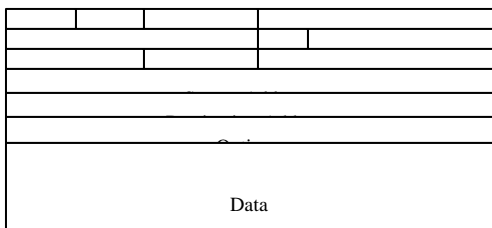

TCP/IP

Emin Gun Sirer

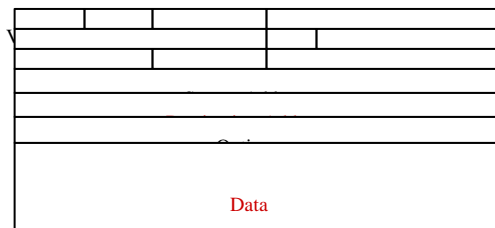
IP

- **Internetworking protocol**
 - Network layer
- **Common packet format for the Internet**
 - Specifies what packets look like
 - *Fragments* long packets into shorter packets
 - *Reassembles* fragments into original shape
- **Some parts are fundamental, and some are arbitrary**
 - IPv4 is what most people use
 - IPv6 clears up some of the messy parts, but is not yet in wide use

IPv4 packet layout



IPv4 packet layout



IP Fragmentation

- Networks have different maximum packet sizes
 - Big packets are sometimes desirable – less overhead
 - Huge packets are not desirable – reduced response time for others
- Higher level protocols (e.g. TCP or UDP) could figure out the max transfer unit and chop data into smaller packets
 - The endpoints do not necessarily know what the MTU is on the path
 - The route can change underneath
- Consequently, IP transparently fragments and reassembles packets

IP Fragmentation Mechanics

- IP divides a long datagram into N smaller datagrams
- Copies the header
- Assigns a Fragment ID to each part
- Sets the More Fragments bit
- Receiving end puts the fragments together based on the new IP headers
- Throws out fragments after a certain amount of time if they have not been reassembled

IP Options

- Source Routing: The source specifies the set of hosts that the packet should traverse
- Record Route: If this option appears in a packet, every router along a path attaches its own IP address to the packet
- Timestamp: Every router along the route attaches a timestamp to the packet
- Security: Packets are marked with user info, and the security classification of the person on whose behalf they travel on the network
 - Most of these options pose security holes and are generally not implemented

UDP

- Unreliable Datagram Protocol
- IP goes from host to host
- We need a way to get datagrams from one application to another
- How do we identify applications on the hosts?
 - Assign *port numbers*
 - E.g. port 13 belongs to the time service

UDP Packet Layout

| | | | | | |
|-----|---------------------|----------|------------------|-----------------|--|
| IP | Version | IHL | TOS | Total Length | |
| | Identification | | Flags | Fragment Offset | |
| | TTI | Protocol | Header Checksum | | |
| | Source Address | | | | |
| UDP | Destination Address | | | | |
| | Source Port | | Destination Port | | |
| | Length | | Checksum | | |
| | Data | | | | |

- UDP adds Ports, Data Length and Data checksum

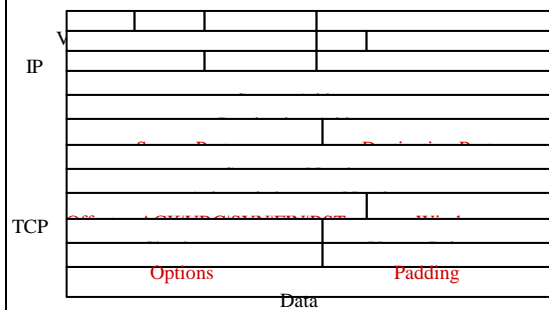
UDP

- UDP is unreliable
 - A UDP packet may get dropped at any time
 - It may get duplicated
 - A series of UDP packets may get reordered
- Applications need to deal with reordering, duplicate suppression, reliable delivery
 - Some apps can ignore these effects and still function
- Unreliable datagrams are the bare-bones network service
 - Good to build on, esp for multimedia applications

TCP

- Transmission Control Protocol
 - Reliable, ordered communication
- Enough applications demand reliable ordered delivery that they should not have to implement their own protocol
- A standard, adaptive protocol that delivers good-enough performance and deals well with congestion
- All web traffic travels over TCP/IP

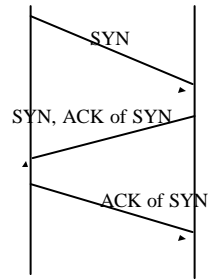
TCP/IP Packets



TCP Packets

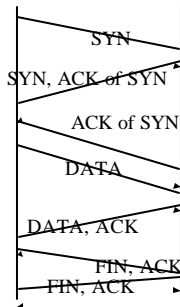
- Each packet carries a unique ID
 - The initial number is chosen randomly
 - The ID is incremented by the data length
- Each packet carries an acknowledgement
 - Can acknowledge a set of packets by ack'ing the latest one received
- Reliable transport is implemented using these identifiers

TCP Connections



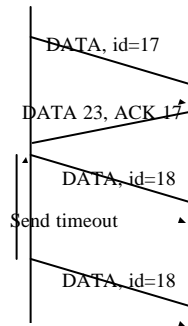
- TCP is *connection* oriented
- A connection is initiated with a three-way handshake
- Three-way handshake ensures against duplicate SYN packets
- Takes 3 packets, 1.5 RTT

Typical TCP Usage



- Three round-trips to set up a connection, send a data packet, receive a response, tear down connection
- FINs work (mostly) like SYNs to tear down connection
 - Need to wait after a FIN for stragglng packets

Reliable transport



- TCP keeps a copy of all sent, but unacknowledged packets
- If acknowledgement does not arrive within a “send timeout” period, packet is resent
- Send timeout adjusts to the round-trip delay

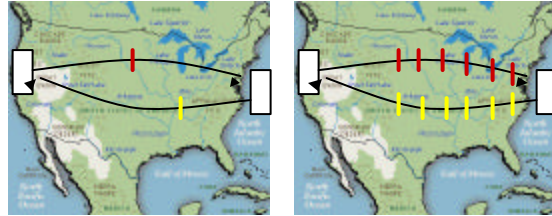
TCP timeouts

- What is a good timeout period ?
 - Want to improve throughput without unnecessary transmissions

$\text{NewAverageRTT} = (1 - \alpha) \text{OldAverageRTT} + \alpha \text{LatestRTT}$
 $\text{NewAverageDev} = (1 - \alpha) \text{OldAverageDev} + \alpha \text{LatestDev}$
 where $\text{LatestRTT} = (\text{ack_receive_time} - \text{send_time})$,
 $\text{LatestDev} = |\text{LatestRTT} - \text{AverageRTT}|$,
 $\alpha = 1/8$, typically.
 $\text{Timeout} = \text{AverageRTT} + 4 * \text{AverageDev}$

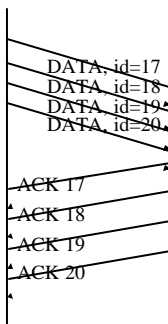
- Timeout is thus a function of RTT and deviation

TCP Windows



- Multiple outstanding packets can increase throughput

TCP Windows

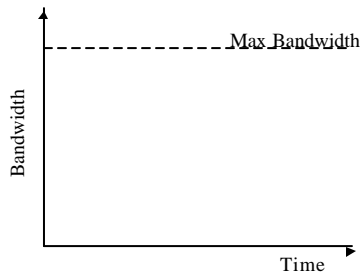


- Can have more than one packet in transit
- Especially over fat pipes, e.g. satellite connection
- Need to keep track of all packets within the window
- Need to adjust window size

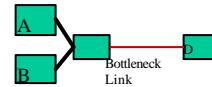
TCP Congestion Control

- TCP Increases its window size as long as no packets are dropped
- It halves the window size when a packet drop occurs
 - A packet drop is evident from the acknowledgements
- Therefore, it will slowly build up to the max bandwidth, and hover around the max
 - It doesn't achieve the max possible though
 - Instead, it shares the bandwidth well with other TCP connections
- This linear-increase, exponential backoff in the face of congestion is termed *TCP-friendliness*

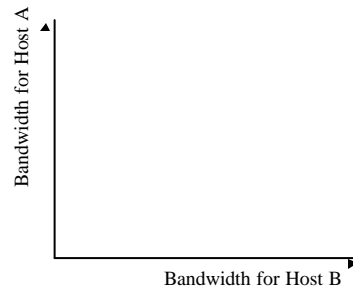
TCP Window Size



- Linear increase
- Exponential backoff
- Assuming no other losses in the network except those due to bandwidth



TCP Fairness

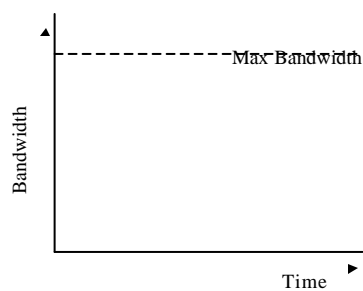


- Want to share the bottleneck link fairly between two flows

TCP Slow Start

- Linear increase takes a long time to build up a window size that matches the link bandwidth*delay
- Most file transactions are not long enough
- Consequently, TCP can spend a lot of time with small windows, never getting the chance to reach a sufficiently large window size
- Fix: Allow TCP to build up to a large window size initially by doubling the window size until first loss

TCP Slow Start



- Initial phase of exponential increase
- Assuming no other losses in the network except those due to bandwidth

TCP Summary

- **Reliable ordered message delivery**
 - Connection oriented, 3-way handshake
- **Transmission window for better throughput**
 - Timeouts based on link parameters
- **Congestion control**
 - Linear increase, exponential backoff
- **Fast adaptation**
 - Exponential increase in the initial phase