Networks

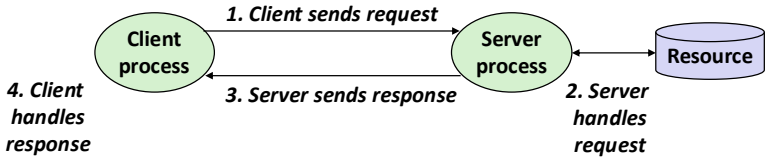
CS 105: Computer Systems
Lecture 14

Melissa O'Neill

March 11, 2026

A Client-Server Transaction

- **Most network applications are based on the client-server model:**
  - A ***server*** process and one or more ***client*** processes (running on ***hosts***)
  - Server manages some ***resource***
  - Server provides ***service*** by manipulating resource for clients

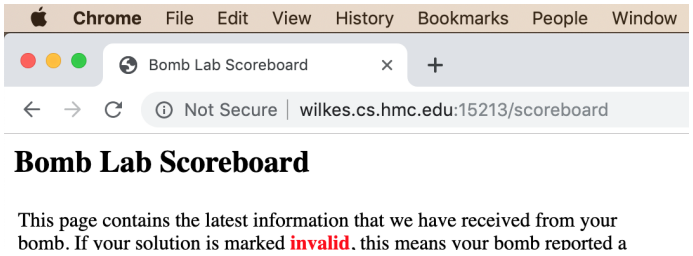


- **A connection is uniquely identified by the *socket addresses* of its endpoints (*socket pair*)**

```
(client_addr:port, server_addr:port)
```

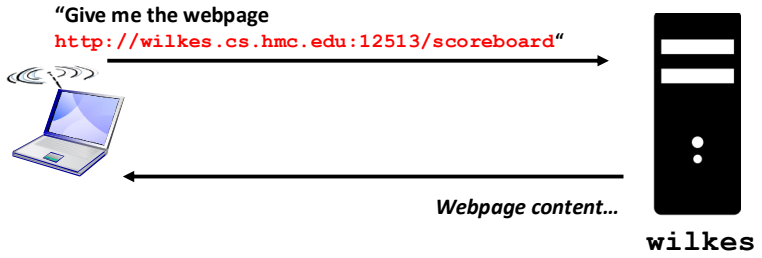

Exercise

- **Consider the Bomb Lab Scoreboard (screenshot below). Discuss with a neighbor:**
  1. Can you guess the server address? Port?
  2. What is the client in this transaction? Can you determine any address or port information?



Example: Web page request

- **Web browser asking for Bomb lab scoreboard**



Sockets and Services

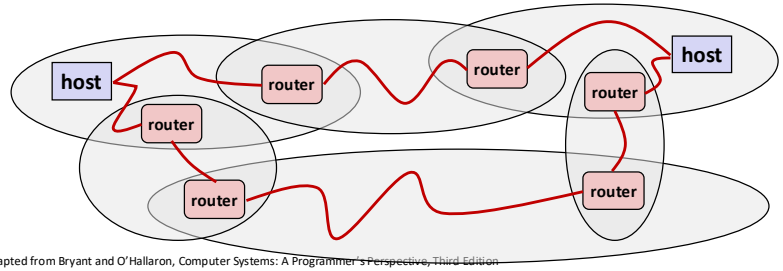
- **Popular services have permanently assigned *well-known ports* and corresponding *well-known service names*:**
  - `ssh` servers: 22/ssh
  - email server: 25/smtp
  - Web servers: 80/http, 443/https

Mappings between well-known ports and service names is in the file `/etc/services` on Linux machine

- **A process could also be identified with *an ephemeral port*, assigned automatically when client makes a connection request**

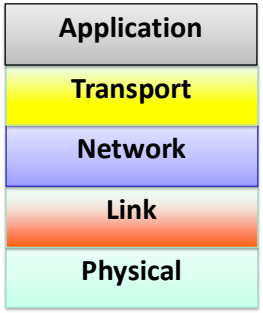
Computer Networks

- **A *network* is a hierarchical system of boxes and wires organized by geographical proximity**
- **An *internetwork* (*internet*) is an interconnected set of networks**
  - The **Global IP Internet** (uppercase "I") is the most famous example of an internet (lowercase "i")



Protocols and the 5-Layer Network Stack

- **A *protocol* defines format and order of messages exchanged between communicating entities, as well as actions taken when messages sent/received**



Application Layer

- **Network application is a pair of processes sending messages over a network**
  - Client process and server process communicate by sending streams of bytes over ***connections***
  - Application endpoint of a connection is an open file called a ***socket***
- **Abstraction provided: don't need to care about how lower network layers are implemented**
  - *Door analogy*: process sends a message out its door (socket) and assumes there is transportation available to get the message to the recipient process' door (socket)

Programmer's Perspective

- **To open socket, process needs to specify destination host and the appropriate process on that host**
  - Hosts are mapped to a set of 32-bit ***IP(v4) addresses***
    E.g., `134.173.42.100` is Knuth
  - (IPv6 uses 128-bit addresses)
  - ***Port*** is 16-bit integer that identifies a process

- **IP addresses are mapped to set of identifiers called Internet *domain names***
  - 134.173.42.100 is mapped to www.cs.hmc.edu
  - Managed by **Domain Naming System** (DNS)

```
linux> nslookup knuth.cs.hmc.edu
Address: 134.173.42.100
```

(some output omitted)

Exercise

- **Typically servers are designed to handle connections from *multiple clients at the same time.* Considering what you've learned in recent lectures, how might a server handle multiple clients?**

Transport Layer

- **At application layer, programmer can only decide *which* transport layer protocol to use**

- **Transport layer protocol breaks up message from higher level into pieces called *segments* (also called *datagrams*)**
  - Application layer message (or part of it) is the **payload**
  - Segment = header information + the payload

- **Example protocols**
  - Transmission Control Protocol (**TCP**)
    Provides reliable, ordered segment delivery process-to-process
  - Unreliable Datagram Protocol (**UDP**)
    Provides unreliable delivery process-to-process

Exercise

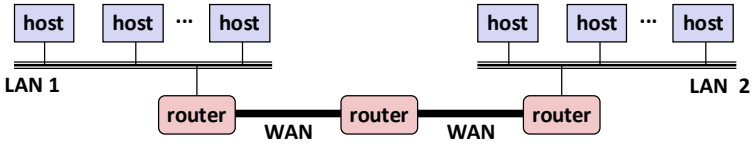
- **Computer networks can be *unreliable*: messages could get dropped on the way to the destination because the network is congested, or different messages could take different routes between hosts.**

- **TCP provides reliable, ordered delivery. With people around you, brainstorm how you might implement this reliable message-passing, for example considering:**
  1. Suppose host A sends five TCP segments to host B. How could host B know which order they should be in?

  2. How could host A know that one of its segments made it successfully to host B? What should A detect a dropped segment? What should A do about it?

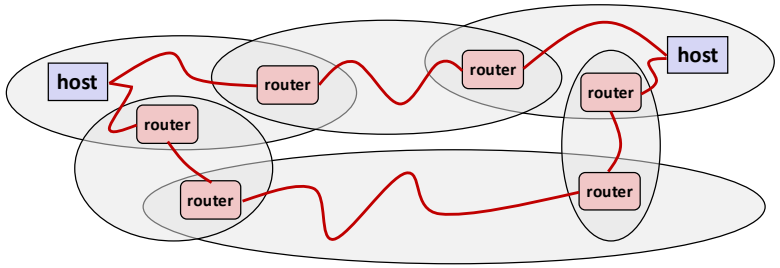
Network Layer: an internet

- Can connect multiple LANs to form an *internet* using specialized computers called *routers*
  - Routers allow LANs with incompatible technologies (e.g., Ethernet vs. IEEE 802.11 aka WiFi) to communicate



- Routers direct information in packets from source to destination IP addresses
  - IP = internet protocol

Logical Structure of an internet



- Ad hoc interconnection of networks
  - No particular topology
- Send packets from source to destination by hopping through networks
  - Router forms bridge from one network to another
  - Different packets may take different routes

Viewing network hops with `traceroute`

```
Beths-MBP:tmp beth$ traceroute www.cs.hmc.edu
traceroute to kay.cs.hmc.edu (134.173.42.2), 64 hops max, 52 byte packets
 1  dsldevice (192.168.1.254)  3.046 ms  3.039 ms  2.584 ms
 2  75-27-240-1.lightspeed.psdnca.sbcglobal.net (75.27.240.1)  4.339 ms  4.0
 3  71.147.184.13 (71.147.184.13)  4.042 ms  3.437 ms  5.670 ms
 4  12.123.136.190 (12.123.136.190)  5.317 ms  11.659 ms  7.352 ms
 5  ggr2.la2ca.ip.att.net (12.122.128.101)  6.081 ms  6.365 ms  3.860 ms
 6  ae-5.a00.lsanca20.us.bb.gin.ntt.net (129.250.9.61)  4.655 ms  5.435 ms
 7  ae-3.r01.lsanca20.us.bb.gin.ntt.net (129.250.2.233)  8.244 ms  4.862 ms
    ae-3.r00.lsanca20.us.bb.gin.ntt.net (129.250.2.253)  4.776 ms
 8  ae-6.r22.lsanca07.us.bb.gin.ntt.net (129.250.6.46)  12.000 ms
    ae-8.r23.lsanca07.us.bb.gin.ntt.net (129.250.6.48)  4.419 ms
    ae-6.r22.lsanca07.us.bb.gin.ntt.net (129.250.6.46)  6.518 ms
 9  ae-1.r00.lsanca07.us.bb.gin.ntt.net (129.250.3.17)  4.395 ms
    ae-2.r01.lsanca07.us.bb.gin.ntt.net (129.250.4.107)  29.225 ms
    ae-2.r00.lsanca07.us.bb.gin.ntt.net (129.250.3.238)  4.906 ms
10  ae-1.a02.lsanca07.us.bb.gin.ntt.net (129.250.3.234)  8.298 ms  9.131 ms
11  ntt-los-nettos-usc.ln.net (165.254.21.242)  5.903 ms  6.942 ms  6.816 ms
12  ln-cit1-clar2016.ln.net (130.152.181.91)  7.563 ms  6.238 ms  8.601 ms
13  hmc-a.router.claremont.edu (134.173.253.23)  6.502 ms  9.916 ms  6.574 m
14  * * *
15  kay.cs.hmc.edu (134.173.42.2)  10.950 ms  5.825 ms  5.900 ms
```

Link Layer: Ethernet example

- **Ethernet segment consists of a collection of *hosts* connected by wires (e.g., twisted pairs) to a *switch***
  - Forms a local area network (LAN)
  - Spans room, building, and can be combined to span a campus
- **Hosts send bits to one another in chunks called frames**
  - Each host and the switch has unique MAC address; E.g., 00:16:ea:e3:54:e6
  - Switch routes frames based on the source and destination MAC addresses

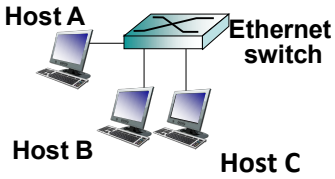
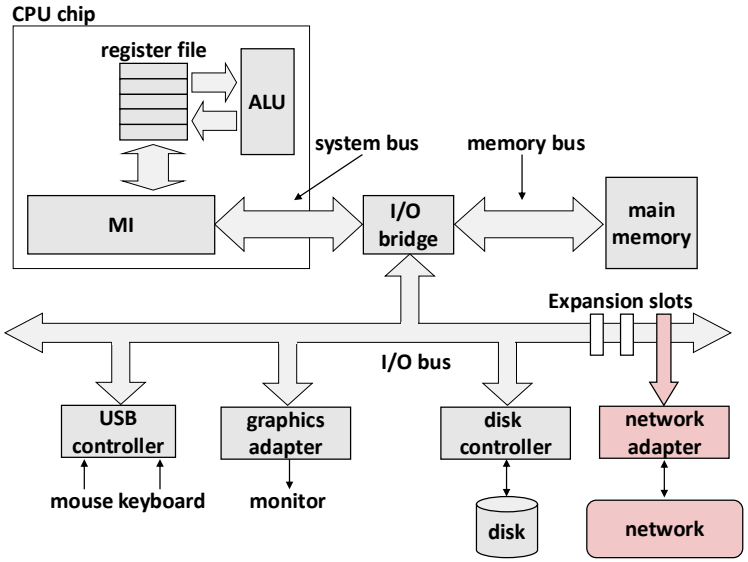


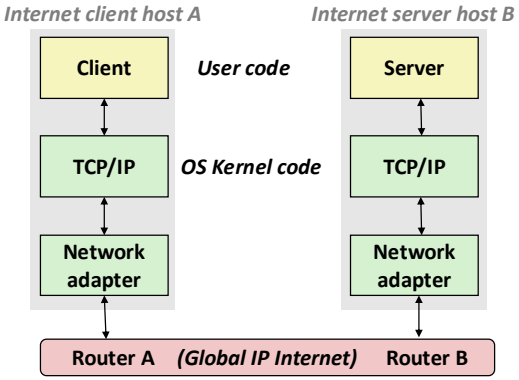
Figure adapted from *Computer Networking: A Top Down Approach, Kurose and Ross*

Hardware Organization of a Network Host



Adapted from Bryant and O'Hallaron, Computer Systems: A Programmer's Perspective, Third Edition

Transferring Data Via Encapsulation



Adapted from Bryant and O'Hallaron, Computer Systems: A Programmer's Perspective, Third Edition