COMP2207:
Transport Layer: TCP and UDP
Network Topology

Host A → Router → Router → Host B

Data Flow

Application → process-to-process → Application

Application

Transport → host-to-host → Transport

Transport

Internet

Internet

Internet

Internet

Link

Link

Link

Link

Ethernet

Fiber, Satellite, etc.

Ethernet

TCP or UDP
UDP – User Datagram Protocol

- Connectionless, ‘send and forget’
- Retransmission/adaptation is up to the application
- No flow control; UDP applications are often fixed bit rate
UDP: User Datagram Protocol

• Properties:
  – Connectionless
    • Sender just sends a datagram to a receiver
    • No sequence numbers, no acknowledgements
  – If retransmission is required, the application has to do it – so it is up to the application layer
    • It may or may not be necessary to resend
  – Some UDP applications may use a constant bit rate
    • e.g. video streaming, up to application to adapt if necessary
  – Low overhead
    • Because no connection management being used
    • Uses less bandwidth for the UDP header
UDP

- Allows sending of datagrams without establishing a connection
- the UDP header is much simpler than TCP
- Checksum is optional

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP length</td>
<td>UDP checksum</td>
</tr>
</tbody>
</table>

Then Payload (variable size)…. 
UDP

• Lossy/congested links can drop packets
  – Higher protocols can send a request back to source if needed
• Lower bandwidth links may drop packets as their buffers fill up
  – Applications could detect this and tell server
Transmission Control Protocol

• TCP
  – Connection oriented
  – Includes acknowledgements and retransmissions
  – Provides flow control/congestion control for segments it sends
  – TCP will adjust sending rate over time
TCP/UDP service model

• sender and receiver each create a **socket** to act as a communication endpoint

• socket has an IP address and **port number**
TCP/UDP service model

• The sockets and protocol used, uniquely identify the application’s subsequent data transmissions
Multiple clients

• multiple clients communicate with the same server

  • Each client endpoint will be different
  • Server multiplexes connection, e.g. 1 thread per client endpoint
Berkeley sockets API

• Example of an API to use sockets (C language)
  – Server-side: socket() and bind()
  – Client-side: socket() and connect()
  – e.g. see http://www.linuxhowtos.org/C_C++/socket.htm
  – More from Corina in next part of module...

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCKET</td>
<td>Create a new communication end point</td>
</tr>
<tr>
<td>BIND</td>
<td>Attach a local address to a socket</td>
</tr>
<tr>
<td>LISTEN</td>
<td>Announce willingness to accept connections; give queue size</td>
</tr>
<tr>
<td>ACCEPT</td>
<td>Block the caller until a connection attempt arrives</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Actively attempt to establish a connection</td>
</tr>
<tr>
<td>SEND</td>
<td>Send some data over the connection</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>Receive some data from the connection</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Release the connection</td>
</tr>
</tbody>
</table>
Transmission Control Protocol

• Properties of TCP
  – Provides connection management
    • Similar to the concepts we briefly looked at for the link layer
  – Provides flow control
    • Adapts sending rate to the capacity of the receiver
  – Uses that capacity but tries also to avoid congestion
    • Backs off in the event of packet loss (indicator of congestion)
  – Retransmission
    • Unacknowledged segments are resent by TCP
  – Receiver reassembles segments in the correct order

• So TCP provides performance and reliability on an otherwise unreliable IP service
TCP header

Seq and Ack numbers used to track sequential packets, with SYN bit

Options include Timestamp, Max segment size (MSS) etc.

Header is padded so is a multiple of 32 bits

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Sequence number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgement number</td>
<td></td>
</tr>
<tr>
<td>TCP header length</td>
<td>Window size</td>
</tr>
<tr>
<td>Header options</td>
<td>Urgent pointer</td>
</tr>
<tr>
<td>Options (0 or more 32-bit words)</td>
<td>Data (optional)</td>
</tr>
</tbody>
</table>

TCP header diagram with fields labeled and explained.
TCP connection establishment

- Three-way handshake
  - SYN, SYN-ACK, ACK

- Each side uses a sequence number
  - Repeat packets can be discarded, lost ones resent
  - Common understanding of position in data stream

SYN opens connection (with random seq num)

Server ACKs

Client ACKs. Connection is now established
TCP Reliability

- ACKs are sent back by receiver
- Sender must detect lost packets
  - By Retransmission timeout
    - Estimate when ACK is expected
  - Cumulative acknowledgements (DupAcks)
    - An ACK acknowledges all past data chunks
    - Designed to cause retransmissions..
TCP flow/congestion control
TCP flow control

• TCP uses a *sliding window protocol* to control the sending rate
  • Receiver has limited buffer capacity (memory) to take packets off the network and process them
• Sender should not send data unless receiver indicates it has buffer space to accept it
  – Otherwise will only have to resend later
  – And wastes network bandwidth
• The ‘sliding window’ is effectively the buffer space the receiver says it has available at any given time
  – This will change over time
Using the sliding window

• The sender sends a segment with a sequence number and starts a timer
• Receiver replies with a segment with an acknowledgment number showing next sequence number it expects to receive and its available window size
  – If the sender’s timer goes off before this is acknowledgement is received, the sender retransmits
• If receiver says window size is 0, the sender may send a 1-byte probe to get a new window advertisement
  – Or wait until receiver indicates it has capacity
Example

- Application does a 2K write
- Receiver's buffer:
  - 0
  - 2K
  - Full
  - Application reads 2K
  - 2K
  - 1K

Sender

- Sender is blocked
- Sender may send up to 2K
- 2K SEQ = 0
- ACK = 2048 WIN = 2048
- 2K SEQ = 2048
- ACK = 4096 WIN = 0
- ACK = 4096 WIN = 2048
- 1K SEQ = 4096

COMP2207: TCP and UDP
TCP congestion control

- The TCP congestion window indicates the number of bytes a sender may put into the network at any time
  - Packet loss is a signal of congestion
  - Runs alongside the receiver’s sliding window
  - Use the smaller of the two windows when sending

- The congestion window size starts low
  - Add one segment’s worth per segment acknowledged before acknowledgement timer goes off
  - Known as ‘slow start’
  - If successful, this doubles the window every round trip
Some congestion control details

• The standard ‘slow start’ method includes a threshold
  – When the window size gets to the threshold size, the window size then only grows additively
  – i.e. add at most one segment per successful round trip

• If there is congestion (packet loss)
  – The threshold is lowered
  – The slow start process repeats

• Leads typically to a ‘saw tooth’ traffic profile
  – Different TCP variants have different thresholds/behaviour
  – There is also MPTCP – multi-path TCP
Example

![Graph showing the relationship between transmission number and congestion window. The graph illustrates the concept of threshold and timeout in TCP or UDP networks.](image_url)
Video: TCP or UDP?

• TCP for video
  – The solution used by YouTube, for example
  – Commonly called ‘web streaming’
  – Client connects via TCP (http) and receives the video data, buffering it before playback
  – On a fast link, the client can typically buffer ahead if watching pre-recorded video
    • Possibly download the whole video while only part-played
  – Problematic if link capacity not enough, and TCP backs off - you may see the ‘hourglass of doom’
    • Will stop playing rather than play ‘broken’ video
  – Not so well-suited for live video
Example: YouTube buffer/playback
Popular use of UDP

• Video/audio
  – streaming clients such as VideoLAN (vlc)
  – Immediate, limited buffering, no retransmission
    – Usually fixed transmission rate, e.g. perhaps 5Mbit/s for a given type of higher quality video encoding
  – May experience ‘glitches’ in the video, rather than video hanging, in the event of congestion
  – Well-suited to live video or voice over IP
    – Especially for low UDP header overhead with VoIP
  – Less overhead; no TCP signalling
  – Very well-suited for multicast
    – TCP cannot establish a connection to more than one peer
<table>
<thead>
<tr>
<th></th>
<th>Transmission Control Protocol</th>
<th>User Datagram Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection:</td>
<td>Connection oriented</td>
<td>Connectionless: “Fire and Forget”</td>
</tr>
<tr>
<td>Reliability:</td>
<td>Handles ACK &amp; retransmissions</td>
<td>Application needs to handle ACK &amp; retransmissions if needed</td>
</tr>
<tr>
<td>Data Order:</td>
<td>Guaranteed that arrives and in the correct order</td>
<td>No guarantee that data is received in the order sent</td>
</tr>
<tr>
<td>Header:</td>
<td>20-bytes minimum</td>
<td>8-bytes</td>
</tr>
<tr>
<td>Good for:</td>
<td>Applications that need high reliability</td>
<td>Applications that need fast and efficient transmission</td>
</tr>
<tr>
<td>Example Protocols:</td>
<td>HTTP(S), FTP, SMTP, Telnet, SSH</td>
<td>DHCP, TFTP, SNMP, RIP, RTP, COAP</td>
</tr>
</tbody>
</table>
Summary

• TCP and UDP offer different transport layer services to applications
  – Choice will depend on scenario and application
• TCP offers reliable connection-oriented service
  – Adapts sending rate based on observed congestion and receiver’s capacity
  – Typically used for web services, for example
• UDP is connectionless and lightweight
  • Typically used for voice over IP, for example
• Application examples
  – Can use TCP or UDP for video; each has advantages
  – For DNS, UDP makes operation more lightweight
reading

• Tanenbaum chapter 6 “transport layer”
• TCP on wikipedia