Transport Layer

- Service Models
- TCP vs UDP
- TCP Connections
- Flow Control and Sliding Window
- TCP Congestion Control
- Newer TCP Implementations
Service Models

- Transport Layer Services
  - Datagrams (UDP): Unreliable Messages
  - Streams (TCP): Reliable Bytestreams

- Socket API: simple abstraction to use the network
  - Port: Identify different applications / application layer protocols on a host
# TCP vs UDP

<table>
<thead>
<tr>
<th>TCP (Streams)</th>
<th>UDP (Datagrams)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connections</td>
<td>Datagrams</td>
</tr>
<tr>
<td>Bytes are delivered once, reliably, and in order</td>
<td>Messages may be lost, reordered, duplicated</td>
</tr>
<tr>
<td>Arbitrary length content</td>
<td>Limited message size</td>
</tr>
<tr>
<td>Flow control matches sender to receiver</td>
<td>Can send regardless of receiver state</td>
</tr>
<tr>
<td>Congestion control matches sender to network</td>
<td>Can send regardless of network state</td>
</tr>
</tbody>
</table>
TCP Connection Establishment

Three-way handshake

Active party (client)  Passive party (server)

CLOSED  
SYN_SENT  
ESTABLISHED  

1 SYN (SEQ=x)
2 SYN (SEQ=y, ACK=x+1)
3 (SEQ=x+1, ACK=y+1)

CSE 461 University of Washington
TCP Connection Release

States...

**Active party**

- ESTABLISHED
- FIN_WAIT_1
- FIN_WAIT_2
- TIME_WAIT (timeout)
- CLOSED

**Passive party**

- ESTABLISHED
- CLOSE_WAIT
- LAST_ACK
- CLOSED

**Messages**

1. FIN (SEQ=x)
   - (SEQ=y, ACK=x+1)

2. FIN (SEQ=y, ACK=x+1)
   - (SEQ=x+1, ACK=y+1)
Flow Control - Sliding Window Protocol

- Instead of stop-and-wait, sends $W$ packets per 1 RTT
  - To fill network path, $W=2BD$ (make sure ans in packets!!!)

- **Shortest bandwidth determines rate**

- Receiver sends ACK upon receiving packets
  - Go-Back-N (similar to project 1 stage b): not efficient
  - **Selective Repeat**
    - Receiver passes data to app in order, and buffers out-of-order segments to reduce retransmissions
    - ACK conveys highest in-order segment
      - As well as hints about out-of-order segments

- **Selective Retransmission** on sender’s side
Flow Control - ACK Clock
Flow Control - Sliding Window Protocol (2)

● Flow control on receiver’s side
  ○ In order to avoid loss caused by user application not calling recv(), receiver tells sender its available buffer space (WIN)
  ○ Sender uses lower of the WIN and $W$ as the effective window size

● How to set a **timeout** for retransmission on sender’s side?
  ○ Adaptively determine timeout value based on smoothed estimate of RTT
Max-Min Fair Allocation

- Start with all flows at rate 0
- Increase the flows until there is a new bottleneck in the network
- Hold fixed the rate of the flows that are bottlenecked
- Go to step 2 for any remaining flows
TCP Bandwidth Allocation

● Closed loop: use feedback to adjust rates
  ○ NOT open loop: reserve bandwidth before use

● Host driven: host sets/enforces allocations
  ○ NOT network driven

● Window based
  ○ NOT rate based

● Congestion signal
  ○ Packet loss, Packet delay, Router indication
AIMD - Additive Increase Multiplicative Decrease
AIMD

- **Slow-Start** (used in AI)
  - Double cwnd until packet timeout
  - Restart and double until cwnd/2, then AI

- **Fast-Retransmit** (used in MD)
  - Three duplicate ACKs = packet loss
  - Don’t have to wait for TIMEOUT

- **Fast-Recovery** (used in MD)
  - MD after fast-retransmit
  - Then pretend further duplicate ACKs are the expected ACKs
TCP Reno

TCP sawtooth

ACK clock running

MD of ½, no slow-start
TCP CUBIC

- Problem with standard TCP?
  - Flows with lower RTT’s “grow” faster than those with higher RTTs
  - Flows grow too “slowly” (linearly) after congestion

TCP BBR

- **Bufferbloat Problem**
  - performance can decrease when buffer size is increased
- **Model based** instead of loss based
  - Measure RTT, latency, bottleneck bandwidth
  - Use this to predict window size
Familiarize yourself with different protocols

- ICMP - Network Layer
- DHCP - Application layer
- OSPF - Network layer
- DCTCP - Internet layer
- Other protocols that are mentioned in lecture.
Network Components

- Parts of a Network
- Types of Links
- Protocols and Layers
- Encapsulation
- Demultiplexing
Parts of a Network

- Parts of a Network

- Types of Links

Simplex

Full-duplex

Half-duplex
## Protocols and Layers

<table>
<thead>
<tr>
<th>Layer</th>
<th>Purpose</th>
<th>Protocols</th>
<th>Unit of Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application</strong></td>
<td>Programs that use network service</td>
<td>HTTP, DNS</td>
<td>Message</td>
</tr>
<tr>
<td><strong>Transport</strong></td>
<td>Provides end-to-end data delivery</td>
<td>TCP, UDP</td>
<td>Segment</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>Sends packets across multiple networks</td>
<td>IP</td>
<td>Packet</td>
</tr>
<tr>
<td><strong>Link</strong></td>
<td>Sends frames across a link</td>
<td>Ethernet, Cable</td>
<td>Frame</td>
</tr>
<tr>
<td><strong>Physical</strong></td>
<td>Transmit bits</td>
<td>__</td>
<td>Bit</td>
</tr>
</tbody>
</table>
Protocols and Layers

ADVANTAGES

● Use information hiding to connect different systems
● Information reuse to build new protocols

DISADVANTAGES

● Adds overhead
● Hides information
Encapsulation
Motivation

- What does the network layer do?
  - Connect different networks (send packets over multiple networks)

- Why do we need the network layer?
  - Switches don’t scale to large networks
  - Switches don’t work across more than one link layer technology
  - Switches don’t give much traffic control
Network Service Models

**Datagram Model**
- Connectionless service
- Packets contain destination address
- Routers look up address in its forwarding table to determine next hop
- Example: IP

**Virtual Circuits**
- Connection-oriented service
- Connection establishment → data transfer → connection teardown
- Packets contain label for circuit
- Router looks up circuit in forwarding table to determine next hop
- Example: MPLS

Both of them use **Store-and-Forward packet switching**
Internetworking - IP

- How do we connect different networks together?
- IP - Internet Protocol
- Lowest Common Denominator
  - Asks little of lower-layer networks
  - Gives little as a higher layer

![Diagram of IP stack and packet structure]
IP Addresses Prefix and Forwarding

- IP prefix \texttt{a.b.c.d/}\texttt{L}
  - Represents addresses that have the same first \texttt{L} bits
  - e.g. 128.13.0.0/16 \to all 65536 addresses between 128.13.0.0 to 128.13.255.255
  - e.g. 18.31.0.0/32 \to 18.31.0.0 (only one address)

- Longest Matching Prefix
  - find the longest prefix that contains the destination address, i.e., the most specific entry
DHCP - Dynamic Host Configuration Protocol

- Bootstrapping problem
- Leases IP address to nodes
- UDP

Also setup other parameters:
- DNS server
- IP address of local router
- Network prefix
ARP - Address Resolution Protocol

- MAC is needed to send a frame over the local link
- ARP to map an IP to MAC
- Sits on top of link layer
ICMP - Internet Control Message Protocol

- Provides error reporting and testing
- Companion protocol to IP
- Traceroute, Ping
NAT - Network Address Translation

- One solution to **IPv4 address exhaustion**
- Map many private IP to one public IP, with different port number
- Pros: useful functionality (firewall), easy to deploy, etc.
- Cons: Connectivity has been broken!
- Many other cons...

<table>
<thead>
<tr>
<th>Internal IP:port</th>
<th>External IP : port</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.12 : 5523</td>
<td>44.25.80.3 : 1500</td>
</tr>
<tr>
<td>192.168.1.13 : 1234</td>
<td>44.25.80.3 : 1501</td>
</tr>
<tr>
<td>192.168.2.20 : 1234</td>
<td>44.25.80.3 : 1502</td>
</tr>
</tbody>
</table>

![Diagram showing NAT process]
IPv6

- A much better solution to IPv4 address exhaustion
- Uses 128-bit addresses, with lots of other changes
- IPv6 version protocols: NDP -> ARP, SLAAC -> DHCP
- Problem: being incompatible with IPV4. Solution: Tunnelling

What's my IP

2601:602:8b00:5f0:30b3:2d19:3fe:db9e
Your public IP address
Routing

- The process of deciding in which direction to send traffic
- Delivery models: unicast, broadcast, multicast, anycast
- Goals: correctness, efficient paths, fair paths, fast convergence, scalability
- Rules: decentralized, distributed setting
Techniques to Scale Routing

Hierarchical Routing
- Route first to the region, then to the IP prefix within the region

IP Prefix Aggregation and Subnets
- Adjusting the size of IP prefixes
  - Internally split one large prefix
  - Externally join multiple IP prefixes
Best Path Routing

**Distance Vector Routing**
Each node maintains a vector of distances (and next hops) to all destinations.

Sometimes doesn’t perform very well: count-to-infinity scenario

**Link State Routing** (widely used)
Phase 1. **Topology Dissemination**: Nodes flood topology

Phase 2. **Route Computation**: running Dijkstra algorithm (or equivalent)

Algorithm details available in lecture slides
BGP - Border Gateway Protocol

- Internet-wide routing between ISPs (ASes)
  - Each has their own policy decisions
- Peer and Transit (Customer) relationship
- Border routers of ISPs announce BGP routes only to other parties who may use those paths.
- Border routers of ISPs select the best path of the ones they hear in any, non-shortest way
BGP example

- **Transit (ISP & Customer)**
  - ISP announce everything it can reach to its customer
    - AS1 to AS2: you can send packet to AS4 through me
  - Customer ISP only announce its customers to ISP
    - AS2 to AS1: you can send packet to A through me

- **Peer (ISP 1 & ISP 2)**
  - ISP 1 only announces its customer to ISP 2
    - AS2 to AS3: you can send packet to A through me
ISP4 to ISP5?
Customer  Provider

ISP4 - ISP1 - ISP3 - ISP5
TCP window size?

Packet is 10 bytes

B = 40 bytes/sec  
D = .5

B = 80 bytes/sec  
D = 2
TCP window size?

Packet is 10 bytes

(40(2+.5)*2) / 10 = 20 packet window size