

**EE3204E Lab Assignment**

**Submitted by**

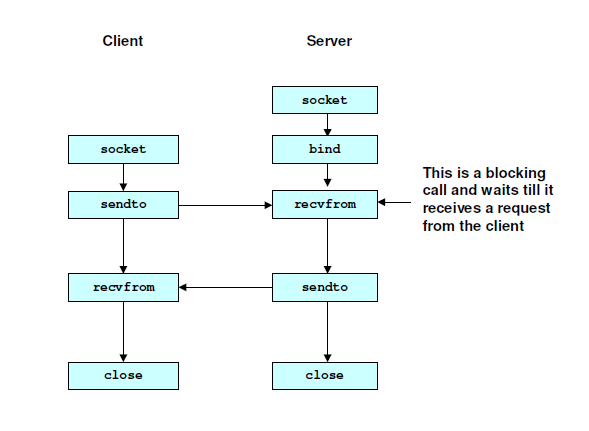
**Shourya Pankaj Moona**

**A0088589B**

**Problem**

Develop a UDP-based client-server socket program for transferring a large message. Here, the message transmitted from the client to server is read from a large file. The message is split into short data-units which are sent by using stop-and-wait flow control. Also, a data-unit sent could be damaged with some error probability. Verify if the file has been sent completely and correctly by comparing the received file with the original file. Measure the message transfer time and throughput for various sizes of data-units. Also, measure the performance for various error probabilities and also for the error-free scenario.

**Introduction**

The flow for a UDP-based client-server socket program can be visualized as:

1. **int socket(int family,int type,int proto);**

The socket() system call returns a socket descriptor (small integer) or -1 on error. It allocates resources needed for a communication endpoint - but it does not deal with endpoint addressing. The parameter family specifies the protocol family: (AF\_INET for Internet, PF\_INET for TCP/IP). ‘type’ specifies the type of service: SOCK\_STREAM or SOCK\_DGRAM and protocol specifies the specific protocol which is usually 0 (default).

1. **int bind( int sockfd, const struct sockaddr \*myaddr, int addrlen);**

The bind() system call is used to assign an address (specified by myaddr) to an existing socket and returns -1 on error. It binds the server to a port which is passed as a parameter.

Note: In this case, we do not use bind for client because the OS can assign it any available number.

1. **sendto()**

It transmits a message to another socket and returns -1 on error.

1. **recvfrom()**

It receives a message from another socket and returns -1 on error.

**Observations**

1. **Varying the Datalength and Frame Error Probability = 0**

When I increased datalength from 1000 bytes to 18000 bytes,

The throughput increased steeply at first, and marginally at higher values of datalength. This is because lesser packets were sent for larger datalengths.

The message transfer time decreased exponentially and then stabilized for larger datalengths.

Error Probability = 0

|  |  |  |
| --- | --- | --- |
| **Datalength**  **(bytes)** | **Throughput**  **(Kbytes/s)** | **Message Transfer**  **Time (ms)** |
| 1000 | 1545.68 | 38.68 |
| 2000 | 2405.00 | 25.00 |
| 3000 | 3474.32 | 17.23 |
| 4000 | 4454.85 | 14.00 |
| 5000 | 4874.98 | 12.27 |
| 6000 | 5228.03 | 11.44 |
| 7000 | 5583.96 | 10.71 |
| 8000 | 5884.56 | 10.16 |
| 9000 | 6331.99 | 9.52 |
| 10000 | 6813.24 | 8.80 |
| 11000 | 7010.00 | 8.53 |
| 12000 | 7323.09 | 8.16 |
| 13000 | 7360.95 | 8.12 |
| 14000 | 7357.75 | 8.13 |
| 15000 | 7589.00 | 7.88 |
| 16000 | 7621.80 | 7.84 |
| 16000 | 7621.797363 | 7.845 |
| 17000 | 7890.340332 | 7.578 |
| 18000 | 7860.260254 | 7.607 |

1. **Varying the Frame Error Probability and Datalength = 1000 bytes**

When I increased error probability from 0.0 to 0.9,

The throughput decreased and message transfer time increased because ‘bad’ packets were sent and good packets had to be retransmitted.

Datalength = 1000 bytes

|  |  |  |
| --- | --- | --- |
| **Error**  **Probability** | **Throughput**  **(Kbytes/s)** | **Message Transfer**  **Time (ms)** |
| 0 | 1545.68 | 38.68 |
| 0.1 | 1277.57 | 46.80 |
| 0.2 | 1185.40 | 50.44 |
| 0.3 | 1101.69 | 54.27 |
| 0.4 | 938.09 | 63.74 |
| 0.5 | 730.40 | 81.86 |
| 0.6 | 656.71 | 91.05 |
| 0.7 | 523.09 | 114.31 |
| 0.8 | 396.91 | 150.64 |
| 0.9 | 193.29 | 309.34 |