



COS 318: Internetworking

Or, how the Internet works


Slides borrowed from Jennifer Rexford




The Internet according to Senator Ted Stevens



The Internet is not something you just dump something on. It's **not a truck**. It's a **series of tubes**. And if you don't understand, those tubes can be filled. And if they are filled, **when you put your message in, it gets in line** and it's going to be delayed by anyone that puts into that tube enormous amounts of material, enormous amounts of material.




The Internet according to Wikipedia...




The **Internet** is the worldwide, **publicly accessible** network of interconnected computer networks that transmit data by **packet switching** using the **standard** Internet Protocol (IP). It is a "**network of networks**" that consists of millions of smaller domestic, academic, business, and government networks, which together carry **various information and services**, such as electronic mail, online chat, file transfer, and the interlinked Web pages and other documents of the World Wide Web.

<http://en.wikipedia.org/wiki/Internet>


Each network is administered independently of all other networks
There is no central authority running the Internet




A Brief History Toward the Internet?




- ◆ 1960s: ARPAnet - Defense Advanced Research Project
 - Research project into packet switching networks
 - Wanted communications infrastructure capable of exploiting redundancy to route around damaged links
- ◆ 1970s: ARPA needed:
 - A common OS for researchers with ARPA funding
 - Technology to keep geographically dispersed ARPA researchers in contact with each other
 - ⇒ funding for BSD Unix project, Univ. of Calif. Berkeley
- ◆ 1980s: BSD Unix
 - Included support for Internet network protocols (**TCP/IP**)



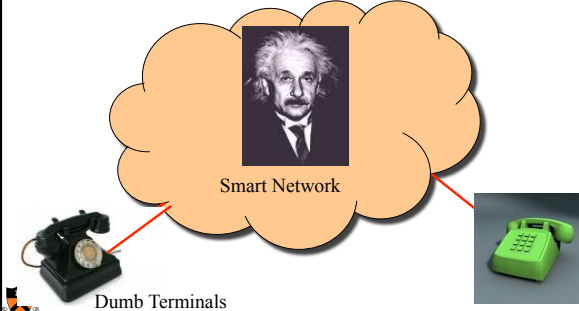
● ● ● | Key Ideas Underlying the Internet




● ● ● | Idea #1: The rise of the stupid network




Telephone Network



Dumb Terminals




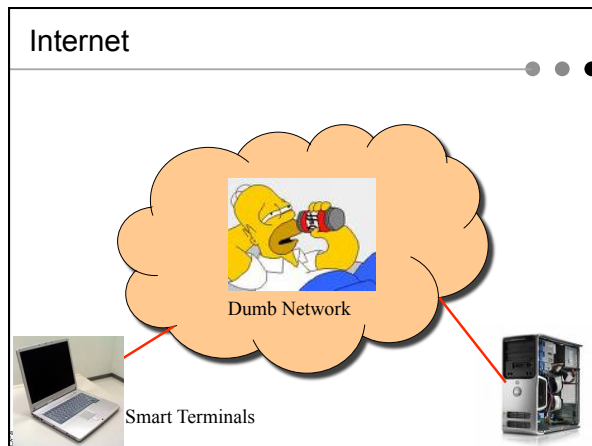
Telephone Network



An early telephone design.

- ◆ Dumb phones
 - Dial a number
 - Speak and listen
- ◆ Smart switches
 - Set up and tear down a circuit
 - Forward audio along the path
- ◆ Limited services
 - Audio
 - Later, fax, caller-id, ...
- ◆ A monopoly for a long time





Power at the Edge

End-to-End Principle

Whenever possible, communications protocol operations should be defined to occur at the **end-points** of a communications system.

Programmability

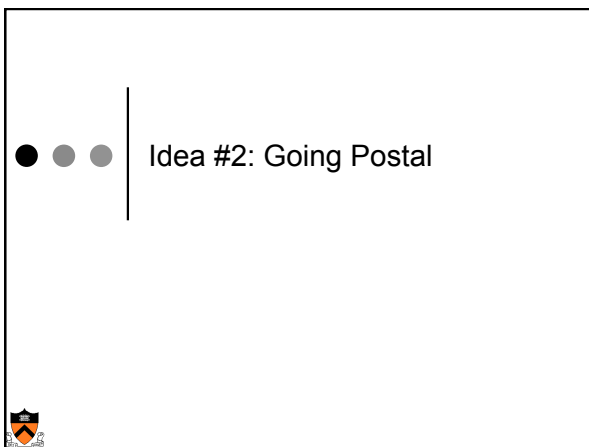
With programmable end hosts, new network services can be added at **any time, by anyone**.

And then end hosts became powerful and ubiquitous...




Let Routers Handle Reliability/Survivability?

- ◆ Need to replicate state
- ◆ End-point handling doesn't require that
- ◆ Place more trust in end hosts
- ◆ Communication involving an end point doesn't survive if end-point goes down





Internet Protocol (IP) Packet Switching



- ◆ Much like the postal system
 - Divide information into letters
 - Put them in envelopes
 - Deliver them independently
 - And usually they get there


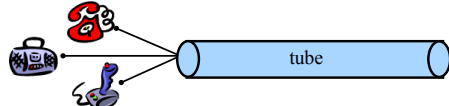

• What's in an IP packet?

- The data you want to send
- A header with the "from" and "to" addresses

Why Packets?

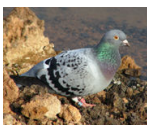

- ◆ Data traffic is bursty
 - Logging in to remote machines
 - Exchanging e-mail messages
- ◆ Don't waste bandwidth
 - No traffic exchanged during idle periods
- ◆ Easy multiplexing
 - Different transfers share access to same links, carrying different addressing information


Why Packets?

- ◆ Packets can be delivered by most anything
 - Serial link, fiber optic link, coaxial cable, wireless
- ◆ Even birds
 - RFC 1149: IP Datagrams over Avian Carriers

IP over Avian Carriers was actually implemented, in April 2001, sending 9 packets over a distance of approximately 5km (3 miles), each carried by an individual pigeon, and they received 4 responses, with a packet loss ratio of 55%, and a response time ranging from 3000 seconds to over 6000 seconds.

● ● ● | Idea #3: Never having to say you're sorry



Best-Effort Packet-Delivery Service

- Packets may be lost
- Packets may be corrupted
- Packets may be delivered out of order

The diagram illustrates a source computer on the left sending three packets (represented by red and blue rectangles) into a cloud labeled 'IP network'. From the cloud, one packet is shown being sent to a destination computer on the right.

IP Service Model: Why Best-Effort?

- ◆ I never promised you a rose garden
 - No error detection and correction
 - No need to remember from one packet to next
 - No need to reserve bandwidth and memory
- ◆ Easier to survive failures
 - Transient disruptions are okay during failover
- ◆ ... but, applications *do* want efficient, accurate transfer of data in order, in a timely fashion
- ◆ Let the end host take care of that!

Retransmit Lost and Delayed Packets

Problem: Lost, Corrupted, or Delayed Data

The diagram shows a computer on the left sending a 'GET index.html' request to a cloud labeled 'Internet'. A server on the right is shown, but no response is received.

Solution: Timeout and Retransmit

The diagram shows the same computer sending a 'GET index.html' request to the 'Internet' cloud. A clock icon is shown below, indicating a timeout. The request is then retransmitted (indicated by a second arrow) and is successfully received by the server.

Waiting for an acknowledgment...

Discard Corrupted Packets

The diagram shows a computer on the left sending a 'GET index.html' request to a cloud labeled 'Internet'. A server on the right is shown sending back a 'GET index.y.html' response, which is crossed out with a red 'X', indicating it is corrupted.

- ◆ Sender computes a checksum
 - Sender sums up all of the bytes
 - And sends the sum to the receiver
- ◆ Receiver checks the checksum
 - Received sums up all of the bytes
 - And compares against the checksum

134	
+ 212	
= 346	
<hr/>	
134	
+ 216	
= 350	

Putting Out of Order Packets Back in Order

Problem: Out of Order

Solution: Add Sequence Numbers

Preventing Buffer Overflow at the Receiver

- ◆ Window size
 - Amount that can be sent without acknowledgment
 - Receiver needs to be able to store this much data
- ◆ Receiver advertises the window to sender
 - Tells the sender the amount of free space left
 - ... and sender agrees not to exceed this amount

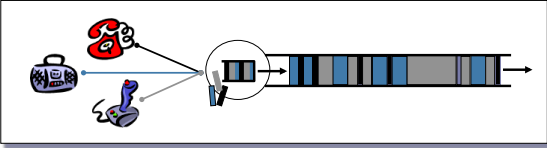
Transmission Control Protocol (TCP)

- ◆ Communication service (socket)
 - Ordered, reliable byte stream
 - Simultaneous transmission in both directions
- ◆ Key mechanisms at end hosts
 - Retransmit lost and corrupted packets
 - Discard duplicate packets and put packets in order
 - Flow control to avoid overloading the receiver buffer


But, what if too many hosts send at once?

Idea #4: Think globally, act locally

Congestion





- ◆ Too many hosts sending packets at once
 - Some packets have to wait in line
 - Eventually the queue runs out of space
 - And some packets gets dropped on the floor
- ◆ As Senator Stevens said: *And if you don't understand, those tubes can be filled. And if they are filled, when you put your message in, it gets in line and it's going to be delayed by anyone that puts into that tube enormous amounts of material, enormous amounts of material.*

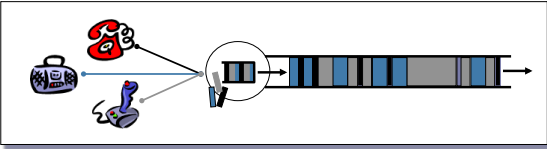


Sharing the Limited Resource


- ◆ Reserve resources
 - Room for ten phone calls
 - Block the 11th call
- ◆ Sub-divide resources
 - Tell the 11 transfers to each use 1/11 of the bandwidth
 - How?
- ◆ Local adaptation
 - Each transfer slows down
 - Voluntarily, for greater good

Congestion Control

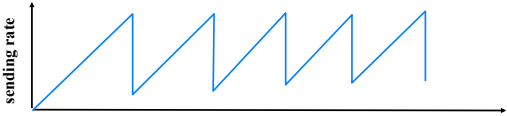



- ◆ What if too many folks are sending data?
 - Senders agree to slow down their sending rates
 - ... in response to their packets getting dropped
 - For the greater good



TCP Congestion Control

- ◆ Detecting congestion
 - My packet was lost
- ◆ Reacting to congestion
 - I voluntarily reduce my sending rate (by 2X)
- ◆ Testing the waters
 - I gradually increase my sending rate (linearly)

Transmission Control Protocol (TCP)

- ◆ Runs on the end host
 - Puts data into packets and sends them
- ◆ Congestion control
 - Speeds up and slows down
- ◆ Ordered reliable byte stream
 - Sender retransmits lost packets
 - Receiver discards corrupted packets
 - Receiver reorders out-of-order packets

Reliable service on an unreliable network



Why not TCP for Everything?

- ◆ Applications have different needs
 - Latency, bandwidth, reliability
- ◆ E.g. real-time speech or video cares more about timing than about getting all bytes
- ◆ So other transport protocols necessary
- ◆ Led to decoupling of TCP and IP
 - TCP: reliable ordered delivery
 - IP: basic datagram service, best-effort delivery
 - (UDP: application-level interface to basic datagram service)



Operating atop a variety of networks

- ◆ Internet operates over a variety of networks
 - Long-haul (X.25)
 - Local-area (ethernet, token rings)
 - Satellite
 - Packet radio
 - Serial links
- ◆ Key: makes very few assumptions about underlying network capabilities
 - Can transfer a packet
 - Can address (unless point-to-point)

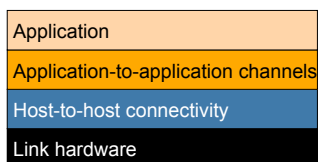


Key idea #5: Layering



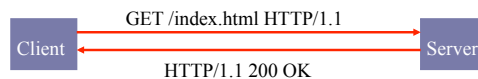
Layering: A Modular Approach

- ◆ Sub-divide the problem
 - Each layer relies on services from layer below
 - Each layer exports services to layer above
- ◆ Interface between layers defines interaction
 - Hides implementation details
 - Layers can change without disturbing other layers

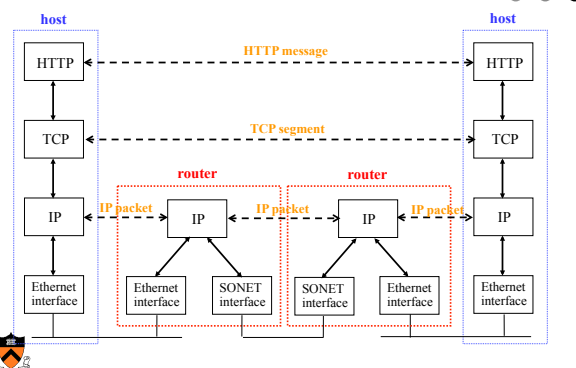


Application-Layer Protocols

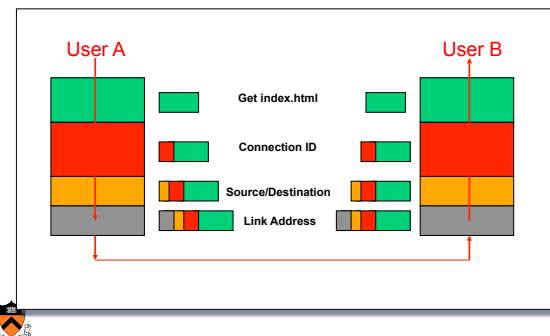
- ◆ Messages exchanged between applications
 - Syntax and semantics of the messages between hosts
 - Tailored to the specific application (e.g., Web, e-mail)
 - Messages transferred over transport connection (e.g., TCP)
- ◆ Popular application-layer protocols
 - Telnet, FTP, SMTP, NNTP, HTTP, BitTorrent, ...

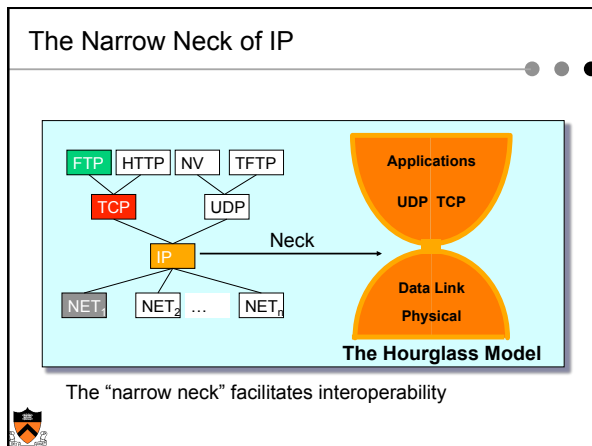
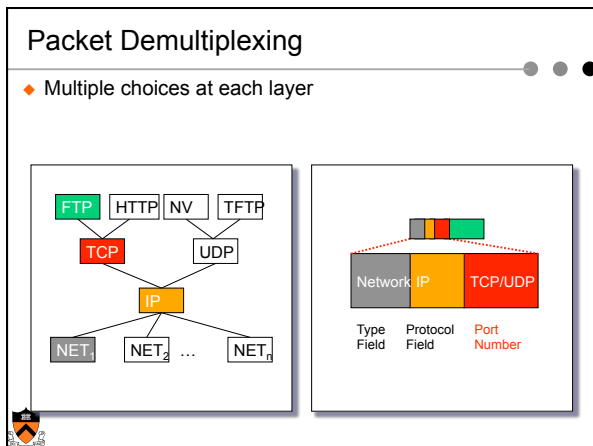


Layering in the Internet



Packet Encapsulation





● ● ● | Idea #6: A rose by any other name

- ### Separating Naming and Addressing
- ◆ Host names
 - Mnemonic name appreciated by humans
 - Variable length, alpha-numeric characters
 - Provide little (if any) information about location
 - Examples: www.cnn.com and ftp.eurocom.fr
 - ◆ IP addresses
 - Numerical address appreciated by routers
 - Fixed length, binary number
 - Hierarchical, related to host location
 - Examples: 64.236.16.20 and 193.30.227.161

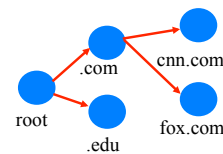
Separating Naming and Addressing

- ◆ Names are easier to remember
 - www.cnn.com vs. 64.236.16.20
- ◆ Addresses can change underneath
 - www.cnn.com needn't be at 64.236.16.20
- ◆ Name could map to multiple IP addresses
 - www.cnn.com to multiple replicas of the Web site
- ◆ Map to different addresses in different places
 - Address of a nearby copy of the Web site
 - E.g., to reduce latency, or return different content
- ◆ Multiple names for the same address
 - E.g., aliases like ee.mit.edu and cs.mit.edu

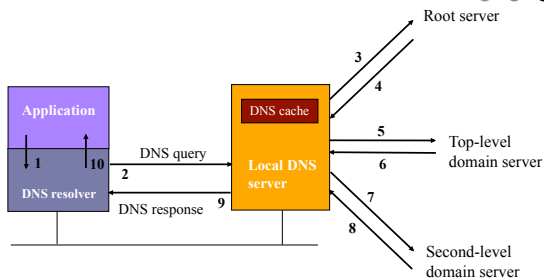


Domain Name System (DNS) Hierarchy

- ◆ Distributed “phone book”
 - Multiple queries to translate name to address
- ◆ Small number of “root servers”
 - Tell you where to look up “.com” names
- ◆ Larger number of “top-level domains”
 - Tell you where to look up “cnn.com” names



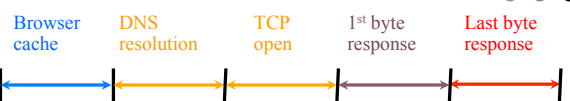
DNS Resolver and Local DNS Server



Caching to reduce latency in DNS translation.



Example: Many Steps in Web Download


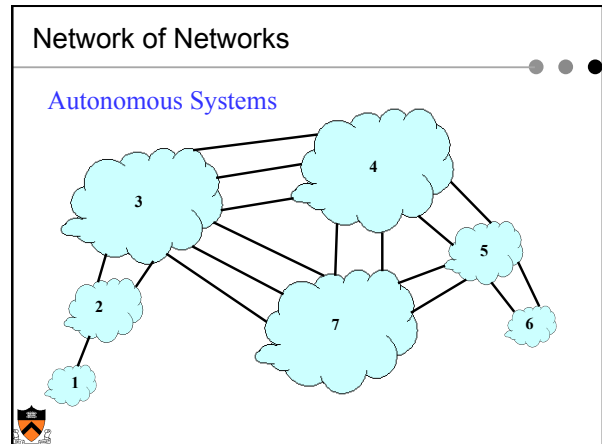


Sources of variability of delay

- Browser cache hit/miss, need for cache revalidation
- DNS cache hit/miss, multiple DNS servers, errors
- Packet loss, round-trip time, server accept queue
- RTT, busy server, CPU overhead (e.g., CGI script)
- Response size, receive buffer size, congestion
- ... downloading embedded image(s) on the page




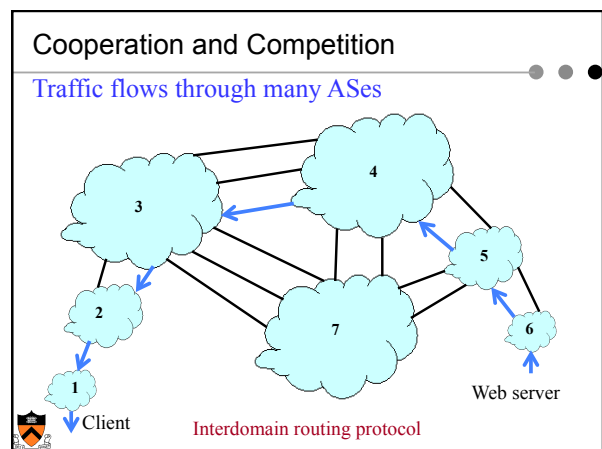
● ● ● | Idea #7: You scratch my back...

Autonomous Systems

Currently about 20,000 ASes.

- Level 3: #1
- MIT: #3
- Harvard: #11
- Yale: #29
- Princeton: #88
- AT&T: #7018, #6341, #5074, ...
- UUNET: #701, #702, #284, #12199, ...
- Sprint: #1239, #1240, #6211, #6242, ...
- ...

Business Relationships

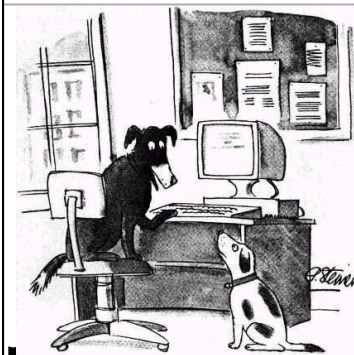
- ◆ Neighboring ASes have business contracts
 - How much traffic to carry
 - Which destinations to reach
 - How much money to pay
- ◆ Common business relationships
 - Customer-provider
 - E.g., Princeton is a customer of AT&T and USLEC
 - E.g., MIT is a customer of Level3
 - Peer-peer
 - E.g., AT&T is a peer of Sprint
 - E.g., Harvard is a peer of Harvard Business School



Problems With the Internet: Cheaters do win



No Strict Notions of Identity



- ◆ Leads to
 - Spam
 - Spoofing
 - Denial-of-service



Nobody in Charge

- ◆ Traffic traverses many Autonomous Systems
 - Who's fault is it when things go wrong?
 - How do you upgrade functionality?
- ◆ Implicit trust in the end host
 - What if some hosts violate congestion control?
- ◆ Anyone can add any application
 - Whether or not it is legal, moral, good, etc.
- ◆ Nobody knows how big the Internet is
 - No global registry of the topology
- ◆ Spans many countries
 - So no government can be in charge



The Internet of the Future

- ◆ Can we fix what ails the Internet
 - Security
 - Performance
 - Upgradability
 - Managability
 - <your favorite gripe here>
- ◆ Without throwing out the baby with bathwater
 - Ease of adding new hosts
 - Ease of adding new services
 - Ease of adding new link technologies
- ◆ An open technical and policy question...

