

Computer Network I

Reti di Calcolatori I

Università di Napoli Federico II – Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Informatica

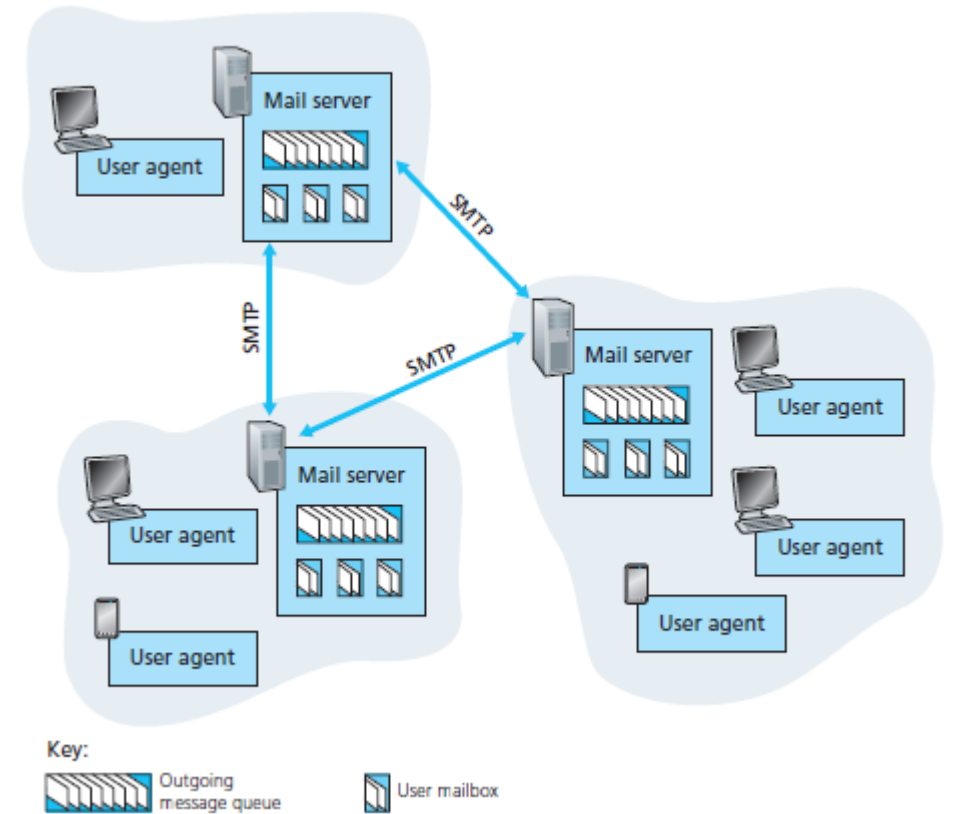
Riccardo Caccavale
(riccardo.caccavale@unina.it)



Application Layer

Mails and SMTP

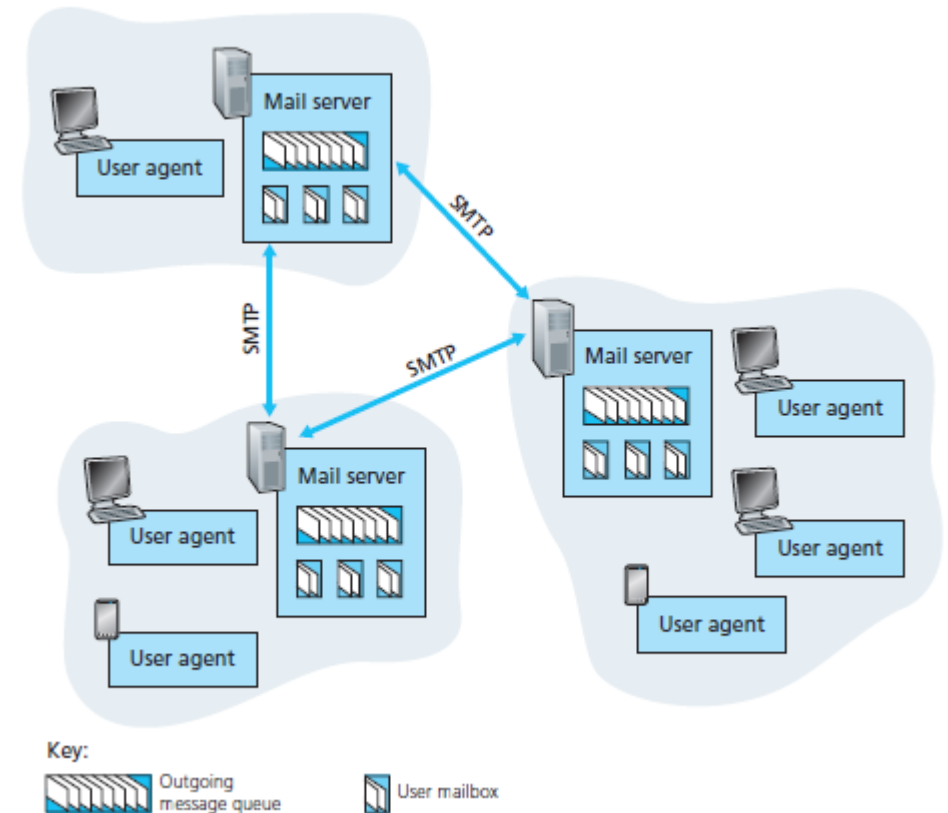
- Electronic mailing (e-mail) is one of the oldest and most important internet applications.
- The **Simple Mail Transfer Protocol (SMTP)**: is the main application-layer protocol for Internet electronic mail.
- There are two elements involved:
 - **User agent**: application allowing mail management (e.g., Outlook, Apple Mail, Gmail, etc.).
 - **Mail Server**: a server that stores mails and maintain user-specific mailboxes.



Application Layer

Mails and SMTP

- SMTP mainly **works on mail servers** to allow them to exchange mails.
- Once a new mail arrives on the server, it is stored inside the **user-specific mailboxes**, waiting to be download locally by the user agent.
- Users **do not exchange mails directly** (as in instant messaging). It is preferred to use more reliable and specialized mail servers.
- In SMTP there are a **client-side** (sender) and a **server-side** (receiver) that run on mail servers, both using the reliable **TCP** to transfer mail.

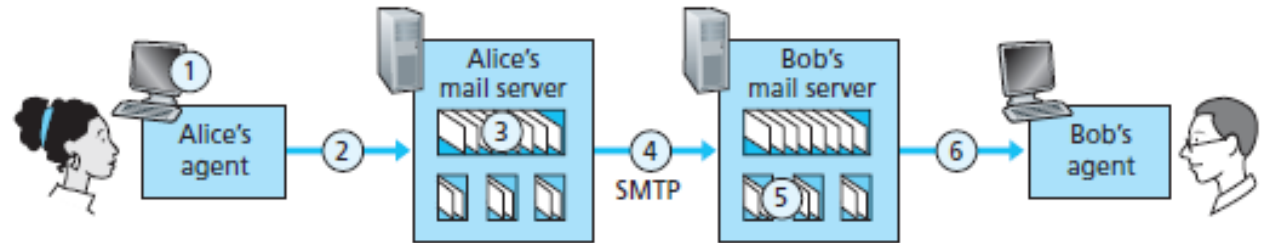


Application Layer

SMTP

- Let's assume to have host A (Alice) that is sending an e-mail to host B (Bob), we have 4 elements involved:

- Alice's agent.
- Alice's mail server.
- Bob's agent.
- Bob's mail server.



- The process works as follows:

1. Alice **invokes her user agent** for e-mail, provides Bob's e-mail address (e.g., bob@someschool.edu), composes a message, and instructs the user agent to send the message.
2. Alice's **user agent sends the message to her mail server**, where it is placed in a message queue.
3. The client side of SMTP, running on Alice's mail server, sees the message in the message queue. It **opens a TCP connection to an SMTP server**, running on Bob's mail server.
4. After some initial SMTP handshaking, the SMTP client sends Alice's **message into the TCP** connection.
5. At Bob's mail server, the **server side of SMTP receives the message**. Bob's mail server then places the message in **Bob's mailbox**.
6. Bob **invokes his user agent** to read the message at his convenience.

Application Layer

SMTP

- First, the client SMTP (running on the sending mail server host) has **TCP establish a connection to port 25 at the server SMTP** (running on the receiving mail server host).
- If the server is down, the client tries again later. Once this connection is established, **the server and client perform some application-layer handshaking**: SMTP clients and servers introduce themselves before transferring information.
 - During this SMTP handshaking phase, the SMTP client indicates the e-mail address of the sender (the person who generated the message) and the e-mail address of the receiver.
- SMTP can count on the reliable data transfer service **of TCP to get the message to the server without errors**.

Application Layer

Accessing mails

- Deployment of **SMTP servers is more reliable** with respect to direct user-to-user mailing:
 - Servers are always on.
 - Servers are certified.
 - The server continuously tries to re-send mails if delivery fails (which is a quite computationally expansive process).
- On the other hand, to access the mails from the server **an additional client-server application is needed** (and a different protocol), most popular are:
 - Post Office Protocol—Version 3 (POP3).
 - Internet Mail Access Protocol (IMAP).
 - HTTP.

Application Layer

Accessing mails: POP3

- The **Post Office Protocol - Version 3** (POP3) is an extremely simple mail access protocol.
- The user agent (the client) opens a **TCP connection** to the mail server (the server) on port 110.
- With the TCP connection established, POP3 progresses through three phases:
 1. **Authorization**: the user agent **sends a username and a password** to authenticate the user.
 2. **Transaction**: the user agent can **retrieve messages**, mark messages for deletion, remove deletion marks, and obtain mail statistics.
 3. **Update**: after the client has issued the quit command, ending the POP3 session, the mail **server deletes the messages** that were marked for deletion.

Application Layer

Accessing mails: IMAP

- With POP3 access, messages can just **be downloaded or deleted**. Action like searching or organizing mails into folders are not considered (these must be done on the local machine).
- The **Internet Mail Access Protocol (IMAP)** allow **servers to provide additional features**:
 - Managing and **creating folders**.
 - Perform **search** into remote folders for messages matching specific criteria.
 - Allow user agent to obtain just parts of messages (e.g., header only, attachments only, etc.).
 - Allow **multiple clients** to be connected to the same server.

Application Layer

Accessing mails: HTTP

- More and more users today are sending and accessing their **e-mails through their Web browsers**.
- HTTP Web-based access was introduced by Hotmail in the mid 1990 and is now the mainstream approach (Google, Yahoo!, Virgilio, etc.) and it is used by almost every major university and corporation (UNINA included).
- With this service, **the user agent is an ordinary Web browser**, and the user communicates with its remote mailbox via HTTP rather than through the POP3 or IMAP protocols.
- This works for user-agents, while mail **servers still rely on the standard SMTP** to exchange messages.

Application Layer

DNS

- There are two ways to identify a host: by **hostname** and by **IP address**. People prefer the more mnemonic hostname identifier, while network devices prefer fixed-length, hierarchically structured IP addresses.
 - Example: www.unina.it -> 143.225.15.50
- The **Domain Name System** (DNS) is an application-layer protocol that manages translation from hostnames to IP addresses.
- It is a **client-server protocol** in which a DNS-client asks to a DNS-server for a specific hostname-to-address translation.
- DNS **servers are often UNIX machines** running the Berkeley Internet Name Domain (BIND) software, **typically using UDP** connection and port number 53.

Application Layer

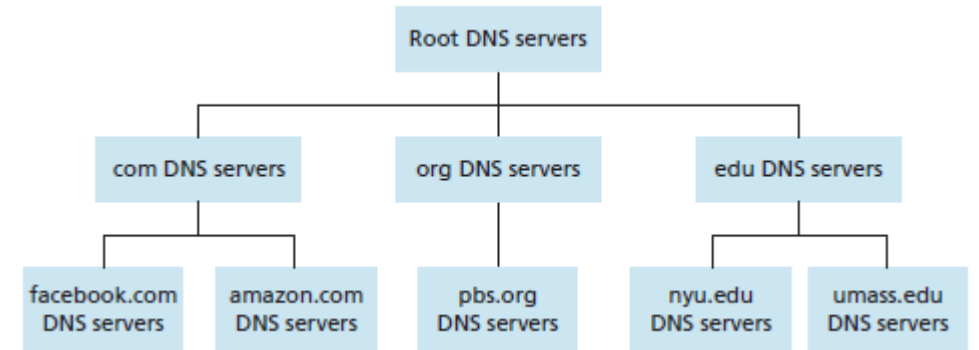
DNS

- For example, if a browser (HTTP client) requests the `www.someschool.edu/index.html` URL, the related IP address is retrieved as follows:
 1. The **browser extracts the hostname**, `www.someschool.edu`, from the URL and passes the hostname to the client side of the DNS application (in C/C++ applications we use the `gethostbyname()` function to perform that).
 2. The DNS **client sends a query** containing the hostname to a DNS server.
 3. The DNS **client receives a reply** containing the IP address for the hostname.
 4. Once the browser receives the IP address from DNS, it can initiate a **TCP connection to the HTTP server** process located at port 80 at that IP address.
- Notice that DNS process is not trivial! **it may add an additional delay** (sometimes substantial) to the Internet applications.

Application Layer

DNS: Distributed Servers

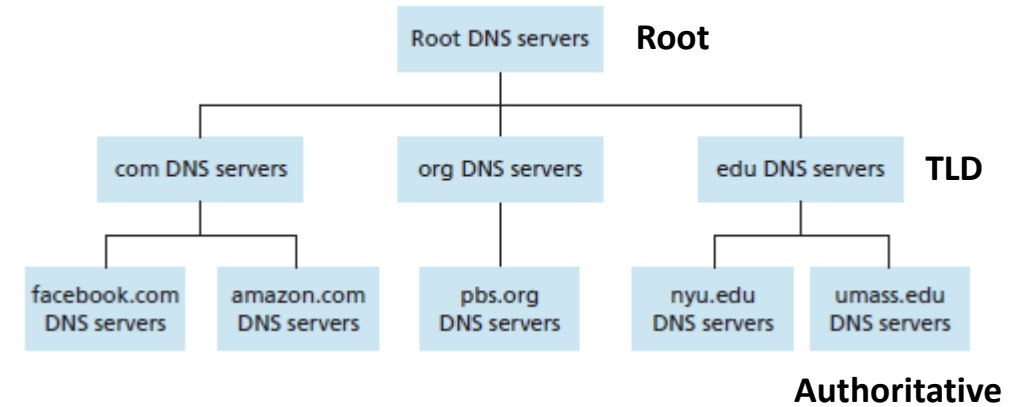
- Internet **DNS is distributed and hierarchically organized**. There are several servers all around the world that provide DNS service.
- This approach is preferred to a centralized DNS for several reasons:
 - **Avoid a single point of failure**: if the DNS server crashes, so does the entire Internet!
 - **Regulate traffic volume**: hundreds of millions of hosts uses DNS.
 - **Better reachability**: a single DNS server cannot be “close to” all clients.
 - **Easy maintenance and update**: we can update a local DNS without effecting the whole Internet.



Application Layer

DNS: Hierarchy

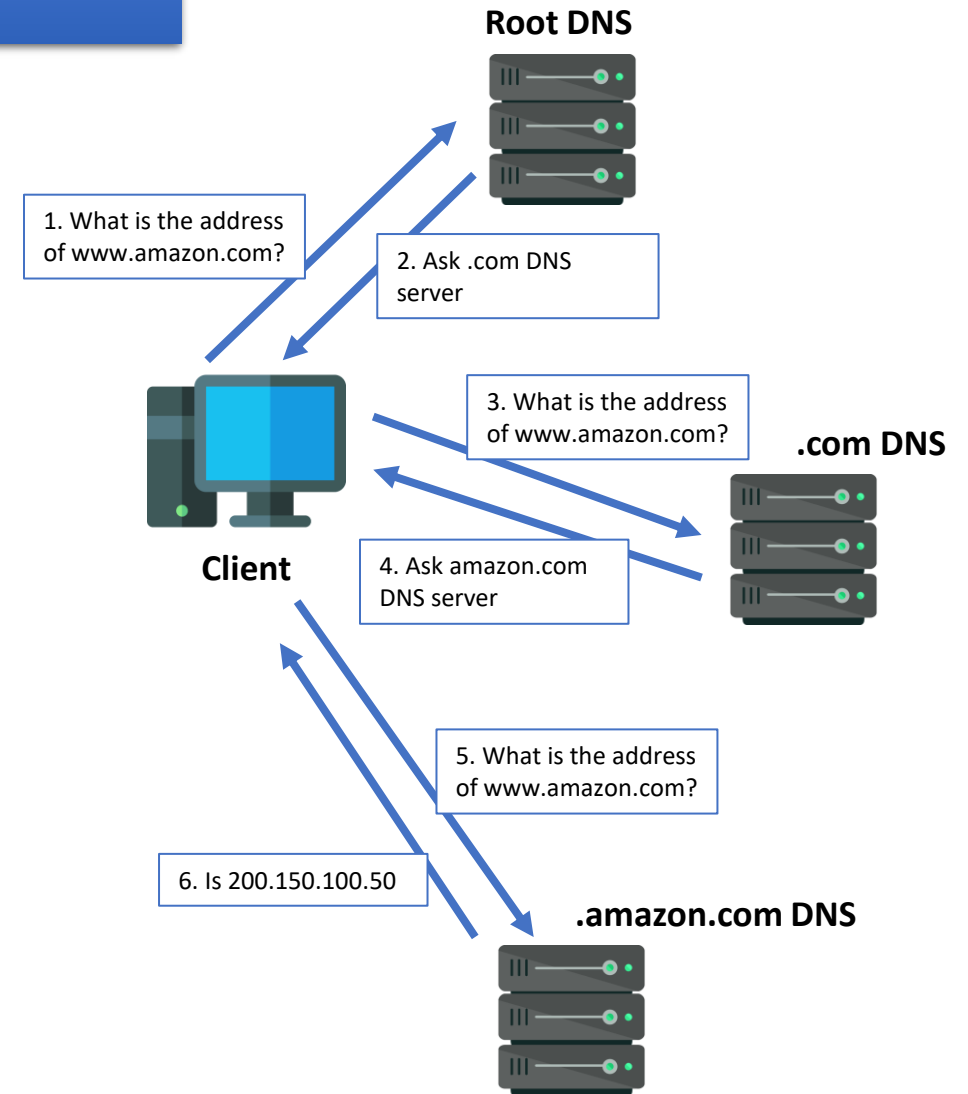
- There are basically 3 classes of DNS servers:
 - **Root DNS servers:** there are over 400 root name servers scattered all over the world, managed by 13 different organizations. Root name servers provide the IP addresses of the TLD servers.
 - **Top-level domain (TLD) servers:** there are one or more for each top-level domain (e.g., .com, .org, .net, .it, .uk, .fr, etc.). TLD servers provide the IP addresses for authoritative DNS servers.
 - **Authoritative DNS servers:** provide real hostname-IP mapping. These can be directly owned by organizations (amazon, facebook, etc.) or offered by third-party.



Application Layer

DNS: Example

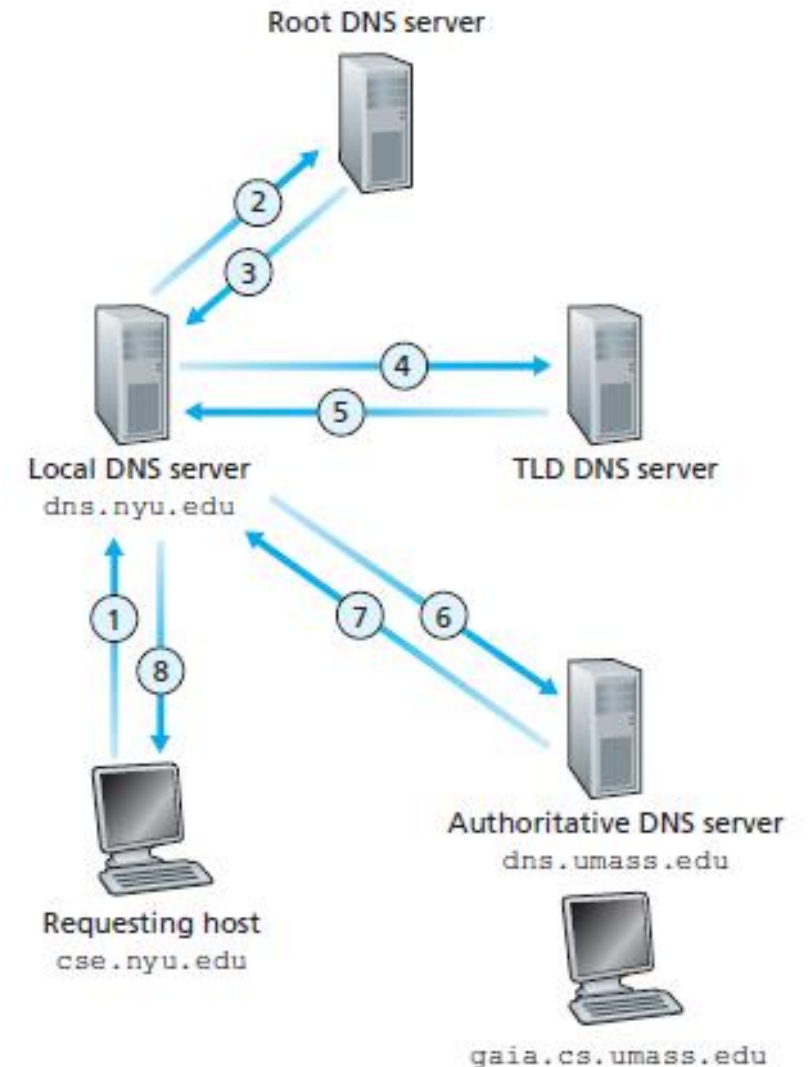
- When a DNS client wants to determine the IP address for the hostname (e.g., `www.amazon.com`) it happens that:
 - The client first **contacts one of the root servers**, which returns IP addresses for TLD servers for the top-level domain “.com”.
 - The client then **contacts one of these TLD servers**, which returns the IP address of an authoritative server for “amazon.com”.
 - Finally, **the client contacts one of the authoritative servers** for amazon.com, which returns the IP address for the hostname “www.amazon.com”.



Application Layer

DNS: Resolver

- There is another important type of DNS server called the **local DNS server** (or **DNS resolver**).
- A local DNS server does not strictly belong to the hierarchy of servers and is **typically managed by ISPs**.
 - Google also provides similar servers all around the world (Google Public DNS) on address 8.8.8.8 or 8.8.4.4.
- When a host connects to an ISP, the ISP provides the host with the IP addresses of **one or more of its local DNS servers** (that are typically “close to” the host).
- When a host makes a DNS query, the query is sent to the local DNS server, which **acts as a proxy**, forwarding the query to the DNS server hierarchy.



Application Layer

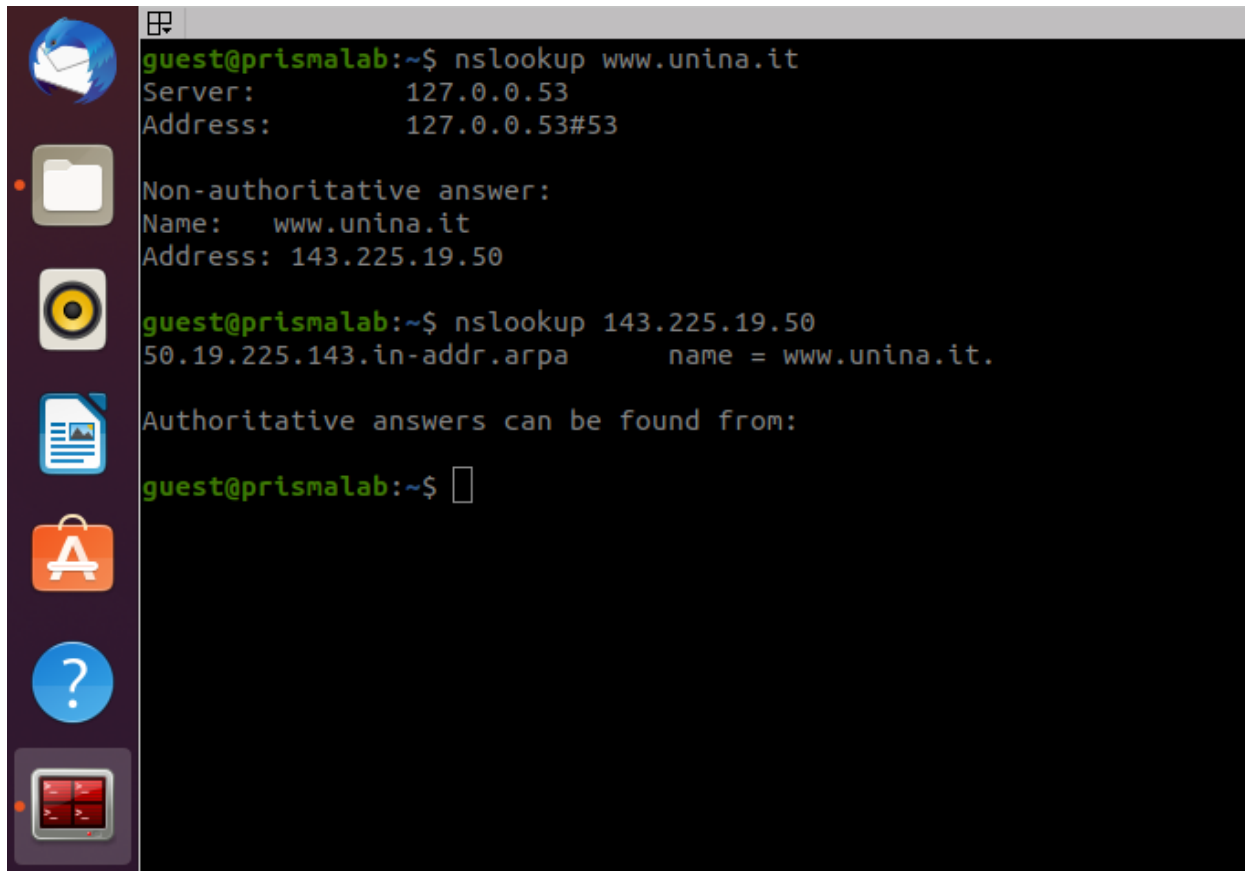
DNS: Aliasing

- DNS provides an **aliasing** service for servers in which complicated/long names (aka canonical) are associated to a more simple/short ones (aka alias).
- **Host aliasing**: hosts with complicated hostnames (e.g., web sites such as relay1.west-coast.enterprise.com) can be **associated to more mnemonic aliases** (e.g., enterprise.com or www.enterprise.com).
- Note that DNS can still be invoked by an application **to obtain the canonical (original) hostname** for a supplied alias as well as the associated IP address.

Application Layer

Lookup Example

- In Linux (and other OSs) we can use nslookup (name server lookup) command to ask for a hostname-address translation.

A terminal window with a dark background and a vertical sidebar on the left containing icons for various applications. The terminal shows the execution of the nslookup command twice. The first command is 'nslookup www.unina.it', which returns the server address (127.0.0.53) and a non-authoritative answer for the IP address of www.unina.it (143.225.19.50). The second command is 'nslookup 143.225.19.50', which returns the authoritative answer for the IP address, identifying it as www.unina.it.

```
guest@prismalab:~$ nslookup www.unina.it
Server:          127.0.0.53
Address:         127.0.0.53#53

Non-authoritative answer:
Name:   www.unina.it
Address: 143.225.19.50

guest@prismalab:~$ nslookup 143.225.19.50
50.19.225.143.in-addr.arpa      name = www.unina.it.

Authoritative answers can be found from:

guest@prismalab:~$
```

We can get translations in both ways:

- To get IP address from hostname (first).
- To get hostname from IP address (second).

Application Layer

DNS: Load Distribution

- A DNS can be used to perform **load distribution** among replicated servers (aka, **round-robin DNS**).
- Typically, busy sites (e.g., google, amazon, CNN, etc.) are **replicated over multiple servers**, with each server running on a different end system and each having a different IP address (multiple IP addresses associated with one canonical hostname).
- The DNS database contains this set of IP addresses. When clients make a DNS query for a name mapped to a set of addresses, the server may **rotate the order in which addresses are replied**.

Application Layer

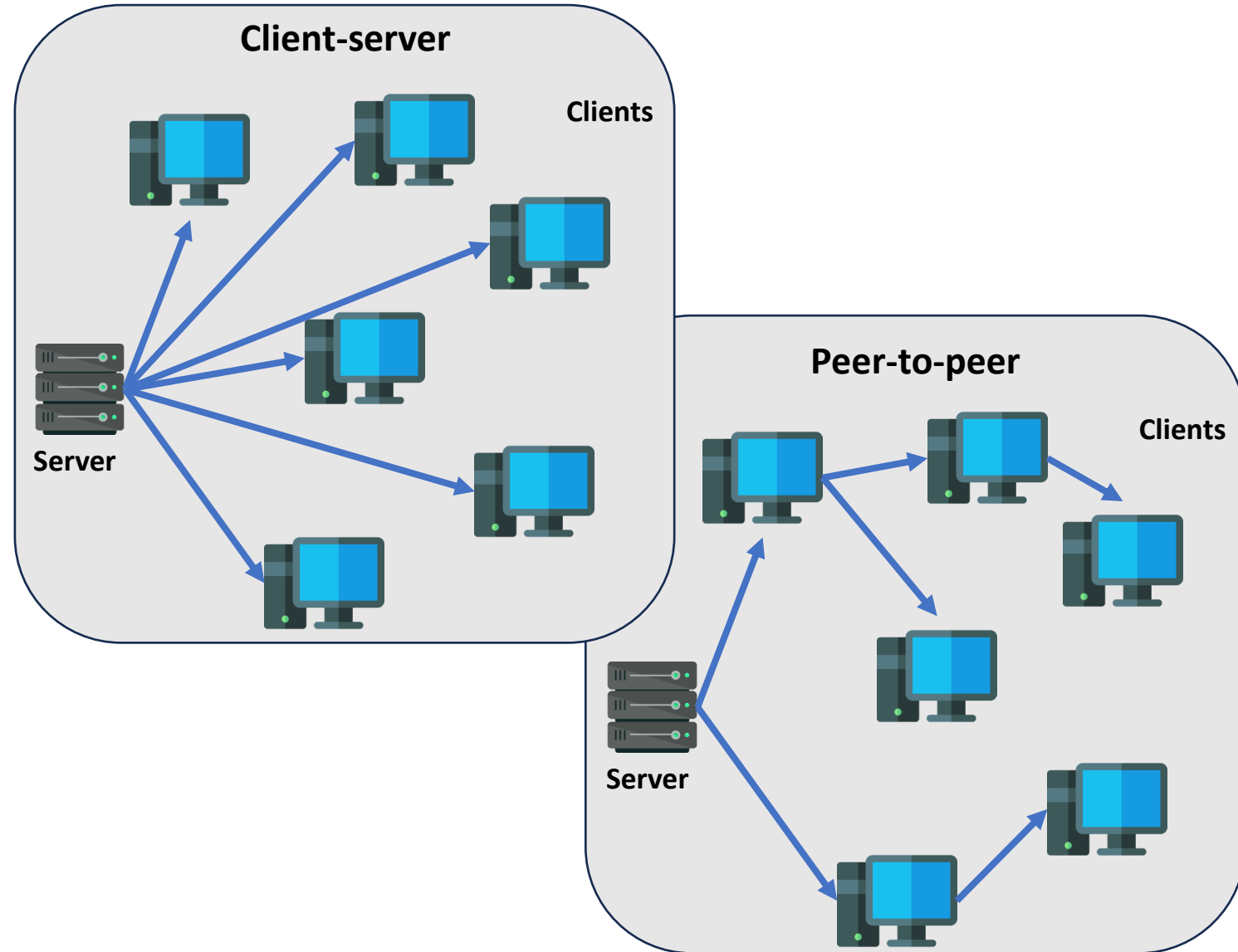
Peer-to-Peer

- Differently from client-server, P2P architecture **makes minimal (if none) use of servers**, here we have **pairs of intermittently connected hosts** (peers) that communicate directly with each other.
- The peers are **not owned by a service provider** but are instead desktops and laptops controlled by users.
- One natural application of P2P is **file sharing**:
 - In a **client-server** architecture, the server must **share files to all clients** (which is a serious burden on the server and requires large bandwidth).
 - In a **P2P** architecture, the peers that receive the file may also **share it to other peers**. Somehow, peers are clients and servers at the same time.

Application Layer

Peer-to-Peer

- In file sharing, the P2P approach typically **scales better** than client-server one.
- On the other hand, P2P approaches can be quite **complex to implement**.
- There may also be **security issues** in having all this clients in direct communication.



Application Layer

BitTorrent

- A popular P2P protocol for file sharing/distribution is **BitTorrent**.
 - BitTorrent estimated users are 150-170 million (in 2023).
- A **torrent** is the collection of all peers participating in the distribution of a particular file. Peers in a torrent download **equal-size chunks** of the file from one another (typical chunk size is 256 kbytes).
- When a peer joins a torrent (having no chunks):
 - It starts by **accumulating chunks**.
 - While the peer downloads chunks **it also uploads chunks** to other peers.
 - Once the peer **acquires the entire file**, it may (selfishly) **leave the torrent**, or (altruistically) **stay in the torrent** and continue to upload chunks to other peers.
- Peers may **leave the torrent at any time** with only a subset of chunks, and later rejoin the torrent.

Application Layer

BitTorrent: tracker

- Each torrent has an infrastructure node called **tracker** that takes track of the peers participating to the torrent (trackers are basically **servers**).
- When a **new peer** joins a torrent:
 - It **registers** itself with the tracker and **periodically informs** the tracker about its status.
 - The tracker **provides the IP addresses** of a randomly selected subset of peers from the torrent.
- Having the list of peers, the new host attempts to establish concurrent TCP **connections with all the peers on this list** (neighbors).
 - During the execution, some of the connected peers **may leave** while other peers (outside the initial list) may attempt to **establish new connections**.
- At any given time, each peer will have a **subset of chunks from the file**, with different peers having different subsets. Periodically, a host will ask each connected peers (over the TCP connections) for **the list of the chunks they have**.

Application Layer

BitTorrent

- The **downloading client** decides which chunks to request (and to whom) following a rarest-first principle:
 - **Rarest-first**: the chunks with **fewest repeated copies** among the neighbors are prioritized. In this way, the **rarest chunks get more quickly redistributed**, so to (roughly) equalize the numbers of copies in the torrent.
- The **uploading client** decides which requests are served following two intertwined principles:
 - **Trading**: hosts give priority to **the best 4 neighbors** that are currently supplying data at the **highest rate**. This check is periodically performed (10 seconds) and the list of 4-best neighbors is updated.
 - **Random selection**: every 30 seconds, a host also picks **one additional neighbor** at random and sends it chunks. If this **random exchange is good** (the 2 hosts are good partners) **they will enter the respective best lists**. This process also allows peers with compatible upload rates to find each other.