State Machine Design

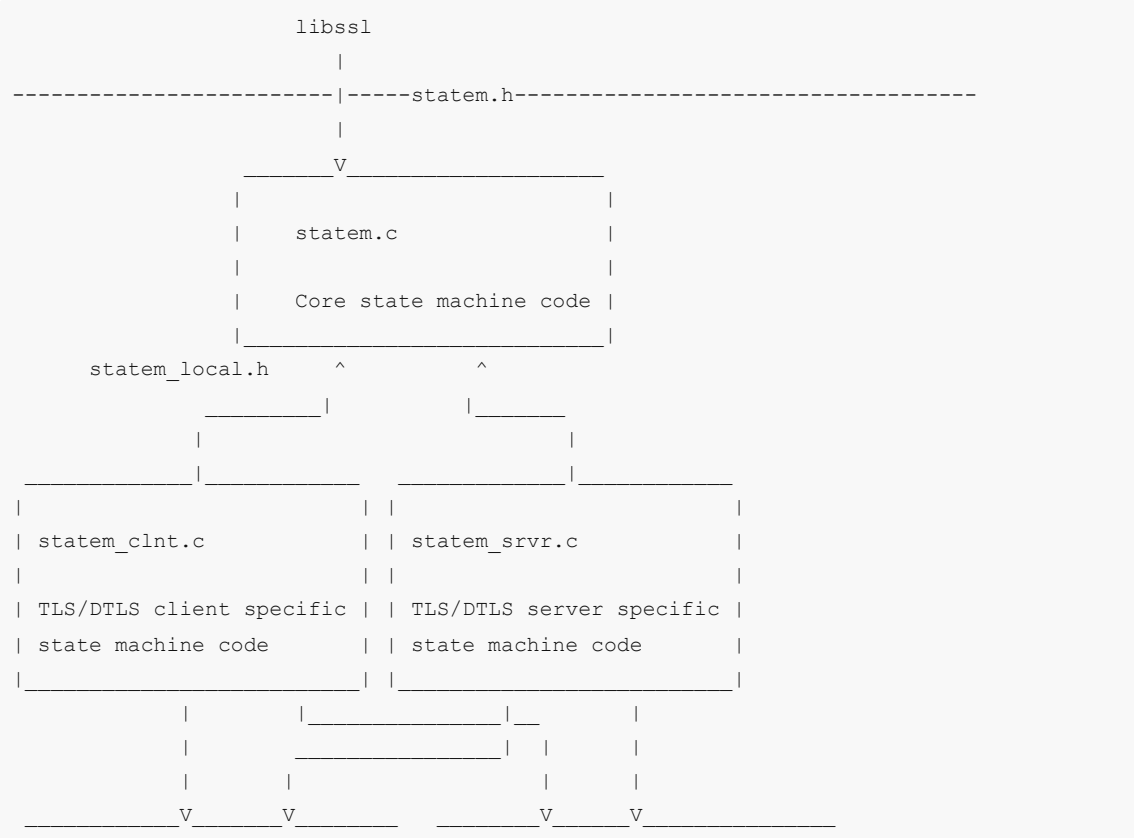
This file provides some guidance on the thinking behind the design of the state machine code to aid future maintenance.

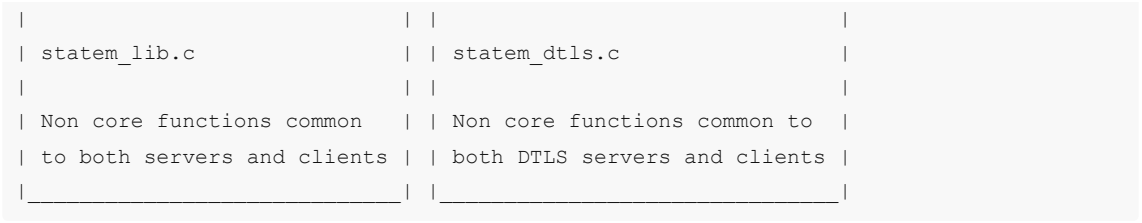
The state machine code replaces an older state machine present in OpenSSL versions 1.0.2 and below. The new state machine has the following objectives:

- Remove duplication of state code between client and server
- Remove duplication of state code between TLS and DTLS
- Simplify transitions and bring the logic together in a single location so that it is easier to validate
- Remove duplication of code between each of the message handling functions
- Receive a message first and then work out whether that is a valid transition - not the other way around (the other way causes lots of issues where we are expecting one type of message next but actually get something else)
- Separate message flow state from handshake state (in order to better understand each)
    - message flow state = when to flush buffers; handling restarts in the event of NBIO events; handling the common flow of steps for reading a message and the common flow of steps for writing a message etc
    - handshake state = what handshake message are we working on now
- Control complexity: only the state machine can change state: keep all the state changes local to the state machine component

The message flow state machine is divided into a reading sub-state machine and a writing sub-state machine. See the source comments in statem.c for a more detailed description of the various states and transitions possible.

Conceptually the state machine component is designed as follows:




statem_lib.c
Non core functions common to both servers and clients
statem_dtls.c
Non core functions common to both DTLS servers and clients