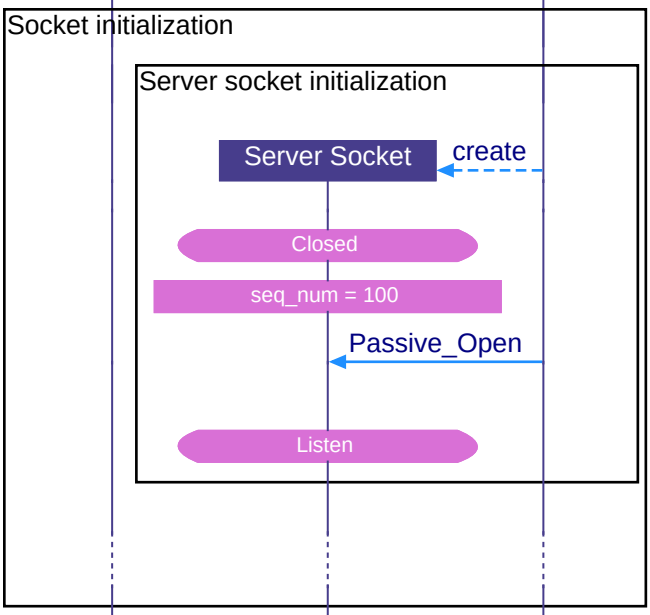
This sequence diagram was generated with EventStudio System Designer (http://www.EventHelix.com/EventStudio).

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.



Server Application creates a Socket

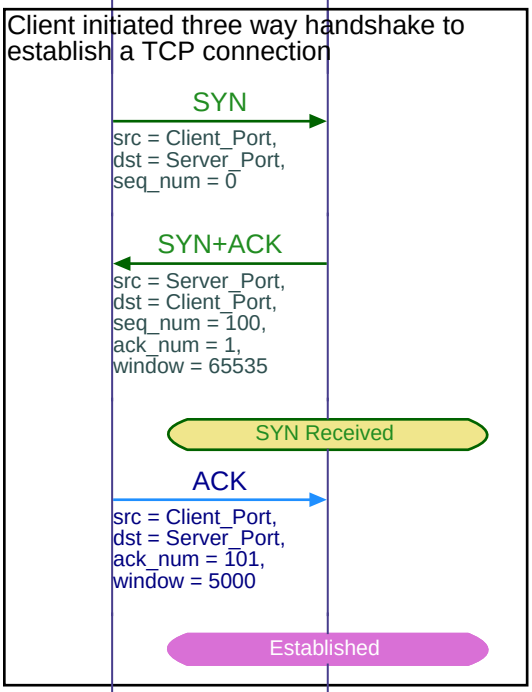
The Socket is created in Closed state

Server sets the initial sequence number to 100

Server application has initiated a passive open. In this mode, the socket does not attempt to establish a TCP connection. The socket listens for TCP connection request from clients

Socket transitions to the Listen state

Server awaits client socket connections.



SYN TCP segment is received by the server

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

Now the server transitions to the SYN Received state

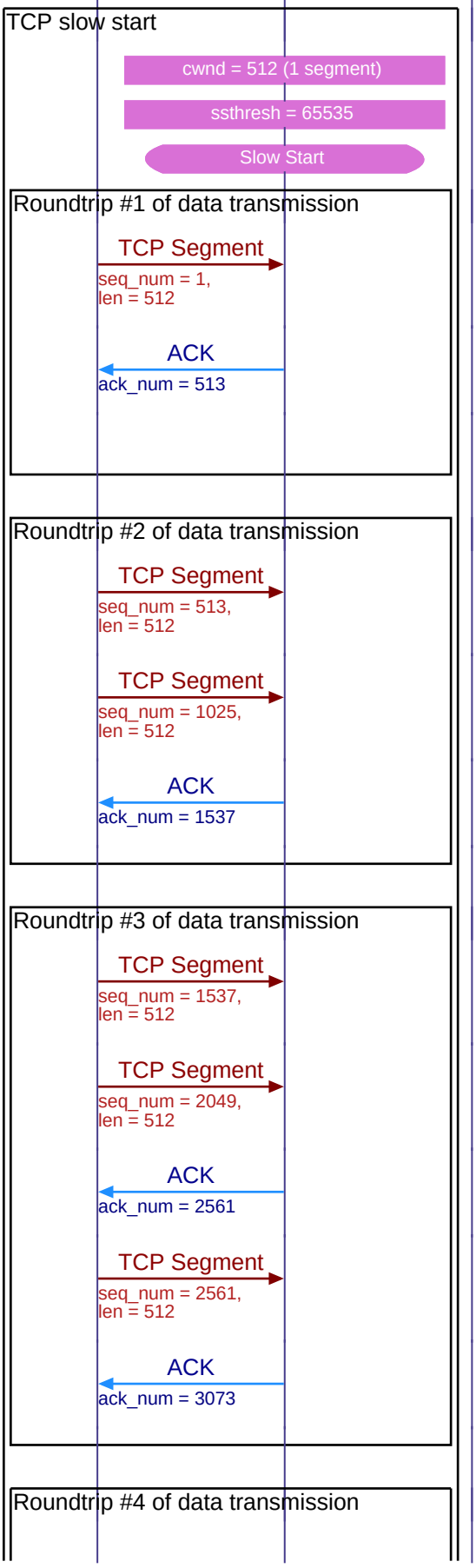
Server receives the TCP ACK segment

Now the server too moves to the Established state

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



By the same logic, the server also sets cwnd to 512

Server end TCP connection moves to slow start state

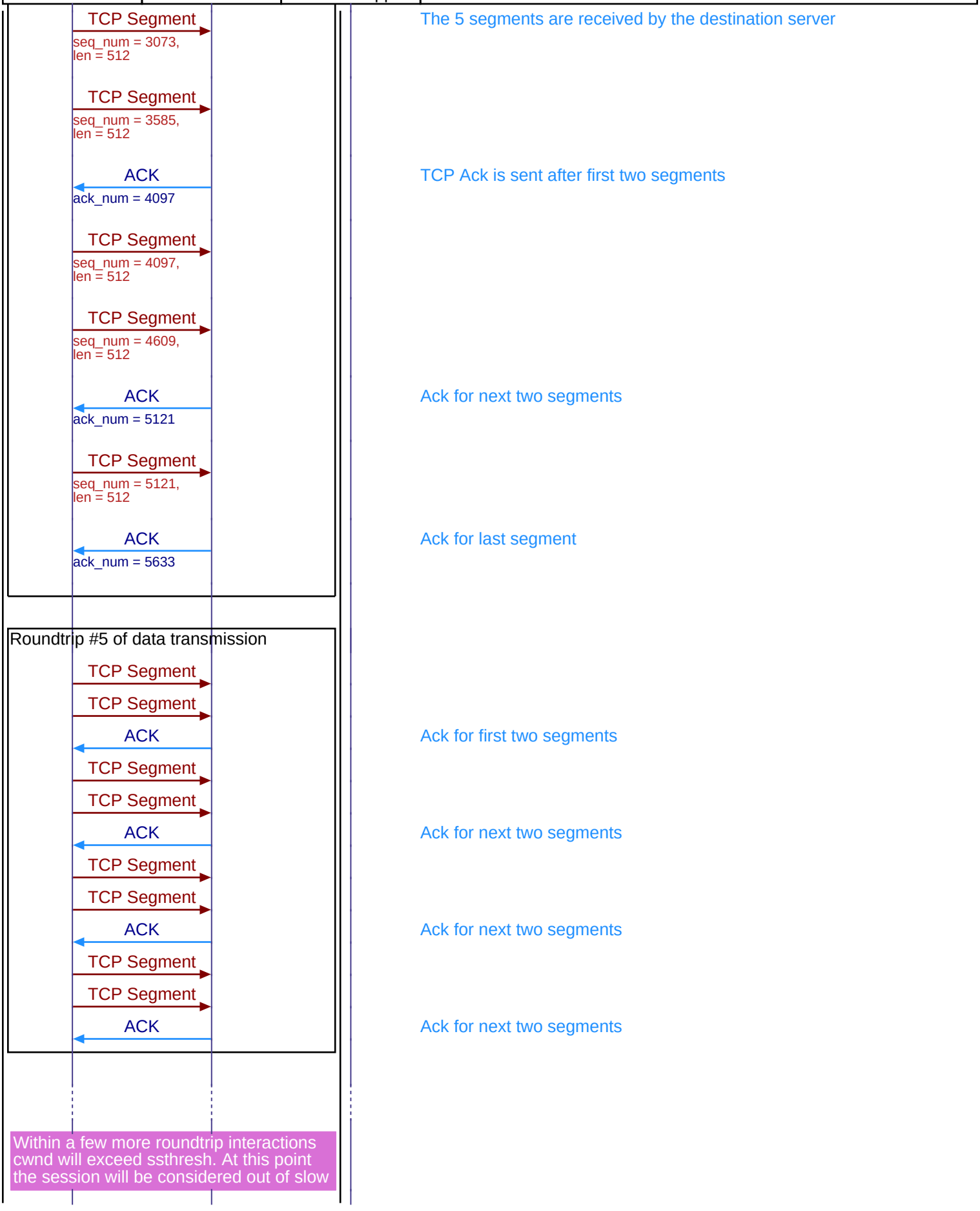
Server acknowledges the data segments with the next expected sequence number as 513
TCP typically sends an acknowledgement every two received segments but in this case it times out for another segment and decides to acknowledge the only segment received.

Receiver generates a TCP ACK on receiving the two segments

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


TCP Segment
seq_num = 3073, len = 512
TCP Segment
seq_num = 3585, len = 512
ACK
ack_num = 4097
TCP Segment
seq_num = 4097, len = 512
TCP Segment
seq_num = 4609, len = 512
ACK
ack_num = 5121
TCP Segment
seq_num = 5121, len = 512
ACK
ack_num = 5633
Roundtrip #5 of data transmission
TCP Segment
TCP Segment
ACK
TCP Segment
TCP Segment
ACK
TCP Segment
TCP Segment
ACK
TCP Segment
TCP Segment
ACK
Within a few more roundtrip interactions cwnd will exceed ssthresh. At this point the session will be considered out of slow
The 5 segments are received by the destination server
TCP Ack is sent after first two segments
Ack for next two segments
Ack for last segment
Ack for first two segments
Ack for next two segments
Ack for next two segments
Ack for next two segments

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

Server receives the FIN

ACK

Server responds back with ACK to acknowledge the FIN

Close Wait

Server changes state to Close Wait. In this state the server waits for the server application to close the connection

Server to client TCP connection release

Close

Server application closes the TCP connection

FIN

FIN is sent out to the client to close the connection

Last Ack

Server changes state to Last Ack. In this state the last acknowledgement from the client will be received

ACK

Server receives the ACK

Closed

Server moves the connection to closed state

delete