



Computer Network Programming

TCP Overview

Dr. Sam Hsu

Computer Science & Engineering

Florida Atlantic University



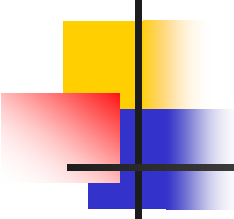
TCP Overview

- Introduction
- TCP Segments
- Sliding Windows
- TCP Checksum
- TCP Data Transfer
- Timeout & Retransmission
- Silly Window Syndrome
- Some TCP Timers
- Congestion Control
- TCP Sequence Numbers
- TCP State Diagram



TCP (1/3)

- TCP is short for *T*ransmission *C*ontrol *P*rotocol.
- It provides a connection-oriented, reliable, end-to-end byte stream delivery service with the following features:
 - Connection-oriented (virtual).
 - Full-duplex data transfer.
 - Unstructured byte stream delivery.

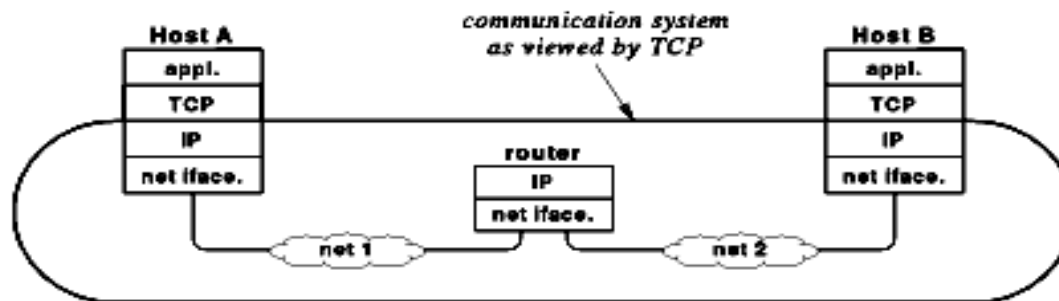


TCP (2/3)

- Buffered transfer.
- Cumulative acknowledgment.
- End-to-end flow control.
- Sender timeout and retransmission.
- TCP makes very few assumptions about the underlying network.
 - It may be employed over a variety of different physical networks.

TCP (3/3)

- TCP provides end-to-end services.



- TCP is an end-to-end transport protocol. It views IP as a mechanism that allows TCP software on a host to exchange messages with TCP software on a remote host.

Ref: *Computer Networks and Internets, 4th* ., Douglas Comer, Prentice Hall, 2004, p. 378.



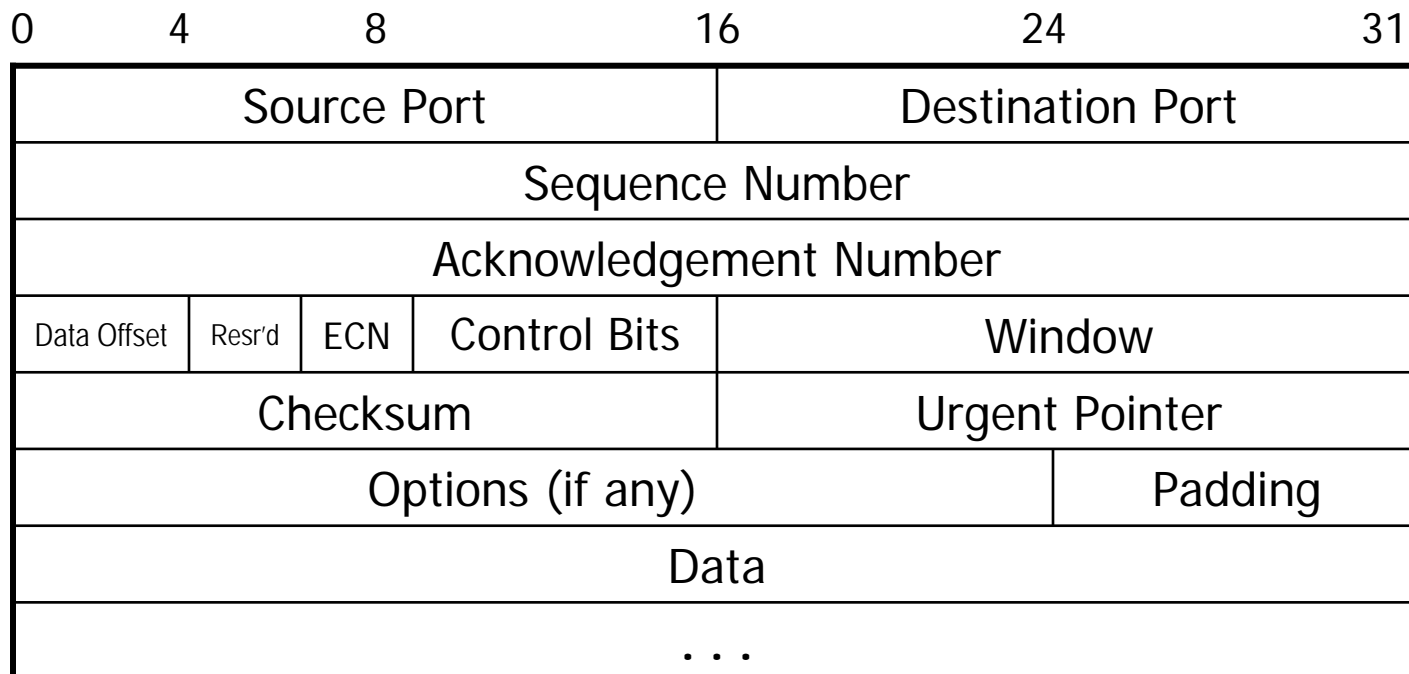
TCP Segments

- TCP breaks a data byte stream into segments for transmission.
 - Segments may be of different sizes.
 - Segments are used to:
 - Establish connections.
 - Transport data and acknowledgements (ACKs).
 - Advertise window sizes.
 - Close connections.



Segment Format (1/3)

- Format of a TCP segment with a TCP header followed by data.





Segment Format (2/3)

- Data Offset
 - Four bits, used to indicate the number of 32-bit words in the TCP header
 - Also known as *Header Length*
- Reserved
 - Three bits, initialized to all zeros
- ECN
 - Three bits, used for explicit congestion notification



Segment Format (3/3)

- Control bits

Bit (left to right)	Meaning (if bit set to 1)
URG	Urgent pointer field is valid
ACK	Acknowledgement field is valid
PSH	This segment requests a push
RST	Reset the connection
SYN	Synchronize sequence numbers
FIN	Sender has reached end of its byte stream



MSS

- Maximum segment size (MSS)
 - MSS refers to the size of the biggest chunk of data that can be received by the destination.
 - The size is negotiated during connection establishment, and is determined by the destination of the segment.
 - The default is chosen otherwise.
 - An MSS may be the minimum MTU along the path or 536 octets.

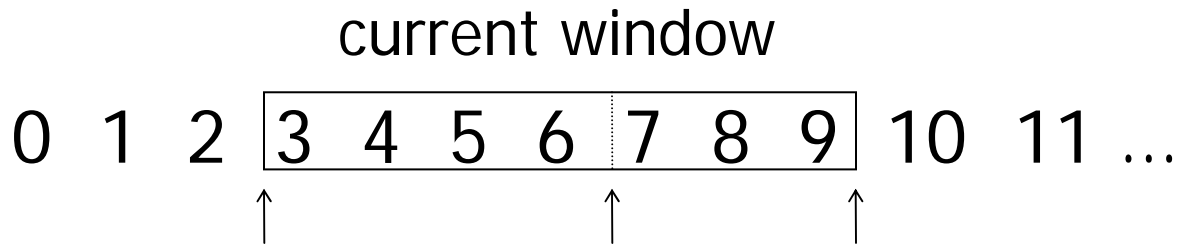


Sliding Windows (1/2)

- TCP uses a sliding-window protocol for end-to-end flow control.
 - Window size is measured in bytes.
 - Acknowledgements are cumulative.
 - Employs a variable window size mechanism.

Sliding Windows (2/2)

- An example of the TCP sliding window:



Octets through 2 have been sent and acknowledged, octets 3 through 6 have been sent but not acknowledged, octets 7 through 9 have not been sent but will be sent without delay, and octets 10 and higher cannot be sent until the window move.

Ref: *Internetworking with TCP/IP Volume I*, 5th ed., Douglas Comer, Prentice Hall, 2005, p. 197.



Port Numbers (1/2)

- A (protocol) port is an abstraction used by TCP to distinguish applications on a given host.
- A port is identified by a 16-bit integer known as the *port number*.
- Three ranges of port numbers:
 - Well-known ports
 - Ranging from 0 to 1,023.
 - A set of pre-assigned port numbers for specific uses.



Port Numbers (2/2)

- Registered ports
 - Ranging from 1,024 to 49,151.
 - Not assigned or controlled by IANA; however their uses need to be registered with IANA to prevent duplications.
- Dynamic ports
 - Ranging from 49,152 to 65,535.
 - Neither assigned or registered. They can be used by anyone.
 - These are ephemeral ports.



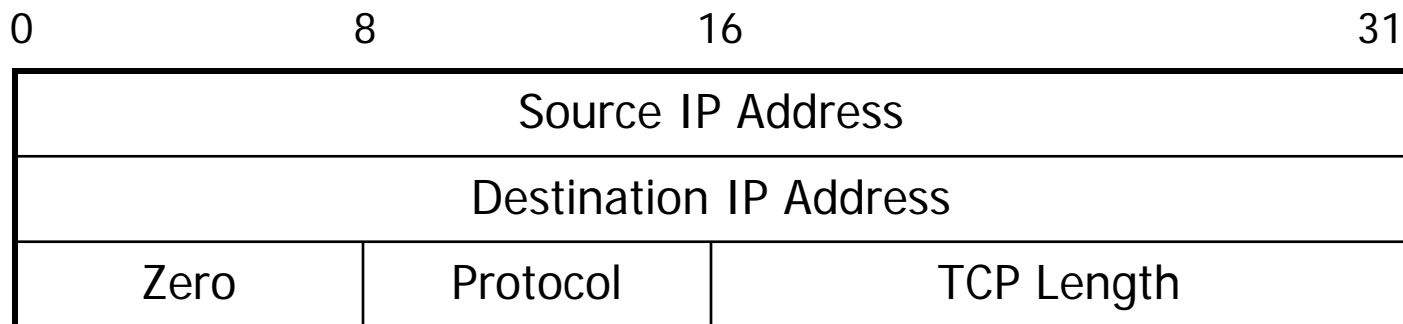
TCP Checksum (1/2)

- Used to ensure integrity of data transferred.
 - Applies to header as well as data.
- Formed by treating the segment as a sequence of 16-bit integers, in network standard byte order, adding them together using 1's complement arithmetic, and then taking 1's complement of the result.
 - Applying the same computation algorithm at the receiving site should result in a zero value.



TCP Checksum (2/2)

- In need of a pseudo IP header to compute the TCP checksum.
- Format of the pseudo IP header:



- At the receiving end, this information is extracted from the IP datagram carrying the segment.



TCP Data Transfer (1/2)

- Application data are broken into the best sized chunks known as *segments*.
- A timer is started after a segment is sent, and retransmission will occur if the timer expires before data in the segment has been acknowledged.
- TCP will send an acknowledgment after receiving a segment(s) successfully from the other end.
- TCP acknowledgment scheme is cumulative.



TCP Data Transfer (2/2)

- A received segment is simply discarded if it fails the checksum error check.
 - No NACK sent, just wait for timeout.
- TCP will resequence segments if necessary.
 - For segments received out of order.
- Duplicate data are discarded.
- TCP adjusts available window size dynamically to enforce end-to-end flow control.
 - A variable window size mechanism.



Timeout & Retransmission (1/3)

- An adaptive retransmission algorithm
 - $RTT = (\alpha * Old_RTT) + ((1 - \alpha) * New_Round_Trip_Sample)$
 - $timeout = \beta * RTT$
 - $new_timeout = \gamma * timeout$

Note: α , β , and γ values are usually 0.9, 2, and 2 respectively.



Timeout & Retransmission (2/3)

- Due to a wide range of variation in delay in computing mean RTT, a better way of estimating RTT and computing timeout, taking variance in delay into consideration, is specified as follows:
 - $DIFF = \text{New_Round_Trip_Sample} - \text{Old_RTT}$
 - $RTT = \text{Old_RTT} + \delta * DIFF$
 - $DEV = \text{Old_DEV} + \rho * (|DIFF| - \text{Old_DEV})$
 - $\text{timeout} = RTT + \eta * DEV$



Timeout & Retransmission (3/3)

where

- δ , ρ , and η are recommended to be $1/2^3$, $1/2^2$ and 4 respectively.

Note that the following still applies:

- $\text{new_timeout} = \gamma * \text{timeout}$



Ambiguity in Retransmission

- Retransmission ambiguity problems
 - An acknowledgement received after a retransmission is activated may be ambiguous:
 - For which segment is this ACK for?
 - The original segment or the retransmitted one?
 - One solution: Use Karn's algorithm to avoid the ambiguity.



Karn's Algorithm

- When a retransmission occurs, don't update the current RTT until a new segment is sent and also ACK'ed successfully without a retransmission.
 - However, the new timeout value will be recomputed for each retransmission of the segment using a backoff strategy and this value is also retained for the subsequent new segment after the current retransmitted segment is ACK'ed
 - A different timeout value will thus be recomputed using the new RTT upon a successful transmission of a new segment without a retransmission.



Silly Window Syndrome (1/2)

- A situation in which small amounts of data are exchanged across the network.
 - Very inefficient use of the network resources.
- It could be caused by either end.
 - The receiver can advertise small windows.
 - The sender can transmit small amounts of data.



Silly Window Syndrome (2/2)

- To avoid this,
 - The receiver must not advertise small segments.
 - The sender tries not to send data smaller than a full-sized segment.
- Some algorithms to solve this problem:
 - Nagle's algorithm (for the sender)
 - Clark's algorithm (for the receiver)
 - Delayed acknowledgement (for the receiver)



Response to Congestion (1/4)

- Congestion is a condition of severe delay caused by an overload in some part of the network.
 - Delay may lead to retransmissions of segments due to timeouts; however, retransmissions aggravate congestion instead of alleviating it.
 - If unchecked, the increased traffic will produce increased delay which, in turn, will generate more traffic, and so on, until the network ends up in *congestion collapse*.



Response to Congestion (2/4)

- TCP uses two window size limits.
 - Receiver window advertisement.
 - Congestion window.
- However, actual window size used in transferring segments is:
 - $\text{Actual_window} = \min(\text{receiver_advertisement}, \text{congestion_window})$



Response to Congestion (3/4)

- TCP uses two techniques to avoid congestion
 - Multiplicative decrease.

Upon loss of a segment, reduce the congestion window by half (down to a minimum of at least one segment). For those segments that remain in the allowed window, backoff the retransmission timer exponentially.



Response to Congestion (4/4)

- Slow start

Whenever starting traffic on a new connection or increasing traffic after a period of congestion, start the congestion window at the size of a single segment and increase the congestion window by one segment each time an acknowledgement arrives.

Ref: *Internetworking with TCP/IP Volume I*, 5th ed., Douglas Comer, Prentice Hall, 2005, pp. 212-213.



Random Early Detection (1/4)

- Random Early Detection (RED) is also known as *Random early Discard*, or *Random Early Drop*.
- RED is for a better TCP performance in case of buffer overflows.
 - A router drops arriving datagrams when its input queue overflows.
 - A technique know as *tail-drop* policy.
 - If the datagrams dropped came from a single TCP connection, the drops may cause it to enter *slow-start*.



Random Early Detection (2/4)

- If the datagrams dropped came from many TCP connections, the drops may cause many TCP connections to enter slow-start.
 - This can cause *global synchronization* problem to occur.
- RED is designed to avoid the global synchronization problem.



Random Early Detection (3/4)

- How RED works?
 - Two threshold values, T_{min} and T_{max} are used to mark the positions in the queue of a router.
 - When a new datagram arrives at the queue,
 - If the queue currently contains fewer than T_{min} datagrams, add the new arrival to the queue.
 - If the queue currently contains more than T_{max} datagrams, discard the new arrival.
 - If the queue currently contains between T_{min} and T_{max} datagrams, randomly discard the new arrival based on a probability p .



Random Early Detection (4/4)

- RED computes a weighted average queue size, *avg*, in octets to determine the probability *p*, for the arriving datagram.

$$\text{avg} = (1 - \gamma) * \text{Old_avg} + \gamma * \text{Current_queue_size}$$

where

γ is a value between 0 and 1.

Note: If γ is small enough, the weighted average will track long term trends, but will remain immune to short data bursts. A suggested value for γ is 0.002.

- In essence, RED is an active queue management algorithm and also a congestion avoidance algorithm.



Some TCP Timers (1/4)

- Persistence timer
 - What if an ACK with the Window field set to non-zero following an ACK with the Window field set to zero, is lost?
 - Deadlock may occur.
 - To avoid this, a persistence timer is set after an ACK with a window size of zero is received.



Some TCP Timers (2/4)

- The sending TCP will send a special segment called a *probe* when the persistence timer goes off.
 - The probe is a one-byte segment, not to be acknowledged.
- The initial timeout value for the probe is the then current RTT. It gets doubled for each timeout until it reaches 60 seconds. It then stays at 60 seconds until the window is reopened.



Some TCP Timers (3/4)

- Keepalive timer
 - Is used by the server to prevent a long idle connection.
 - The server sends a probe segment to a client after an initial idle period of two hours.
 - It then sends the subsequent probes at an interval of 75 seconds each until a response is received.
 - The server will terminate the connection after the client has not responded to 10 probes.



Some TCP Timers (4/4)

- Time-waited timer
 - Is also known as the 2MSL timer
 - MSL may be 30 seconds, 1 minute, or 2 minutes.
 - Is used for normal connection termination.
 - To give TCP enough time to handle the loss of any of the last four segments during connection termination.
 - To prevent old duplicates being misinterpreted as belonging to a new incarnation of the same connection.



TCP Connection Reset

- For abnormal termination of connections.
- By sending a segment with the RST bit set.

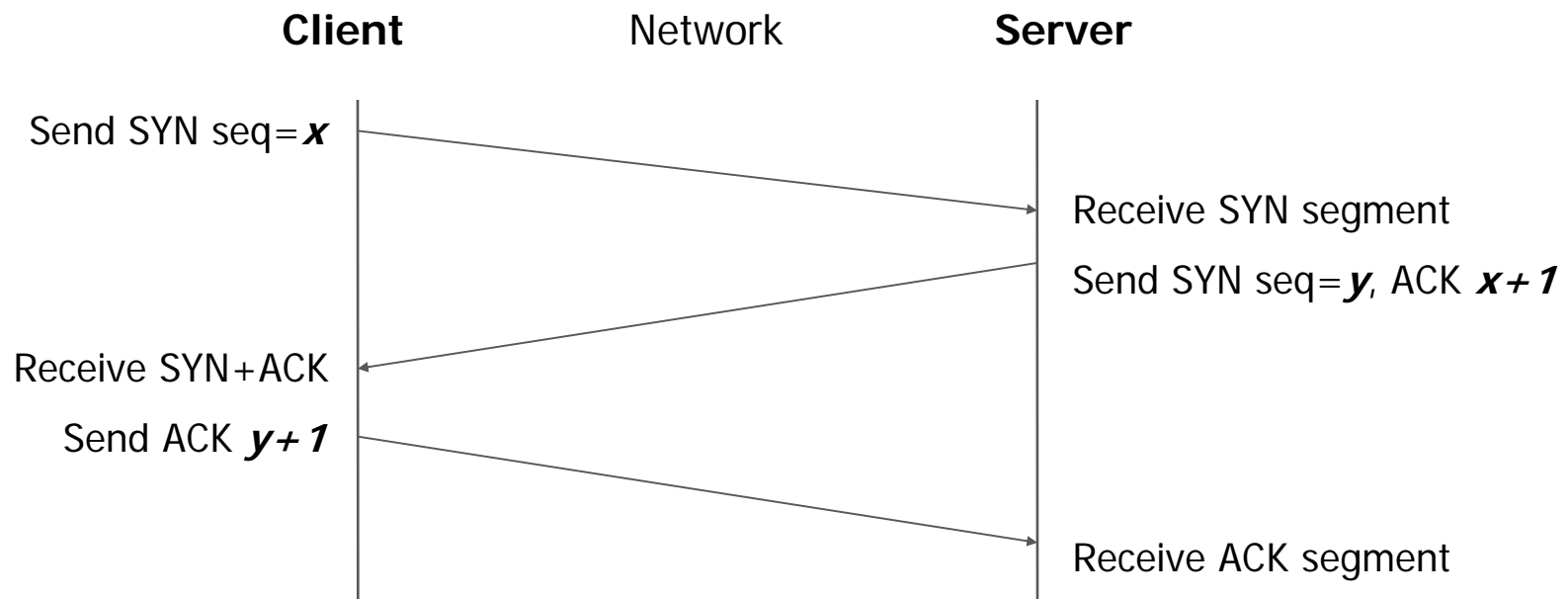


TCP Sequence Numbers

- A finite 32-bit number space
 - $0 - 4,294,967,295$ ($2^{32} - 1$)
- Sequence numbers wrap around.
- One point to ponder:
 - *How to tell that a segment with a sequence number smaller than expected is not a retransmission?*

TCP Connection Establishment

- A 3-way handshaking process



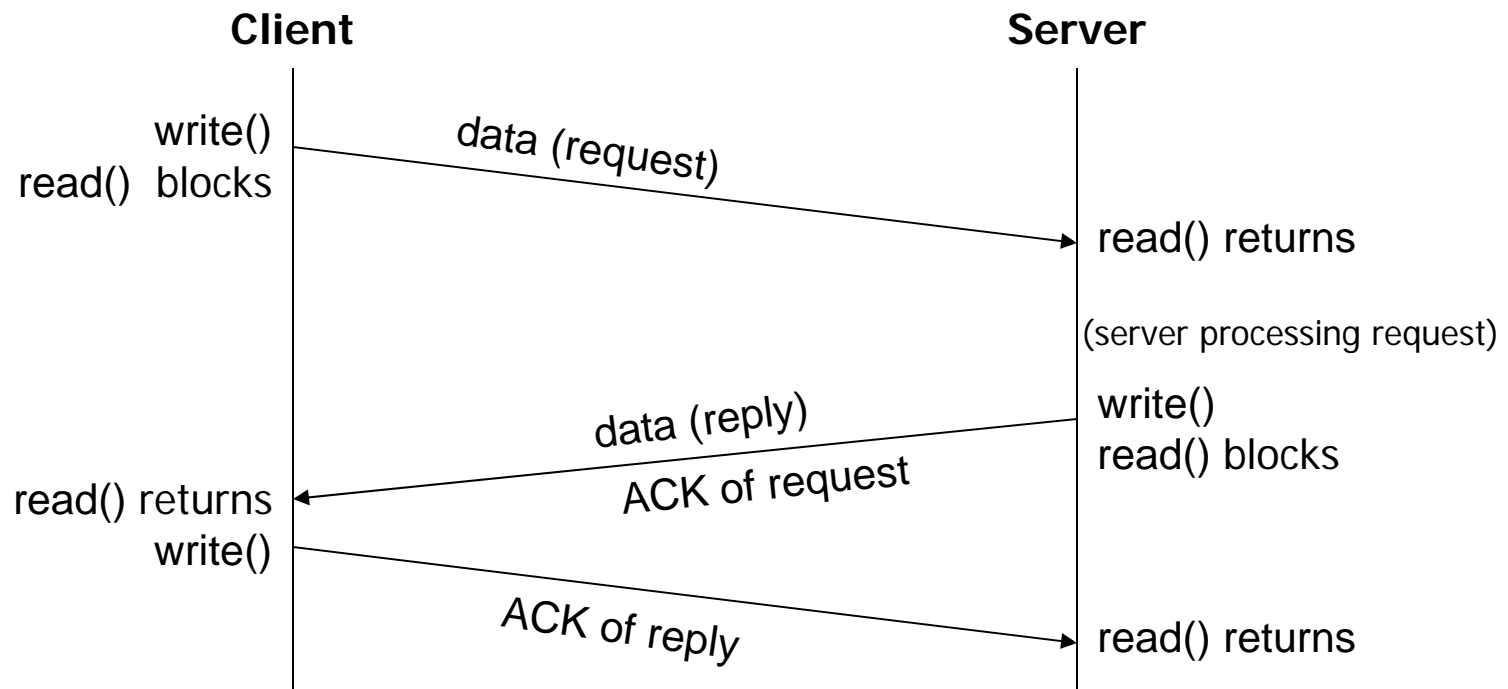


TCP Connection Options

- Three common TCP options (sent in SYN):
 - MSS option: The *maximum segment size* that can be received.
 - 65,535 (16 bits for the window size field in the TCP header) is the upper limit.
 - Window scale option: To allow the advertised window size to be scaled (left-shift) 0-14 bits.
 - MSS can thus reach up to $65,535 \times 2^{14}$ bytes.
 - Timestamp option: May be used for high-speed connections to prevent possible data corruption caused by old, delayed, or duplicated segments.

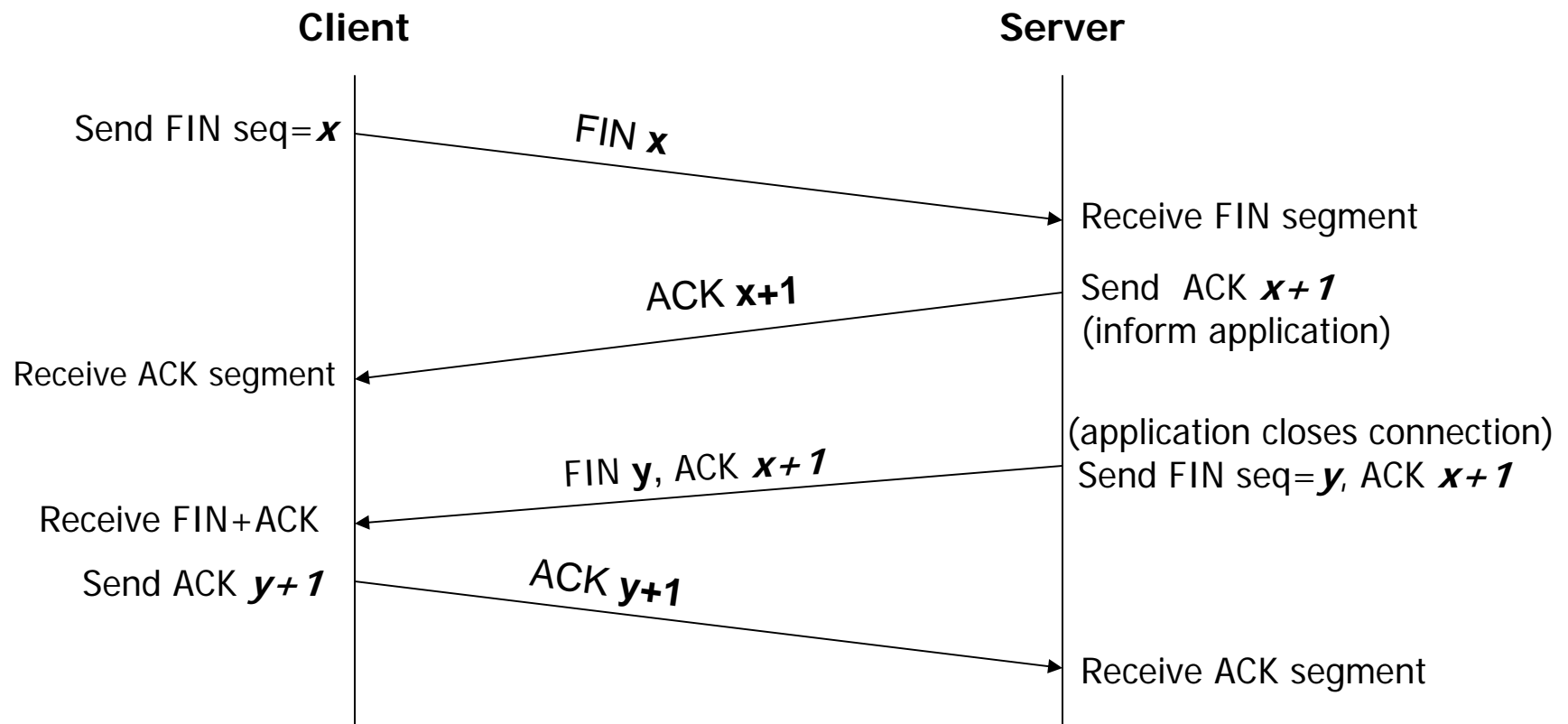
TCP Data Transfer

- Packets exchanged for data transfer



TCP Connection Termination

- A 4-way handshake



TCP State Transition Diagram

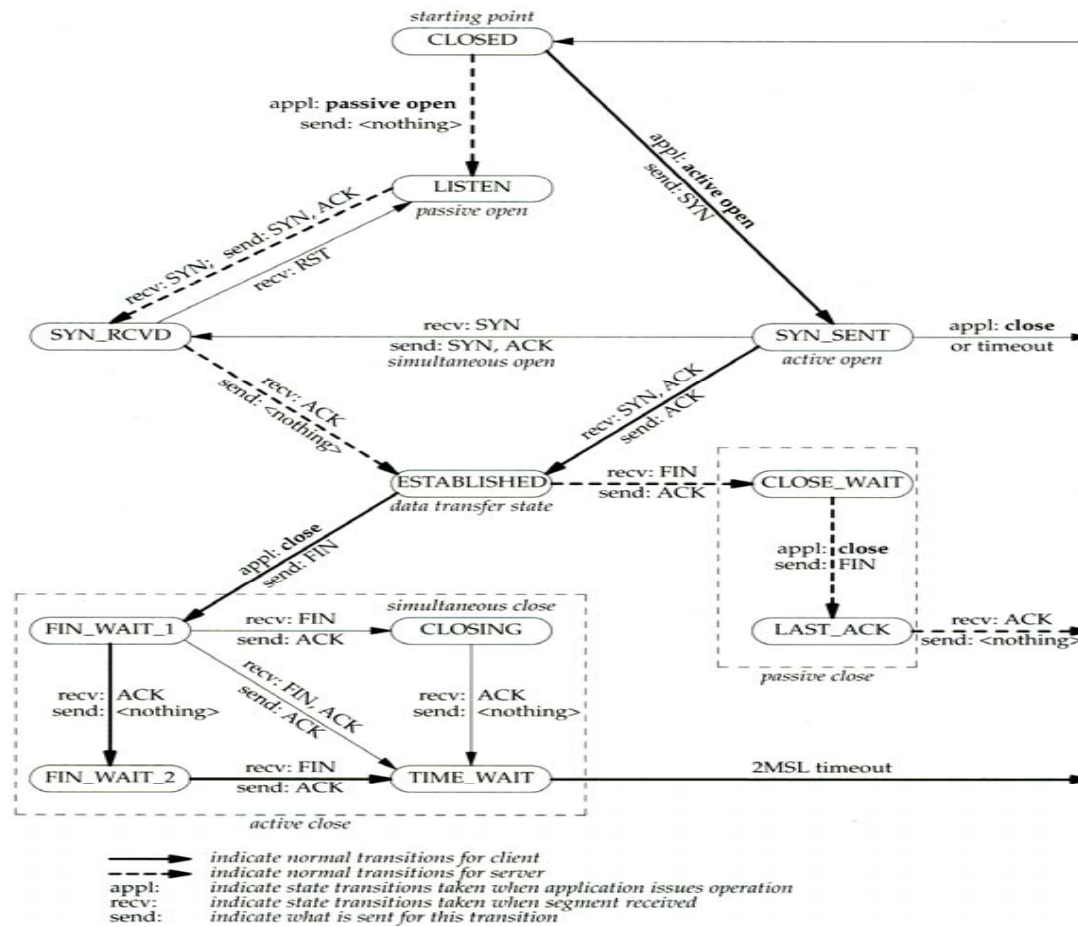


Figure 2.4 TCP state transition diagram.



For More Information

- RFC 793 – Transmission Control Protocol, Sep-01-1981
- RFC 2581 – TCP Congestion Control, April 1999
- RFC 3168 – The Addition of Explicit Congestion Notification (ECN) to IP, September 2001.
- RFC 3390 – Increasing TCP's Initial Window, October 2002
- RFC 3742 – Limited Slow-Start for TCP with Large Congestion Windows, March 2004
- RFC 4727 – Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers, November 2006
- TCP tutorials:
 - <http://www.garykessler.net/library/tcpip.html#TCP>
 - <http://cities.lk.net/tcp.html>