Transport Layer Contd.

COMP90007

Internet Technologies

Looking under the hood of TCP: Connections

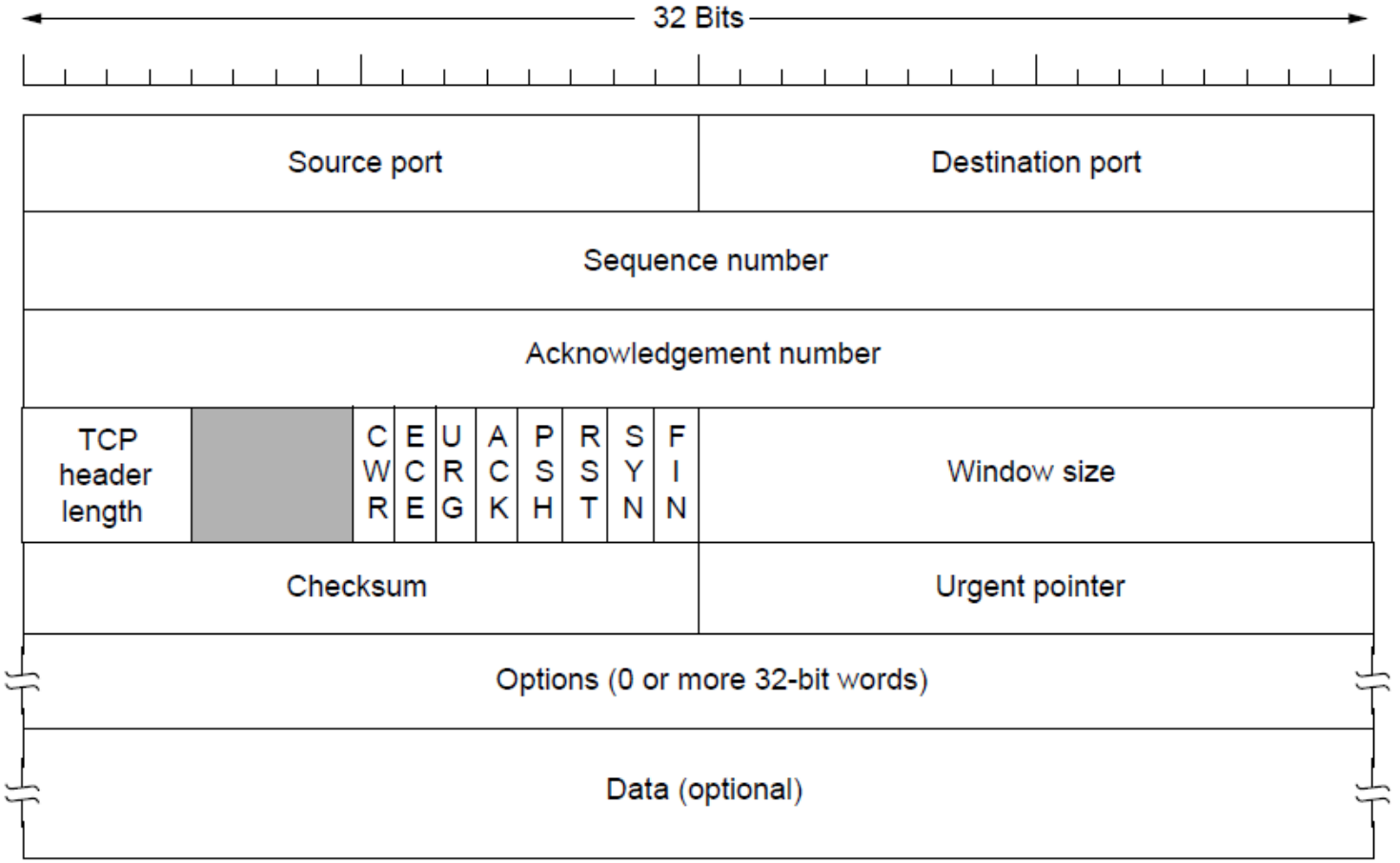
- TCP connections are:
- Full duplex - data in both directions simultaneously
- Point to point - exact pairs of senders and receivers
- Byte streams, not message streams - message boundaries are not preserved
- Buffer options - TCP entity can choose to buffer prior to sending or not depending on the context
  - **TCP_NODELAY in Java**
  - **Socket.setTcpNoDelay(boolean)**

TCP Details Contd

- Data sent between TCP entities in segments - segment has a **20 byte header plus zero or more data bytes**
- TCP entities decide how large segments should be mainly with 2 constraints:
  - **65,515 byte IP payload**
  - **Ethernet unit size - generally 1500 bytes**
- Sliding window - **sender transmits and starts a timer**
- Receiver sends back an acknowledgement which is the next sequence number expected - if sender's timer expires before acknowledgement, then the sender **transmits the original segment again**

The TCP Segment Header

- TCP header includes addressing (ports), sliding window (seq. / ack. number), flow control (window), error control (checksum) and more



The TCP Segment Header

- ***Source port* and *Destination port*** fields identify the local end points of the connection
- ***Sequence number and Acknowledgement number*** fields perform their usual functions
- ***TCP header* length** tells how many 32-bit words are contained in the TCP header
- ***Window size*** field tells how many bytes may be sent starting at the byte acknowledged
- ***Checksum*** is also provided for extra reliability. It checksums the header, the data
- ***Options*** field provides a way to add extra facilities not covered by the regular header
- ***URG*** is set to 1 if the *Urgent pointer* is in use. The Urgent pointer is used to indicate a byte offset from the current sequence number at which urgent data are to be found

The TCP Segment Header

- ***CWR*** and ***ECE*** are used to signal congestion when *ECN* (Explicit Congestion Notification) is used
- ***ECE*** is set to signal an ECN-Echo to a TCP sender to tell it to slow down when the TCP receiver gets a congestion indication from the network
- ***CWR*** is set to signal Congestion Window Reduced from the TCP sender to the TCP receiver so that it knows the sender has slowed down and can stop sending the ECN-Echo
- The ***ACK*** bit is set to 1 to indicate that the Acknowledgement number is valid. This is the case for nearly all packets. 0 means ignore ACK number field
- ***PSH*** bit indicates PUSHed data. The receiver is hereby kindly requested to deliver the data to the application upon arrival and not buffer it until a full buffer has been received

The TCP Segment Header

- The ***RST*** bit is used to abruptly reset a connection that has become confused due to a host crash or some other reason. It is also used to reject an invalid segment or refuse an attempt to open a connection
- The ***SYN*** bit is used to establish connections. The connection request has SYN = 1 and ACK = 0. The connection reply does bear an acknowledgement, so it has SYN = 1 and ACK = 1.
- In essence, the SYN bit is used to **denote both CONNECTION REQUEST and CONNECTION ACCEPTED**, with the ACK bit used to distinguish between those two possibilities.
- The ***FIN*** bit is used to release a connection. It specifies that the sender has no more data to transmit. However, after closing a connection, the closing process may continue to receive data.

Details of TCP Connection Establishment and Release

- Connections established using three-way handshake
- Two **simultaneous connection attempts results in only one connection** (uniquely identified by end points)
- Connections released with **symmetric release**
- Timers used for lost connection releases due to **three army problem**

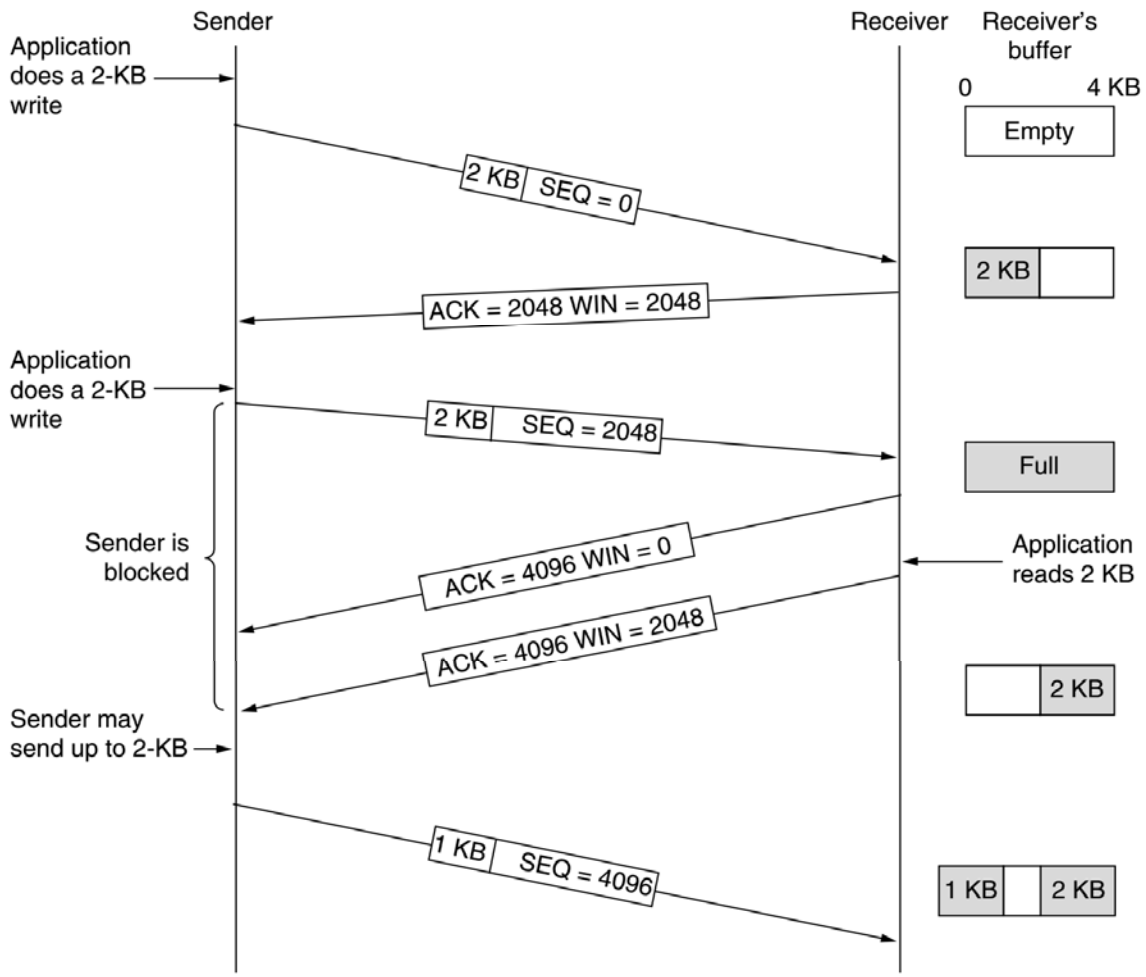
TCP Connection Management – The Full Set of States

- The full TCP connection finite state machine has more states than the simple example from earlier.

State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIME WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

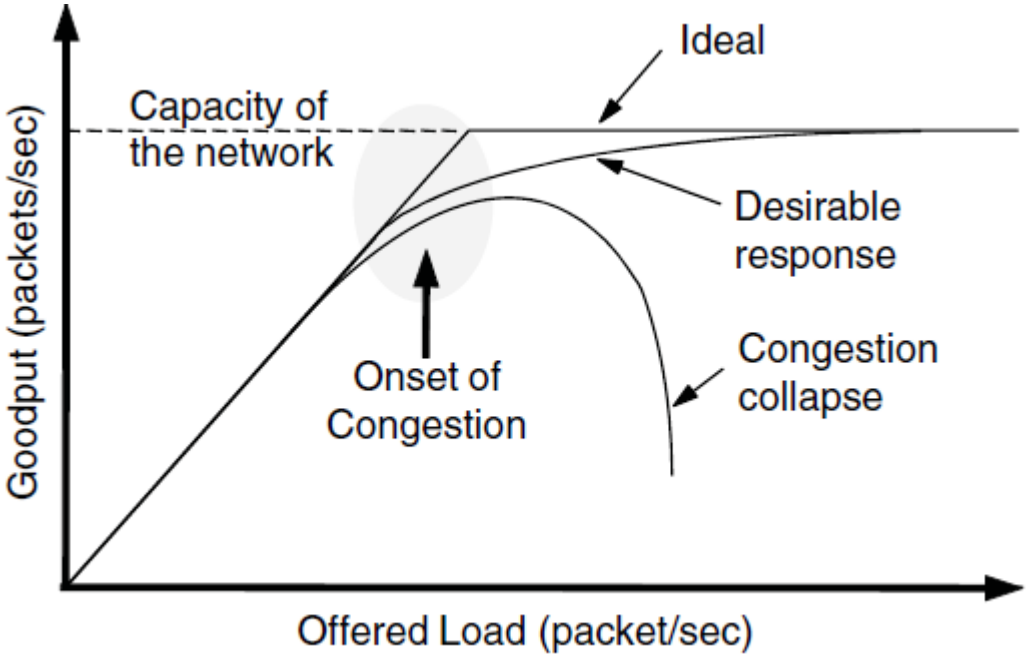
TCP Transmission Policy

- TCP acknowledges bytes
- Receiver advertises window based on available buffer space



What happens if there is congestion?

- Congestion results when too much traffic is offered; performance degrades due to loss/retransmissions
  - Goodput (=useful packets) trails offered load



Congestion Control vs Flow Control

- Flow control is an issue for point to point traffic, primarily concerned with preventing sender transmitting data faster than receiver can receive it
- Congestion control is an issue affecting the ability of the subnet to actually carry the available traffic, in a global context

Load Shedding

- When congestion control mechanisms fail, load shedding is the key remaining possibility - **drop packets**

- In order to ameliorate impact, applications can mark certain **packets as priority** to avoid discard policy

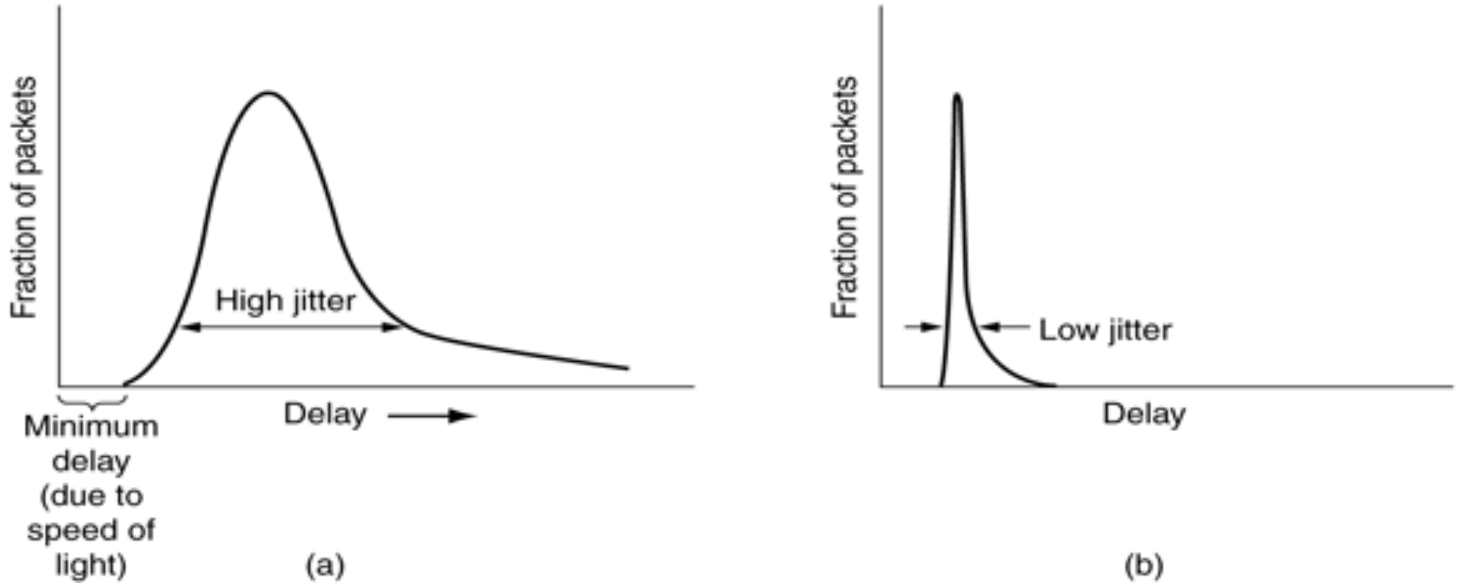
But what is the key problem if network is not delivering properly?

- Quality of Service becomes low: Unhappy User
- **Expected network performance** is an important criterion for a wide range of network applications
- Some **engineering techniques are available to guarantee QoS** (Quality of Service)
- 4 dimensions to watch out for:

  bandwidth, reliability, delay, jitter

Lets Look at Details of Jitter at Transport Layer

- Jitter is the **variation in packet arrival times**
  - a) high jitter
  - b) low jitter



Mechanisms for Jitter Control

- Jitter is important for some application??
- Jitter can be contained by **determining the expected transit time** of a packet
- Packets can be **shuffled at each hop in order to minimise jitter** - slower packets sent first, faster packets wait in a queue
- For certain applications jitter control is extremely important as it mainly directly affects the **quality perceived by the application user**