15-440 Distributed Systems

Lecture 2 & 3 – 15-441 in 2 Days

WARNING!!!!

- These slides (and any other slides posted for class) are *NOT* meant to be a complete guide or class notes
- Please take notes in class
  - There is often additional material presented
  - Slides are often hard to understand after the fact
- I will make sure to post at least a draft of the slides in advance of class
  - Gives you a start on notes ☺

Distributed Systems vs. Networks

- Low level (c/go)
- Run forever
- Support others
- Adversarial environment
- Distributed & concurrent
- Resources matter

- And have it implemented/run by vast numbers of different people with different goals/skills

Keep an eye out for…

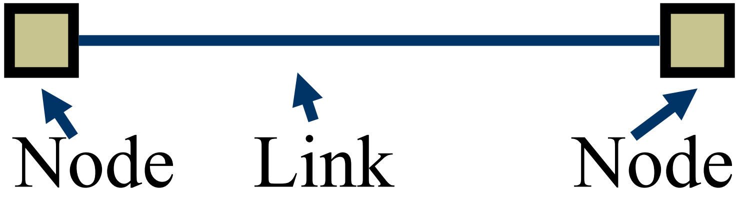
- Modularity, Layering, and Decomposition:
  - Techniques for dividing the work of building systems
  - Hiding the complexity of components from each other
  - Hiding implementation details to deal with heterogeneity
- Naming/lookup/routing
- Resource sharing and isolation

- Models and assumptions about the environment and components
- Understanding and estimating performance

Today’s & Tuesday’s Lecture

- Network links and LANs
- Layering and protocols
- Internet design
- Transport protocols
- Application design

Basic Building Block: Links

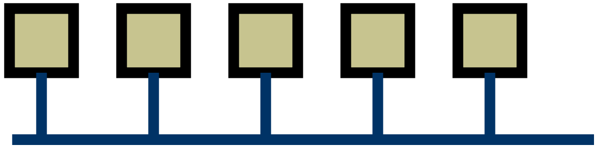


- Electrical questions
  - Voltage, frequency, …
  - Wired or wireless?

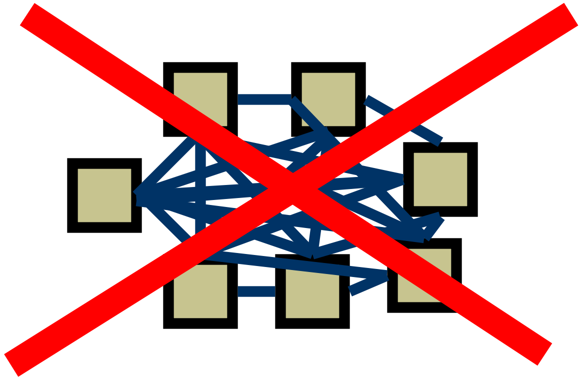
- Link-layer issues: How to send data?
  - When to talk – can either side talk at once?
  - What to say – low-level format?

Basic Building Block: Links

- … But what if we want more hosts?



One wire

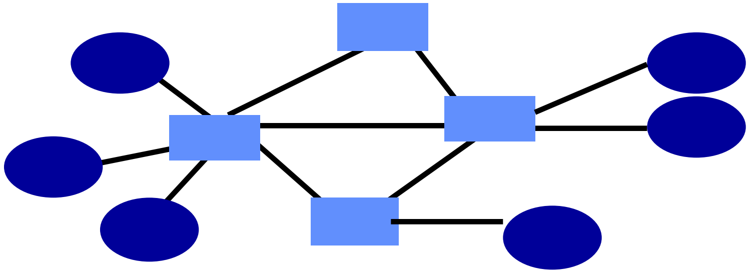


Wires for everybody!

- Scalability?!

Multiplexing

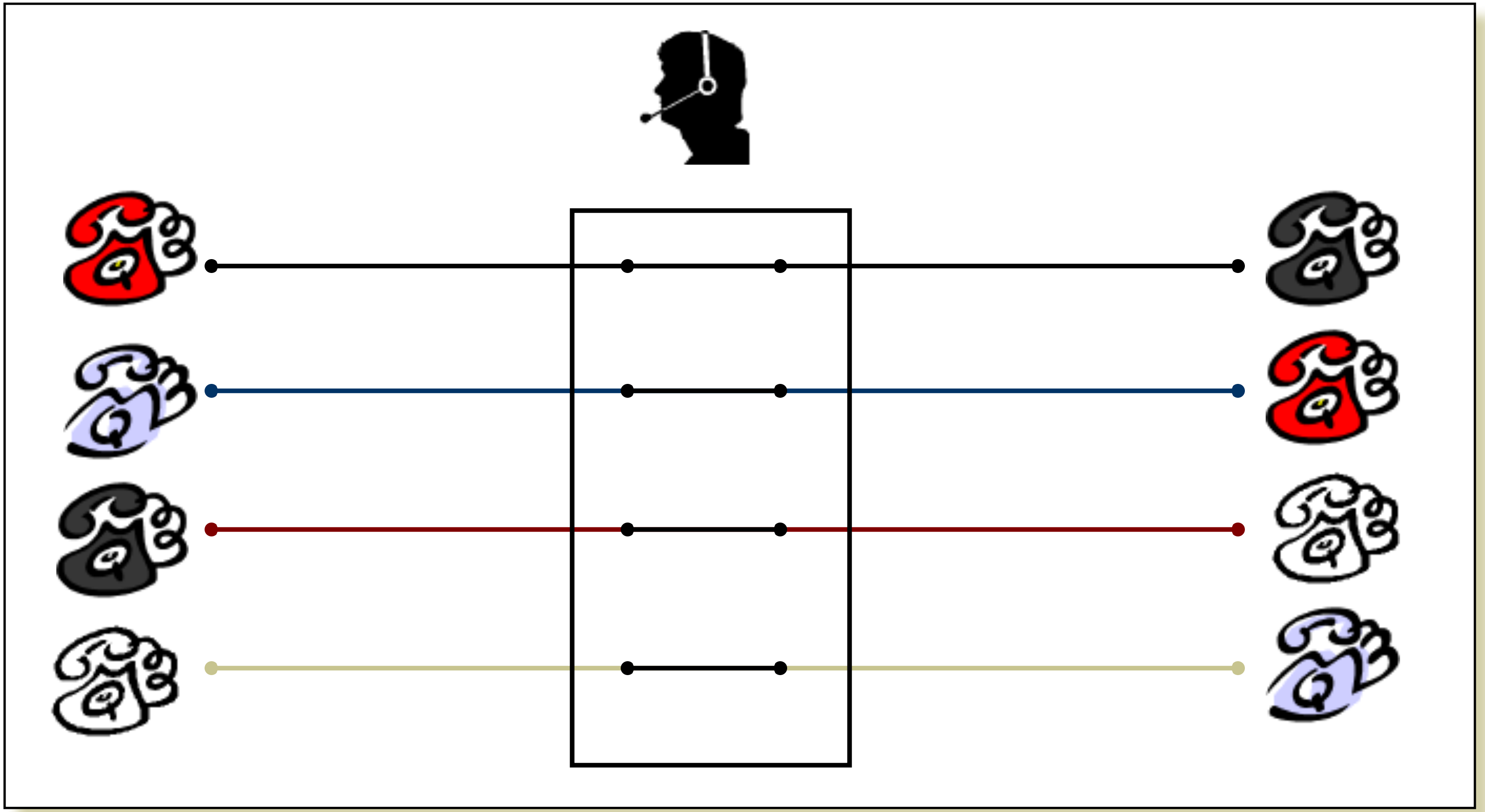
- Need to share network resources



- How? Switched network
  - Party "A" gets resources sometimes
  - Party "B" gets them sometimes
- Interior nodes act as "Switches"

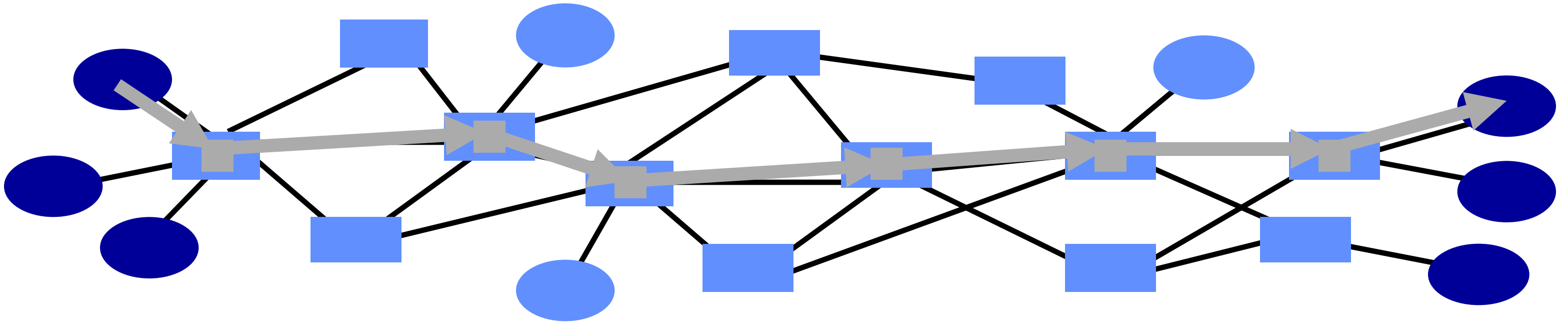
- What mechanisms to share resources?

In the Old Days…Circuit Switching

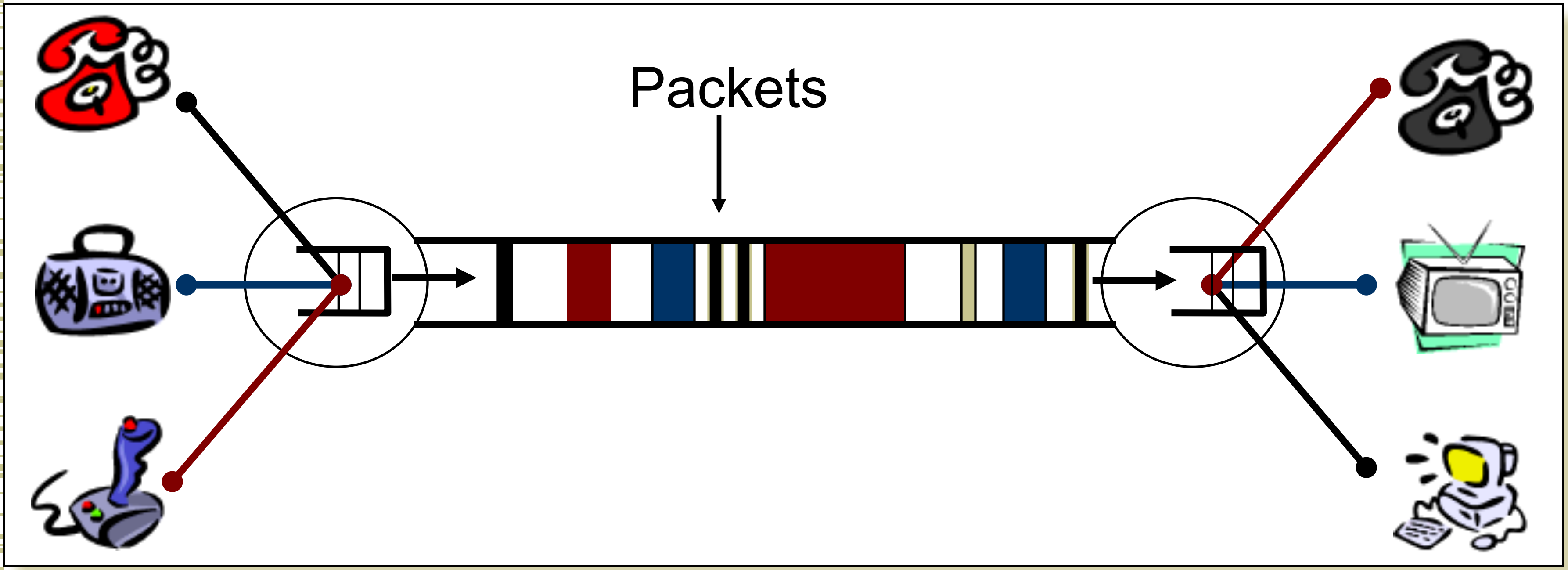


Packet Switching

- Source sends information as self-contained packets that have an address.
  - Source may have to break up single message in multiple
- Each packet travels independently to the destination host.
  - Switches use the address in the packet to determine how to forward the packets
  - Store and forward
- Analogy: a letter in surface mail.



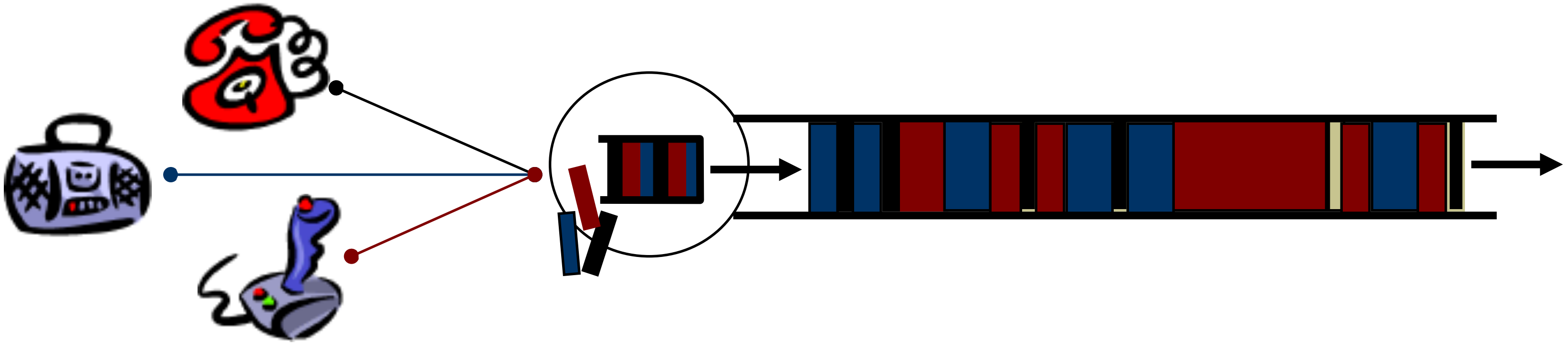
Packet Switching – Statistical Multiplexing



- Switches arbitrate between inputs
- Can send from *any* input that's ready
  - Links never idle when traffic to send
  - (Efficiency!)

What if Network is Overloaded?

Problem: Network Overload



Solution: Buffering and Congestion Control

- Short bursts: buffer
- What if buffer overflows?
  - Packets dropped
  - Sender adjusts rate until load = resources → “congestion control”

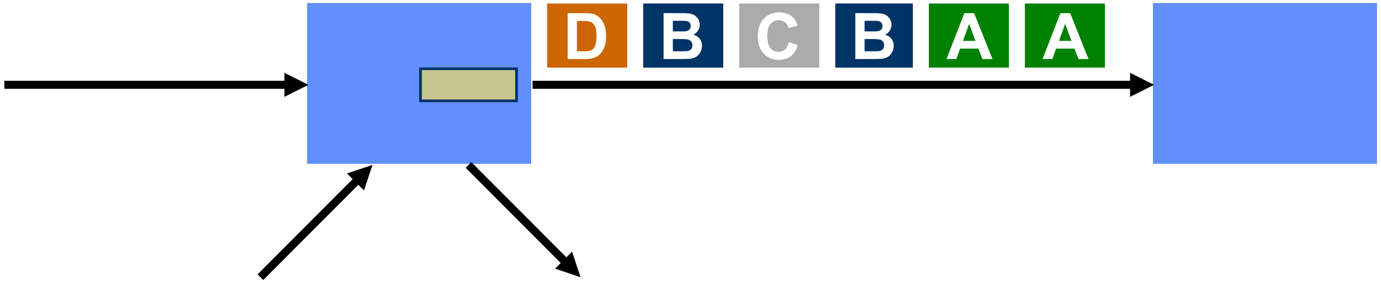
Model of a communication channel

- Latency - how long does it take for the first bit to reach destination
- Capacity - how many bits/sec can we push through? (often termed “bandwidth”)
- Jitter - how much variation in latency?
- Loss / Reliability - can the channel drop packets?
- Reordering

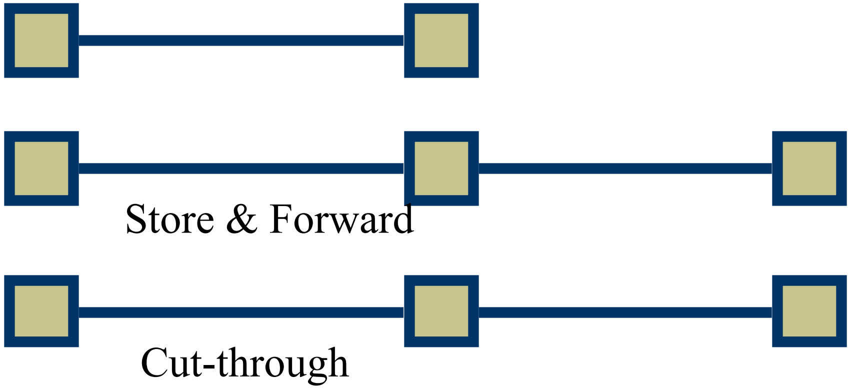
Packet Delay

- Sum of a number of different delay components:

- Propagation delay on each link.
  - Proportional to the length of the link
- Transmission delay on each link.
  - Proportional to the packet size and 1/link speed
- Processing delay on each router.
  - Depends on the speed of the router
- Queuing delay on each router.
  - Depends on the traffic load and queue size



Packet Delay



Prop + xmit

2*(Prop + xmit)

2*prop + xmit

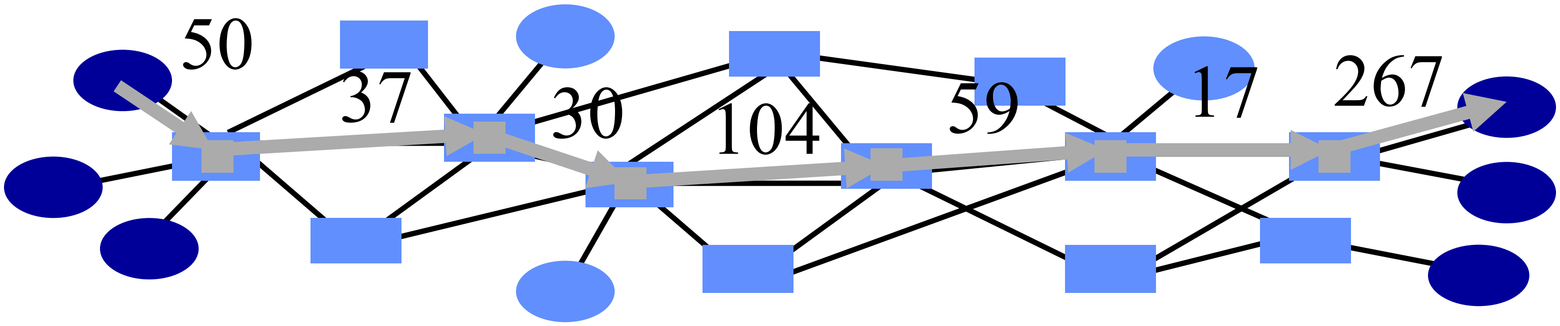
When does cut-through matter?

Next: Routers have finite speed (processing delay)

Routers may buffer packets (queueing delay)

Sustained Throughput

- When streaming packets, the network works like a pipeline.
  - All links forward different packets in parallel
- Throughput is determined by the slowest stage.
  - Called the bottleneck link
- Does not really matter why the link is slow.
  - Low link bandwidth
  - Many users sharing the link bandwidth



Some simple calculations

- Cross country latency
  - Distance/speed = 5 * 10^6m / 2x10^8m/s = 25 * 10^-3 s = 25ms
  - 50ms RTT
- Link speed (capacity) 100Mbps
- Packet size = 1250 bytes = 10 kbits
  - Packet size on networks usually = 1500bytes across wide area or 9000bytes in local area
- 1 packet takes
  - 10k/100M = .1 ms to transmit
  - 25ms to reach there
  - ACKs are small → so 0ms to transmit
  - 25ms to get back
- Effective bandwidth = 10kbits/50.1ms = 200kbits/sec ☹

Think about this…

- What if we sent two packets before waiting for an ACK
  - What if we sent N packets?
  - How many packets do we need to send before we use up the capacity of the link?

Some Examples

- How long does it take to send a 100 Kbit file?
  - Assume a perfect world
  - And a 10 Kbit file

Throughput Latency	100 Kbit/s	1 Mbit/s	100 Mbit/s
500 μsec	0.1005	0.0105	0.0006
10 msec	0.11	0.02	0.0101
100 msec	0.2	0.11	0.1001

Some Examples

- How long does it take to send a 100 Kbit file?
  - Assume a perfect world

Throughput Latency	100 Kbit/s	1 Mbit/s	100 Mbit/s
500 µsec	1.0005	0.1005	0.0015
10 msec	1.01	0.11	0.011
100 msec	1.1	0.2	0.101

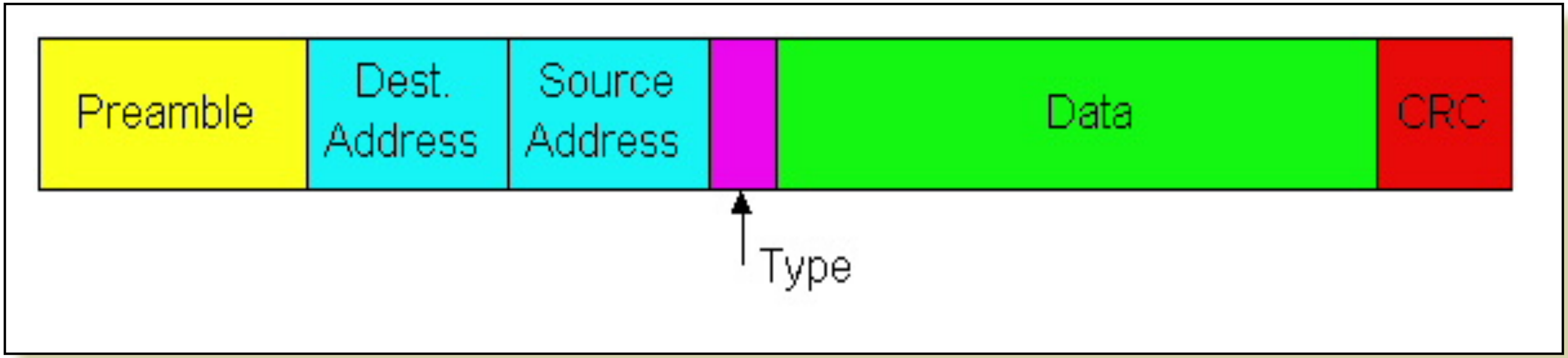
Some Examples

- How long does it take to send a 10 Kbit file?
  - Assume a perfect world

Throughput Latency	100 Kbit/s	1 Mbit/s	100 Mbit/s
500 µsec	0.1005	0.0105	0.0006
10 msec	0.11	0.02	0.0101
100 msec	0.2	0.11	0.1001

Example: Ethernet Packet

- Sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame



Ethernet Frame Structure

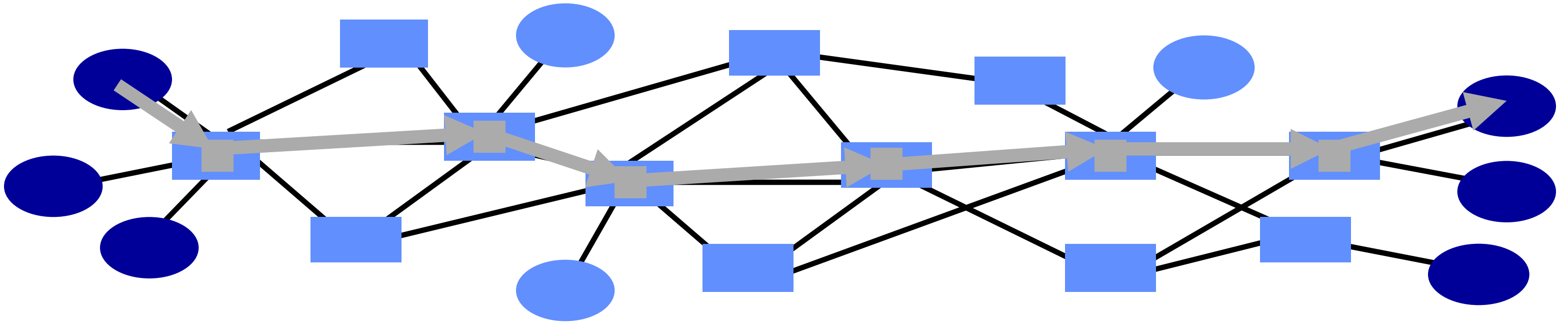
- Each protocol layer needs to provide some hooks to upper layer protocols
  - Demultiplexing: identify which upper layer protocol packet belongs to
  - E.g., port numbers allow TCP/UDP to identify target application
  - Ethernet uses Type field
- Type: 2 bytes
  - Indicates the higher layer protocol, mostly IP but others may be supported such as Novell IPX and AppleTalk

Ethernet Frame Structure (cont.)

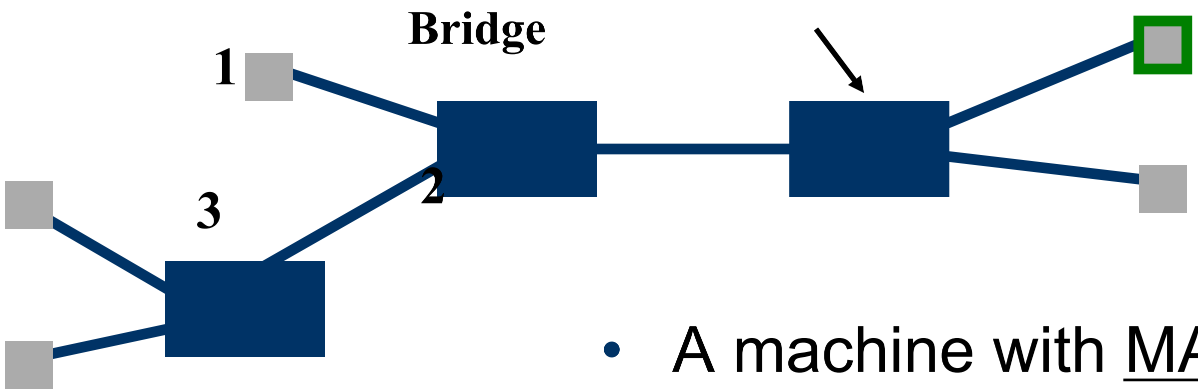
- Addresses:
  - 6 bytes
  - Each adapter is given a globally unique address at manufacturing time
    - Address space is allocated to manufacturers
      - 24 bits identify manufacturer
      - E.g., 0:0:15:* → 3com adapter
    - Frame is received by all adapters on a LAN and dropped if address does not match
  - Special addresses
    - Broadcast – FF:FF:FF:FF:FF:FF is “everybody”
    - Range of addresses allocated to multicast
      - Adapter maintains list of multicast groups node is interested in

Packet Switching

- Source sends information as self-contained packets that have an address.
  - Source may have to break up single message in multiple
- Each packet travels independently to the destination host.
  - Switches use the address in the packet to determine how to forward the packets
  - Store and forward
- Analogy: a letter in surface mail.



Frame Forwarding

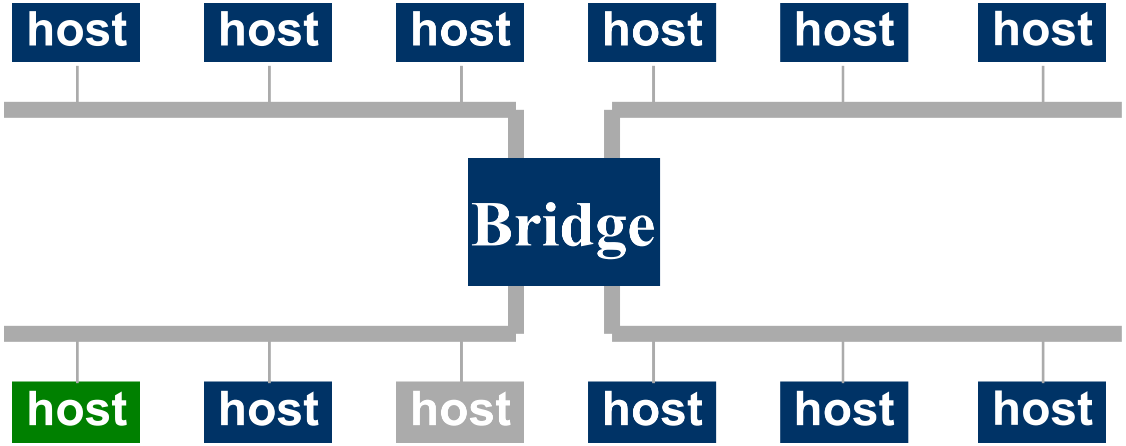


MAC Address	Port	Age
A21032C9A591	1	36
99A323C90842	2	01
8711C98900AA	2	15
301B2369011C	2	16
695519001190	3	11

- A machine with MAC Address lies in the direction of number port of the bridge
- For every packet, the bridge “looks up” the entry for the packets destination MAC address and forwards the packet on that port.
  - Other packets are broadcast – why?
- Timer is used to flush old entries

Learning Bridges

- Manually filling in bridge tables?
  - Time consuming, error-prone
- Keep track of source address of packets arriving on every link, showing what segment hosts are on
  - Fill in the forwarding table based on this information

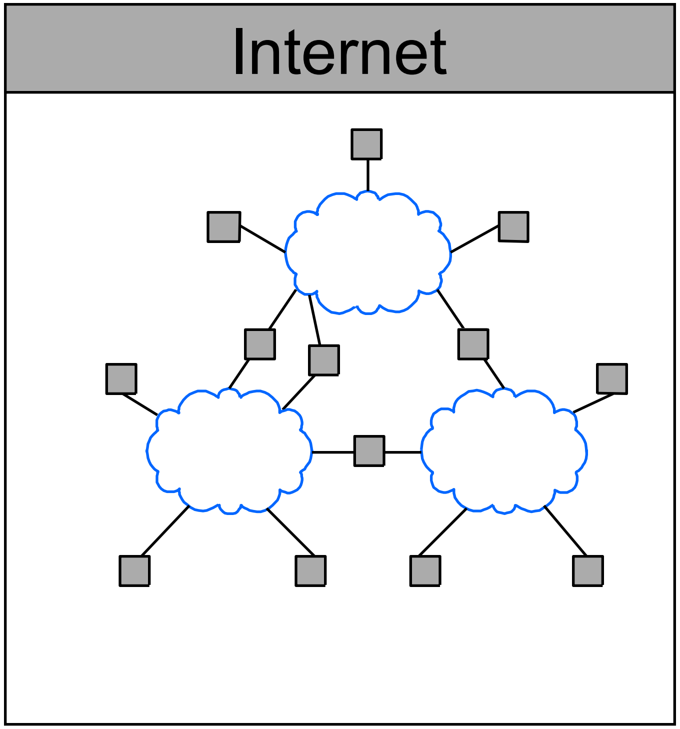


Today's & Tuesday's Lecture

- Network links and LANs
- Layering and protocols
- Internet design
- Transport protocols
- Application design

Internet

- An inter-net: a network of networks.
  - Networks are connected using routers that support communication in a hierarchical fashion
  - Often need other special devices at the boundaries for security, accounting, ..
- The Internet: the interconnected set of networks of the Internet Service Providers (ISPs)
  - About 17,000 different networks make up the Internet

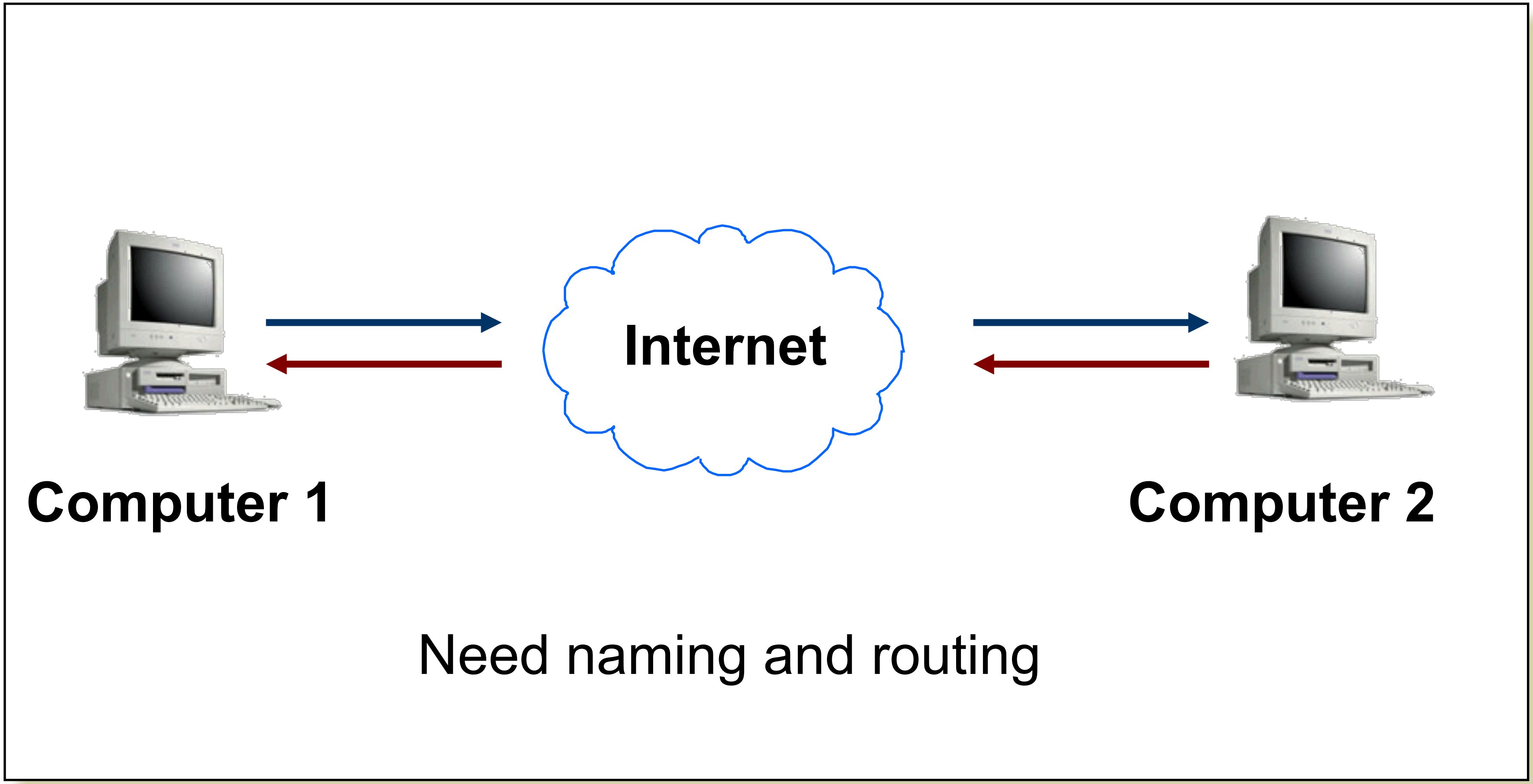


Challenges of an internet

- Heterogeneity
  - Address formats
  - Performance – bandwidth/latency
  - Packet size
  - Loss rate/pattern/handling
  - Routing
  - Diverse network technologies → satellite links, cellular links, carrier pigeons
  - In-order delivery

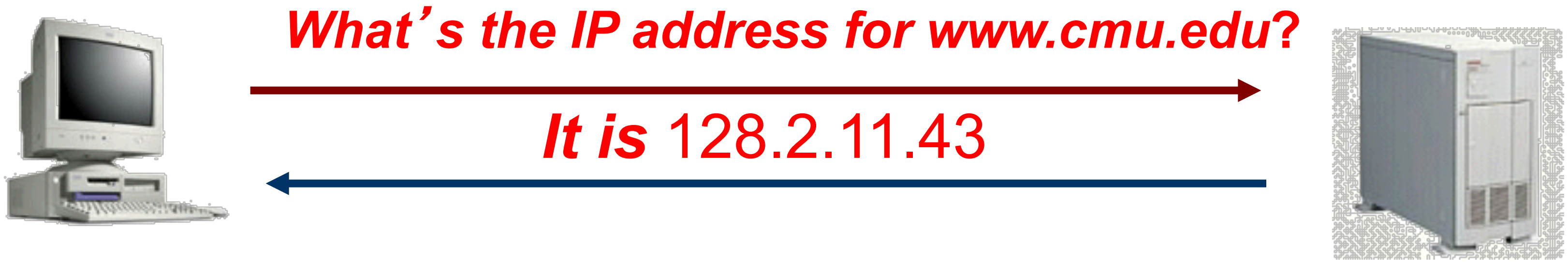
- Need a “standard” that everyone can use → IP

How To Find Nodes?



Need naming and routing

Naming



What's the IP address for www.cmu.edu?

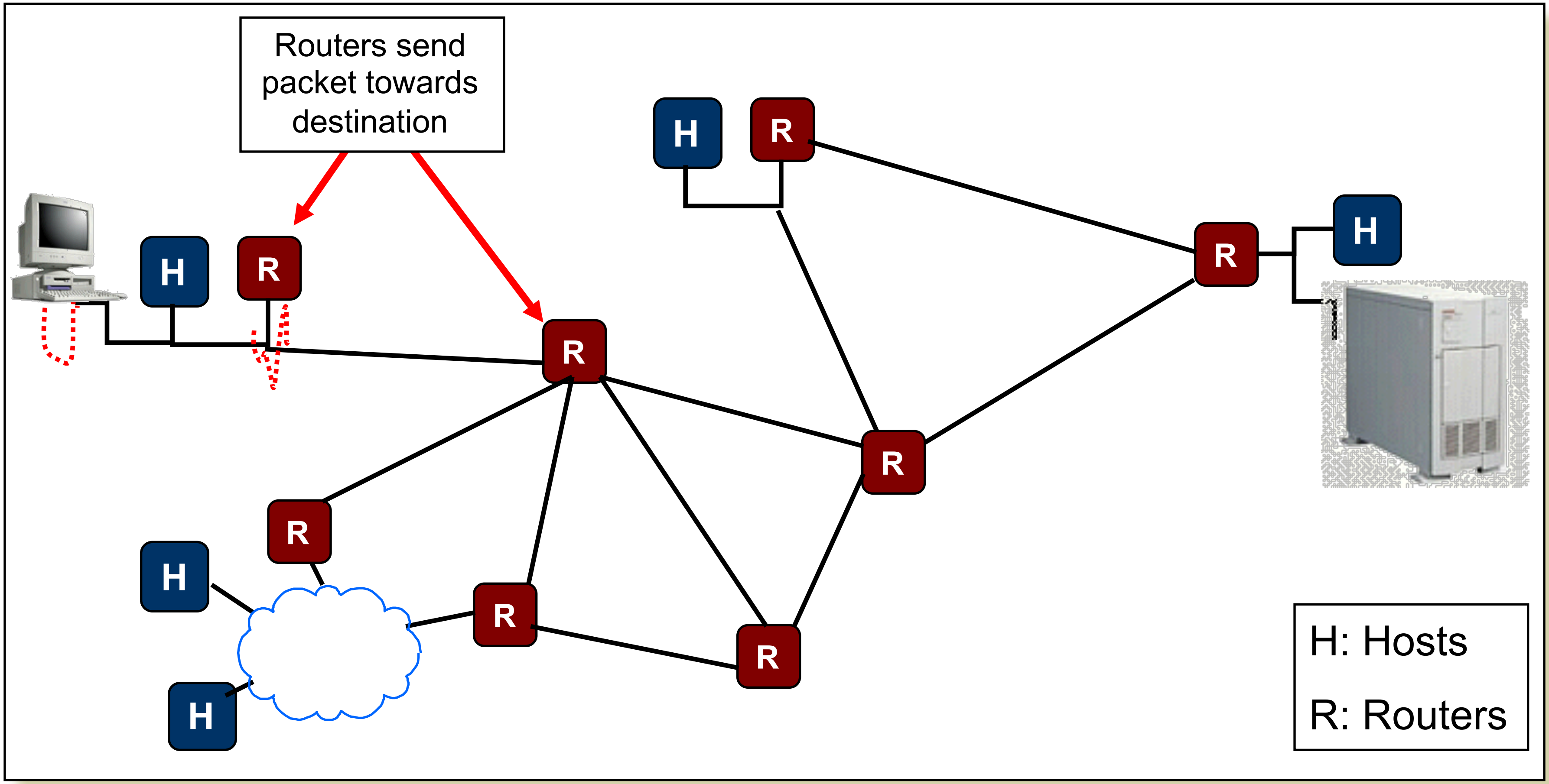
It is 128.2.11.43

Computer 1

Local DNS Server

Translates human readable names to logical endpoints

Routing



Routers send packet towards destination

H: Hosts

R: Routers

Network Service Model

- What is the *service model* for inter-network?
  - Defines what promises that the network gives for any transmission
  - Defines what type of failures to expect
- ***best-effort***
  - Ethernet/Internet– packets can get lost, etc.

Possible Failure models

- Fail-stop:
  - When something goes wrong, the process stops / crashes / etc.
- Fail-slow or fail-stutter:
  - Performance may vary on failures as well
- Byzantine:
  - Anything that can go wrong, will.
  - Including malicious entities taking over your computers and making them do whatever they want.
- These models are useful for proving things;
- The real world typically has a bit of everything.

- Deciding which model to use is important!

Example: project 1

- Project 1: Build a bitcoin miner
- Server --- many clients
- Communication:
  - Send job
  - ACK job
  - do some work
  - send result to server
  - (repeat)

- IP communication model:
  - Messages may be lost, re-ordered, corrupted (we'll ignore corruption, mostly, except for some sanity checking)
- Fail-stop node model:
  - You don't need to worry about evil participants faking you out.

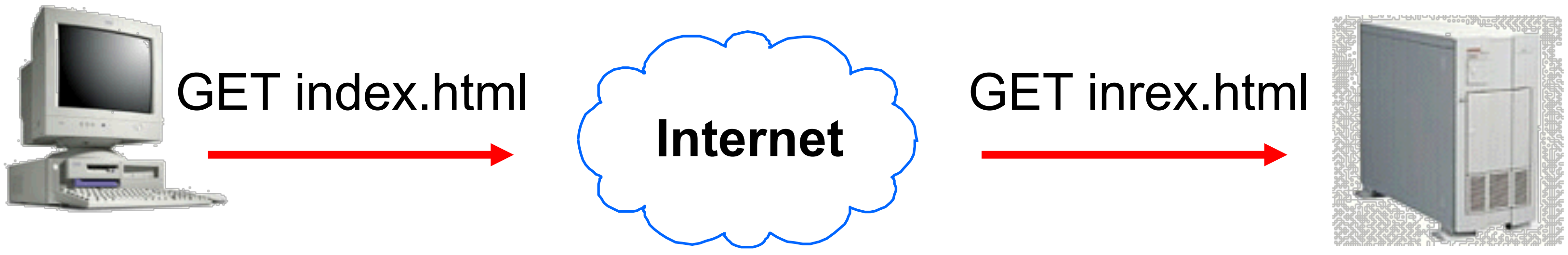
Fancier Network Service Models

- What if you want more?
  - Performance guarantees (QoS)
  - Reliability
    - Corruption
    - Lost packets
  - Flow and congestion control
  - Fragmentation
  - In-order delivery
  - Etc…

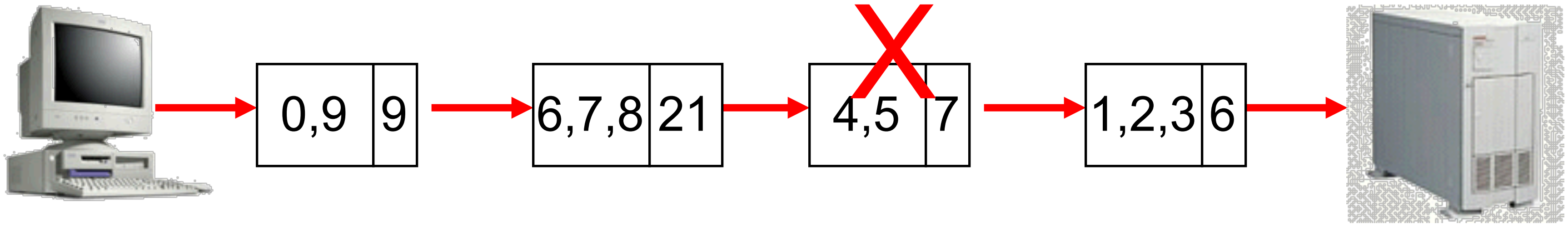
- If network provided this, programmers don’t have to implement these features in every application
- But note limitations: this can’t turn a byzantine failure model into a fail-stop model...

What if the Data gets Corrupted?

Problem: Data Corruption

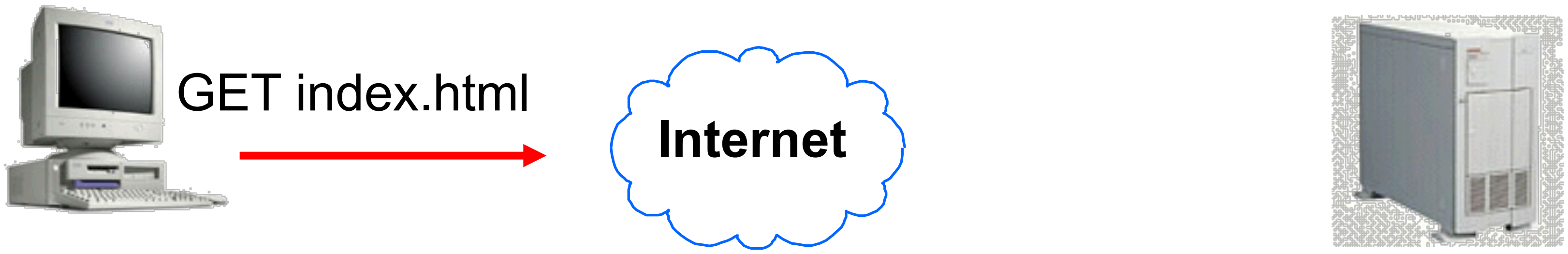


Solution: Add a *checksum*



What if the Data gets Lost?

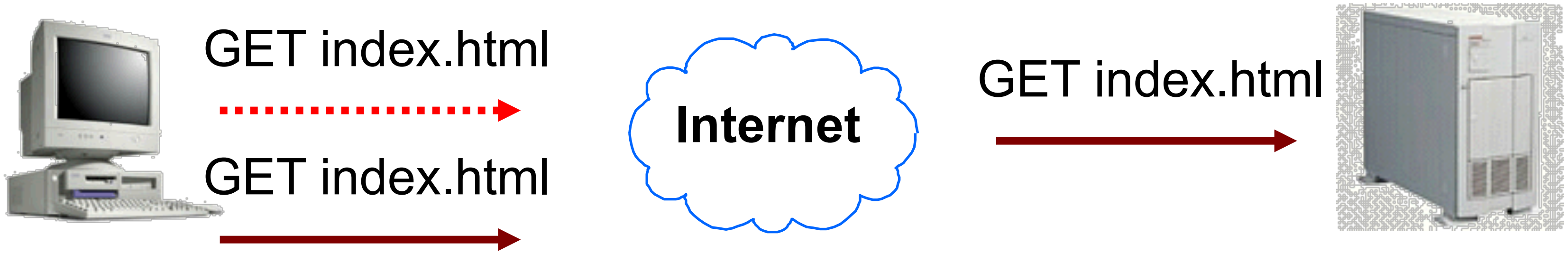
Problem: Lost Data



GET index.html

Internet

Solution: Timeout and Retransmit



GET index.html

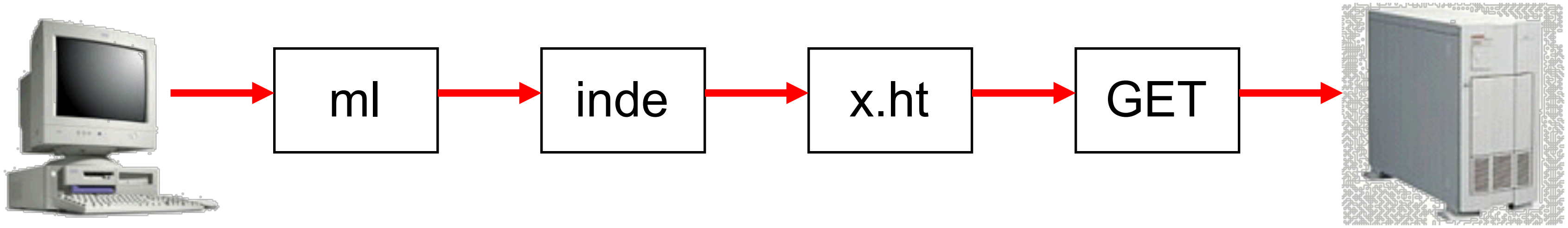
GET index.html

Internet

GET index.html

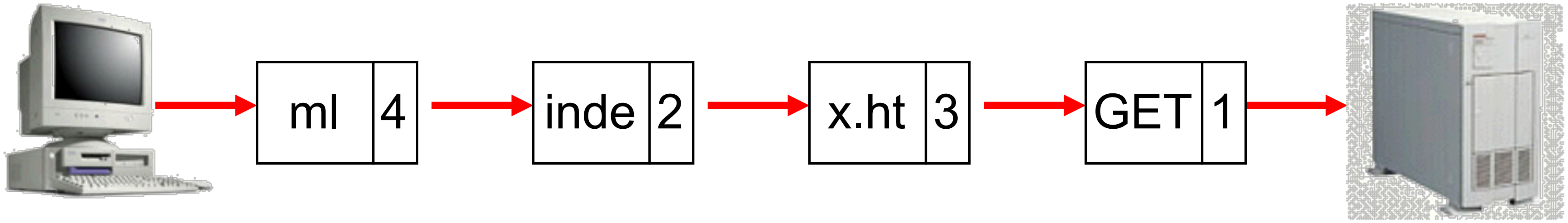
What if the Data is Out of Order?

Problem: Out of Order



GET x.htindeml

Solution: Add Sequence Numbers



GET index.html

15-440 Distributed Systems

Lecture 2 & 3 – 15-441 in 2 Days

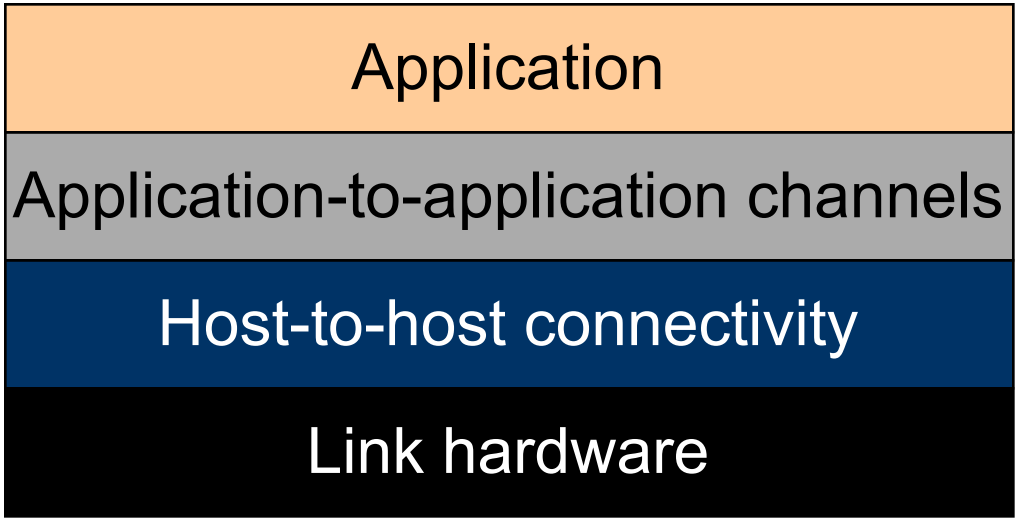
(Part 2)

Networks [including end points] Implement Many Functions

- Link
- Multiplexing
- Routing
- Addressing/naming (locating peers)
- Reliability
- Flow control
- Fragmentation
- Etc.…

What is Layering?

- Modular approach to network functionality
- Example:



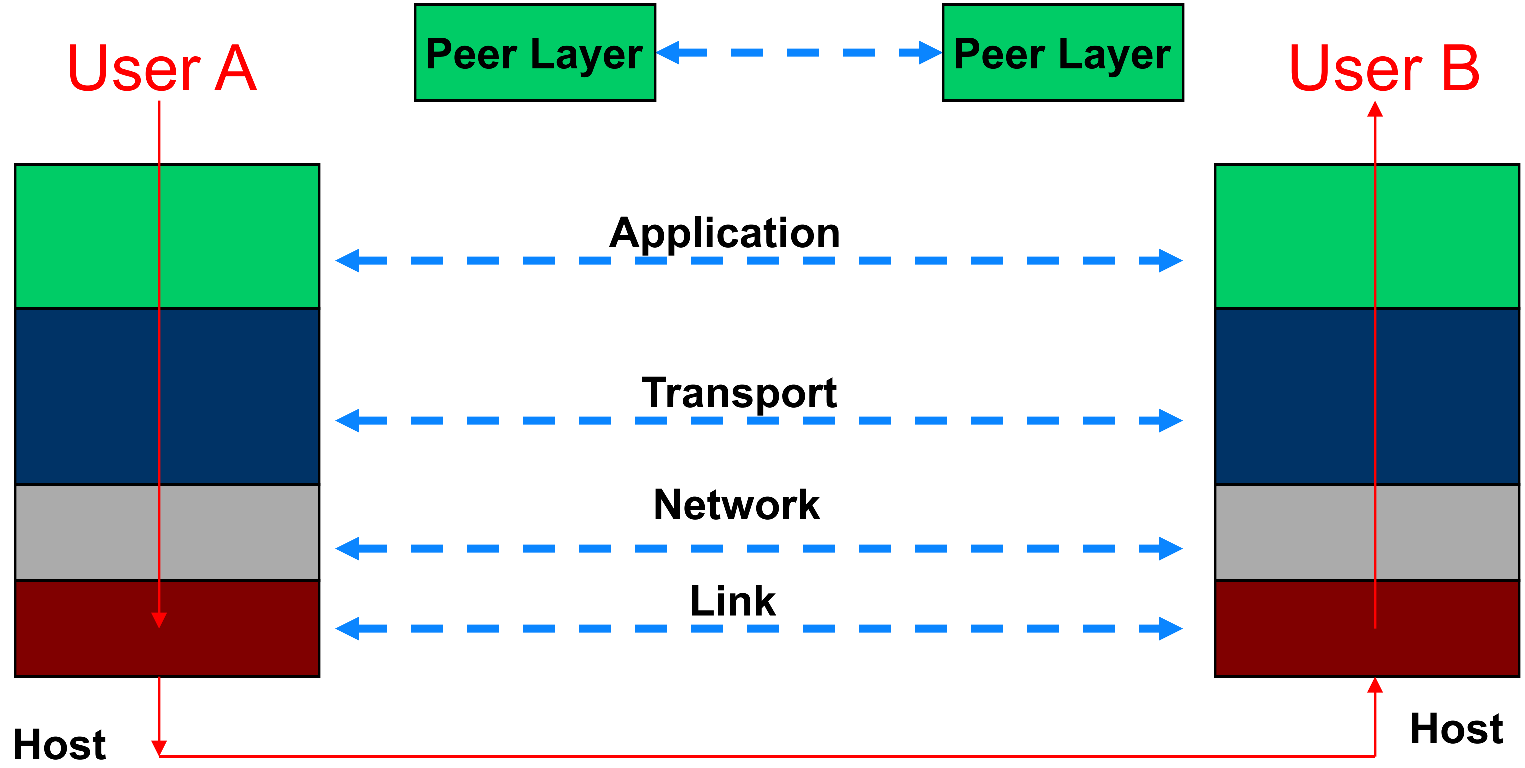
Application

Application-to-application channels

Host-to-host connectivity

Link hardware

What is Layering?



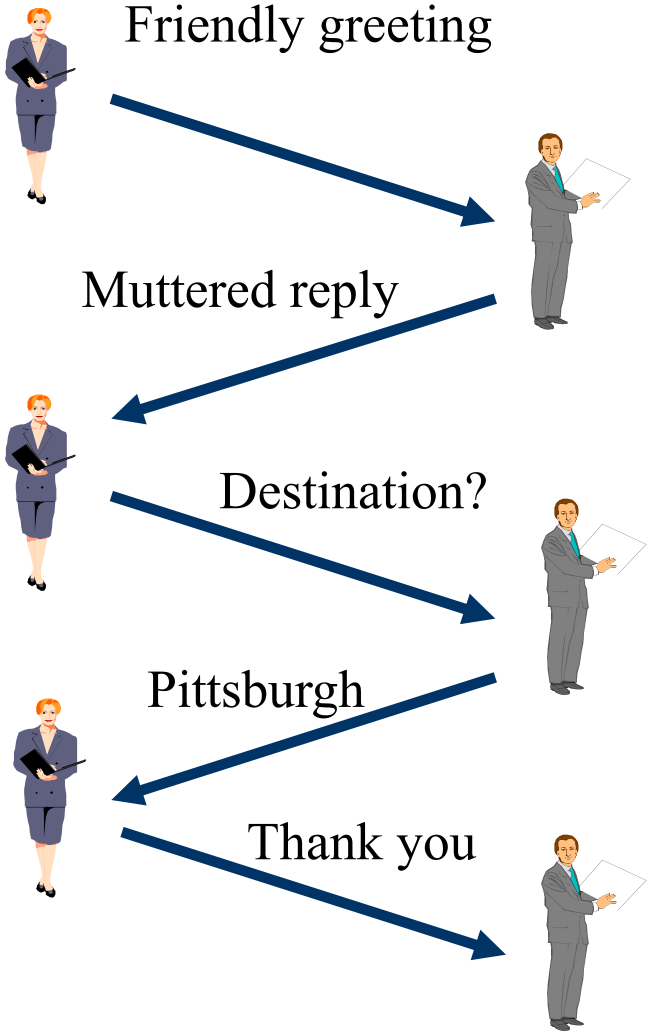
Modular approach to network functionality

Layering Characteristics

- Each layer relies on services from layer below and exports services to layer above
- Interface defines interaction with peer on other hosts
- Hides implementation - layers can change without disturbing other layers (black box)

What are Protocols?

- An agreement between parties on how communication should take place
- Module in layered structure
- Protocols define:
  - Interface to higher layers (API)
  - Interface to peer (syntax & semantics)
    - Actions taken on receipt of a messages
    - Format and order of messages
    - Error handling, termination, ordering of requests, etc.
- Example: Buying airline ticket



Friendly greeting

Muttered reply

Destination?

Pittsburgh

Thank you

Today’s & Tuesday’s Lecture

- Network links and LANs
- Layering and protocols
- Internet design
- Transport protocols
- Application design

Goals [Clark88]

0 **Connect existing networks**

initially ARPANET and ARPA packet radio network

1. Survivability

   ensure communication service even in the presence of network and router failures

2. Support multiple types of services
3. Must accommodate a variety of networks
4. Allow distributed management
5. Allow host attachment with a low level of effort
6. Be cost effective
7. Allow resource accountability

Goal 0: Connecting Networks

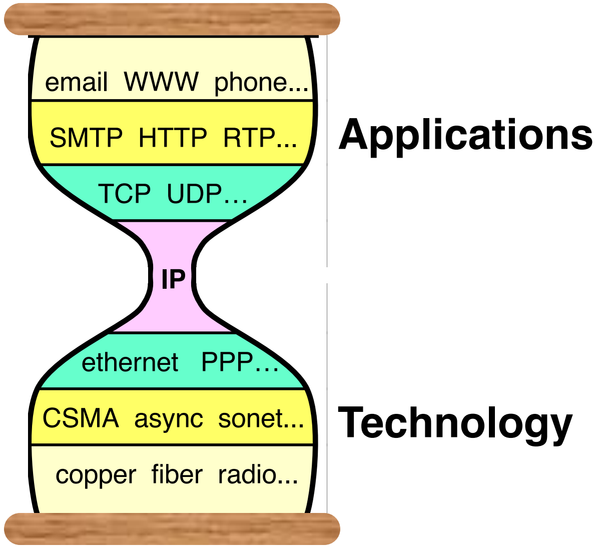
- How to internetwork various network technologies
  - ARPANET, X.25 networks, LANs, satellite networks, packet networks, serial links…
- Many differences between networks
  - Address formats
  - Performance – bandwidth/latency
  - Packet size
  - Loss rate/pattern/handling
  - Routing

Gateway Alternatives

- Translation
  - Difficulty in dealing with different features supported by networks
  - Scales poorly with number of network types (N^2 conversions)
- Standardization
  - “IP over everything” (**Design Principle 1**)
  - Minimal assumptions about network
  - Hourglass design

IP Hourglass

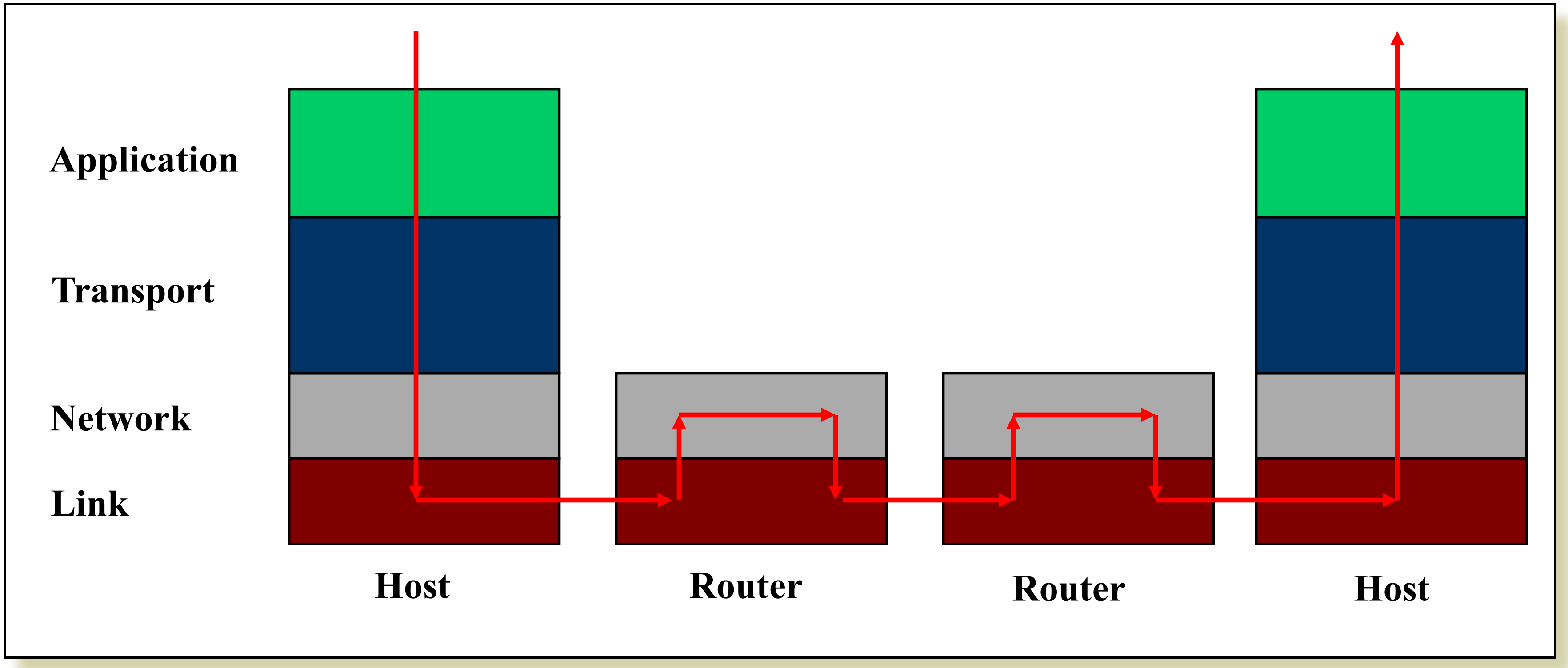
- Need to interconnect many existing networks
- Hide underlying technology from applications
- Decisions:
  - Network provides minimal functionality
  - “Narrow waist”



Tradeoff: **No assumptions, no guarantees.**

IP Layering (Principle 2)

- Relatively simple
- Sometimes taken too far



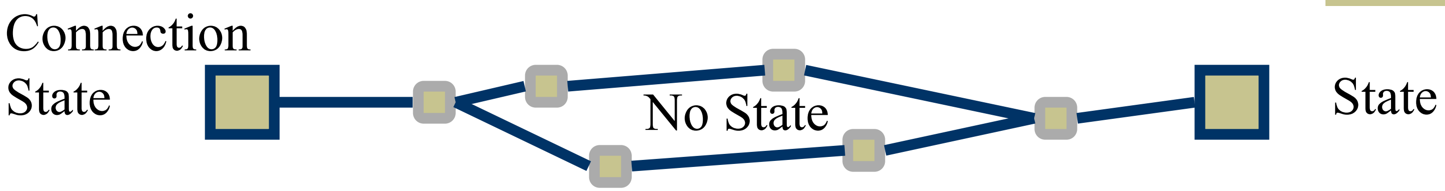
Goal 1: Survivability

- If network is disrupted and reconfigured…
  - Communicating entities should not care!
  - No higher-level state reconfiguration

- How to achieve such reliability?
  - Where can communication state be stored?

	Network	Host
Failure handing	Replication	“Fate sharing”
Net Engineering	Tough	Simple
Switches	Maintain state	Stateless
Host trust	Less	More

Fate Sharing



- Lose state information for an entity if and only if the entity itself is lost.
- Examples:
  - OK to lose TCP state if one endpoint crashes
    - NOT okay to lose if an intermediate router reboots
- Tradeoffs
  - Survivability: Heterogeneous network → less information available to end hosts and Internet level recovery mechanisms
  - Trust: must trust endpoints more

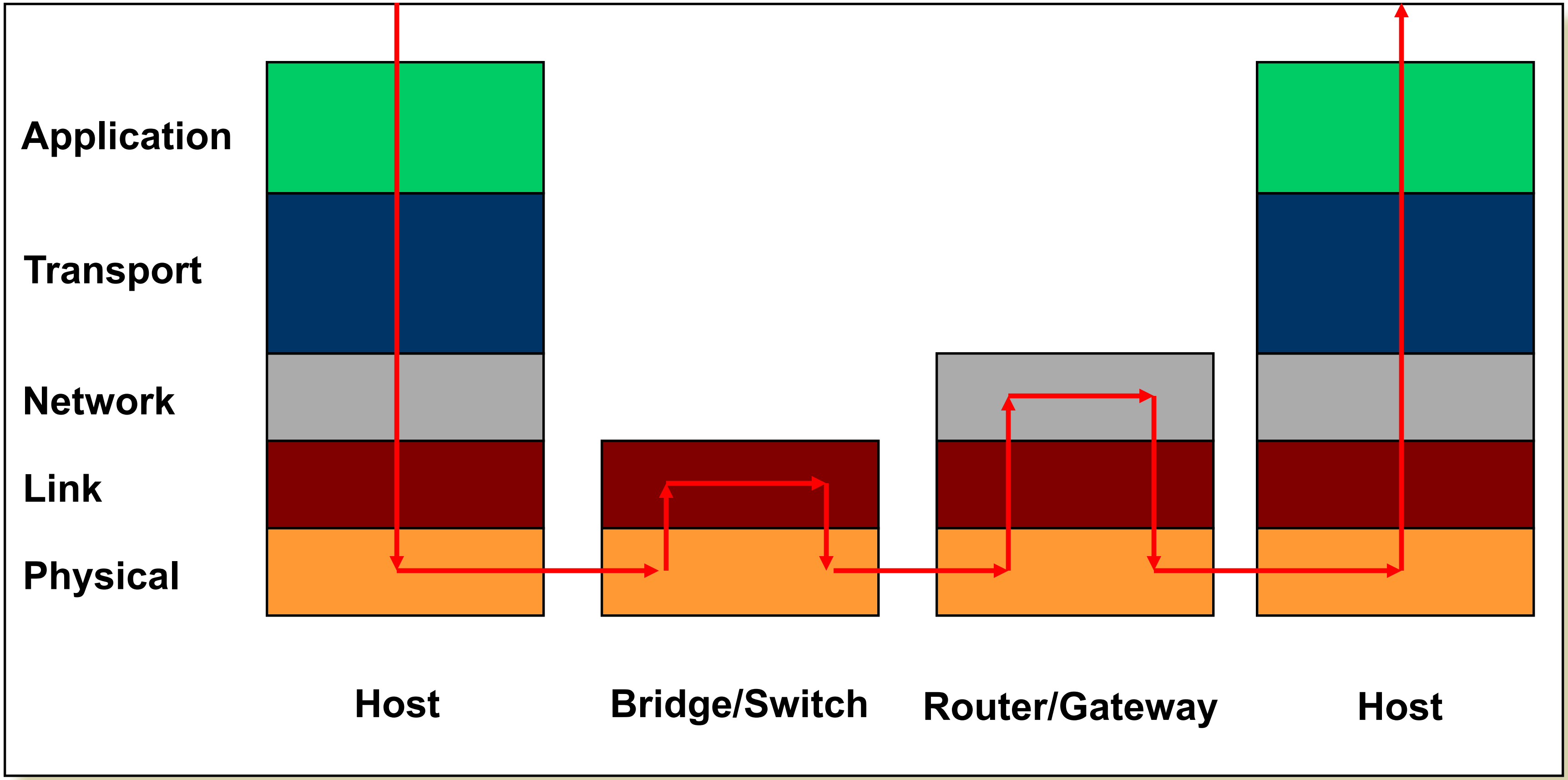
End-to-End Argument

- Deals with where to place functionality
  - Inside the network (in switching elements)
  - At the edges
- Argument
  - If you have to implement a function end-to-end anyway (e.g., because it requires the knowledge and help of the end-point host or application), **don't implement it inside the communication system**
  - Unless there's a compelling performance enhancement
- Key motivation for split of functionality between TCP,UPD and IP

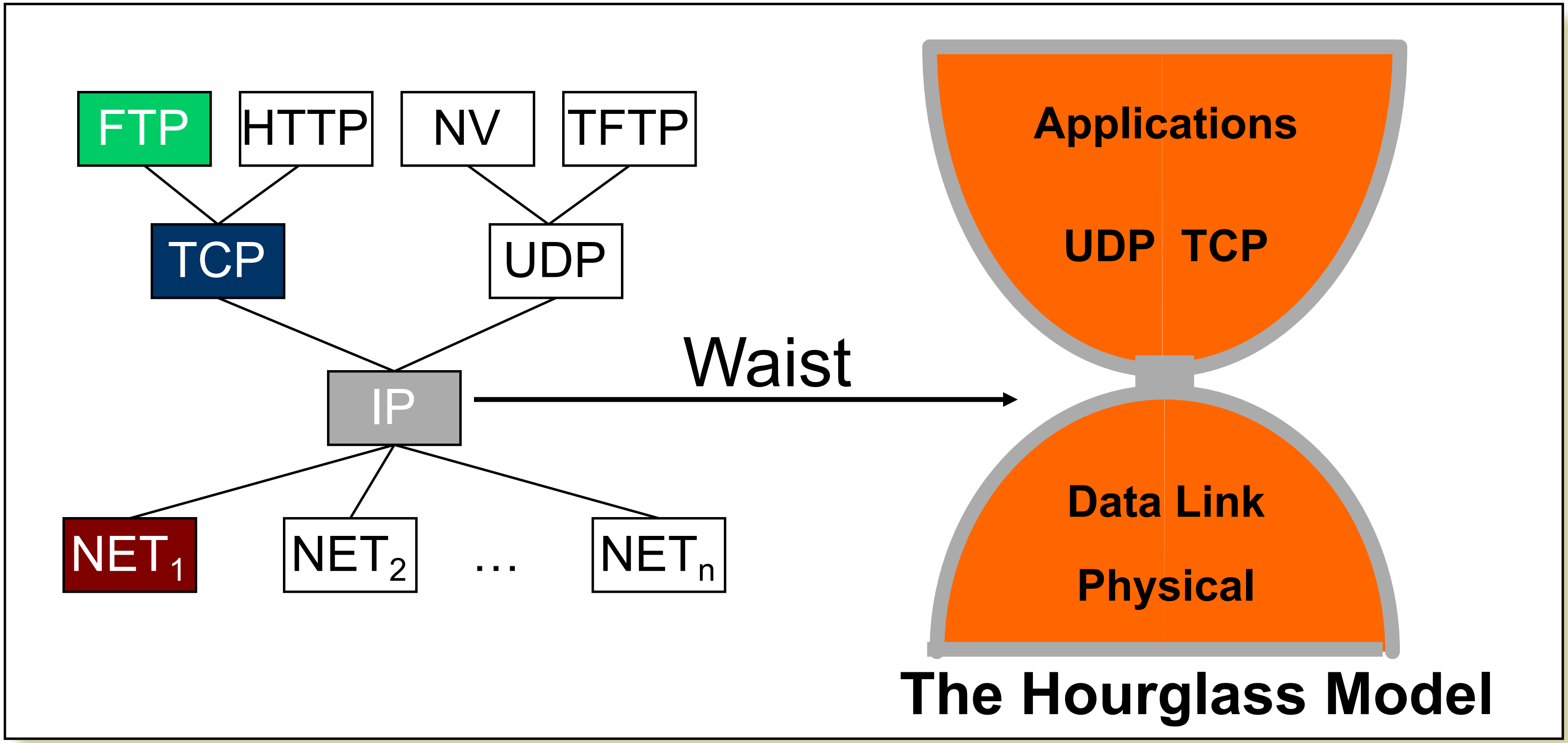
Further Reading: "End-to-End Arguments in System Design." Saltzer, Reed, and Clark.

IP Layering

- Relatively simple



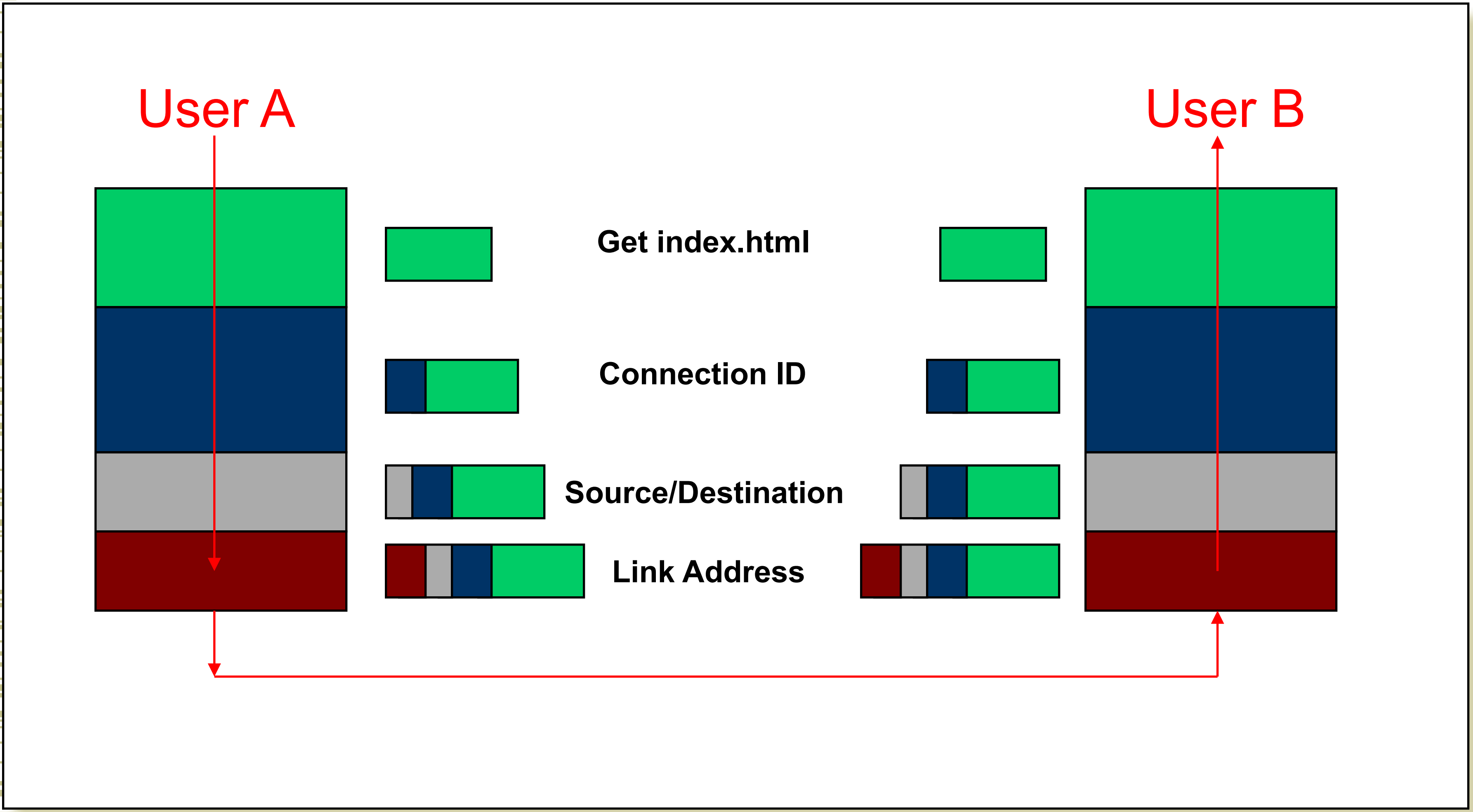
The Internet Protocol Suite



The Hourglass Model

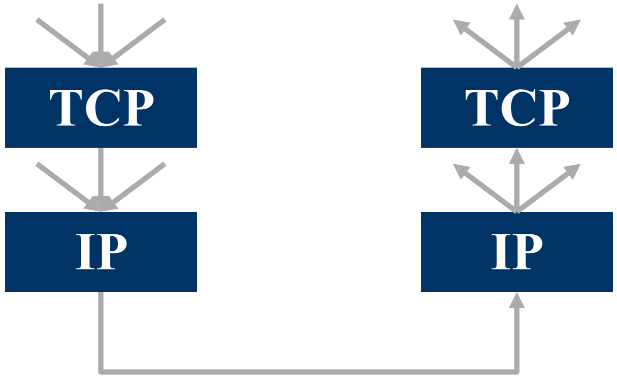
The waist facilitates interoperability

Layer Encapsulation



Multiplexing and Demultiplexing

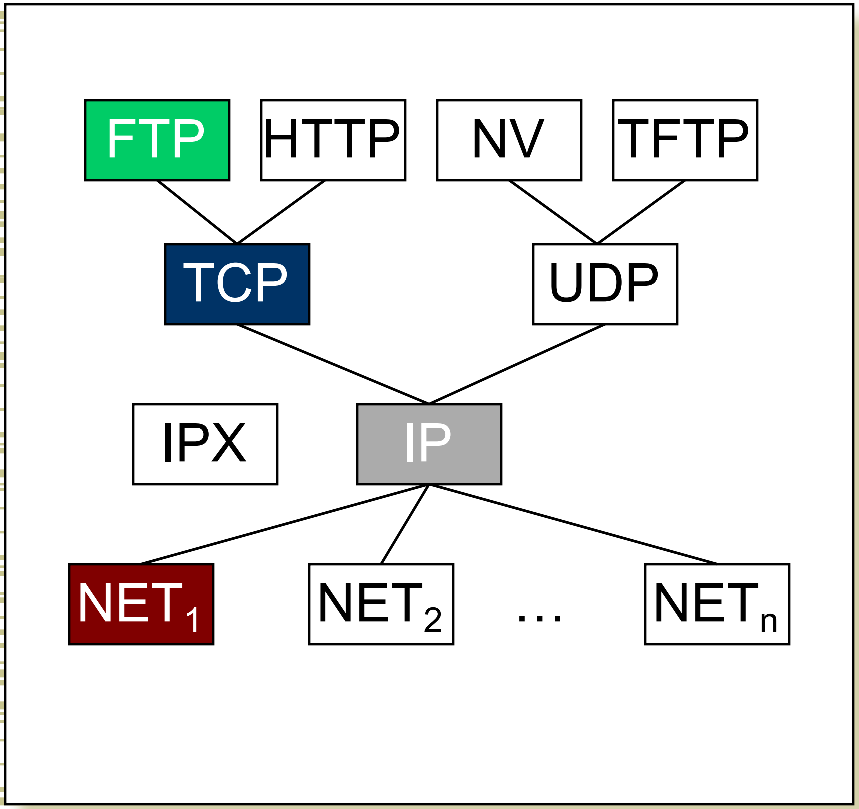
- There may be multiple implementations of each layer.
  - How does the receiver know what version of a layer to use?
- Each header includes a demultiplexing field that is used to identify the next layer.
  - Filled in by the sender
  - Used by the receiver
- Multiplexing occurs at multiple layers. E.g., IP, TCP, …

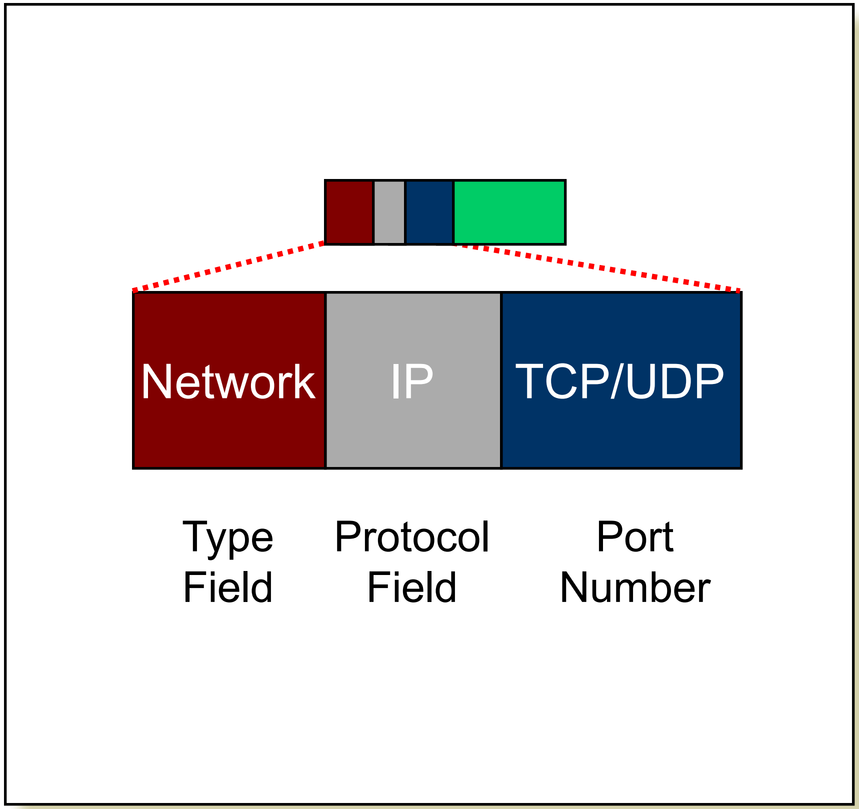


<table>
<tr><td>V/HL</td><td>TOS</td><td>Length</td></tr>
<tr><td colspan="2">ID</td><td>Flags/Offset</td></tr>
<tr><td>TTL</td><td>Prot.</td><td>H. Checksum</td></tr>
<tr><td colspan="3">Source IP address</td></tr>
<tr><td colspan="3">Destination IP address</td></tr>
<tr><td colspan="3">Options..</td></tr>
</table>

Protocol Demultiplexing

- Multiple choices at each layer





IP Packets/Service Model

- Low-level communication model provided by Internet
- Datagram
  - Each packet self-contained
    - All information needed to get to destination
    - No advance setup or connection maintenance
  - Analogous to letter or telegram

IPv4 Packet Format

0 4 8 12 16 19 24 28 31

<table>
<tr><td>version</td><td>HLen</td><td>TOS</td><td colspan="2">Length</td></tr>
<tr><td colspan="3">Identifier</td><td>Flag</td><td>Offset</td></tr>
<tr><td colspan="2">TTL</td><td>Protocol</td><td colspan="2">Checksum</td></tr>
<tr><td colspan="5">Source Address</td></tr>
<tr><td colspan="5">Destination Address</td></tr>
<tr><td colspan="5">Options (if any)</td></tr>
<tr><td colspan="5">Data</td></tr>
</table>

Header

Aside: Interaction with Link Layer

- How does one find the Ethernet address of a IP host?
- ARP
  - Broadcast search for IP address
    - E.g., “who-has 128.2.184.45 tell 128.2.206.138” sent to Ethernet broadcast (all FF address)
  - Destination responds (only to requester using unicast) with appropriate 48-bit Ethernet address
    - E.g, “reply 128.2.184.45 is-at 0:d0:bc:f2:18:58” sent to 0:c0:4f:d:ed:c6

IP Addresses: How to Get One?

Network (network portion):

- Get allocated portion of ISP’s address space:

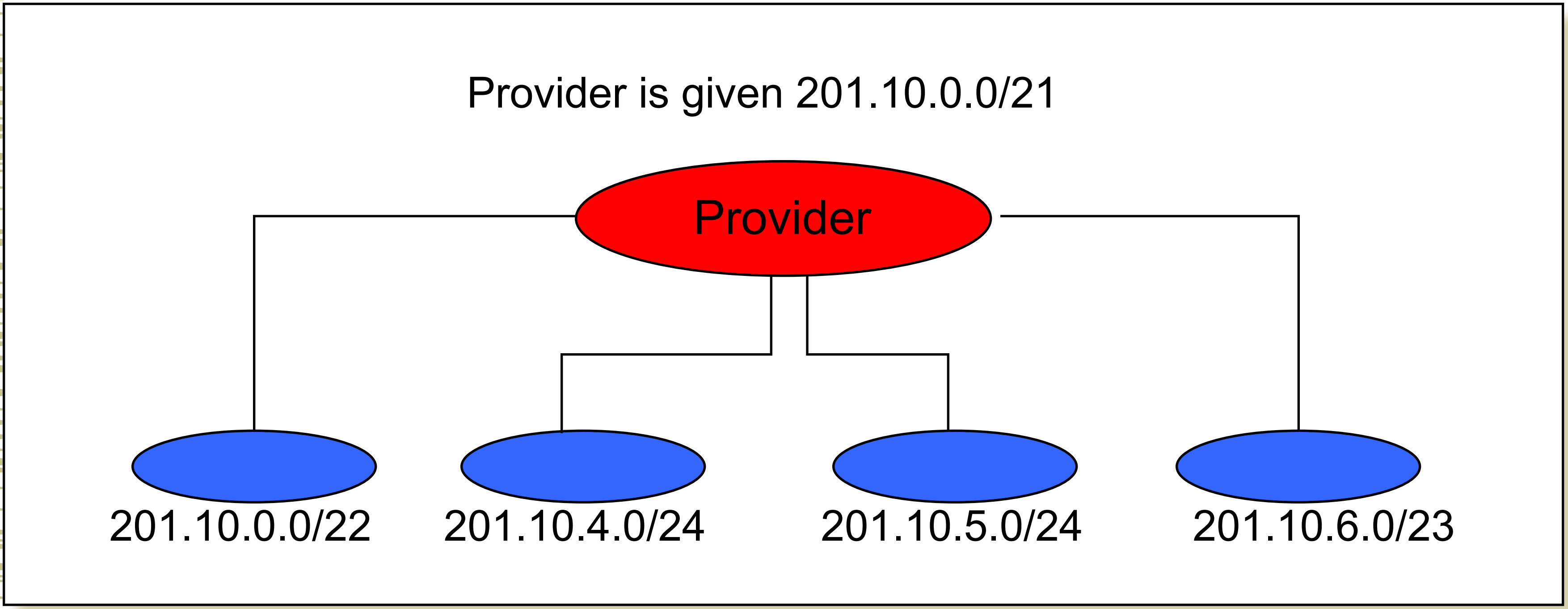
ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	…..	….	….
Organization 7	11001000 00010111 00011110	00000000	200.23.30.0/23

IP Addresses: How to Get One?

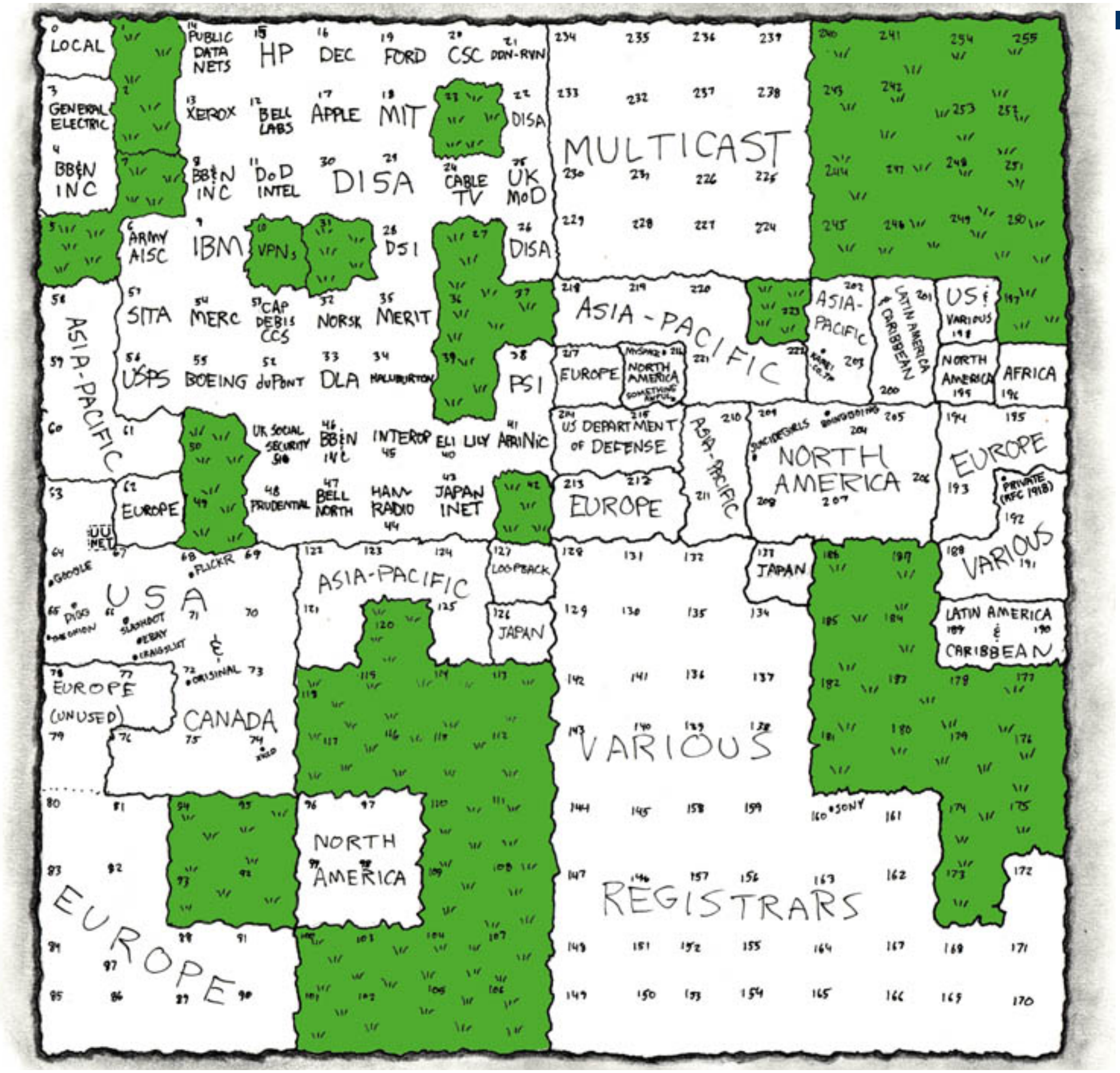
- How does an ISP get block of addresses?
  - From **Regional Internet Registries** (RIRs)
    - ARIN (North America, Southern Africa), APNIC (Asia-Pacific), RIPE (Europe, Northern Africa), LACNIC (South America)

- How about a single host?
  - Hard-coded by system admin in a file
  - DHCP: Dynamic Host Configuration Protocol: dynamically get address: "plug-and-play"
    - Host broadcasts "DHCP discover" msg
    - DHCP server responds with "DHCP offer" msg
    - Host requests IP address: "DHCP request" msg
    - DHCP server sends address: "DHCP ack" msg

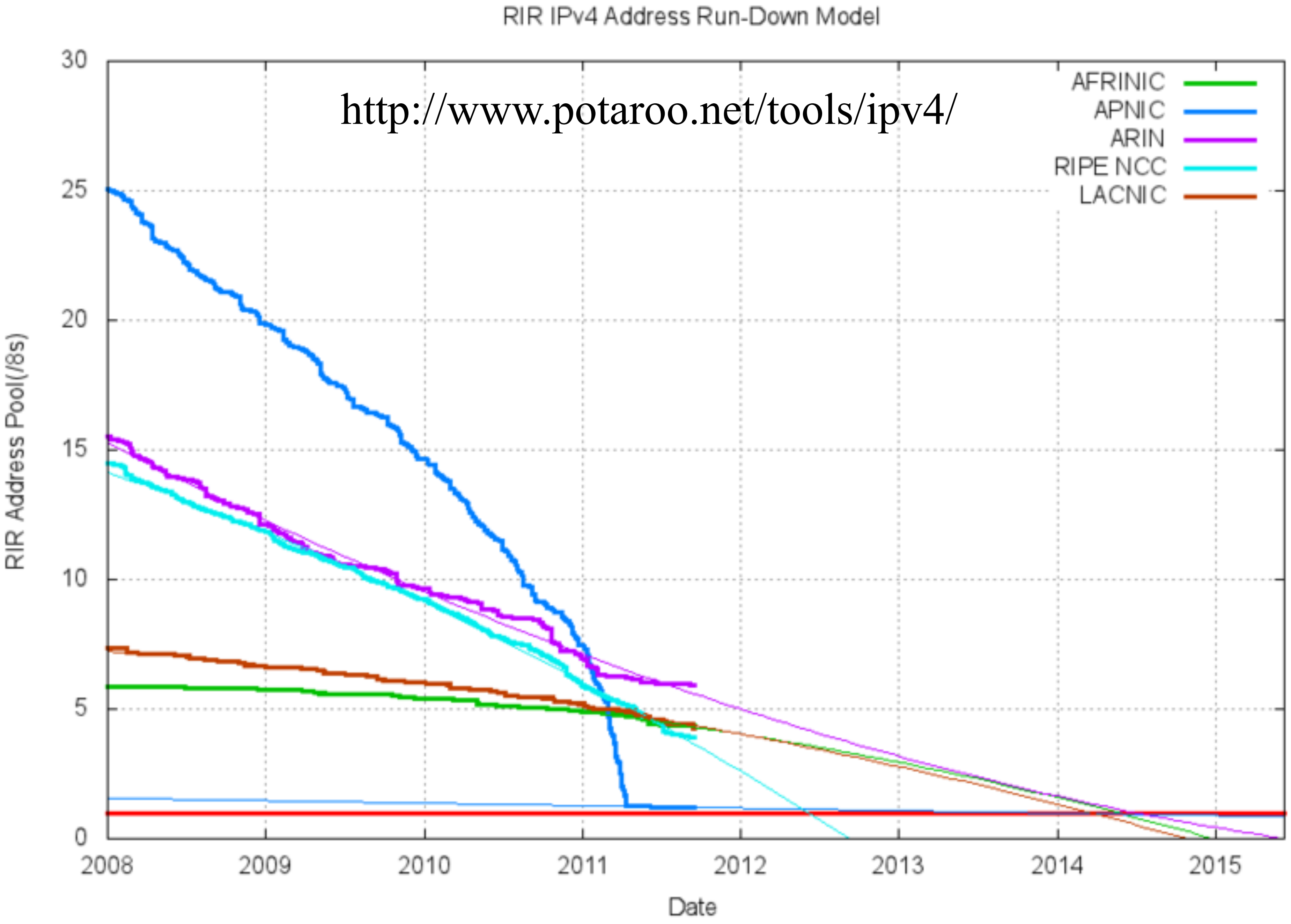
CIDR IP Address Allocation



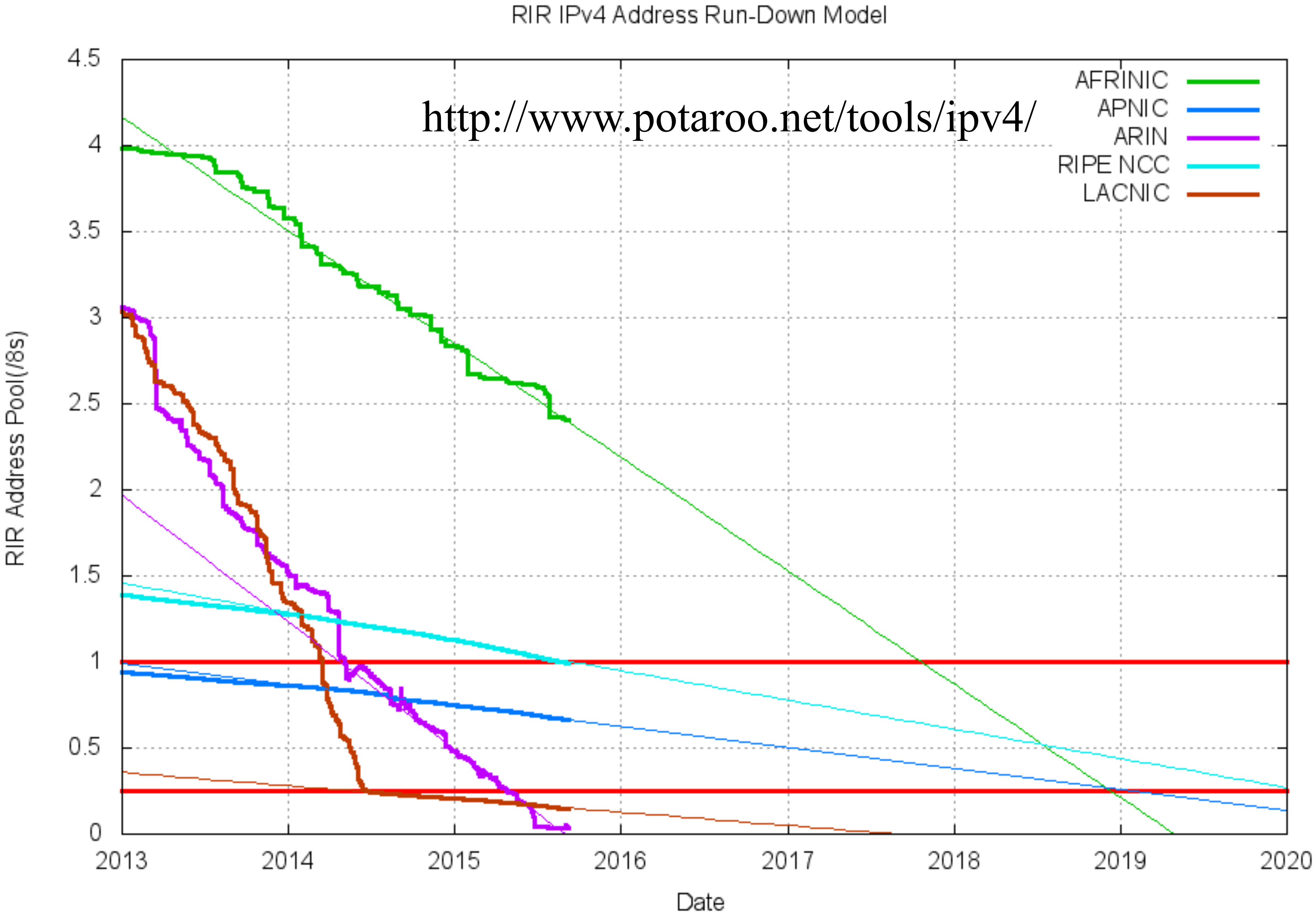
IP Address Utilization ('06)



http://xkcd.com/195/



http://www.potaroo.net/tools/ipv4/



http://www.potaroo.net/tools/ipv4/

What Now?

- Last /8 given to RIR in 1/2011
- Mitigation
  - Reclaim addresses (e.g. Stanford gave back class A in 2000)
  - More NAT?
  - Resale markets
  - Slow down allocation from RIRs to LIRs (i.e. ISPs)
- IPv6?

Host Routing Table Example

```
Destination     Gateway         Genmask         Iface
128.2.209.100   0.0.0.0         255.255.255.255 eth0
128.2.0.0       0.0.0.0         255.255.0.0     eth0
127.0.0.0       0.0.0.0         255.0.0.0       lo
0.0.0.0         128.2.254.36    0.0.0.0         eth0
```

- From “netstat –rn”
- Host 128.2.209.100 when plugged into CS ethernet
- Dest 128.2.209.100 → routing to same machine
- Dest 128.2.0.0 → other hosts on same ethernet
- Dest 127.0.0.0 → special loopback address
- Dest 0.0.0.0 → default route to rest of Internet
  - Main CS router: gigrouter.net.cs.cmu.edu (128.2.254.36)

Today's & Tuesday's Lecture

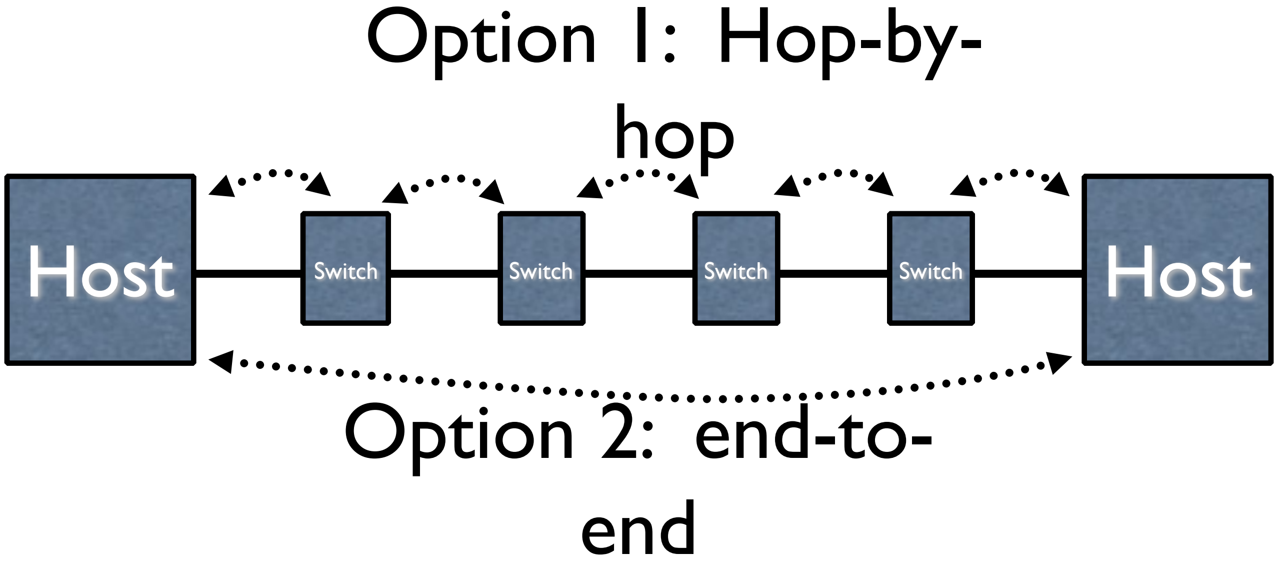
- Network links and LANs
- Layering and protocols
- Internet design
- Transport protocols
- Application design

Networks [including end points] Implement Many Functions

- Link
- Multiplexing
- Routing
- Addressing/naming (locating peers)
- **Reliability**
- Flow control
- Fragmentation
- Etc….

Design Question

- If you want reliability, etc.
- Where should you implement it?



A question

- Is hop-by-hop enough?
  - [hint: What happens if a switch crashes? What if it’ s buggy and goofs up a packet?]

Internet Design: Types of Service

- **Principle**: network layer provides one simple service: best effort datagram (packet) delivery
  - All packets are treated the same

- Relatively simple core network elements
- Building block from which other services (such as reliable data stream) can be built
- Contributes to scalability of network

- No QoS support assumed from below
  - In fact, some underlying nets only supported reliable delivery
    - Made Internet datagram service less useful!
  - Hard to implement without network support
  - QoS is an ongoing debate…

Types of Service

- TCP vs. UDP
  - Elastic apps that need reliability: remote login or email
  - Inelastic, loss-tolerant apps: real-time voice or video
  - Others in between, or with stronger requirements
  - Biggest cause of delay variation: reliable delivery
    - Today’s net: ~100ms RTT
    - Reliable delivery can add *seconds*.
- Original Internet model: “TCP/IP” one layer
  - First app was remote login…
  - But then came debugging, voice, etc.
  - These differences caused the layer split, added UDP

Transport Protocols

- UDP provides just integrity and demux
- TCP adds…
  - Connection-oriented
  - Reliable
  - Ordered
  - Point-to-point
  - Byte-stream
  - Full duplex
  - Flow and congestion controlled

User Datagram Protocol (UDP): An Analogy

UDP

- Single socket to receive messages
- No guarantee of delivery
- Not necessarily in-order delivery
- Datagram – independent packets
- Must address each packet

Postal Mail

- Single mailbox to receive letters
- Unreliable ☺
- Not necessarily in-order delivery
- Letters sent independently
- Must address each letter

Example UDP applications

Multimedia, voice over IP

Transmission Control Protocol (TCP): An Analogy

TCP

- Reliable – guarantee delivery
- Byte stream – in-order delivery
- Connection-oriented – single socket per connection
- Setup connection followed by data transfer

Telephone Call

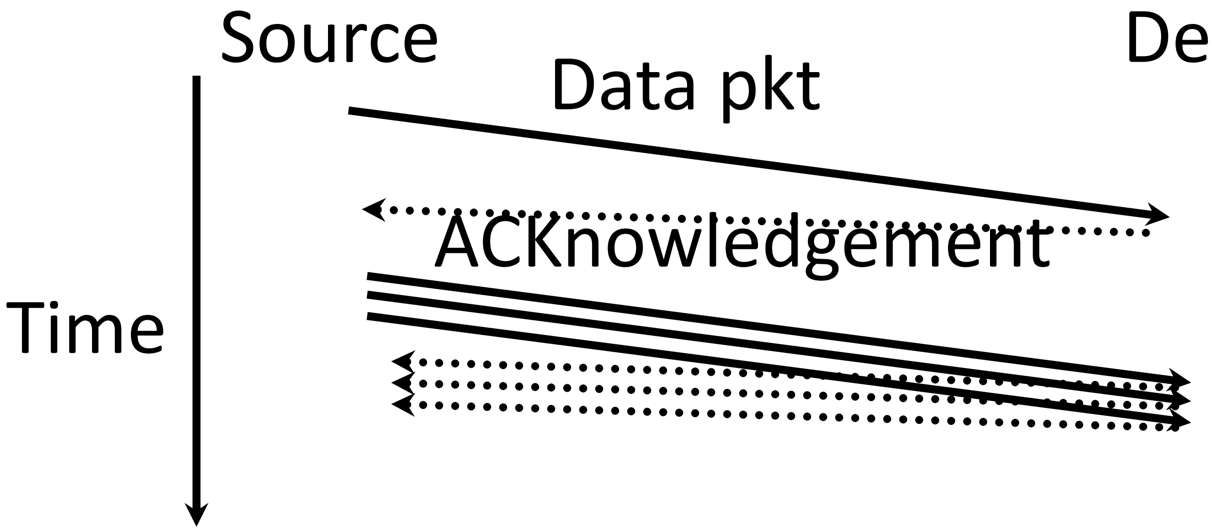
- Guaranteed delivery
- In-order delivery
- Connection-oriented
- Setup connection followed by conversation

Example TCP applications

Web, Email, Telnet

Rough view of TCP

(This is a *very* incomplete view - take 15-441. :)



What TCP does:

1) Figures out which packets got through/lost

2) Figures out how fast to send packets to use all of the unused capacity,

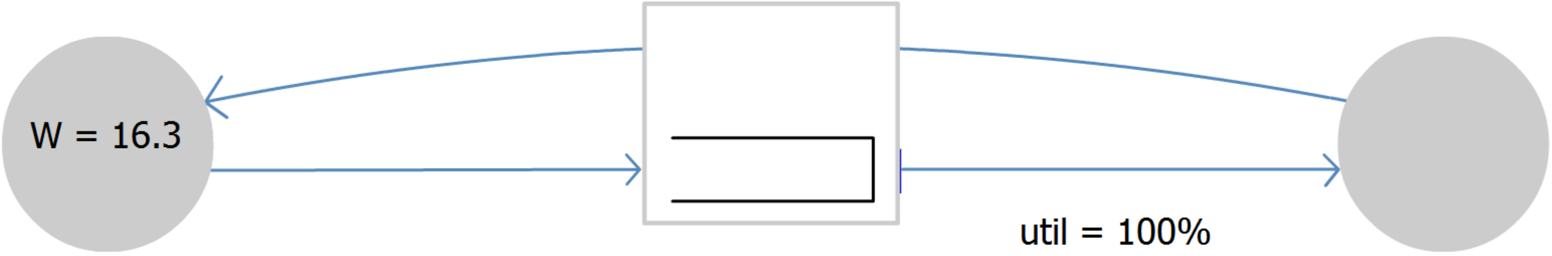
- But not more
- And to share the link approx. equally with other senders

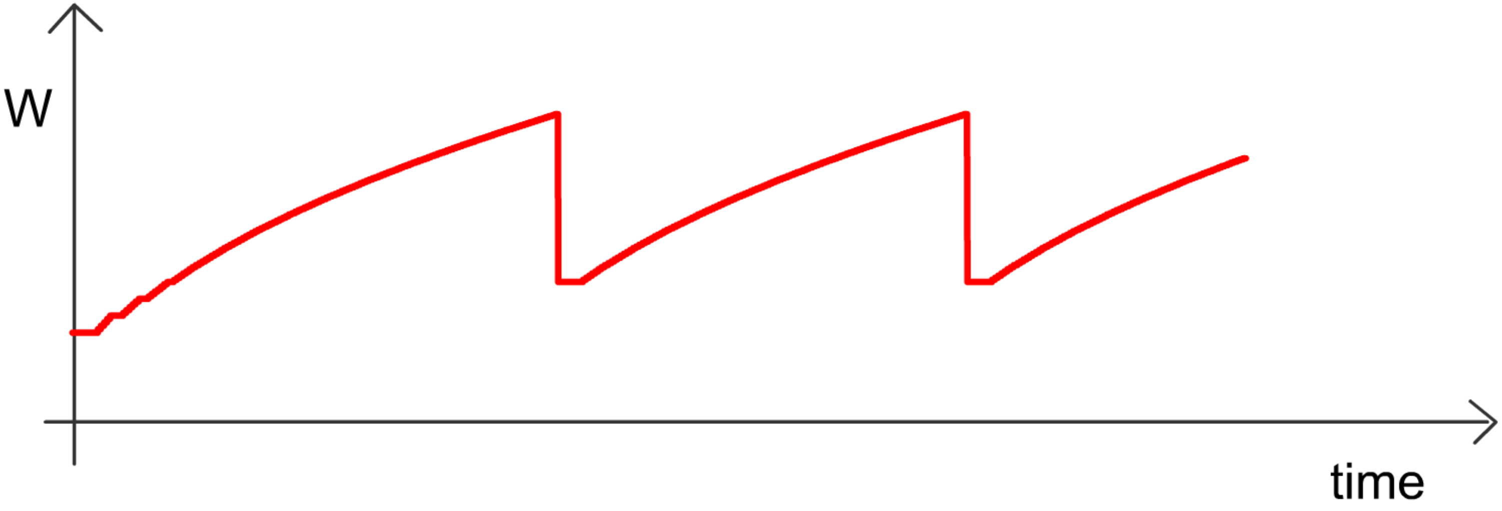
Questions to ponder

- If you have a whole file to transmit,
  how do you send it over the Internet?
  - You break it into packets (packet-switched medium)
  - TCP, roughly speaking, has the sender tell the receiver "got it!" every time it gets a packet. The sender uses this to make sure that the data's getting through.
  - If you acknowledge the correct receipt of the entire file (e.g. due to e2e argument)... why bother acknowledging the receipt of the individual packets???
- The answer: Imagine the waste if you had to retransmit the entire file because one packet was lost. Ow.

Single TCP Flow

Router with large enough buffers for full link utilization





Today’s & Tuesday’s Lecture

- Network links and LANs
- Layering and protocols
- Internet design
- Transport protocols
- Application design

Client-Server Paradigm

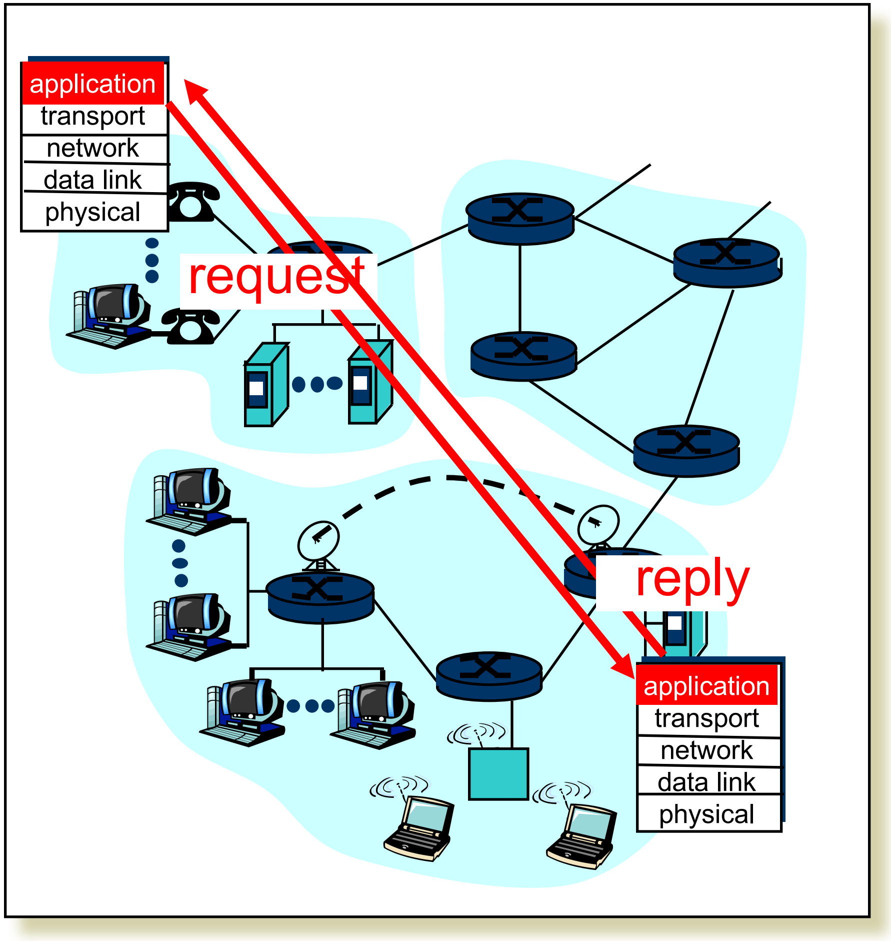
Typical network app has two pieces: *client* and *server*

Client:

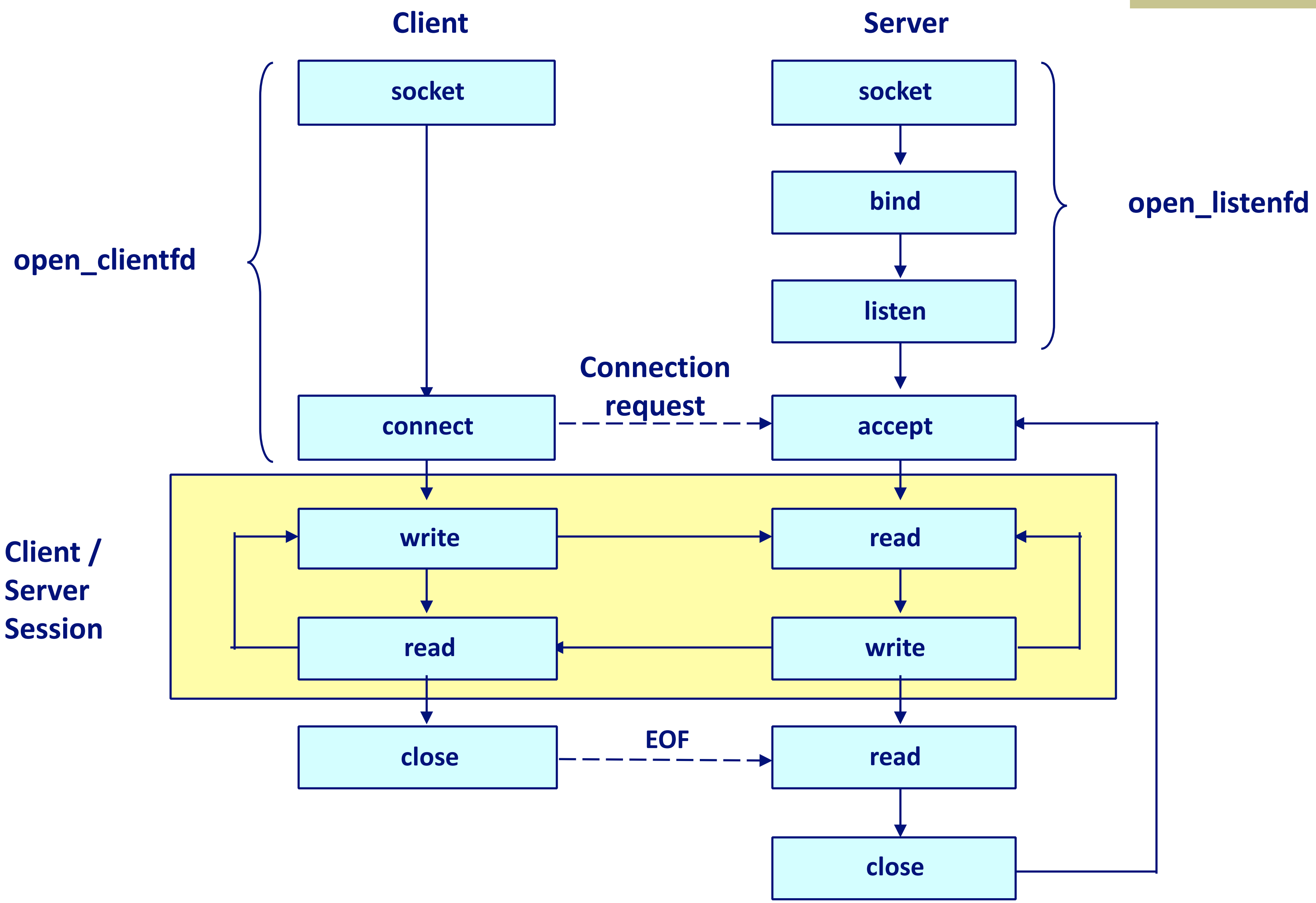
- Initiates contact with server (“speaks first”)
- Typically requests service from server,
- For Web, client is implemented in browser; for e-mail, in mail reader

Server:

- Provides requested service to client
- e.g., Web server sends requested Web page, mail server delivers e-mail



Socket API Operation Overview



What Service Does an Application Need?

Data loss

- Some apps (e.g., audio) can tolerate some loss
- Other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Timing

- Some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”

Bandwidth

- Some apps (e.g., multimedia) require minimum amount of bandwidth to be “effective”
- Other apps (“elastic apps”) make use of whatever bandwidth they get

Transport Service Requirements of Common Apps

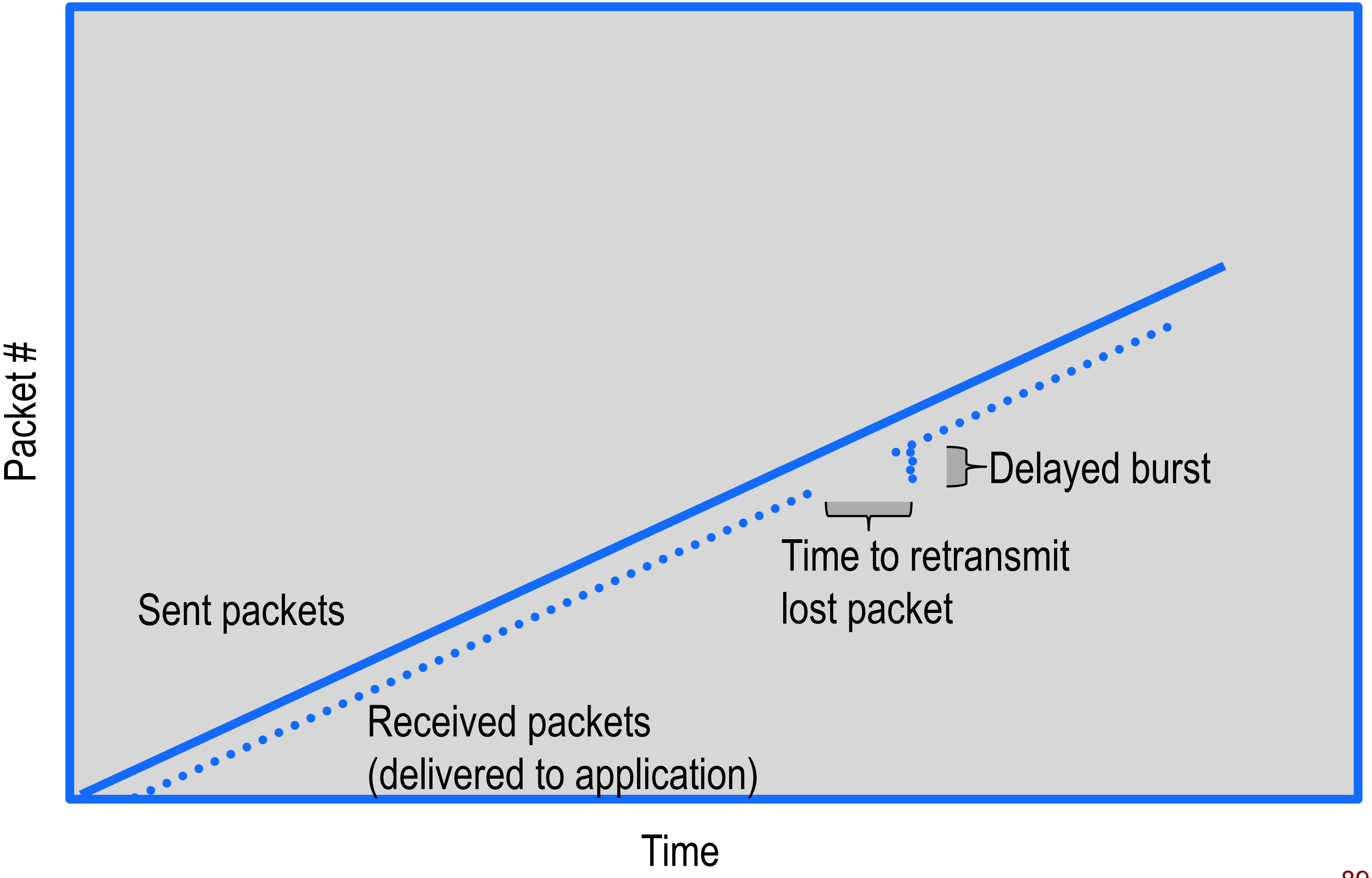
Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
web documents	no loss	elastic	no
interactive audio/video	loss-tolerant (often)	audio: 5Kb-1Mb video:10Kb-5Mb	yes, 100’ s msec
non-interactve audio/video	loss-tolerant (sometimes)	same as above	yes, few secs
interactive games	loss-tolerant	few Kbps	yes, 100’ s msec
financial apps	no loss	elastic	yes and no: μs?

Why not always use TCP?

- TCP provides “more” than UDP
- Why not use it for everything??

- A: Nothing comes for free...
  - Connection setup (take on faith) -- TCP requires one round-trip time to setup the connection state before it can chat...
  - How long does it take, using TCP, to fix a lost packet?
    - At minimum, one “round-trip time” (2x the latency of the network)
    - That could be 100+ milliseconds!
  - If I guarantee in-order delivery,
    what happens if I lose one packet in a stream of packets?

One lost packet



Design trade-off

- If you’ re building an app...

- Do you need everything TCP provides?
- If not:
  - Can you deal with its drawbacks to take advantage of the subset of its features you need?

    OR
  - You’re going to have to implement the ones you need on top of UDP
    - Caveat: There are some libraries, protocols, etc., that can help provide a middle ground.
    - Takes some looking around - they’ re not as standard as UDP and TCP.

Blocking sockets

- What happens if an application write()s to a socket waaaaay faster than the network can send the data?
    - TCP figures out how fast to send the data...
    - And it builds up in the kernel socket buffers at the sender... and builds...
    - until they fill. The next write() call *blocks* (by default).
    - What’s blocking? It suspends execution of the blocked thread until enough space frees up...

In contrast to UDP

- UDP doesn’t figure out how fast to send data, or make it reliable, etc.
- So if you write() like mad to a UDP socket...
- It often silently disappears. *Maybe* if you’re lucky the write() call will return an error. But no promises.

Web Page Retreival

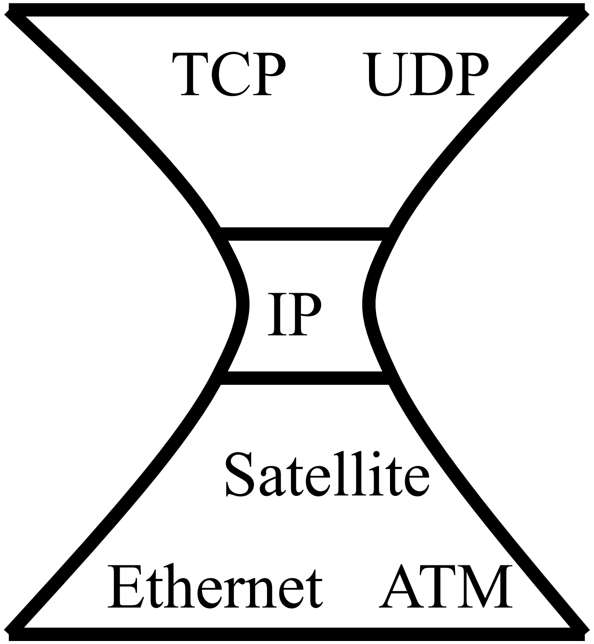
1. Static configuration
   - IP address, DNS server IP address, IP address of routers,
2. ARP for router
3. DNS lookup for web server
   - Several packet exchanges for lookup
4. TCP SYN exchange
5. HTTP Get request
6. HTTP response
   - Slow start, retransmissions, etc.

Caching Helps

1. Static configuration
    - IP address, DNS server IP address, IP address of routers,
2. ARP for router
3. DNS lookup for web server
    - Several packet exchanges for lookup
4. TCP SYN exchange
5. HTTP Get request
6. HTTP response
    - Slow start, retransmissions, etc.

Summary: Internet Architecture

- Packet-switched datagram network
- IP is the “compatibility layer”
  - Hourglass architecture
  - All hosts and routers run IP
- Stateless architecture
  - no per flow state inside network



Summary: Minimalist Approach

- Dumb network
  - IP provide minimal functionalities to support connectivity
    - Addressing, forwarding, routing
- Smart end system
  - Transport layer or application performs more sophisticated functionalities
    - Flow control, error control, congestion control
- Advantages
  - Accommodate heterogeneous technologies (Ethernet, modem, satellite, wireless)
  - Support diverse applications (telnet, ftp, Web, X windows)
  - Decentralized network administration

Rehashing all of that...

- TCP is layered on top of IP
  - IP understands only the IP header
  - The IP header has a “protocol” ID that gets set to TCP
  - The TCP at the receiver understands how to parse the TCP information
- IP provides only “best-effort” service
- TCP adds value to IP by adding retransmission, in-order delivery, data checksums, etc., so that programmers don’t have to re-implement the wheel every time. It also helps figure out how fast to send data. This is why TCP sockets can “block” from the app perspective.
- The e2e argument suggests that functionality that must be implemented end-to-end anyway (like retransmission in the case of dead routers) should probably be implemented only there -- unless there’s a compelling perf. optimization

Proj 1 and today’ s material

- You’ ll use UDP. Why?
  - A1: The course staff is full of sadists who want you to do a lot of work. This is true in part: timeouts and retransmission are a core aspect of using the network.
  - A2: The communication needed is very small, and you have to implement a lot of reliability stuff anyway to ensure that the work gets done...
  - Honestly? This one seems to me like a middle ground. You might use TCP for “other” reasons (firewalls that block everything but TCP), or to avoid the need for the “job ack” part of the protocol. Or you might stick with UDP to reduce the overhead at the server.

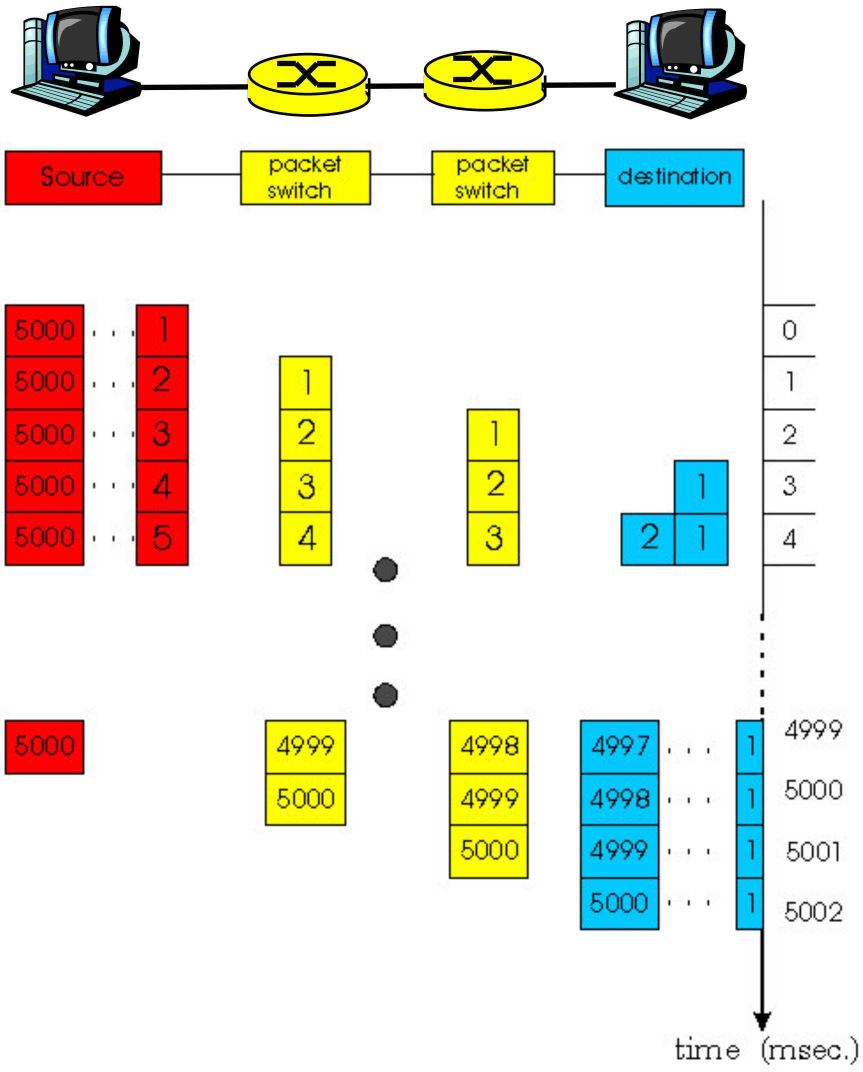
Last Time

- Modularity, Layering, and Decomposition
  - Example: UDP layered on top of IP to provide application demux (“ports”)
- Resource sharing and isolation
  - Statistical multiplexing - packet switching
- Dealing with heterogenity
  - IP “narrow waist” -- allows many apps, many network technologies
  - IP standard -- allows many impls, same proto

Application-level Delay

Delay of one packet

Average sustained throughput



$$\text{Delay}^{*} + \frac{\text{Size}}{\text{Throughput}}$$

Units: seconds + bits/(bits/seconds)

* For minimum sized packet

A Word about Units

- What do “Kilo” and “Mega” mean?
  - Depends on context
- Storage works in powers of two.
  - 1 Byte = 8 bits
  - 1 KByte = 1024 Bytes
  - 1 MByte = 1024 Kbytes
- Networks work in decimal units.
  - Network hardware sends bits, not Bytes
  - 1 Kbps = 1000 bits per second
  - To avoid confusion, use 1 Kbit/second
- Why? Historical: CS versus ECE.