Transport Layer

Process-to-process communication
The unit of data at this level is usually called a "segment" – sometimes a "Protocol Data Unit (PDU)"

"The Applications"
- App
- Pres

"The Network OS"
- Sess
- Trans
- Net

"The Hardware"
- DL
- PHY
Transport-layer protocols in the TCP/IP stack

• We'll discuss:
  – User Datagram Protocol (UDP)
    • Packet-oriented, Best effort
  – Transmission Control Protocol (TCP)
    • Byte-stream oriented, Reliable

• Note that there are others (covered in CS670)
  – Remote Procedure Call (RPC)
    • Request/Reply paradigm
  – Real-Time Protocol (RTP)
    • For real-time (e.g., multimedia) apps
  – Others added through RFC ballotting

UDP
UDP

- “IP with ports”
- Best effort (not reliable)
- connectionless

Why a Best-Effort protocol (instead of a reliable one)?
- Speed
- Simplicity

UDP

<table>
<thead>
<tr>
<th>16 bits</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>Dest Port</td>
</tr>
<tr>
<td>Checksum</td>
<td>Length</td>
</tr>
<tr>
<td>Payload</td>
<td></td>
</tr>
</tbody>
</table>

Features that give simplicity, speed
- Connectionless, so no pre-transmit setup
- Small headers
- Unregulated flow rate, no error re-xmits (but application can do that)
- “stateless” (segments are unrelated to previous segments)
UDP applications

- Used for many apps where reliability is less critical than speed
  - Streaming multimedia
  - Internet telephony
  - DNS service

TCP
TCP

- TCP = Transmission Control Protocol
  - Reliable (Guarantees all bytes will be delivered, in-order, no errors)
  - Connection-Oriented
  - Designed for “byte-stream” data

- Includes:
  - Flow control (Sliding Window)
  - Congestion control (Discussed later)

Byte stream data

App port

Send Buffer

Segments

Rcv Buffer
TCP header format

- **Source port**: 16 bits
- **Dest port**: 16 bits
- **Sequence number**: 32 bits
- **Ack number**: 32 bits
- **Header length**: 4 bits
- **URG**: 1 if this segment contains Urgent Data (Urgent Pointer is valid)
- **ACK**: 1 if this is an ACK segment (Acknowledgment Number is valid)
- **PSH**: 1 if segment is to be delivered to receiving process immediately
- **RST**: 1 error flag (invalid seg, connection refused,...)
- **SYN**: Used when establishing a connection
- **FIN**: Used to release a connection
"pseudo header"

<table>
<thead>
<tr>
<th>Source IP address</th>
<th>Dest IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0000</td>
<td>Protocol ID</td>
</tr>
<tr>
<td>TCP segment length</td>
<td></td>
</tr>
</tbody>
</table>

- Included in Checksum to help detect misrouted, mis-synched segments – if the address was changed en-route, the receiver would see a checksum error
- Also used in calculating Checksum for UDP
- Note that this violates the normal assumption of independence of protocol layers, but it's needed to catch mis-routed packets

TCP options field

- Used to extend the protocol
- Some examples:
  - Specify max segment size you will accept (during setup)
  - Specify a scaling factor for the window size field
Establishing a TCP connection

"Three-Way Handshake"

Client

- CONNECT
- IP
- Max seg
- Starting SN

Server

- LISTEN, ACCEPT
- Server ACKs clients desired Starting SN, and sends back its own
- Recall that the ACK specifies the SN of the next byte that the receiver expects

Client ACKs server’s desired SN

“SYN SN=x”

“SYN SN=y, ACK=x+1”

“SYN SN=x+1, ACK=y+1”

Connection Established

TCP connection management

Legend:
- Event / Action
- CLOSE/FIN
- FIN
- FIN/Ack
- FIN/ack
- ACK
- ACK/Ack
- SYN
- SYN/SYN+ACK
- SYN+ACK
- Close wait
- Last ACK
- Timeout
- close
- CONNECT/SYN

G. W. Cox -- Fall 2007

TCP
Opening: 3-way handshake

Legend:
- Event / Action
- Client
- Server

Sliding Window Algorithm -- refresher

- Guaranteed delivery
  - All frames ACKed
  - Timeouts to detect lost frames, ACKs
  - No window movement until preceding block of frames delivered

- In order
  - All frames numbered
  - Receiver inserts frame in correct position, according to number
  - No ACK until frames in order

- Flow Control
  - Send rate regulated by send window size
  - No window movement until receiver ACKs
Extending SWA for TCP:
Dynamic window size

- In each ACK, receiver specifies a max Send Window size “Advertised Window Size”
- This allows receiver to regulate the flow dynamically according to the traffic it is able to handle at any moment

Extending SWA for TCP:
Larger sequence numbers

- In a network with high-speed links, a small SN field can roll over quickly
- That might be OK if all data was delivered quickly, but big networks can have big delays – segments can be as late as the IP TTL (120 sec, typ)
- Recall that if there are two items in flight with the same SN, the SWA algorithm can fail
- The fix: 32-bit sequence numbers
In a big network, response times can vary widely from moment to moment – this complicates the strategy of using RTT to set the timeout time.

The fix:
- Sender keeps running average of time between SEND and ACK to each receiver
- Timeout adjusted dynamically according to current average (for example, timeout = 2 x current_RTT)

How does TCP decide when to send a segment?

1. When a predetermined number of bytes have been accumulated in the send buffer (negotiated seg size)
2. On demand from higher level (e.g., to send urgent data)
3. On a time basis (so that segs are not delayed waiting on data from a slow-talking application)
A final note

- TCP has a surprising number of parameters and control options
- To learn more, read Peterson 5.2.
- To learn still more, take CS670