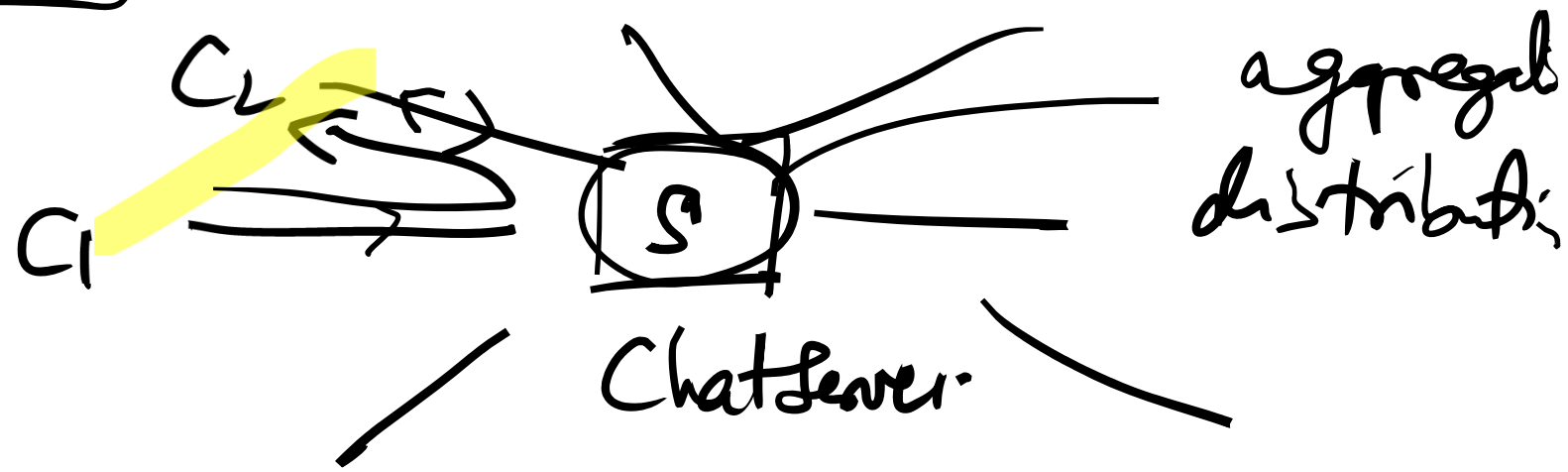
p2p architecture

peer to peer.

Client-Server:

mail.

chat appl (peers)

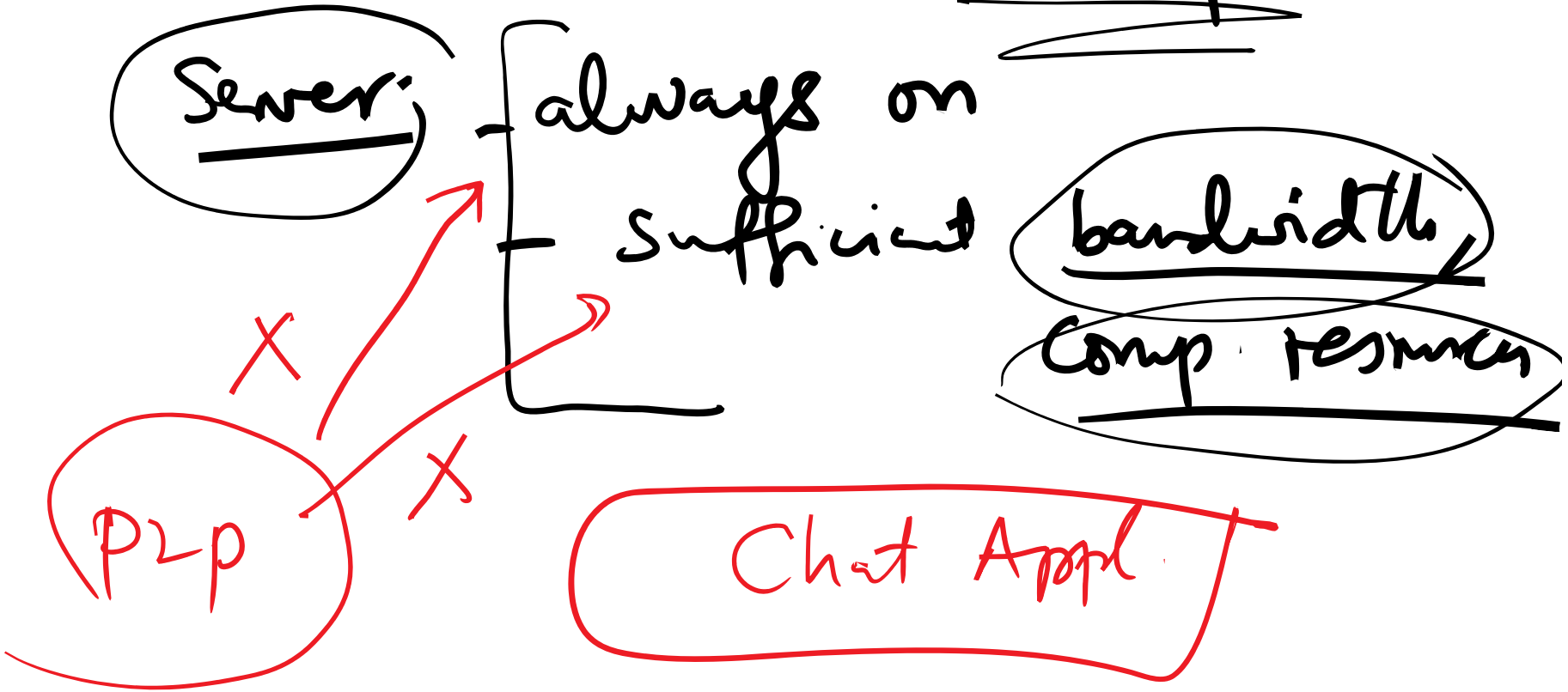


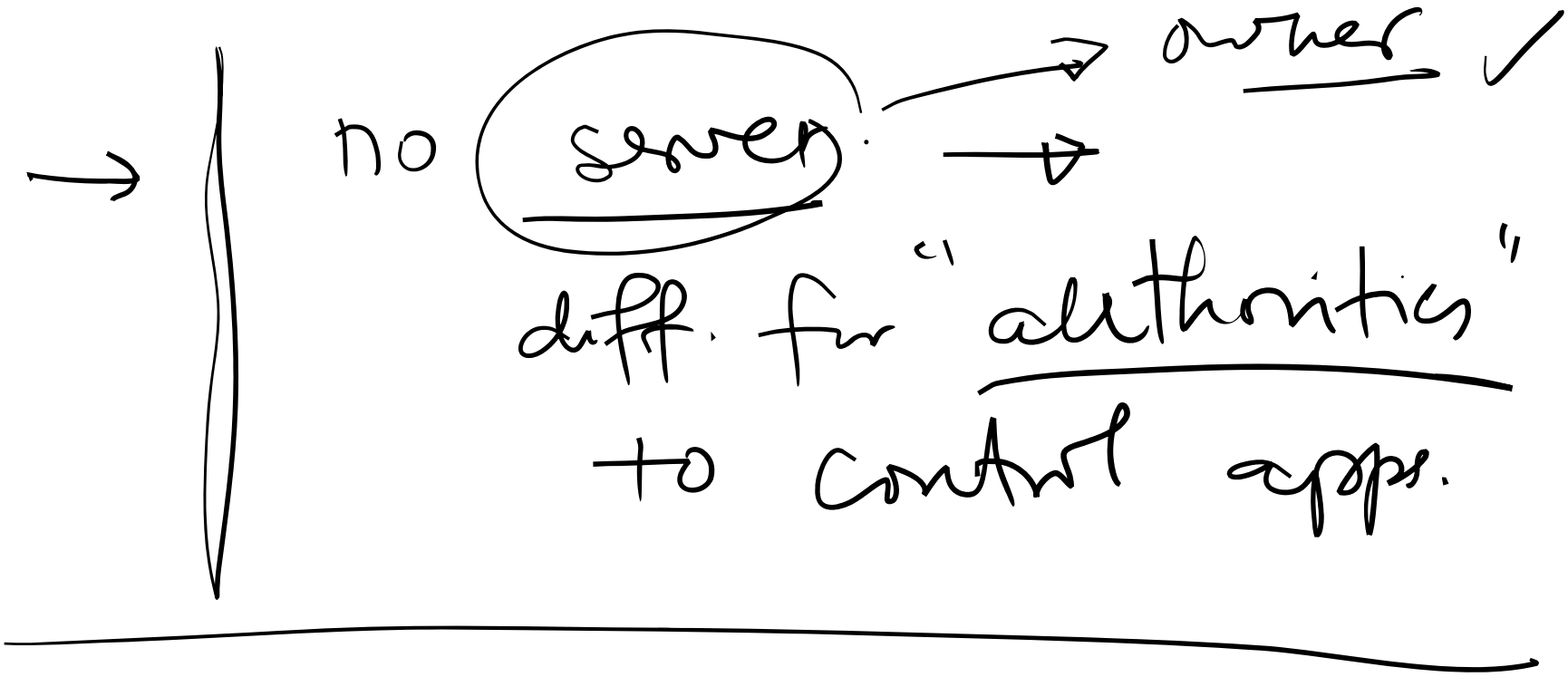
P2P:

→ absence of infrastructure (Server)

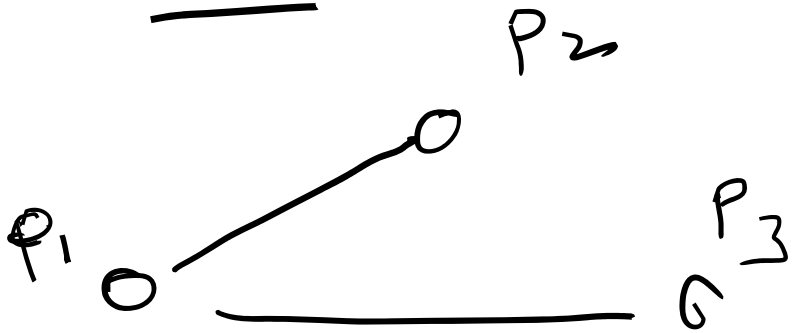
→ "distributed" → truly

Server:
- always on
- sufficient bandwidth, comp resources




no server
owner ✓
diff. for "authorities"
to control apps.

P2P.

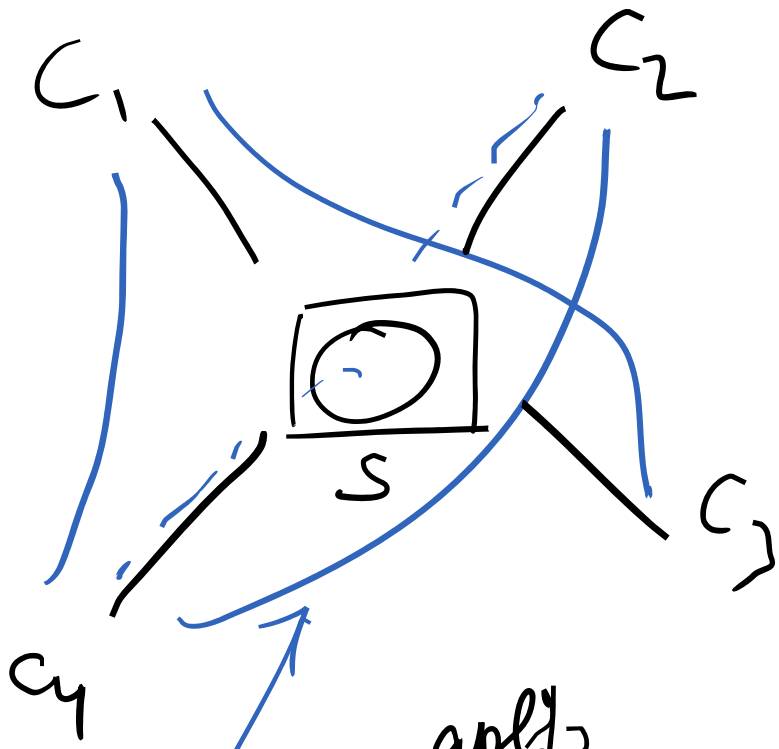


N = no of nodes

- Manually set up N/w
- Small # of nodes

N

Cl - Ser.



apply n/w.

logical links

P2P

1. How to form p2p network.

2. Resources are also distributed

3. Distributed Search Algo

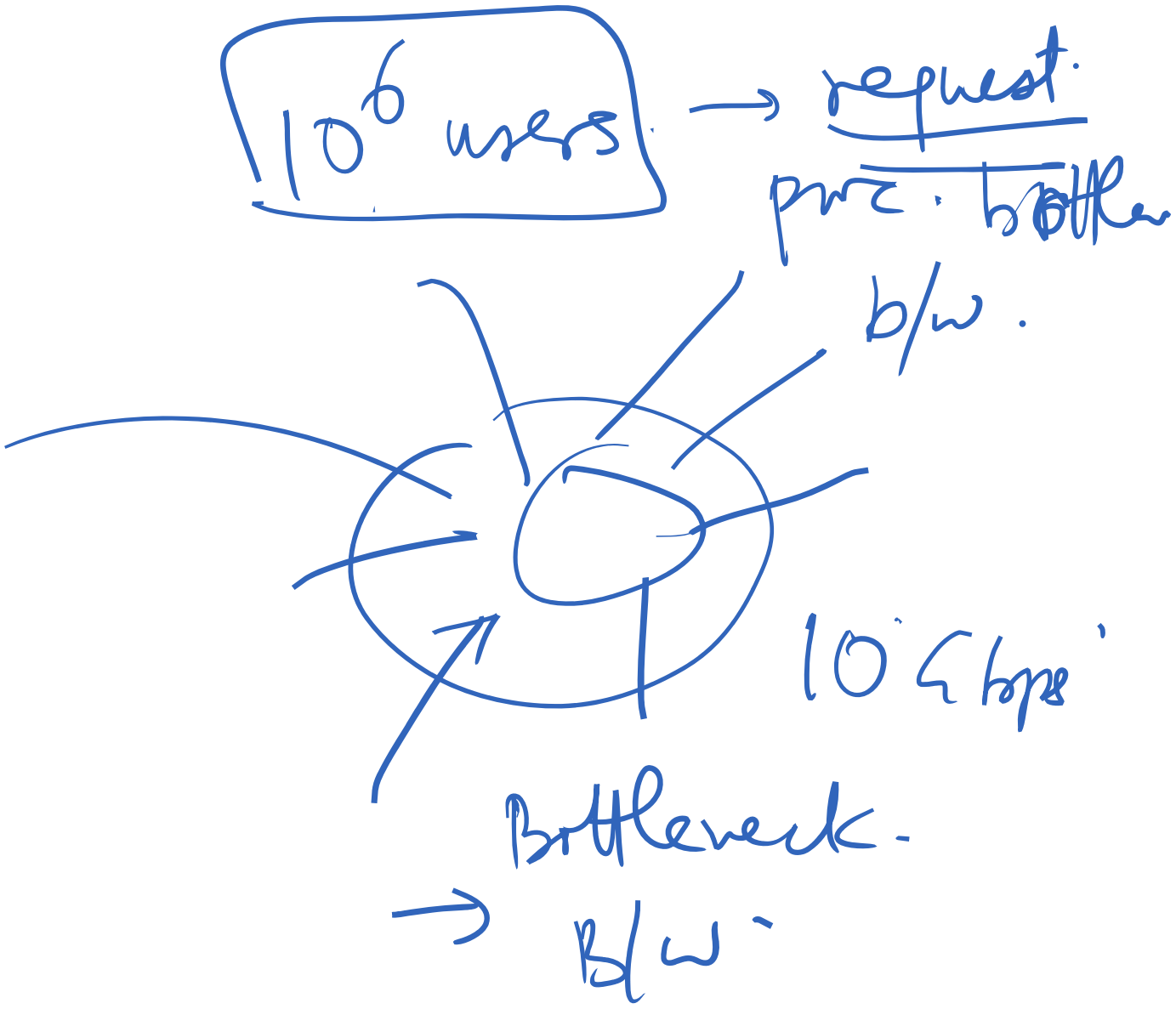
→ Brute force approach →

$n \rightarrow$ all nodes

expensive.

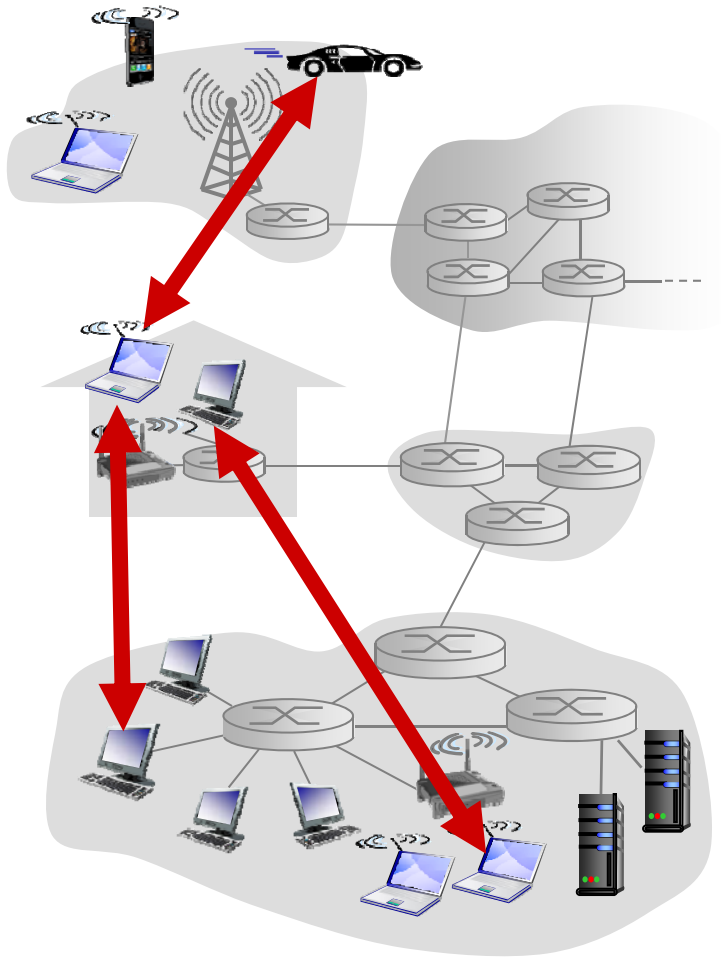
n^2

Scheme for distribution.


10^6 users
request
proc. bottleneck
b/w.
10 Gbps
Bottleneck.
B/w.

Pure P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses



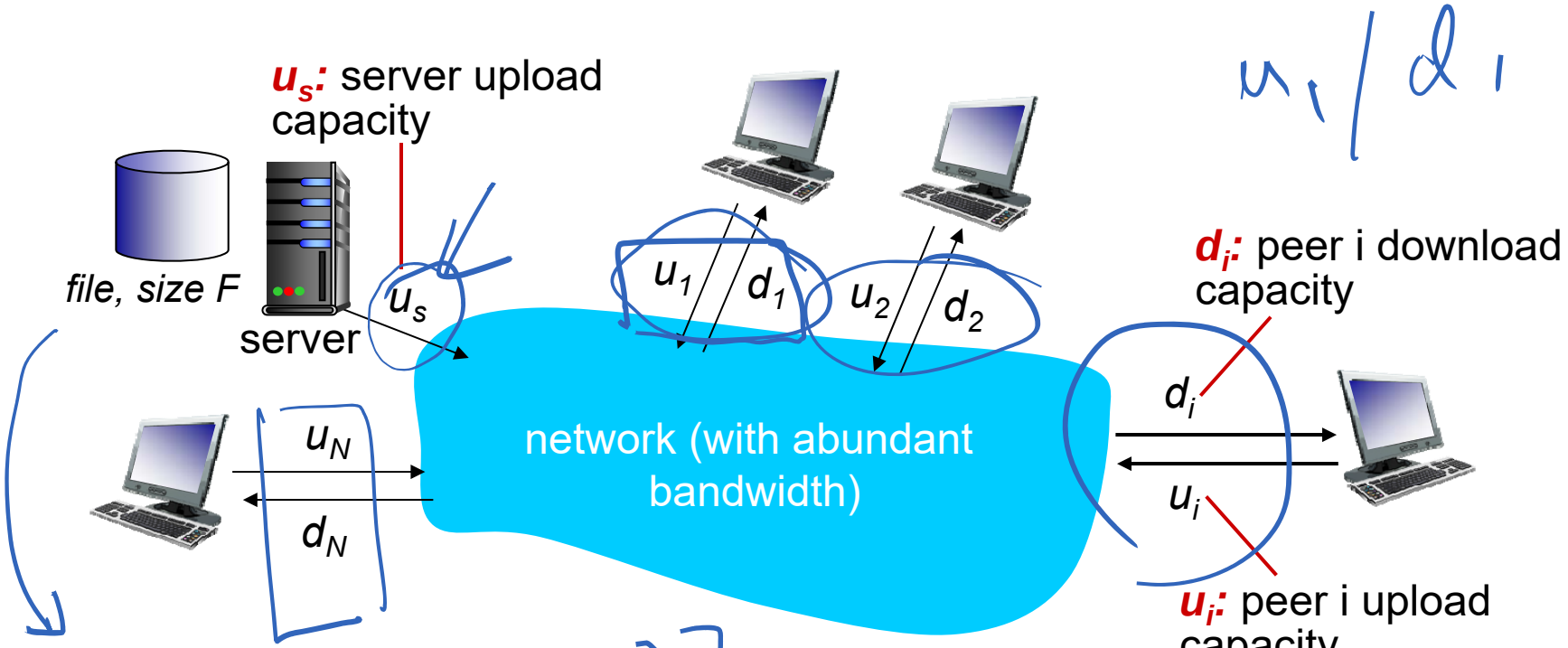
examples:

- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)

File distribution: client-server vs P2P

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

- peer upload/download capacity is limited resource

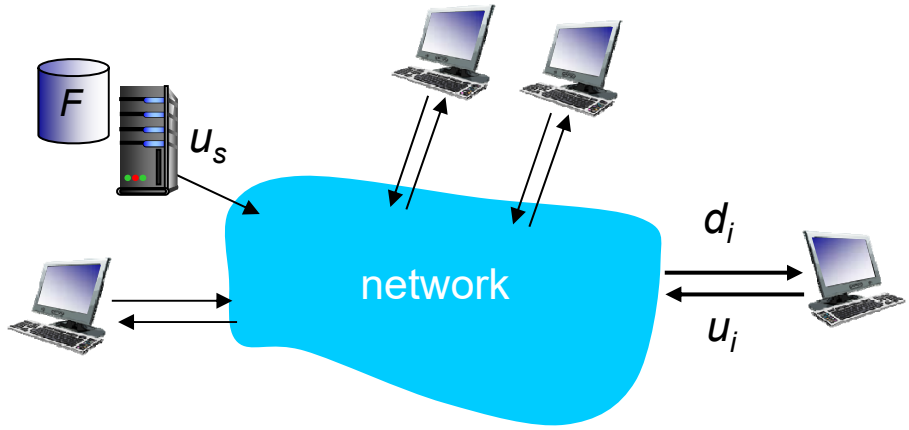


u_1 / d_1

$N * \left[\frac{B(\text{size})}{u_s}\right]$ = Total Time.

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}



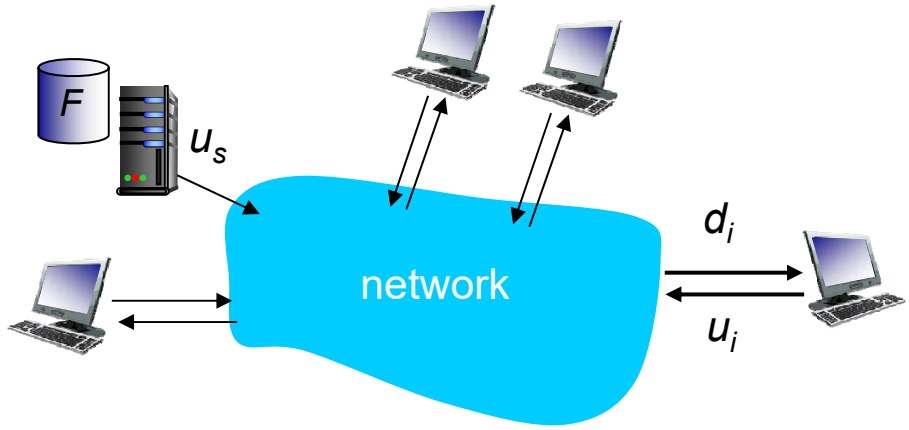
time to distribute F to N clients using client-server approach

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

increases linearly in N

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

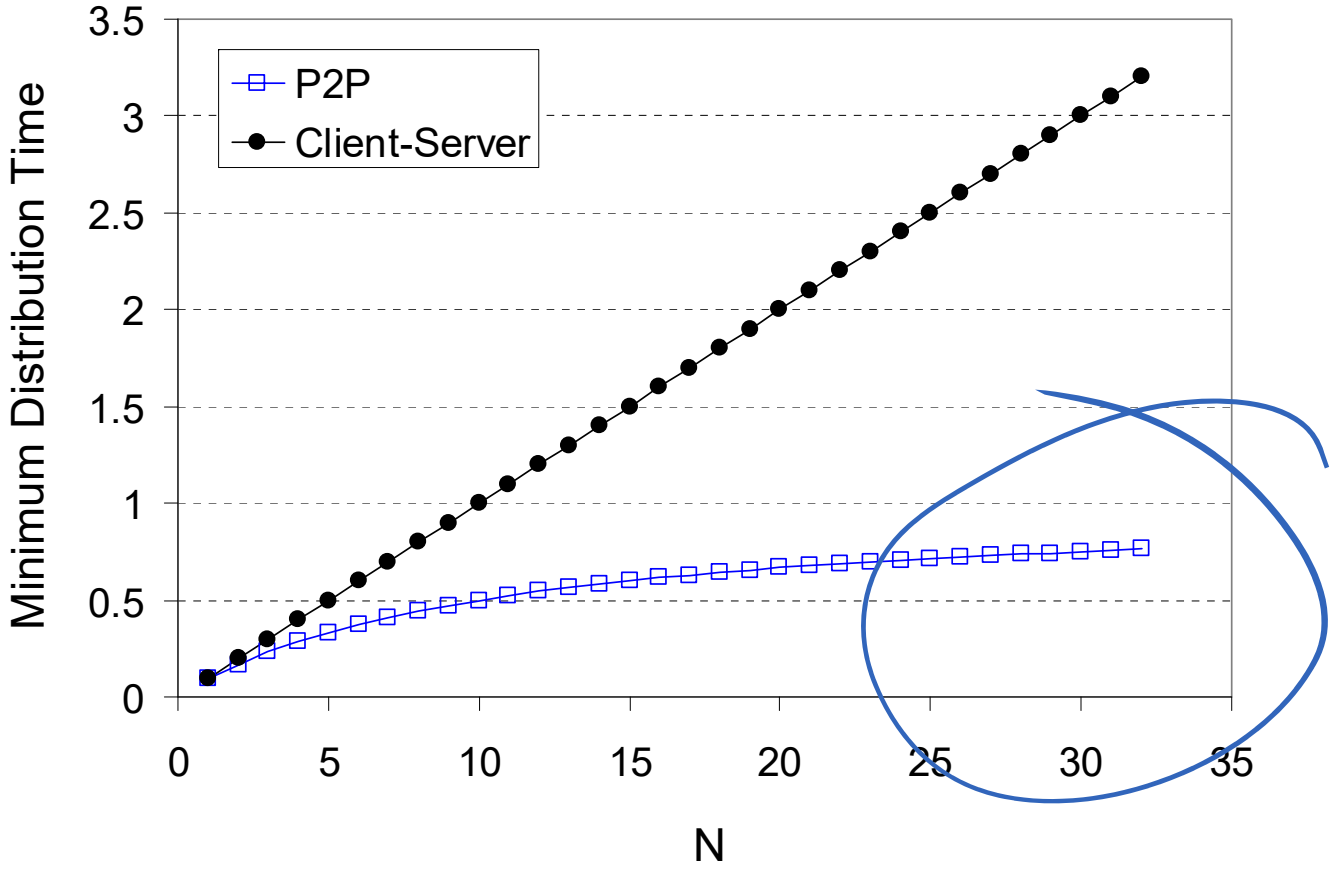
$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in N …

… but so does this, as each peer brings service capacity

Client-server vs. P2P: example

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

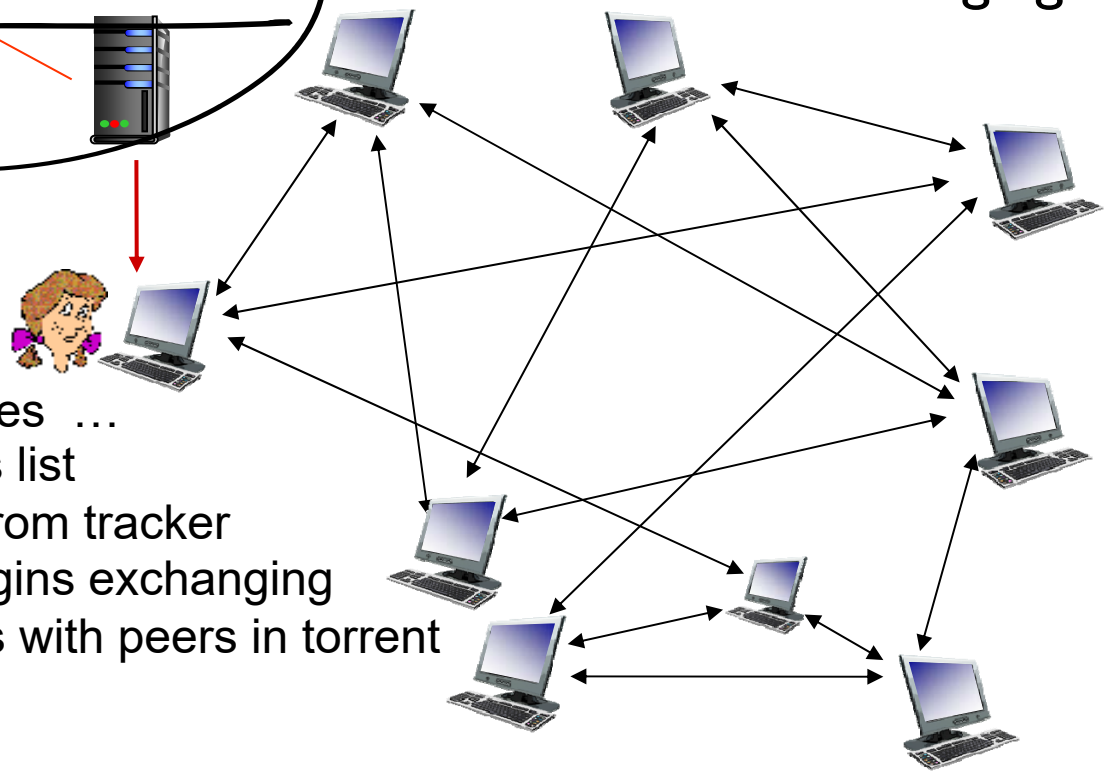


P2P file distribution: BitTorrent

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

tracker: tracks peers participating in torrent

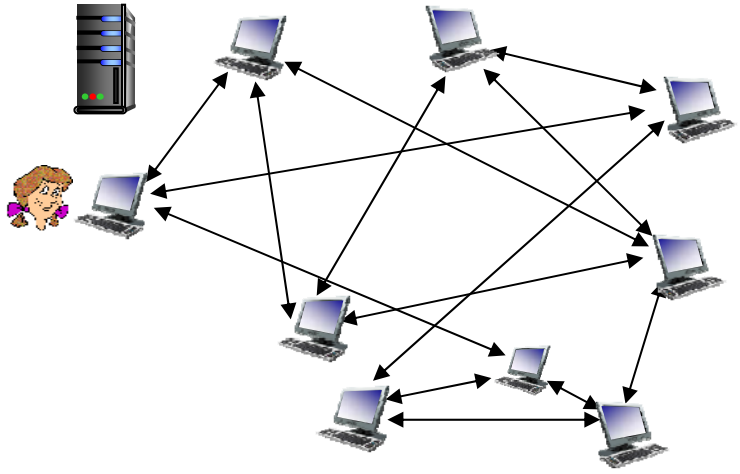
torrent: group of peers exchanging chunks of a file



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

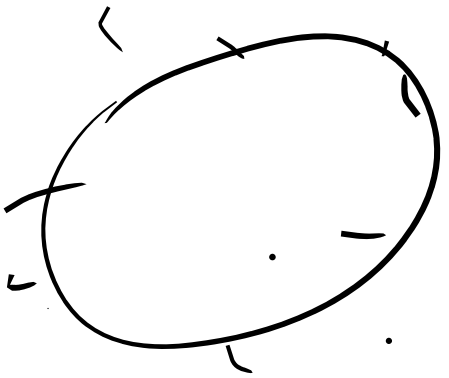
Network Formulation

Assignment

protocol.

P2P → "Chat"

Cafetria. → with

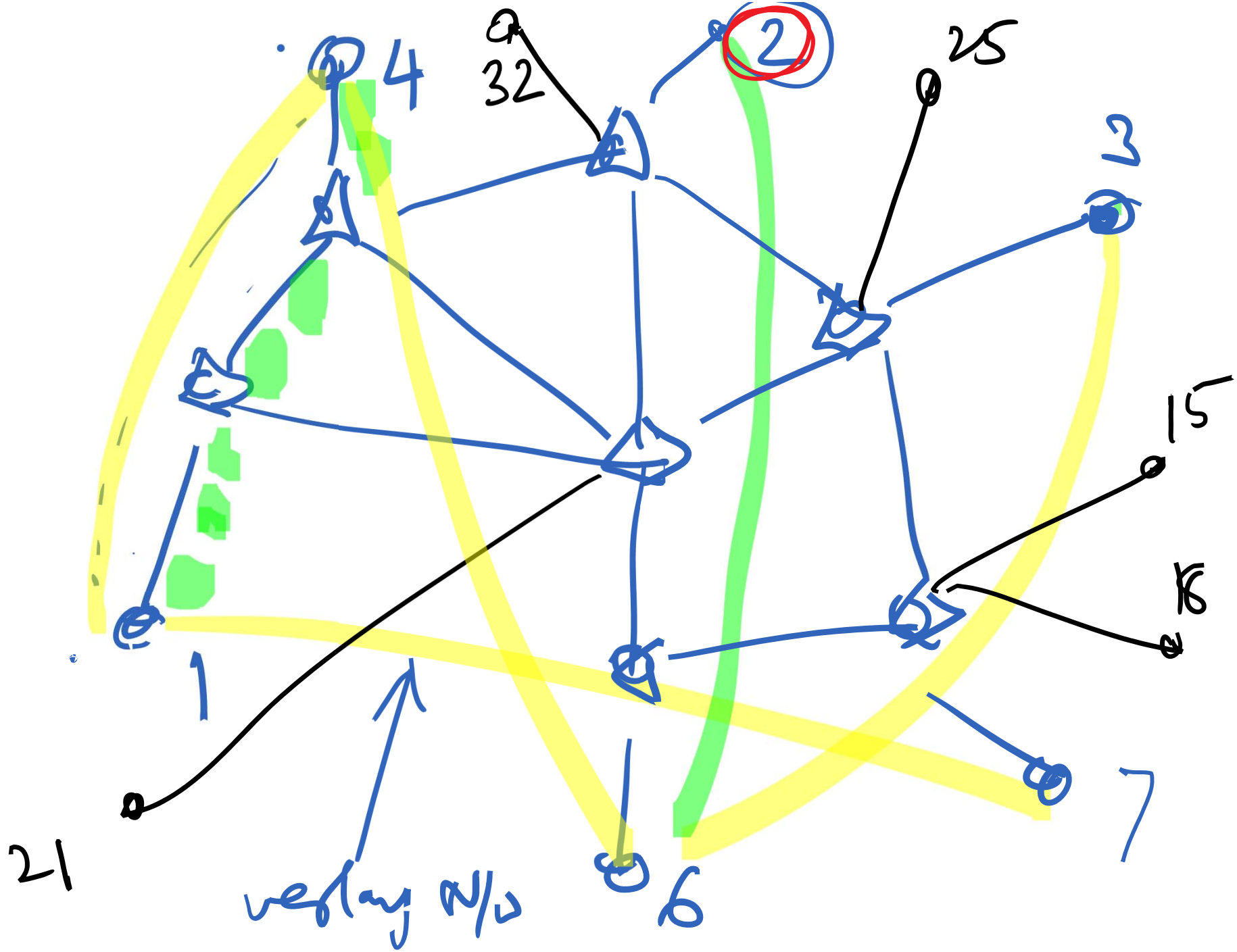


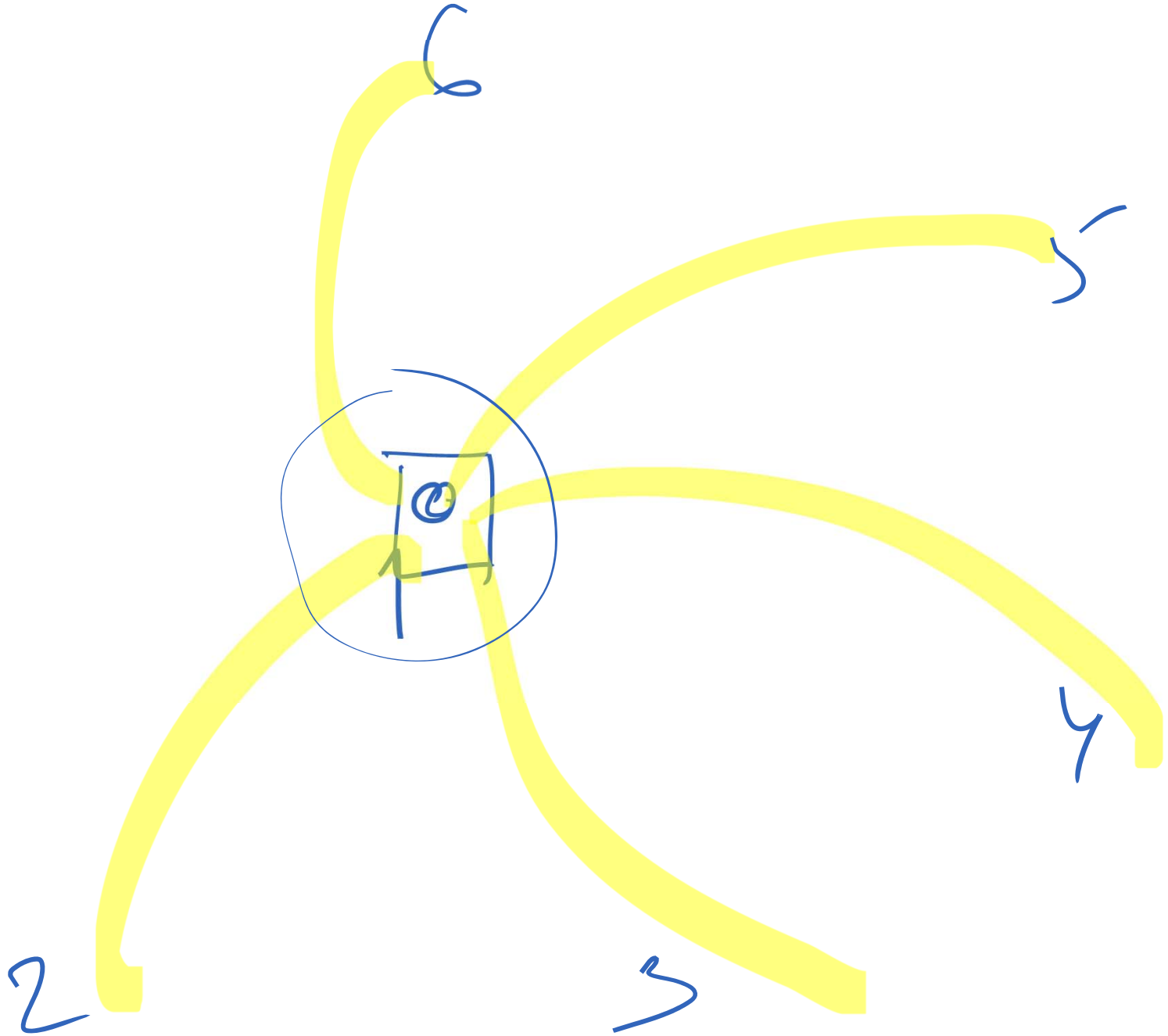
→ underlying protocol.

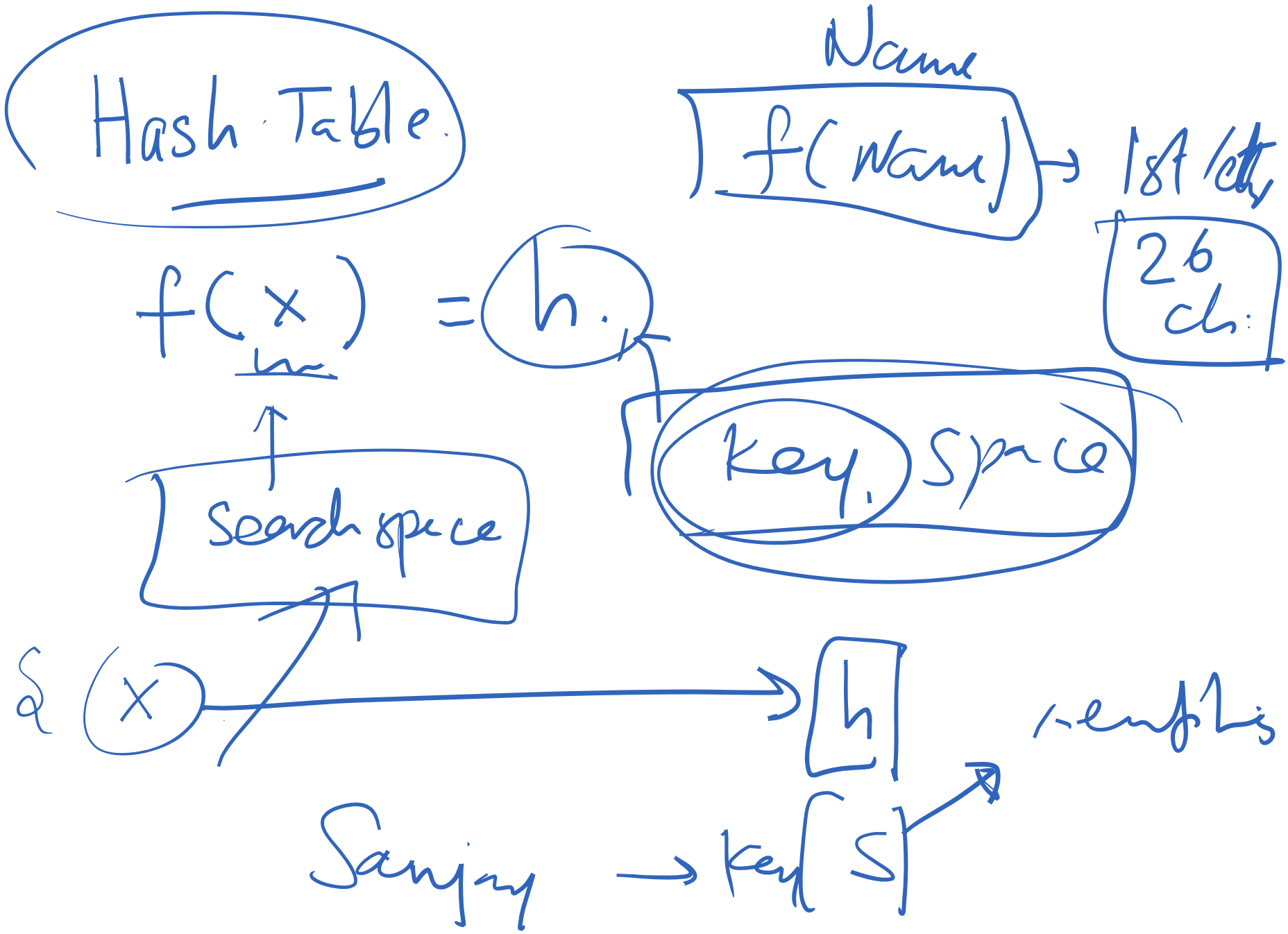
P2P

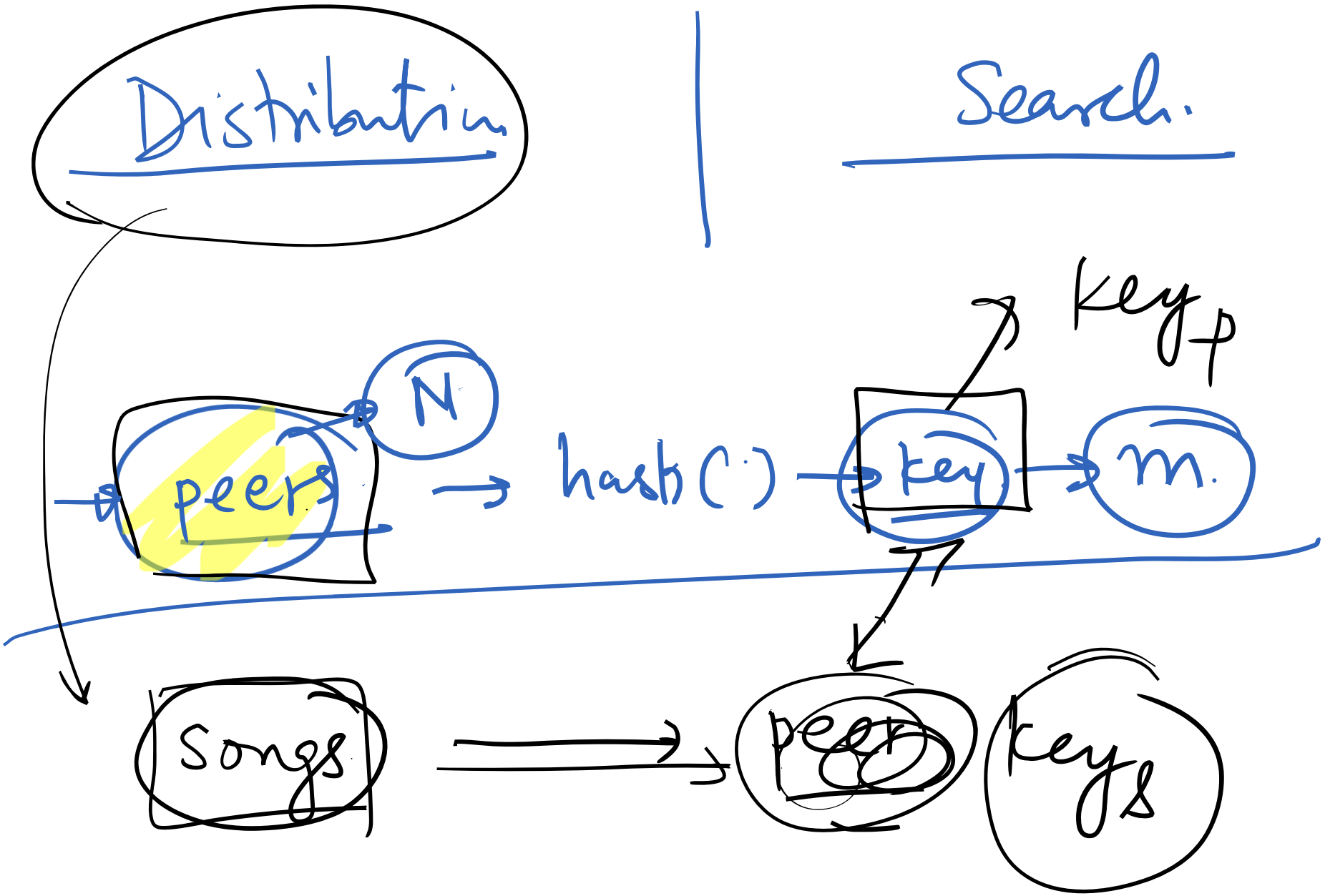
- Formation of p2p N/w
- Resource distribution and Search
- Cooperation

P2P N/w (Overlay n/w)


4
32
2
25
3
15
16
1
21
7
6


6
5
1
4
2
3


Hash Table
Name
f(Name) → 1st letter
26 ch.
f(x) = h.
key space
Search space
x
h
Sanjay → key[S]

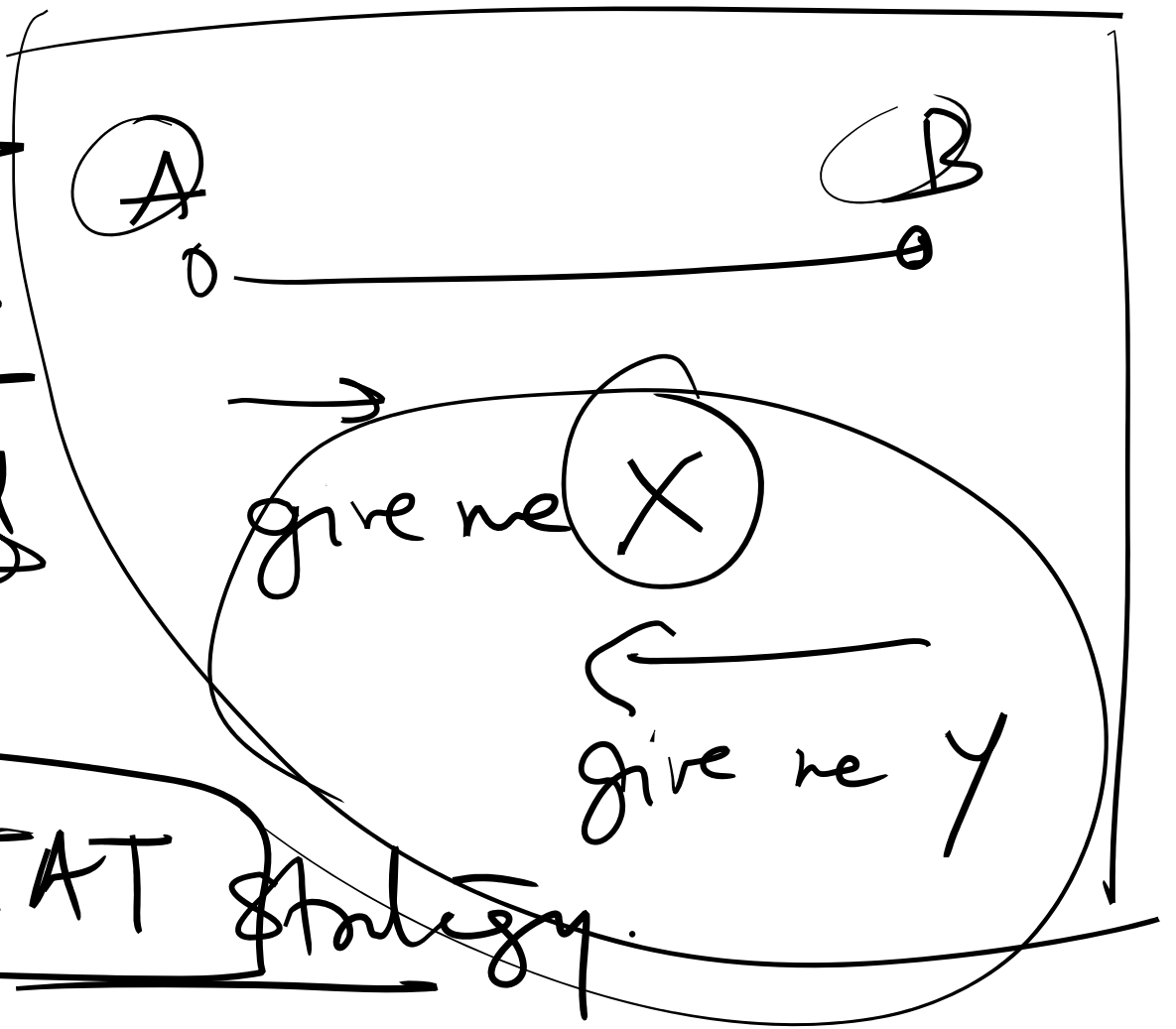

Distribution
Search.
key_p
N
peers
hash(·)
key
m.
Songs
peer
keys

Cooperation

File Transfer

Token based Coop

TIT for TAT Strategy



Generous TFT

1st time always Coop.

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

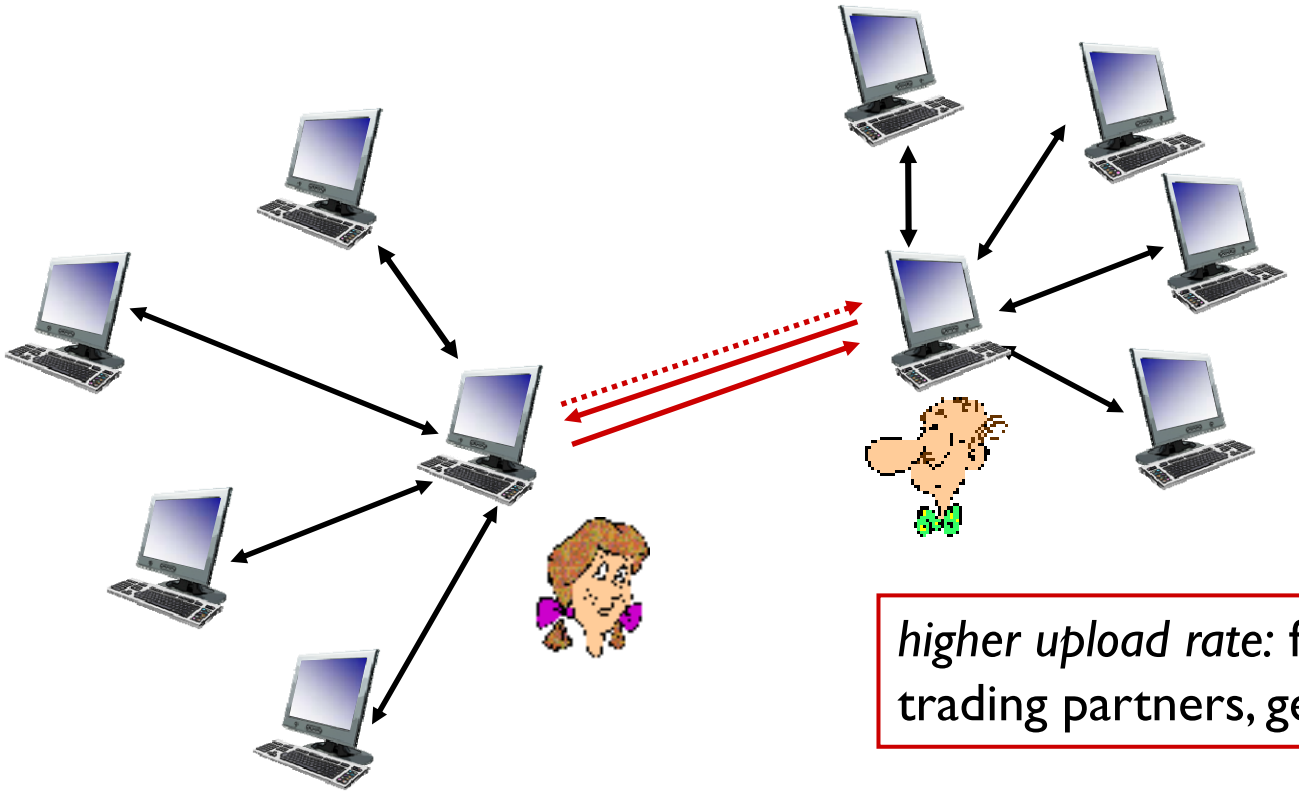
- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
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

BitTorrent: tit-for-tat

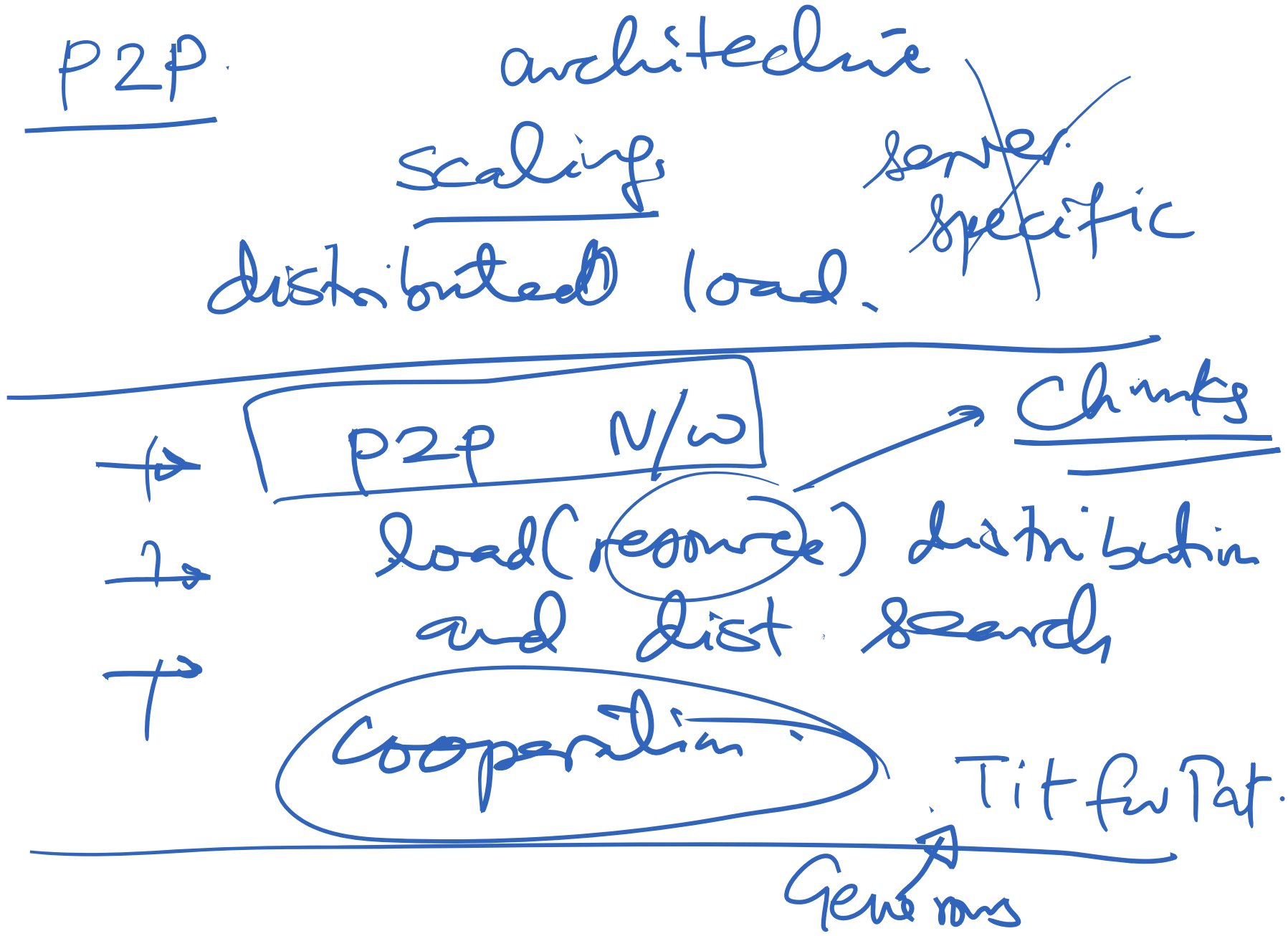
(1) Alice "optimistically unchokes" Bob

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

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



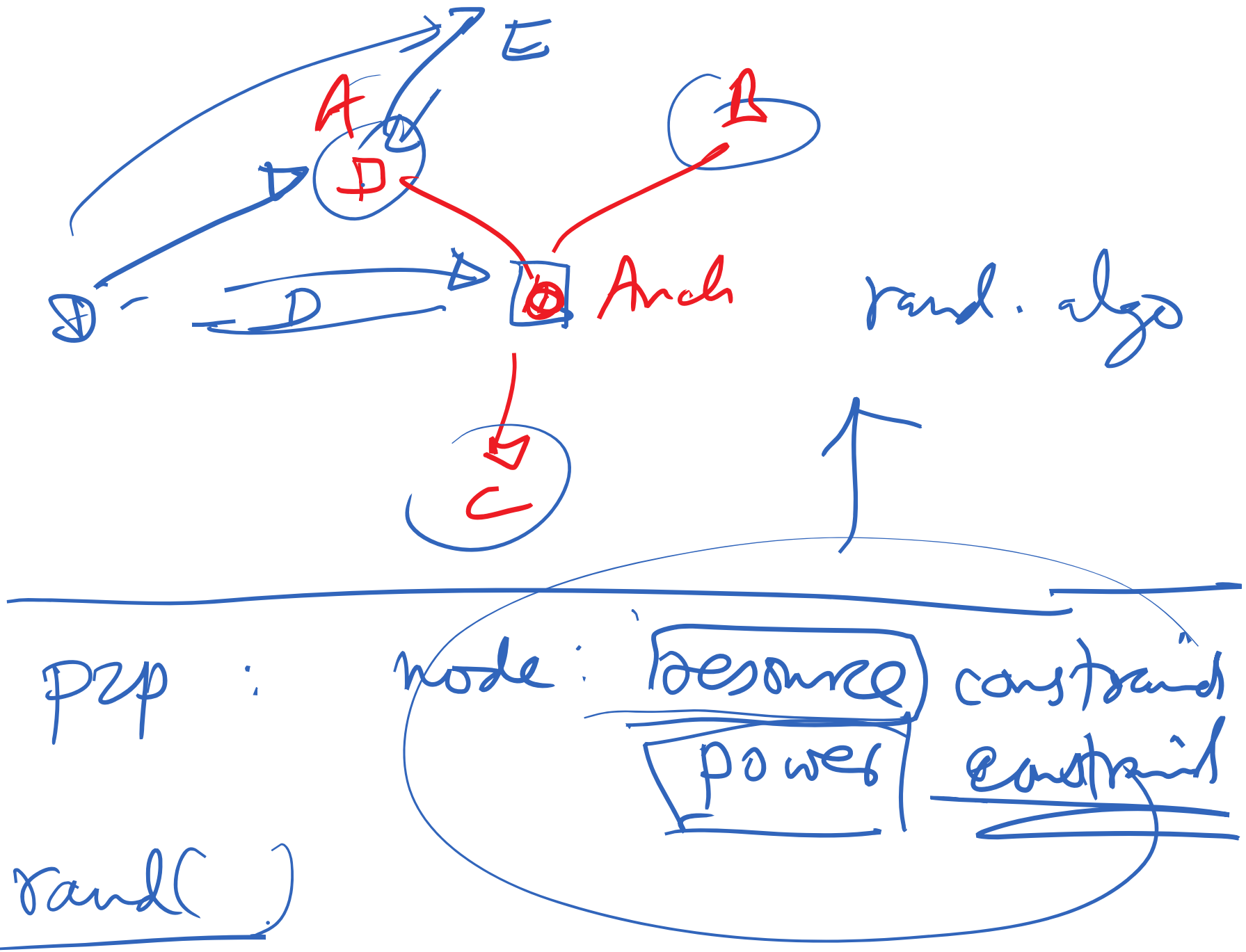
higher upload rate: find better trading partners, get file faster !

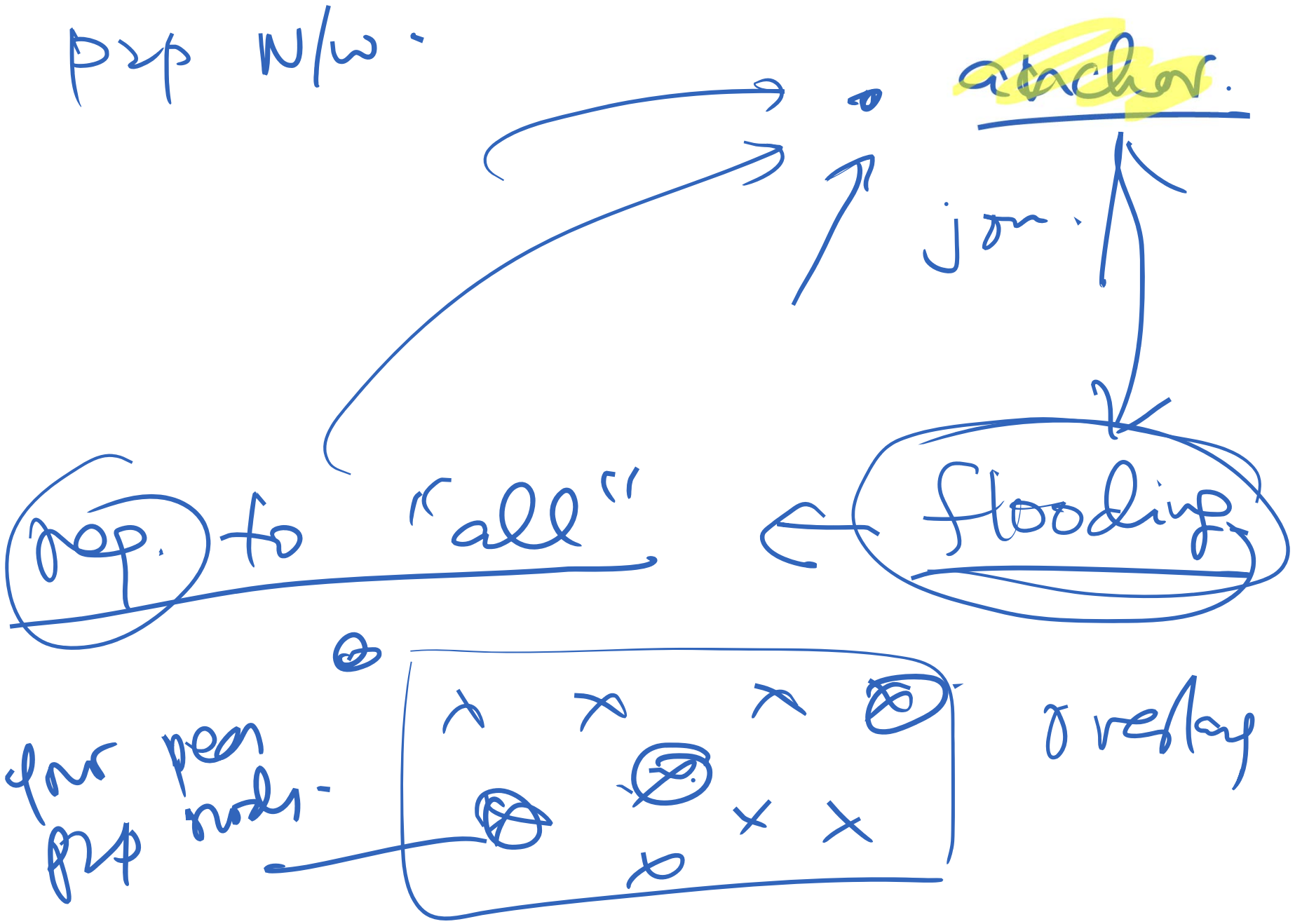

P2P
architecture
scaling
server
specific
distributed load.
P2P N/w
Chunks
load(resource) distribution
and dist search
Cooperation
Tit for Tat.
Genous

Anchor (node) → maintain a list of neighbors

join req. ⟶ accept

↳ send a random neighbor id.


E
A
D
B
D
Arch
C
rand. algo
p2p :
node : resource constraint
power constraint
rand( )


p2p N/w.
anchor.
join.
p2p to "all"
flooding
for per p2p node.
overlay

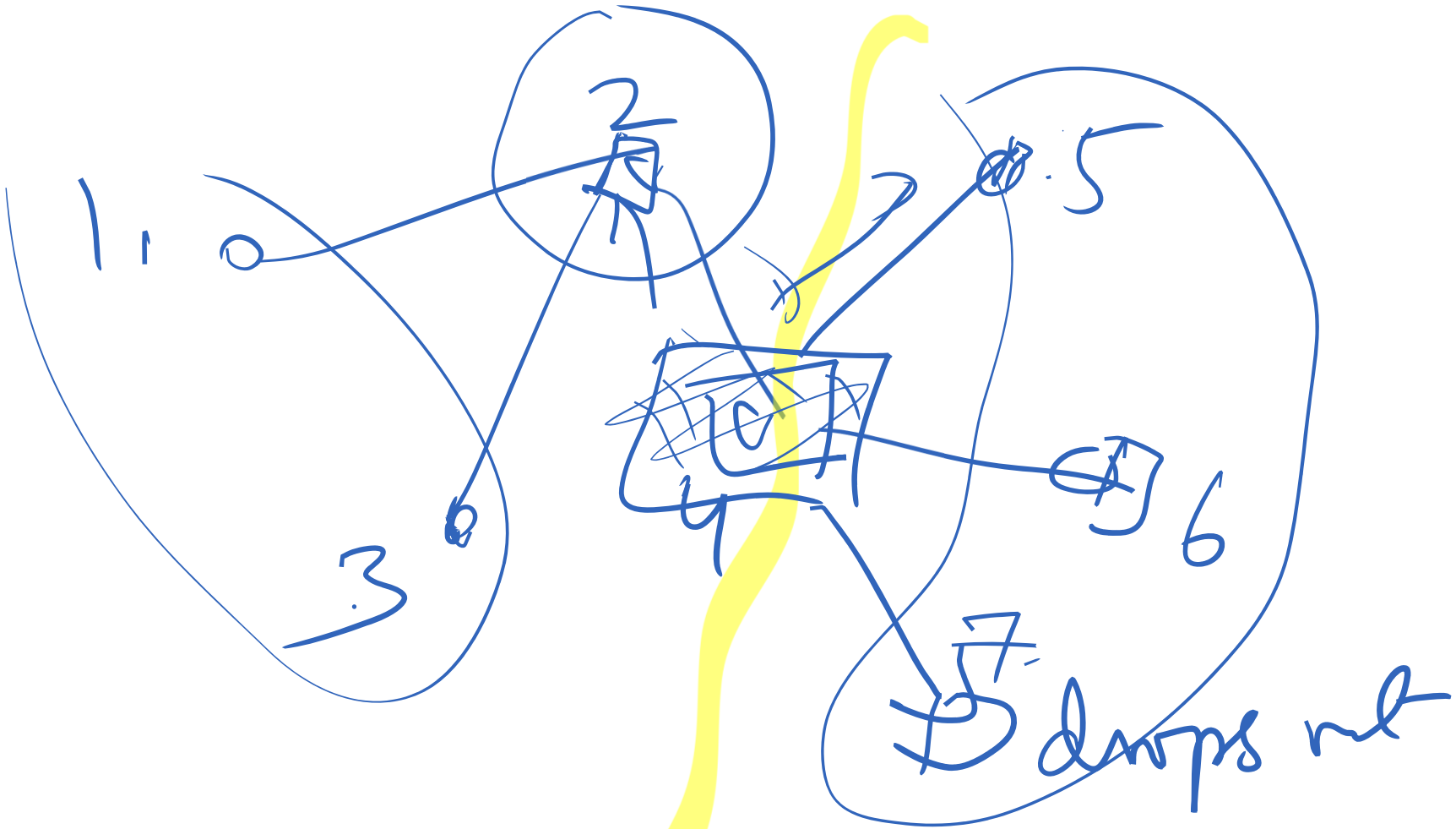
if (energy left < threshold)

don't accept

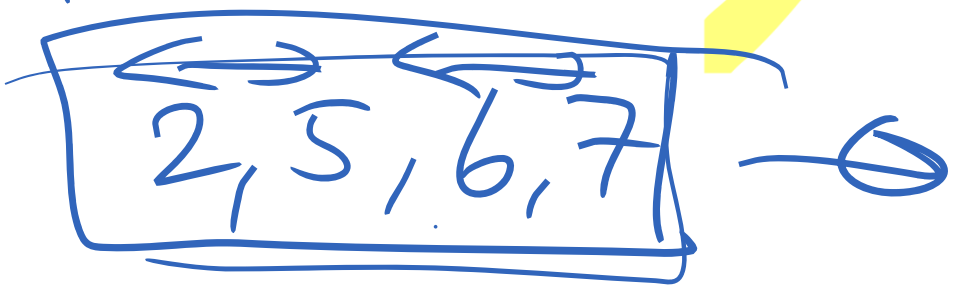
if (deg (node) > deg_{max})

don't accept

Anchor → nucleus.



How to ensure connected



ness

→ neighbor of neighbor (2-hop) list

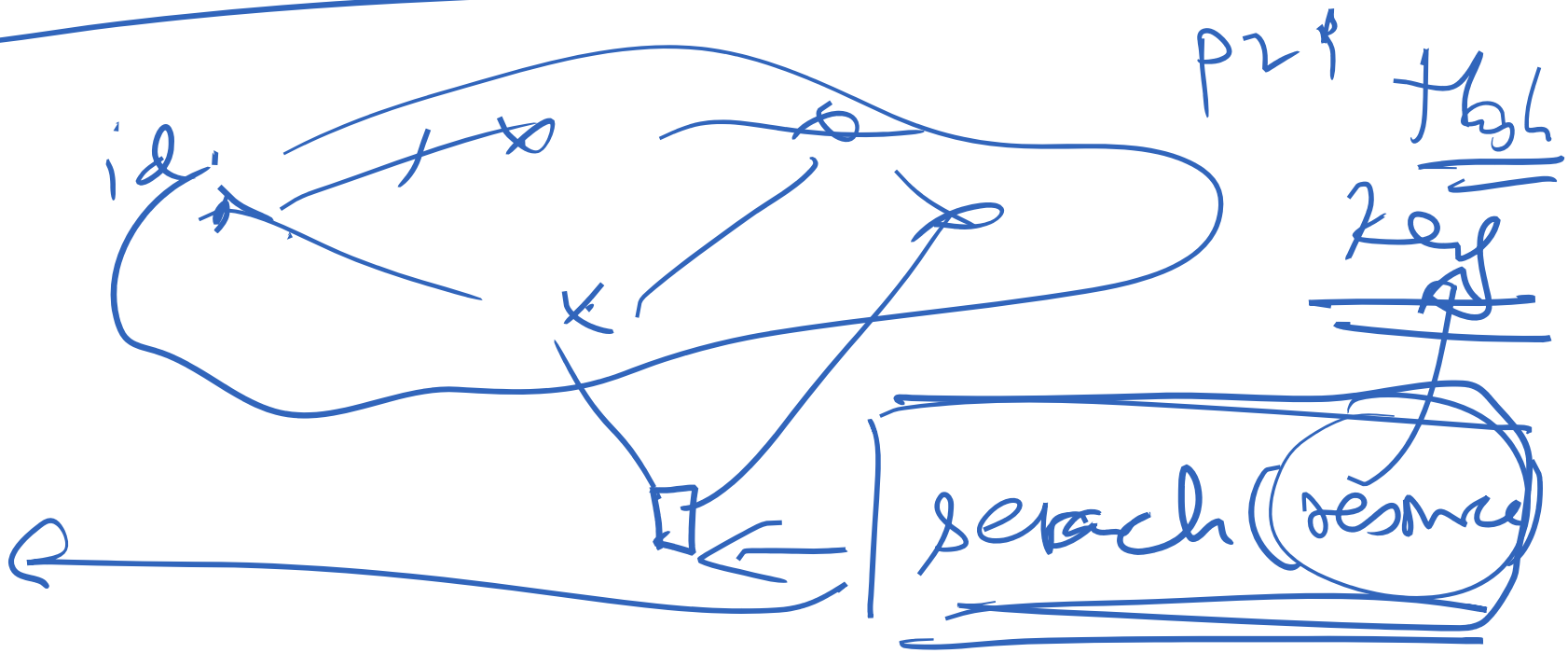
N/w formation

→ Anchor node (s) (well known)

→ Conn. req → Anch.

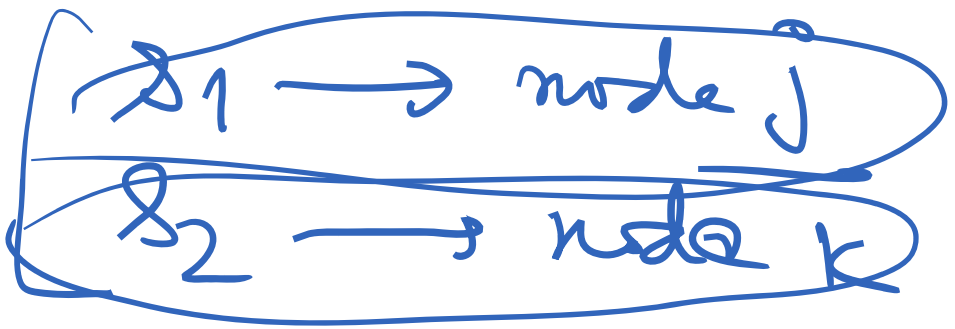
→ N/w partition → fix that

resource dist. & search



resource → distribute it in p2p

song →



Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks (CDNs)

2.7 socket programming with UDP and TCP

Multimedia Streaming

Real Time application

- delay guarantees

MM → bandwidth constraint

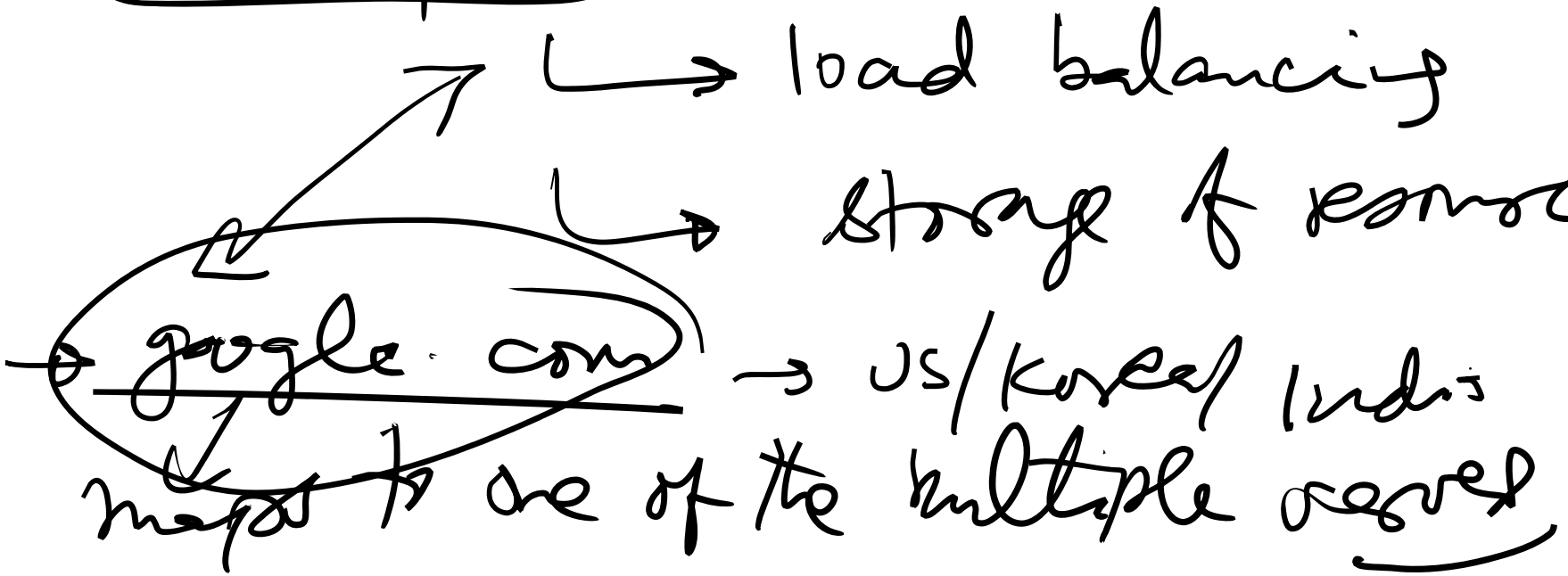
+ Client-Server → requirements are very heavy

P2P

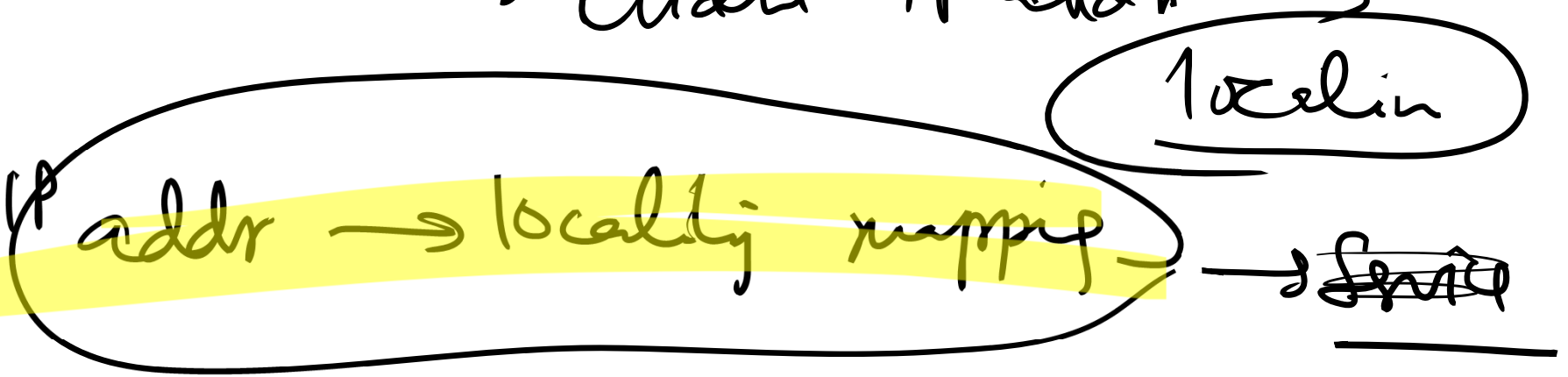
- cooperation problem / Monetized problem
- availability

problem. Stay within Client-Server model and provide scalable streaming service.

Multiple Servers.

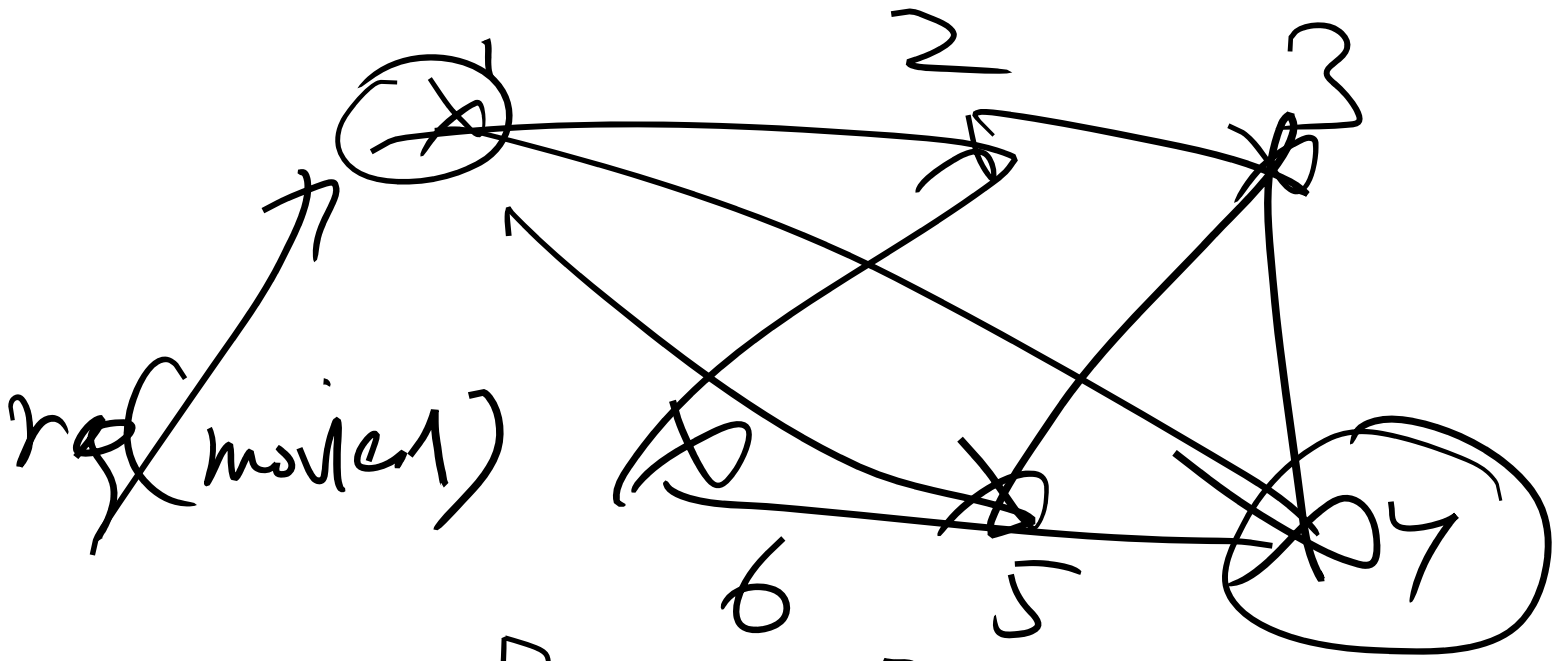


- DNS DB has list of mappings
  + clients (asks for a map)
    - client IP addr ⟹



Server → ~~replicable~~ (multiple)

resource → default



Server [Table]

movie1	Server 4
movie2	Server 5

Server:

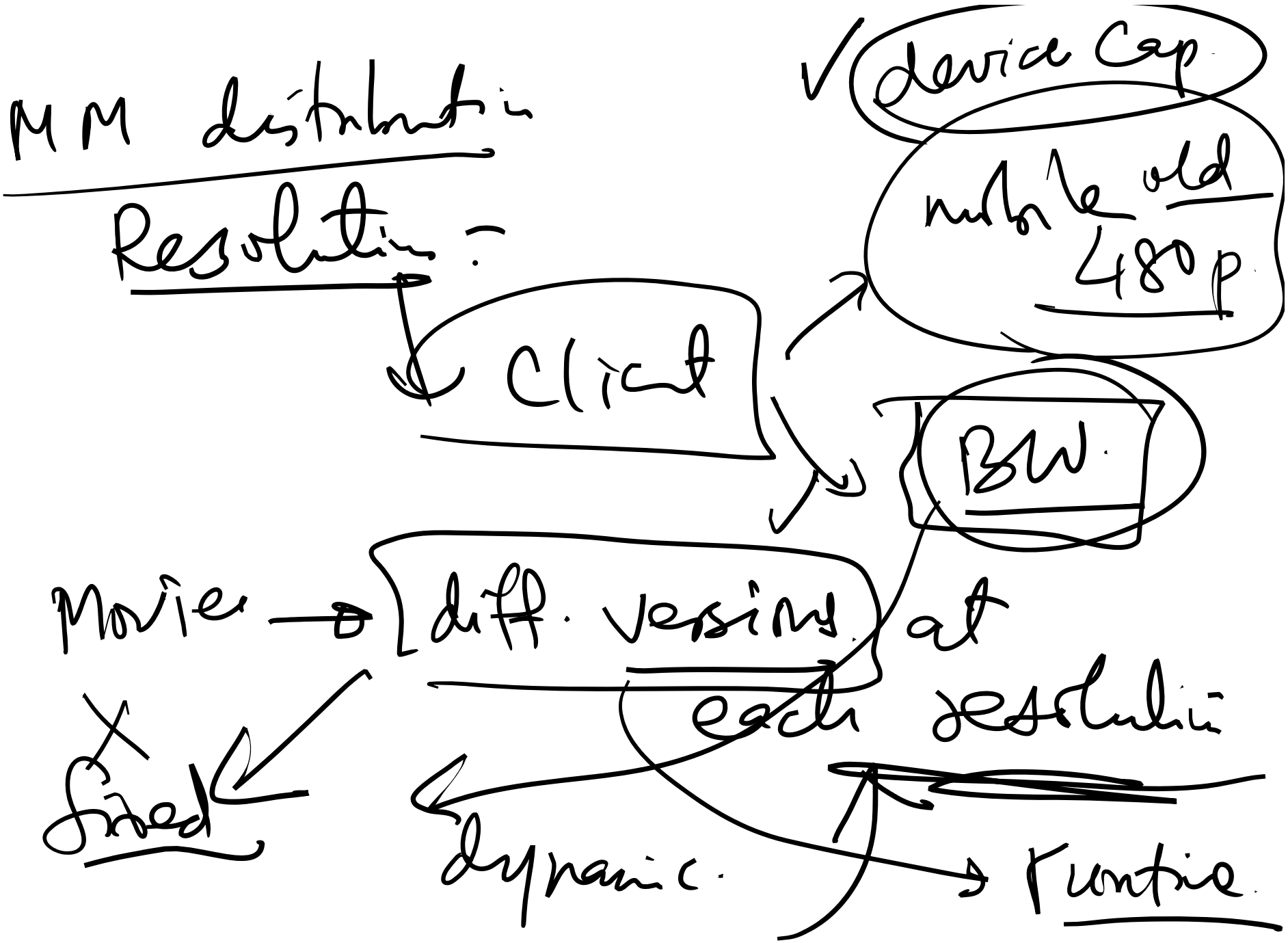
Table.

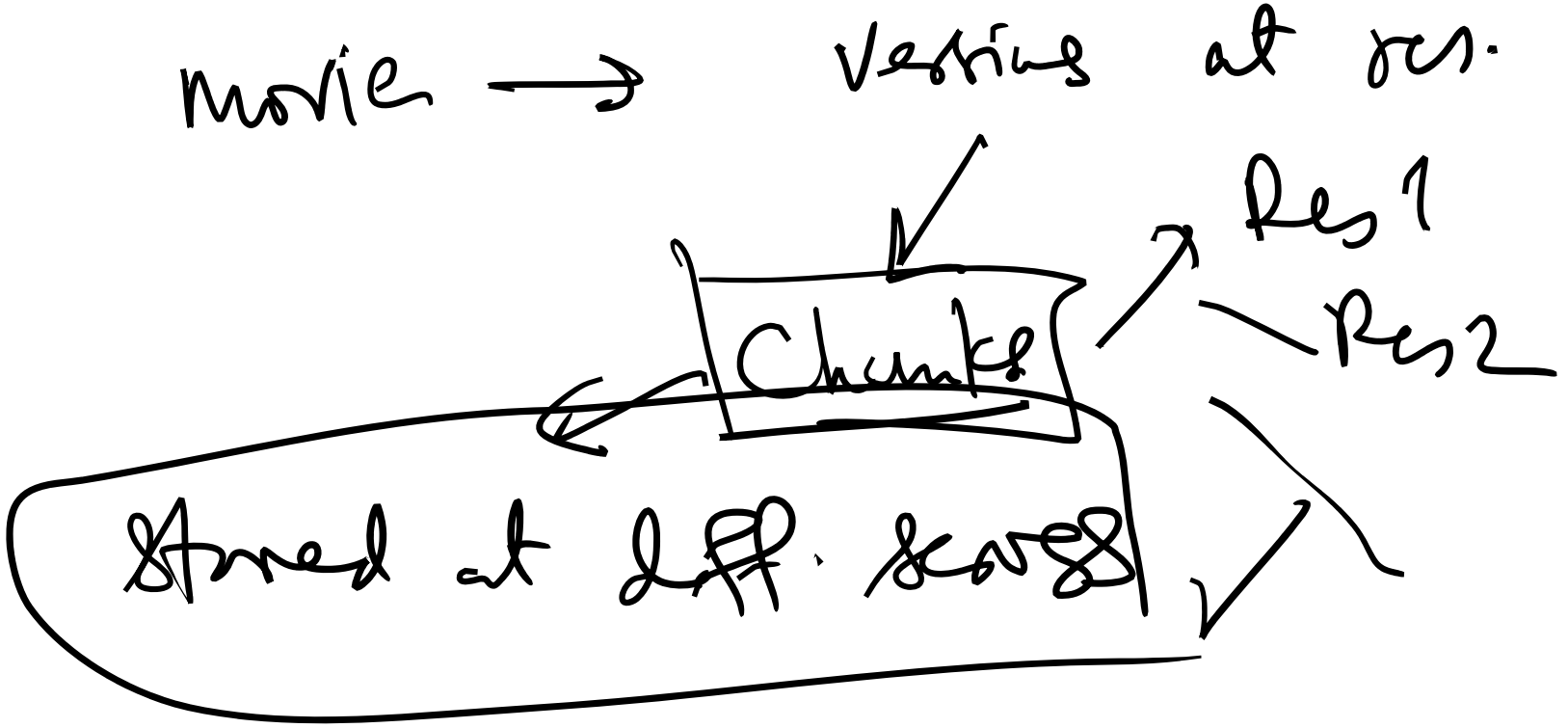
Movie key	Server IDs
movie	S1, S3, S9

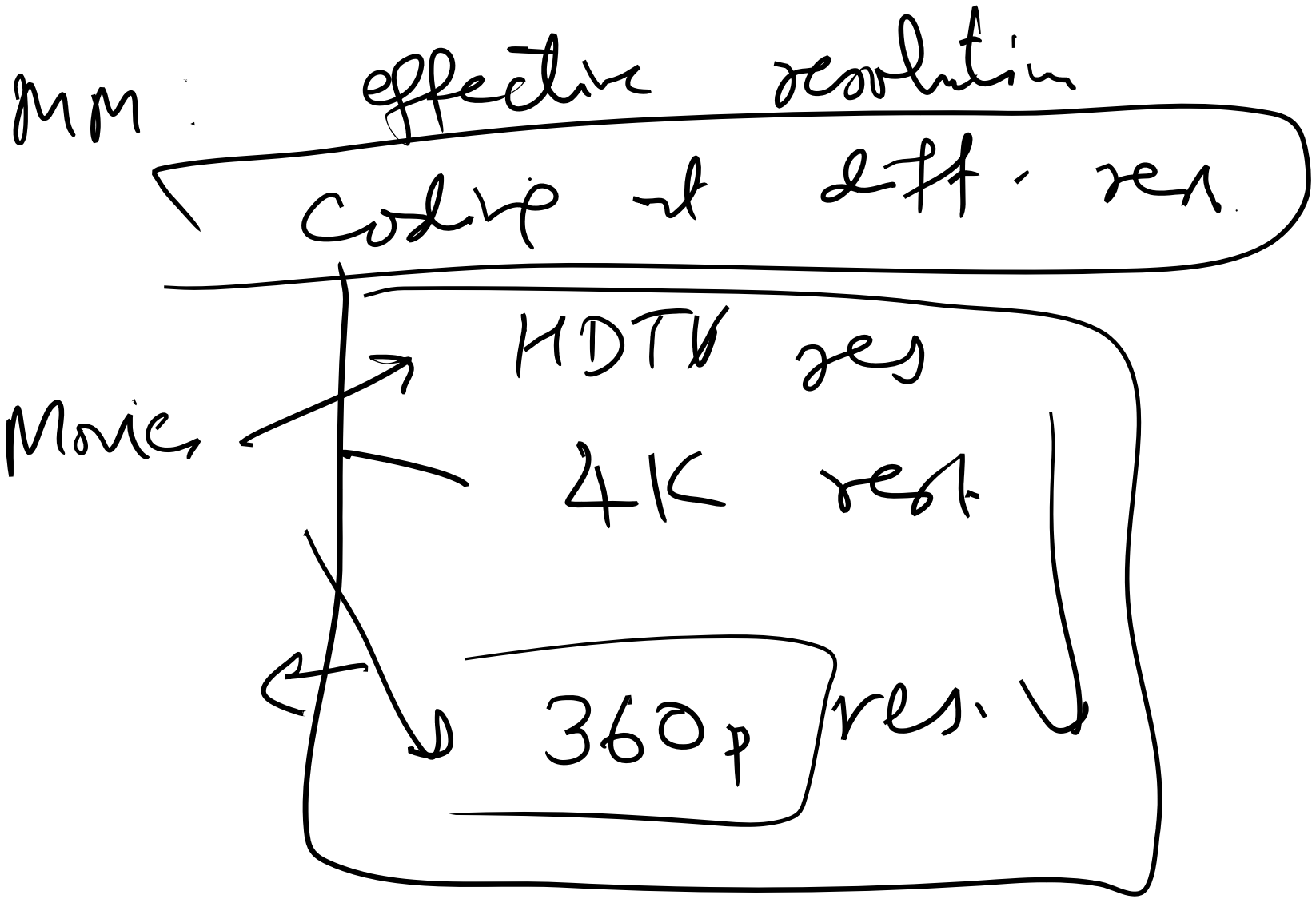
Client rep(movie1) → S1

Server S1 : resp(movie1 → S1, S3, S9)

Client → rep(movie1, S3)


MM distribution
Resolution :-
Device Cap
mobile old
480p
Client
BW
Movie
diff. versions at
each resolution
Fixed
Dynamic
Runtime


Movie
Vertica at res.
Chunks
Res 1
Res 2
Stored at diff. servers


MM
effective resolution
coding at diff. res.
Movie
HDTV res
4K res.
360p res.

Video Streaming and CDNs: context

- video traffic: major consumer of Internet bandwidth
  - Netflix, YouTube: 37%, 16% of downstream residential ISP traffic
  - ~1B YouTube users, ~75M Netflix users
- challenge: scale - how to reach ~1B users?
  - single mega-video server won't work (why?)
- challenge: heterogeneity
  - different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)
- *solution:* distributed, application-level infrastructure

Multimedia: video

- video: sequence of images displayed at constant rate
  - e.g., 24 images/sec
- digital image: array of pixels
  - each pixel represented by bits
- coding: use redundancy *within* and *between* images to decrease # bits used to encode image
  - spatial (within image)
  - temporal (from one image to next)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) and *number of repeated values* (N)

frame *i*

frame *i+1*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

Multimedia: video

- CBR: (constant bit rate): video encoding rate fixed
- VBR: (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes
- examples:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, < 1 Mbps)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)

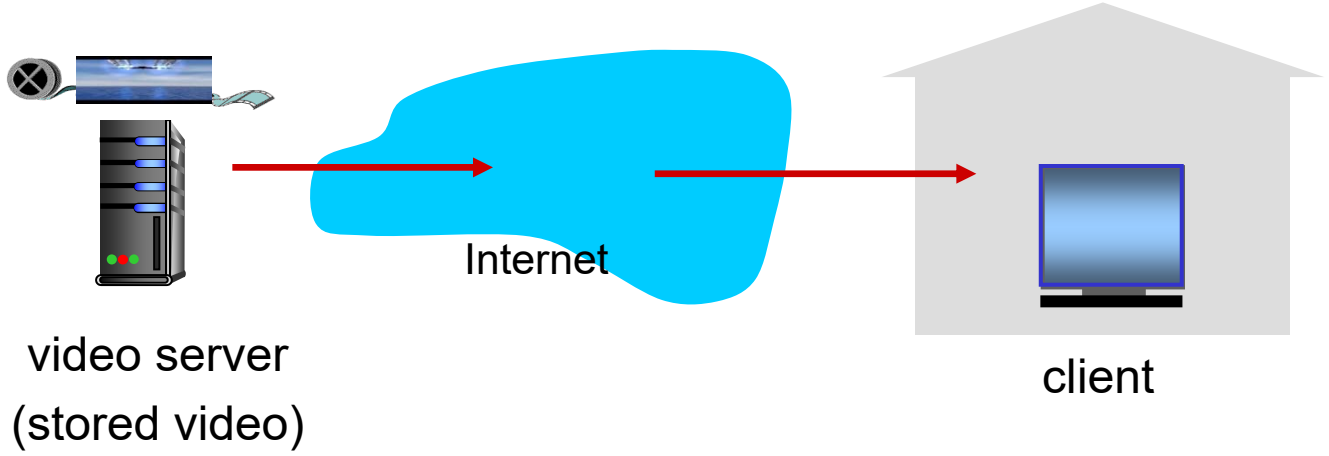
frame *i*

frame *i+1*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

Streaming stored video:

simple scenario:



Streaming multimedia: DASH

- *DASH: D*ynamic, *A*daptive *S*treaming over *H*TTP
- *server:*
  - divides video file into multiple chunks
  - each chunk stored, encoded at different rates
  - *manifest file:* provides URLs for different chunks
- *client:*
  - periodically measures server-to-client bandwidth
  - consulting manifest, requests one chunk at a time
    - chooses maximum coding rate sustainable given current bandwidth
    - can choose different coding rates at different points in time (depending on available bandwidth at time)

Streaming multimedia: DASH

- *DASH: Dynamic, Adaptive Streaming over HTTP*
- *"intelligence"* at client: client determines
  - *when* to request chunk (so that buffer starvation, or overflow does not occur)
  - *what encoding rate* to request (higher quality when more bandwidth available)
  - *where* to request chunk (can request from URL server that is "close" to client or has high available bandwidth)

Content distribution networks

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?

- *option 1:* single, large "mega-server"
  - single point of failure
  - point of network congestion
  - long path to distant clients
  - multiple copies of video sent over outgoing link

….quite simply: this solution *doesn't scale*

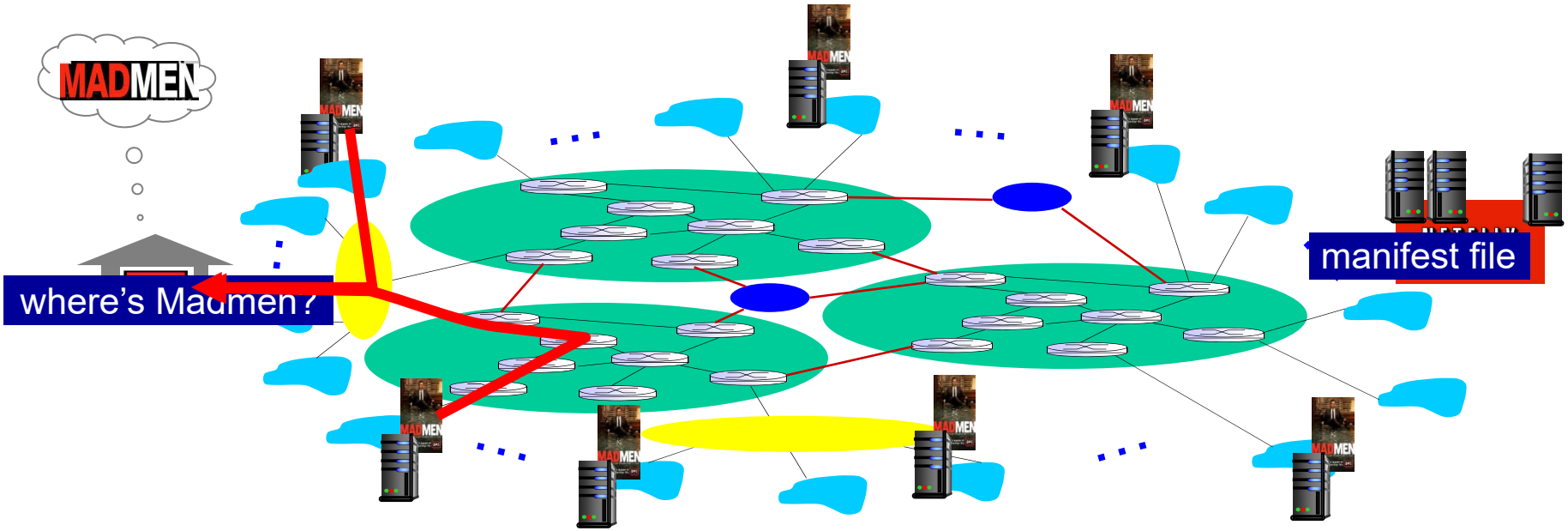
Content distribution networks

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

- *option 2:* store/serve multiple copies of videos at multiple geographically distributed sites *(CDN)*
  - *enter deep:* push CDN servers deep into many access networks
    - close to users
    - used by Akamai, 1700 locations
  - *bring home:* smaller number (10's) of larger clusters in POPs near (but not within) access networks
    - used by Limelight

Content Distribution Networks (CDNs)

- CDN: stores copies of content at CDN nodes
  - e.g. Netflix stores copies of MadMen
- subscriber requests content from CDN
  - directed to nearby copy, retrieves content
  - may choose different copy if network path congested



Content Distribution Networks (CDNs)

OTT challenges: coping with a congested Internet

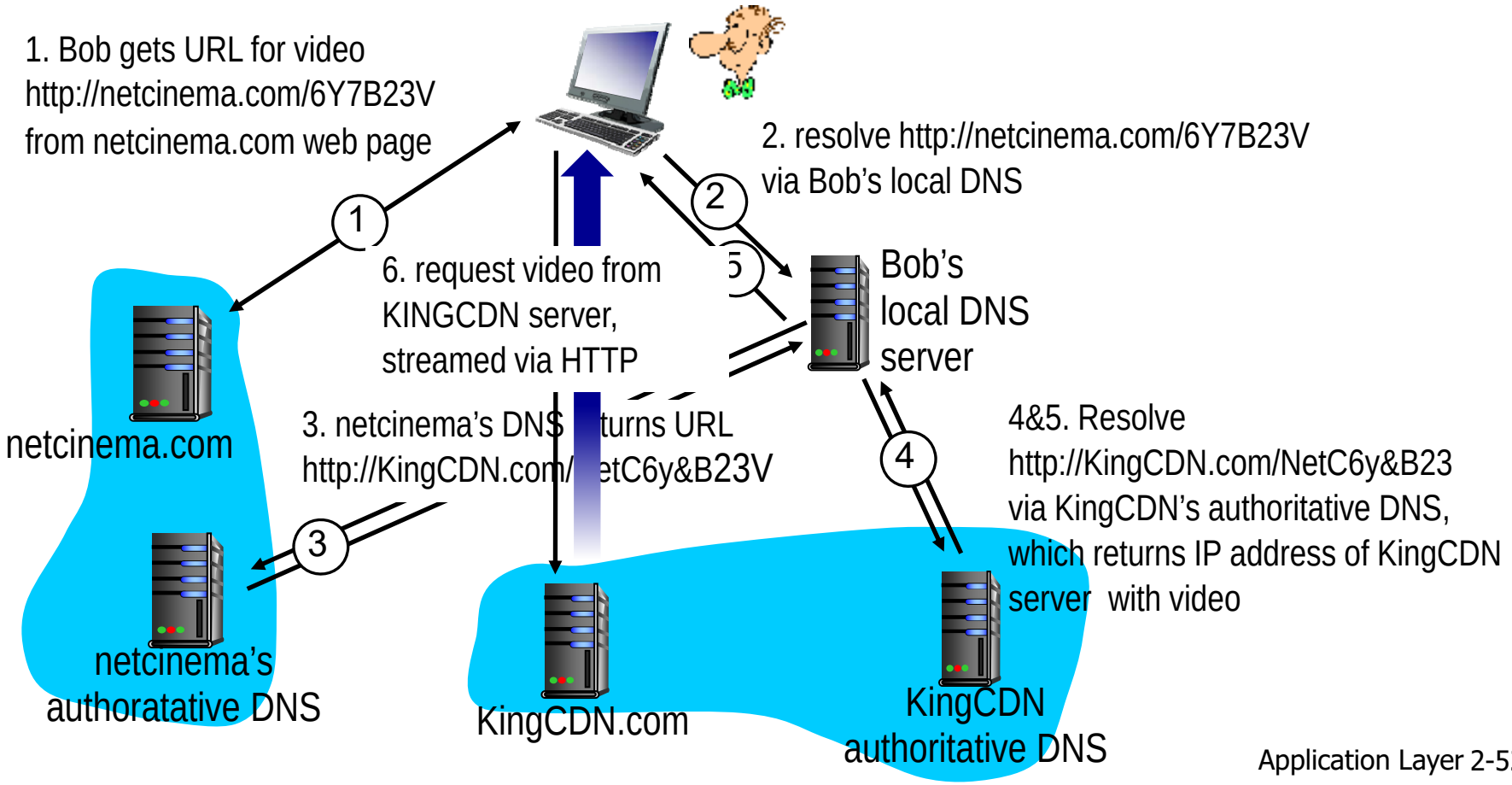
- from which CDN node to retrieve content?
- viewer behavior in presence of congestion?
- what content to place in which CDN node?

more .. in chapter 7

CDN content access: a closer look

Bob (client) requests video http://netcinema.com/6Y7B

- video stored in CDN at http://KingCDN.com/NetC6y&B23V



Case study: Netflix