CS570 Fall 2007 Programming Assignment 3

Implement the packet-oriented file transfer protocol described below. Due 19 November 2007.

DELIVERABLES:

- Demonstration on two of the Windows PCs in Tech Hall N329. Your demonstration will show that your program can correctly transfer an instructor-provided file from one PC to the other, including simulated dropped packets.
- Complete design documentation and user's guide.
- CD copy of source, documentation, and complied code (including any support files needed to execute the program)
- Printed copy of source and documentation
- Printed copy of logs for a sample run (including error handling)

PACKET TYPES AND FORMATS:

- Data packet. Fields:
  - Start marker ('$$$$START$$$$')
  - Sequence number (range 0-1024)
  - Payload (0-512 ACSII upper and lower-case letters, numerals, punctuation)
  - End marker ('$$$$END$$$$')
- ACK packet. Fields:
  - Start marker ('$$$$ACK$$$$')
  - Sequence number of packet being ACKed
  - End Marker ('$$$$END$$$$')

PROTOCOL DESCRIPTION:

The sending part of the protocol reads blocks of data from the file "protodata.txt". The data contained in the file consists of ASCII text (upper and lower-case letters, numerals, punctuation). The data in the file has been has been divided by a higher-level protocol into blocks, one per packet to be sent. The beginning of a block is marked with '****STARTOFBLOCK****' and the end of a block is marked with '****ENDOFBLOCK****'. A block may contain any number of characters from 1 to 512. The file may contain any number of blocks, from 1 to 1024.

Packets are numbered sequentially in the order transmitted starting from 1. Re-transmitted packets have the same sequence number as the original packet.

The sending protocol is:

1. read next data block from protodata.txt. If end of file, stop.
2. stuff bytes as needed
3. Assign sequence number
4. build packet
5. transmit packet or "drop" packet (see Special Instructions)
6. start 5-second timer

7. If ACK received before timer times out, go to 1
8. If timer times out, go to 5

The receiving protocol is:

1. receive packet
2. resolve byte stuffing as needed
3. generate and send ACK

SPECIAL INSTRUCTIONS:

The user will be able to cause the sender to drop a packet, simulating loss of a packet in transit.

You may assume that the ASCII sequence '****' is reserved and does not occur in the data part of the block. However, the sequence '$$$$' (which is a control sequence that is used in packet format) may occur in the data and you should use byte stuffing to prevent that sequence from being transmitted as data (the original text must be restored on the receiving end).

You are not required to implement the bit-level structure of a packet. You are not required to implement bi-directional transfer for this assignment.

The user will be able to make the protocol work in a "single-block" mode, where one block is processed and sent (or dropped) and ACKed, then the sending-side waits for a user input before processing the next block.

You will generate time-stamped send and receive logs that record the major steps taken by the protocol. The log will be saved for later printout and will also be displayed in real-time on the screens.

- On the sending side, the log will record:
  - User control actions (sending side)
  - Sequence number and data contents of each packet
  - Simulated dropped packets
  - Byte stuffing actions
  - ACK received with sequence number
  - Timeouts
- On the receiving side, the log will record:
  - User control actions (receiving side)
  - Sequence number of received packet
  - Received data (after packet processing)
  - ACK sent with sequence number
  - receipt of an out-of-order packet
  - Byte stuffing actions