Computer Networks COL 334/672

Application Layer: DNS and P2P

Tarun Mangla

Slides adapted from KR

Sem 1, 2024-25

Recap: Application Layer

- HTTP
- Email
- DNS → Domain Name System
- P2P
- Video streaming

Domain Name → IP address

↓

Distributed & Decentralized Database

Recap: DNS

- Mapping between domain name and IP address
- *distributed database* implemented in hierarchy of many *name servers*

Resource Record

Name, Value, Type, TTL

Root NS (Top-level domains)

.com	dns.xyz.com	NS	—
dns.xyz.com	x.y.z.w	A	—

.com NS .org .in

Primary NS

Secondary NS google.com

ns1.google.com

google.com	x.y.z.w1	A	—

Client

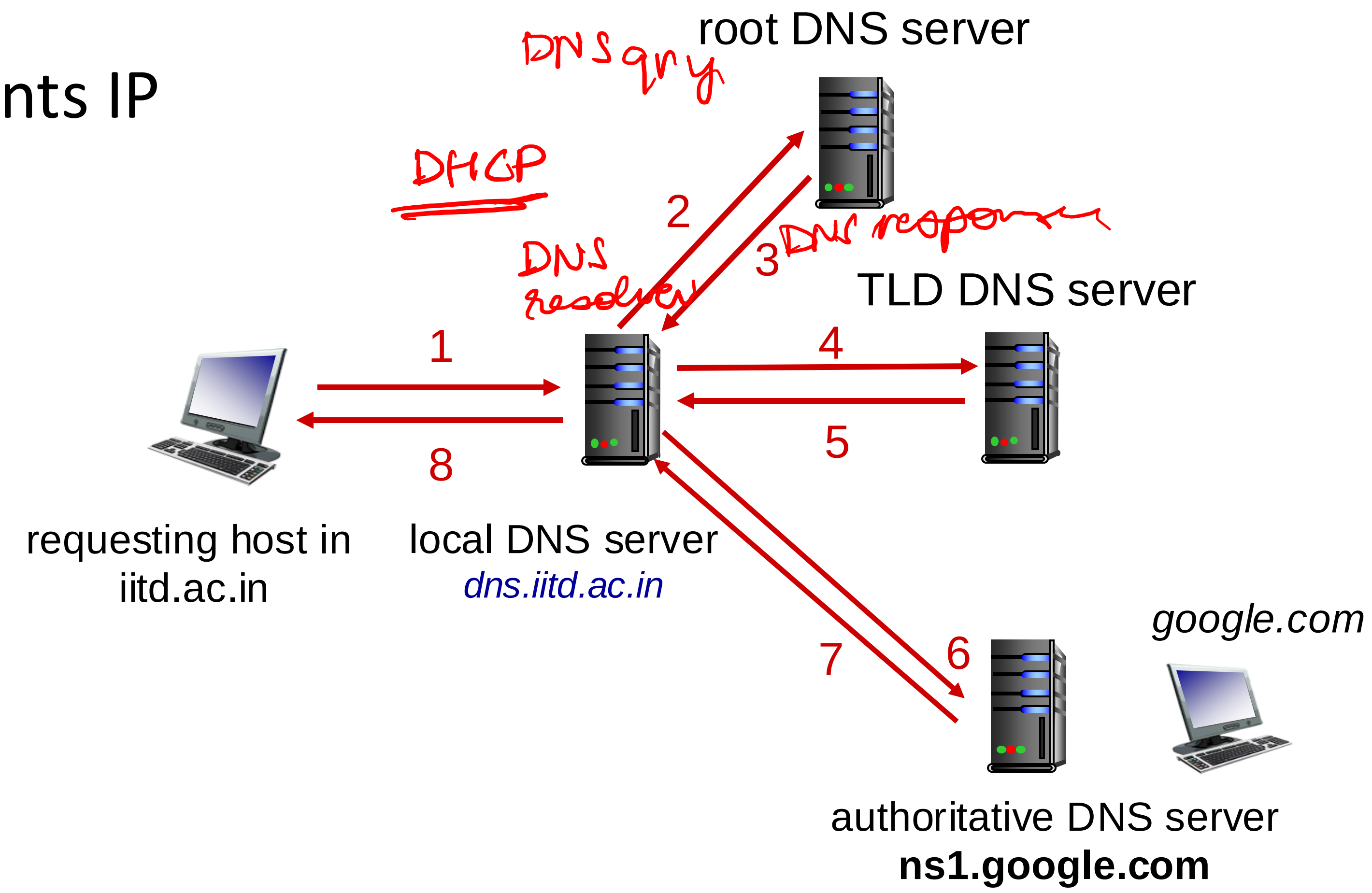
google.com

DNS name resolution: iterated query

Example: host at iitd.ac.in wants IP address for google.com

Iterated query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



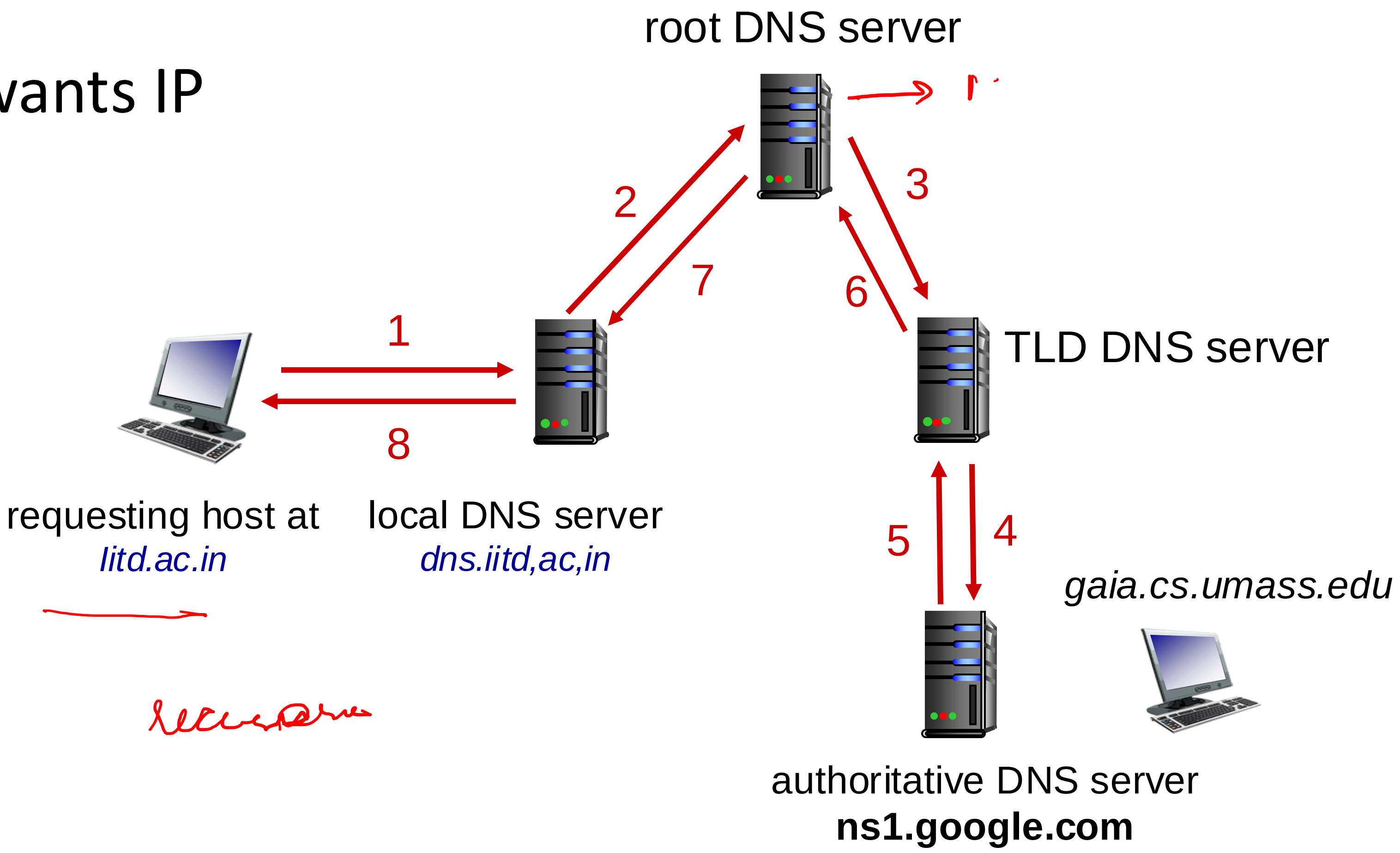
iterative query

DNS name resolution: recursive query

Example: host at iitd.ac.in wants IP address for google.com

Recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



DNS protocol messages

DNS *query* and *reply* messages, both have same *format:*

message header:

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative

2 bytes	2 bytes
identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions (variable # of questions)	
answers (variable # of RRs)	
authority (variable # of RRs)	
additional info (variable # of RRs)	

IP geolocation databases

DNS is also used for load balancing

LB

Authoritative NS

DC1

IP_A google.com

IP_B google.com

IP_C google.com

IP_D google.com

DC2

DNS protocol messages

DNS *query* and *reply* messages, both have same *format*:

8.8.8.8 → Public DNS resolver

NS do not have a way to know client IP addrs

Response Message

IP

UDP

DNS

```
Queries
  google.com: type A, class IN
    Name: google.com
    [Name Length: 10]
    [Label Count: 2]
    Type: A (Host Address) (1)
    Class: IN (0x0001)
Answers
  google.com: type A, class IN, addr 142.250.194.142
    Name: google.com
    Type: A (Host Address) (1)
    Class: IN (0x0001)
    Time to live: 227 (3 minutes, 47 seconds)
    Data length: 4
    Address: 142.250.194.142
```

← 2 bytes → ← 2 bytes →

identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions (variable # of questions)	
answers (variable # of RRs)	
authority (variable # of RRs)	
additional info (variable # of RRs)	

Caching DNS Information

dig google.com

localDNS
(cache the response)

- once (any) name server learns mapping, it *caches* mapping, and *immediately* returns a cached mapping in response to a query
  - caching improves response time
  - cache entries timeout (disappear) after some time (TTL)
  - TLD servers typically cached in local name servers
- cached entries may be *out-of-date*
  - if named host changes IP address, may not be known Internet-wide until all TTLs expire!
  - *best-effort name-to-address translation!*

Getting your info into the DNS

Domain Name

ICANN

.in

Domain -

example: new startup "Network Utopia"

- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts NS, A RRs into .com TLD server:

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

free records

- create authoritative server locally with IP address `212.212.212.1`
  - type A record for www.networkuptopia.com
  - type MX record for networkutopia.com

DNS observations

C → S (DNS)

Adversary

google.com 2.2.2.2

Distributed Denial of Service → overwhelm server (Traffic limit 3 req/s)

Unencrypted

DDoS attacks

- bombard root servers with traffic TLD servers
  - not successful to date
  - traffic filtering
  - local DNS servers cache IPs of TLD servers, allowing root server bypass

e.g. via amazon / cloudflare

Spoofing attacks

- intercept DNS queries, returning bogus replies
  - DNS cache poisoning
  - RFC 4033: DNSSEC authentication services

Centralization of dns

Censorship → Network ISP can block some DNS request and can spoof the responses

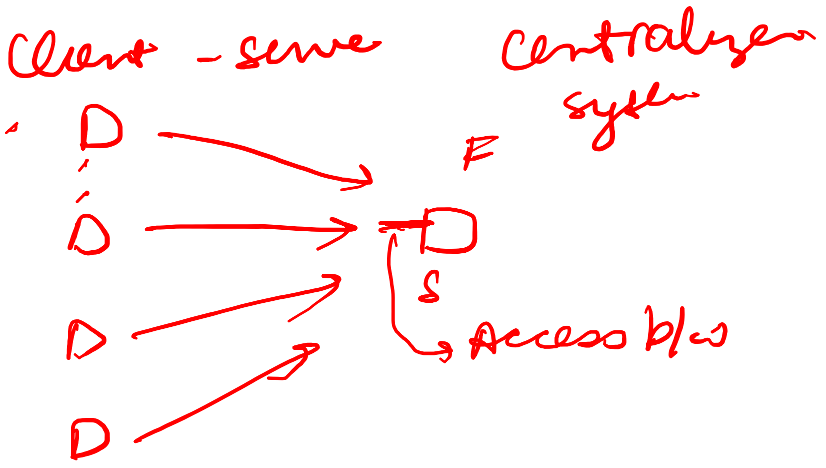
Great Firewall of China

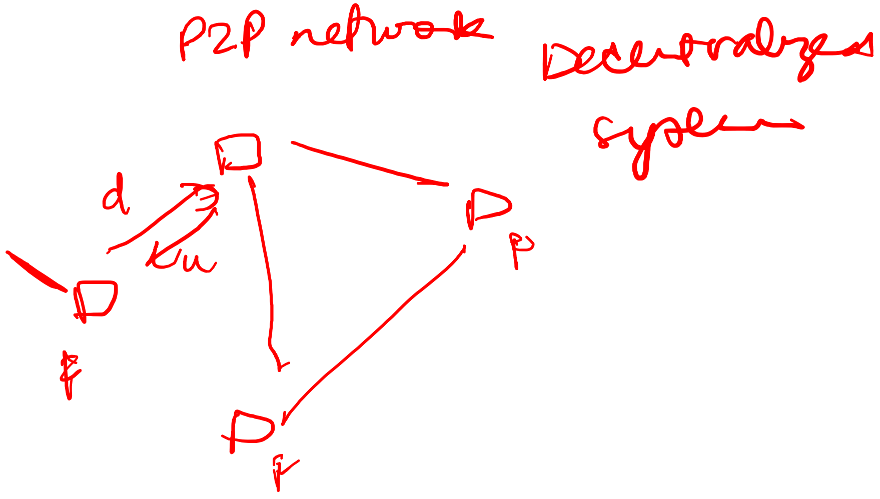
- Name servers hosted by third-party (e.g., cloudflare, amazon)
  - Why?
  - Single point of failure?

Recap: Application Layer

- HTTP
- Email
- DNS
- **P2P**
- Video streaming

Peer-to-peer architecture (P2P)





- *self scalability* – new peers bring new service capacity, and new service demands
- No single point of failure

- *No always-on server, clients can come and go anytime*
- Complex management

Why P2P for content distribution?

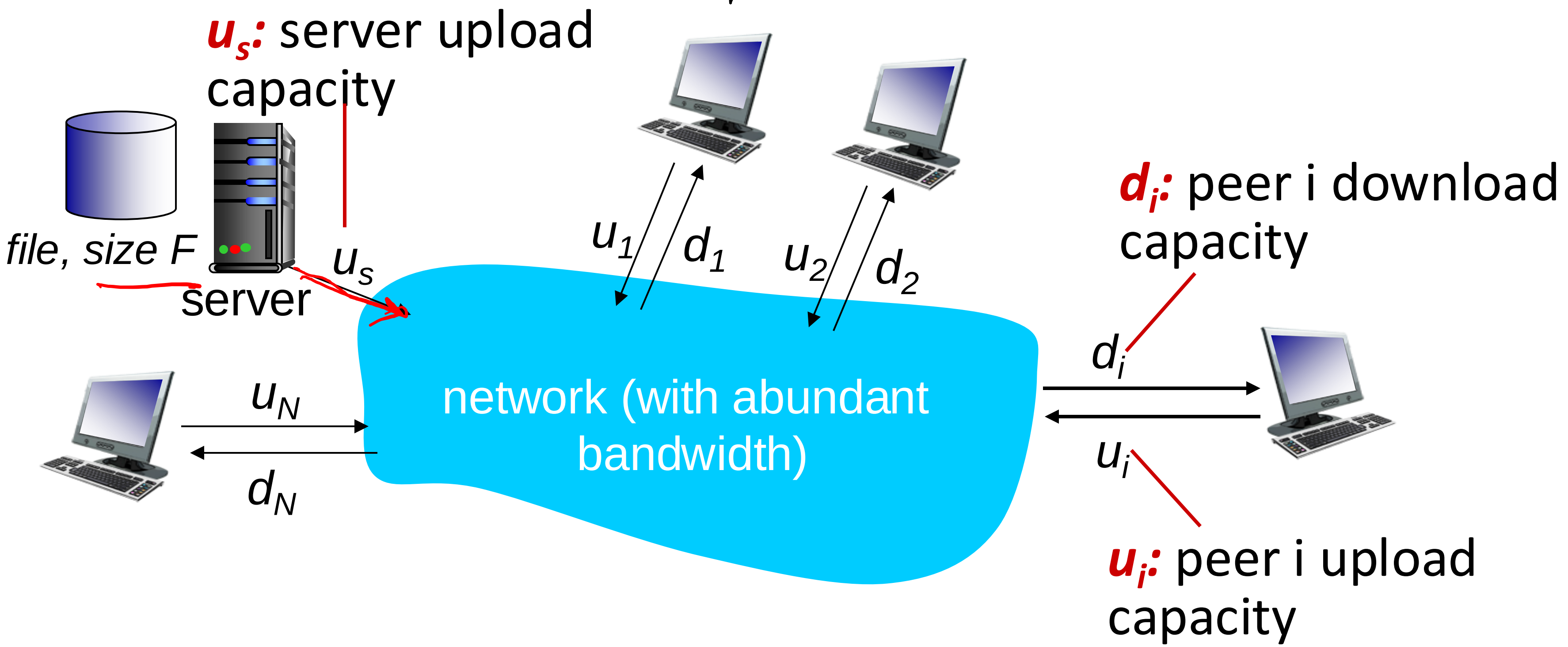
- Scales better than client-server architecture ***Why?***

Min TTD client-server architecture: $\max\left(\frac{NF}{u_s}, \frac{F}{d_{min}}\right)$

Min TTD

P2P architecture: $\max\left(\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s+\sum u_i}\right)$

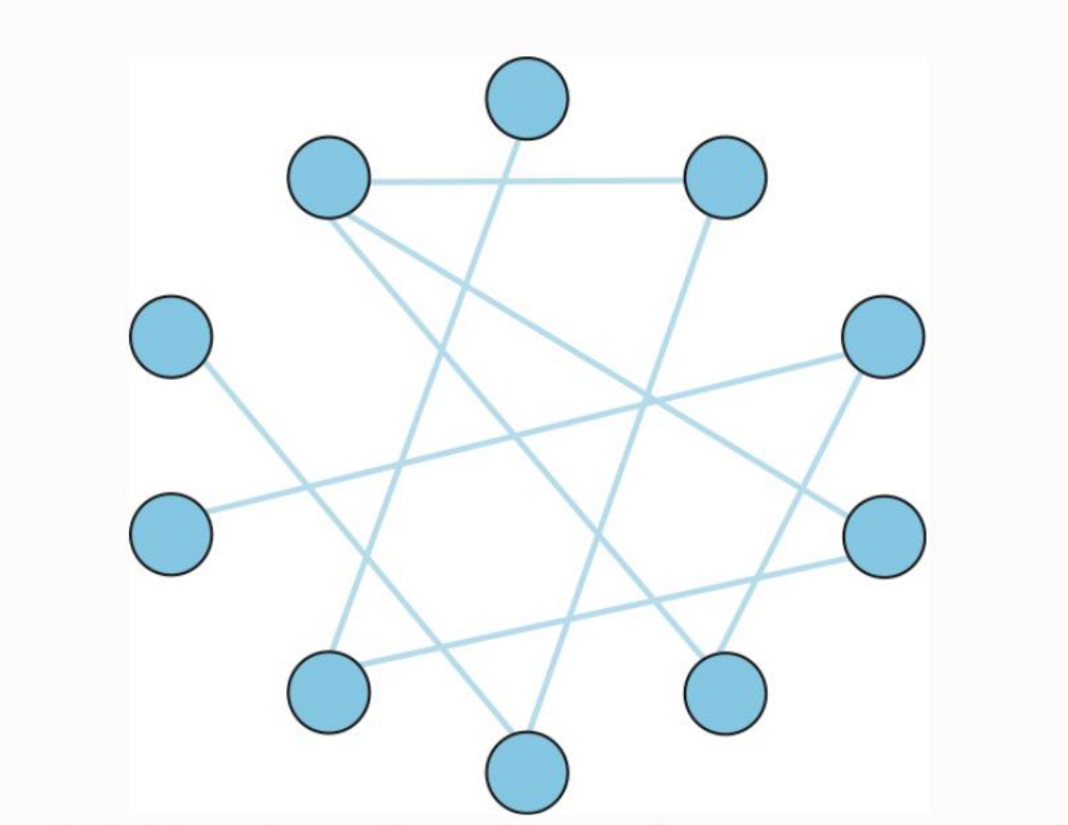
F



File Distribution in a P2P Network

- Two interesting questions
  - 1 → How to find a file?
  - 2 → How to download a file?

- Constraints:
  - → Not every node knows every other neighbor ((N can be very large) ↳ hard to keep
  - Nodes can come and go



Finding a File: Approaches

(NAPSTER)

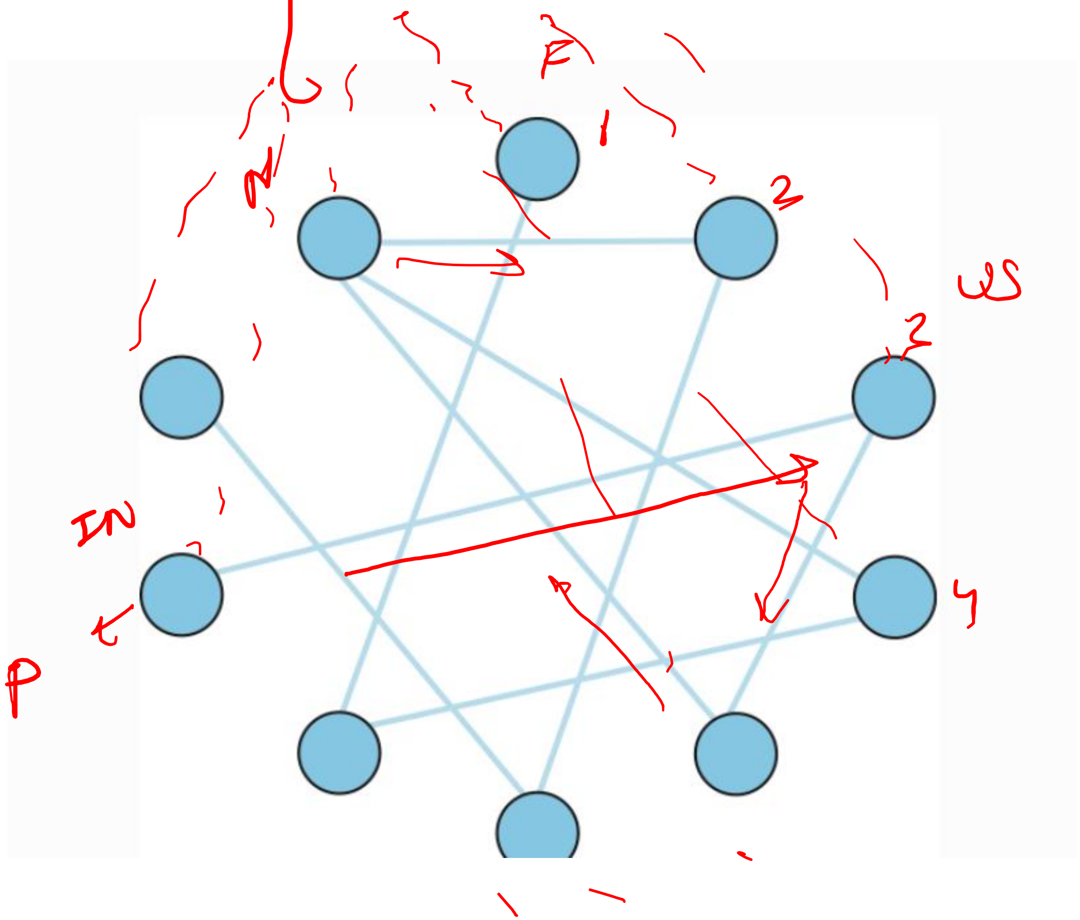
① Broadcast request

centralized serve

Database

Legally Accountable

- Approach #1:
  - Use a centralized server with 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 TTL to avoid indefinite broadcast messages
  - Cons: high overhead



Can we do better?