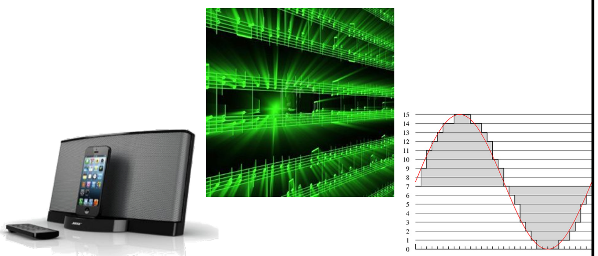
Penn Engineering
ESE



Lecture #11 – Networking

ESE 150 – DIGITAL AUDIO BASICS

TODAY'S SEATING (FOR INCLASS EXERCISE)

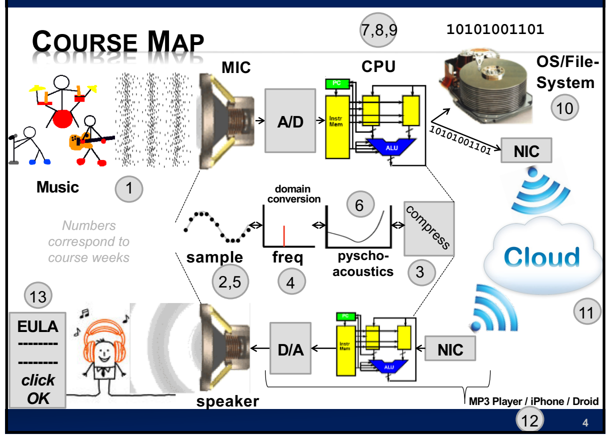
Screen

First Row	Student	(empty)	Student
	Student	(empty)	Student
	Student	Student	Student
	Student	(empty)	Student
	Student	(empty)	Student

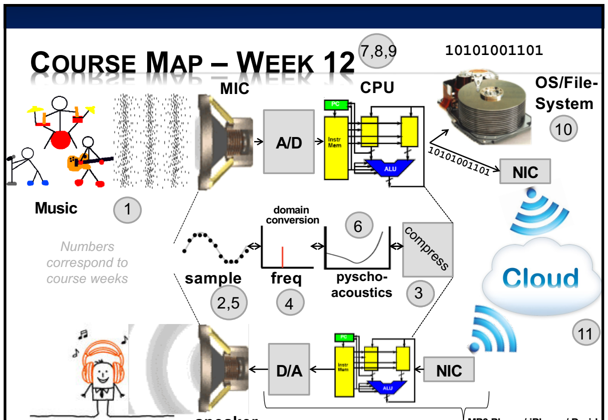
LECTURE TOPICS

- **Where are we on course map?**
- **Networks**
  - Communicating Between Machines
  - Bandwidth Requirements
  - Technology Costs
  - Network Layering
    - Transport
    - Network – Routing – what can go wrong?
    - Physical (physical layer independence)
    - By end: seen TCP/IP basics
- **Next Lab**

COURSE MAP



COURSE MAP – WEEK 12



WHAT WE'LL COVER TODAY…

- **Established can**
  - represent things (sound, computations, images, movies, 3D objects…) as bits
  - Store and reconstruct from bits
- **If we can send bits between machines…**
  - Communicate (from MP3 player to Cell Phone)
  - Transport (from scanner and 3D printer to a transporter?)

Communicating between Machines

Fundamentals of Networks

Networked Systems

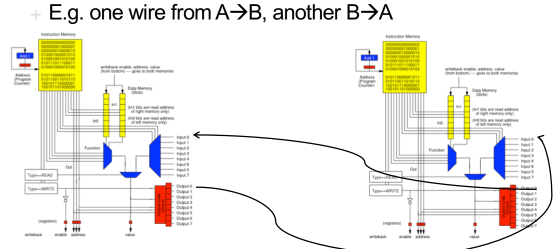
- **Today**
  - We expect our computers to be networked
    - Google, wikipedia, Email, IM, …
  - Can work stand alone
    - Airplane mode?
  - But, are crippled when not connected
  - Phone isn't a phone unless its networked

Minimal Setup

- **Have two computers**
  - think raw processors for the moment
- **Want them to communicate**
  - Send an mp3 file from A to B

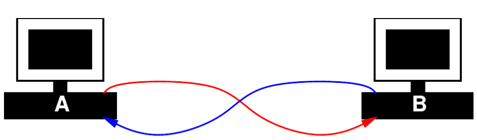
Physical Connection

- **Place an I/O datapath in each computer**
- **String wire between computer's IO peripheral**
  - E.g. one wire from A→B, another B→A



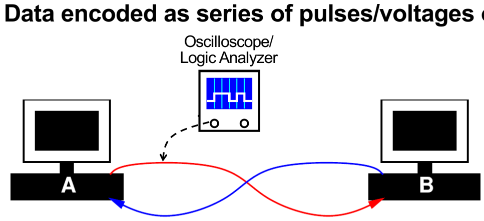
Physical Connection

- **Place an I/O datapath in each computer**
- **String wire between computer's IO peripheral**
  - E.g. one wire from A→B, another B→A



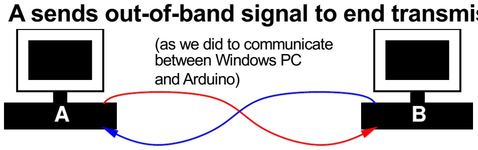
Signaling

- **Communicate with Voltage pulses**
  - A pulls line low (0)
  - B senses low (0) line
- **Data encoded as series of pulses/voltages on line**



Communication Basic Steps

1. Start program on B to receive data (file)
2. Start program on A to send data (file)
3. B waits for valid symbols
4. A sends data
5. B receives
6. A sends out-of-band signal to end transmission

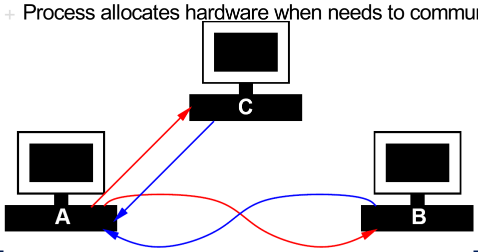


Preclass 1

- How many computers does your laptop communicate with?
  - E-mail
  - Weather
  - Canvas, Piazza
  - Source code repositories (svn, git, …)
  - eniac
  - Web servers
    - Seas, news, facebook, youtube, wikipedia, google, ….
  - iTunes, Windows Update

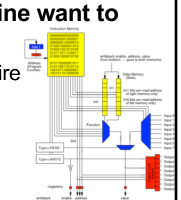
Multiple Tasks – *Multiple Wires?*

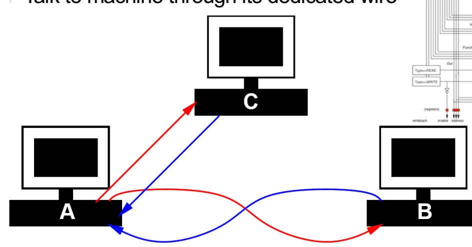
- Back to wired connections
- E.g. download song and browse
  - Could have a separate interface/wire for each application
  - Process allocates hardware when needs to communicate



Connect to Multiple Machines

- Add interface/wire for every machine want to talk to
  - Talk to machine through its dedicated wire

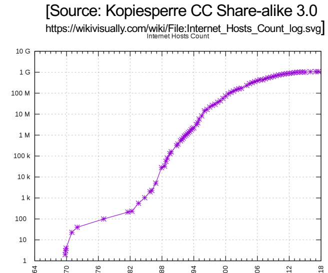




Scalability

[Source: Kopiesperre CC Share-alike 3.0 https://wikivisually.com/wiki/File:Internet_Hosts_Count_log.svg]

- Do we like where this is going?
- Hosts on Internet



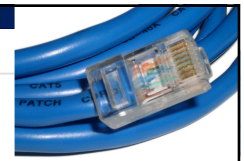
- How many things are connected to Internet?
  - Estimate as of 8 Billion connected devices!
  - Growing to 50—100 Billion in next few years…

How many connections?

- **Conclusion:** need to look at capacity as well as scalability of a network solution

Bandwidth Requirements and Costs

Wires



- How fast can I send data over a wire?
- Consider a Category-5 Ethernet cable
  - Bandwidth (bits/s)
    - 1Gbit/s – 1000Base-T (Gigabit ethernet)
  - Latency/transit time (distance/time)
    - 0.64 c [c=speed of light = 3×10^8 m/s]
    - 0.192 m/ns or roughly 5ns/m

[image: http://en.wikipedia.org/wiki/File:Cat_5.jpg]

Comparison: Audio (Preclass 3)

- Real-Time stereo (2-channel) MP3
  - 128Kbits/s
  - How many can share 1Gbit/s link?
- How long to download 3 minute song at full rate?
- How long for first bit to travel across 4000km wire at 0.6 × speed-of-light?

Comparison: Video (Preclass 3)

- HDTV compressed
  - Around 36Mbits/s
  - How many can share 1 Gbit/s link?

Costs (Preclass 4)

- Cat 5e per foot ~ $0.20/foot
  - Say $0.60/m
  - Raw wire
    - Ignoring handling to run
    - Ignoring rent/lease/buy land to run
  - Philly → San Francisco: ~4,000km
  - Wire cost?

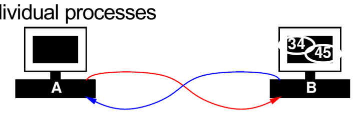
Implications?

- Today's wire bandwidth exceeds the throughput needs of any real-time single-stream data
  - Can afford to share the wire
- Wires are not cheap
  - Cannot afford not to share the wire

Sharing (Virtualizing) Connections

Sharing Link

- **Idea: Tag data with target**
  - "this is for process 34"
  - "this is for process 45"
- **Have transport layer deal with…**
  - Mixing data from separate streams
  - Separating data out into individual streams
  - Delivering to individual processes



34: and then she said...

45: 80004010 00001200

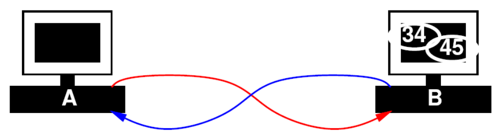
Packet

and then she said...	34

- **Begin to form a packet**
  - Header: says where to go
  - Payload: the data to send
- **Header:**
  - Added, consumed by network handling in routing
- **Payload:**
  - Only thing seen by the application processes

Packets

80004010 00001200	45	and then she said...	34

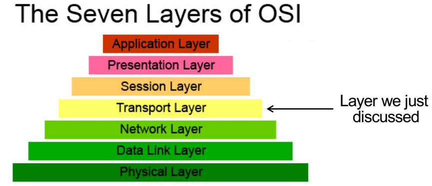


Transport Layer

- **Call this the "Transport" Layer**
  - responsible for delivering data to the individual application process on the computer

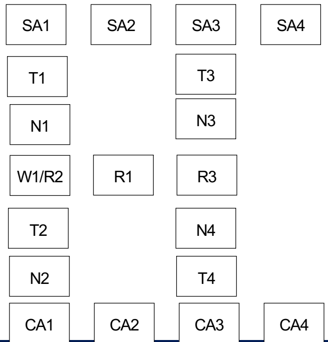
OSI Model of a Network

The Seven Layers of OSI



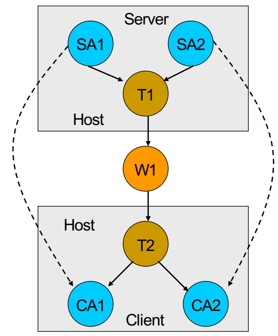
- **OSI – Open Systems Interconnection Reference Model**
  - Developed in 1980's; maintained by ISO
  - Abstract different functions of a network into layers
    - Each layer only knows about layer above and below (at the interface level)
  - Think of it like this: your "Application" doesn't know if its on a wired or wireless network (*physical layer*)…but it knows it needs a network!

Possible Role Assignment



Simulation 1

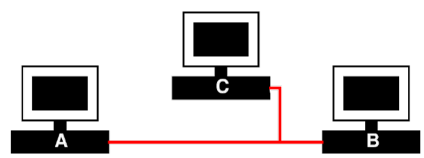
- **Send 4 verses or digits from each**
  - from song-server-app, π-server-app
  - to song-listener-app, π-consumer
- All go through one wire W1
- T1 – Transport tagging
- T2 – Transport sorting



Virtualize Physical Wires

Start Simple

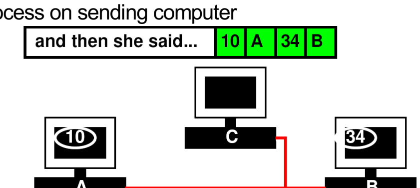
- **Add more computers to same pair of wires**



- All computers on wire see all the data on the wire
  - *How do computers know who the message is for?*

Extended Packet

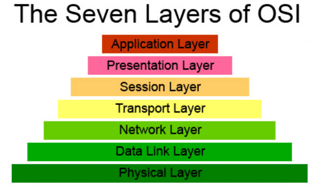
- **Extend our packet header:**
  - Destination computer
  - Process on destination computer
  - Sending computer
  - Process on sending computer



Network Layer

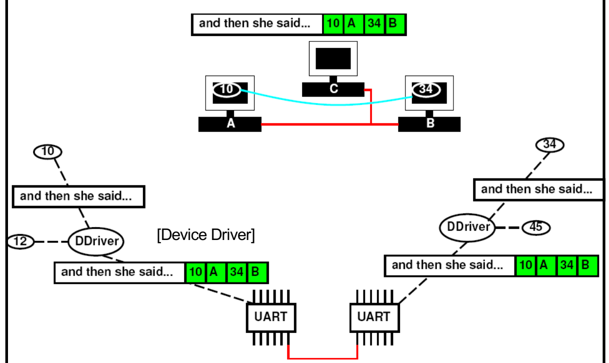
- **responsible for end-to-end (source to destination) packet delivery**

The Seven Layers of OSI



Virtualization Effect

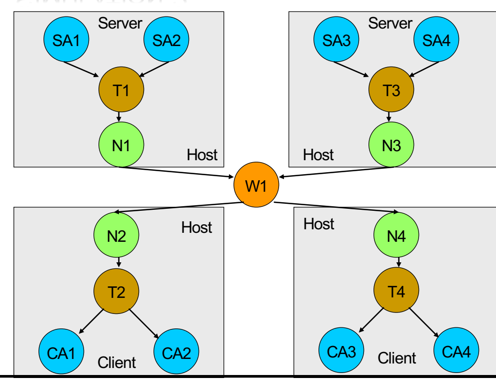
- **Each pair of processes on different computers**
  - Has the view of a point-to-point connection
  - Each process, thinks it "owns the network" and has a dedicated connection to the other node



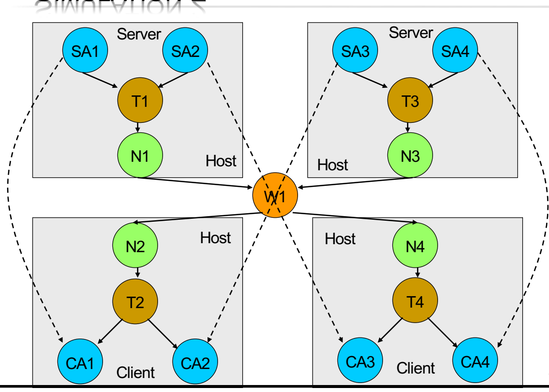
Simulation 2

- **Send 4 verses or digits from each**
  - from song-server serving 2 songs
  - And digit-server serving 2 fundamental constants
  - To two clients

Simulation 2



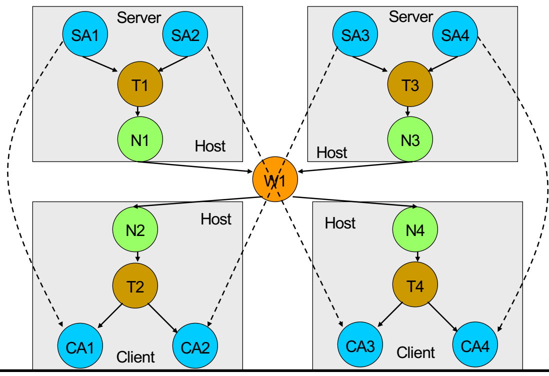
Simulation 2



Simulation 2

- **N1, N3**
  - Add network-layer source/destination packet headers
- **W1 – Wire**
  - Duplicate packets to both destinations
  - Simulate shared wire
- **N2, N4**
  - Look at network-layer source/destination header
  - Discard packets not destined for this computer

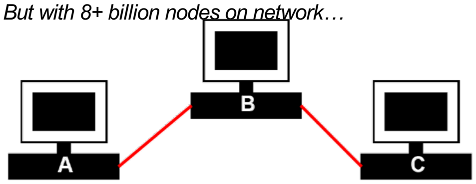
Simulation 2



Extending the Virtual Link

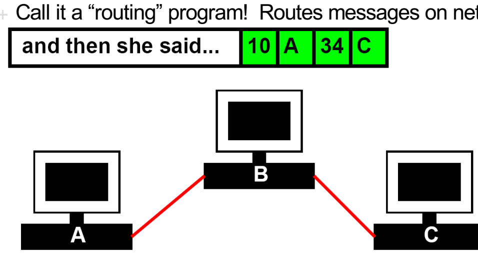
Indirect Connections

- A and B are connected
- B and C are connected
- How get message from A to C?
  - *We could add a wire between A and C…*
  - *But with 8+ billion nodes on network…*



Indirect Connections

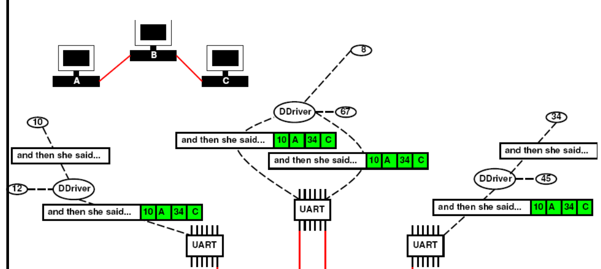
- Run program/process on B to forward messages from A to C
  - Call it a "routing" program! Routes messages on network



Routing

- B runs a general program
  - If packet destined for B, takes it
  - Otherwise, sends on to (toward) destination
- Extension of the network handling process that is sorting data to processes

Routing



Reachability

- **If everyone plays along**
  - We can communicate with any computer reachable *transitively* from my computer
- **Don't need direct connections**

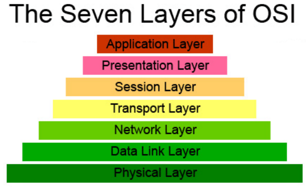
Routing → Route Tables

- **To make efficient**
  - Each computer should route *close* to destination
  - …and not route in circles
- **E.g. compute all-pairs shortest paths (CIS160,121)**
  - Store result, each machine knows where to send packet next
  - How much storage?
    - Cleverness to compress/summarize
  - Additional cleverness to compute incremental updates
    - When add a computer or a link breaks

Network Layer

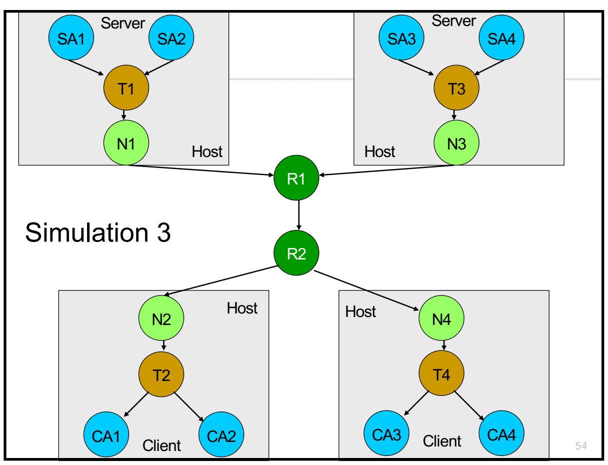
- **Responsible for end-to-end packet delivery**
  - Source to Destination
  - This includes routing packets through intermediate hosts

The Seven Layers of OSI



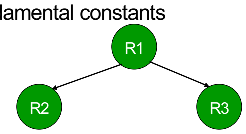
Simulation 3

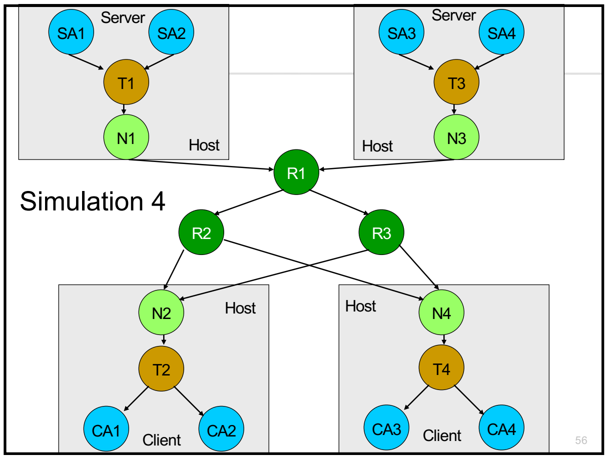
- **Send 4 verses or digits from each**
  - from song-server serving 2 songs
  - And digit-server serving 2 fundamental constants
  - To two clients
- **R1 –** pass along packets to R2 (for now)
- **R2 –** look at address and send to N2 or N4



Simulation 4

- **Send 4 verses or digits from each**
  - from song-server serving 2 songs
  - And digit-server serving 2 fundamental constants
  - To two clients
- **Roles:**
  - 4 server apps
  - Network Interface, 2 servers
  - 3 routers
    - R1 – flip a coin and send to R2 or R3
    - R2, R3 – send to N2, N4 based on address
  - Network Interface for each of 2 clients
  - 4 client apps



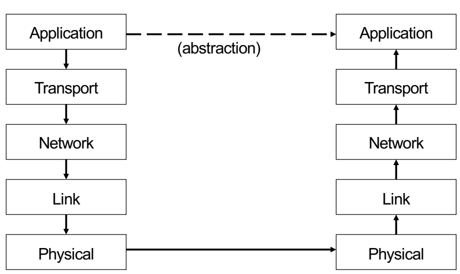


Where are we now?

- **Can communicate**
  - From one process on a computer
  - to any other process on any other computer
  - *if* the two are transitively connected
    - By a set of participating computers which route data
- **Layers have provided "Abstraction"**
  - Processes just see streams of data between the endpoints

57

Layers



58

Protocols

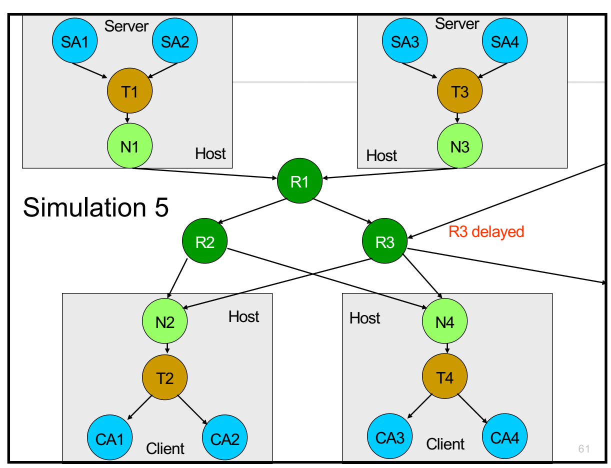
- **So far, we've discussed a protocol called IP:**
  - IP = Internet Protocol
- **Delivery to processes (rather than hosts): UDP**
  - UDP = Unreliable Datagram Protocol

59

Simulation 5

- **Send 4 verses or digits from each**
  - from song-server serving 2 songs
  - And digit-server serving 2 fundamental constants
  - To two clients
- **Deliberately delay data through R3**
  - Model non-determinism in route timing

60



What can go wrong?

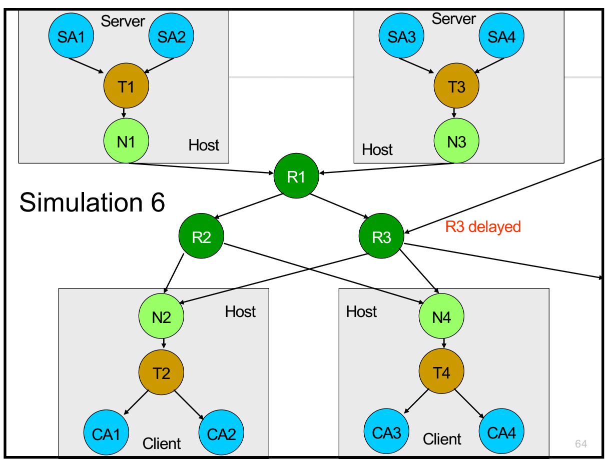
- **Packets arrive out of order**
- **Solution?**
  - Add a sequence number

62

Simulation 6

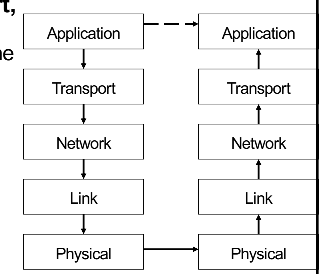
- **Send 4 verses or digits from each**
  - from song-server serving 2 songs
  - And digit-server serving 2 fundamental constants
  - To two clients
- **T1/T3 –** add sequence number to packet
- **T2/T4 –** hold packets, reorder, and deliver in order of sequence number
- **R3** – still delaying packets

63



Abstracting Physical Layer

- **Application, transport, network**
  - Don't really care how the bits are moved from machine-to-machine
- **What are other ways we send bits?**
  - Beyond wires
  - Optically
  - RF/wireless
  - Pneumatic tubes, passing paper notes, SMS Text messages…

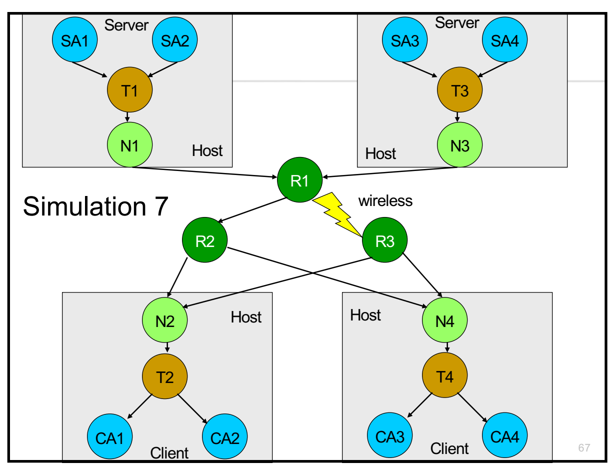


65

Simulation 7

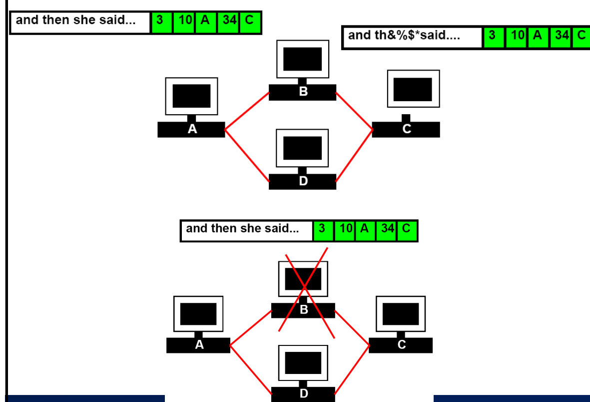
- **Send 4 verses or digits from each**
  - from song-server serving 2 songs
  - And digit-server serving 2 fundamental constants
  - To two clients
- **Roles:**
  - 4 server apps
  - Network Interface for each of 2 servers
  - 3 routers, connect to both servers and endpoints
    - One link is via text messaging
  - Network Interface for each of 2 clients
  - 4 client apps

66



What else can go wrong?

- Bits get corrupted
- Intermediate machines holding messages can crash
- Messages can get misrouted



Data Corruption

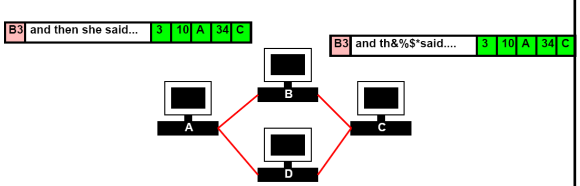
- How do we deal with data corruption?
  - Use redundancy
- Two strategies:
  - Use enough redundancy to correct
  - Use just enough redundancy to detect it
    - Have the sender resend

Data Corruption

- Relatively uncommon
  - Most packets are fine
- We have efficient (low overhead) ways to detect
  - Compute a hash of the message data
  - Highly unlikely one (few) message bit errors will result in same hash
  - → checksum

Revised Packet

- Header
- Data payload
- Checksum



Lost Packet

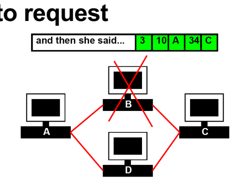
- How can we deal with lost packets?

Lost Packet Strategy

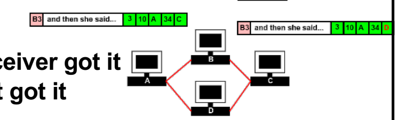
- Sender sends packet
  - But keeps a copy
- Receiver gets packet
  - Checks checksum
  - OK, uses packet and sends ACK
    - "got your last packet in tact"
  - Not ok, discard packet
- Sender
  - Receives ACK, can discard packet and send next
  - No ACK (after timeout), resend packet

Retransmission Discipline

- Don't depend on receiver to request retransmission
  - Why?
- Header may be corrupted
  - Not deliver to receiver



- Only know receiver got it when it says it got it



Corrupted ACK

- What if the ack is lost?
  - Sender resends
- Receiver receives a second copy
  - Oops, don't want that to be interpreted as new data
  - i.e. send: "rm *; cd ..\n"
    - Receive: "rm *; cd ..\n rm *; cd ..\n"

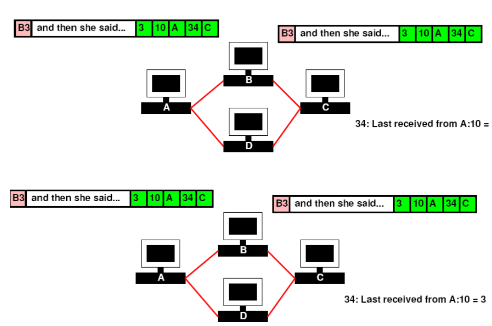
Avoid Duplication

- How can we avoid duplication?

Accommodating Duplication

- Use packet sequence number
  - Keep track of last packet received
  - If receive packet again,
    - Discard the packet

Sequence Numbers



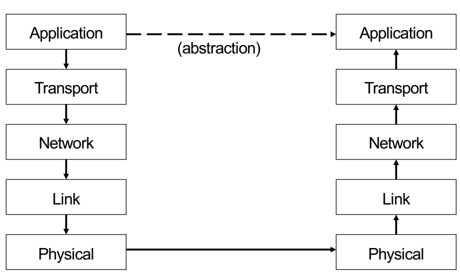
TCP

- **TCP = Transmission Control Protocol**
  - Provides Reliable delivery
  - Deals with
    - Retransmission
    - Duplication
    - Out of sequence / resequence / reconstruction

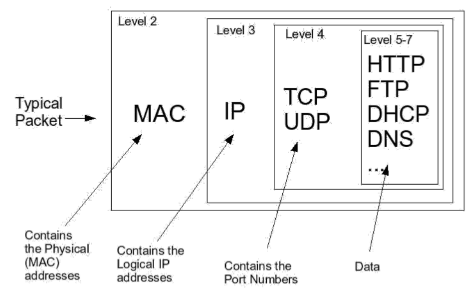
Transport Layer

- **Call this the "Transport" Layer**
  - responsible for reliably delivering data to the individual application process on the computer

Layers



Layers and the Packet



Big Ideas

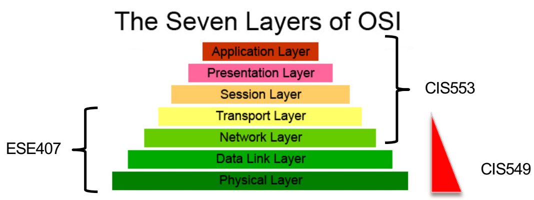
- **Sharing – Network interface, wires**
  - Previously gates, processor, memory
- **Virtualization – datastream abstracts physical point-to-point link**
- **Layering**
  - Divide-and-conquer functionality
  - Implementation hiding/technology independence
  - Reliable communication link from unreliable elements

This Week In Lab

- **Lab 11:**
  - Look at naming, addressing, network diagnostics, …
  - Including a packet sniffer!
    - …see all the bits on the network you aren't supposed to see!
    - Get an appreciation for what is going on, on the lower network layers

Learn More @ Penn

- **Courses**
  - ESE407 – Intro Networks and Protocols
  - CIS553 – Networked Systems
  - CIS549 – Wireless Mobile Communications



86