Programming Assignment 2: Naughty Receiver – Reliable Data Transfer

In the previous programming assignment, you learned socket programming. In this assignment, you will explore reliable data transfer mechanism utilizing client-server socket communication. We have introduced RDT 3.0 (Chapter 3-part2.ppt, Slide No. 9) in chapter 3. RDT3.0 is a stop-and-wait protocol that uses alternative bit to realize the reliable data transfer. In this assignment, you will implement client-server communication on two separate processes similar to what you did in last programming assignment. Then, depending on your preference, you may implement the RDT sender on the server side and the RDT receiver on the client side or vice versa. The primary function for the RDT sender is to transfer data message (through socket) to the RDT receiver according to RDT 3.0 mechanism. The RDT receiver would ACK the message based on RDT 3.0 mechanism. To simplify the problem, we assume that the sender will always have data to transmit. The data can be defined as a set of phrases such as the days of the week.

data=["Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"]

The sender fetches one of phrases and attaches a proper sequence number to generate a message, which would be sent by the socket to the receiver side. The sender picks the phrases in order. When reaching the last phrase of data set, the sender would go back to the first phrase and continue assembling the message. In this way, the sender would always have data for transfer. As discussed in Chapter 3, the FSM of RDT 3.0 is shown in Figure 1.

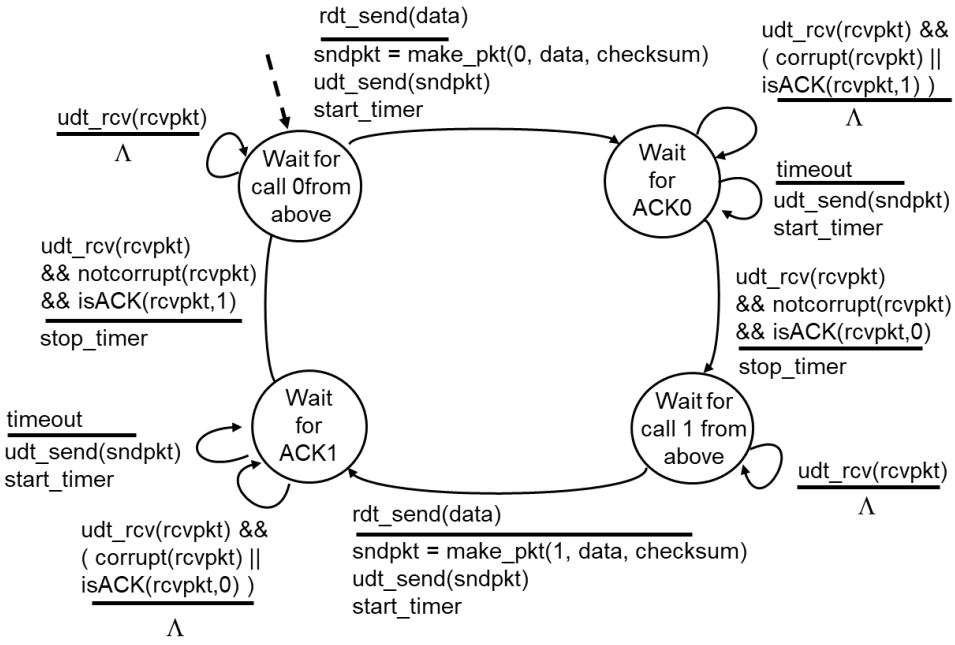

Figure 1: RDT 3.0 Sender FSM

Because of our assumption of continuous data, the sender no longer needs to wait for data from above. So the FSM can be simplified. The modified FSM is shown in Figure 2.

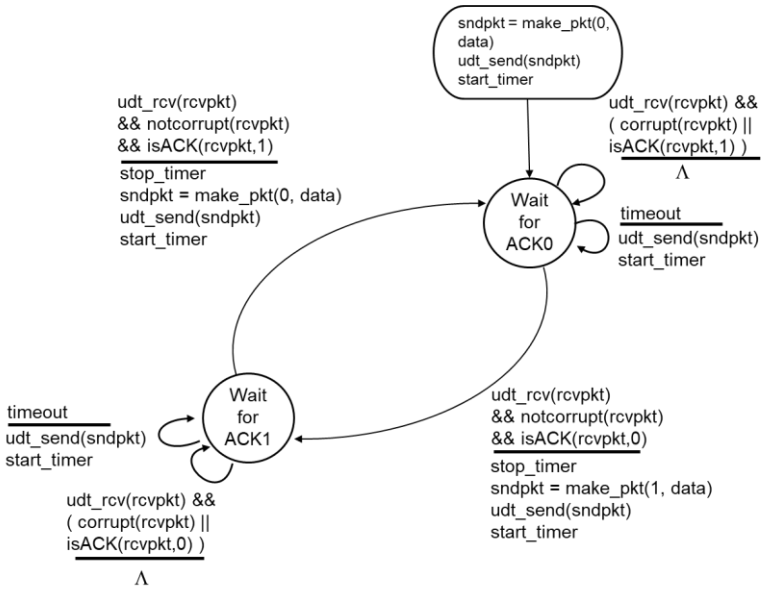

Figure 2: Modified Sender FSM

As you recall, the RDT 3.0 assumes an unreliable channel between the sender and the receiver. However, the socket communication is reliable. To simulate the effect of unreliable packet transmission, we add a naughty receiver function to the receiver side. The FSM of the revised receiver is shown in Figure 3 and the Naughty_Receiver() function is shown in Figure 4. As shown in Figure 4, instead of just sending an ACK with correct sequence number, the Naughty_Receiver function has options to send corrupted ACK, not to send ACK or to send ACK with incorrect sequence number. This function will randomly choose one of the four options (i.e. generate a random number to choose one of the four options with equal chance) to make the sender experience the different error conditions and thus go through all the transitions and actions defined in the RDT 3.0. Note: the checksum is not mandatory in this assignment. When generating the corrupted ACK, the receiver can simply generate a message of "corrupted ACK" (as long as the sender understands it is a corrupted ACK).

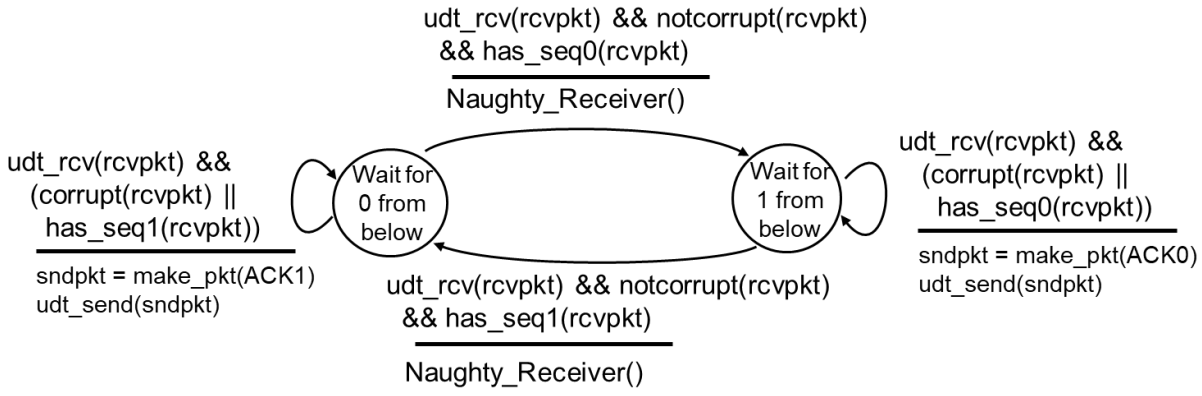

Figure 3: Receiver FSM

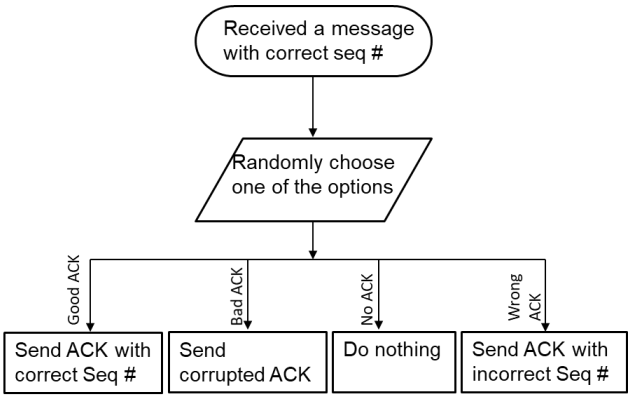

Figure 4: Naughty Receiver Function

In this assignment, you will use printout messages to indicate the actions of the sender and the receiver. The sample printout messages are shown in the appendix. You can define your termination condition. It could be stop after sending certain number of messages or after fixed number of iterations. Like programming assignment 1, you will implement both client-server processes on the same server (eustis.eecs.ucf.edu). You can use C/C++, Java, and Python to program this project. Eustis supports all these programming languages.

Submission:

1. You will submit your source codes
2. You will submit a report to describe the design of your programs, explain how to compile/run your codes and include screenshots of the running sender and receiver screens (your screens only need to show part of output messages since the complete set of messages may be much longer). Then, copy and paste the complete printout messages for both sender and receiver to your report (or to text files). You printout messages should show all four options (i.e. each option happened at least once) in the Naughty_Receiver().
3. Since you need to submit several files, please zip all your files into one file and submit through webcourses.

Appendix: Sample Printout Messages

RDT Sender:

```
Lets start
Connected by ('127.0.0.1', 57586)
Sender sent a message: Sunday 0
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Timeout. Send the message again
Sender sent a message: Sunday 0
Sender received a valid ACK for 0 , send next message
Sender sent a message: Monday 1
Sender received a corrupted ACK, keep waiting
Continue waiting
```

Continue waiting
Continue waiting
Continue waiting
Continue waiting
Timeout. Send the message again
Sender sent a message: Monday 1
Sender received a valid ACK for 1 , send next message
Sender sent a message: Tuesday 0
Sender received an ACK with wrong sequency number, keep waiting
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Timeout. Send the message again
Sender sent a message: Tuesday 0
Sender received a valid ACK for 0 , send next message
Sender sent a message: Wednesday 1
Sender received a valid ACK for 1 , send next message
Sender sent a message: Thursday 0
Sender received an ACK with wrong sequency number, keep waiting
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Timeout. Send the message again
Sender sent a message: Thursday 0
Sender received a valid ACK for 0 , send next message
Sender sent a message: Friday 1
Sender received a valid ACK for 1 , send next message
Sender sent a message: Saturday 0
Sender received a corrupted ACK, keep waiting
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Timeout. Send the message again
Sender sent a message: Saturday 0
Sender received a valid ACK for 0 , send next message
Sender sent a message: Sunday 1
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Timeout. Send the message again
Sender sent a message: Sunday 1
Sender received a valid ACK for 1 , send next message
Sender sent a message: Monday 0

Sender received an ACK with wrong sequency number, keep waiting
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Timeout. Send the message again
Sender sent a message: Monday 0
Sender received a valid ACK for 0 , send next message
Sender sent a message: Tuesday 1
Sender received a corrupted ACK, keep waiting
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Continue waiting
Timeout. Send the message again
Sender sent a message: Tuesday 1
Sender received a valid ACK for 1 , send next message
Sender sent a message: Wednesday 0
Sender received a corrupted ACK, keep waiting

RDT Receiver:

Receiver just correctly received a message: Sunday 0
How do you respond?
(1) send a correct ACK; (2) send a corrupted ACK; (3) do not send ACK; (4) send a wrong ACK
Receiver do not send ACK
Receiver just correctly received a duplicated message: Sunday 0
Receiver responds with ACK 0
Receiver just correctly received a message: Monday 1
How do you respond?
(1) send a correct ACK; (2) send a corrupted ACK; (3) do not send ACK; (4) send a wrong ACK
A corrupted ACK is sent
Receiver just correctly received a duplicated message: Monday 1
Receiver responds with ACK 1
Receiver just correctly received a message: Tuesday 0
How do you respond?
(1) send a correct ACK; (2) send a corrupted ACK; (3) do not send ACK; (4) send a wrong ACK
Receiver incorrectly responds with ACK 1
Receiver just correctly received a duplicated message: Tuesday 0
Receiver responds with ACK 0
Receiver just correctly received a message: Wednesday 1
How do you respond?
(1) send a correct ACK; (2) send a corrupted ACK; (3) do not send ACK; (4) send a wrong ACK
Receiver correctly responds with ACK 1
Receiver just correctly received a message: Thursday 0
How do you respond?
(1) send a correct ACK; (2) send a corrupted ACK; (3) do not send ACK; (4) send a wrong ACK
Receiver incorrectly responds with ACK 1
Receiver just correctly received a duplicated message: Thursday 0
Receiver responds with ACK 0

Receiver just correctly received a message: Friday 1
How do you respond?
(1) send a correct ACK; (2) send a corrupted ACK; (3) do not send ACK; (4) send a wrong ACK
Receiver correctly responds with ACK 1
Receiver just correctly received a message: Saturday 0
How do you respond?
(1) send a correct ACK; (2) send a corrupted ACK; (3) do not send ACK; (4) send a wrong ACK
A corrupted ACK is sent
Receiver just correctly received a duplicated message: Saturday 0
Receiver responds with ACK 0
Receiver just correctly received a message: Sunday 1
How do you respond?
(1) send a correct ACK; (2) send a corrupted ACK; (3) do not send ACK; (4) send a wrong ACK
Receiver do not send ACK
Receiver just correctly received a duplicated message: Sunday 1
Receiver responds with ACK 1
Receiver just correctly received a message: Monday 0
How do you respond?
(1) send a correct ACK; (2) send a corrupted ACK; (3) do not send ACK; (4) send a wrong ACK
Receiver incorrectly responds with ACK 1
Receiver just correctly received a duplicated message: Monday 0
Receiver responds with ACK 0
Receiver just correctly received a message: Tuesday 1
How do you respond?
(1) send a correct ACK; (2) send a corrupted ACK; (3) do not send ACK; (4) send a wrong ACK
A corrupted ACK is sent
Receiver just correctly received a duplicated message: Tuesday 1
Receiver responds with ACK 1
Receiver just correctly received a message: Wednesday 0
How do you respond?
(1) send a correct ACK; (2) send a corrupted ACK; (3) do not send ACK; (4) send a wrong ACK
A corrupted ACK is sent