Multimedia Sharing Application

Advanced Computer Networks Project
CSN-503

Supervised by

Prof. Durga Toshniwal

Introduction

In the modern world, Data has overtook oil as the most valuable commodity.

Sharing data will help one connect with the people, who the sender wants to use it. People share content because they want to connect with other humans, while Businesses share content- both public/ advertising content to people and also confidential content to other Business Officials.

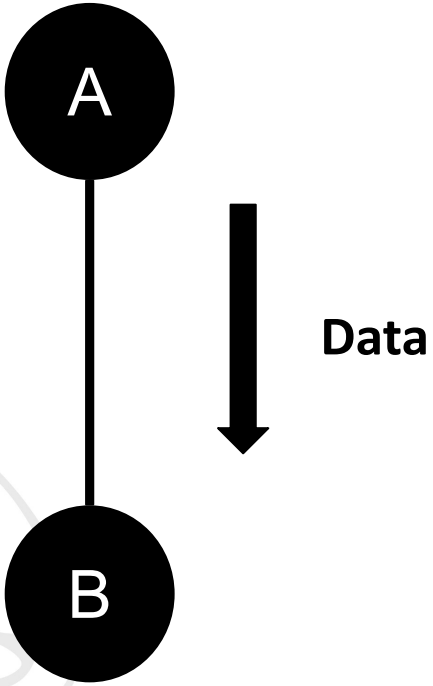
Few metrics like bandwidth or Maximum transfer sizes, Routing, Internet Protocols used, etc, requires concern on, and Security measures are to be followed to perform a Data Transfer between two hosts in a network. Cost, performance and time, all act as important factors in the Data Transfer, and optimizing them is considered to be a main goal in Computer Networking.

Problem Statement

Multimedia Sharing Application:

Develop an application which has the following features:

a) Share large multimedia files between two nodes over a private network.
b) Use multithreading to optimize the large files transfer.
c) Show the results of multithreading over a single threaded transfer.



Types of Network Models

Client-Server Network Model

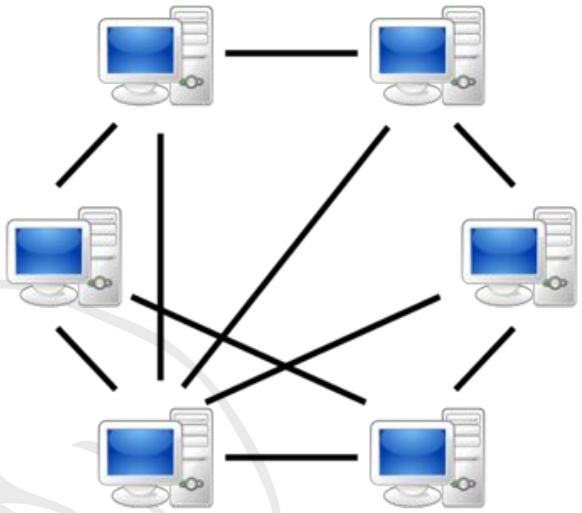
- There exists a powerful machine (server) which listens to requests in a constant loop from less powerful machines(clients).
- Server does the processing of the request, logic, assembling data and returns the result to the client machine.
- Various protocols e.g. FTP, SMTP, HTTP, use a client server model, even the WEB we use today.

Peer-to-Peer Network Model

- In peer to peer network there is a decentralized model where each node acts as a peer.
- There is no dedicated server and each peer can act as both a server and a client.
- Each peer can make request to other peers and at the same time can respond to request from other peers.

Types of Network Models

Client-Servers Network Model



P2P Network Model

Why not peer-to-peer network model?

Advantages:

[+] Good performance. //more bandwidth utilization with more nodes

[+] Low cost.

[+] Fault tolerant

Disadvantages:

[-] Security Threats

- Distribution of virus.
- TCP port issues.
- Risks of downloading illegal content.
- Vulnerabilities in peer to peer software.

[-] Files and folders cannot be centrally backed up. //not an industry standard.

Methodology

Client Side

On Client Side, the client/sender sends the data to the respective recipient.

Since to transfer the data, multithreading is being used, the data first needs to be divided into smaller chunks, in order to distribute it to each thread.

This function is performed as following:

- We can send the data by dividing it on the basis of allocating equal byte-codes [for example 4096 or 4KB] per thread.
- Considering the data to be a text or doc file, the file can be divided into smaller blocks of data based on the number of lines present per file.

Data Transfer- Multithreading

For the transmission of data from client to server, we have explored different ways of protocols, different ways to use these protocols, also explored which form (data form) the data needs to be for optimal and error free transmission.

TCP vs UDP

We have explored UDP, TCP. In using UDP, which being a connectionless protocol, the data transfer from one thread at client to another at server was complex, causing a lot of overhead and was quite Unreliable.

So, we decided to use TCP as it was reliable and connection oriented protocol.

Data Transfer Implementation

For implementing TCP data transfer we decided to use Socket in Python.

- The thread on client side first makes connection with thread on server side with help of load balancer at server.
- This thread takes the data it receives and sends it out in format of “utf-8”

Server Side

On Server side, We need recipient to receive data sent by client, we aim to allocate a separate thread on server side to receive data from each thread on client side, For this action we need a load balancer.

The main program on server side acts as a load balancer, it actively listens for requests to connect from clients. When it receives a request from thread on client side, main program creates a new thread that communicates with the client side thread.

Each thread on server side receives data and stores it separately and send acknowledgment upon receiving successfully.

Server Side- Implementation

Upon confirmation of completion of data transfer from client side, Server side creates a new thread (final) that combines data stored separately **in order** and creates the final file.

- We have created a method for thread that receives data
- For storing the data, we tried storing it in a buffer and dividing data into equal byte-code chunks. But output has undergone memory leakage, i.e., upon combining data fragments, we are getting a damaged file [missing data].
- Therefore, we have implemented in a way that each thread stores it as a file, and in the end, the files are adjoined to form final file. This method had minimum/no data loss.

Observed Results

Let us compare the transmission time taken and file size in both single-threaded and multi-threaded scenarios.

File Size	Single Thread	Multi Thread
1 MB	0.3s	1.5s
10 MB	0.9s	2.1s
100MB	8s	7.8s
1GB	122s	26s

Interpretation

- By observing the table in before slide we can observe that multi threading is leading to less transfer time for large file [eg: 100MB, 1GB], while more time comparatively for small files [eg: 1MB, 10MB].

- This is because the preprocessing that we need for forming threads, causes more overhead, when compared to that of single threaded scenario.

- So we can conclude that using multithreading is useful for large files

- We can also estimate the size, above which multithreading is useful.

DEMO

What have we achieved

- ★ Shared text and doc files between two nodes in a private network.
- ★ The data in the large files is divided into smaller chunks of data fragments, and is sent for data transfer at the Client side.
- ★ Through Multi-threading, data transfer of large files is made easy and time is efficiently utilised.
- ★ Received data fragments are collected and combined in order at the Server Side.

Future Work

- Application capable of using Single thread for small files and Multi thread for large files through real-time analysis.
- Application capable of sharing other multimedia files such as audio, video and image files.
- Including Error detecting methods such as using Checksum and possible Error Correcting methods.
- Rather than just Data Transfer, the application can enable clients to
  - List all the files in their Client database
  - Delete files in their Server database
- Provide a good User Interface

Thank You