Santosh Kumar Dubey
Person# 5009 7059

Remote File Sharing System

Table of Contents

Santosh Kumar Dubey
Person# 5009 7059

Remote File Sharing System

Implementation :

The following section explains the implementation of the program.

The program can be run in two modes:
1. Server
2. Client

In each mode a set of commands are available for performing different actions, which can be viewed by executing the HELP command.

When run in server mode the process listens for incoming connections and registers the clients, it also sends the list of registered clients to each of the registered clients.

When run in client mode, the process first registers to the server, then connect to other clients (peers). Once connected, the clients can initiate file transfers between each other either using upload or download requests.

Various message types are used in the program for different types of communications. These are defined in proj1.h header file.

The implementation of some of the important functionality are explained below:

1. Client registration:

a) When the REGISTER command is executed on the client, the client connects to the server over a TCP socket, and sends its listening port number over the connection. The client uses the message type MSG_MYPORT to send this information.
b) The server, on receiving the connection and the port number, registers the client and sends the update list of registered client to all the currently registered clients. The server uses the message type MSG_PEER_LIST for this.
c) The message is formatted as the message type, followed by the number of peers, followed by IP-port pair of each peer.
d) Clients uses this list to connect to peers.

2. Peer connections:

a) When a CONECT command is executed on the client, the client connects to the peer over a TCP socket, and sends the message type MSG_CONNECT_REQUEST followed by its port number.
b) The peer receives the connection and adds the peer to its connected client list.
c) A client can have a maximum of 3 connections to peers – this includes both incoming and outgoing connections.

Santosh Kumar Dubey
Person# 5009 7059

Remote File Sharing System

3. File upload:

a) Once connected, a client can initiate a file upload to its peer.
b) When the UPLOAD command is executed on the client, the client sends a message of type MSG_UPLOAD_REQUEST to the peer.
c) The message contains the message type, followed by the file size, file name length and the file name in that order.
d) The peer on receiving the message, creates a file in its current directory with the given name. If something goes wrong in the process, a message with type MSG_UPLOAD_REJECT is sent back to the peer. If everything is OK, a message of type MSG_UPLOAD_ACCPET is sent back.
e) The originator client on receiving the MSG_UPLOAD_ACCEPT starts sending the file.
f) The file is read PACKET_SIZE bytes at a time and sent over to the peer. The peer receives a PACKET_SIZE chunk of data and writes to the file.
g) A peer can receive UPLOAD request from multiple peers and send file simultaneously.

4. File download:

a) Once connected, a client can request the peer for a file.
b) When a DOWNLOAD command is executed on a client, the client sends a message of type MSG_DOWNLOAD_REQUEST to the peer.
c) The message contains the message type, followed by the file name length and the file name in that order.
d) The peer on receiving the message, looks for the file in its current directory with the given name. If something goes wrong in the process like file not found or no accedd, a message with type MSG_DOWNLOAD_REJECT is sent back to the peer. If everything is OK, a message of type MSG_DOWNLOAD_ACCPET is sent back. The MSG_DOWNLOAD_ACCPET contains the file size too.
e) The originator client on receiving the MSG_DOWNLOAD_ACCEPT creates the file and starts receiving the data.
f) As above, the file is read/sent and received/written in PACKET_SIZE bytes chunk at a time.
g) A client can initiate multiple (upto 3) download requests at a time and receives them simultaneously.

5. Showing the available Peers (Connected) :

Once the client is connected to each other the available peers will be displayed by simply typing **"peers"** this command will show all the connected client and their respective port number. This command have been implemented to give user flexibility to check the available peers with their IP and port number.

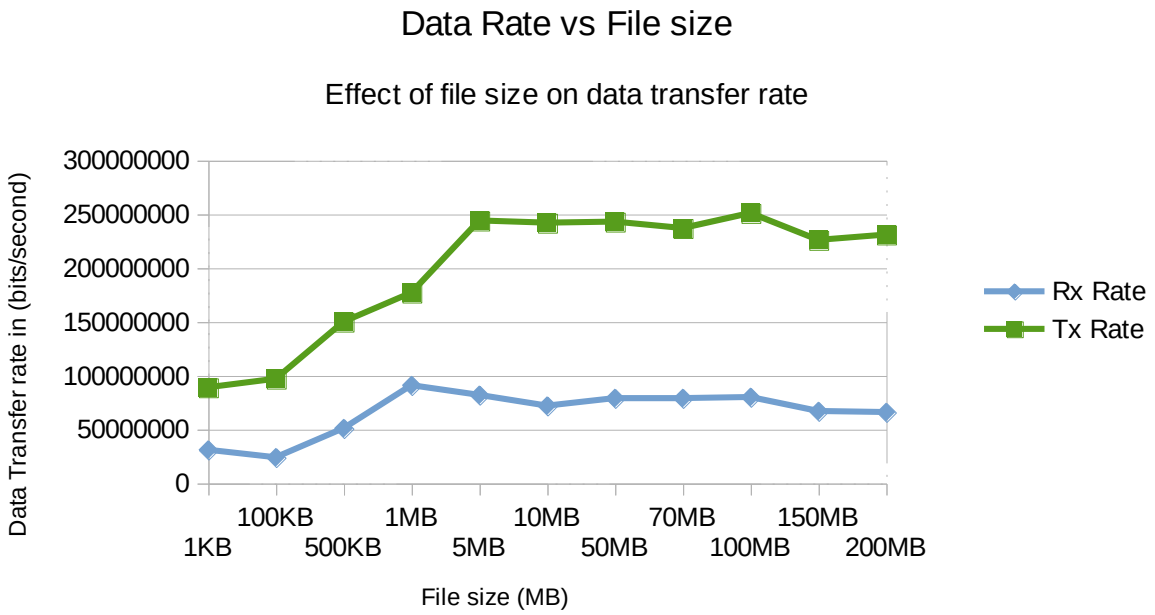
Santosh Kumar Dubey
Person# 5009 7059

Remote File Sharing System

Observation and Analysis:

4.1. Data rate vs. File Size:

Observation:



I, observe that as the size of the file increases, the rate of transfer increases, till the file size is around ~5M, and then the transfer rate more or less stabilizes, with very little fluctuations.

Analysis:

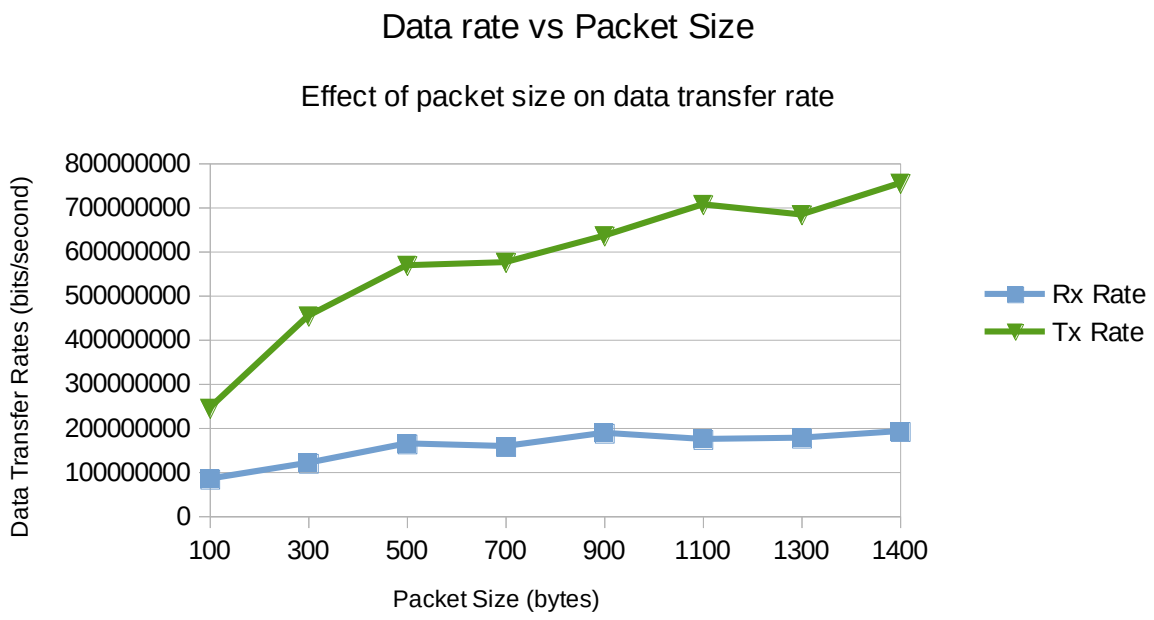
a) The observation can be largely attributed to disk I/O performance.
b) Since the packet size is constant, the effect of network latency or bandwidth is minimal on the data transfer rate.
c) The overall rate can be largely affected by the disk performance.
d) Most, if not all, disks uses a small cache called disk buffer, to speed up the disk I/O performance.
e) When block of data is written to the disk, it is actually stored in the disk buffer and the disk signals that the write is complete immediately after receiving the data. The data is actually written to the disk later.
f) This reduces the I/O latency in the process doing the write.
g) Similarly the disk may do some read-ahead and saves the data on the disk buffer, as a result the process requesting the data may receive it without much delay.
h) However, when the amount of data to be written/read increases, the disk buffer may become full. When the amount of data is more than the disk buffer size, the process no longer gets benefited from the buffer, and the disk performance becomes the bottle neck.
i) As a result of the above, I see that the transfer rate keeps increasing. At a certain point, when I hit the disk buffer limit, the transfer rate stops increasing as disk performance becomes a bottleneck.

Santosh Kumar Dubey
Person# 5009 7059

Remote File Sharing System

4.2. Data rate vs. Packet Size:

Observation:



I, observe that as the packet size increase, the data transfer rate also gradually increases.

Analysis:

Transferring each TCP segment has some overhead:

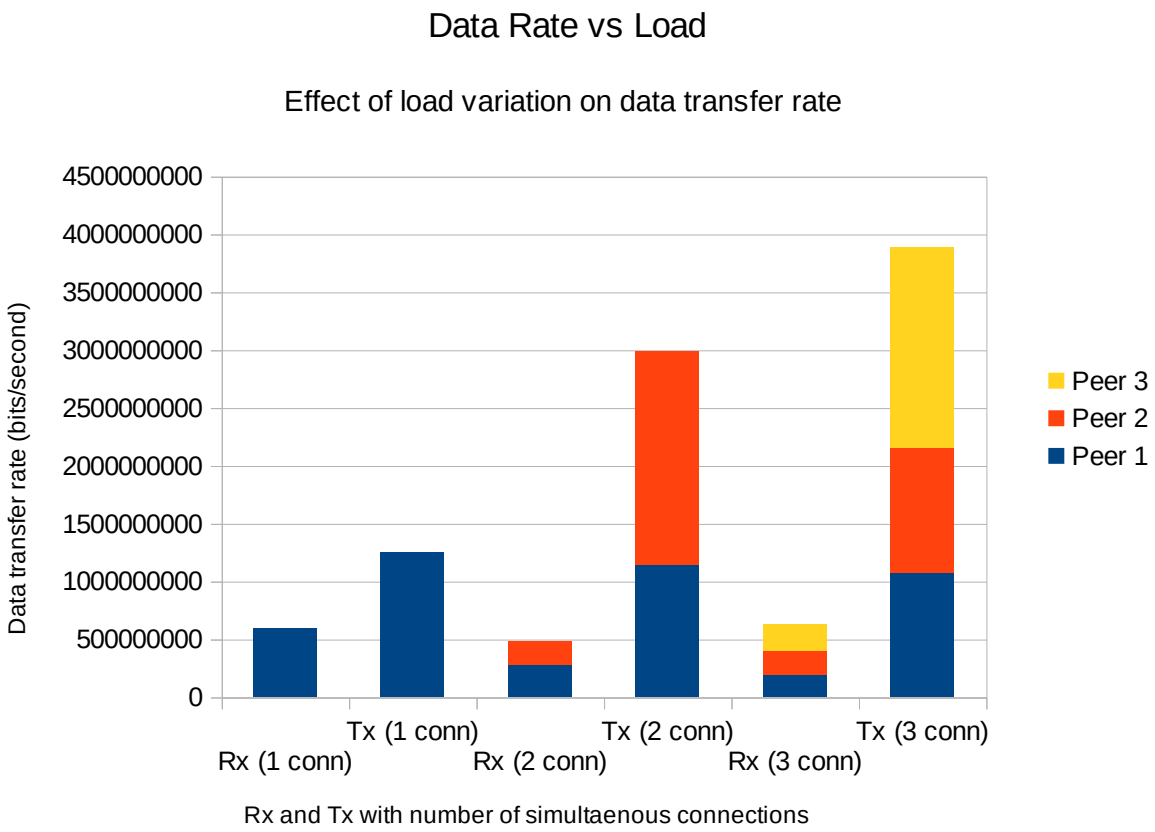
- Adding TCP headers to each segment increases the actual amount of data transferred on the wire
- Each TCP segment adds a small processing overhead and latency because of protocol processing at both ends like sending/waiting for ACK for the TCP segment.

a) As a result of the above overhead the larger each block of data is, the smaller the overhead becomes measured over the entire transfer.
Therefore, larger packet size results in higher throughput.

b) Here the file size being constant the disk performance does not have any impact on the overall performance.

Santosh Kumar Dubey
Person# 5009 7059

4.3. Date Rate vs. Load variation:



Observation:

a) This test was performed by once client receiving files simultaneously from 1, 2 or 3 other clients in each iteration respectively.
b) I, observe that for the client receiving the files simultaneously, the data transfer rate on individual connection decreases as the load increase, but the cumulative transfer rate remains more or less the same.
c) For the clients sending the files, the transfer rate remains almost same.

Analysis:

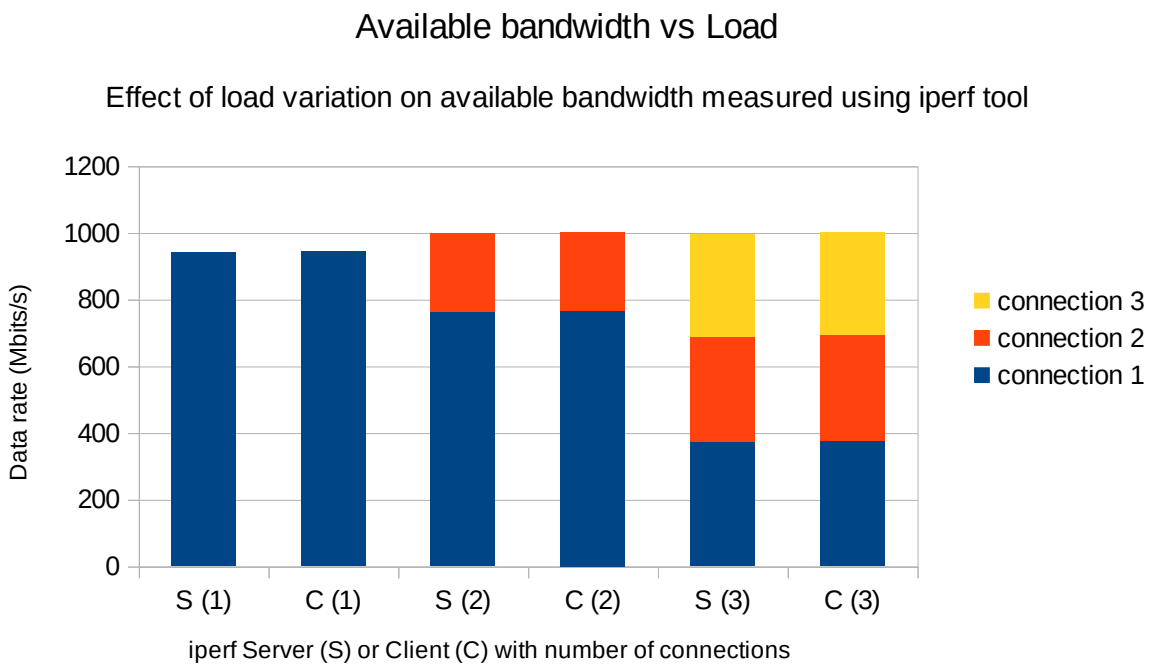
a) This results are expected.
b) In the client receiving the files, both the network performance and the disk I/O is divided among the connections. The resources being finite, the overall performance remains constant but gets divided among the various connections as the load increases.
c) For the clients sending the files, irrespective of how many connections the receiving client is part of, tries to send as much as possible and as fast as possible, hence the overall transfer rate remains constant.
d) I, would have expected the rate to go down a little because the receiving client must be

busy serving other connections and may delay the ACK. But probably because the performance bottleneck here is the disk I/O and not the network bandwidth, hence the network delay cause by the receiver would not impact the transmitter's performance.

Santosh Kumar Dubey
Person# 5009 7059

4.4. Bandwidth measurement using iperf :

Observation:



a) Although I would have expected that the results would be similar to that of the previous test, the results are slightly different.

b) On the server side (the iperf server) I see that the results are almost identical to that of the previous test – the individual throughput of each connection decreases but the cumulative throughput remains almost same.

c) On the client side (the iperf clients) I see the same – the individual throughput of each client reduces as the load increases while the sum of the bandwidth of all the clients remains the same. This is very different from what I observed in the previous test.

d) Another observation is that the maximum available bandwidth calculated by this test is more than achieved by the previous test.

Santosh Kumar Dubey
Person# 5009 7059

Remote File Sharing System

Analysis:

a) The major difference, and the cause of the different result, between this test and the previous one is that this test measures ONLY the network bandwidth – there is no disk I/O involved.

b) At the server, the total available bandwidth is divided in processing the simultaneous connections. So I see that individual throughput of each connection reduces but the overall throughput remains almost same.

c) Because the server is busy is processing all the connections simultaneously, it introduces a delay in sending TCP ACKs to the clients or data to the clients.

d) At the clients end, even though they would try to send as fast as possible, they couldn't because of the delay caused by the server's response. So the server's processing power becomes the bottleneck.

e) I also observed that the maximum throughput achieved by this test is more than that of the previous test, this is mainly because the iperf only measures network bandwidth and disk I/O is not involved, this leaves the process more time to send and receive data over the network, hence more throughput.

Santosh Kumar Dubey
Person# 5009 7059

References :

1) http://unixhelp.ed.ac.uk/CGI/man-cgi?inet_ntoa
2) http://www.beej.us/guide/bgnet/output/html/multipage/zindex.html
3) http://linux.die.net/man/3/getaddrinfo
4) http://msdn.microsoft.com/en-us/library/windows/desktop/ms738564%28v=vs.85%29.aspx
5) http://pubs.opengroup.org/onlinepubs/009695399/functions/inet_addr.html
6) http://www.gta.ufrj.br/ensino/eel878/sockets/inet_ntoaman.html