CSSE373

Milestone 2

Fred Zhang, Songyu Wang

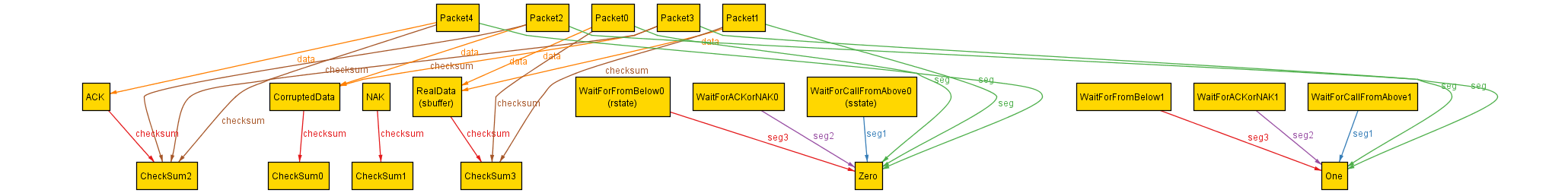
**Using the given protocol, it is possible to transmit all of the data in the sender’s buffer to the receiver’s buffer.**

We ran:

run possibleReliabe for 10 but 7 Time, exactly 1 RealData

The result is:

Time 0



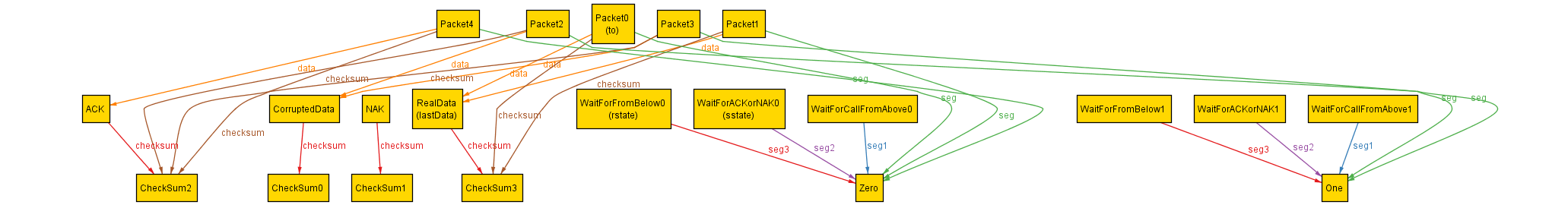
At time 0,

Sender state is Wait for call from above and sequence number is 0

Receiver state is Wait for from below and sequence number is 0

The data we want to send is in sender buffer.

Time 1

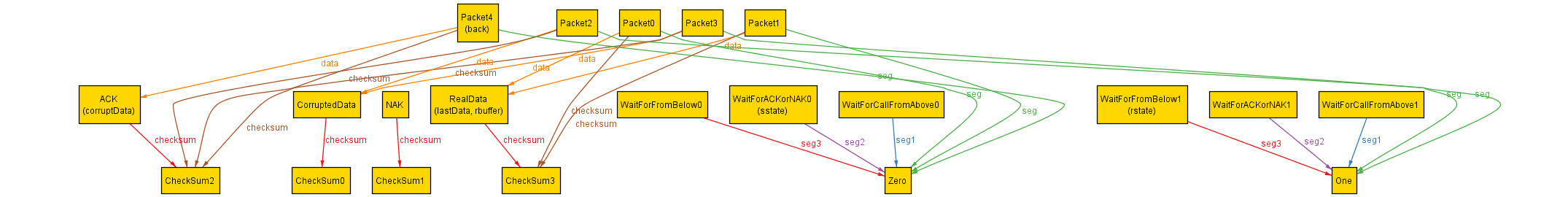


At time 1,

Sender state change to Wait from ACK or NAK and the sequence number is still 0.

The data is packed and put into the channel.

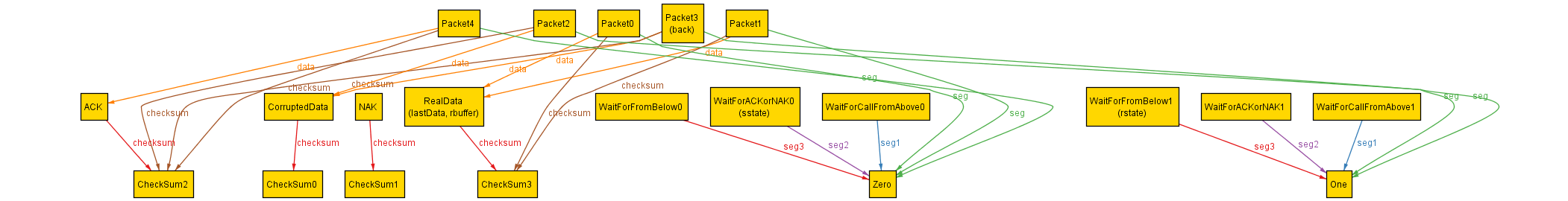
Time 2



At Time 2,

The Receiver receives data, switch its state to Wait for from below with sequence number 1, put the data into receive buffer and put an ACK packet to the channel (The ACK packet will corrupt at next time).

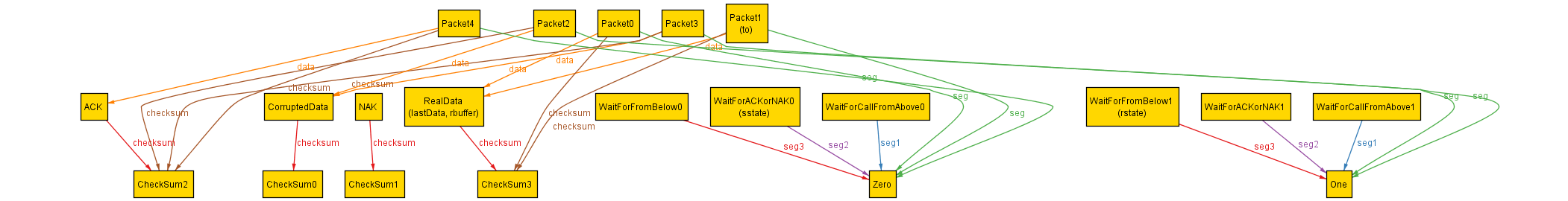
Time 3



At Time3,

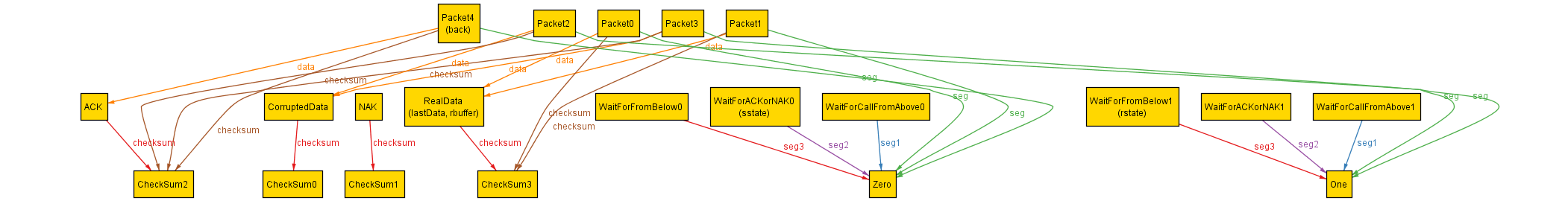
The ACK packet corrupted to CorruptedData.

Time 4



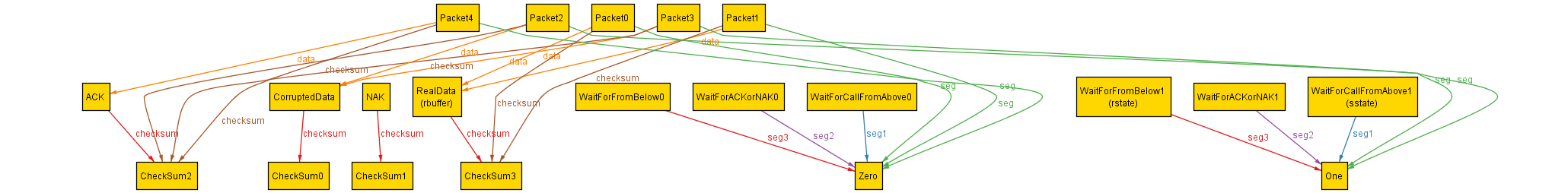
The sender receives the corruptedData and put the ReadData into channel again. No state changes.

Time 5



Receiver gets the data again and put ACK back to the channel. No state changes.

Time 6



Sender gets the ACK and state switches to Wait for a call from above with sequence 1.

And it is done transmit the data.

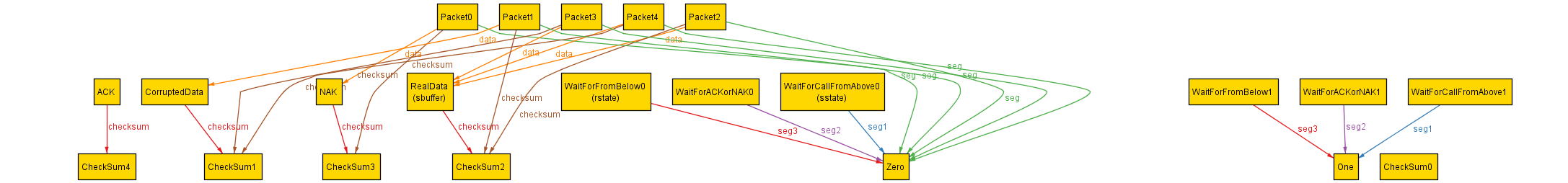
This is a possible way.

**Using the given protocol, it is not** always **possible to transmit all of the data in the sender’s buffer to the receiver buffer.**

We ran

check alwaysReliable for 5 but exactly 8 Time, 2 RealData

And we have a counterexample:

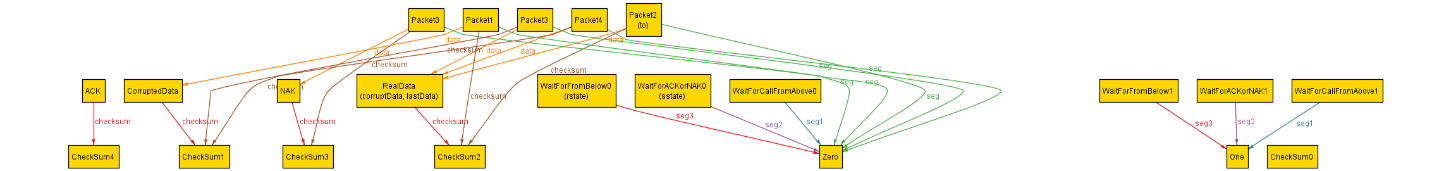


At time 0,

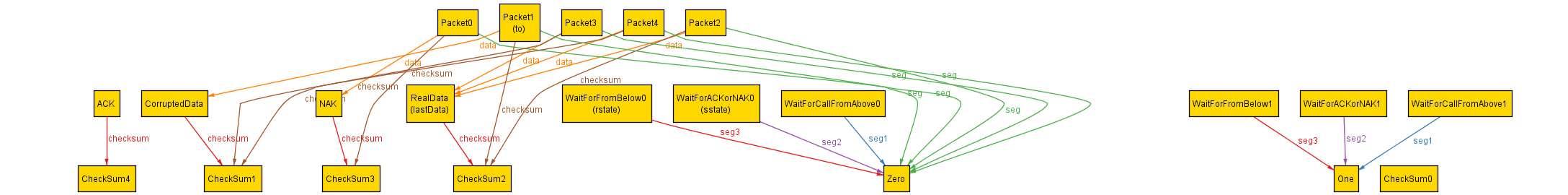
Sender state is Wait for call from above, and sequence number is 0

Receiver state is Wait for from below, and sequence number is 0

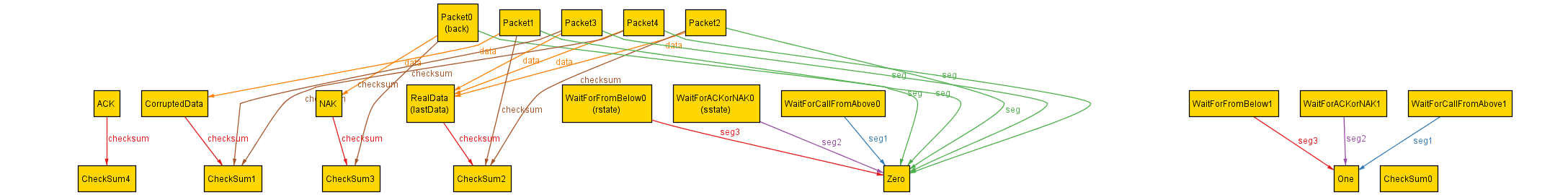
The data we want to send is in sender buffer.



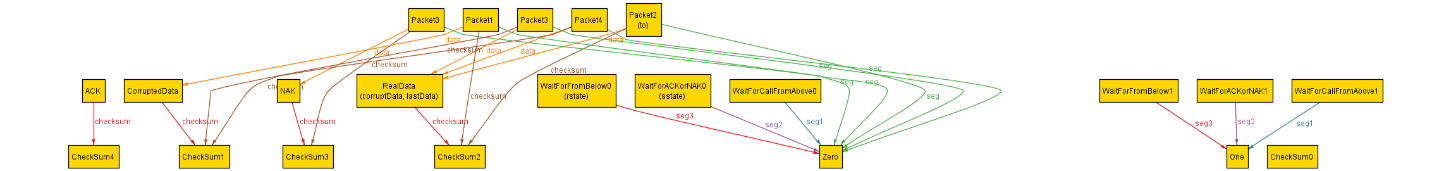
The data is packed and put into the channel.



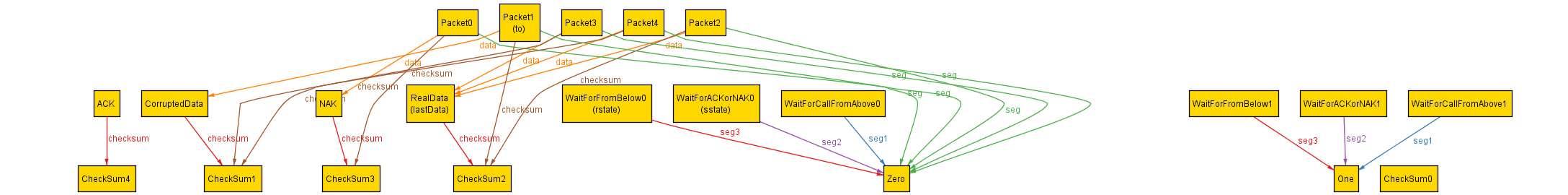
The data is corrupted.



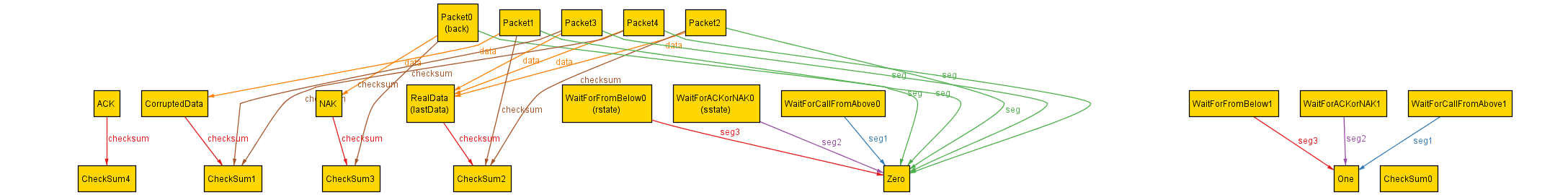
The receiver receives the corrupted data and send NAK back.



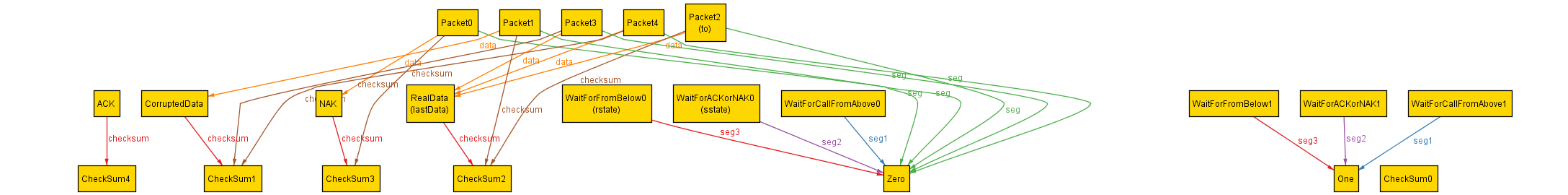
Sender gets the NAK and resend the last data.



The data is corrupted again.



Receiver received corrupted data and sent NAK again.



Sender received NAK and resend that data again.

As you can see above, There exist a “loop”:

Sender sends the data

The packet gets corrupted

Receiver sends NAK to indicate sender that the data is corrupted

Sender send the data

The packet gets corrupted

…

…

Consider this scenario:

Assume the link is not reliable at all, and all packets are corrupted before it arrives, then the sender and receiver would just sending and responding NAK forever.

That is the same scenario described by Alloy.

This protocol cannot guarantee to send all the data from sender to receiver. If there is one packet cannot be sent properly, that data may stick in the channel forever and block the rest of data in the sender.

**If the network guarantees that for every packet there can be no more than one send/receive error in the wire, is it always possible to send the entire data from the sender buffer to the receiver buffer using the protocol.**

We run:

check alwaysReliableWithMaxOneCorruptionPerData for 5 but 10 Time, 2 RealData

check alwaysReliableWithMaxOneCorruptionPerData for 5 but 11 Time, 2 RealData

We have counterexample when we use 10 Times, but not 11 Times.



After seeing the diagram, we found that it is doing the correct behavior just like what we described in the previous section. However, since we only allow 10 Times, the transmitting process does not have enough time to finish. That is why we have this counterexample. We concluded that this is not a flaw in the protocol. It is a problem of how Alloy model the protocol.

So, there is no real counterexample that disproves the property. This property holds.