**Application overview:**

* Một server keep track xem client nào đang kết nối và mỗi client đang có file gì.
* Sử dụng tracker protocol, client thông báo cho server nó đang chứa file gì, nhưng không transmit file cho server
* Khi một client thiếu file, request được gửi đến cho server
* Nhiều client có thể tải nhiều file từ một client cùng 1 lúc 🡪 requires the client code to be multithreaded.

**The simple Like-torrent essential components:**

* Magnet text: chứa mọi thông tin cần thiết để tạo metainfo file (yêu cầu tối thiểu là một mã băm trỏ đến tệp metainfo trên cổng theo dõi trung tâm của bạn)
* Metainfo File (.torrent file): Giữ tất cả chi tiết về torrent của bạn, (including where the tracker address (IP address) is, what is the piece length, piece-count)
* Pieces: Được chỉ định trong Metainfo File. (kích thước bằng nhau, thường là 512KB)
* Files: được chỉ định trong Metainfo File. Có thể có nhiều hơn 1 file trong 1 torrent -> map the piece address space to the file address space if you have N parts and M files. Simple math mistakes are very likely possible here, but you MUST carry out all of the address mapping by hand and by yourself definition for later referring.

**Tracker HTTP protocol:**

* Tracker request parameters:
* Contact the tracker server (the global centralized tracker portal), submit all the appropriate fields. You should **include the compact flag or metainfo in your request, a bare minimum requirement is a magnet text of your torrent** that be able to parse.
* Submit started, stopped and completed requests to the tracker server when appropriate. When you **send the started request**, you need to **include how many bytes you have downloaded already** (at the piece level detail)

**Tracker response:** The tracker responds with "text/plain" document consisting of a dictionary with the following keys:

* Failure reason: If present, then no other keys may be present. The value is a human readable error message as to why the request failed (string).
* Warning message: the response still gets processed normally. The warning message is shown just like an error.
* *Tracker id*: A string that the client should send back on its next announcements. If absent and a previous announce sent a tracker id, do not discard the old value; keep using it.
* Peers: (dictionary model) the value is a list of dictionaries, each with the following keys:
* peer id: peer's self-selected ID, as described above for the tracker request (string)
* ip: peer's IP address either IPv6 (hexed) or IPv4 (dotted quad) or DNS name (string)
* port: peer's port number (integer)

PEER – Downloading

* These are simple TCP connections with a 2-way handshake to enter the established state.
* After you receive an establish command from the peer, you can start downloading pieces. You download pieces by requesting blocks.
* The last requirement for downloading is that you send has messages each time you finish downloading a piece to close the connection.

\*\*\* Advanced \*\*\*

* Have a request queue, which keeps a list of all blocks you have sent a request message to the peer for.
* Your algorithm for choosing which blocks to request from which peer is completely up to you, with the minimum requirements that you don’t send a request for a piece the peer does not have, and *you don’t send a request for a piece you already have*.

PEER – Uploading:

* Your application has to begin seeding the file to other peers who are also interested in downloading it after you download it.

\*\*\* Advanced \*\*\*

The "tit-for-tat" theory underlies peer-to-peer file sharing, implying that the more you share, the more files will be shared with you. This project is a chance to apply your learned theory. The currently deployed peer selecting algorithm prevents free-rider by only changing peer  
selecting decision once every ten seconds.

User Interface:

Minimum requirements: downloading and seeding a single torrent to multiple peers, and provides us ability to see all the upload/download statistics.

Extra credit:

* Distributed hash table (DHT):
* **Distributed hash table (DHT):** transmit your torrent file without the centralized tracker. Typically, this is a better source to find more peers here than by using the torrent’s centralized tracker.
* **Simultaneous torrents**: require your client show how to download and upload several torrents simultaneously (at the same time), as well as access the statistics through your user interface.
* **Tracker scrape**: https://en.wikipedia.org/wiki/Tracker\_scrape is the exchange to obtain the metainfo on behalf of the client. It should be mentioned that a peer's involvement in a data transmission is unaffected by scrape exchanges.
* **Download/Seeding strategies**: a simple download strategy is required for your client, but if you choose to apply something fancier (like Super Seeding, Rarest-First, End Game), you need to document it and provide supplemental materials.
* Optimizing peer selection: a simple peer selection strategy is required for your client, meaning you can just response to all your peers. If you want to use the wiki's actual standards (of only 4+1 peers at a time), please make sure you document it