CSSE373

Milestone 3, RDT2.2

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.**

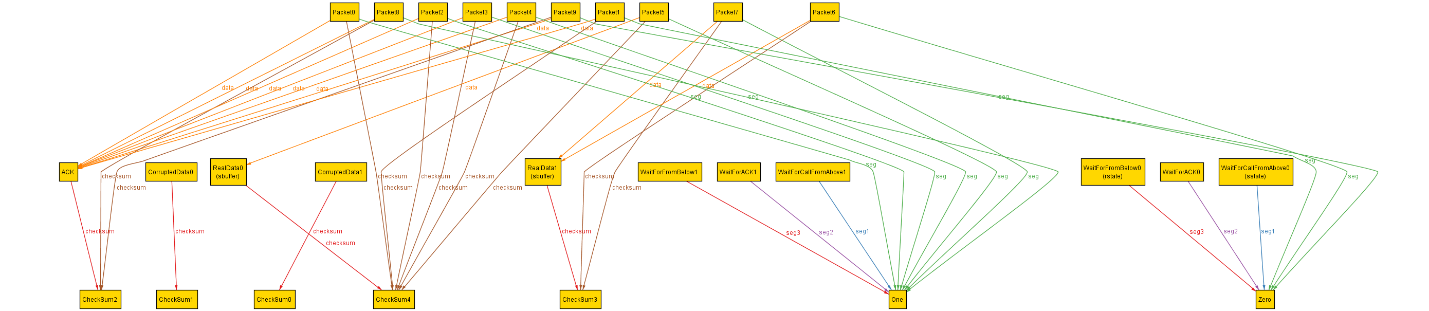
We ran:

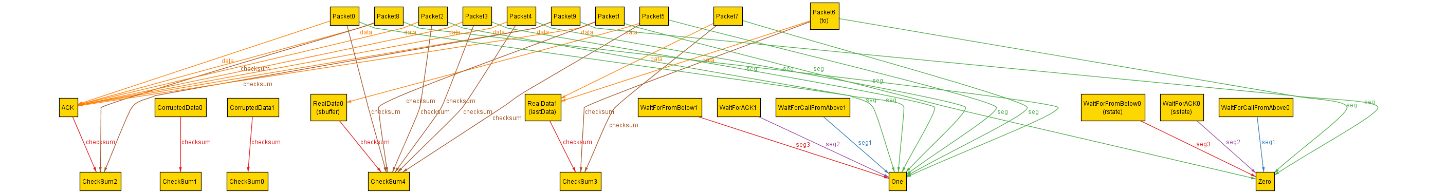
run possibleReliabe for 10 but 7 Time, exactly 2 RealData

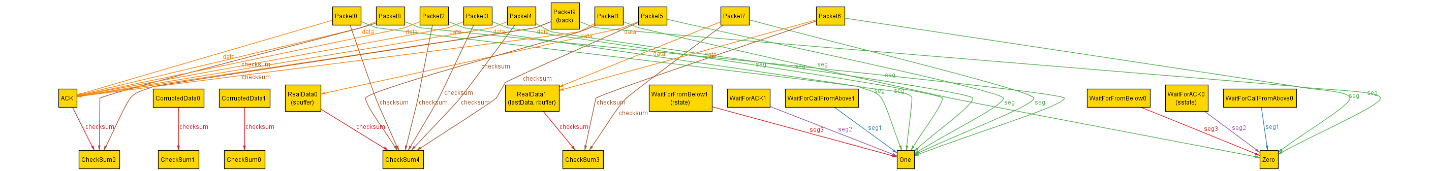
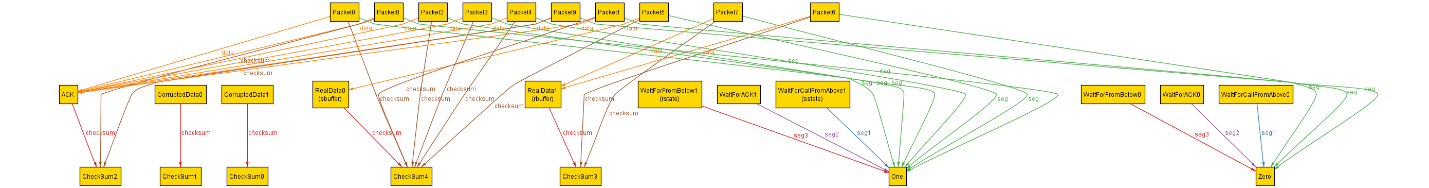
The result is:

The images get a bit large and hard to read. I think it is probably easier for you to run it yourself.

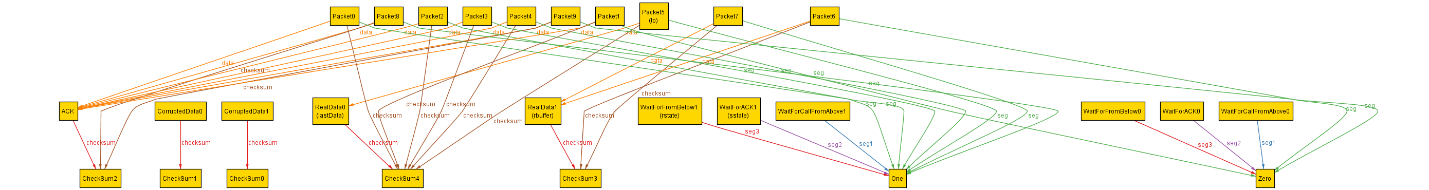
Time0: RealData0 and RealData1 is in sbuffer, rbuffer and links are empty. Sender is in WaitForCallFromAbove0 state, while the receiver is in WaitforFromBelow0 state.

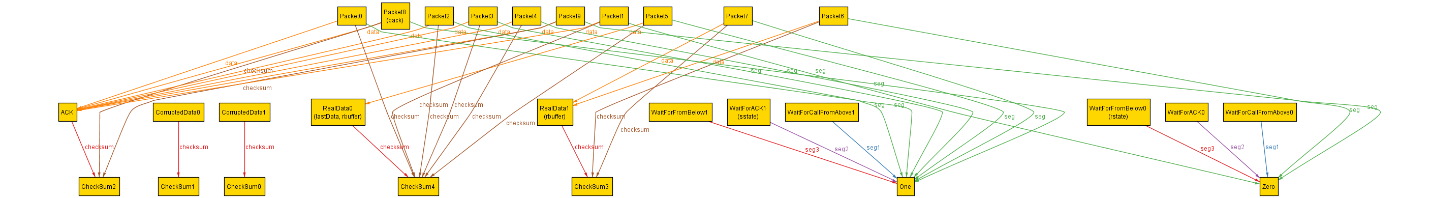


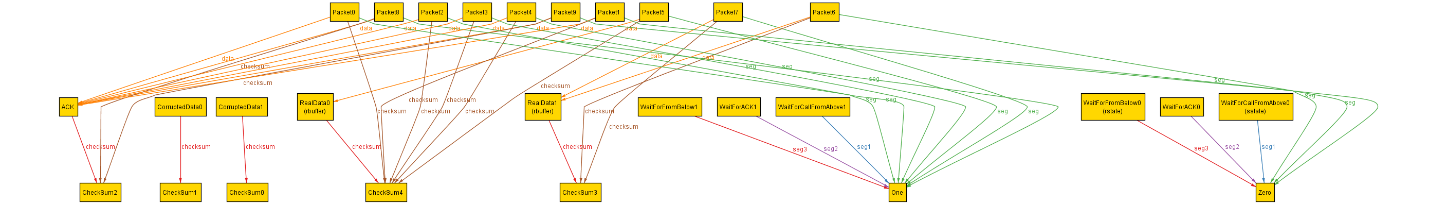
Time1: packet6 is sent with RealData1. It appears in the to link and lastData of sender. Sender switches to WaitForACK0 state, while the receiver is still in WaitforFromBelow0 state.

Time2: Packet6 arrived and the receiver replies with ACK in Packet9. The receiver switches to WaitforFromBelow1 after sending the ACK. Time3: The receiver gets the ACK in packet9 and switches to WaitForCallFromAbove1.

Time4: RealData2 gets sent in similar sequence as above. Packet5 is sent with RealData0. It appears in the to link and lastData of sender. Sender switches to WaitForACK1 state, while the receiver is still in WaitforFromBelow1 state.



Time5: Packet5 arrived and the receiver replies with ACK in Packet8. The receiver switches to WaitforFromBelow0 after sending the ACK. Time6: The receiver gets the ACK in packet9 and switches to WaitForCallFromAbove1.

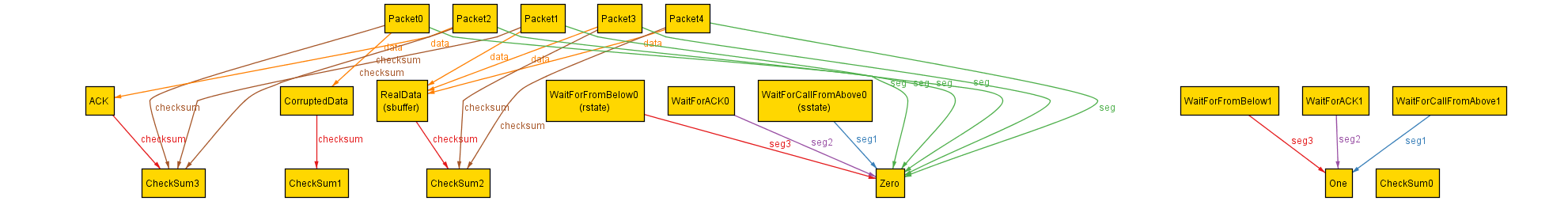


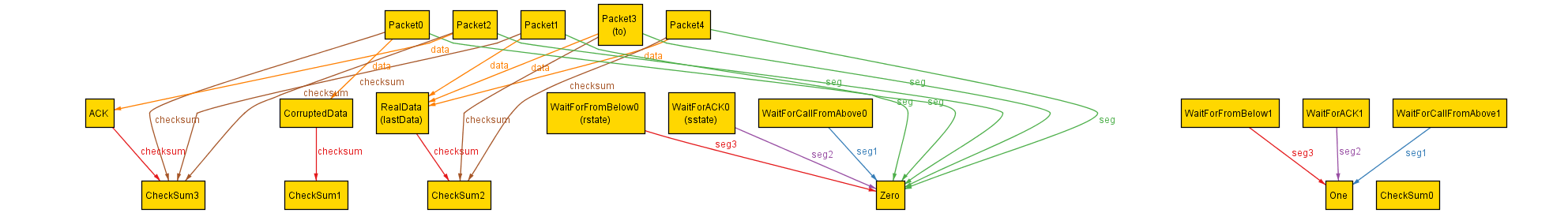
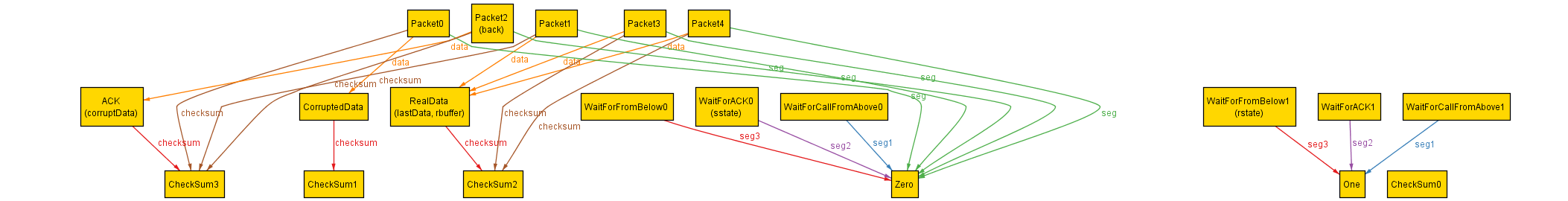
**Using the given protocol, it is not** always **possible to transmit all of the data in the sender’s buffer to the receiver buffer.**

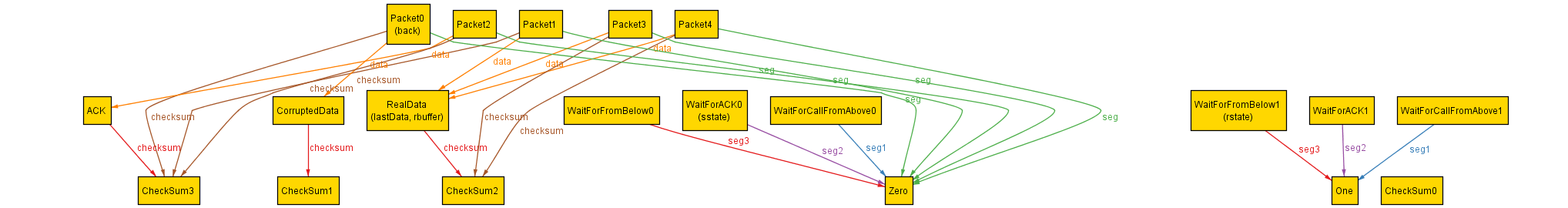
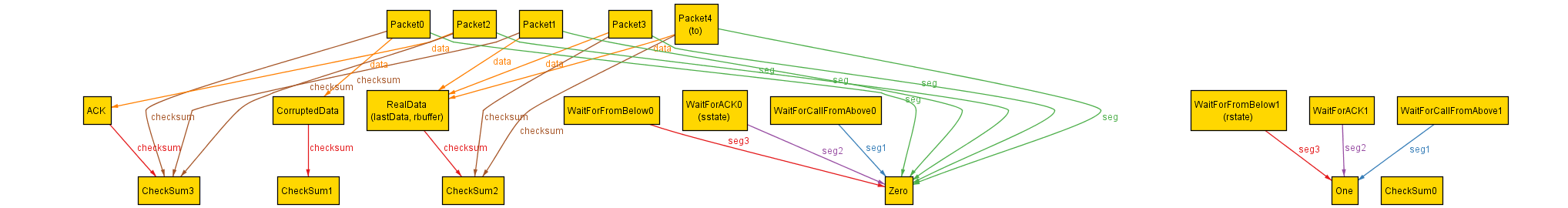
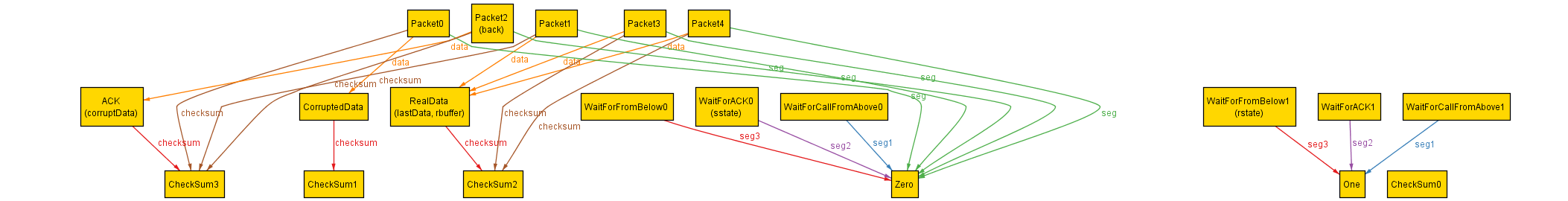
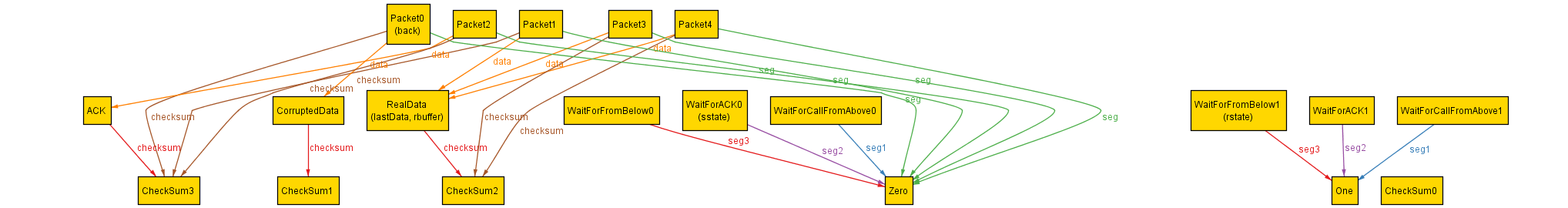
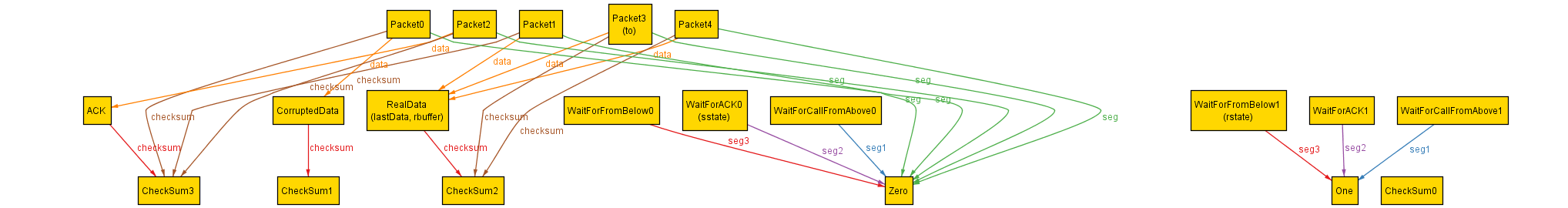
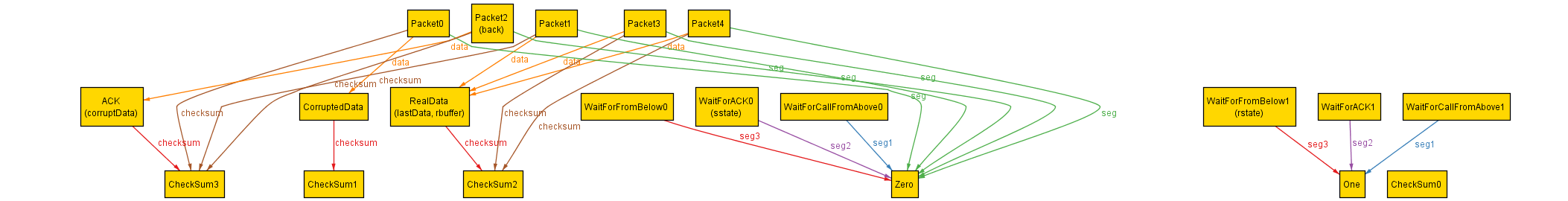
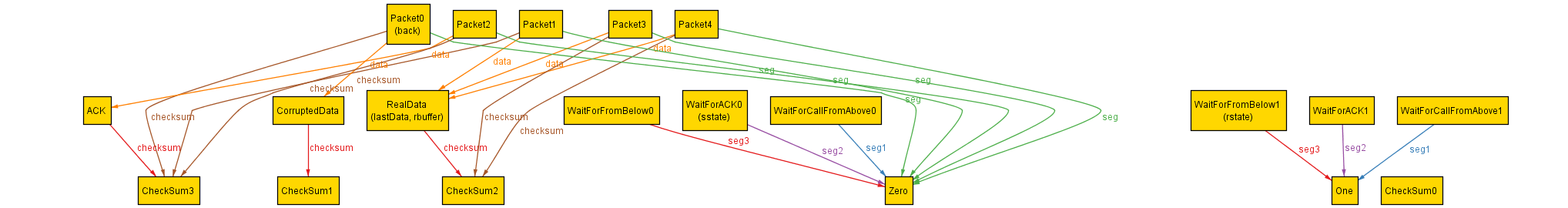
We ran

check alwaysReliable for 5 but exactly 10 Time, 1 RealData

Time0: RealData is in sbuffer, rbuffer and links are empty. Sender is in WaitForCallFromAbove0 state, while the receiver is in WaitforFromBelow0 state.



Time1: packet3 is sent with RealData0. It appears in the to link and lastData of sender. Sender switches to WaitForACK0 state, while the receiver is still in WaitforFromBelow0 state. Time2: Packet3 arrived and the receiver replies with ACK in Packet2. The receiver switches to WaitforFromBelow1 after sending the ACK. Notice that ACK is highlighted with corruptData, this suggests that Packet2 is going to be corrupted in the next state.

Time3: Packet3 corrupted at this moment and becomes Packet0. It is still traveling in the back link.  Time4: The sender receives Packet0 and found out that it is corrupted, so that it resends the lastData in Packet4. The sender state remains to be WaitForACK0, because it gets a corrupted one.Time5: Packet4 arrived and the receiver replies with ACK in Packet2. The receiver switches to WaitforFromBelow1 after sending the ACK. Notice that ACK is highlighted with corruptData, this suggests that Packet2 is going to be corrupted again in the next state. Time6: Packet2 corrupted at this moment and becomes Packet0. It is still traveling in the back link.  Time7: The sender receives Packet0 and found out that it is corrupted, so that it resends the lastData in Packet3. The sender state remains to be WaitForACK0, because it gets a corrupted one. Time8: Packet4 arrived and the receiver replies with ACK in Packet2. The receiver switches to WaitforFromBelow1 after sending the ACK. Notice that ACK is highlighted with corruptData, this suggests that Packet2 is going to be corrupted again in the next state. Time9: Packet2 corrupted at this moment and becomes Packet0. It is still traveling in the back link. 

We think 10 Time step is reasonable for one data to be transmitted.

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

The sender sends the data

The receiver gets the data and send a ACK

The ACK gets corrupted

Sender receives the corrupted ACK, and resends the data

…

…

Consider this scenario:

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. Without limiting the number of times packet can corrupt, it gets into a corrupt and resends loop.

**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 ran

check alwaysReliableWithMaxOneCorruptionPerData for 5 but 11 Time, 2 RealData

check alwaysReliableWithMaxOneCorruptionPerData for 5 but 12 Time, 2 RealData

The first one returns us a counter example, while the second gives no counter example.

Since we ensure that each Data can only be corrupted once, only RealData1, RealData2 and ACK can gets corrupted. Each corruption would consume 3 cycles. Therefore, 3\*3 = 9 times are wasted in resolving corrupted data, and we still need 2 cycles to send the Data.

Thus, 11 Time does not guarantee transimission, but 12 Time is enough.

According to Alloy, if we have enough time, in this case 12Time, the data would be transmitted reliabled.