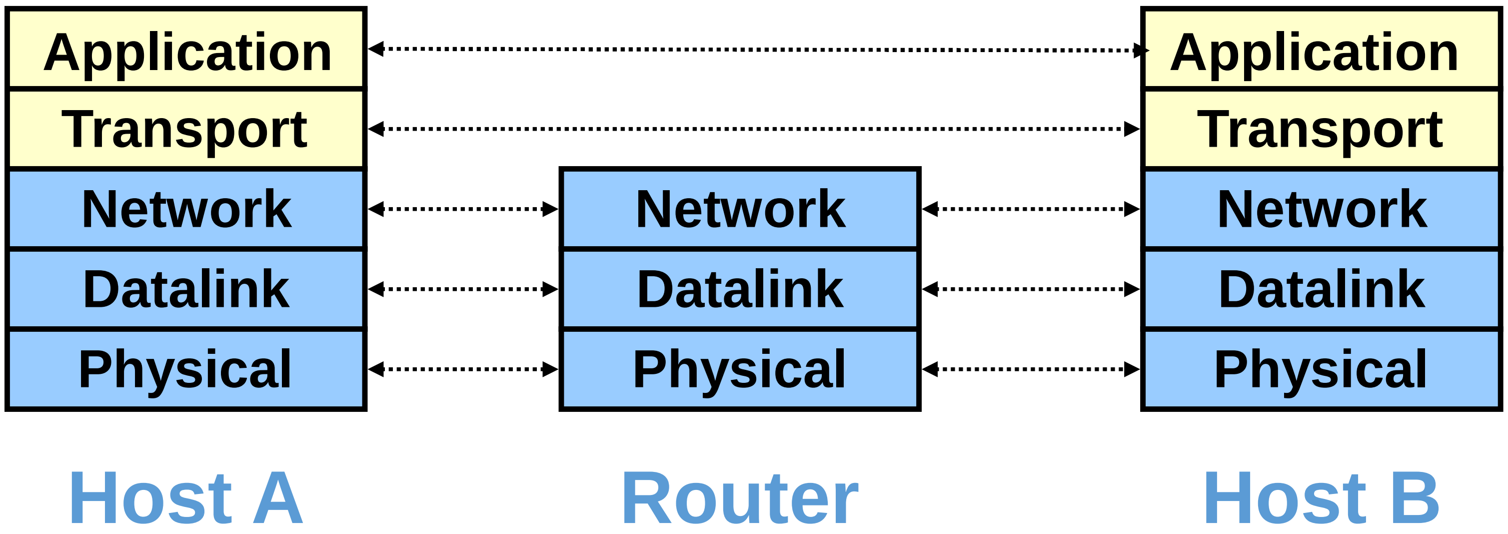
- Bits arrive on wire
  - Physical layer necessary
- Packets must be delivered to next-hop and across local networks
  - Datalink layer necessary
- Packets must be delivered between networks for global delivery
  - Network layer necessary
- The network doesn’t support reliable delivery
  - Transport layer (and above) ***<u>not</u>*** supported

Switches *vs.* Routers

- Switches do what routers do, except they don't participate in global delivery, just local delivery
- Switches only need to support Physical and Datalink
  - Don't need to support Network layer
- Routers support Physical, Datalink and Network layers
- 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

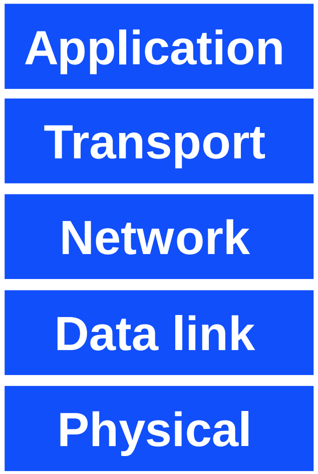
Simple diagram

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



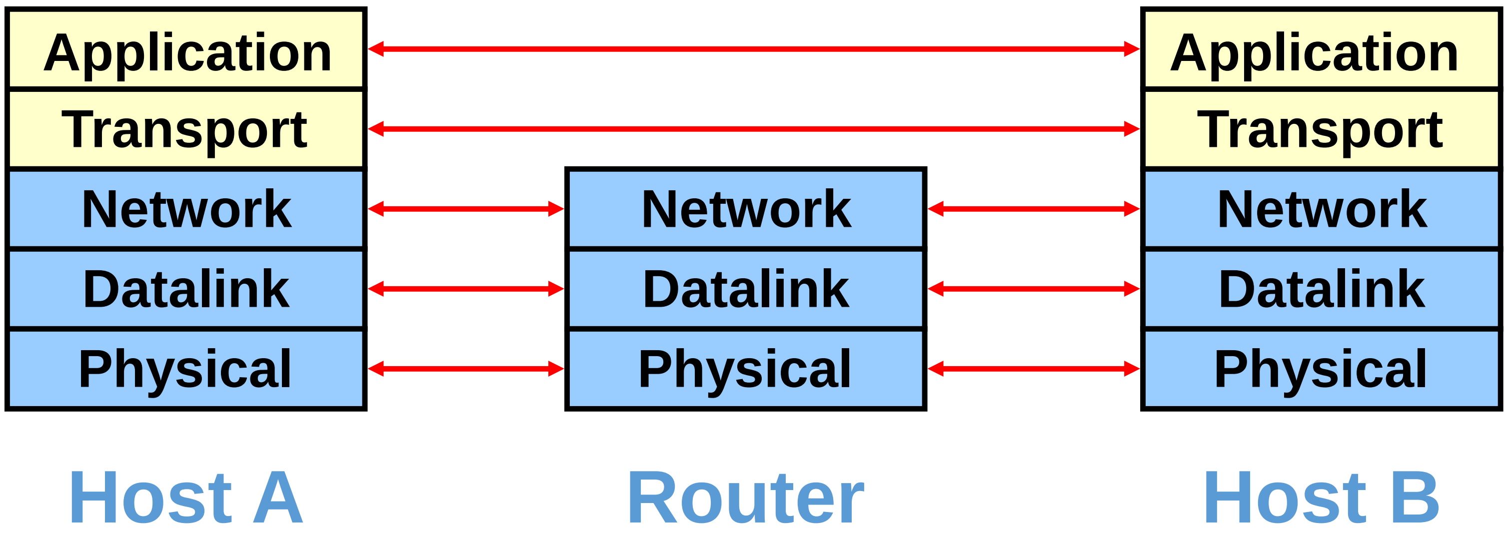
Looking a little closer

- At the end host
- Application
  - user space: web server, browser, mail, game
- Transport and network.
  - Typically part of the operating system
- Datalink
  - Often written by vendor of the network interface hardware
- Physical
  - Hardware: network interface card and link



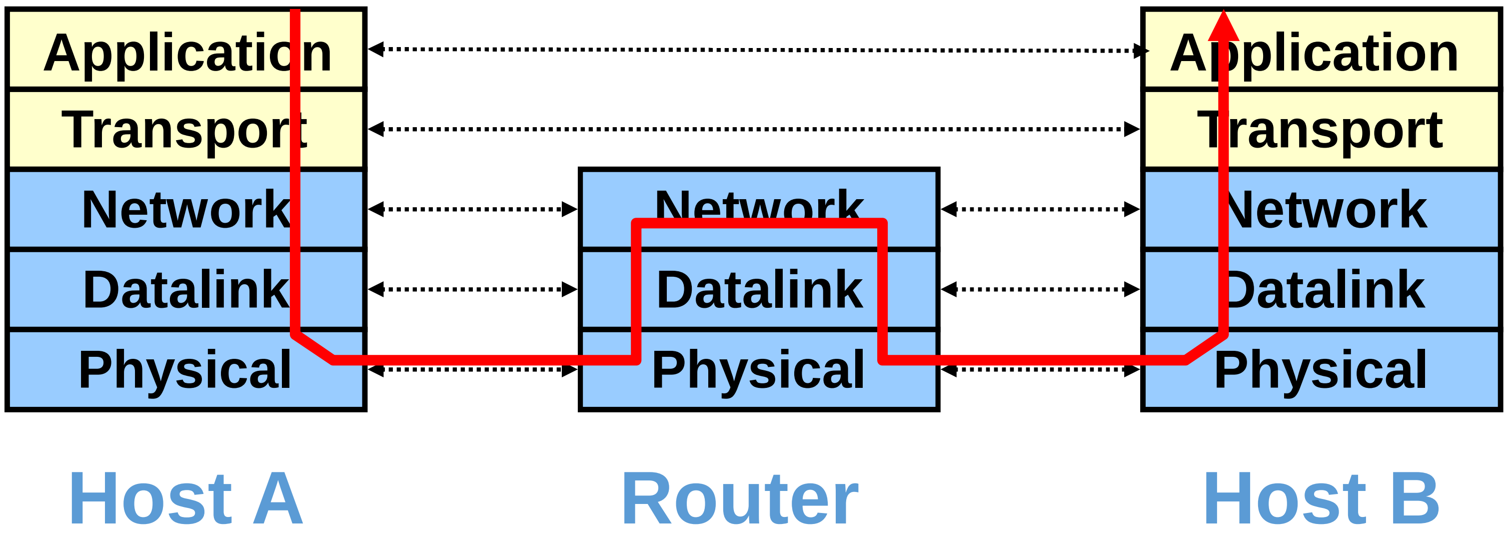
Logical Communication

- Layers interacts with peer's corresponding layer

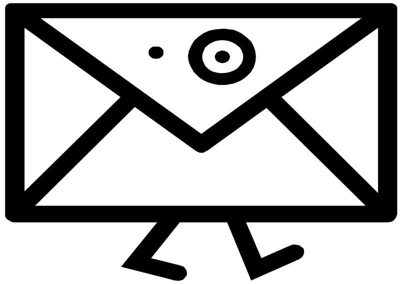


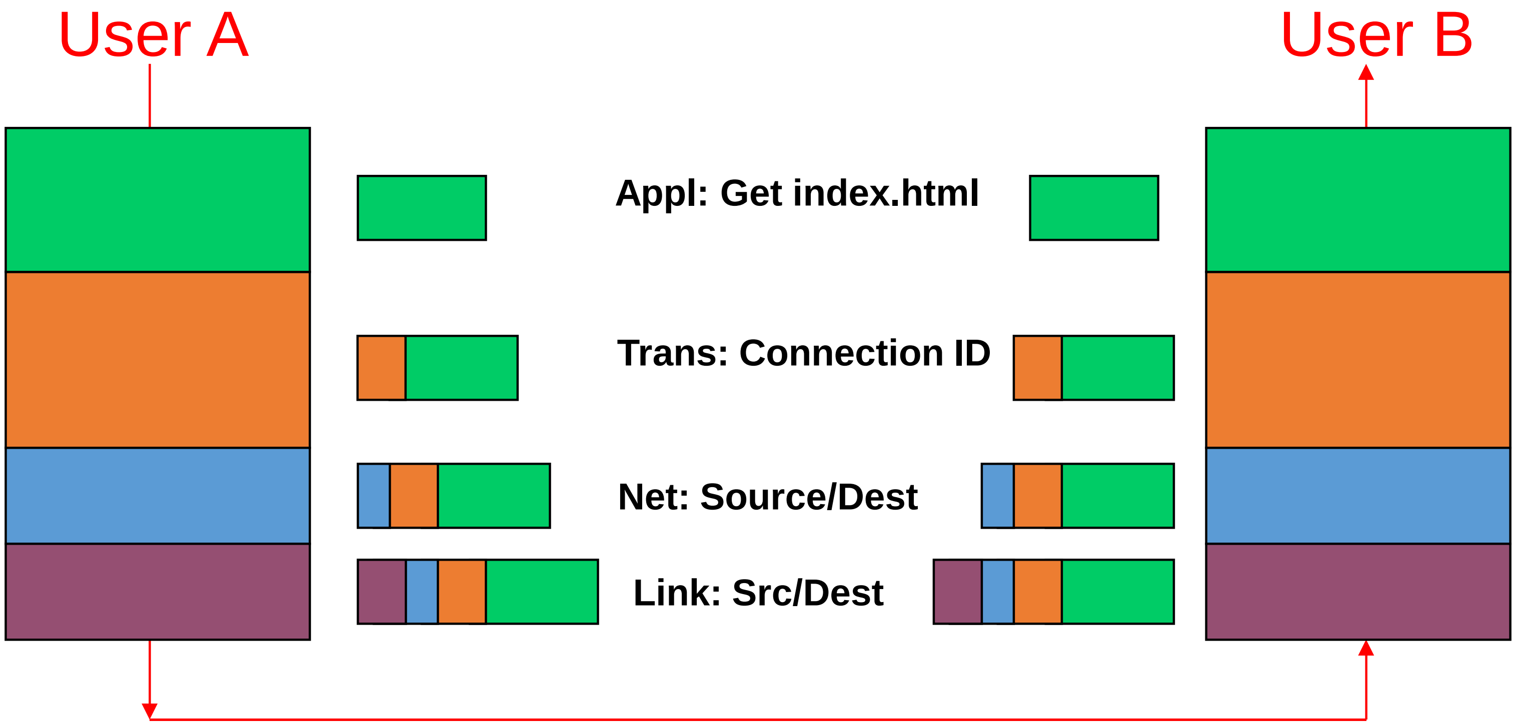
Physical Communication

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

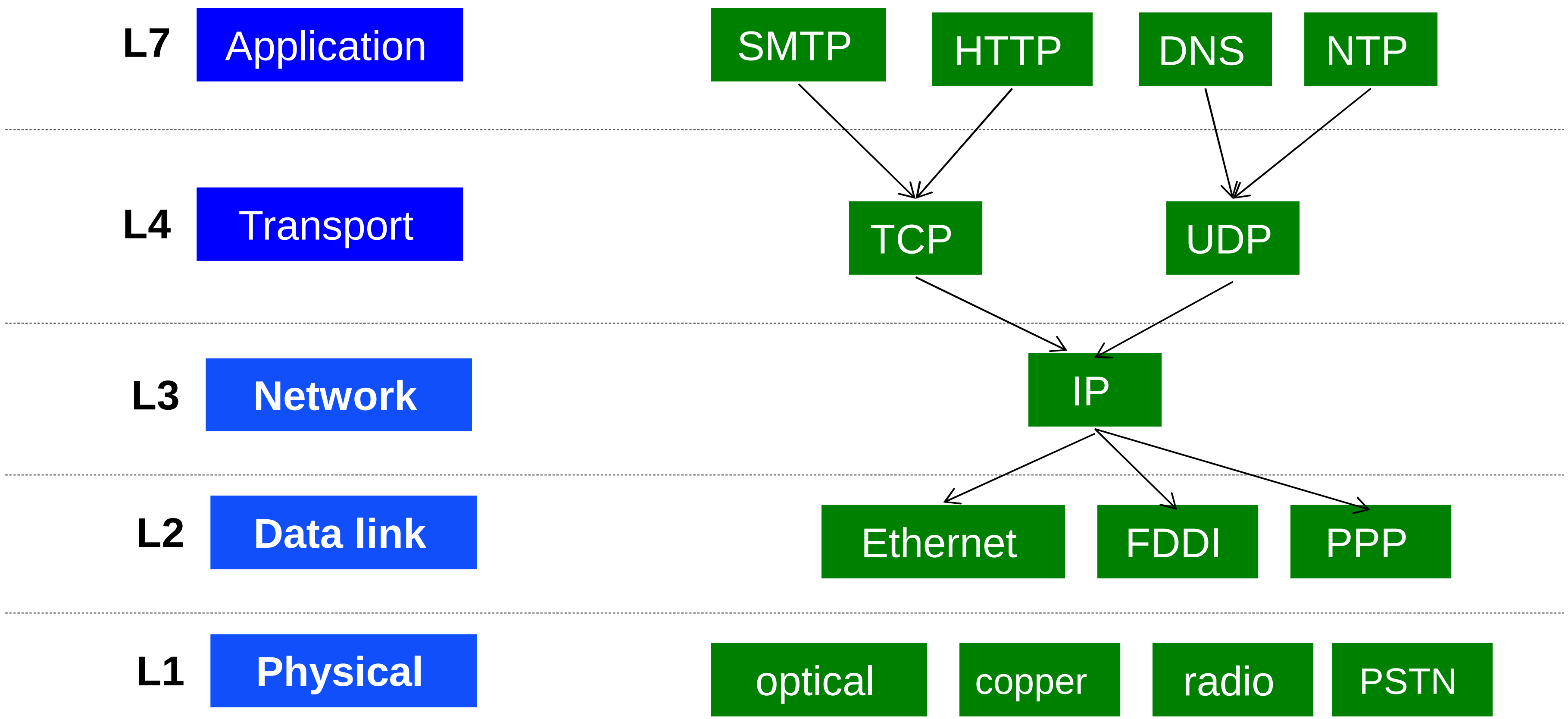


Layer Encapsulation





Protocols at different layers



There is just one network-layer protocol, IP

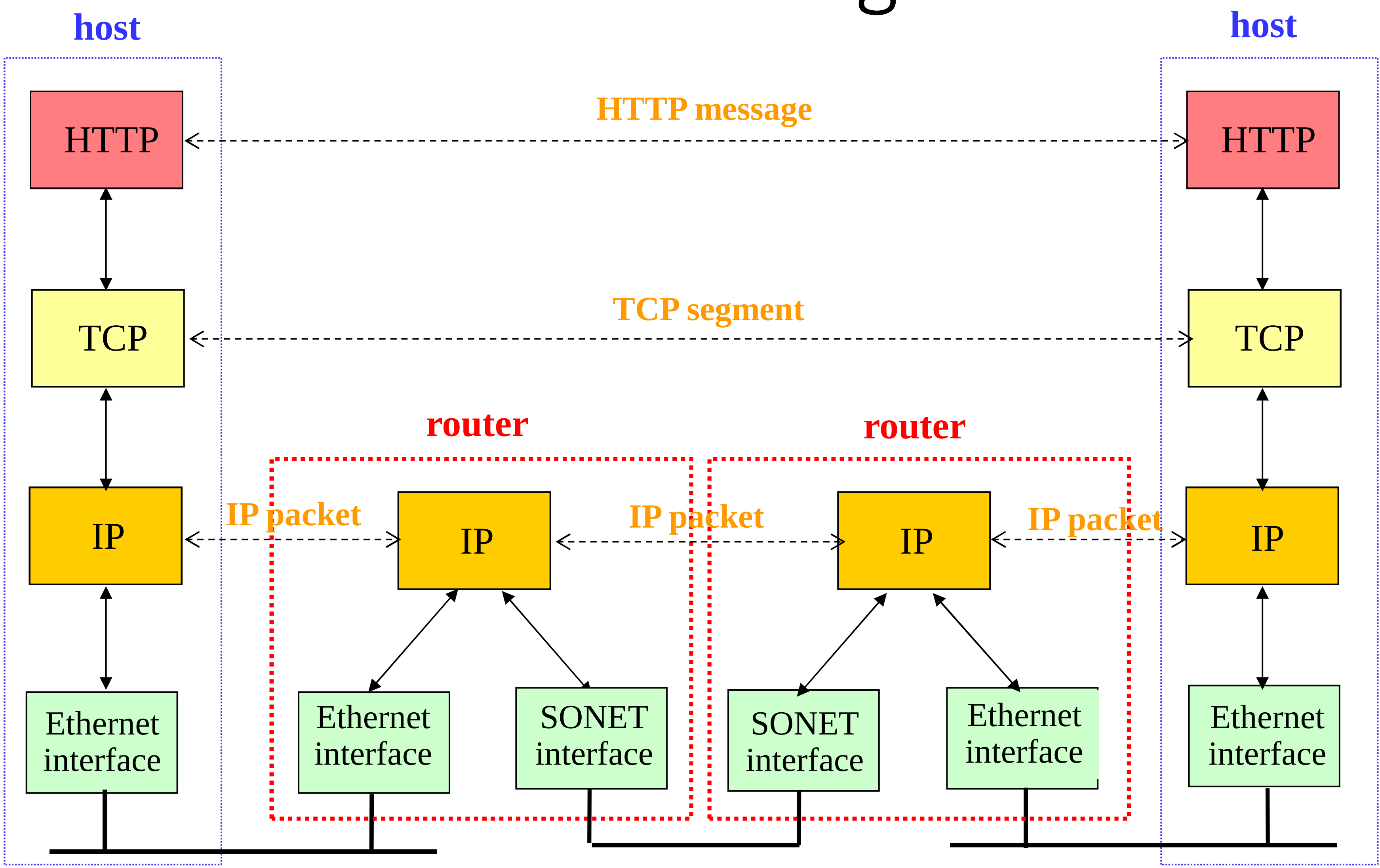
The “narrow waist” of the Internet hourglass

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 to function on *all* networks
- Supports simultaneous innovations above and below IP
- But changing IP itself is hard (e.g., IPv4 → IPv6)

A Protocol-Centric Diagram



What are some of the benefits of protocols and layering?

Interoperability

- Many implementations of many technologies
  - Hosts running FreeBSD, Linux, Windows, MacOS, …
  - People using Mozilla, Explorer, Opera, …
  - Routers made by cisco, juniper, …
  - Hardware made by IBM, Dell, Apple, …
- And it changes all the time.
- Phew!

But they can all talk together because they use the same protocol(s)

Abstraction & Reuse

- Multiple choices of protocol at many layers
  - Physical: copper, fiber, air, carrier pigeon
  - Link: ethernet, token ring, SONET, FDDI
  - Transport: TCP, UDP, SCTP

- But we don't want to have to write "a web (HTTP) browser for TCP networks running IP over Ethernet on Copper" and another for the fiber version…
  - Protocols provide a standard interface to write to
  - Layers hide the details of the protocols below

Decoupling aids innovation

- Technologies at each layer pursued by very different communities
- Innovation at each layer can proceed in parallel

What are some of the drawbacks of protocols and layering?

Drawbacks of Layering

- Layer N may duplicate lower layer functionality
  - e.g., error recovery to retransmit lost data
- Information hiding may hurt performance
  - e.g., packet loss due to corruption vs. congestion
- Headers start to get really big
  - e.g., typical TCP+IP+Ethernet is 54 bytes
- Layer violations when the gains too great to resist
  - e.g., TCP-over-wireless
- Layer violations when network doesn’t trust ends
  - e.g., firewalls

Where to place 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

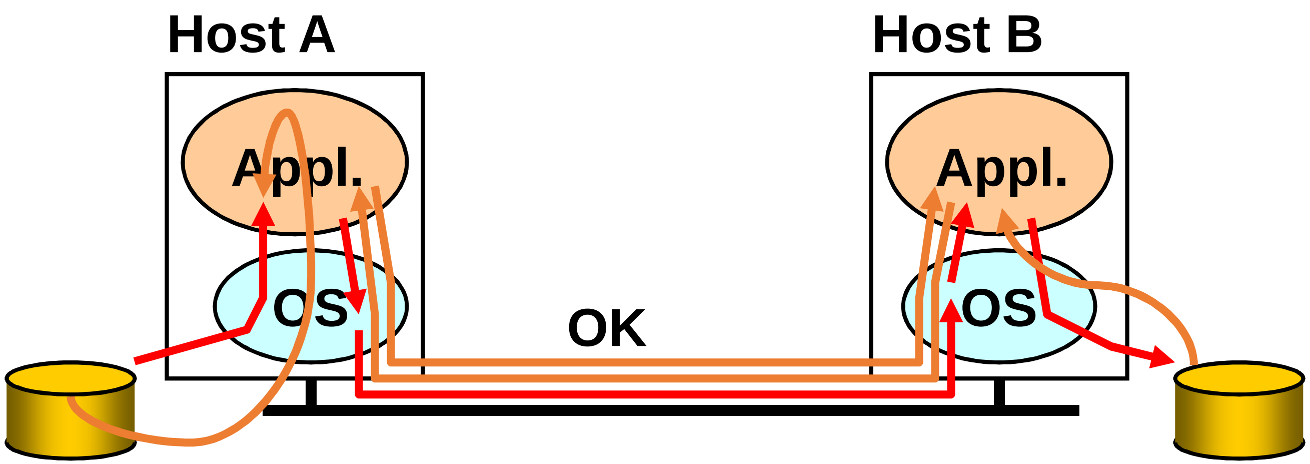
Basic Observation

- Some application requirements can only be correctly implemented **end-to-end**
  - reliability, security, *etc.*

- Implementing these in the network is hard
  - every step along the way must be fail proof

- Hosts
  - **Can** satisfy the requirement without network's help
  - **Will/must** do so, since they can't rely on the network

Example: Reliable File Transfer



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

Discussion

- Solution 1 (make each step reliable) is incomplete
  - What happens if any network element misbehaves?
  - Receiver has to do the check anyway!
- Solution 2 (end to end check) 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?

Summary of End-to-End 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

“Only if sufficient” interpretation

- Don’t implement a function at the lower levels of the system unless it can be completely implemented at this level

- *Unless you can relieve the burden from hosts, don’t bother*

"Only if necessary" interpretation

- Don't implement *anything* in the network that can be implemented correctly by the hosts

- Make network layer absolutely minimal
  - This E2E interpretation trumps performance issues
  - Increases flexibility, since lower layers stay **simple**

"Only if useful" interpretation

- If hosts can implement functionality correctly, implement it in a lower layer only as a performance enhancement
- But do so only if it does not impose burden on applications that do not require that functionality

Taking stock of where we're at…

First Step: Basic Concepts and Decisions

- Plumbing: links, switches
- Packet Switching winner over circuit switching
- Best-effort service model

Second Step: Architectural Principles

- Protocols and Layering

- End-to-End Principle

Third Step: Design Challenges and Solutions

- Let’s go layer by layer
  - Physical
  - Datalink
  - Network
  - Transport
  - Application

Two Layers We'll Worry About Less

- Physical:
  - Technology dependent
  - Lots of possible solutions
  - Not specific to the Internet

- Application:
  - Application-dependent
  - Lots of possible solutions

Datalink and Network Layers

- Both support best-effort delivery
  - Datalink over local scope
  - Network over global scope
- Key challenge: scalable, robust **routing**
  - How do we address destinations
  - How to direct packets to destination

Transport Layer

- Provide reliable delivery over unreliable network