EECE 4830-5830 Network Design, Dr. Vinod Vokkarane

Programming Project Phase 2: Implement RDT 1.0 over a reliable UDP channel

Deadline: Sunday, Feb 12th (Midnight)

Project description: The TCP/IP stack has five layers, namely application, transport, network, link, and physical. In Phase 1, each of you implemented the standard user datagram protocol (UDP) sockets. In Phase 2, the intention is to transfer a file (say JPEG) between a UDP client process and a UDP server process. In Phase 2, we will provide reliable data transfer (RDT) service assuming that the underlying layer is perfectly reliable using UDP connection developed in Phase 1. Each group has to implement the RDT 1.0 protocol described in Section 3.4.1, page 218 of the course textbook.

The end-to-end system architecture is given below:

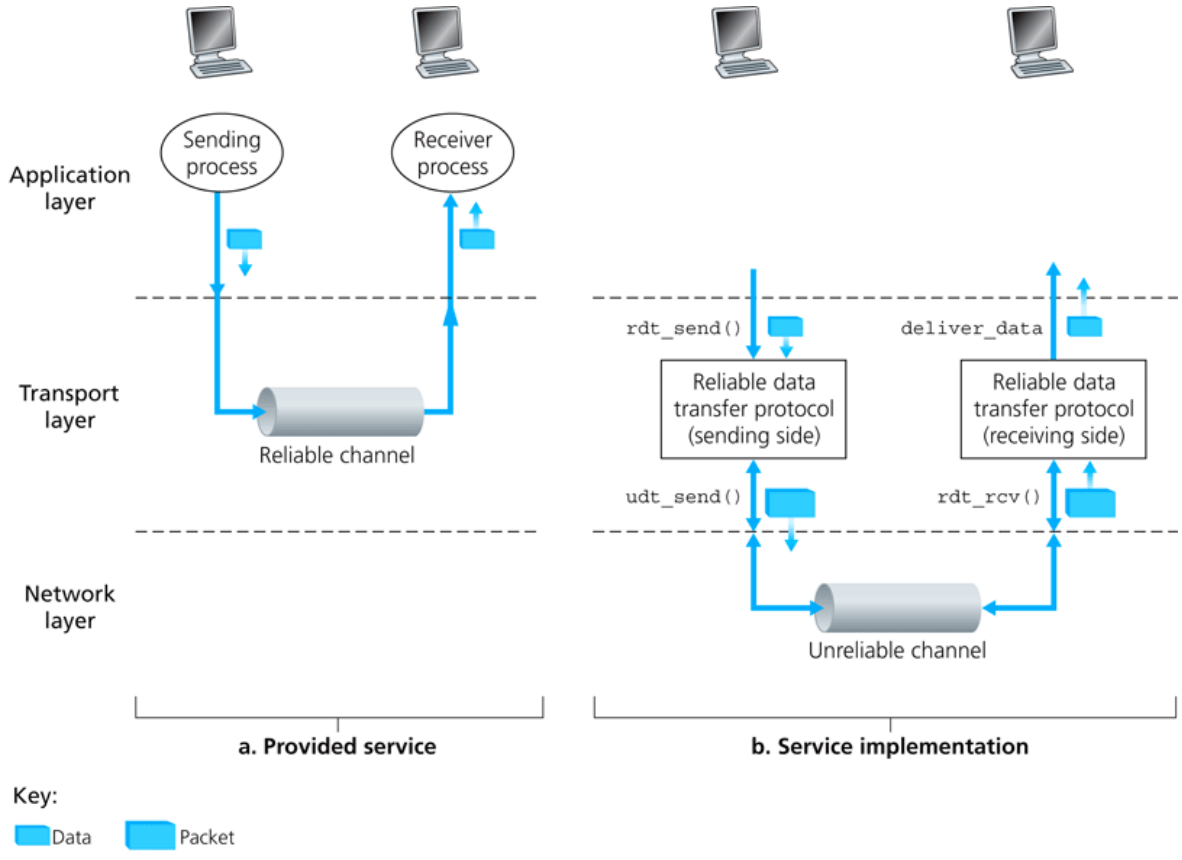
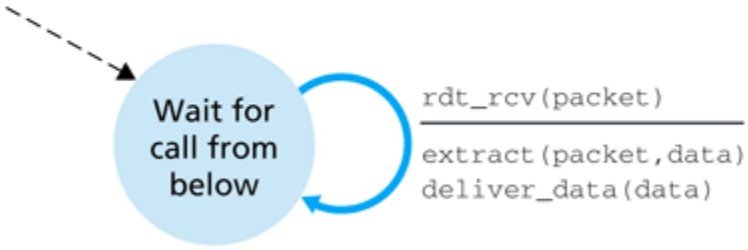


Figure 3.8 ♦ Reliable data transfer: Service model and service implementation

The finite state machines of the sender and received are given below:

a. rdt1.0: sending side



b. rdt1.0: receiving side

Figure 3.9 ♦ `rdt1.0` – A protocol for a completely reliable channel

The following are the basic implementation steps:

1. Pick a transfer file - JPEG image file (recommended), easier identifying loss of packets in an 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 1024B).

3. Use the UDP sockets (Phase 1) to send and receive RDT1.0 packets.

4. The RDT1.0 receiver should assemble packets in order and deliver the entire transfer file to the application.

5. You could write a simple GUI that handles this process. (Optional - extra-credit)

In your implementation, use Binary variables for representing bits (instead of strings).

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/doc):**

a) Bulleted list of tasks implemented by each team member and
b) Teammate rating between 1(not good) – 5 (excellent) on
   - project time commitment,
   - design contribution,
   - coding contribution,
   - debugging contribution, and
   - report preparation (confidential submission for each member).

One team member can submit all your documents (except item 5) in a single compressed file with name **Student1LastName_Student2LastName_Phase2.zip**.

Both team members should submit Item 5 (contribution rating).

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).