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

TCP is an end to end protocol which operates over the heterogeneous Internet. TCP has no advance knowledge of the network characteristics, thus it has to adjust its behavior according to the current state of the network. TCP has built in support for congestion control. Congestion control ensures that TCP does not pump data at a rate higher than what the network can handle.

In this sequence diagram we will analyse "Slow start", an important part of the congestion control mechanisms built right into TCP. As the name suggests, "Slow Start" starts slowly, increasing its window size as it gains confidence about the networks throughput.

Socket initialization

Server awaits client socket connections.

Client initiated three way handshake to establish a TCP connection

SYN
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
src = Client_Port,
dst = Server_Port,
seq_num = 0

SYN TCP segment is received by the server

SYN+ACK
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
src = Server_Port,
dst = Client_Port,
seq_num = 100,
ack_num = 1,
window = 65535

Client receives the "SYN+ACK" TCP segment

ACK
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.

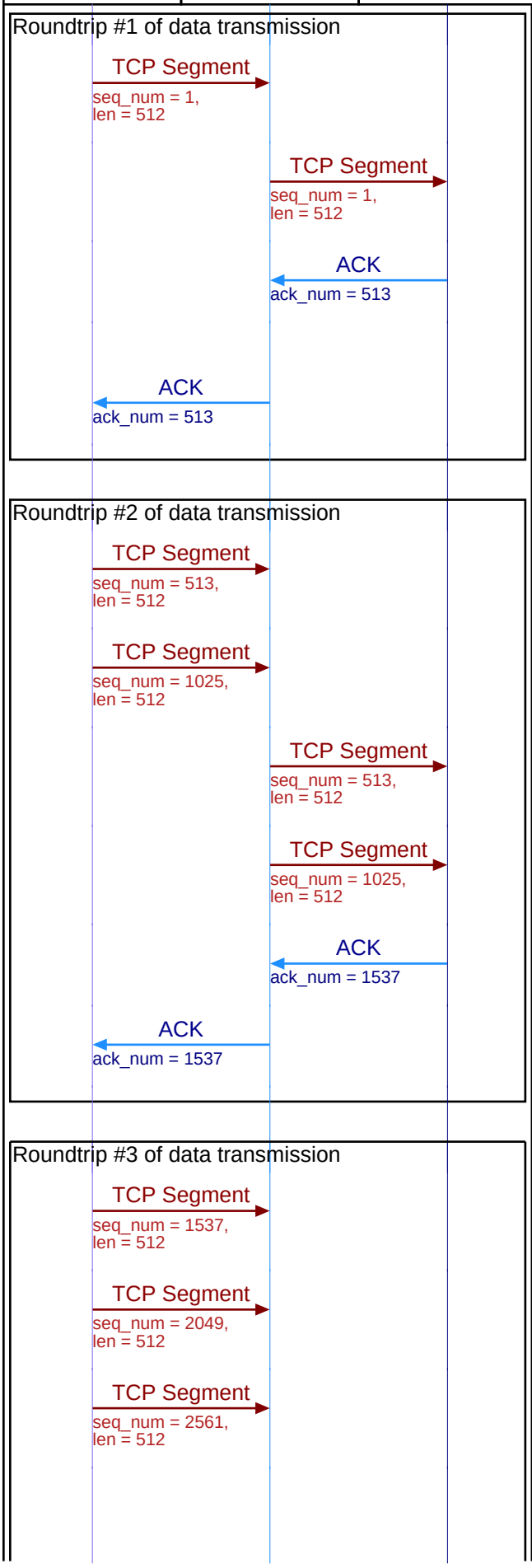
ACK
src = Client_Port,
dst = Server_Port,
ack_num = 101,
window = 5000

Server receives the TCP ACK segment

A TCP connection starts in the "Slow Start" state. In this state, TCP adjusts its transmission rate based on the rate at which the acknowledgements are received from the other end.

TCP Slow start is implemented using two variables, viz cwnd (Congestion Window)and ssthresh (Slow Start Threshold). cwnd is a self imposed transmit window restriction at the sender end. cwnd will increase as TCP gains more confidence on the network's ability to handle traffic. ssthresh is the threshold for determining the point at which TCP exits slow start. If cwnd increases beyond ssthresh, the TCP session in that direction is considered to be out of slow start phase

TCP slow start



The first TCP segment is sent with a sequence number of 1. This is the sequence number for the first byte in the segment.

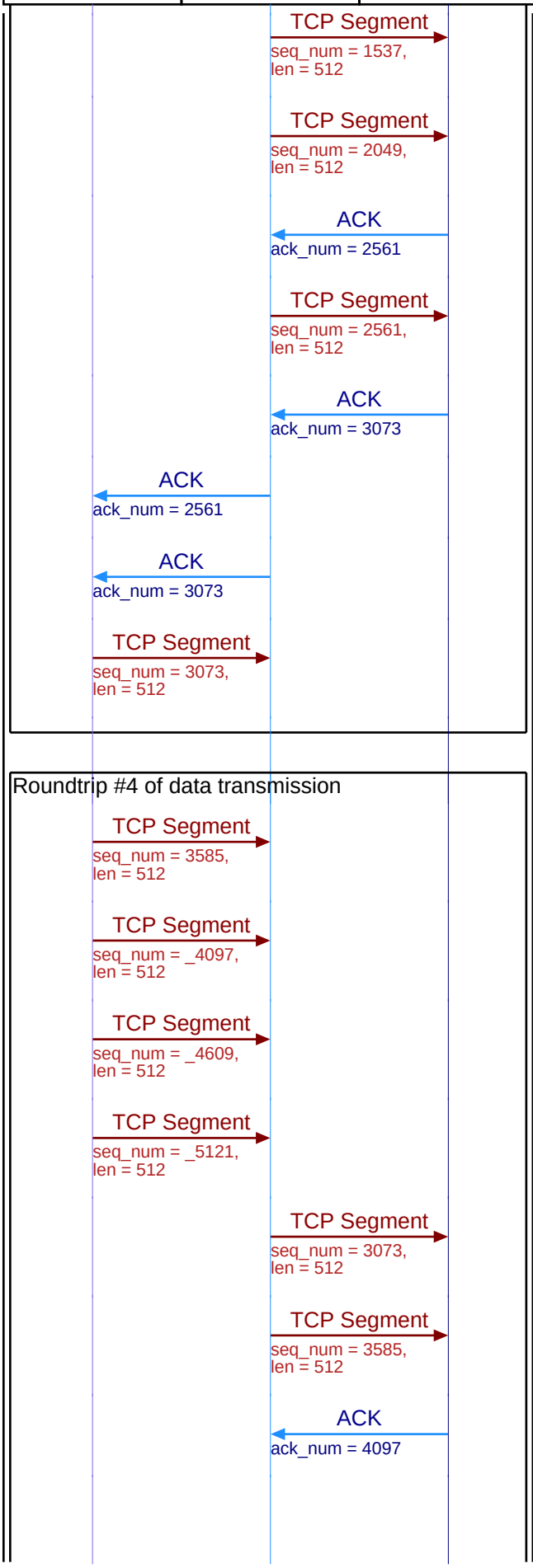
Server acknowledges the data segments with the next expected sequence number as 513
TCP typically sends an acknowledgement every two received segments but in this case it times out for another segment and decides to acknowledge the only segment received.

Client receives the acknowledgement for the first TCP data segment

Since the congestion window has increased to 2, TCP can now send two segments without waiting for an ack

Receiver generates a TCP ACK on receiving the two segments

Now three segments can be sent without waiting for an ack



Network delivers the three segments to the destination server

TCP acknowledges receipt of two segments

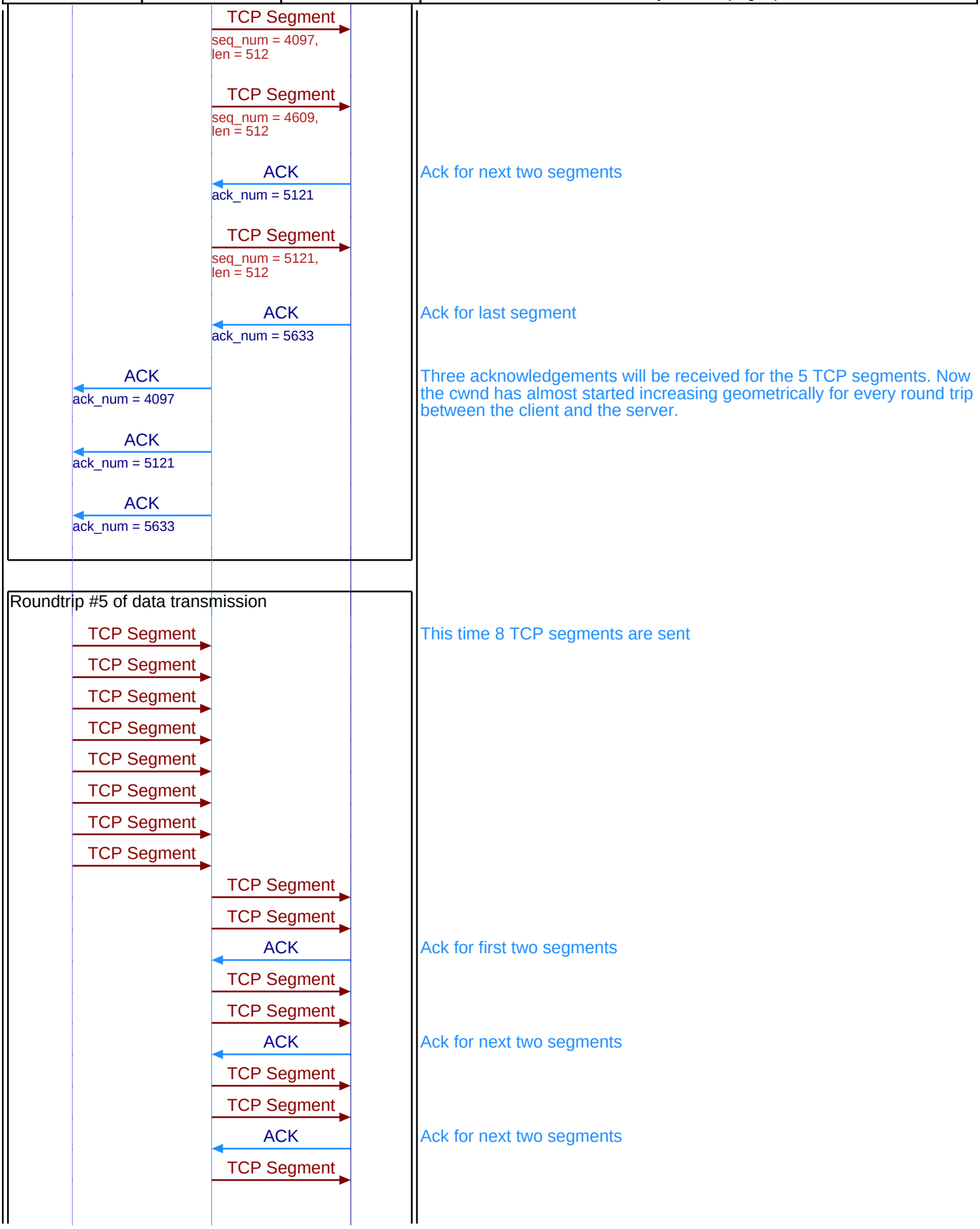
TCP times for another segment and acknowledges the only pending segment

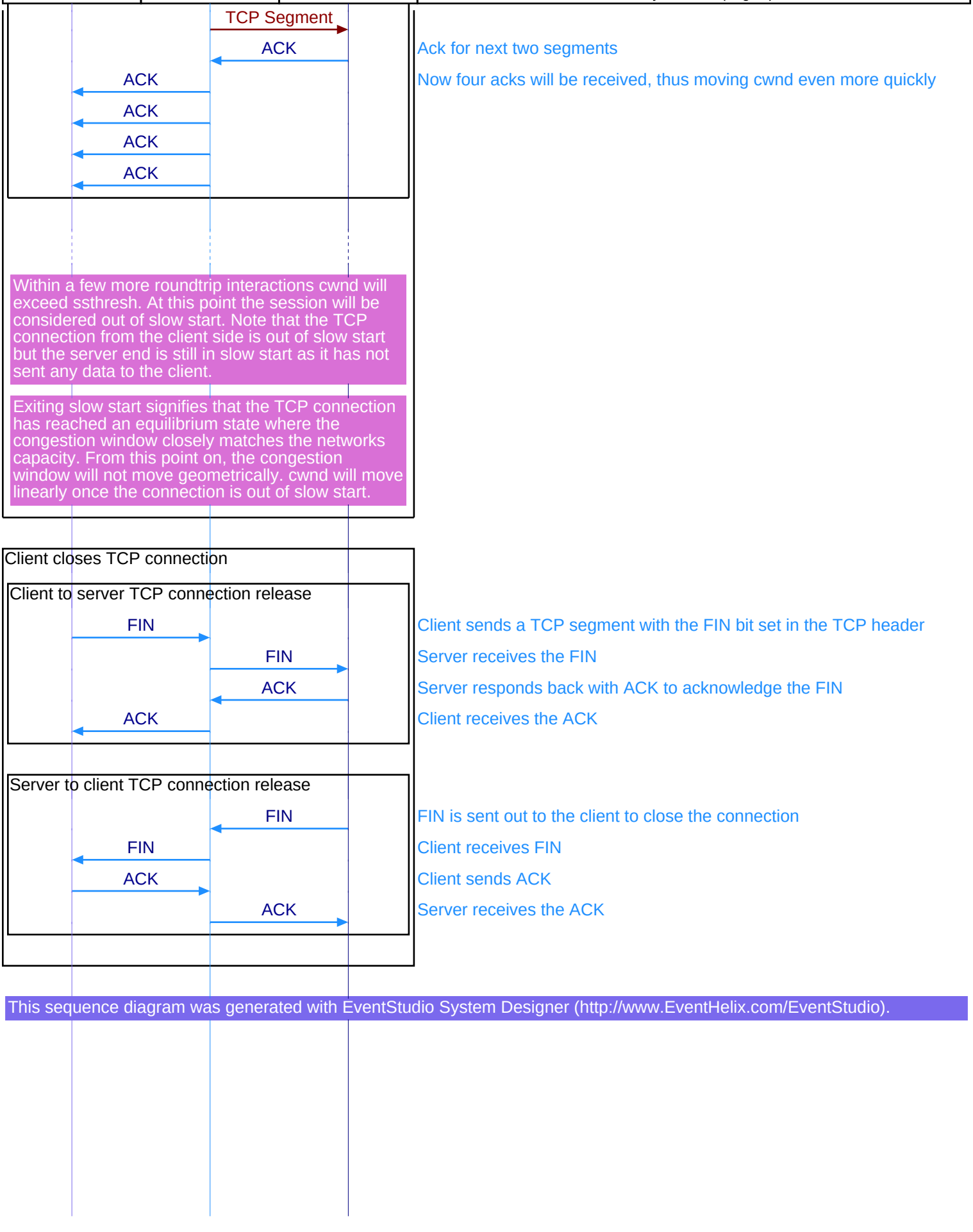
The TCP acknowlegements again increment cwnd. This time two acks are received, so cwnd will get incremented by 2

Since cwnd has reached 5 segments, TCP is allowed to send 5 segments without waiting for the ack

The 5 segments are received by the destination server

TCP Ack is sent after first two segments


TCP Segment
seq_num = 4097, len = 512
TCP Segment
seq_num = 4609, len = 512
ACK
ack_num = 5121
Ack for next two segments
TCP Segment
seq_num = 5121, len = 512
ACK
ack_num = 5633
Ack for last segment
ACK
ack_num = 4097
ACK
ack_num = 5121
ACK
ack_num = 5633
Three acknowledgements will be received for the 5 TCP segments. Now the cwnd has almost started increasing geometrically for every round trip between the client and the server.
Roundtrip #5 of data transmission
TCP Segment
TCP Segment
TCP Segment
TCP Segment
TCP Segment
TCP Segment
TCP Segment
TCP Segment
This time 8 TCP segments are sent
TCP Segment
TCP Segment
ACK
Ack for first two segments
TCP Segment
TCP Segment
ACK
Ack for next two segments
TCP Segment
TCP Segment
ACK
Ack for next two segments
TCP Segment


TCP Segment
ACK
Ack for next two segments
ACK
Now four acks will be received, thus moving cwnd even more quickly
ACK
ACK
ACK
Within a few more roundtrip interactions cwnd will exceed ssthresh. At this point the session will be considered out of slow start. Note that the TCP connection from the client side is out of slow start but the server end is still in slow start as it has not sent any data to the client.
Exiting slow start signifies that the TCP connection has reached an equilibrium state where the congestion window closely matches the networks capacity. From this point on, the congestion window will not move geometrically. cwnd will move linearly once the connection is out of slow start.
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