Computer Networks
COL 334/672

Application Layer: DNS and P2P

Tarun Mangla

Slides adapted from KR

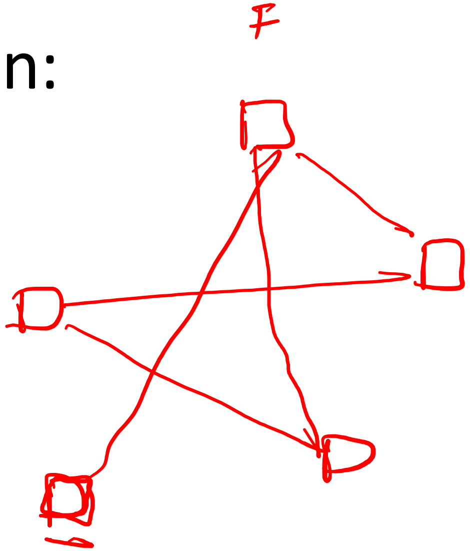
Sem 1, 2024-25

Recap

- Peer-to-Peer (P2P) networks for content distribution:
  - Scale better as they can make use of client uplink
  - Particularly popular in the early 2000s

Napster, Gnutella, BitTorrent

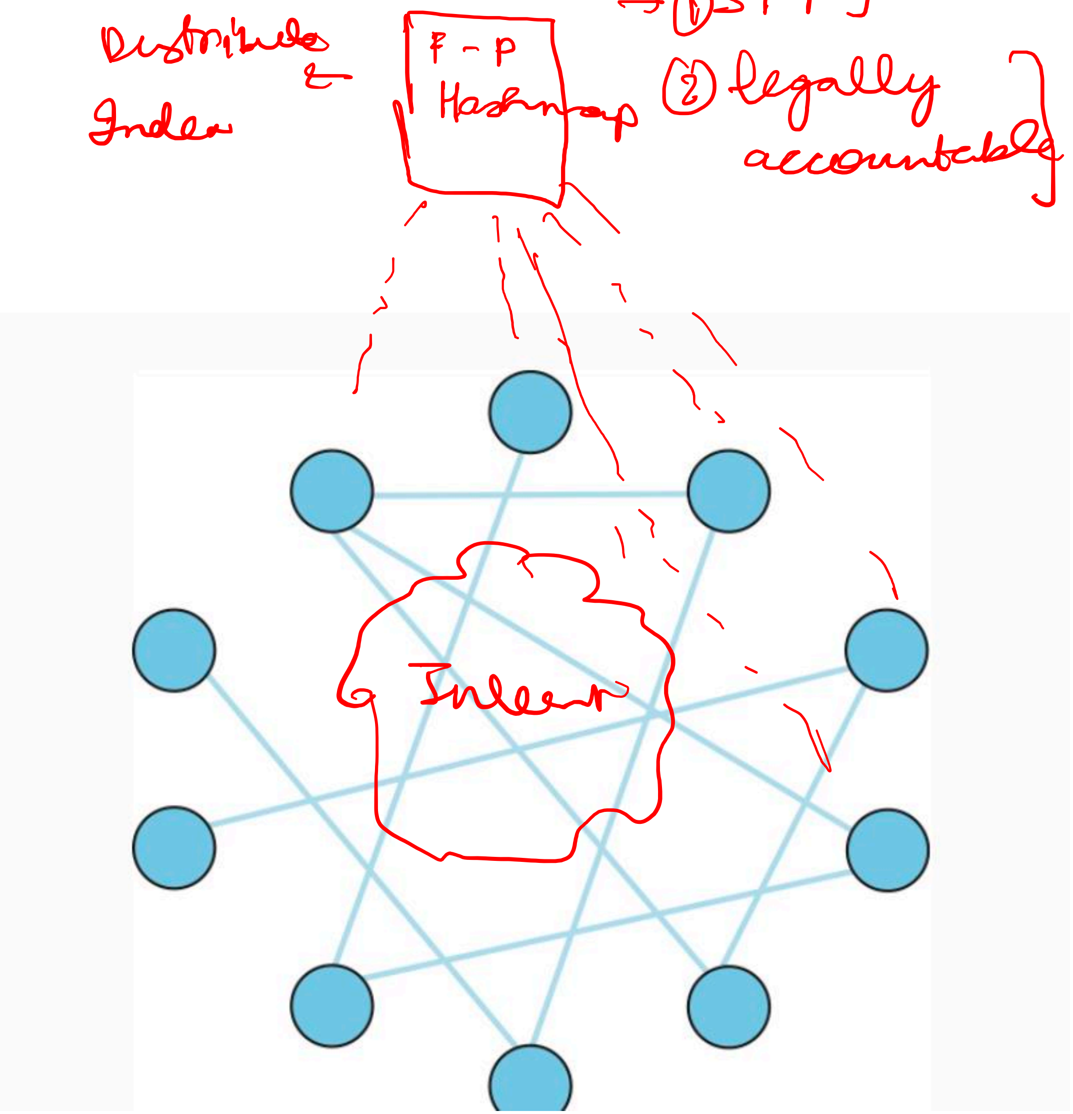
- Two interesting questions:
  - ① How to find content?
  - How to download content?



Finding a File: Approaches

✓NAPSTER (architecture)

- Approach #1: use indexing a centralized server
  - The centralized server contains information about nodes and the files
  - A new node communicates with the centralized server for file search
  - Cons:
    - Single point of failure
    - Accountable
- Approach #2:
  - Node broadcasts query to its neighbors which in turn broadcast it to their neighbors
  - Use TTLs to avoid indefinite broadcast messages
  - Cons: high overhead



Can we do better?

Finding a file in a P2P network

① Distributed manner

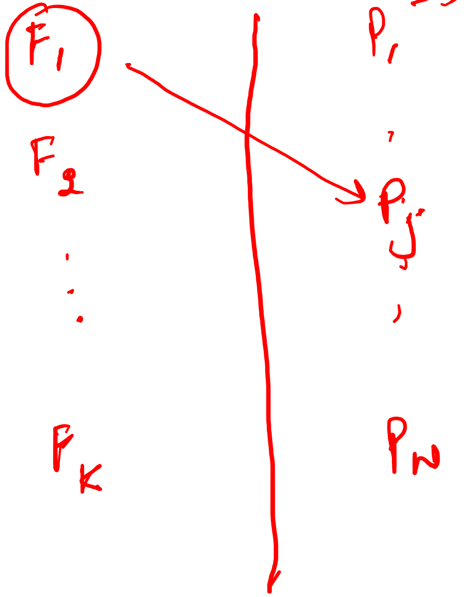
②

hash tables : File → which peer

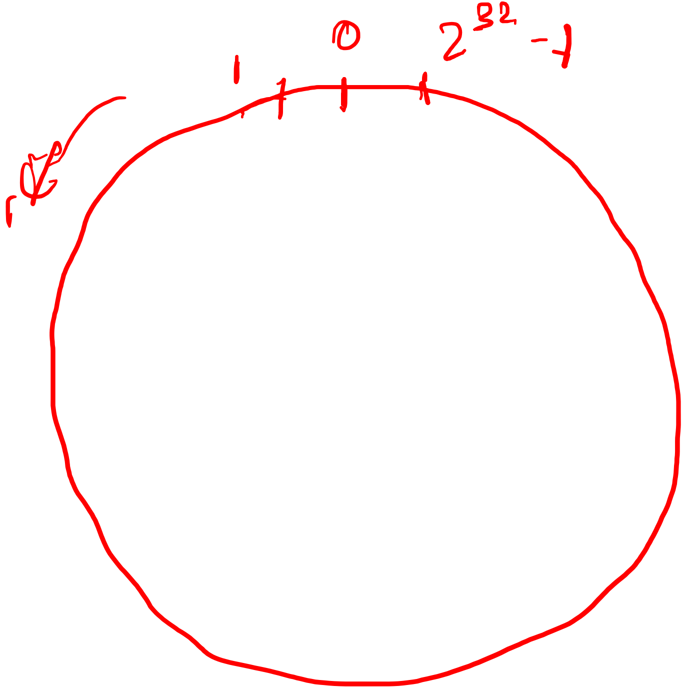
- Intuition: Some indexing is useful for a faster lookup. What kind?
- Challenge: But can't have a centralized hash table
- Solution: Use a distributed hash table (DHT)

$H(F_1) \rightarrow P_j$

P_K = want to look up for F_r



P_1 → IP-address (32-bit address space)

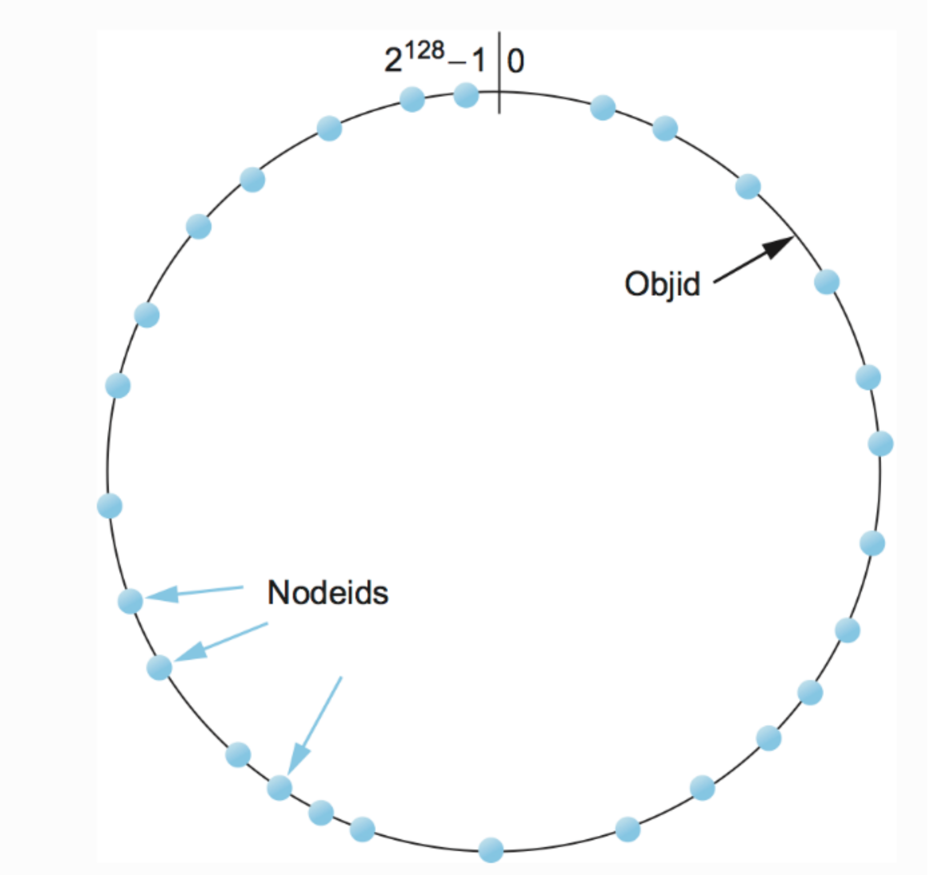


~~H(F...)~~ $\overset{H(F_1)}{=}$: $F_1 \rightarrow P_j$

PASTRY

Idea:

- Map the objects and the nodes to a common virtual space
- Store the object information in a node that is closest to it in the abstract space



How do we search for the closest node?

PASTRY

$2^3 - 1$

H(P): 0, 2, 6

H(F): 2, 3, 5

F1: 6

f2: 2

f3: 0

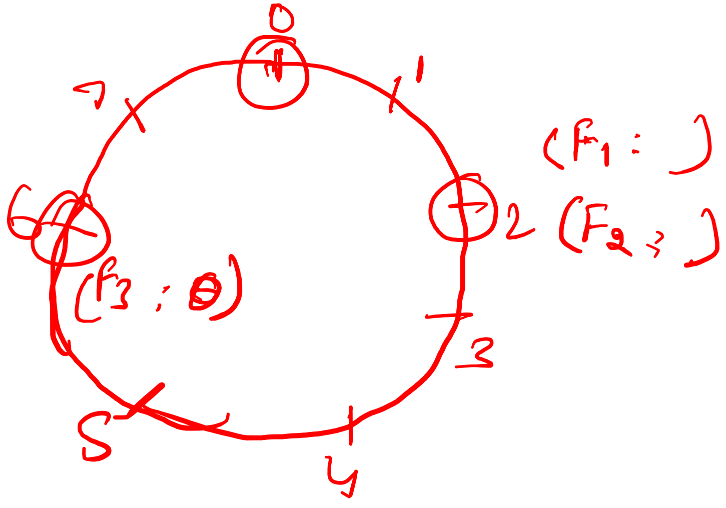
Idea:

- Map the objects and the nodes to a common virtual space : IP addr space
- Store the object information in a node that is closest to it in the abstract space

① what if the peer disconnects

H(P1) →

Peers may not be unfo



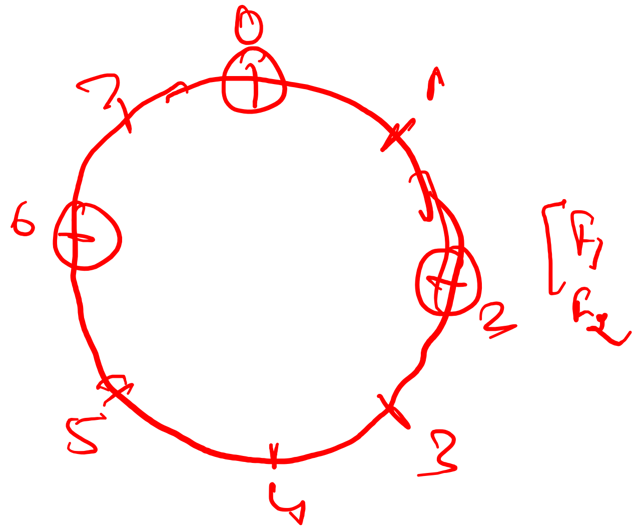
How do we search for the closest node?

PASTRY: Searching Closest Node

H(F₁): 2 H(F₃)= 5

H(F₂): 3

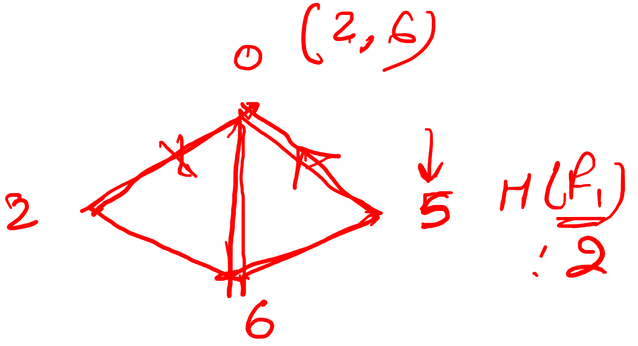
Idea: *To search for a file f, route query messages closer to H(f) in the virtual space until you find the node containing information about f*



Challenge: *How do we ensure that we can always go to a closer node?*

Dynamically change

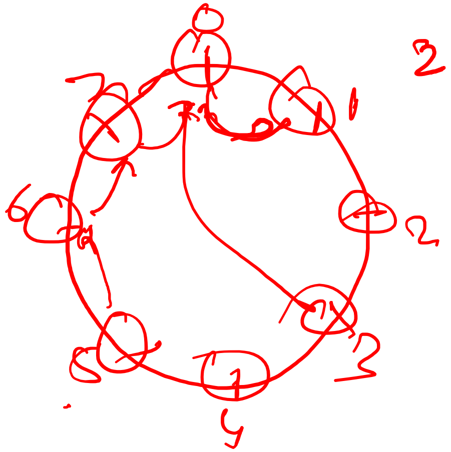
L = 2



Solution: *Each node should store L nodes (L/2 successors, L/2 predecessors) and log(N) nodes distributed randomly in the virtual space* (Randomized algorithm)

= 4

4



Broadcast

CHORD

6

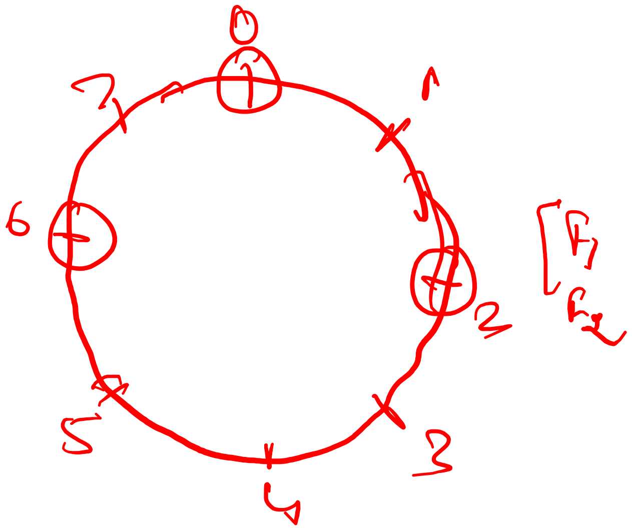
log(N)

2^m +5

PASTRY: Searching Closest Node

H(F1) : 2
H(F2) : 3
H(F3) = 5

Idea: *To search for a file f, route query messages closer to H(f) in the virtual space until you find the node containing information about f*



Challenge: *How do we ensure that we can always go to a closer node?*

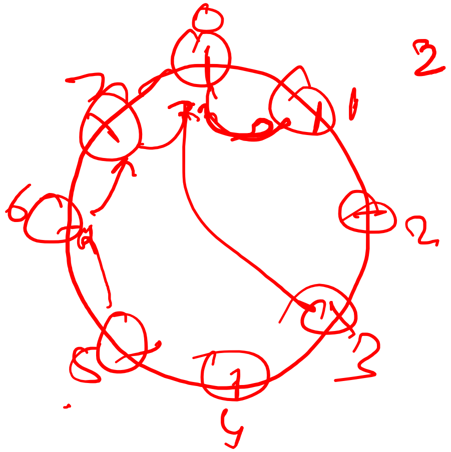
L = 2

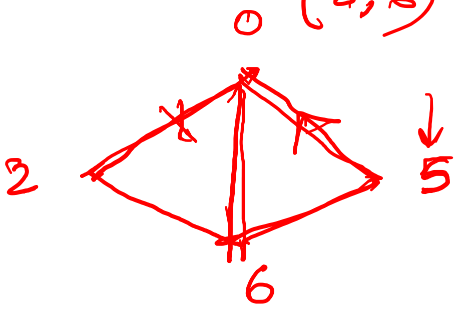
Dynamically change

Solution: *Each node should store L nodes (L/2 successors, L/2 predecessors) and log(N) nodes distributed randomly in the virtual space* => 4 (Randomized algorithm)

CHORD

log(N)





H(F1) : 2

Broadcast

2^m +1

Distributed Hash Table

- You should think about the following:
  - How the neighbor are maintained in the first place

- Various optimizations exist for DHTs

- Used in other domains such as distributed file system, web caching etc.

Next question: How to download content?

P2P file distribution: BitTorrent

①. What content to download from whom?

② who to send the data?

① Rarest one first

File chunks

② In-order (Application consider

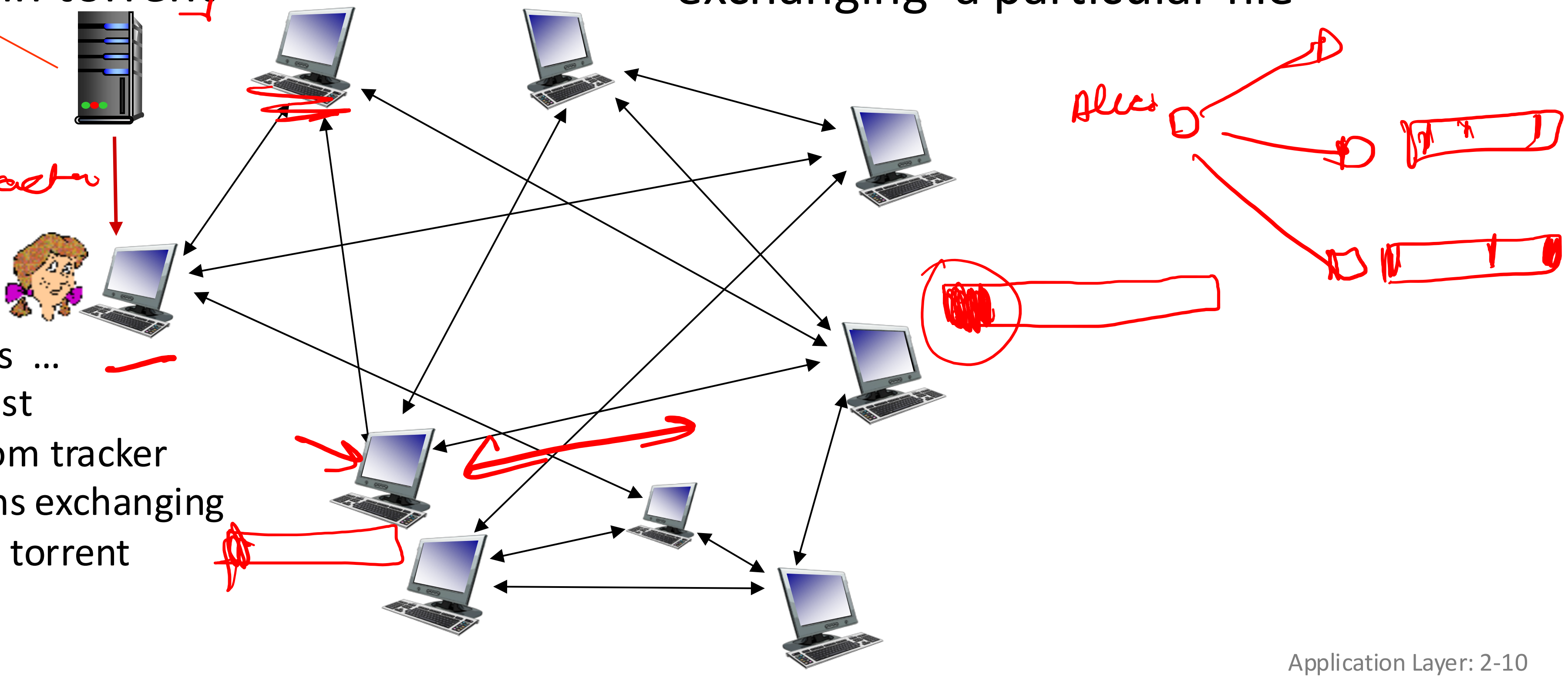
③ Randomly down

tracker: tracks peers participating in torrent

① centralized tracker

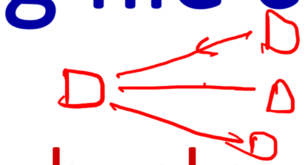
② Decentralized tracker (DHTs)

torrent: group of peers exchanging a particular file



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

BitTorrent: requesting, sending file chunks



Which chunks to request?

- 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: whom to send chunks?

Exploration and exploitation

N

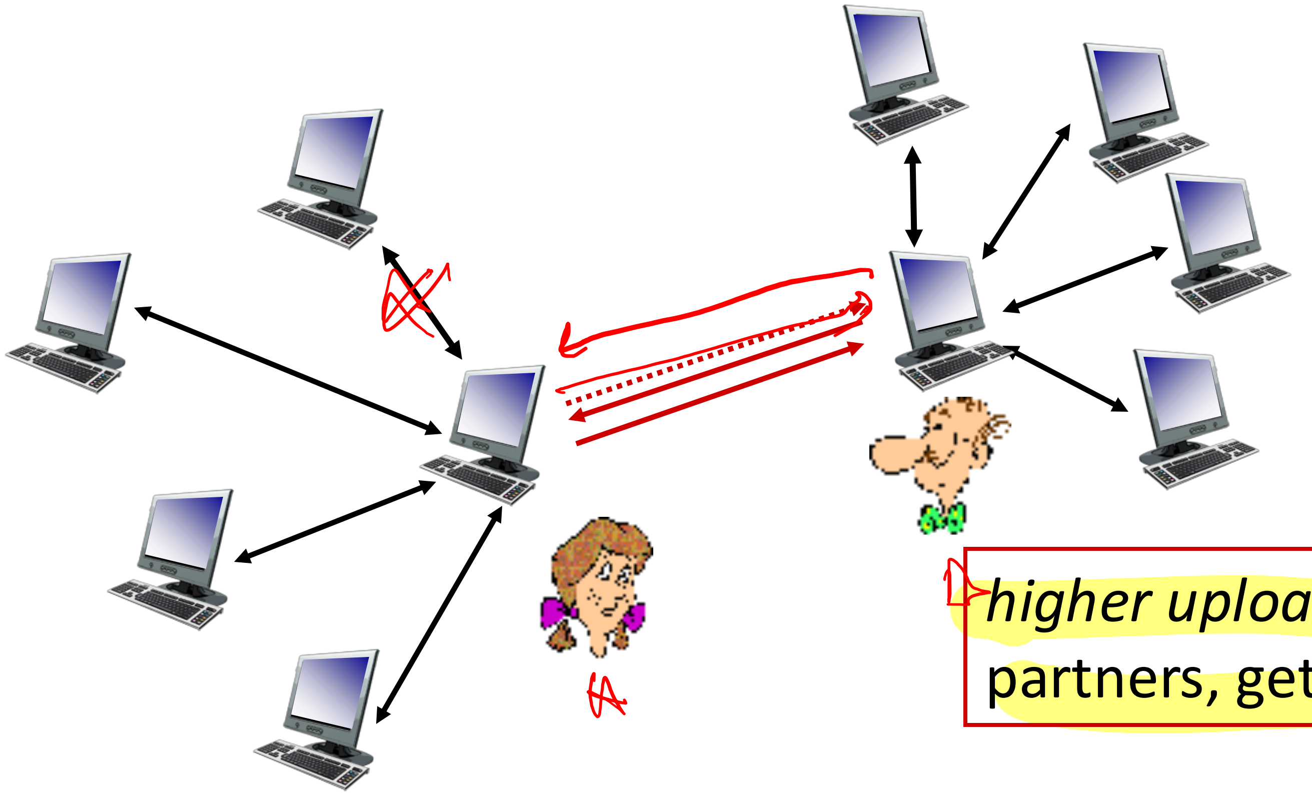
- Uses tit for tat
- sends chunks to those four peers currently sending chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks

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 !

Need for alternate "faster" content distribution mechanism

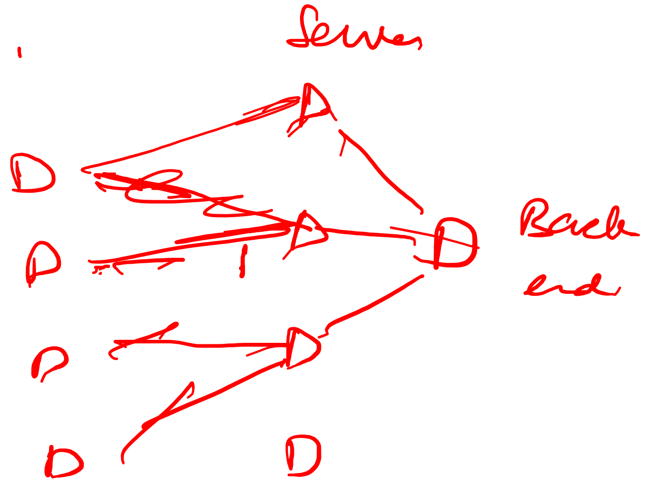
Content Delivery Network

- P2P would not work for latency-sensitive applications (e.g., web)
- Need an alternate mechanism that scales well (number of users and geography)
- Question: how to scale client-server paradigm?

Use geographically distributed servers

- Too expensive to do that for every content provider!

Use Content Distribution Networks



Content distribution networks (CDNs)

Akamai → MIT

- CDN: geographically distribute collection of *server surrogates*
- Servers can be leased by many customers
- Popular CDNs: Limelight, Akamai, Level3
- Two kinds of server placement policies:

  - *enter deep:* push CDN servers deep into many access networks
    - close to users
    - Akamai: 240,000 servers deployed in > 120 countries (2015)

    latency

  - *bring home:* smaller number (10's) of larger clusters in POPs near access nets
    - used by Limelight

    Tier-1 ISPs

    low maintain

Akamai

Limelight NETWORKS