Federated Architecture

...

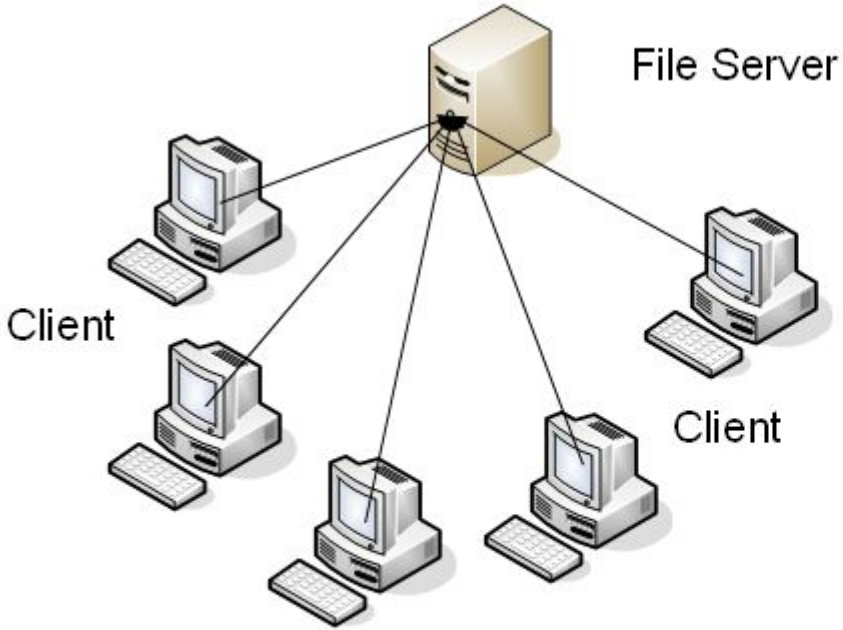
Mini Survey

- Have you started **coding** your projects?
- Are you using django, flask, serverless (AWS lambda), or something else?
- Do you plan to try to get jobs as software engineers after graduating?
- Is there any topic you're dying to learn, that we haven't yet?
    - Feel free to email me after!

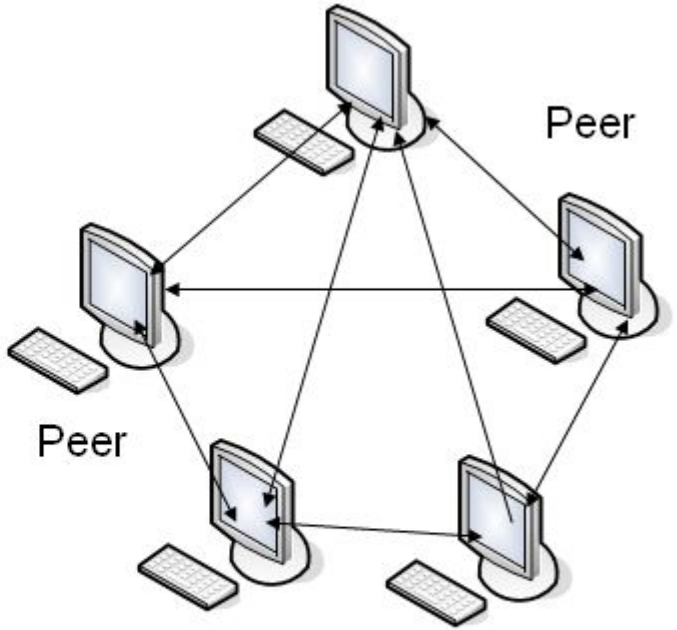
Peer to Peer

- In the common client-server architecture, multiple clients will communicate with a central server.
- A peer-to-peer (P2P) architecture consists of a decentralized network of peers - nodes that are both clients and servers.
- P2P networks distribute the workload between peers, and all peers contribute and consume resources within the network without the need for a centralized server.

Peer to Peer



Client server mode



Peer to Peer mode

P2P: Examples

- Some examples of P2P architecture:
  - Napster - shut down in 2001 since they used a centralized tracking server
  - BitTorrent - popular P2P file-sharing protocol, usually associated with piracy
  - Skype - used to use proprietary hybrid P2P protocol, now uses client-server model after Microsoft's acquisition
  - Bitcoin - P2P cryptocurrency without a central monetary authority
  - Spotify - used to use P2P, changed to central server

Peer to Peer: Types of Peers

- Not all peers are necessarily equal.
- **Super peers** may have more resources and can contribute more than they consume.
- **Edge peers** do not contribute any resources, they only consume from the network.

Peer to Peer: Overlay Network

- The P2P **overlay network** consists of all the participating peers as network nodes.
- There are links between any two nodes that know each other
    - if a participating peer knows the location of another peer in the P2P network, then there is a directed edge from the former node to the latter in the overlay network.
- Based on how the nodes in the overlay network are linked to each other, we can classify the P2P networks as **unstructured** or **structured**

Types of P2P

- Unstructured
- Structured
- Hybrid (client-server + P2P)

P2P: Unstructured

- All nodes can talk to all other nodes at same time
- Given a random set of neighbors upon entry to network
- Hard to “search” for a specific file

P2P: Unstructured

- An unstructured P2P network is formed when the overlay links are established **arbitrarily**.
- Such networks can be easily constructed
    - new peer that wants to join the network can copy existing links of another node and then form its own links over time.
- What might some disadvantages be?

P2P: Unstructured

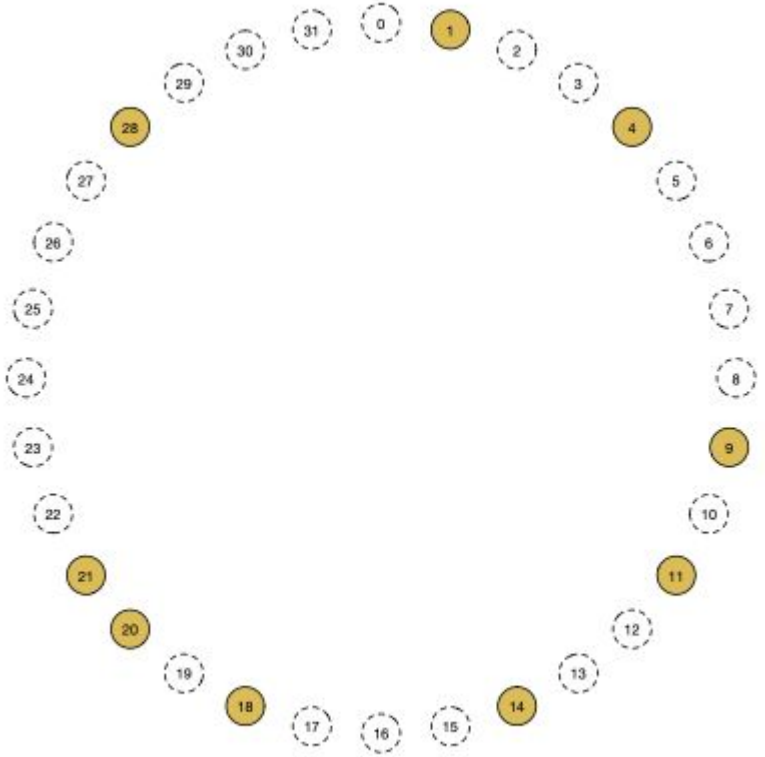
- If a peer wants to find a desired piece of data in the network, the query has to be flooded through the network in order to find as many peers as possible that share the data.
    - the queries may not always be resolved.
- A popular content is likely to be available at several peers and any peer searching for it is likely to find the sam
    - but if a peer is looking for a rare or not-so-popular data shared by only a few other peers, then it is highly unlikely that search will be successful.
- Since there is no correlation between a peer and the content managed by it, there is no guarantee that flooding will find a peer that has the desired data.
- Flooding also causes a high amount of signalling traffic in the network and hence such networks typically have a very poor search efficiency.

P2P: Structured

- Nodes interact with each other in organized manner
- Each peer responsible for a specific part of the content in the network.
- Use hash functions and assign values to every content and every peer in the network
- Then follow a global protocol in determining which peer is responsible for which content.
- This way, whenever a peer wants to search for some data, it uses the global protocol to determine the peer(s) responsible for the data and then directs the search towards the responsible peer(s).

Distributed Hash Table

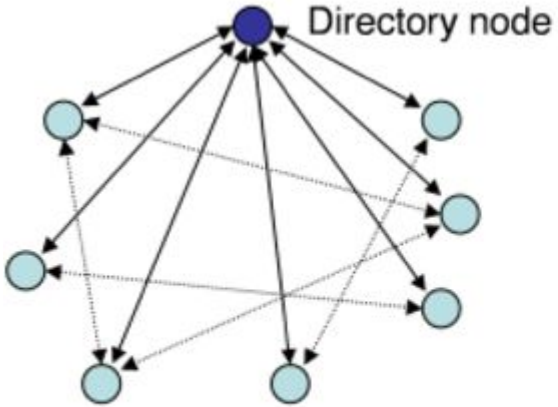
- A distributed hash table (DHT) is a directory with no central service.
- DHTs automatically heal when nodes fail. New nodes can also join at any time.
- DHTs form a ring in the address space with each node knowing about its successor and predecessor nodes.
- A simple lookup algorithm is to linearly ask down the chain until the key is found, but that's slower than it needs to be.
    - Instead with an address space of N, a table of log(N) entries can give us sufficient data so that log(N) hops will get us the value associated with a particular key.
- DHTs provide a fast and efficient decentralized directory system.



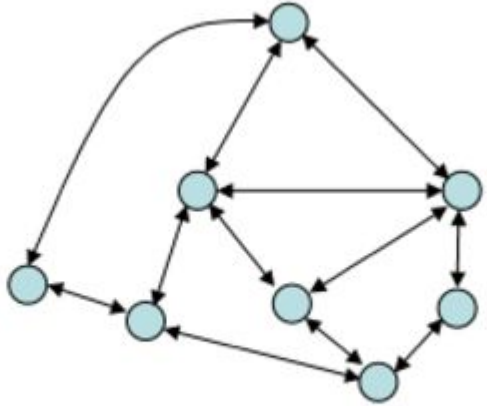
P2P: Hybrid

- In its purest form, P2P architecture is completely decentralized.
- “Hybrid”: hybrid between client-server and P2P
- There is a central tracking server layered on top of the P2P network to help peers find each other
- Central tracking layer directs requests to nodes

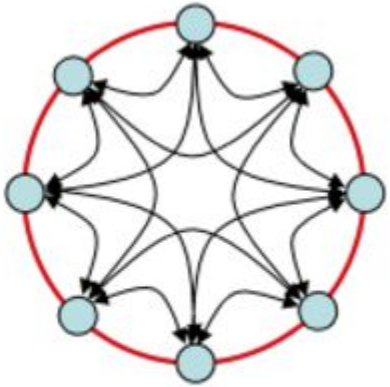
P2P: Types



Hybrid (Napster)



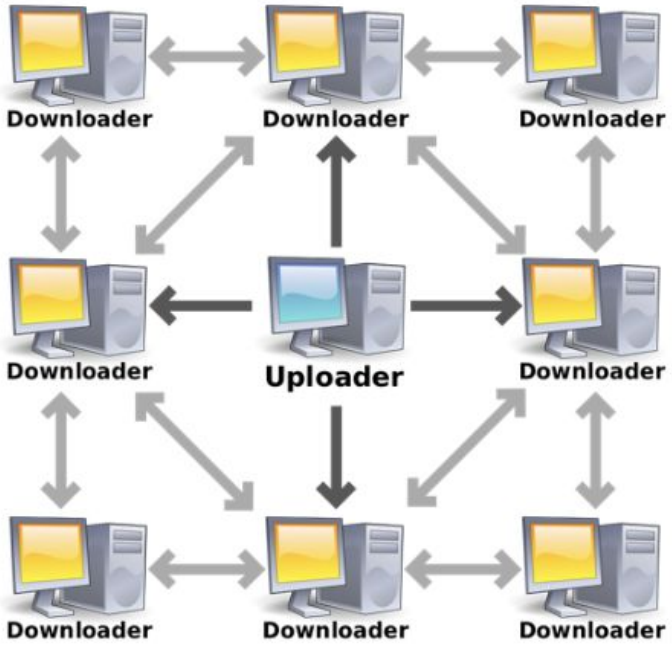
Unstructured overlay



Structured overlay

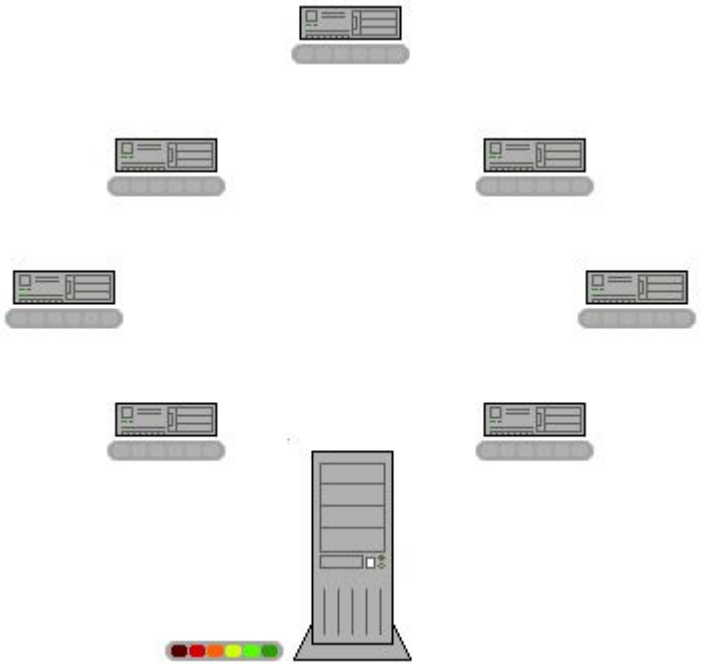
P2P File Transfer: Swarm

- How to share files?
- In swarming file transfer:
  - an uploader starts everything off
  - then clients download files from each other.



P2P: File Transfer: Segmented

- Breaks the file into segments that clients can exchange as smaller chunks to assemble the complete file.
- The seed (bottom) transfers segments to different machines in the network.
- After the initial segment's transfer, the segments are transferred directly from client to client.
- The original seeder only needs to send one copy of the file for all the clients to receive a copy.
    - The load is spread out among all the clients.



P2P Example: Bitcoin

- The term "bitcoin network" refers to the collection of nodes running the bitcoin P2P protocol.
- Decentralized ledger of transactions, creating a system that is trustless
    - No central server or authority deciding if your bitcoin is “valid”
- The “next” node is decided by consensus between peers
    - Incentive to go to consensus node since it’s likeliest to continue chain

Byzantine Fault Tolerance

- Byzantine Fault Tolerance is a computer system's ability to continue operating even if some of its nodes fail or act maliciously
- Byzantine node refers to the traitor node which could lie or mislead other nodes in the network intentionally.
- Types of failures:
  - Failing to return a result
  - Providing responses with incorrect results
  - Responding with deliberately misleading results for queries
  - Providing a response to a single query with different results to different components of the system

Byzantine Fault Tolerance

- Practical Byzantine Fault Tolerance emerged as one of the prominent optimizations of BFT in 1999
  - Barbara Liskov and Miguel Castro, paper: ‘Practical Byzantine Fault Tolerance.’
  - If you’re interested in theoretical CS, read the paper!

Byzantine Fault Tolerance: the simple explanation

- All of the nodes in the pBFT model are ordered in a sequence with one node being the primary node (leader) and the others referred to as the backup nodes.
- All of the nodes within the system communicate with each other
- The goal is for all of the honest nodes to come to an agreement of the state of the system through a majority.
- Nodes communicate with each other heavily
    - have to prove that messages came from a specific peer node
    - also need to verify that the message was not modified during transmission.

Byzantine Fault Tolerance: the simple explanation

- For the pBFT model to work, the assumption is that the amount of malicious nodes in the network cannot simultaneously equal or exceed ⅓ of the overall nodes in the system
- The more nodes in the system, then the more mathematically unlikely it is for a number approaching ⅓ of the overall nodes to be malicious.

P2P: Advantages

- There is no central server to maintain and to pay for (disregarding tracking servers), so this type of network can be more economical
    - Costs are also distributed between nodes, who have incentive to be there
- There is no single point of failure
- P2P networks are very resilient to the change in peers; if one peer leaves, there is minimal impact on the overall network.
- P2P networks can survive attacks fairly well since there is no centralized server.

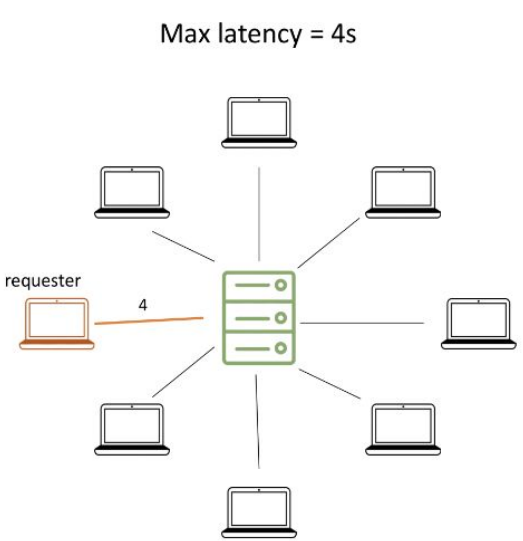
P2P: Disadvantages

- Introduce many security concerns.
    - One peer is infected with a virus and uploads a chuck of the file that contains the virus,
- P2P networks often contain a large number of users who utilize resources shared by other nodes, but who do not share anything themselves.
    - These type of freeriders are called the leechers.
- Although being hard to shut down is an “advantage”, it can also be a disadvantage if it is used to facilitate illegal or immoral activities.
    - Silk road: guns, body parts, people being sold

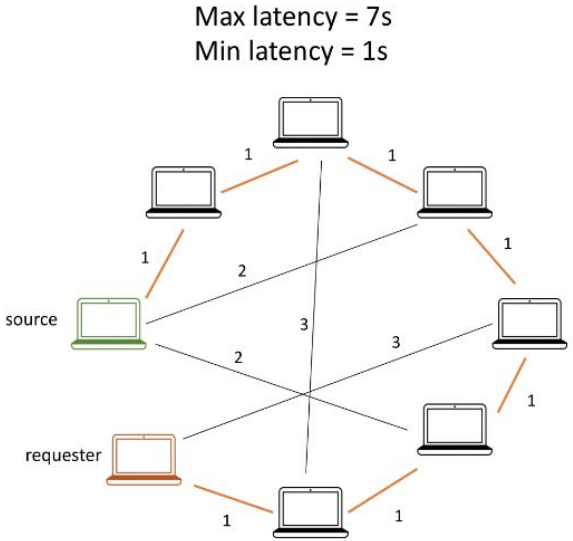
P2P: Disadvantages, Continued

- If there are many peers in the network, it can be difficult to ensure they have the proper permissions to access the network if a peer is sharing a confidential file.
    - “Wild west” or total openness of information usually the norm
- Consistency is hard!
    - No central server controlling consistency
    - Nodes have to pull updates and resolve inconsistencies on their own timeline
- Latency (following slide)

P2P: Latency Tradeoffs



Centralized System



Peer-to-Peer System

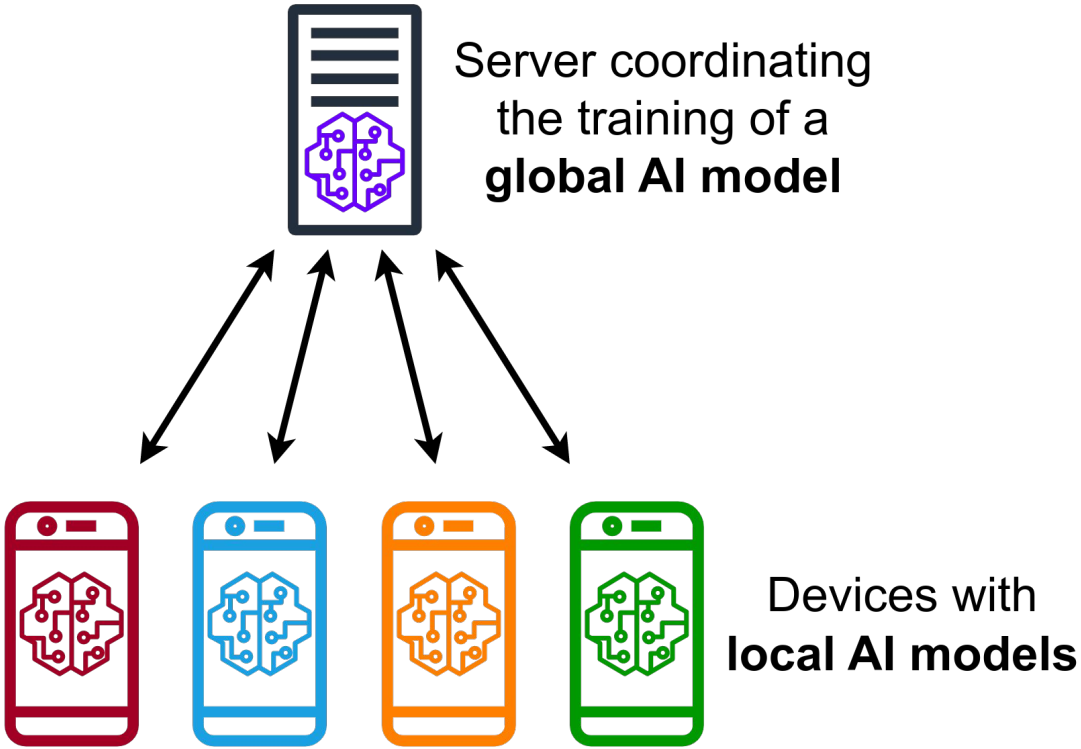
When to use P2P: When there are enough peers

- P2P architecture works best when there are lots of active peers
- New peers joining the network can easily find other peers to connect to.
- If a large number of peers drop out of the network, there should still enough remaining peers to pick up the slack.
- If there are only a few peers, there are less resources available overall.
- For example, in a P2P file-sharing application, the more popular a file is, which means that lots of peers are sharing the file, the faster it can be downloaded.

Federated Learning

- Federated learning (often referred to as collaborative learning) is a decentralized approach to training machine learning models.
- It doesn't require an exchange of data from client devices to global servers.
- Instead, the raw data on edge devices is used to train the model locally, increasing data privacy.

Federated Learning



Federated Learning

- Your device downloads the current model, improves it by learning from data on your phone, and then summarizes the changes as a small focused update.
- Only this update to the model is sent to the cloud, using encrypted communication, where it is immediately averaged with other user updates to improve the shared model.
- All the training data remains on your device, and no individual updates are stored in the cloud.

Federated Learning: Harder on the ML Engineers

- Can’t look at data to understand issues
- Can’t retrain different models with same data for experimentation
- Code to “average” the update, staleness of update, etc

Federated Learning: Better for privacy

- Your data stays on your device
- No personal data ever sent to server
- Configured to only train when phone not in use
- Example: GBoard

In Class Exercise

- “Design” the frontend of your quiz app
  - You are not being graded on design, but it should be usable
  - Beautiful websites: https://loadmo.re/
  - Ugly Websites: http://www2.pnwx.com/
- Show me your “design” when done, even if rought/WIP
  - All pages “views” should be represented
- Ok for one person to work on this while others code