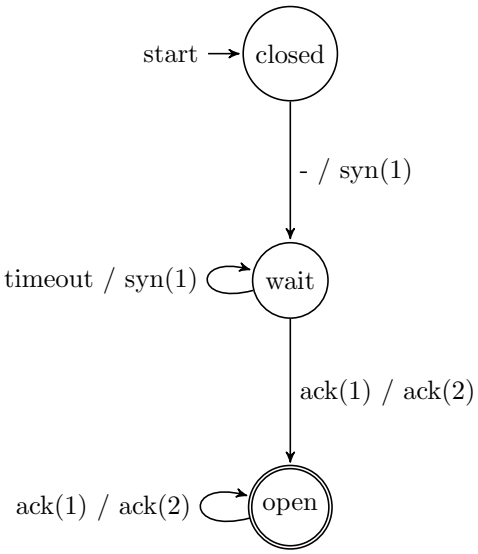
DVA218 - Lab 3a: Mealy State Machines

Sebastian Lindfors

April 24, 2018

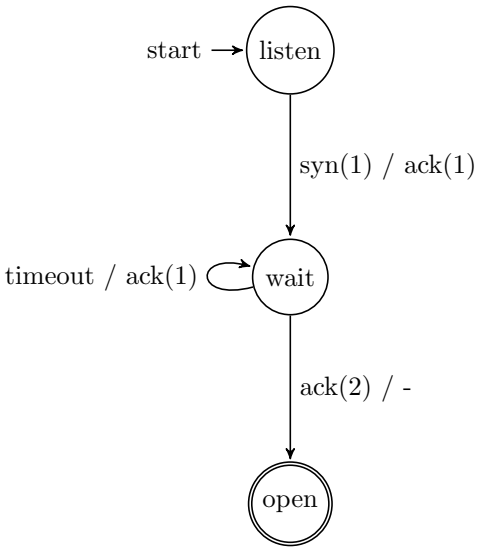
Handshake: Client

The client initiates the handshake by sending a SYN to the server. If the SYN timeout is reached, the SYN is resent. When the server acknowledges the SYN, it responds with an ACK. When the ACK is received on the client side, an ACK of the ACK is sent to the server. Since each party has agreed to establish a connection, the client can assume that the connection is open, even if the final ACK is lost.



Handshake: Server

The server waits for a SYN from a client to arrive on the socket. When a SYN arrives, the server responds with an ACK. If the client fails to respond, the ACK is resent. When the client acknowledges the ACK with another ACK, the server can assume that the connection is open.

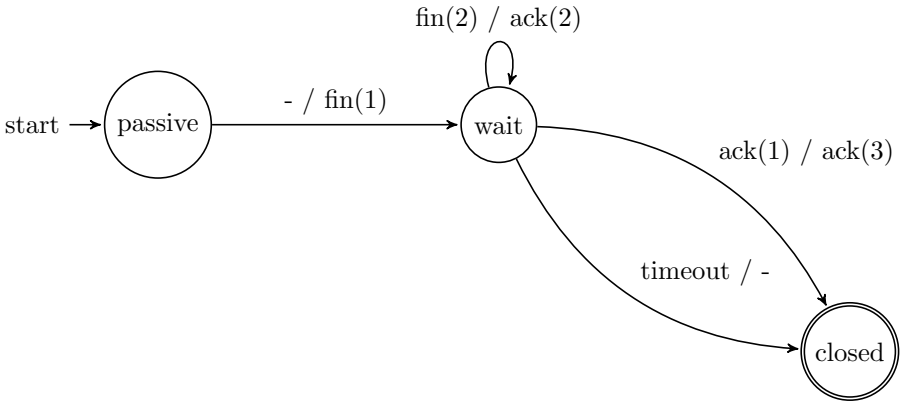


Teardown: Initiator

Either the client or the server can initiate a teardown request, so in the following state machines they are denoted as initiator and responder.

The initiator starts the process by sending a FIN to the responder. The initiator then awaits a consecutive FIN from the responder, and sends an ACK in response. At this point, the responder should send an ACK for the initial FIN, in which case the initiator will respond with a final ACK.

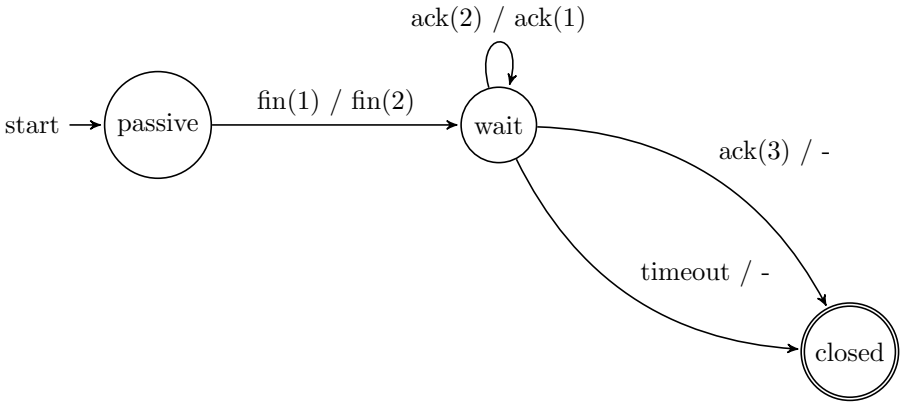
If the responder is unresponsive at any point during teardown, the initiator will eventually close the connection anyway.



Teardown: Responder

When a FIN is received from the initiator, the responder sends a consecutive FIN to the initiator. It then awaits an ACK for the consecutive FIN, in which case it responds with an ACK for the initial FIN. It then awaits the final ACK and closes instantly when it arrives.

If the initiator becomes unresponsive at any point, the responder can eventually close the connection anyway, because it's locally aware that the other party wants to close the connection.

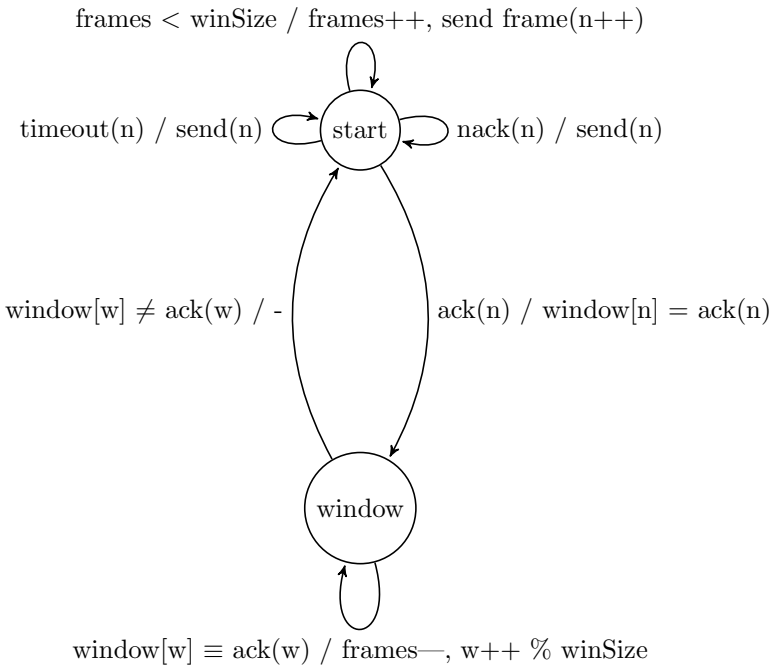


Sliding Window: Sender

The following state machines implement selective repeat to handle packet loss, out-of-order packets, and the sliding window mechanism itself.

From the sender's perspective, it will initially send all packets that fit within the window, in order. For each packet sent, a timeout is present in case the receiver doesn't respond to it, and it is automatically resent. If the sender receives a NACK for a specific packet (corrupt data), it will also be resent.

When the sender receives an ACK for a packet, it places that ACK in its own window and tries to process the window. If the lowest index in the window is an ACK, it will open up to send another frame and slide the window forward. When it reaches an index that doesn't have an ACK, it goes back to the starting point, and repeats the process. This should effectively catch out-of-order packets that need to be resent before moving on.



Sliding Window: Receiver

When the receiver receives a frame from the sender, it runs a checksum comparison on the packet content. If the checksum is false, the receiver sends a NACK in return. If the frame is fine, the receiver sends an ACK to the sender, and inserts the frame in its window buffer.

If the lowest index in the window has a frame, the window slides forward. When it reaches an empty index, it goes back to the starting point and awaits more packets to arrive.