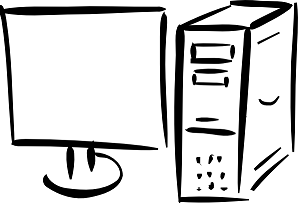
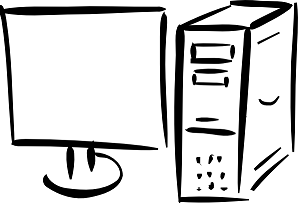
Overview Architecture:

4. Connect

Chunk Transfer

1. Connect/Ack

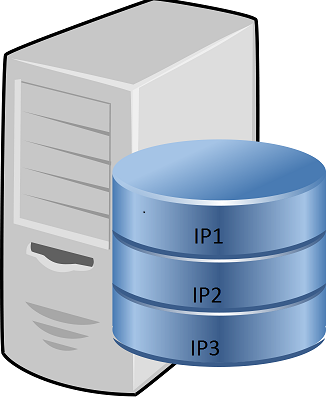
3. Do you have the file?

2. Query/5.Ack+Loc of file

Yes!

Tracker server

4. Connect



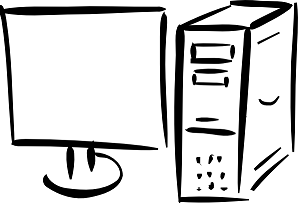
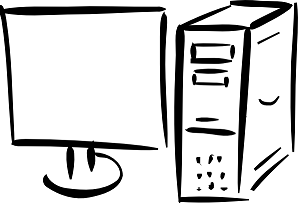
Chunk Transfer

3. Do you have the file?

3. Do you have the file?

No!

Yes!

Peers:

1. Peers run on virtual machine.
2. Each peer has a unique identifier which is its IP address.
3. Each peer has storage, a shared folder, to store the file it downloads.
4. Each peer is capable of splitting files into chunks and uploads into all other peers with replication.
5. Each peer can download chunks from other peers.
6. A peer can contact server to obtain information about other peers.
7. All peers operates on multi-threading.

Server:

1. Server stores IP address of all peers.
2. Server tells the peer where to upload chunks.
3. When a peer uploads a file, it has to tell the server where each chunk is stored.
4. Server tells peer where it each chunks are present, to download.

Interaction:

1. Server to peer and peer to peer interacts via **socket communication**.
2. Server and each peer have a particular **IP and port number** to establish connection.

Splitting of files into chunks:

1. Read the file and get the file length.
2. Specify chunk size
3. Read file till chunk size and store it as a temporary array.
4. Store it as a file with sequence number.
5. Transfer chunks via socket to peers.
6. Manage replication of chunks in peers.
7. Merging of chunks is done while downloading the chunks at the peer.

Validation:

1. Using MD5 checksum to validate file
2. Each chunk will be put through MD5 checksum and will be validated after being uploaded into peers.
3. Again, each chunk will be validated for MD5 checksum when it is downloaded.
4. Checksum compare is done based on the initial checksum and the checksum that the chunk has.
5. If any chunk is found corrupted after uploading/downloading, the chunk has to be uploaded/downloaded again.

Communication Overhead:

1. Handshake is an important overhead.
   1. Each peer has to connect to every other peer via two-way handshake.
2. Accessing peers to check if it has the chunk or not.
   1. If a peer needs a data, it has to ask other peers if it has a packet or not.
3. Validating via checksum
   1. Each peer has to send a checksum to all other peers to/from which it uploads/downloads the chunk to validate the chunk.

Socket communication uses IP and port no. which has lesser communication overload than Ethernet.

Computational overhead:

1. Validating the chunks after upload and download is a computational over head.
2. Any corrupted chunk will lead to upload or download of chunks again, which increases the computation as well.
3. Each peer has to search their shared folder for chunks.

Storage overhead:

1. A chunk has to be stored in multiple peers to avoid loss of data when a peer leaves.
2. Each peer has to store information about the chunks it has.

Challenges in splitting the file:

1. Loss of data:

* A peer gets IPs of other peers from the server.
* Server returns only the IPs of active peers and hence chunks are not lost by sending it to an inactive peer.
* Also, multiple copies of single file are present in different nodes. Hence, when a peer leaves, the chunk can be retrieved from other peers.

1. Corrupted data:

* Each chunk is appended with a MD5 Checksum.
* The same checksum is also sent separately to the peer to whom the chunk is uploaded.
* Then both the checksum are validated. If there is a mismatch, the chunk will be uploaded or downloaded again.
* Checksum validation is during merging the chunks too.

Work split:

Two members worked on communication between peers, tracker server and checking storage availability on each peer.

Two members worked on splitting files into chunks and merging it into complete file and validating whether the chunks are proper.