1. **What is the fundamental difference between P2P and client–server architectures? Is a P2P system with a centralized index equivalent to a client–server system? List the main advantages and drawbacks of P2P file sharing systems from different points of view:**
    - **end-users;**
    - **file owners;**
    - **network administrators.**

The fundamental difference between P2P and client-server architecture is that in P2P architecture there are multiples nodes and each of them is a client and the server at the same time. As each node functions as a client and a server they can make requests to other nodes and respond back to them. Whereas in the client-server architecture there can be multiple clients that each are connected to the same server via network so then clients only send requests to the server and server responses to clients.

P2P system with a centralized index is quite similar (but not exactly equivalent) to a client-server system as each node always ask for a resource from the main server which then responses to the request node by giving it the peer node that has the resource. Then the request node asks for the resource from the peer that the server provided, and it sends back the resource back to the request node. So it is somewhat similar to the client-server system but the server does not send back the resource to the request node and then the resource peer node gives responses with the wanted resources. So the rest of the communication is happening between nodes and not the dedicated server and the request node as in the client-server system.

Advantages of P2P file sharing system

- End-user:
    - Fast and easy file sharing
    - Faster download speed as nodes can load files from multiple other nodes at the same time
- File owners:
    - Fast and easy file sharing
    - Larger availability as every one that is connected to P2P network can request files
- Network administrators:
    - Probably cheaper than client-server to operate as there is no need to get server as each node is a server by themselves
    - Can add new clients(nodes) easily, so it is more flexible
    - If one node crashes, others are working, compared to client-server where if the server goes down the whole system stops working

Disadvantages of P2P file sharing system

- End-user:
    - Slower browsing speed as the node can simultaneously share files to other nodes and download files which both take up a lot of bandwidth.

    - Need to know from who you are downloading files as there could be malicious individuals
    - Transaction duration can be somewhat unpredictable (resources from across the world vs 10km away might take different time to download)
- File owners:
    - Not as secure as the client-server as everyone can be part of the network and be part of the file sharing
- Network administrators:
    - Not as secure as the client-server as everyone can be part of the network and be part of the file sharing
    - Energy consumption can be high if the network is large
    - How to add and remove nodes efficiently from the network
    - Need to figure out how the sharing of resources, data indexing(centralized, decentralized), and searching is done. Also does follow have hybrid systems, structured systems or unstructured systems network architecture.

2. **Propose a P2P architecture of the social network application, with the (key, data) pairs for the different entities which need be distributed. Describe how the following operations would work:**
    - **create or remove a user;**
    - **create or remove a friendship;**
    - **read message.**

The proposed P2P architecture follows the unstructured P2P networks architecture using the centralized index search where there is one central server which forwards the node requests to other nodes. That way nodes can always identify the right peers that they want to talk to. Also each node have their unique public and private key that is used to identify the user and encrypt/decrypt messages. Routing is done using the public/private key pairs where IP and public key are value pair so each message can be send to correct location using the public key of that user. Every message that is send across the network will need to have each user can be identified using public key/IP value pairs. Overall functionality is similar to Scuttlebutt gossip protocol (https://scuttlebutt.nz/about/) except with centralized index.

- Create or remove a user:
    - To create an user a public and private key is made for that node and all user data that they will. Private key is only stored in the local storage of the users device so they only have access to it. This means that they cannot use their device across multiple devices as private key is only stored in the one device. When a new user is created public key of the user is send to the centralized server which then records the public key and new user node data (like IP and not user data) so other can send messages to the new user using the public key and the key is forward to all peers in the network. And if user forgets their private key they cannot access their account anymore.

    - To remove and user a request is send to all friends and the centralized index which deletes all data about that node. Then the node is taken offline so it is not part of the network any more
- Create or remove a friendship:
    - In order to create a friendship with another user, the user needs to send a friend request to the node that they want to be friends with. So request is send to the central server node which then looks through the request data (public key of the receiver) and forwards receiver IP to the request node so they can send the request to the right node. Then if the receiver accepts the friendship, it adds the requester to their file sync and vice versa so each post the friend makes is added to both of theirs storage.
    - To remove a friendship request node sends the request to the friend node (does not need to use the central server as it is a friend, and they have the friends IP stored locally) and with data containing that they want to remove them as a friend. Then both of them remove the other ones from their file sync so they will not sync the posts that the other has made any more.
- Read a message:
    - When user wants to talk to another they will send the message to the other using the receivers public key and encrypting the message using receiver public key. Request data has the encrypted message, receivers public key and sender data. Then the centralized server will give the knowledge where to contact based on the receivers public key and then sender will send the message to that IP where the receiver is. Then the receiver will read the message by decrypting it with their private key.

3. Compare and contrast the different (at least four) approaches to storage system design in terms of
    - scalability, ease of use (ingesting data, etc.)
    - architecture (shared, shared-nothing, etc.)
    - consistency scheme
    - fault-tolerance
    - meta-data management.

    Try to generate a comparison table in addition to a short discussion.

Name	Description	Scalability, ease of use	Architecture	Consistency scheme	Fault-tolerance	Meta-data management
Distributed file system (Google file system)	It is distributed file system created by Google for their internal use for search engine and data analysis	Easy to add more data and read data from the chunks. Large chunks might also reduce the network overhead(https://www.techopedia.com/definition/26906/google-file-system-gfs). Probably meant to work with Google's systems. High scalability is provided by the shadow master as it can provide read-only access to file	Files are organized in tree directories and they are identified by pathnames. Files ae divided into fixed chucks sizes. There is application that uses the file system and GFS client. Application gets the chunk location and GFS client get the chunk from that location. Chunk server stores the chunks. One master maintain the file metadata	Relaxed consistency model for concurrent writes and appends to records. It uses checkpointing and self-validation record writing.	Replication and failover. Each chunk is replicated at many servers (min 3). When one server fails then master redirects all file access to the replica. Shadow master is also functional in case of if the master fails.	One master maintains the file metadata (namespace, access control info, chunk placement info)

		system even if the master is down (https://static.googleusercontent.com/media/research.google.com/fi//archive/gfs-sosp2003.pdf)				
Cloud storage	Storage model that enables that files and data can be stored on the internet and can be accessed using public internet.	Scalability is high as each user can specify their wanted storage amount and pay only how much they need.	Only ones that can access the data are the ones that you have given permission. You can make HTTP requests to the cloud storage provider and you can access the cloud web server if you have specific key. On server there can be multiple other servers like database server where different kinds of data is stored	Has consistent actions which happens after each operations for data and the metadata. Although (https://cloud.google.com/storage/docs/consistency)	Can have back-ups database in case of faults that can be used instead of the regular ones. But depends on what kind of cloud storage has been selected	Each object in the cloud storage have their own metadata. It tells how the object should be handles and what kind of properties it has. It exists as a key-value pair. Some of the metadata can eb edited and some cannot. (https://cloud.google.com/storage/docs/metadata)
Data lake	Centralized storage where data can be stored in any scale and data can be unstructured or structured. Stores relation data (https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/)	Very scalable as is supports any kind of data. It is also very flexible as it provide multiple access methods for the same data	Any kind of data can be stored there. It is a large container. It consists of three data layers where one is used to analyze data, one is used to store data and one is used to access data from external sources. Application are built on top of these. Then there is platform management where all data governance and security(authentication, data protection, etc) is handled. (https://moodle.lut.fi/pluginfile.php/1235797/mod_resource/content/1/Principles%20of%20Distributed%20Database%20Systems%204th%20edition.pdf)	Lacks data consistency as data can be in any format and any kind of data	Historical data is store so data accuracy is ensured, and this enables the system to recover from failures and get old data (https://www.upsolver.com/glossary/data-lake)	Each data has unique identifier with their own metadata information (https://www.guru99.com/data-lake-architecture.html) Metadata capture can be automated. (https://www.oreilly.com/content/tips-for-managing-metadata-in-a-data-lake/)
Data warehouse	Is distributed database which is used in BI and BA. Good for large scale enterprise sysstem(https://moodle.lut.fi/pluginfile.php/1320036/mod_resource/content/1/L11%20NoSQL.pdf)	Highly scalable as it support huge number of data and many different users. Also it supports unstructured big data although data first needs to be transformed in to same kind of data format.	Data is split to 3 smaller parts (OLAP, reporting and data mining) and they can be accessed by different users for different purposes. Relies on relational DBMS for managing the data. Has three layer architecture. Data comes from the source layer which is then pipe through ETL tools which transforms the data into common format. Then in reconciled layer which verifies the data. Then it goes into data warehouse layer where it is directed into data marts which are used by the analysis tool. (https://moodle.lut.fi/pluginfile.php/1320036/mod_resource/content/1/L11%20NoSQL.pdf).	Data can from multiple sources but is transformed into a common format to keep the data more consistent. Has a global schema (https://moodle.lut.fi/pluginfile.php/1320036/mod_resource/content/1/L11%20NoSQL.pdf)	ELT phase checks for duplicates, inconsistencies, or missing/unexpected data. Then whole data warehouse is rewritten after each refresh (https://moodle.lut.fi/pluginfile.php/1320036/mod_resource/content/1/L11%20NoSQL.pdf)	Metadata is important part as it helps the administrator to decide what to do with data (delete, retain, update, usage in the future). This metadata information is used when new data arrives to data warehouse (https://www.interviewbit.com/blog/data-warehouse-architecture/)

- If no additional link is provided then the information is from the course book (), the lecture video or the lecture slides.

Overall, these different storage solutions differ quite a lot from each other. Data warehouse is more organized version of data lake which is primarily used by Business intelligence and enterprise companies due to it supporting large number of file and provides analysis tools. Google file system is used by Google, and it is intended for large chunks of data like Google's own search engine. Cloud storage provides a lot of flexibility (in terms of options, size, location, etc.) as different data can be stored in there using different storage methods like in SQL or in Non-SQL databases. Data lake should be used if you don’t know what kind of data you want to store, or it is in multiple different format as data lake supports unstructured and structured data in any scale.