Assignmen4 RDT App Programming

Based on assignment3, you will implement a simple file transfer application using UDP Socket APIs. And implement RDT 3.0 at the application layer. Functions include but not limited to:

1. Implement unreliable file transfer
    1) transfer a large file to a remote host using UDP Socket APIs.
    2) record ~~the number of error bits,~~ and the total delay of transmission from the first bit sending out to the last bit received.
2. Implement reliable file transfer using RDT 3.0 protocol at the application layer, based on the implementation of step 1. Compare ~~the number of error bits~~, and the total delay of transmission. Remember RDT3.0 uses stop-and-wait, so the delay will be much larger than that of pure UDP. ~~(optional)To improve the performance, sliding window can be used to replace stop-and-wait.~~
    a. To set the value of the timer appropriately, you need to measure the round-trip delay between the client and server frequently and adapt the value accordingly. You can refer to TCP dynamic timer algorithm.
    b. Acknowledgements are cumulative rather than selective. Remember that like TCP, you acknowledge the next sequence number you are expecting to receive, which is 1 more than the largest in-order sequence number you have received.
    c. You can retry packets infinitely many times, and should make sure you retry at least FIVE times, after which, if you want, the client can terminate the connection with an error.
    ~~d. (optional)As reliable transport is inherently a stateful protocol, your transport layer should handle simple connection management (support multiple simultaneous connections).~~
3. Implement file transfer using TCP Socket APIs, compare the number of error bits, and the total delay of transmission (no need implement RDT, it's done by TCP).
4. Your solution MUST handle large and binary (non-ASCII) files correctly.
5. UI design is very flexible. Please concentrate on the RDT protocol implementation instead of time-consuming UI.

Language recommended: Java, C/C++ or Python.

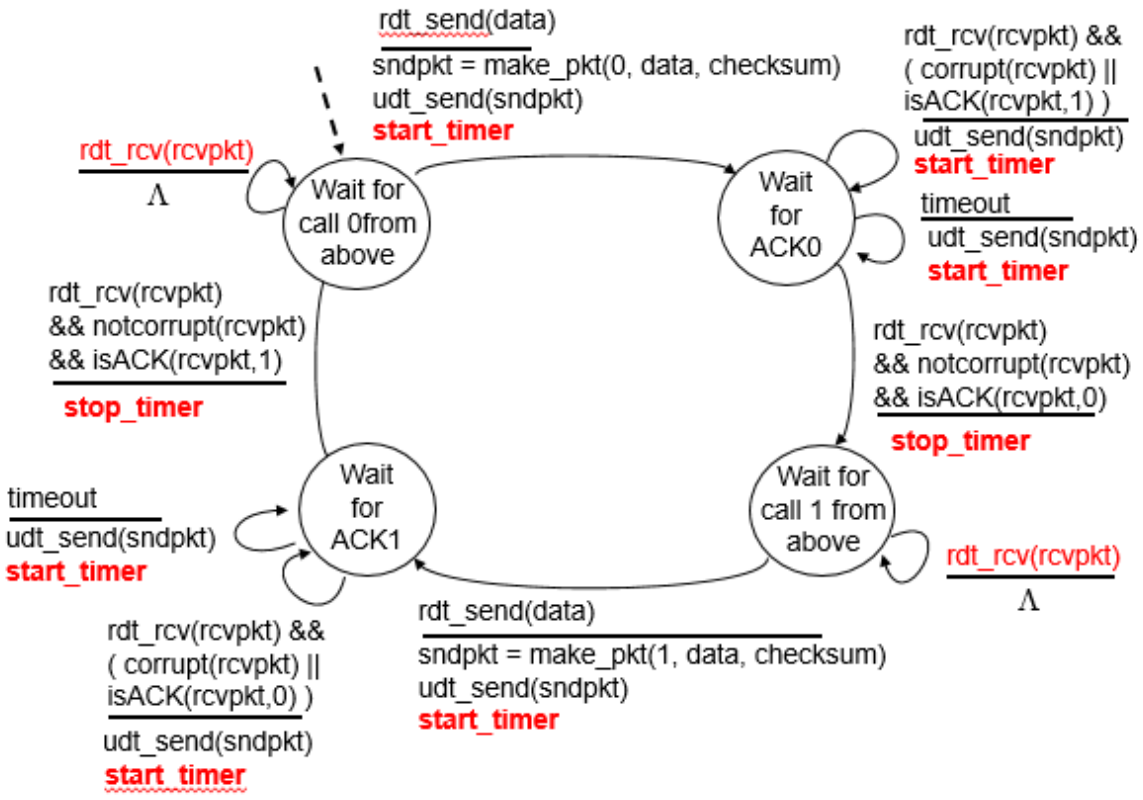
Note:

1. For RDT 3.0 protocol, please refer to textbook section 3.4.1 where RDT3.0 Sender FSM is depicted by Figure3.15, and RDT3.0 receiver FSM is same as RDT2.2 in Figure 3.14.
2. For C/C++ programmers, the Winsock tutorial can be found at: http://msdn.microsoft.com/en-us/library/windows/desktop/ms740673(v=vs.85).aspx
3. For java programmers, the java Socket API tutorial can be found at: http://docs.oracle.com/javase/7/docs/api/java/net/Socket.html

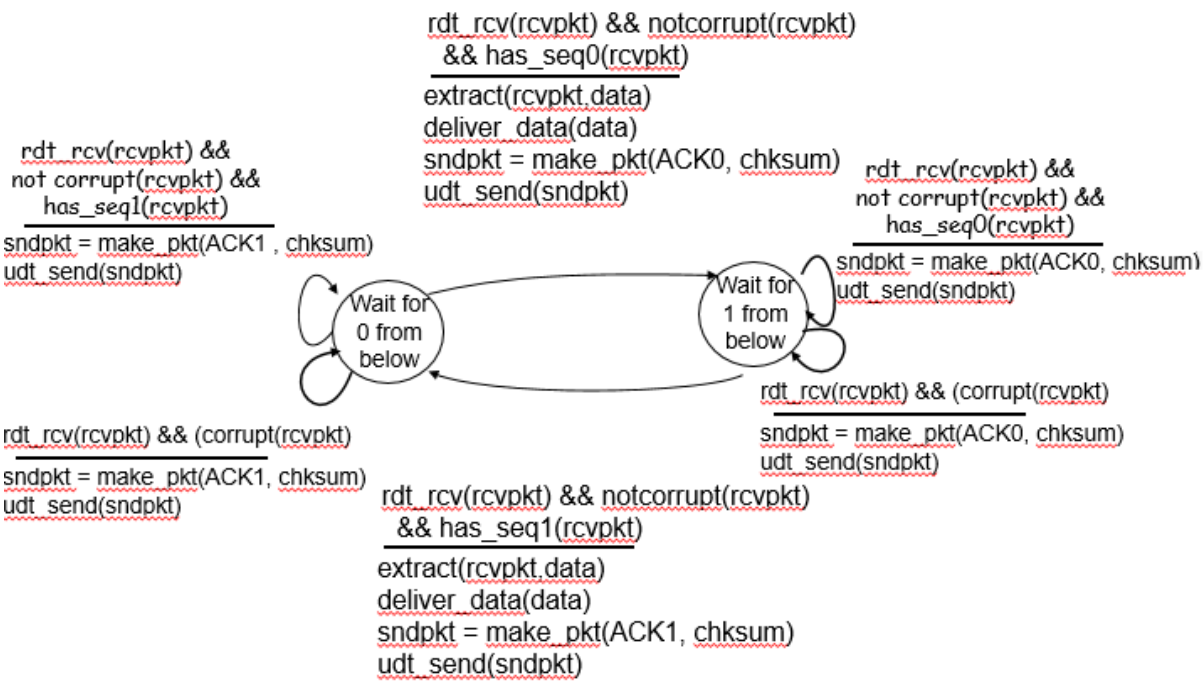
Submission:

1. Please submit to ftp://lpshen:public@public.sjtu.edu.cn/upload/assign4
2. Due date: 24:00 26 April. 2017
3. Filename: assign4-xxxxxxxxxx.zip/rar(xxxxxxxxxx is your student ID) including source codes, compiled runnable files and a simple report.
4. The simple readme (Keep it short! It should be a 1 or 2 page document) includes:
   - ✓ How to run the application.
   - ✓ Describe high-level structure of your RDT code.
   - ✓ Describe the tests you performed on your code to evaluate functionality and robustness.
   - ✓ Problems and experiences (troubleshooting experiences, comments and suggestion for the project or lectures are welcome)
5. Any questions please send email to TA Mr. Wang wqsdot7public@163.com

RDT3.0 sender FSM:



RDT3.0 receiver FSM:


rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
&& has_seq0(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
sndpkt = make_pkt(ACK0, chksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) &&
not corrupt(rcvpkt) &&
has_seq1(rcvpkt)
sndpkt = make_pkt(ACK1 , chksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) &&
not corrupt(rcvpkt) &&
has_seq0(rcvpkt)
sndpkt = make_pkt(ACK0, chksum)
udt_send(sndpkt)
Wait for
0 from
below
Wait for
1 from
below
rdt_rcv(rcvpkt) && (corrupt(rcvpkt)
sndpkt = make_pkt(ACK0, chksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && (corrupt(rcvpkt)
sndpkt = make_pkt(ACK1, chksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
&& has_seq1(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
sndpkt = make_pkt(ACK1, chksum)
udt_send(sndpkt)