*Services:
1. Login service – performs signups / logins.

2. Data Service – performs uploads / downloads / modify.

3. Encryption Service – performs the handshake to rebuild the key used to encrypt and decrypt files.

*Internal API (all API is in form of TCP packets).

1. Check Status (used between the gateway and other nodes to check the health of the connection, used periodically on a set interval).

2. Checkfiles (if multiple data nodes are present, the nodes will sync for redundancy sake and to ease of load. Checkfiles call will check if the nodes are in sync).

*Outbound API (the API that the end-user will use)

1. signup(username, password)

Username – desired username (will be checked on the user service)

Password – desired password

Redirected to the user service.

Returns: 2 – username taken, 3 – account created.

2. login(username, password)

Redirected to the user service.

Returns: 0 – wrong account, 1-identity confirmed, id – identification number, salt (security)

3. connect(id)

Redirected to the data service.

Id – identification number, retrieved on login from user service)

4. upload(id,service,username,password)

Redirected to data Service.

5. download(id)

Returns array of all user's content to be displayed in the GUI.

Redirected to data service.

6. getKey(id,password,salt)

Performs multiple calls between the client and the encryption service.

Returns: data needed to reconstruct the encryption / decryption key.

Redirected to encryption service.

7. delete(id,password,service-id)

Deletes specific data from the data service.

*Load balancing

Architecture type star – there is only one gateway, but multiple nodes of other services. Using service discovery, the gateway caches a connection to each and performs queries round-robin style.

*Cache

1. Service ip and port. (cached cold)

2. Most active users will have their id cached (cached in-memory)

3. Recently downloaded data will be cached (will be expunged upon a timeout). (cached cold)