Networks

Where We Are In The Course

- We've done:
  - Processes
  - Threads
  - Synchronization
  - Virtual Memory
  - Disks
  - File Systems
- We have our *almost* all our building blocks!
- One more: networks
- And then… combinations:
  - Parallel and Distributed Computing
  - Distributed and Cluster File Systems
  - Security

Networks

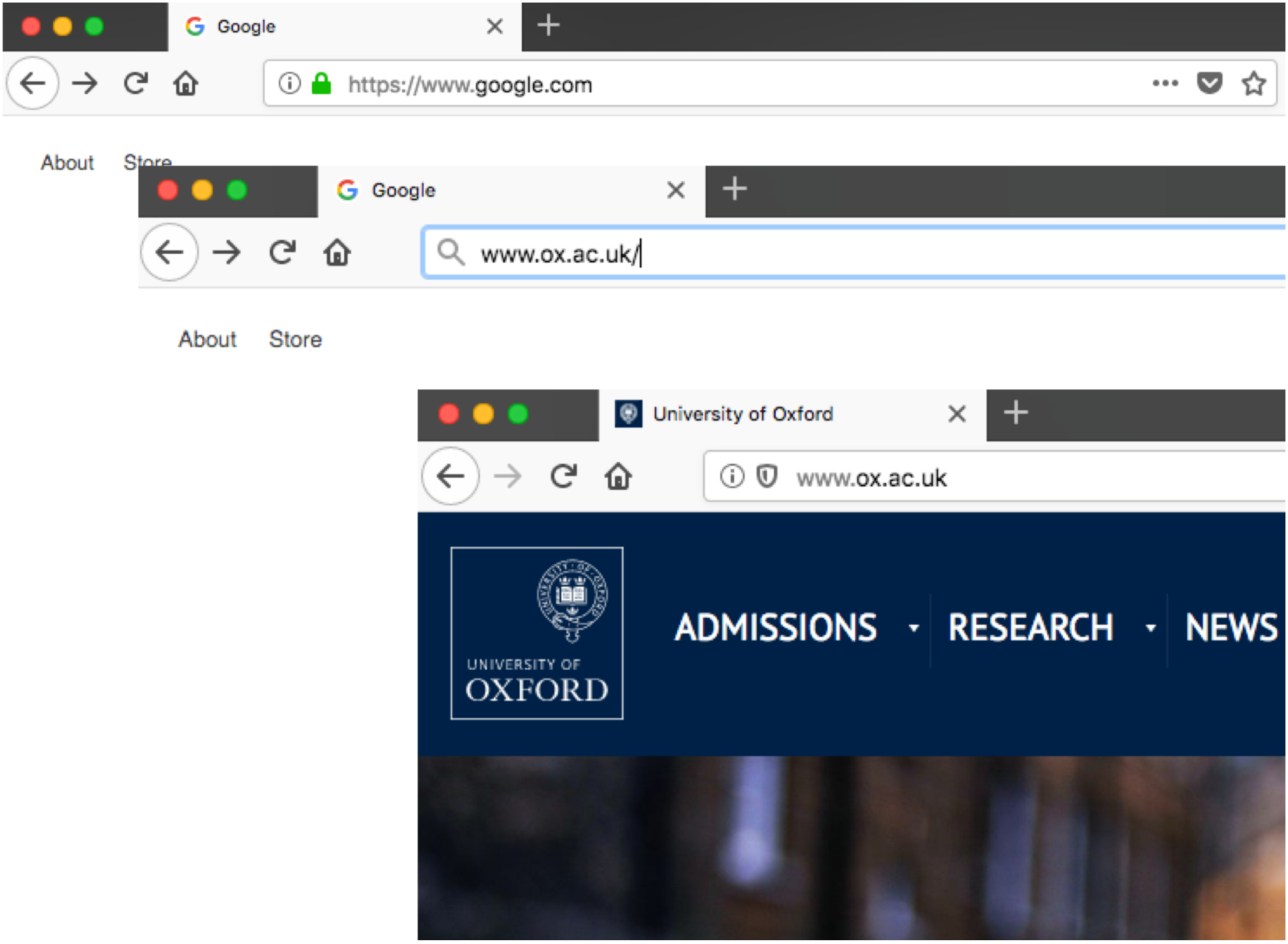
- A system of lines or channels that interconnect
  - Railroads, telephones
- Computer networks
  - Hierarchical systems of boxes and wires
    - usually concerned with providing efficient, correct, and robust message passing between two separate nodes (which are usually computers)
    - organized by geographical proximity
  - Complex software is necessary to make them work

OS View

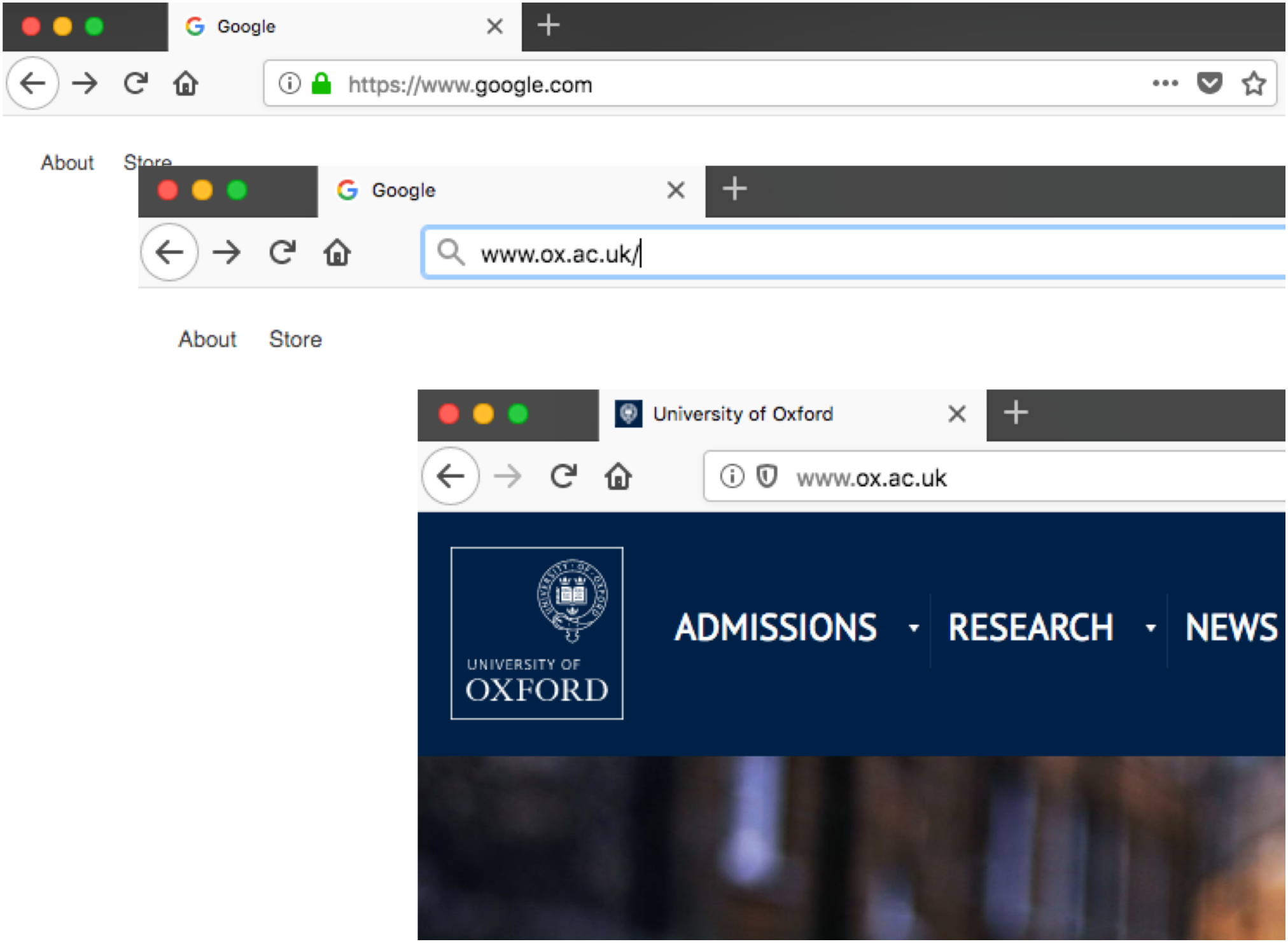
OS sees the network as just another device

- Network Interface Controller (NIC) is added to the bus
- Transfer data to/from memory to NIC through DMA or memory-mapped I/O

But like any device, there is a lot of software and metadata for its management


Google
https://www.google.com
About
Store
Google
www.ox.ac.uk/
About
Store
University of Oxford
www.ox.ac.uk
UNIVERSITY OF
OXFORD
ADMISSIONS
RESEARCH
NEWS

Finding a Web Page


Google
https://www.google.com
About
Store
Google
www.ox.ac.uk/
About
Store
University of Oxford
www.ox.ac.uk
UNIVERSITY OF
OXFORD
ADMISSIONS
RESEARCH
NEWS

Translating from Text to Numbers

- These symbolic names are Internet domain names
- Internet domain names are mapped to IP addresses
  - skipper.cs.utexas.edu is mapped to 128.83.139.82
  - www.ox.ac.uk is mapped to 129.67.242.154
- IP addresses are changing…
  - IPv4: 32-bit addresses such as 128.83.139.82
  - IPv6: 128-bit addresses such as fe80::46a8:42ff:fe44:a726
  - 1996: Internet Engineering Task Force (IETF) introduced Internet Protocol Version 6 (IPv6) with 128-bit addresses
    - Intended as the successor to IPv4
  - As of 2018, vast majority of Internet traffic still carried by IPv4
    - 22% of users access Google services using IPv6 up from 4% in 2015

```
[~ $ uname -a                                                                  ]
Linux skipper 4.4.0-138-generic #164-Ubuntu SMP Tue Oct 2 17:16:02 UTC 2018 x86_
64 x86_64 x86_64 GNU/Linux
[~ $ ifconfig                                                                  ]
eno1      Link encap:Ethernet  HWaddr 44:a8:42:44:a7:26
          inet addr:128.83.139.132  Bcast:128.83.255.255  Mask:255.255.0.0
          inet6 addr: fe80::46a8:42ff:fe44:a726/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:62012840 errors:0 dropped:75 overruns:0 frame:0
          TX packets:60248962 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:39249990385 (39.2 GB)  TX bytes:56169449162 (56.1 GB)
          Interrupt:18

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:9452917 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9452917 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:235017083238 (235.0 GB)  TX bytes:235017083238 (235.0 GB)

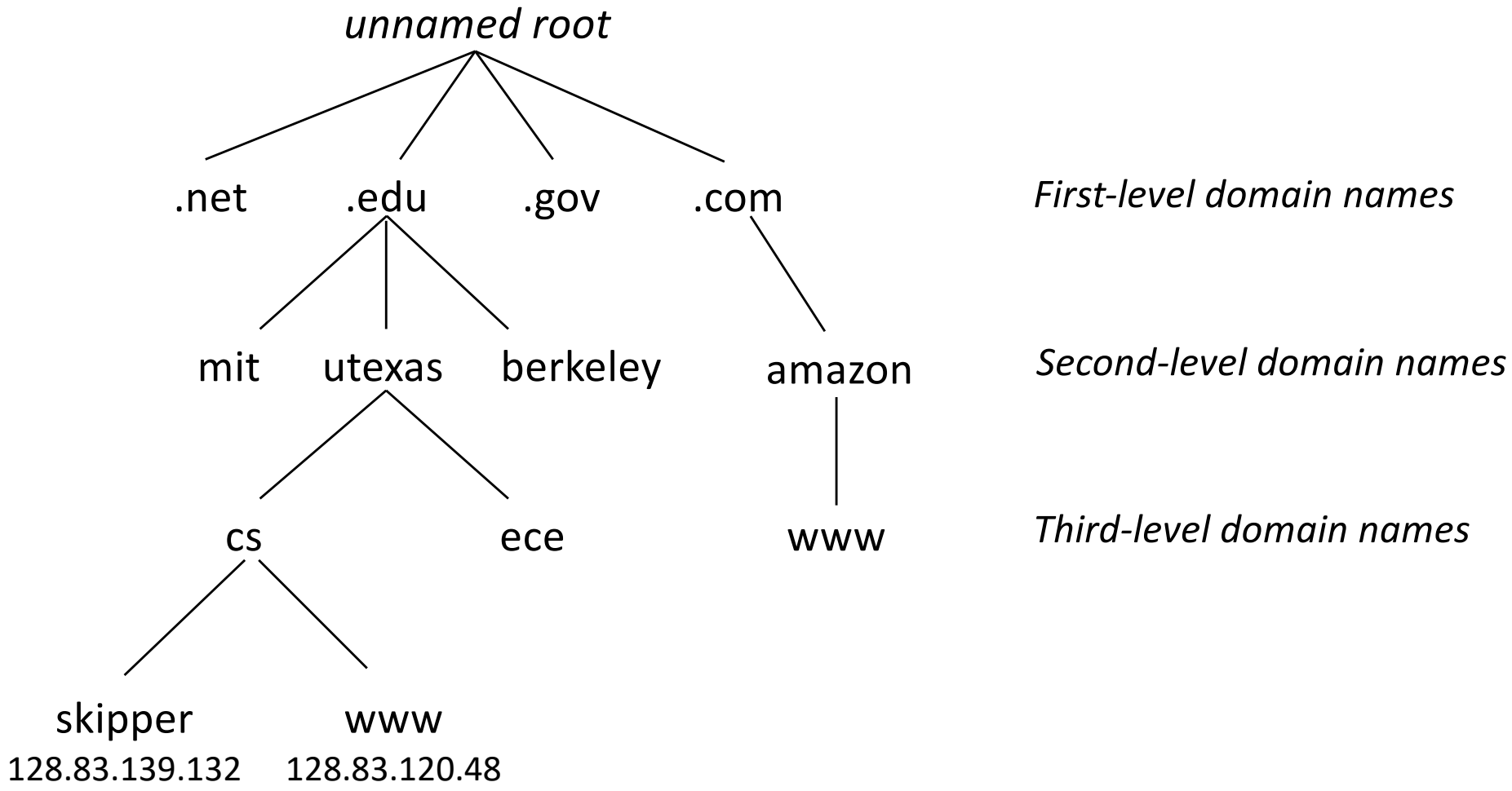
~ $
```

Domain Naming System (DNS)

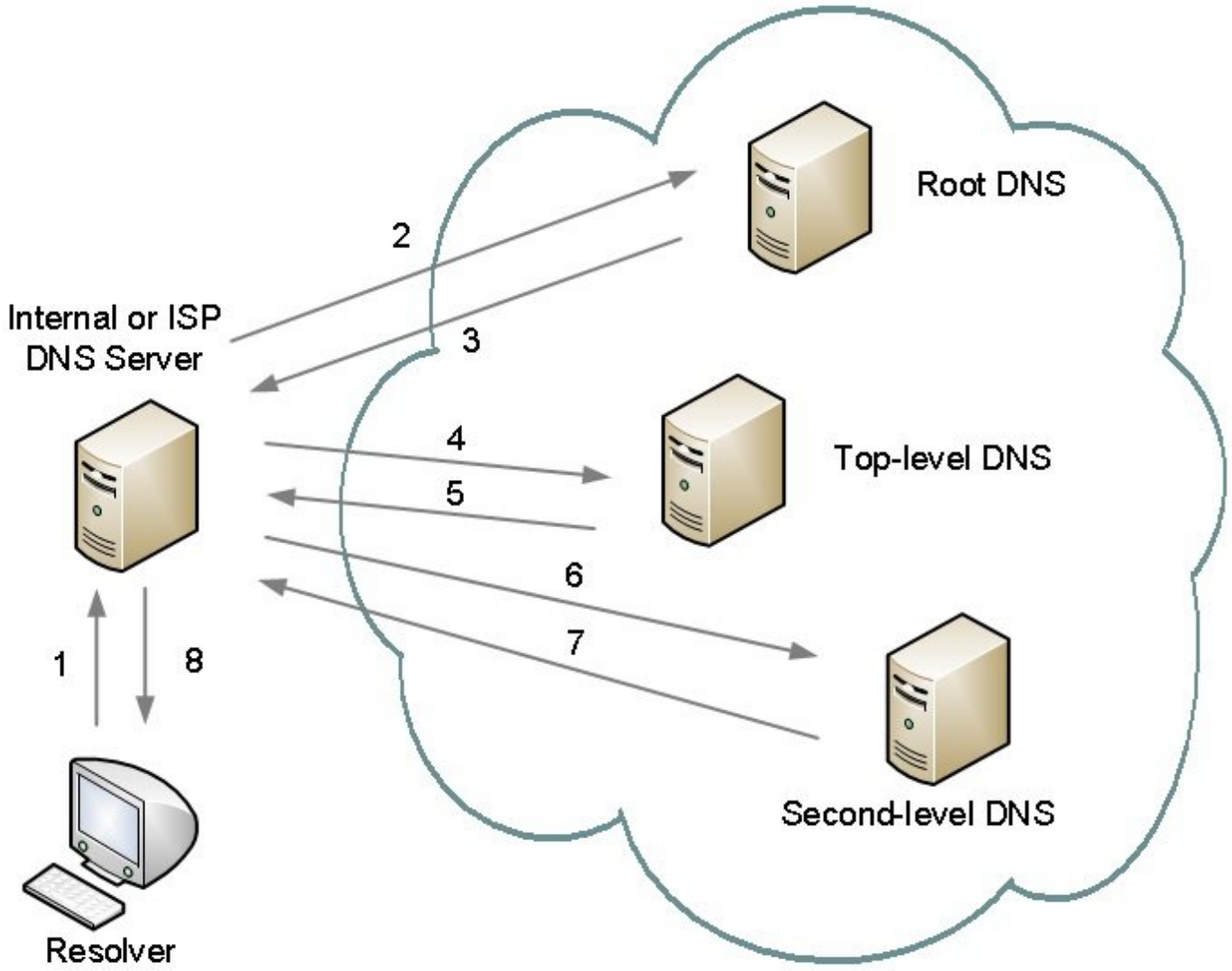
- The Internet maintains a mapping between IP addresses and domain names in a huge worldwide distributed database called DNS

- Conceptually, programmers can view the DNS database as a collection of millions of host entries.
  - Each host entry defines the mapping between a set of domain names and IP addresses.
  - In a mathematical sense, a host entry is an equivalence class of domain names and IP addresses.

Internet Domain Names



DNS


Root DNS
2
3
Internal or ISP
DNS Server
4
5
Top-level DNS
6
7
1
8
Second-level DNS
Resolver

You: GDC 3rd Floor Lab



Do we get to just send the data directly there? *Is there a direct connection?*

Information you want: Oxford, UK



Getting the Data

You: GDC 3rd Floor Lab

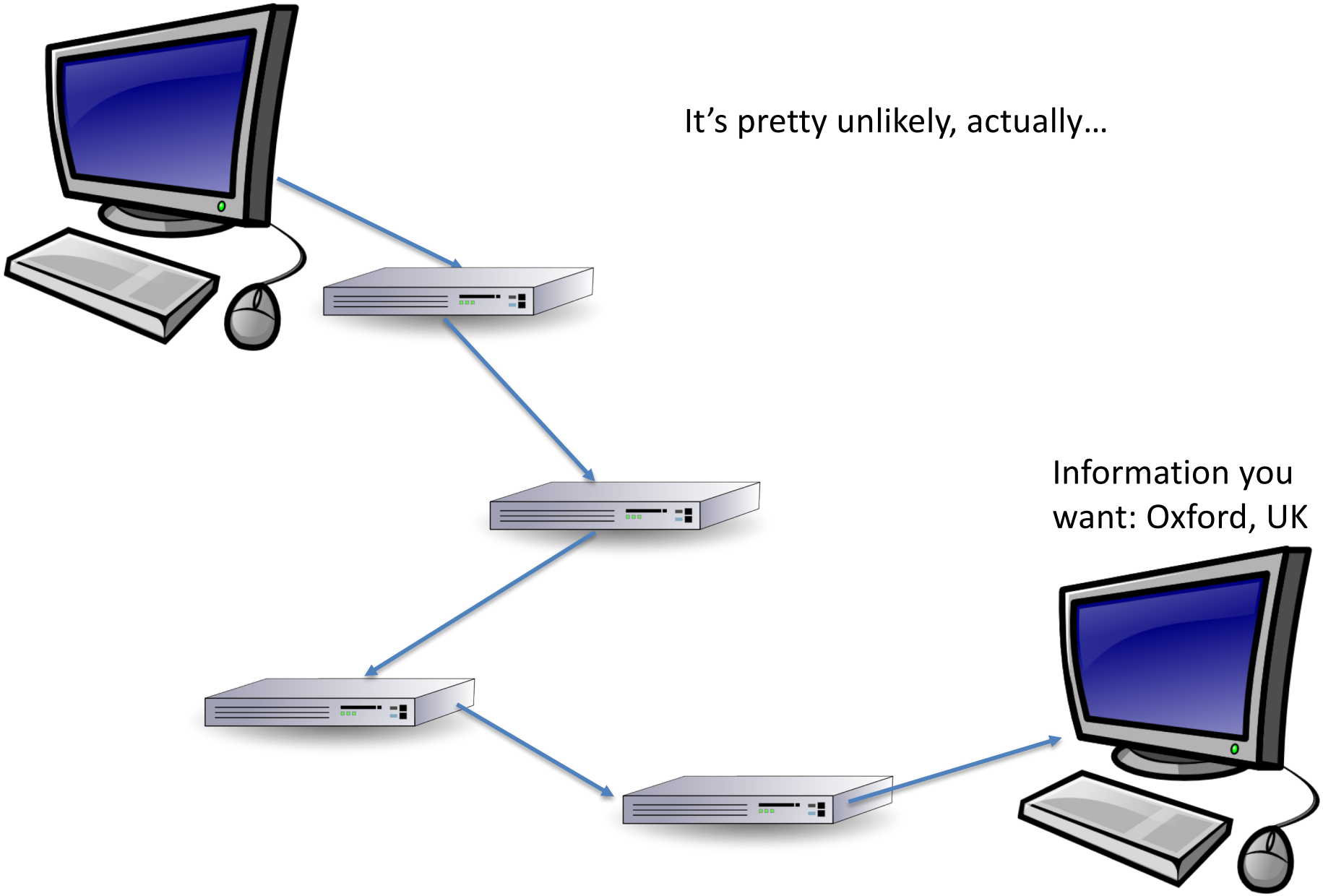


Do we get to just send the data directly there? *Is there a direct connection?*

Information you want: Oxford, UK



You: GDC 3rd Floor Lab

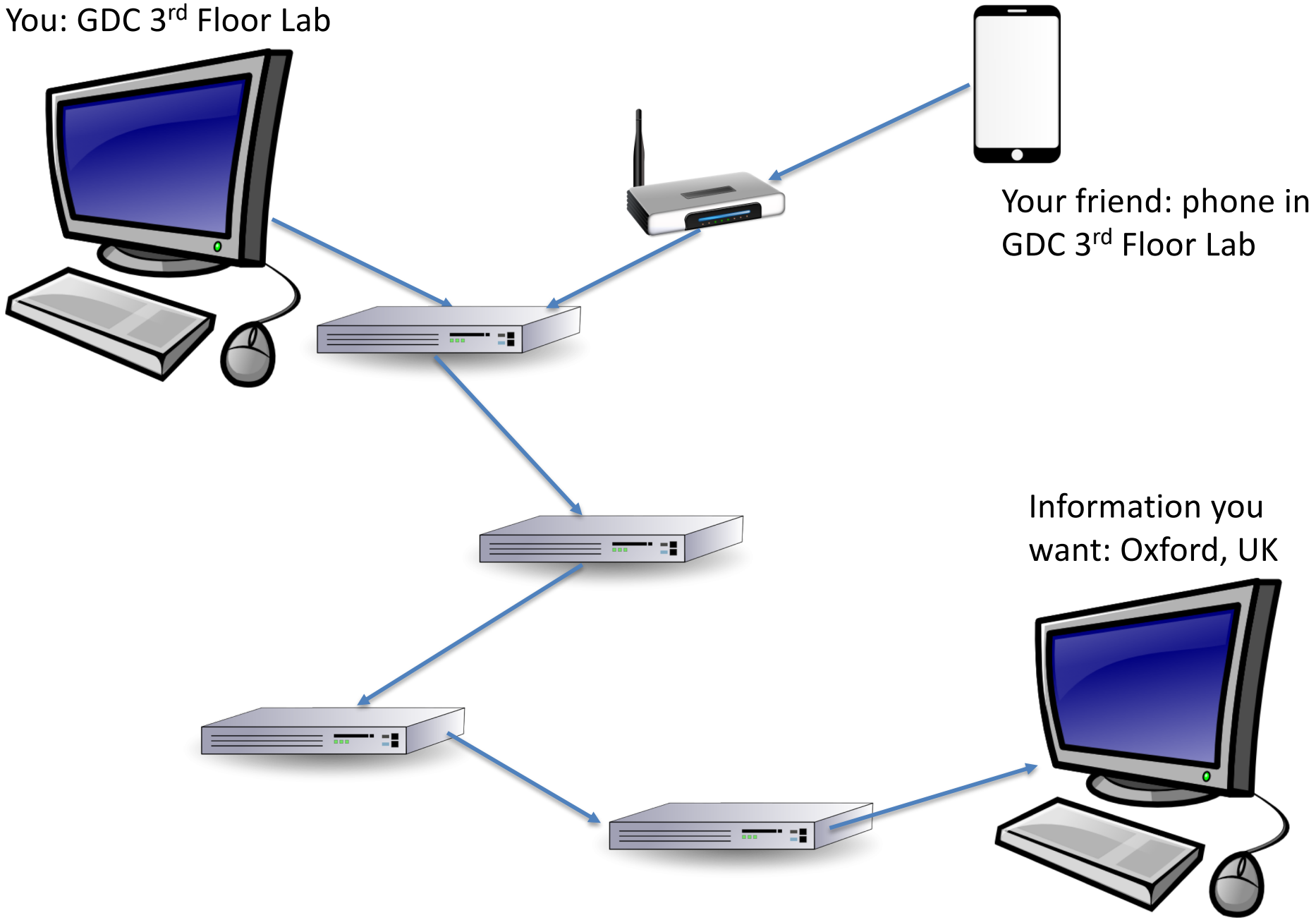


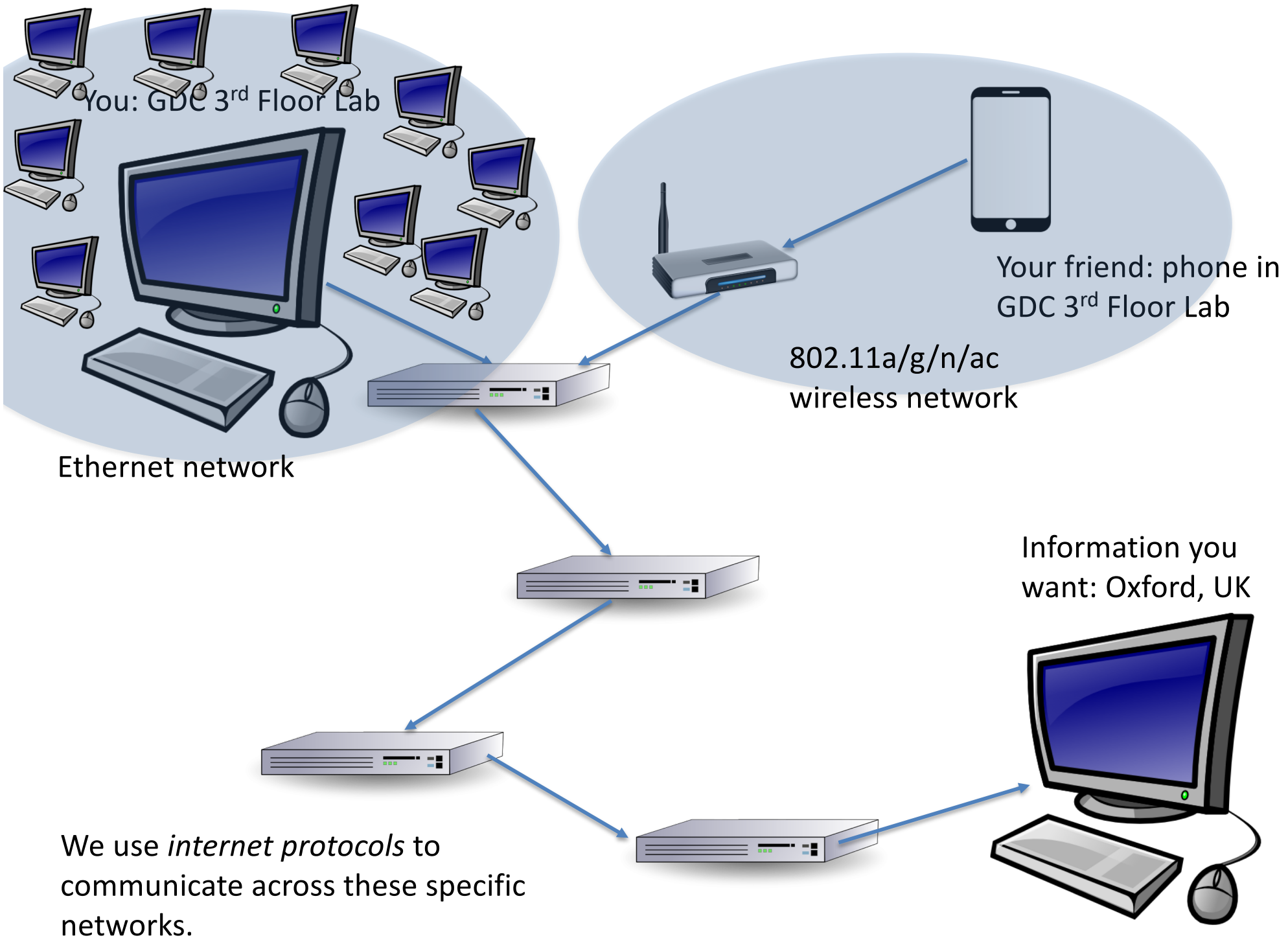
It’s pretty unlikely, actually…

Information you want: Oxford, UK

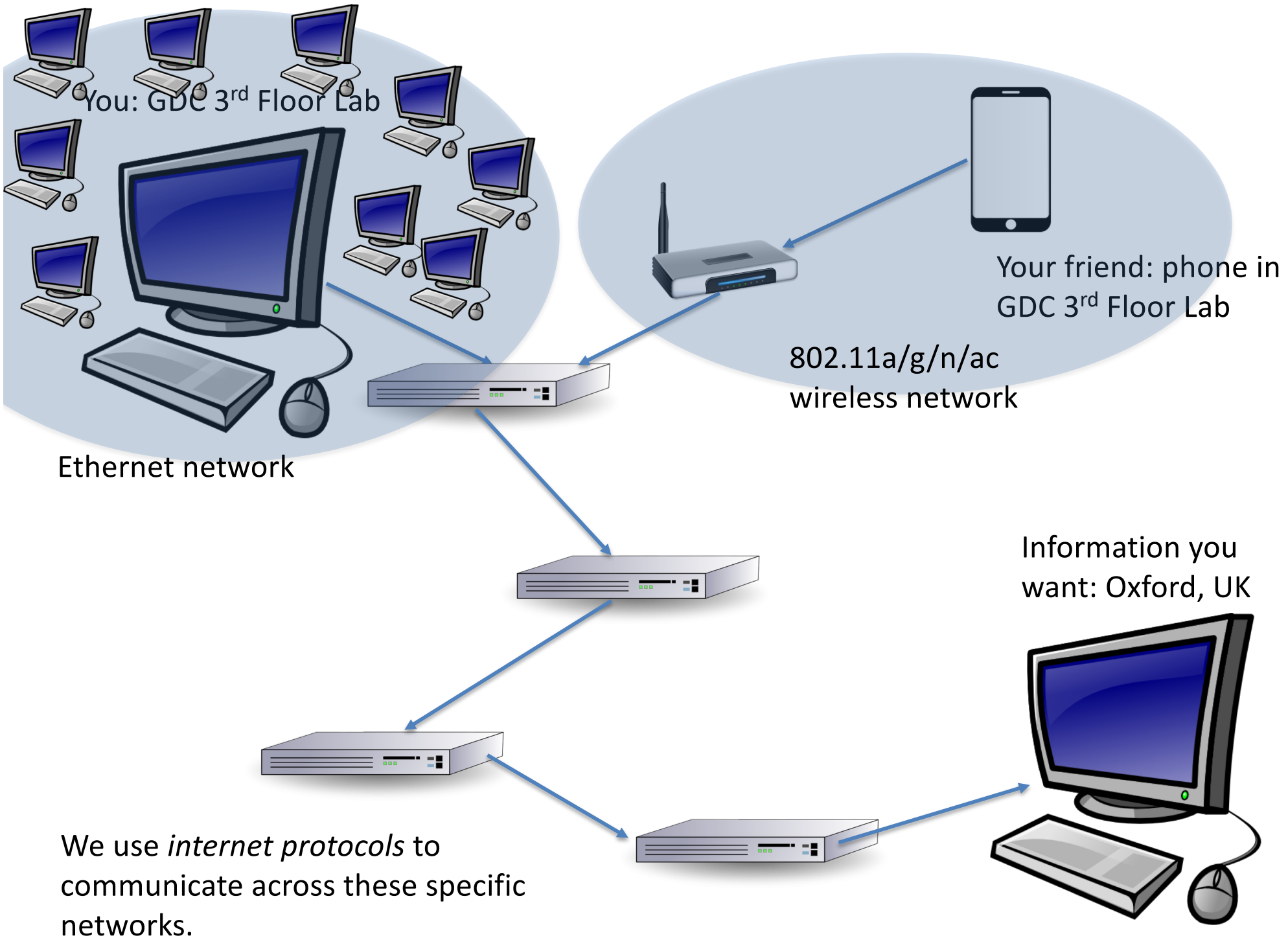
Internet Protocol: Routing

- *Packets* are routed based on destination IP address
  - Address space is structured to make routing practical at global scale
  - For example, 128.83.*.* goes to UT
- Each machine sends messages directly to a set of local names and a *Gateway,* which provides access to the rest of the Internet


You: GDC 3rd Floor Lab
Your friend: phone in GDC 3rd Floor Lab
Information you want: Oxford, UK


You: GDC 3rd Floor Lab
Your friend: phone in GDC 3rd Floor Lab
802.11a/g/n/ac wireless network
Ethernet network
Information you want: Oxford, UK
We use *internet protocols* to communicate across these specific networks.

Connecting Networks

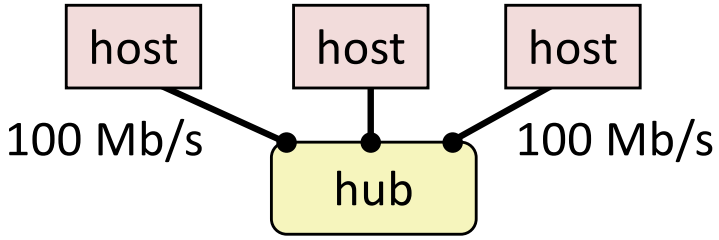

You: GDC 3rd Floor Lab
Your friend: phone in GDC 3rd Floor Lab
802.11a/g/n/ac wireless network
Ethernet network
Information you want: Oxford, UK
We use *internet protocols* to communicate across these specific networks.

The internets

- Multiple incompatible networks can be physically connected by specialized computers called *routers*
- The connected networks are called an *internet*

- The Global IP **I**nternet is the most famous example of an **i**nternet
  - Most computers today speak IP
  - Our examples so far have assumed this Global IP Internet

Example Network: Ethernet

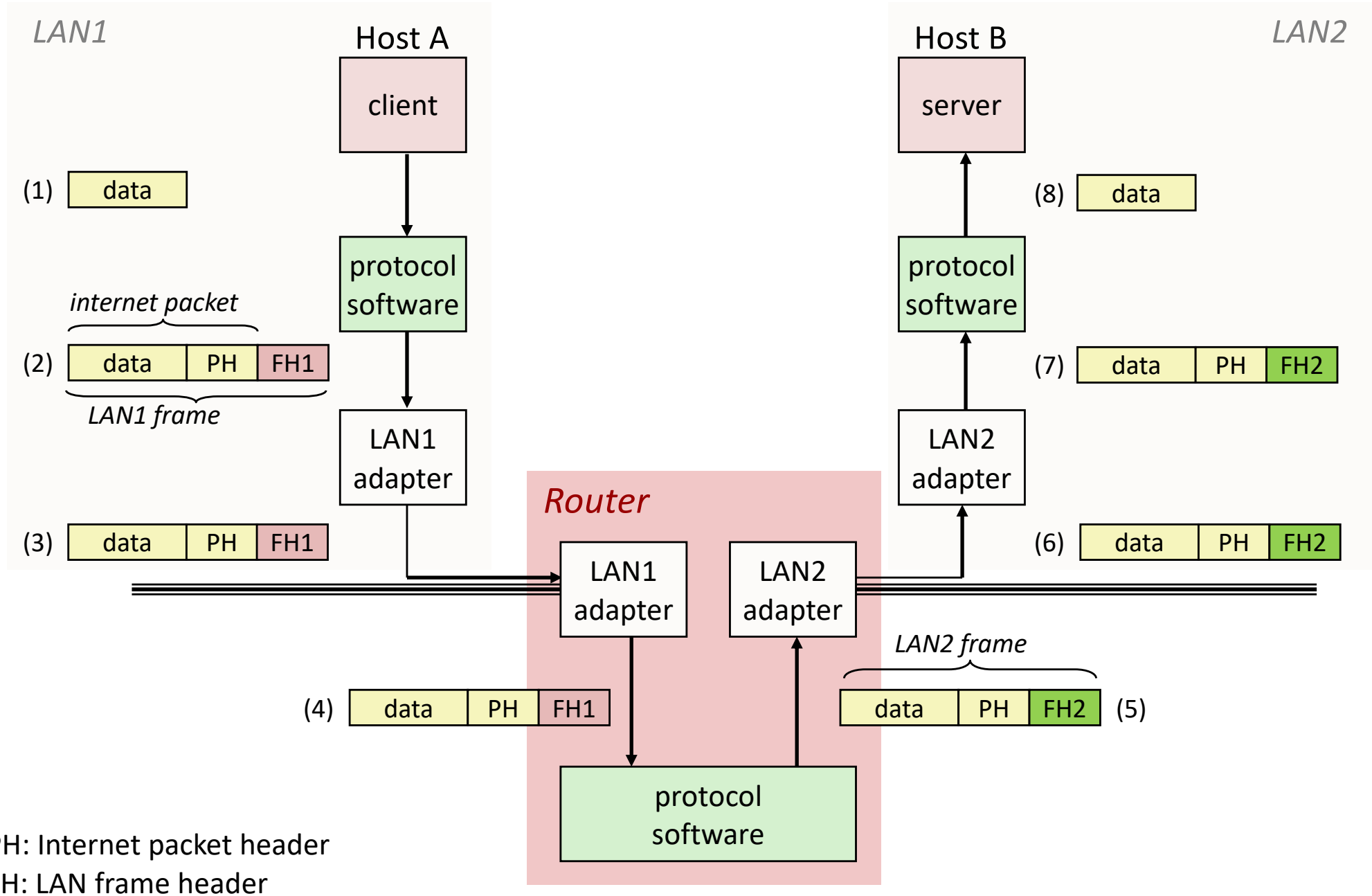


- Invented Xerox PARC, 1973
  - co-invented by Bob Metcalfe, who is now at UT!
- Connects collection of *hosts* connected by wires to a *hub*
- Each host has at least one Ethernet adaptor
- Each Ethernet adapter has a unique 48-bit MAC address (**M**edia **A**ccess **C**ontrol)
  - e.g., 00:16:ea:e3:54:e6
- Hub copies each bit from each adaptor to every adaptor on that hub
  - Every host sees every bit

Classifications of Networks

- System Area Network (SAN)
  - Connects cluster or machine room
  - e.g., Quadrics, Fibre
- Local Area Network (LAN)
  - Connects nodes in a single building
  - Needs to be fast and reliable
  - e.g., Ethernet
- Wide Area Network (WAN)
  - Connects nodes across the state, country, or planet
  - Typically slower and less reliable than LAN
  - Often high-speed point-to-point phone lines

Transferring Data Over an internet



PH: Internet packet header
FH: LAN frame header

We've still got some problems…

1. How do we get data to the right process?
   - We are in a multiprogramming environment!
   - The right application, the right chrome tab, …

   We're going to add another layer!
2. What do we do about data loss?
   - Wait. What? How do we even get data loss?

Processes and Reliability

Layers of the Network

Open Source Interconnection
7 Layer Model

Provides common vocabulary for network engineers (generally the developers stay out of this level of detail).

Layer 1 is Hardware
Layer 2 is Switching
Layer 3 is Routing
Layer 4 is TCP
Layer 5 is often OS/Library
Layer 6 is often OS/Library
Layer 7 is Application

OSI (Open Source Interconnection) 7 Layer Model

Layer	Application/Example		Central Device/ Protocols		DOD4 Model
Application (7) Serves as the window for users and application processes to access the network services.	**End User layer** Program that opens what was sent or creates what is to be sent Resource sharing • Remote file access • Remote printer access • Directory services • Network management		**User Applications** SMTP	GATEWAY Can be used on all layers	Process
Presentation (6) Formats the data to be presented to the Application layer. It can be viewed as the "Translator" for the network.	**Syntax layer** encrypt & decrypt (if needed) Character code translation • Data conversion • Data compression • Data encryption • **Character Set Translation**		JPEG/ASCII EBDIC/TIFF/GIF PICT		
Session (5) Allows session establishment between processes running on different stations.	**Synch & send to ports** (logical ports) Session establishment, maintenance and termination • Session support - perform security, name recognition, logging, etc.		**Logical Ports** RPC/SQL/NFS NetBIOS names		
Transport (4) Ensures that messages are delivered error-free, in sequence, and with no losses or duplications.	**TCP** Host to Host, Flow Control Message segmentation • Message acknowledgement • Message traffic control • Session multiplexing	PACKET FILTERING	TCP/SPX/UDP		Host to Host
Network (3) Controls the operations of the subnet, deciding which physical path the data takes.	**Packets** ("letter", contains IP address) Routing • Subnet traffic control • Frame fragmentation • Logical-physical address mapping • Subnet usage accounting		**Routers** IP/IPX/ICMP		Internet
Data Link (2) Provides error-free transfer of data frames from one node to another over the Physical layer.	**Frames** ("envelopes", contains MAC address) [NIC card — Switch — NIC card] (end to end) Establishes & terminates the logical link between nodes • Frame traffic control • Frame sequencing • Frame acknowledgment • Frame delimiting • Frame error checking • Media access control		**Switch Bridge WAP** PPP/SLIP	Land Based Layers	Network
Physical (1) Concerned with the transmission and reception of the unstructured raw bit stream over the physical medium.	**Physical structure** Cables, hubs, etc. Data Encoding • Physical medium attachment • Transmission technique - Baseband or Broadband • Physical medium transmission Bits & Volts		**Hub**		

Layer 4: TCP/IP Protocol Family

- Layer 4 guides the data to the process involved in the communication
- Includes both TCP/IP and UDP, which are the most popular protocols on IP
  - Both are built on top of IP

UDP: User Datagram Protocol

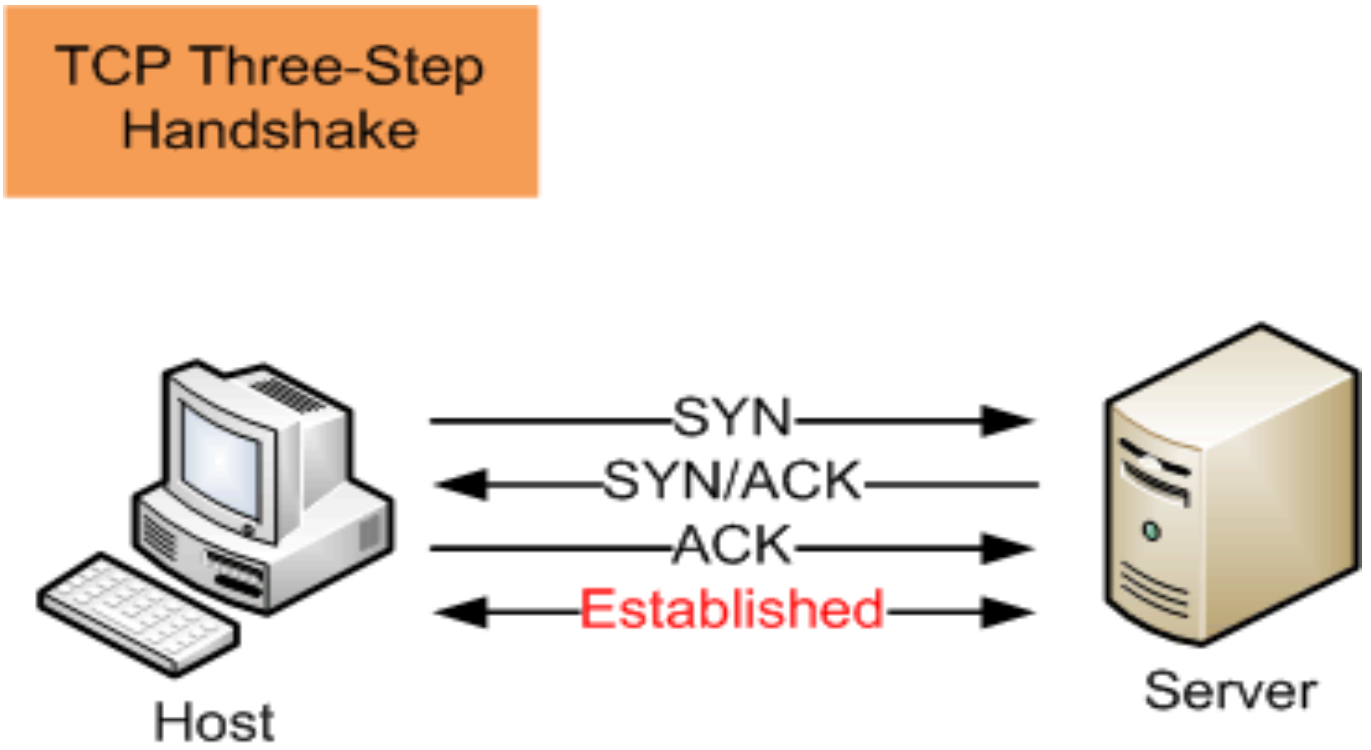
- Uses IP to provide *unreliable* datagram delivery from process-to-process
  - UDP sends *datagrams*
- Network does not deliver data reliably
  - Several types of error can affect data delivery
    - Bit errors (electrical interference, cosmic rays…)
    - Loss (everything is buffering!)
    - Link and node failure
- Sent datagrams may be delayed, dropped, reordered, or duplicated
- These issues are exposed to the application

TCP: Transmission Control Protocol

- Uses IP to provide *reliable* byte streams from process-to-process over connections
  - TCP/IP sends *segments* through the network
- Provides illusion of a reliable “pipe” between two processes on two different machines
- Masks lost & reordered segments so applications don’t have to worry
- Handles congestion and flow control
- Most Internet applications use TCP

TCP: The Appearance of Reliability

- Guaranteed delivery
- Session-oriented
- Step 1: Set up a session



Next: Ensure Data Arrives

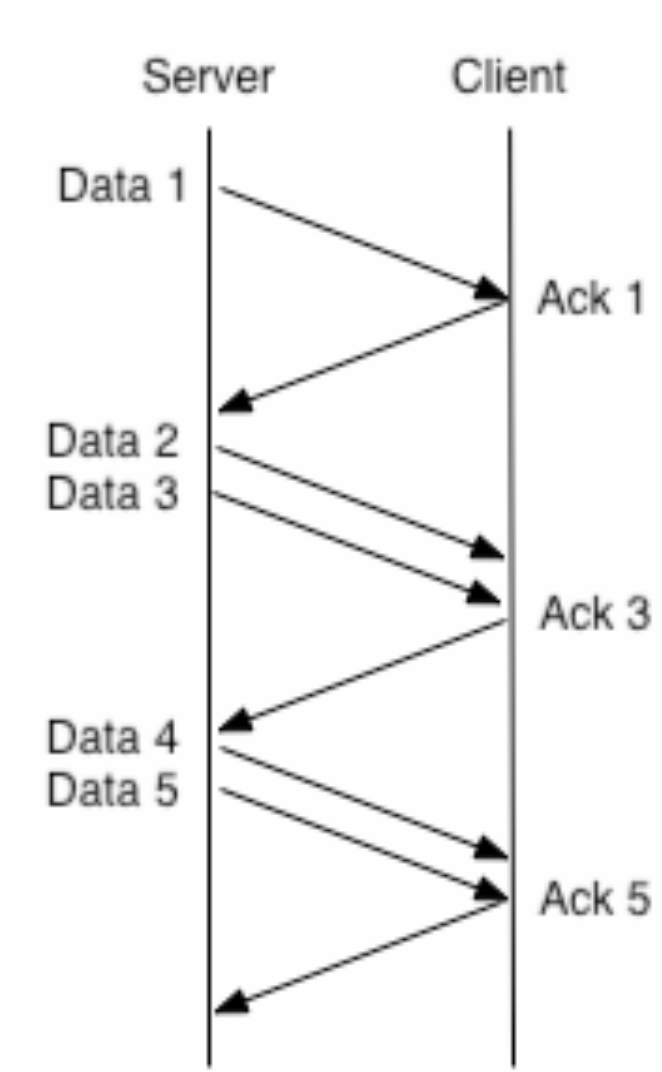
- Solution: use ACKs
  - Sender sends a segment and sets a timer
  - Receiver receives a segment and sends an ACK (acknowledgement)
  - Sender receives ACK and clears timer
  - (If timer goes off, start again)
- Implementation Details:
  - Buffers at sender, receiver, and router
  - Data is ACKed at the receiver as it is read from the buffer by the application
    - Until that point, segments could still be dropped
- Problem:
  - Bandwidth is 1 segment per round trip time (RTT)

- Solution: We want multiple outstanding segments at a time

Update: Ensure Data Arrives Efficiently

- New solution: pipeline old solution
  - Multiple segments in flight
  - Resend un-ack-ed segments after timeout
- Optimizations
  - Cumulative acks
    - ack i acks segments 1 to i
    - Immediate resend on nack
  - Delayed acks
    - For bidirectional communication, use application response as implicit ack

TCP Example



Question

The ACK in the TCP protocol implies:

A. Message received
B. Segment arrived safely
C. Transmission is complete
D. Nothing

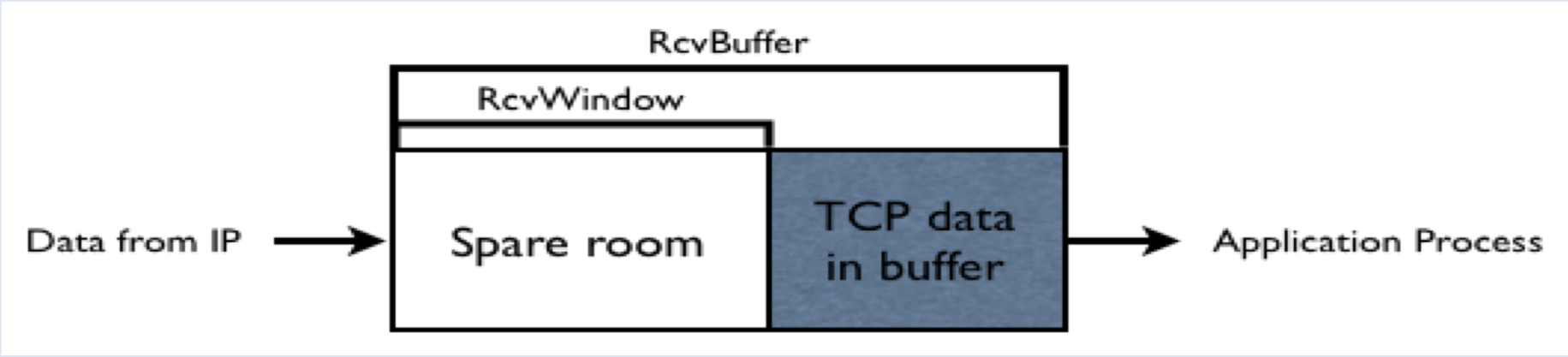
We Still Have Questions…

How many segments can we send before waiting for the cumulative ACK?

TCP/IP

TCP Flow Control

- Receiver has a buffer to receive incoming data (RcvBuffer)
  - If buffer overflows, segments are dropped!
- Sender estimates free buffer space at the receiver (RcvWindow)
  - Sender tracks how much data has been sent vs. how much is known to be received (by ACK)
- Sender ensures that the amount of data that has been sent but not ACKed is <= the free buffer space at the receiver



We Still Have Questions…

I mean… that's great and all, but what if the data is being lost because the network is heavily loaded?

TCP Congestion Window

- Sets maximum number of bytes that can be sent without acknowledgement (ACK)
  - Prevents sender from overflowing receiver's buffer
- Dynamically sized to the network environment
  - Increases in size until loss occurs
- Implications on performance
  - Larger windows suited to high-latency networks
  - Smaller windows suited to high-loss networks

TCP Congestion Control

- Congestion Control algorithm is centered around the size of the congestion window
- Five major components
  - Additive increase, multiplicative decrease
  - Slow start
  - Reaction to timeout events
  - Round trip variance estimation
  - Exponential retransmit timer backoff

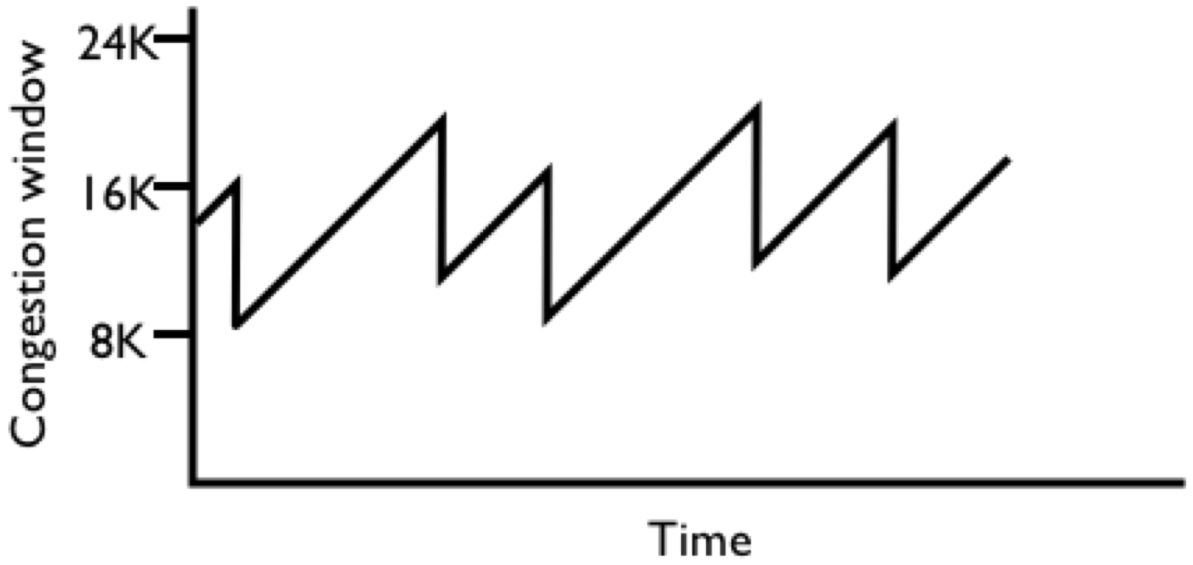
TCP/IP Congestion Control

TCP Congestion Control

- Congestion Control algorithm is centered around the size of the congestion window
- Five major components
  - Additive increase, multiplicative decrease
  - Slow start
  - Reaction to timeout events
  - Round trip variance estimation
  - Exponential retransmit timer backoff

Additive Increase, Multiplicative Decrease

- Additive increase
  - Congestion window will grow by one segment size every roundtrip
- Multiplicative decrease
  - After a loss event, congestions window is halved
  - Never less than one segment size



Slow Start

- At start, congestion window is set to one segment
- But growing linearly from there is too slow.
- So…
  - During slow start phase, exponential growth...
    - Congestion window grows by 1 segment size for each received ack
    - Results in doubling every RTT
  - Until first loss event occurs…

Reaction to Timeout Events

- Drop congestion window to one segment size
- Slow start mode until congestion window reaches some threshold
  - typically half of its size before the loss
- Additive increase after threshold

But wait!

How do we set the timer?

RTT Variance and Retransmit Backoff

- Original TCP set timeout to twice the estimated RTT
  - But with high load (above 75%) RTT can vary by 16X
    - Lots of unnecessary sends under load (bad)
- Current versions set timeout to
  - Estimated RTT + 4 x MeanDev(RTT)
- Retransmit backoff is exponential
  - Provably needed for stability
  - This is why web browser stalls for 5 sec, then for 10 then...
    - Hint: hit reload if page no there after 5 sec

TCP and the OS

TCP is implemented in the OS.

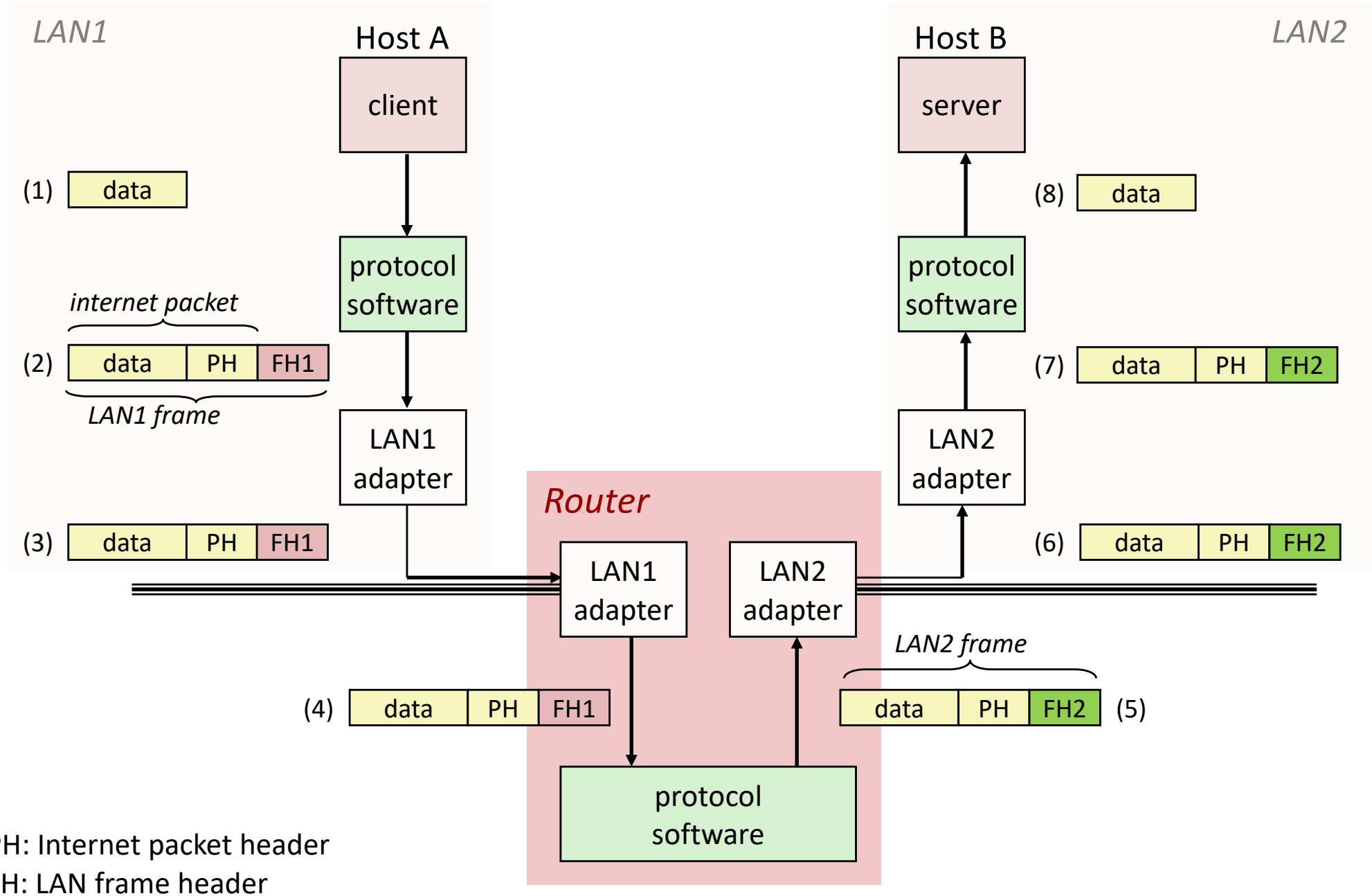
Thus, inside the kernel we must:

- Track unacknowledged data
  - Keep a copy, keep timer
  - Receiver re-orders out of order segments
  - Where does it keep all this information?
    - The PCB… *Protocol* Control Block!
- Must ACK received segments very quickly
- Must decide when to wake processes on receipt
  - Is there more data coming? Should it wait?
- Must decide when to send data
  - If send is small, should it wait for more?


I TCP/IP
but mostly IP

Considering the Cost

Transferring Data Over an internet

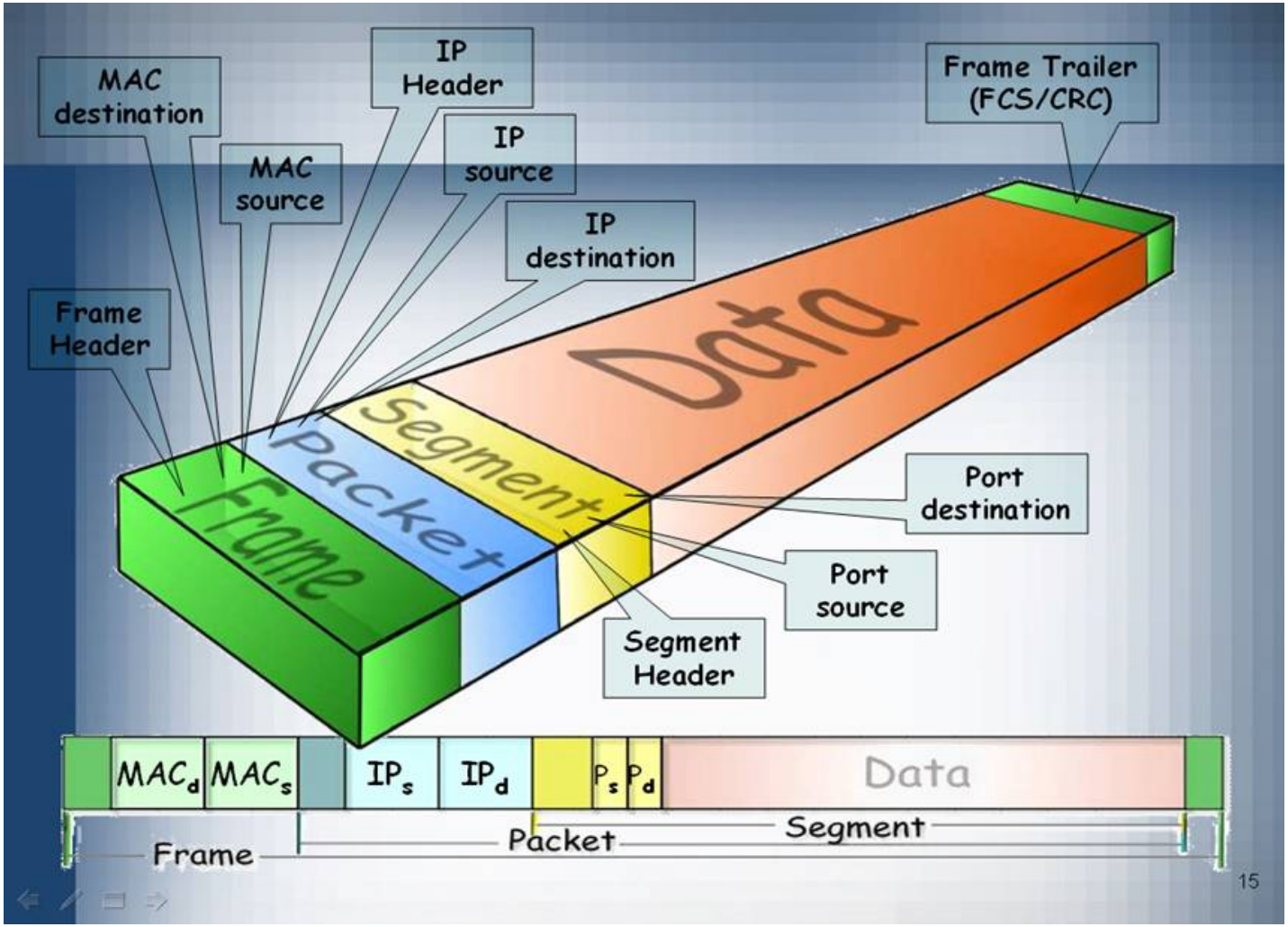


PH: Internet packet header
FH: LAN frame header

Costs

- Across the network:
  - Latency: time for one byte to go from one place to another
    - Latency from New York to San Francisco:
      3000 miles * 1 sec/186,282 miles= 15 ms
  - Throughput: Maximum bytes per second (bandwidth)
  - *Note: Bandwidth is not the whole story!*
  - Key to good performance:
    - LAN: minimize overhead
    - WAN: keep the pipeline full!
- On the source and destination machines:
  - Overhead: CPU time to put the data on or pull it off the wire. Includes protocol processing time.

Structure of “Packet”



The Network Layers

- Layer 2 communicates from one piece of hardware to the next using MAC addresses and frames
- Layer 3 allows communication from one host to another across various Layer 2 networks using IP addresses and packets
- Layer 4 allows communication from one process to another using ports and segments (TCP/IP) or datagrams (UDP)
- (But we call everything packets)
- Shared resources can lead to contention
  - TCP/IP is careful to avoid bringing down the network through congestive failure

Hardware and Software Organization of an Internet Application

