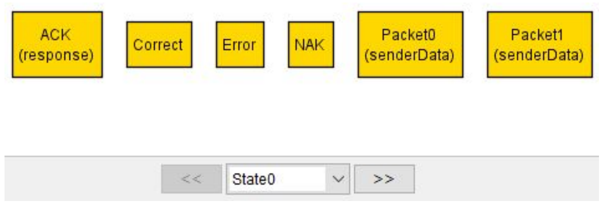
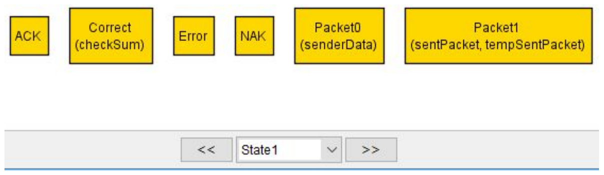
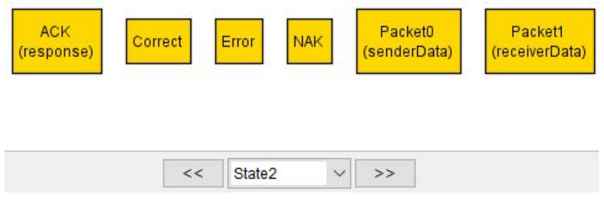
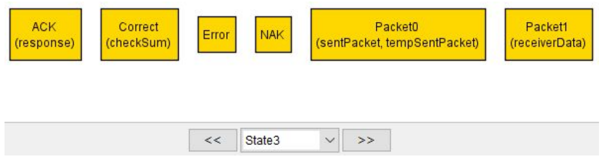
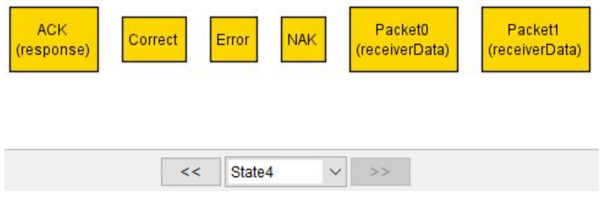
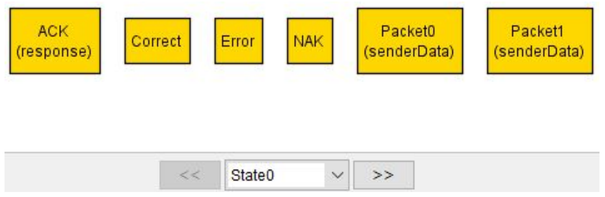
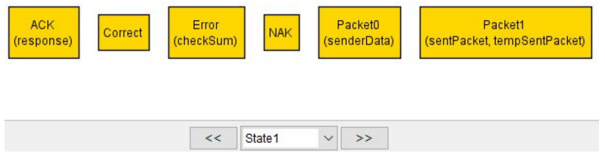
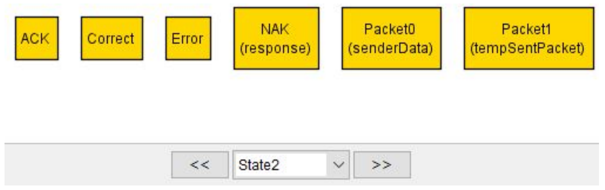
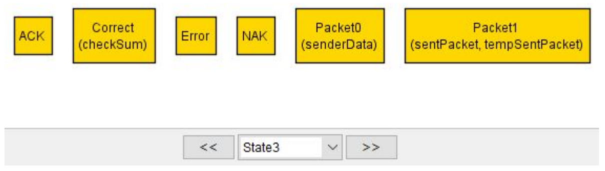
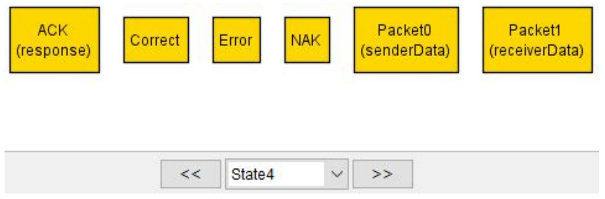
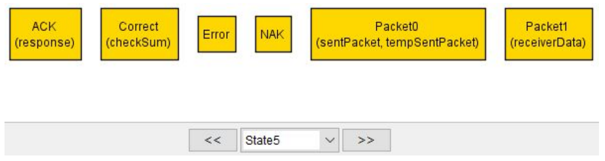
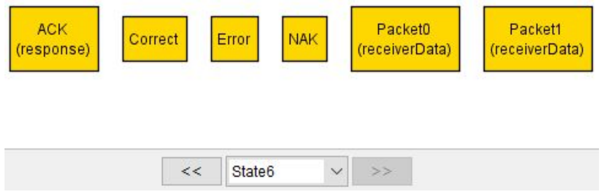
Chris Budo
Jonathan Jenkins
Melissa Thai
Milestone 2 - Checking RDT 2.0

Below you can see a walkthrough of two scenarios of two packets of data being sent projected over state. The first scenario has no errors during transfer, while the second scenario has an error, causing the sender to resend the error packet. Note that every piece of data starts with the sender in *senderData* - with *tempSendData* being used as a buffer to contain the original packet in case of an error requiring a resend - then transitions to being in the *sentPacket* state, which represents it being in the network before finally finishing in the *receiverData* which represents it being received by the person who requested it. Before it can be saved into *receiverData* however the *checkSum* is checked. If it was correct then it saves and sends an ACK. However if there was an error then it sends a *NAK* which will cause it to be retransmitted to the receiver.

5 States - No Error	
 ACK (response) Correct Error NAK Packet0 (senderData) Packet1 (senderData) << State0 >>	In the initial state all packets are stored in the sender data. There is an *ACK* response that represents the two machines completing their hand shaking. Note there is no *checkSum* yet.
 ACK Correct (checkSum) Error NAK Packet0 (senderData) Packet1 (sentPacket, tempSentPacket) << State1 >>	In the second state, a packet is selected and set as the *sentPacket* and the *tempSendPacket*. *sentPacket* is the packet to be send to the receiver and *tempSendPacket* is what will be sent again in case the receiver gets an error. Note that the *checkSum* is now set to *Correct*.
 ACK (response) Correct Error NAK Packet0 (senderData) Packet1 (receiverData) << State2 >>	In the third state the *checkSum* is evaluated (It is correct) and therefore an *ACK* response is sent. This causes the *Packet1* to transition into *receiverData*.
 ACK (response) Correct (checkSum) Error NAK Packet0 (sentPacket, tempSentPacket) Packet1 (receiverData) << State3 >>	In the fourth state, we see that *Packet0* is being sent to the receiver (since it's the *sentPacket* and the *tempSendPacket*). *checkSum* is set to *Correct*.

 ACK (response) Correct Error NAK Packet0 (receiverData) Packet1 (receiverData) << State4 >>	In the final state the *checkSum* is again evaluated to correct which causes an *ACK* to be sent and *Packet0* to be saved to *receiverData*. The data transfer is now complete.

7 States - One Error	
 ACK (response) Correct Error NAK Packet0 (senderData) Packet1 (senderData) << State0 >>	In the initial state all packets are stored in the sender data. There is an *ACK* response that represents the two machines completing their hand shaking. Note there is no *checkSum* yet.
 ACK (response) Correct Error (checkSum) NAK Packet0 (senderData) Packet1 (sentPacket, tempSentPacket) << State1 >>	In the second state, a packet is selected and set as the *sentPacket* and the *tempSendPacket*. *sentPacket* is the packet to be sent to the receiver and *tempSendPacket* is what will be sent again in case the receiver gets an error. *checkSum* is set to *ERROR*.
 ACK Correct Error NAK (response) Packet0 (senderData) Packet1 (tempSentPacket) << State2 >>	In the third state the *checkSum* is evaluated (It is an *ERROR*) and therefore a *NAC* response is sent. This causes the *Packet1* to *NOT* enter in as receiver data, and instead simply hang out as the *tempSentPacket* ready to be resent.
 ACK Correct (checkSum) Error NAK Packet0 (senderData) Packet1 (sentPacket, tempSentPacket) << State3 >>	In the fourth state, we see that *Packet1* is being resent, now causing the *checkSum* to be correct.
 ACK (response) Correct Error NAK Packet0 (senderData) Packet1 (receiverData) << State4 >>	*Packet1* is now saved into *receiverData* as the *checkSum* is correct. An *ACK* message is sent back allowing for progression.

 ACK (response) Correct (checkSum) Error NAK Packet0 (sentPacket, tempSentPacket) Packet1 (receiverData) << State5 >>	*Packet0* is now free to be sent along with a *Correct* check sum.
 ACK (response) Correct Error NAK Packet0 (receiverData) Packet1 (receiverData) << State6 >>	*Packet0* is accepted by *receiverData* and an *ACK* was sent. The data transfer is now complete, despite an error in transmission.