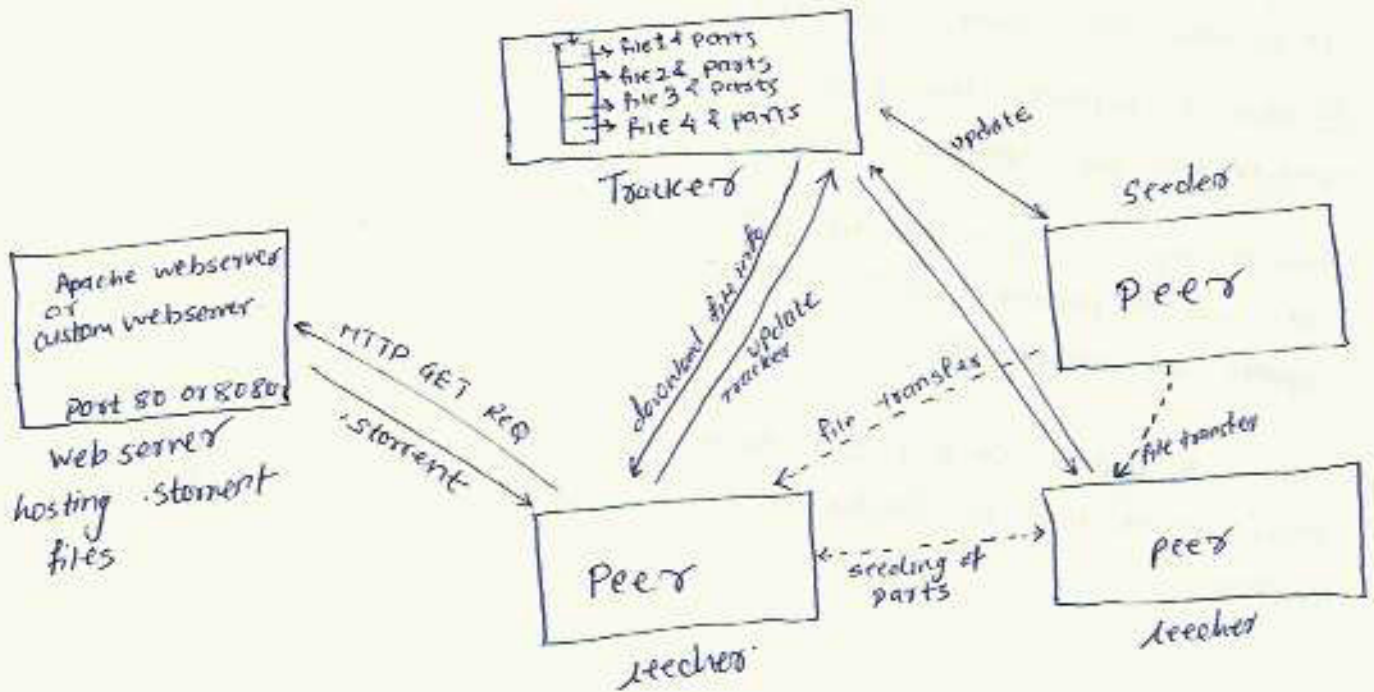


Software Architecture.



* Functions of each block:-

A] Webserver:-

- Host .torrent files with filename & tracker info.
- Allow peers to download .torrent files using webbrowser (port 8080 or 80)
- (ie create & run a ~~web~~ HTTP webserver on this machine)

Prerequisites:-

- ① Generate .torrent file
 - ↳ contents of .torrent file
- filename: file.torrent
- Tracker ip: 192.168.1.101
or
hostname: tracker.com
- Tracker port: 2700
- file pieces: 4 e.g.
- piece length: 64 KB

* How to decide parts & keep track of them? Should tracker keep info. on parts of file by storing offset or should it be written in the .torrent file?

B] Tracker:- (concurrent servers)

- Keep track and store info about seeders of specific file.
- give information about seeders to the leeches/peer.
 - ↳ information like:
 - ↳ seeders ip & port.
- Do i need to keep track of leeches?
 - ↳ if i do can it overload my tracker.
- ↳ from point 1. info to be stored.
 - filename:
 - ↳ parts (pieces)
 - ↳ which seeder has what parts.
- ↳ if possible :- keep info about leeches.
 - ↳ filename
 - ↳ ongoing download of parts
 - ↳ if the part is completely downloaded then move that leeches to seeder array for that file part.
 - ↳ update it's data structures based on data coming from seeders & leeches.
 - ↳ (optional) If seeder fails send error code to the leeches???

②

B] Tracker.

→ Prerequisites:-

* ① At least one seeder is present.

* ② How to populate Tracker D.S when tracker has just started.

③ What type of data transfer should be used to populate tracker D.S or update an existing D.S.

④ How to update tracker D.S to show most recent info on seeder and leecher.

C] Peer:-

This entity is a client software we use.

functions:- parse .torrent & get tracker info.

→ Get seeder info from tracker.

→ populate D.S with the info. from the tracker.

→ Then decide which seeders to choose & what part to download.

→ connect to the seeder and download the part. (client)

→ check update tracker about the download.

→ After download complete update the tracker.

* → Also check which part are present & update tracker that it can seed those parts.

→ if at least one part is completely downloaded then start seeder program (server) & listen for incoming requests. (concurrent)

→ property to retry if no seeder is currently present.

→ property to gracefully close all connections after download complete. or when seeder crashes or after a certain time limit.

A) Webserver Design:-

- webserver hosted on `www.about-webs.com`
- .torrent file uploaded to projects page.
- file can be downloaded from the webserver.

B) Tracker Design:-

```

int main ()
{
    // Create head array for storing file info. (D.S) [To be shared by all incoming & outgoing threads/process]
    // Start server and listen on port e.g 2700.
    for incoming requests/updates. (concurrent server) → use of thread
    // if leech requests & file not present in D.S then send error code to the peer (tracker)
    error code should specify retry property.
    # [How to detect if incoming request is for leeching or seeding]
    → Define a protocol
    

| Request type | Payload. |
|--------------|----------|
|--------------|----------|


    for leeching → leech 0xF1 : filename
    for seeding → seed 0xF2 : filename & port details (name) & ip
                   or update : ————
    // if request is that from seeder then extract data from message; acquire mutex & update the data structures. (populates respective fields).
    Also retrieve ip.
    // go into listen mode
    * Keep limit on the max. number of files for ongoing transfer.

```