**COSC 5328\_02\_1**

**Computer Networking**

**Fall - 2015**

**Project: Reliable Communication over Unreliable Channel**

**Date: 11/25/2015**

**Project Team:**

**Adithya Narayanan (L20375088)**

**Rahul Pandya (L20355202)**

**Harsh Bhavsar (L20373507)**

**Pavan Rajashekhar Bantu (L20375845)**

**Index:**

|  |  |  |
| --- | --- | --- |
| **1.** | **Introduction** | **1** |
|  | a. what is the purpose of project? | 1 |
| b. principles which are being studied? | 1 |
| c. Summary of how you designed the project? | 1 |
| **2**. | **Background** | **2** |
|  | a. Relevant information regarding transport layer and reliability in the internet? | 2 |
| **3.** | **Software Design** | **3** |
|  | a. Diagram (flowchart , Dependency graph with directed edges) | 3 |
| b. Challenges and how you resolved them? | 4 |
| **4.** | **Implementation** | **7** |
|  | a. Discuss coding and Algorithms, describe your new functions and what you did to link your libraries. | 7 |
| **Table 4.1** : Functionality | 10 |
| **5.** | **Tests and data collected** | **10** |
|  | **Table 5.1 :** Test Cases | 11 |
| **Figure 5.1 :** Probability Vs Average | 11 |
| Output screenshots | 12 |
| **6.** | **Brief Conclusion** | **13** |

**1.Introduction**

a. What is purpose of the project ?

* **Unreliable data transfer**: An unreliable channel is in which there is no guarantee of the data transfer. Packets may be lost, or may not be received in the same order in which they are transmitted.
* **Reliable data transfer**: Reliable data transfer ensures the data transfer of all packets and to enable the receiver to deliver the packets in order to its application layer.
* The main purpose of our project is to support reliable communication over unreliable channel.
* To perform this, we are implementing our own socket type, called Basic Reliable Protocol, which would guarantee the packets which are sent are delivered to the receiver without any data loss.

b. Principles which are being studied ?

* By understanding the concepts of UDP and RDT protocols, we're declaring the type of socket which is used to overcome the drawbacks of UDP. The parameters required to define the socket has been declared in the header file. This is implemented by creation of the link list data structure which references all the parameters to build the socket for establishing reliable data transfer.

c. Summary of how you designed the project?

* data structure for each module in the project. we are define data structure based on the data type that is both receiving data and transmission data. we also created structure which contains the parameters required for building the socket. In this structure we are using a datatype called index distinguishes the socket between each of the users we are also creating another structure for defining data packets and acknowledgement packets. We have declared rthread and sthread in the header file which makes use of this data structure. the rthread executed when the data is sent from user 1 to user 2 and its par alley over run by sthread when there is a drop in data packet for enabling retransmission of the lost packet to destination user. the dropping of packets is simulated by the drop function which is defying rsocket.c.

**2. Background**

a. Relevant information regarding transport layer and reliability in the internet?

* Transport layer provides for the transparent transfer of data for users, systems, and applications and reliable data transfer services to the upper levels.
* The transport layer controls the reliability of communications through flow control, segmentation, and error control.

**Various functions performed by Transport layer are :**

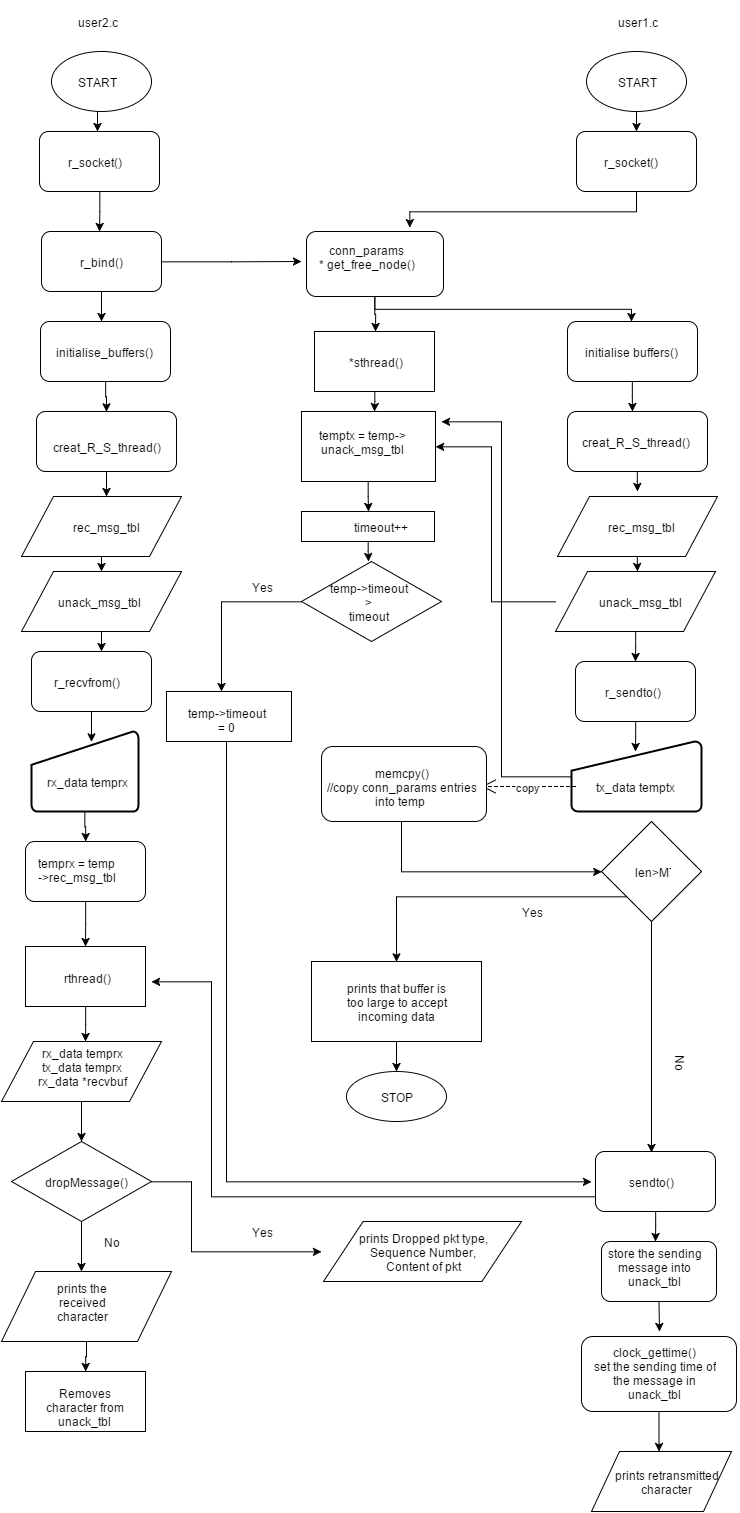
* This Layer is the first one which breaks the information data, supplied by Application layer in to smaller units called segments. It numbers every byte in the segment and maintains their accounting.
* This layer ensures that data must be received in the same sequence in which it was sent.
* This layer provides end-to-end delivery of data between hosts which may or may not belong to the same subnet.

**Reliability :**

* Reliability data transfer guarantees that all packets sent by the sender are delivered to the receiver in the same order, without any packet loss.
* It ensures accuracy and integrity of the data.
* It verifies that the data is transferred is received in the correct order and intact.

**3.Software Design**

a. Diagram (Flowchart)



b. Challenges and how you solved them?

The main issue that we faced while designing the BRP socket was to ensure that the threads don’t crash and lead to segmentation failure. Since our design was a concurrent server and client system, we protected our data from concurrent access by ensuring that the events were separately handled. This was enabled by the use of flags and thread synchronization. We designed the protocol such that the receiver end program takes data only if its buffer is filled in the time interval. This helped us to avoid data manipulation and memory leaks.

**4. Implementation**

a. discuss coding and algorithms, Describe your new functions and what you did to link your libraries.

**rsocket.h**

**#define SOCK\_BRP 1234**

* Definition of SOCK\_BRP type, used in 'r\_socket()' function to select BRP socket type.

**#define MAX\_BUF 50**

* We have set the maximum length of the buffer. This means that the buffer can’t accept data more than 50 characters or bytes.

**#define prob\_drop 0.35**

* We are setting the input value for the drop packet function for setting the probability value, p, for dropping the packets

**#define MTU 512**

* We are specifying the value for the Maximum Transmission Unit. This is the largest size packet or frame, specified in octets that can be sent in a packet- or frame-based network such as the Internet.

**#define datapkt 0x01**

* Sets the value for the packet type as 1 to denote that it is a data packet.

**#define ackpkt 0x02**

* Sets the value for the packet type as 1 to denote that it is an acknowledgement packet.

**Data structures:**

**struct rx\_data{**

**char pkt\_type;**

**char data;**

**int sequence\_num;**

**};**

* We have defined the parameters required for creating the received messages table that stores the data packets that are received by user2 or server under a structure named as rx\_data and have also been acknowledged back to user1 by user2.
* pkt type denoted whether it is a data packet or an acknowledgement packet.
* data represents the content of the packet and sequence\_num denotes the sequence number of the packet.
* This structure triggers the execution of rthread when rsend() or recvfrom() are called.

**struct tx\_data{**

**char pkt\_type;**

**char data;**

**int sequence\_num;**

**int timeout;**

**int lock;**

**struct tx\_data \*next;**

**};**

* We have defined the parameters required for defining the unacknowledged message table that stores the details of the packet are transmitted by user1 or the client to the server(user2) but these packets haven’t been acknowledged user1(server) to user2(client).
* pkt\_type, data and sequence\_num are the same paramters as used in rx\_data.
* Timeout denotes the time to live (TTL) that is specified in the header of the data packet.
* Lock denotes that the thread (process) is currently making use of this data packet if it set to 1. If lock =0 then it means that the packet is not being executed and is in waiting.
* This structure triggers the execution of sthread when rsend() or recvfrom() are called.

**struct conn\_params{**

**int sk;**

**struct sockaddr serverinfo;**

**pthread\_t Rth;**

**pthread\_t Sth;**

**int seq;**

**struct rx\_data \*rec\_msg\_tbl;**

**struct tx\_data \*unack\_msg\_tbl;**

**struct conn\_params \*next;**

**};**

* This structure defines the parameters that are used to create and bind the socket between each user.
* The parameter sockaddrserverinfo defines the following which are described in user1.c:
  + serverinfo.sin\_family = AF\_USPEC
    - denotes that the address is IPv4 or IPv6
  + serverinfo.sin\_port
    - denotes the port number in which the server is binding to initiate connection
* socketType
* creates object to reference the data stored in the received message table and

unacknowledged message table.

### **Table 4.1: Functionality.**

|  |  |
| --- | --- |
| **Functions** | **Description** |
| **r\_bind()** | Binds brp socket between user1 and user2 |
| **r\_recvfrom()** | calls the recvfrom() function in both the threads for receiving incoming data via socket into the buffers |
| **initialize\_buffers()** | Allocates memory for every data that is stored in the received message table and the unacknowledged message table |
| **r\_sendto()** | send data between the users through brp socket |

**5.Testing and collected data**

We take average value of 10 samples for every probability step.

Test string - **"The link layer is the second layer in the OSI model".**

Test string length = **48**.

* **Probability**: Probability of packet loss
* **Samples**: No of times the test was run with each p value
* **Total**: Average number of transmissions made to transmit the entire string
* **Average**: Rate of transmission (transmission per character)

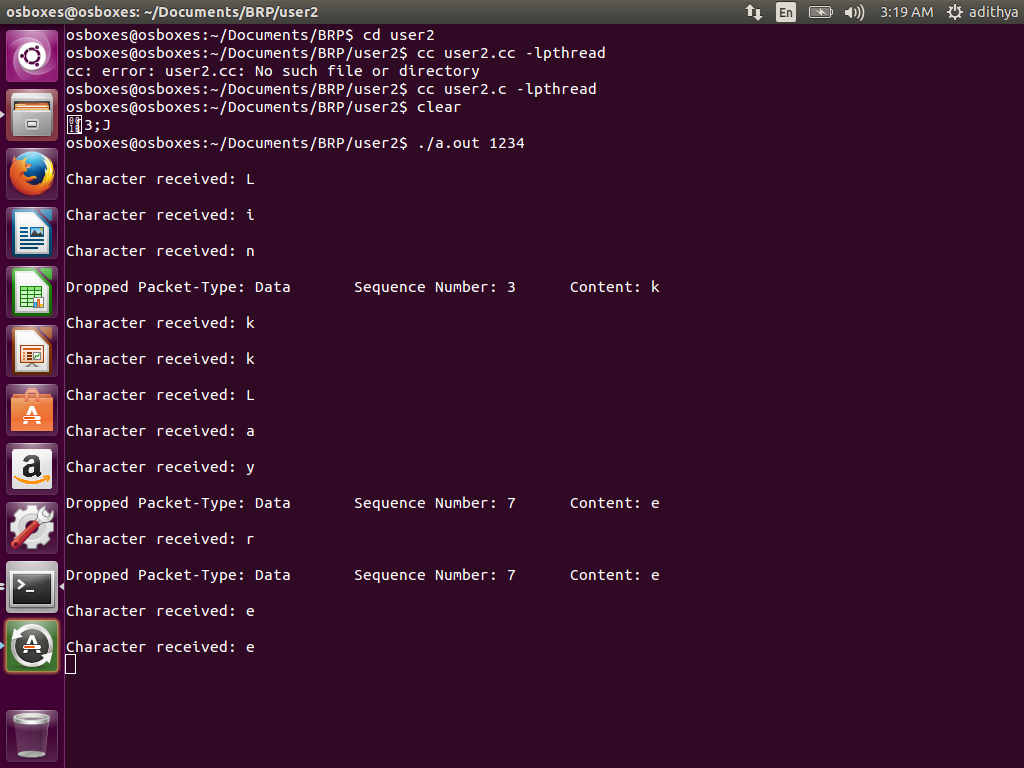
### Table 5.1: Test cases.

|  |  |  |  |
| --- | --- | --- | --- |
| Probability | Samples | Total | Average |
| 0.050000 | 5 | 49.2 | 1.0400 |
| 0.100000 | 5 | 49.2 | 1.0400 |
| 0.150000 | 5 | 53.4 | 1.1250 |
| 0.200000 | 5 | 54.6 | 1.1375 |
| 0.250000 | 5 | 63.0 | 1.3125 |
| 0.300000 | 5 | 89.6 | 1.8667 |
| 0.350000 | 5 | 99.4 | 1.0708 |
| 0.400000 | 5 | 103.5 | 1.1562 |
| 0.450000 | 5 | 104.6 | 1.1792 |
| 0.500000 | 5 | 106.4 | 2.2167 |

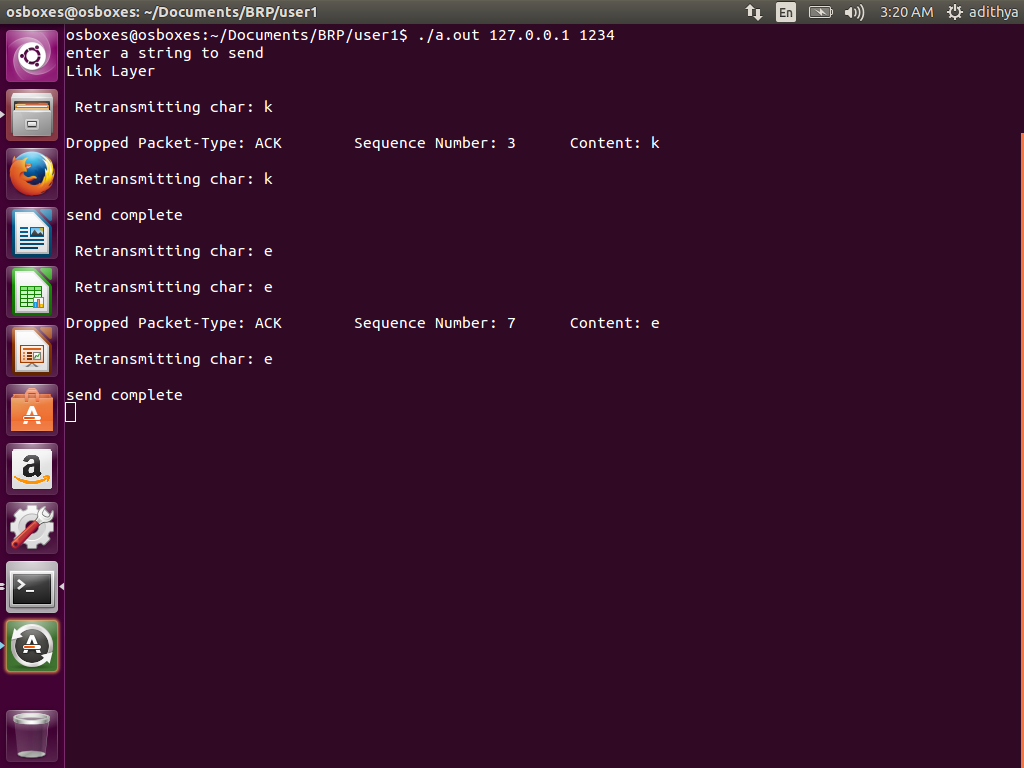
Figure : 5.1 Probability Vs Average

# Output Screenshots

**user2.c**



**user1.c**



**Conclusion**

As per the given requirements, we were able to design a reliable protocol using the basic concepts of UDP. We have studied about UDP and have understood how it becomes less reliable because of packet loss. Using link list data structures, we were able to design and implement two tables that stored information on unacknowledged data packets as well as received data.

Using certain concepts of rdt v2.0 (Reliable Data Transfer) protocol, we were able to overcome the drawbacks of UDP owing to good co-ordination and effective team work.