16.483-583 Network Design, Fall 2015, Dr. Vinod Vokkarane

Programming Project Phase 3: Implement RDT 2.2 over an unreliable UDP channel with bit-errors

Deadline: Sunday, Oct 11th (Midnight)

Project description: The TCP/IP stack has five layers, namely application, transport, network, link, and physical. In Phase 1, each student implemented the standard user datagram protocol (UDP) sockets. The intention was to transfer a file (say JPEG) between a UDP client process and a UDP server process. In Phase 2, each of you implemented reliable data transfer service over the reliable UDP connection developed in Phase 1. In Phase 3, each of you has to implement the RDT 2.2 protocol described in Section 3.4.1, page 218-227 of the course textbook. The finite state machines (FSM) of the sender and received are given below:

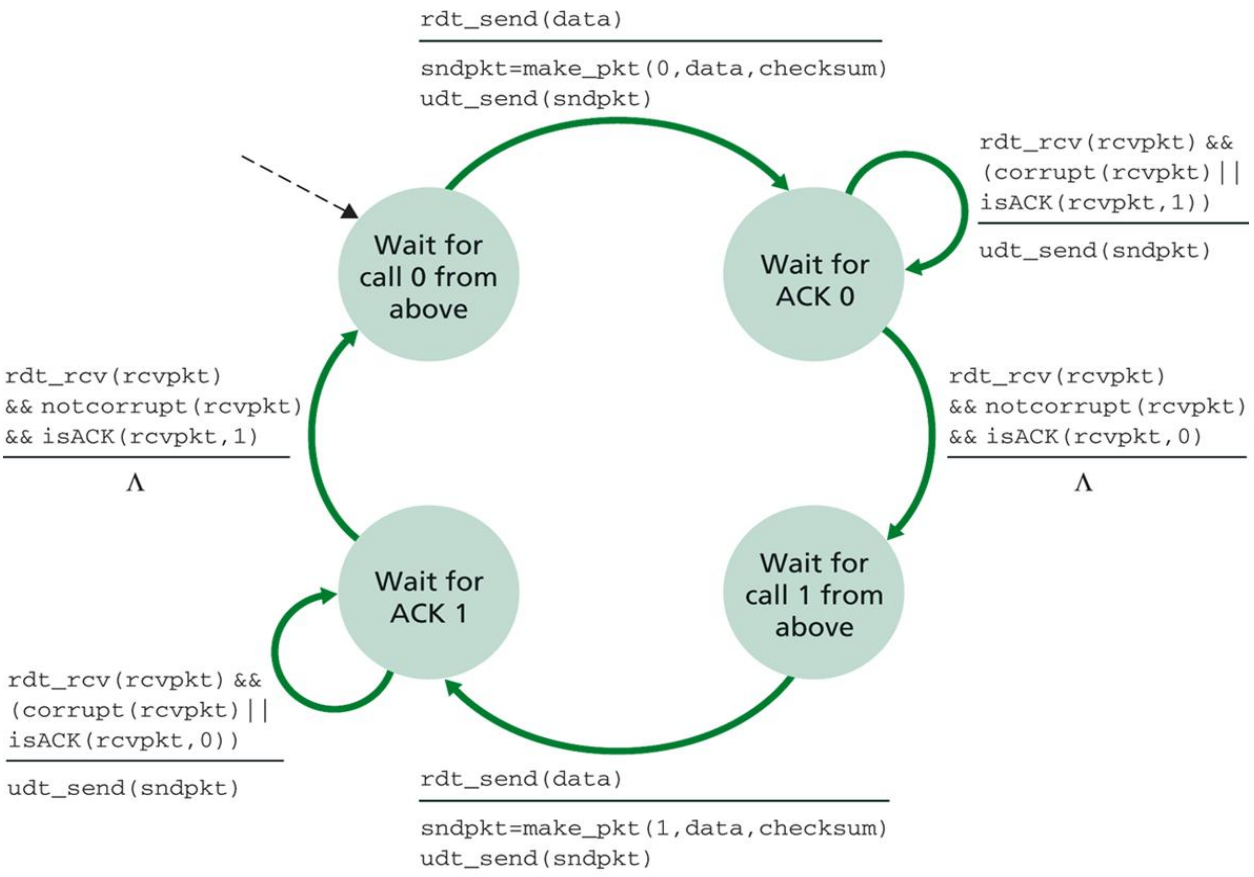

Figure 3.13 ♦ rdt2.2 sender

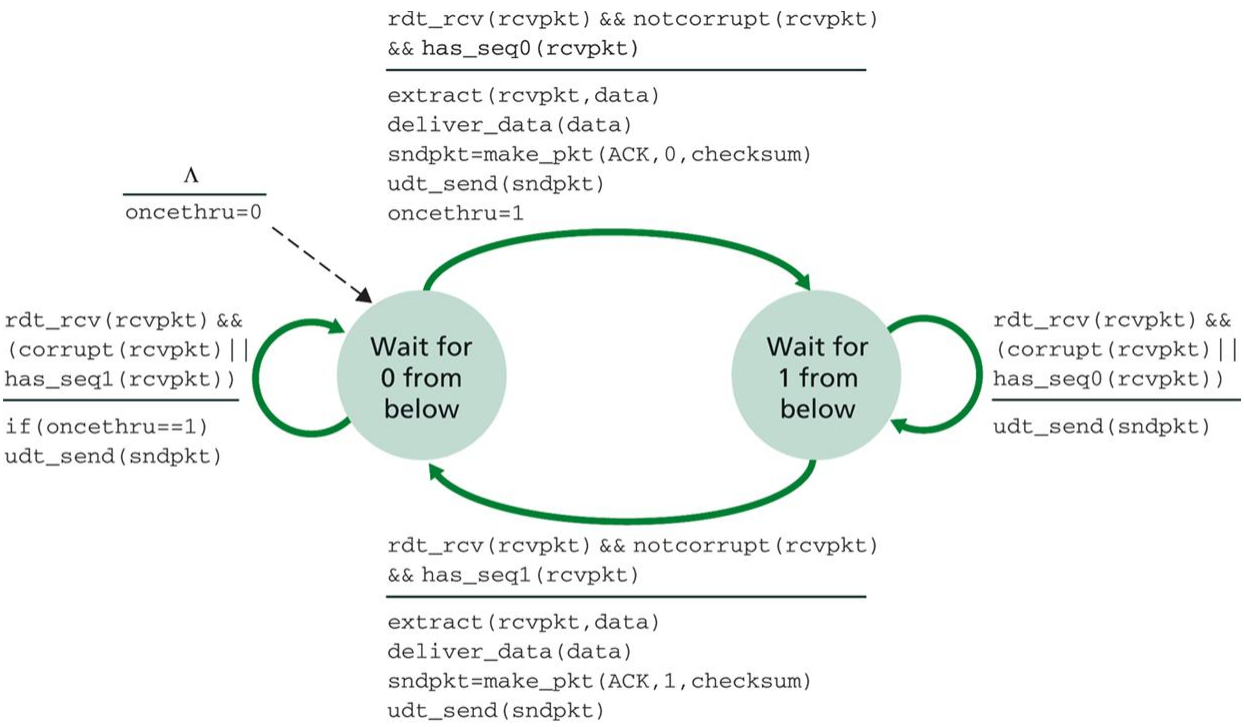

Figure 3.14 ♦ rdt2.2 receiver

The following are the basic implementation steps:

1. Pick a transfer file - JPEG image file (recommended), easier to identify loss of packets in a image file (lost pixels).
2. Make_packet function - parses the image file and breaks it down to several packets (set a fixed packet size, say 1024 bytes).
3. Use the UDP sockets (Phase 2) to send and receive RDT2.2 packets (UDP sockets are unreliable, so they simulate the unreliable Internet IP (network) layer).
4. Implement: Sequence Numbers (to identify duplicates), Checksum (implement your own (Do not use Java buit-in function!), similar to UDP), and ACKs (remember, RDT 2.2 is a NAK-free protocol).
5. The RDT2.2 receiver should assemble packets in order and deliver the entire transfer file to the application.

In your implementation, use binary variables for representing bits (instead of strings). In this phase, you will need to implement different data transfer scenarios

1. Option 1 - No loss/bit-errors.
2. Option 2 - ACK packet bit-error: you need to intentionally change the data bits of the received ACK packet at the sender and implement suitable recovery mechanism.
3. Option 3 - Data packet bit-error: you need to intentionally change the data bits of the received DATA packet at the receiver and implement suitable recovery mechanism.

Implement Optional Feature for Extra Credit (50% of Phase 3): Implement an applet/GUI to show the data transfer (display of image as the transfer happens) and the (sender and receiver) FSM.

Expectations: In this phase of the project, you will learn about the basic principles used to provide non-pipelined reliable data transfer over a data channel with bit errors.

Programming language: C, C++, C-sharp, Python or Java (your choice).

Deliverables:

Sender and Receiver source files (groupname.tar): well documented source code; mention ALL references for reuse of source code (if any).

Transfer File (No .EXE files): sample file used to test the functionality of your program. (Optional)

Updated Design Documents (groupname.doc): updated detailed design document with possibly screen-shots of a sample scenario.

ReadMe.txt: List the names of files submitted and their purpose, and explain steps required to set up and execute your program. Also, how to check for the different scenarios (1 - no loss/error; 2 - ACK bit error; 3 - Data packet error).

MyContribution.doc: I want each team member to submit this document individually (only one person still submits rest of the deliverables). Submit a bulleted list of your contributions during this project phase and also rate your teammate's contribution on a scale of 0 (did not do any work) - 5 (Awesome! did all the work). Please include a brief justification for your rating.

Note: All deliverables should be submitted through Blackboard. All submission will be inspected using the Plagiarism detection software.

Expectations: In this phase of the project, you will learn about the basic principles used to provide non-pipelined reliable data transfer over a perfectly reliable channel.

Programming language: C, C++, Python or Java.

Deliverables:

1. **ReadMe.txt**: Name of the team members, list the names of files submitted and their purpose, and explain steps required to set up and execute your program.
2. **Updated Design Documents** (yourlastname.doc): updated detailed design document with possibly screen-shots of a sample scenario.
3. **Sender and Receiver source files** (yourlastname.tar.gz/.zip): well documented source code; mention ALL references for reuse of source code (if any).
4. **Transfer File** (No .EXE files): sample file used to test the functionality of your program. (File is Optional)
5. **Individual Contribution (contribution.txt):** Bulleted list of tasks implemented by each team member.

Submit all your documents in a single compressed file with the name **StudentLastNamePhase3.zip**. All submission will be inspected using the **Plagiarism detection software.**

References:

1. Socket Programming
   - JAVA Socket Programming
   - Linux Gazette's Socket Programming and Beej's Guide
   - Socket Programming by JAVA World and O'REILLY JAVA Network Programming
2. UDP: RFC768
3. Principles of Reliable Data Transfer, Section 3.4.1, Page 206 -207, Kurose/Ross (6th Ed).