Chapter 1

L11 - DNS & P2P

1.1 Global Process Naming and Addressing

Every process communicating over the Internet is uniquely identified by a combination of an IP address and a port number.

```
172.67.2.106:443
```

- **172.67.2.106**: IP address of a network interface.
- **443**: Port number (details below).

1.1.1 Network Interfaces

A network interface connects an end-system (such as a computer or smart device) to a network.

- Each end-system has at least one network interface.
- Each network interface has at least one unique IP address.
- Examples: Your laptop's network card, or your car's onboard Wi-Fi.

1.1.2 Port Numbers

Port numbers identify specific processes within a local system.

- Each process using network communication is assigned a unique port number.
- On a server, port numbers for certain applications are restricted and standardized.

Reserved Port Numbers

Some port numbers are universally reserved for well-known services.

- Reserved ports are used exclusively by specific server processes.
- Client processes and other servers should not use these reserved ports.
- Examples: Ports 80, 8080, 443, and 8443 are reserved for web servers.

1.1.3 Web Server Port Numbers

Web servers use standardized ports for HTTP and HTTPS communication:

- **80, 8080**: HTTP (standard web traffic)
- **443, 8443**: HTTPS (secure web traffic)

Note: HTTPS is the encrypted, secure version of HTTP.

1.1.4 The Domain Name System (DNS)

DNS is an essential Internet system that translates human-friendly domain names into IP addresses.

Example

When you enter a URL such as:

- **https://www.epfl.ch/labs/nal/publications**
- **www.epfl.ch**: Hostname (DNS name)
- **labs/nal/publications**: Path to the resource

The Domain Name System (DNS) maps domain names like `www.epfl.ch` to IP addresses (e.g., `128.178.211.3`). This translation allows users to access network interfaces by name rather than by numeric address, making the Internet more user-friendly and scalable.

DNS Translation in Action

1. The web client reads the URL and identifies the protocol: **https**.
2. For HTTPS, the default port number is **443** (or sometimes 8443).
3. The web client extracts the DNS name (`www.epfl.ch`) and sends a DNS query to obtain its IP address.
4. Once resolved, it can contact the web server process at `128.178.211.3:443`.

1.1.5 Application Design

The design of a distributed application requires answering:

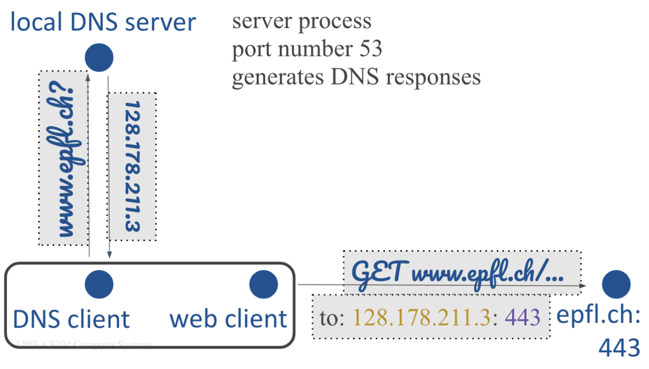
- How many processes/threads are involved, and their roles.
- What communication protocols are used between them.

For DNS, this means defining client and server roles and their communication.

1.1.6 How Does DNS Work?

DNS uses a client-server architecture with specialized processes:

- When you type a URL, your web client extracts the DNS name (e.g., `www.epfl.ch`).
- A local process, called the **DNS client** or **stub resolver**, creates a DNS query.
- The DNS client sends this query to a **local DNS server** (resolver), typically nearby.
- The DNS server process listens on port 53 and replies with the IP address.
- The DNS client relays the response back to the web client.
- The web client can now communicate with the web server using the IP address and correct port.



Why Not a Single DNS Server?

A single-server DNS design would not scale:

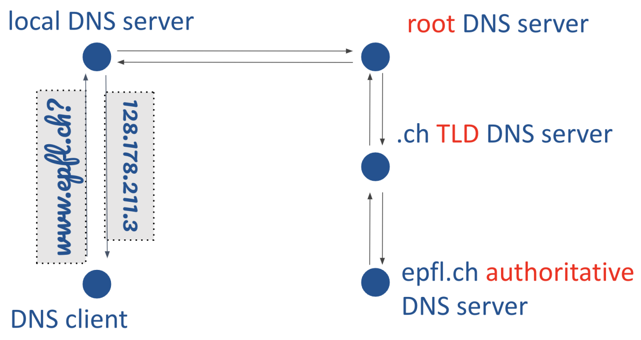
- It would be overloaded by global traffic.
- It could not provide low latency to all clients.
- It would be a single point of failure.
- Maintenance would be unmanageable at Internet scale.

1.1.7 Scalability

Scalability is the ability of a system to grow while maintaining good performance and reasonable cost, independently from size.

1.1.8 Distributed DNS

The Domain Name System (DNS) is not managed by a single server, but as a distributed, hierarchical system. This structure ensures scalability, reliability, and high performance for name resolution on the Internet.

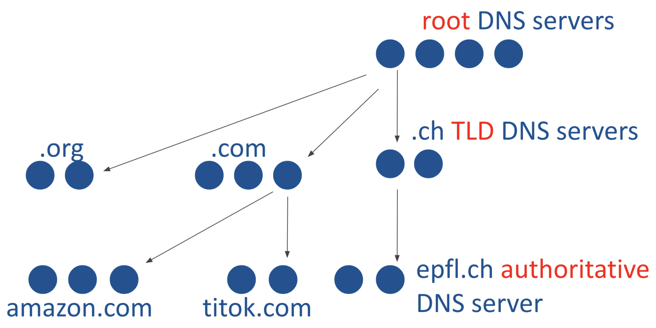


When a DNS client wants to resolve a domain such as `epfl.ch`, the following process occurs:

- The client queries its local DNS server.
- If the local server does not know the answer, it queries a **root DNS server**.
- If the root server does not know the answer, it forwards the query to a **TLD (Top-Level Domain) server** (e.g., for `.ch`).
- If the TLD server does not know, it asks the **authoritative DNS server** for the specific domain (e.g., `epfl.ch`).
- The authoritative DNS server provides the definitive answer for the requested domain.

DNS Hierarchy

DNS servers are organized hierarchically to efficiently manage and resolve domain names. At each level, servers have specific responsibilities.



- **Root DNS servers** are at the top of the hierarchy. They know how to reach all TLD servers.
- **TLD DNS servers** manage domains for top-level domains (e.g., `.ch`, `.com`, `.org`). They know the authoritative servers for all second-level domains under their TLD.
- **Authoritative DNS servers** hold the final, definitive information about their specific domain (e.g., `epfl.ch`, `ricardo.ch`). They know the IP addresses for all hostnames in their domain.

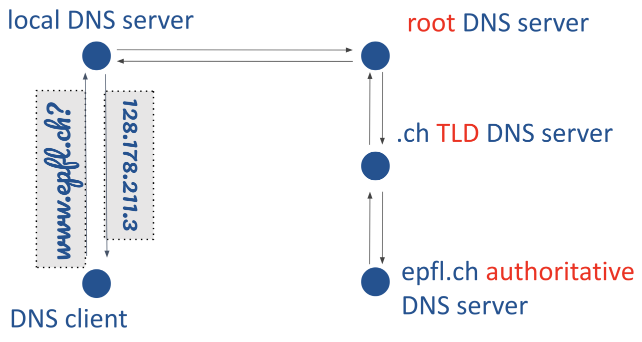
For example, an authoritative server for `epfl.ch` knows the addresses of all hosts ending with `epfl.ch`. Similarly, a TLD server for `.ch` knows how to reach authoritative servers for all `.ch` domains.

1.1.9 DNS Query Resolution

There are two main approaches for a DNS query to reach a server that can answer it. Each approach offers different trade-offs in terms of efficiency and server workload.

Recursive Query

In a **recursive query**, the client asks a DNS server to resolve the name entirely. The server takes responsibility for querying other DNS servers until it obtains the answer, which it then returns to the client.

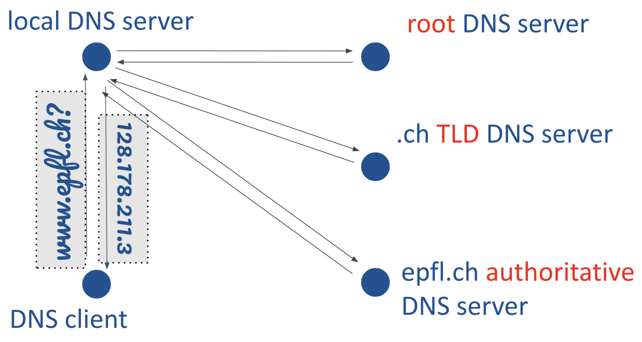


- Every DNS client knows at least one local DNS server.
- Every DNS server knows at least one root DNS server.
- Each root DNS server knows at least one TLD DNS server per TLD.
- Each TLD DNS server knows at least one authoritative DNS server for each second-level domain it manages.

If the hierarchy works correctly, every DNS query should eventually receive a response.

Iterative Query

In an **iterative query**, a DNS server responds to the client with the address of the next server to contact if it cannot answer directly. The client then queries that server, repeating the process until it finds the authoritative answer.



This model is common in practice, often as a mix of recursive and iterative queries: the local DNS server handles recursion for the client, while upstream servers use iteration.

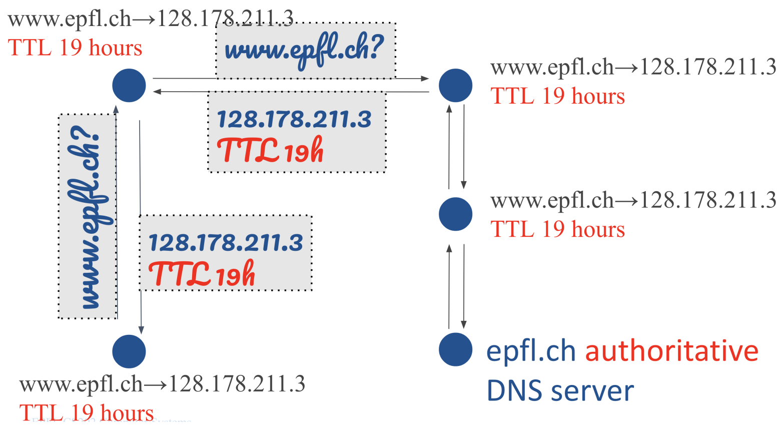
Benefits of the Hierarchy

This hierarchical structure significantly improves DNS scalability:

- No DNS server needs to know all domain names worldwide.
- Each server only needs to know where to find the next more specific server.

DNS Caching

DNS queries can involve multiple servers, which may introduce noticeable delays. To improve performance, DNS extensively uses **caching**.
Caching allows both DNS clients and servers to store recent DNS query results, so repeated requests for the same domain name can be answered much faster without contacting other servers.
The primary challenge of caching is ensuring the data remains up-to-date. Each DNS response includes a **Time To Live (TTL)**, which specifies the maximum time the result should be cached. When the authoritative server sends a response, it sets an initial TTL. As the response is passed along the DNS hierarchy, each server reduces the TTL by the elapsed time before forwarding or serving the cached result.



Example:

- The authoritative server for `epfl.ch` responds with a TTL of 24 hours.
- After 4 hours, a TLD server responds with a TTL of 20 hours.
- After another hour, a root server responds with a TTL of 19 hours, and so on.
- Each caching server and client will use the remaining TTL value.

DNS Processes

DNS operates through a combination of queries and responses, with the help of a distributed hierarchy:

- The **DNS client** generates queries to resolve domain names.
- The **local DNS server** (also called a resolver) processes client requests and can respond directly from cache or contact higher-level servers.
- The **hierarchy of DNS servers** assists in generating accurate responses for the client.

DNS Hierarchy: Summary

The DNS hierarchy consists of three main layers:

- **Root servers** (top): Know how to reach all TLD servers.
- **Top-level domain (TLD) servers** (middle): Responsible for specific TLDs (e.g., `.ch`, `.com`), know authoritative servers for each second-level domain.
- **Authoritative servers** (bottom): Responsible for a specific domain (e.g., `epfl.ch`), and know all hostnames within that domain.

DNS Protocol

DNS communication is based on a simple query-response protocol:

- A **query** (or request) is sent from a client or server to another DNS server to obtain information about a domain name.
- A **response** contains the answer, provided as a list of *resource records* (RRs).

A **resource record (RR)** is a data structure containing specific DNS information. There are several common types:

- **A record:** Maps a DNS name to an IPv4 address.
- **AAAA record:** Maps a DNS name to an IPv6 address.
- **CNAME record:** Maps a DNS name to another DNS name (alias).
- **MX record:** Specifies the mail server for a domain.
- **SOA record:** Indicates the Start of Authority for a domain.

Prof. Note: You do not need to memorize all types, but it is useful to know the main examples and their purpose.