XSFTP

(eXtremely Simple File Transfer Protocol)

Purpose

We intend to build a reliable network file transfer application, using a protocol which is built over UDP in the application layer.

Data Structure Format

Types of Packets and Their Structures

Each packet has its own Opcode Number. Other details are added as necessary according to the packet specification.

Connection Request Packet

A client must initially request the Server for a file. It sends a packet to the transport layer in the below given format.

1 byte	Variable (510 bytes maximum)	1 byte (NULL char)
OPCODE=1	Requested Filename	0

Connection Assign Packet

The Server assigns Sequence Number and Receive Window Size.

1 byte	4 bytes	4 bytes
OPCODE=2	Sequence Number	Receive Window

Data Packet (always sent by server only)

1 byte	4 bytes	Variable (507 bytes maximum)
OPCODE=3	Sequence Number	File Data

Acknowledgment Packet

1 byte	4 bytes
OPCODE=4	Sequence Number

Error Packet

1 byte	4 bytes
OPCODE=5	Error Code

XSFTP in Action

Initial Connection protocol

As the Protocol follows a Client-Server paradigm, the client first must send a connection request to the server, which must include the required data for a file transfer. This is done by the client sending a ‘connection request packet’.
Once the server receives the packet, it can do one of the following things:

- Server accepts the request and sends a ‘connection assign packet’ to the client.
- Server is unable to process the request or wants to reject it. This prompts the server to send an ‘error packet’ citing the reason it is unable to process the request such as RequestRejected, FileNotFound.

The client now depending on the received packet chooses to do one of the following:

- If the client receives a client ‘connection assign packet’ it will acknowledge it with an ‘acknowledge packet’ containing the sequence number received from the server.
- If the client receives an error message, it sends an ‘acknowledge packet’ containing the error code it received and closes the connection.
- If the client times out, then it is assumed the server has rejected the request and the client closes the connection.

Now considering no error packets are received by the client, the connection is successfully established. It then waits for ‘data packets’ which contain the file data requested. The server sends the first data packet after it receives an acknowledgment for the ‘connection assign packet’.
After a successful connection, from the first data packet sent by the server, our protocol adheres to the Selective Repeat Protocol. (Sender is Server; Receiver is Client)

Overview of XSFTP

Sequence numbers: Sequence of packet numbers in the buffer on both sender and receiver ends of the protocol. Using this sequence, we can send the acknowledgement from the receiver end to the sender end on whether the packet is received at its side.

Acknowledgments - Whenever the receiver receives an uncorrupted data packet it sends an acknowledgment to the sender. When a sender receives an ACK, it marks that packet as sent, provided the packet is in its sliding window.

Negative acknowledgments - Our protocol does not have negative acknowledgments. Whenever the received data is corrupted, the receiver does not send any ACK and causes a timeout on the sender's side, which emulates a NACK.

Retransmissions - The sender selectively retransmits the data whenever an ACK is not received within the timeout. In this protocol, the sender has different timers for different sequence numbers within a window. After the timeout, only that packet is retransmitted to the receiver.

Window size- Window size of sender and receiver's side is the same and should not be more than half of the sequence number space.

Ordering of packets:- Our protocol ensures that the packets are ordered according to the sequence number. Though the window size of the receiver is greater than 1, ordering of packets is ensured because the receiver slides the window and sends the packets to the upper layer only when the received packet matches the base sequence of its window.

Scenarios

1. Data packet lost :- Sender never receives an ACK from the receiver and retransmits that packet after a timeout.
2. Data packet corrupted :- If the received packet is corrupted, the receiver does nothing, and therefore a timeout happens at the sender side (no ACK), and the packet is retransmitted.
3. ACK lost :- Sender never receives an ACK from the receiver and retransmits that packet after a timeout
4. ACK corrupted :- Sender checks for a corrupted ACK and does nothing. Hence a timeout occurs for that packet and is retransmitted.

Connection Termination

Normal Termination

The end of transfer is marked by a Data packet that contains a packet size of less than 512 bytes. The connection is terminated by the client in this situation. The server terminates the connection after an acknowledgment is received for this packet from the client. *However*, the client is encouraged to wait for a while after sending the final acknowledgement in order to retransmit the final ACK if it had been lost. The client will know that the ACK is lost if the server sends the final packet again. In such a case the client receives a duplicate final packet, for which the client ACKs again. For this to happen the client must not terminate its connection for a set timeout for which it can still receive packets even after sending the final ACK.

Premature termination

If either the client or the server encounters an error, it sends an error packet to the recipient. This is just a courtesy, no ACKs are expected or are sent for an Error packet. Hence if an error packet is lost, there will be no retransmission. Hence timeouts should also be taken into account to imply an error.

Error Codes

ERROR CODE	ERROR
0 (0000)	Not Defined
1 (0001)	File not found
2 (0010)	RequestRejectedByServer
3 (0011)	Access violation
4 (0100)	Incorrectly formed packet