# Project Sprint 2

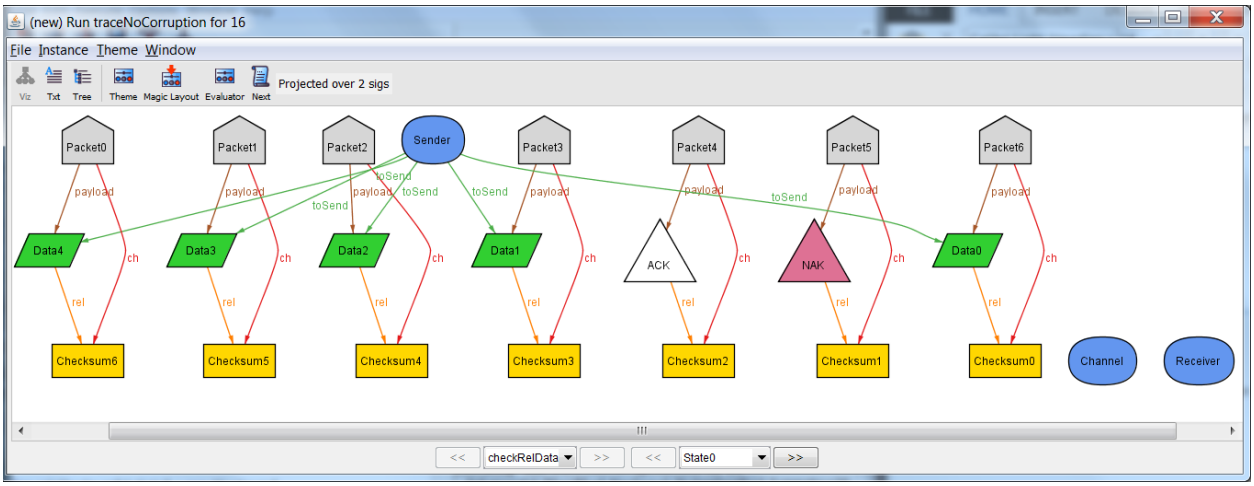
## Property 1A – Successful Transfer without Corruption

It is possible for the data to be successfully transferred. In this case, no corruption of data occurred during transfer.

For this trace, five instances of data (green parallelograms) will be transferred. These pieces of data are initially held in Sender (blue circle). Packets (grey pentagons) each contain a piece of data as well as a checksum (yellow boxes). The Receiver (blue circle) will contain data after it is successfully transferred. The Channel (blue circle) will contain packets while they are being transferred. Finally, the Ack (white triangle) and Nak (red triangle) are sent after each receive operation to indicate if the transfer was successful or corrupted. For this trace, no corruption will occur, so the Nak will never be sent.

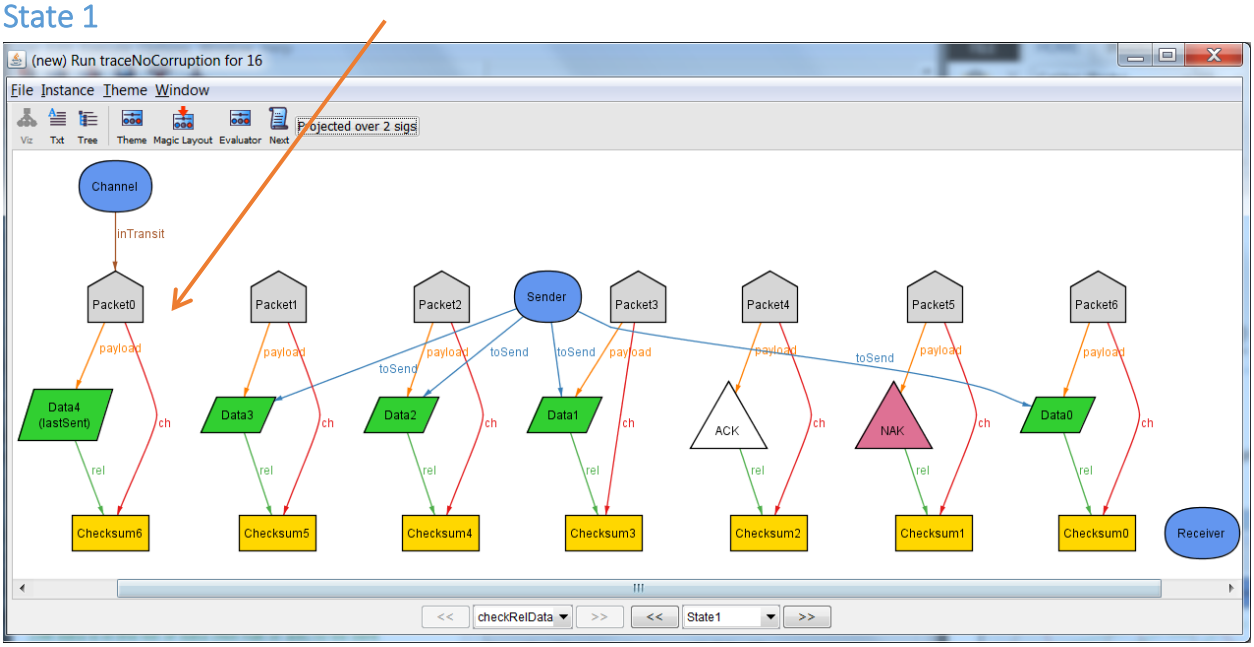
These states follow the pattern of Send Data, Receive Data, Send Response. In Send Data, a data from Sender is transferred through the Channel. In Received Data, the transferred packet's checksum is compared to the packet's contents. If they match, then the data is added to the Receiver. If they do not, the data is ignored. In Send Response, if the previous receive was successful, then an Ack is sent through the Channel. Otherwise, a Nak is sent through the Channel. The process then repeats, with the Data selected by Sent Data being chosen based on whether an Ack or a Nak was sent in the previous step.

### State 0



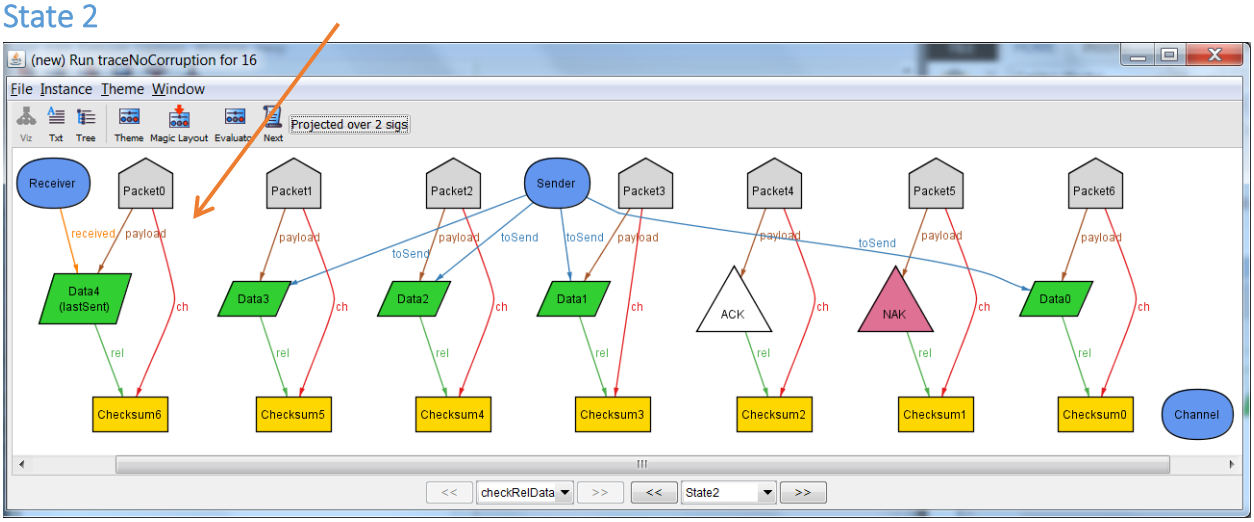

In the initial state, all five instances of data are contained in the Sender. Nothing is being transferred in the channel. Nothing has been received in the receiver.

## State 1



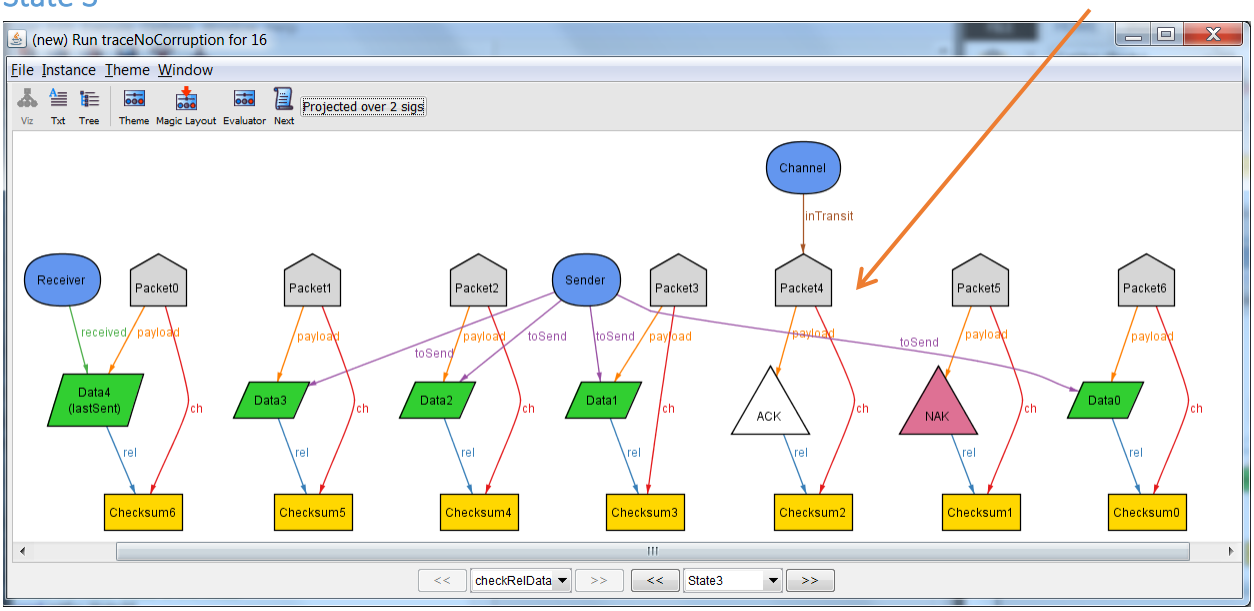

In this state, the first send occurs. Data 4 is no longer contained in the Sender. Packet 0, which contains Data 4, is now being transferred through the Channel.

## State 2



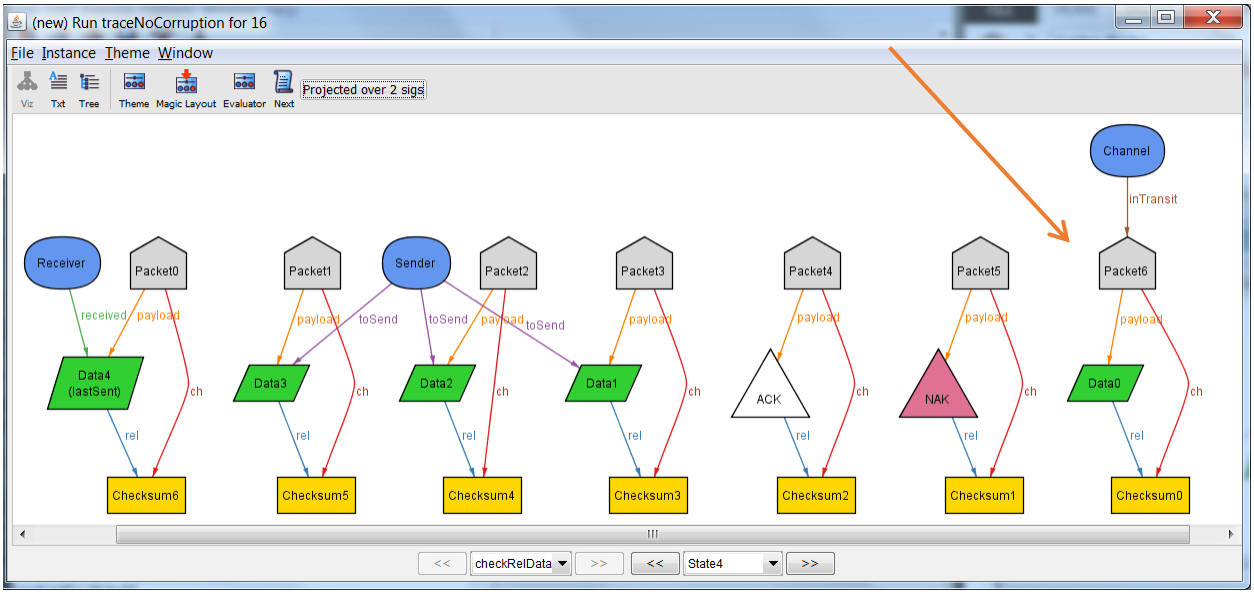

In this state, the first receive occurs. Data 4 is no longer being transferred through the Channel. The Checksum of the transferred Packet matches the contents of the Packet, so no corruption occurred. Data 4 is now contained in the Receiver.

## State 3



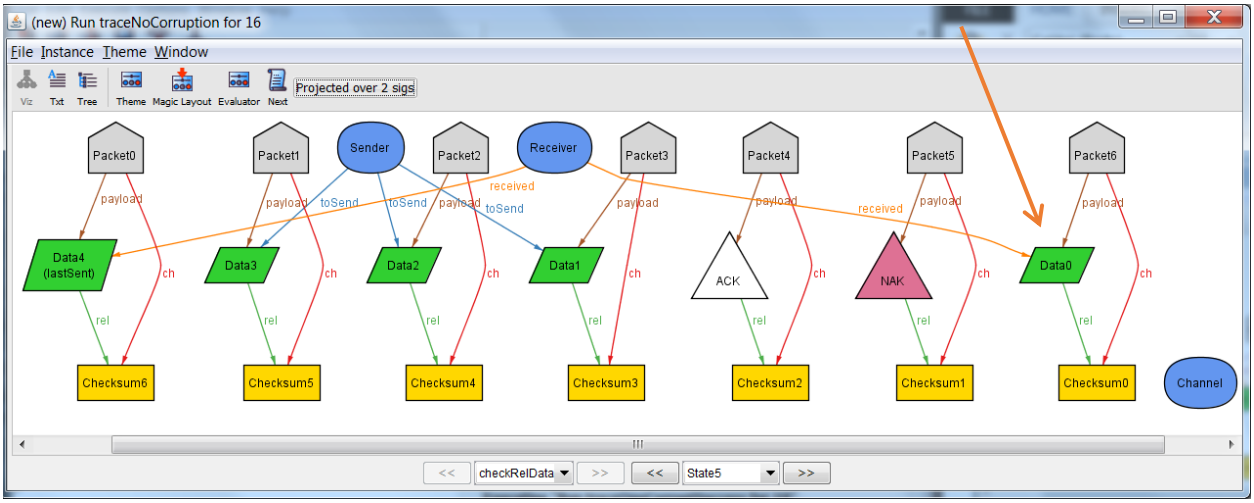

Since the data received in the previous state was not corrupted, the Ack is now sent through the Channel. A full cycle of steps has now come to an end.

## State 4



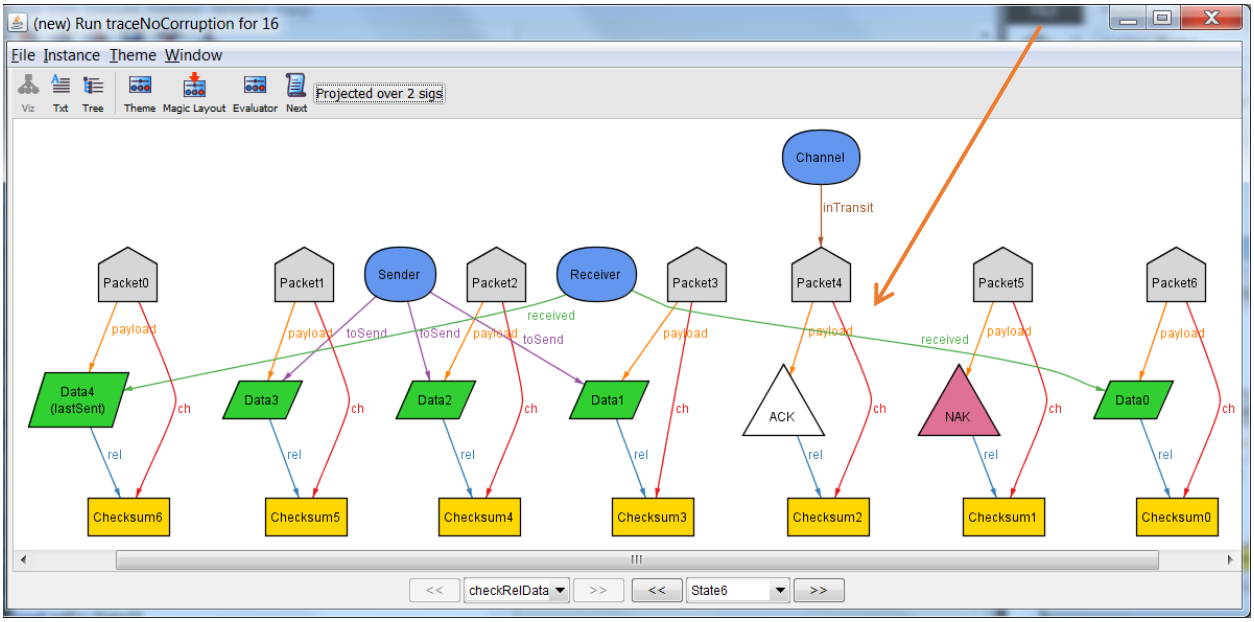

A new cycle starts with a send. Since the previous state transferred an Ack, a new piece of data is sent.

## State 5



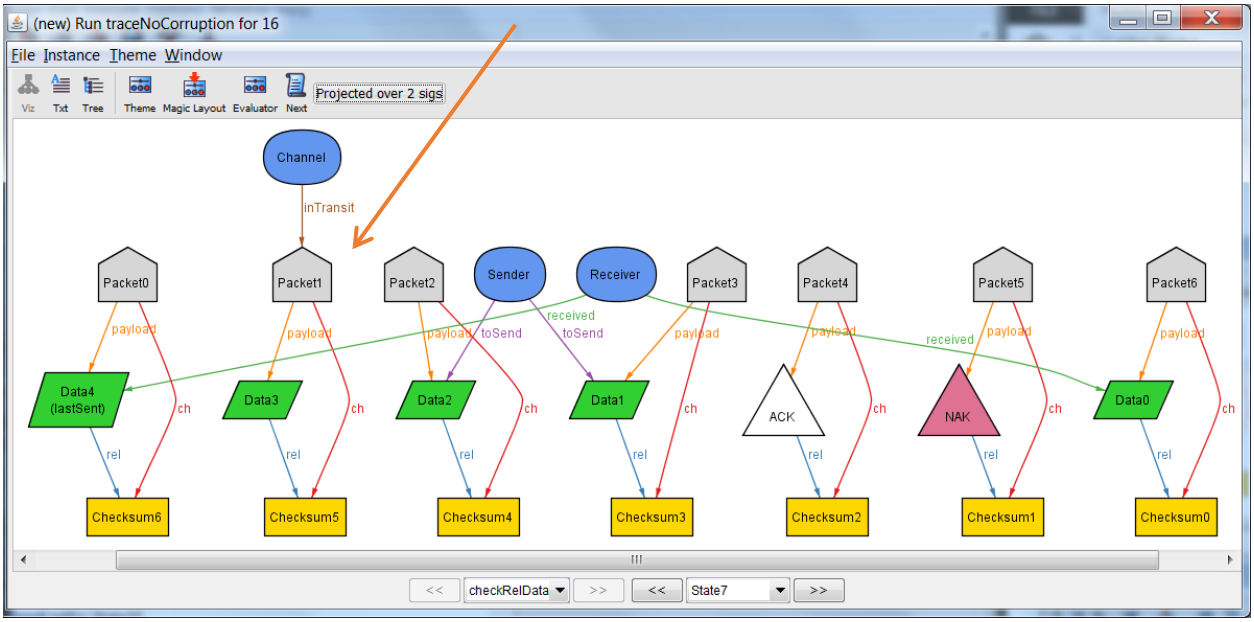

Receive: The checksum and contents matched, so the data is added to the receiver.

## State 6



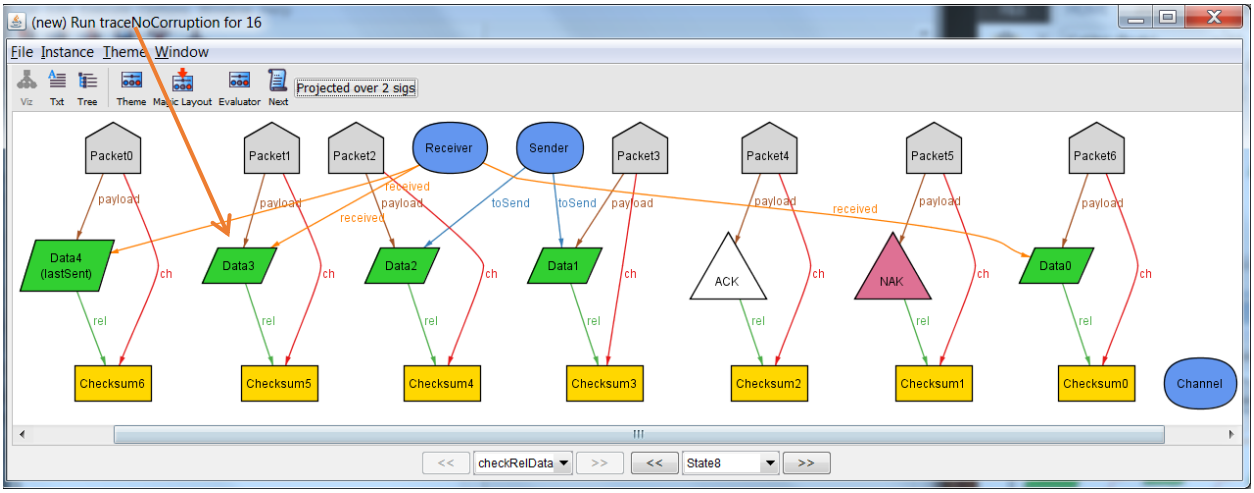

Response: The last transfer was successful, so Ack is sent back.

## State 7



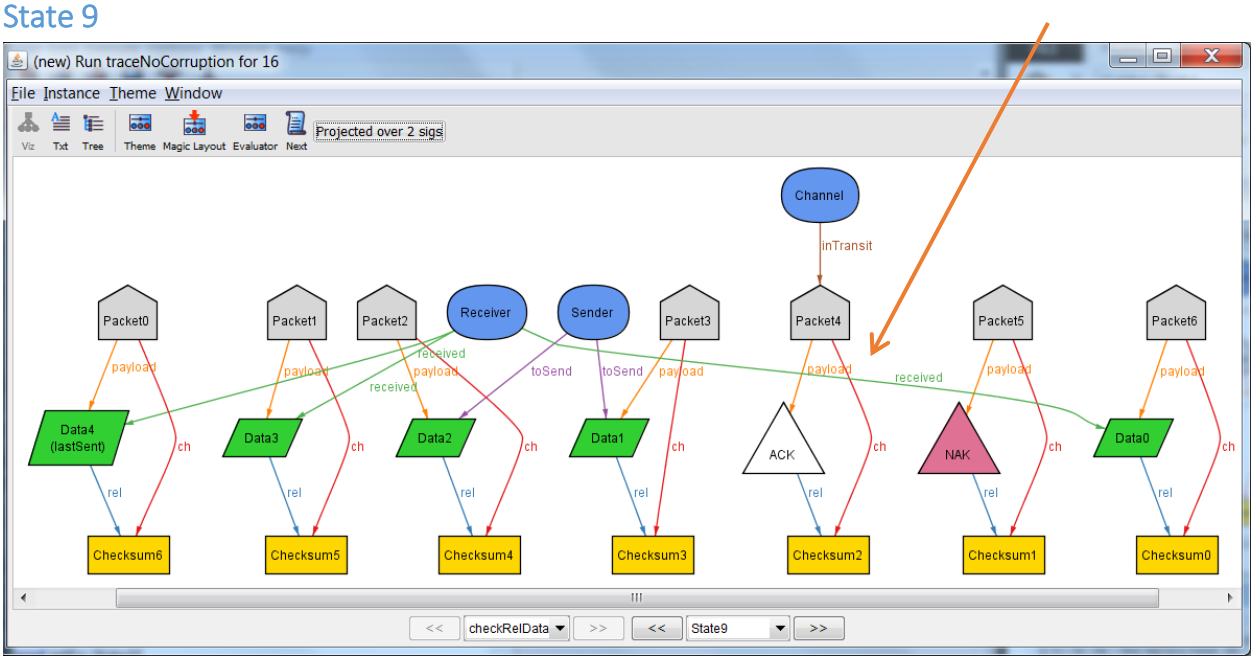

Send: Ack was received, so send a new data.

## State 8



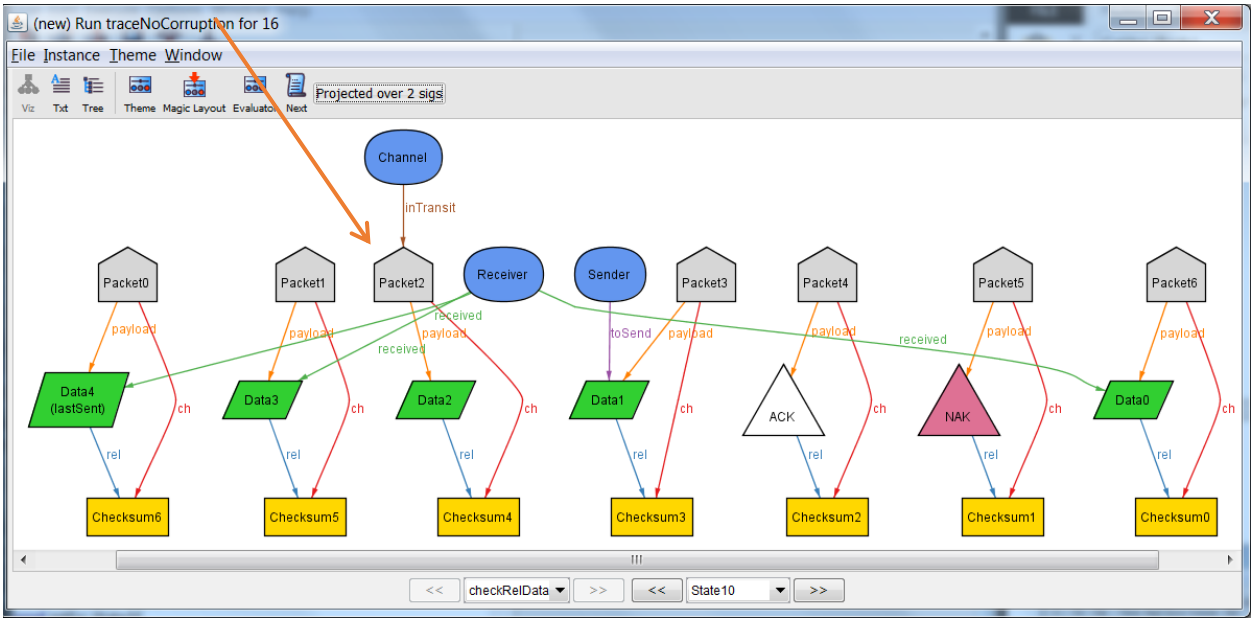

Receive: The checksum matched the contents, so the receive was successful.

## State 9



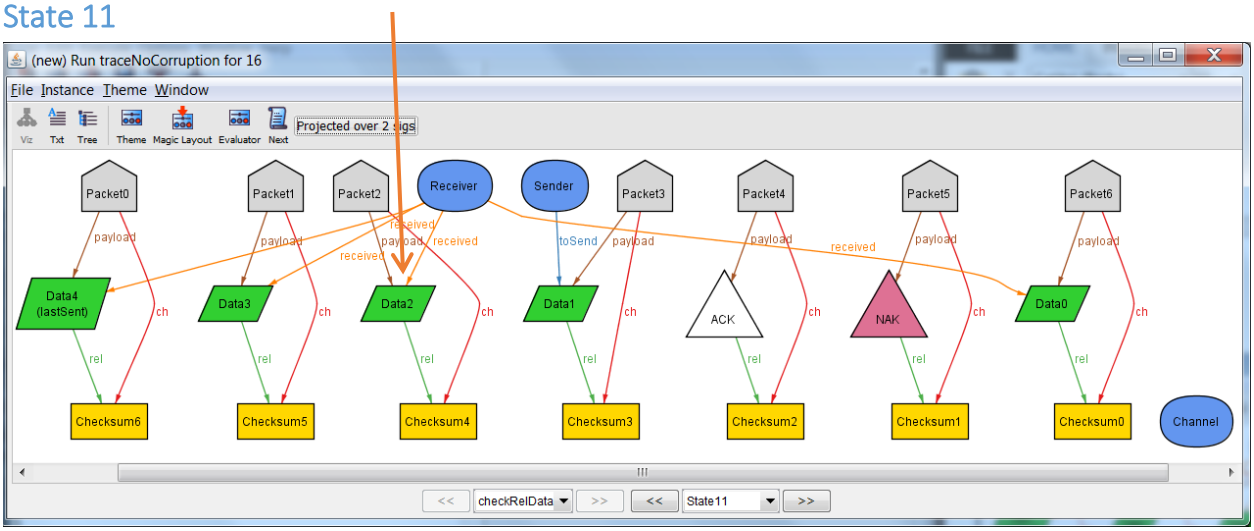

Send Response: The last receive was successful, so send Ack.

## State 10



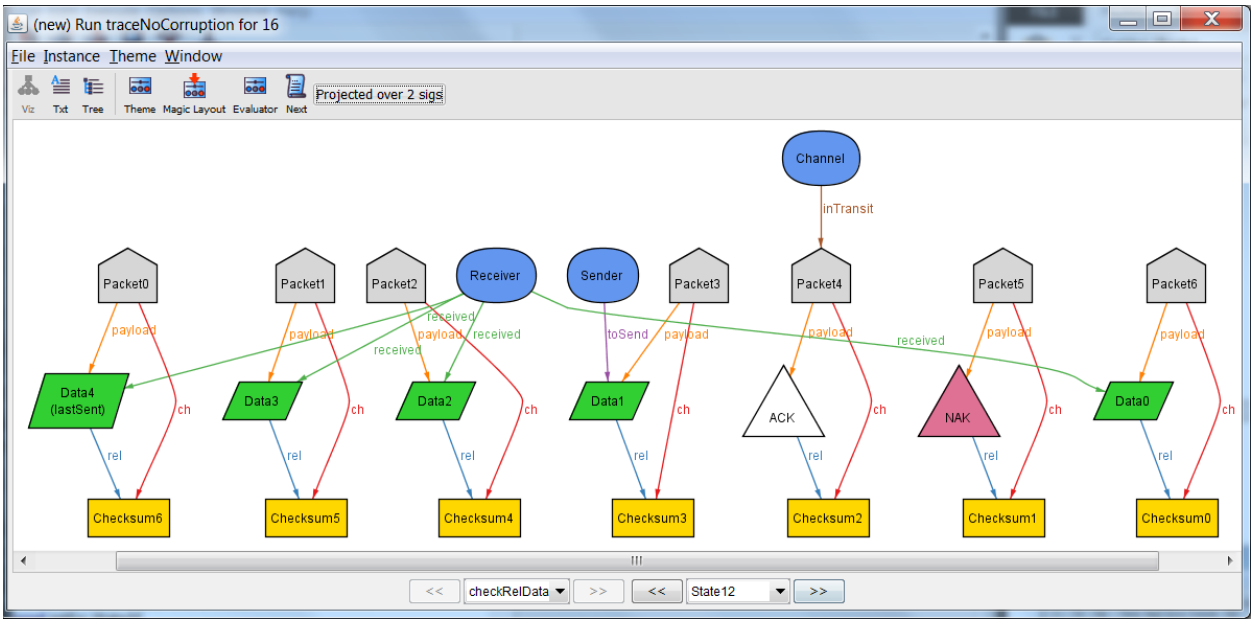

Send: Ack was received, so send new data.

## State 11



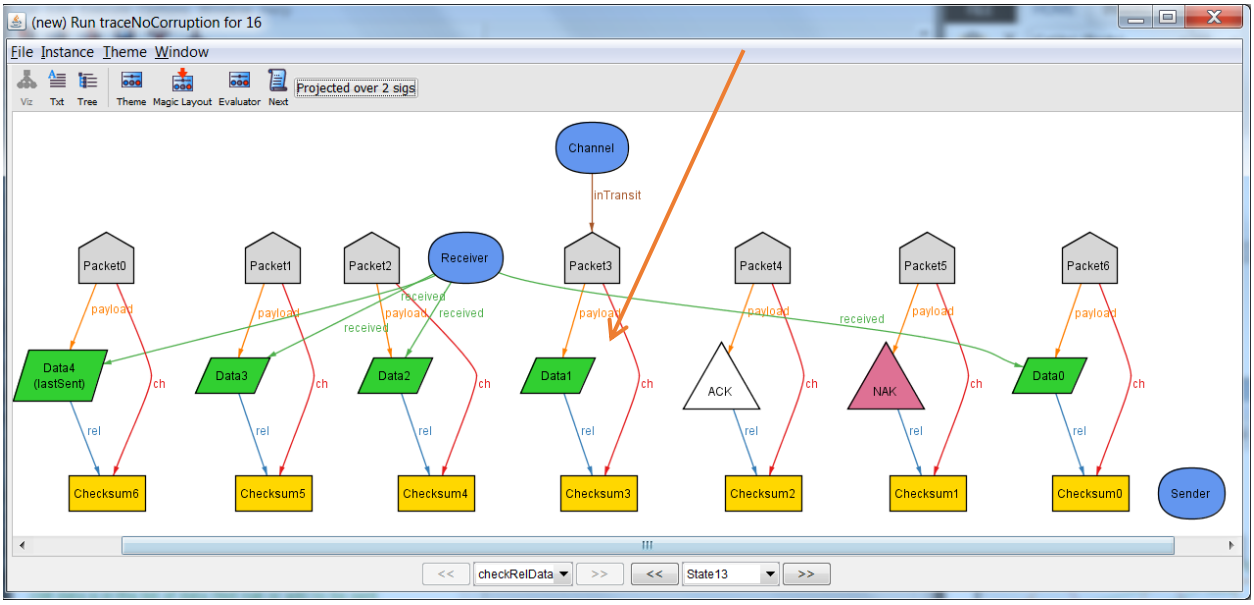

Receive: The checksum matches the contents, so the receive is successful.

## State 12



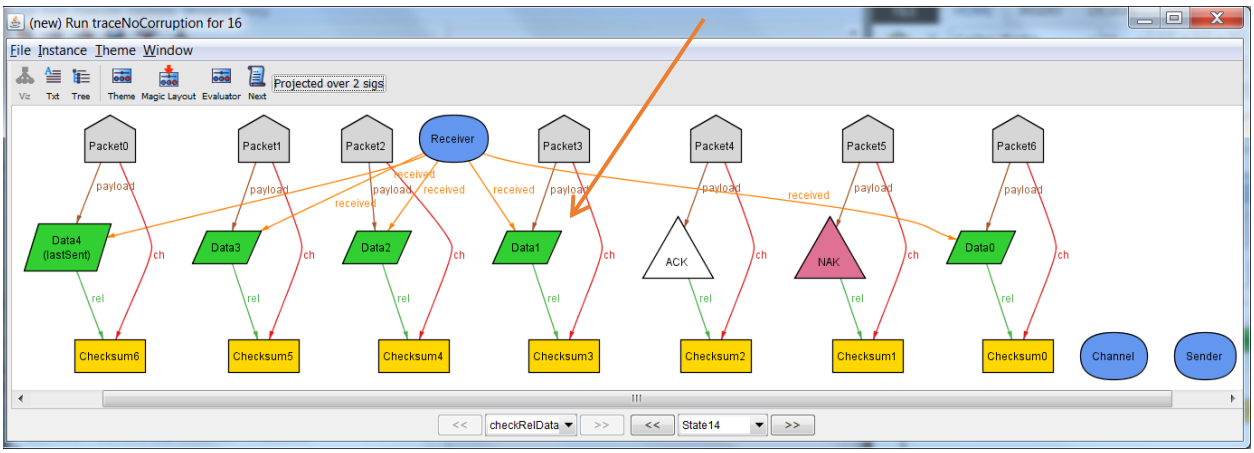

Send Response: The last receive was successful, so send Ack.

## State 13



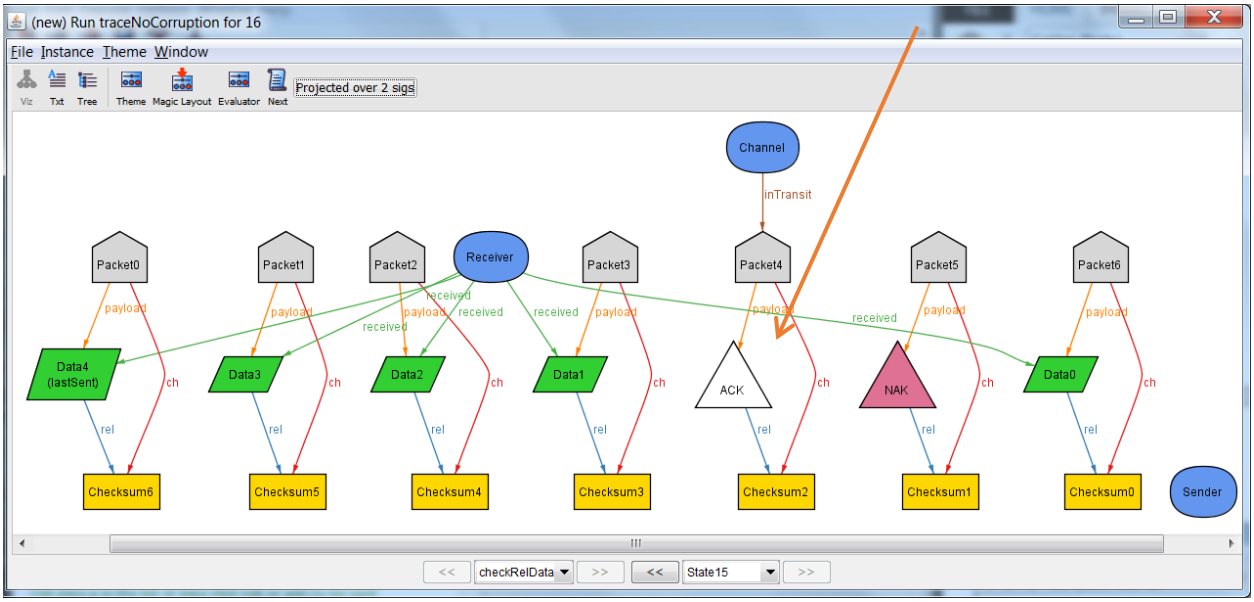

Send: Ack was received, so send new data.

## State 14




Receive: The checksum and contents match, so the receive was successful.

## State 15



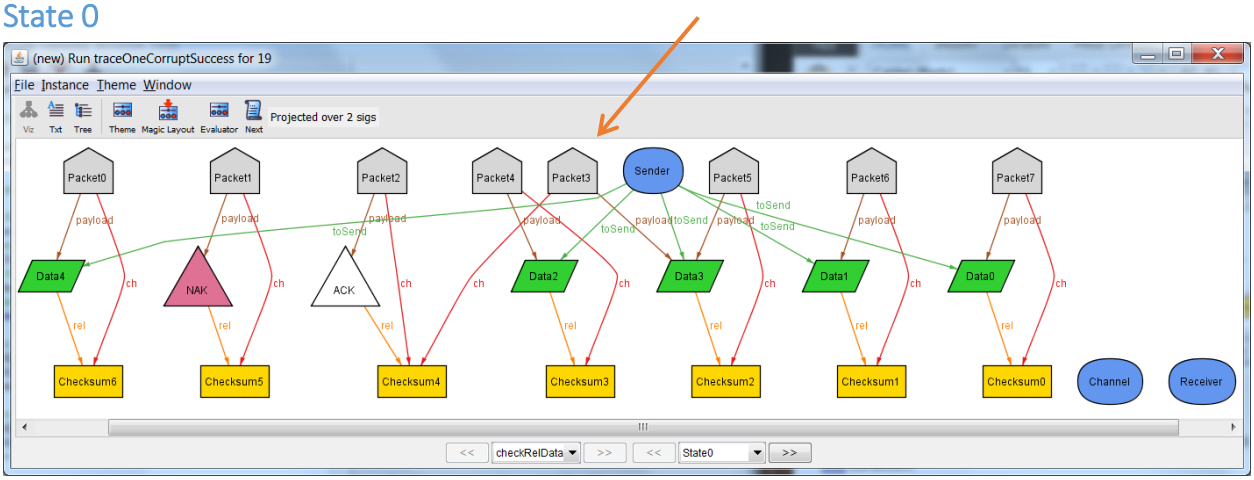

Send Response: The previous receive was successful, so send Ack.

All of the data was successfully received by the Receiver. Thus property 1 holds when there is no corruption.

## Property 1B – Successful Transfer with Corruption

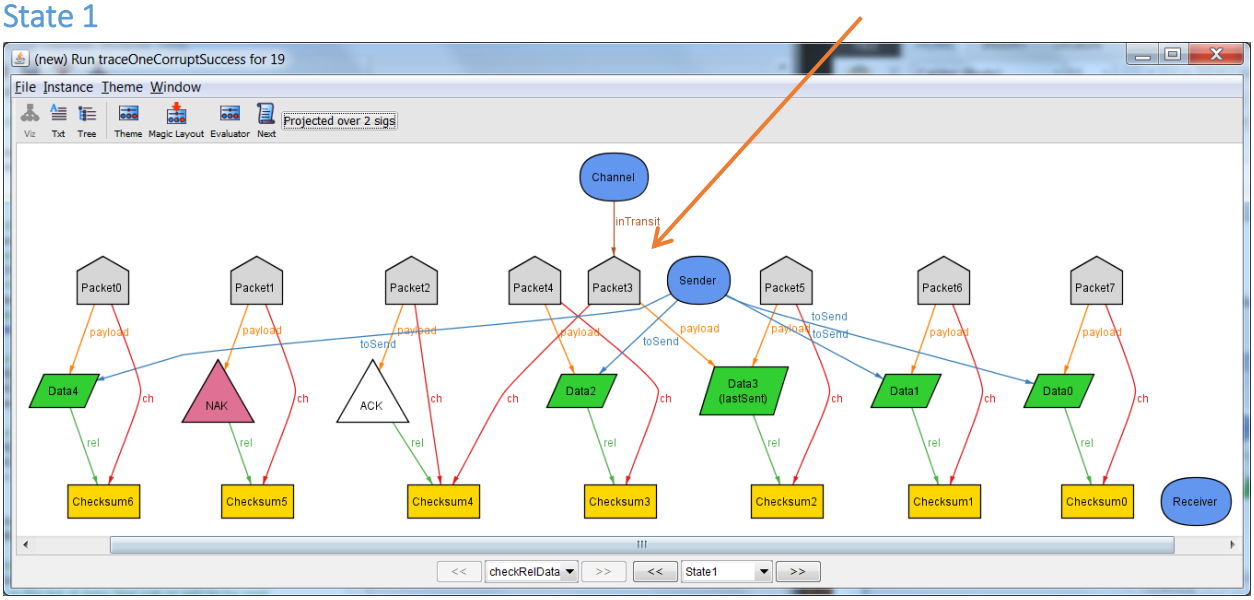
It is also possible to successfully transfer all of the data even if corruption is present. This trace follows the same pattern as the previous example: Send, Receive, and Respond. If the Response is an Ack, the next data is sent. If the Response is a Nak, then the data that was last sent is re-sent.

### State 0



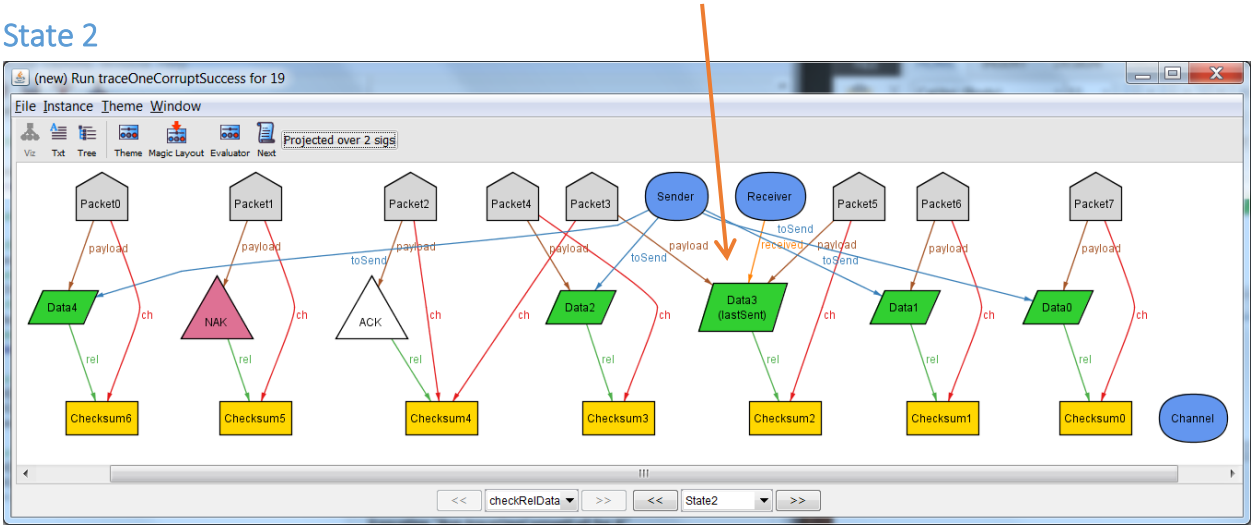

To start with, all of the data is contained in Packets and referenced by Sender. Packet 3 contains an error, as its checksum refers to Ack, while its data contains Data 3.

### State 1



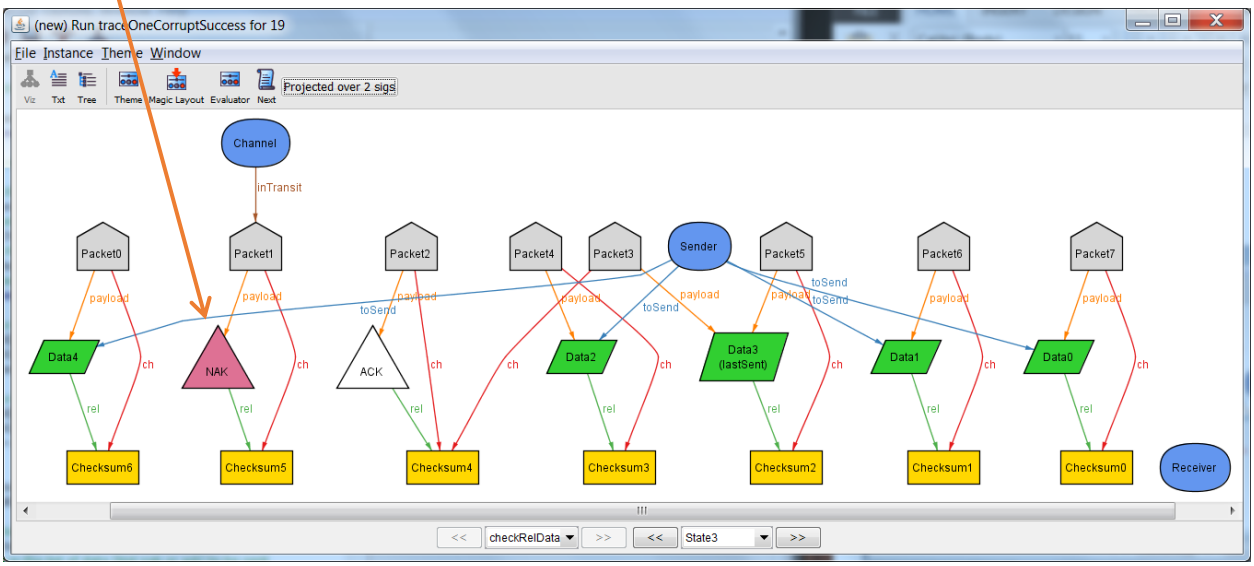

Send: It attempts to send Packet 3.

## State 2



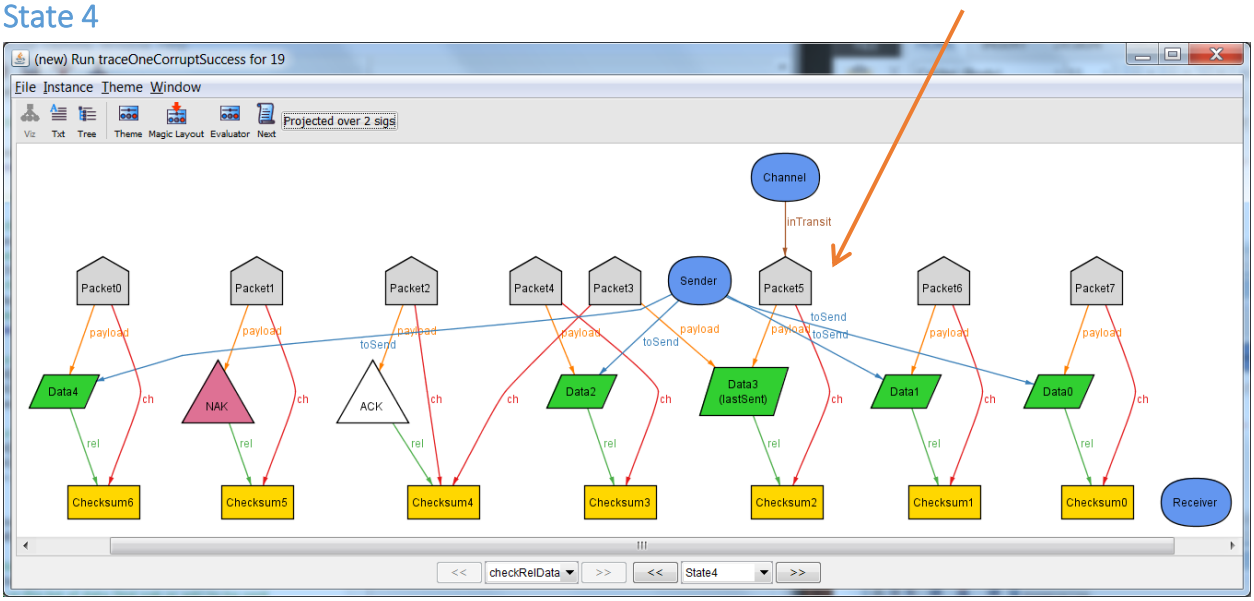

Receive: Data 3 is received.

## State 3



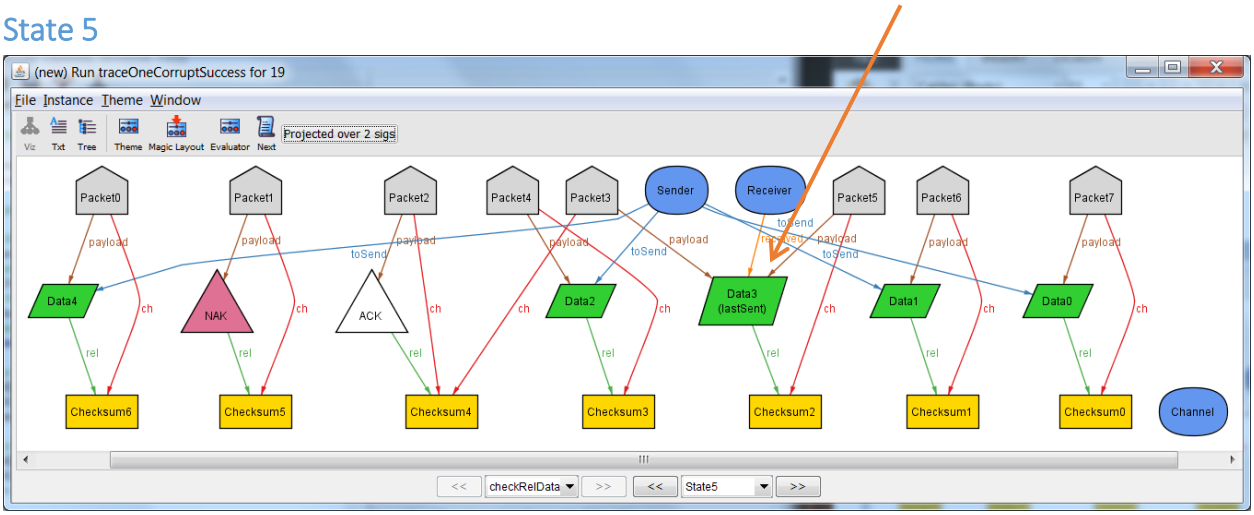

Response: The checksum of the received packet was invalid. The data is removed from the Receiver, and a Nak is sent back to Sender.

## State 4



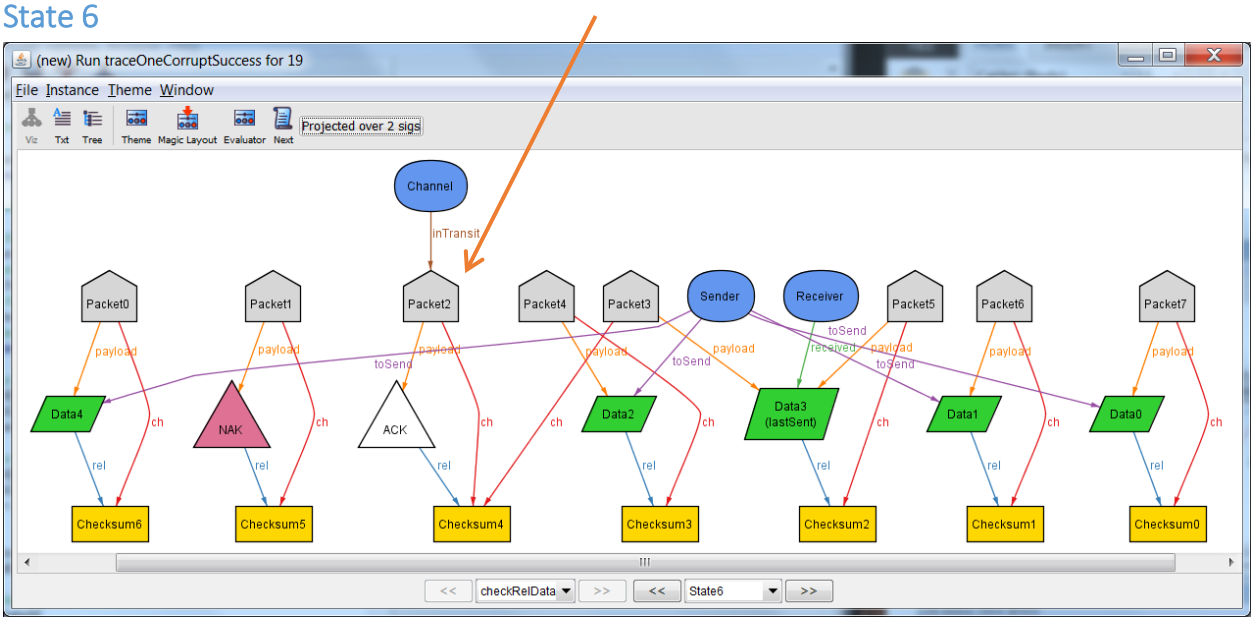

Send: Send Packet 5.

## State 5



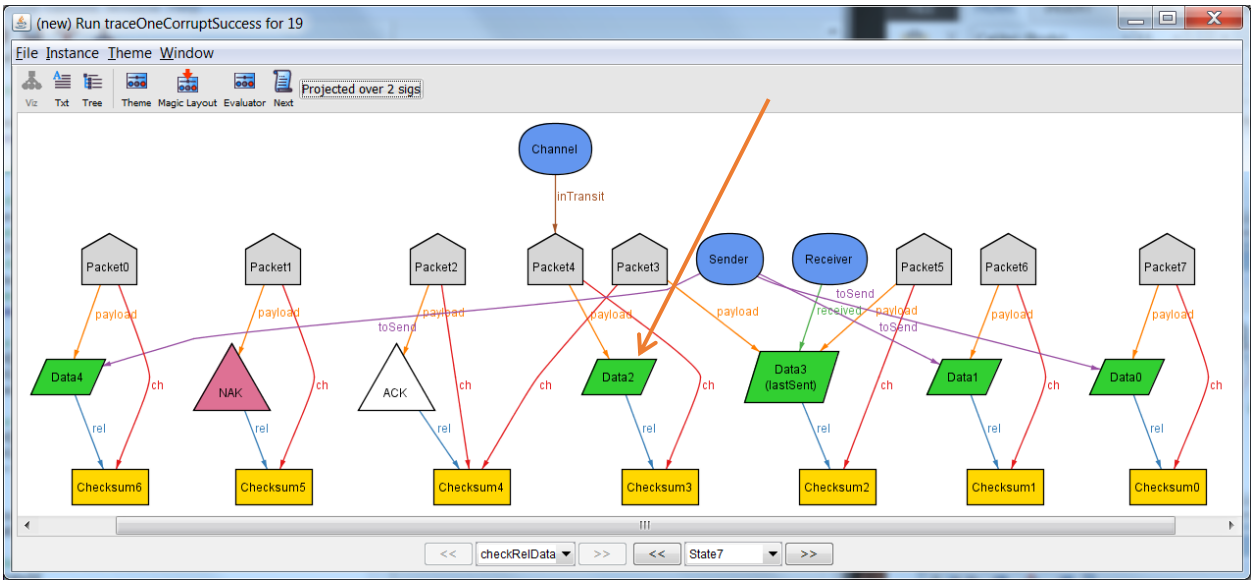

Receive: Data 3 is received.

## State 6



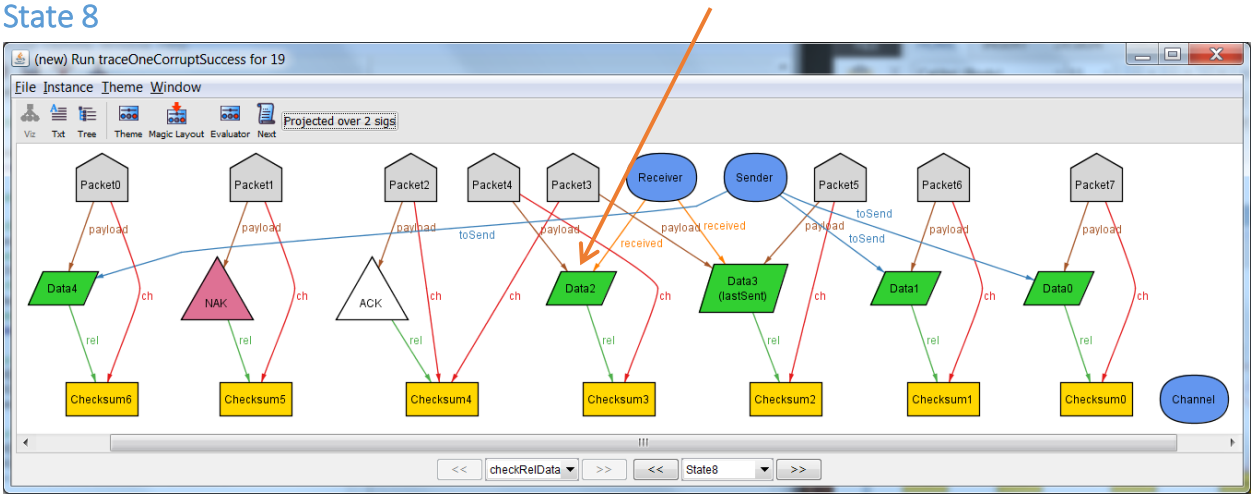

Response: The checksum of the last receive was valid, so the data is kept in the Receiver and an Ack is sent.

## State 7



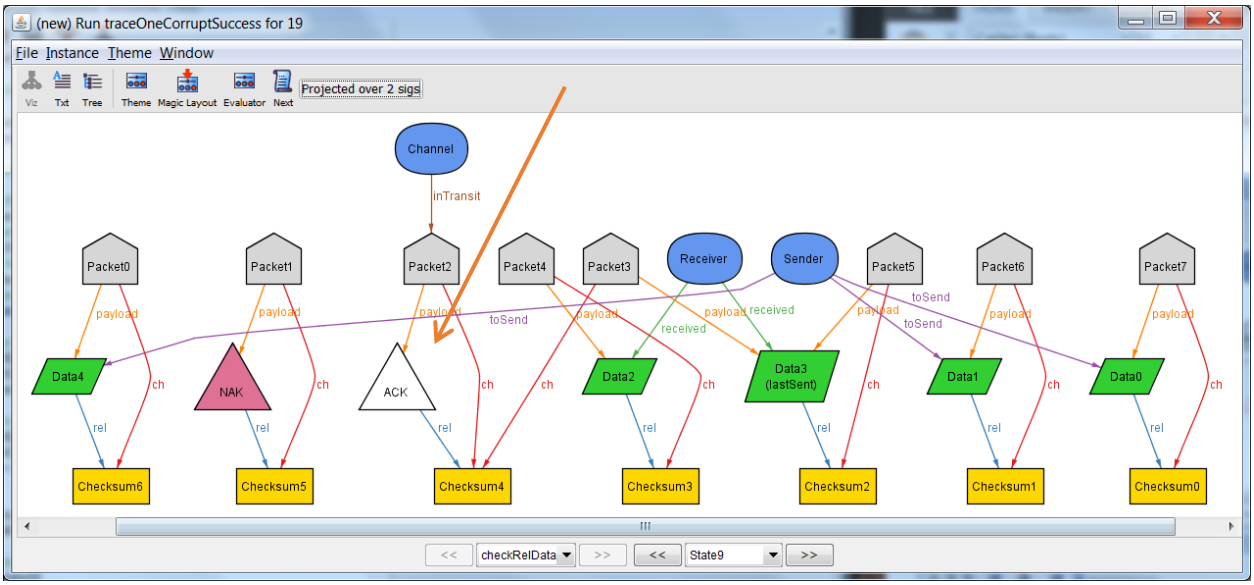

Send: Send Packet 4.

## State 8



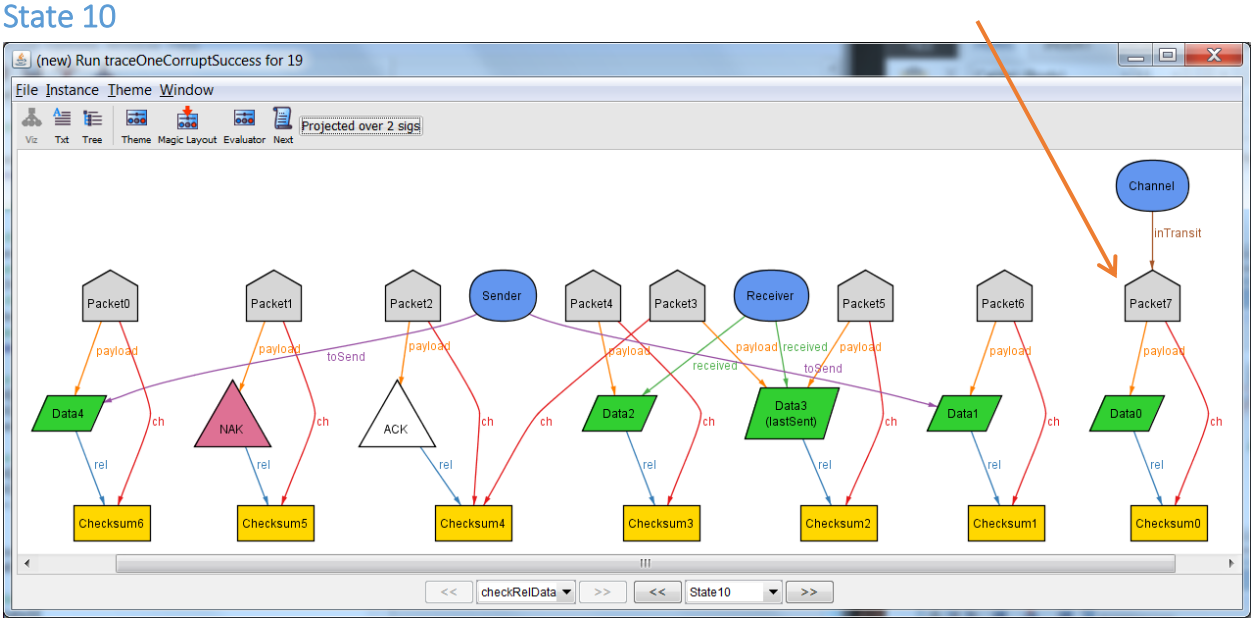

Receive: Data 2 is received.

## State 9



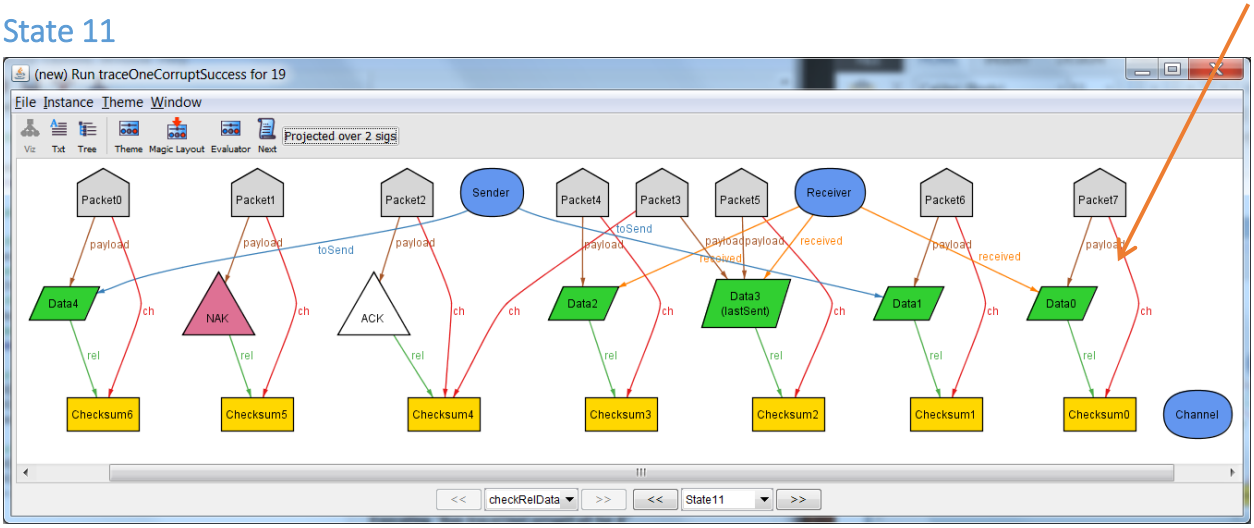

Response: The checksum was valid, so store the data and send Ack.

## State 10



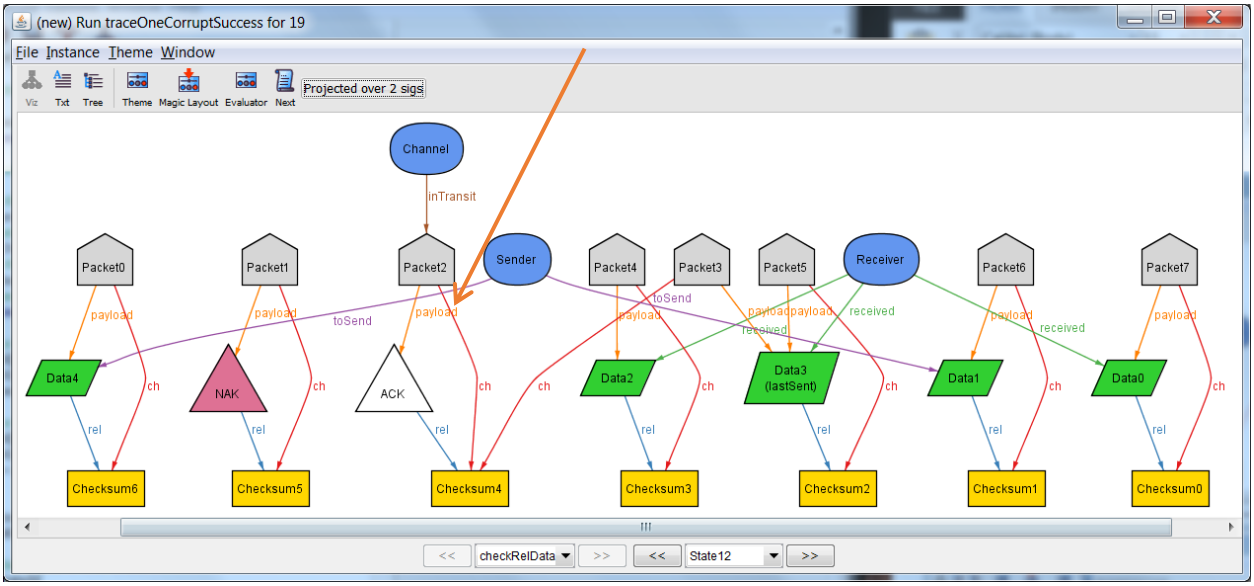

Send: Send packet 7.

## State 11



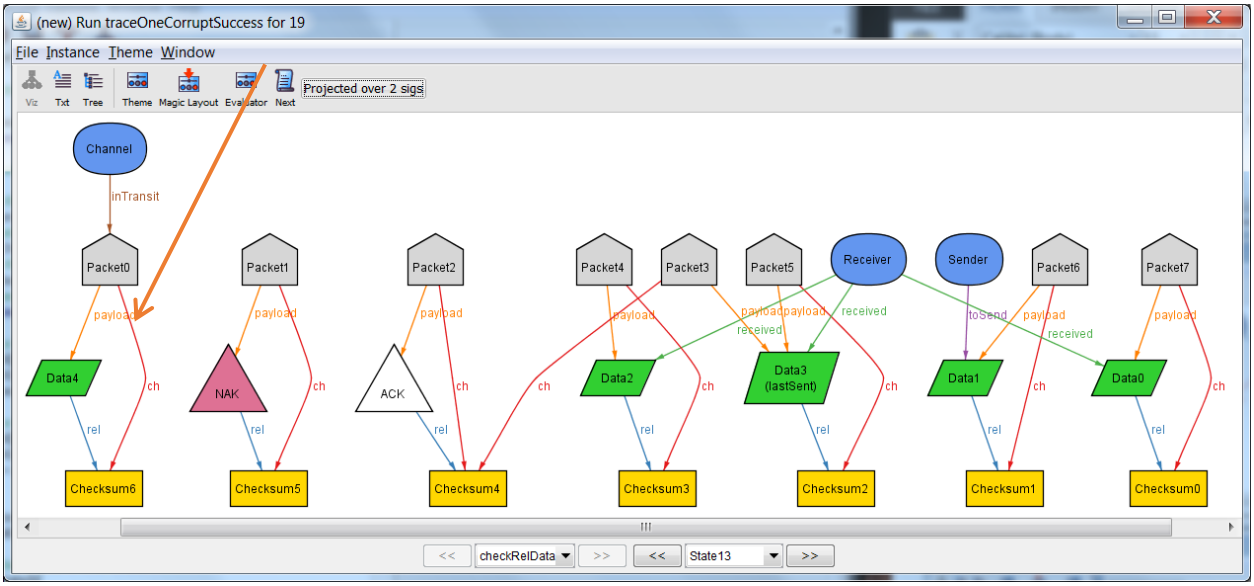

Receive: Receive data 0.

## State 12



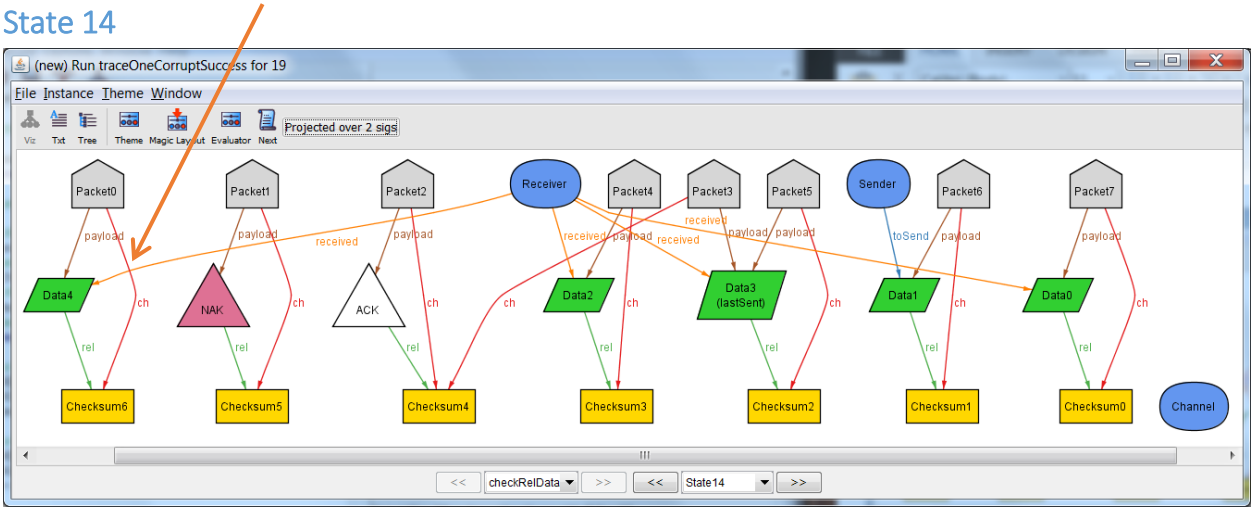

Response: The checksum was valid, so store the data and send Ack.

## State 13



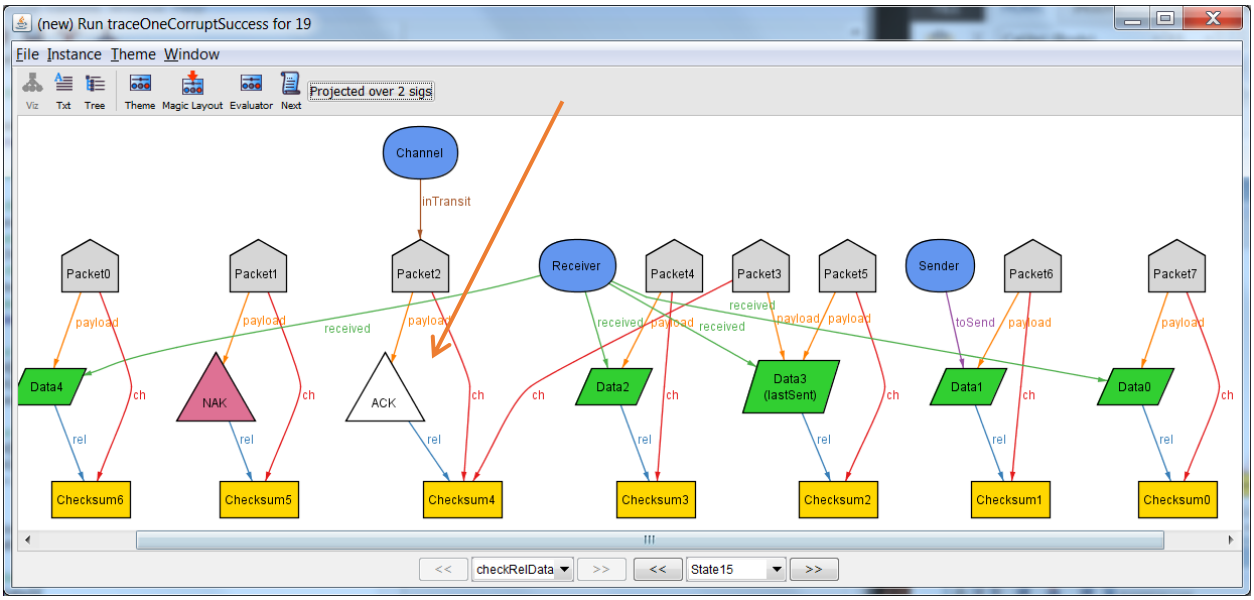

Send: Send Packet0.

## State 14



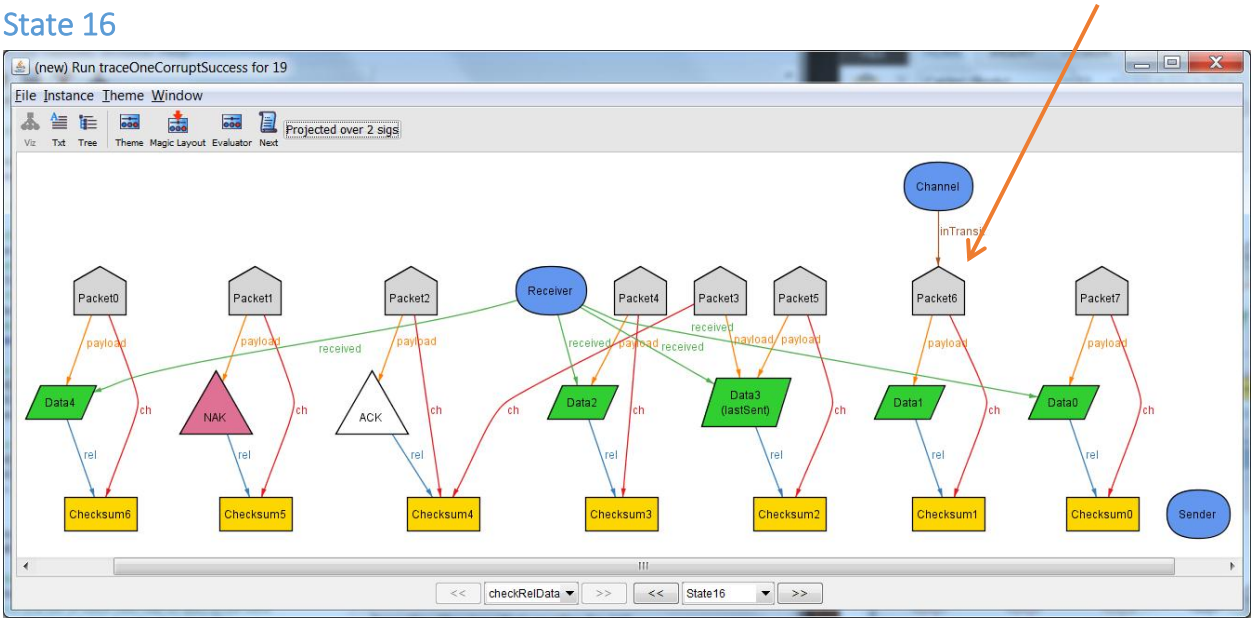

Receive: Receive Data 4.

## State 15



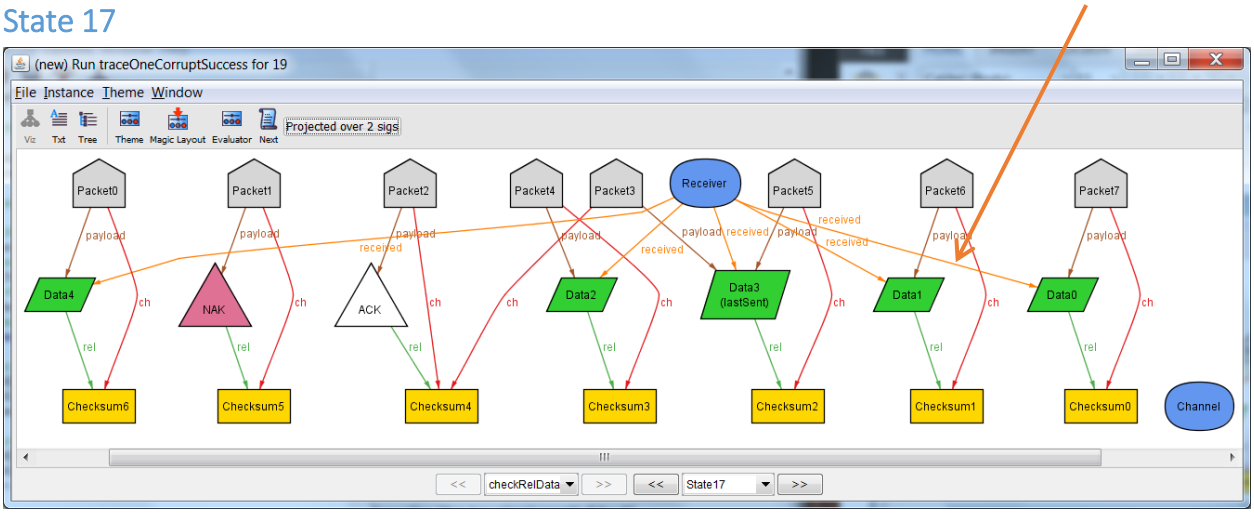

Respond: The checksum was valid, so store the data and send an Ack.

## State 16



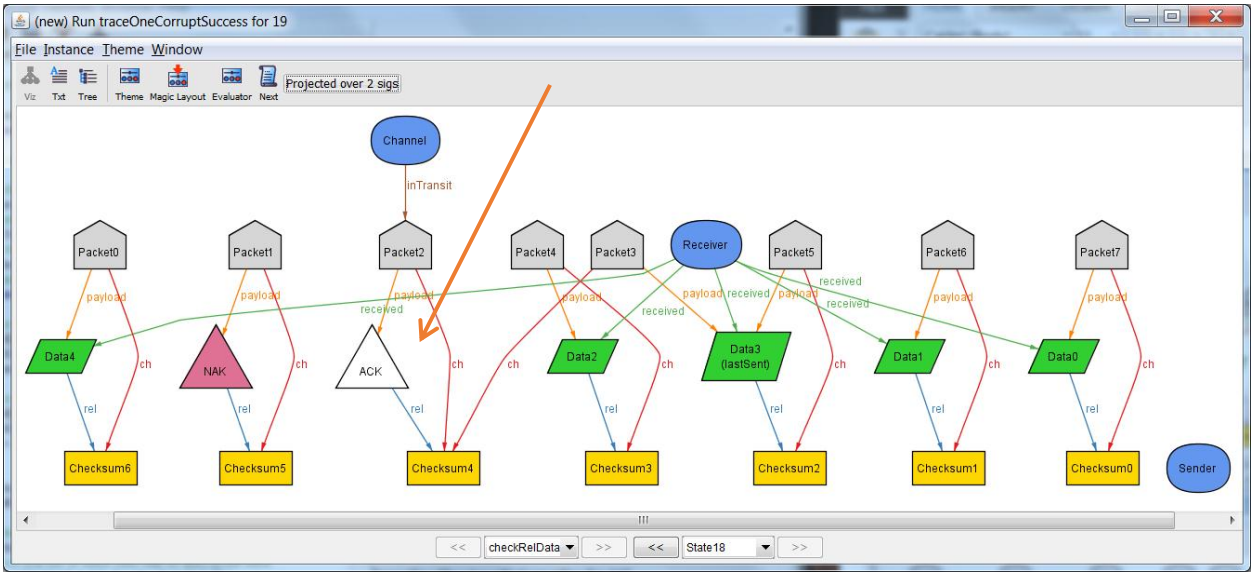

Send: Send Packet 6.

## State 17




Receive: Receive data 1.

## State 18




Respond: The checksum was valid, so store the data and send back Ack.

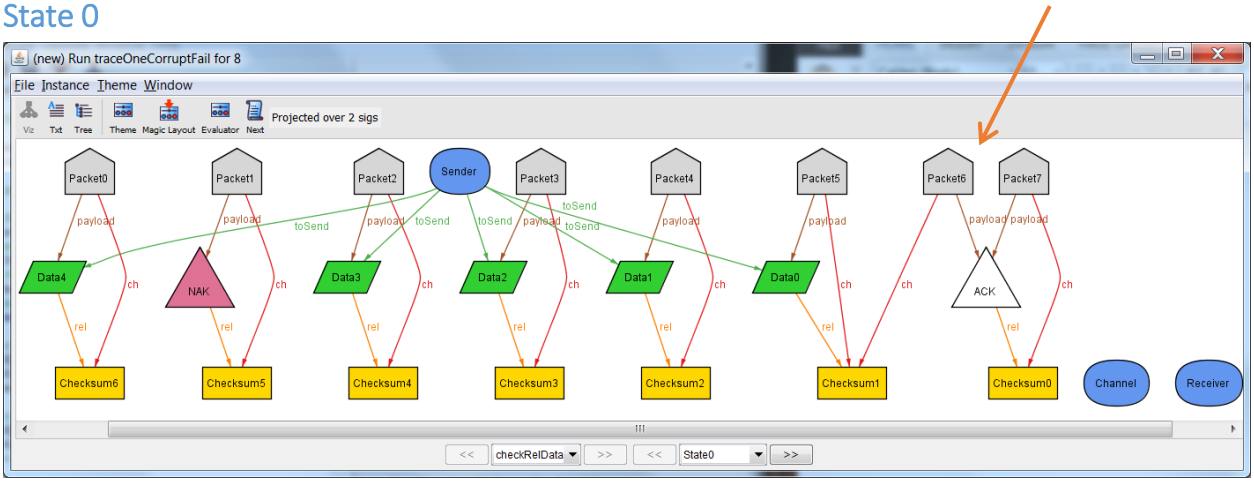
Thus it is possible to successfully send every data even if some corruption is present.

## Property 2 – Unsuccessful Transfer

It is possible that a transfer could be unsuccessful. This event occurs if a packet containing a response (Ack or Nak) is corrupted. No contingency is made for this occasion. Thus, the system will at that point freeze and be unable to complete the transfer. Thus property 2 does not hold.

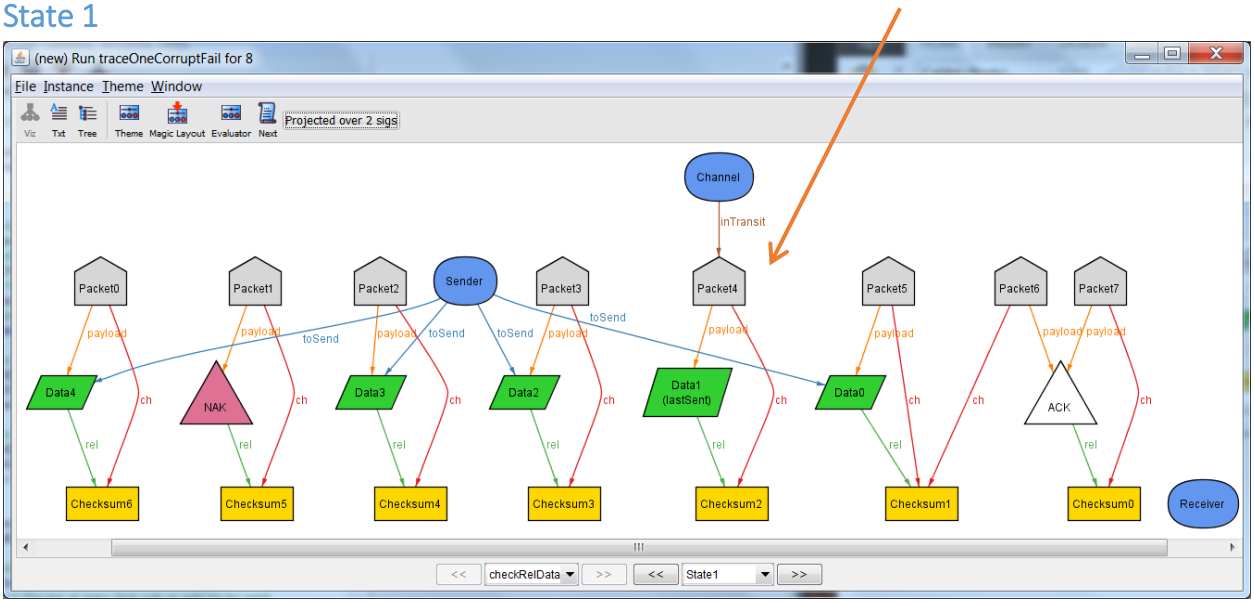
This trace again follows the Send, Receive, Response pattern.

### State 0



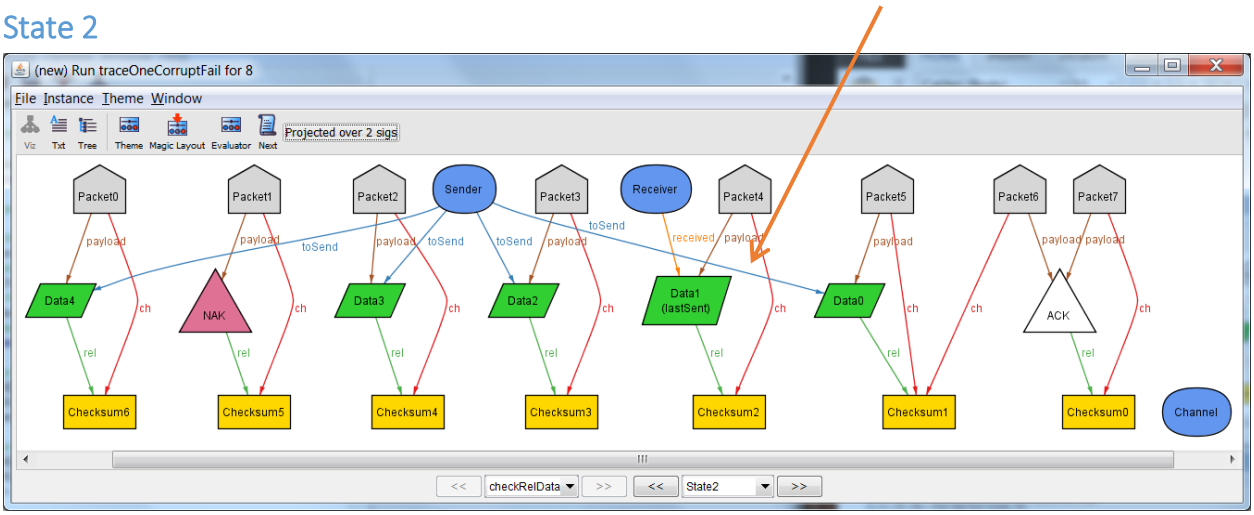

In the initial state, all of the data is referenced by the Sender. Additionally, we notice the presence of a corrupted Packet, number 6, that contains Ack but whose checksum references Data 0.

### State 1



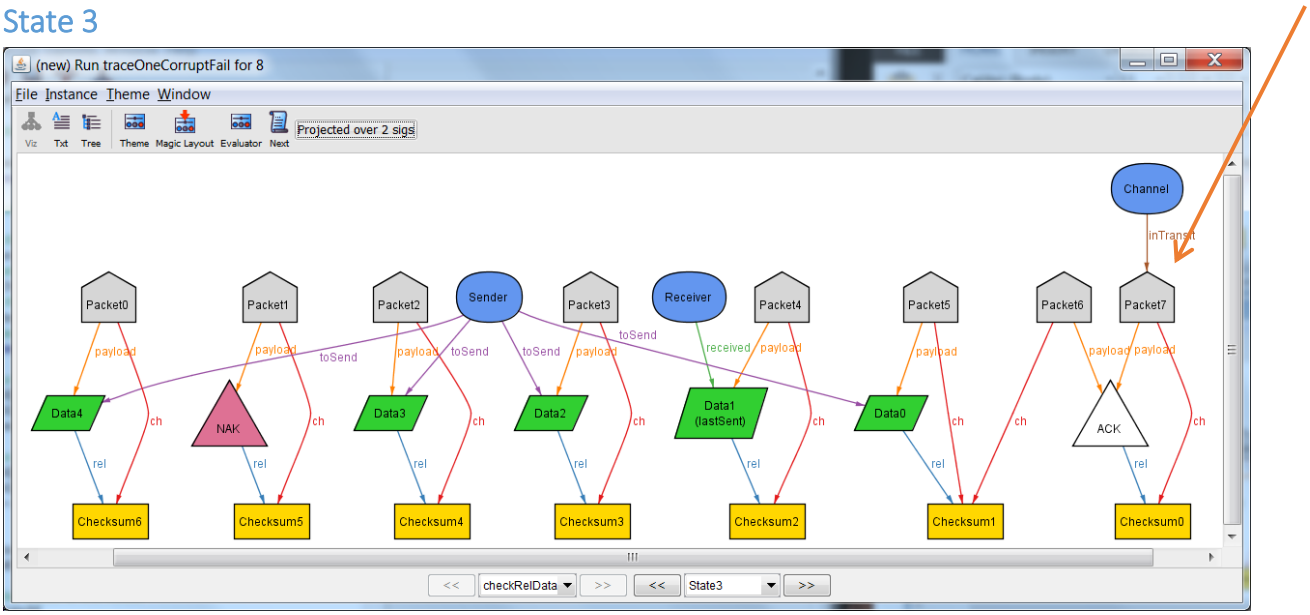

Send: Data 1 is sent normally.

## State 2



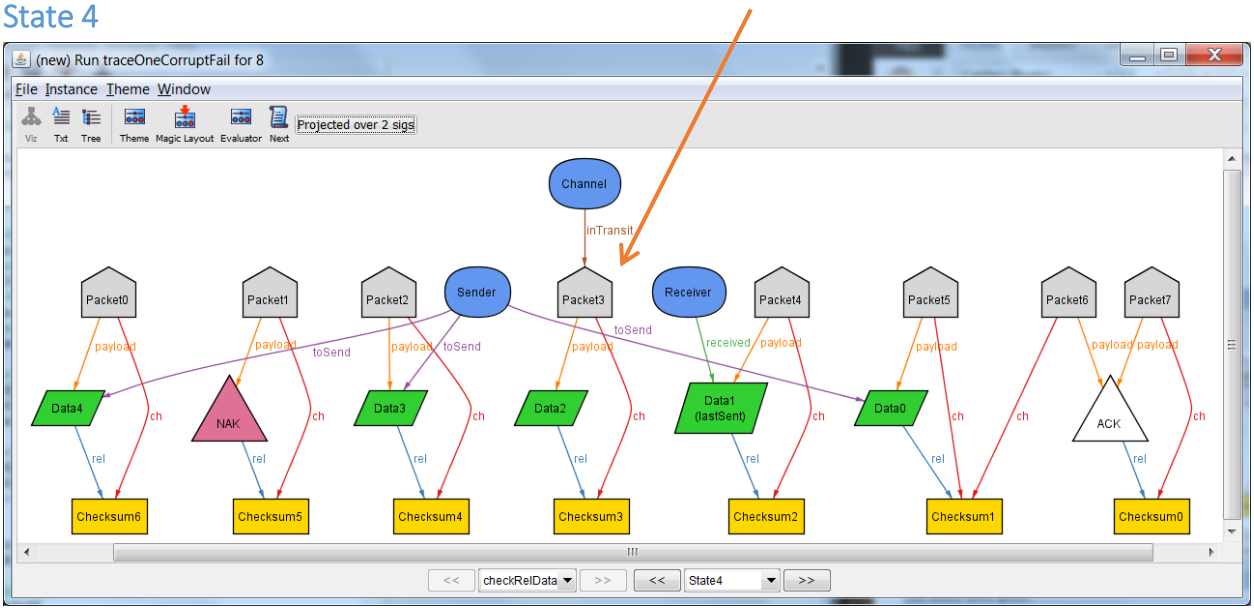

Receive: Data 1 is successfully received.

## State 3



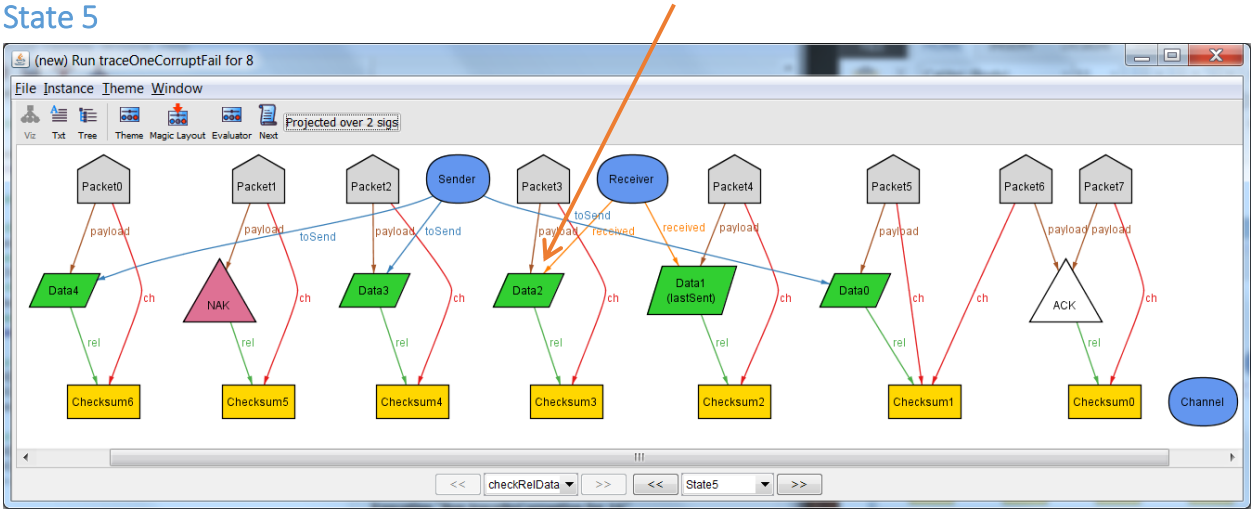

Response: The receive was successful, so respond with Ack.

## State 4



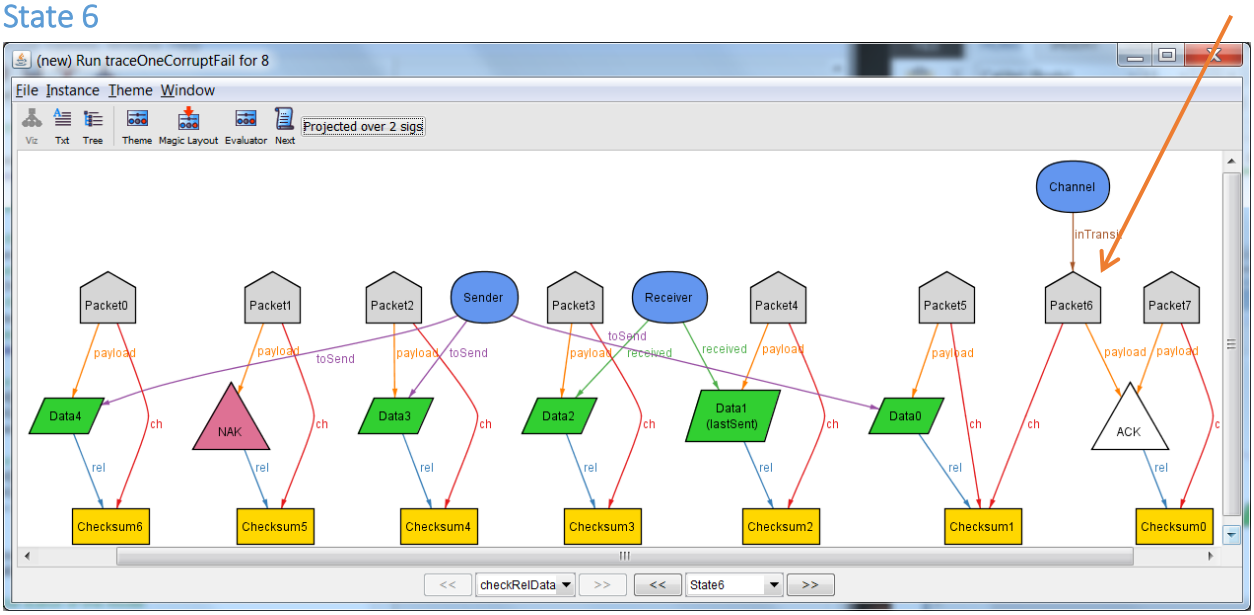

Send: Data 2 is sent normally.

## State 5



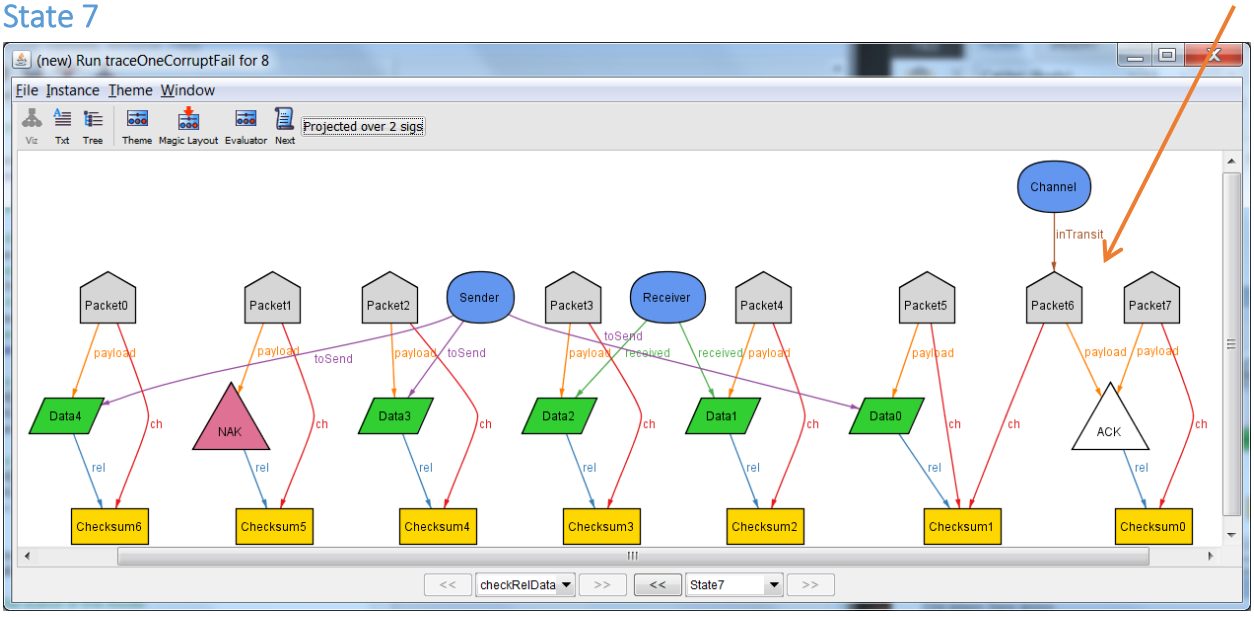

Receive: Data 2 is successfully received.

## State 6




Response: The receive was successful, so we respond with an Ack. However, a bad Ack packet, number 6, is being sent.

## State 7




Skip: The sender receives the bad Ack packet. According to our model, the Sender has no instructions at this point. Thus, the system is stuck in this state. The transfer can no longer be completed. Property 2 therefore does not hold.