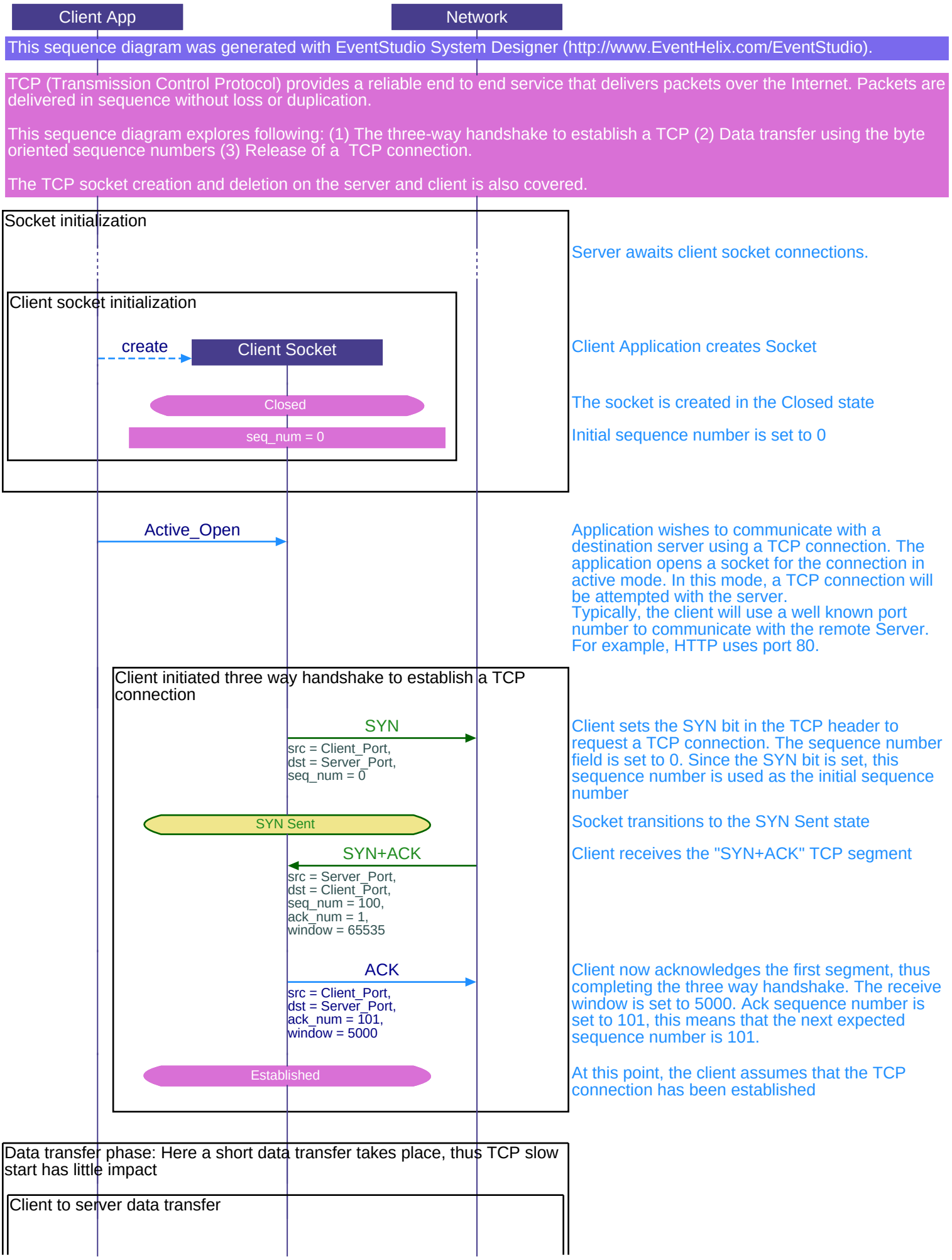
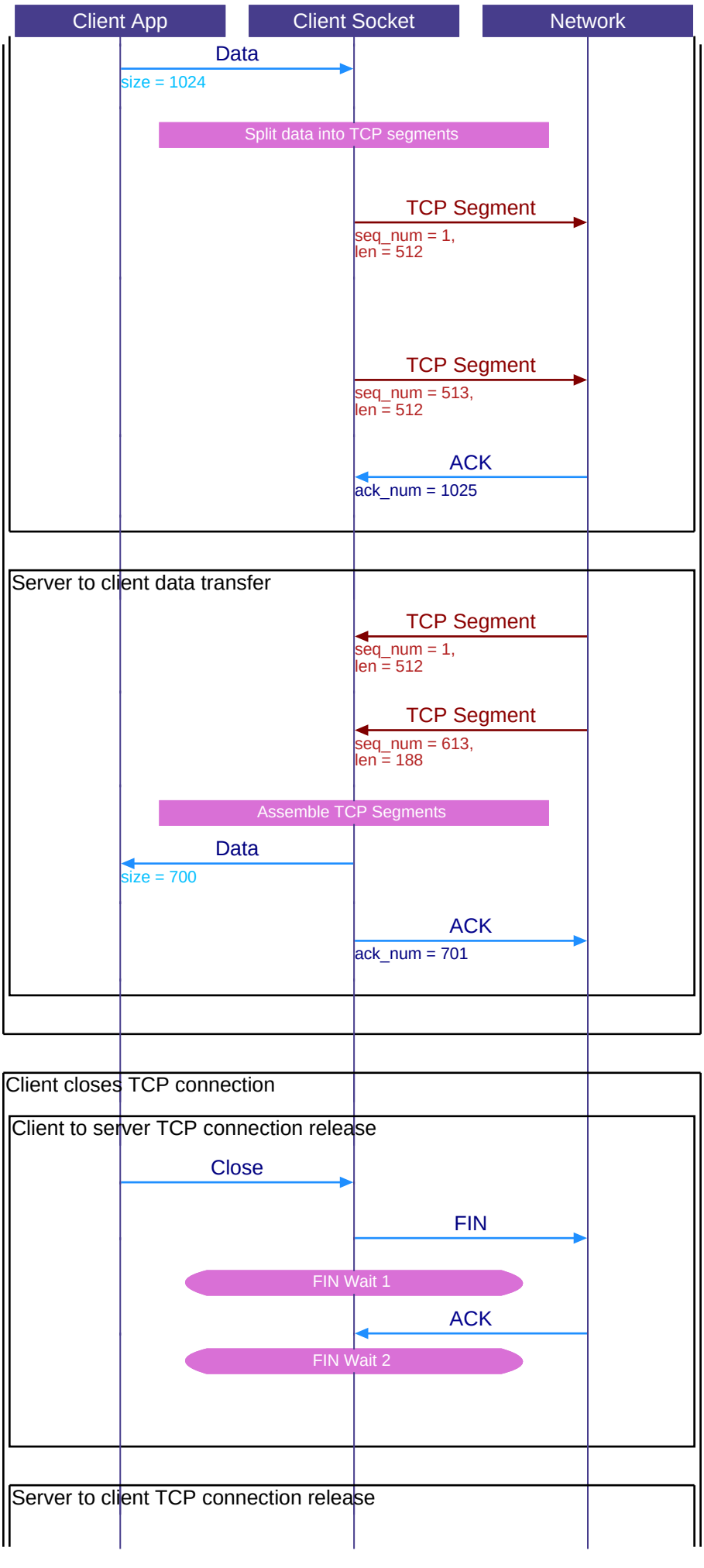

Client App
Network
This sequence diagram was generated with EventStudio System Designer (http://www.EventHelix.com/EventStudio).
TCP (Transmission Control Protocol) provides a reliable end to end service that delivers packets over the Internet. Packets are delivered in sequence without loss or duplication.
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 socket initialization
create
Client Socket
Client Application creates Socket
Closed
The socket is created in the Closed state
seq_num = 0
Initial sequence number is set to 0
Active_Open
Application wishes to communicate with a destination server using a TCP connection. The application opens a socket for the connection in active mode. In this mode, a TCP connection will be attempted with the server.
Typically, the client will use a well known port number to communicate with the remote Server. For example, HTTP uses port 80.
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 Sent
Socket transitions to the SYN Sent state
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.
Established
At this point, the client assumes that the TCP connection has been established
Data transfer phase: Here a short data transfer takes place, thus TCP slow start has little impact
Client to server data transfer



Client application sends 1024 bytes of data to the socket

This TCP connection limits TCP segments to 512 bytes, thus the received data is split into 2 TCP segments

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.)

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

Client has received both the TCP segments

Socket passes data to Client application

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

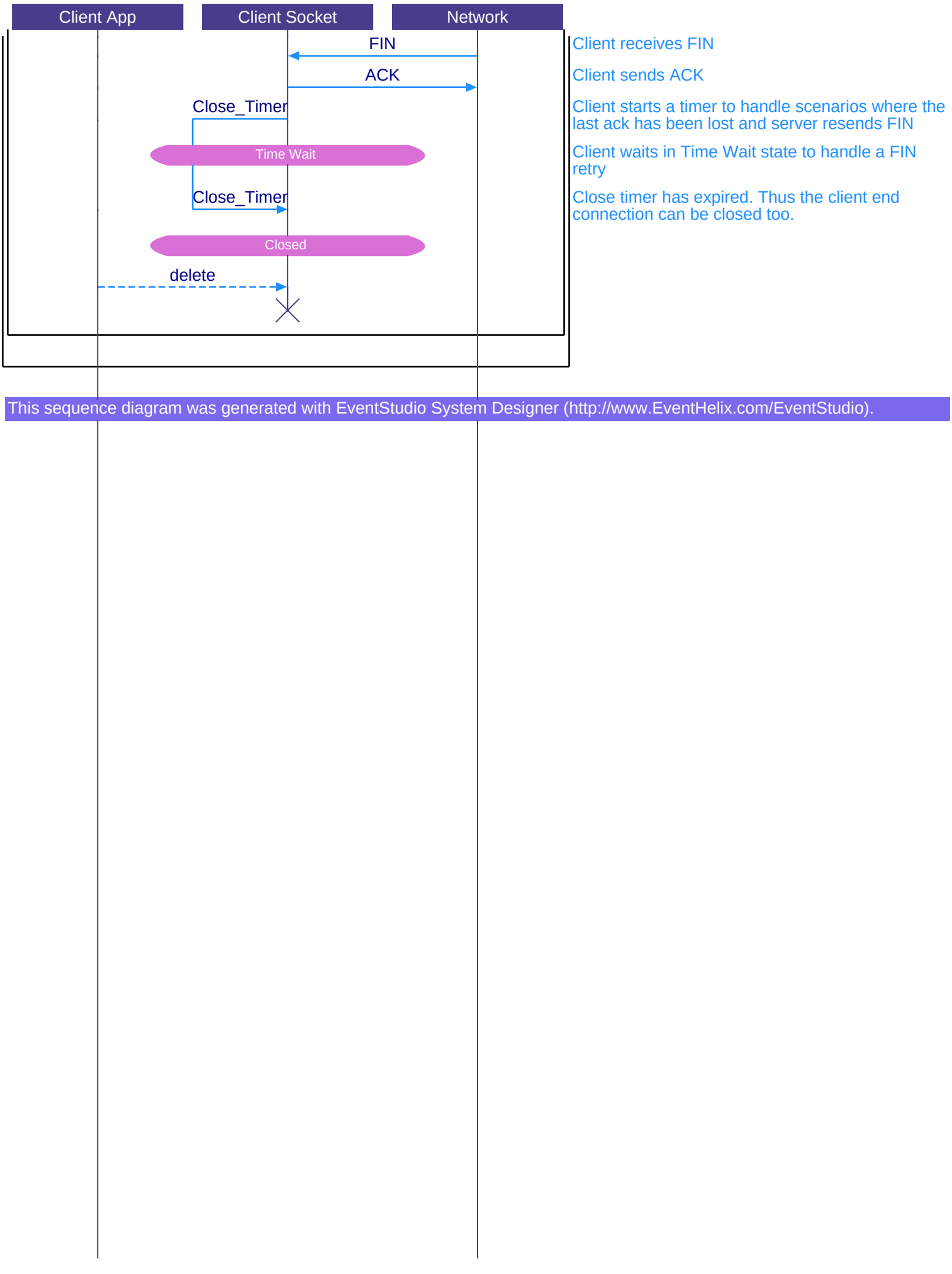
Client application wishes to release the TCP connection

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

Client changes state to FIN Wait 1 state

Client receives the ACK

Client changes state to FIN Wait 2. In this state, the TCP connection from the client to server is closed. Client now waits close of TCP connection from the server end


Client App
Client Socket
Network
FIN
ACK
Close_Timer
Time Wait
Close_Timer
Closed
delete
Client receives FIN
Client sends ACK
Client starts a timer to handle scenarios where the last ack has been lost and server resends FIN
Client waits in Time Wait state to handle a FIN retry
Close timer has expired. Thus the client end connection can be closed too.
This sequence diagram was generated with EventStudio System Designer (http://www.EventHelix.com/EventStudio).