EE 542
Lecture 3: Internet Protocols
Internet and Cloud Computing

Young Cho
Department of Electrical Engineering
University of Southern California

Announcements

- Office Hours
  - At the end of the Lecture outside
- Lab/Project Team
  - 3 Student Team
- Tools
  - Sign up for AWS Account
  - Purchase xDot(s)
    - https://www.arrow.com/en/products/mtxdot-na1-a01-100/multi-tech-systems
    - JTAG programmer

[redacted] Fri, Aug 10, 10:33 AM (10 days ago)

to Young

Hello Professor,

My name is [redacted] and I had taken the CS558L class with you last fall. I have been interning at Sony PlayStation with the PlayStation Now team for the summer(referred by you) and will also be continuing during the fall. I wanted to tell you a little about the work I have been doing this summer. I can also send a detailed report on the same by next week.
I have been interning with the Core Engineering team at PlayStation Now which is responsible for maintaining the backbone infrastructure for the streaming service ie they have their own reliable communication protocol which runs on top of UDP along with their own Congestion Control Algorithm which helps with efficient data delivery.
I have been working on improving their congestion control protocol and wrote a Version 2 of it which addresses specific important changes as to how it reacts when the Client side network bandwidth changes along with better response to loss/delay conditions on the network as well as increased the speed of recovery when it falls low making sure it doesn't cause any corruption during the fast increase . Additionally they employ a FEC methodology to check for available bandwidth on the network which I had to modify a bit for the Congestion Control Algorithm.

This was just a brief summary but I could send you the entire report by next week. I also wanted to mention that now that I have started working in the industry, it's amazing to see the startling resemblance with the type of projects we had done as part of our academic coursework with real world applications . I would never have imagined that the world's biggest companies are working with the kind of technology we had played with in class and I am very grateful to you for providing us with such an enriching experience.

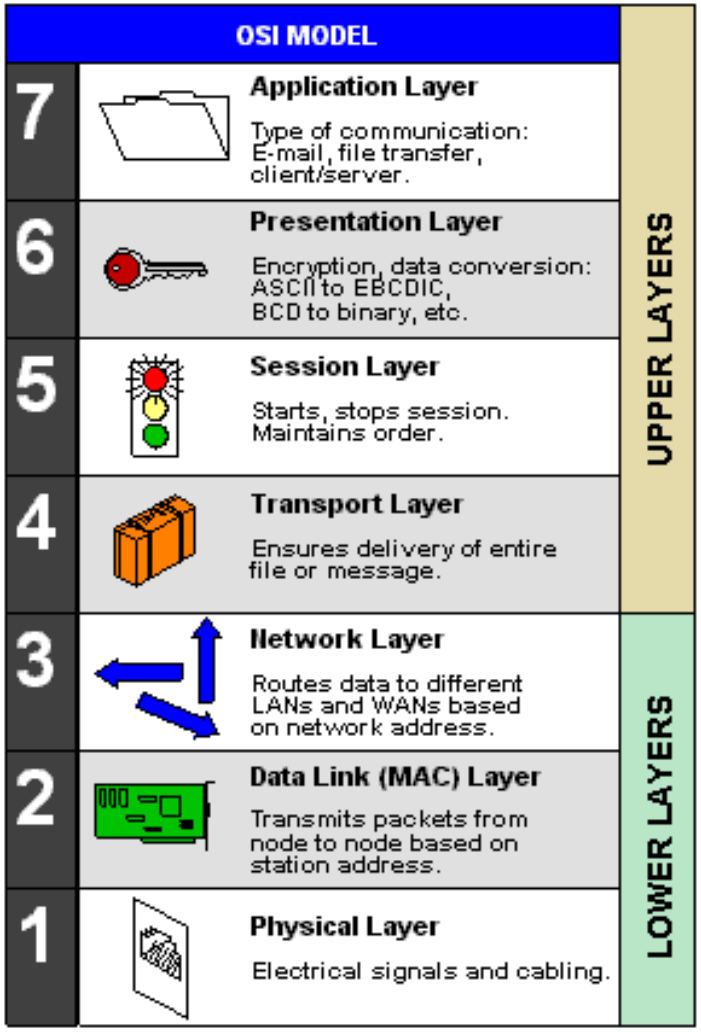
Additionally, I would like to know if you could supervise my Fall internship CPT in the same company.
Looking forward to your reply.

Thank You once again for everything :)

•••

[redacted]

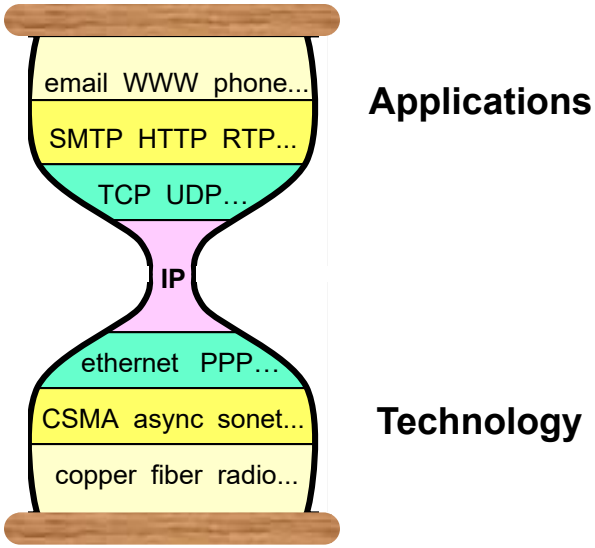
Open System Interconnection Reference Model



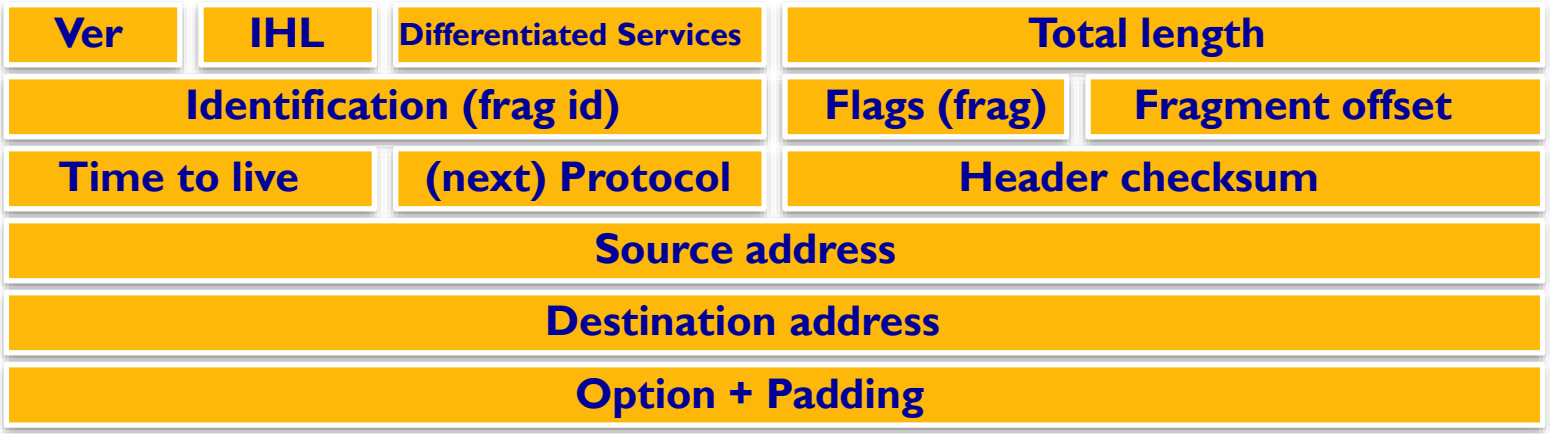
From computer desktop encyclopedia © 2004

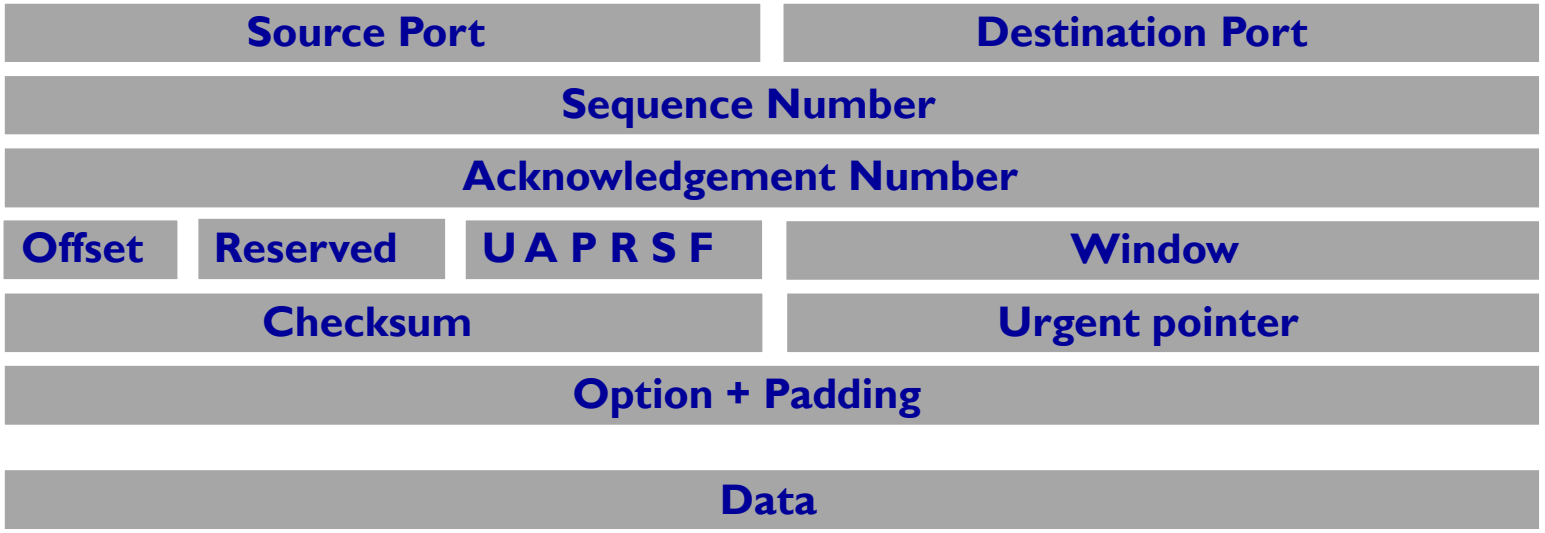
Internet Protocol (Layer 3)

- Interconnect many networks
- Hide underlying technology
  - IP is the "compatibility layer"
- Hourglass architecture
  - All hosts and routers run IP
  - Stateless architecture
- Trade-off
  - No assumptions
  - No guarantees



IPv4 Packet Structure





Fields of the IP Header

- **Time To Live (TTL) (1 byte):**
  - Specifies longest paths before datagram is dropped
  - Role of TTL field: Ensure that packet is eventually dropped when a routing loop occurs

  Used as follows:
  - Sender sets the value (e.g., 64)
  - Each router decrements the value by 1
  - When the value reaches 0, the datagram is dropped

Fragmentation

- IP packets can be 64KB
  - Different link-layers have different MTUs
  - Split IP packet into multiple fragments
    - IP header on each fragment
    - Various fields in header to help process
    - Intermediate router may fragment as needed
- Where to do reassembly?
  - End nodes – avoids unnecessary work
  - Dangerous to do at intermediate nodes
    - Buffer space
    - Multiple paths through network

Fragmentation Disadvantages

- Uses resources poorly
  - Forwarding costs per packet
  - Best if we can send large chunks of data
  - Worst case: packet just bigger than MTU
- Poor end-to-end performance
  - Loss of a fragment
- Reassembly is hard
  - Buffering constraints

IP Address Problem (1991)

- Address space depletion
  - In danger of running out of classes A and B
- Why?
  - Class C too small for most domains
  - Very few class A – IANA (Internet Assigned Numbers Authority) very careful about giving
  - Class B – greatest problem
    - Sparsely populated – but people refuse to give it back

Mitigation Efforts

- Classless Inter-Domain Routing (CIDR)
- Network address translation (NAT)
- Use of private network addressing
- Name-based virtual hosting of web sites
- Tighter control by regional Internet registries on the allocation of addresses to local Internet registries
- Network renumbering and subnetting to reclaim large blocks of address space allocated in the early days of the Internet, when the Internet used inefficient classful network addressing

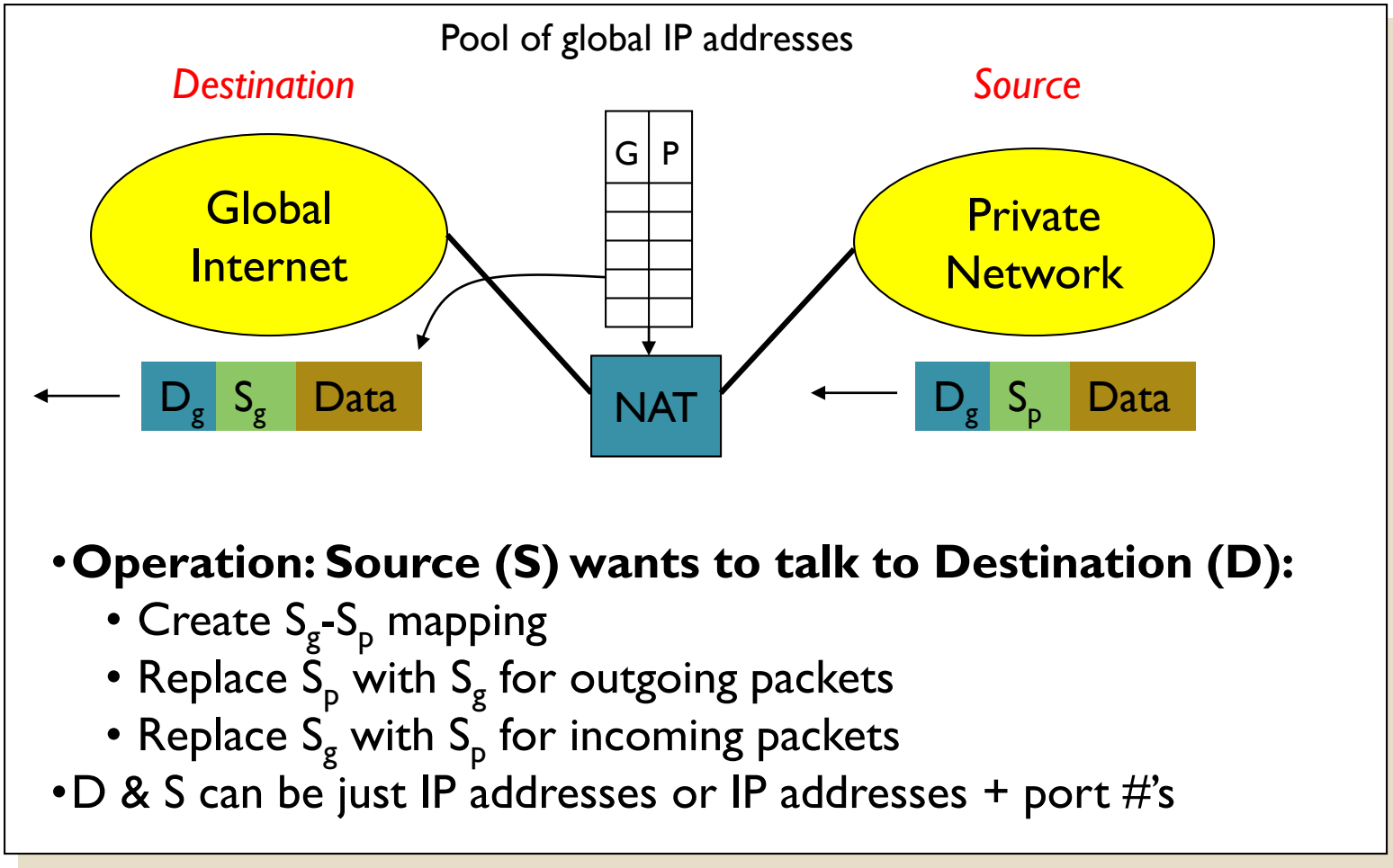
Classless Inter-Domain Routing

- Do not use classes to determine network ID
- Assign any range of addresses to network
  - Use common part of address as network number
  - e.g., addresses 192.4.16 - 196.4.31 have the first 20 bits in common. Thus, we use this as the network number
  - netmask is /20, /xx is valid for almost any xx
- Enables more efficient usage of address space (and router tables)

NAT

- Network Address Translation (NAT)
- Alternate solution to address space
  - Kludge (but useful)
- Sits between your network and the Internet
- Translates local network layer addresses to global IP addresses
- Has a pool of global IP addresses (less than number of hosts on your network)

NAT Illustration

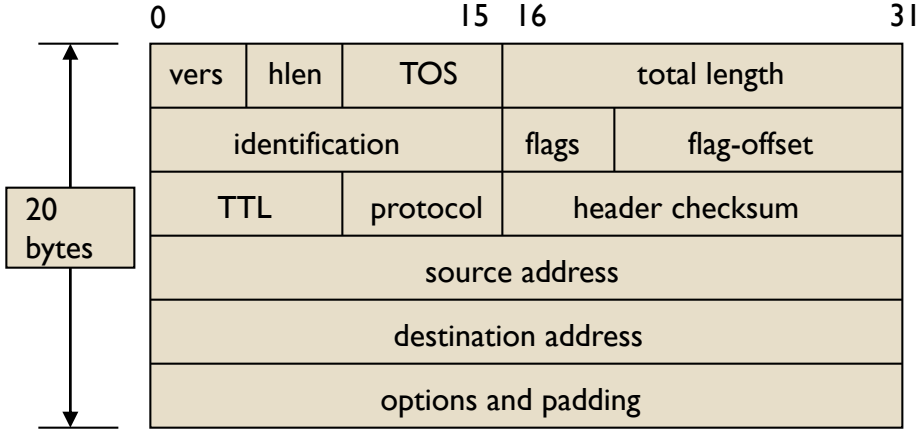


- **Operation: Source (S) wants to talk to Destination (D):**
  - Create S_g-S_p mapping
  - Replace S_p with S_g for outgoing packets
  - Replace S_g with S_p for incoming packets
- D & S can be just IP addresses or IP addresses + port #'s

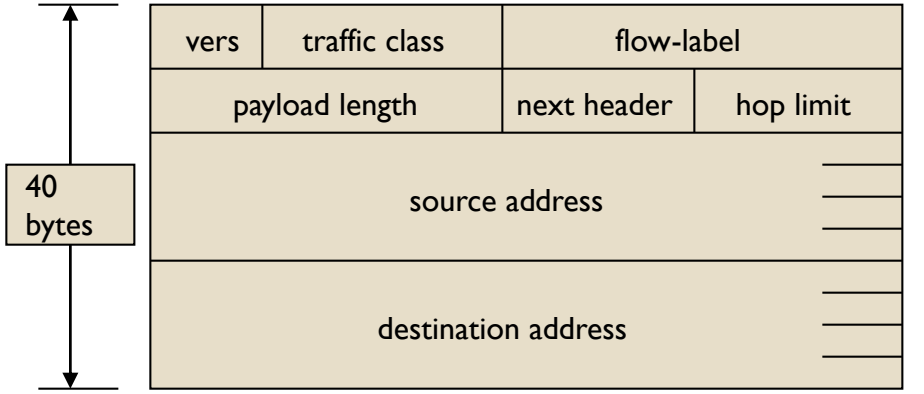
IPv6

- Larger Address Space
- Aggregation-based address hierarchy
  - Efficient backbone routing
- Efficient and Extensible IP datagram
- Stateless Address Autoconfiguration
- Security (IPsec mandatory)
- Mobility

Header comparison



IPv4



IPv6

Removed (6)

- ID, flags, flag offset
- TOS, hlen
- header checksum

Changed (3)

- total length => payload
- protocol => next header
- TTL => hop limit

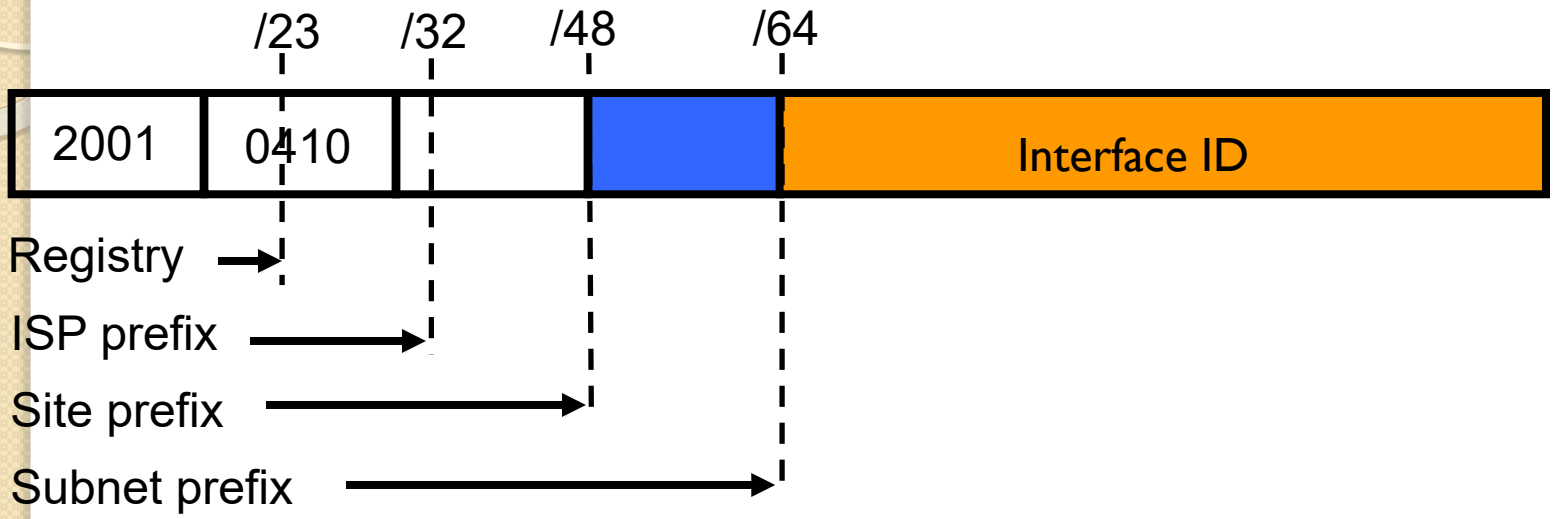
Added (2)

- traffic class
- flow label

Expanded

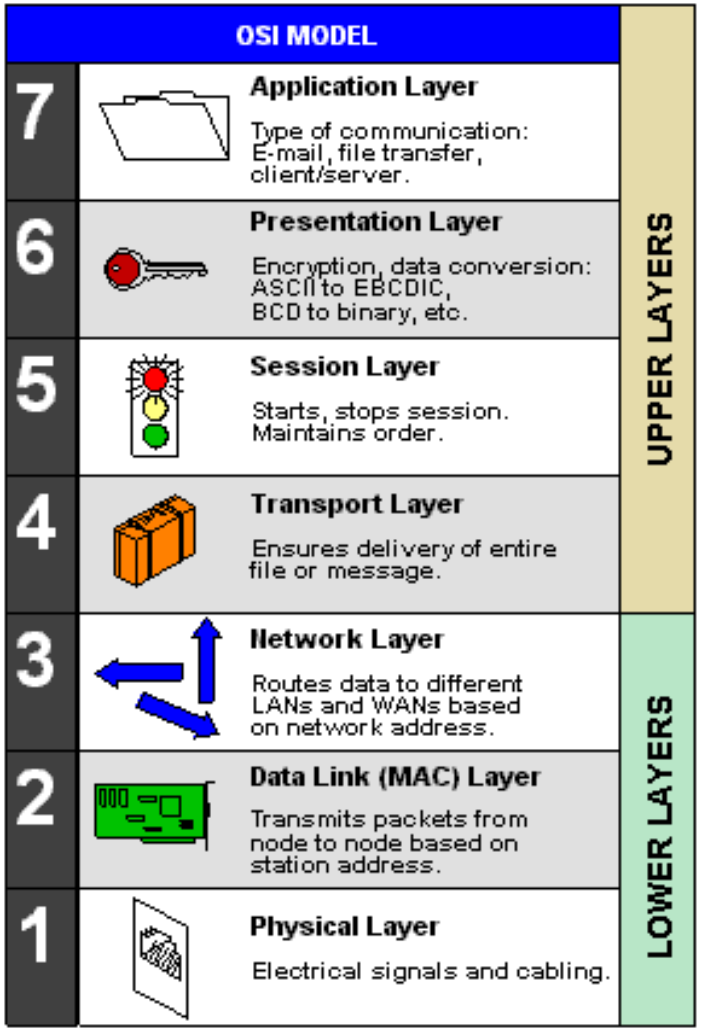
- address 32 to 128 bits

Address Allocation



- The allocation process was recently updated by the registries:
  - International Assigned Numbers Authority (IANA) allocates from 2001::/16 to regional registries
  - Each regional registry allocation is a ::/23
  - ISP allocations from the regional registry is a ::/36 (immediate allocation) or ::/32 (initial allocation) or shorter with justification
  - Policy expectation that an ISP allocates a ::/48 prefix to each customer

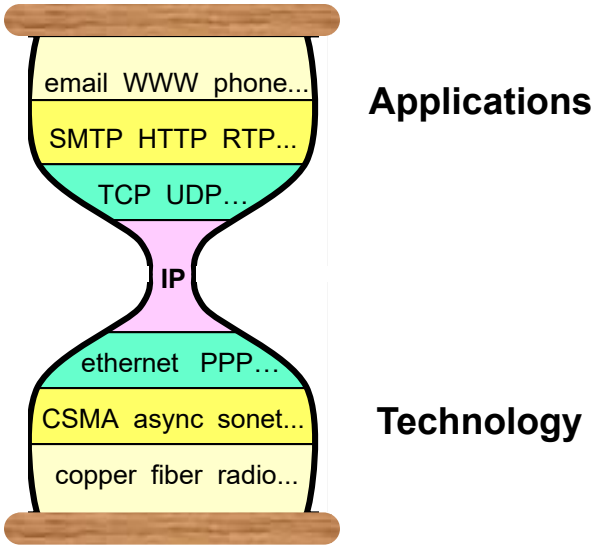
Open System Interconnection Reference Model



From computer desktop encyclopedia © 2004

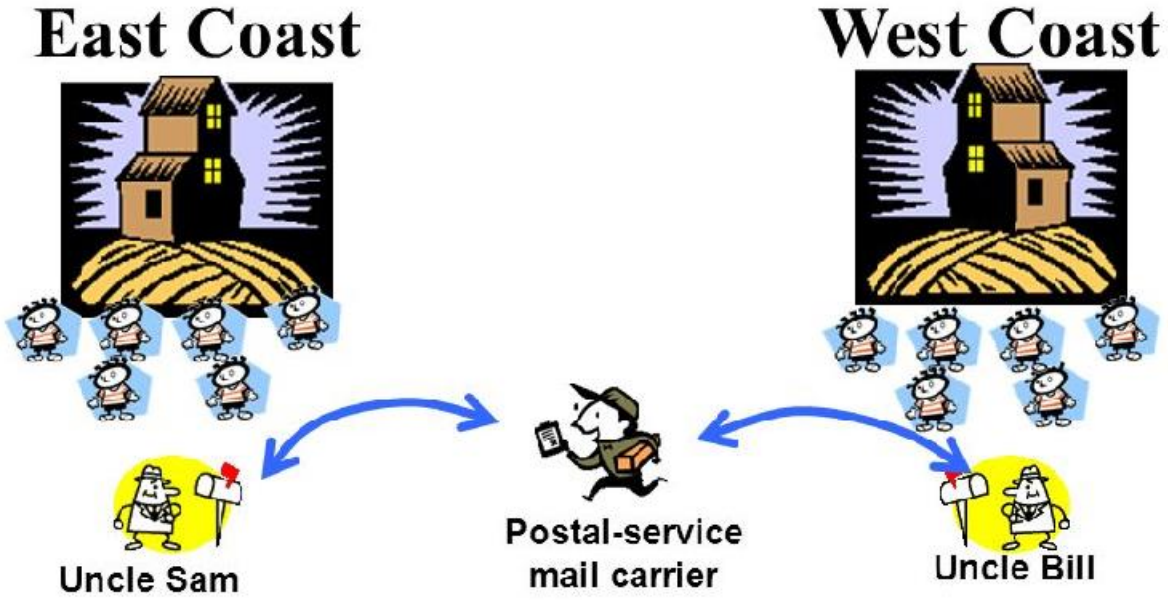
TCP and UDP (Layer 4)

- Network layer
  - Logical communication between hosts
  - IP Addresses
- Transport layer
  - Logical communication between processes
  - Relies on, enhances, network layer services
  - Port numbers



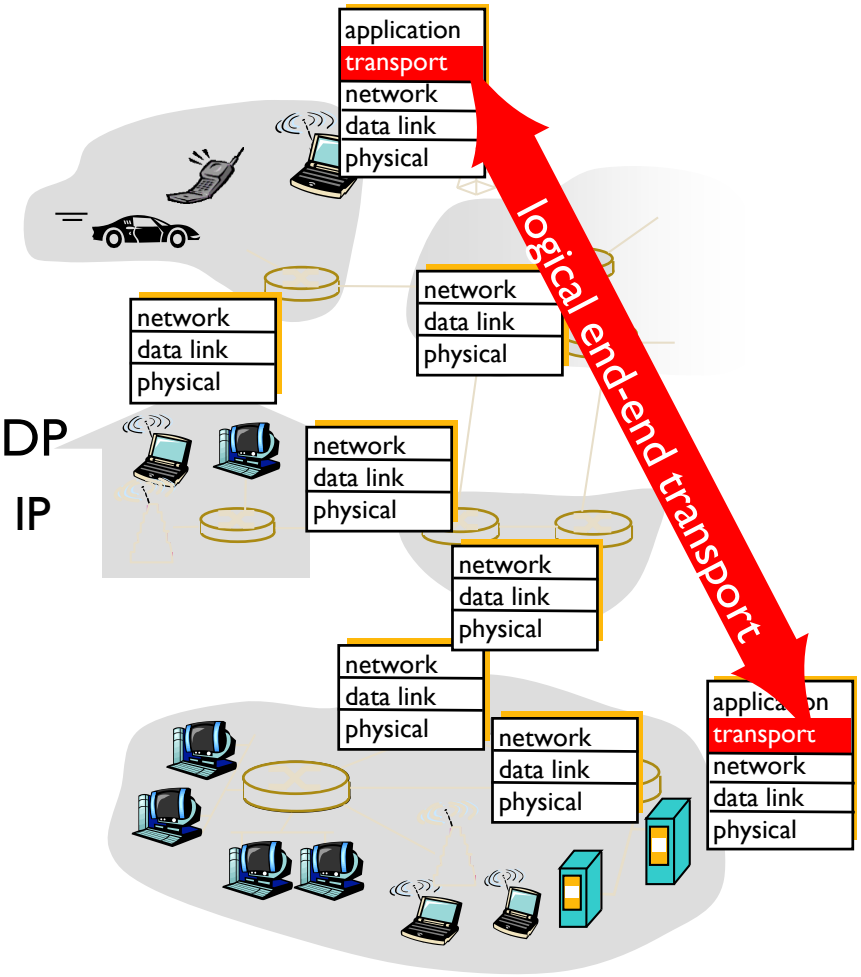
Analogy: Postal Service

- Hosts = Houses
- Processes = Persons in the House
- Data = Letters
- Transport Layer Protocol = Distribution by Uncles
- Network Layer Protocol = Postal Service



Internet transport-layer protocols

- Reliable, in-order delivery (TCP)
  - congestion control
  - flow control
  - connection setup
- Unreliable, unordered delivery: UDP
  - no-frills extension of "best-effort" IP
- Services not available:
  - delay guarantees
  - bandwidth guarantees



UDP: User Datagram Protocol [RFC 768]

- "no frills," "bare bones" Internet transport protocol
- "best effort" service, UDP segments may be:
  - lost
  - delivered out of order to app
- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

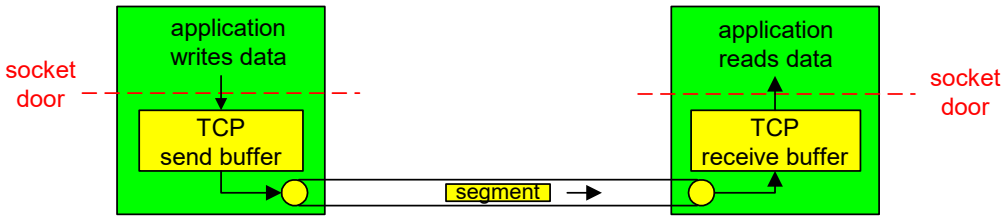
Why is there a UDP?

- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small segment header
- no congestion control: UDP can blast away as fast as desired

TCP: Overview

RFCs: 793, 1122, 1323, 2018, 2581

- full duplex data:
  - bi-directional data flow in same connection
  - MSS: maximum segment size
- connection-oriented:
  - handshaking (exchange of control msgs) init's sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver
- point-to-point:
  - one sender, one receiver
- reliable, in-order *byte steam:*
  - no "message boundaries"
- pipelined:
  - TCP congestion and flow control set window size
- *send & receive buffers*



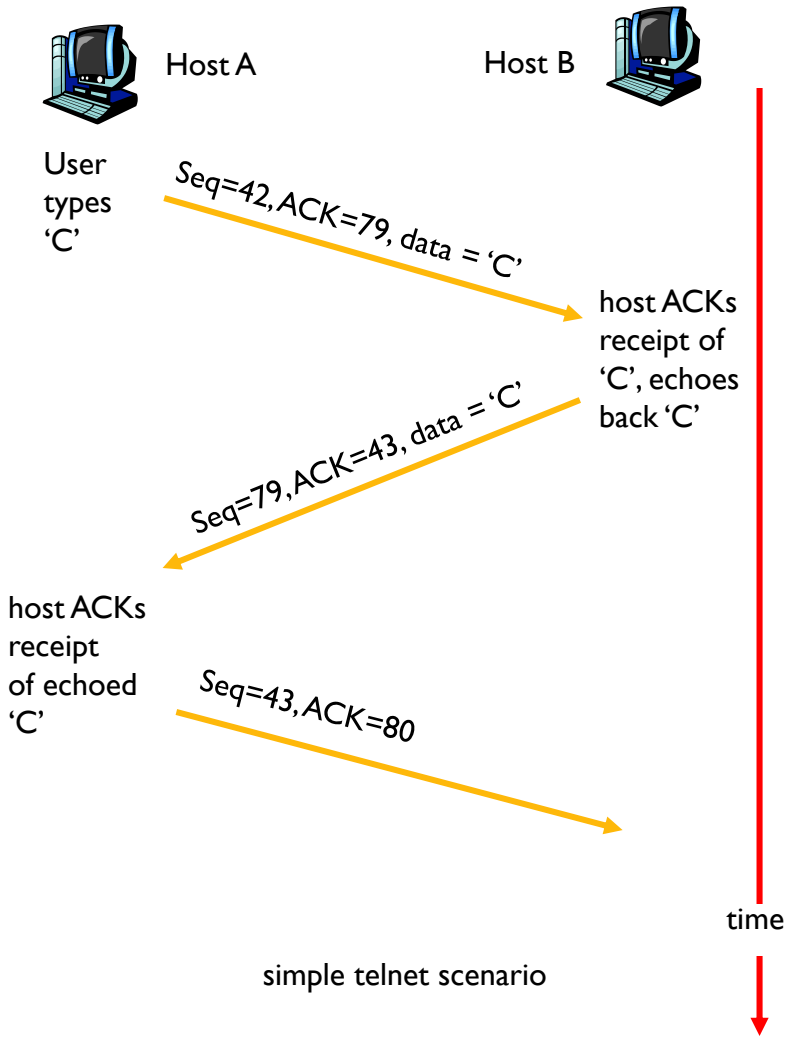
TCP seq. #'s and ACKs

Seq. #'s:

- byte stream "number" of first byte in segment's data

ACKs:

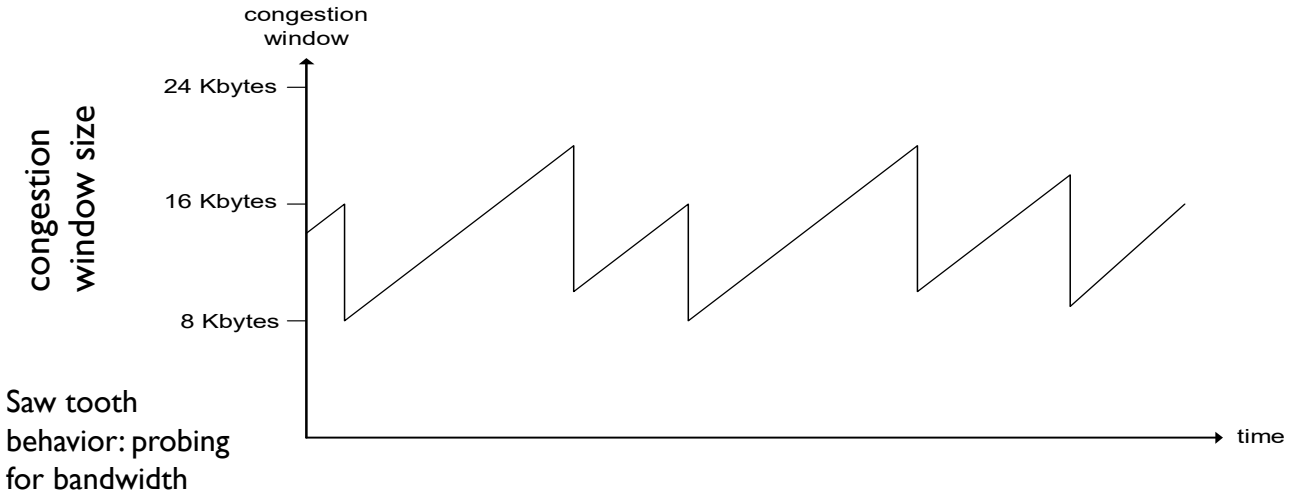
- seq # of next byte expected from other side
- cumulative ACK



simple telnet scenario

Problem: TCP congestion control

- *Approach:* increase transmission rate (congestion window size), probing for usable bandwidth, until loss occurs
  - *additive increase:* increase **CongWin** by 1 MSS every RTT until loss detected
  - *multiplicative decrease:* cut **CongWin** in half after loss



Lab 2

- In essence, Lab 1 on AWS
- Report due on coming Saturday 11:59PM
- Demo Video on Youtube on the following Monday 11:59PM
- The Goal of these submission is to prove to me that you did the lab assignment
- So, use screen video captures and your explanations to convince me that you did the work