

CSSE373

Milestone 2

Fred Zhang, Songyu Wang

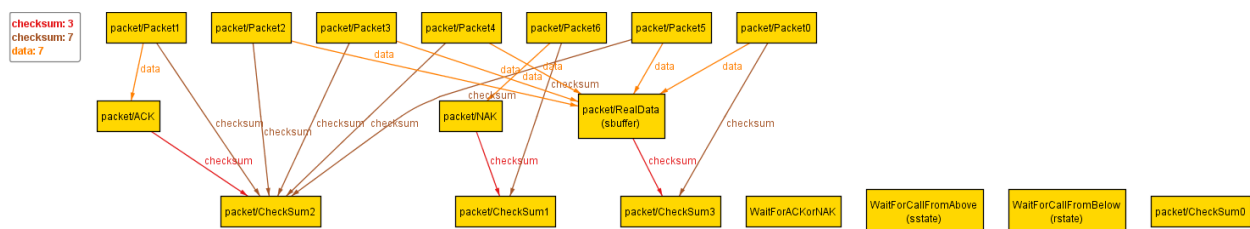
Using the given protocol, it is possible to transmit all of the data in the sender's buffer to the receiver's buffer.

We ran:

run possibleReliabe for 7 but exactly 1 RealData

The result is:

Time0

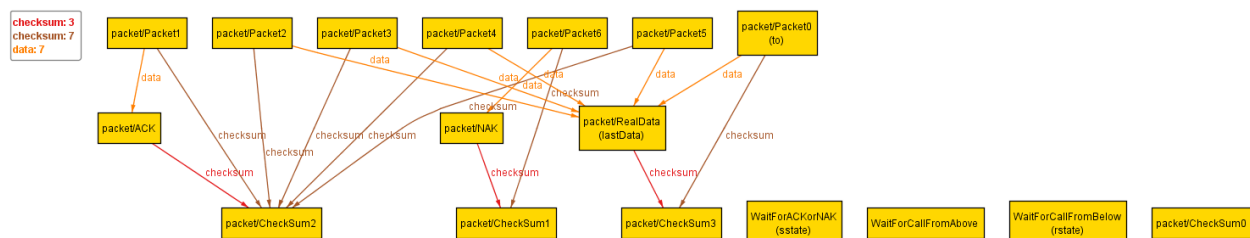


At Time 0, The RealData is in sender buffer.

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

checksum: 3
checksum: 7
data: 7

packetACK

packetChecksum2

packetChecksum1

packetChecksum3

WaitForACKorNAK (state)

WaitForCallFromAbove

WaitForCallFromBelow (state)

packetChecksum0

packetPacket1

packetPacket2

packetPacket3

packetPacket4

packetPacket5

packetPacket6

packetPacket5 (to)

packetPacket0

packet/RealData (lastData)

packet/NAK

data

checksum

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

[illegible]

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

The diagram illustrates a reliable data transfer protocol state. A legend box in the top left corner specifies:

- checksum: 3
- checksum: 7
- data: 7

 The diagram shows several components:

- Packets:** A row of yellow boxes labeled packet1Packet1 through packet6Packet6, followed by packet0Packet0 (to). Arrows labeled 'data' point from packet1Packet1 to packetACK, and from packet6Packet6 to packetRealData (lastData).
- Checksums:** A row of yellow boxes labeled packetChecksum2, packetChecksum1, packetChecksum3, WaitForACKorNAK (sstate), WaitForCallFromAbove, WaitForCallFromBelow (rstate), and packetChecksum0. Arrows labeled 'checksum' point from packetACK to packetChecksum2, from packet6Packet6 to packetChecksum1, and from packetRealData to packetChecksum3.
- NAK:** A yellow box labeled packetNAK. An arrow labeled 'checksum' points from packet6Packet6 to packetNAK, and an arrow labeled 'checksum' points from packetNAK to packetChecksum1.
- Wait States:** Three yellow boxes labeled WaitForACKorNAK (sstate), WaitForCallFromAbove, and WaitForCallFromBelow (rstate).

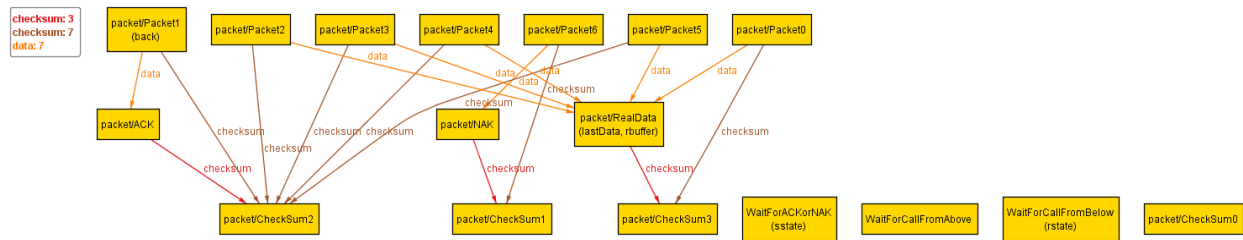
 The diagram uses color-coded arrows: orange for 'data', red for 'checksum', and brown for 'checksum' (from packets to checksums).

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

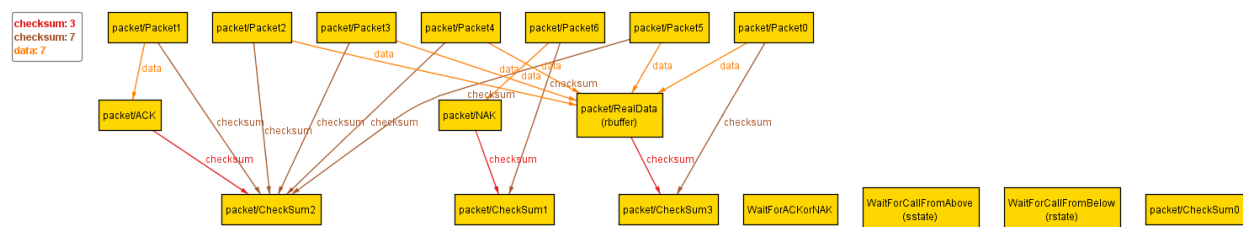


At Time 5, Receiver receives the uncorrupted data. The data is in the receiver buffer. Receiver sends an ACK message.

Sender state is Wait for ACK or NAK

Receiver state is Wait for Call From Below

Time6

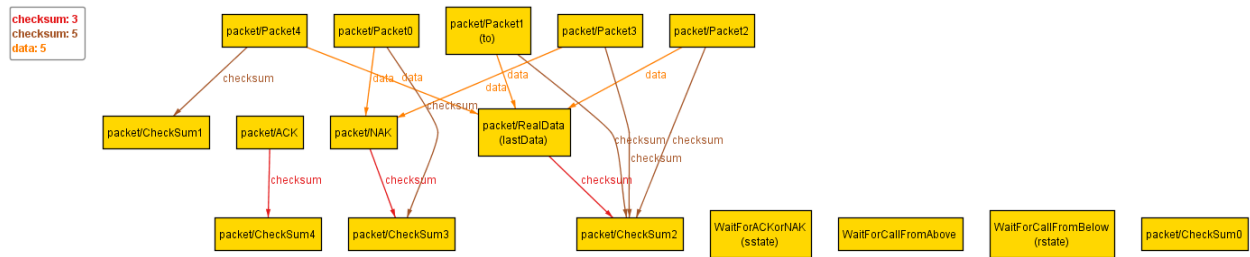


At Time 6, Sender receives the ACK message. Since there is no more data to send. This is the last state.

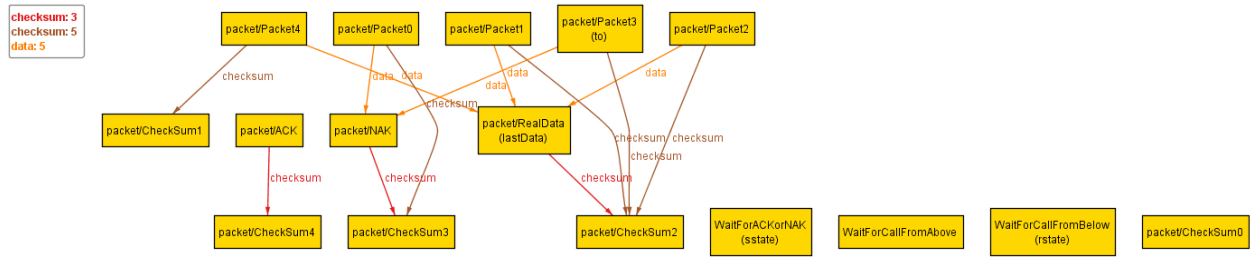
Sender state is Wait for Call From Above

Receiver state is Wait for Call From Below

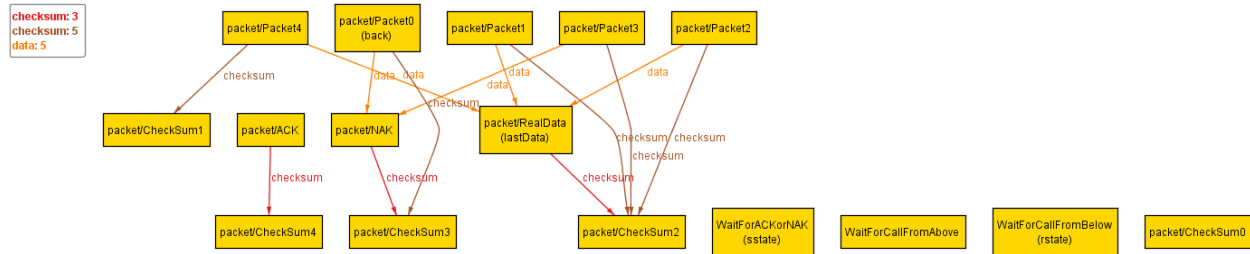
The NAK reaches the sender so the it resends the data again in Packet1.



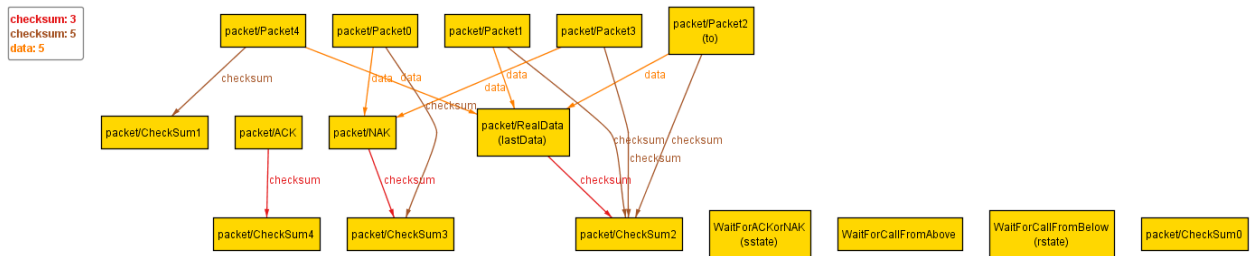
The packet gets corrupted again from Packet1 to Packet3



The corrupted packet arrived and the receiver 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 corrupted before it arrives, then the sender and receiver would just keep sending and responding NAK forever.

That is the same scenario described by Aloof.

This protocol cannot guarantee to send all the data from sender to receiver. If there is one packet that 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 through 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 protocol also has another flaw that, if a corrupted packet happens to pass the checksum test by flipping proper number of bits, the corrupted data can still get through somehow. In network, it is hard to achieve 100% corruption detection.