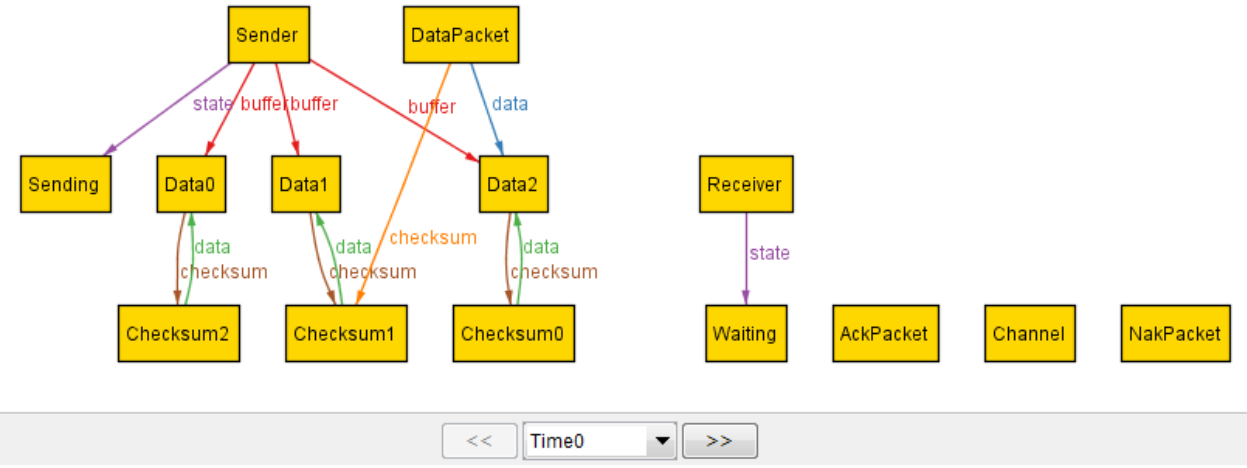
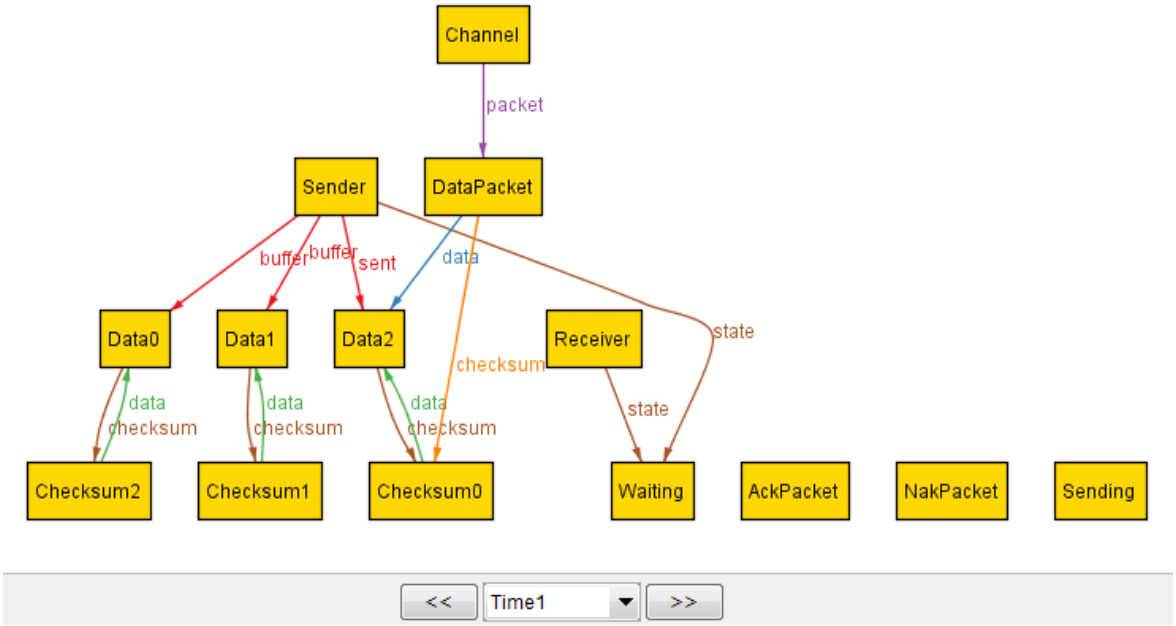
Team TripleT :: Tyler Duffy, Tyler Rockwood, & Trent Punt
Formal Methods Milestone 2 Submission

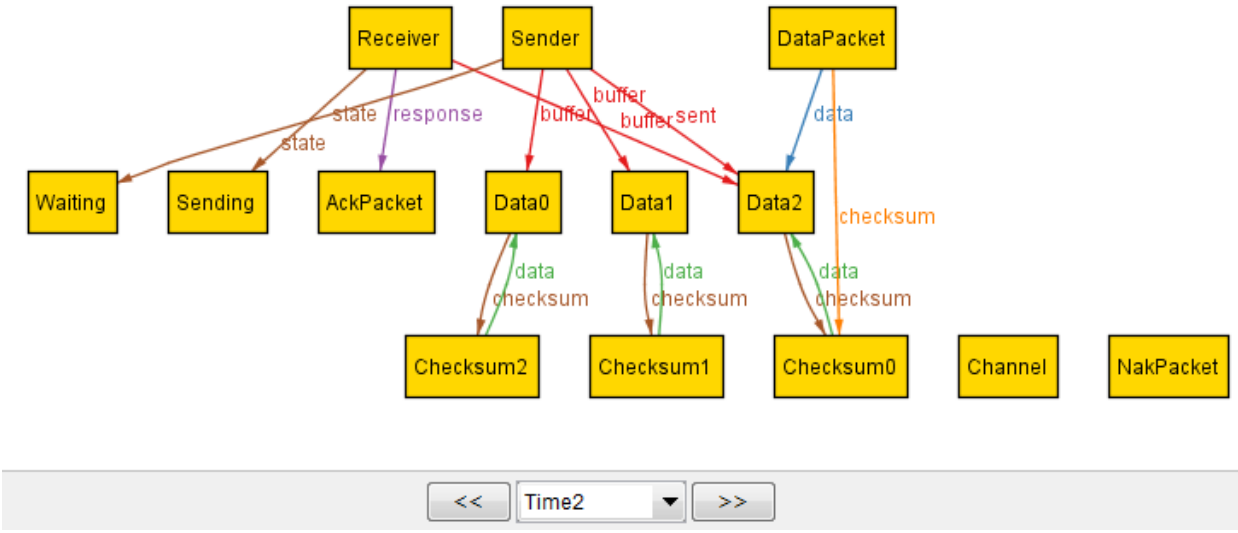
Transfer of Three Data Packets With No Malformed Packets



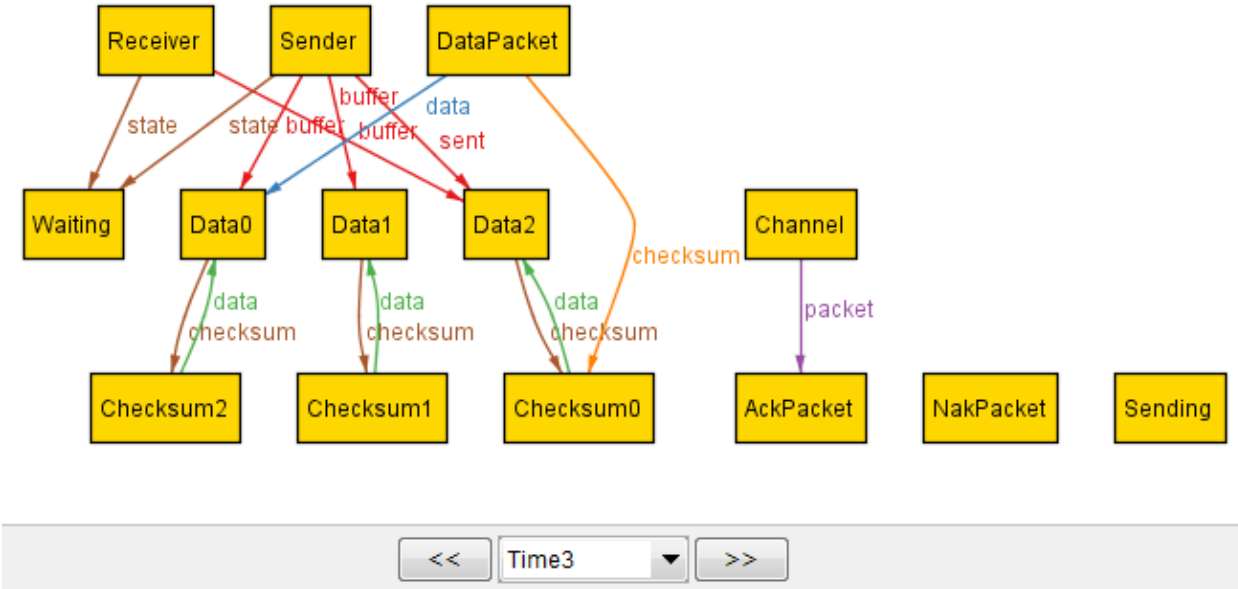
Time 0: Initial state. The Sender has all the data, the receiver has none of the data. Each of the data pieces has an assigned checksum value. Each checksum also has a data piece assigned to it for the one to one relation.



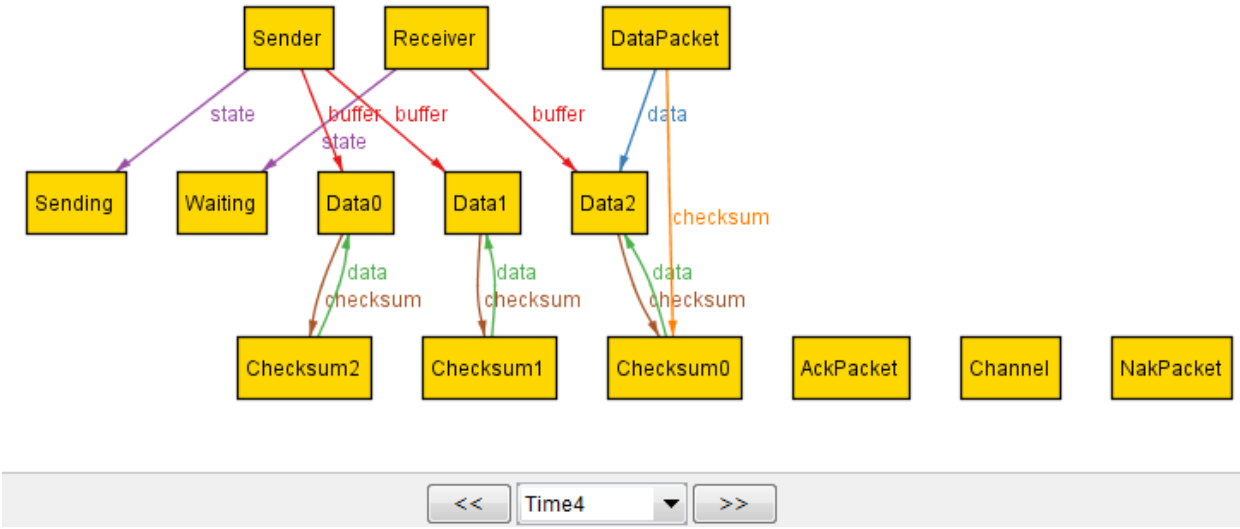
Time 1: The Sender has now built a DataPacket with the Data2 piece of data and its corresponding checksum. The Sender has prepared this data packet and dropped it into the channel to be picked up later by the Receiver. We also see that Sender has a new field called "sent" (red relation off of Sender above) that it has assigned to Data2. This serves as memory of what we are trying to send in the event that the data is malformed and not properly received. If this is the case, we can easily retransmit the same data. This data is now removed from the Sender buffer.



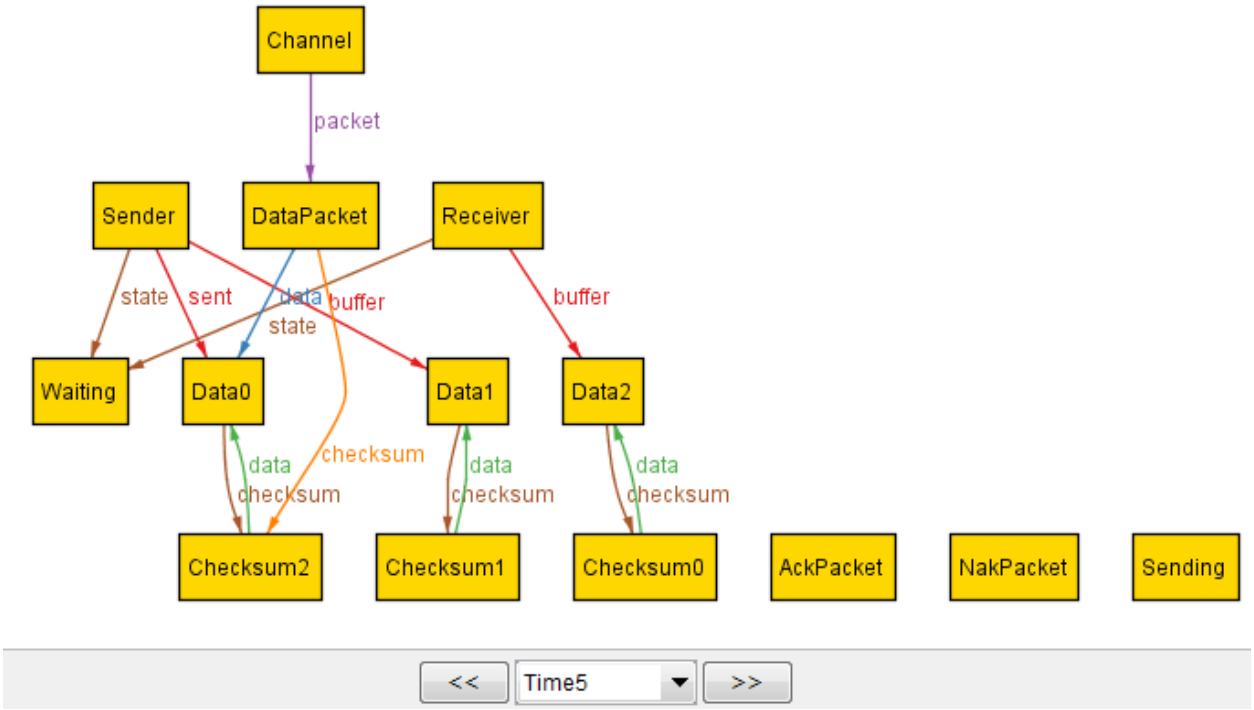
Time 2: The Receiver has now grabbed the data out of the Channel and checked its checksum for validity. Since the checksum passes, the Receiver prepares an AckPacket to send back for confirmation (we can see this in the purple "response" field of Receiver). Also, this new data appears in the Receiver buffer since it is the right data.



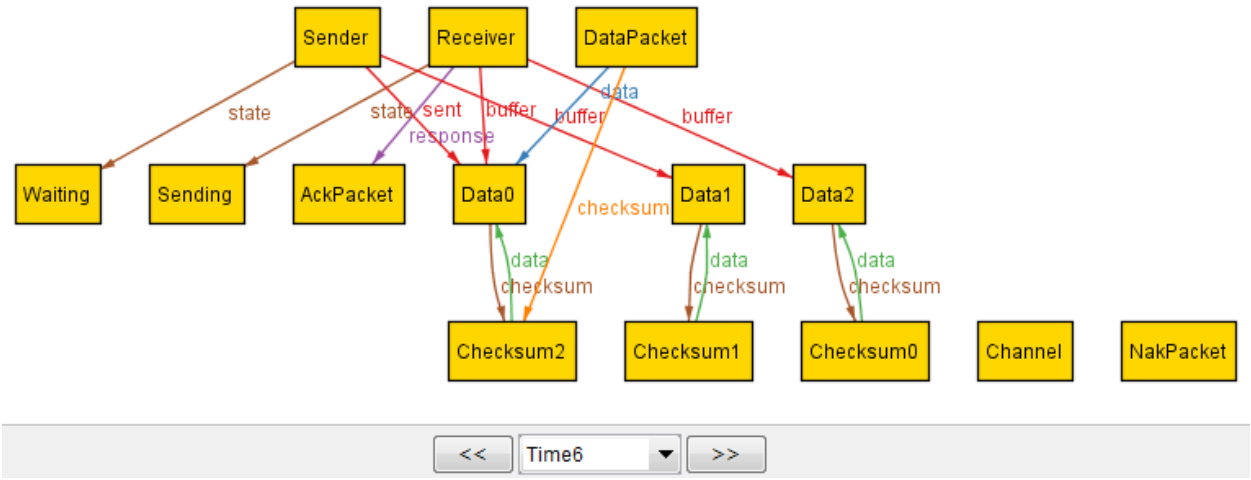
Time 3: The AckPacket is dropped into the Channel to be picked up later by the Sender.



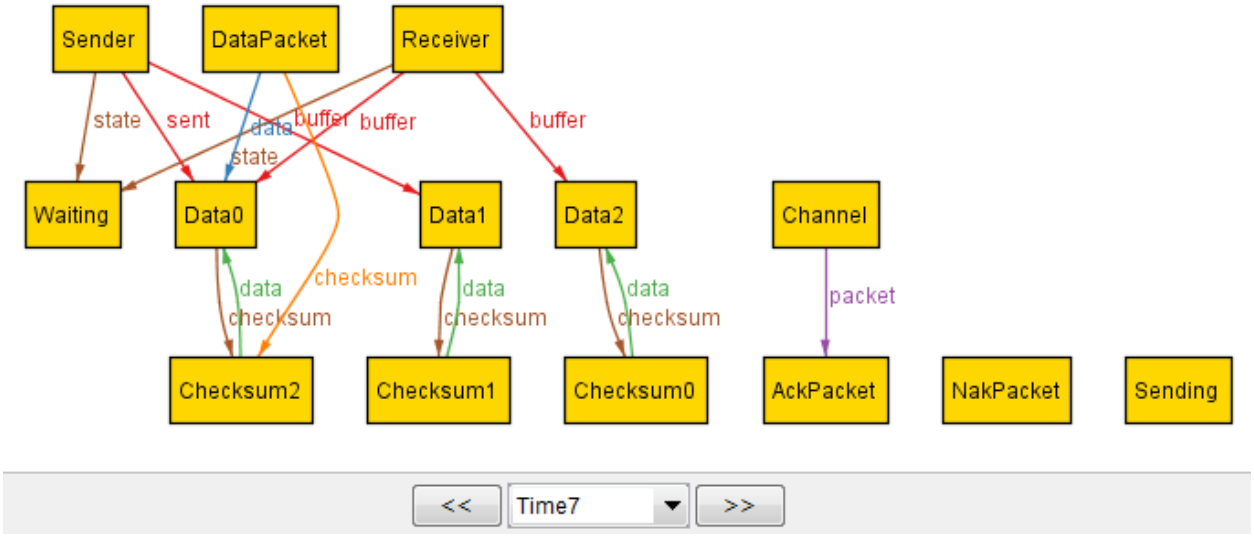
Time 4: The AckPacket is now grabbed from the Channel by the Sender. Since we have confirmed the successful transmission of data, we clear the memory of the "sent" field that was storing the last data that we were trying to transmit. The Sender is now back in the Sending state again and ready to prepare more data packets!



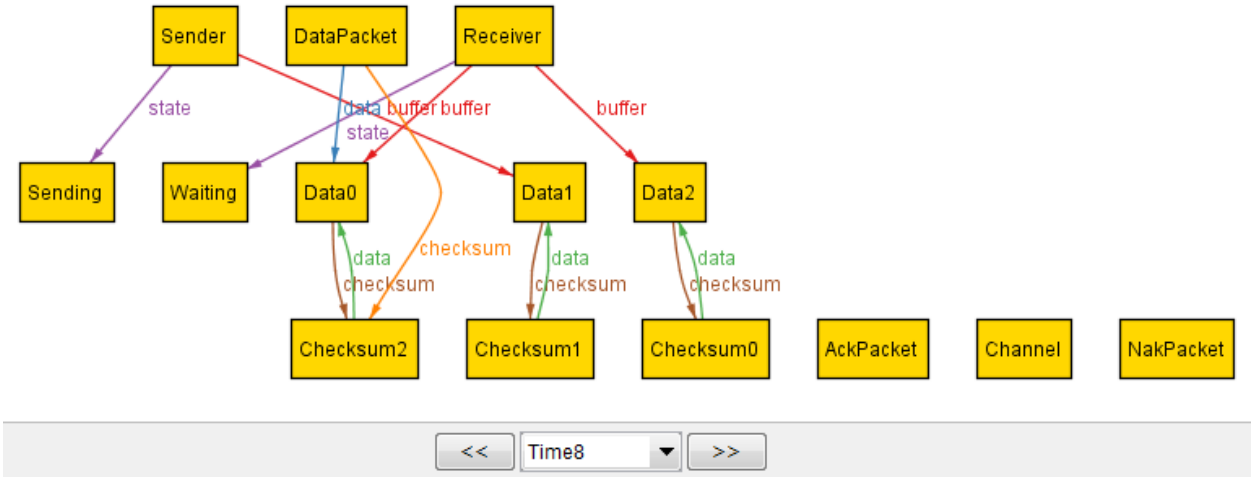
Time 5: The Sender has now built a DataPacket with the Data0 piece of data and its corresponding checksum. The Sender has prepared this data packet and dropped it into the channel to be picked up later by the Receiver. We also see that Sender has a new field called "sent" that it has assigned to Data0. This serves as memory of what we are trying to send in the event that the data is malformed and not properly received. If this is the case, we can easily retransmit the same data. This data is now removed from the Sender buffer.



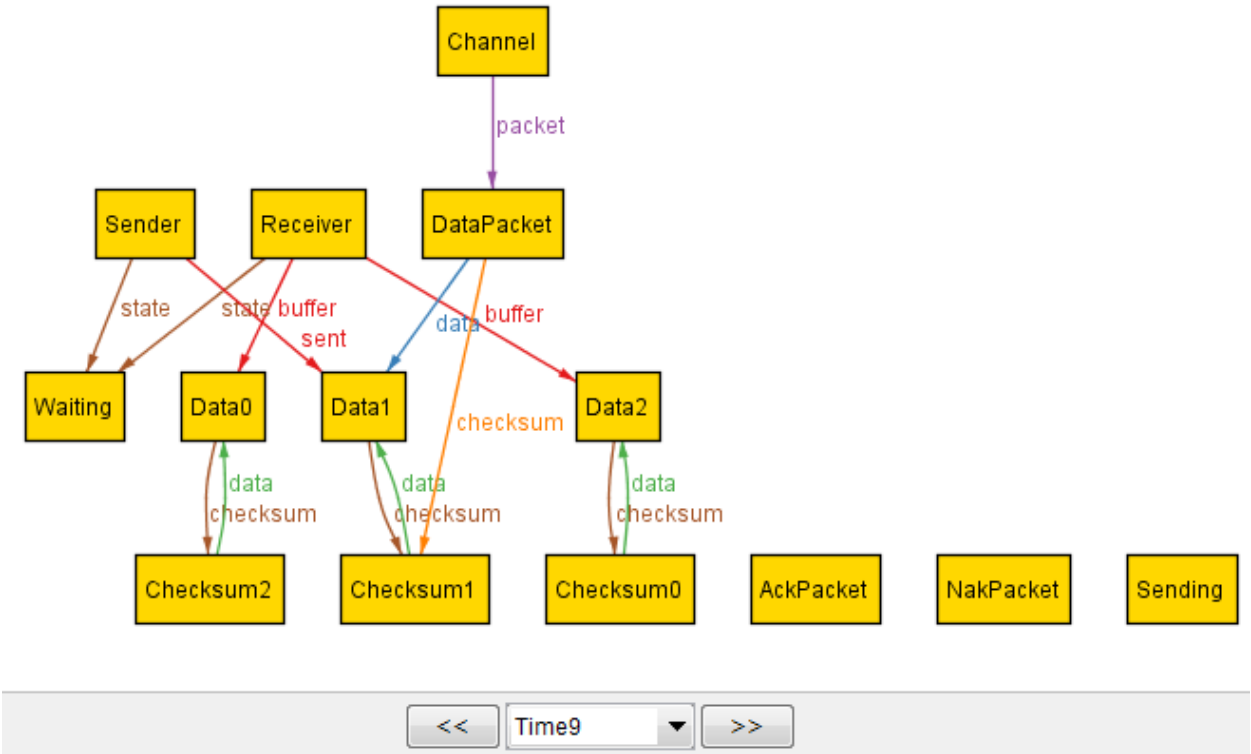
Time 6: The Receiver has now grabbed the data out of the Channel and checked its checksum for validity. Since the checksum passes, the Receiver prepares an AckPacket to send back for confirmation (we can see this in the purple "response" field of Receiver). Also, this new data appears in the Receiver buffer since it is the right data.



Time 7: The AckPacket is dropped into the Channel to be picked up later by the Sender.

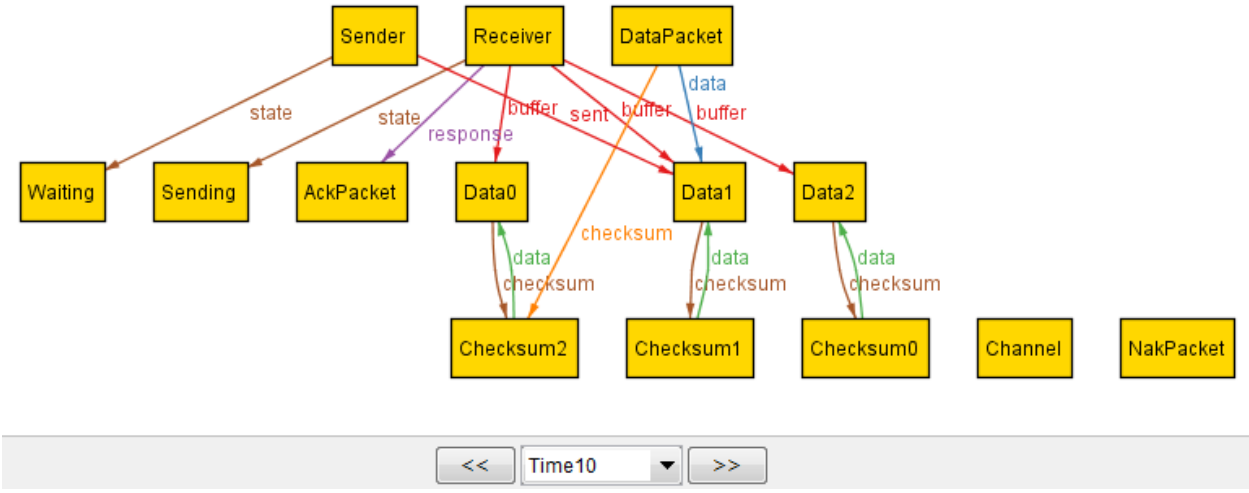


Time 8: The AckPacket is now grabbed from the Channel by the Sender. Since we have confirmed the successful transmission of data, we clear the memory of the "sent" field that was storing the last data that we were trying to transmit. The Sender is now back in the Sending state again and ready to prepare more data packets!

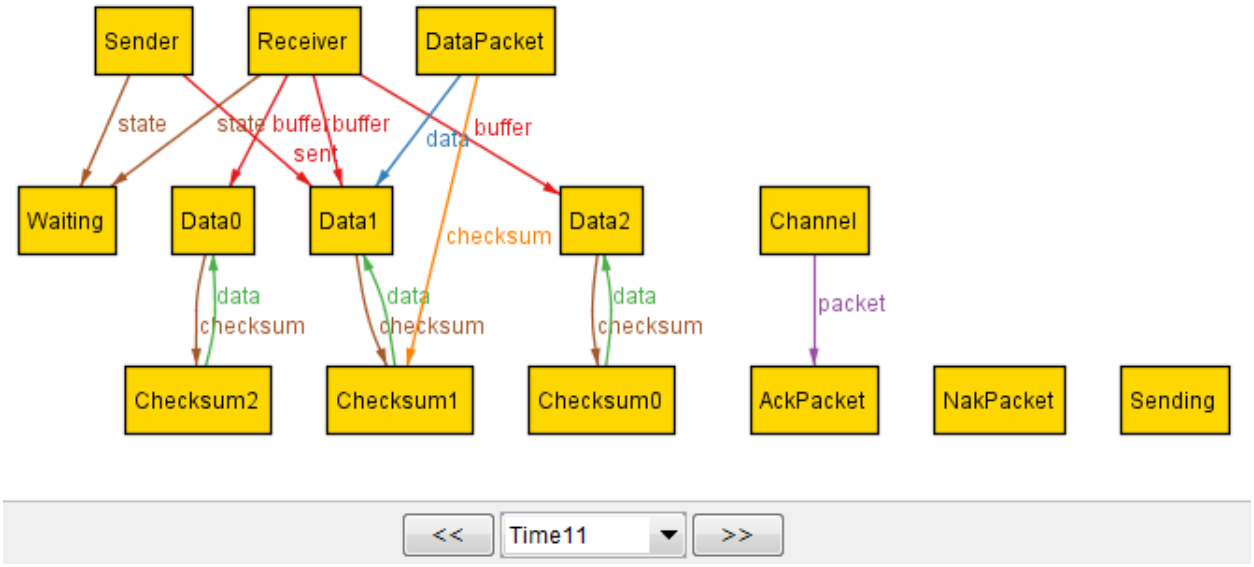


Time 9: The Sender has now built a DataPacket with the Data1 piece of data and its corresponding checksum. The Sender has prepared this data packet and dropped it into the channel to be picked up later by the Receiver. We also see that Sender has a new field called "sent" that it has assigned to Data1. This serves as memory of what we are trying to send in the event that the data is malformed and not properly received. If this is the case, we can easily retransmit the same data. This data is now removed from the Sender buffer.

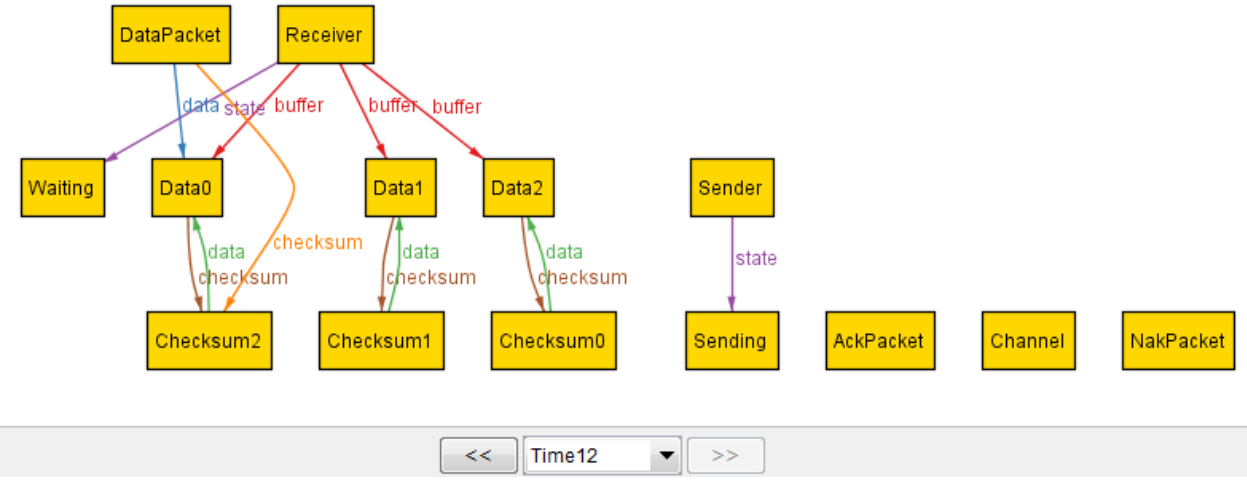
Team TripleT :: Tyler Duffy, Tyler Rockwood, & Trent Punt
Formal Methods Milestone 2 Submission



Time 10: The Receiver has now grabbed the data out of the Channel and checked its checksum for validity. Since the checksum passes, the Receiver prepares an AckPacket to send back for confirmation (we can see this in the purple "response" field of Receiver). Also, this new data appears in the Receiver buffer since it is the right data.

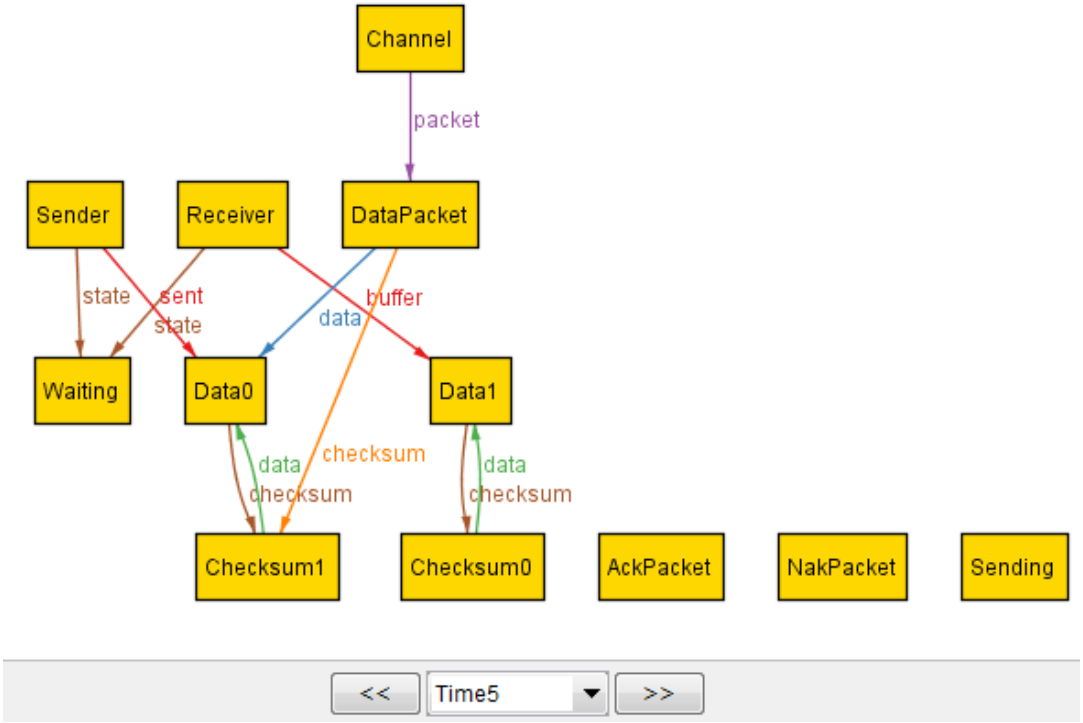


Time 11: The AckPacket is dropped into the Channel to be picked up later by the Sender.

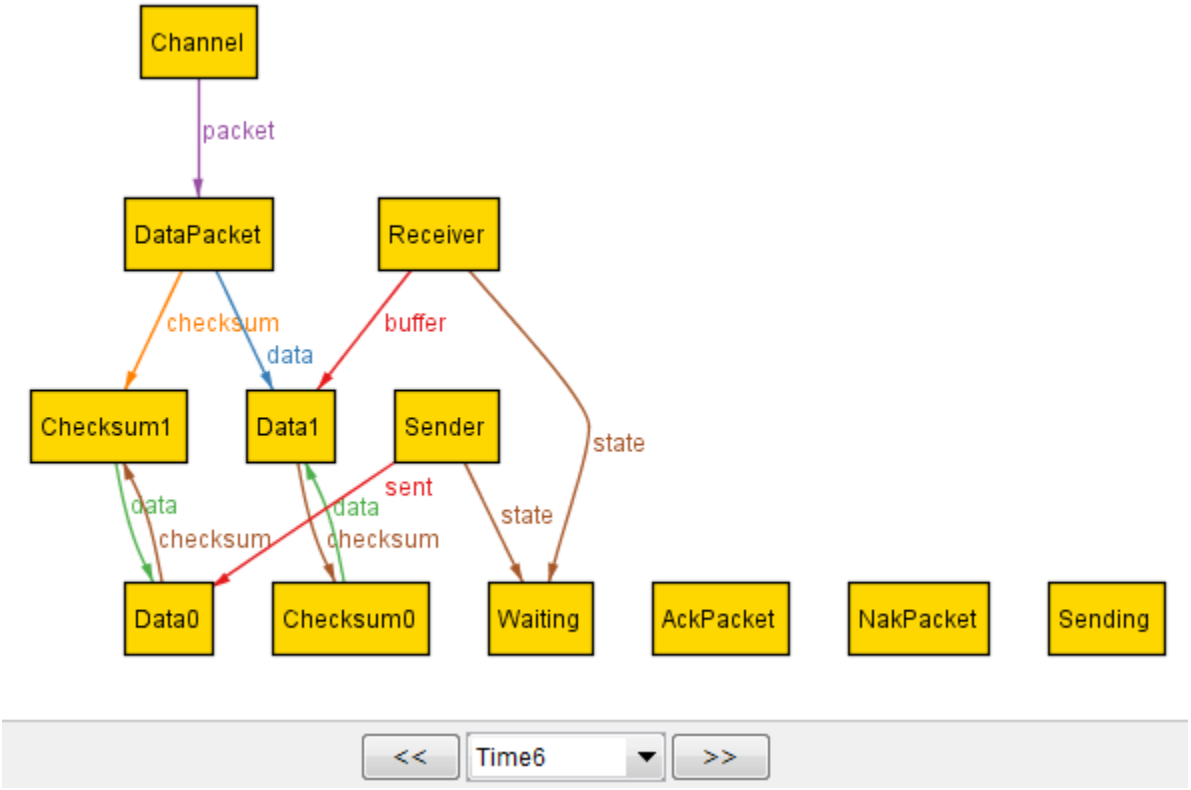


Time 12: End state! The Sender has no data left to send from its buffer. The Receiver has all the data properly transmitted into its buffer. Neither computer is trying to send any more packets back and forth (in the "sent" or "response" fields). The Channel is also empty. All done!

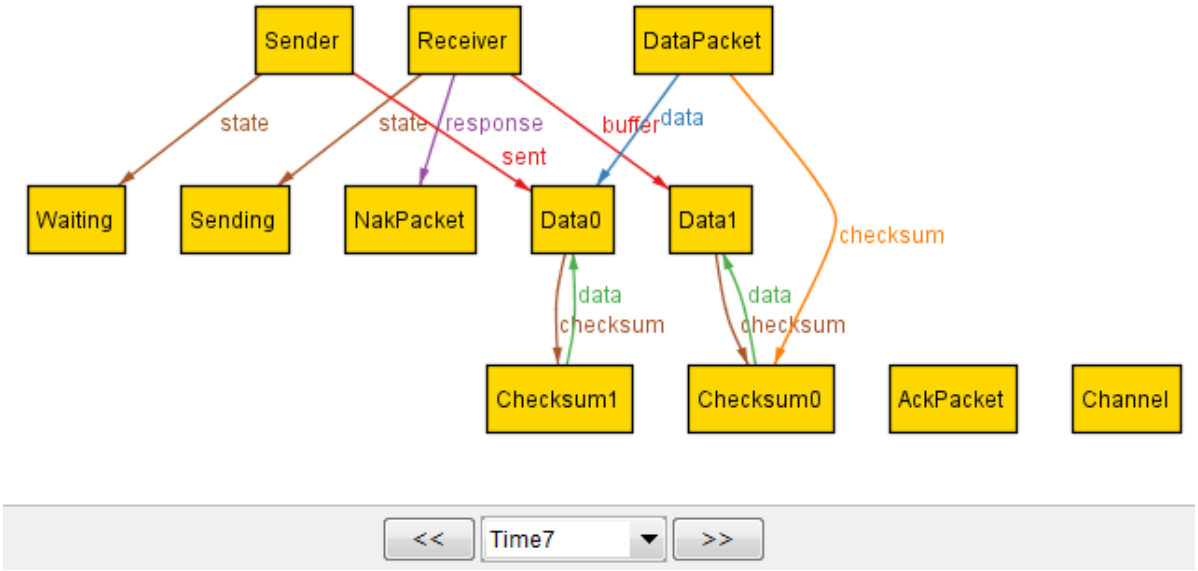
Sample Transfer of Data Packets With Malformed Packets



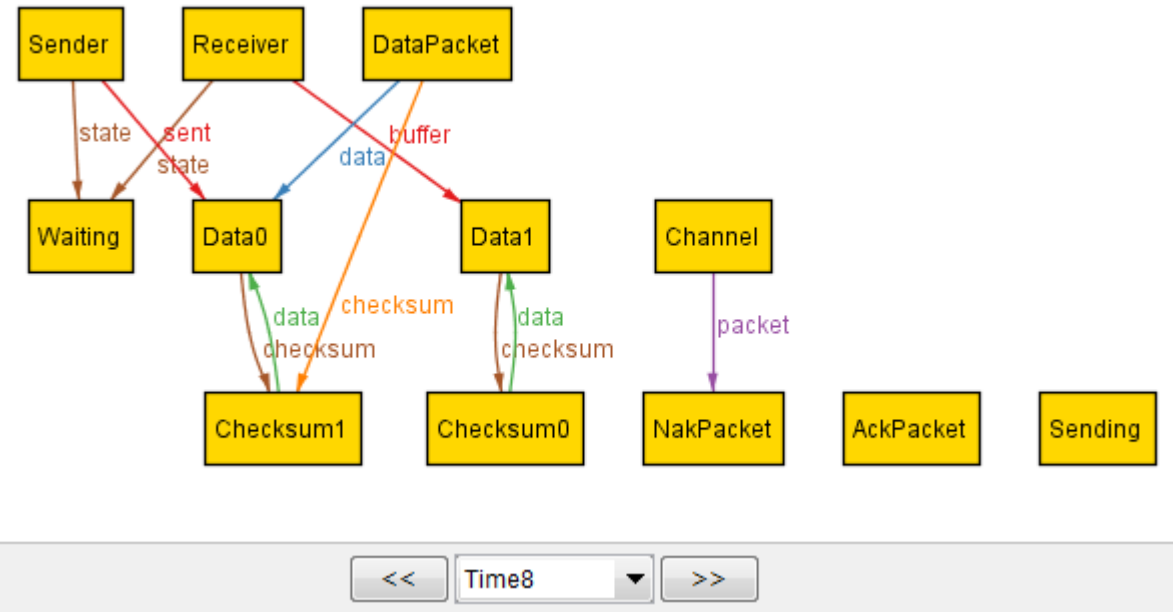
Time 5: Here the Sender has prepared a DataPacket with Data0 and the appropriate Checksum for this Data and dropped it into the Channel for the Receiver to pickup.



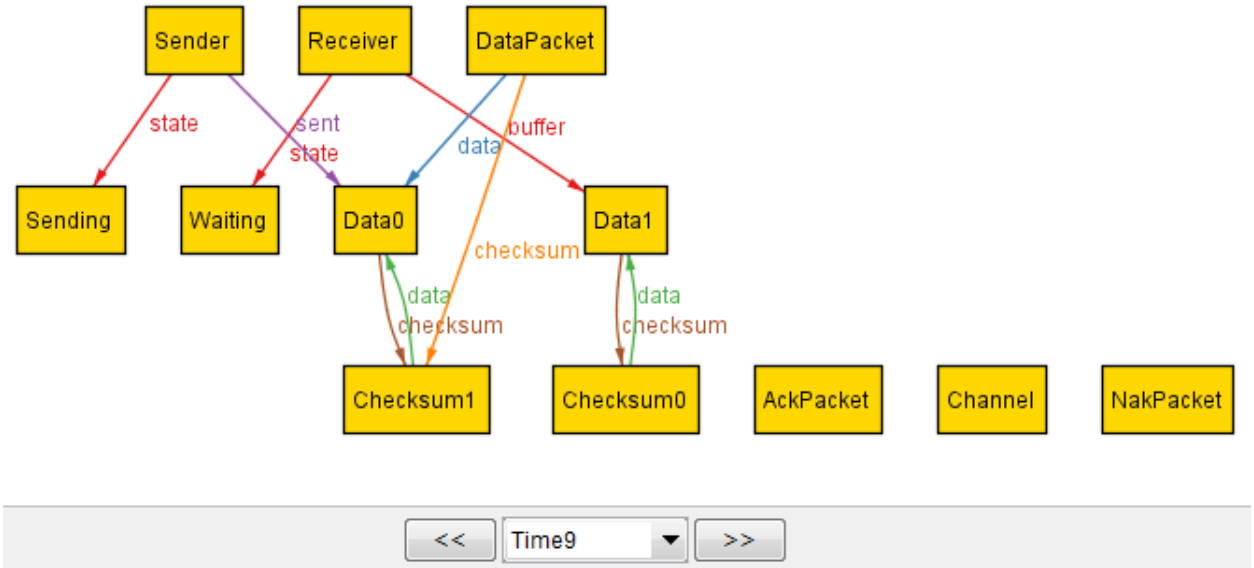
Time 6: We can see that the data has now been malformed inside of the Channel en route to the Receiver. The DataPacket is now pointing to Data1 instead of Data0.



Time 7: The Receiver has now pulled the data from the Channel. Since the received data's checksum does not match the packets, we see that the Receiver is preparing a NakPacket response to send back to the Sender (see the purple "response" relation off of Receiver above).



Time 8: The Receiver has now placed its NakPacket response into the Channel.



Time 9: The Sender has now received the NakPacket from the Receiver, so it must get ready to retransmit its last attempt. This attempt is stored under the purple “sent” relation off of Sender above. The Sender will now retransmit the Data0 packet to the Receiver (hopefully correctly!).

Can have both successful transfers and unsuccessful transfers depending on eventual transmission of any malformed packets…