Important Note:

You can do this assignment in pairs like the last one. You can either write code in python or java. If there are questions ask the course staff; they are there to help you. Also please try to post questions and problems on the facebook page, it is there to help you out outside the office hours anytime 24/7. Other than your partner you are not allowed to discuss your code with anyone. For this assignment you must do all the thinking, research and coding by yourself.

- This assign is due on 18th and you must submit at least 70% work on the deadline. Any less work will be penalized and no work will automatically give you 0 marks. Also before the viva you can complete as much as you want but you will be graded on what you show and explain during the time of viva. This will be final. (

- **Code must work on Linux Terminals.** (

- No environment such as eclipse will be allowed. (Students are strongly advised that any act of plagiarism will be reported to the Disciplinary Committee. (

Intro: In this assignment you are required to construct a peer to peer file sharing model based on BitTorrent. BitTorrent was developed in response to centralized file sharing services where the centralized service did not allow multiple file uploaders to send files directly to multiple file downloaders. BitTorrent is designed to solve such problems by enabling file downloaders and file hosters to interact directly in a peer-to-peer fashion i.e in the BitTorrent protocol each participant acts as both the server and the client. Moreover using BitTorrent large number of downloaders can collaborate in downloading files from a much smaller number of file uploaders. As of 2013, BitTorrent has 15–27 million concurrent users at any time. As of January 2012,

BitTorrent is utilized by 150 million active users. Based on this figure, the total number of

monthly BitTorrent users may be estimated to more than a quarter of a billion

- **KeyWords:**

  - The heart of the BitTorrent protocol is a torrent file. The torrent file is a meta- information file containing information regarding the content to be shared/distributed.

  - Any participant (peer) has to have access to the torrent file. An initial peer needs to have access to the complete file for bootstrapping the transfer.

As of 2013, BitTorrent has 15–27 million concurrent users at any time. As of January 2012,

BitTorrent is utilized by 150 million active users. Based on this figure, the total number of

monthly BitTorrent users may be estimated to more than a quarter of a billion.

- A peer that has access to the complete content and it's only uploading it is called a seeder .

- A peer who is downloading and uploading and has incomplete access to the file, is called a leecher.

- A collection of peers (seeders or leechers) who are participating in a transfer based on torrent file forms a swarm.

- The core of the BitTorrent protocol is the tit for tat mechanism, also called optimistic unchoking allowing for upload bandwidth to be exchanged for download bandwidth. A peer is hoping another peer will provide data, but in case this peer doesn't upload, it will be choked.

- Another important mechanism for BitTorrent is rarest piece first allowing rapid distribution of content across peers. If a piece of the content is owned by a small group of peers it will be rapidly requested in order to increase its availability and, thus, the overall swarm speed and performance.

- The tracker server monitors the number of clients downloading/uploading files. When a file downloader contacts the tracker server, a list of other clients ip and ports are returned. Clients learn about other BitTorrent clients via the tracker server, as well as accepting new request from peer clients directly in a peer-to-peer fashion.

Problem (100): In this project you are going to write a file peer.py/.java which would include elements from both the server and the client file. From previous search engine try to think and remember which elements were unique to the server and which to the client. Moreover you are going to write the file tracker_server.py/.java which would represent the tracker server. We will assume, tracker server never crashes. A peer who wants to share his file will first generate a torrent file with the extension (.torrent) which would be based on the following format. **abc.torrent** Suppose a peer named X wants to share a file named got5.mp4 with other peers around the bitTorrent

network. Peer X will keep this file in local upload folder (please make sure you do not restart peer if new files are added to this folder). The torrent file abc.torrent contains the relevant info about the got5.mp4 file which X wants to share. The torrent file should be generated by X. To share the file the peer must first connect with the tracker server and send

```
Name : got5.mp4          <the name of the file>
Key : 13490              <the unique key different for every file>
size : 41,633,813 bytes  <the total size of the file>
n_parts : 30             <the number of parts the file is to divided>
```

in the torrent file. The tracker server must store the torrent file,port and ip of X. You have to decide on you own how the tracker server will store this info. In addition to this the tracker server should also parse the abc.torrent file and extract the name and write it to a file named search_content.txt. The format of the search_content.txt is shown below.

This contains the file got5 of the type mp4 which belongs to the torrent abc. Now there is another peer Y who wants to download a movie (mp4 file) named got5. He would first contact the tracker server and search for his relevant content. The tracker server would search the serach_content.txt file and would compare the keywords with the name got5 and if there is a match the server would send the corresponding torrent file which in this case is abc.torrent and the port and ip of all users which have the file in this case its only X.

After Y has received this info he can chose to close his connection with the tracker server. Now peer Y will contact peer X using his port and ip and request to download the file. The file can be divided into the number of parts listed in the torrent file. In the previous assignment you have been opening up a file and sending it completely to the other user. But here you have to divide the file in chunks and then send in. This means if there is a disconnection (or peer process is terminated) the file should resume the download where it was left previously. For

example the user has downloaded 4 parts out of 30 and then there is a disconnection; then after the connection is established again he would resume the download from where it was left eg from the 5th chunk. Also he can download the file from any other user eg Z who has the file and whose ip and port have been given by the tracker server for the same file.

Furthermore the peer must keep track of all other peers from whom he has downloaded a file or they have downloaded it from him. The peer must prefer those peers from whom he has downloaded files the most. You can keep track of this locally in any particular file format you like. To give preference to a particular user means that if there is a request from 2 users who want to download a particular file from you. You must first allow the "preferred" user to start downloading first and then the next user.

Peer will use ip-port of the tracker server ip port to connect to the torrent network. Peer.py <IP> <PORT>. When some peer wants to leave the network intentionally, he will notify the tracker server, and tracker server will update accordingly. If some peer unexpectedly leaves, you must handle this case appropriately without sending out information to the tracker.

This assignment will not be as simple as the first one. This assignment would only be completed if both the partners work together and contribute. This assignment requires multi

```
Name : got5  type: mp4  tor : abc.torrent

Name : score  type: txt  tor : xyz.torrent
```

threading and the error handling and the distributed network is a bit tricky to handle so please start early.

Checklist and Marks Division:

- Proper torrent file format which can include any type of file eg pdf,txt,mp4 etc. All files to be for upload to be present in upload folder (there can be multiple subfolders). Do not hard code for any

particular file format.(10)

- Peer connection with tracker server (10)
- Argument parsing when a peer becomes active (establishes a connection) eg peer.py <ip>

<port> from the command-line (5)
- Do not hard code any paths at all; your code must be generic (5)
- Tracker server complete working and storing content in the search_content.txt (10)
- Peer to peer connection (10)
- Multiple peer connection with both tracker server and themselves (requires multi-

threading) (15)
- File downloading in chunks+resume download mechanism; also multi downloading case

(20)
- Optimistic un-chocking and preferred downloading (10)
- Good interface of peer(5)