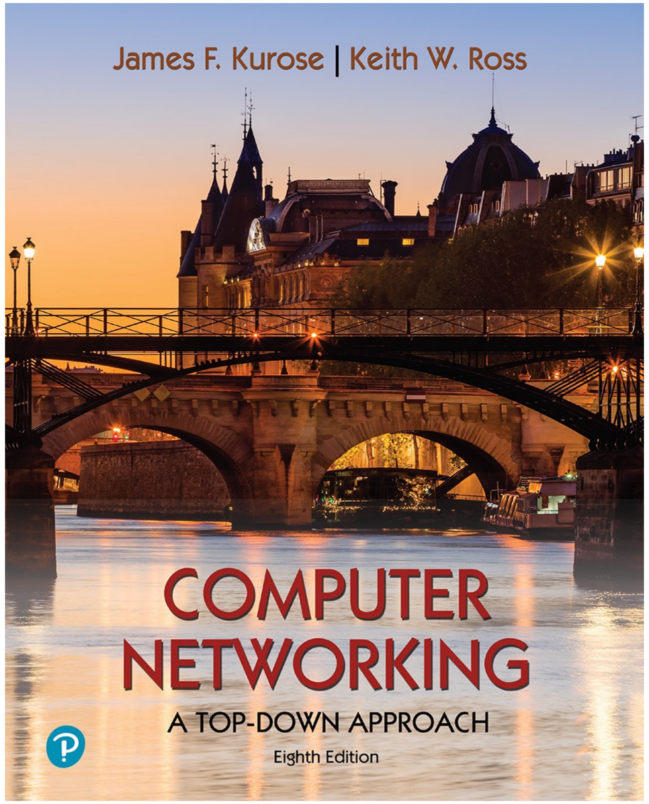
Chapter 1
Introduction

Yaxiong Xie

Department of Computer Science and Engineering
University at Buffalo, SUNY



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Adapted from the slides of the book's authors

Chapter 1: roadmap

- What *is* the Internet?
- What *is* a protocol?
- Network edge: hosts, access network, physical media
- Network core: packet/circuit switching, internet structure
- Performance: loss, delay, throughput
- Security
- Protocol layers, service models
- History

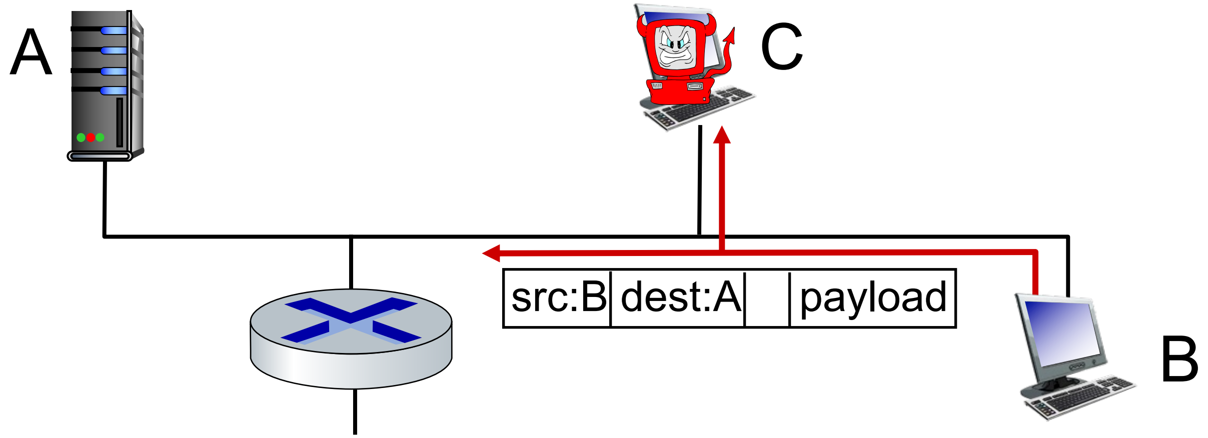
Network security

- Internet not originally designed with (much) security in mind
  - *original vision:* "a group of mutually trusting users attached to a transparent network" ☺
  - Internet protocol designers playing "catch-up"
  - security considerations in all layers!
- We now need to think about:
  - how bad guys can attack computer networks
  - how we can defend networks against attacks
  - how to design architectures that are immune to attacks

Bad guys: packet interception

packet "sniffing":

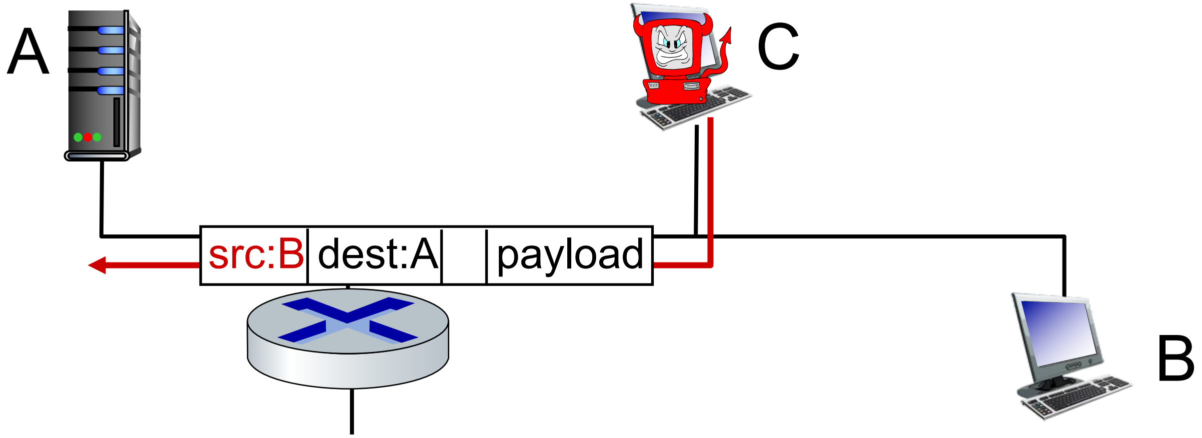
- broadcast media (shared Ethernet, wireless)
- promiscuous network interface reads/records all packets (e.g., including passwords!) passing by



Wireshark software used for our end-of-chapter labs is a (free) packet-sniffer

Bad guys: fake identity

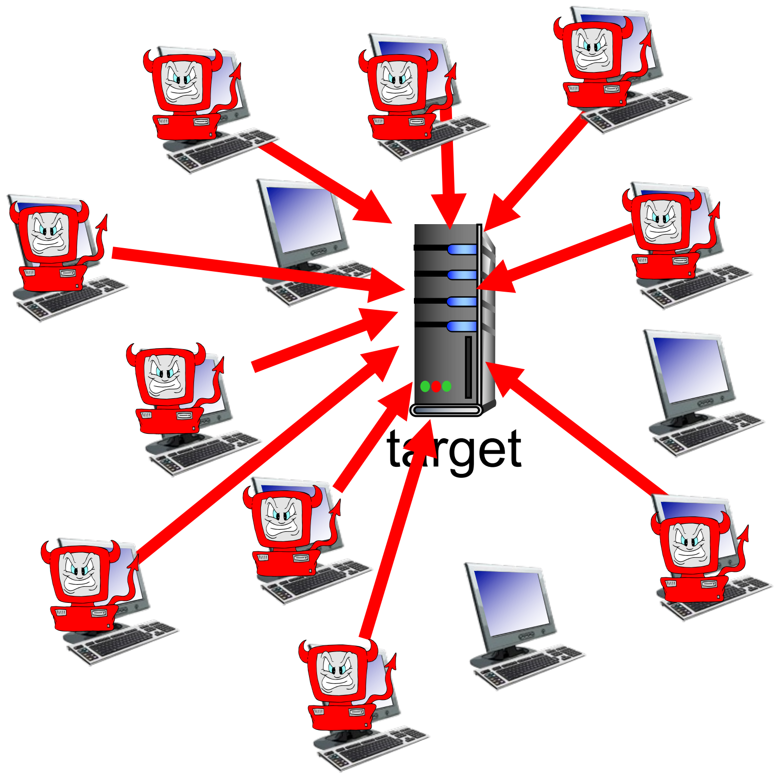
IP spoofing: injection of packet with false source address



Bad guys: denial of service

Denial of Service (DoS): attackers make resource (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic

1. select target
2. break into hosts around the network (see botnet)
3. send packets to target from compromised hosts



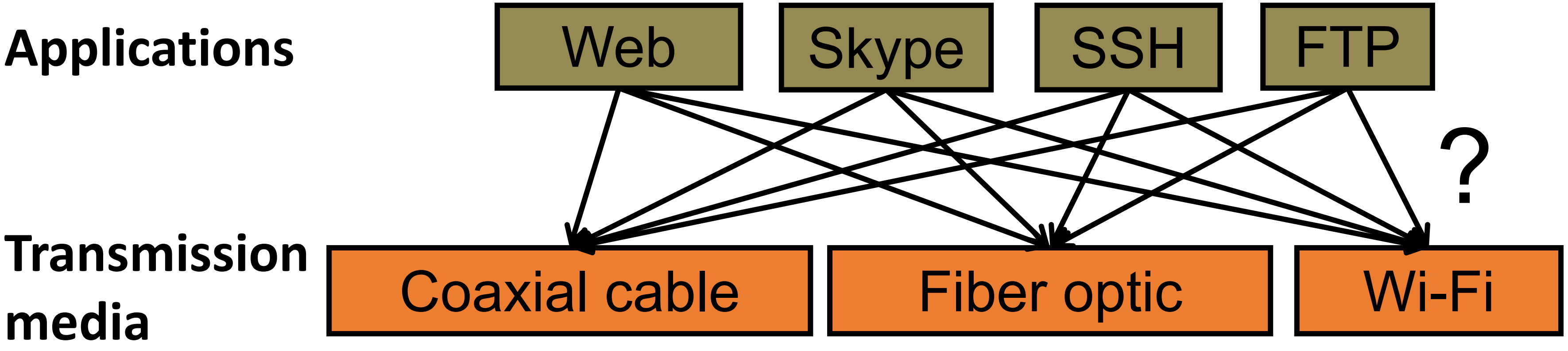
Lines of defense:

- authentication: proving you are who you say you are
  - cellular networks provides hardware identity via SIM card; no such hardware assist in traditional Internet
- confidentiality: via encryption
- integrity checks: digital signatures prevent/detect tampering
- access restrictions: password-protected VPNs
- firewalls: specialized “middleboxes” in access and core networks:
  - off-by-default: filter incoming packets to restrict senders, receivers, applications
  - detecting/reacting to DOS attacks

Chapter 1: roadmap

- What *is* the Internet?
- What *is* a protocol?
- Network edge: hosts, access network, physical media
- Network core: packet/circuit switching, internet structure
- Performance: loss, delay, throughput
- Security
- Protocol layers, service models
- History

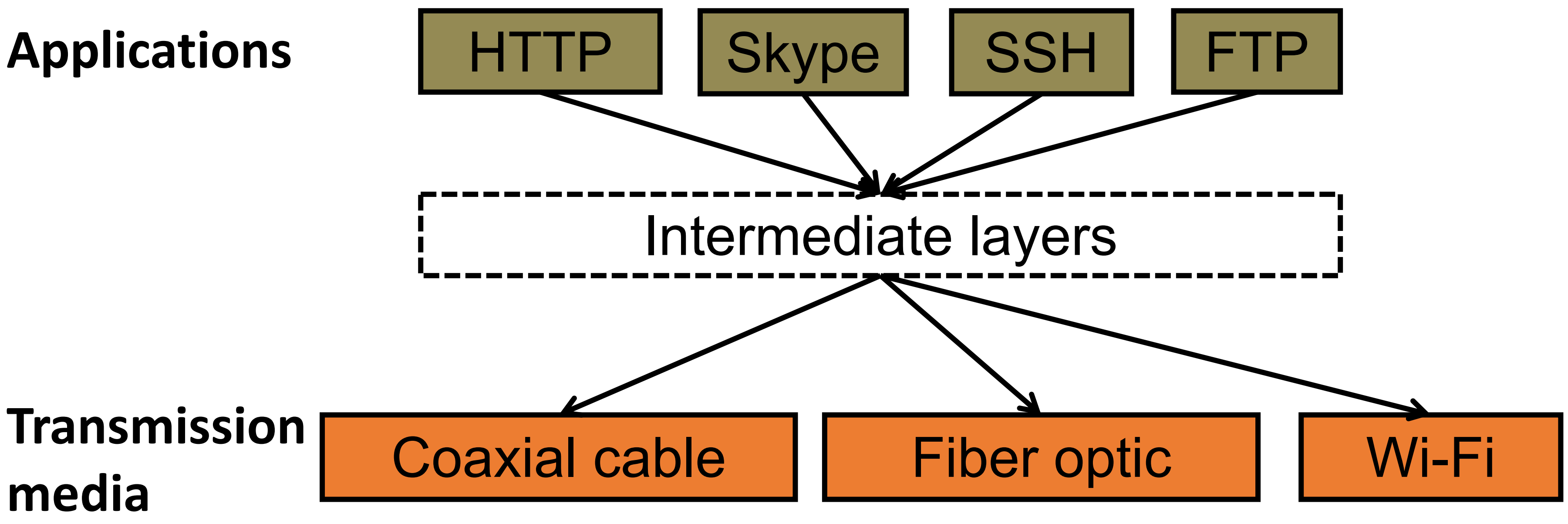
Layering: Motivation



- **Re-implement every application** for every new underlying transmission medium?
  - **Change** every application on any **change** to an underlying transmission medium (and vice-versa)?

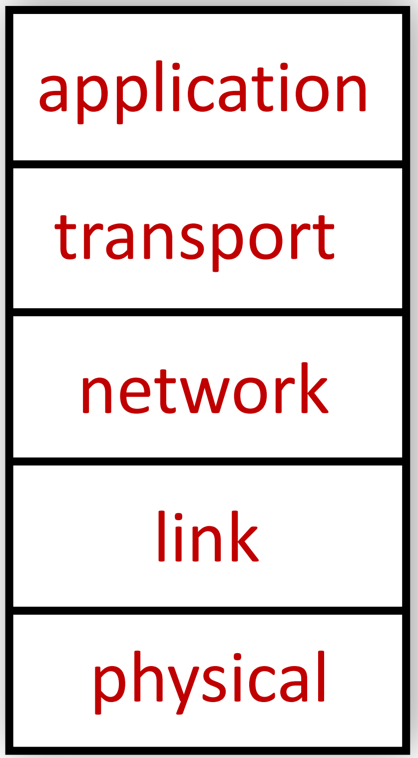
- **No!** But how does the Internet design avoid this?

Internet solution: Intermediate layers



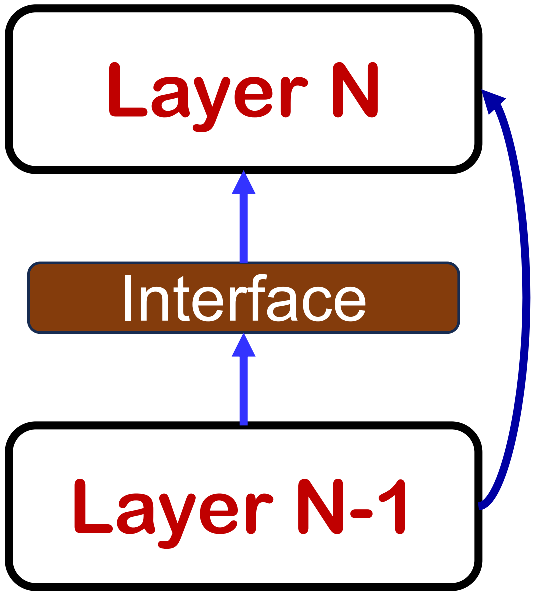
- **Intermediate layers** provide a **set of abstractions** for applications and media
- New applications or media **need only implement for** intermediate layer's **interface**

Properties of layers



Properties of layers

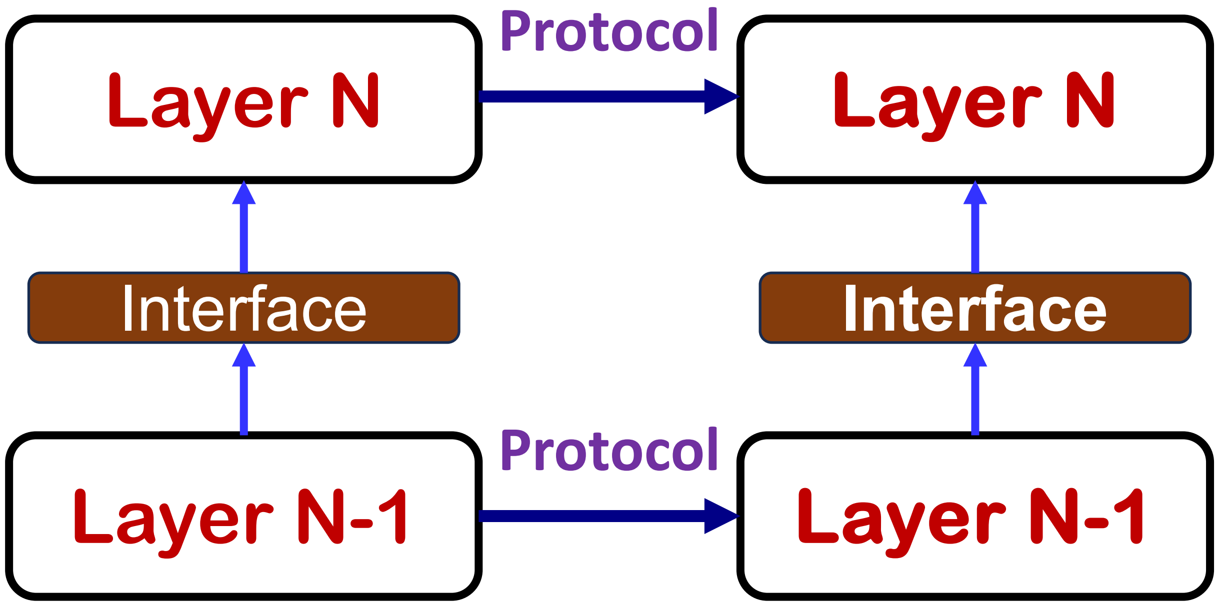
- **Service:** **What** a layer does
- **Service interface:** **How to access** the service
  - Interface for the layer **above**



Layer N uses the services provided by **layer N-1**

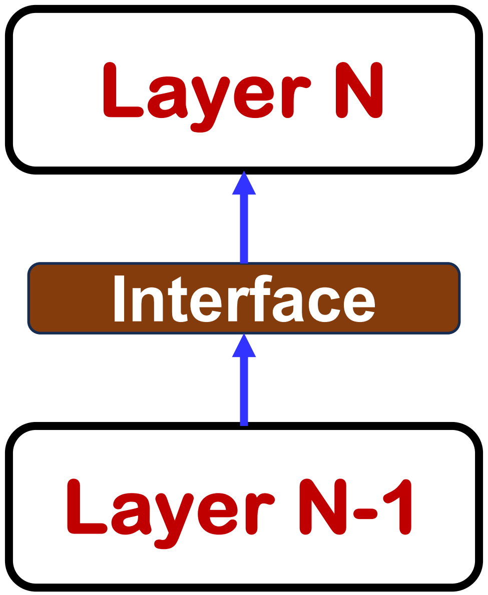
Properties of layers

- **Service: What** a layer does
- **Service interface: How to access** the service
  - Interface for the layer **above**
- **Protocol interface: How peers communicate** to implement service
  - Set of rules and formats that govern the communication **between two Internet hosts**



Physical layer (L1)

- **Service**: **Move bits** between two systems connected by a **single physical link**
- **Interface**: specifies **how to send, receive bits**
  - *e.g.*, require quantities and timing
- **Protocols**: coding scheme used to represent bits, voltage levels, duration of a bit

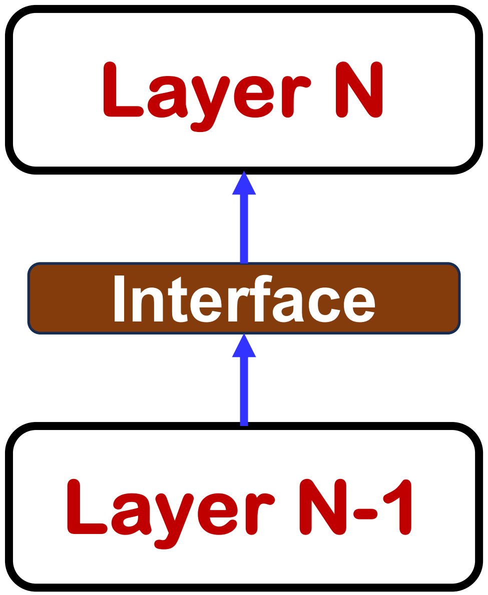


Data link layer (L2)

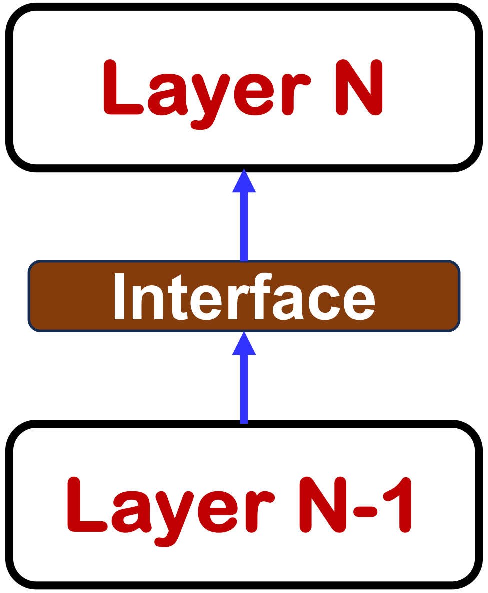
- **Service**: data transfer between neighboring network elements
  - Arbitrates access to common physical media

- **Interface**: send messages (frames) to other network elements; receive messages addressed to network elements

- **Protocols**: medium access control, retransmission



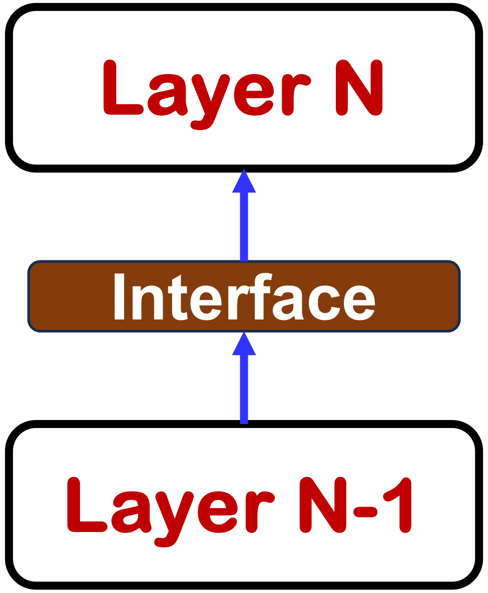
Network layer (L3)



- **Service**: routing of datagrams from source to destination
  - IP, routing protocols
- **Interface**:
  - Send packets to specified internetwork destination
  - Receive packets destined for end host
- **Protocols**:
  - Define inter-network addresses (globally unique)
  - Construct routing tables and forward datagrams

Transport layer (L4)

- **Service**: Provide **end-to-end** communication between **processes on different hosts**
  - Demultiplex communication between hosts
  - Possibly reliability in the presence of errors
  - Rate adaptation (**flow control, congestion control**)



- **Interface**: send message to specific process at given destination; local process receives messages sent to it

- **Protocol**: perhaps implement reliability, flow control, packetization of large messages, framing

Why stack or layering?

Approaches to dealing with complex systems

- Explicit structure allows identification, relationship of system's pieces
  - Layered reference model for discussion

- Modularization ease maintenance, updating the system
  - Change in layer's service implementation: transparent to rest of system

Drawbacks of layering

- Layer *n* may **duplicate** lower level functionality
  - *e.g.*, error recovery to retransmit lost data

- Layers may need **same information in headers**
  - *e.g.*, timestamps, maximum transmission unit size

- Layering can **hurt performance**
  - *e.g.*, headers

Layer violations

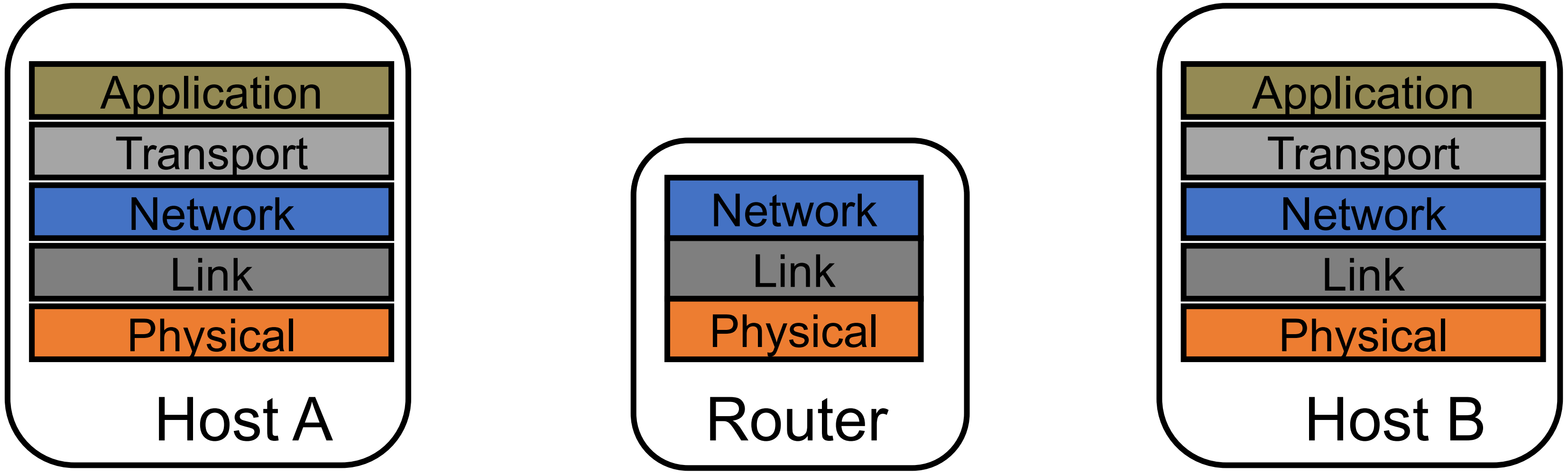
- Two types:

1. **Overlying** layer examines **underlying** layer's state
   - *e.g.*, transport monitors wireless link-layer to see whether packet loss from congestion or corruption

2. **Underlying** layer inspecting **overlying** layer's state
   - *e.g.*, firewalls, NATs (network address translators), "transparent proxies"

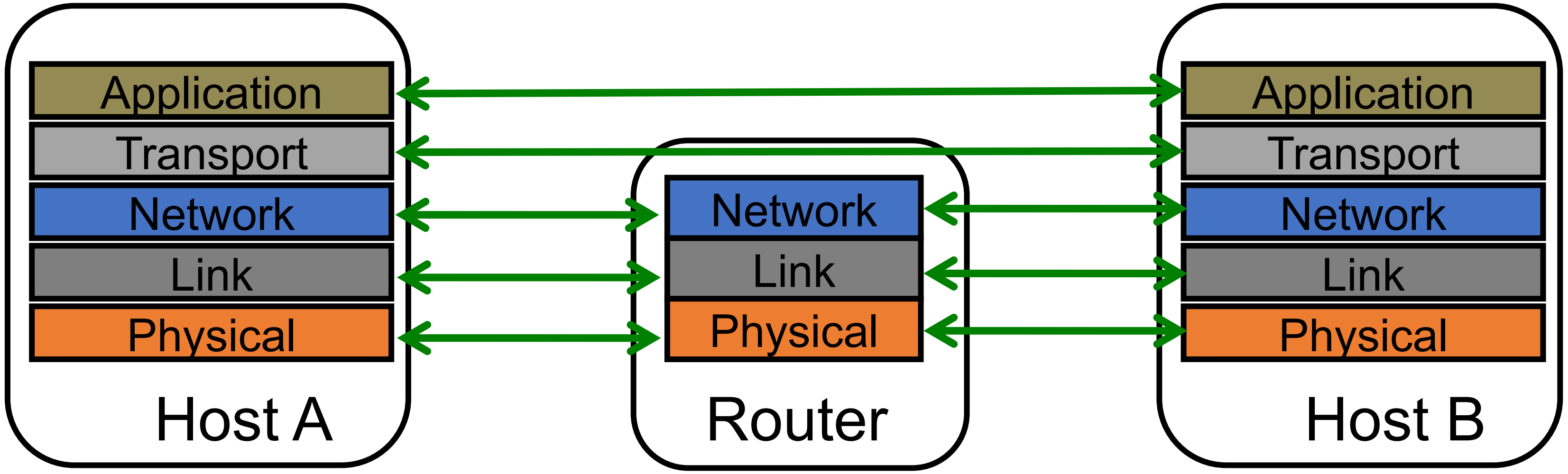
Who does what?

- Five layers
  - **Lower three layers** are implemented **everywhere**
  - **Top two layers** are implemented **only at end hosts**
    - **Their protocols are *end-to-end***



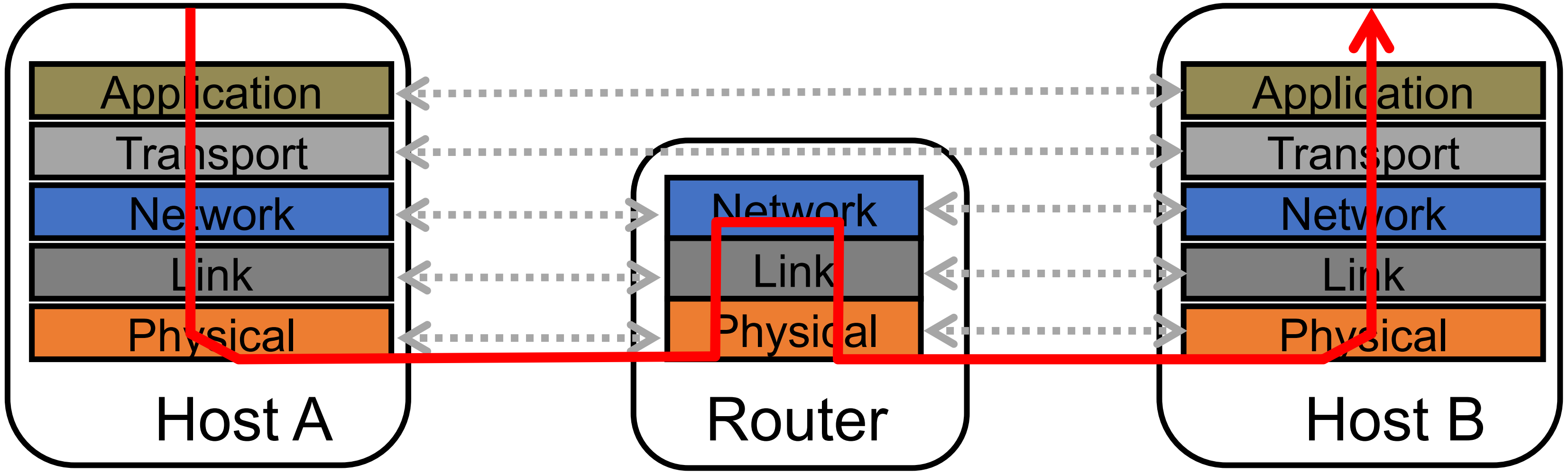
Logical communication

- Each layer on a host interacts with its **peer** host's corresponding layer via the **protocol interface**



Physical path across the Internet

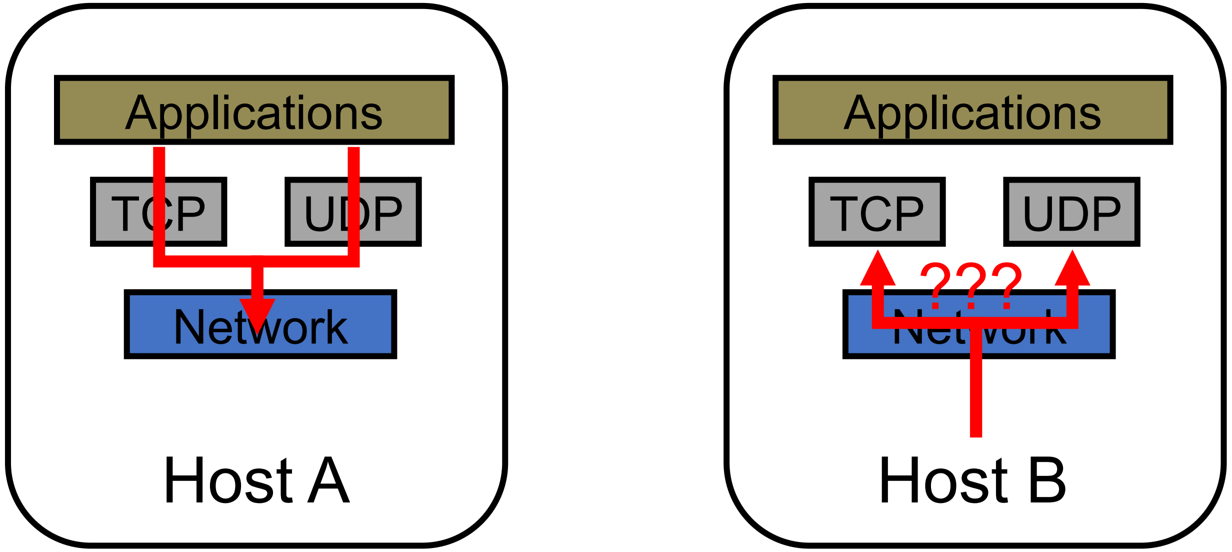
- Communication **goes down to physical network**
- Then from **network peer to peer**
- Then **up to the relevant layer**



Protocol multiplexing

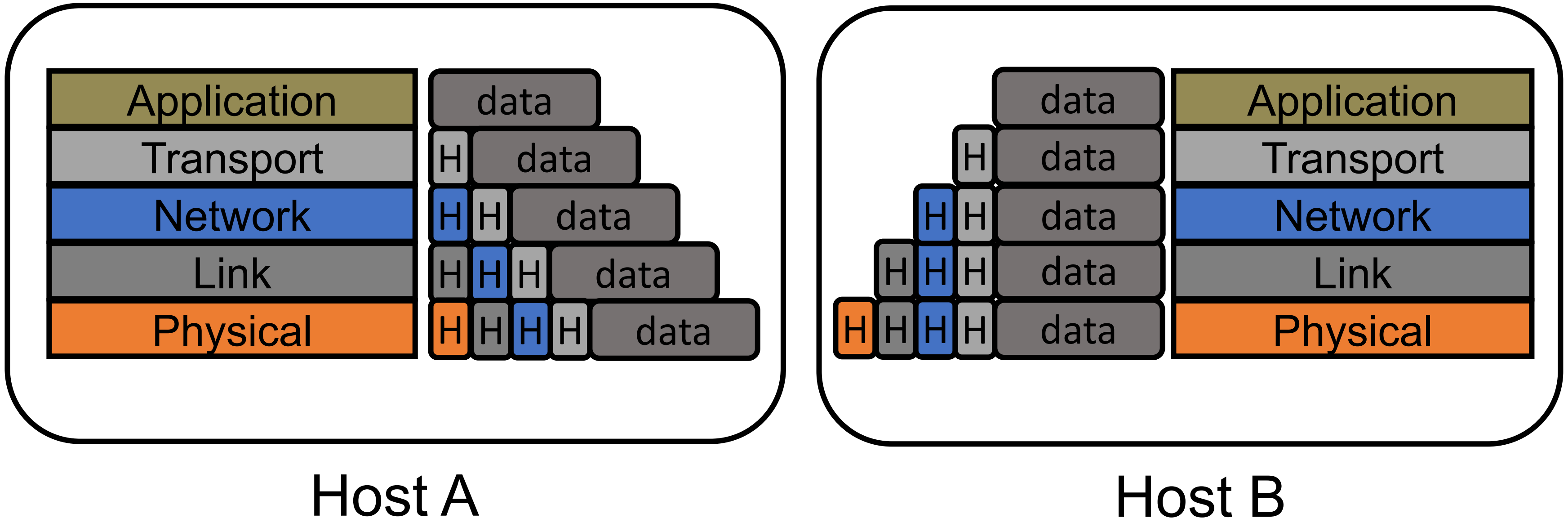
- **Multiplexing:** Multiple **overlying** protocols share use of a single **underlying** protocol

- **Problem:** How does the underlying protocol decide **which overlying protocol** messages go to?



Protocol headers

- Each layer attaches its own header (H) to facilitate communication between peer protocols
- On reception, layer **inspects and removes** its own header
  - Higher layers **don't see** lower layers' headers



Services, Layering and Encapsulation

application
transport
network
link
physical

source

M

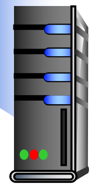
Application exchanges messages to implement some application service using *services* of transport layer

H_t M

Transport-layer protocol transfers M (e.g., reliably) from one *process* to another, using services of network layer

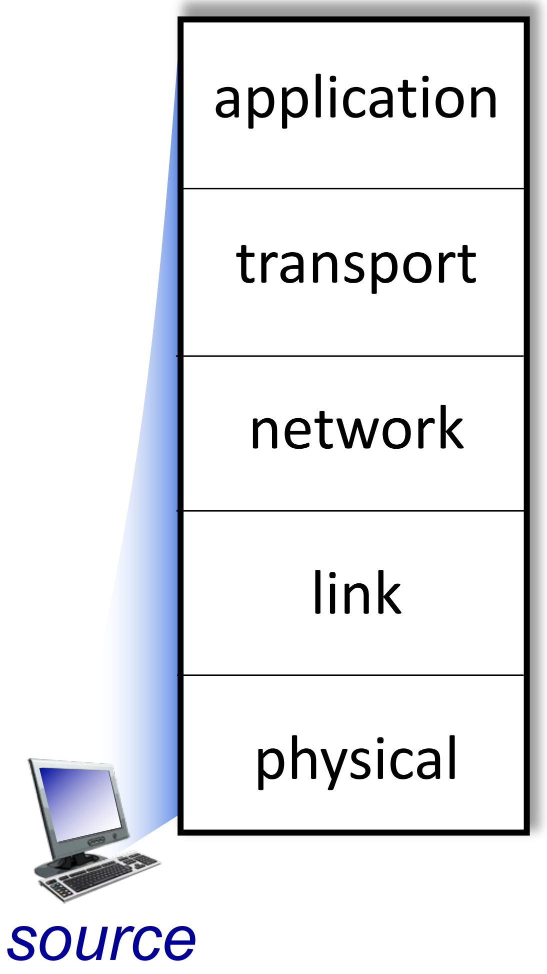
- transport-layer protocol encapsulates application-layer message, M, with *transport* layer-layer header H_t to create a transport-layer segment
  - H_t used by transport layer protocol to implement its service

application
transport
network
link
physical



destination

Services, Layering and Encapsulation



M

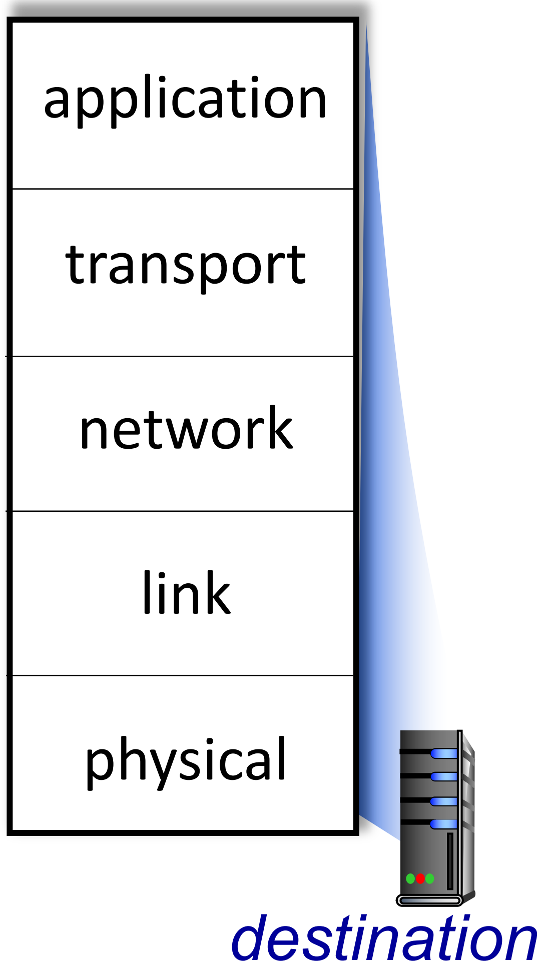
H_t M

Transport-layer protocol transfers M (e.g., reliably) from one *process* to another, using services of network layer

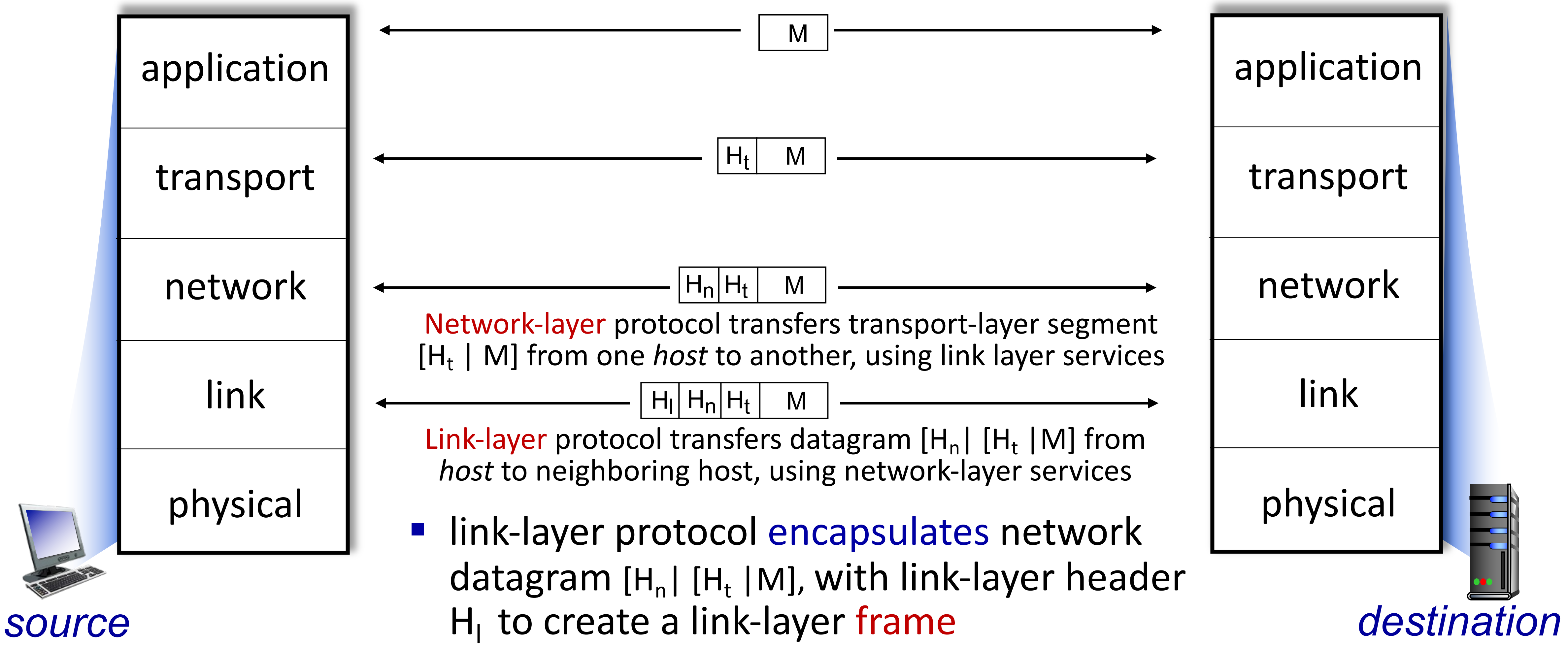
H_n H_t M

Network-layer protocol transfers transport-layer segment [H_t | M] from one *host* to another, using link layer services

- network-layer protocol encapsulates transport-layer segment [H_t | M] with network layer-layer header H_n to create a network-layer datagram
  - H_n used by network layer protocol to implement its service

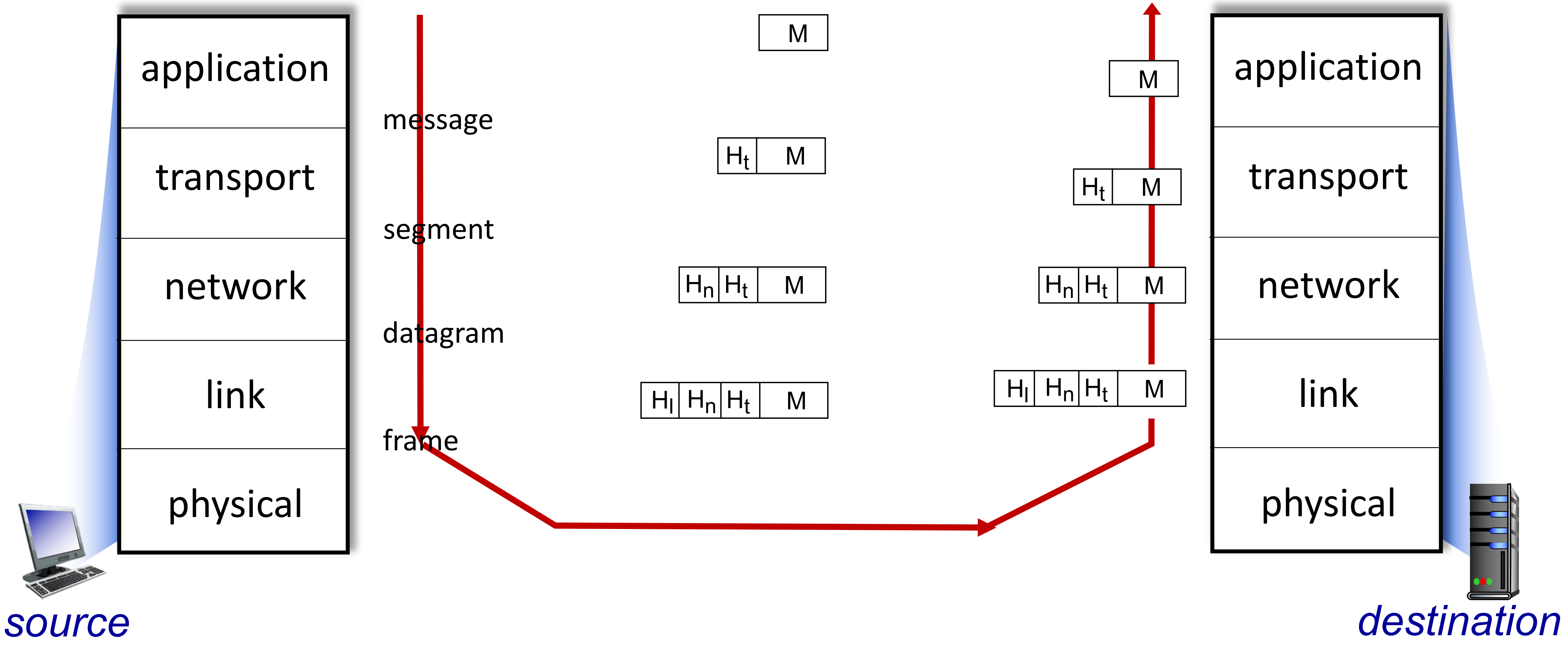


Services, Layering and Encapsulation

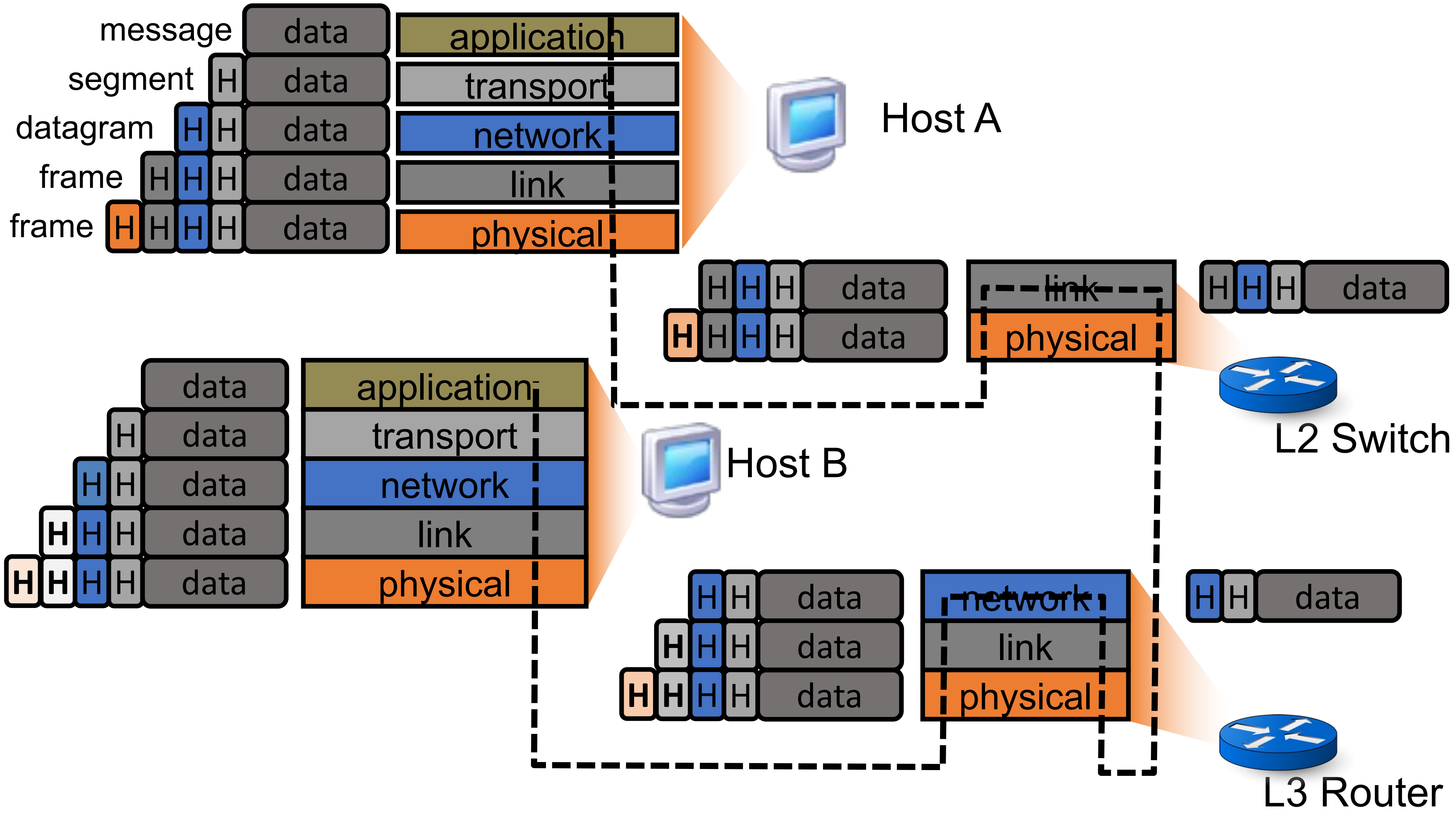


- link-layer protocol encapsulates network datagram [H_n| [H_t |M], with link-layer header H_l to create a link-layer frame

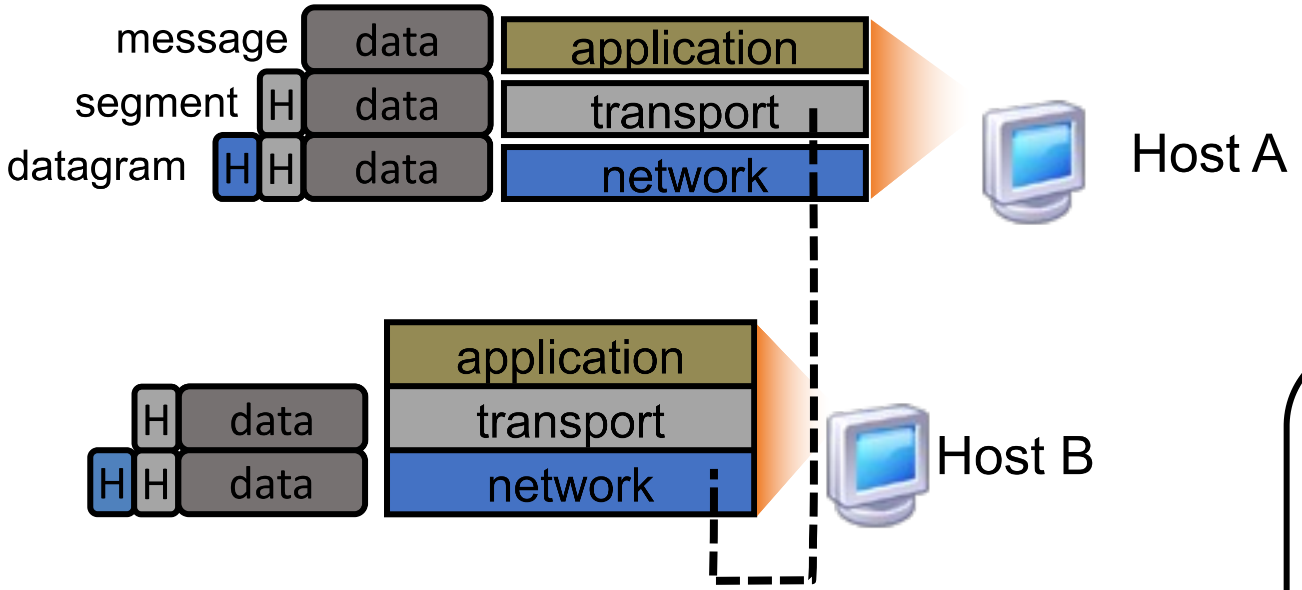
Services, Layering and Encapsulation

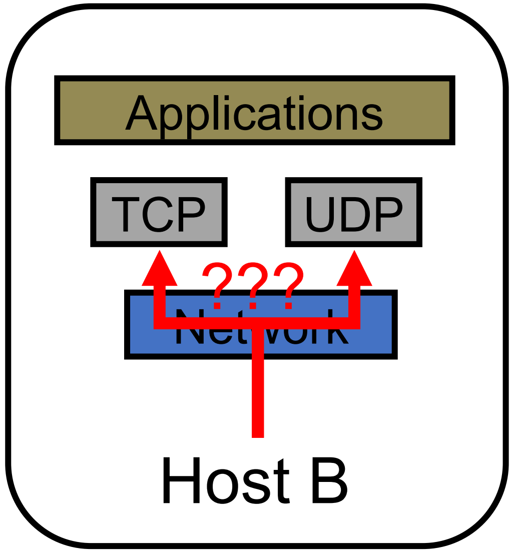


Encapsulation in the Internet



Protocol demultiplexing





- Lower-layer header contains demultiplexing information
- **Network header** contains **Protocol** field specifying overlying protocol

Chapter 1: roadmap

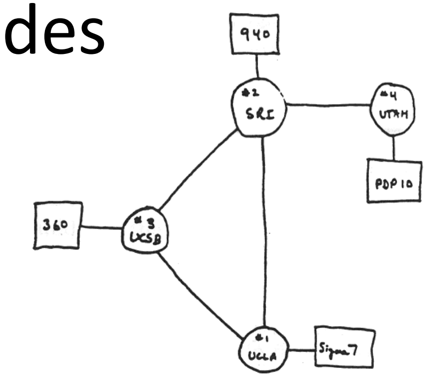
- What *is* the Internet?
- What *is* a protocol?
- Network edge: hosts, access network, physical media
- Network core: packet/circuit switching, internet structure
- Performance: loss, delay, throughput
- Security
- Protocol layers, service models
- History

Internet history

1961-1972: Early packet-switching principles

- 1961: Kleinrock - queueing theory shows effectiveness of packet-switching
- 1964: Baran - packet-switching in military nets
- 1967: ARPAnet conceived by Advanced Research Projects Agency
- 1969: first ARPAnet node operational

- 1972:
  - ARPAnet public demo
  - NCP (Network Control Protocol) first host-host protocol
  - first e-mail program
  - ARPAnet has 15 nodes



THE ARPA NETWORK

Internet history

1972-1980: Internetworking, new and proprietary networks

- 1970: ALOHAnet satellite network in Hawaii
- 1974: Cerf and Kahn - architecture for interconnecting networks
- 1976: Ethernet at Xerox PARC
- late70's: proprietary architectures: DECnet, SNA, XNA
- 1979: ARPAnet has 200 nodes

Cerf and Kahn's internetworking principles:

- minimalism, autonomy - no internal changes required to interconnect networks
- best-effort service model
- stateless routing
- decentralized control

define today's Internet architecture

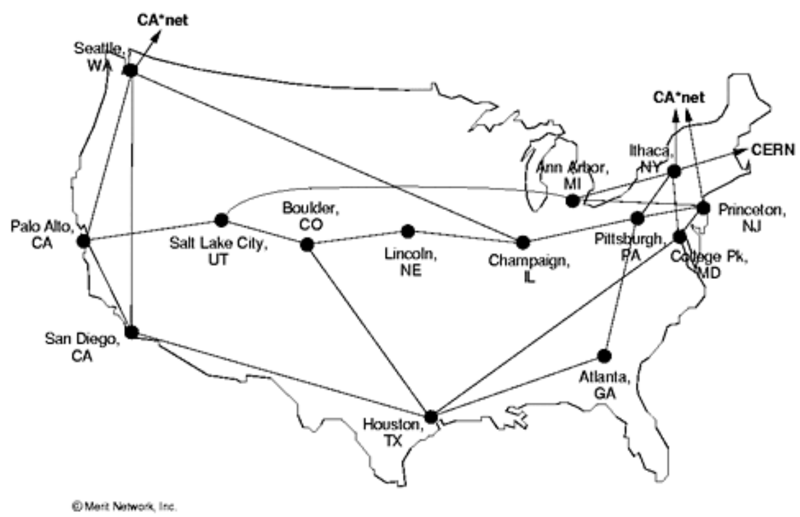
Internet history

1980-1990: new protocols, a proliferation of networks

- 1983: deployment of TCP/IP
- 1982: smtp e-mail protocol defined
- 1983: DNS defined for name-to-IP-address translation
- 1985: FTP protocol defined
- 1988: TCP congestion control

- new national networks: CSnet, BITnet, NSFnet, Minitel
- 100,000 hosts connected to confederation of networks

NSFNET T1 Network 1991



Internet history

1990, 2000s: commercialization, the Web, new applications

- early 1990s: ARPAnet decommissioned
- 1991: NSF lifts restrictions on commercial use of NSFnet (decommissioned, 1995)
- early 1990s: Web
  - hypertext [Bush 1945, Nelson 1960's]
  - HTML, HTTP: Berners-Lee
  - 1994: Mosaic, later Netscape
  - late 1990s: commercialization of the Web

late 1990s – 2000s:

- more killer apps: instant messaging, P2P file sharing
- network security to forefront
- est. 50 million host, 100 million+ users
- backbone links running at Gbps

Internet history

2005-present: scale, SDN, mobility, cloud

- aggressive deployment of broadband home access (10-100's Mbps)
- 2008: software-defined networking (SDN)
- increasing ubiquity of high-speed wireless access: 4G/5G, WiFi
- service providers (Google, FB, Microsoft) create their own networks
  - bypass commercial Internet to connect "close" to end user, providing "instantaneous" access to social media, search, video content, …
- enterprises run their services in "cloud" (e.g., Amazon Web Services, Microsoft Azure)
- rise of smartphones: more mobile than fixed devices on Internet (2017)
- ~21B devices attached to Internet (2021)

Chapter 1: summary

We've covered a "ton" of material!

- Internet overview
- what's a protocol?
- network edge, access network, core
  - packet-switching versus circuit-switching
  - Internet structure
- performance: loss, delay, throughput
- layering, service models
- security
- history

You now have:

- context, overview, vocabulary, "feel" of networking
- more depth, detail, *and fun* to follow*!*