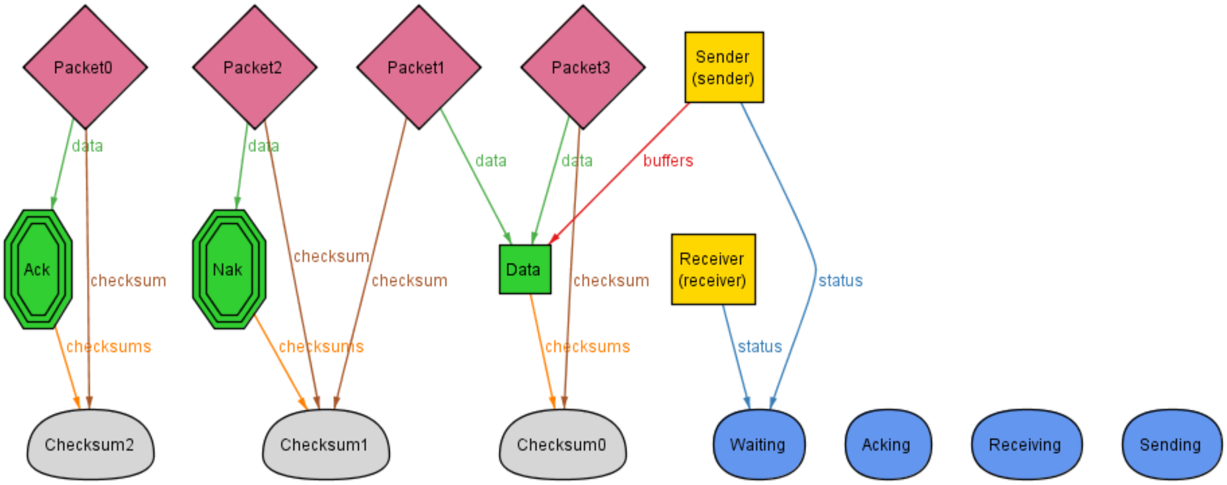
RDT20 Writeup
Team Carry Harder
John Krasich
Matt Lash
Caleb Post

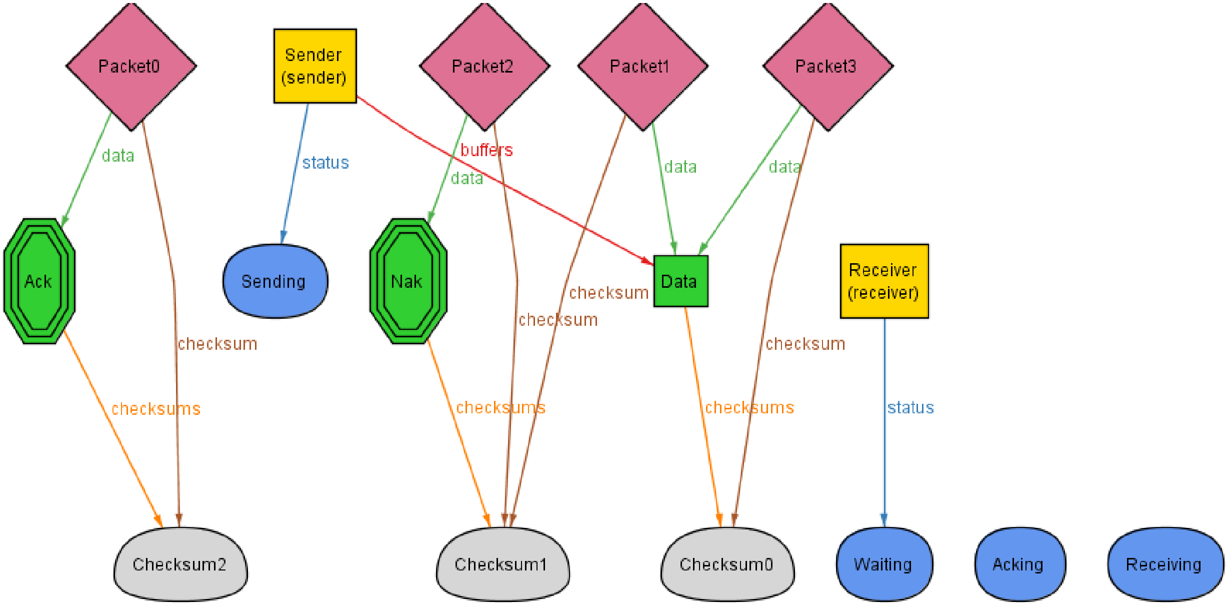
1. Is it possible to transmit all of the data in the sender's buffer to the receiver's buffer with
   i. No errors?

   Yes. Using four packets and four data we can generate the path as follows:

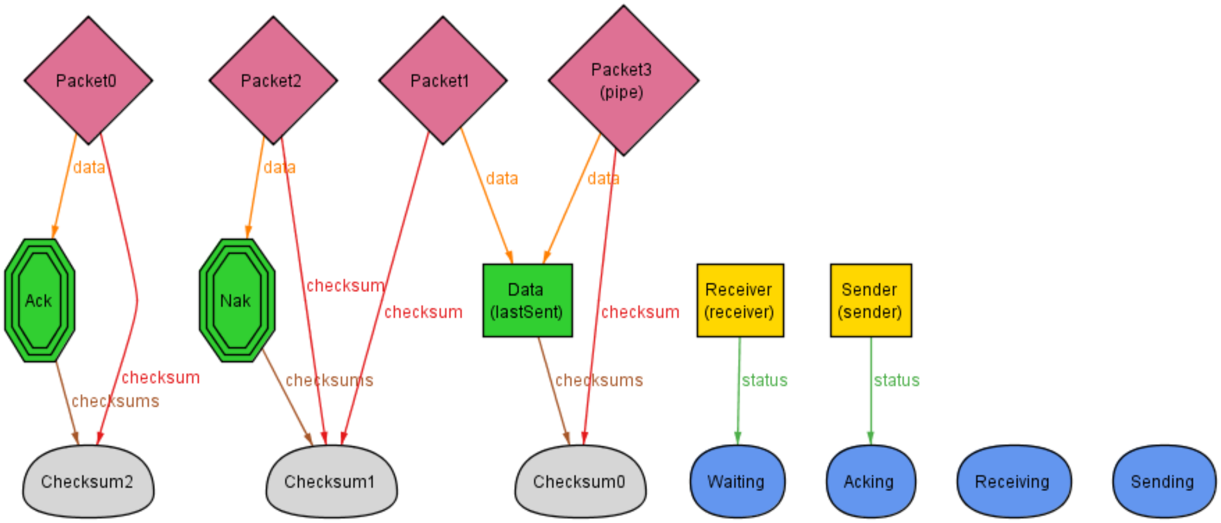
   State 1: Sender has one data to send and both the sender and receiver are waiting



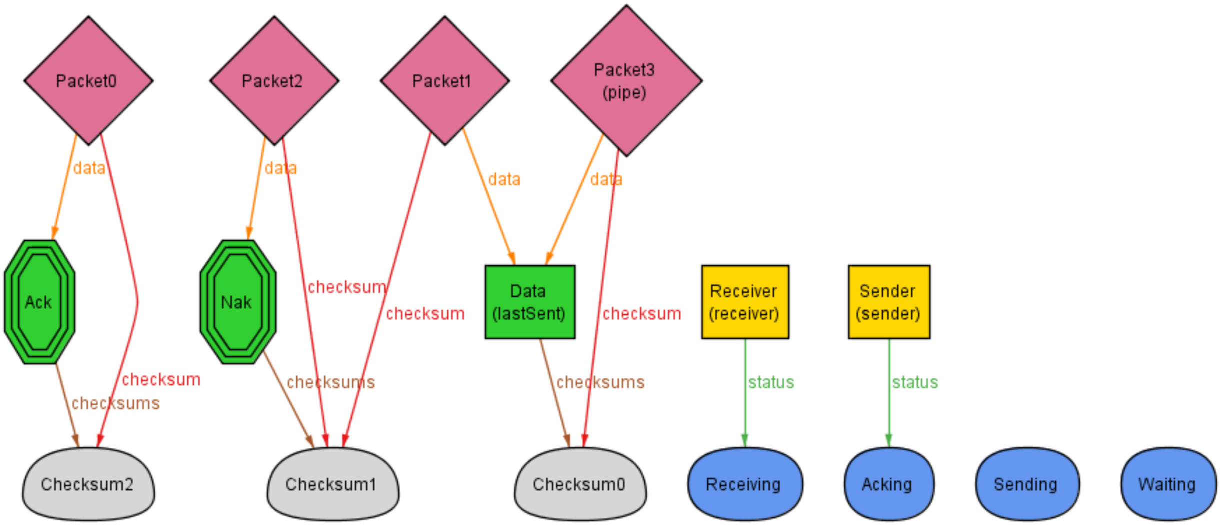
   State2: Sender has gotten into the sending state and is about to transmit data in a packet



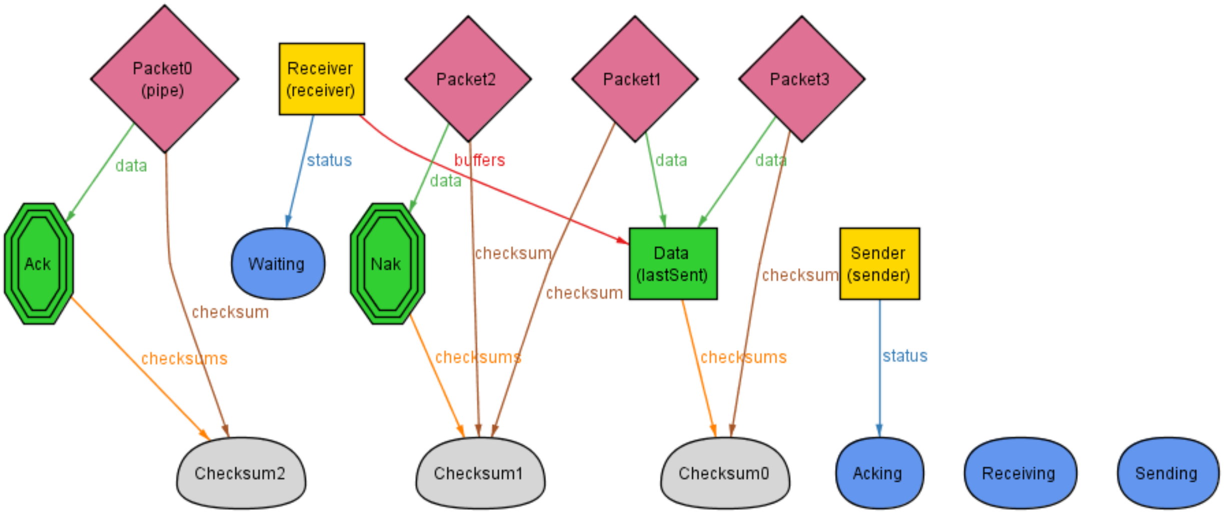
   State 3: Sender has put data in the pipe and has gone into the Acking state



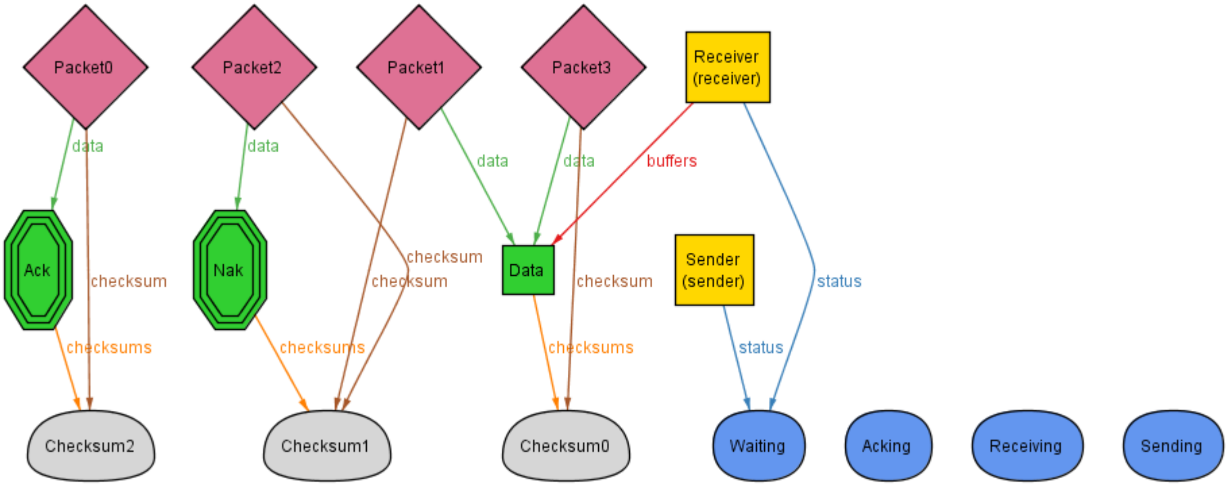
State 4: Sender is in the Acking stage waiting for the response from the Receiver.



State 5: Receiver got the packet and put an Ack packet into the pipe



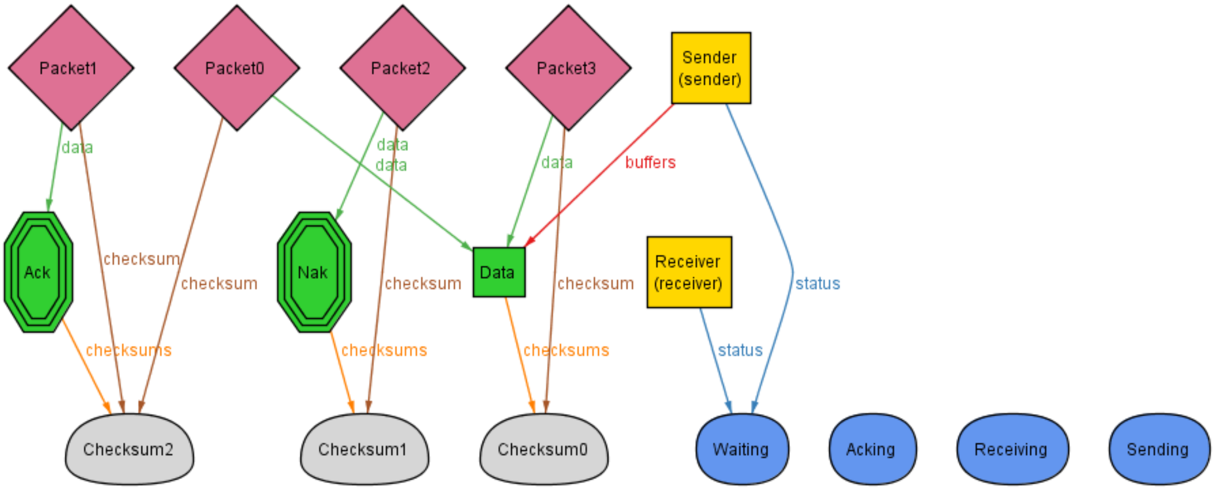
State 6: Sender got the Ack packet and now both sender and receiver are waiting and all the data is in the receiver's buffer.



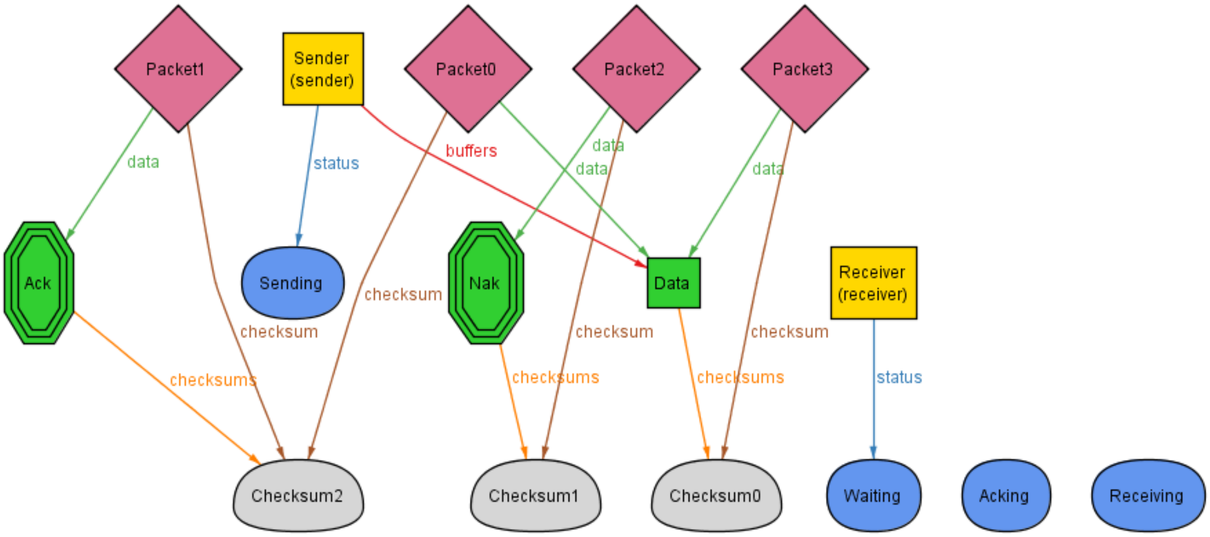
ii. 1 error?

Yes. Using four packets and three data we can generate the path as follows:

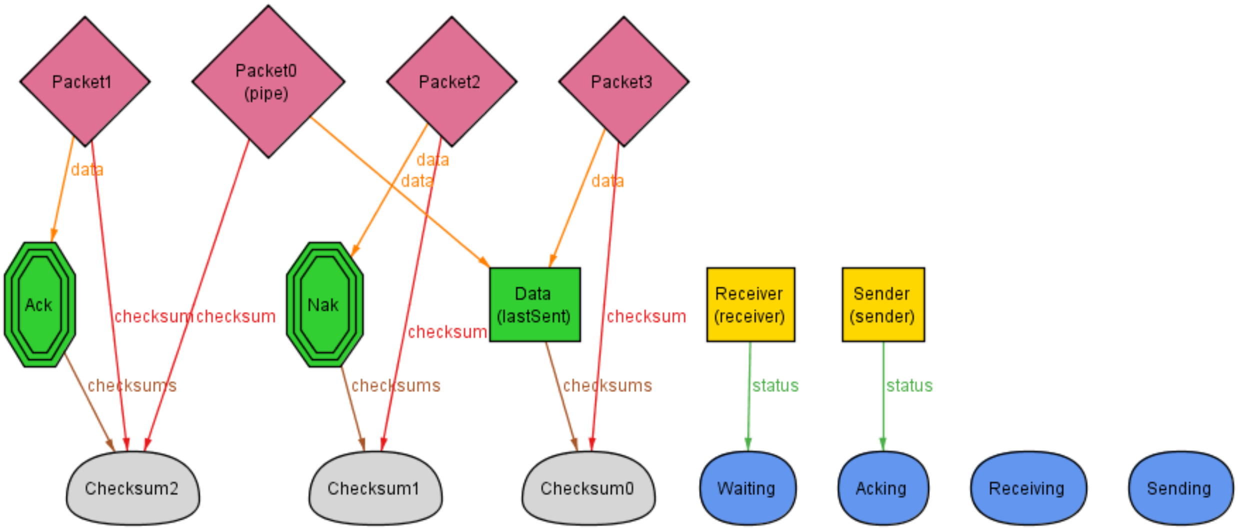
State 1: Sender has data to send, both sender and receiver are waiting



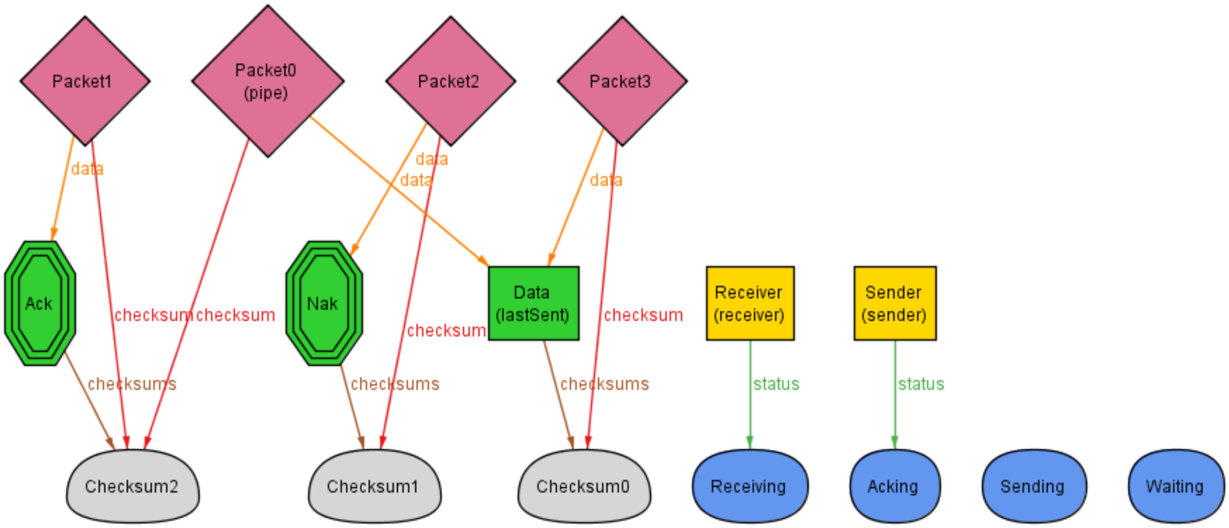
State2: Sender has gotten into the sending state and is about to transmit data in a packet



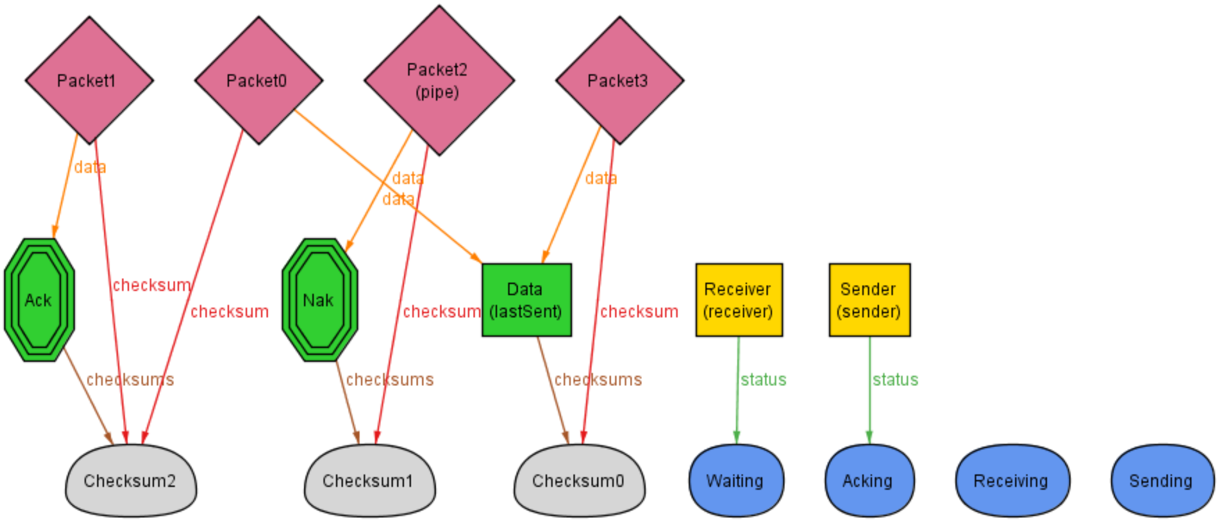
State 3: Sender has put some data in the pipe and has gone to the Acking state



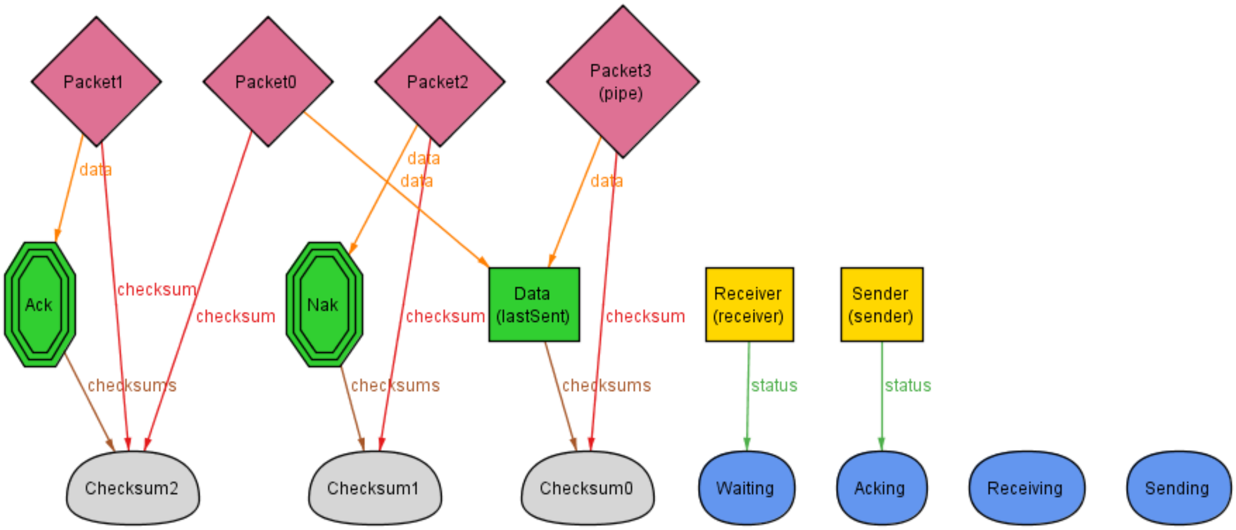
State 4: Receiver has transitions to the receiving state so that it can get the data from the pipe. (notice that this is corrupted data)



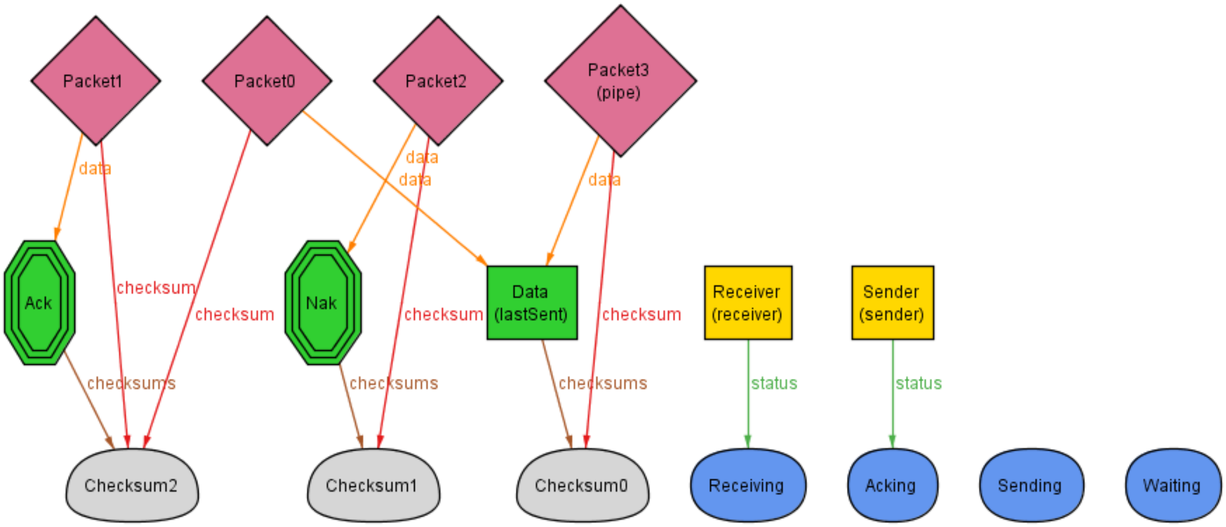
State 5: The receiver has gone back to the Waiting state and realizing the data was currupted has put a Nak packet into the pipe



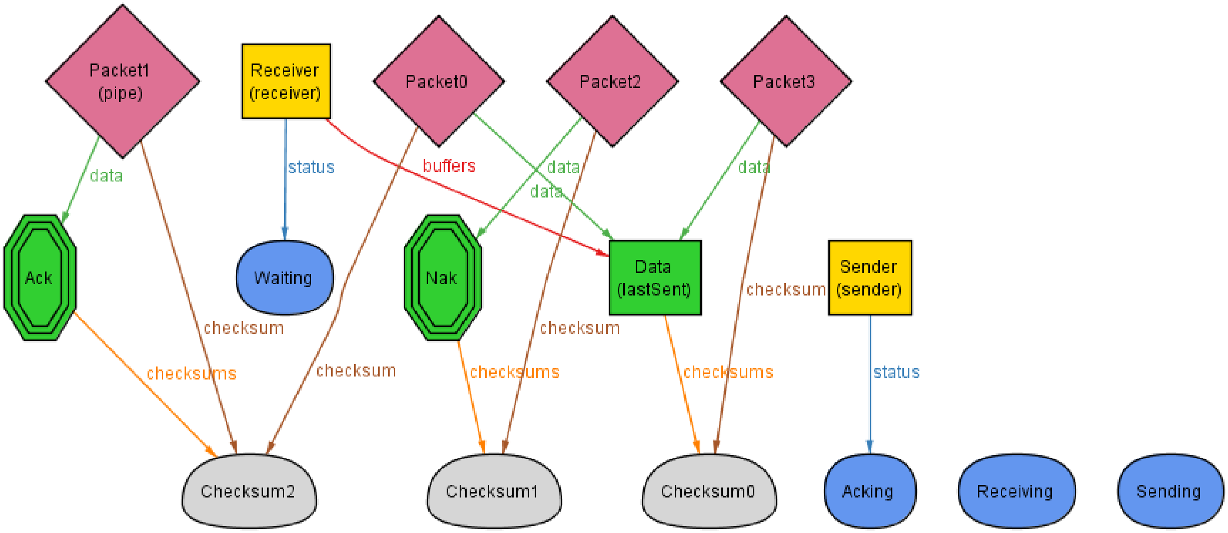
State 6: The Sender got the Nak packet and sent the data again (this time it is not corrupted) and is Acking again



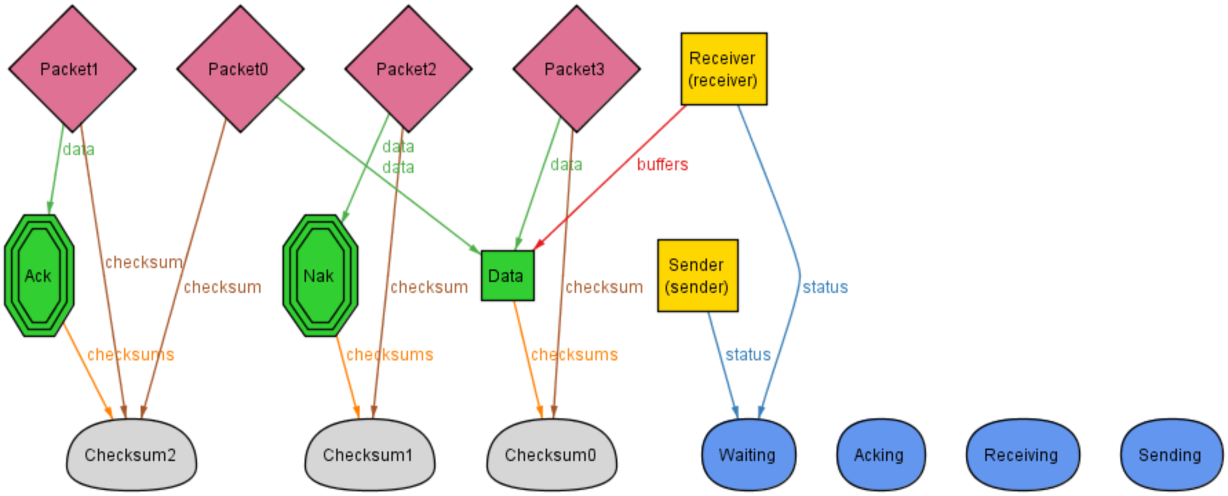
State 7: The receiver detects data intended for it in the pipe and transitions to the receiving state



State 8: The receiver pulls the uncorrupted data from the pipe and sends an Ack packet back for the sender



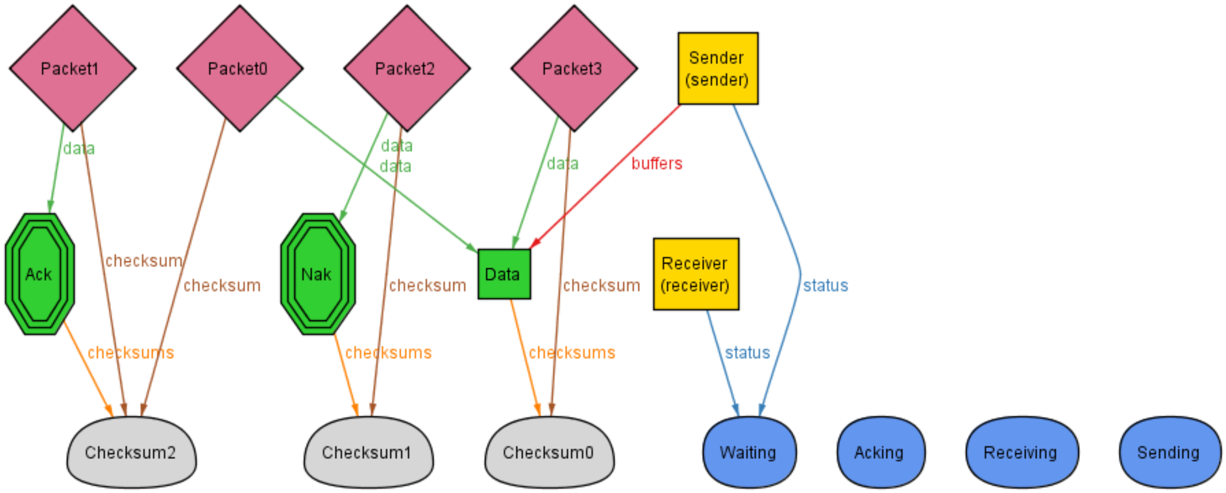
State 9: The sender gets the Ack packet and moved to Waiting. Now both Sender and receiver are waiting and all the data is in the receiver's buffer.



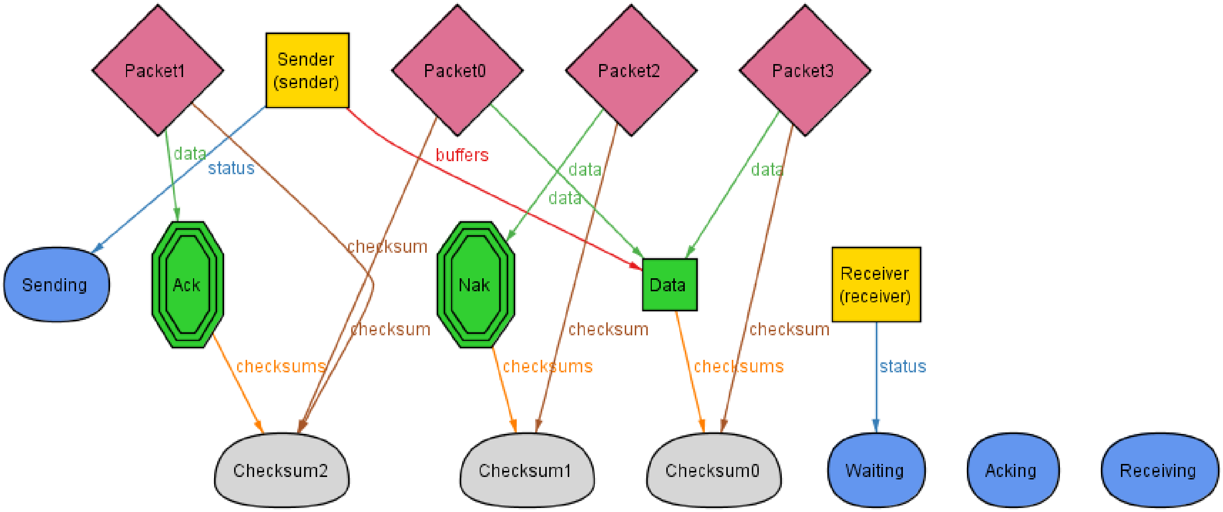
iii. Two errors?

Yes. Using four packets and three data we can generate the path as follows:

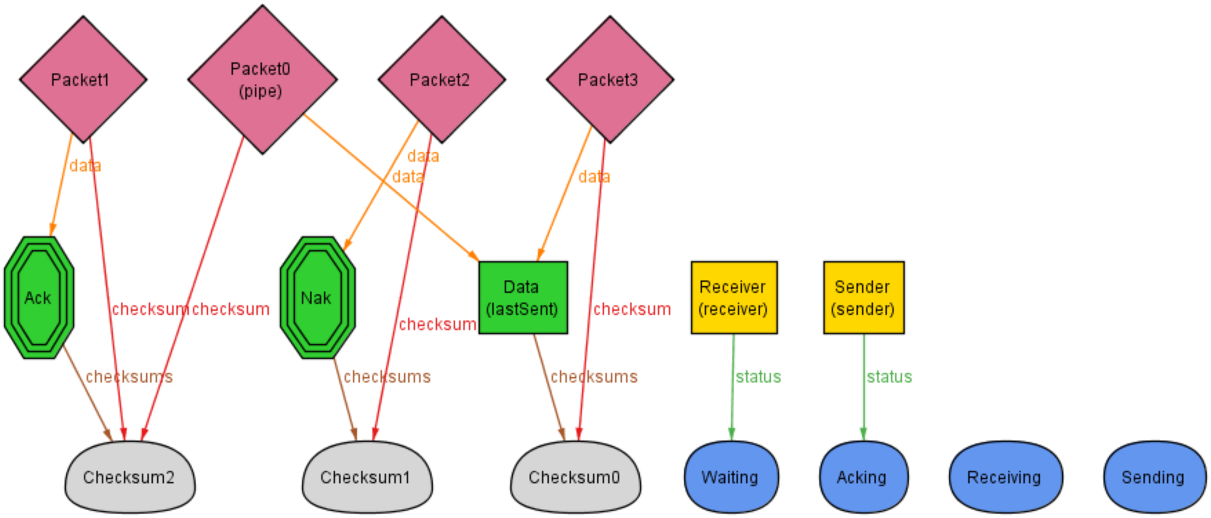
State 1: Sender has data to send, both sender and receiver are waiting



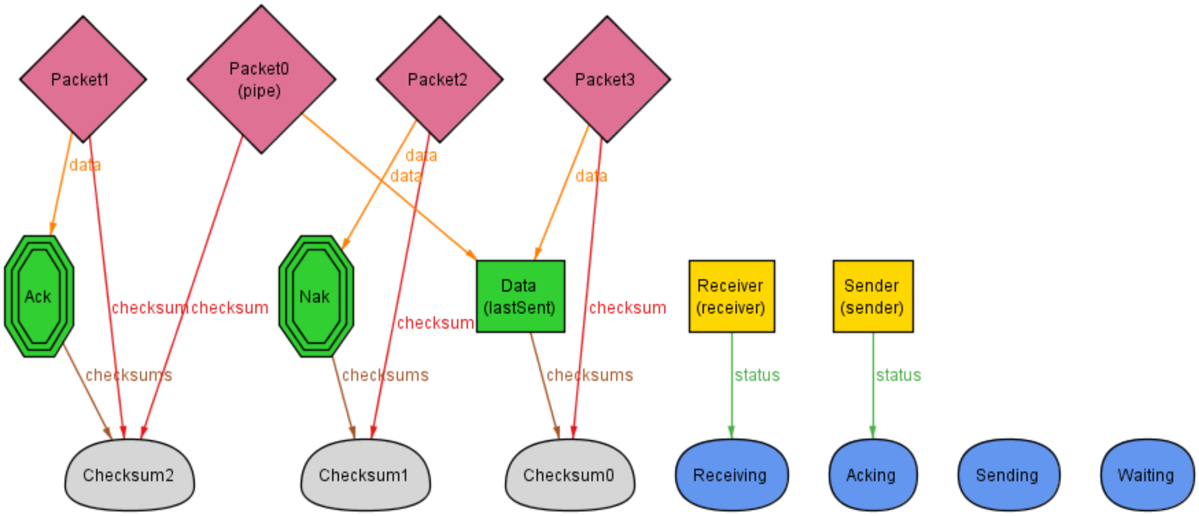
State 2: Sender has gotten into the sending state and is about to transmit data in a packet



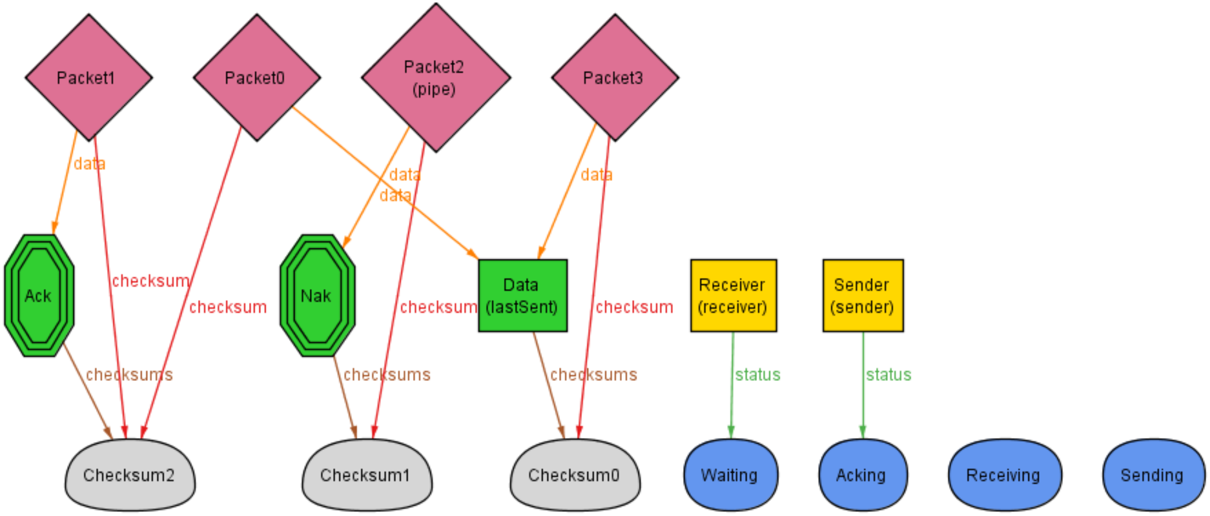
State 3: Sender has put some data in the pipe and has gone to the Acking state



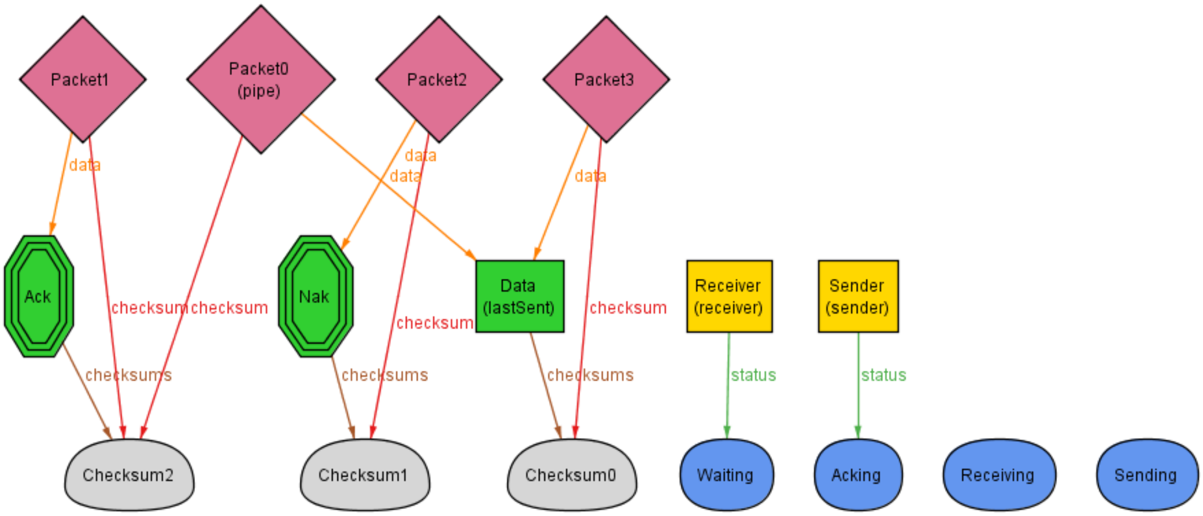
State 4: Receiver has transitions to the receiving state so that it can get the data from the pipe. (notice that this is corrupted data)



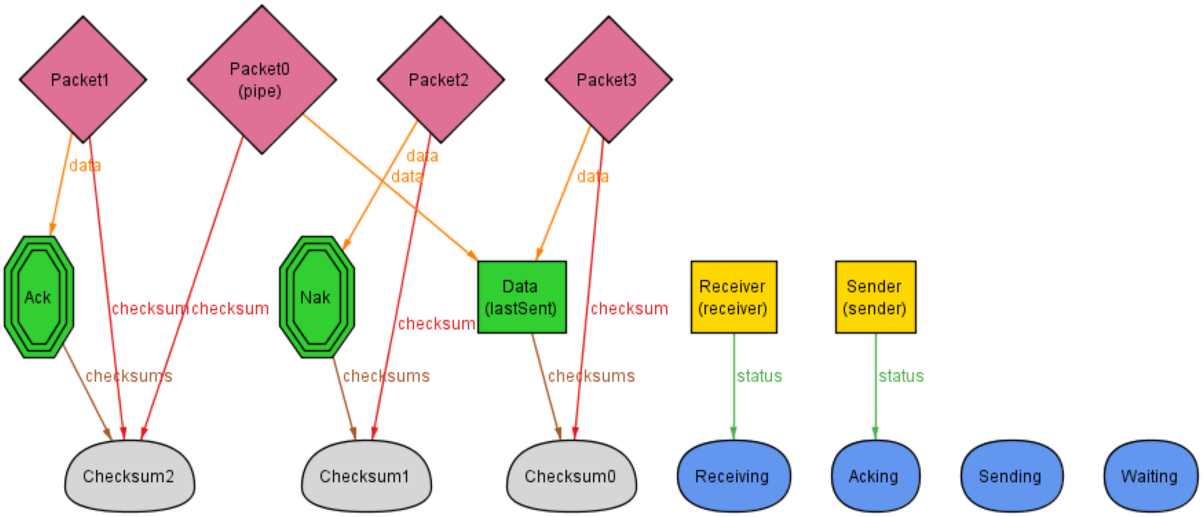
State 5: The receiver has gone back to the Waiting state and realizing the data was currupted has put a Nak packet into the pipe



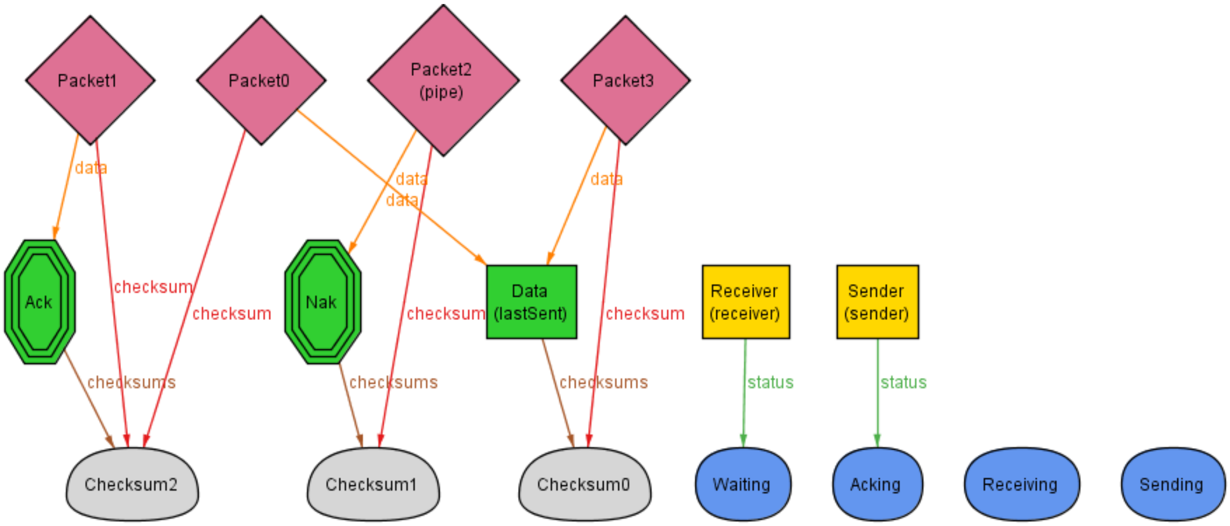
State 6: The Sender got the Nak packet and sent the data again (silly sender sent corrupted data again) and is Acking again



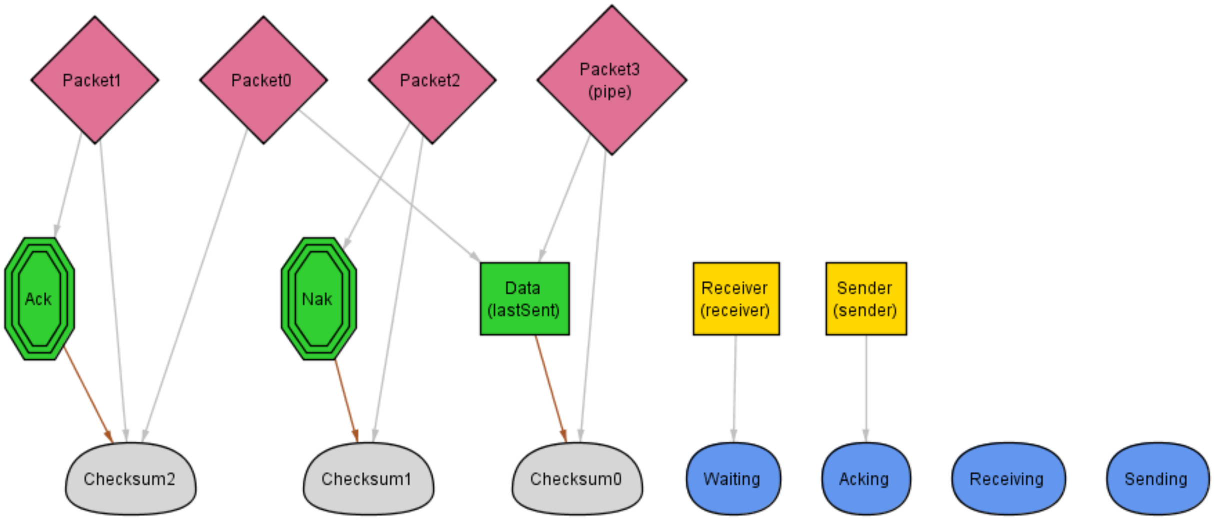
State 7: The receiver detects data intended for it in the pipe and transitions to the receiving state



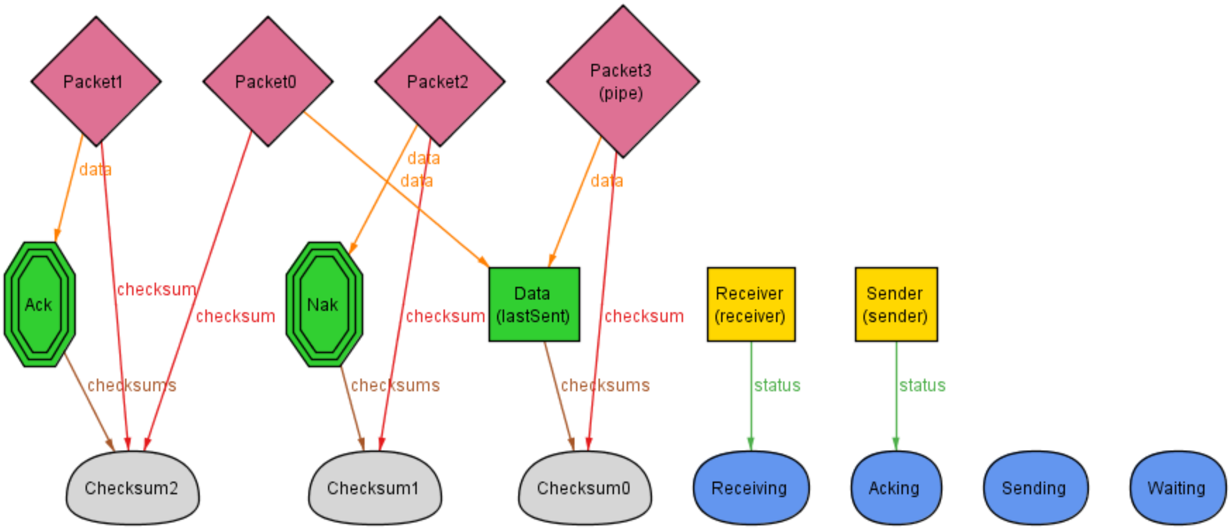
State 8: The receiver has gone back to the Waiting state and realizing the data was currupted has put a Nak packet into the pipe



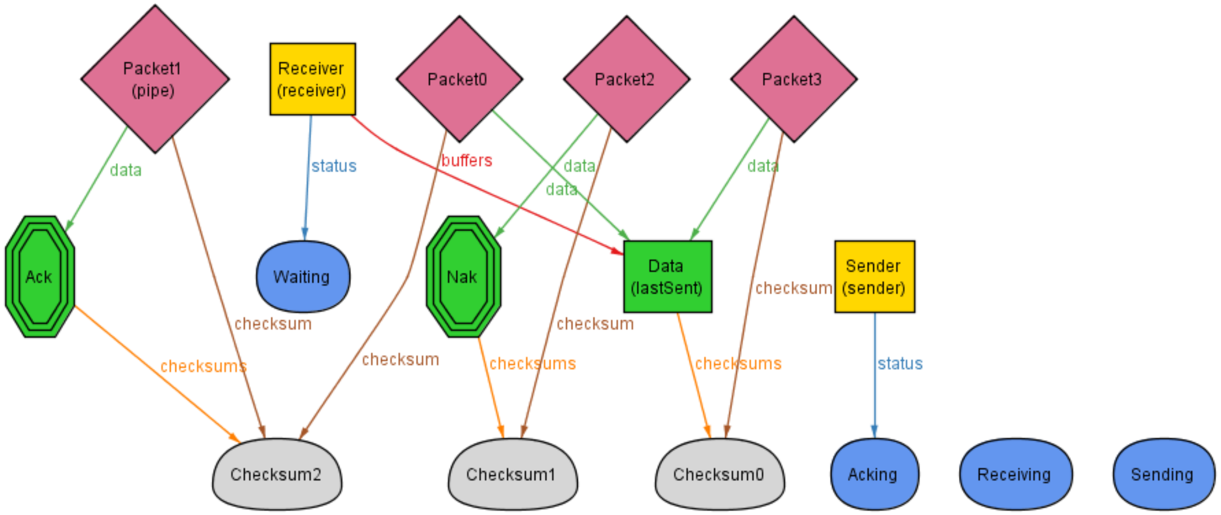
State 9: The Sender got the Nak packet and sent the data again (notice the data is uncorrupted this time) and is Acking again



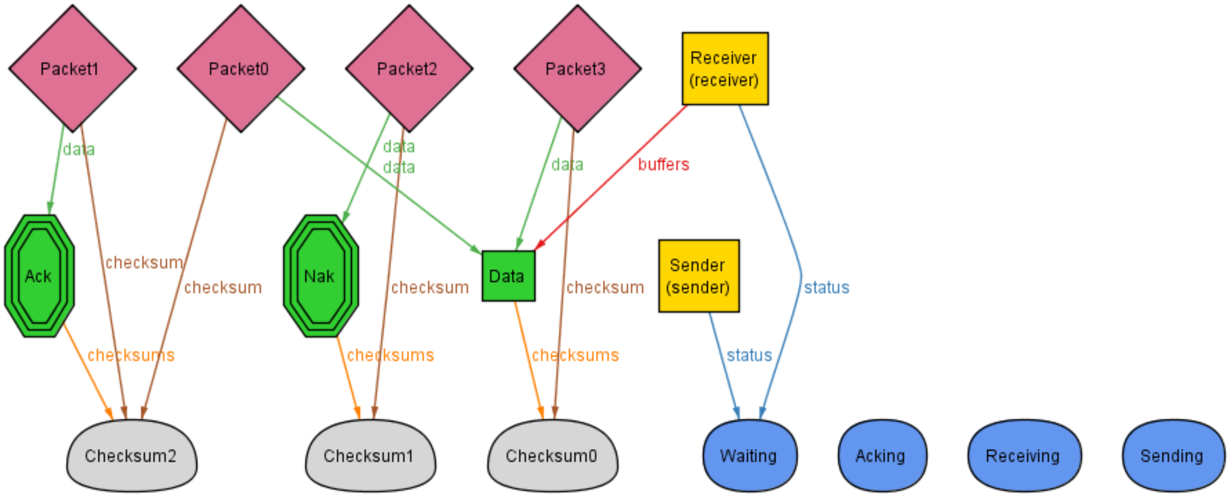
State 10: The receiver detects data intended for it in the pipe and transitions to the receiving state



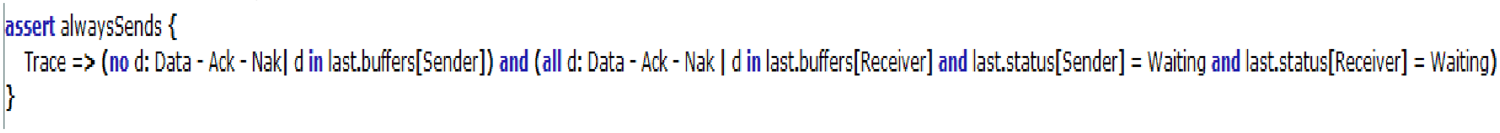
State 11: The receiver pulls the uncorrupted data from the pipe and sends an Ack packet back for the sender



State 12: The sender gets the Ack packet and moved to Waiting. Now both Sender and receiver are waiting and all the data is in the receiver’s buffer.

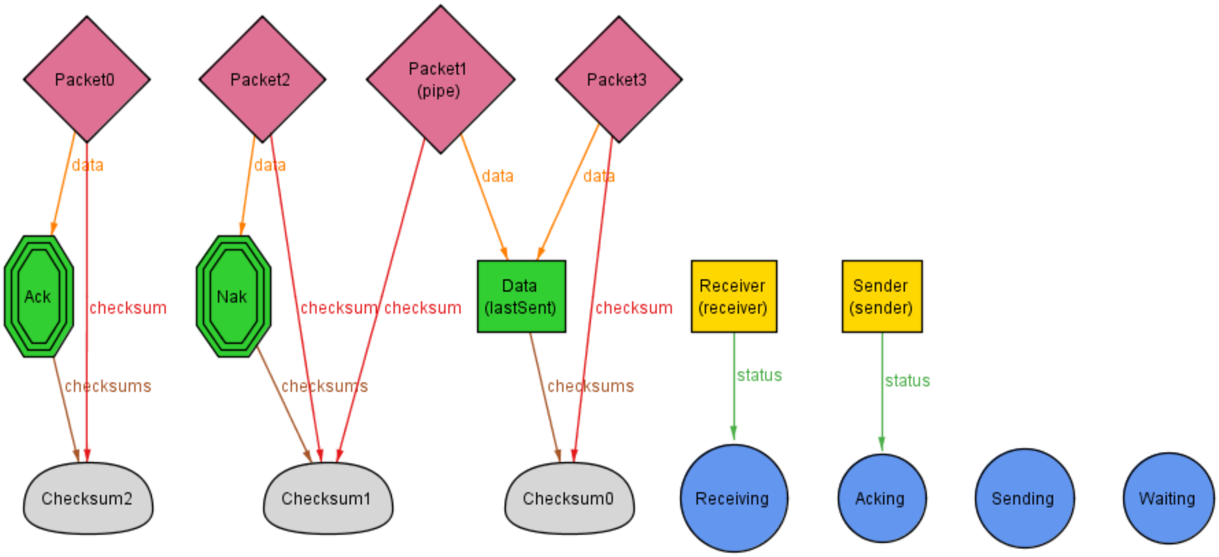


2. Is it always possible to transmit all of the data in the sender’s buffer to the receiver’s buffer? No. The data has to possibility to eternally sent corrupted data and never send the proper data thus it is possible to not send the right data. Another possibility is for the Nak packet to be corrupted. If this happened then the sender wouldn’t know to try to resend the data making the receiver unable to get the correct data. The last possibility is for the sender to send the correct data but for the Ack packet to be corrupted. In this case the data goes into the receicver’s buffer but the sender is going to be in the Acking stage forever because it never received a valid Ack or Nak packet. If the sender only had one packet to send the receiver then all the data will be in the receiver’s buffer at the end but if there are two or more packets to send then the receiver will only receive the first packet of data. We used this assertion to check if it is possible to always send the data:

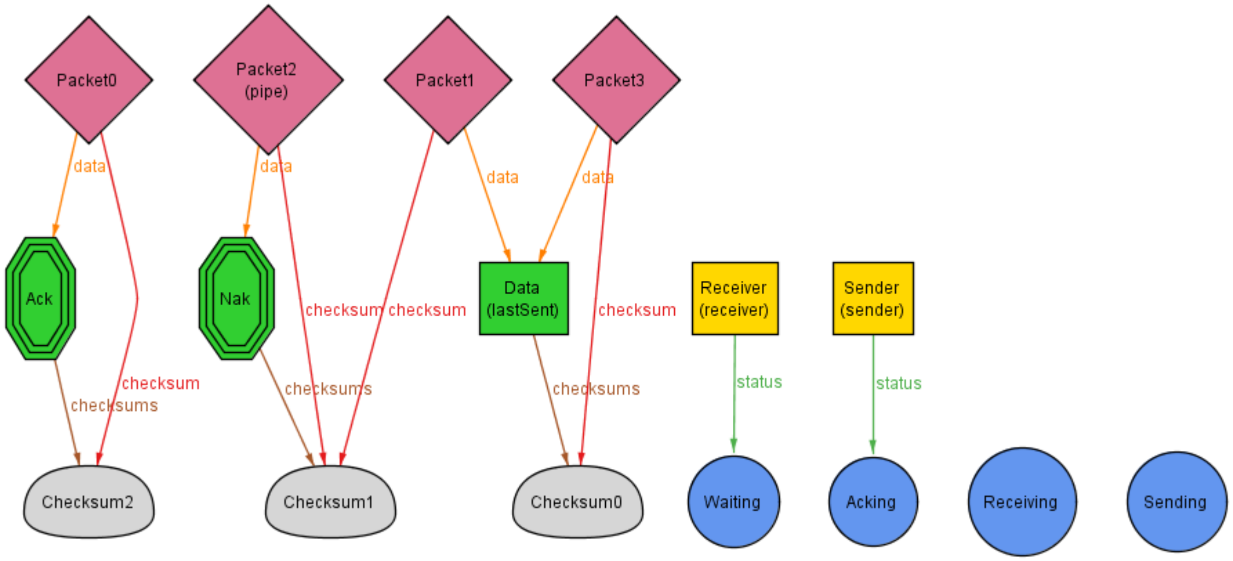


```
assert alwaysSends {
   Trace => (no d: Data - Ack - Nak| d in last.buffers[Sender]) and (all d: Data - Ack - Nak | d in last.buffers[Receiver] and last.status[Sender] = Waiting and last.status[Receiver] = Waiting)
}
```

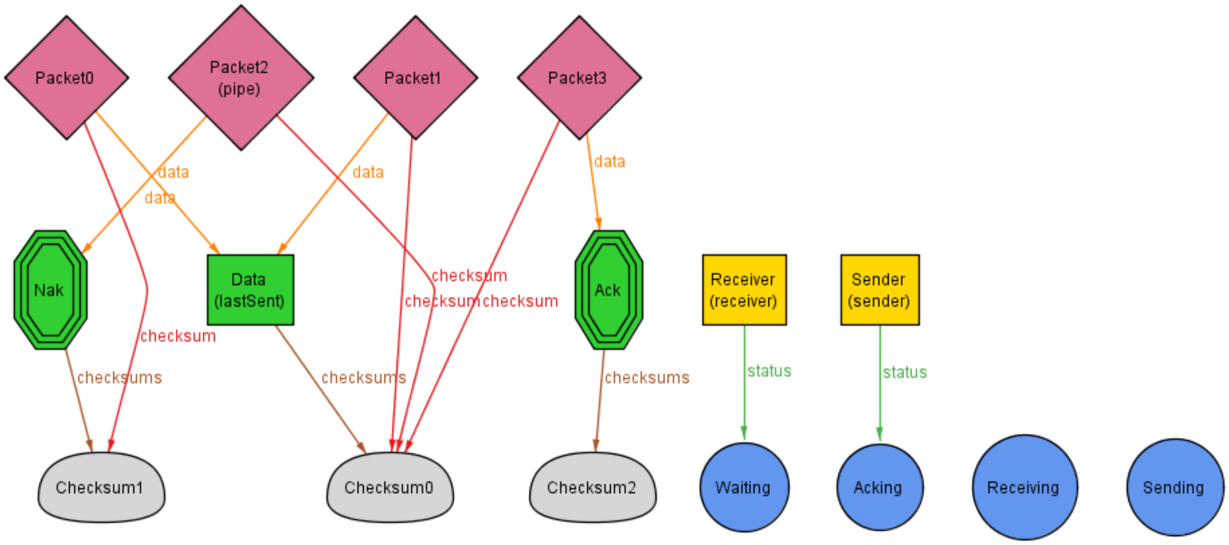
Case 1: This is the case where it contintually sends the wrong data.
In this state It sends corrupted data



In this state the receiver gets the corrupted data and sends a Nak packet back to the sender. It continuously sends the switchs between these two states.



Case 2/3: In this case it sends a corrupted Nak packet then the sender is continually stuck Acking because it is waiting to receive an uncorrupted Nak or Ack packet.


Packet0
Packet2 (pipe)
Packet1
Packet3
data
data
data
data
Nak
Data (lastSent)
Ack
Receiver (receiver)
Sender (sender)
checksum
checksum
checksum
checksum
checksums
checksums
checksums
status
status
Checksum1
Checksum0
Checksum2
Waiting
Acking
Receiving
Sending