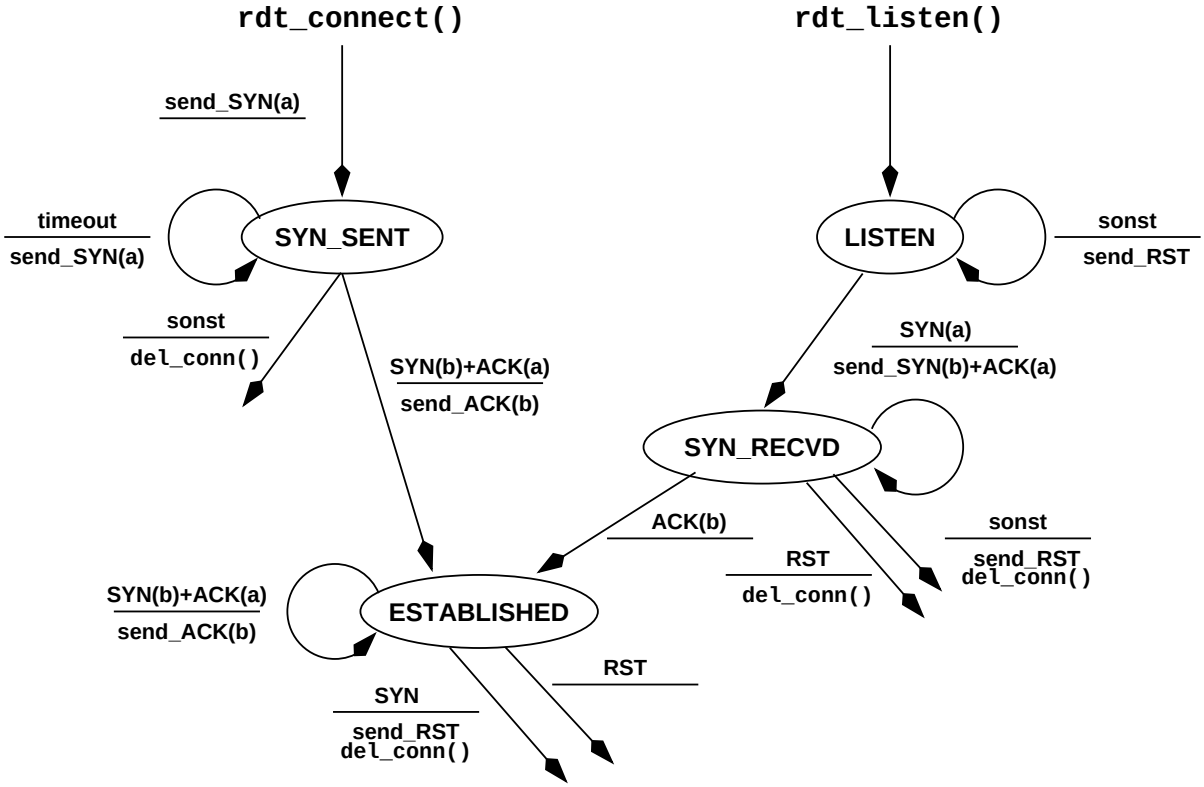
TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet Intelligente Netze und Management verteilter Systeme
Prof. Anja Feldmann, Ph. D.
Harald Schiöberg, Bernhard Ager, Vlad Manilici

8. Assignment Protocol Design WS 08/09

Question 1: (50 points) Connection Handshake

In this exercise you shall extend you existing RDT Implentation to the functionality of RDT 4.0. This knows in addition to RDT 3.0 the functionalities of connection handshake and teardown. First we introduce the explicit connection handshake. For that you have to extend the RDT Header by the 3 flag bits `SYN, FIN, RST`. Furthermore the RDT magic is changed to `RDT4.0`. Additionally the following statemachine is to be implemented:



IN the state `ESTABLISHED` RDT 4.0 exactly like RDT 3.0. The call of `rdt_listen` starts the automaton in the state `LISTEN`. IN this state the server waits for packet with `SYN` state set. Any other packet is answered with a `RST`. If a `SYN` is received, the server replies with `SYN ACK` and switches to `SYN_RECVD`

Generally: For a `SYN` the sequence number can be chosen freely, after that only the next sequence number is allowed for data packets. `SYN` packets are to be acknowledged like normal packets, and the sequence number is increased.

In the state `SYN_RECVD` the server expects an `ACK` packet for his `SYN`. When this is arriving, it changes in the state `ESTABLISHED`. Any other packet is answered with a reset packet.

Generally: After sending a `RST`, the whole state that is kept for a connection is immediately deleted.

On the client side you need to implement the branch of `rdt_connect` If `rdt_connect` is called, a `SYN` is sent immediately. The connection goes to the state `SYN_SENT`. In this state the connection waits for a `SYN ACK`, which is acknowledged and the connection moves to state `ESTABLISHED`.

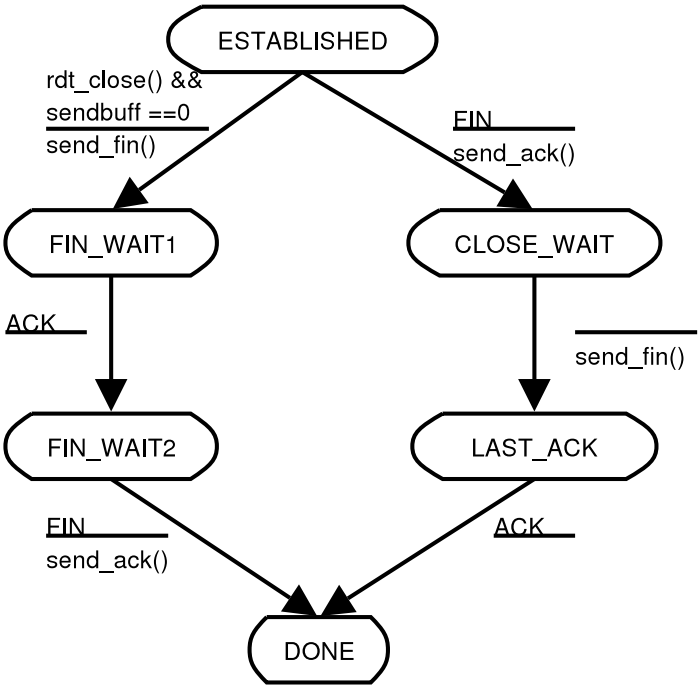
Generally: in each of the states, the last sent packet is retransmitted on a timeout. Furthermore calling `rdt_send` or `rdt_recv` should yield an error code if the connection is not in the state `ESTABLISHED` .

Question 2: (50 points) Connection teardown

Now the connection should be teared down. For that you need the function `rdt_close`. It sets a flag in the connection state indicating that the connection should be closed. If this flag is set, and the sender buffer is empty, a `FIN` packet is sent and the connection state is `FIN_WAIT1`. In `FIN_WAIT1` the connection expects an ack for the fin and moves on to `FIN_WAIT2`. If a `FIN` is received, it is acknowledged and the connection is terminated.

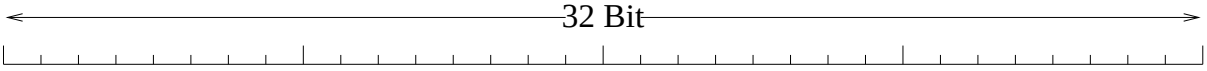
As opposed to this *active close*, there exists also the passive close: If a `FIN` is received in `ESTABLISHED`, it is acknowledged and the connection goes into `CLOSE_WAIT`. In `CLOSE_WAIT` the connection can still send data through send, but not receive new data. A call to `rdt_close` again sends a `FIN`, and moves on to `LAST_ACK`, which waits for an ack and terminates the connection.

Here is the statemachine:



Hint: you should introduce a function (e.g. statemachine_connection()), which does the statechanges for the connection state, and which calls statemachine_send and statemachine_recv if the connection is in the right state.

For reference: The new RDT-Header:



R	D	T	4
.	0	Reserviert	
Reserviert			
Reserviert			
Reserviert		Payload Length	
Seq number	Ack number	Reserviert Rst Fin Syn A	Checksum
Reserviert			
Reserviert			
Reserviert		Payload....	

For clarification:

ACK has the value 0x01, SYN is 0x04, FIN 0x08 and RST 0x10.
Submission: until Wed, 17th Dec 2008, 23:59h