IK2213 - Network Services and Internet-based Applications

Assignment 4 - Reliable UDP

Ganapathy Raman Madanagopal, Tien Thanh Bui

1. Introduction

In this project, we have developed a library for Reliable UDP, which enables the application using it to send multiple files to multiple destinations. The library inserts a RUDP header on the payload of normal UDP datagram by which a reliable transmission can be achieved. It does the flow control using Go-Back-N ARQ sliding window technique same as in TCP, but in contrast RUDP doesn't provide congestion control, error recovery and other TCP capabilities. RUDP has its application in the scenarios where user do not want to add more overhead to the data (as like in TCP header of 20 bytes), but requires overall reliable data transfer.

2. Problems & Our Solutions

The following is the Reliable UDP header of 8 bytes long:

```
0               7               15              23              31
+---------------+---------------+---------------+---------------+
|            Version            |             Type              |
+---------------+---------------+---------------+---------------+
|                        Sequence Number                        |
+---------------+---------------+---------------+---------------+
```

Fields:

1. VERSION - (8 bits) - Denotes the version of RDUP and the current version is 1.

2. TYPE - (8 bits) - Denotes the type of payload

    * DATA = 1

    * ACK = 2

    * SYN = 4

    * FIN = 5

3. Sequence Number - (32 bits) - Unique number for each segment. Used for identification purpose.

Even though RUDP act as a common library for both sending and receiving application, the implementation of RUDP has been fine tuned to adapt the requirements. The RUDP protocol implementation and its functionality can be broadly viewed from two perspectives, from sender side and other hand from receiver side.

2.1. Sender Side

The application above the RUDP, passes the entire data to RUDP as chunks. For N receivers/client and for the M specified files, the application "vs_send" passes N copies of M files data to the RUDP library. Initially the application sends the "BEGIN" (Type = 1) packet which contains the filename terminated with a null string. On receiving the begin packet, we create a link and start storing the data (Type = 2) on the link list as the VSFTP application sends on till the "END" (Type = 3) is sent. For a each file a new linked list is created and the data chunks are added to the list. On reception of all chunks of a file, RUDP start to send the file to the receiver.

The sender starts the transmission of a file by sending a SYN packet to the receiver.

Now a randomly created sequence number is added to the RUDP header on the SYN packet and using the generated number as base sequence number the sequence number is incremented for each packet. The sequence number gets incremented until the FIN packet is sent. For each file a new and for each sender a new sequence number is generated and used. Also the sequence number are mapped to the data chunk index. Initially RUDP will send three segments. Based on the reception of Acknowledgment from the receiver, the window slides on. If the sender doesn't receive an ACK within the specified duration (two seconds), then the event handler generates the event "Timeout" is generated and the chunks/segments after the last acknowledged segment are retransmitted. Maximum of five retransmission are allowed. After sending all the files to a particular receiver, finally "FIN" packet is send out the receiver on reception of "ACK" for the FIN, the RUDP socket is closed for the particular client. Same process is repeated for the all the clients and finally, the data stored in the linked list are freed.

2.2. Receiving Side

The function of RUDP on receiving side is almost same as the sender side, with major difference in handling the sequence number and the event generation. Receiver does not need to regenerate a sequence number or to maintain a timer for processing timeout signals. On reception of SYN, it needs to create an Acknowledgment with the sequence number equals to received sequence number + 1. This is the next expected sequence number from the sender and it is stored for future comparison purpose. In the same manner, the expected sequence number increases till the FIN packet is received for each session. No event handling mechanism is required on receiver end. Also if RUDP receives the old data i.e. the data with old sequence number which is different from the expected sequence number or receives data in out of order, then the data segments which are received in out of order are discarded, and send the ack only to last correct segment which is received. All the received data (BEGIN, DATA, END) are passed to the application "vs_rec". Finally on the reception of FIN an acknowledgment to FIN is send out.

3. Discussion & Conclusion

Our RUDP service performs the full functionality of sliding window and the can detect and respond to the EVENTS quite well. However, it is limited by the fact that currently our RUDP supports only four types (SYN, ACK, DATA, FIN). Support for other common types such as RST, NUL, EACK, CHK,TCS and handling of checksum hasn't been implemented. We will try to improve it in the future. However, although only basic functionalities were developed, this project has given us a solid background in the areas of Socket Programming, Data Structures, Flow control mechanisms.