**VALLEY VIEW UNIVERSITY**

**DEPARTMENT OF COMPUTING SCIENCE AND ENGINEERING**

**REPORT:**

**BUILDING A BIT-TORRENT CLIENT FOR A LOCAL AREA NETWORK**

**SUBMITTED IN PARTIAL FULFILMENT OF THE COURSE INFT430 DISTRIBUTED COMPUTING**

**BY GROUP 3:**

**JUSTICE MARKWEI (LEADER) - 215IT02003338**

**FRANCIS AIKINS HOTWARDS - 214CS01002026**

**TIMBAU AYINNAH SAMUEL - 215CS02003285**

**MASTURA LUKMAN - 215IT02003321**

**SAMIRA ABUKARI - 215IT02003374**

**TO:**

**J. ABANDOH-SAM**

**13TH APRIL 2018**

**INTRODUCTION**

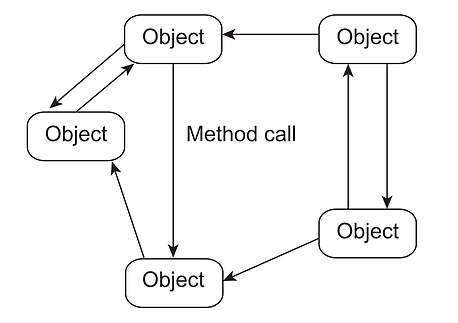
This report is to explain what architecture we used in building our bittorrent client, how it fulfills the distributed system qualities, what type of synchronization it uses and explanation of codes that was committed by individual members of the group.

**WHAT IS BITTORRENT?**

The bittorrent protocol was designed to allow heavy files to be shared across the network or internet in a decentralized manner, in small pieces by all computers or peers who have the same file. This is meant to increase download and upload speed of the file. All peers with the same file share a piece of that file to the interested node on the network until the full file is complete.

**ARCHITECTURE STYLE: OBJECT BASED ARCHITECTURE**

I would be ok to use a centralized approach; client server approach. But because of its disadvantage of all peers losing access to files when the server goes down, we will go with a decentralized approach instead where all peers are the server. We are using the **Object-based architecture**. In this architecture, the tracker only gives you the information about the peers who have your file, and then after you connect to those peers, you will download the files from them. The advantage of this is that because it is decentralized, even when one peers goes off, the other peers are still alive to serve you your file until you finish downloading the file. Below is an illustration of the object-based architecture:



As shown in the above image, communication between object happen as method invocations. These are generally called Remote Procedure Calls (RPC). Some popular examples are Java RMI, Web Services and REST API Calls. This has the following properties. This architecture style is less structured.

* Component = object
* connector = RPC or RMI

When decoupling these processes in space, people wanted the components to be anonymous and replaceable. And the synchronization process needed to be asynchronous.

**PROGRAMMING LANGUAGE CHOICE**

We chose to use python because it is easy to understand and code in. But most importantly because of its ‘asyncio’ library feature that it has, which will help us build a distributed client what can both input and output asynchronously.

**KEY CHARACTERISTICS OF DISTRIBUTED SYSTEMS THAT IT FULFILS**

* **Synchronization**

We used the asynchronous synchronization process because bittorrent is about uploading and download files at the same time. Therefore, the clients or peers need to be able to be downloading and uploading files at the same time. In python we imported the library asyncio to help us do that

* **Concurrency**

This involves multiprogramming and multi-processing. We made sure there can be parallel distribution between all peers and server. Many users can use the same resources and application interactions. Also many servers can respond to client requests.

* **Fault Tolerance**

Because computers fail we need hardware redundancy and software recovery. Therefore, by using object-based architecture, there is increase in availability in services. When one peer fails, the others are still there to serve.

* **Processes**

The processes we used include;

Read file.txt.torrent and get Tracker information out of it.

Request list of peers from tracker.

Connect to each peer and download file.

For each file piece download, write to disk.

* **Communication**

We can transfer data in an asynchronous manner where there is input and output at the same time. The communication methods include message passing where we send and receive primitives. During transfer, data comes in blocks from the peers. We use the port mechanism where the file requester and the peers communicate through open ports.

**EXPLANATION OF CODES WE COMMITTED**

* **File\_saver.py (committed by Mastura Lukman)**

This python script is meant to save the file that is being downloaded onto the computer. While the files are coming in blocks, every block should be received in a queue and saved into a file path on your computer hard disk till the download had finished.

* **Peer.py (committed by Gloria Samuel Timbau)**

This script helps the client to communicate with the available peers who have the requested files. To do this the client gets the peers information from tracker and opens a port to communicate with them. They now make a ‘handshake’, where the client sends an ‘interested’ request to the to each peer, when the peer agrees to give a piece of the file it has, the client now gets the privilege to download that piece. When that piece is finished it moves on to the next client to download another piece of the same file, till all the pieces come together to make up the full file.

* **Torrio.py (committed by Aikins Hotwards Francis)**

This script is deals with input and output torrent pieces that are coming in. Whenever any block comes in it hashes it in an index form to identity it. It takes information about the pieces in the block, when it begins, the length of the block and data in the block. Then upon any block received it checks for the piece and downloads it.

* **Tracaker.py (committed by Justice Markwei)**

The tracker file is responsible for looking for peers who have the file that is being requested, so that it can be downloaded, using the torrent file. It gets the peers, then it requests the peers. It gets peers parameters like; peer id, compact, number of peers ids, whether the event has started or not, peer port numbers, where they are uploading or downloading, and how much size is left of the torrent file.

* **Show\_torrent\_file.py (committed by Samira Abukari)**

This script displays the torrent file that was requested and downloaded from the peers after download has finished. It opens the file directory where the file has been saved and opens it upon download finished.

**CONCLUSION**

Sharing files is any important thing that has to be made easy and fast. Therefore, the best solution so far is to use a distributed system using a bittorrent protocol. Embarking on this project has taught us as a group how to collaborate to work and bring out an output and, it has also taught us how to build a torrent client. We express out heart felt gratitude to our dear lecturer Mr. Abandoh Sam for making us do this project. Than you sir.