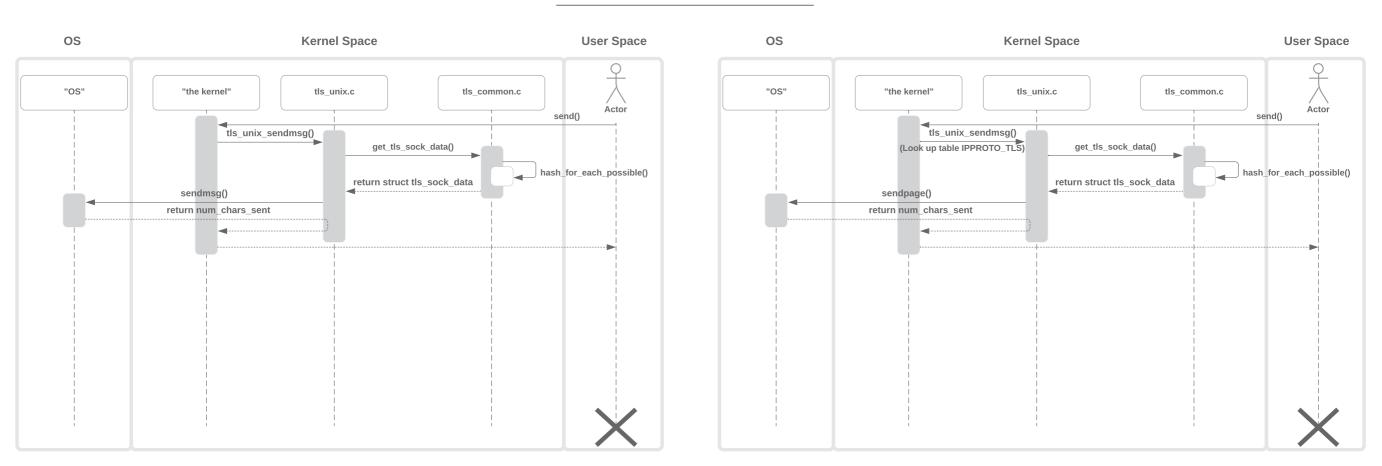
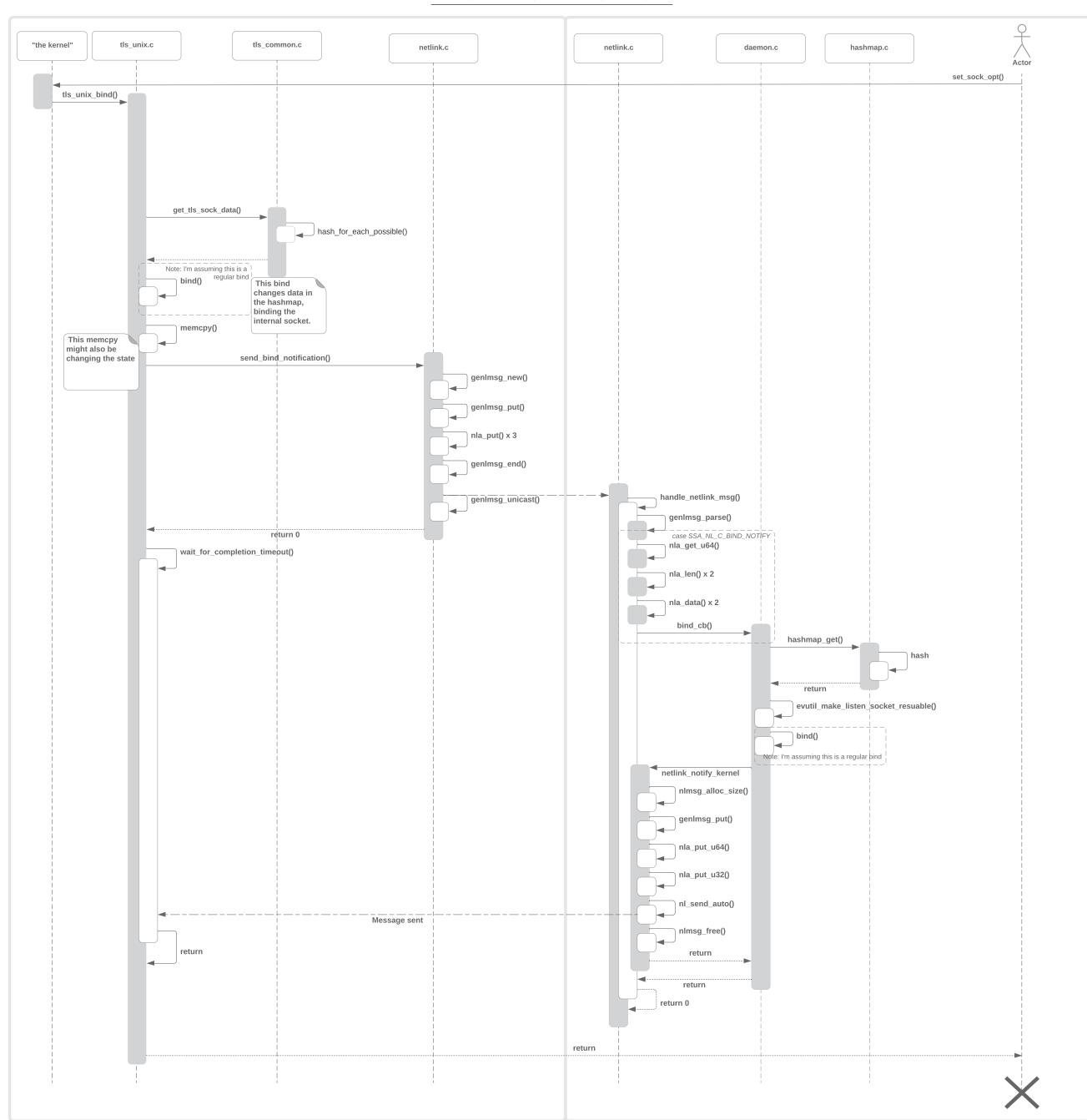


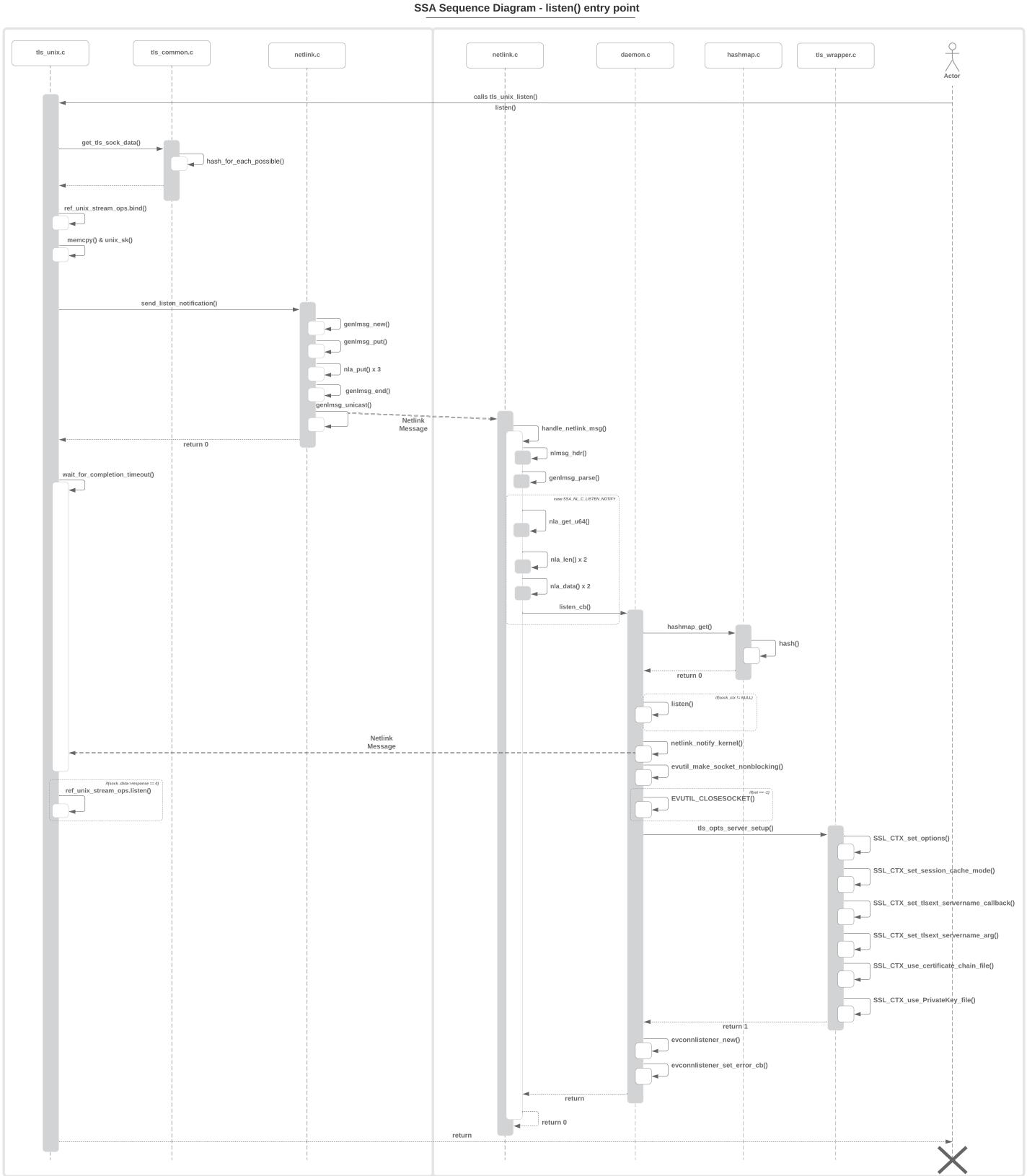
SSA Sequence Diagram - connect() entry point hashmap.c netlink.c netlink.c "the kernel" daemon.c tls\_wrapper.c tls\_unix.c tls\_common.c connect() tls\_unix\_connect() get\_tls\_sock\_data() return struct tls\_sock\_data bind() sizeof() memcpy() unix\_sk() send\_connect\_notification() wait\_for\_completion\_timeout() genlmsg\_new() genImsg\_put() genImsg\_end() genlmsg\_unicast() handle\_netlink\_msg() nlmsg\_data() genlmsg\_parse() case SSA\_NL\_C\_CONNECT\_NOTIFY
nla\_get\_u64() nla\_get\_u32() ntohs() tls\_opts\_client\_setup() SSL\_CTX\_set\_options() SSL\_CTX\_set\_verify() return 1 tls\_client\_wrapper\_setup() new\_tls\_conn\_ctx() tls\_client\_setup() SSL\_CTX\_set\_client\_cert\_cb() SSL\_set\_tlsext\_host\_name() SSL\_force\_post\_handshake\_auth() SSL\_set\_ex\_data() bufferevent\_socket\_new if (is\_accepting == 1) //TLS server bufferevent\_openssl\_socket\_new() bufferevent\_openssl\_socket\_new() bufferevent\_openssl\_socket\_new() bufferevent\_setcb() bufferevent\_enable() bufferevent\_setcb() return ctx set\_netlink\_cb\_params() bufferevent\_socket\_connect() if (sock\_ctx->is\_connected == 0)
netlink\_notify\_kernel() hashmap\_add() hash() return 0 if (blocking == 0)
netlink\_notify\_kernel() nlmsg\_alloc\_size() genImsg\_put() nla\_put\_u64() nla\_put\_u32() nl\_send\_auto() nlmsg\_free() return -EHOSTUNREACH return if (sock\_data->response != 0)

return sock\_data->response return 0 return ret return 0 return

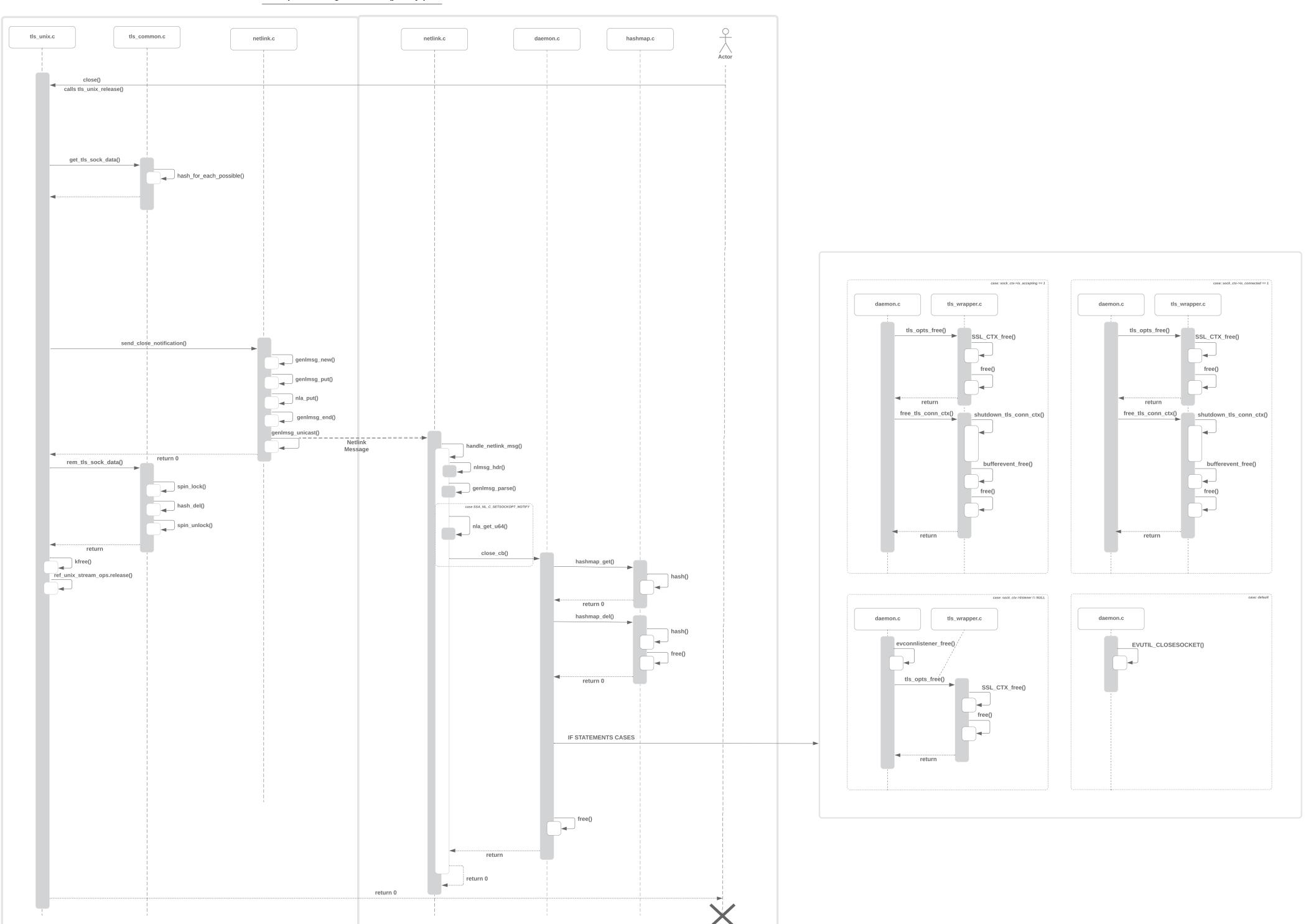
## SSA Sequence Diagram - send() entry point

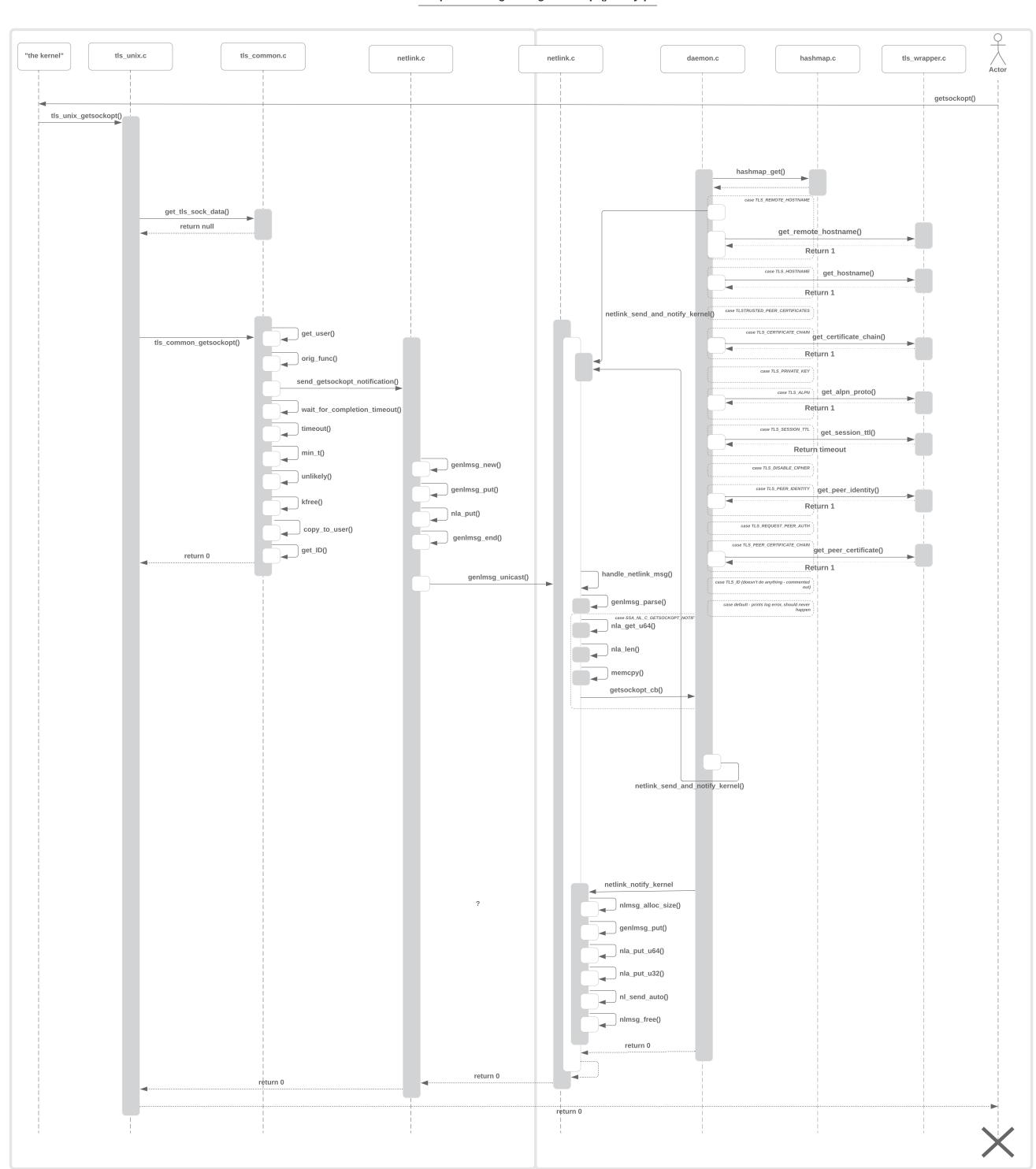


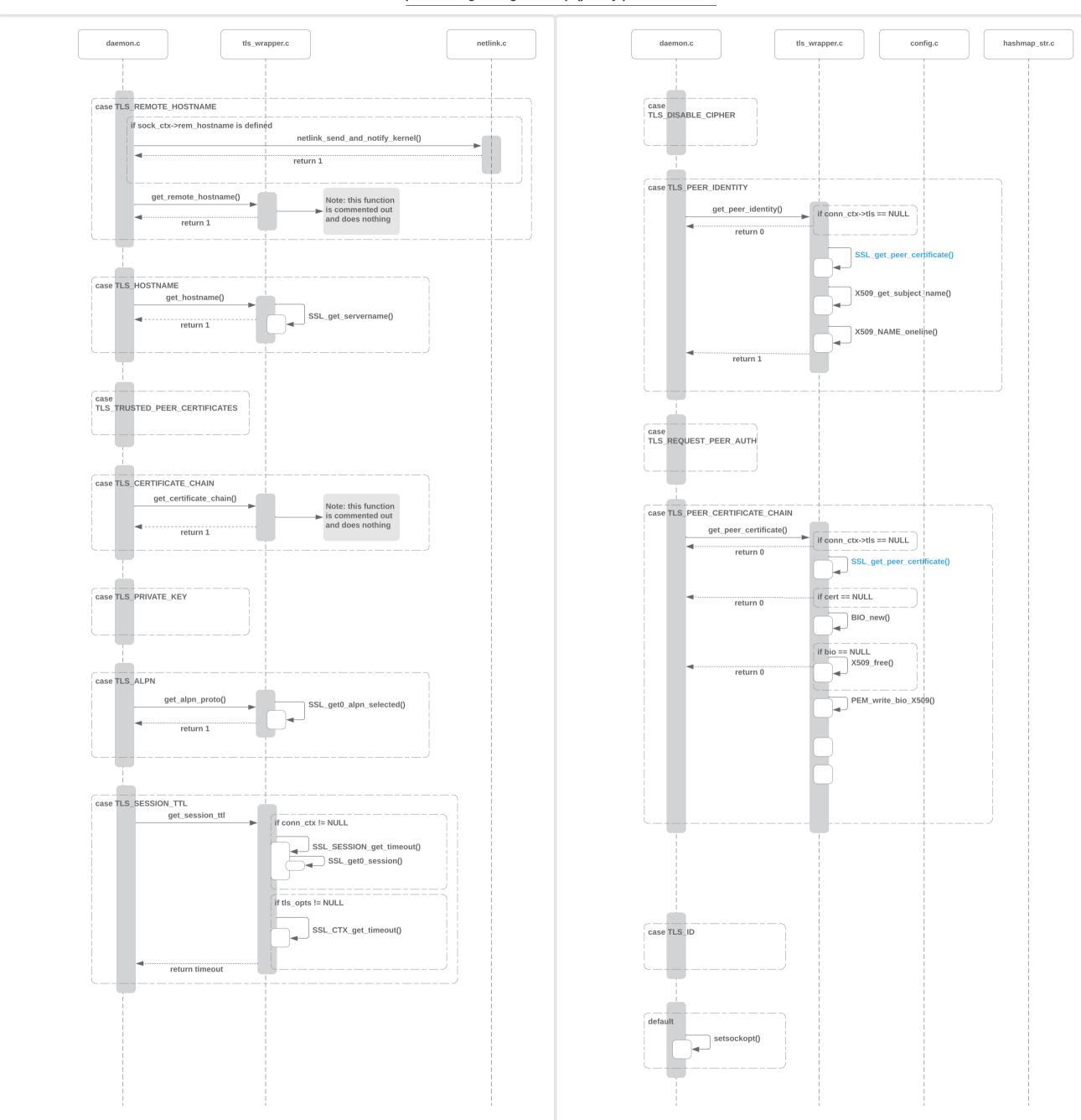


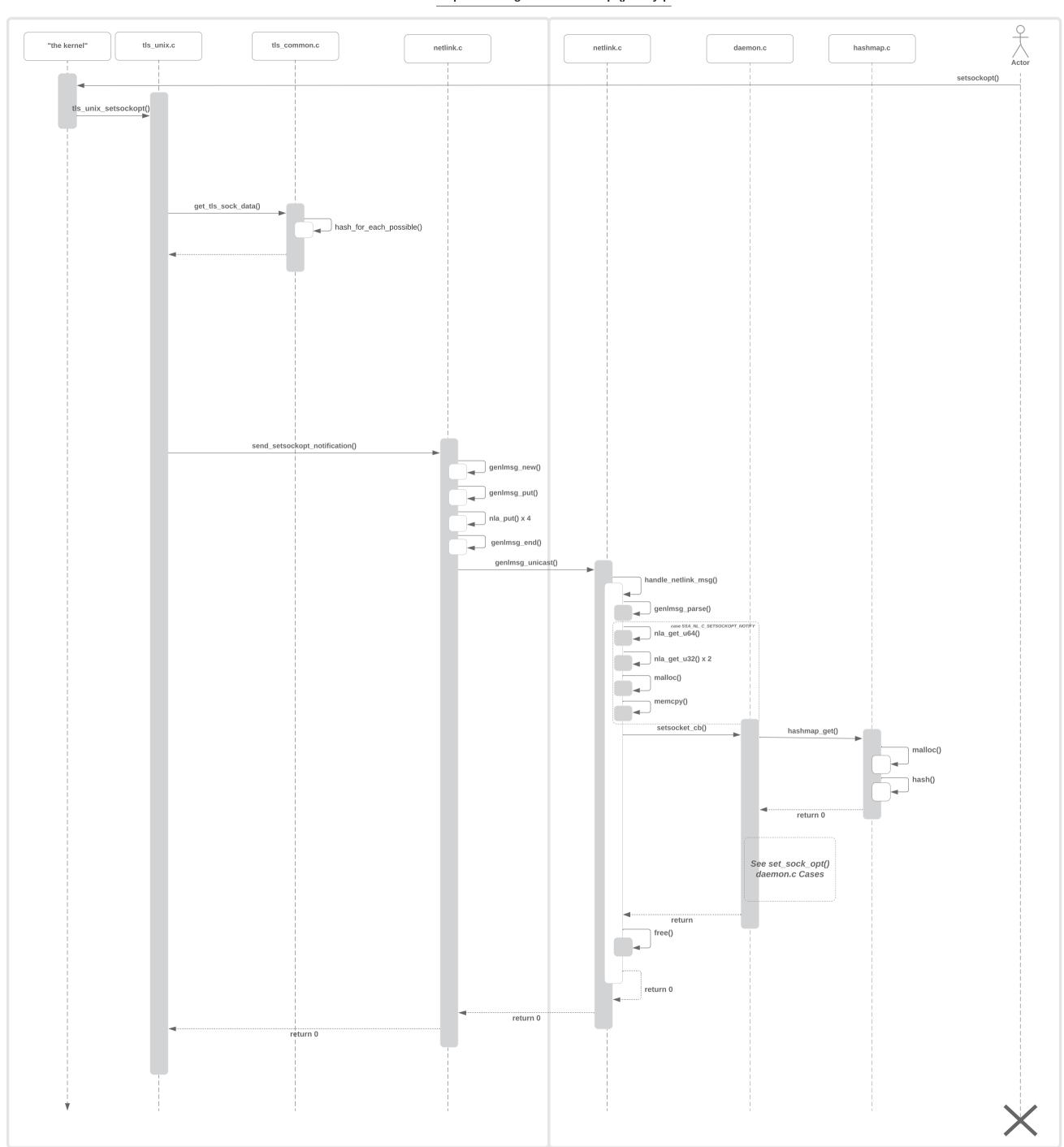


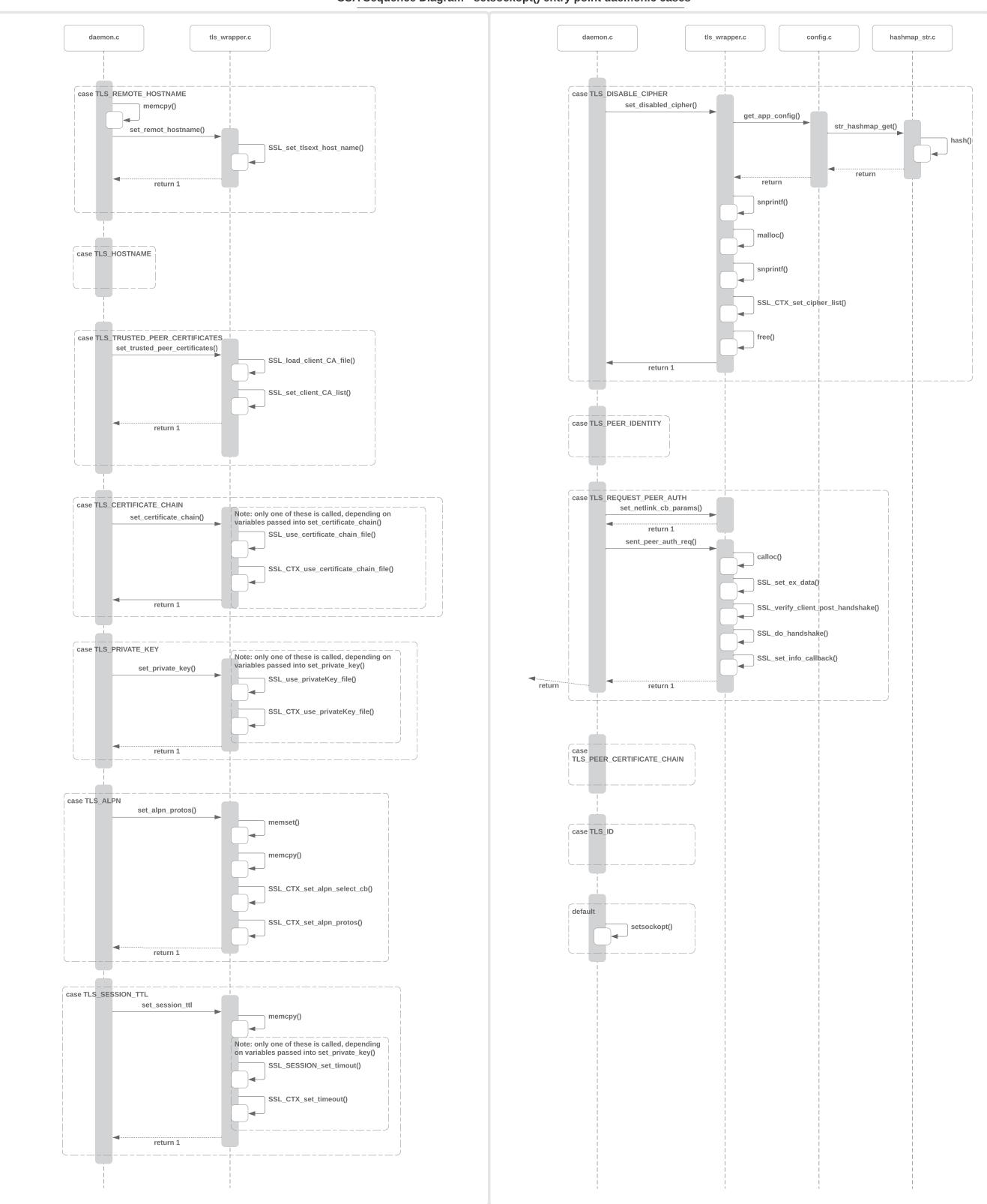
## SSA Sequence Diagram - close() entry point











## **SSA Sequence Diagram - Other Entry Points**

