Decentralized networks, decentralized Internet

Sept 19, 2024

application layer

Min Suk Kang
Associate Professor
School of Computing/Graduate School of Information Security

NetS&P Research Lab @ KAIST

KAIST

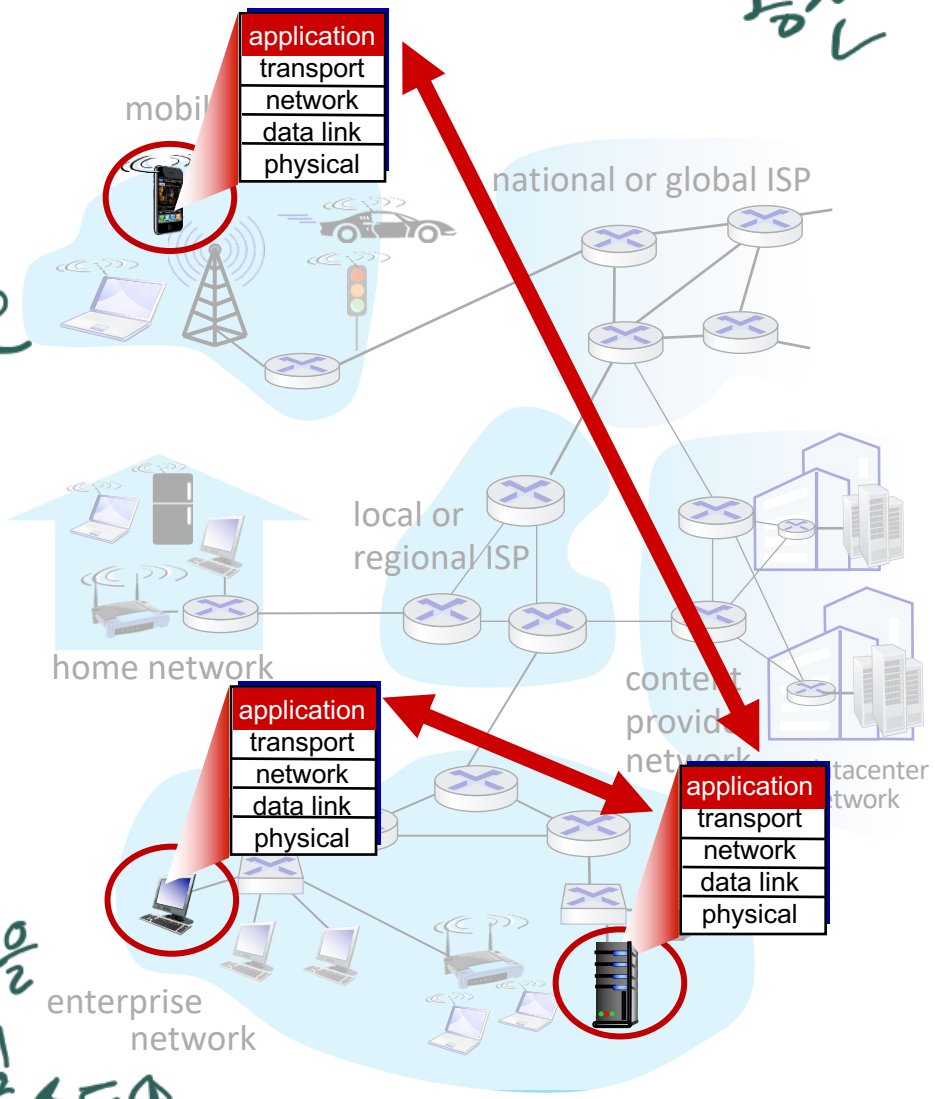
Creating a network app

write programs that:

- run on (different) end systems
- communicate over network
- e.g., web server software communicates with browser software

no need to write software for network-core devices

- network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation



Client-server paradigm

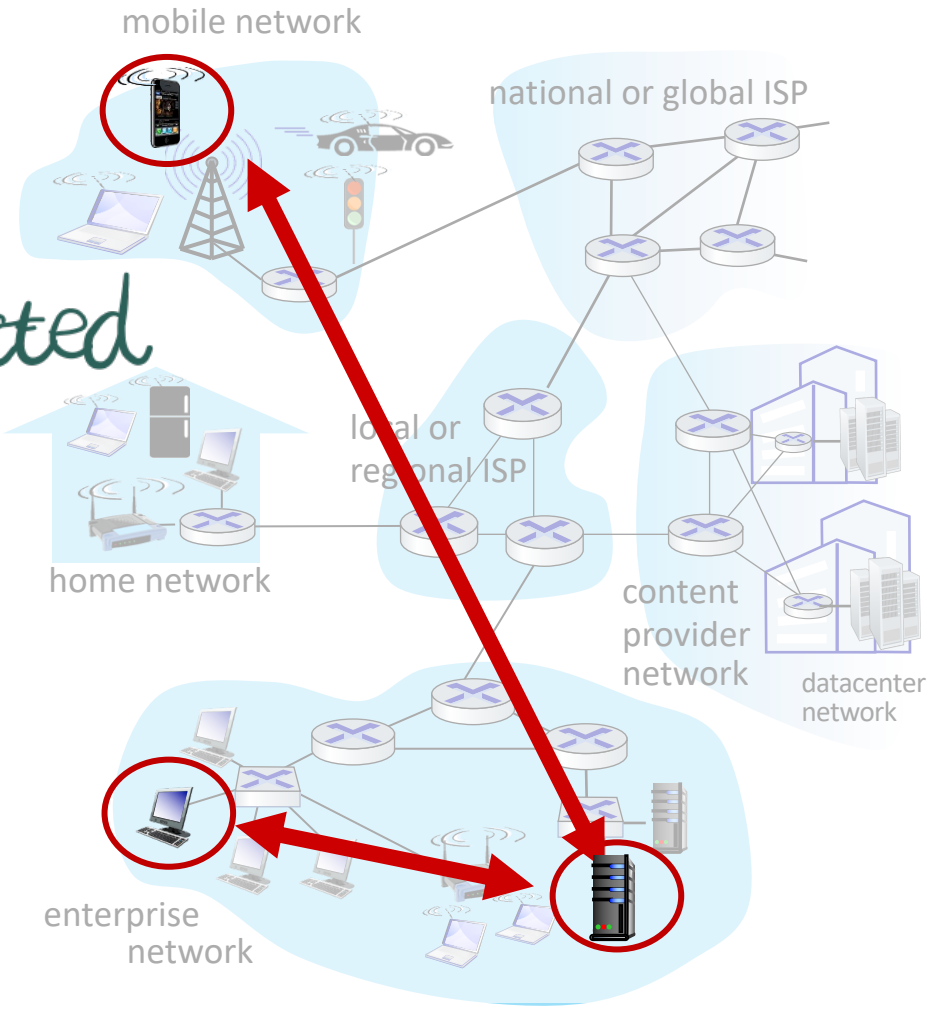
server:

- always-on host
- permanent IP address
- often in data centers, for scaling

clients: don't need always connected

- contact, communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do *not* communicate directly with each other

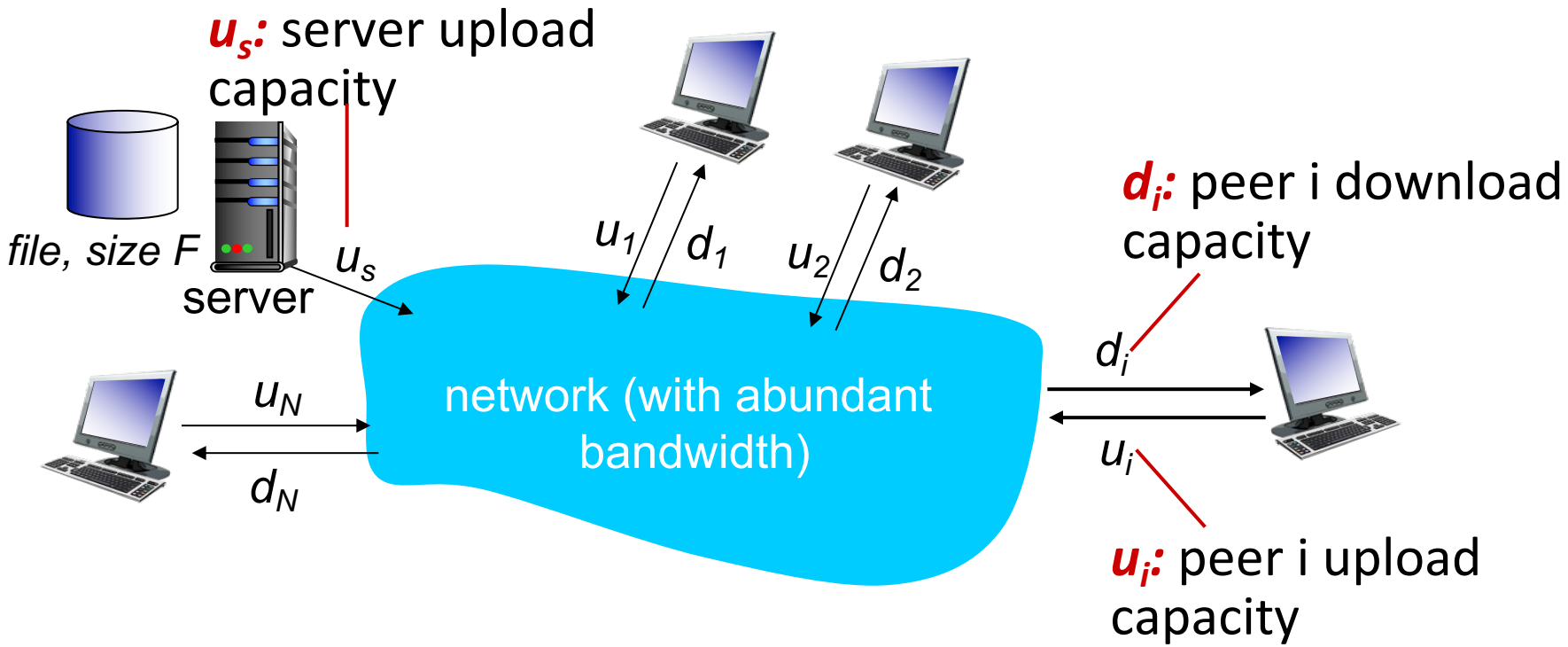
- examples: HTTP, IMAP, FTP



File distribution and network paradigms

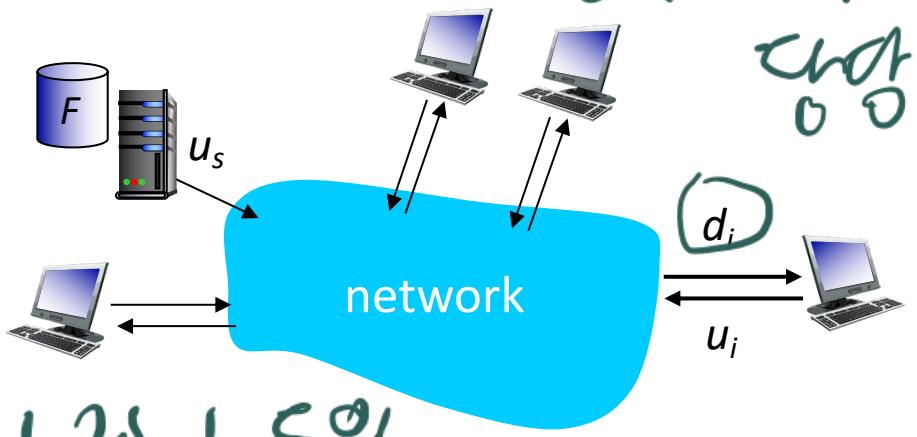
Q: how much time to distribute file (size *F*) from one server to *N peers*?

- peer upload/download capacity is limited resource



File distribution time: client-server

- *server transmission:* must sequentially send (upload) *N* file copies:
  - time to send one copy: F/u_s
  - time to send *N* copies: NF/u_s
- *client:* each client must download file copy
  - d_{min} = min client download rate
  - min client download time: F/d_{min}

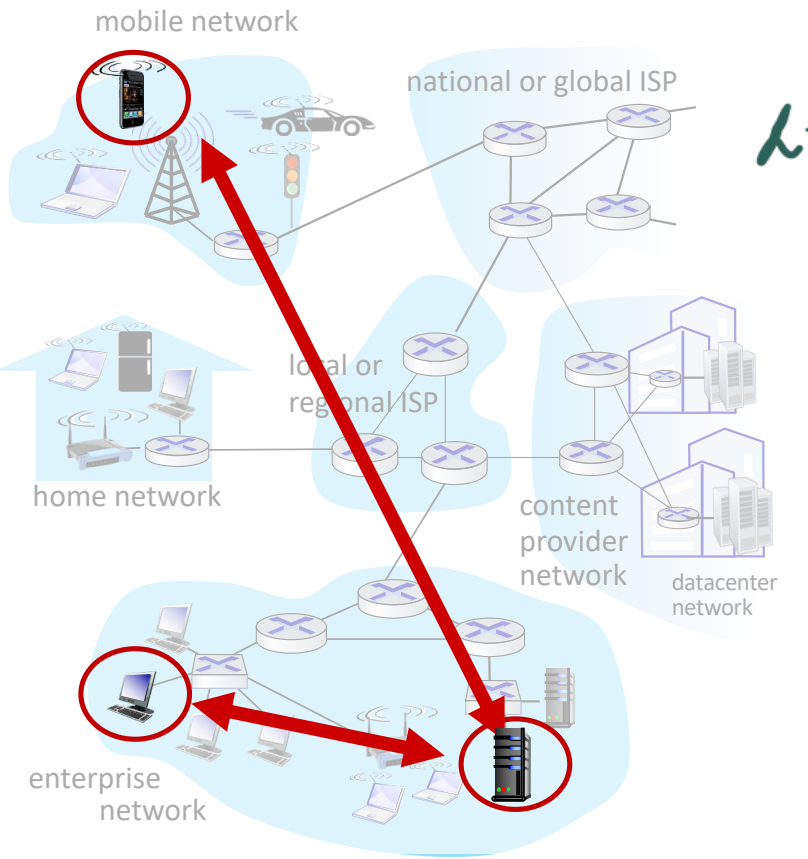


time to distribute F to N clients using client-server approach

$$D_{c\text{-}s} \geq max\{NF/u_s, F/d_{min}\}$$

increases linearly in N

Client-server model:
Root causes of many problems



network performance

값싼 시간대

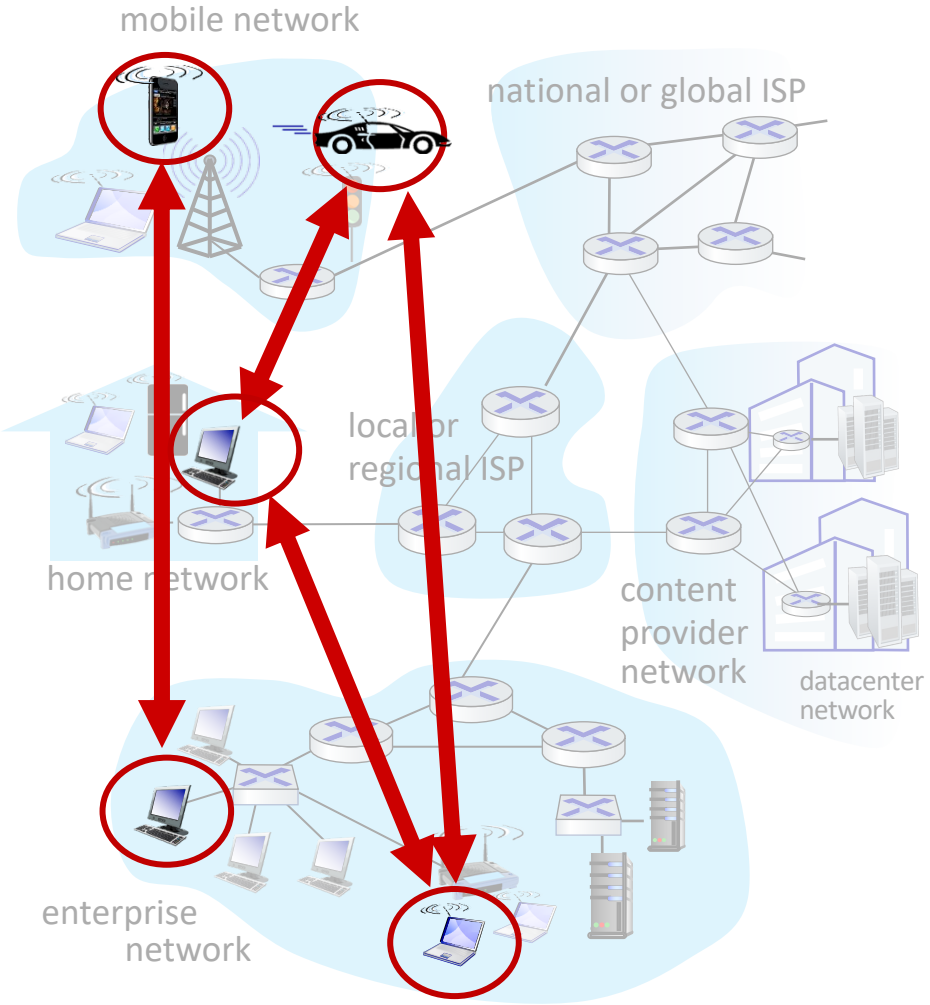
⋮

Getting rid of content servers?

- Old: a few powerful servers and the rest are dummy terminals
- New: large numbers of always-connected server-grade machines

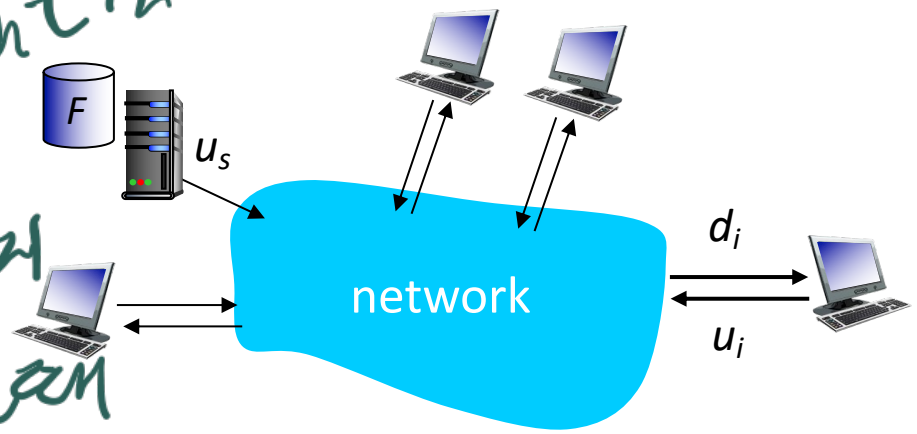
Peer-peer architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
  - *self scalability* – new peers bring new service capacity, as well as new service demands
- peers may be intermittently connected and change IP addresses
  - complex management
- example: P2P file sharing



File distribution time: P2P

- *server transmission:* must upload at least one copy:
  - time to send one copy: F/u_s
- *client:* each client must download file copy
  - ~~min~~ client download time: F/d_{min}
- *clients:* as aggregate must download NF bits
  - max upload rate (limiting max download rate) is $u_s + \Sigma u_i$



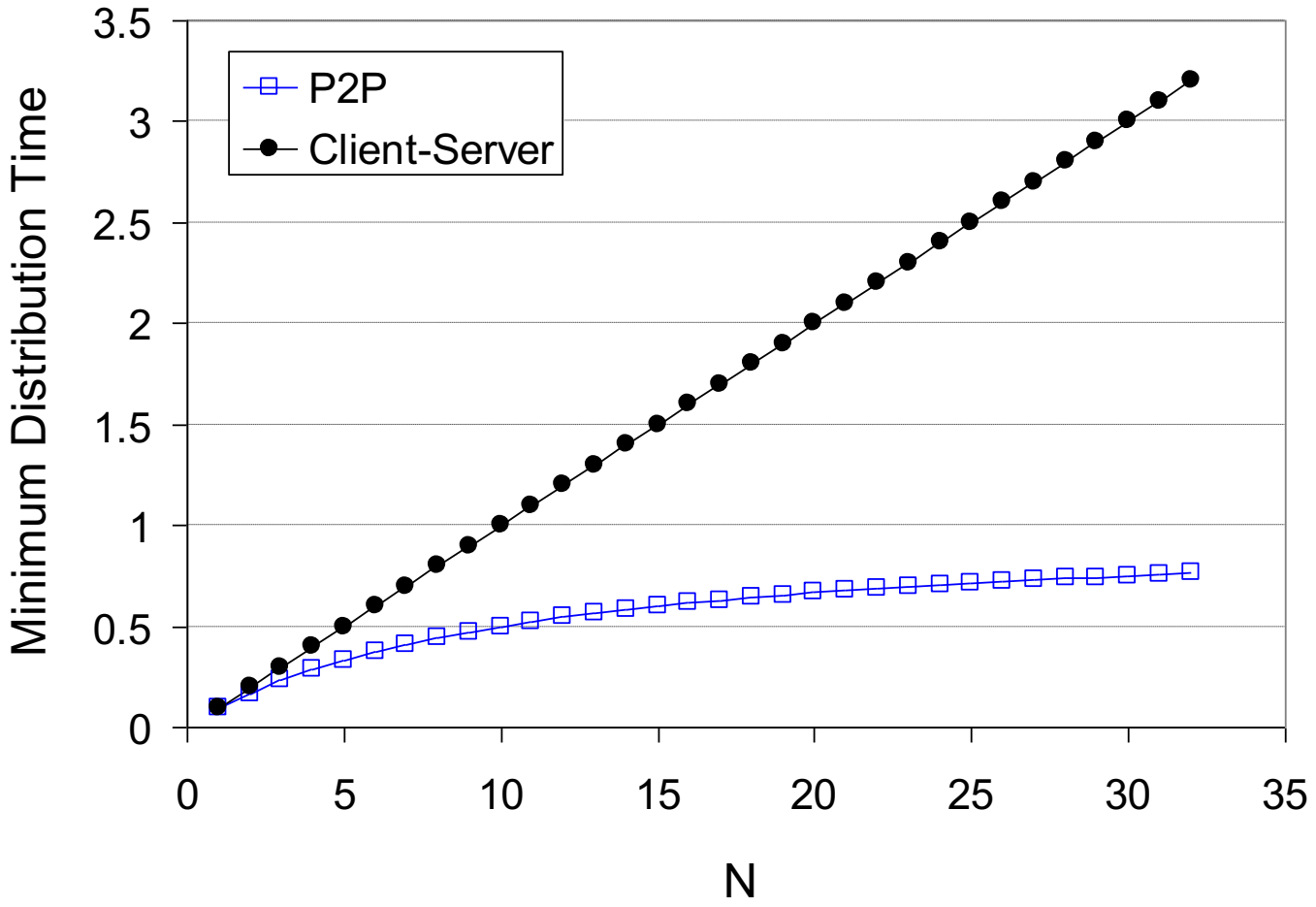
time to distribute F to N clients using P2P approach

$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in N …?

Client-server vs. P2P: example

client upload rate = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$



Real-world P2P applications?

bit-torrent

blockchain

- We'll focus on P2P today (because we won't for the rest of the semester)

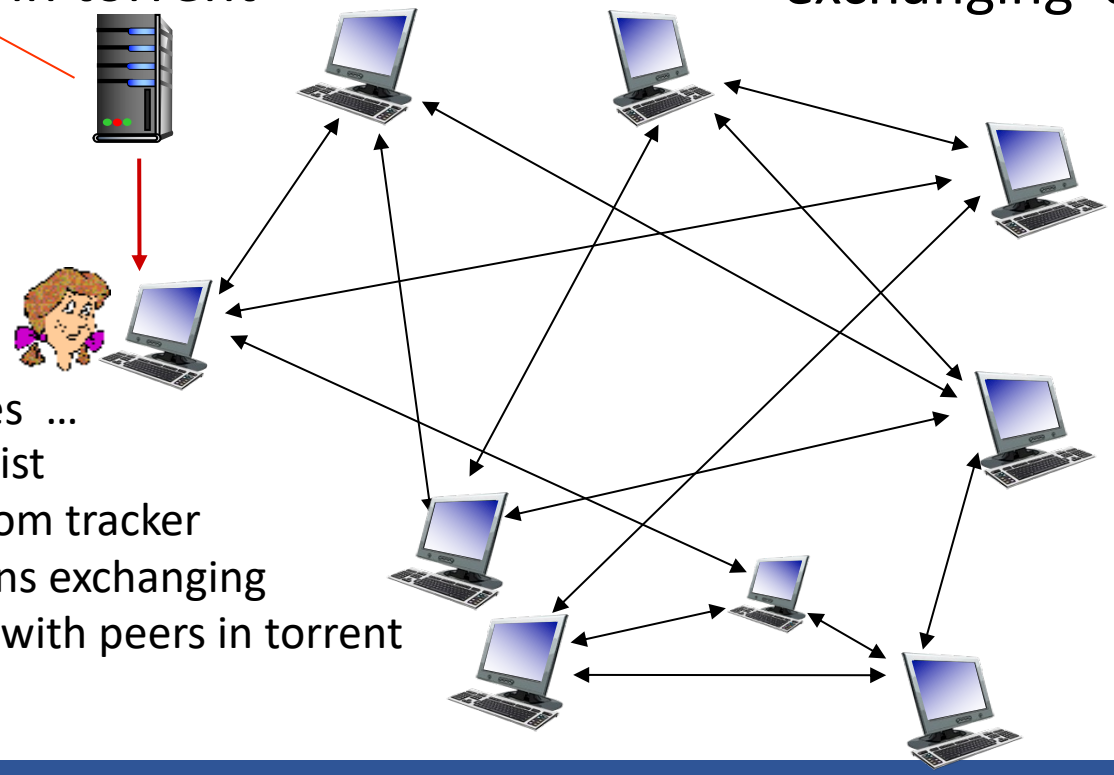
P2P file distribution: BitTorrent

↳ unit of the file sharing

- file divided into 256Kb chunks
- peers in torrent send/receive file chunks

tracker: tracks peers participating in torrent

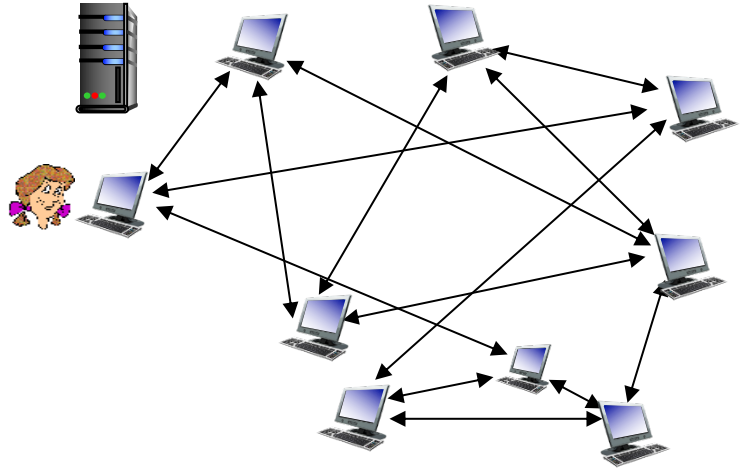
torrent: group of peers exchanging chunks of a file



Alice arrives …
… obtains list
of peers from tracker
… and begins exchanging
file chunks with peers in torrent

P2P file distribution: BitTorrent

- peer joining torrent:
  - has no chunks, but will accumulate them over time from other peers
  - registers with tracker to get list of peers, connects to subset of peers ("neighbors")



- while downloading, peer uploads chunks to other peers
- peer may change peers with whom it exchanges chunks
- *churn:* peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

client끼리 chunk를 교환하여 전체 파일 획득

BitTorrent: requesting, sending file chunks

Requesting chunks:

- at any given time, different peers have different subsets of file chunks
- periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first

Sending chunks: tit-for-tat

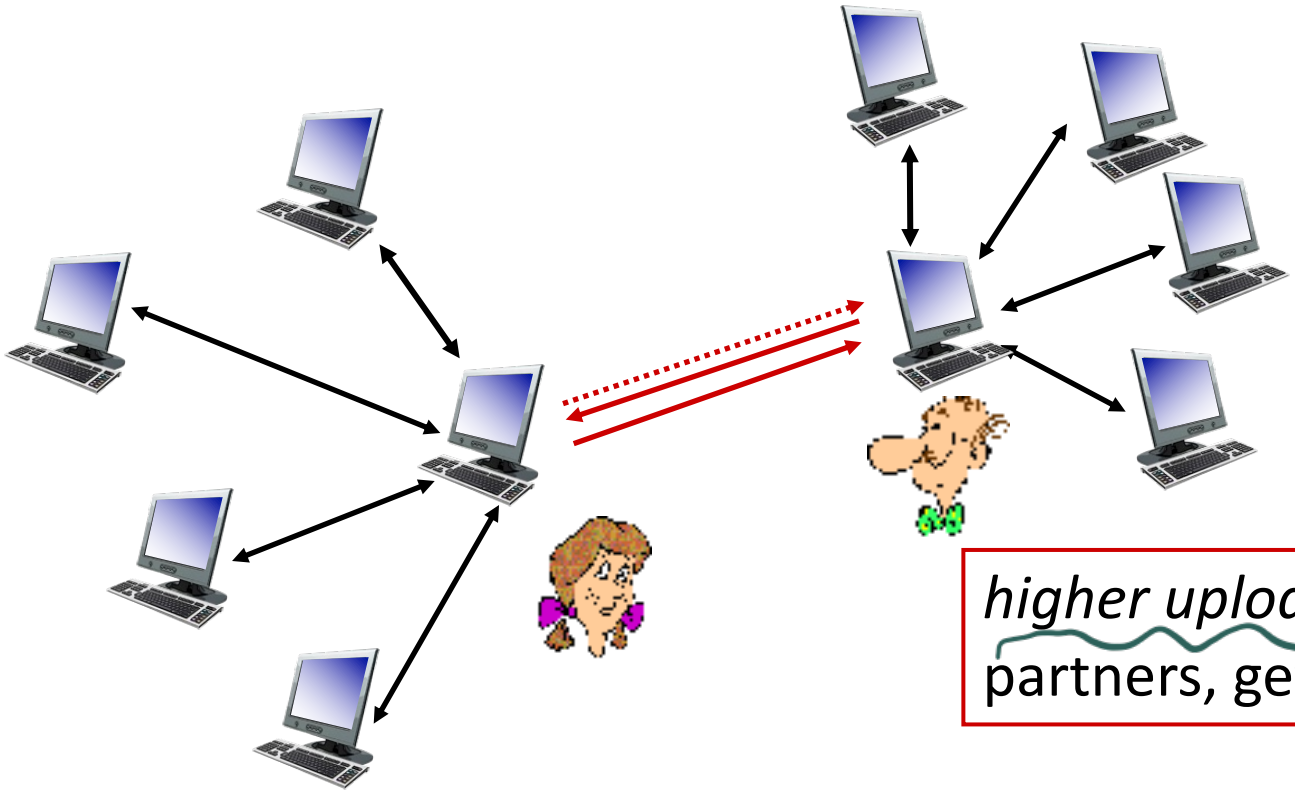
- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
  - newly chosen peer may join top 4

BitTorrent: tit-for-tat

(1) Alice “optimistically unchokes” Bob

(2) Alice becomes one of Bob’s top-four providers; Bob reciprocates

(3) Bob becomes one of Alice’s top-four providers



higher upload rate: find better trading partners, get file faster !

BitTorrent:
Fine for sharing files. Can we do more with it?

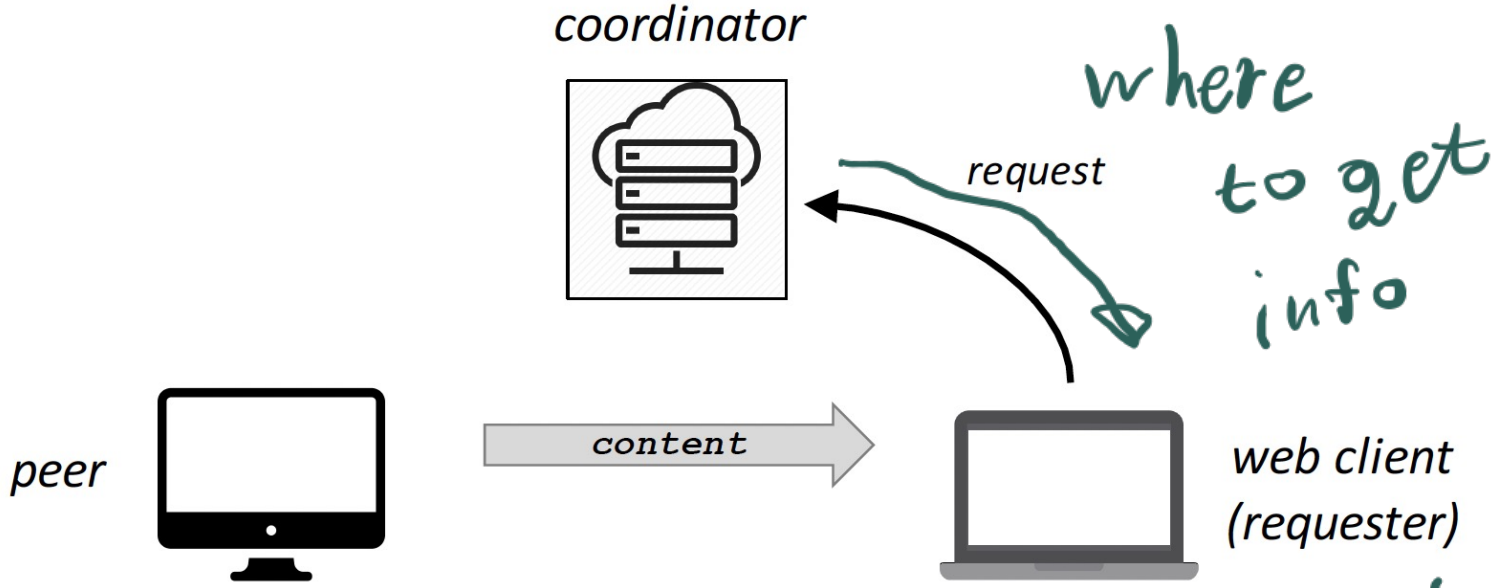
- How about “Decentralized web over BitTorrent”?

Real-world P2P applications?

- BitTorrent
- Decentralized webs
- Blockchain
- …

Decentralized (not entirely though) Webs

- Peer-assisted CDNs

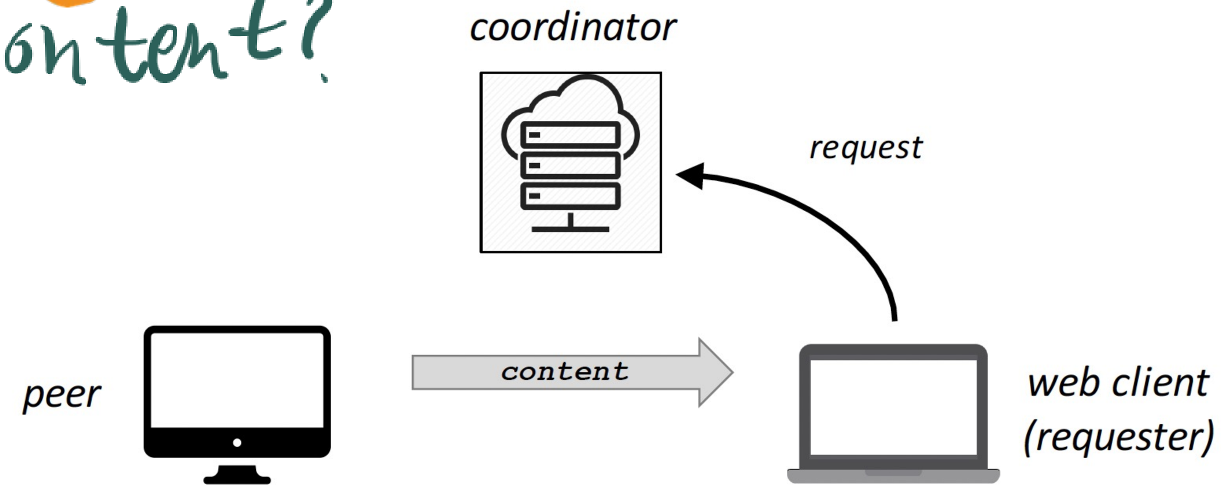


- Several start-ups: Peer5, Swarmify, Greta
- Big ones: Akamai NetSession

서버는 정보가 존재하는 peer들 리스트, peer가 직접 전달

Solved challenges

how to distribute content?



how to locate the content?

검색 속도를 높일 방법은? → 지리적으로 가까운 망에 접근

자료의 신뢰성? → photographic hash tag

Unsolved challenges

본인 resource를 희생하며 자원을 공유할 동기(인센티브)

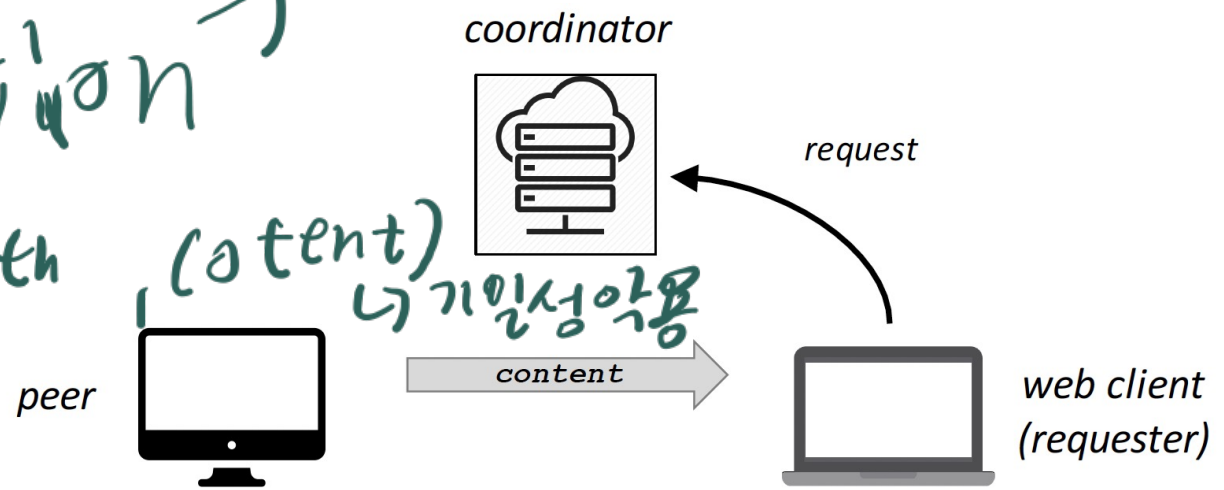
incentivization

legal (bandwidth, latent)

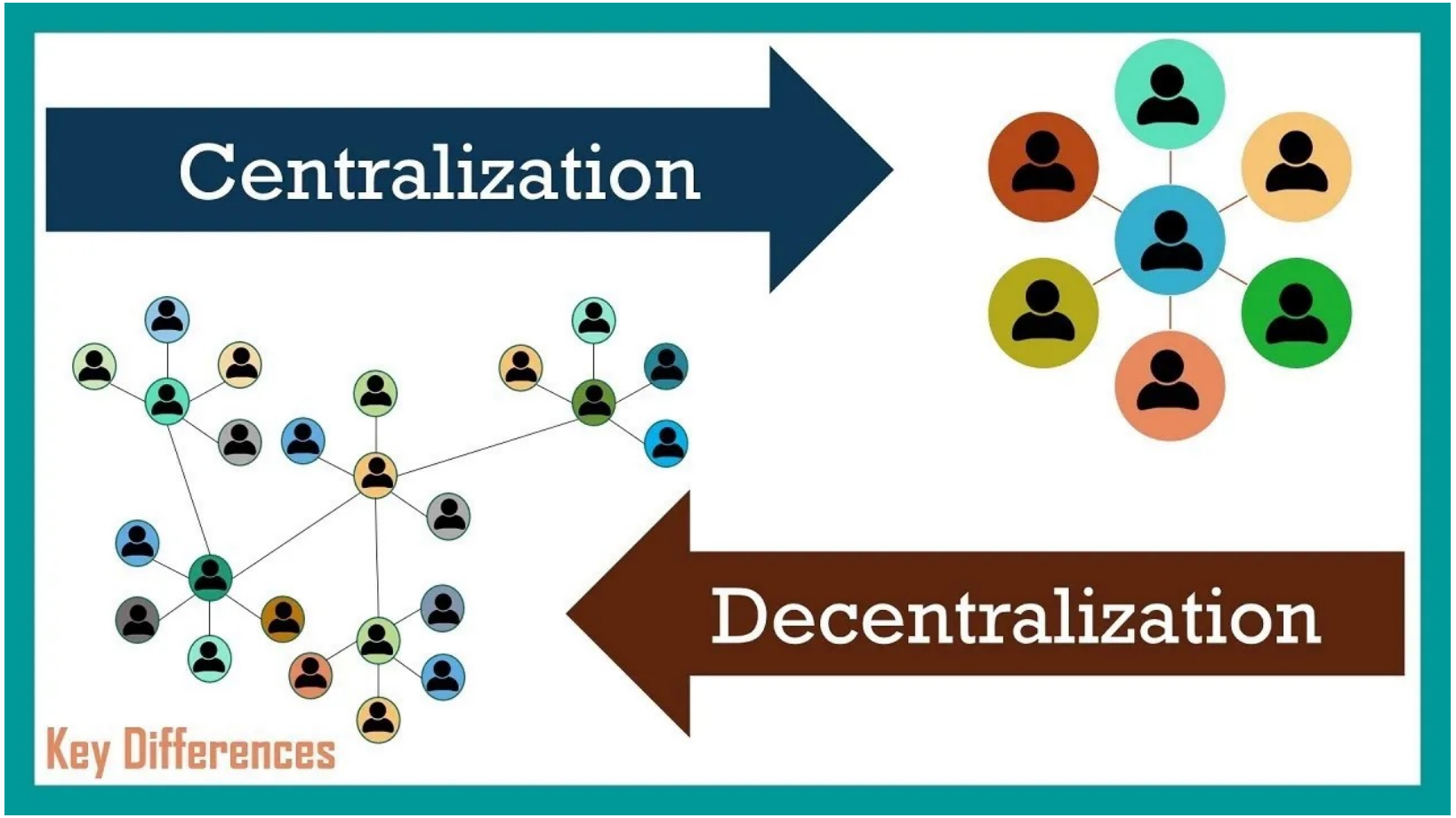
privacy

ethical concerns

⋮



Centralization vs. decentralization



(source: https://medium.com/@matt_61320/)

Real-world P2P applications?

- BitTorrent
- Decentralized webs
- Blockchain
- …


BLOCK CHAIN
TECHNOLOGY

Too many jargons…

- Bitcoin
- Mining (burning electricity)
- Ethereum
- Smart Contract
- DeFi
- NFT
- Web3
- …
- Decentralized Web?

Defining blockchain ***without*** any jargons

- "**Agreeing** on **something** among a **group of parties** who **care about it**"

Defining blockchain ***with*** some jargons

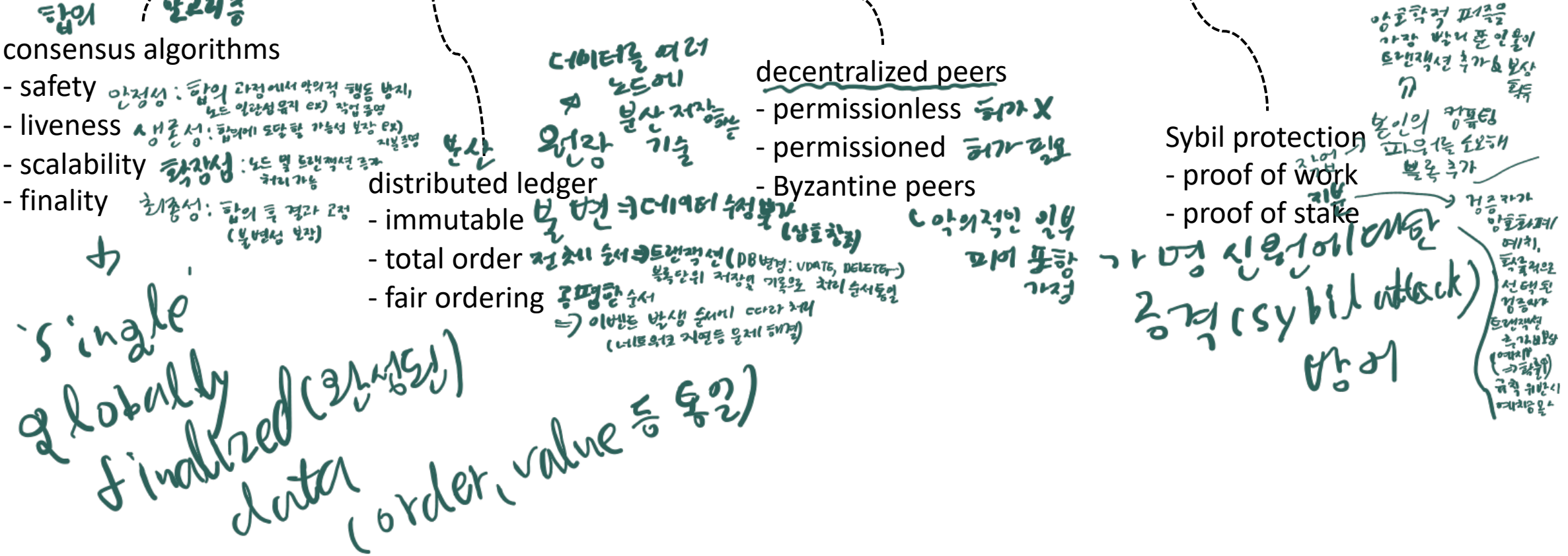
- "**Agreeing** on **something** among a **group of parties** who **care about it**"

consensus algorithms
- safety
- liveness
- scalability
- finality

distributed ledger
- immutable
- total order
- fair ordering

decentralized peers
- permissionless
- permissioned
- Byzantine peers

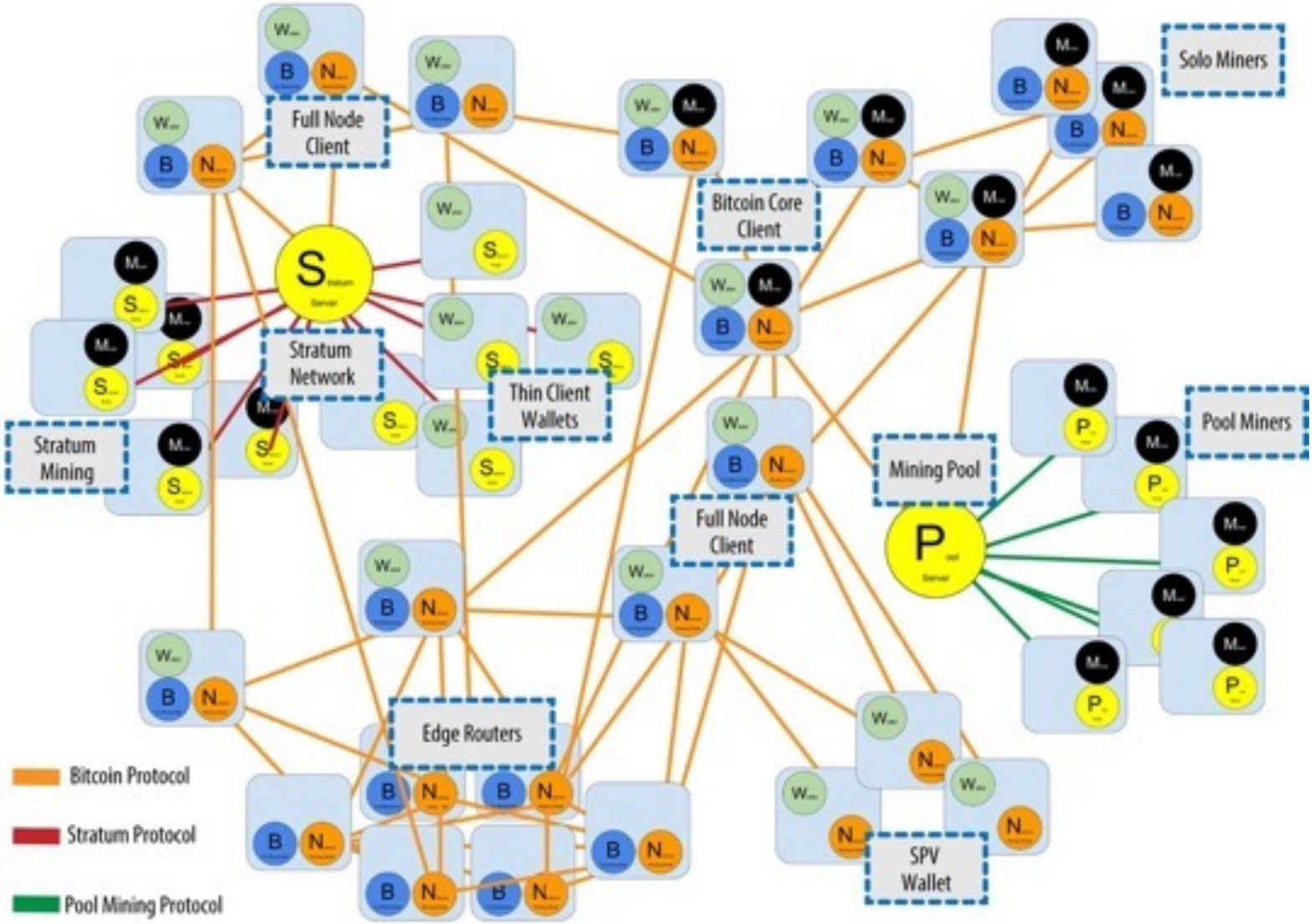
Sybil protection
- proof of work
- proof of stake



Blockchain p2p networks

- Decentralized, permissionless peer-to-peer broadcast network used to announce new transactions and proposed blocks
- Requirements
  - low latency
    - 10 minute block creation time handles latency issues
  - robust against malicious miners
    - e.g., censor transactions
- Network topology and discovery
  - Bitcoin: 8 outgoing, 117 incoming connections
- Communication protocol
  - Flooding new blocks and pending transactions

Extended Bitcoin network



Bitcoin network

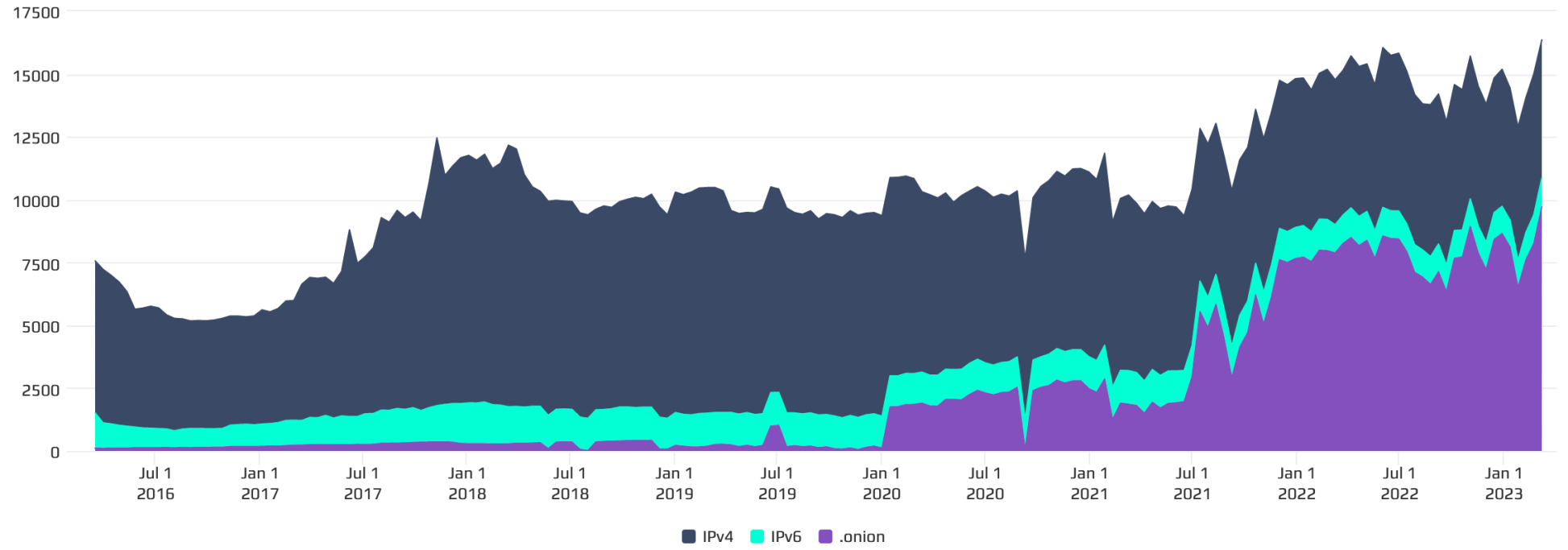
16359	10370	8793 ▲ 116.22%
Reachable nodes	Average	Since 7 years ago

NODES

Chart shows the number of reachable Bitcoin nodes during the last 7 years. Series can be enabled or disabled from the legend to view the chart for specific networks.

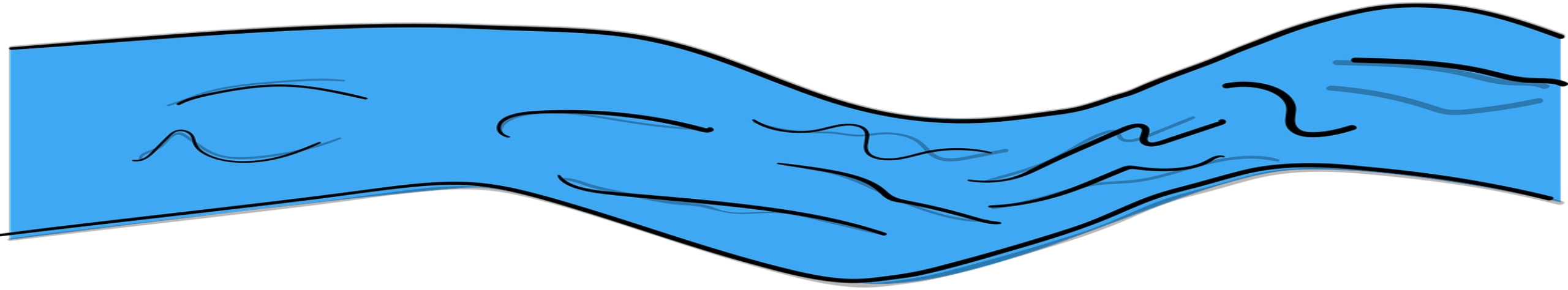
24h 90d 1y 7y

Lo 5176 Hi 16359 Avg 10370 Last 16359 nodes



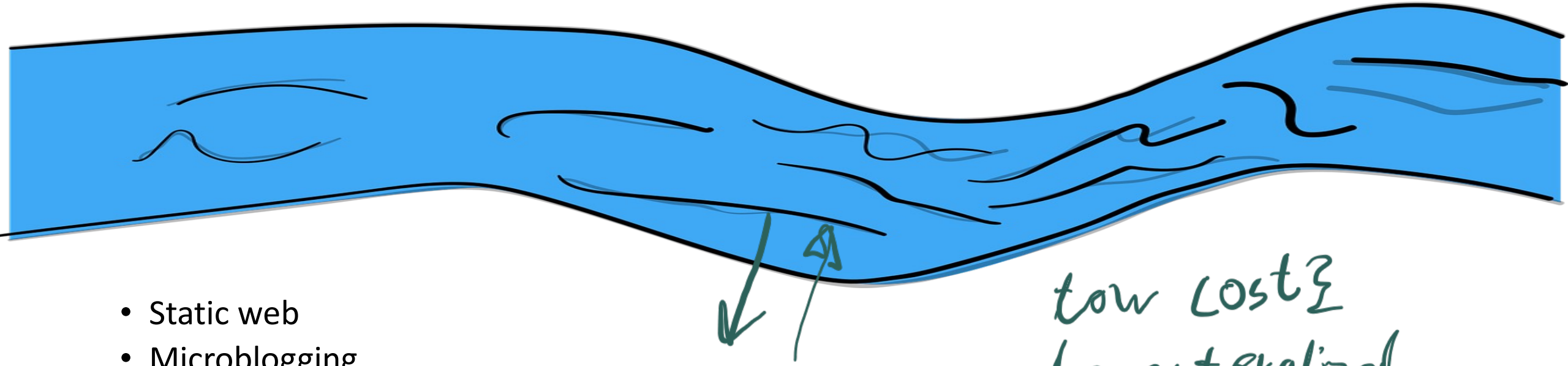
Let’s Imagine “Ideal” Blockchain

- A publicly visible, verifiable, totally-ordered, fully-decentralized ledger
- Handling requests instantly with infinite bandwidth at no cost



- **Can we build a new decentralized web on this “ideal” blockchain?** 🤔

Building a New Web



- Static web
- Microblogging
- Facebook
  - Social graph, chats
- ...

front-end

low cost로
decentralized
streamon
검증해 정보 등급

- How would this new web change the ecosystem?

Back to Reality: Existing Blockchains

- Reality check:
  - 7 transactions per second, 1-hour for finizing a transaction, burning too much fossil fuel?

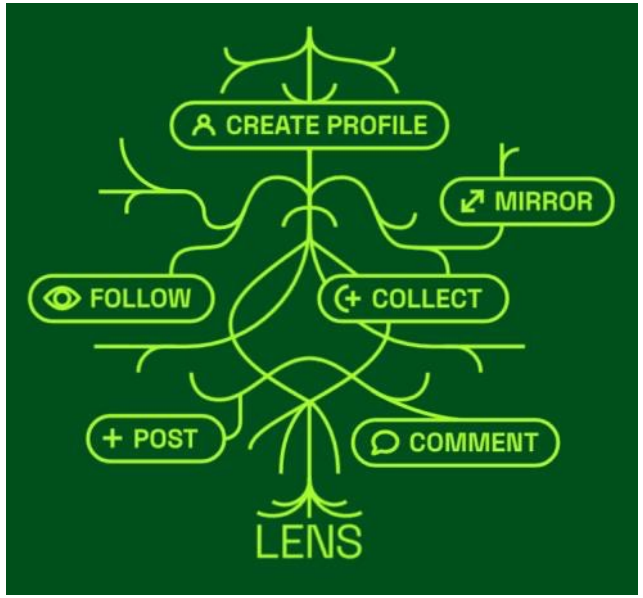
- Getting better (very quickly)
  - >100k transactions per second
  - <1sec finalization
  - near zero cost operation

- Technologies behind them
  - new distributed system techniques (e.g., DAG, proof of stake)
  - new cryptography (e.g., zero-knowledge proof)
  - …

Use Case: Lens Protocol

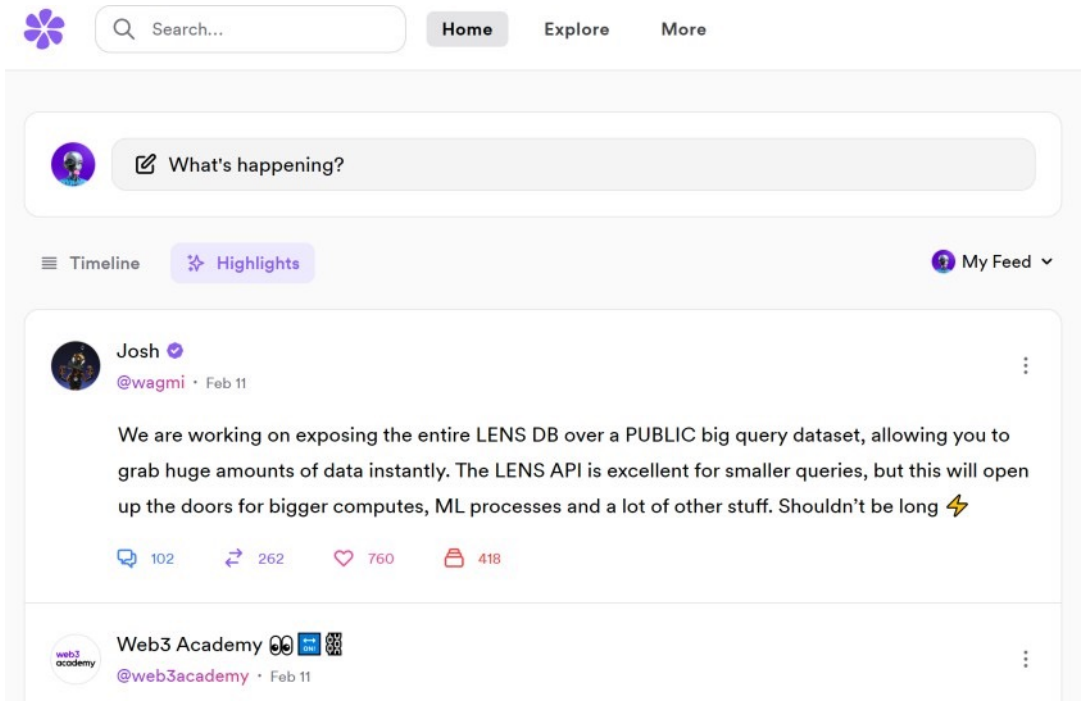
A user-owned, open social graph
that any application can plug into

social graph
data elements



https://www.lens.xyz/

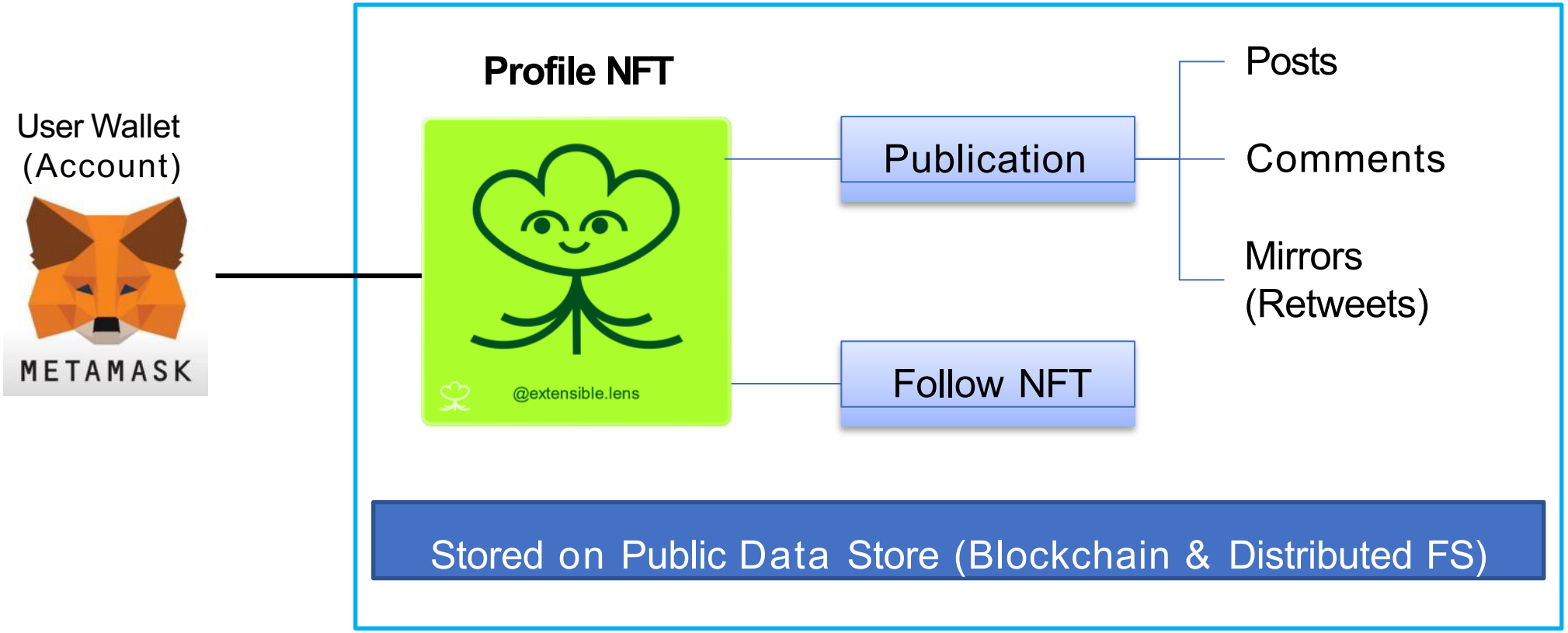
Lenster – Web3 twitter



https://lenster.xyz/

1. Data Ownership to Users

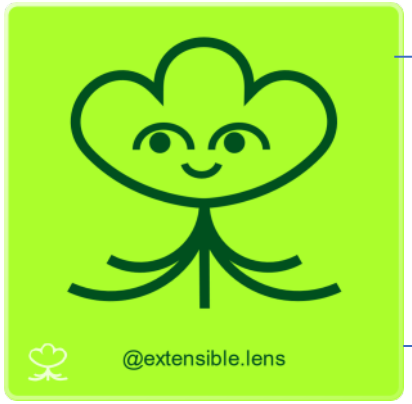
Profile NFT is a key element to owns your data and gives you control of your contents



2. Data as Digital Assets

Follow NFT, Collect, Profile NFT as a digital asset

Profile NFT



Buy & Sell
a Profile NFT
with all content

Publication

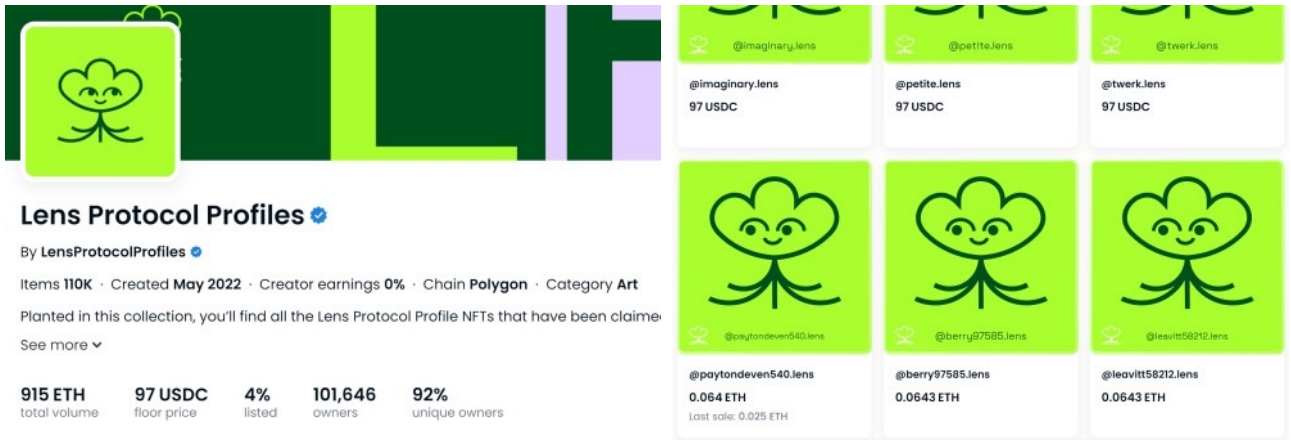
- Posts
- Comments
- Mirrors (Retweets)

Collect
Paid content

Follow NFT

Follow
with a fee

Profile NFT market on OpenSea



https://opensea.io/collection/lens-protocol-profiles

3. Communities as data governance actors

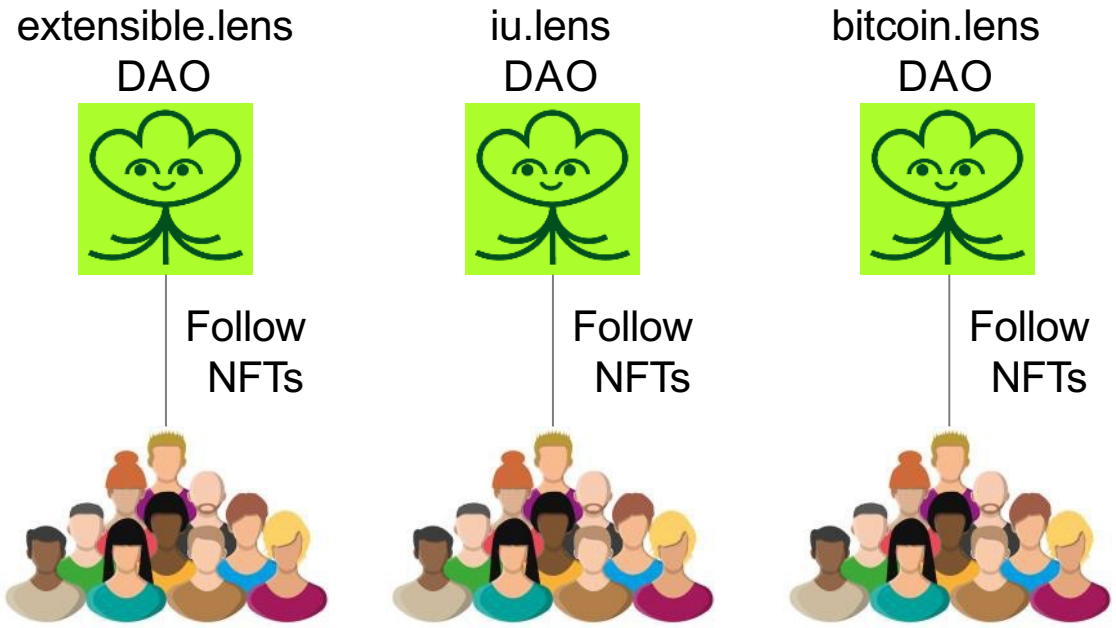
Governance for Lens Protocol

Community Multisig
by the Lens community

- Setting up Governance and Emergency Admin Addresses
- Setting Treasury Addresses and Fees
- Whitelisting Assets
- Moving the Lens Protocol System into a Publishing Paused or fully Paused state
- Whitelisting addresses to create profiles
- Whitelisting Follow, Collect, and Reference Modules
- Upgrading the Lens Protocol Hub Contract

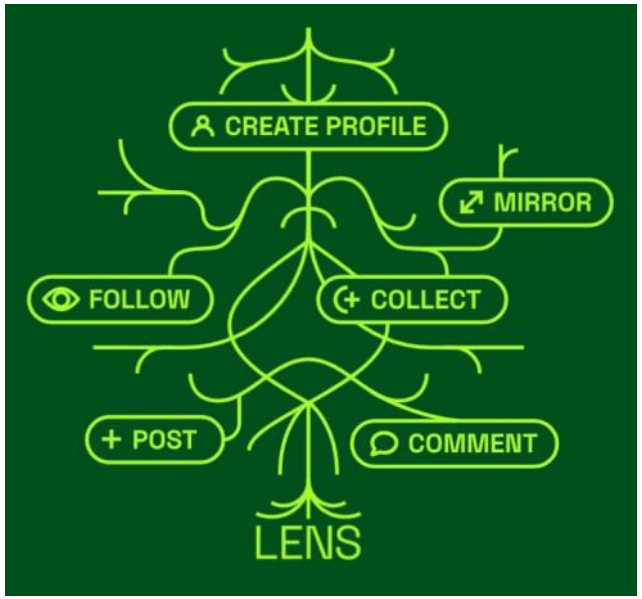
Governance for social communities

Built-in governance of Follow NFTs
to create a social DAO
(Decentralized Autonomous Organization)



4. Protocols as data utilization tools

Lens Protocol makes user's social graph data **a social graph protocol**
Applications utilize Lens Protocol to access user's social graph

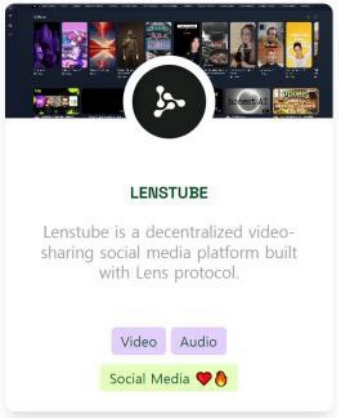


Lensverse

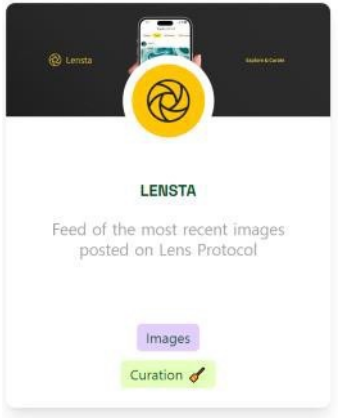
Hundreds of applications built on top of Lens Protocol

Web3 Twitter

Web3 Youtube

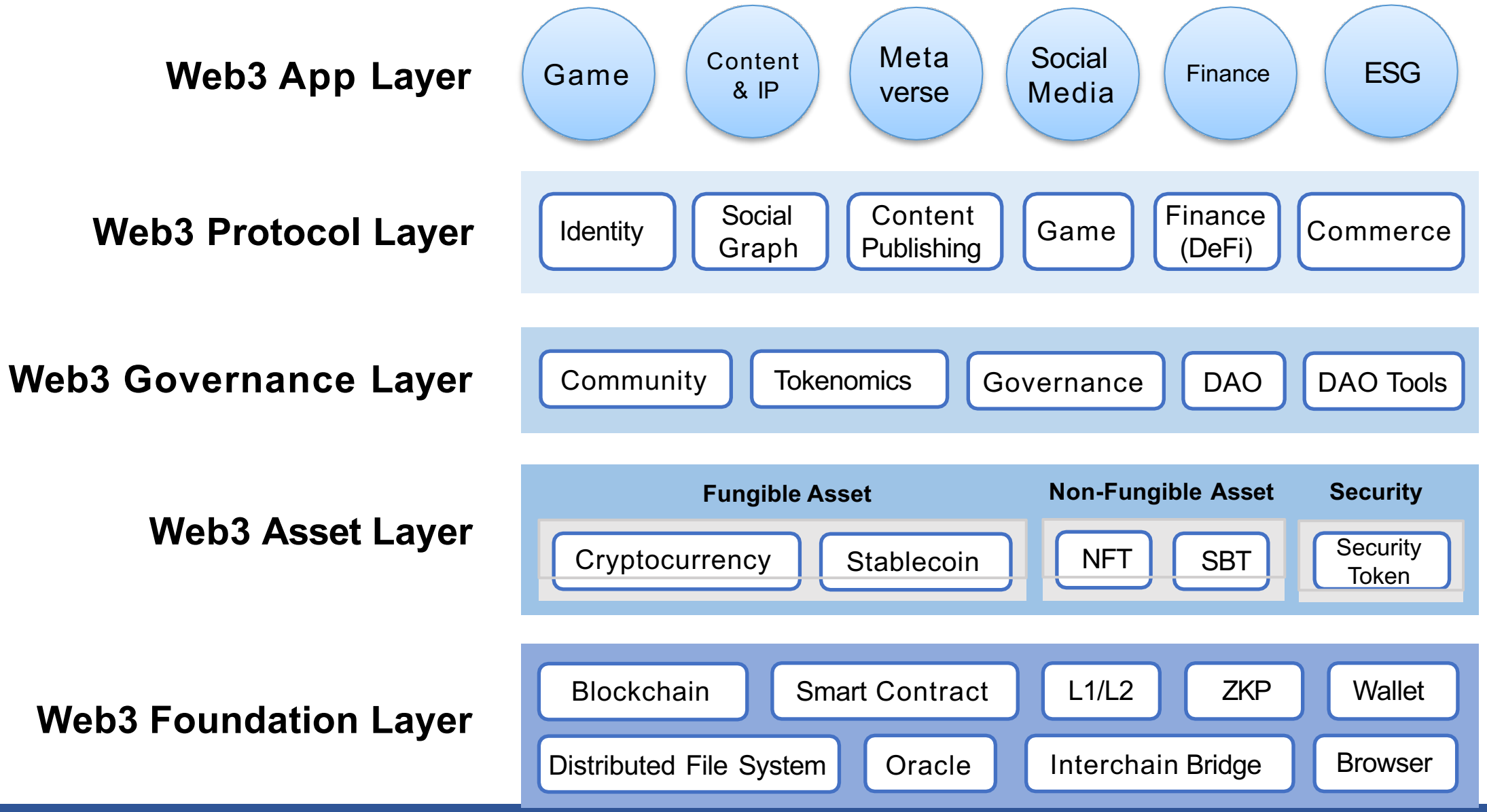


Web3 Instagram

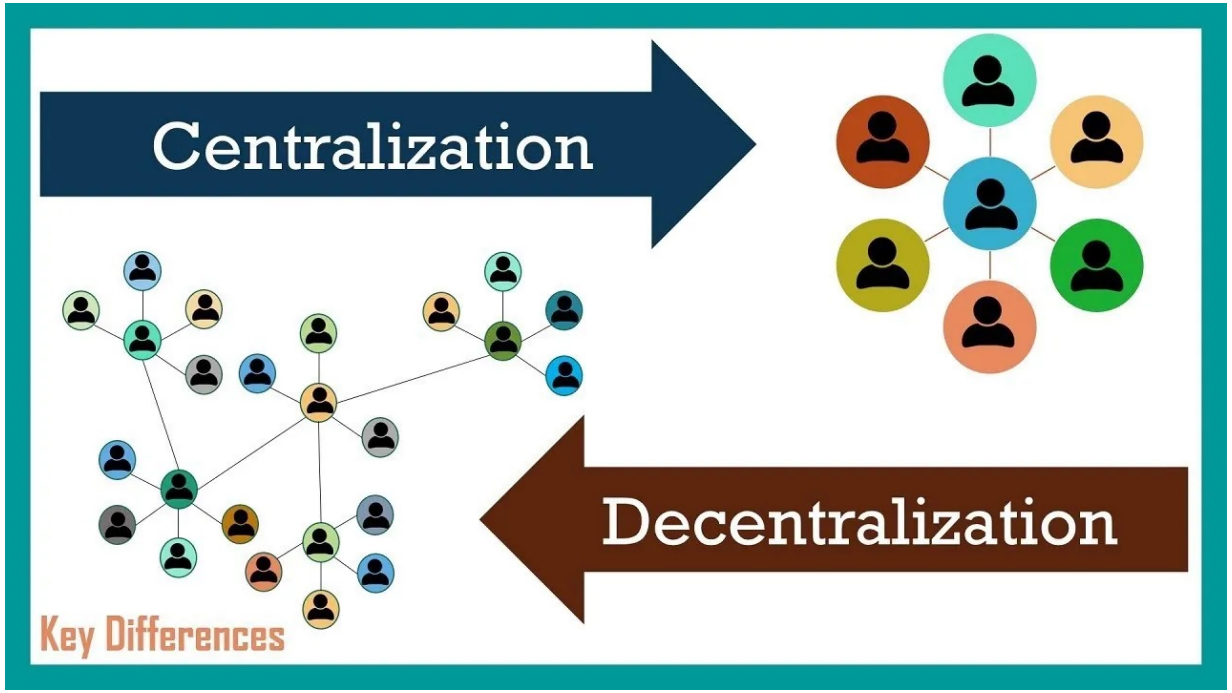


https://www.lens.xyz/apps

Web3 Stack (in the view of data)



Takeaways



- Can you imagine the next Internet or Web?

Next…

- *Chapter 2.6 Video Streaming and Content Distribution Networks*