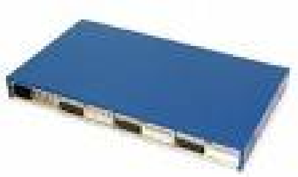


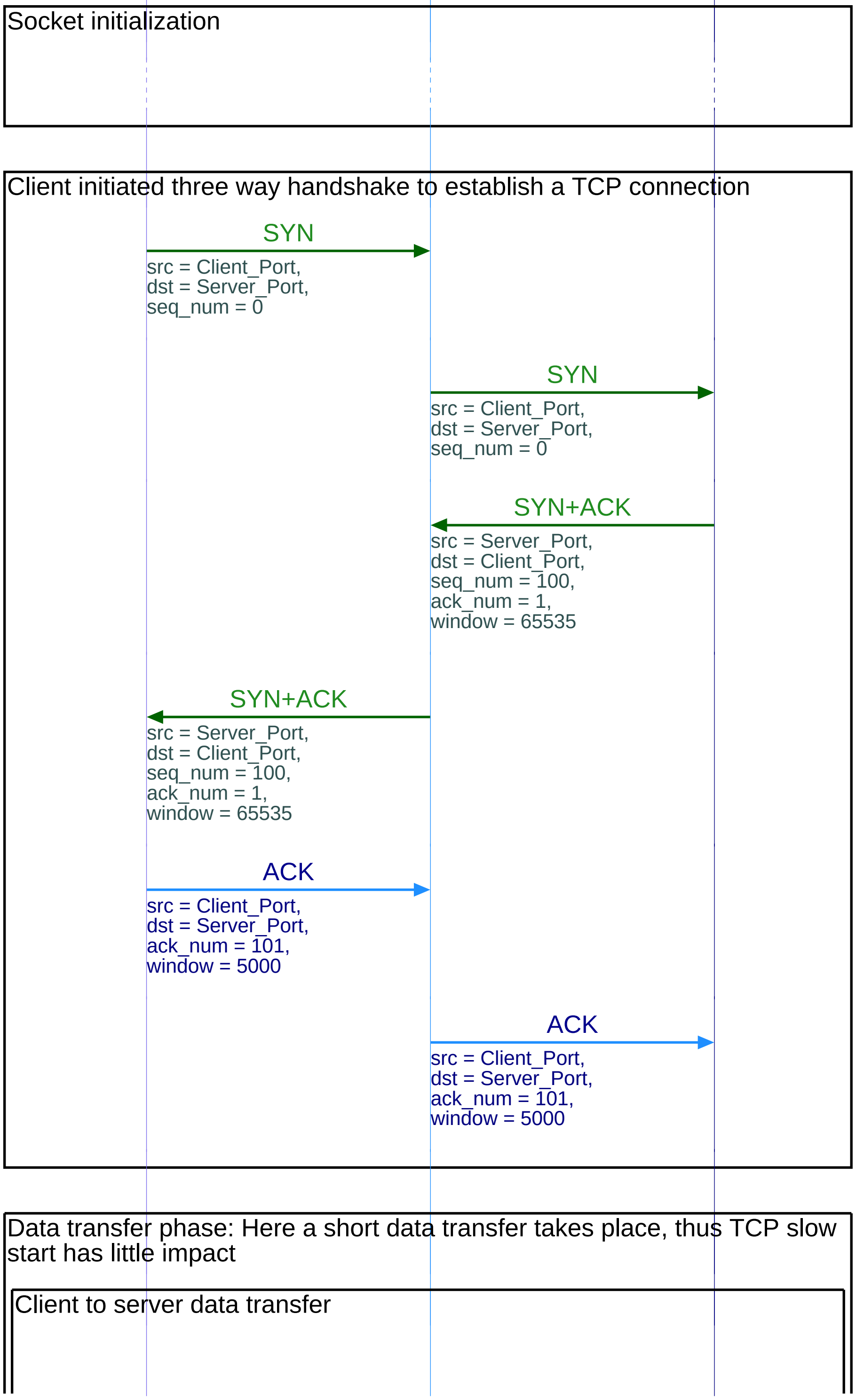
Client | Net | Server

This sequence diagram was generated with EventStudio System Designer (http://www.EventHelix.com/EventStudio).

TCP (Transmission Control Protocol) provides a reliable end to end service that delivers packets over the Internet. Packets are delivered in sequence without loss or duplication.

This sequence diagram explores following: (1) The three-way handshake to establish a TCP (2) Data transfer using the byte oriented sequence numbers (3) Release of a TCP connection.

The TCP socket creation and deletion on the server and client is also covered.



Socket initialization

Server awaits client socket connections.

Client initiated three way handshake to establish a TCP connection

SYN (Client → Net)
src = Client_Port,
dst = Server_Port,
seq_num = 0

Client sets the SYN bit in the TCP header to request a TCP connection. The sequence number field is set to 0. Since the SYN bit is set, this sequence number is used as the initial sequence number

SYN (Net → Server)
src = Client_Port,
dst = Server_Port,
seq_num = 0

SYN TCP segment is received by the server

SYN+ACK (Server → Net)
src = Server_Port,
dst = Client_Port,
seq_num = 100,
ack_num = 1,
window = 65535

Server sets the SYN and the ACK bits in the TCP header. Server sends its initial sequence number as 100. Server also sets its window to 65535 bytes. i.e. Server has buffer space for 65535 bytes of data. Also note that the ack sequence numer is set to 1. This signifies that the server expects a next byte sequence number of 1

SYN+ACK (Net → Client)
src = Server_Port,
dst = Client_Port,
seq_num = 100,
ack_num = 1,
window = 65535

Client receives the "SYN+ACK" TCP segment

ACK (Client → Net)
src = Client_Port,
dst = Server_Port,
ack_num = 101,
window = 5000

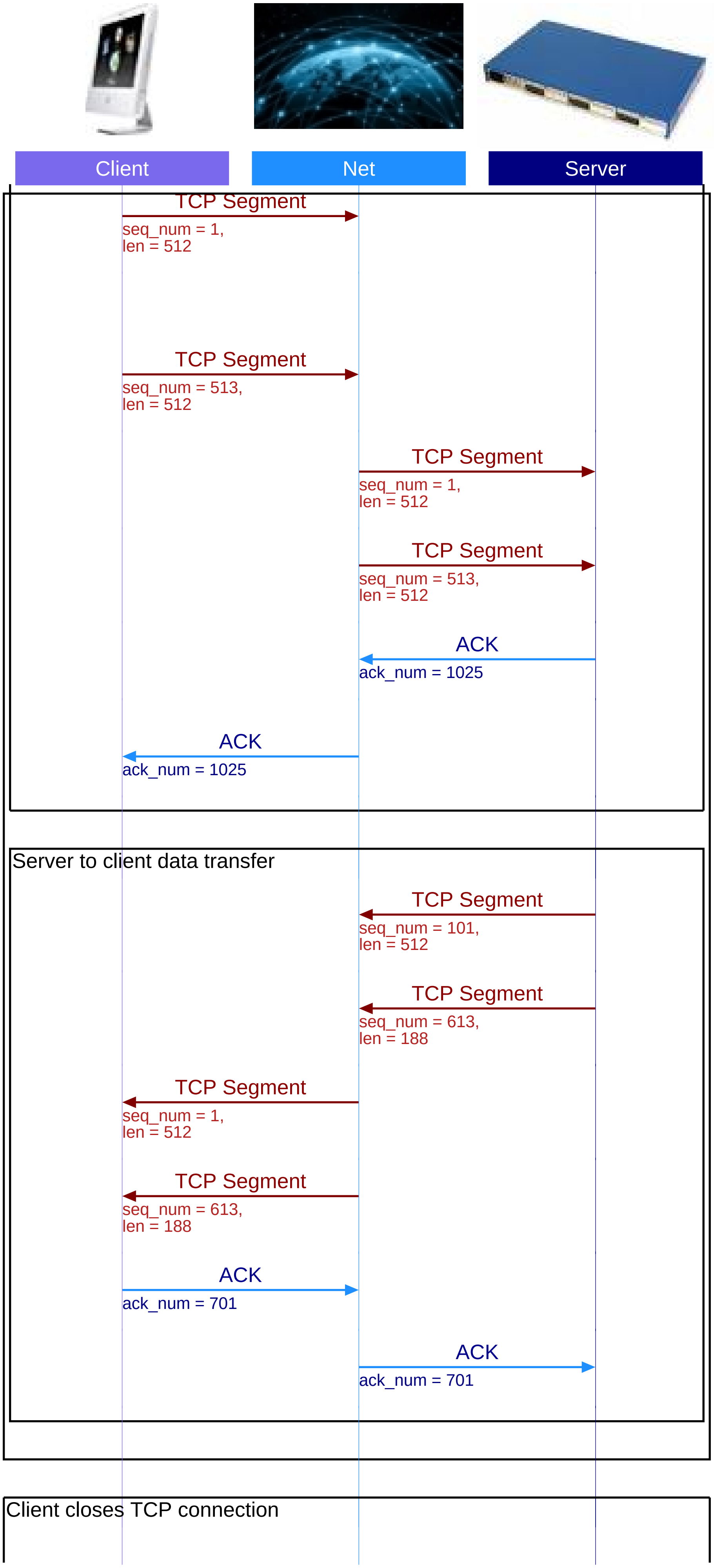
Client now acknowledges the first segment, thus completing the three way handshake. The receive window is set to 5000. Ack sequence number is set to 101, this means that the next expected sequence number is 101.

ACK (Net → Server)
src = Client_Port,
dst = Server_Port,
ack_num = 101,
window = 5000

Server receives the TCP ACK segment

Data transfer phase: Here a short data transfer takes place, thus TCP slow start has little impact

Client to server data transfer



Client | Net | Server

TCP Segment
seq_num = 1,
len = 512

The first TCP segment is sent with a sequence number of 1. This is the sequence number for the first byte in the segment.
(Note that unlike other protocols, TCP maintains sequence numbers at byte level. The sequence number field in the TCP header corresponds to the first byte in the segment.)

TCP Segment
seq_num = 513,
len = 512

Bytes in the first TCP segment correspond to 1 to 512 sequence numbers. Thus, the second TCP segment contains data starting with 513 sequence number

TCP Segment
seq_num = 1,
len = 512

TCP Segment
seq_num = 513,
len = 512

Server receives both the segments

ACK
ack_num = 1025

Server acknowledges the data segments with the next expected sequence number as 1025 (TCP typically sends an acknowledgement every two received segments)

ACK
ack_num = 1025

Server to client data transfer

TCP Segment
seq_num = 101,
len = 512

TCP Segment
seq_num = 613,
len = 188

TCP Segment
seq_num = 1,
len = 512

TCP Segment
seq_num = 613,
len = 188

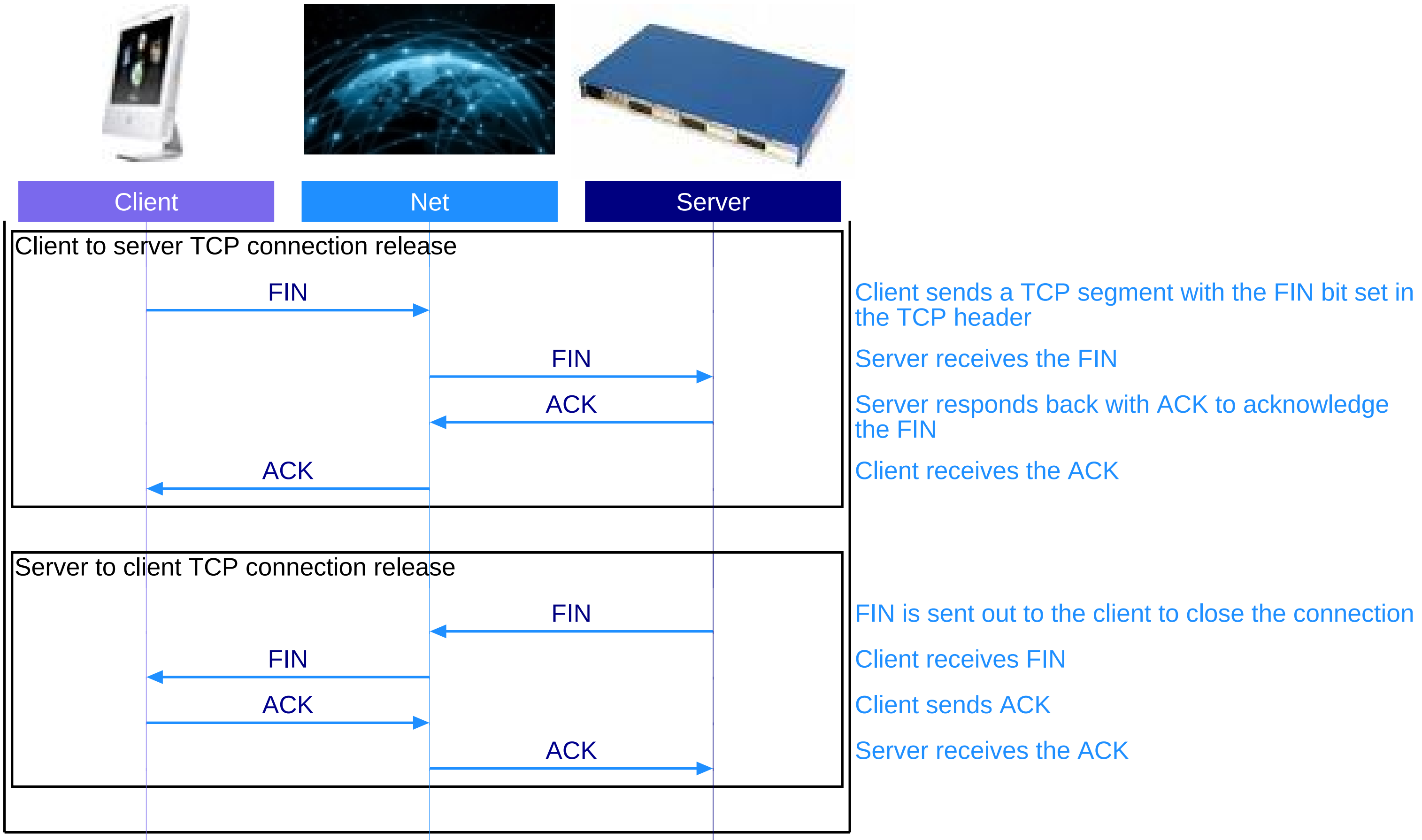
Client has received both the TCP segments

ACK
ack_num = 701

Client sends a TCP ACK with the next expected sequence number set to 701

ACK
ack_num = 701

Client closes TCP connection



Client to server TCP connection release

FIN — Client sends a TCP segment with the FIN bit set in the TCP header

FIN — Server receives the FIN

ACK — Server responds back with ACK to acknowledge the FIN

ACK — Client receives the ACK

Server to client TCP connection release

FIN — FIN is sent out to the client to close the connection

FIN — Client receives FIN

ACK — Client sends ACK

ACK — Server receives the ACK

This sequence diagram was generated with EventStudio System Designer (http://www.EventHelix.com/EventStudio).