CSSE373

Milestone 2

Fred Zhang, Songyu Wang

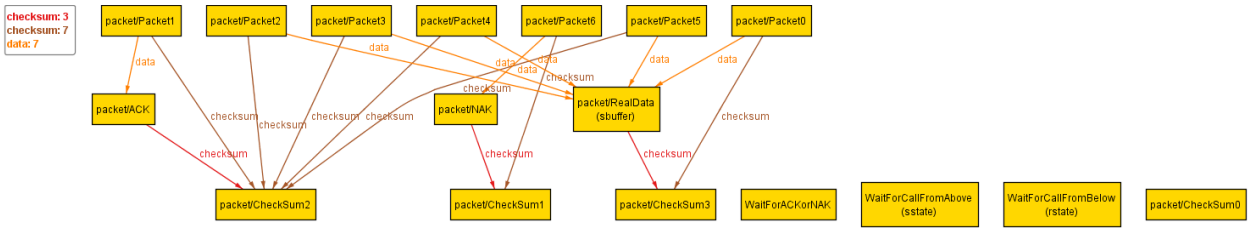
Using the given protocol, it is possible to transmit all of the data in the sender's buffer to the receiver's buffer.

We ran:

run possibleReliabe for 7 but exactly 1 RealData

The result is:

Time0

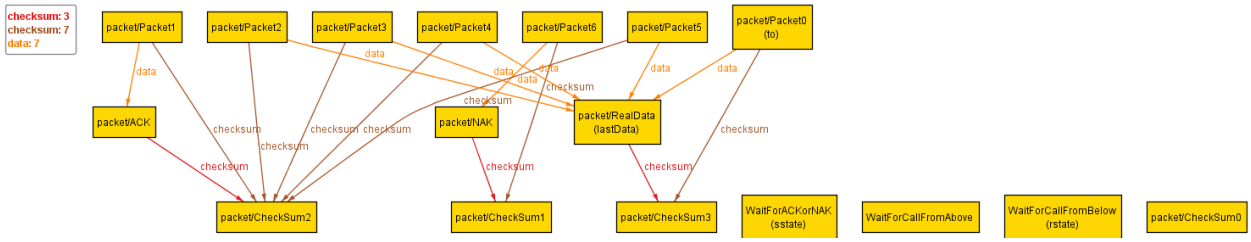


At Time 0, The RealData is in sender buffer.

Sender state is Wait for Call From Above

Receiver state is Wait for Call From Below

Time1

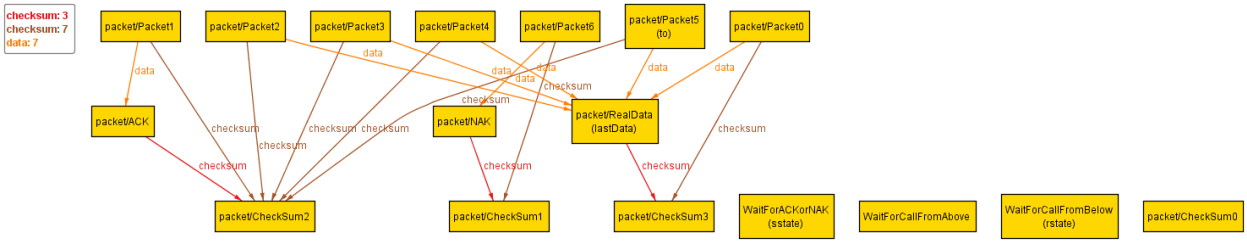


At Time 1, The RealData is packed in Packet0 and put in the channel. It is waiting for being received by Receiver.

Sender state is Wait for ACK or NAK

Receiver state is Wait for Call From Below

Time2

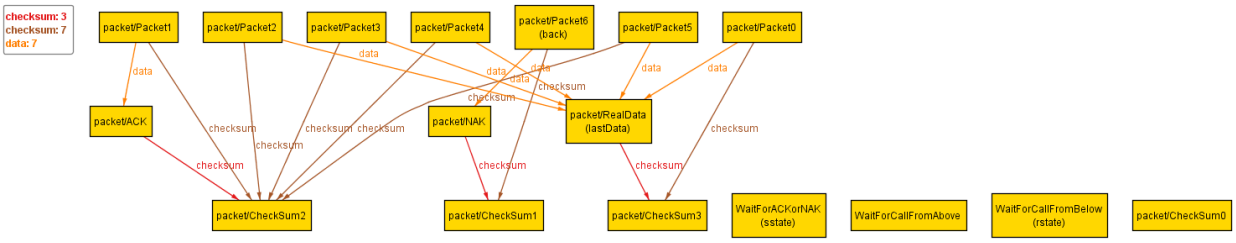


At Time 2, The RealData is corrupted (Packet0 becomes Packet5)

Sender state is Wait for ACK or NAK

Receiver state is Wait for Call From Below

Time3

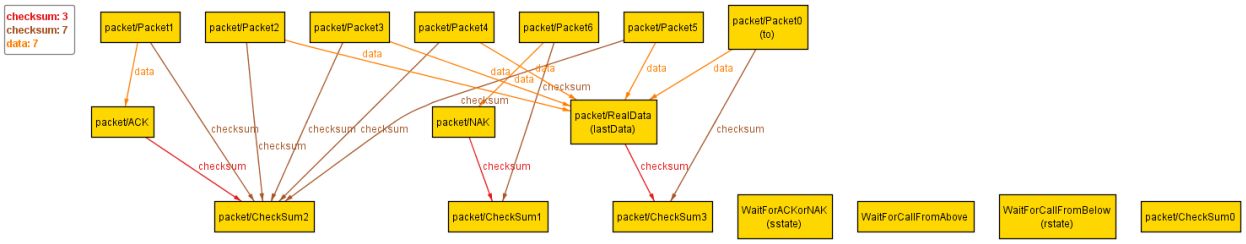


At Time 3, An NAK packet is sent by receiver because it detects the corrupted data

Sender state is Wait for ACK or NAK

Receiver state is Wait for Call From Below

Time4

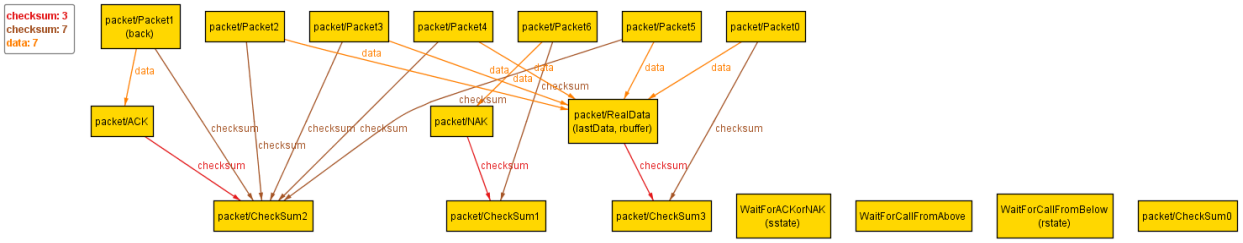


At Time 4, Sender receives the NAK data and Resend RealData/Packet0 into the channel.

Sender state is Wait for ACK or NAK

Receiver state is Wait for Call From Below

Time5

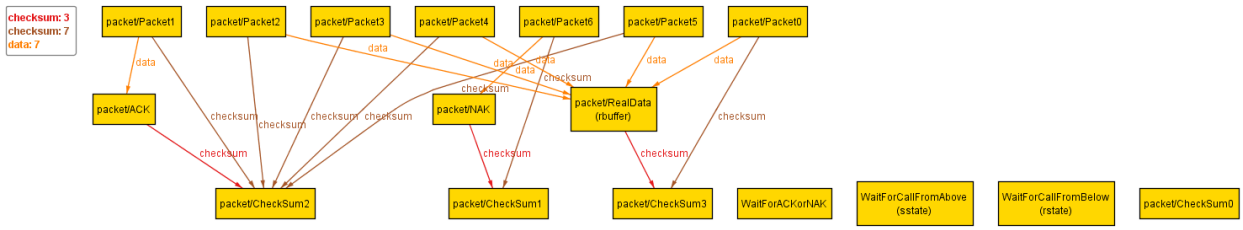


At Time 5, Receiver receives the uncorrupted data. The data is in the receiver buffer. Receiver sends an ACK message.

Sender state is Wait for ACK or NAK

Receiver state is Wait for Call From Below

Time6



At Time 6, Sender receives the ACK message. Since there is no more data to send. This is the last state.

Sender state is Wait for Call From Above

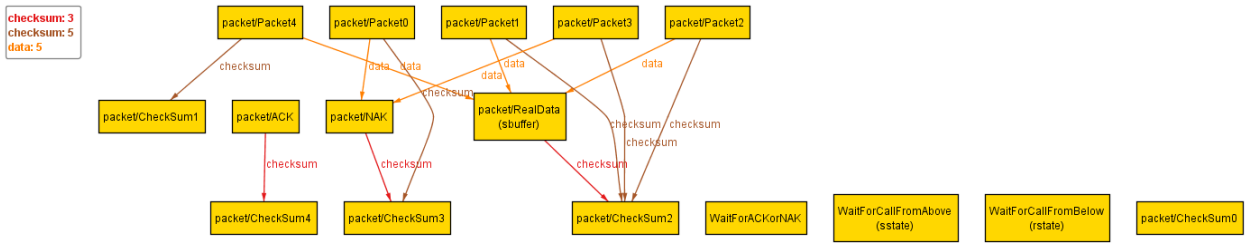
Receiver state is Wait for Call From Below

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

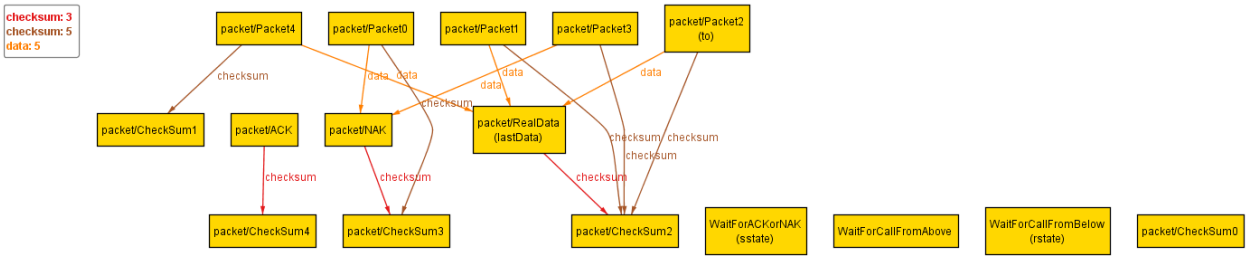
We ran

check alwaysReliable for 5 but exactly 8 Time, 1 RealData

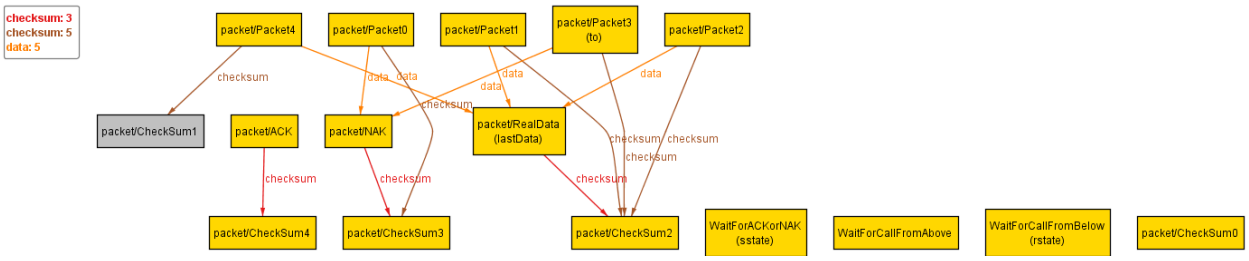
And we have a counterexample:



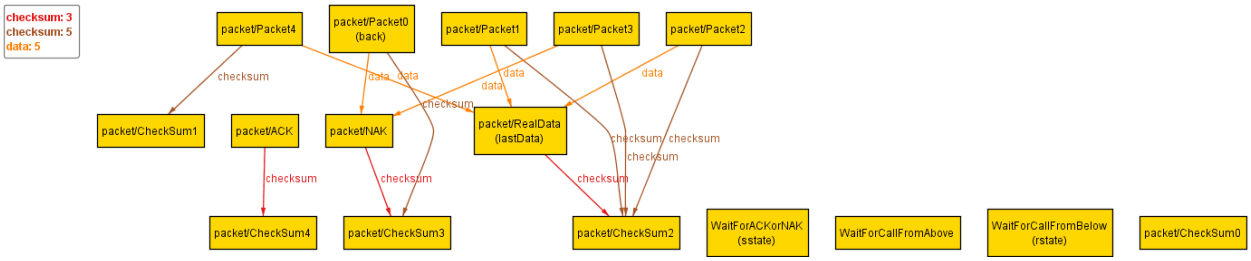
Sends the data, so the lastData becomes packet/RealData, packet2 is in the link



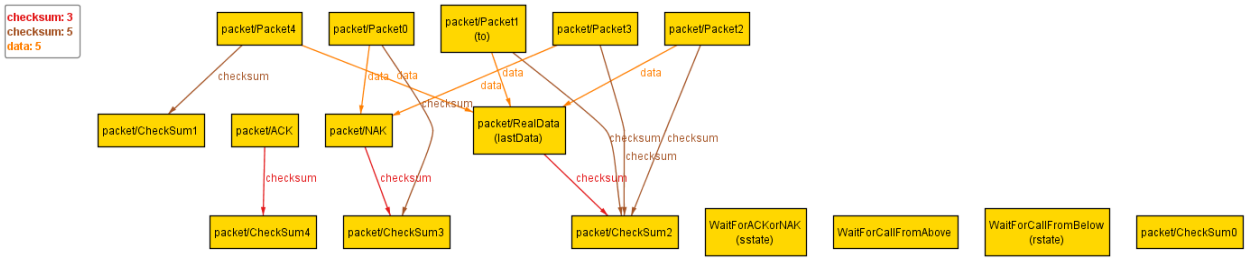
The packet gets corrupted.



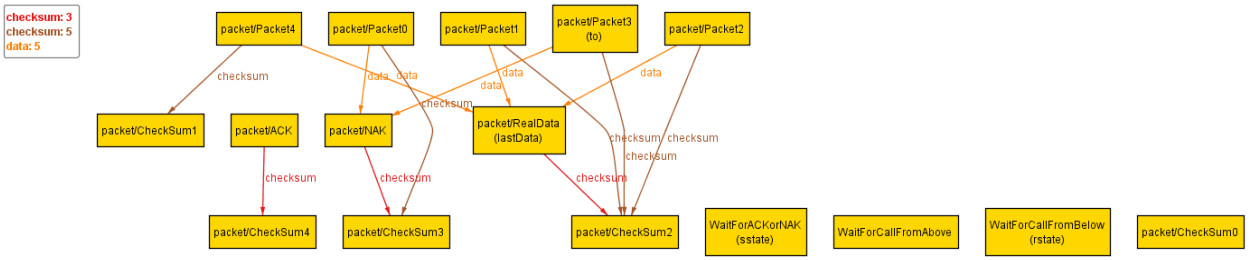
The corrupted packet arrived and the reciever sends an NAK.



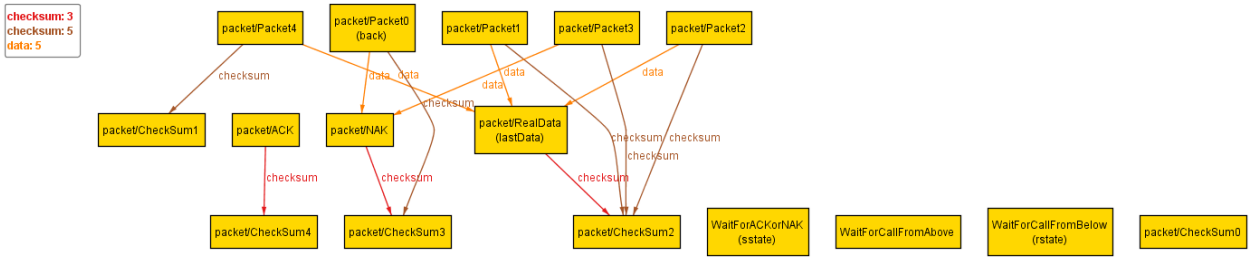
The NAK reaches the sender so the it resends the data again in Packet1.



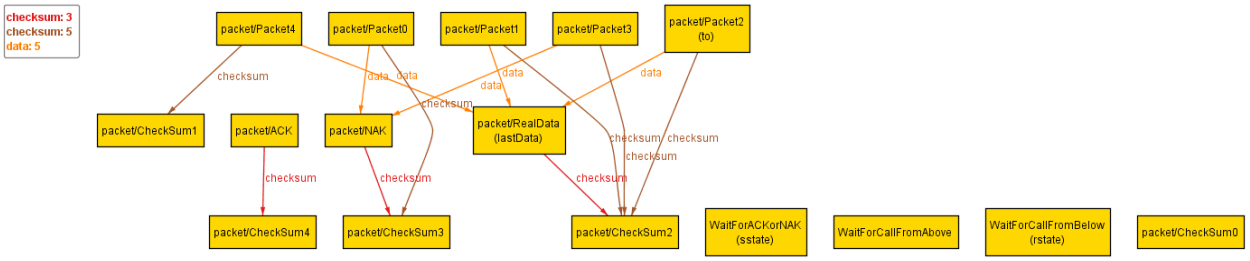
The packet gets corrupted again from Packet1 to Packet3



The corrupted packet arrived and the reciever sends an NAK.



The NAK reaches the sender so the it resends the data again in Packet1.



We think 8 Time step is reasonable for one data to be transmitted.

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

Sender sends the data

The packet gets corrupted (In our case, the packet corrupted and became an ACK packet)

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 corrputed before it arrives, then the sender and reciever 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.

To make this protocol reliable, we probably need to provide an upper bound probability of the corruption, so that the message gets throw the majority of the time. If we allow arbitrary corruption in the link, the packet can easily get lost over and over again.

Also, this protocal also has a nother flaw that, if a corrupted packet happens to pass the checksum test by flipping proper number of bits, the corrupted data can still gets through somehow. In network, it is hard to achieve 100% corruption detection.