

# Introduction to Networking

CAN201 – Lecture 3

Module Leader: Dr. Wenjun Fan Co-teacher: Dr. Fei Cheng

**HTTP = WEBPAGE?**

**A**pplication **P**rogramming **I**nterface

# RESTful API

- Representation State Transfer API
- A set of recommended styles/rules for API design
  - Use of standard HTTP methods (GET, POST, PUT, DELETE, etc.) to operate on resources.
  - Resources are identified via URIs, where each URI represents a resource.
  - Statelessness, meaning the server does not store any client session information between requests.
  - Use of standard HTTP status codes to represent the outcome of requests.
- Over 93% of developers using RESTful API in their projects

# Examples of RESTful

- Yes:
  - **Retrieving User Information**
  - Request: GET /users/123
  - Description: Retrieves information about the user with ID 123
- **Creating a New User**
  - Request: POST /users
  - Request Body: json
    - { "name": "Alice", "email": "alice@example.com" }
  - Description: Creates a new user on the server.

# Examples of RESTful

- Yes:
  - **Updating User Information**
  - Request: PUT /users/123
  - Request Body:json
    - { "email": "newemail@example.com" }
  - Description: Updates the email of the user with ID 123.
- **Deleting a User**
  - Request: DELETE /users/123
  - Description: Deletes the user with ID 123.

# Examples of RESTful

- **No:**
  - Request: GET /getUser?id=123
  - Description: Retrieves information about the user with ID 123.
  - **Reason: Using Verbs in the URI**
- Request: POST /users/getUserInfo
  - Request Body: json
    - { "id": 123 }
  - Description: Retrieves information about the user with ID 123.
  - **Reason: Using POST for Retrieval**

# Examples of RESTful

- **No:**
  - Request: GET  
/users/updateEmail?id=123&email=newemail@example.com
  - Description: Updates the email of a user.
  - Reason: Including Actions in the URI

# HTTP/2

- **Multiplexing:**
  - Multiple requests can be sent over a single connection simultaneously, reducing latency.
- **Header Compression:**
  - Compresses HTTP headers to reduce overhead and improve load times.
- **Server Push:**
  - Allows the server to send resources to the client before they are requested, improving page load speed.
- **Binary Framing:**
  - Uses binary format instead of text, making it more efficient to parse and less error-prone.

# HTTP/3

- **Based on QUIC Protocol:**
  - Uses UDP instead of TCP, reducing connection setup time and providing better performance in high-latency networks.
- **Improved Multiplexing:**
  - Eliminates head-of-line blocking issues seen in HTTP/2, further enhancing performance.
- **Better Connection Resumption:**
  - Faster connection recovery and continuation even if network changes occur (e.g., switching between Wi-Fi and mobile data).
- **Built-in Security:**
  - Integrates TLS (Transport Layer Security) for encrypted and secure data transmission by default.

# Lecture III – Application Layer (2)

1. Domain Name System (DNS)
2. P2P Applications
3. Socket Programming



# Identification



Fei Cheng

01000010XX

32031119XXXXXXXXXX

G46564XXX

- You and me:
  - Uni-ID, CNID / Passport, Name, Nickname
- Internet hosts:
  - IP address (32 bit for IPv4) - used for addressing datagrams
  - “Name”, eg. [www.xjtu.edu.cn](http://www.xjtu.edu.cn) - used by humans
- Q: How to map between IP address and name?

L  
To  
Z

Auckland 2004

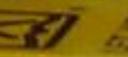
 **Yellow Pages®**

A  
To  
K

Auckland 2004

 **Yellow Pages®**

Auckland 2004

 **Yellow Pages®**  
LEAD THE WAY TO VALUE



Telecom



Telecom

# DNS: domain name system

- **Application-layer protocol:**
  - C/S architecture
  - UDP (port 53)
  - hosts, name servers communicate to resolve names (name / address translation)
- **Distributed database implemented in hierarchy of many name servers**

# DNS: services, structure

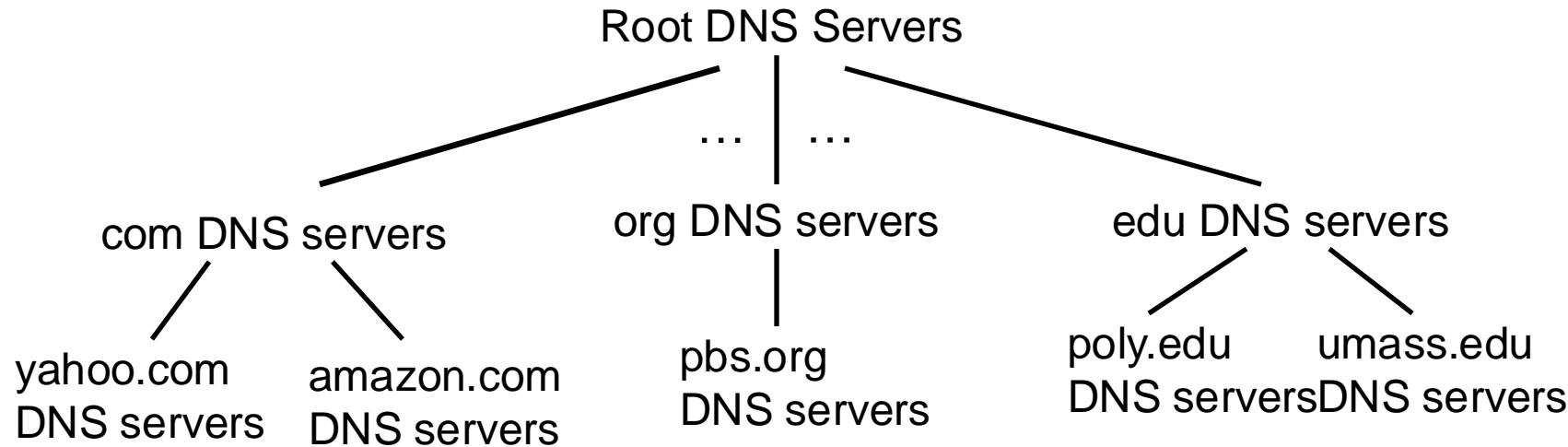
- **DNS services**

- Hostname to IP address translation(A)
- Host aliasing (cname)
  - canonical, alias names
- Mail server aliasing(mx)
- Load distribution
- Replicated Web servers: many IP addresses correspond to one name

- Why not centralize DNS?

- Single point of failure
- Traffic volume
- Distant centralized database
- Maintenance

# DNS: a distributed, hierarchical database

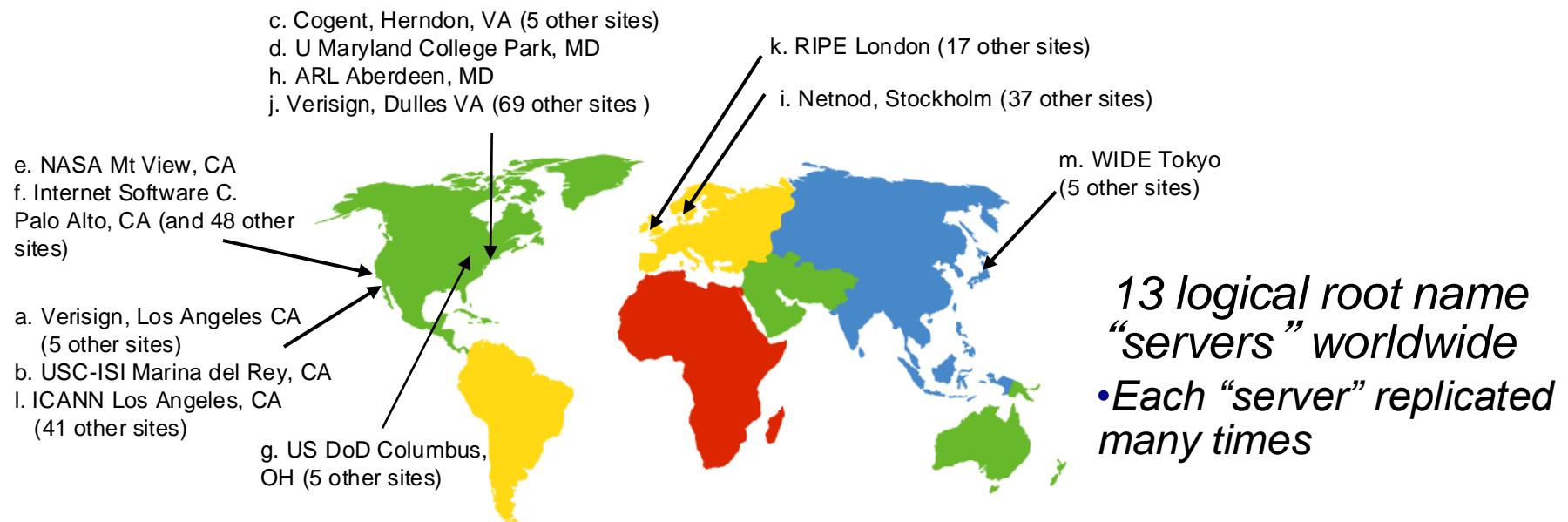


**Client wants IP for [www.amazon.com](http://www.amazon.com):**

- Client queries root server to find com DNS server
- Client queries .com DNS server to get amazon.com DNS server
- Client queries amazon.com DNS server to get IP address for [www.amazon.com](http://www.amazon.com)

# DNS: root name servers

- Contacted by local name server that can not resolve name
- Root name server:
  - Contacts authoritative name server if name mapping not known
  - Gets mapping
  - Returns mapping to local name server



# TLD, authoritative servers

- **Top-level domain (TLD) servers:**

- Responsible for com, org, net, edu, aero, jobs, museums, and all Top-level country domains, e.g.: cn, uk, fr, ca, jp

- *Eg.:*

- *Network Solutions* maintains servers for .com TLD
- *Educause* for .edu TLD (<https://net.educause.edu/>)

- **Authoritative DNS servers:**

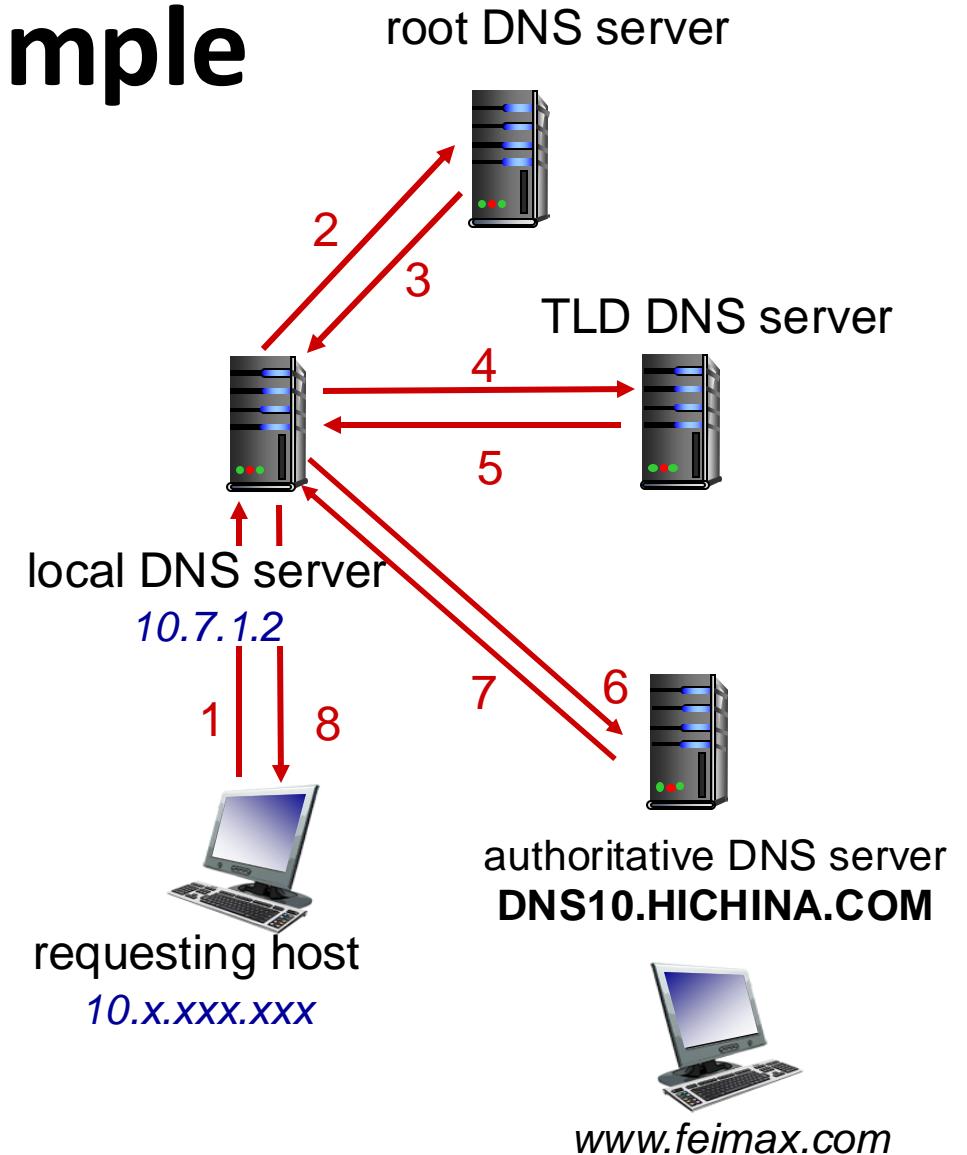
- Organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- Can be maintained by organization or service provider

# Local DNS name server

- Does not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one
  - Also called “default name server”
- When host makes DNS query, query is sent to its local DNS server
  - Has local cache of recent name-to-address translation pairs (but may be out of date!)
  - Acts as proxy, forwards query into hierarchy

# DNS name resolution example

- Host at XJTLU wants IP address for [www.feimax.com](http://www.feimax.com)
- Iterated query:
  - contacted server replies with name of server to contact
  - “I don’t know this name, but ask this server”

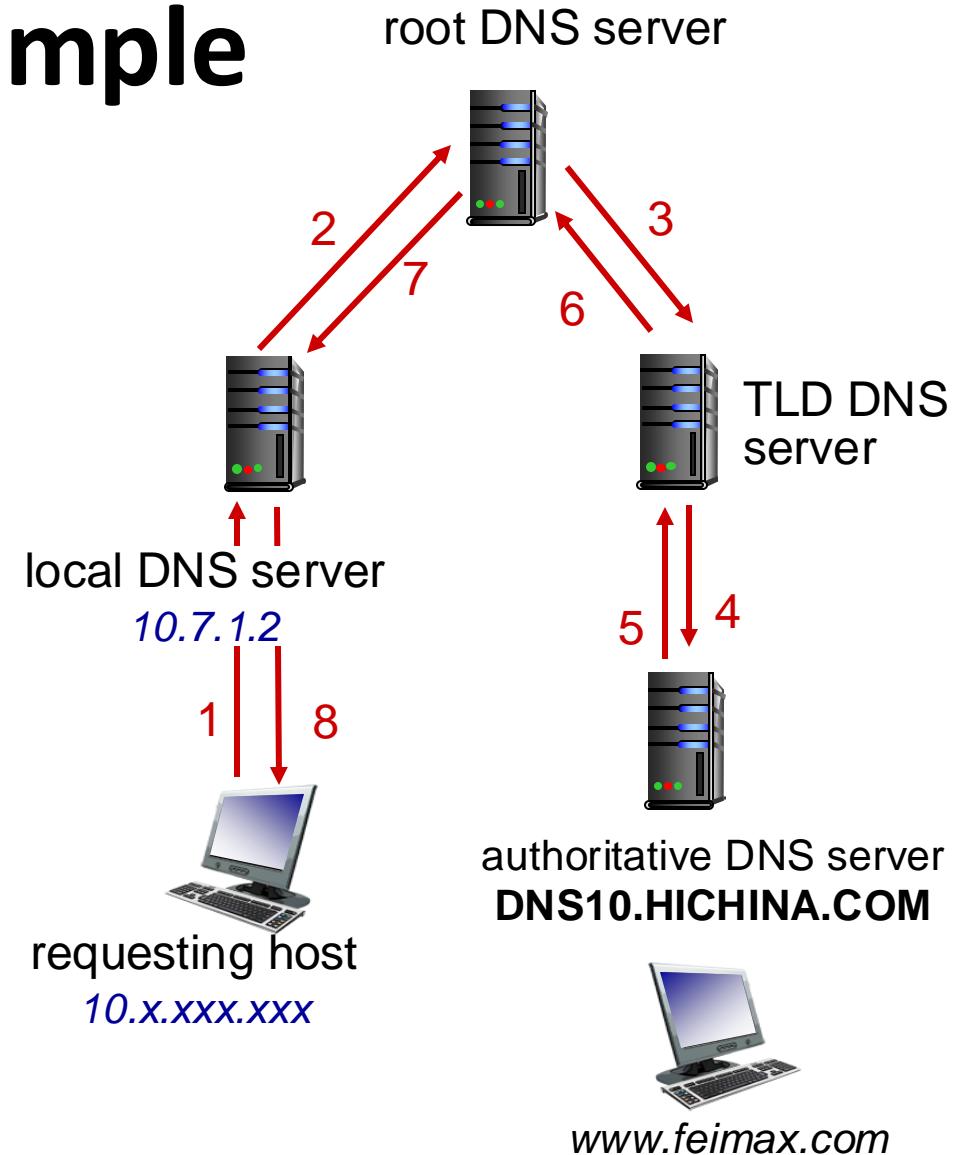


# DNS name resolution example

- **Recursive query:**

- Puts burden of name resolution on contacted name server
- Heavy load at upper levels of hierarchy

dig +trace



# DNS: caching, updating records

- Once (any) name server learns mapping, it *caches* mapping
  - Cache entries timeout (disappear) after some time (TTL)
  - TLD servers typically cached in local name servers
    - thus root name servers not often visited
- Cached entries may be *out-of-date*
  - If name host changes IP address, may not be known Internet-wide until all TTLs expire
- Update/notify mechanisms proposed IETF standard
  - RFC 2136

# DNS records

**DNS:** distributed database storing resource records (**RR**)

RR format: `(name, value, type, ttl)`

## type=A

- **name** is hostname
- **value** is IP address

## type=NS

- **name** is domain (e.g.,  
foo.com)
- **value** is hostname of  
authoritative name  
server for this domain

## type=CNAME

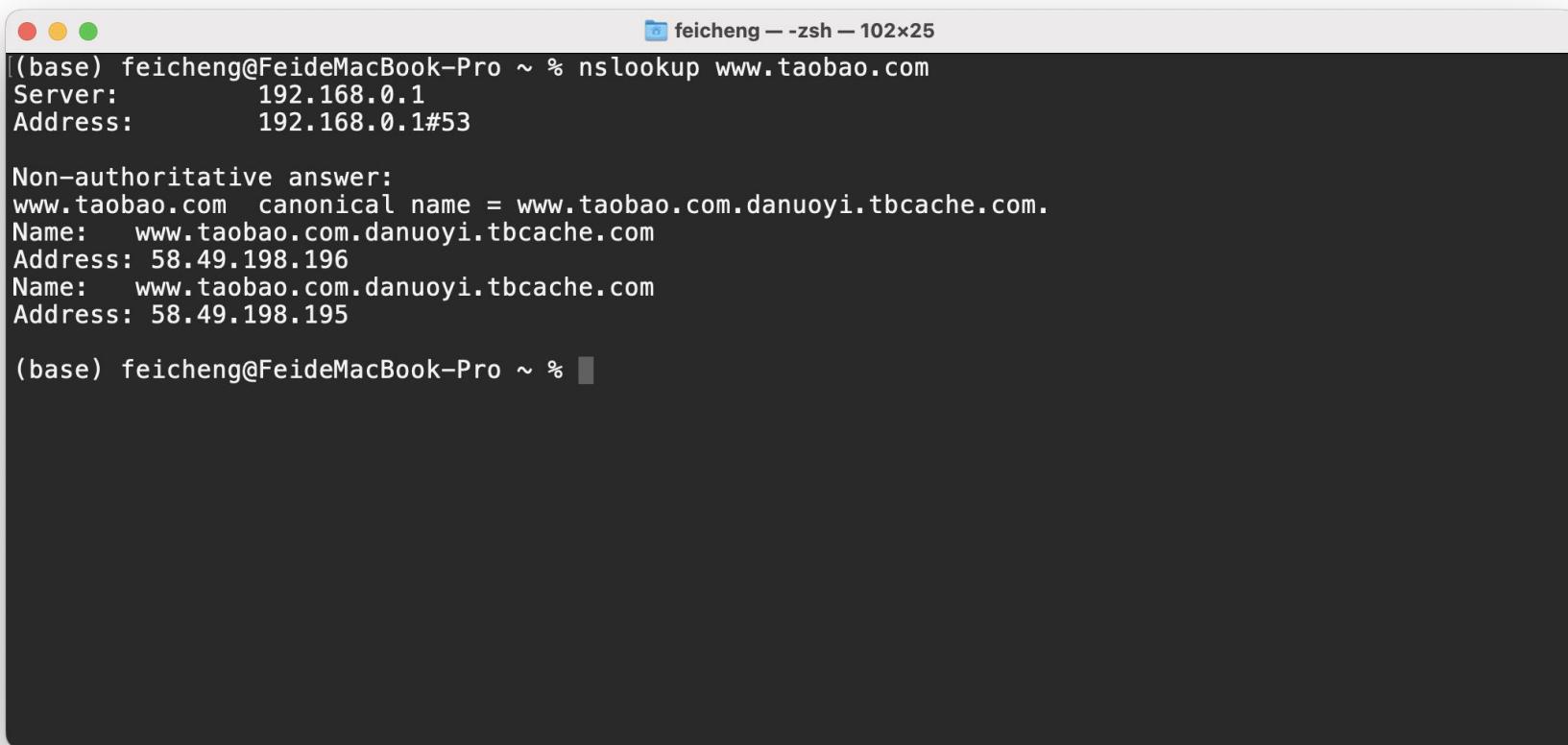
- **name** is alias name for some  
“canonical” (the real) name
- **www.taobao.com** is really  
www.taobao.com.danuoyi.tbcache.com
- **value** is canonical name

## type=MX

- **value** is name of mailserver  
associated with **name**

# Loop up a domain name

- Use *nslookup* command:



```
feicheng -- zsh -- 102x25
(base) feicheng@FeideMacBook-Pro ~ % nslookup www.taobao.com
Server:      192.168.0.1
Address:     192.168.0.1#53

Non-authoritative answer:
www.taobao.com canonical name = www.taobao.com.danuoyi.tbcache.com.
Name: www.taobao.com.danuoyi.tbcache.com
Address: 58.49.198.196
Name: www.taobao.com.danuoyi.tbcache.com
Address: 58.49.198.195

(base) feicheng@FeideMacBook-Pro ~ %
```

A screenshot of a macOS terminal window titled "feicheng -- zsh -- 102x25". The window shows the command "nslookup www.taobao.com" being run. The output indicates that the server is 192.168.0.1 and the address is 192.168.0.1#53. It then provides a non-authoritative answer, showing two entries for the canonical name www.taobao.com.danuoyi.tbcache.com, each with an address of 58.49.198.196 and 58.49.198.195 respectively.

```
feicheng -- zsh -- 105x33
(base) feicheng@FeideMacBook-Pro ~ % nslookup www.xjtlu.edu.cn
Server:      192.168.0.1
Address:     192.168.0.1#53

Non-authoritative answer:
www.xjtlu.edu.cn      canonical name = www.xjtlu.edu.cn.w.cdngslb.com.
Name:   www.xjtlu.edu.cn.w.cdngslb.com
Address: 150.138.252.211
Name:   www.xjtlu.edu.cn.w.cdngslb.com
Address: 221.230.245.106
Name:   www.xjtlu.edu.cn.w.cdngslb.com
Address: 124.72.130.240
Name:   www.xjtlu.edu.cn.w.cdngslb.com
Address: 124.72.130.238
Name:   www.xjtlu.edu.cn.w.cdngslb.com
Address: 124.72.130.239
Name:   www.xjtlu.edu.cn.w.cdngslb.com
Address: 124.72.130.237
Name:   www.xjtlu.edu.cn.w.cdngslb.com
Address: 150.138.252.209
Name:   www.xjtlu.edu.cn.w.cdngslb.com
Address: 150.138.252.216
Name:   www.xjtlu.edu.cn.w.cdngslb.com
Address: 122.225.213.252
Name:   www.xjtlu.edu.cn.w.cdngslb.com
Address: 122.228.95.140
Name:   www.xjtlu.edu.cn.w.cdngslb.com
Address: 122.228.95.141
Name:   www.xjtlu.edu.cn.w.cdngslb.com
Address: 122.225.213.249

(base) feicheng@FeideMacBook-Pro ~ %
```

```
fei -- bash -- 80x12
[FeideMacBook-Pro-2:~ fei$ nslookup www.xjtlu.edu.cn
Server:      10.7.1.2
Address:     10.7.1.2#53

www.xjtlu.edu.cn      canonical name = lnx-wwwweb01.xjtlu.edu.cn.
Name:   lnx-wwwweb01.xjtlu.edu.cn
Address: 10.7.1.32

FeideMacBook-Pro-2:~ fei$
```

When the internet access point changes, the IP address from DNS may change

Try it at home, or at XJTLU

# Get information of a domain name

- Use *whois* command:

- whois feimax.com

```
Domain Name: FEIMAX.COM
Registry Domain ID: 1957417764_DOMAIN_COM-VRSN
Registrar WHOIS Server: grs-whois.hichina.com
Registrar URL: http://www.net.cn
Updated Date: 2019-08-16T04:36:58Z
Creation Date: 2015-09-04T05:56:45Z
Registry Expiry Date: 2020-09-04T05:56:45Z
Registrar: Alibaba Cloud Computing (Beijing) Co., Ltd.
Registrar IANA ID: 420
Registrar Abuse Contact Email: DomainAbuse@service.aliyun.com
Registrar Abuse Contact Phone: +86.95187
Domain Status: ok https://icann.org/epp#ok
Name Server: DNS10.HICHINA.COM
Name Server: DNS9.HICHINA.COM
```



**BAYMAX**



[feimax.com](http://feimax.com)



[feimid.com](http://feimid.com)



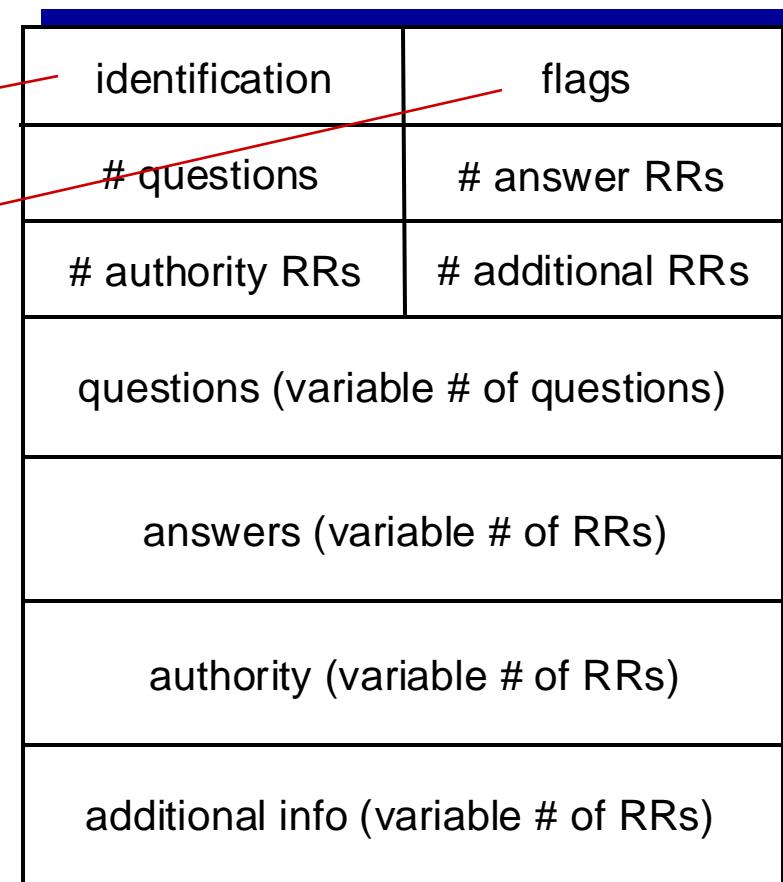
[feimin.me](http://feimin.me)

# DNS protocol, messages

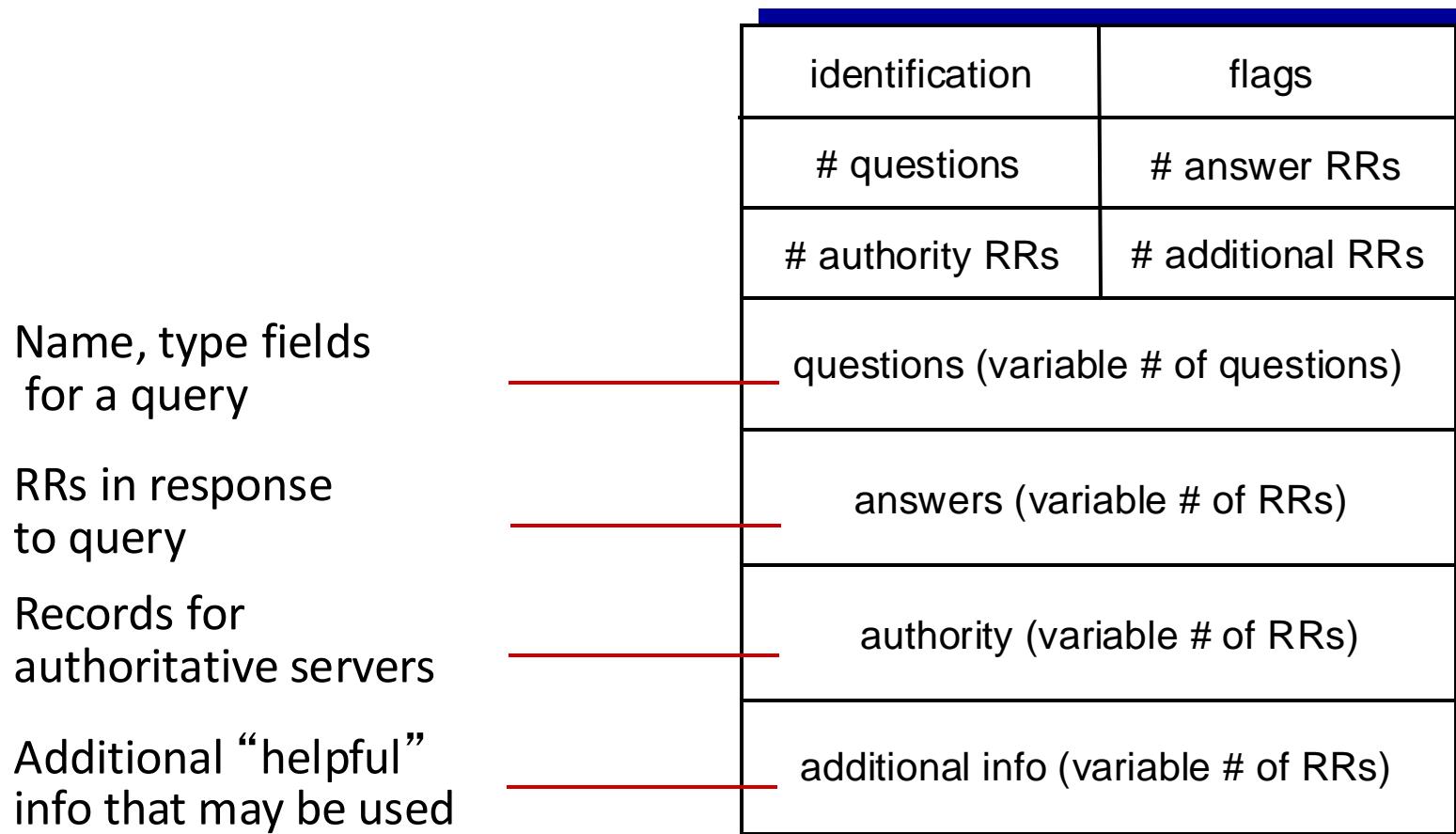
- *Query* and *reply* messages, both with same *message format*

## Message header

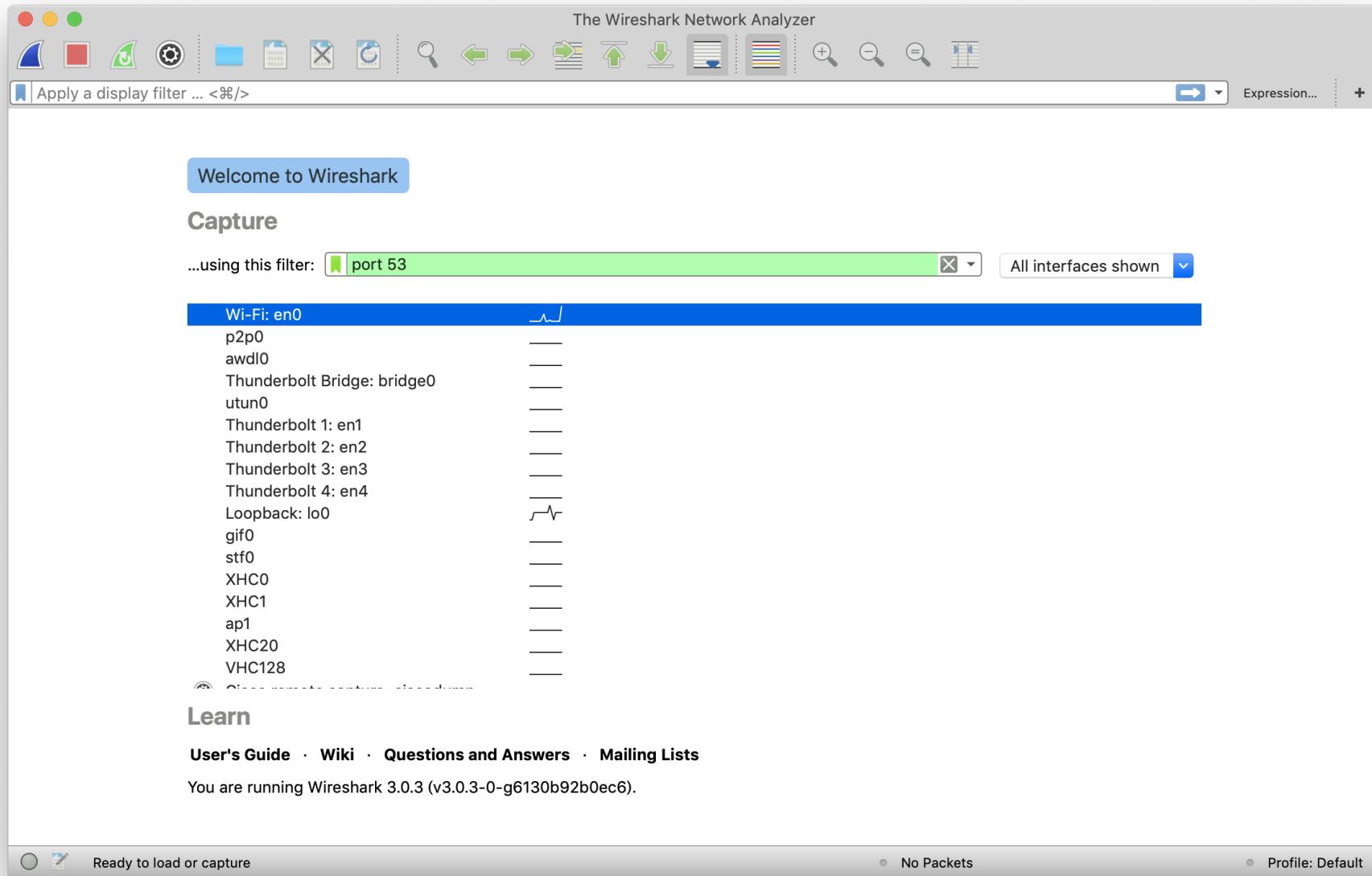
- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



# DNS protocol, messages



# Wireshark examples





# Request

Wireshark · Packet 106 · Wi-Fi: en0 (port 53)

> Frame 106: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface en0, id 0  
> Ethernet II, Src: Apple\_74:26:5d (f4:d4:88:74:26:5d), Dst: Tp-LinkT\_0b:d8:5c (f4:2a:7d:0b:d8:5c)  
> Internet Protocol Version 4, Src: 192.168.0.24, Dst: 192.168.0.1  
> User Datagram Protocol, Src Port: 53876, Dst Port: 53  
  Domain Name System (query)  
    Transaction ID: 0xdb53  
    Flags: 0x0100 Standard query  
      0... .... .... = Response: Message is a query  
      .000 0... .... .... = Opcode: Standard query (0)  
      .... .0. .... .... = Truncated: Message is not truncated  
      .... .1 .... .... = Recursion desired: Do query recursively  
      .... .0. .... .... = Z: reserved (0)  
      .... .... .0 .... = Non-authenticated data: Unacceptable  
    Questions: 1  
    Answer RRs: 0  
    Authority RRs: 0  
    Additional RRs: 0  
  Queries  
    www.feimax.com: type A, class IN  
      Name: www.feimax.com  
      [Name Length: 14]  
      [Label Count: 3]  
      Type: A (Host Address) (1)  
      Class: IN (0x0001)  
      [Response In: 108]  
  
0000 f4 2a 7d 0b d8 5c f4 d4 88 74 26 5d 08 00 45 00 \* } \ . t& ] E  
0010 00 3c 0b ff 00 00 40 11 ed 48 c0 a8 00 18 c0 a8 < . @ H ..  
0020 00 01 d2 74 00 35 00 28 32 a1 db 53 01 00 00 01 .. t 5 ( 2 . S ..  
0030 00 00 00 00 00 00 03 77 77 77 06 66 65 69 6d 61 .. . w ww . feima  
0040 78 03 63 6f 6d 00 00 01 00 01 x . com .. .  
  

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)	

Show packet bytes

Help Close

# Request

Wireshark · Packet 106 · Wi-Fi: en0 (port 53)

> Frame 106: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface en0, id 0  
> Ethernet II, Src: Apple\_74:26:5d (f4:d4:88:74:26:5d), Dst: Tp-LinkT\_0b:d8:5c (f4:2a:7d:0b:d8:5c)  
> Internet Protocol Version 4, Src: 192.168.0.24, Dst: 192.168.0.1  
> User Datagram Protocol, Src Port: 53876, Dst Port: 53  
  Domain Name System (query)  
    Transaction ID: 0xdb53  
    Flags: 0x0100 Standard query  
      0... .... .... = Response: Message is a query  
      .000 0... .... .... = Opcode: Standard query (0)  
      .... .0. .... .... = Truncated: Message is not truncated  
      .... .1 .... .... = Recursion desired: Do query recursively  
      .... .0. .... .... = Z: reserved (0)  
      .... .... 0 .... .... = Non-authenticated data: Unacceptable  
    Questions: 1  
    Answer RRs: 0  
    Authority RRs: 0  
    Additional RRs: 0  
  Queries  
    www.feimax.com: type A, class IN  
      Name: www.feimax.com  
      [Name Length: 14]  
      [Label Count: 3]  
      Type: A (Host Address) (1)  
      Class: IN (0x0001)  
      [Response In: 108]  

0000 f4 2a 7d 0b d8 5c f4 d4 88 74 26 5d 08 00 45 00	.*}..\\... t&].E.
0010 00 3c 0b ff 00 00 40 11 ed 48 c0 a8 00 18 c0 a8	<....@.H.....
0020 00 01 d2 74 00 35 00 28 32 a1 db 53 01 00 00 01	....t.5.( 2 .S....
0030 00 00 00 00 00 00 03 77 77 06 66 65 69 6d 61	.....w ww:feima
0040 78 03 63 6f 6d 00 00 01 00 01	x.com....

Show packet bytes

Help Close

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)	

# Request

Wireshark · Packet 106 · Wi-Fi: en0 (port 53)

> Frame 106: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface en0, id 0  
> Ethernet II, Src: Apple\_74:26:5d (f4:d4:88:74:26:5d), Dst: Tp-LinkT\_0b:d8:5c (f4:2a:7d:0b:d8:5c)  
> Internet Protocol Version 4, Src: 192.168.0.24, Dst: 192.168.0.1  
> User Datagram Protocol, Src Port: 53876, Dst Port: 53  
  Domain Name System (query)  
    Transaction ID: 0xdb53  
    Flags: 0x0100 Standard query  
      0... .... .... = Response: Message is a query  
      .000 0... .... .... = Opcode: Standard query (0)  
      .... .0. .... .... = Truncated: Message is not truncated  
      .... .1 .... .... = Recursion desired: Do query recursively  
      .... .0. .... .... = Z: reserved (0)  
      .... .0 .... .... = Non-authenticated data: Unacceptable  
    Questions: 1  
    Answer RRs: 0  
    Authority RRs: 0  
    Additional RRs: 0  
  Queries  
    www.feimax.com: type A, class IN  
      Name: www.feimax.com  
      [Name Length: 14]  
      [Label Count: 3]  
      Type: A (Host Address) (1)  
      Class: IN (0x0001)  
      [Response In: 108]  

0000 f4 2a 7d 0b d8 5c f4 d4 88 74 26 5d 08 00 45 00	.*}..\\... t&]..E.
0010 00 3c 0b ff 00 00 40 11 ed 48 c0 a8 00 18 c0 a8	<.....@ ..H.....
0020 00 01 d2 74 00 35 00 28 32 a1 db 53 01 00 00 01	....t.5.( 2-S... ..
0030 00 00 00 00 00 00 03 77 77 06 66 65 69 6d 61	.....w ww·feima
0040 78 03 63 6f 6d 00 00 01 00 01	x.com....

Show packet bytes

Help Close

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)	

# Request

Wireshark · Packet 106 · Wi-Fi: en0 (port 53)

> Frame 106: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface en0, id 0  
> Ethernet II, Src: Apple\_74:26:5d (f4:d4:88:74:26:5d), Dst: Tp-LinkT\_0b:d8:5c (f4:2a:7d:0b:d8:5c)  
> Internet Protocol Version 4, Src: 192.168.0.24, Dst: 192.168.0.1  
> User Datagram Protocol, Src Port: 53876, Dst Port: 53  
> Domain Name System (query)  
    Transaction ID: 0xdb53  
    Flags: 0x0100 Standard query  
        0... .... .... = Response: Message is a query  
        .000 0... .... .... = Opcode: Standard query (0)  
        .... .0. .... .... = Truncated: Message is not truncated  
        .... .1 .... .... = Recursion desired: Do query recursively  
        .... .0. .... .... = Z: reserved (0)  
        .... .... .0 .... = Non-authenticated data: Unacceptable  
    Questions: 1  
    Answer RRs: 0  
    Authority RRs: 0  
    Additional RRs: 0  
    Queries  
        www.feimax.com: type A, class IN  
            Name: www.feimax.com  
            [Name Length: 14]  
            [Label Count: 3]  
            Type: A (Host Address) (1)  
            Class: IN (0x0001)  
            [Response In: 108]  

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)	

0000 f4 2a 7d 0b d8 5c f4 d4 88 74 26 5d 08 00 45 00 \* } \ . t& ] E  
0010 00 3c 0b ff 00 00 40 11 ed 48 c0 a8 00 18 c0 a8 < . @ H ..  
0020 00 01 d2 74 00 35 00 28 32 a1 db 53 01 00 00 01 .. t . 5 ( 2 - S ..  
0030 00 00 00 00 00 03 77 77 06 66 65 69 6d 61 .. . w ww . feima  
0040 78 03 63 6f 6d 00 00 01 00 01 x . com .. .

Show packet bytes

Help Close

# Request

Wireshark · Packet 106 · Wi-Fi: en0 (port 53)

> Frame 106: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface en0, id 0  
> Ethernet II, Src: Apple\_74:26:5d (f4:d4:88:74:26:5d), Dst: Tp-LinkT\_0b:d8:5c (f4:2a:7d:0b:d8:5c)  
> Internet Protocol Version 4, Src: 192.168.0.24, Dst: 192.168.0.1  
> User Datagram Protocol, Src Port: 53876, Dst Port: 53  
  Domain Name System (query)  
    Transaction ID: 0xdb53  
    Flags: 0x0100 Standard query  
      0... .... .... = Response: Message is a query  
      .000 0... .... .... = Opcode: Standard query (0)  
      .... .0. .... .... = Truncated: Message is not truncated  
      .... .1 .... .... = Recursion desired: Do query recursively  
      .... .0. .... .... = Z: reserved (0)  
      .... .... .0 .... = Non-authenticated data: Unacceptable  
    Questions: 1  
    Answer RRs: 0  
    Authority RRs: 0  
    Additional RRs: 0  
  Queries  
    www.feimax.com: type A, class IN  
      Name: www.feimax.com  
      [Name Length: 14]  
      [Label Count: 3]  
      Type: A (Host Address) (1)  
      Class: IN (0x0001)  
      [Response In: 108]  
  
0000 f4 2a 7d 0b d8 5c f4 d4 88 74 26 5d 08 00 45 00 \* } \ . t& ] E  
0010 00 3c 0b ff 00 00 40 11 ed 48 c0 a8 00 18 c0 a8 < . @ H ..  
0020 00 01 d2 74 00 35 00 28 32 a1 db 53 01 00 00 01 .. t . 5 ( 2 . S ..  
0030 00 00 00 00 00 03 77 77 06 66 65 69 6d 61 .. w ww . feima  
0040 78 03 63 6f 6d 00 00 01 00 01 x . com .. .  
  

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)	

Show packet bytes

Help Close

# Request

Wireshark · Packet 106 · Wi-Fi: en0 (port 53)

> Frame 106: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface en0, id 0  
> Ethernet II, Src: Apple\_74:26:5d (f4:d4:88:74:26:5d), Dst: Tp-LinkT\_0b:d8:5c (f4:2a:7d:0b:d8:5c)  
> Internet Protocol Version 4, Src: 192.168.0.24, Dst: 192.168.0.1  
> User Datagram Protocol, Src Port: 53876, Dst Port: 53  
  Domain Name System (query)  
    Transaction ID: 0xdb53  
    Flags: 0x0100 Standard query  
      0... .... .... = Response: Message is a query  
      .000 0.... .... = Opcode: Standard query (0)  
      .... .0. .... = Truncated: Message is not truncated  
      .... .1 .... = Recursion desired: Do query recursively  
      .... .0. .... = Z: reserved (0)  
      .... ....0 .... = Non-authenticated data: Unacceptable  
    Questions: 1  
    Answer RRs: 0  
    Authority RRs: 0  
    Additional RRs: 0  
    Queries  
      www.feimax.com: type A, class IN  
        Name: www.feimax.com  
        [Name Length: 14]  
        [Label Count: 3]  
        Type: A (Host Address) (1)  
        Class: IN (0x0001)  
        [Response In: 108]  

0000	f4	2a	7d	0b	d8	5c	f4	d4	88	74	26	5d	08	00	45	00	* } \ . . . t& ] . E .
0010	00	3c	0b	ff	00	00	40	11	ed	48	c0	a8	00	18	c0	a8	< . . . @ . H . . . .
0020	00	01	d2	74	00	35	00	28	32	a1	db	53	01	00	00	01	. . . t . 5 . ( 2 . S . . . .
0030	00	00	00	00	00	00	03	77	77	77	06	66	65	69	6d	61	. . . . w ww . feima
0040	78	03	63	6f	6d	00	00	01	00	01	x . com . . . .						

Show packet bytes

Help Close

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)	

# Request

Wireshark · Packet 106 · Wi-Fi: en0 (port 53)

```

> Frame 106: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface en0, id 0
> Ethernet II, Src: Apple_74:26:5d (f4:d4:88:74:26:5d), Dst: Tp-LinkT_0b:d8:5c (f4:2a:7d:0b:d8:5c)
> Internet Protocol Version 4, Src: 192.168.0.24, Dst: 192.168.0.1
> User Datagram Protocol, Src Port: 53876, Dst Port: 53
  Domain Name System (query)
    Transaction ID: 0xdb53
    Flags: 0x0100 Standard query
      0... .... .... = Response: Message is a query
      .000 0.... .... = Opcode: Standard query (0)
      .... 0. .... .... = Truncated: Message is not truncated
      .... 1.... .... = Recursion desired: Do query recursively
      .... 0.... .... = Z: reserved (0)
      .... 0.... .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
    Queries
      www.feimax.com: type A, class IN
        Name: www.feimax.com
          [Name Length: 14]
          [Label Count: 3]
          Type: A (Host Address) (1)
          Class: IN (0x0001)
        [Response In: 108]
        0000 f4 2a 7d 0b d8 5c f4 d4 88 74 26 5d 08 00 45 00
        0010 00 3c 0b ff 00 00 40 11 ed 48 c0 a8 00 18 c0 a8
        0020 00 01 d2 74 00 35 00 28 32 a1 db 53 01 00 00 01
        0030 00 00 00 00 00 03 77 77 06 66 65 69 6d 61
        0040 78 03 63 6f 6d 00 00 01 00 01
        ...} \... t&] E
        <....@ H....
        ...t 5( 2-S...
        ....w ww.feima
        x.com...
  
```

Show packet bytes

Help Close

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)	

# Request

Wireshark · Packet 106 · Wi-Fi: en0 (port 53)

> Frame 106: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface en0, id 0  
> Ethernet II, Src: Apple\_74:26:5d (f4:d4:88:74:26:5d), Dst: Tp-LinkT\_0b:d8:5c (f4:2a:7d:0b:d8:5c)  
> Internet Protocol Version 4, Src: 192.168.0.24, Dst: 192.168.0.1  
> User Datagram Protocol, Src Port: 53876, Dst Port: 53  
  Domain Name System (query)  
    Transaction ID: 0xdb53  
    Flags: 0x0100 Standard query  
      0... .... .... = Response: Message is a query  
      .000 0.... .... = Opcode: Standard query (0)  
      .... .0. .... = Truncated: Message is not truncated  
      .... .1 .... = Recursion desired: Do query recursively  
      .... .0. .... = Z: reserved (0)  
      .... ....0 .... = Non-authenticated data: Unacceptable  
    Questions: 1  
    Answer RRs: 0  
    Authority RRs: 0  
    Additional RRs: 0  
  Queries  
    www.feimax.com: type A, class IN  
      Name: www.feimax.com  
      [Name Length: 14]  
      [Label Count: 3]  
      Type: A (Host Address) (1)  
      Class: IN (0x0001)  
      [Response In: 108]  

0000 f4 2a 7d 0b d8 5c f4 d4 88 74 26 5d 08 00 45 00	.*}..\\... t&]..E.
0010 00 3c 0b ff 00 00 40 11 ed 48 c0 a8 00 18 c0 a8	<.....@ ..H.....
0020 00 01 d2 74 00 35 00 28 32 a1 db 53 01 00 00 01	....t.5( 2-S.....
0030 00 00 00 00 00 00 03 77 77 06 66 65 69 6d 61	.....w ww·feima
0040 78 03 63 6f 6d 00 00 01 00 01	x.com... . .

Show packet bytes

Help Close

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)	

# Request

Wireshark · Packet 106 · Wi-Fi: en0 (port 53)

> Frame 106: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface en0, id 0  
> Ethernet II, Src: Apple\_74:26:5d (f4:d4:88:74:26:5d), Dst: Tp-LinkT\_0b:d8:5c (f4:2a:7d:0b:d8:5c)  
> Internet Protocol Version 4, Src: 192.168.0.24, Dst: 192.168.0.1  
> User Datagram Protocol, Src Port: 53876, Dst Port: 53  
  Domain Name System (query)  
    Transaction ID: 0xdb53  
    Flags: 0x0100 Standard query  
      0... .... .... = Response: Message is a query  
      .000 0... .... .... = Opcode: Standard query (0)  
      .... .0. .... .... = Truncated: Message is not truncated  
      .... .1 .... .... = Recursion desired: Do query recursively  
      .... .0. .... .... = Z: reserved (0)  
      .... .... .0 .... = Non-authenticated data: Unacceptable  
    Questions: 1  
    Answer RRs: 0  
    Authority RRs: 0  
    Additional RRs: 0  
  Queries  
    www.feimax.com: type A, class IN  
      Name: www.feimax.com  
      [Name Length: 14]  
      [Label Count: 3]  
      Type: A (Host Address) (1)  
      Class: IN (0x0001)  
      [Response In: 108]  

0000	f4	2a	7d	0b	d8	5c	f4	d4	88	74	26	5d	08	00	45	00	* } \ . . . t& ] . E .
0010	00	3c	0b	ff	00	00	40	11	ed	48	c0	a8	00	18	c0	a8	< . . . @ . H . . . .
0020	00	01	d2	74	00	35	00	28	32	a1	db	53	01	00	00	01	. . . t . 5 . ( 2 . S . . . .
0030	00	00	00	00	00	00	03	77	77	06	66	65	69	6d	61	..... w ww . feima	
0040	78	03	63	6f	6d	00	00	01	00	01						x . com . . . . . . . .	

Show packet bytes

Help Close

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)

# Response

Wireshark · Packet 108 · Wi-Fi: en0 (port 53)

transaction ID: 0xaab53

Flags: 0x8180 Standard query response, No error

1.... .... .... = Response: Message is a response  
.000 0.... .... = Opcode: Standard query (0)  
.... .0.... .... = Authoritative: Server is not an authority for domain  
.... .0.... .... = Truncated: Message is not truncated  
.... ..1 .... .... = Recursion desired: Do query recursively  
.... ...1.... .... = Recursion available: Server can do recursive queries  
.... ...0.... .... = Z: reserved (0)  
.... ...0.... .... = Answer authenticated: Answer/authority portion was not authenticated by  
.... ...0.... .... = Non-authenticated data: Unacceptable  
.... .... .... 0000 = Reply code: No error (0)

Questions: 1

Answer RRs: 1

Authority RRs: 0

Additional RRs: 0

Queries

www.feimax.com: type A, class IN

Name: www.feimax.com  
[Name Length: 14]  
[Label Count: 3]  
Type: A (Host Address) (1)  
Class: IN (0x0001)

Answers

www.feimax.com: type A, class IN, addr 106.14.20.246  
[Request In: 106]  
[Time: 0.031764000 seconds]

0000	f4 d4 88 74 26 5d f4 2a	7d 0b d8 5c 08 00 45 00	...t&]* }...\..E.
0010	00 4c 00 00 40 00 40 11	b9 37 c0 a8 00 01 c0 a8	.L..@.. 7.....
0020	00 18 00 35 d2 74 00 38	72 ac db 53 81 80 00 01	..5-t-8 r-S.....
0030	00 01 00 00 00 00 03 77	77 77 06 66 65 69 6d 61	....w ww.feima
0040	78 03 63 6f 6d 00 00 01	00 01 c0 0c 00 01 00 01	x.com.....
0050	00 00 00 3c 00 04 6a 0e	14 f6	...<..j.. ..

Show packet bytes

Help Close

# Inserting records into DNS

- Example: new startup “feimax.com”
- Register name feimax.com at DNS registrar (e.g., net.cn)
  - Normally, you don’t need to set up the NS record
  - Insert A record for the IP address of your host
  - Insert MX record for email
  - ...

Name	Type	RR	TTL
<input type="checkbox"/> 主机记录 ②	记录类型 ②	解析请求来源(isp) ②	记录值 ②
<input type="checkbox"/> group	CNAME	默认	plugin.vsais.com 10 分钟
<input type="checkbox"/> nas	显性URL	默认	https://feimax.cn6.quickconnect.cn/ 10 分钟
<input type="checkbox"/> www	A	默认	106.14.20.246 10 分钟
<input type="checkbox"/> @	显性URL	默认	http://www.feimax.com 10 分钟
<input type="checkbox"/> @	TXT	默认	v=spf1 include:spf.mxhichina.com -all 10 分钟
<input type="checkbox"/> mail	CNAME	默认	mail.mxhichina.com 10 分钟
<input type="checkbox"/> @	MX	默认	mxw.mxhichina.com.   10 10 分钟
<input type="checkbox"/> @	MX	默认	mxn.mxhichina.com   5 10 分钟

Domain name is a scarce resource.

# Lecture III – Application Layer (2)

1. Domain Name System (DNS)
2. P2P Applications
3. Socket Programming

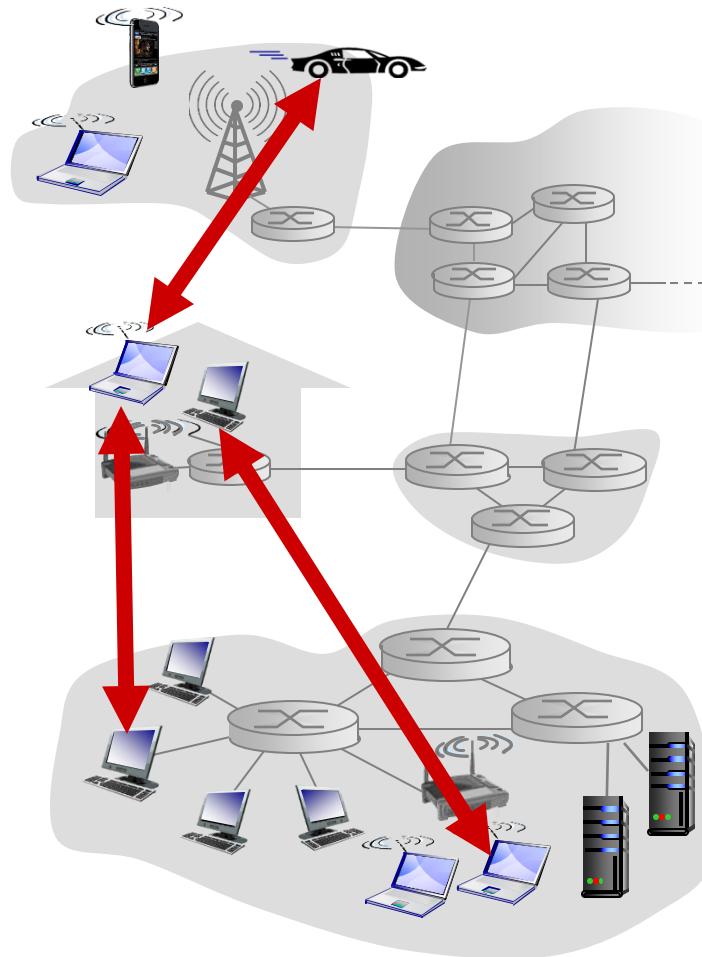


# Pure P2P architecture

- **No always-on server**
- **Arbitrary end systems directly communicate**
- **Peers change IP addresses**

## *Examples:*

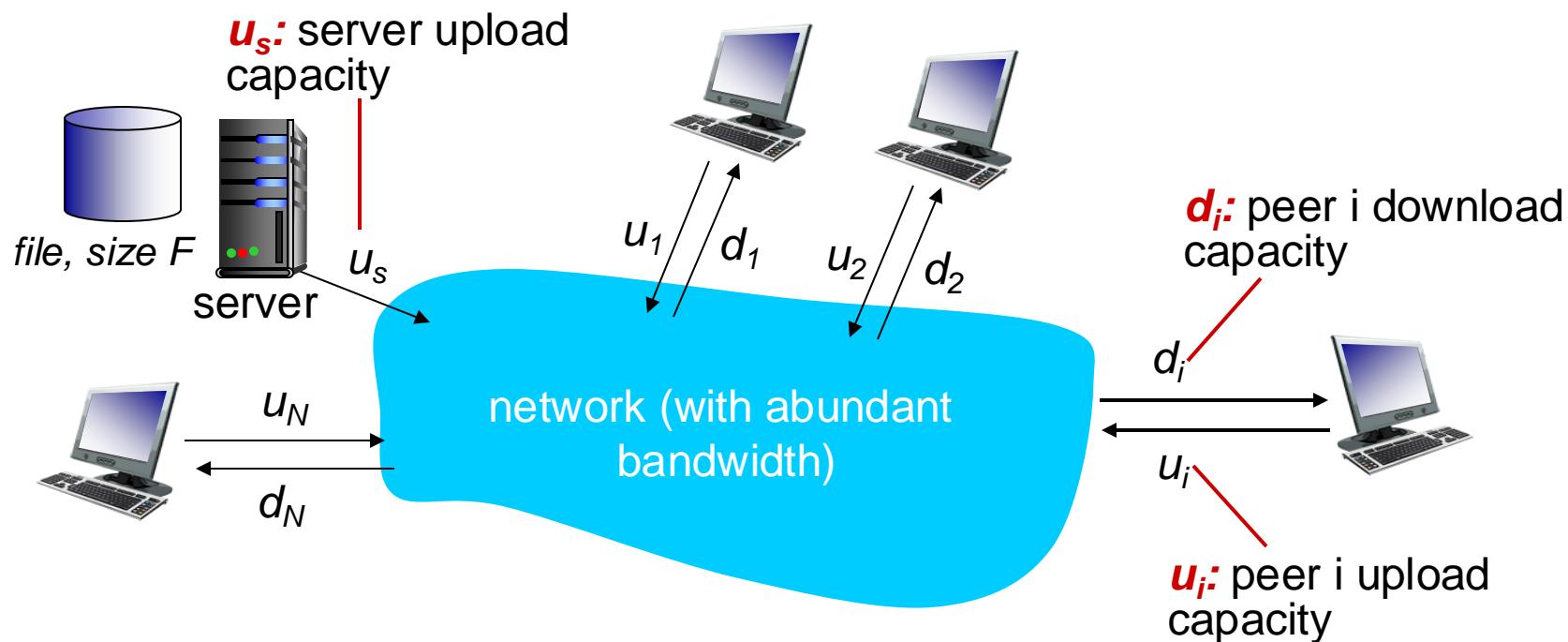
- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)



# File distribution: client-server vs P2P

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

- peer upload/download capacity is limited resource



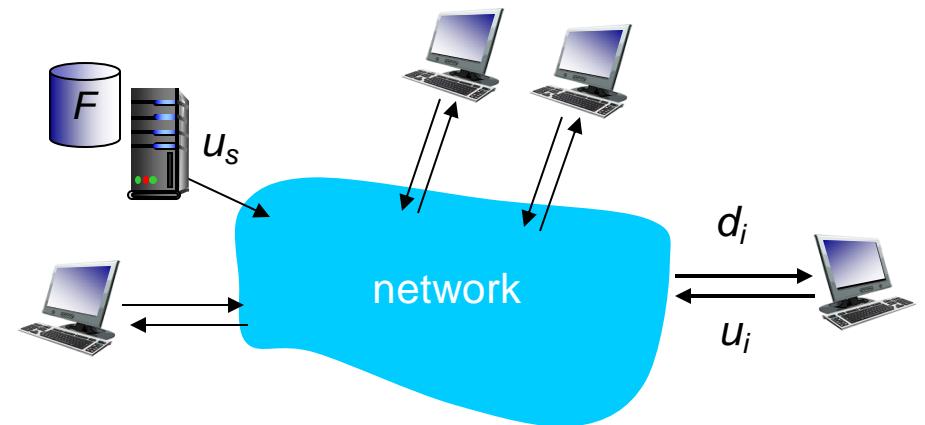
# File distribution time: client-server

- **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
- **max client download time:  $F/d_{min}$**



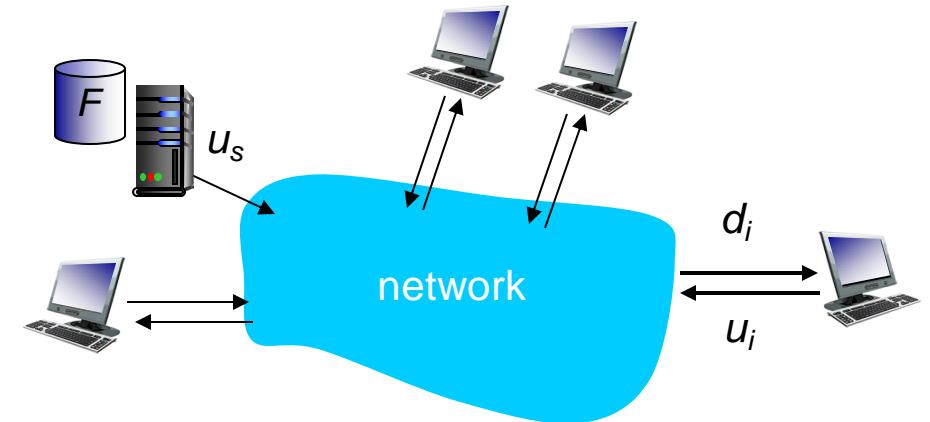
*time to distribute  $F$   
to  $N$  clients using  
client-server approach*

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

increases linearly in  $N$

# File distribution time: P2P

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



time to distribute  $F$   
to  $N$  clients using  
P2P approach

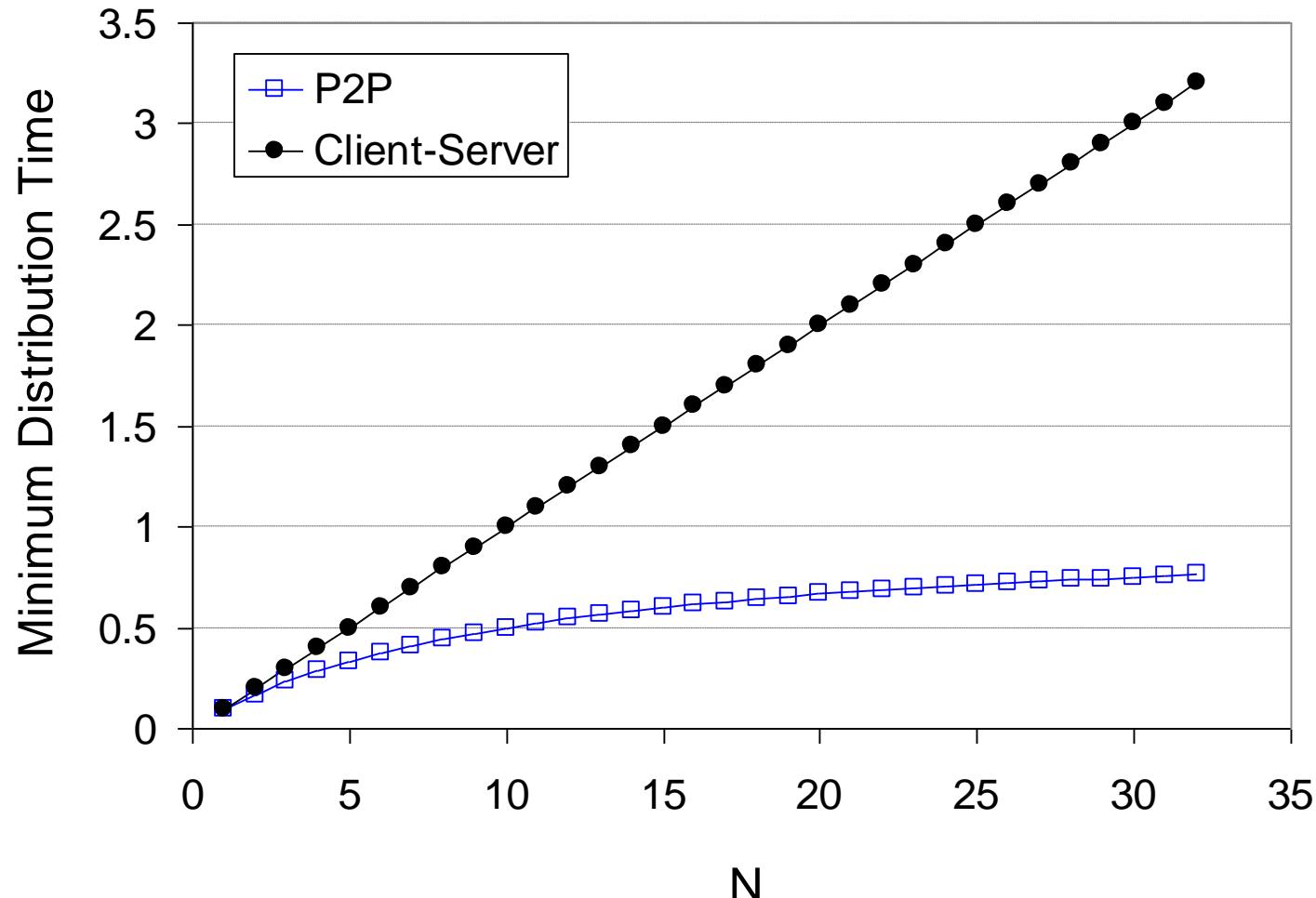
$$D_{P2P} > \max\{F/u_s, F/d_{min}, NF/(u_s + \sum u_i)\}$$

increases linearly in  $N$  ...

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

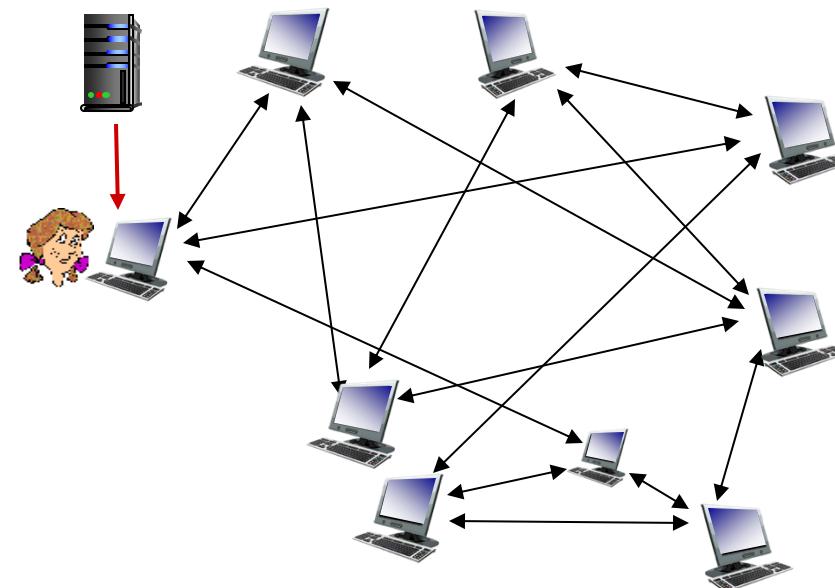
# Client-server vs. P2P: example

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



# BitTorrent

- Efficient content distribution system using file swarming.
- The throughput increases with the number of downloaders via the efficient use of network bandwidth.

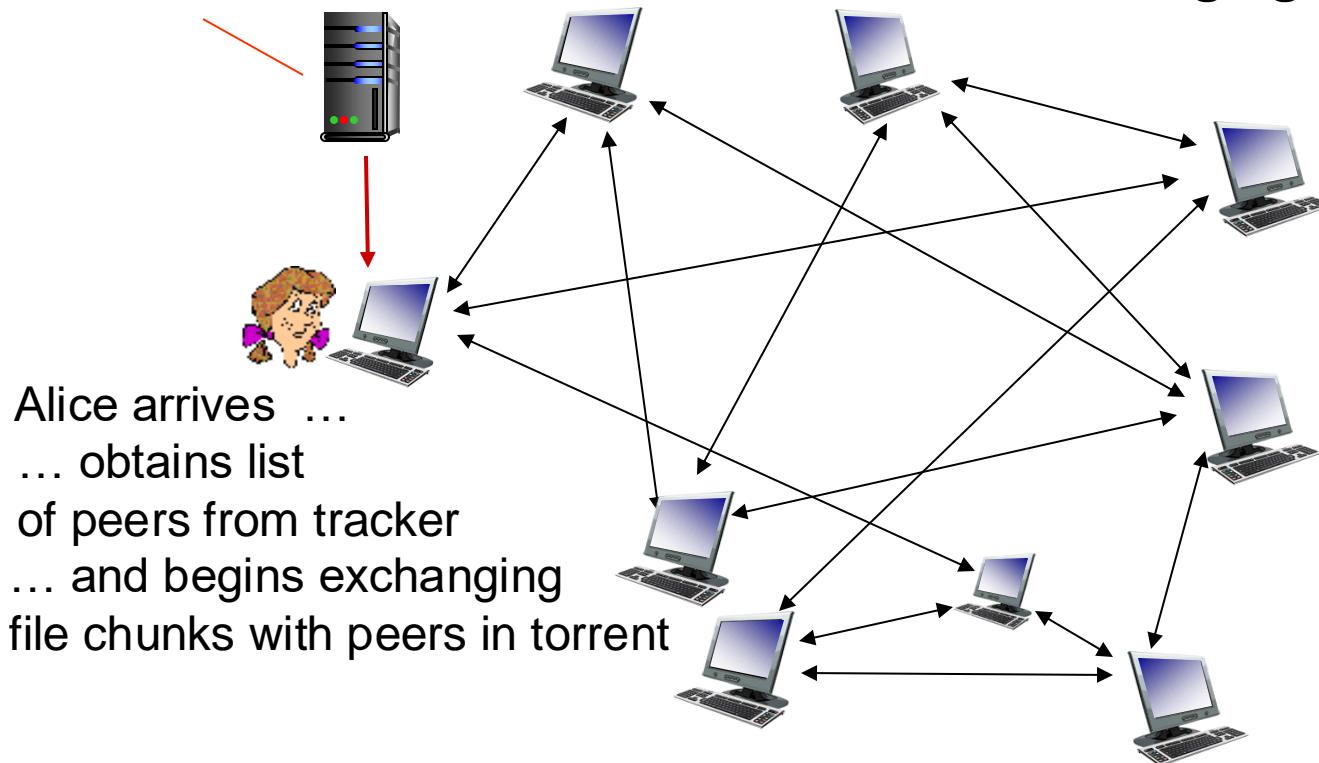


# BitTorrent

- Peers in torrent send/receive file pieces(chunks)

*Tracker*: a central server keeping a list of all peers  
tracks peers participating in torrent

*Torrent/Swarm*: group of peers  
exchanging chunks of a file



# BitTorrent

- To share a file or group of files, the initiator first creates a **.torrent** file, a small file that contains:
  - **Metadata** about the files to be shared
  - Information about the **tracker**, the computer that coordinates the file distribution
- Downloaders first obtain a **.torrent** file, and then connect to the specified **tracker**, which tells them from which other peers to download the **pieces** of the file.

# Metadata of .torrent

- SHA-1 hashes of all pieces
- A mapping of the pieces to files
- Piece size
- Length of the file
- A tracker reference

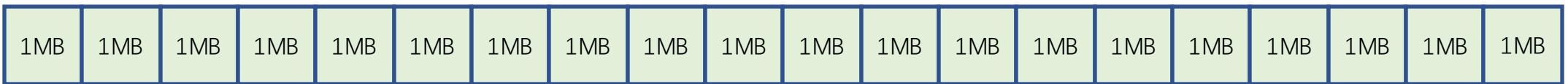
# Pieces of the file

A 20MB File

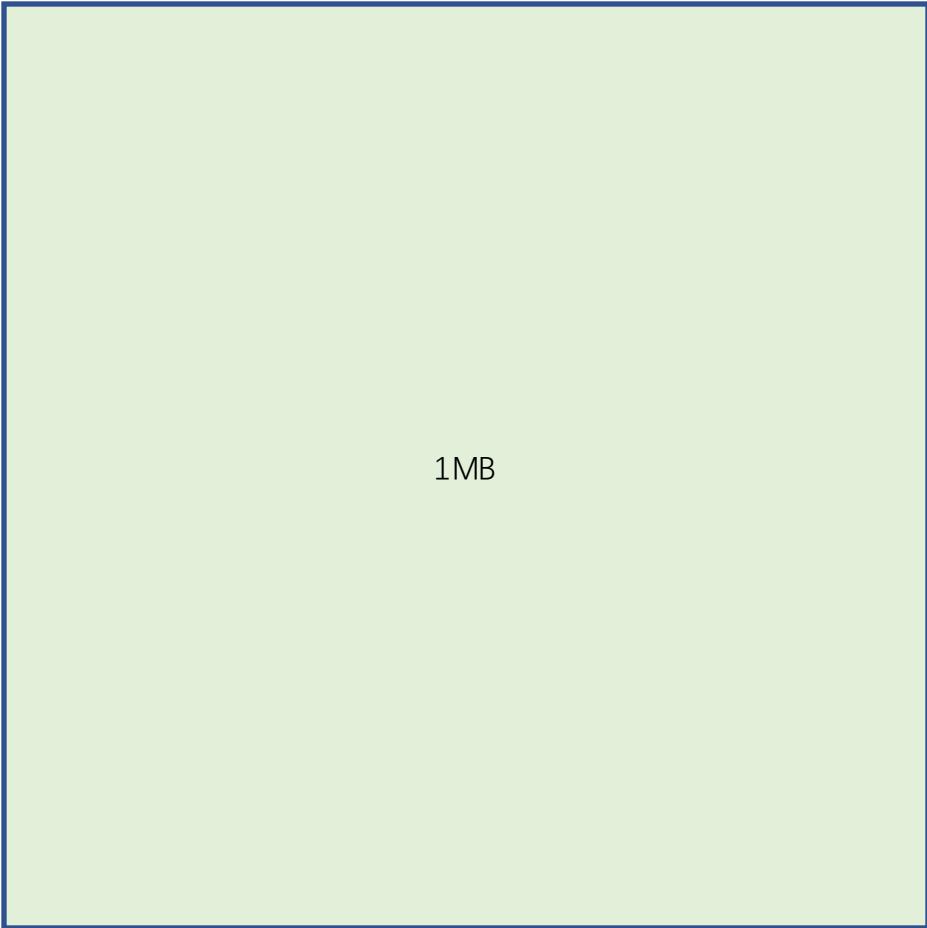


# Pieces of the file

A 20MB File



# Pieces of the file



1MB

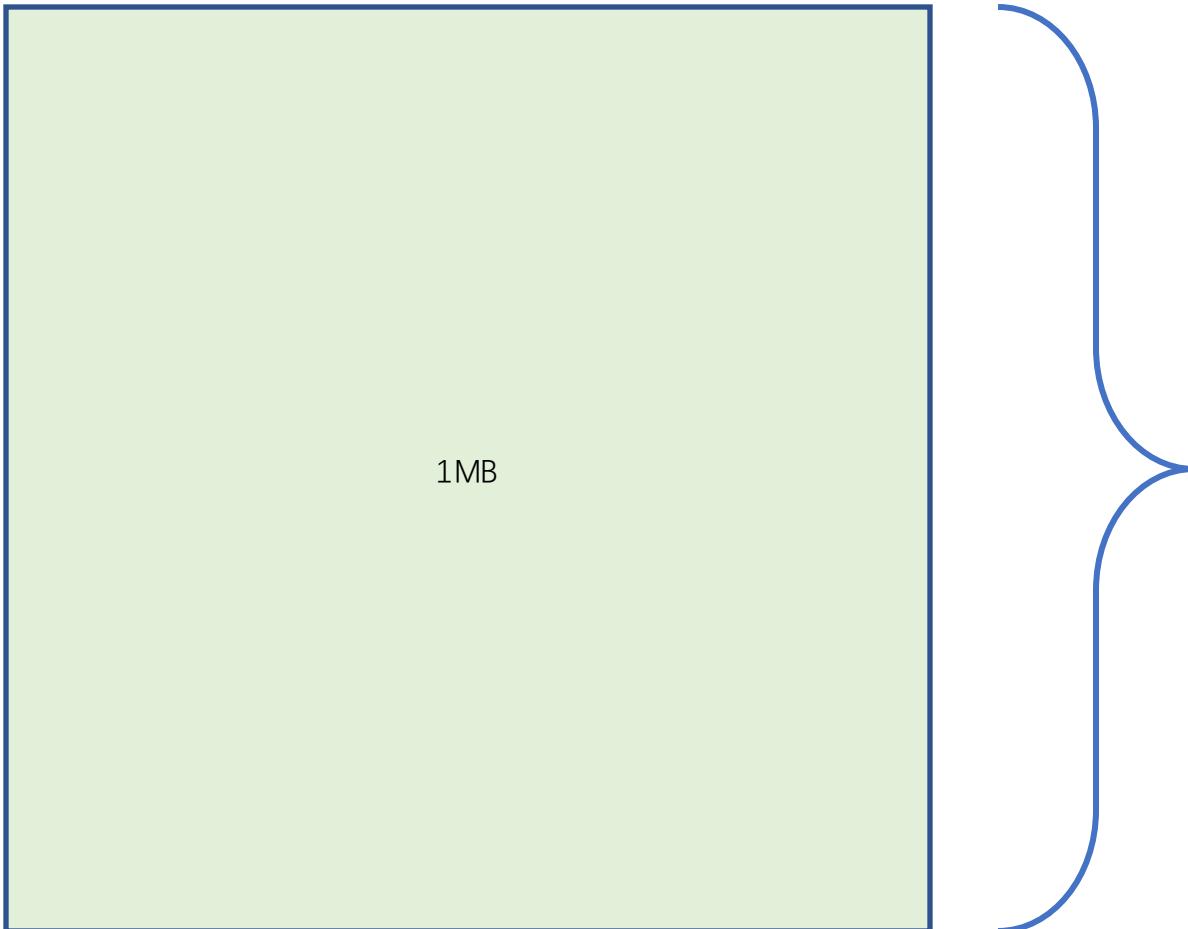
# Pieces of the file

```
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
000010100101010010101001111001001010  
.....
```



a64fdc88f6adfc3755cc8395f2f1a3bfe2fe203e

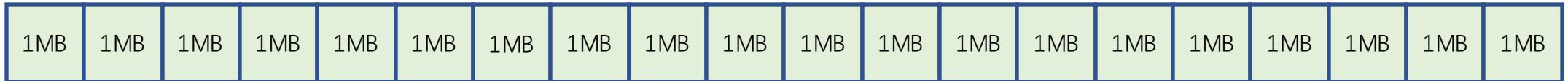
# Pieces of the file



a64fdc88f6adfc3755cc8395f2f1a3bfe2fe203e

# Pieces of the file – SHA1 List

A 20MB File

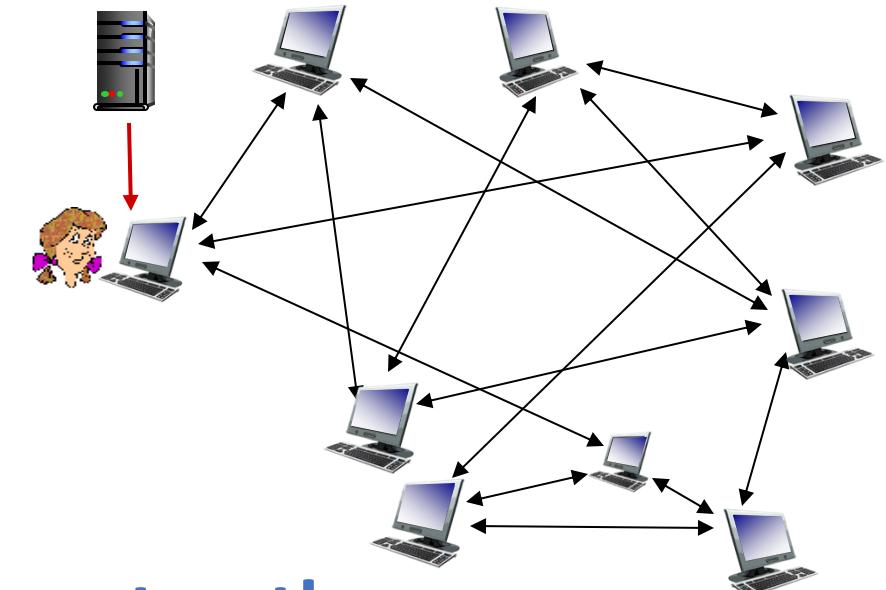


d01b701843e3fe7866aaa33458504d4d72c2d2b7  
e6d9e8325c4dd58f75924b6b0a4d914df4dc2d2d  
a0c60e0ac1adec5908b923db2d91c94e56802c62  
4bb19bd734729d831d16e8a756fc85919712f779  
51f8bc061e66c297b9655902b4a432ae1d68bdf6  
fdfdcfef8be54a1a0aa2492d99f038d29309c71c  
8bfb3ea7fd6f0f28bcfac1bb978342d3897dcc9  
f00c928c5084b42dc9256ea6a708c8f2e580957d  
3be062723dd90986594e542fab731750f40f1ae2  
71e1dc348aaeb13f65b6c2e46d6ccd358a5000d4

.....

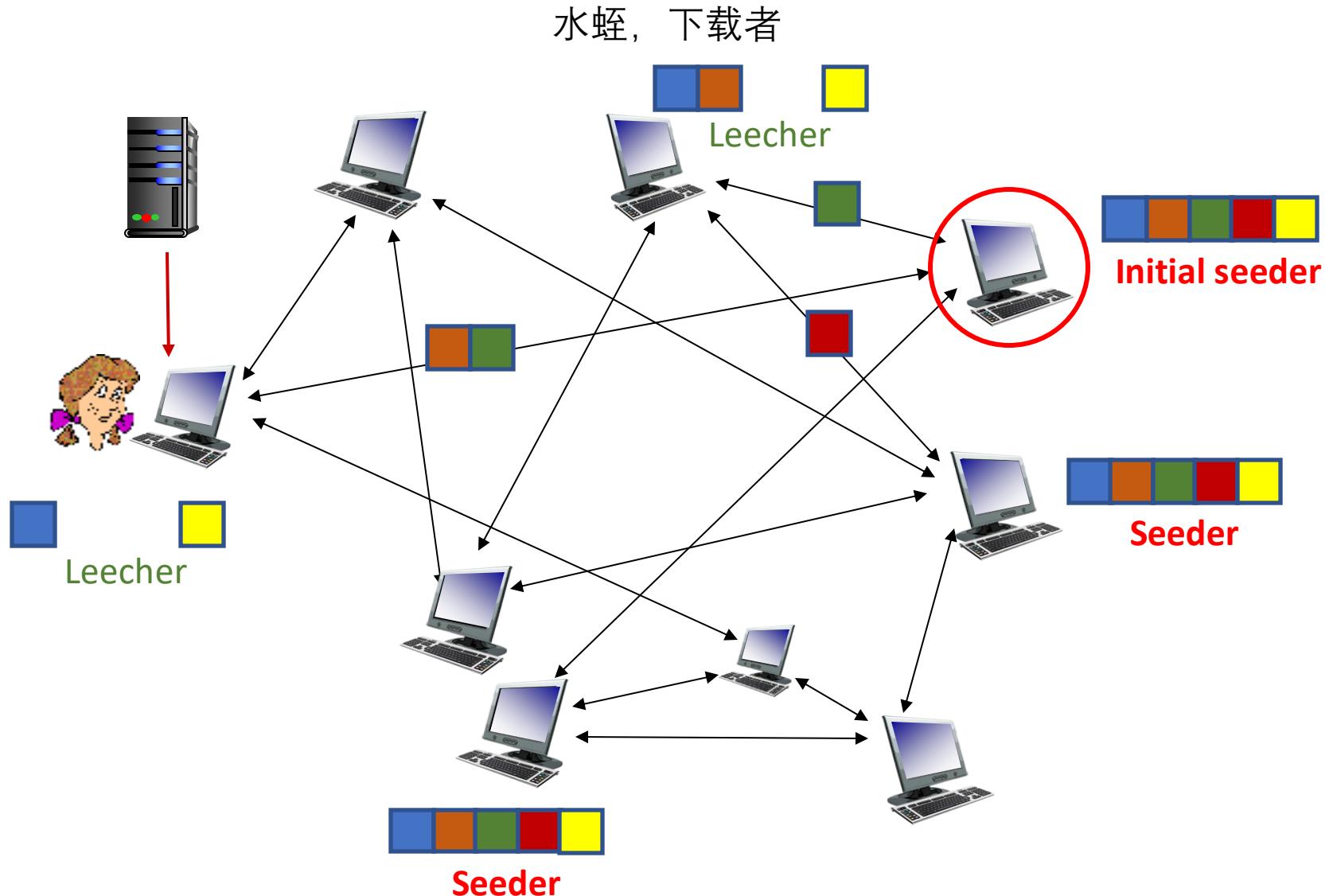
# BitTorrent

- Peer joining torrent:
  - has no pieces, 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 pieces to other peers
- Peer may change peers with whom it exchanges pieces
- Peers may come and go
- Once peer has entire file, it may (selfishly) leave or remain in torrent



**Seeder** = a peer that provides the complete file.

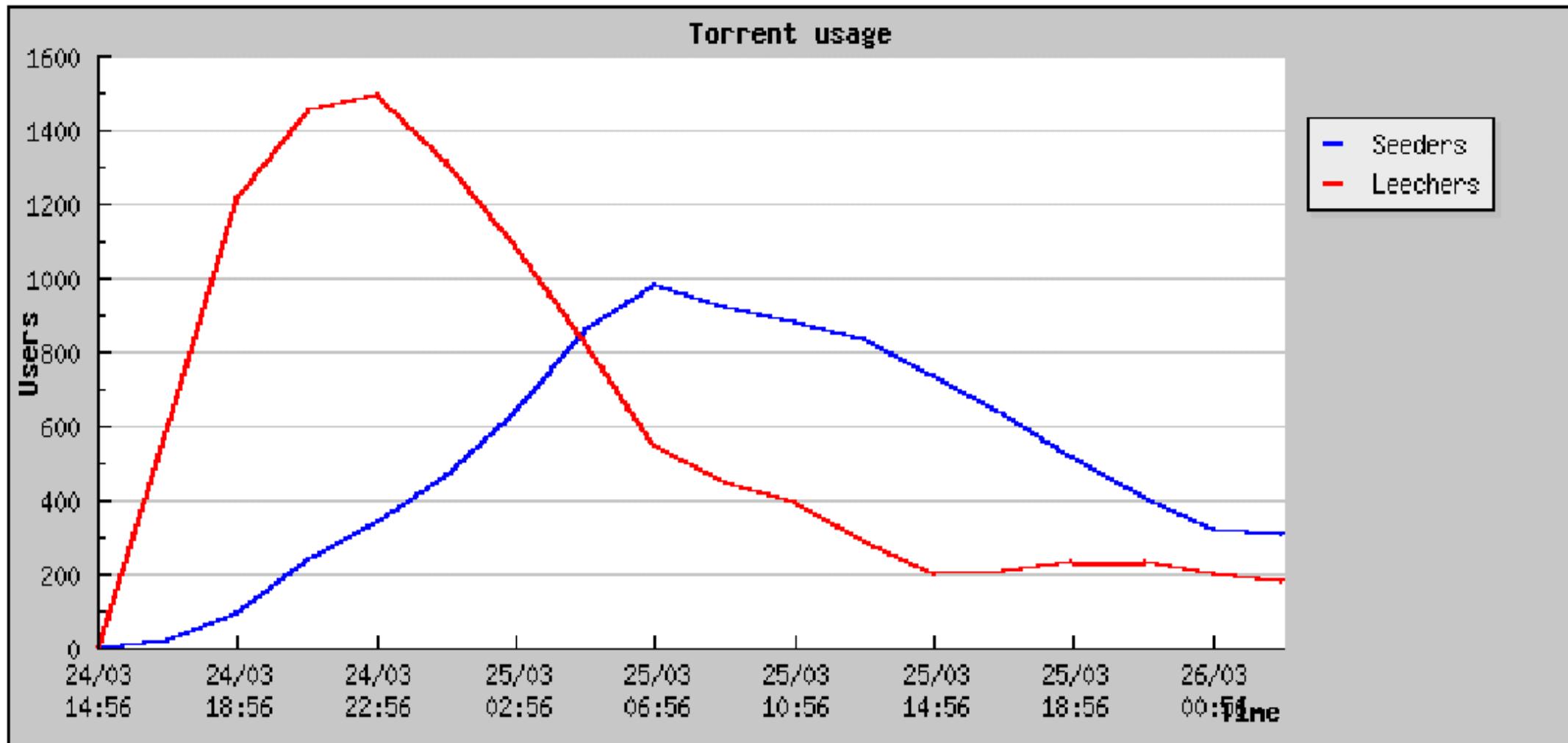
**Initial seeder** = a peer that provides the initial copy.



# Leecher -> Seeder

- As soon as a leecher has a complete piece, it can potentially share it with other downloaders.
- Eventually each leecher becomes a seeder by obtaining all the pieces, and assembles the file. Verifies the “checksum” of the file.

# Leecher -> Seeder



# Piece selection policy

- The **order** in which pieces are selected by different peers is critical for good performance.
- If an inefficient policy is used, then peers may end up in a situation where each has all identical set of easily available pieces, and **none of the missing ones**.
- If the original seed is prematurely taken down, then the file cannot be completely downloaded!

# Piece selection - Macro view



# Piece selection - Micro view

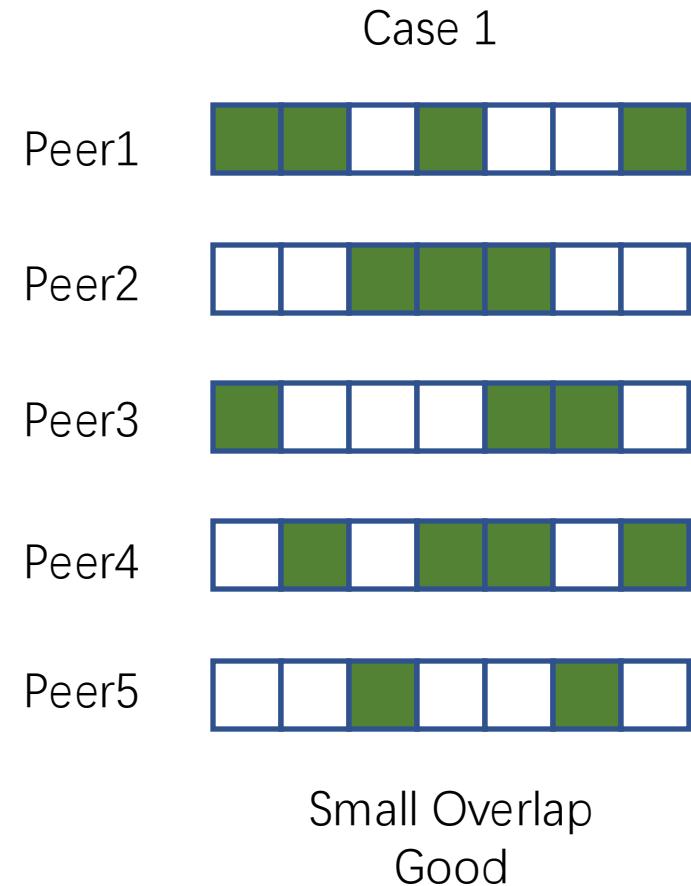
- **Rarest First**
  - General rule
- **Random First Piece**
  - Special case, at the beginning
- **Endgame Mode**
  - Special case

# Random First Piece

- Initially, a peer has nothing to trade
- Important to get a complete piece ASAP
- Select a random piece of the file and download it

# Rarest First

- Determine the pieces that are **most rare among your peers**, and download those first.
- This ensures that the most commonly available pieces are left till the end to download.



# Endgame Mode

- Near the end, missing pieces are requested from every peer containing them.
- This ensures that a download is not prevented from completion due to a single peer with a slow transfer rate.
- Some bandwidth is wasted, but in practice, this is not too much.

# BitTorrent – Internal Mechanism

- Built-in incentive mechanism (where all the magic happens):
  - Choking Algorithm
  - Optimistic Unchoking

# Choking

- Choking is a *temporary refusal* to upload. It is one of BT's most powerful idea to deal with free riders (those who only download but never upload).
  - For avoiding free riders and avoiding network congestion
- *Tit-for-tat strategy* is based on game-theoretic concepts.  
以牙还牙

# Optimistic unchoking

- A peer sends pieces 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 every 10 secs
- Every 30 secs: randomly select another peer, starts sending chunks
  - “optimistically unchoke” this peer
  - newly chosen peer may join top 4
- Reasons:
  - To discover currently unused connections that are better than the ones being used
  - To provide minimal service to new peers

# Upload-Only mode

- Once download is complete, a peer can **only upload**. The question is, which nodes to upload to?
- Policy: Upload to those with the best upload rate. This ensures that pieces get replicated faster, and new seeders are created fast

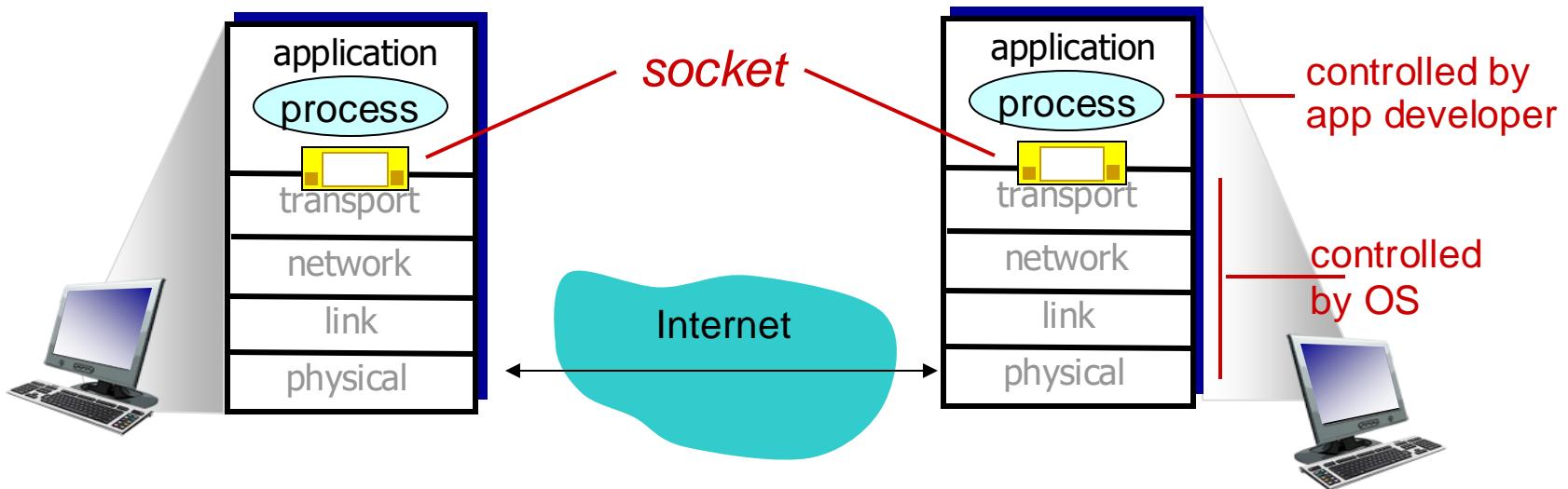
# Lecture III – Application Layer (2)

1. Domain Name System (DNS)
2. P2P Applications
3. Socket Programming



# Socket programming

- **Goal:** learn how to build client/server applications that communicate using sockets
- **Socket:** door between application process and end-end-transport protocol



# Socket programming

- Two socket types for two transport services:
  - UDP: unreliable datagram
  - TCP: reliable, byte stream-oriented
- Application Example:
  1. client reads a line of characters (data) from its keyboard and sends data to server
  2. server receives the data and converts characters to uppercase
  3. server sends modified data to client
  4. client receives modified data and displays line on its screen

# Socket programming with UDP

- **UDP: no “connection” between client & server**
  - No handshaking before sending data
  - Sender explicitly attaches IP destination address and port # to each packet
  - Receiver extracts sender IP address and port# from received packet
- **UDP: transmitted data may be lost or received out-of-order**
- **Application viewpoint:**
  - UDP provides unreliable transfer of groups of bytes (“datagrams”) between client and server

# Client/server socket interaction: UDP

## server (running on serverIP)

create socket, port= x:

```
serverSocket =  
socket(AF_INET,SOCK_DGRAM)
```

read datagram from  
**serverSocket**

write reply to  
**serverSocket**  
specifying  
client address,  
port number

## client

create socket:

```
clientSocket =  
socket(AF_INET,SOCK_DGRAM)
```

Create datagram with server IP and  
port=x; send datagram via  
**clientSocket**

read datagram from  
**clientSocket**  
close  
**clientSocket**

# Example app: UDP server

include Python's socket library

create UDP socket  
bind socket to local port number 12000

loop forever

Read from UDP socket into message, getting client's address (client IP and port)

send upper case string back to this client

## *Python UDPServer*

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print ('The server is ready to receive')
while True:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(),
                        clientAddress)
```

# Example app: UDP client

## *Python UDPCClient*

```
include Python's socket  
library → from socket import *  
  
create UDP socket for  
server → serverName = 'hostname'  
get user keyboard  
input → serverPort = 12000  
Attach server name, port to  
message; send into socket → clientSocket = socket(AF_INET, SOCK_DGRAM)  
  
print out received string  
and close socket → message = input('Input lowercase sentence: ')  
→ clientSocket.sendto(message.encode(),  
                      (serverName, serverPort))  
→ modifiedMessage, serverAddress = clientSocket.recvfrom(2048)  
→ print(modifiedMessage.decode())  
→ clientSocket.close()
```

# Socket programming with TCP

## Client must contact server

- Server process must first be running
- Server must have created socket (door) that welcomes client's contact

## client contacts server by:

- Creating TCP socket, specifying IP address, port number of server process
- ***When client creates socket:*** client TCP establishes connection to server TCP

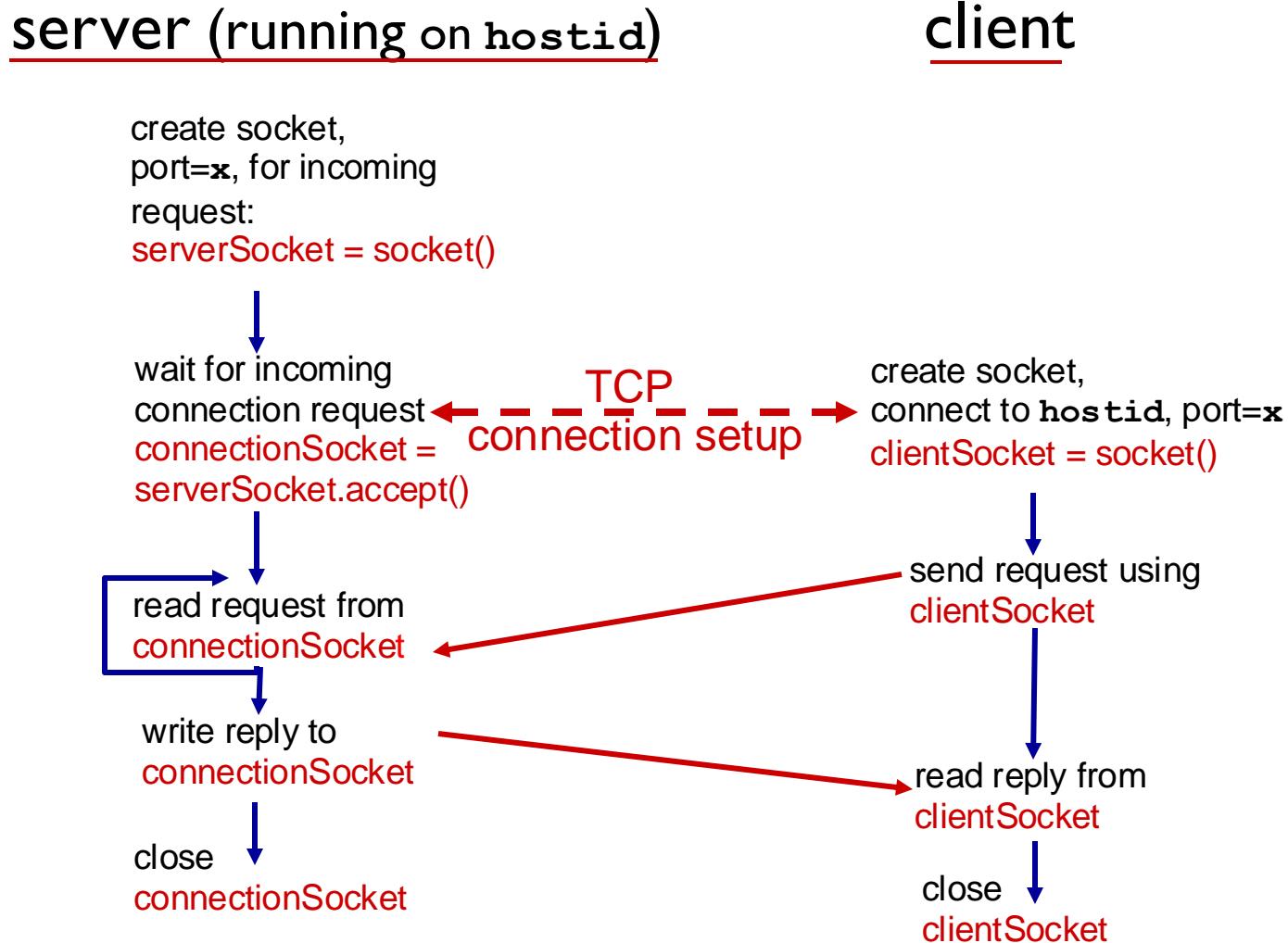
- When contacted by client, server TCP creates new socket for server process to communicate with that particular client

- Allows server to talk with multiple clients
- Source port numbers used to distinguish clients (more in Chap 3)

## Application viewpoint:

TCP provides reliable, in-order byte-stream transfer ("pipe") between client and server

# Client/server socket interaction: TCP



# Example app: TCP server

## *Python TCP Server*

```
from socket import *

create TCP welcoming  
socket → serverPort = 12000  
  
server begins listening for  
incoming TCP requests → serverSocket = socket(AF_INET, SOCK_STREAM)  
serverSocket.bind(('', serverPort))  
serverSocket.listen(1)  
print('The server is ready to receive')

loop forever → while True:  
  
server waits on accept()  
for incoming requests, new  
socket created on return → connectionSocket, addr = serverSocket.accept()  
  
read bytes from socket (but  
not address as in UDP) → sentence = connectionSocket.recv(1024).decode()  
capitalizedSentence = sentence.upper()  
connectionSocket.send(capitalizedSentence.  
encode())  
  
close connection to this  
client (but not welcoming  
socket) → connectionSocket.close()
```

# Example app: TCP client

## *Python TCPClient*

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input('Input lowercase sentence:')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print('From Server:', modifiedSentence.decode())
clientSocket.close()
```

create TCP socket for  
server, remote port 12000 → clientSocket = socket(AF\_INET, **SOCK\_STREAM**)

No need to attach server  
name, port → clientSocket.connect((serverName, serverPort))

# Application Layer - Summary

- Application architectures
  - client-server
  - P2P
- Application service requirements
  - reliability, bandwidth, delay
- Internet transport service model
  - connection-oriented, reliable: TCP
  - unreliable, datagrams: UDP
- Specific protocols:
  - HTTP
  - DNS
  - P2P: BitTorrent
- Socket programming:
  - TCP, UDP sockets

# Thanks.