SPL-1 Project Report

STORRENTO

Submitted by

SIFAT SIKDER

BSSE Roll No. : 1221

BSSE Session: 2019-2020

Submitted to

Supervised By:-

Dr. B M Mainul Hossain

Associate Professor, Institute of Information Technology (IIT)

University of Dhaka

Institute of Information Technology

University of Dhaka

28-05-2022

Contents

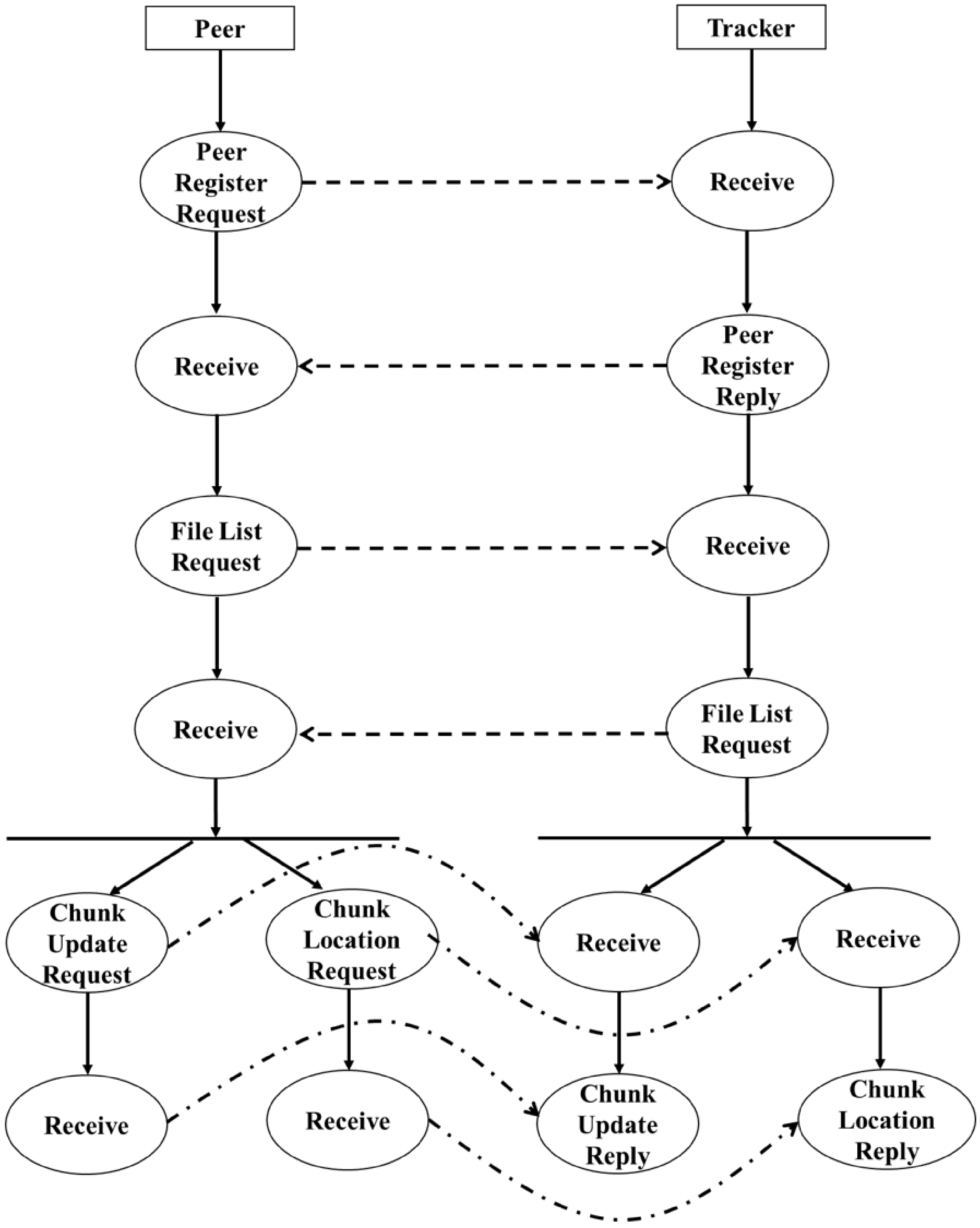
1. Introduction

With the development of peer-to-peer (P2P) networks the traffic landscape of the internet has radically changed. Today, P2P applications are still responsible for a vast amount of internet traffic. Even though, many peer-to-peer applications have been proposed and implemented in the past 10 years and different concepts have been improved and expended, the behavior of P2P networks and its traffic handling is still a major field of research.

2. Background of the Project

2.1 Peer-to-Peer Architecture

In a BitTorrent P2P network a tracker coordinates the communication between different clients. Whenever a peer connects or disconnects from a tracker an announce message is sent. Due to this fact, it is inevitable that a tracker has complete knowledge of all clients connected to its BitTorrent networks. Trackers offer a service to give a snapshot of the client constellation of a particular network, but only for a specific point in time. Observing the client behavior over a longer period has an important impact on research and further protocol improvements. Therefore. this project designs, develops and tests an application to acquire sets of data from different torrent swarms. A BitTorrent swarm refers to the set of peers currently connected to a tracker. Due to the fact that this tool should provide flexibility, reusability as well as extensibility, a detailed documentation is required. To assure that the application is able to perform this task absolutely reliable, deployment in a distributed environment is crucial. In a final step, a collection of data from a real life environment is acquired for a period of at least one week and subsequently evaluated



2.2 BEncoding

An ordinary HTTP Tracker responds to an announce request in an encoded format. The so called BEncoding offers the file sharing system BitTorrent to store and transmit data in an endian unspecific way, which is important for cross-platform communication. Torrent meta-data files take advantage of the same encoding mechanism. Bencoding allows the encoding of four different data types:

BEncoded strings: Strings are encoded without a beginning or ending delimiter and have always the following format: :. For example 5:world represents the string world.

BEncoded integers: Integers encoded in BEncoding format contain a starting and an ending delimiter, negative values are allowed and have the format: ie. The encoding of the integer 54 would be encoded with string i54e.

BEncoded lists: The encoding of a list follows the format: le, the character ’l’ is the starting delimiter and the ’e’ is used as an ending delimiter. An example for a BEncoded list is the sequence l5:plant5:ocean4:snowe, it is the representation of the list entries plant, ocean, snow.

BEncoded dictionaries: The final data type is the dictionary, it contains for each key a value. An initial ’d’ and a trailing ’e’ are used as delimiters. The dictionary values are BEncoded types as integers, strings, lists or even other dictionaries. The key must store a string value. The following encoding d5:plant5:green:5ocean4:blue4:snow5:white are dictionary entries plant ⇒ green, ocean ⇒ blue, snow ⇒ white.

3 Solution Design and Implementation

3.1 Design Specifications

According to the main goal of this project, which is acquiring a set of data about peer behaviour from trackers, an application has to be designed that provides the requirements flexibility, extensibility, good documentation and deployment.

- Flexibility must be provided regarding the application’s domain. There is a huge number of different tracker implementations connected to the global BitTorrent network. Hence, tracker behaviour may be unpredictable and hardly reproducible. Different trackers need to be announced in different intervals and the time to receive a response varies. This tool approaches this issue by providing as many reasonable starting configuration parameters as possible, in addition, sending announce requests for single or multiple torrents is supported.

- Extensibility, because BitTorrent offers beside the already discussed topics a wide range of other functionalities. For example instead of downloading .torrent files manually, magnet-links offer a service to acquire the meta-data information automatically. It is assumed that the BitTorrent protocol will be extended or changed in the future. Any newly designed tool related to BitTorrent has to take this fact into consideration

3.2 Implementation

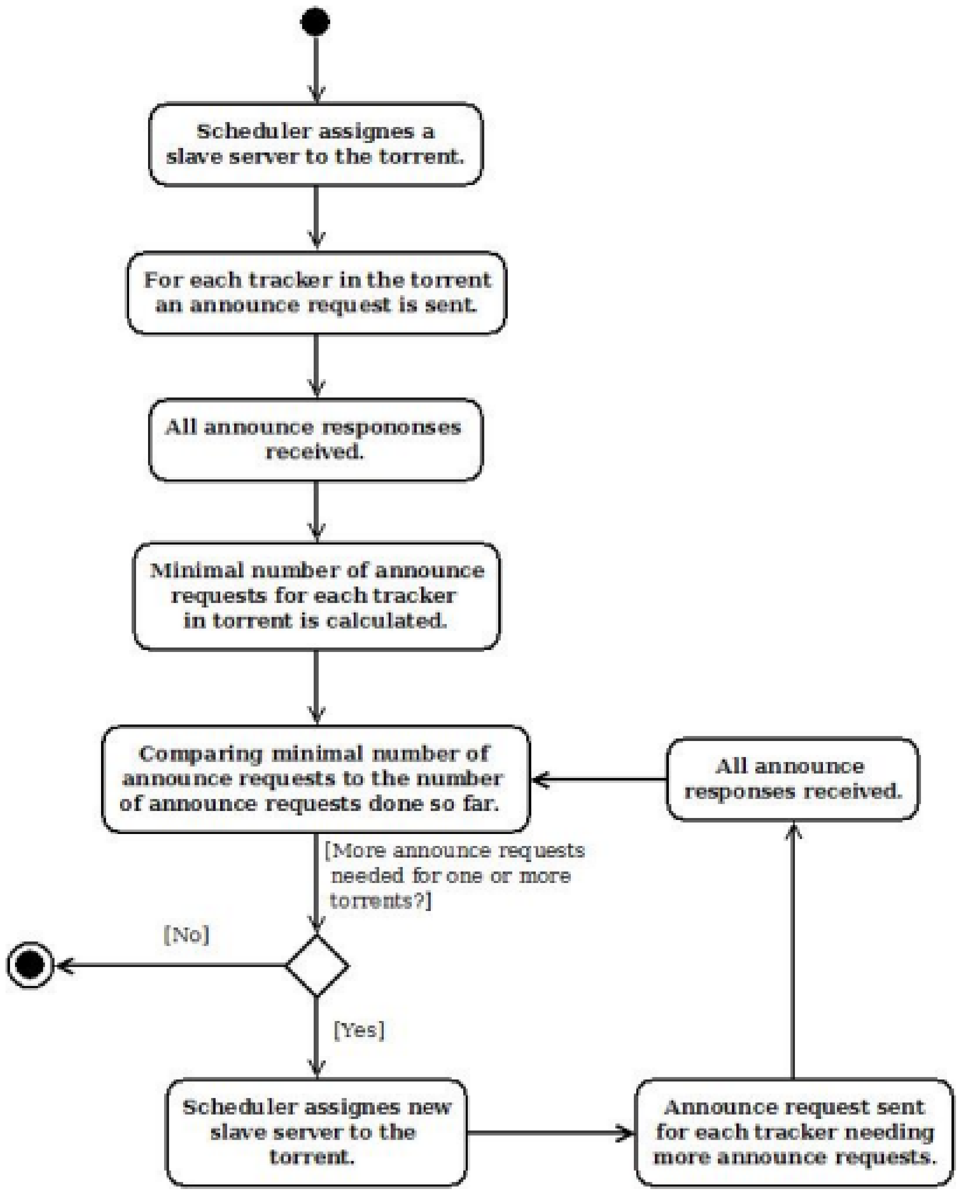

Scheduler assignes a slave server to the torrent.
For each tracker in the torrent an announce request is sent.
All announce repononses received.
Minimal number of announce requests for each tracker in torrent is calculated.
Comparing minimal number of announce requests to the number of announce requests done so far.
All announce responses received.
[More announce requests needed for one or more torrents?]
[No]
[Yes]
Scheduler assignes new slave server to the torrent.
Announce request sent for each tracker needing more announce requests.

Figure 3.7: Server assignment

3.3.1 Conversion of BEncoded Values

Torrent meta-data files and HTTP Tracker responses are generally encoded in a BEncoded format. This format is less efficient than pure binary encoding as used in UDP tracker implementations, but it is unaffected by endianness, which assures cross-platform communication. Although, BEncoding is an easily readable information representation, handling and processing this encoding can become exhausting. Due to this fact, another way of storing announce responses has been chosen. Especially regarding the extensibility of the tool, a different encoding seems to be a good choice. BValues are used in other BitTorrent related implementations, but are still not an official way to translate and store BEncoded values

3.3.2 Decoding the tracker address

After parsing the torrent file the announce portion contains the tracker address

3.2.3 Parsing the server response

Peers try to connect the tracker and then parses the server response

3.2.4 Connect to server after registering as a client

In order to upload or download a file a peer must have to register in the P2P architecture.

3.2.5 Download

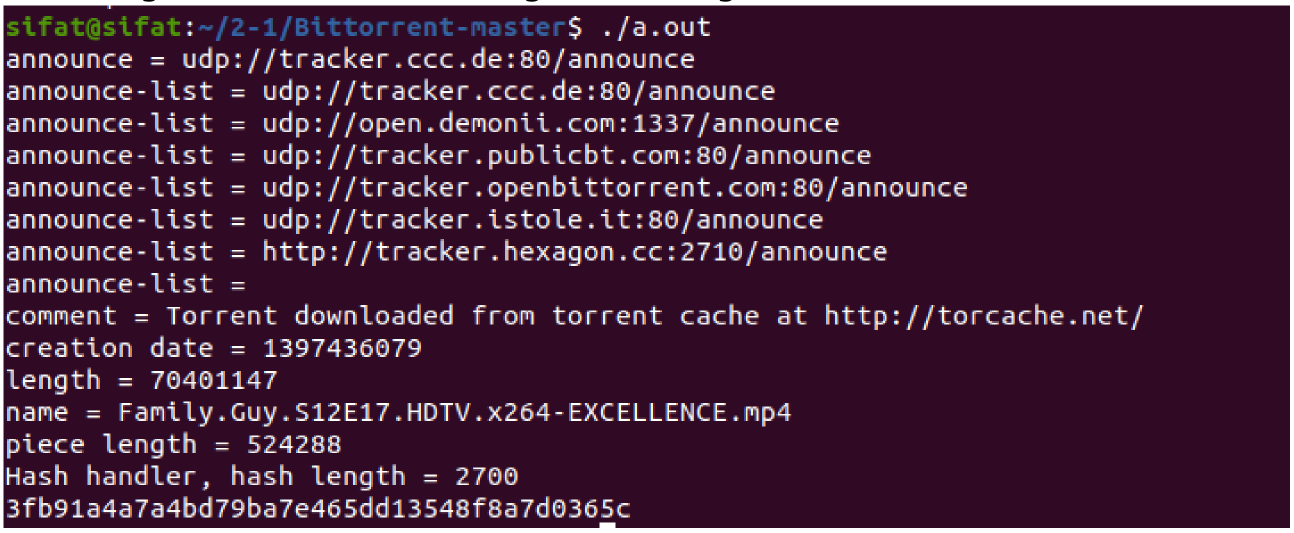
Peer can download different piece from another different peer then concatenate to get the whole file

3.2.5 Upload

When a peer uploads a file tracker saves the IP of that peer containing the file to serve other peers in future.

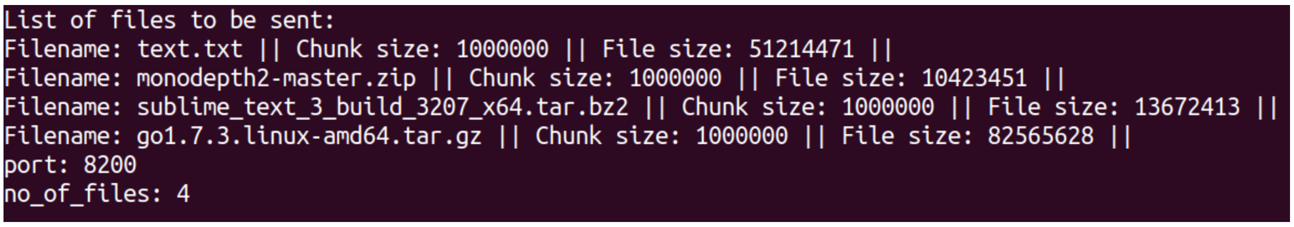
3. User Interface

1.Parsing the .torrent file using BEncoding



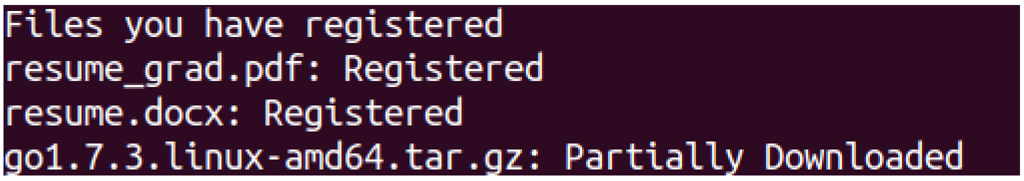
```
sifat@sifat:~/2-1/Bittorrent-master$ ./a.out
announce = udp://tracker.ccc.de:80/announce
announce-list = udp://tracker.ccc.de:80/announce
announce-list = udp://open.demonii.com:1337/announce
announce-list = udp://tracker.publicbt.com:80/announce
announce-list = udp://tracker.openbittorrent.com:80/announce
announce-list = udp://tracker.istole.it:80/announce
announce-list = http://tracker.hexagon.cc:2710/announce
announce-list =
comment = Torrent downloaded from torrent cache at http://torcache.net/
creation date = 1397436079
length = 70401147
name = Family.Guy.S12E17.HDTV.x264-EXCELLENCE.mp4
piece length = 524288
Hash handler, hash length = 2700
3fb91a4a7a4bd79ba7e465dd13548f8a7d0365c
```

2.register request to enter the P2P architecture



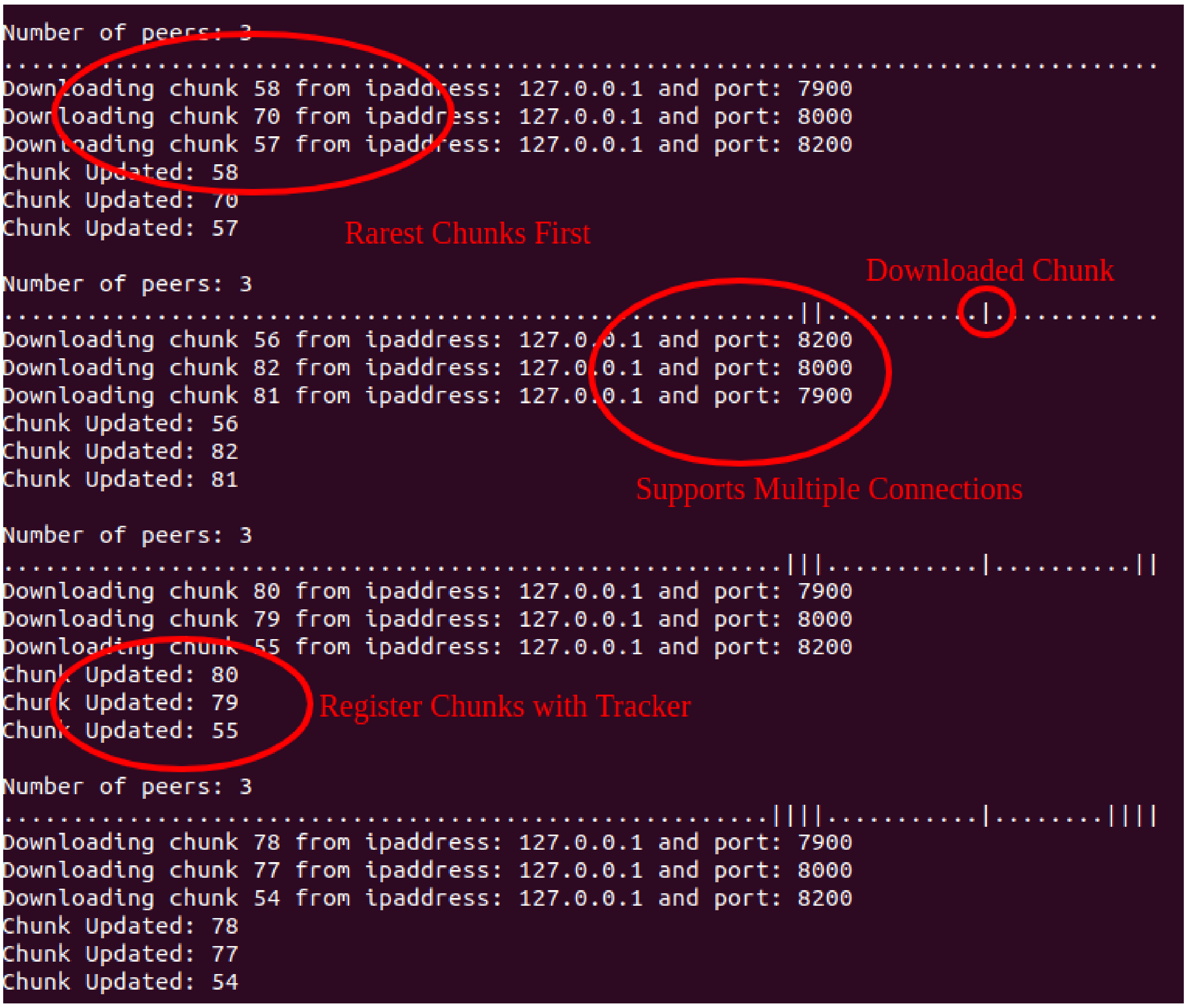
```
List of files to be sent:
Filename: text.txt || Chunk size: 1000000 || File size: 51214471 ||
Filename: monodepth2-master.zip || Chunk size: 1000000 || File size: 10423451 ||
Filename: sublime_text_3_build_3207_x64.tar.bz2 || Chunk size: 1000000 || File size: 13672413 ||
Filename: go1.7.3.linux-amd64.tar.gz || Chunk size: 1000000 || File size: 82565628 ||
port: 8200
no_of_files: 4
```

3.register reply



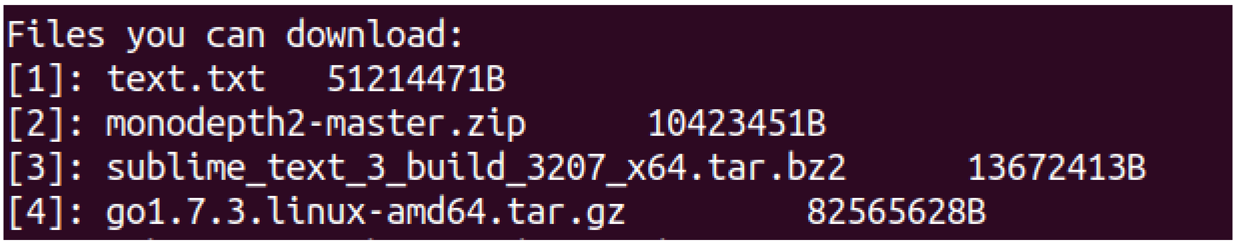
```
Files you have registered
resume_grad.pdf: Registered
resume.docx: Registered
go1.7.3.linux-amd64.tar.gz: Partially Downloaded
```

4.uploading the rarest chunk first and checking the multiple clients supporting or not



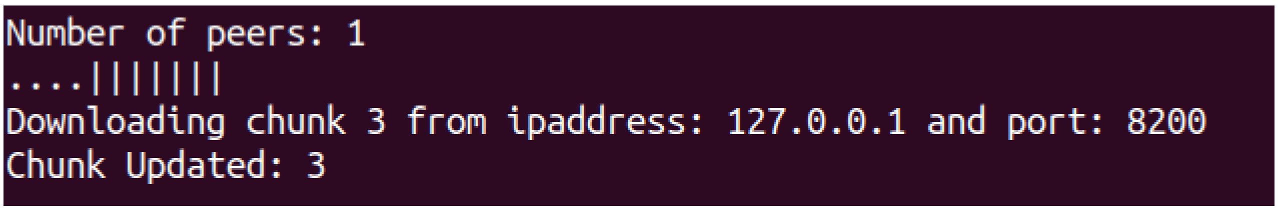
```
Number of peers: 3
....................................................................................
Downloading chunk 58 from ipaddress: 127.0.0.1 and port: 7900
Downloading chunk 70 from ipaddress: 127.0.0.1 and port: 8000
Downloading chunk 57 from ipaddress: 127.0.0.1 and port: 8200
Chunk Updated: 58
Chunk Updated: 70
Chunk Updated: 57

Number of peers: 3
..........................................................||...........|............
Downloading chunk 56 from ipaddress: 127.0.0.1 and port: 8200
Downloading chunk 82 from ipaddress: 127.0.0.1 and port: 8000
Downloading chunk 81 from ipaddress: 127.0.0.1 and port: 7900
Chunk Updated: 56
Chunk Updated: 82
Chunk Updated: 81

Number of peers: 3
.........................................................|||...........|..........||
Downloading chunk 80 from ipaddress: 127.0.0.1 and port: 7900
Downloading chunk 79 from ipaddress: 127.0.0.1 and port: 8000
Downloading chunk 55 from ipaddress: 127.0.0.1 and port: 8200
Chunk Updated: 80
Chunk Updated: 79
Chunk Updated: 55

Number of peers: 3
........................................................||||...........|........||||
Downloading chunk 78 from ipaddress: 127.0.0.1 and port: 7900
Downloading chunk 77 from ipaddress: 127.0.0.1 and port: 8000
Downloading chunk 54 from ipaddress: 127.0.0.1 and port: 8200
Chunk Updated: 78
Chunk Updated: 77
Chunk Updated: 54
```

5.Giving File list reply



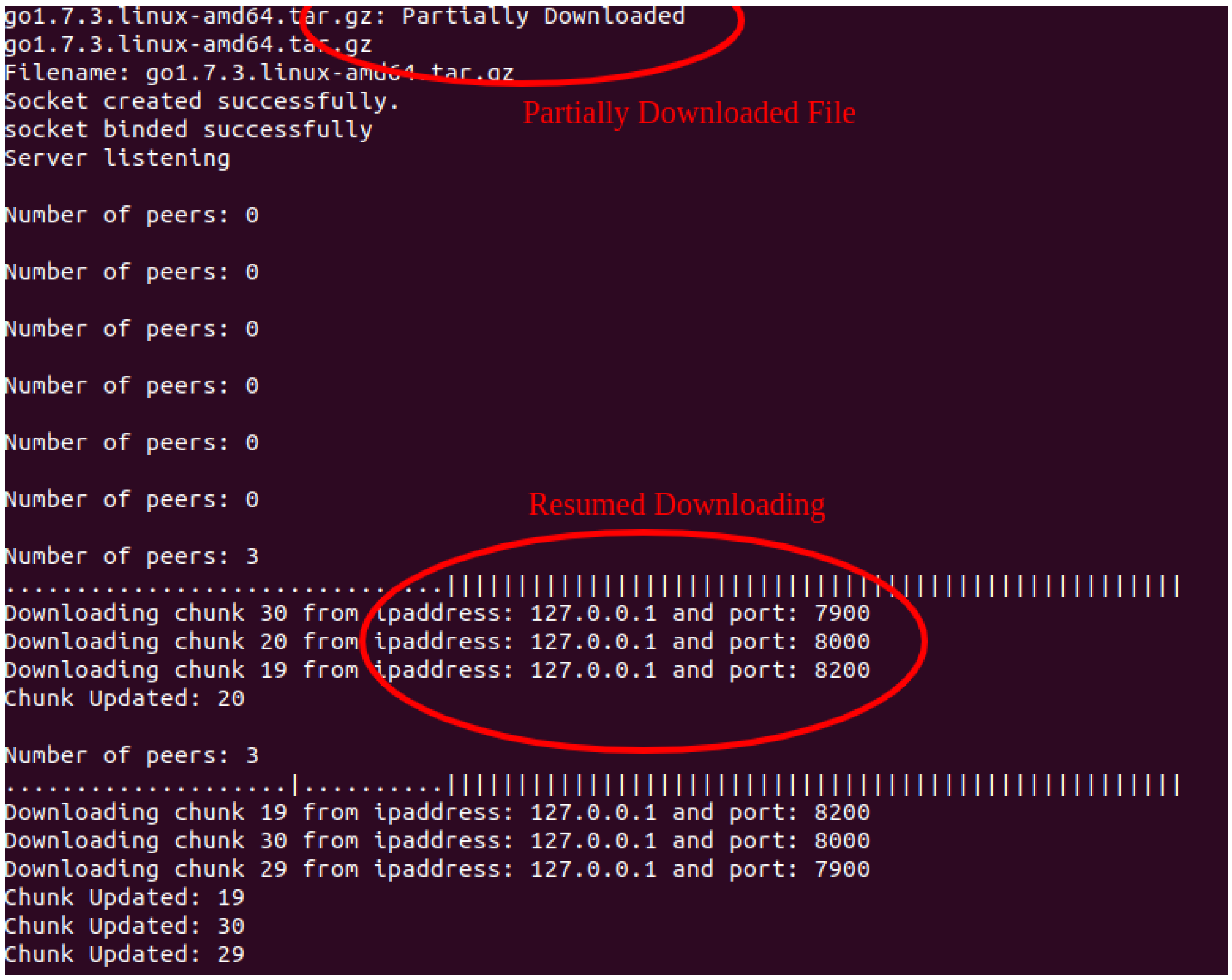
```
Files you can download:
[1]: text.txt   51214471B
[2]: monodepth2-master.zip      10423451B
[3]: sublime_text_3_build_3207_x64.tar.bz2       13672413B
[4]: go1.7.3.linux-amd64.tar.gz          82565628B
```

6.Download Progress



```
Number of peers: 1
....|||||||
Downloading chunk 3 from ipaddress: 127.0.0.1 and port: 8200
Chunk Updated: 3
```

7.Partially Downloaded File resume to download



```
go1.7.3.linux-amd64.tar.gz: Partially Downloaded
go1.7.3.linux-amd64.tar.gz
Filename: go1.7.3.linux-amd64.tar.gz
Socket created successfully.
socket binded successfully
Server listening

Number of peers: 0

Number of peers: 0

Number of peers: 0

Number of peers: 0

Number of peers: 0

Number of peers: 0

Number of peers: 3
...............................|||||||||||||||||||||||||||||||||||||||||||||||||||||||
Downloading chunk 30 from ipaddress: 127.0.0.1 and port: 7900
Downloading chunk 20 from ipaddress: 127.0.0.1 and port: 8000
Downloading chunk 19 from ipaddress: 127.0.0.1 and port: 8200
Chunk Updated: 20

Number of peers: 3
....................|..........|||||||||||||||||||||||||||||||||||||||||||||||||||||||
Downloading chunk 19 from ipaddress: 127.0.0.1 and port: 8200
Downloading chunk 30 from ipaddress: 127.0.0.1 and port: 8000
Downloading chunk 29 from ipaddress: 127.0.0.1 and port: 7900
Chunk Updated: 19
Chunk Updated: 30
Chunk Updated: 29
```

4. Challenges Faced

As BEncoding is a completely different encoding algorithm so it was hard to parse the .torrent file.

5. Conclusion

The evaluation of the Scraper application has shown that smaller numbers of peers can be easily acquired. Although, there is no guarantee that the received peer numbers and the individual IP addresses are correct or the acquired data complete. Regarding torrents with larger peer numbers, an accurate set of peers is difficult and not possible to acquire with the chosen design. Time delay caused by not responding trackers have a significant impact on the announce procedure.

I have learned new concepts such as socket programming,Piece Selection Algorithm, etc

6.Future Work

The work performed in this project provides a basis for further future research in different fields. The observed peer numbers in the one week acquiring process showed a periodic behavior of BitTorrent users. One possible direction of research is investigating the geographical location of different peers.