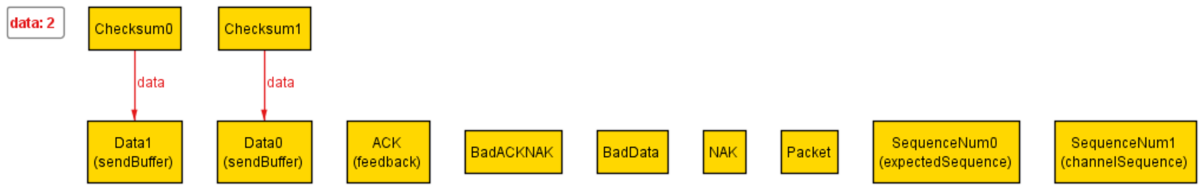
CSSE373: Project Milestone 3, RDT 2.1
Sean Carter and Austin Willis ("The Awesome Team")

Property 1 Screenshots

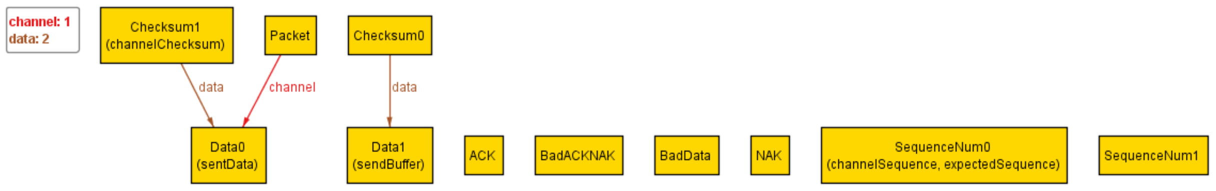
Key: sendBuffer is the sender's buffer, recvBuffer is the receiver's buffer

State0 (init)



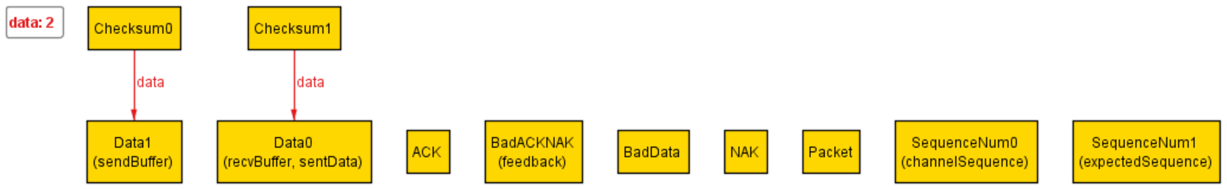
All data is in the sendBuffer, none is in the recvBuffer. The default feedback is set to ACK, deach piece of Data has its own checksum. The receiver is expecting the next incoming packet ot have Sequence Num 0.

State1



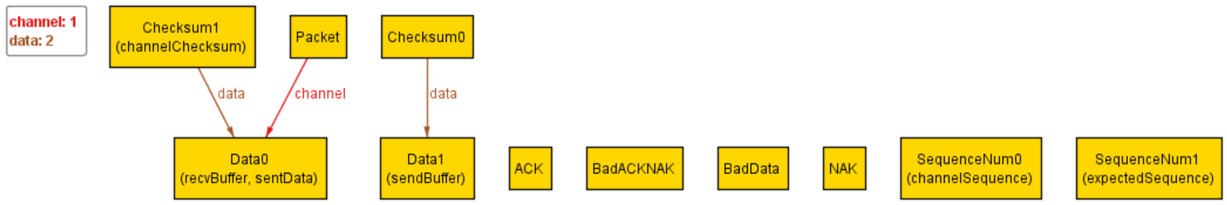
Data0 is put into the channel to be sent and is not corrupted during transmission. It is given the SequenceNum of 0, the same as what the receiver is expecting.

State2



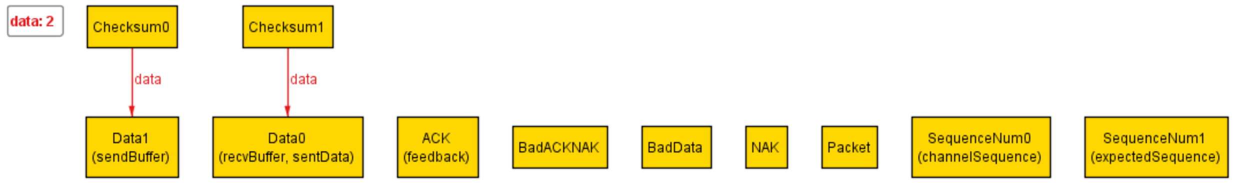
The receiver has received the uncorrupted Data0, and because the sequence numbers matched, it has placed it into its recvBuffer. Thus is will change its expectedSequence to 1. Because it received uncorrupted data, it sends a feedback of Ack. However, the Ack is corrupted during transmission and will be received as a BadAckNak instead.

State3


channel: 1
data: 2
Checksum1 (channelChecksum)
Packet
Checksum0
data
channel
data
Data0 (recvBuffer, sentData)
Data1 (sendBuffer)
ACK
BadACKNAK
BadData
NAK
SequenceNum0 (channelSequence)
SequenceNum1 (expectedSequence)

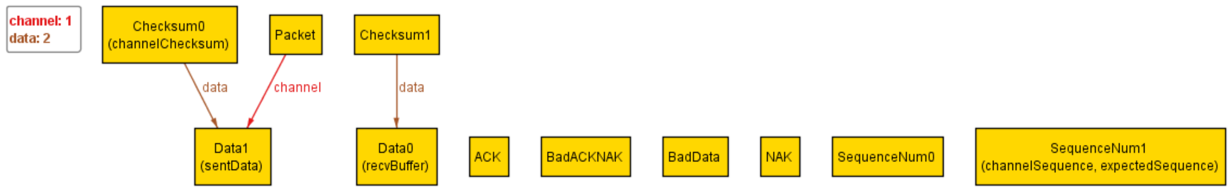
The sender, having received a BadAckNak, does not know whether or not the receiver received corrupted or uncorrupted data. So it resends Data0 and keeps its SequenceNum at 0.

State4



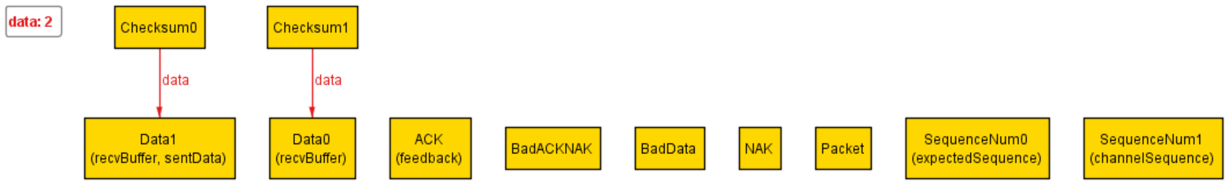
The receiver receives Data0 again. Because the data was not corrupt, it sends back an ACK. However, because the SequenceNum was 0, and not the 1 it was expecting, it does not add Data0 to the recvBuffer (because it already has). It keeps its expected SequenceNum as 1.

State5



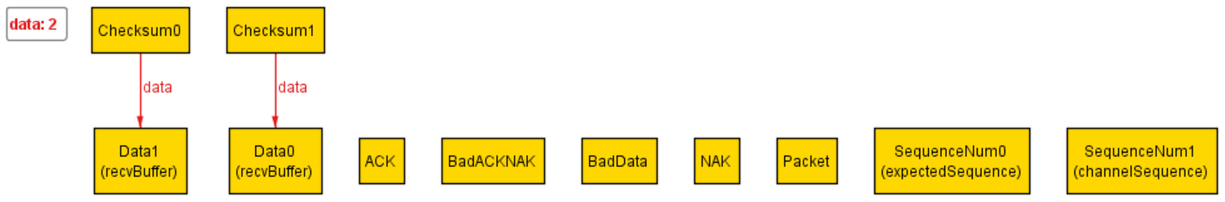
The sender receives the receiver's ACK, and knows that it properly received Data0. The sender now sends Data1 with a SequenceNum of 1. Data1 does not get corrupted during transmission.

State6



The receiver receives an uncorrupted Data1 with the expected SequenceNum of 1. It sends feedback of Ack.

State6

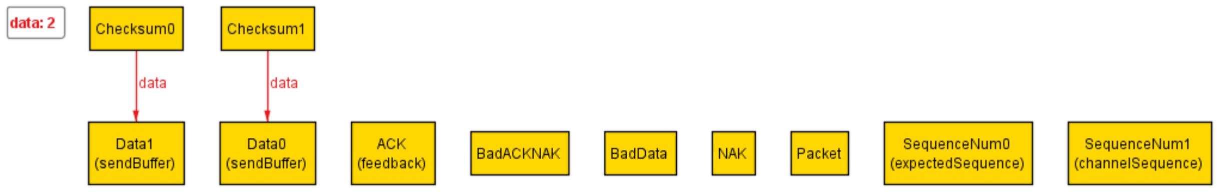


The sender receives the receiver’s Ack and knows that Data1 has been received uncorrupted.

Property 2

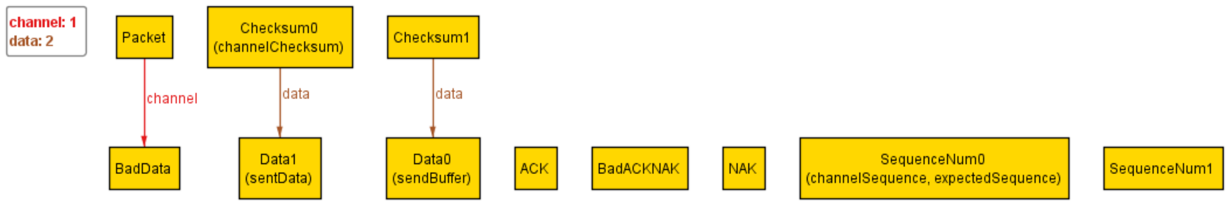
Our assertion found an instance where a combination of data being corrupted during transmission from the sender to the receiver and the feedback being corrupted during transmission from the receiver to the sender cause the receiver to never receive all of the sender’s data.

State0 (init)



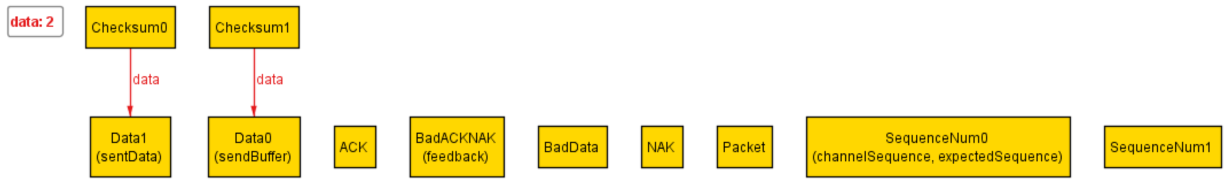
All data is in the sendBuffer, none is in the recvBuffer. The default feedback is set to ACK, deach piece of Data has its own checksum. The receiver is expecting the next incoming packet ot have Sequence Num 0.

State1



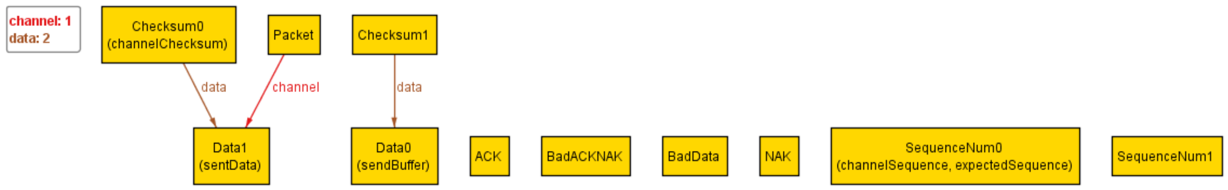
The sender tries to send Data1 with a SequenceNum of 0 (what the receiver is expecting). However, the data is corrupted during transmission.

State2



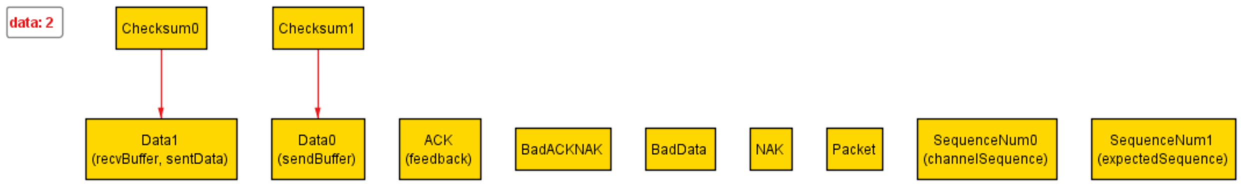
The receiver receives corrupted data, and tries to return a NAK to the sender as feedback. However, the NAK is corrupted during transmission.

State3



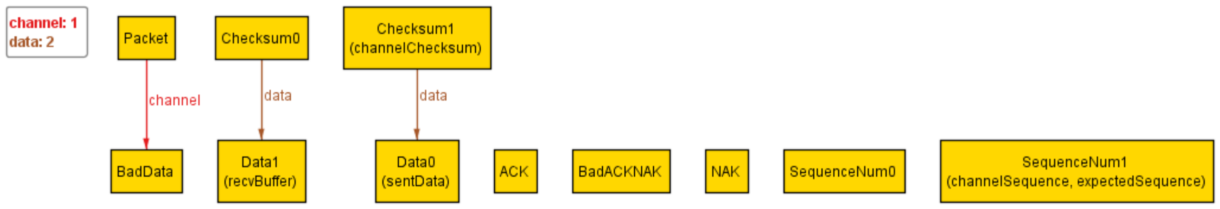
The sender has received the corrupted NAK as feedback and does not know whether the receiver received Data1. So it attempts to send Data1 again, with the same SequenceNum as last time. This time, the data is not corrupted during transmission.

State4


data: 2
Checksum0
Checksum1
Data1 (recvBuffer, sentData)
Data0 (sendBuffer)
ACK (feedback)
BadACKNAK
BadData
NAK
Packet
SequenceNum0 (channelSequence)
SequenceNum1 (expectedSequence)

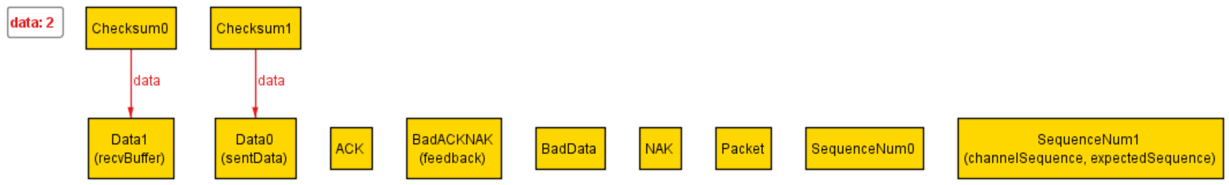
The receiver receives the uncorrupted Data1, so it sends back an ACK (this time the feedback is not corrupted). The SequenceNum it was expecting matched the SequenceNum of the data it received, so its puts Data1 into its recvBuffer. Then it changes its expected SequenceNum to 1.

State5



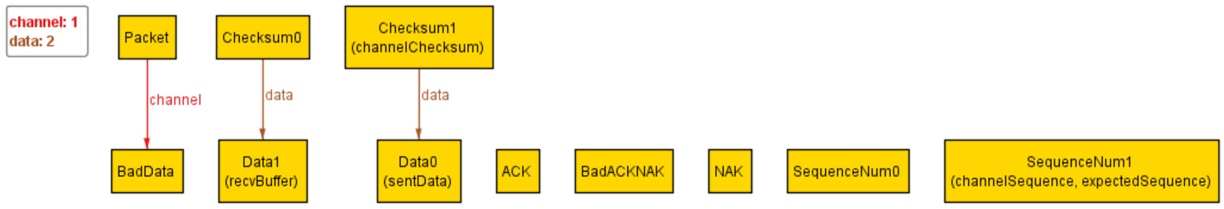
The sender has received the receiver's ACK and knows that it has received Data1. It now sends Data0 with a SequenceNum of 1. However, the data is again corrupted during transmission.

State6



The receiver receives the corrupted Data, and thus sends back a NAK. However, the NAK is corrupted during transmission.

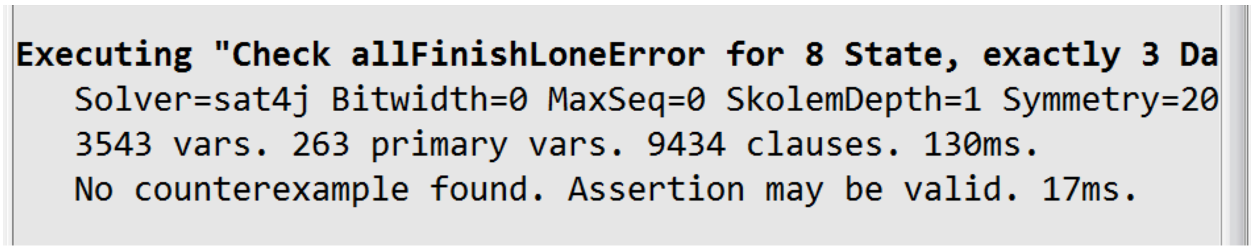
State7



The sender receives the corrupted NAK and does not know whether or not the receiver has received Data1 properly. So it attempts to send Data1 again with the same Sequence Number (1).

Corrupted feedbacks and Datas will cause the program to be stuck in an infinite loop.

Property 3



```
Executing "Check allFinishLoneError for 8 State, exactly 3 Da
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   3543 vars. 263 primary vars. 9434 clauses. 130ms.
   No counterexample found. Assertion may be valid. 17ms.
```

By putting in the assert that there could not be more than one sender/receiver error, we verified that the data will always be fully transmitted from the sender to the receiver. No counterexample was found.