Transport Layer Review

Mahalingam Ramkumar
Mississippi State University, MS

October 1, 2014
Transport Layer Functions

- Distinguish between different application instances through port numbers
- Make it easy for applications to use the unreliable “virtual link” created by network layer
- Two popular transport layer protocols
- TCP (transmission control protocol) and UDP (user datagram protocol)
TCP vs UDP

- TCP is connection oriented
- Can be used to send and recv data packet only after connection is established
- `connect()` should be used before applications can use `send()` and `recv()`
- UDP is not connection oriented
- `sendto()`, `recvfrom()`
TCP vs UDP

TCP guarantees
- all data bytes will be delivered in the right order to the correct process
- flow control to ensure that a fast sender can not saturate a slow receiver

UDP guarantees that *if* the packet is received, it will be provided to the correct process

Applications will have to create acknowledgments (and do everything else TCP does) *if* reliable delivery needs to be guaranteed
Important components of TCP header
- One-bit FLAGS (SYN,ACK,FIN,RST,PSH,URG),
- sequence number and acknowledgment number (32-bit numbers)
- Window size

Three phases
- connection establishment (three way handshake, SYN, ACK flags, sequence and acknowledgement numbers)
- sending/receiving data (sequence numbers, acknowledgement numbers, ACK flag)
- Connection termination (FIN and ACK flags, acknowledgement numbers)
A and $B$ TCP end-points

SYN - SYN/ACK - ACK handshake

Only TCP headers, no application data bytes.

$A \rightarrow B$: [SYN, seq=45678] (request for connection; my starting sequence number is 45678)

$B \rightarrow A$: [SYN, seq=43234, ACK, ack=45679] (my starting seq is 43234, received your request, next data byte I am expecting from you is 45679)

$A \rightarrow B$: [SYN, seq=45679, ACK, ack=43235], no data sent
TCP layer has to keep track of all current connections

A connection is identified using four values sender and destination port, sender and destination IP

For each connection TCP remembers starting sequence numbers, keeps track of current sequence and ack numbers, buffer space etc., and uses various timers
After Connection is Established

- Starting sequence numbers (A-45678, B-43234)
- A → B: [seq=45900, ACK, ack=46456, W=20000] + [100 data bytes] conveys the following
  - The first byte included in this packet has sequence number 45900 (which is actually the 45900-45678=222nd byte of app data): bytes 222 to 321 included in the packet.
  - The next packet I am expecting from you should begin with the data byte with sequence number 46456 (I have so far received the 3221 bytes corresponding to sequence numbers 43235 to 46455)
  - I have 20000 bytes of buffer space left — you can send me 20000 bytes (46456 to 66455) before you receive my next ack.
Connection Termination

- Two parallel full duplex connections
- one for sending data from $A \rightarrow B$ and acks from $B \rightarrow A$
- the other for sending data from $B \rightarrow A$ and acks from $A \rightarrow B$
- Terminated individually

If $A$ has no more data to send $A$ can initiate termination of the line $A \rightarrow B$ (by sending a TCP header with FIN flag set)

- $B$ acknowledges FIN by sending an acknowledgement number that counts the FIN flag as a byte.

- $A$ starting seq=45678, $A$ sent bytes 422 bytes 45679 to 46100. If FIN was not received the acknowledgement number from $B$ will be 46101. If FIN was received the number is 46102

- $B$ can continue to send data bytes to $A$ even after one connection is terminated.
Connection Termination

- FIN, ACK, FIN, ACK (four packets)
- If both are ready to terminate around the same time it can be three packets FIN, ACK/FIN, ACK
Window Size

- Window size is dynamic
- Remaining buffer space
- TCP layer cannot control when the application will read the buffer and clear it.
- Necessary for flow control (can send window size 0 to stop the other end from sending more data)
- Necessary for pipe-lining (permit other end can send $W$ bytes before it receives the next acknowledgement: the next ack can indicate a different window size)
- 16-bit restricted maximum window size to 65536 initially
- A window scaling (WS) option can be specified in the header to overcome this limitation
- $WS = 5$, $W = 20000$ implies window size is $2^5 \times W = 640,000$
Other TCP Topics

- Timers, Congestion control, Transactional TCP
- Will be revisited later
Not very different from directly using the IP layer to send a packet.
The only difference is that port numbers in the UDP header is needed to address processes (IP can address only the platform).
UDP has very little overhead.
Have to use UDP for multitasking (can not use TCP. Why?)