The University Of Alabama in Huntsville Computer Science

cs570

# Background (a)



The University Of Alabama in Huntsville Computer Science

## Background

cs570

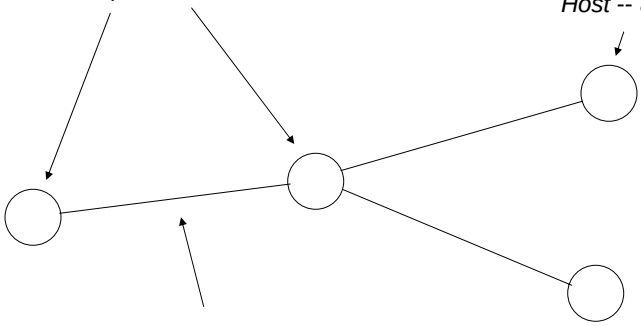
- Basic terminology
- Communication models
- Protocols
- Standard reference models (OSI and TCP/IP)

# Basic Network Diagrams



*Node* -- a computer or a network device

*Host* -- a computer



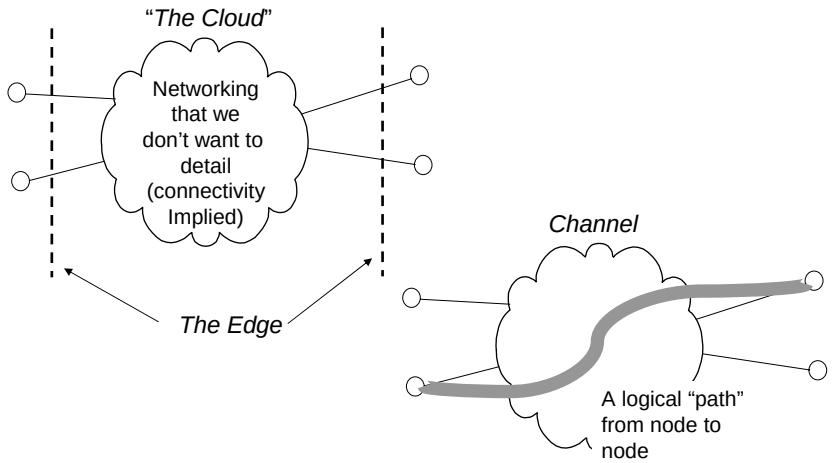
*Link* -- a physical data path between nodes





# Terms

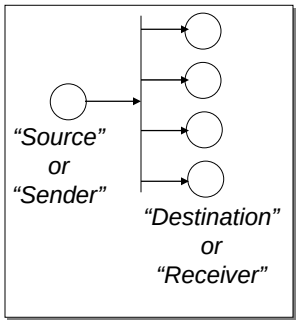




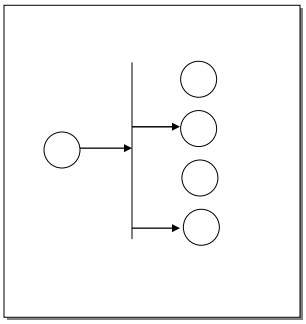
# Some Communication Types



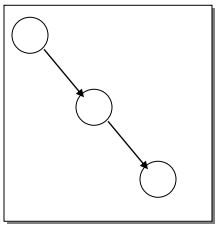
*Broadcast* --
Same message
sent to every receiver




*Multicast* --
Same message
sent to some receivers



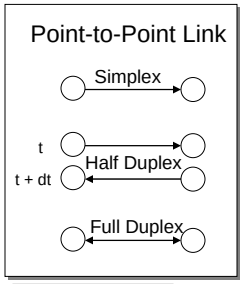
*Point-to-point* --
Messages relayed

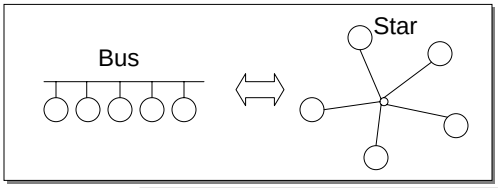


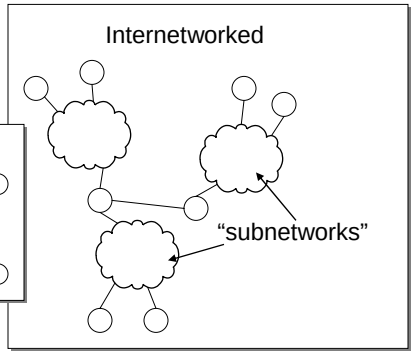


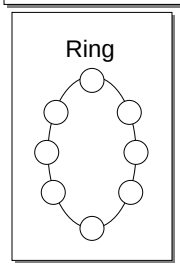
# Some Network Types

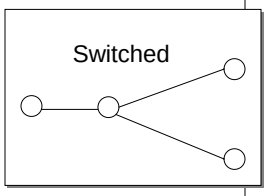




# Some Problem Terminology



***"Bandwidth"***

Means:
1. Data Rate in Bits per second
2. Frequency Range in Hertz

Assume I mean #1.
If I mean #2, I'll say so.

***"K", "M", "G" …***

1. "K" = $2^{10}$, "M" = $2^{20}$, "G" = $2^{30}$
2. "K" = $10^3$, "M" = $10^6$, "G" = $10^9$

Use #1 when you mean a quantity of data
("100MB" = 100 x $2^{20}$ Bytes)

Use #2 when you mean a Rate or distance
("100Mbps" = 100 x $10^6$ bps,
"100Km" = 100 x $10^3$ m)

***"b", "B"***

"b" --> bit
"B" --> 8-bit byte

***"internet" / "Internet"***

"internet" -- short for internetwork
"Internet" -- THE Global Internet





# The Ans



| | # Nodes | Range | Bandwidth |
|---|---|---|---|
| Personal Area Network (PAN) | 1-10 | couple of meters (body, car) | ~1Mbps |
| Local Area Network (LAN) | Up to a few hundred | few hundred meters (building) | 10-1000 Mbps |
| Metro Area Network (MAN) | Hundreds to Thousands | km (city) | 100-1000 Mbps |
| Wide Area Network (WAN) | Tens of thousands and up | 10s-1000s km (country, world-wide) | 1000 Mbps and up |

# Two Models of Communication





## The Postal Service model

- "*Connectionless*" (a.K.A "*packet-oriented*", "*datagram*")
- The "data" carries destination address
- The "network" moves data from place to place, forwarding it toward destination
- There is some overhead associated with reading/ processing the address at each intermediate station, but "setup" overhead is low

***Better than connection-oriented when we have short interactions among changing sets of nodes***

## The Telco model

- "*Connection-oriented*" (aka "*circuit-switched*")
- A source-to-destination path is established beforehand based on addressing info
- "Data" needs no addressing -- it simply follows the established path
- There is some overhead associated with setting up the path and tearing it down, but "forwarding" overhead is low

***Better than connectionless when we have fairly long interactions among a constant set of nodes***

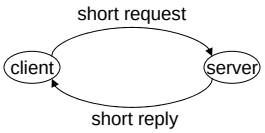


# Two Common Ways That Computers Communicate





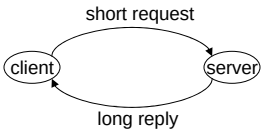
**Conventional Client / Server Communication (e.g. web surfing)**




- Relatively short messages exchanged
- A single client-server episode may be short
- client accesses many servers per unit time
- server serves many clients per unit time
- performance usually not critical

***Fits the Postal Service Model***

**Streaming media (e.g. real-time video)**




- A single client-server episode may be hours long
- client-server pairings are fairly stable
- performance may be critical

***Fits the Telco Model***

# Protocols



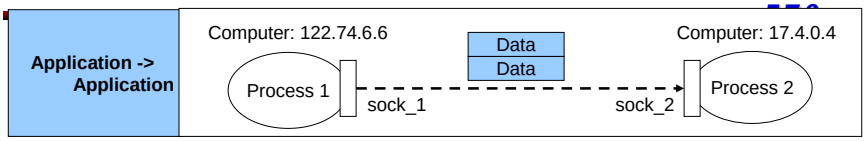
- A *protocol* is a set of
  - **rules** and
  - **formats**

  for data communication between peers. It allows agreement on the meaning and validity of messages.

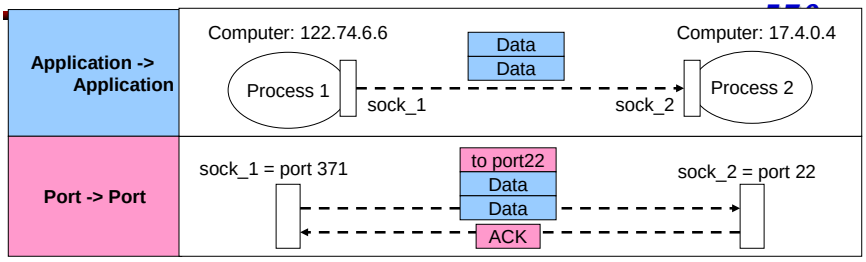




# A Protocol Stack



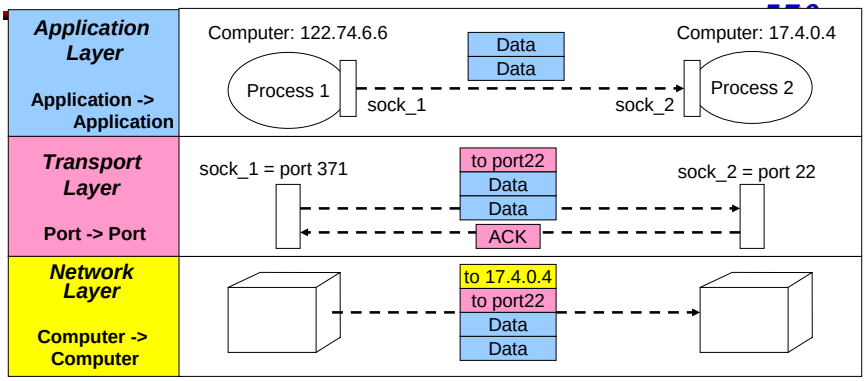
# Protocol Stack



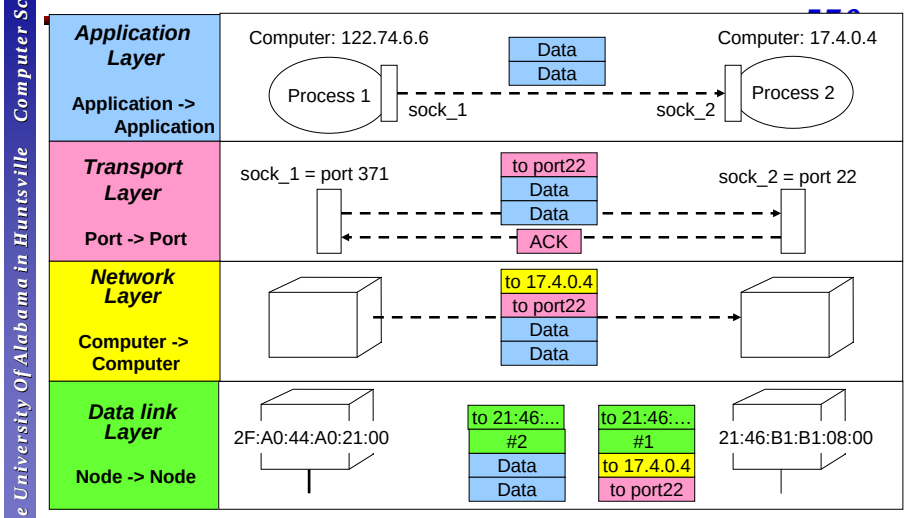





# Protocol Stack

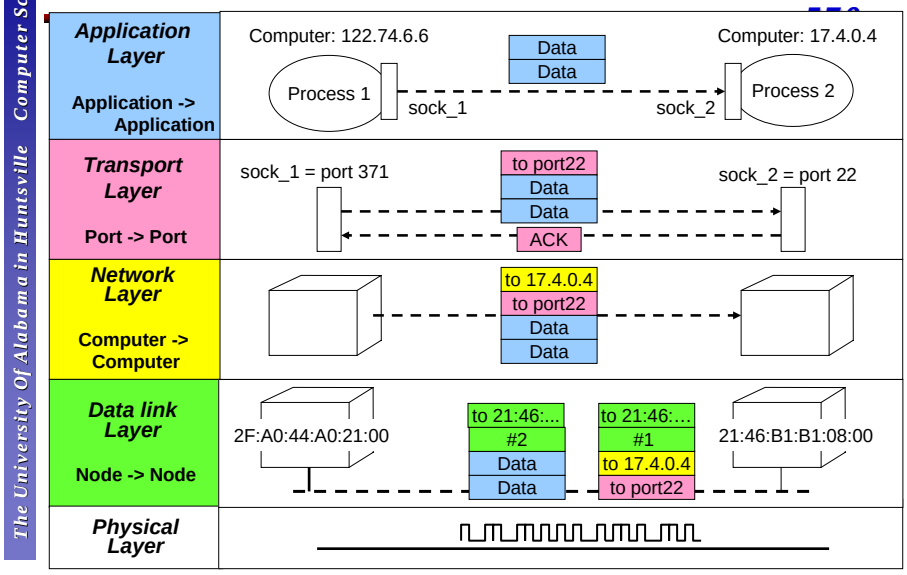


# Protocol Stack



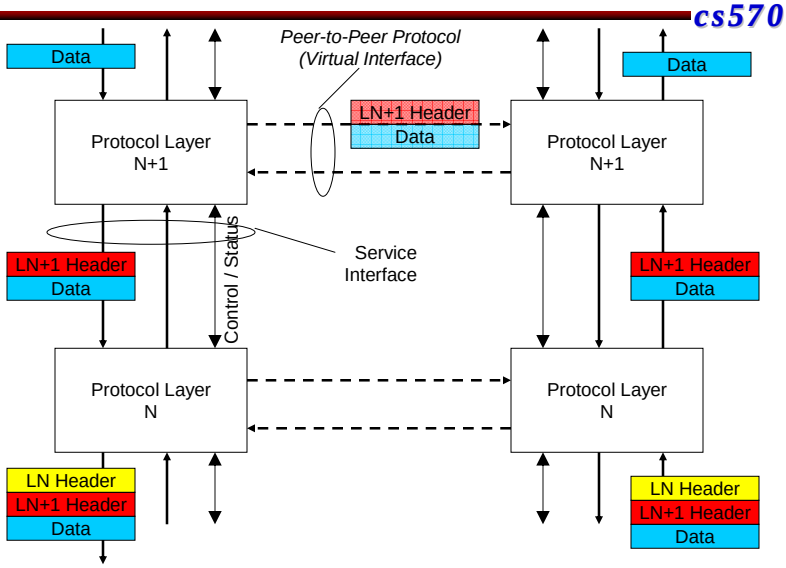



# Protocol Stack



## How the Protocol Layers Interoperate








## Some Things to Note



- The sender's layer N behaves as though it communicates directly with its peer at the receiver
- Each layer adds some header/ trailer info -- this is called **encapsulation**

# Why a Layered Protocol?



- Modularity -> flexibility
- Higher layers don't depend on the way the lower layers do their job -- this limits the "breakage" when we incorporate new protocols or technologies
- Lower layers don't depend on the type of information that the higher layers send down -- this makes it possible to build simple "exception-free" code
- By limiting the functions of each layer, we can build efficient, fast-running code.





# The ISO OSI Reference Model



*ISO OSI = International Standards Organization Open Systems Interconnect*

| | | |
|---|---|---|
| L7 | Application | User applications |
| L6 | Presentation | Provides utility functions (e.g, format conversion)<br>(These functions usually bundled into Application layer) |
| L5 | Session | Manages end-to-end "communication episodes", ties together data streams<br>(These functions usually bundled into Application layer) |
| L4 | Transport | Manages "port-to-port" transfers<br>Data unit is the **segment** |
| L3 | Network | Moves data across network(s) "computer-to-computer"<br>Data unit is the **packet** or **virtual circuit** |
| L2 | Data Link | Manages data movement across links: "node-to-node"<br>Data unit is the **frame** |
| L1 | Physical | Transfers raw binary data across links<br>Data unit is the **bit** |

**AKA:**
- **"The OSI Stack"**
- **"The Wedding Cake Model"**

# How the Heck Am I Supposed to Remember That?



| | | | | | | |
|---|---|---|---|---|---|---|
| Physical | Data Link | Network | Transport | Session | Presentation | Application |
| PLEASE | DO | NOT | TAKE | SAUSAGE | IZZA | AWAY |



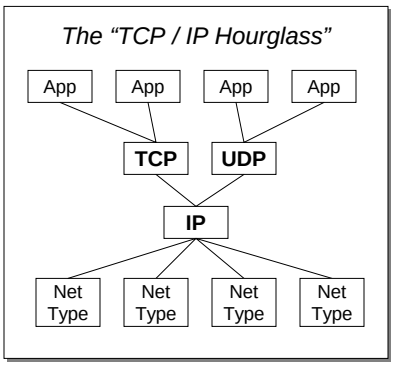


# The TCP/IP Reference Model



TCP/IP = “Transmission Control Protocol / Internet Protocol”

| *OSI* | *TCP / IP* |
|---|---|
| Application | Application |
| Presentation | XXX |
| Session | XXX |
| Transport | TCP |
| Network | IP |
| Data Link | Often Combined |
| Physical | Often Combined |



# Why Two Reference Models?



Both emerged about the same time, but radically different philosophies

| OSI | TCP / IP |
|---|---|
| *Based on standards developed by international organization* | *Based on programmer's, user's needs* |
| *Many years to "paper", early implementations bad* | *Solid running code early, widespread free distribution (BSD unix)* |
| *Mandated, fixed standard* | *Open standard w/ developer participation* |
| *"Useless" layers* | *Layers do what programmers need to do* |
| *Only connection-oriented service is visible to user* | *Both connection-oriented and connectionless service visible* |

The result:

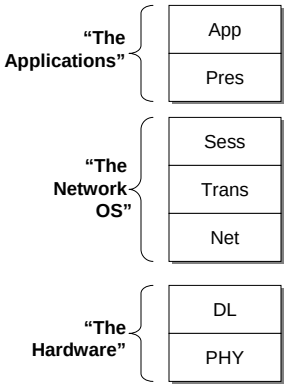
TCP/IP dominates by far, but OSI still useful as a reference model





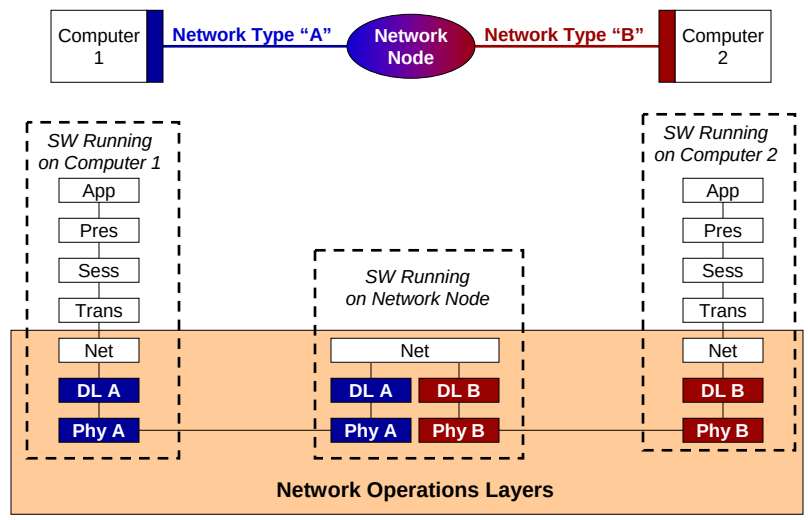
# The reference model we'll use and how to think about it





# Where Is the Protocol Software in a Network?








# Applications Interface: Sockets



**Most OS's provide a "socket" construct that serves as the connection between an application and the networking protocols**