Last Lecture: Putting it Together

CS 118: Computer Networks

George Varghese

(some slides from Alex Snoeren)

Exam Overview

- Focus on topics since the midterm
  - No question on earlier lectures, but can't completely avoid
  - Do not study before Ethernet
- Roughly the same style, as midterm, maybe 7 questions instead of 5
  - It won't take the whole time
  - Will post a sample final as well (two different old finals)
  - Final review this Wednesday during class hours
- Open book, open notes
  - But suggest you make a quick reference guide of 2 pages

Part 1: The essential simplicity of the Internet via abstraction

Complexity conquered by abstractions

Abstractions are idealizations we invent to deal with the perversity and complexity of reality. We use them all the time to deal with the complexity of the world around us. Examples include

- ***Models in Physics:*** We abstract planets as point masses interacting via gravitation to predict orbits ignoring different shapes tec)
- ***Music:*** We abstract the complexity of music (tone, timber) using notes
- ***Math:*** We abstract matrices and numbers as fields ignoring their differences.
- ***Computer Science:*** We would like to hide the messy details of computer and networks from programmers to make them more ***productive*** by hiding implementation details behind an interface

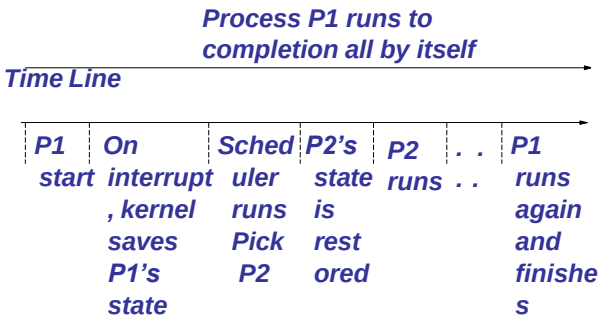
Just as in OSes which abstract a computer

Create clean abstractions so that application programmers on say Windows OS can write large programs quickly and correctly.
Examples include

- ***Processes: Abstraction of Uninterrupted Computation:*** Dealing with idiosyncrasies of interrupts would be intolerable.
- ***Virtual Memory: Abstraction of Infinite Memory:*** Dealing with the details of moving data from disk to memory would be a pain
- ***Simple I/O: Abstracting devices*** Dealing with the idiosyncrasies of various devices (USBs, cameras, microphones) would be hard.

  simpler illusion of reading and writing devices from memory

Abstractions require underlying mechanisms



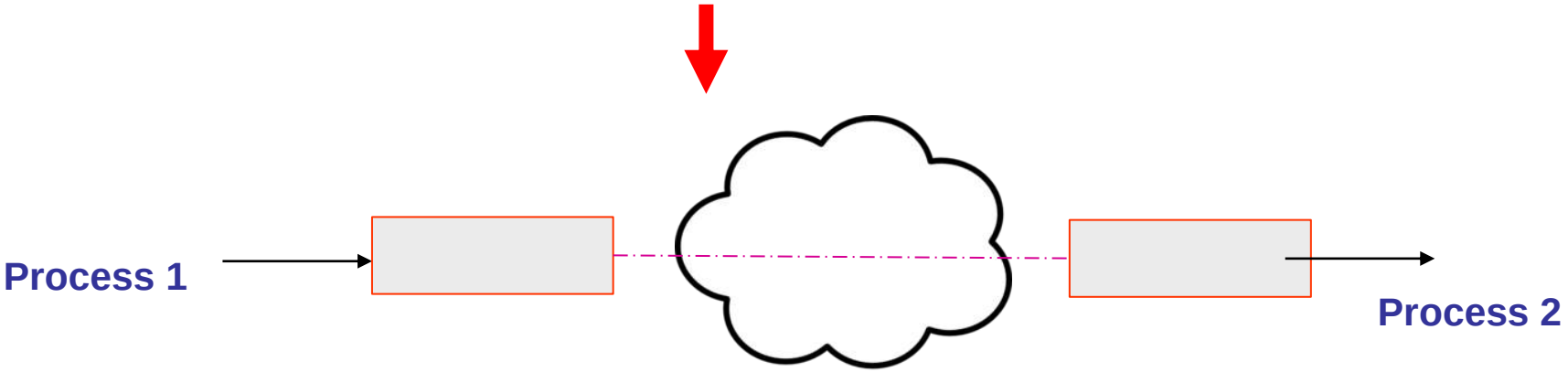
ILLUSION

REALITY

- Underlying mechanisms: Context switch, scheduling, protection

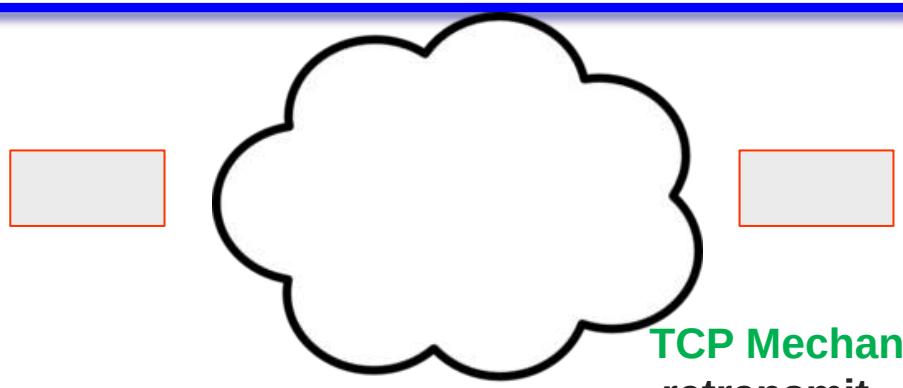
Essential Simplicity: TCP

FROM: Single Machine Communication (reliable, in-order)

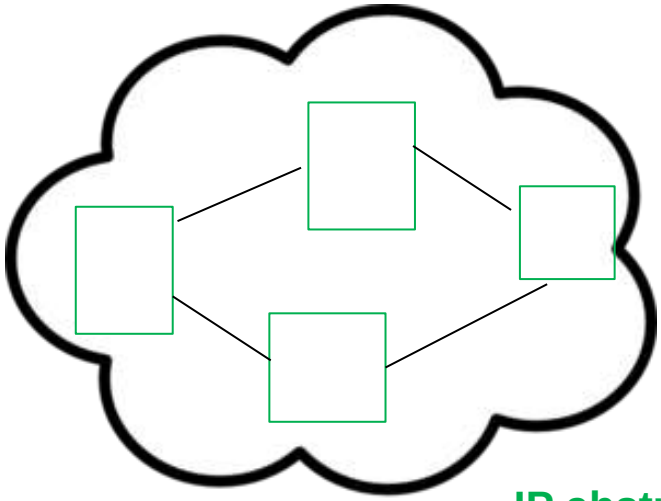


TO: Reliable In-order across network (TCP Abstraction)

Essential Simplicity: IP



TCP Mechanism: Number segments, retransmit



Source Host

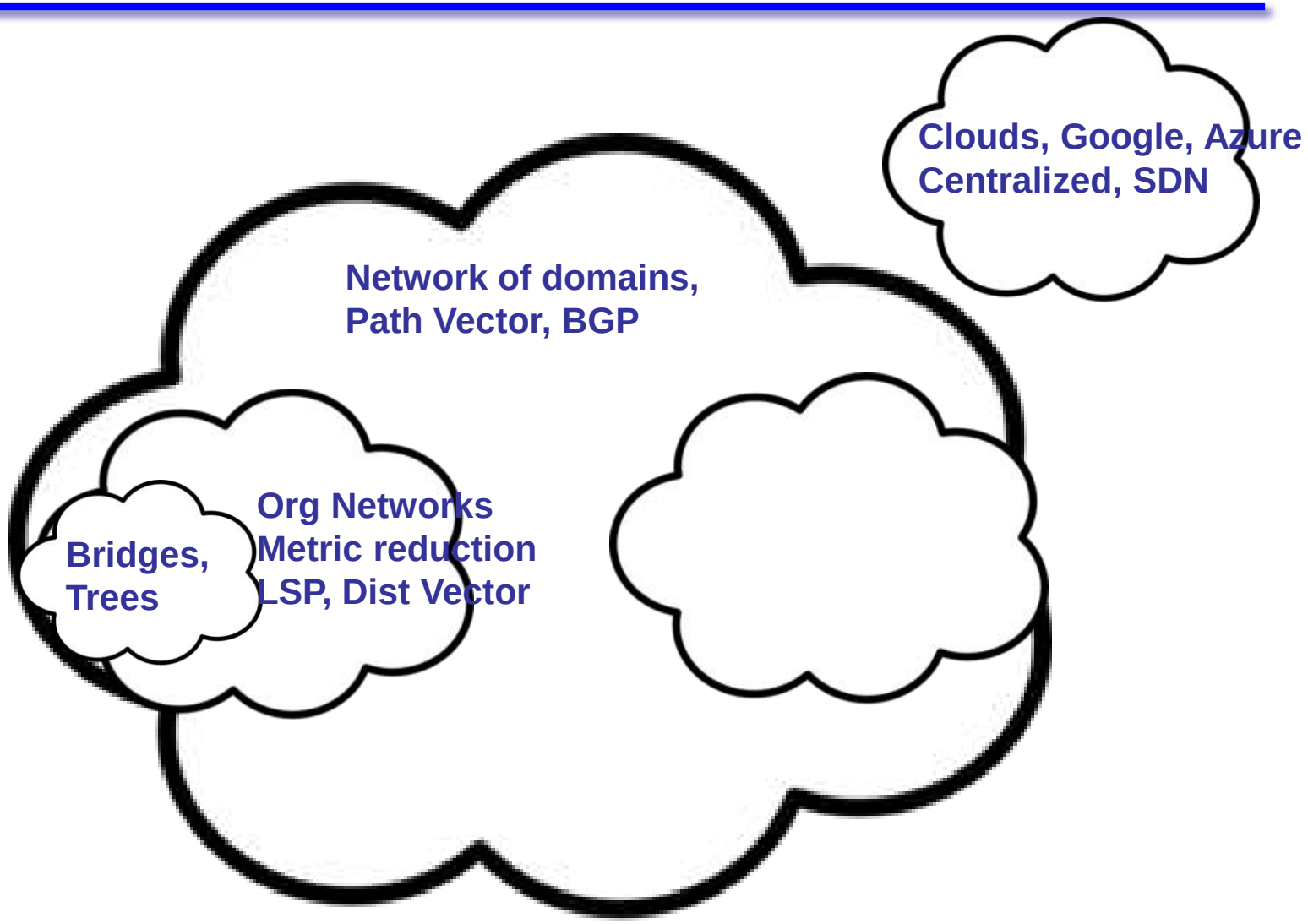
Destination Host

IP abstraction: forward packets Along path to destination

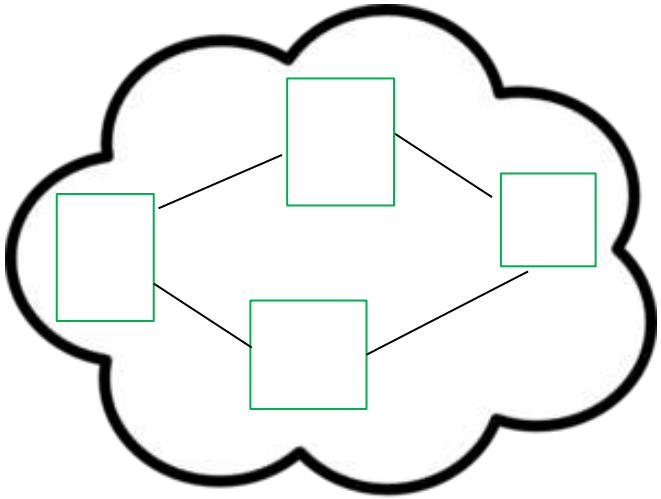
Essential Simplicity IP (Scott)

- Compute Routes: avoid loops, make progress
- Mechanisms: distance vector, LSP (metric gets smaller), path vector, spanning tree, SDN (centralized)
- Hierarchy for scaling: Spanning Tree and bridges at lowest, then Distance Vector/LSP to create organizational networks, then BGP to interconnect organizations

Scaling IP by hierarchies



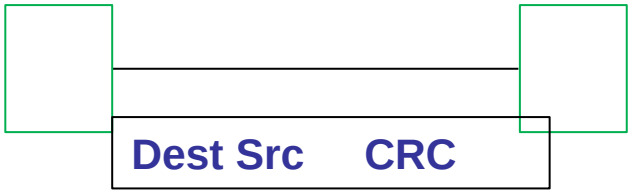
Essential Simplicity: Data Link



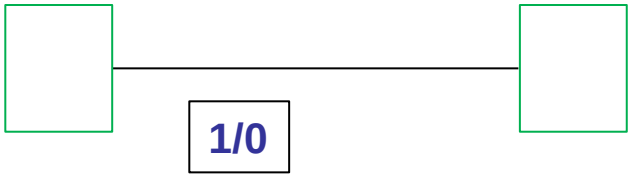
IP mechanism: compute routes, forward data, route packets to next hop

Data Link abstraction: send a group of bits (frame) between two machines on a single link

Essential Simplicity: Physical

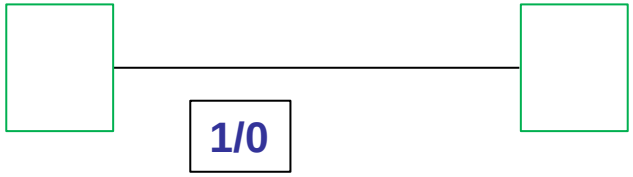


Data Link mechanism: divide bit stream into frames, do error detection (checksums) and media access if there are many senders

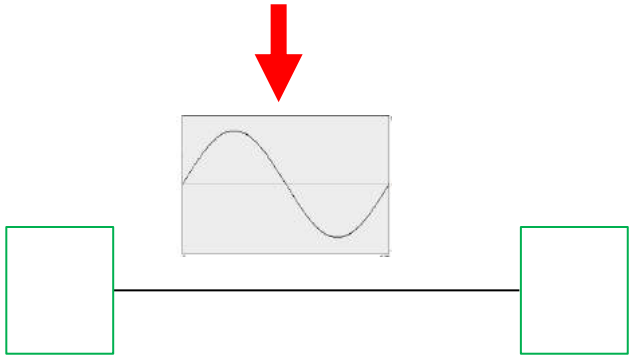


Phy Abstraction: send a group of bits (frame) between two machines on a single link

Essential Simplicity: Physical

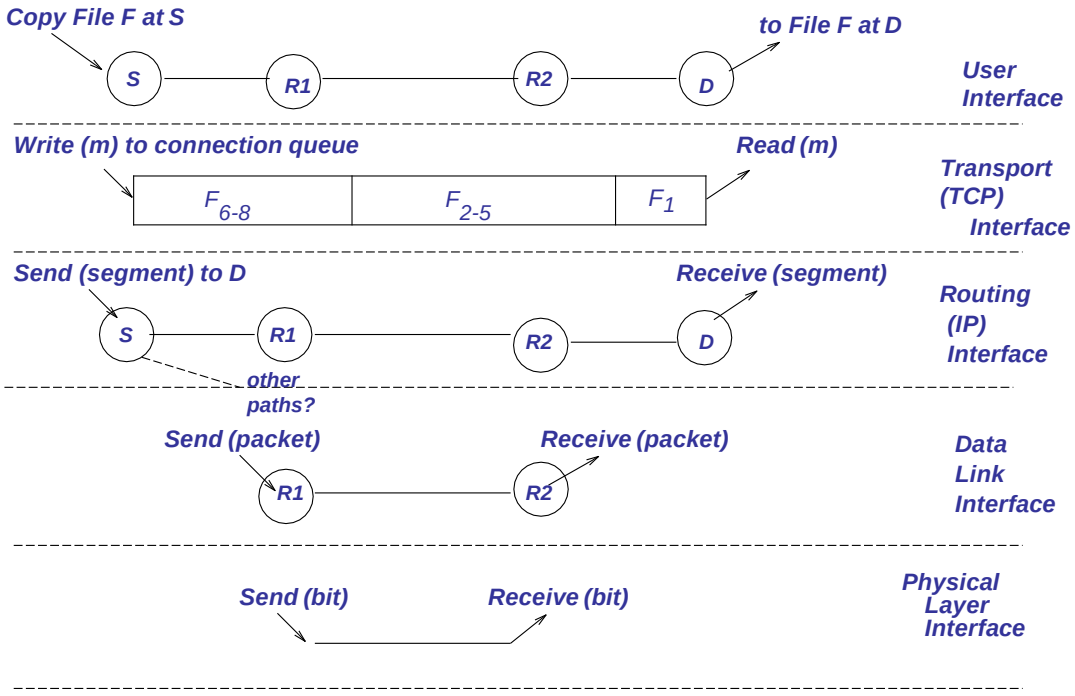


Phy mechanism: learn how to sample energy (clock recovery), convert logical bits to energy

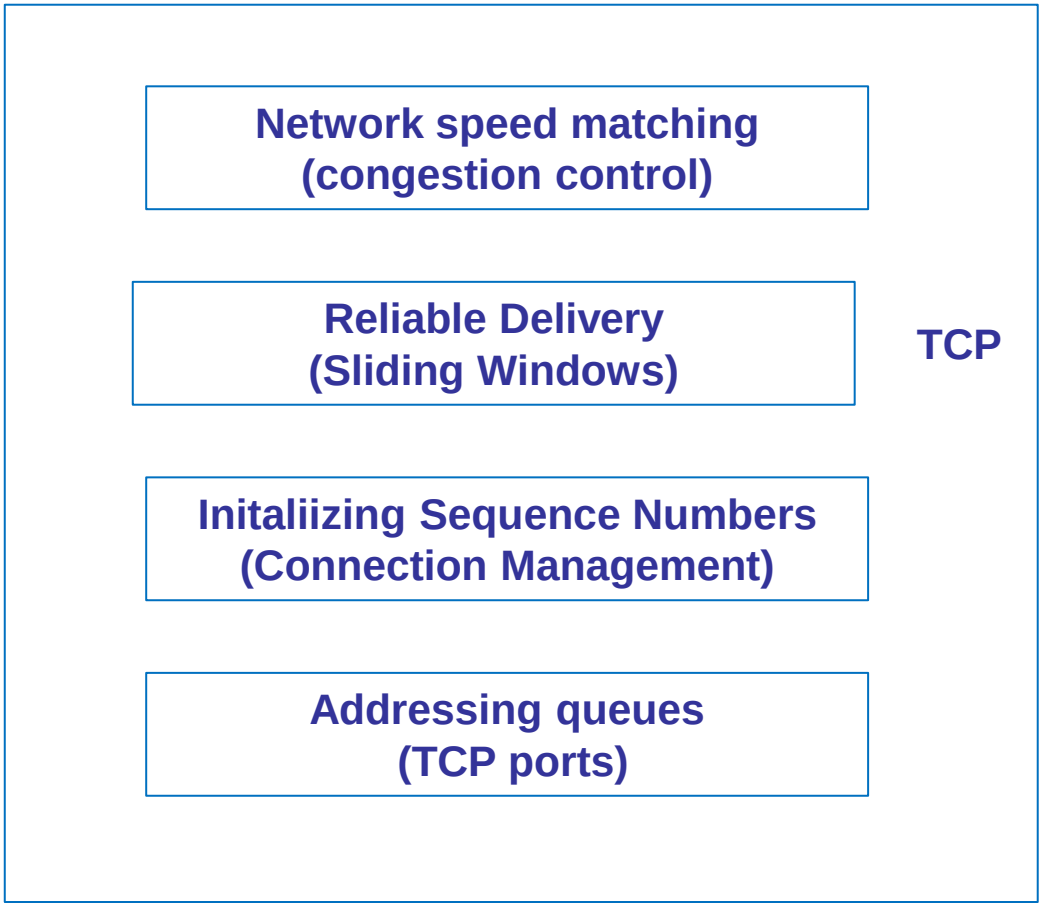


Phy Mechanism: send bits by modulating physical energy waves that travel distance (channel abstraction)

All together now, behold our abstractions



Going deeper: sublayers of TCP



Mechanisms within each abstraction

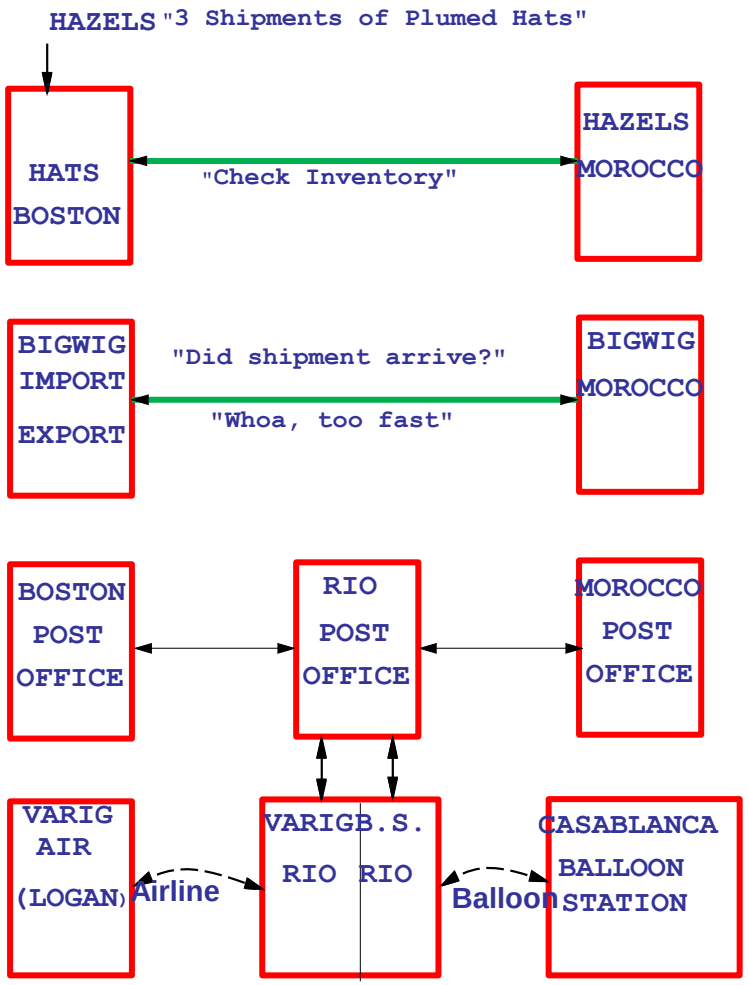
TRANSPORT LAYER:
MULTIHOP PACKET PIPE
- Same as Data Link
- Congestion Control

NETWORK LAYER:
PACKET SWITCH
- Addressing
- Routing
- Forwarding

DATA LINK LAYER:
1-HOP FRAME PIPE
- Error Detection
- Framing
- Error Recovery
- Flow Control
- Taking Turns
- Extending Data Links

PHYSICAL LAYER:
BIT PIPE
- Clock Recovery
- Multiplexing
- Media

SIMILAR ABSTRACTIONS USED BY SOCIETY



Similar Problems at all layers

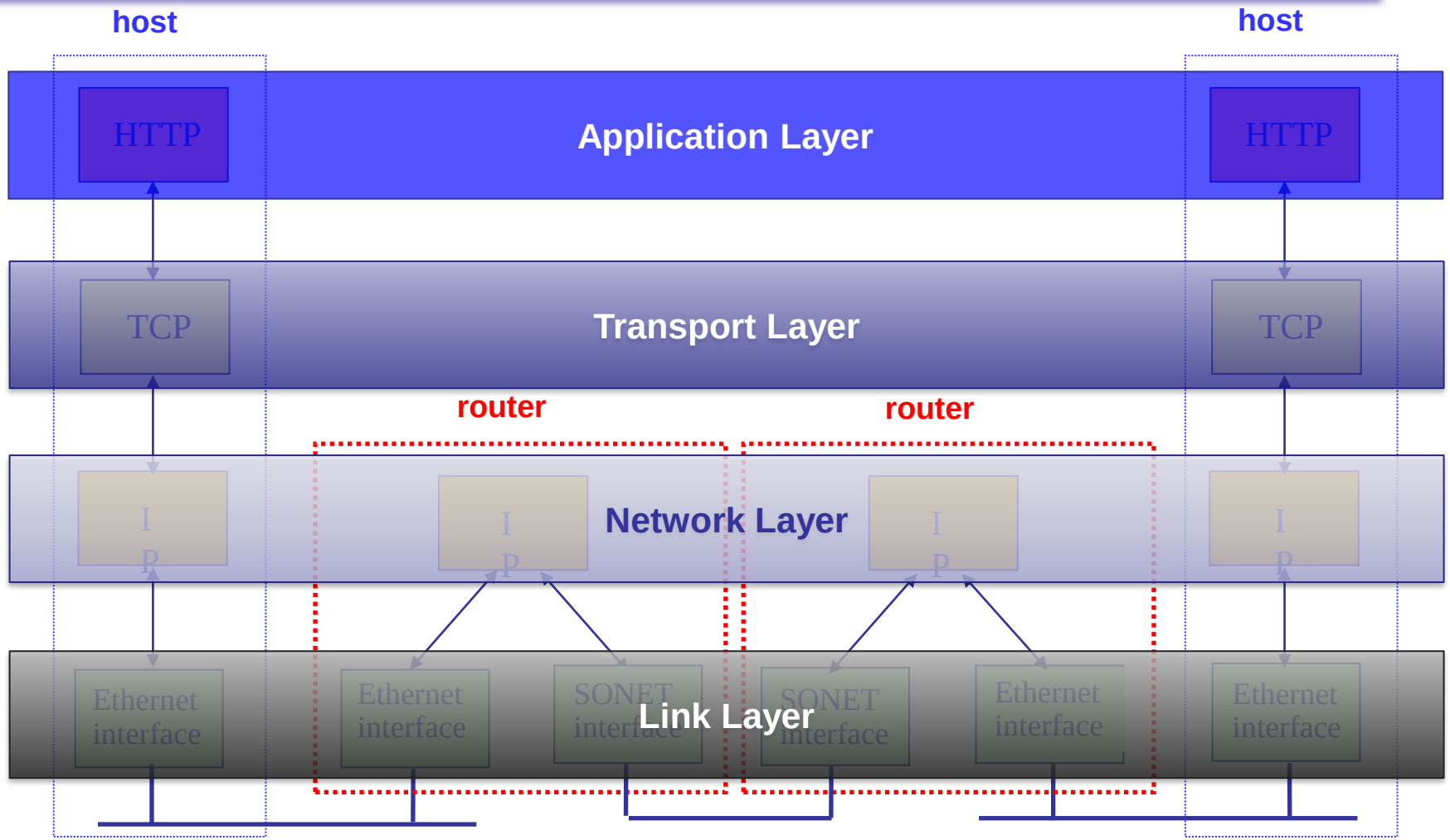
- Naming: time slots (physical), MAC addresses (DL), IP addresses (IP), IP address + ports (TCP)
- Synchronization: clock recovery (physical), packet sending (Ethernet), route estimates (IP), sequence numbers (TCP): keep state variables in synch
- Error control: spacing levels (Physical), CRCs (Data Link), LSP jumps (routing), retransmission (TCP)

True for all systems (databases, OS) but one new one

- Interconnection strategy: repeater (Physical), bridge (Data Link), router (IP), gateway (TCP and higher)

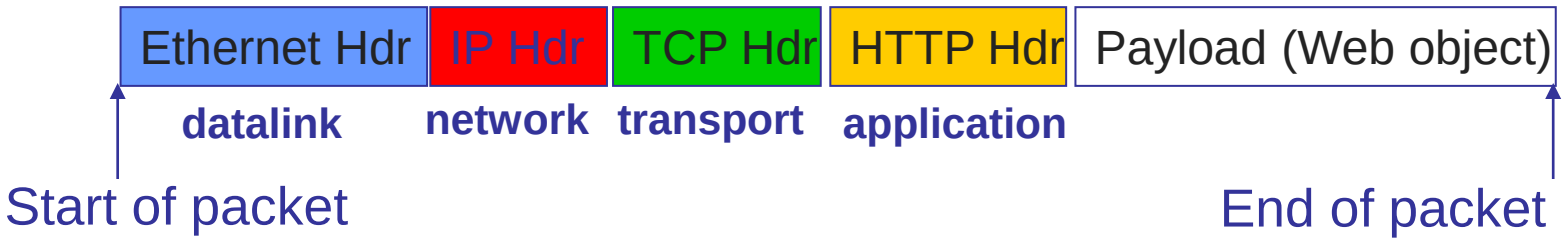
Part 2: The essential details of each layer

The Protocol Stack



Encapsulation via Headers

- Typical Web packet



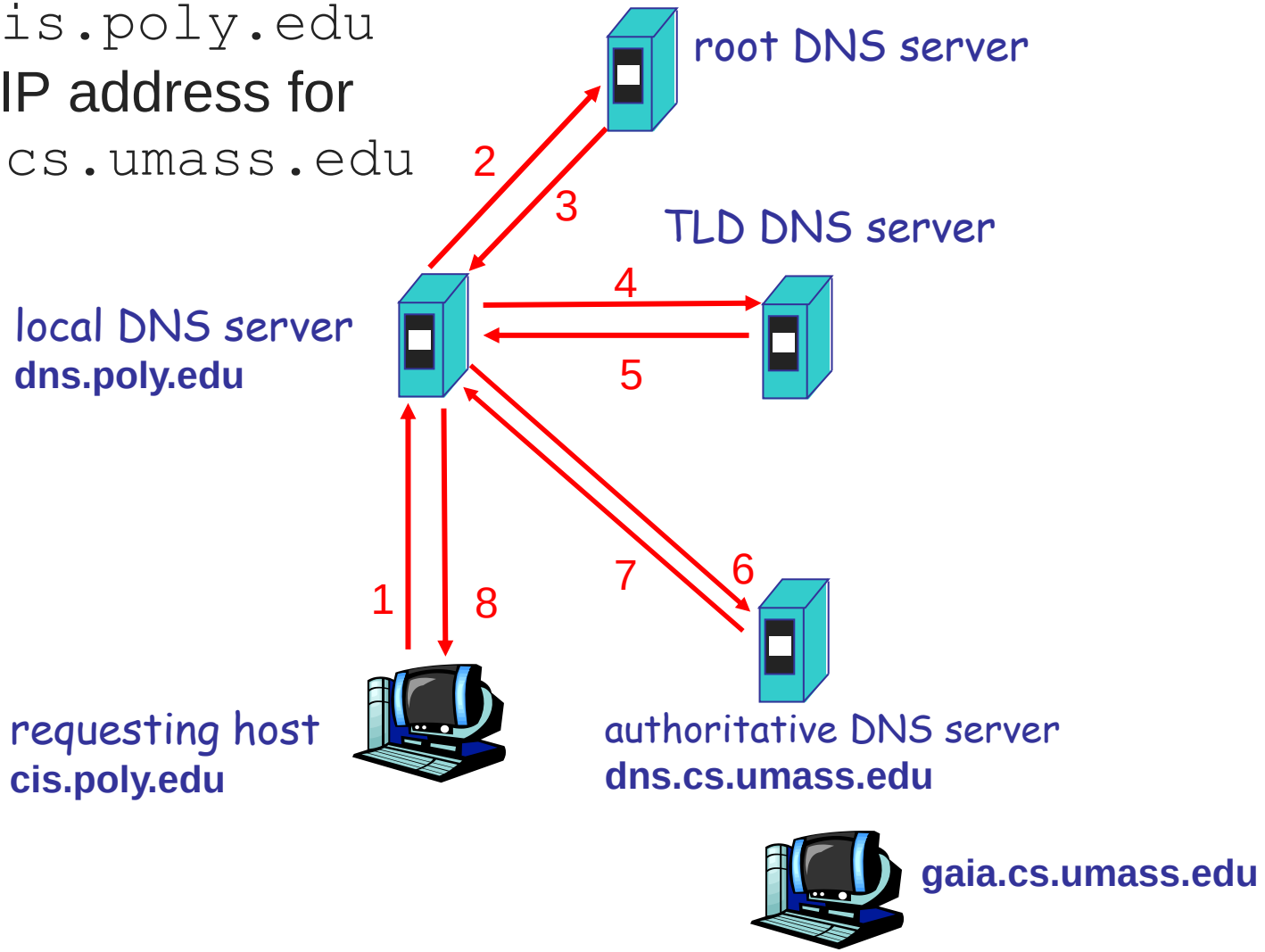
- Notice that layers add overhead
  - Space (headers), effective bandwidth
  - Time (processing headers, “peeling the onion”), latency

Quick Review of Layers

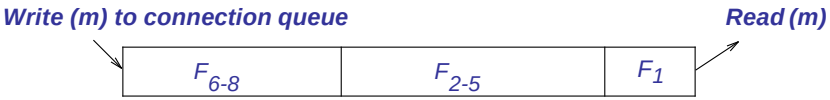
- In the course we went bottom up from physical to transport
- Lets now review top down just as we started in first lecture
- From DNS down to the physical layer, but by now we've learned a lot

It all starts with DNS

Host at `cis.poly.edu` wants IP address for `gaia.cs.umass.edu`

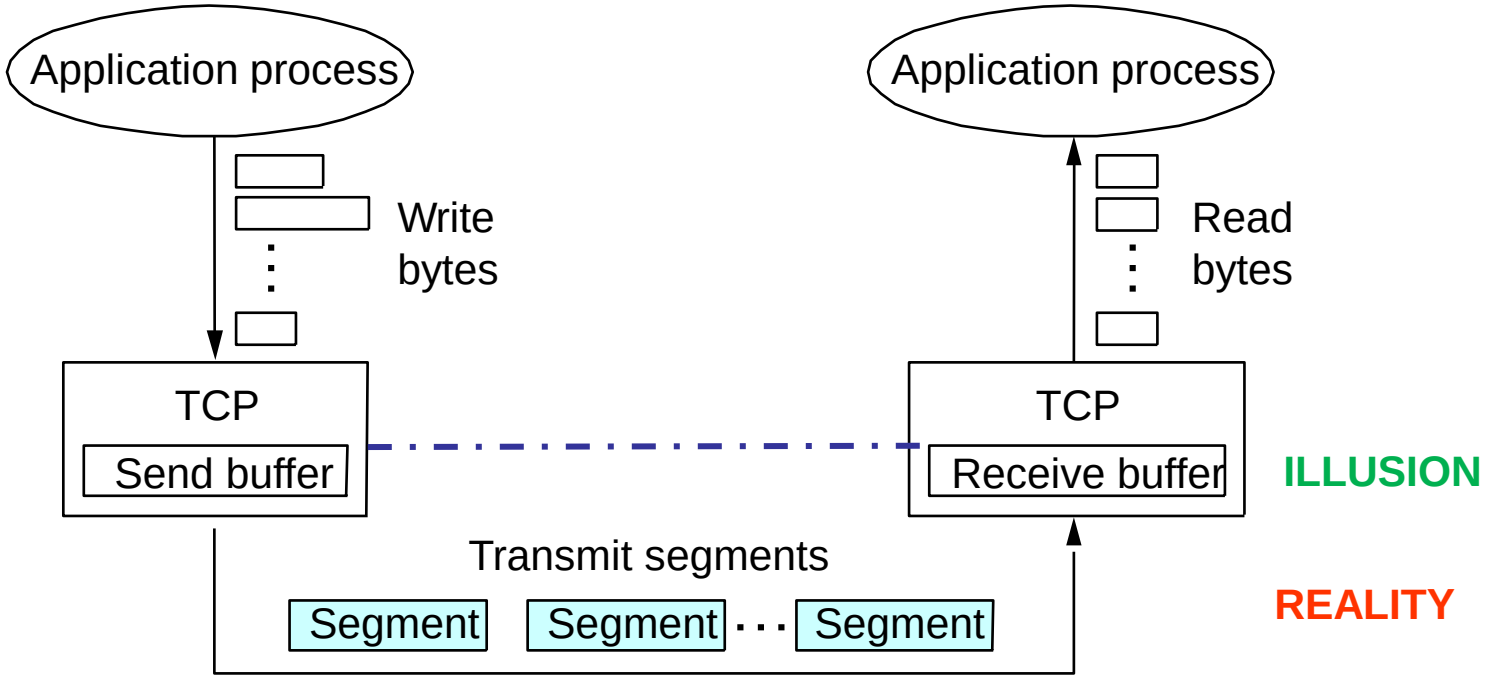


Then Transport Layer



(TCP)

TCP Abstraction



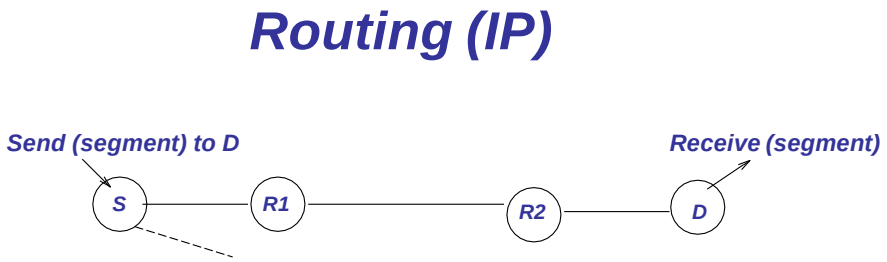
TCP Mechanisms

- 3-way handshake using ISN to defend against delayed duplicates
- Sliding Window for reliability
- Flow and congestion control by dynamically adjusting window

Transport Protocol Ideas

IDEAS	TRANSPORT MECHANISMS
Just like reliable Data Links except	**Error Control, Sequencing** Multiple connections, larger sequence numbers
Fault-tolerance Delayed duplicates Crashes 2 generals	**Connection Management** 3-way handshakes, Timer-based Management Graceful disconnection
Adaptive Protocols Passing info in headers. Deadlock vs. Livelock	**Congestion Control** Matching to network speed Control vs. Avoidance Congestion Collapse Bit or timeout based feedback.

Routing Review



Mechanism: Link-state Routing

- Two phases
  - Reliable flooding
    - » Tell all routers what you know about your local topology
  - Path calculation (Dijkstra's algorithm)
    - » Each router computes best path over complete network
- Motivation
  - Global information allows optimal route computation
  - Straightforward to implement and verify

Mechanism: Distance Vector Algorithm

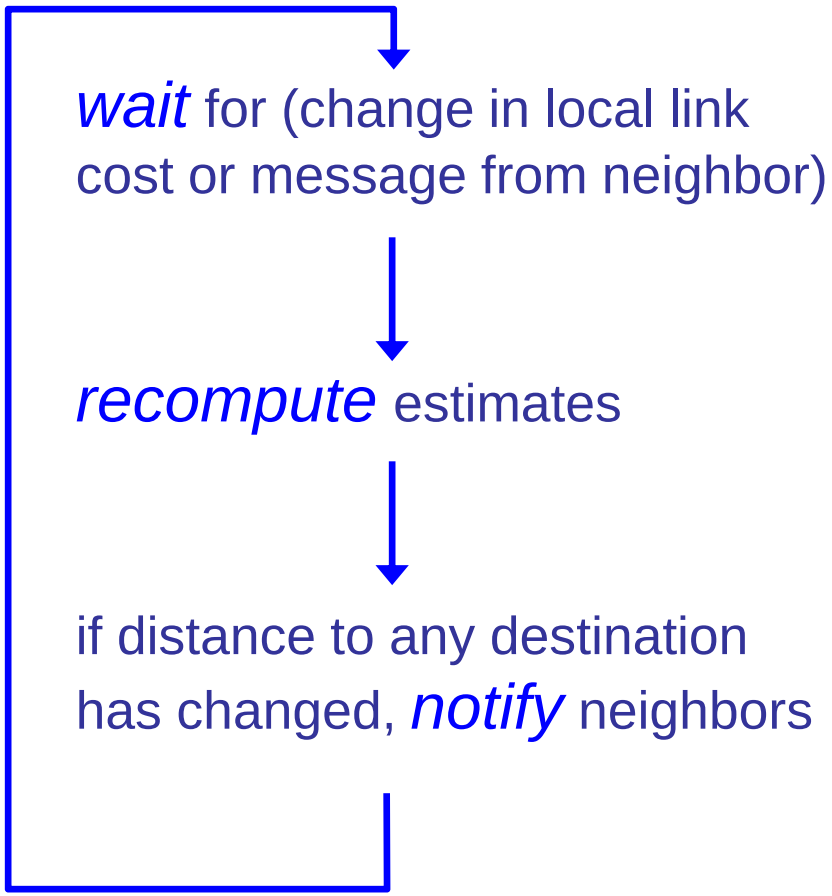
Iterative, asynchronous: each local iteration caused by:

- Local link cost change
- Distance vector update message from neighbor

Distributed:

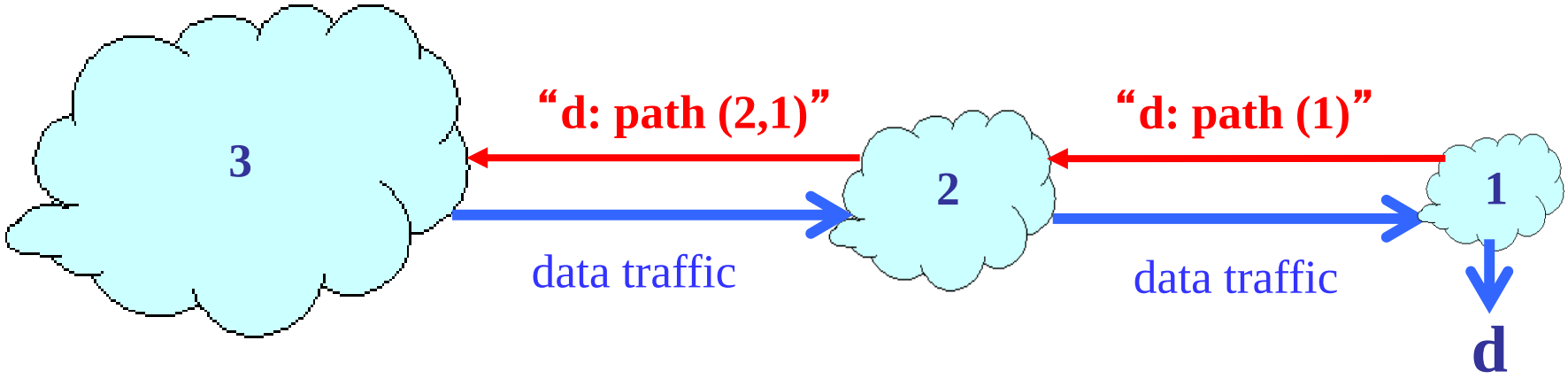
- Each node notifies neighbors when its DV changes
- Neighbors then notify their neighbors if necessary

Each node:



Mechanism 3: Path-vector Routing

- Extension of distance-vector routing
  - Support flexible routing policies
  - Avoid count-to-infinity problem
- Key idea: advertise the entire path
  - Distance vector: send *distance metric* per destination
  - Path vector: send the *entire path* for each destination



Routing Ideas

TIDEAS	ROUTING MECHANISMS
Flexible: Hierarchies Scalability vs Efficiency	**Addressing** Partition address to reflect hierarchy. Flexible Partition (IP masks) Unlimited levels (OSI)
Autoconfiguration Initialization Failures	**Neighbor Greeting** Send hellos to discover neighbor and detect failure Endnode Routing: must be simple Send to Router + Redirects
Post Office vs. Phone Model Failure Methods Crash Recovery	**Connectionless Routing** Distance Vector: Similar to Spanning Tree except for Age Fields: Count to infinity. Solved by limit on cost Link State: Intelligent flooding
2 generals Handle Diversity!	**Forwarding** Stop Looping packets Hop Counts,Time to Live

Data Link Review

Data Link

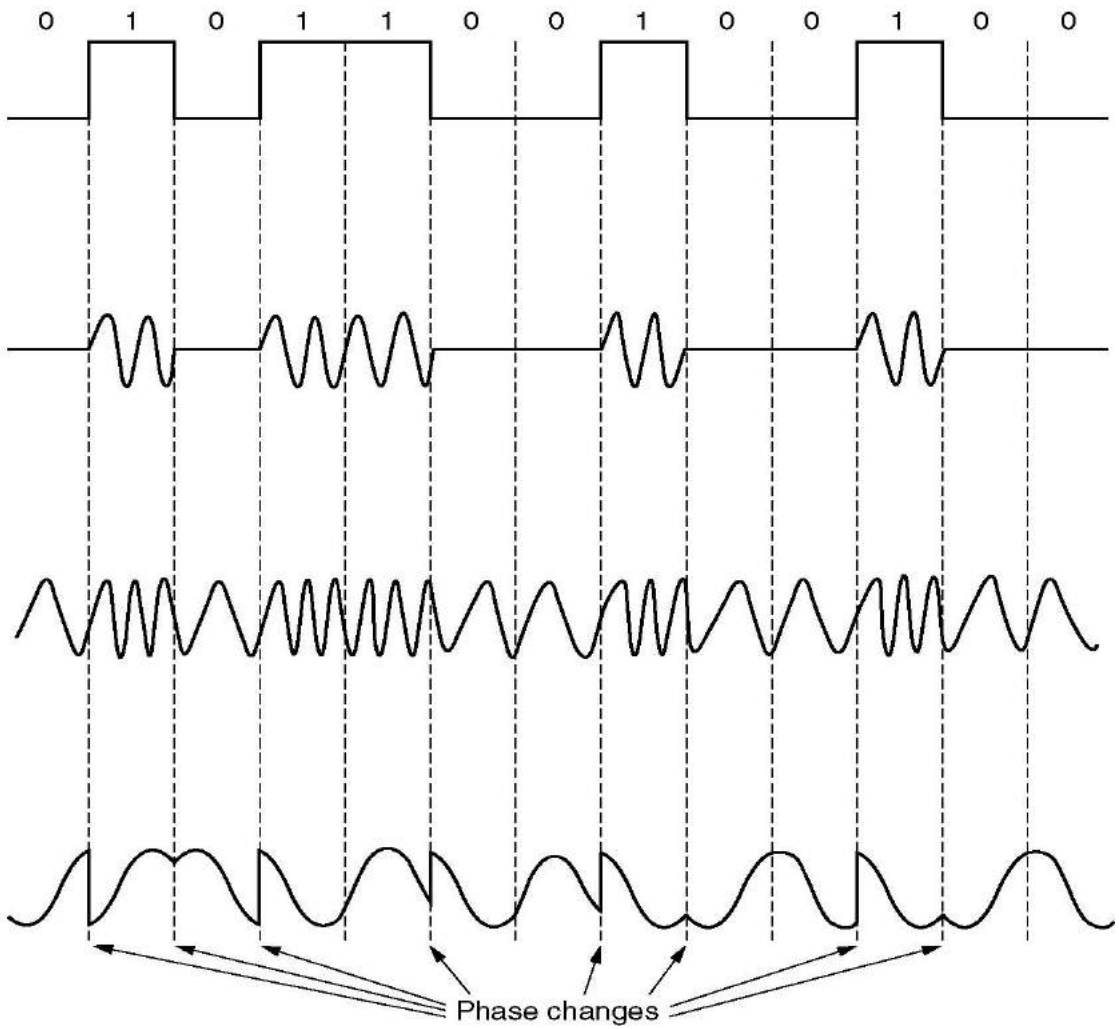
Data Link Ideas

IDEAS	DATA LINK MECHANISMS
Correctness: Invariants Reduction Performance Initialization Impossibility Results	Point-to-point Error Detection (CRCs and Parity) Framing Error Recovery: Alternating Bit Sliding Window Selective Reject Flow Control
Performance: No Error Recovery Randomization	LANs: Taking Turns Ethernet Collision Sense, Detection & Backoff 802.5
Layer n Relays Protocol Design! In Stages Failures Autoconfiguration Self-stabilization	Extending Data Link Protocols Basic Bridging Loop Detection Attempt Spanning Tree

Physical Layer Review

Physical Layer

Forms of Digital Modulation



Clock Recovery

- Using a training sequence to get receiver lined up
  - Send a few, known initial training bits
  - Adds inefficiency: only *m* data bits out of *n* transmitted
- Need to combat clock drift as signal proceeds
  - Use transitions to keep clocks synched up
- Question is, how often do we do this?
  - Quick and dirty every time: asynchronous coding
  - Spend a lot of effort to get it right, but amortize over lots of data: synchronous coding

Part 3: Beyond the basics

Fast Routers and Servers

- Servers: VIA (bypass kernel) from app to adaptor
- DPDK: libraries to offload TCP processing to user space with polling to reduce interrupt overhead
- Router Lookups: pipelined multibit tries
- Router Switching: Crossbar switches using VoQs
- Router Scheduling: Deficit Round Robin, Token Bucket rate control for each queuu

More complex services out there

- Load balancers: route a request to a VC implementing a service
- Firewalls: drop packets that match certain fields
- IDS devices: like firewalls, but also inspect content
- NAT boxes: Allow an organization to manage with a single IP address by mapping (using TCP port numbers) to private IP addresses
- WAN Acceleration: compression, TCP termination, deduplication
- Edge Computing: Moving services to user edge to get better performance

New Protocols out there

- Akamai: Hack the DNS to get shortest path to a replicated service
- QUIC: Improvement of TCP to get reduce roundtrips
- DCTCP: Improvement of TCP to ramp up and down more smoothly in the face of congestion
- BBR: Google's new congestion control algorithm
- Espresso: Hack BGP: use centralized system & extra info to get better routes from Google to other ISPs
- B4: Within Google, routes between data centers are done centrally, longer path routes used to improve throughput and not waste bandwidth!