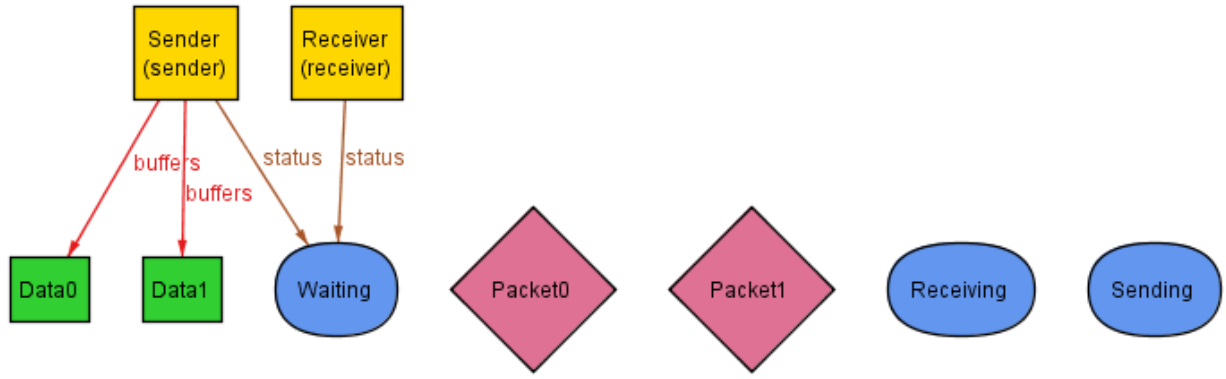
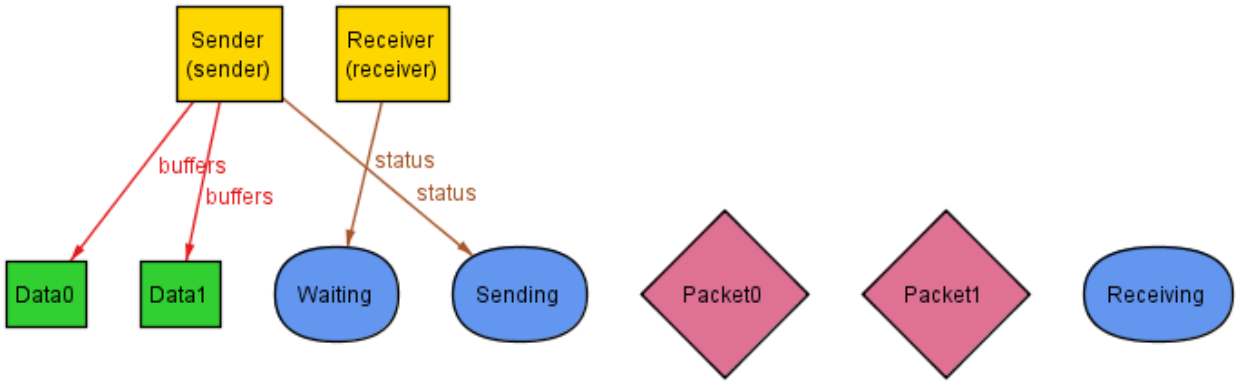
RDT10 Writeup
Team Carry Harder
John Krasich
Matt Lash
Caleb Post

1. Is it possible to transmit all of the data in the sender's buffer to the receiver's buffer?

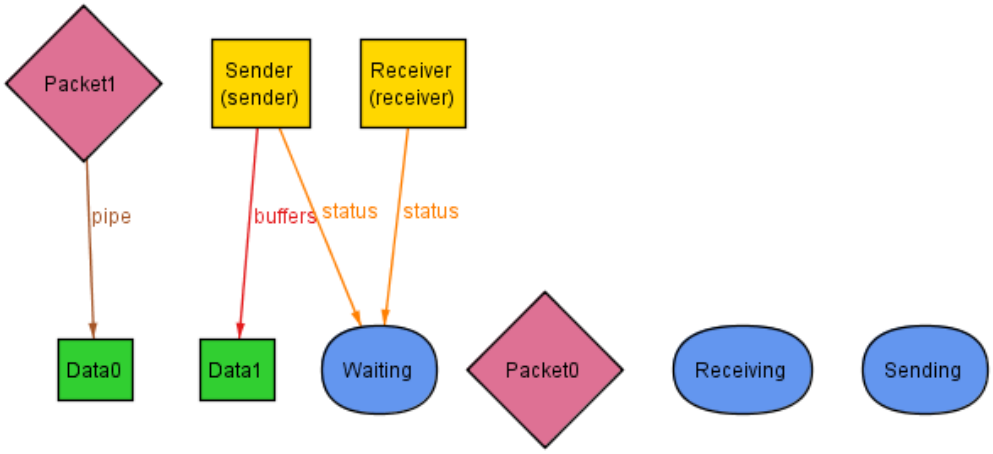
   Yes. Using two packets we can generate the path as follows:
   State 1: Sender has data to send, both sender and receiver are waiting.



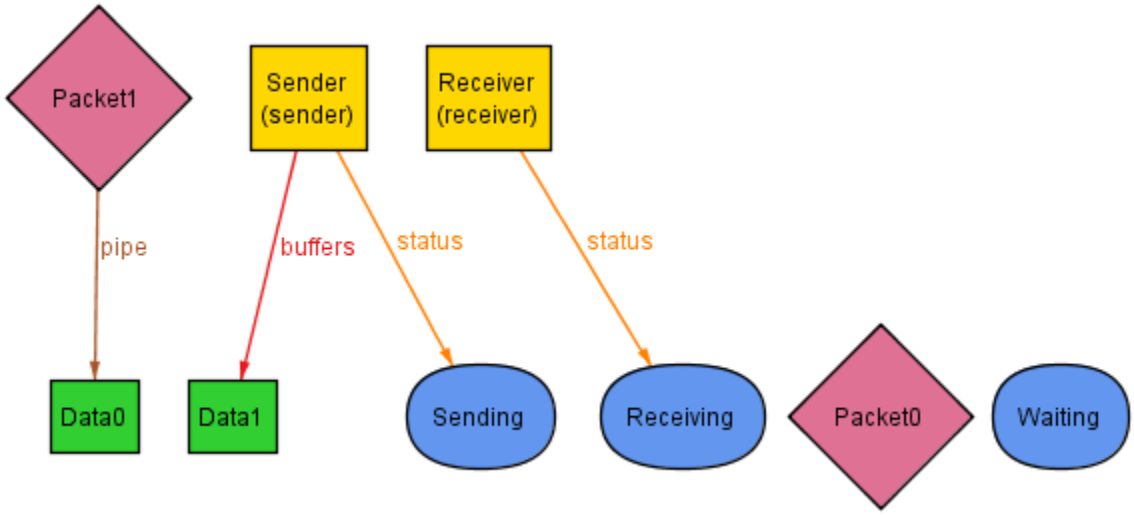
   State 2: Sender has gotten into the Sending state and is about to transmit data in a packet.



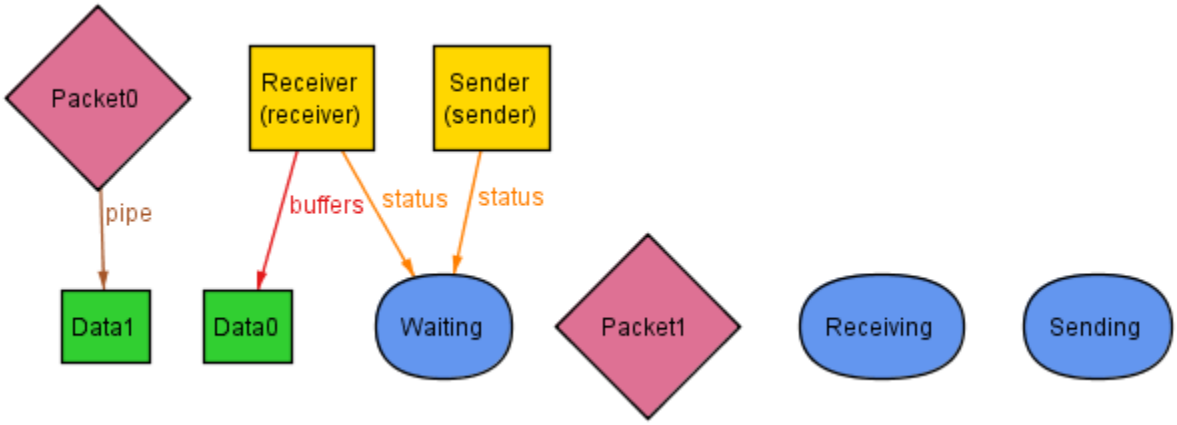
   State 3: Sender has put some data into the pipe, and has gone back into the waiting state.



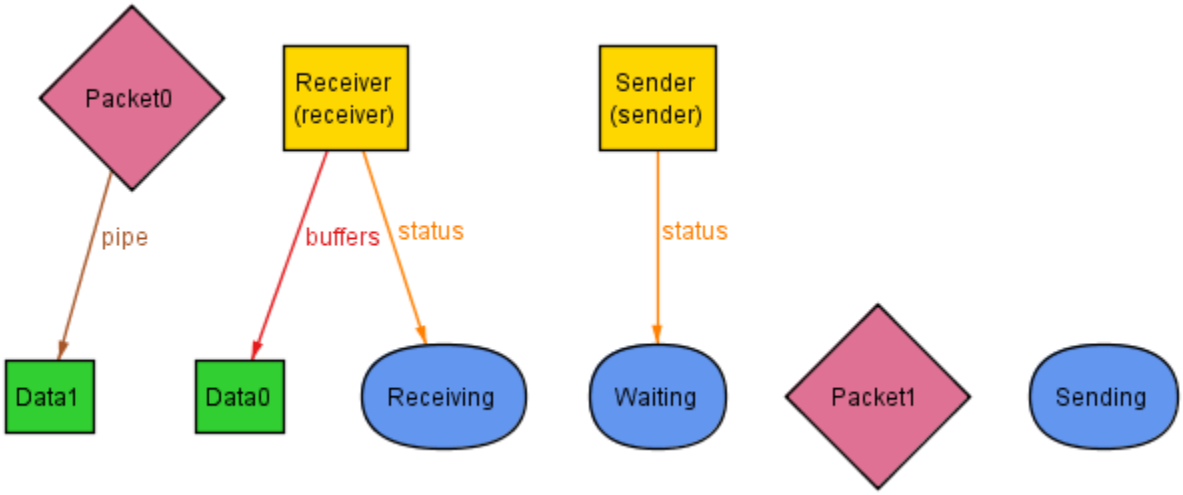
State 4: Sender has once again gotten ready to send, and the receiver has gone into receiving state since there is data in the pipe that it needs to get.



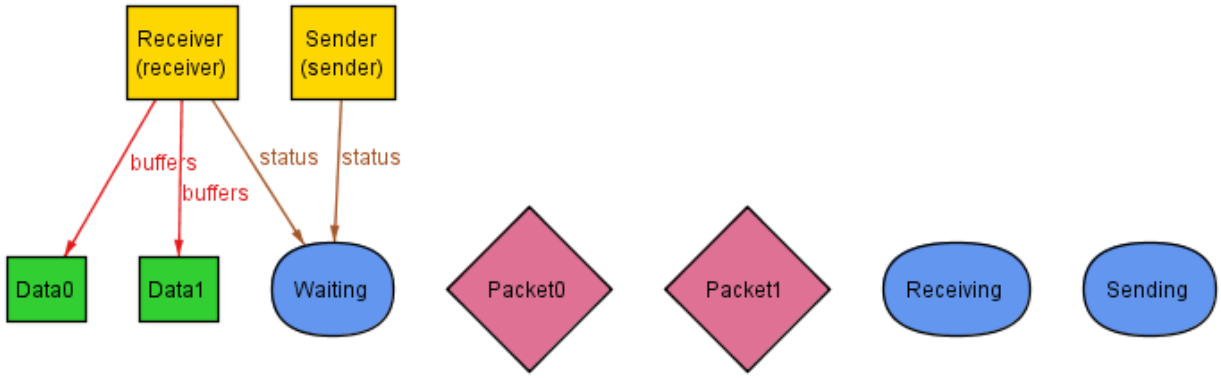
State 5: Sender and Receiver have both gone back to waiting, having put its data into the pipe and receiving it from the pipe respectively.



State 6: Receiver has once again gone into receiving mode since it senses data in the pipe.



State 7: Receiver has extracted the data from the pipeline, getting all the data out of the pipe, and all the data from the sender.



2. Is it always possible to transmit all of the data in the sender's buffer to the receiver's buffer?
   Yes. Under two key assumptions this is true with the model that we have created. We added the constraint that there will be no loss of packets in the pipe, and there is no corruption of packets in the pipe. Using these two additional constraints in the model lets us create no counterexample when running the following assertion. (which checks to make sure that the last system state has no packets left in the pip and the receiver received all the data)

```
assert alwaysSends {
   (no p: Packet | some last.pipe[p]) and (all d: Data | d in last.buffers[last.receiver]) and (no d: Data|d in last.buffers[last.sender])
}
```

   The loss of packets in the pipe is a known flaw with RDT 1.0 and is why there are more RDT Protocols, so we did not allow for them in our model since it is a known error and we were looking for unknown errors which could occur in the protocol. If we impose a possible loss of packets into the pipe, then the protocol becomes unreliable and there is a counterexample there.