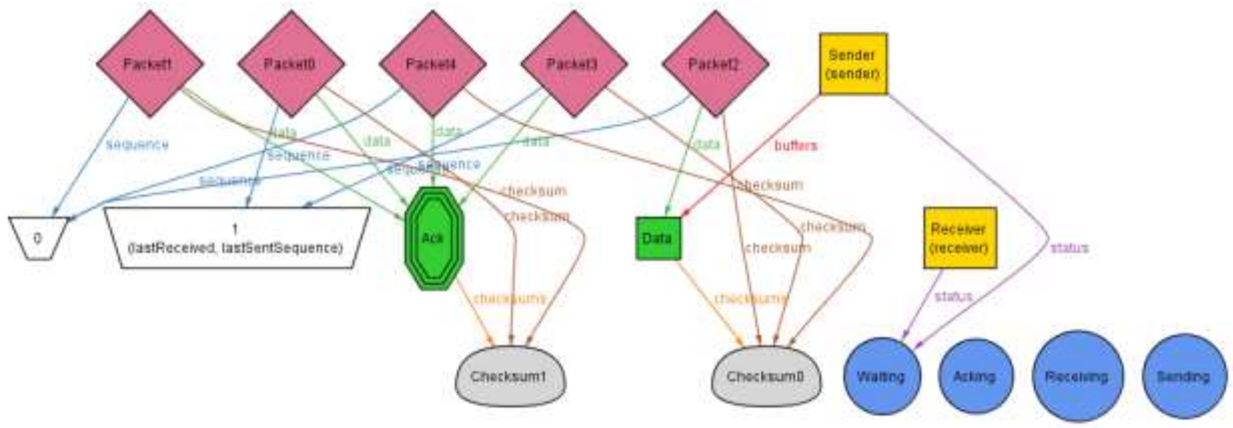
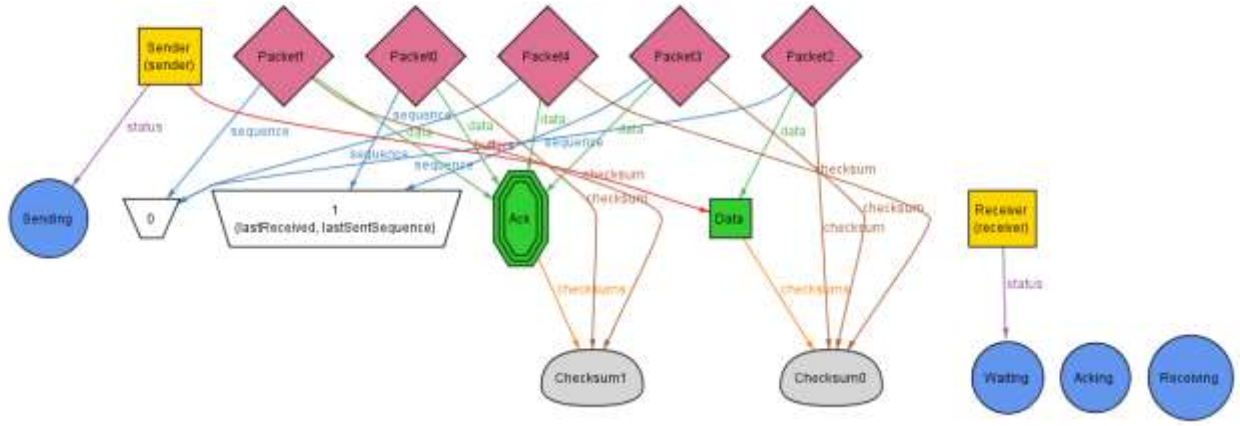
RDT2.2 Writeup
Team Carry Harder
John Krasich
Matt Lash
Caleb Post

1. Is it possible to transmit all the data in the sender's buffer to the receiver's buffer?
   Yes. There are now many more possibilities for this tor occur, it can occur without errors, with errors in data, with errors in the ACK, and with errors in both the data and the ACK. I will show the cases below with 1 Data where there is errors in the ACK and where there is an error in the ACK and Data.

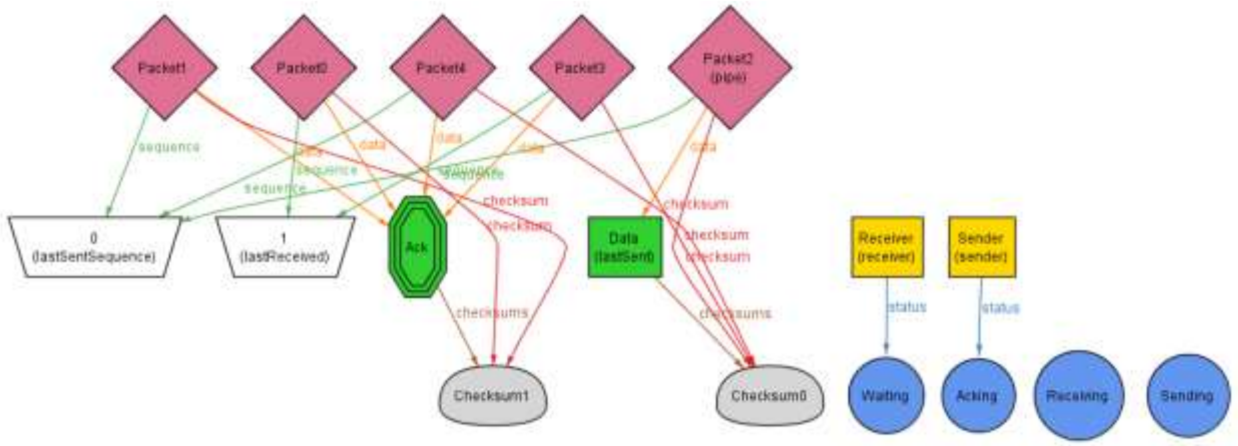
   Error with ACK:
   Initial State:
   We have one piece of data that the sender wants to send to the receiver. The lastReceived packet sequence number and lastSentSequence number are both initialized to 1 to match the protocol.



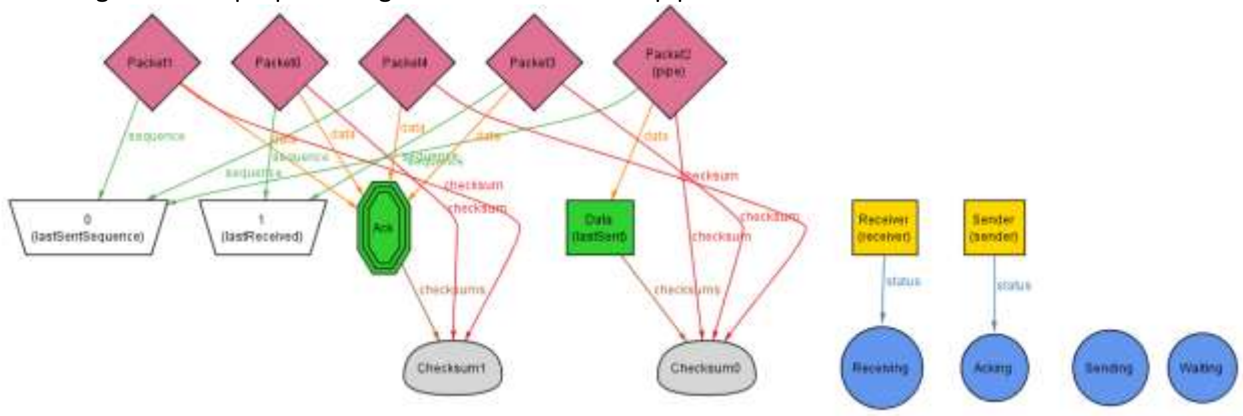
   State 2:
   The sender transitions into the sending state and he prepares to place the data in the pipe.



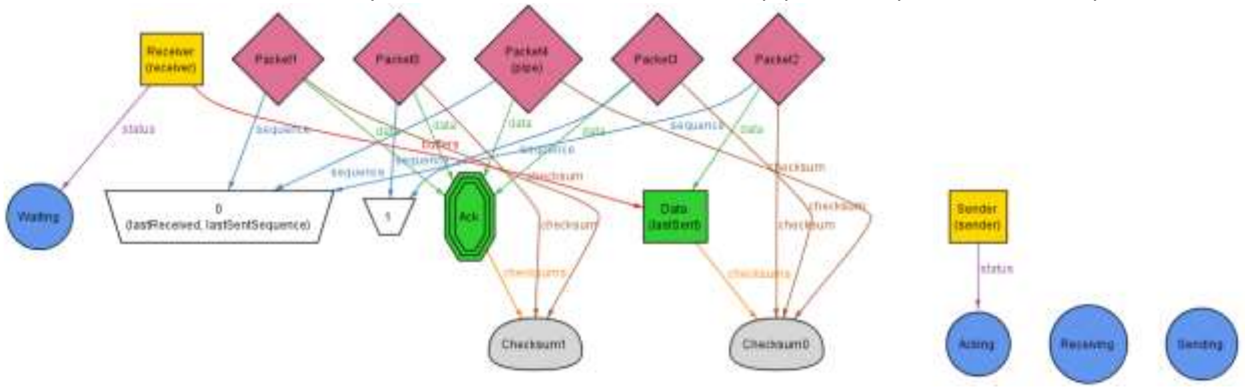
State 3:
The sender has sent the data into the pipe. The last sent sequence number is updated to be 0 and the last sent data is updated to be Data. The sender enters the Acking state because he is now waiting on a reply from the receiver that it has gotten the data.



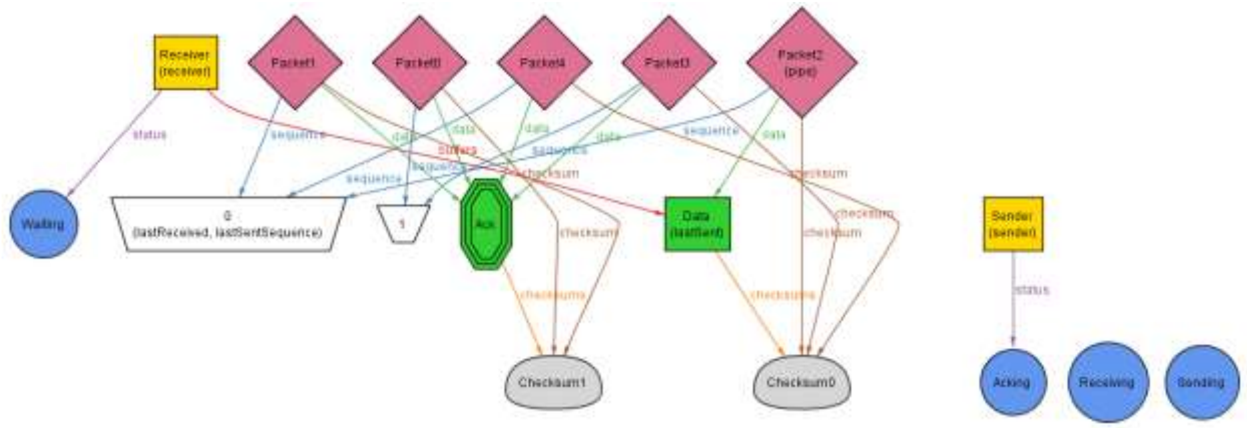
State 4:
The receiver sees that there is data in the pipe that he needs to get, so he transitions into his receiving state and prepares to get the data out of the pipe.



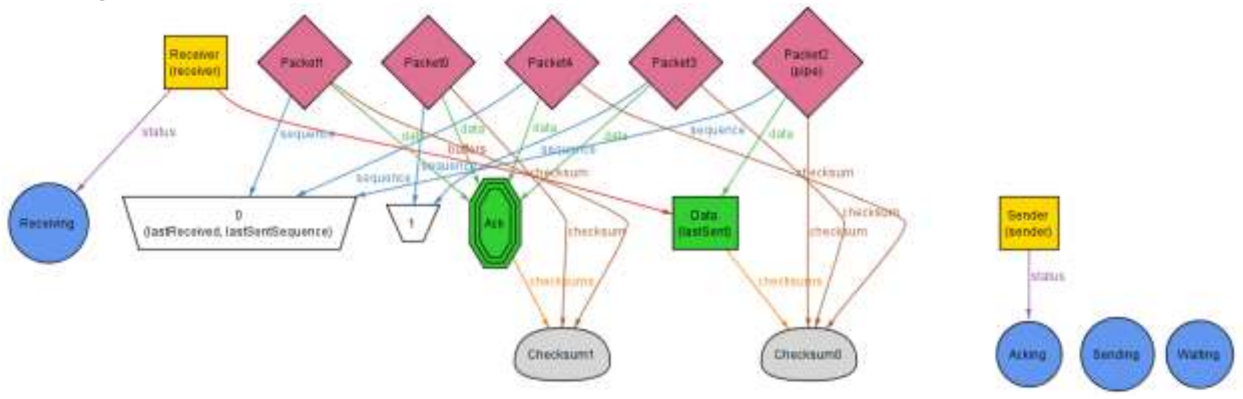
State 5:
The receiver pulls the data packet from the pipe and sees that it is an uncorrupt packet. He updates his last received sequence number to be 0 and then places an ACK in the pipe to the sender. However, look carefully and we see that the ACK in the pipe for sequence 0 is corrupt.



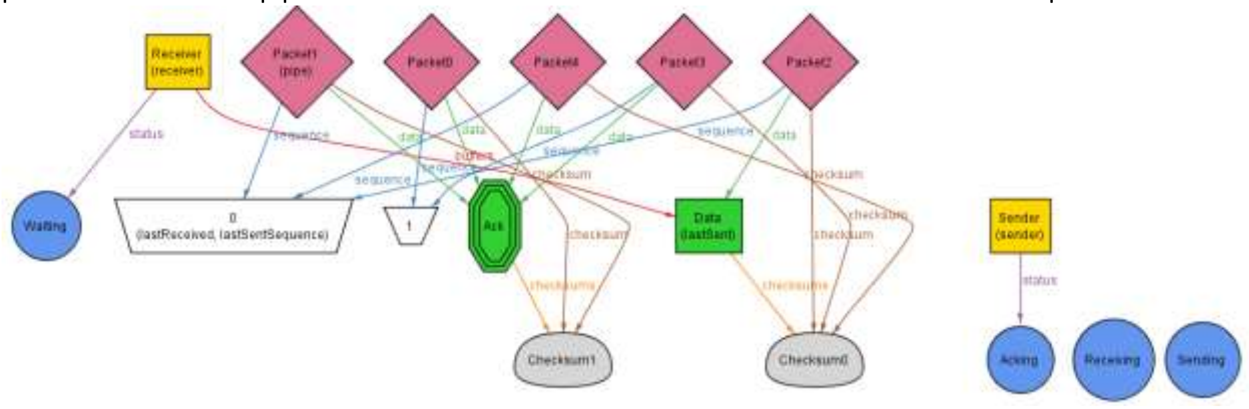
State 6:
The sender sees that the ACK is corrupt, so he responds by immediately resending packet number 0, his last sent packet.



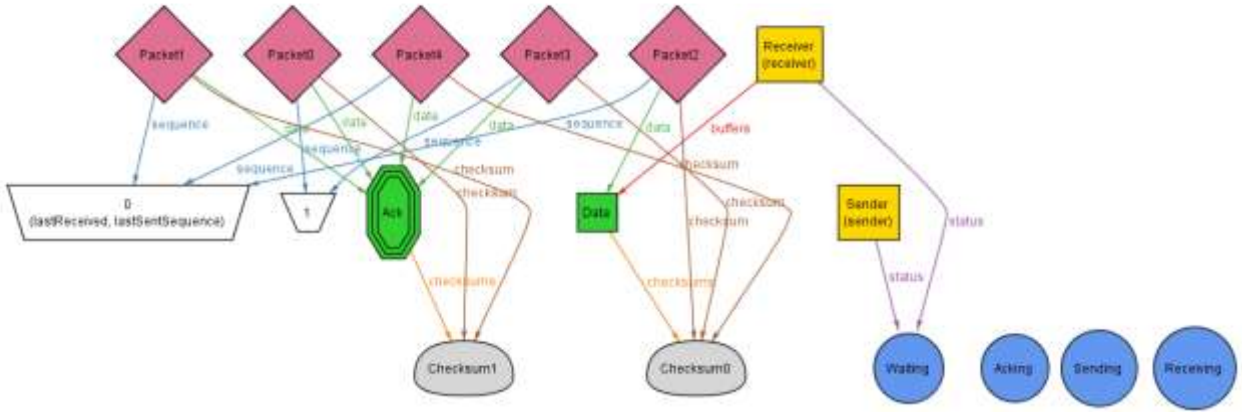
State 7:
The receiver sees that there is a packet in the pipe that he needs to get so he transitions into the receiving state.



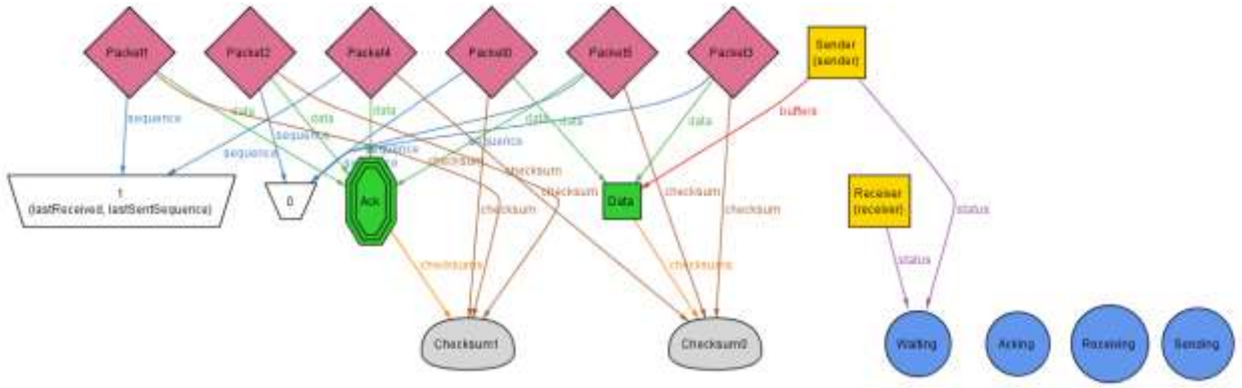
State 8:
The receiver sees that he has already gotten this piece of data with sequence number 0 so he places the ACK in the pipe for the sender. This time we can see that the ACK is not corrupt.



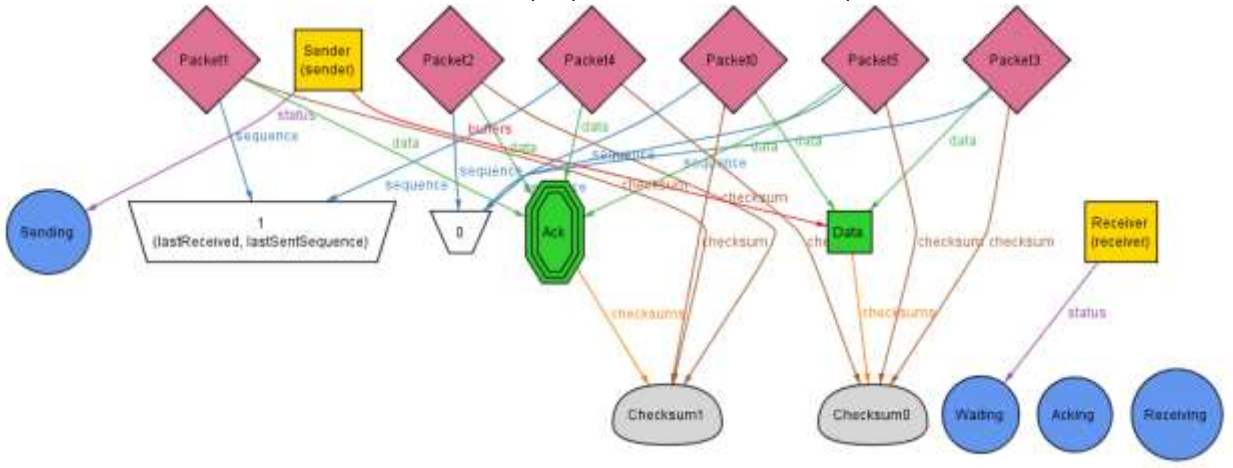
State 9:
The sender gets the ACK from the pipe and so transitions back to waiting. All the data has now been transferred.



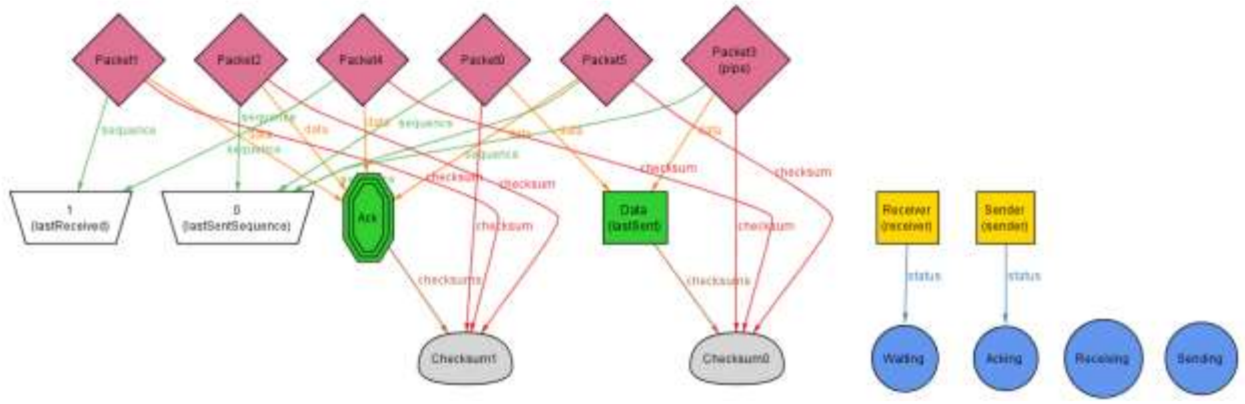
Errors with ACK and DATA corruption
Initial State:
The initial state is very similar to the above state, however there is an additional packet which represents the possibility of sending a corrupt data packet to the receiver.



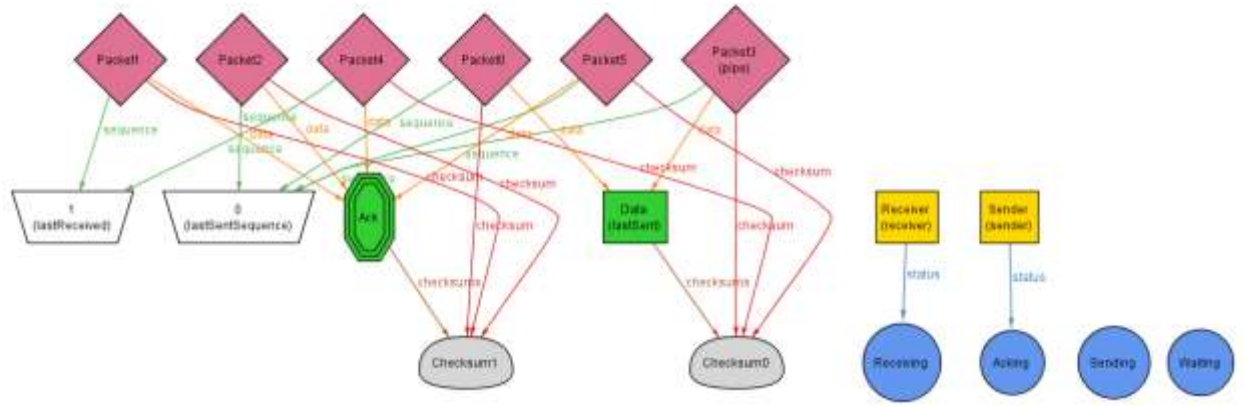
State 2:
The sender transitions to the send state, and prepares to send the data packet to the receiver.



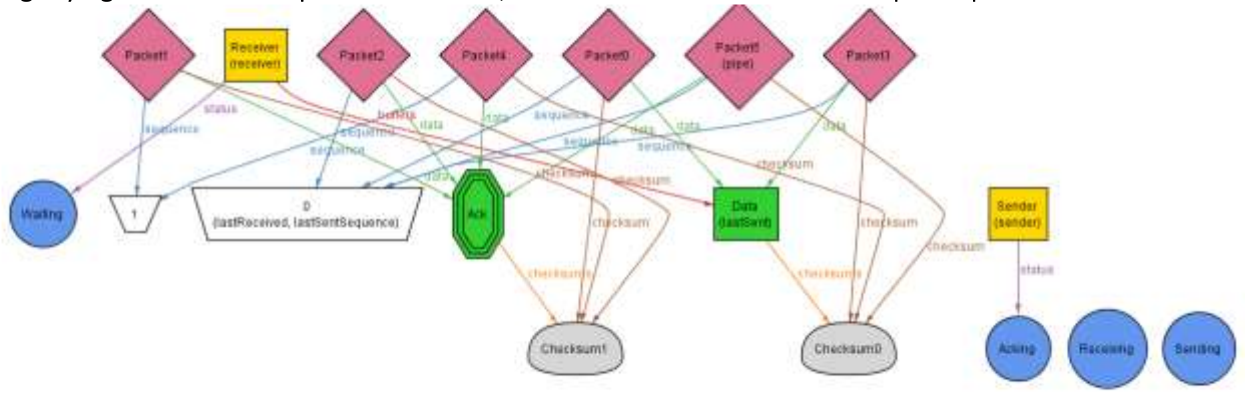
State 3:
Last Sent Sequence gets updated to 0 to reflect that the packet has been put in the pipe, and we can see that the correct packet has been put into the pipe, it is not corrupt.



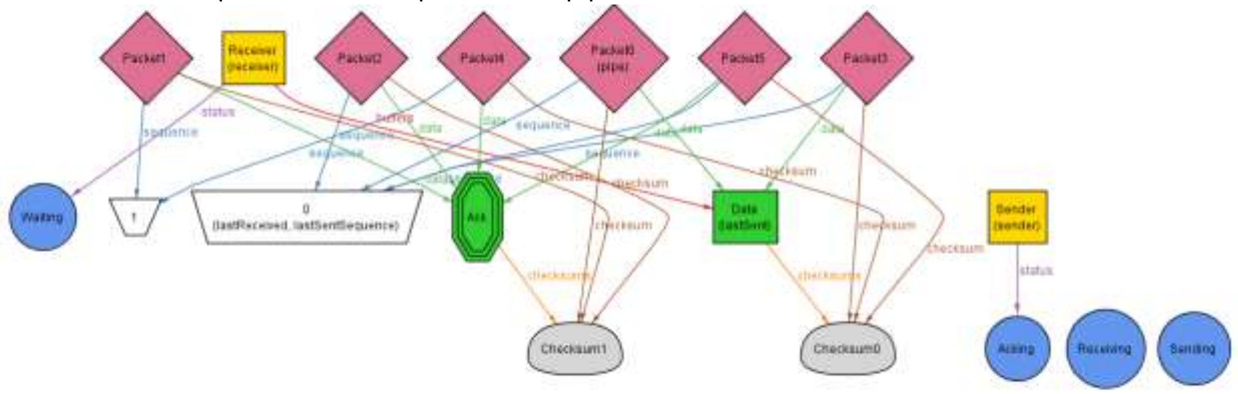
State 4:
The receiver sees that there is data in the pipe that is destined for it and transitions to the receiver state.



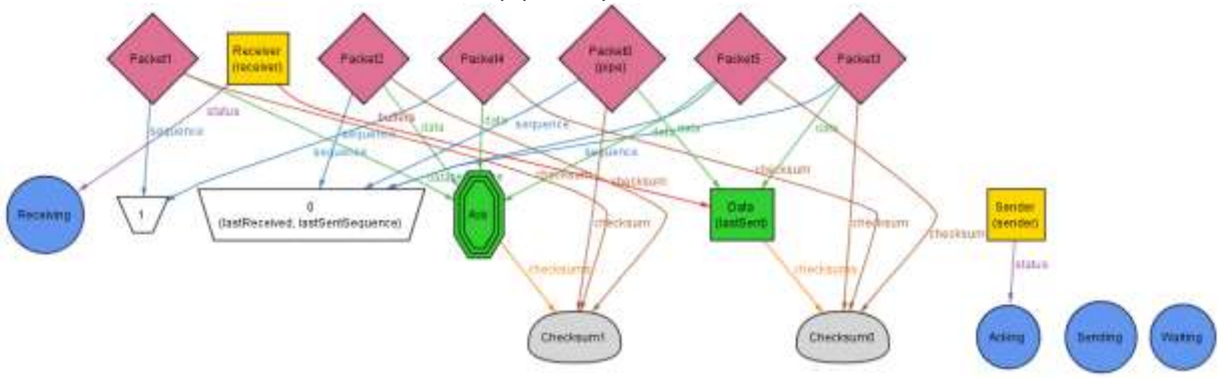
State 5:
The receiver sees that the packet is correct and therefore sends back the ack to the sender signifying it received the packet. However, we can see that this is the corrupt ack packet.



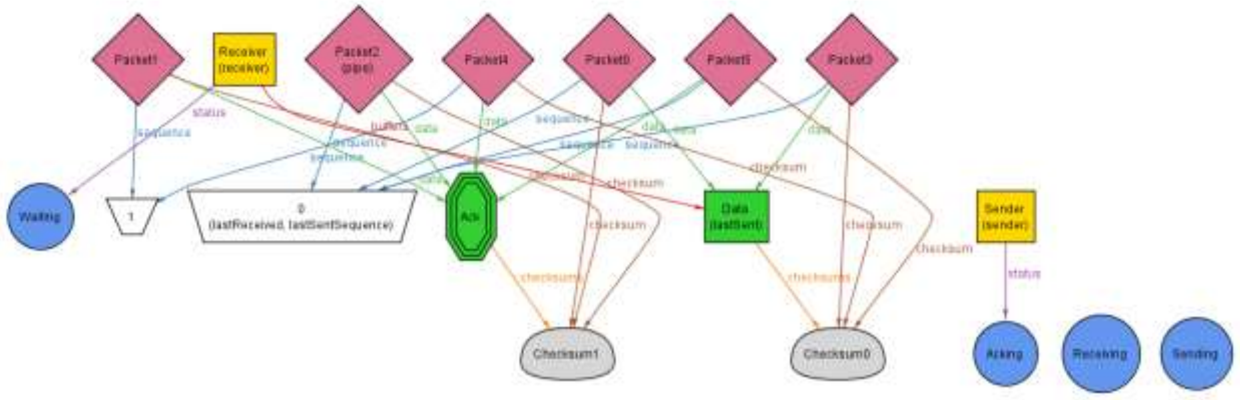
State 6:
The sender gets the corrupt ACK and immediately resends the data packet. This time however the incorrect data packet has been put into the pipe.



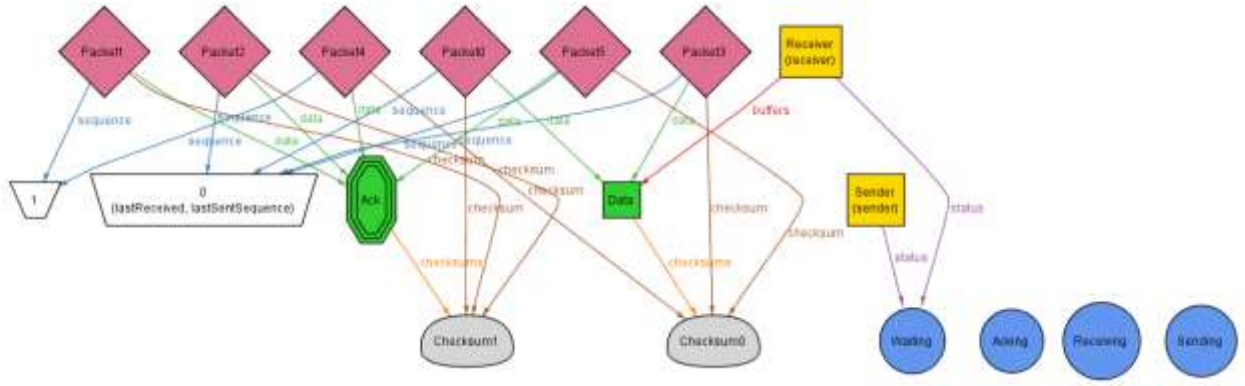
State 7:
The receiver sees that there is data in the pipe for you



State 8:
The correct ACK is now put in the pipe and the incorrect data packet is dropped since that data has already been received.



State 9:
The correct ack received the sender returns back to waiting and the system is now complete for the transfer.

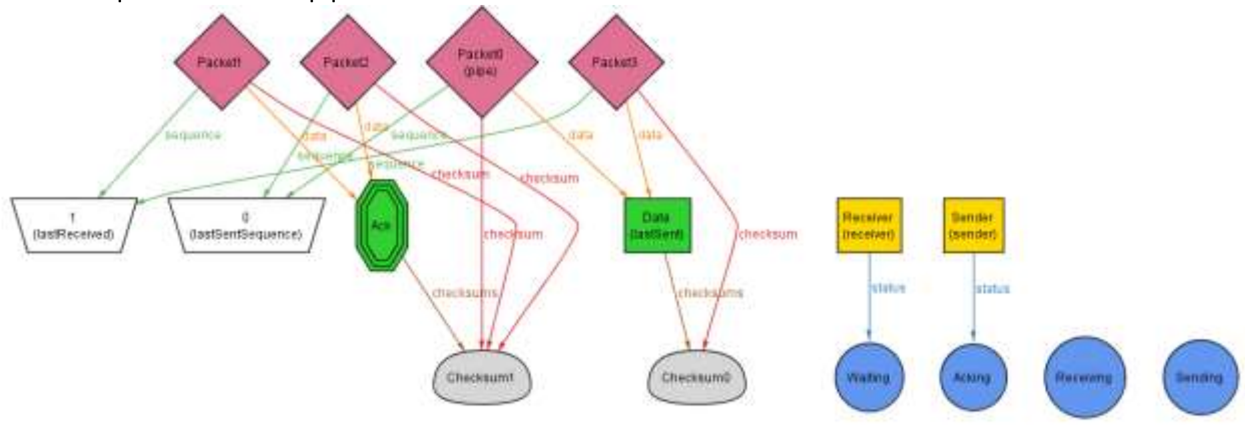


There is also the case that the incorrect data packet is sent first, and the ACK goes back to the sender, but this case has already been described.

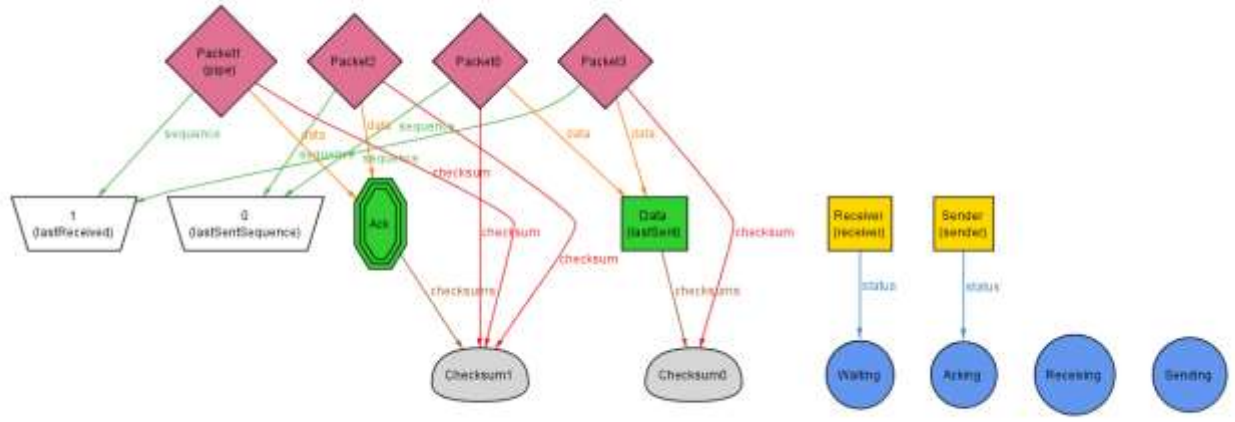
2. Is it always possible to transmit all of the data in the sender's buffer to the receiver's buffer?
No. This is the same case as the previous, RDT 2.0 protocol. If the data sent is always corrupted in the pipe then the protocol will never be able to fully send the information and will never complete the data transfer.

This happens because the system can always oscillate between these states: the state where the incorrect, corrupted, data packet is put into the pipe, and then the acknowledgement is sent back indicating that the packet was incorrect. This will continue happening indefinitely.

The corrupt data is in the pipe.



The acknowledgement is sent back indicating that the data was corrupt, resend. So it sends ack 1 instead of ack 0 for the packet it received.



It will oscillate between these states forever, never putting the correct data into the pipe.

Another case that can cause it never to complete is if the acknowledgement packet sent back is also always corrupt. This means that the data will always be resent even if the data was correct since the acknowledgement is always corrupted.