Lecture 4:

The Application Layer (part 2)

Katerina Argyraki, EPFL

Interface

- A point where two systems, subjects, organizations, … meet and interact.

Application Programming Interface

- Interface between application and transport layers
- A set of functions that are the only way for processes to exchange messages over the Internet

Network interface

- Interface between an end-system and the network
- A piece of hardware or software that sends and receives packets
- Example: your network card is a (hardware) network interface

DNS name

- Identifies a network interface
  = identifies an end-system

- Also called a "hostname"
  - an end-system is also called a "host"

URL

- Identifies a web object
  - example: www.epfl.ch/index.html

- Format: DNS name + file name
  - www.epfl.ch identifies a network interface
  - index.html identifies a file

Process name/address

- Identifies a process
  = app-layer piece of code
  - example: 128.178.50.12, 80

- Format: IP address + port number
  - 128.178.50.12 identifies a network interface
  - 80 identifies a process

Web request revisited

- You enter a URL into your web client
  - http://www.epfl.ch/index.html
- Web client extracts DNS name
  - www.epfl.ch
- Translates DNS name to IP address
  - 104.20.228.42
- Forms web-server process name
  - 104.20.228.42, 80

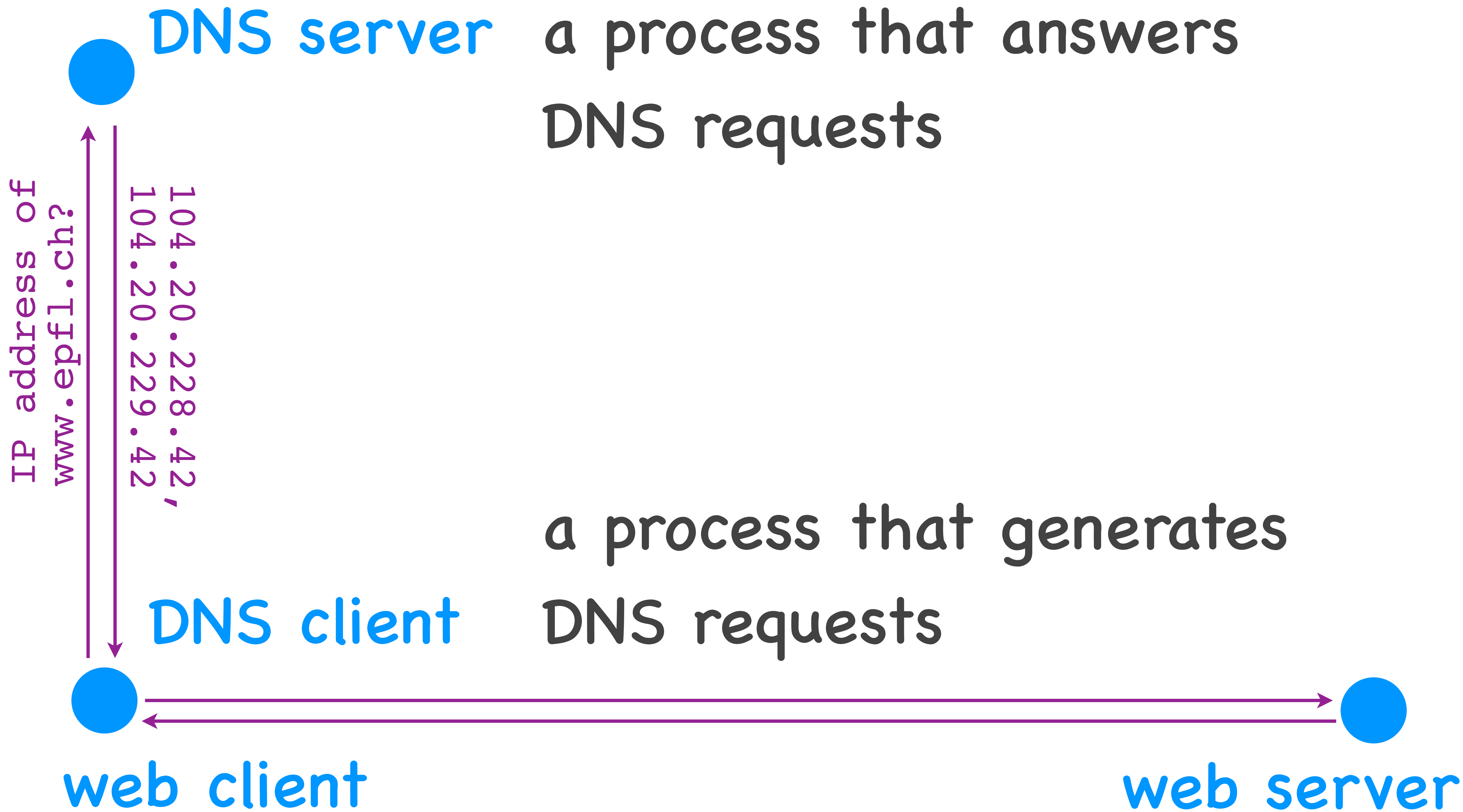
Web request revisited

- You enter a URL into your web client
  - www.epfl.ch/index.html
- Web client extracts DNS name
  - www.epfl.ch
- Translates DNS name to IP address
  - 104.20.228.42
- Forms web-server process name
  - 104.20.228.42, 80

Example 2: DNS

Design an application =

- Design the architecture
  - which process does what?
- Design the communication protocol
  - what sequences of messages can be exchanged?
- Choose the transport-layer technology
  - what kind of delivery is needed?


DNS server
a process that answers DNS requests
IP address of www.epfl.ch?
104.20.228.42, 104.20.229.42
a process that generates DNS requests
DNS client
web client
web server

www.epfl.ch	104.20.228.42, 104.20.229.42
www.search.ch	195.141.85.90
facebook.com	157.240.201.35
google.com	172.217.168.14
www.stanford.edu	34.196.104.129, 3.90.95.150

Could we have a single DNS server in the entire Internet?

Scalability (informally)

- Ability to grow

- As the system grows,
  it maintains its properties
  at a reasonable cost

Hierarchy of DNS servers

root servers

TLD (top-level domain) servers

authoritative servers

Hierarchy of DNS servers

root servers

.com servers .org servers .ch servers

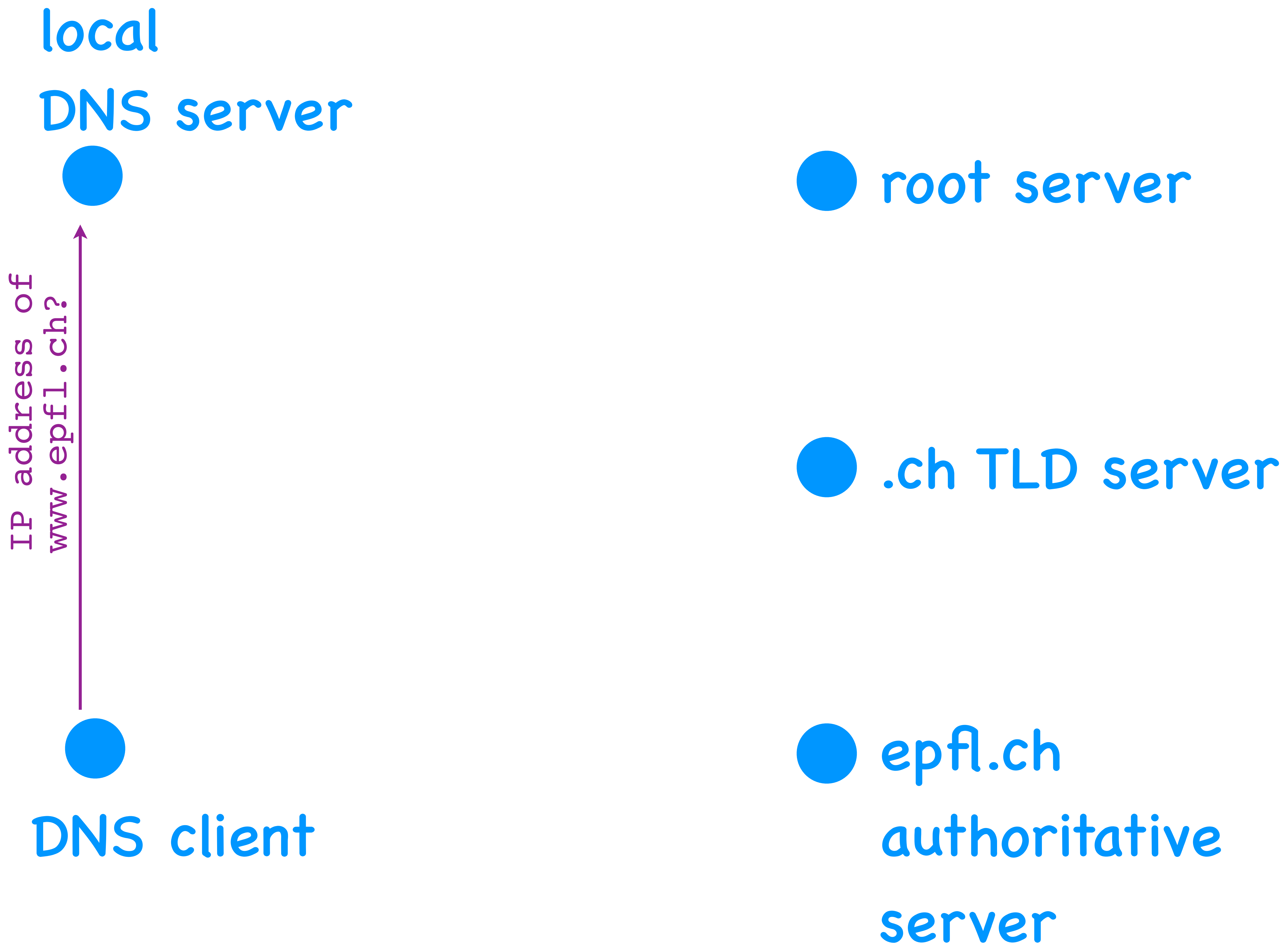
yahoo.com servers

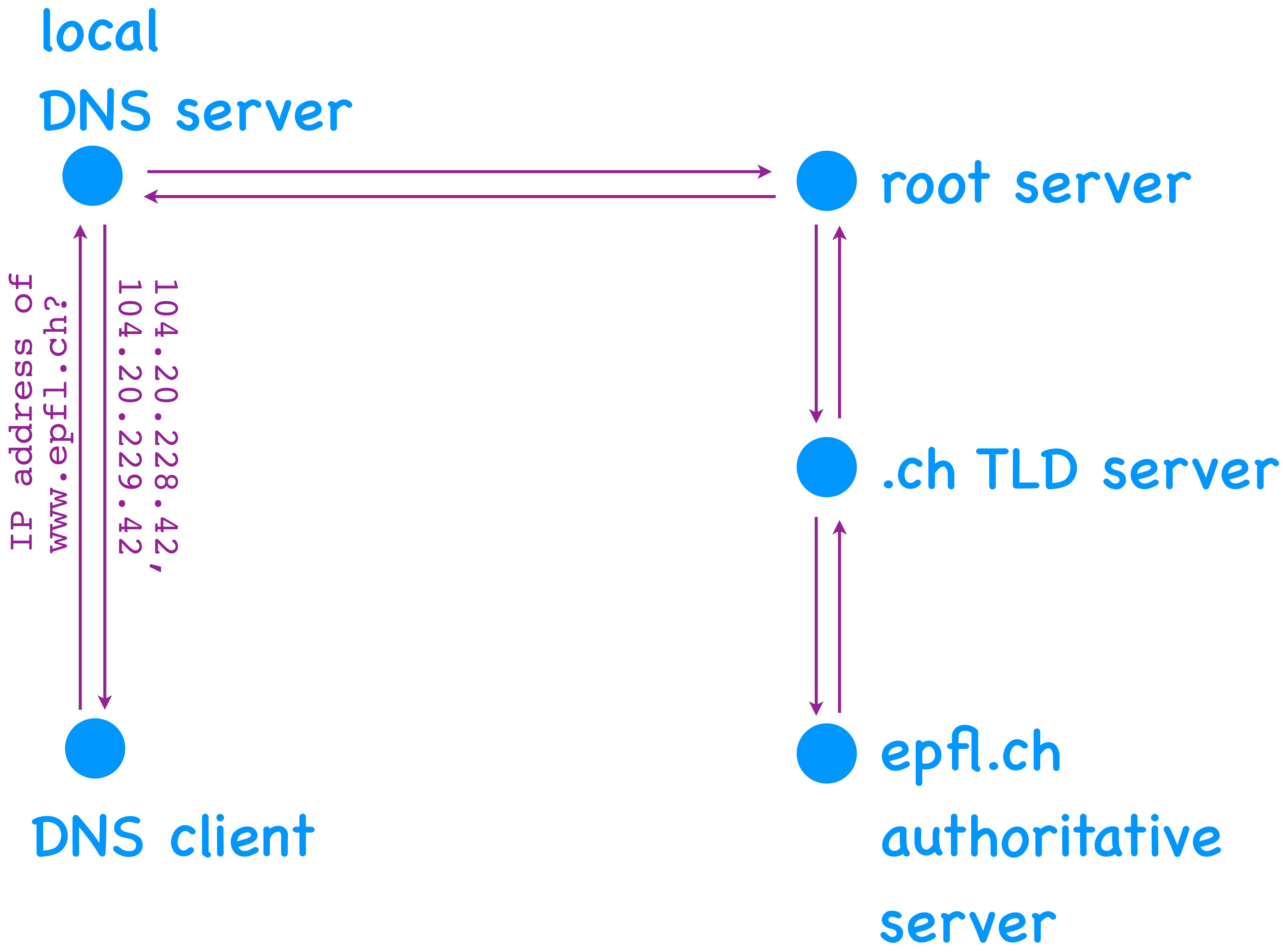
amazon.com servers

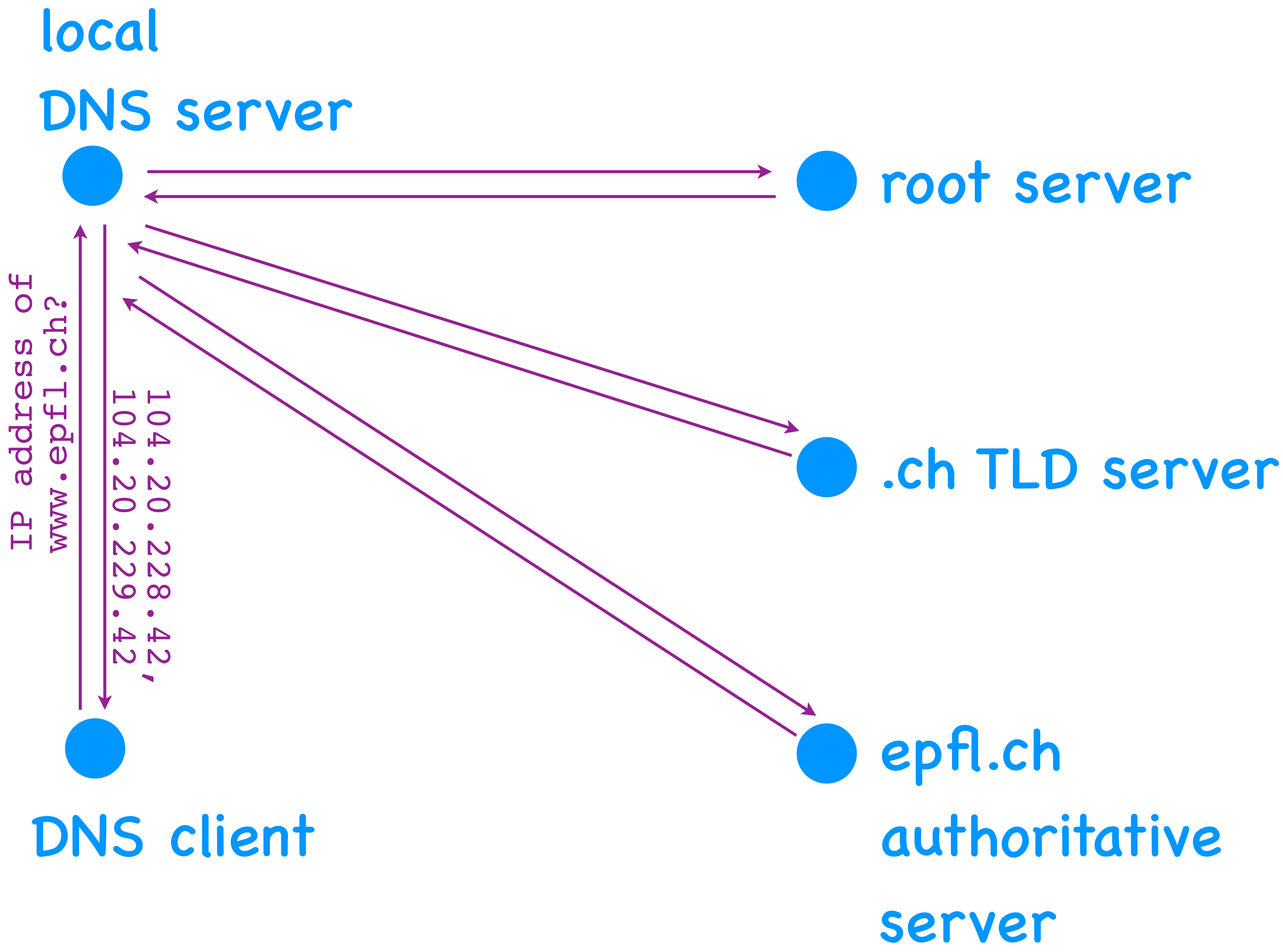
pbs.org servers

search.ch servers

epfl.ch servers


local
DNS server
root server
IP address of
www.epfl.ch?
.ch TLD server
epfl.ch
authoritative
server
DNS client


local
DNS server
root server
.ch TLD server
epfl.ch
authoritative
server
DNS client
IP address of
www.epfl.ch?
104.20.228.42,
104.20.229.42


local
DNS server
root server
.ch TLD server
epfl.ch
authoritative
server
DNS client
IP address of
www.epfl.ch?
104.20.228.42,
104.20.229.42

DNS processes

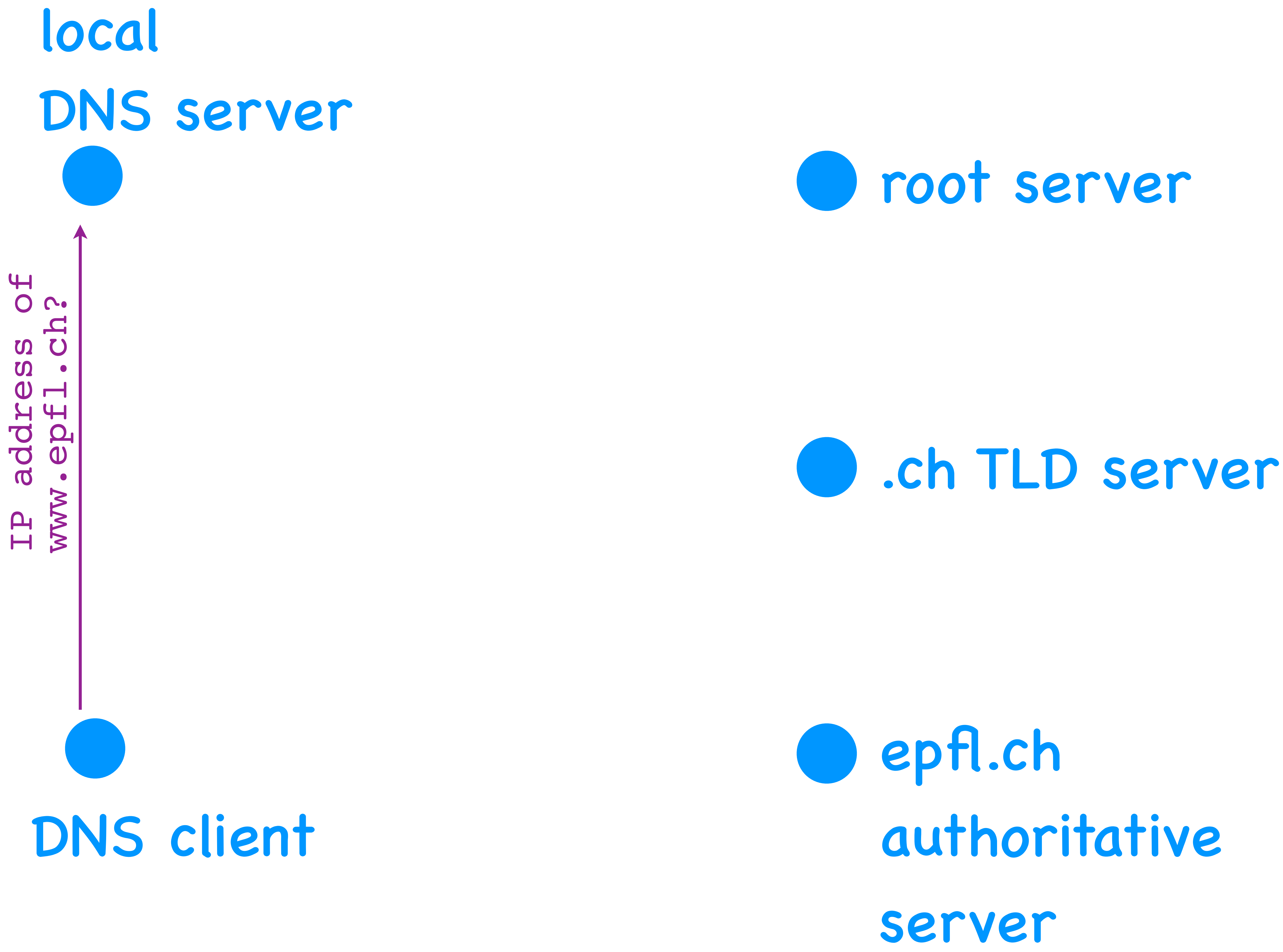
- DNS client
  - helps apps map DNS names to IP addresses
- Local DNS server
  - answers requests from nearby DNS clients
- Hierarchy of DNS servers
  - answers requests from local DNS servers

Hierarchy of DNS servers

- Three levels: root servers, TLD servers, authoritative servers
- Each node knows how to reach its children
  - root servers know TLD servers for each TLD
  - TLD servers know authoritative servers for each lower-level domain within their TLD

Hierarchy

- Universal technique for scaling large systems


local
DNS server
root server
IP address of
www.epfl.ch?
.ch TLD server
epfl.ch
authoritative
server
DNS client

How to prevent stale data?

local DNS server

www.epfl.ch –> 104.20.228.42
expires Dec. 31, 2019, 00:00 GMT

root server

www.epfl.ch –> 104.20.228.42
expires Dec. 31, 2019, 00:00 GMT

Mapping cannot change until expiration date

.ch TLD server

www.epfl.ch –> 104.20.228.42
expires Dec. 31, 2019, 00:00 GMT

DNS client

www.epfl.ch –> 104.20.228.42
expires Dec. 31, 2019, 00:00 GMT

epfl.ch server

www.epfl.ch –> 104.20.228.42
expires Dec. 31, 2019, 00:00 GMT

DNS caching

- All DNS clients and servers cache name-to-IP address mappings

- Reduces load at all levels
- Reduces delay experienced by apps

- Relies on expiration dates to ensure mapping freshness

Caching

- Universal technique
  for improving performance

- Challenge: stale data
  - option #1: dynamic check for staleness
  - may introduce significant delay

Caching

- Universal technique
  for improving performance

- Challenge: stale data
  - option #1: dynamic check for staleness
  - option #2: limit data update rate

Why do we use option #1 for web caching but option #2 for DNS caching?

Which of the following DNS servers is guaranteed to know the IP address of www.epfl.ch?

(a) A local DNS server.

(b) A root DNS server.

(c) An epfl.ch authoritative server.

Every Internet end-system must know the IP address of at least one

(a) DNS server.

(b) root DNS server.

(c) authoritative DNS server for each lower-level domain it wants to communicate with.

Every DNS server must know the IP address of at least one

(a) local DNS server.

(b) root DNS server.

(c) authoritative DNS server for each lower-level domain in the world.

Design an application =

- Design the architecture
  - which process does what?

- Design the communication protocol
  - what sequences of messages can be exchanged?

- Choose the transport-layer technology
  - what kind of delivery is needed?

DNS protocol elements

- Resource Record (RR)
  - piece of information,
    e.g., DNS name to IP address mapping
  - multiple types: A, CNAME, MX, SOA, ...
- Question: request for an RR
- Answer: response to a question

DNS protocol elements

- Message
  - contains sets of questions and answers
  - (plus other elements...)

- A DNS client and server or two DNS servers can exchange any sequence of messages

Design an application =

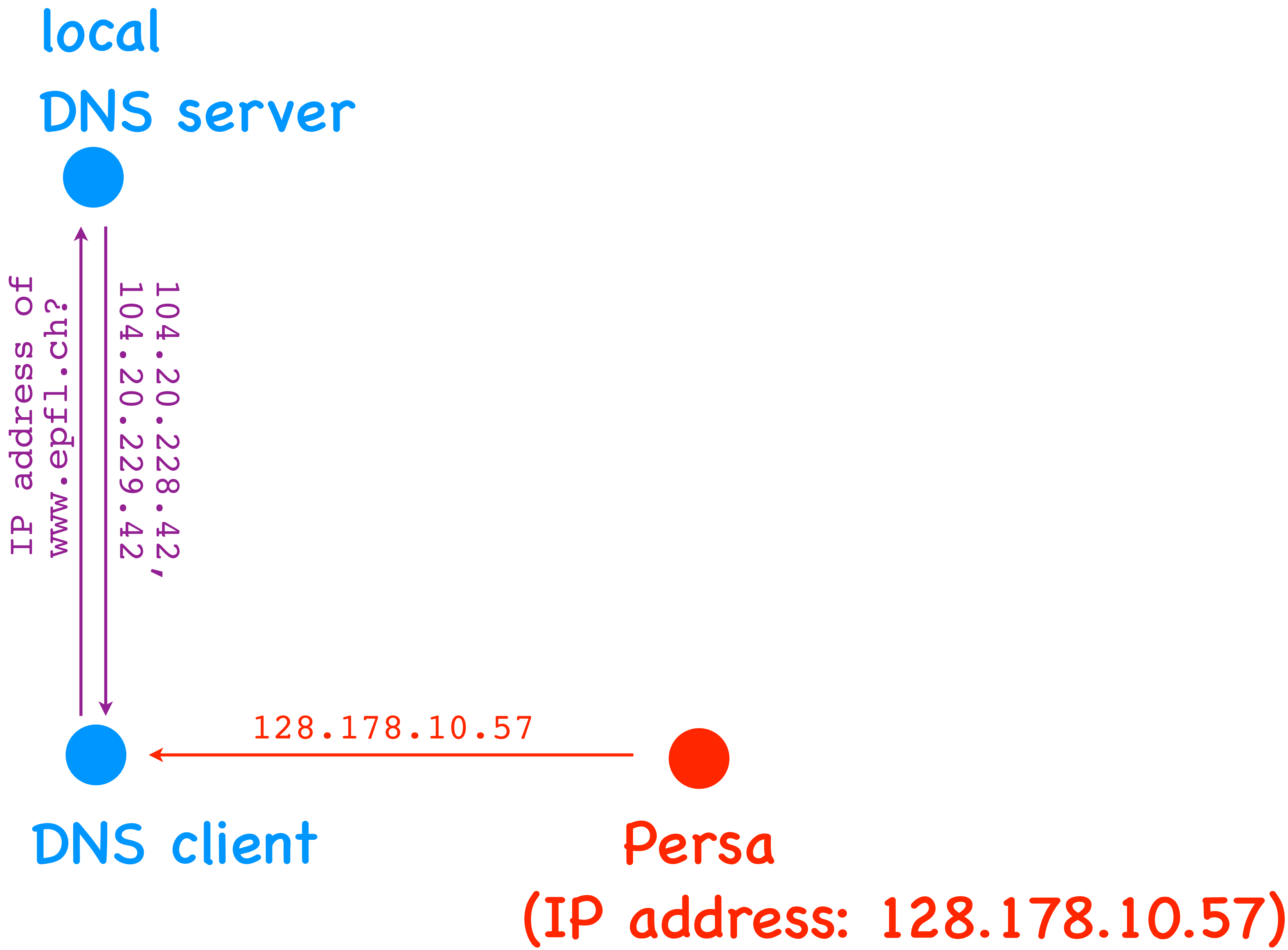
- Design the architecture
  - which process does what?
- Design the communication protocol
  - what sequences of messages can be exchanged?
- Choose the transport-layer technology
  - what kind of delivery is needed?

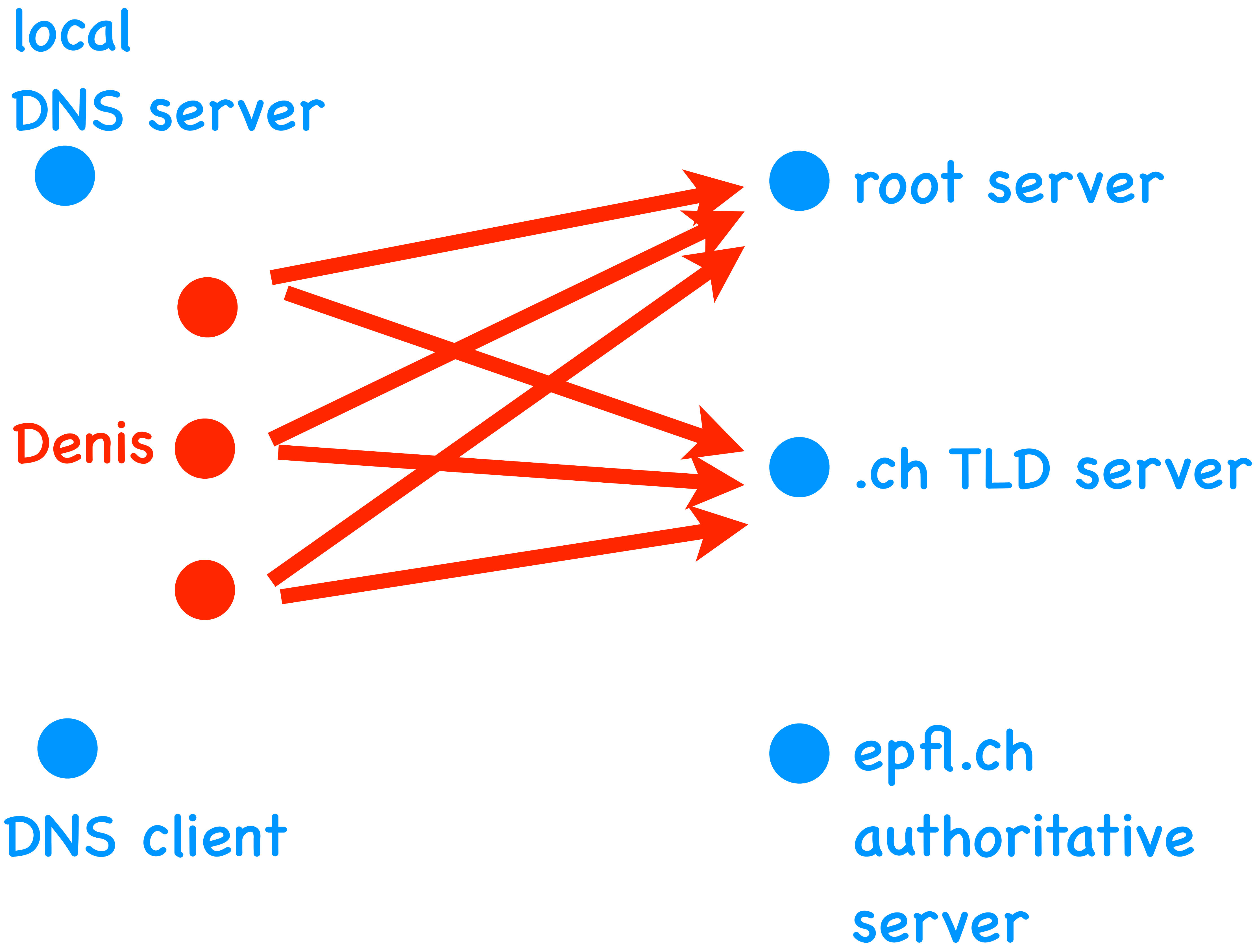
Would you use TCP or UDP
for DNS's transport layer? Why?

DNS transport layer

- UDP (for short exchanges)
  - does not make sense to pay the cost of TCP connection setup
- TCP (typically between DNS servers)
  - can amortise the cost of TCP connection setup

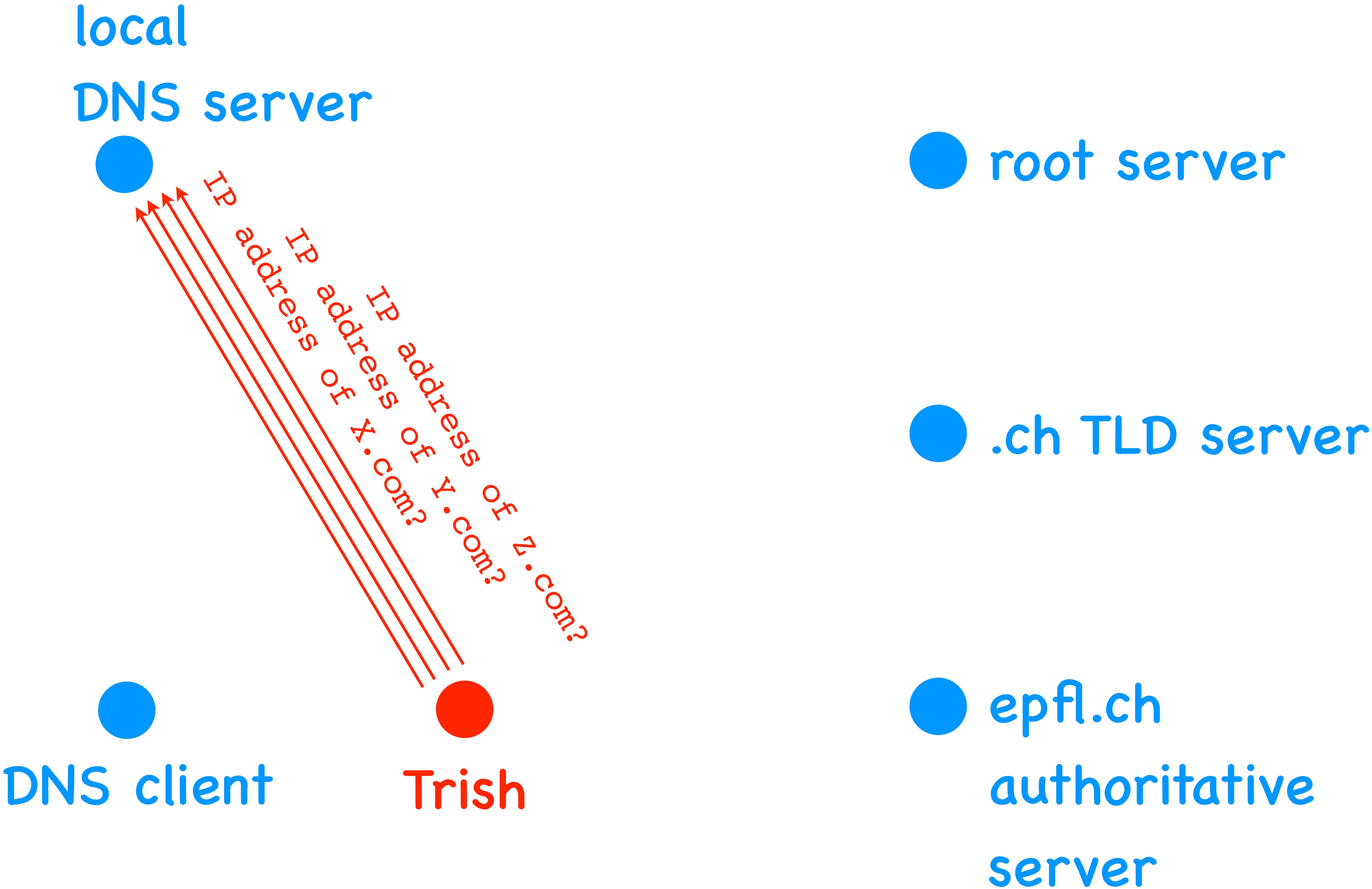
How can one attack the DNS system?


local
DNS server
IP address of www.epfl.ch?
104.20.228.42, 104.20.229.42
128.178.10.57
DNS client
Persa
(IP address: 128.178.10.57)


local
DNS server
root server
Denis
.ch TLD server
epfl.ch
authoritative
server
DNS client

Hierarchy

- Universal technique for scaling large systems
- Nodes that are high up in the hierarchy make good attack targets


local
DNS server
root server
.ch TLD server
epfl.ch
authoritative
server
DNS client
Trish
IP address of x.com?
IP address of y.com?
IP address of z.com?

Caching

- Universal technique
  for improving performance

- Trashing the cash is a potential
  vulnerability

Attacks against DNS

- **Impersonate** a DNS server and provide an incorrect mapping
- **DoS** the root servers and/or TLD servers
- **Trash the cache** of a DNS server to slow down its responses

667

Original story 1:26 pm EDT: Facebook—and apparently all the major services Facebook owns—are down today. We first noticed the problem at about 11:30 am Eastern time, when some Facebook links stopped working. Investigating a bit further showed major DNS failures at Facebook:

Jim Salter
@jrssnet

So, @facebook's DNS is broken this morning...

TL;DR: Google anycast DNS returns SERVFAIL for Facebook queries; querying a.ns.facebook.com directly times out.

```
oot@jrs-router:/etc/bind# dig @a.ns.facebook.com www.facebook.com

; <<>> DiG 9.11.3-1ubuntu1.15-Ubuntu <<>> @a.ns.facebook.com www.facebook.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
root@jrs-router:/etc/bind# dig @8.8.8.8 m.facebook.com

; <<>> DiG 9.11.3-1ubuntu1.15-Ubuntu <<>> @8.8.8.8 m.facebook.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 49071
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;m.facebook.com.                IN      A

;; Query time: 15 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Oct 04 11:46:05 EDT 2021
;; MSG SIZE  rcvd: 43
```

667

Original story 1:26 pm EDT: Facebook—and apparently all the major services Facebook owns—are down today. We first noticed the problem at about 11:30 am Eastern time, when some Facebook links stopped working. Investigating a bit further showed major DNS failures at Facebook:

Jim Salter
@jrssnet

So, @facebook's DNS is broken this morning...

TL;DR: Google anycast DNS returns SERVFAIL for Facebook queries; querying a.ns.facebook.com directly times out.

```
oot@jrs-router:/etc/bind# dig @a.ns.facebook.com www.facebook.com

; <<>> DiG 9.11.3-1ubuntu1.15-Ubuntu <<>> @a.ns.facebook.com www.facebook.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
root@jrs-router:/etc/bind# dig @8.8.8.8 m.facebook.com

; <<>> DiG 9.11.3-1ubuntu1.15-Ubuntu <<>> @8.8.8.8 m.facebook.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 49071
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;m.facebook.com.                IN      A

;; Query time: 15 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Oct 04 11:46:05 EDT 2021
;; MSG SIZE  rcvd: 43
```

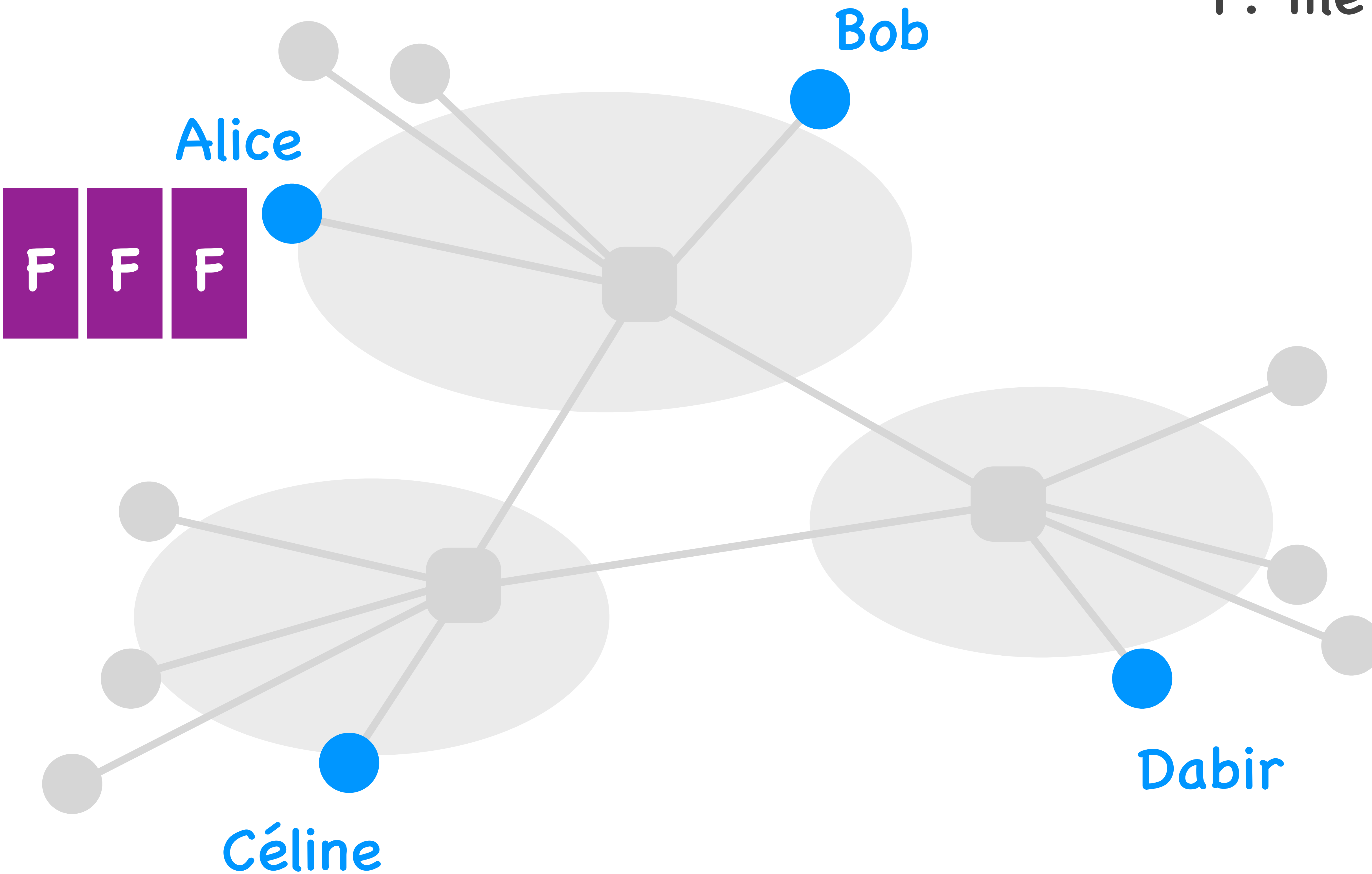
Example 3: BitTorrent (almost)

Design an application =

- Design the architecture
  - which process does what?
- Design the communication protocol
  - what sequences of messages can be exchanged?
- Choose the transport-layer technology
  - what kind of delivery is needed?

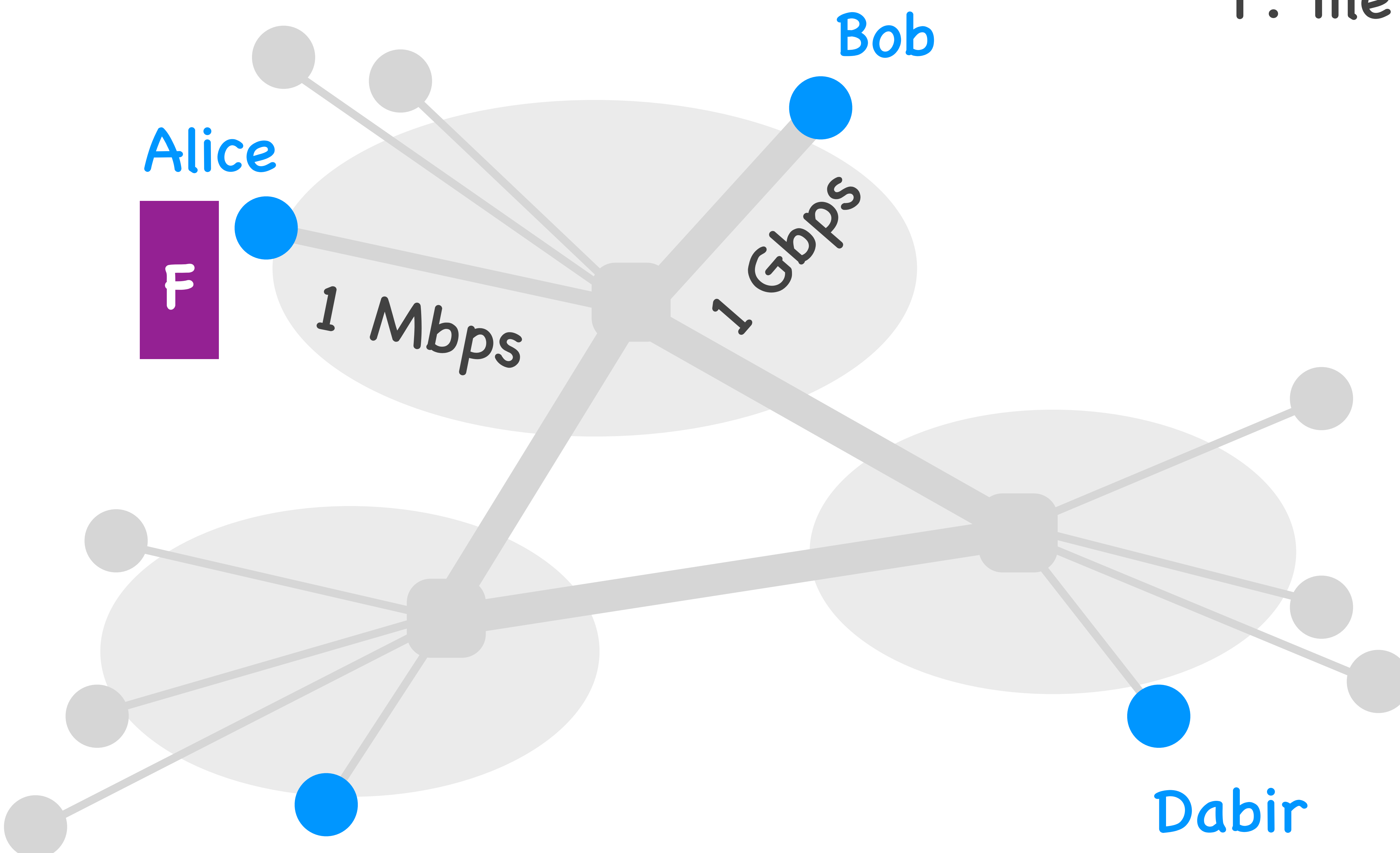
What does it mean that peer-to-peer "scales better" than client-server?

F: file size



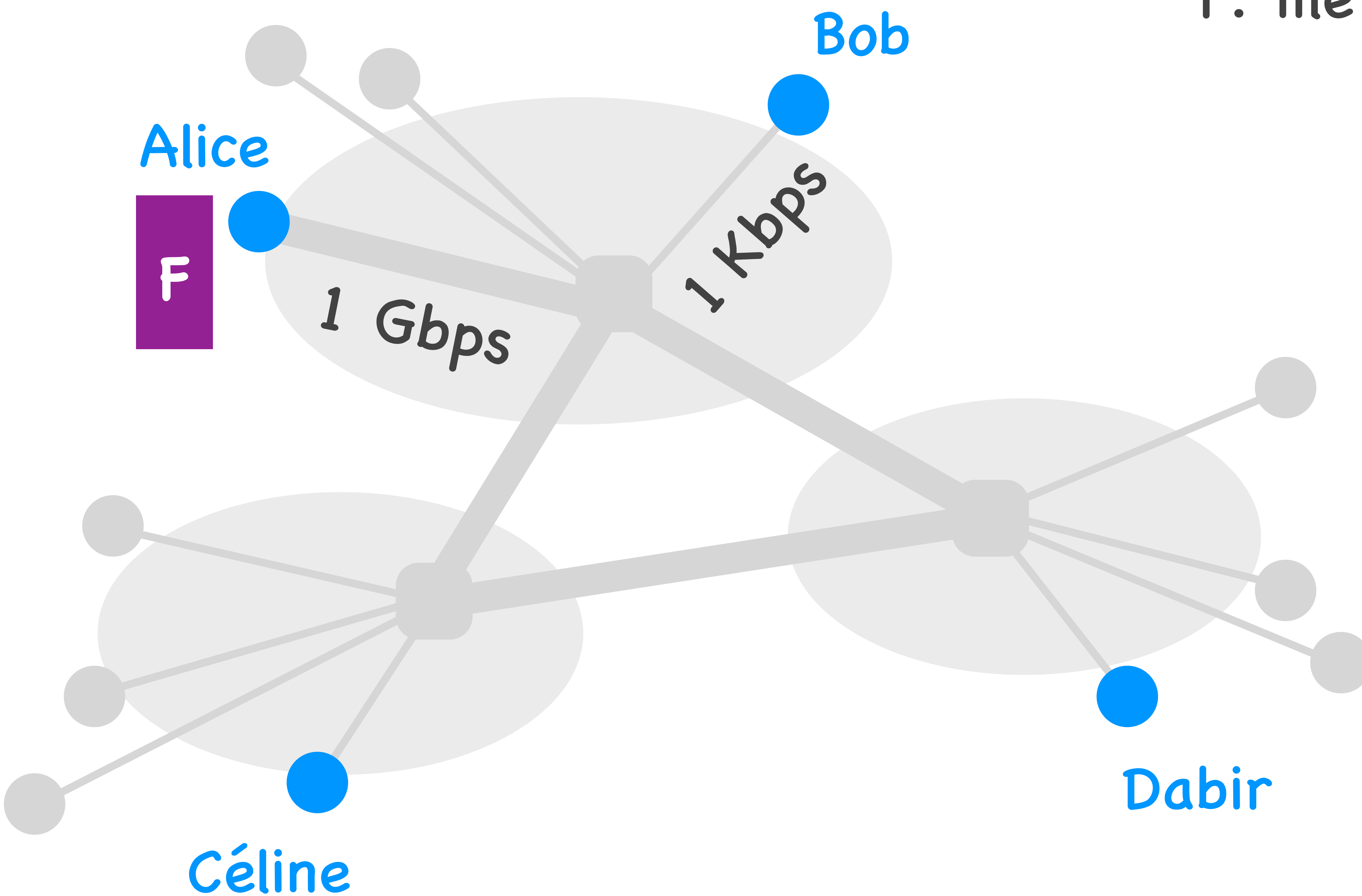
D_{cs} =

F: file size



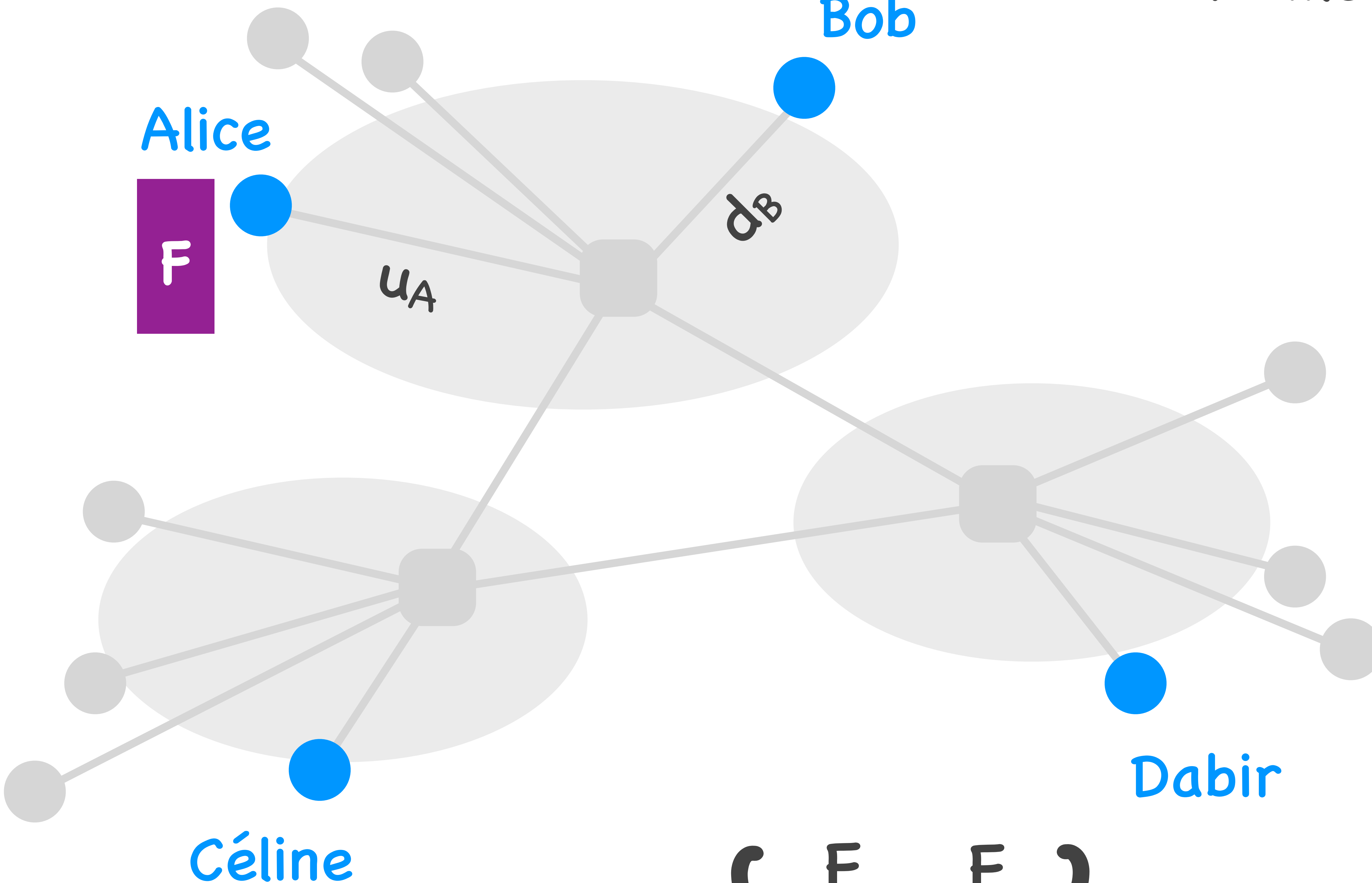
$D_{cs} =$

F: file size



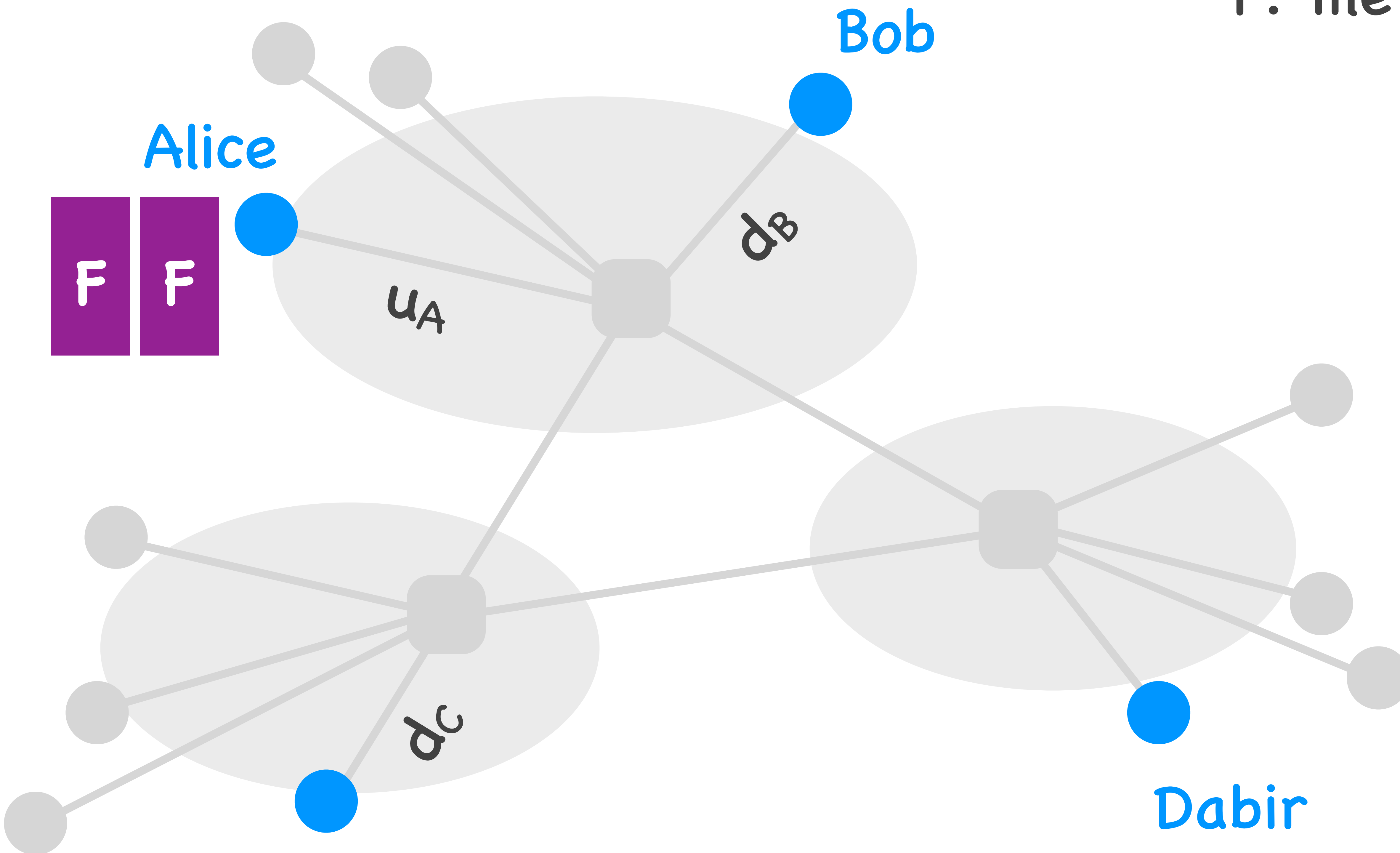
D_{cs} =

F: file size



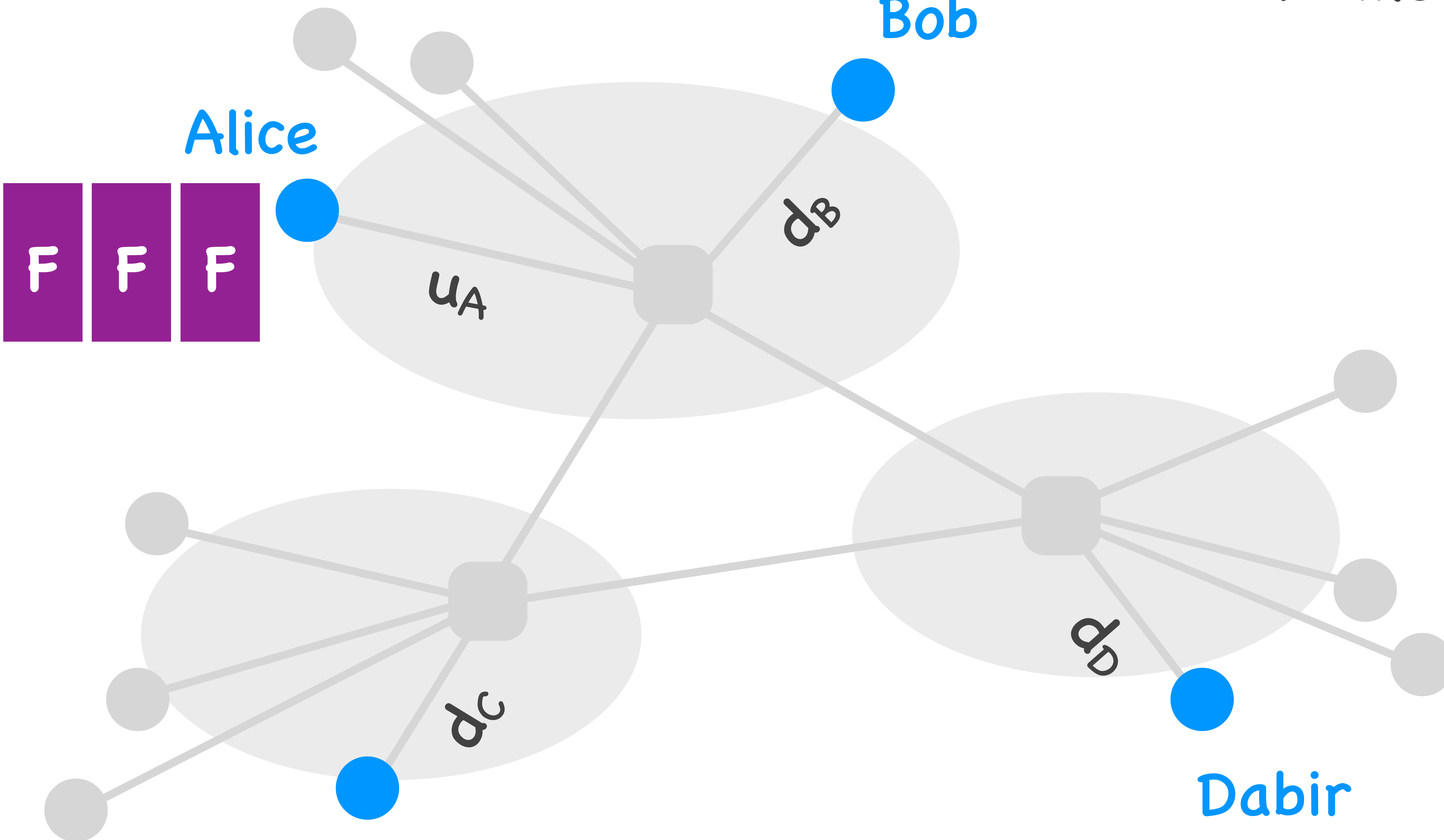
$$D_{cs} = \max\left\{ \frac{F}{u_A}, \frac{F}{d_B} \right\}$$

F: file size



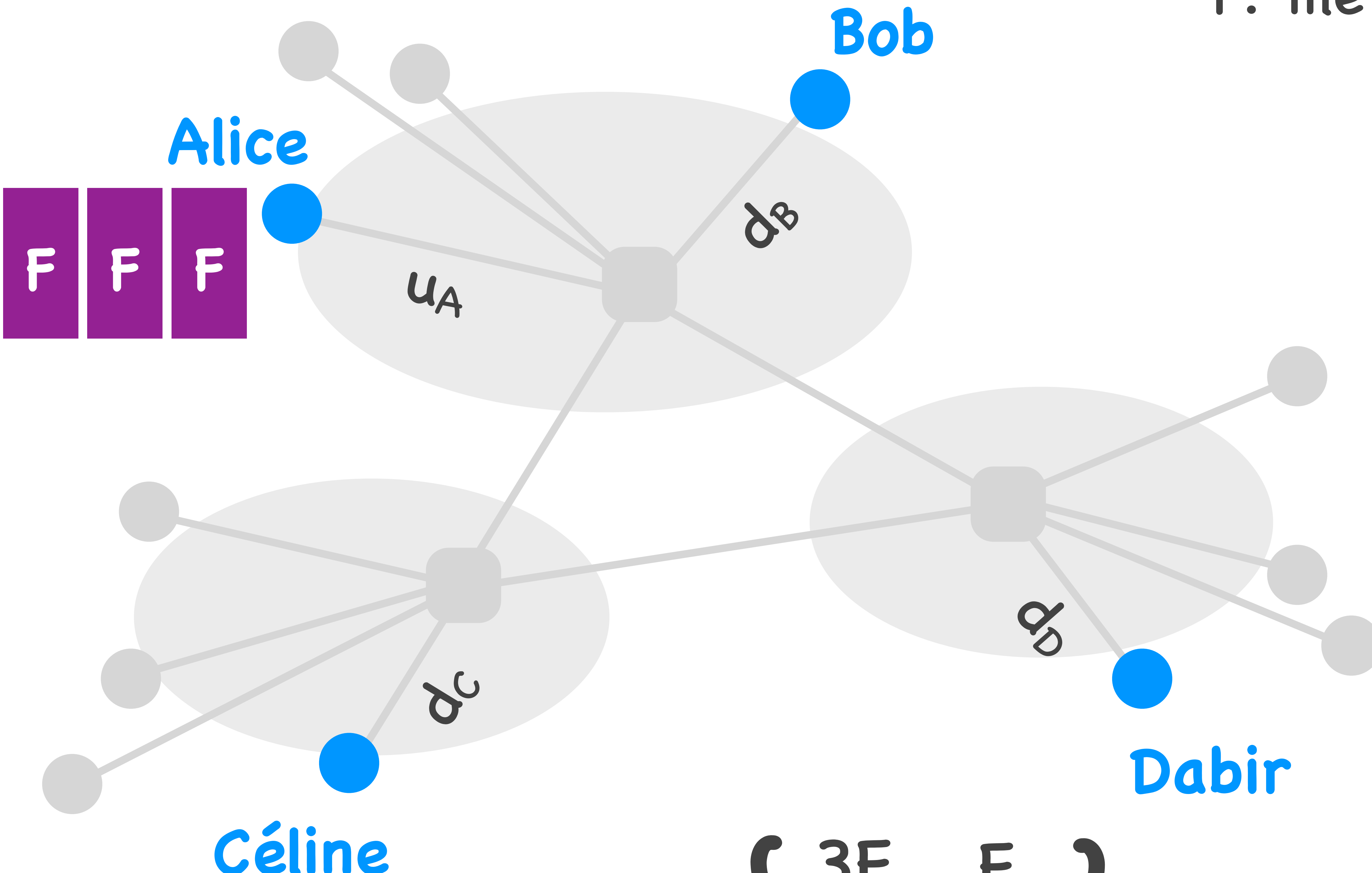
$$D_{cs} \geq \max\left\{ \frac{2F}{u_A}, \frac{F}{d_B}, \frac{F}{d_C} \right\}$$

F: file size



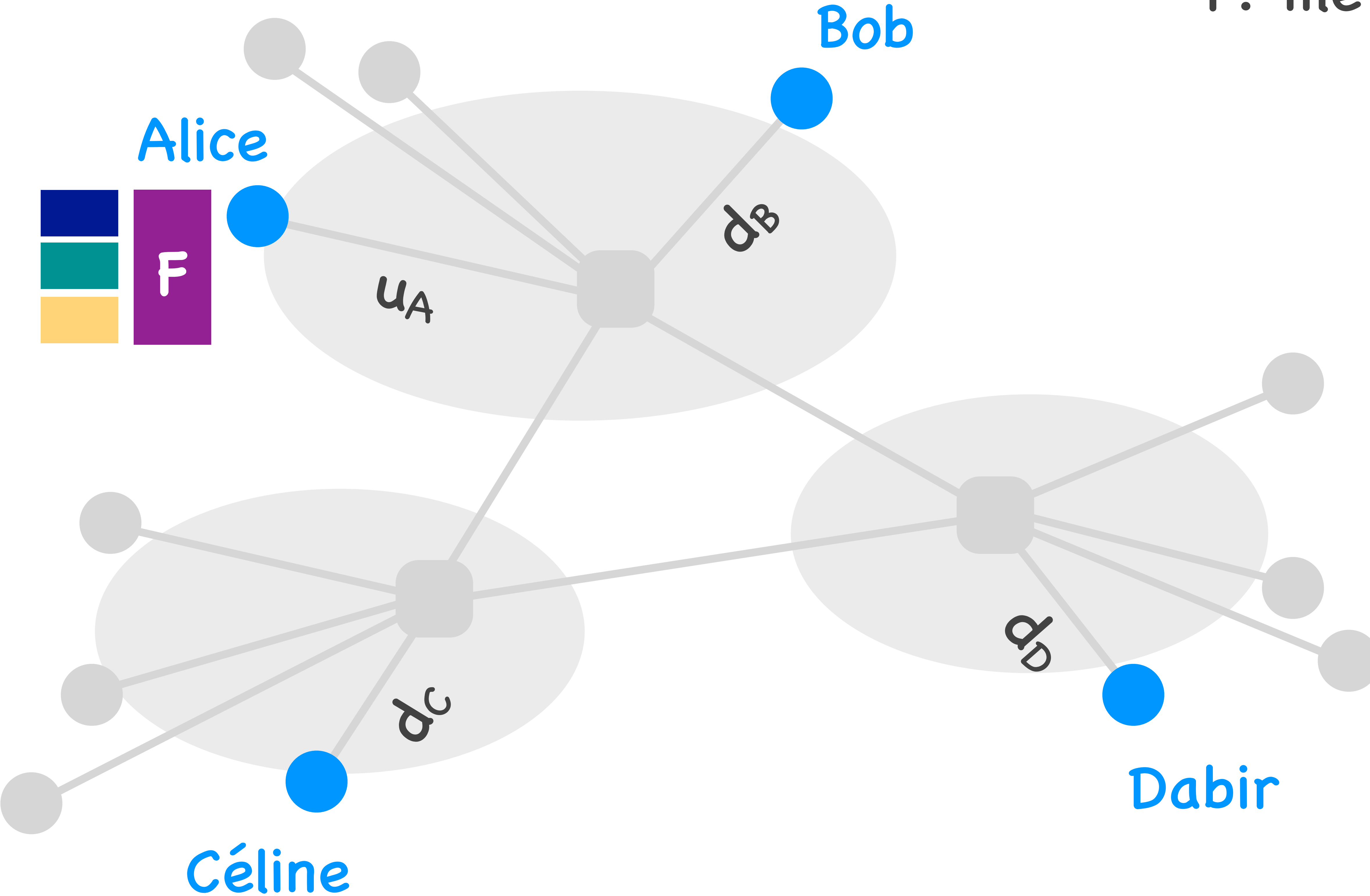
$$D_{cs} \geq \max\left\{\frac{3F}{u_A}, \frac{F}{d_B}, \frac{F}{d_C}, \frac{F}{d_D}\right\}$$

F: file size

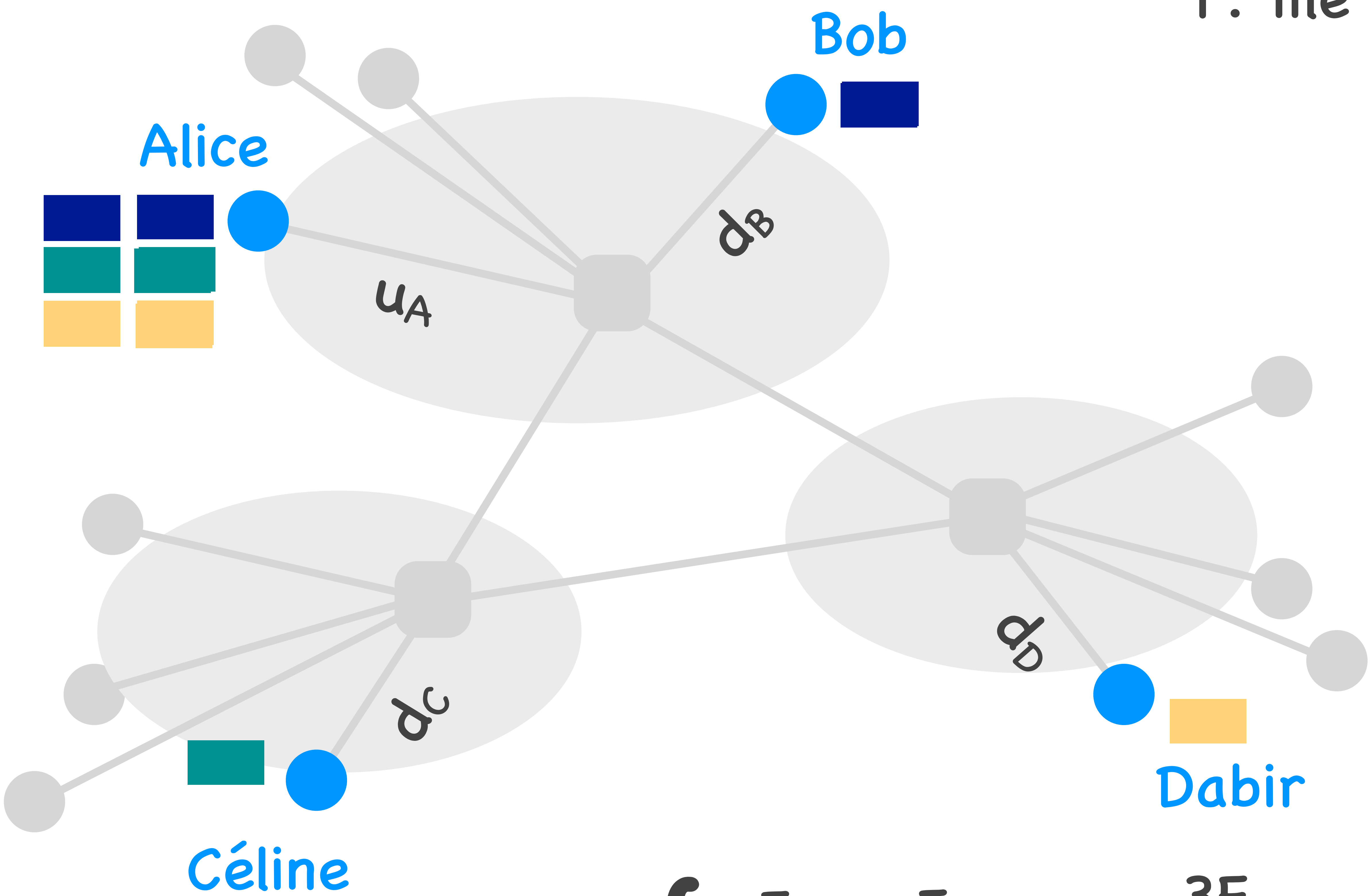


$$D_{cs} \geq \max\left\{\frac{3F}{u_A}, \frac{F}{d_{min}}\right\}$$

F: file size



F: file size



$$D_{P2P} \geq \max\left\{ \frac{F}{u_A}, \frac{F}{d_{min}}, \frac{3F}{u_A+u_B+u_C+u_D} \right\}$$

$$D_{cs} \geq \max \left\{ \frac{3F}{u_A}, \frac{F}{d_{min}} \right\}$$

$$D_{P2P} \geq \max \left\{ \frac{F}{u_A}, \frac{F}{d_{min}}, \frac{3F}{u_A+u_B+u_C+u_D} \right\}$$

number of downloaders

file size

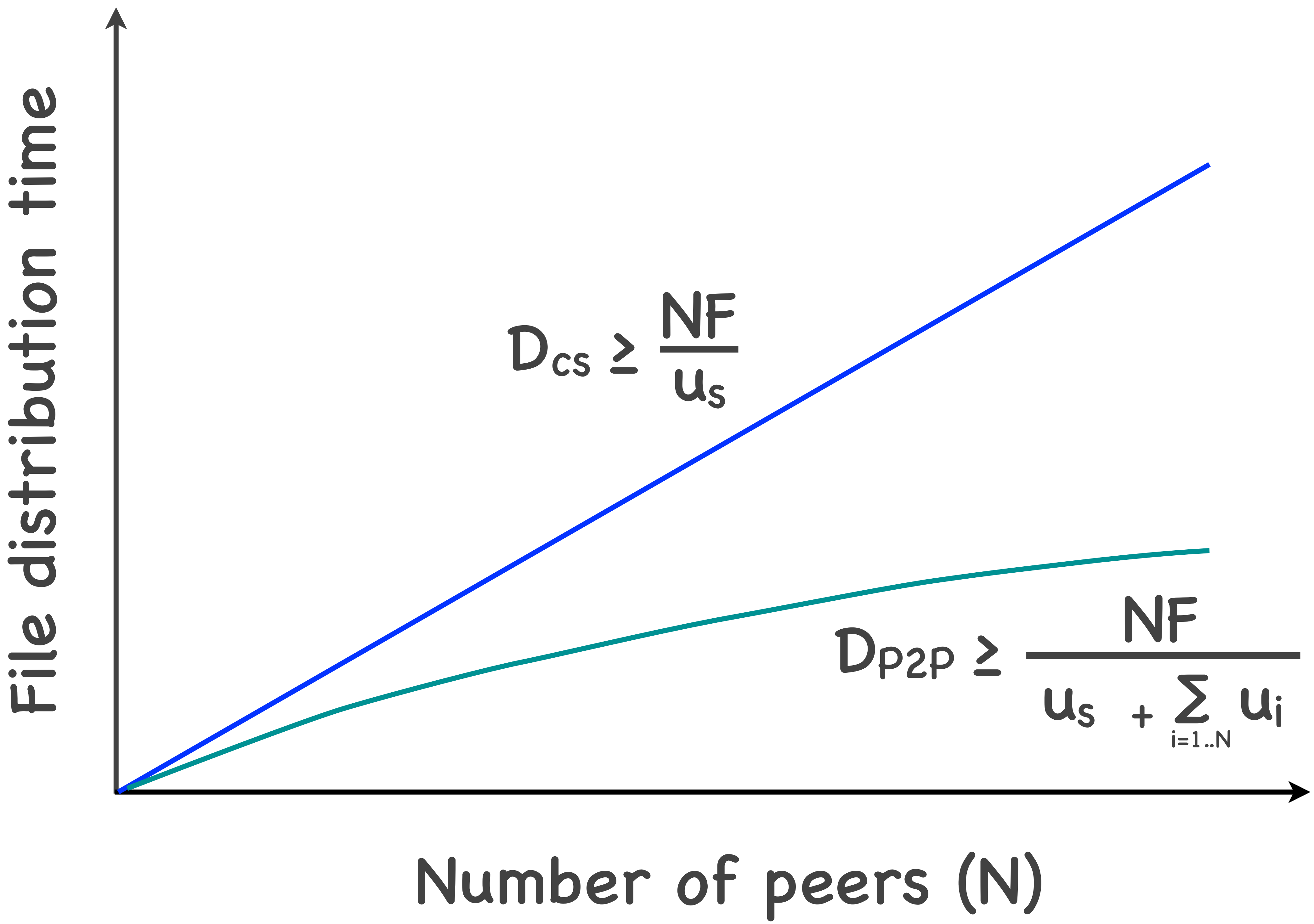
$$D_{cs} \geq \max\left\{ \frac{NF}{u_s}, \frac{F}{d_{min}} \right\}$$

$$D_{P2P} \geq \max\left\{ \frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1..N} u_i} \right\}$$

server upload capacity

min peer download capacity

aggregate peer upload capacity


File distribution time
$D_{cs} \geq \frac{NF}{u_s}$
$D_{P2P} \geq \frac{NF}{u_s + \sum_{i=1..N} u_i}$
Number of peers (N)

Scalability (informally)

- Ability to grow
- As the system grows,
  it maintains its properties
  at a reasonable cost

File distribution

- Client-server: time increases linearly with the number of clients
- Peer-to-peer: time increases sub-linearly with the number of peers
- Peer-to-peer scales better than client-server

How to retrieve content from a P2P file distribution system?

Content

- Set of data files
- Stored in a peer

Metadata file

- Special file that stores information about the data files
  - file identities
  - (optionally) location information
- May be on a web server or a peer
- BitTorrent: metadata file = .torrent file

Steps to retrieve content

- (Learn metadata file ID)
- Find metadata file location
- Get metadata file (from web server or peer), read data file IDs
- Find data file locations
- Get data files (from peers)

How to find file location?

Tracker

- An end-system that knows the locations of the files
  - the IP addresses of the peers that store each file

Distributed Hash Table (DHT)

- An distributed system that knows the locations of the files
  - the IP addresses of the peers that store each file

Tracker vs. DHT

- Different implementations of the same service
  - input: file ID
  - output: IP(s) of peer(s) that have the file
- Tracker is centralized, DHT is distributed/decentralized
- You don't need both

Steps to retrieve content

- (Learn metadata file ID)
- Find metadata file location
- Get metadata file (from web server or peer), read data file IDs
- Find data file locations
- Get data files (from peers)

Steps to retrieve content

- (Learn metadata file ID)
- Find metadata file location
- Get metadata file (from web server or peer), read data file IDs
- Find data file locations
- Get data files (from peers)

Where is the metadata file?

- Option #1: on a web server
  - you download it from the web server
  - you don't need to learn any ID
- Option #2: on a peer
  - you learn its ID from a web server
  - you learn its location from a tracker or DHT
- BitTorrent: metadata file ID = magnet link
  - e.g., magnet:xt=urn:btih:c12fe1c06bba25...

Steps to retrieve content

- (Learn metadata file ID)
- Find metadata file location
- Get metadata file (from web server or peer), read data file IDs
- Find data file locations
- Get data files (from peers)

Why use magnet links?

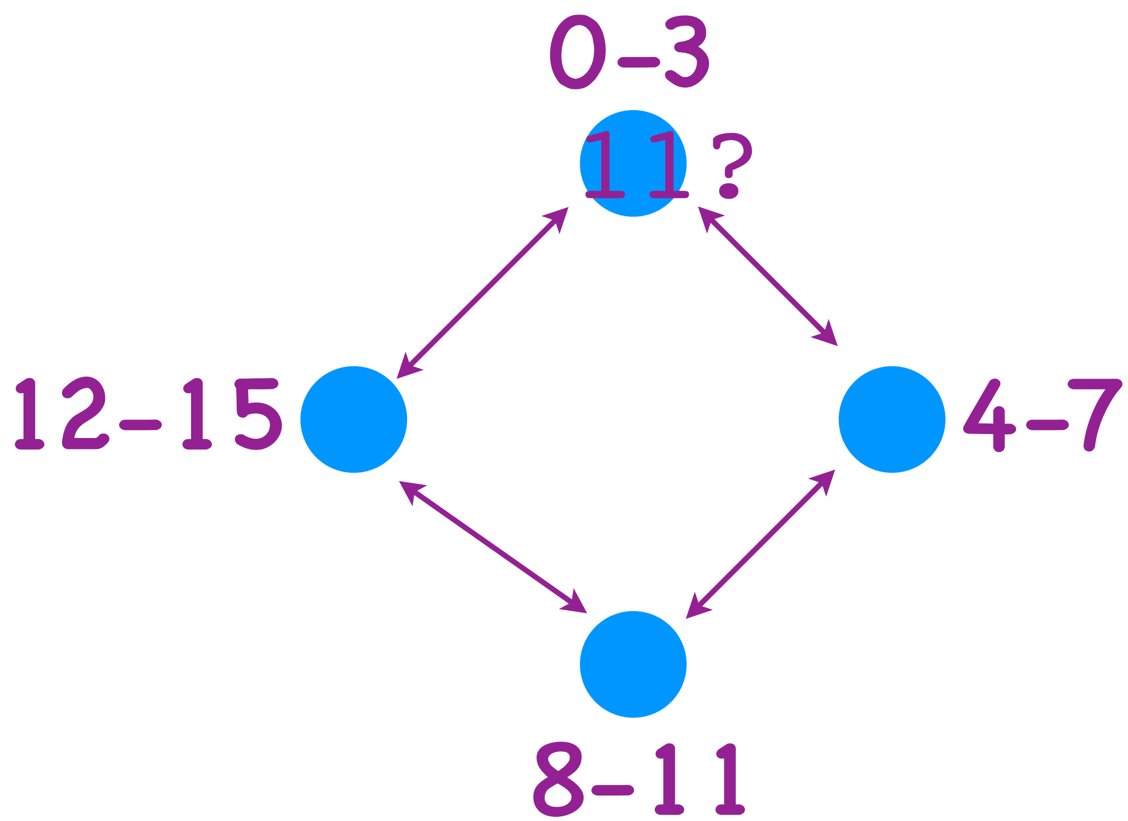
How does a DHT work?

Simplifying assumption

- We can have only 16 files
- File IDs are from 0 to 15

IP: 1.1.1.1
stored
file IDs: 1,5,12

IP: 4.4.4.4
stored
file IDs: 13



IP: 2.2.2.2
stored
file IDs: 10, 11

IP: 3.3.3.3
stored
file IDs: 3, 8

Basic DHT concepts

- File ID space partitioned:
  each peer "owns" an ID range
- Each peer knows the location
  of the files whose IDs it owns
- Each peer knows its own range
  + the ranges owned by its neighbors

Basic DHT concepts

- The DHT receives requests to locate a file ID
- Each peer forwards the request to the neighbor whose range is closest to the target file ID