15-440 Distributed Systems

Lecture 3 – 15-440 in 2 Days

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

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

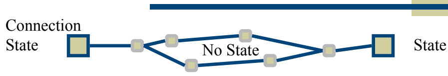
2. Support multiple types of services
3. Must accommodate a variety of networks
4. Allow distributed management
5. Allow host attachment with a low level of effort
6. Be cost effective
7. Allow resource accountability

Goal 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



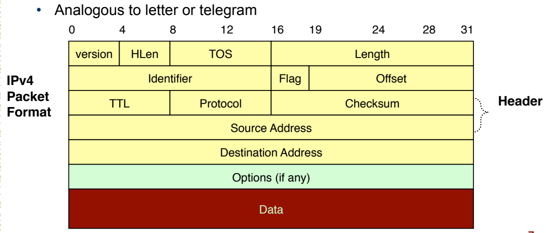
- Lose state information for an entity if and only if the entity itself is lost.
- Examples:
  - OK to lose TCP state if one endpoint crashes
    - NOT okay to lose if an intermediate router reboots
  - Is this still true in today's network?
    - NATs and firewalls
- Tradeoffs
  - Survivability: Heterogeneous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

Today's Lecture

- Internet design
- Transport protocols
- Application design

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



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):

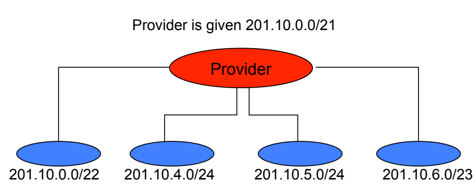
- Get allocated portion of ISP's address space:

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

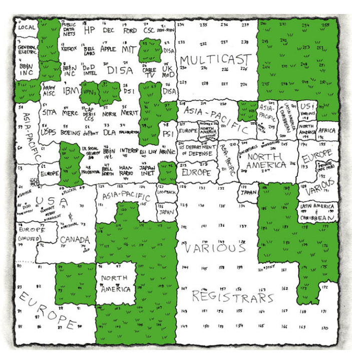
IP Addresses: How to Get One?

- 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

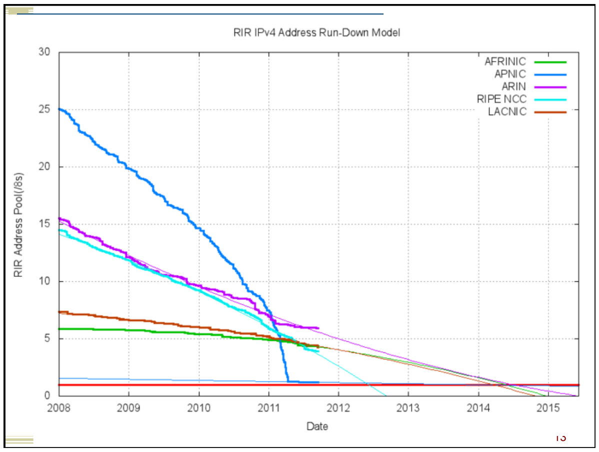
CIDR IP Address Allocation



IP Address Utilization ('06)



http://xkcd.com/195/



13

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?

14

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)

15

Today’s Lecture

- Internet design
- Transport protocols
- Application design

16

Networks [including end points] Implement Many Functions

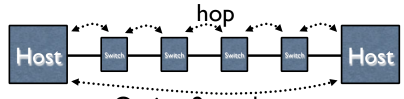
- Link
- Multiplexing
- Routing
- Addressing/naming (locating peers)
- **Reliability**
- Flow control
- Fragmentation
- Etc.…

17

Design Question

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

Option 1: Hop-by-hop



Option 2: end-to-end

Options

- Hop-by-hop: Have each switch/router along the path ensure that the packet gets to the next hop
- End-to-end: Have just the end-hosts ensure that the packet made it through
- What do we have to think about to make this decision??

A question

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

End-to-End Argument

- Deals with where to place functionality
  - Inside the network (in switching elements)
  - At the edges
- Guideline not a law
- 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

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

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

User Datagram Protocol (UDP): An Analogy

UDP	Postal Mail
Single socket to receive messages	Single mailbox to receive letters
No guarantee of delivery	Unreliable ☺
Not necessarily in-order delivery	Not necessarily in-order delivery
Datagram – independent packets	Letters sent independently
Must address each packet	Must address each letter

Example UDP applications
Multimedia, voice over IP

Transmission Control Protocol (TCP): An Analogy

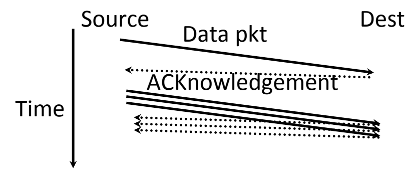
TCP	Telephone Call
• Reliable – guarantee delivery • Byte stream – in-order delivery • Connection-oriented – single socket per connection • Setup connection followed by data transfer	• Guaranteed delivery • In-order delivery • Connection-oriented • Setup connection followed by conversation

Example TCP applications
Web, Email, Telnet

25

Rough view of TCP

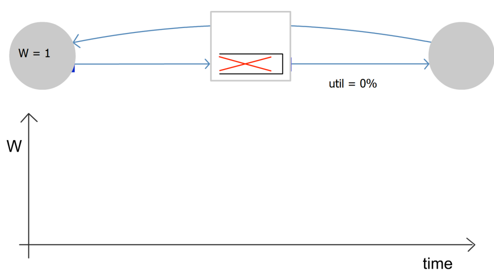
(This is a *very* incomplete view - take 15-441. :)



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

Single TCP Flow

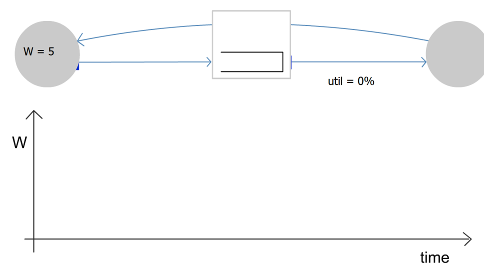
Router without buffers



27

Single TCP Flow

Router with large enough buffers for full link utilization



28

Today's Lecture

- Internet design
- Transport protocols
- Application design

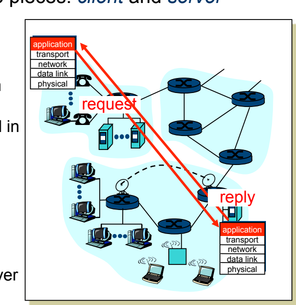
29

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



30

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	
financial apps	no loss	elastic	yes, 100’s msec

yes and no: μs?

Transport Protocols

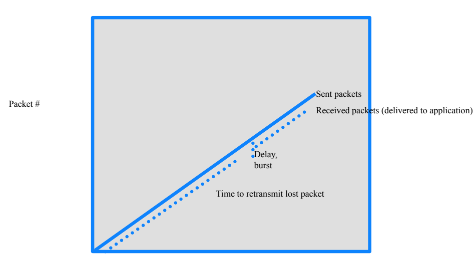
- UDP provides just integrity and demux
- TCP adds…
  - Connection-oriented
  - Reliable
  - Ordered
  - Point-to-point
  - Byte-stream
  - Full duplex
  - Flow and congestion controlled

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



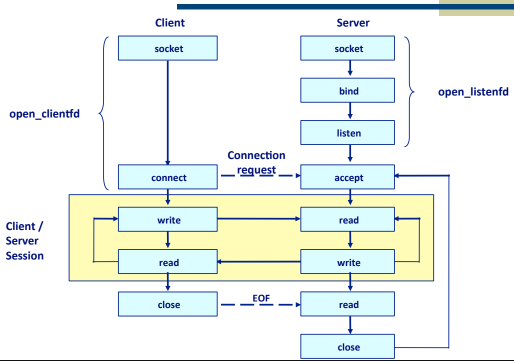
Design trade-off

- 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.

Socket API Operation Overview



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

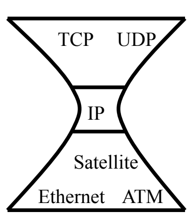
- UDP doesn't figure out how fast to send data, or make it reliable, etc.
- So if you write() like mad to a UDP socket...
- It often silently disappears. *Maybe* if you're lucky the write() call will return an error. But no promises.

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.
  - But by e2e, if you have to acknowledge the correct receipt of the entire file... why bother acknowledging the receipt of the individual packets???
- This is a bit of a trick question -- it's not asking e2e vs in-network. :)
  The answer: Imagine the waste if you had to retransmit the entire file because one packet was lost. Ow.

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



41

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

42

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.

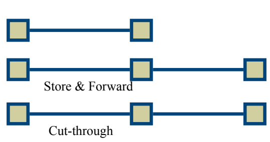
Web Page Retreival

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.

Packet Delay



Prop + xmit

2*(Prop + xmit)

2*prop + xmit

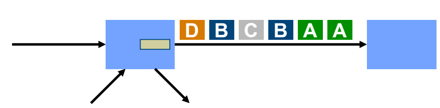
When does cut-through matter?

Next: Routers have finite speed (processing delay)

Routers may buffer packets (queueing delay)

Packet Delay

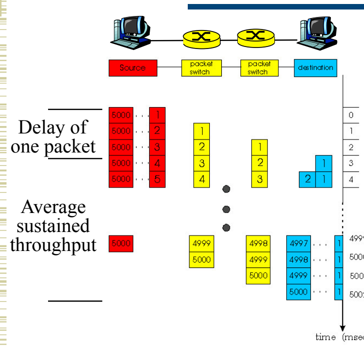
- Sum of a number of different delay components.
- Propagation delay on each link.
  - Proportional to the length of the link
- Transmission delay on each link.
  - Proportional to the packet size and 1/link speed
- Processing delay on each router.
  - Depends on the speed of the router
- Queuing delay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



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.

Application-level Delay



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

Units: seconds + bits/(bits/seconds)

* For minimum sized packet

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
  - And a 100 Kbit file

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
  - 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

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