How the Domain Name System Works

Understand the detailed working of the domain name system.

We'll cover the following

Through this lesson, we'll answer the following questions:

- How is the DNS hierarchy formed using various types of DNS name servers?
- How is caching performed at different levels of the Internet to reduce the querying burden over the DNS infrastructure?
- How does the distributed nature of the DNS infrastructure help its robustness?

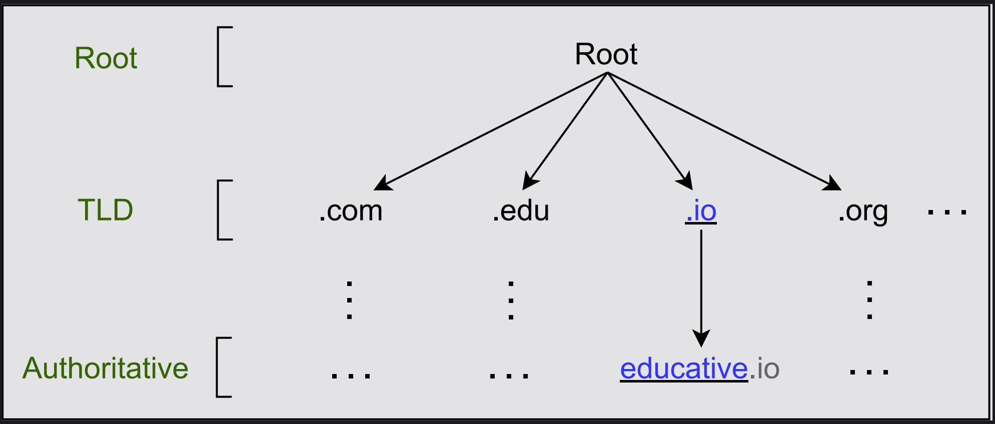
Let's get started.

DNS hierarchy

As stated before, the DNS isn't a single server that accepts requests and responds to user queries. It's a complete infrastructure with name servers at different hierarchies.

There are mainly four types of servers in the DNS hierarchy:

1. **DNS resolver:** Resolvers initiate the querying sequence and forward requests to the other DNS name servers. Typically, DNS resolvers lie within the premise of the user's network. However, DNS resolvers can also cater to users' DNS queries through caching techniques, as we will see shortly. These servers can also be called local or default servers.
2. **Root-level name servers:** These servers receive requests from local servers. Root name servers maintain name servers based on top-level domain names, such as `.com`, `.edu`, `.us`, and so on. For instance, when a user requests the IP address of educative.io, root-level name servers will return a list of top-level domain (TLD) servers that hold the IP addresses of the `.io` domain.
3. **Top-level domain (TLD) name servers:** These servers hold the IP addresses of authoritative name servers. The querying party will get a list of IP addresses that belong to the authoritative servers of the organization.
4. **Authoritative name servers:** These are the organization's DNS name servers that provide the IP addresses of the web or application servers.



DNS hierarchy for resolution of domain/host names

Point to Ponder

Question

How are DNS names processed? For example, will educative.io be processed from left to right or right to left?

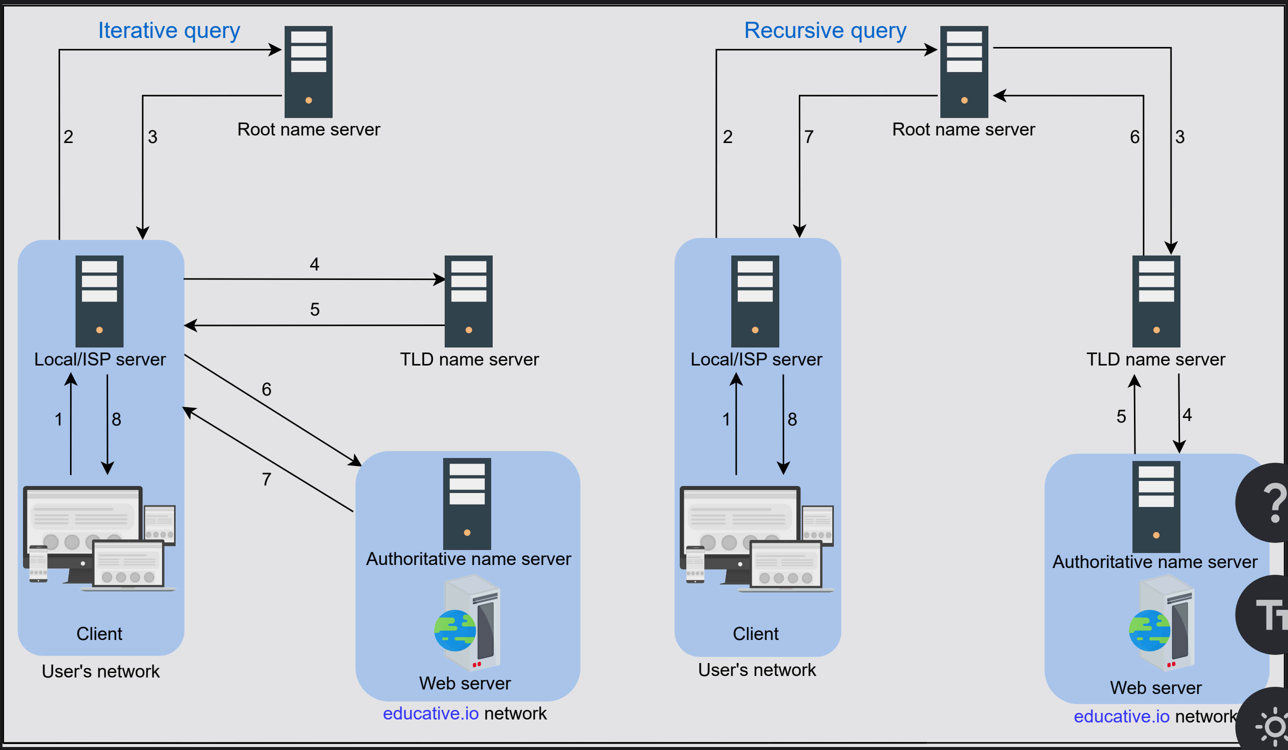
Show Answer

Iterative versus recursive query resolution

There are two ways to perform a DNS query:

1. **Iterative:** The local server requests the root, TLD, and the authoritative servers for the IP address.
2. **Recursive:** The end user requests the local server. The local server further requests the root DNS name servers. The root name servers forward the requests to other name servers.

In the following illustration (on the left), DNS query resolution is iterative from the perspective of the local/ISP server:



Iterative versus recursive query

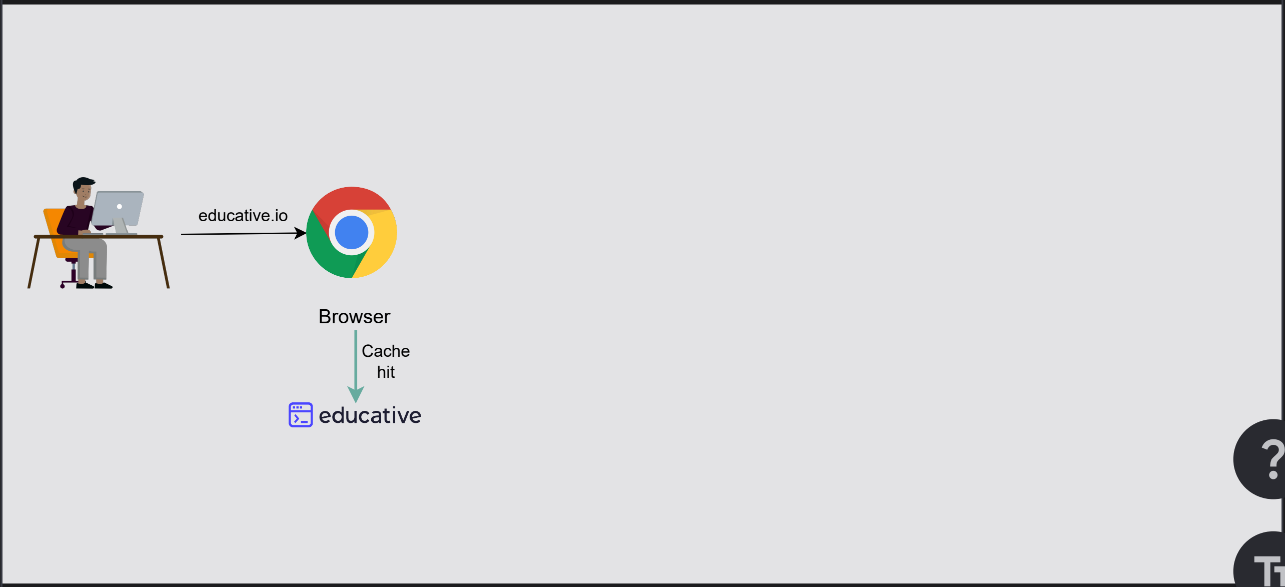
> **Note:** Typically, an iterative query is preferred to reduce query load on DNS infrastructure.

Fun Fact

Caching

Caching refers to the temporary storage of frequently requested resource records. A ***record*** is a data unit within the DNS database that shows a name-to-value binding. Caching reduces response time to the user and decreases network traffic. When we use caching at different hierarchies, it can reduce a lot of querying burden on the DNS infrastructure. Caching can be implemented in the browser, operating systems, local name server within the user's network, or the ISP's DNS resolvers.

The slideshow below demonstrates the power of caching in the DNS:



The user requests to visit a URL, and the browser has cached the domain name to IP address mapping

1 of 7

Note: Even if there is no cache available to resolve a user's query and it's imperative to visit the DNS infrastructure, caching can still be beneficial. The local server or ISP DNS resolver can cache the IP addresses of TLD servers or authoritative servers and avoid requesting the root-level server.

DNS as a distributed system

Although the DNS hierarchy facilitates the distributed Internet that we know today, it's a distributed system itself. The distributed nature of DNS has the following advantages:

- It avoids becoming a single point of failure (SPOF).
- It achieves low query latency so users can get responses from nearby servers.
- It gets a higher degree of flexibility during maintenance and updates or upgrades. For example, if one DNS server is down or overburdened, another DNS server can respond to user queries.

There are 13 logical root name servers (named letter **A** through **M**) with many instances spread throughout the globe. These servers are managed by 12 different organizations.

Let's now go over how DNS is scalable, reliable, and consistent.

Highly scalable

Due to its hierarchical nature, DNS is a highly scalable system. Roughly 1,000 replicated instances of 13 root-level servers are spread throughout the world strategically to handle user queries. The working labor is divided among TLD and root servers to handle a query and, finally, the authoritative servers that are managed by the organizations themselves to make the entire system work. As shown in the DNS hierarchy tree above, different services handle different portions of the tree enabling scalability and manageability of the system.

Reliable

Three main reasons make the DNS a reliable system:

1. **Caching:** The caching is done in the browser, the operating system, and the local name server, and the ISP DNS resolvers also maintain a rich cache of frequently visited services. Even if some DNS servers are temporarily down, cached records can be served to make DNS a reliable system.
2. **Server replication:** DNS has replicated copies of each logical server spread systematically across the globe to entertain user requests at low latency. The redundant servers improve the reliability of the overall system.
3. **Protocol:** Although many clients rely on the unreliable User Datagram Protocol (UDP) to request and receive DNS responses, it's important to acknowledge that UDP also offers distinct advantages. It is much faster, and therefore, improves DNS performance. Furthermore, internet service reliability has improved since its inception, so UDP is usually favored over TCP. DNS queries are usually retransmitted at the transport layer if there's no response for the previous one. Therefore, request-response might need additional round trips, which provides a shorter delay as compared to TCP, which needs a three-way handshake every time before data exchange.

Point to Ponder

Question

What happens if a network is congested? Should DNS continue using UDP?

Show Answer

Consistent

DNS uses various protocols to update and transfer information among replicated servers in a hierarchy. DNS compromises on strong consistency to achieve high performance because data is read frequently from DNS databases as compared to writing. However, DNS provides eventual consistency and updates records on replicated servers lazily. Typically, it can take from a few seconds up to three days to update records on the DNS servers across the Internet. The time it takes to propagate information among different DNS clusters depends on the DNS infrastructure, the size of the update, and which part of the DNS tree is being updated.

Consistency can suffer because of caching too. Since authoritative servers are located within the organization, it may be possible that certain resource records are updated on the authoritative servers in case of server failures at the organization. Therefore, cached records at the default/local and ISP servers may be outdated. To mitigate this issue, each cached record comes with an expiration time called **time-to-live (TTL)**.

Point to Ponder

Question

To maintain high availability, should the TTL value be large or small?

Show Answer

Test it out

Let's run a couple of commands. Click on the terminal to execute the following commands. Copy the following commands in the terminal to run them. Study the output of the commands:

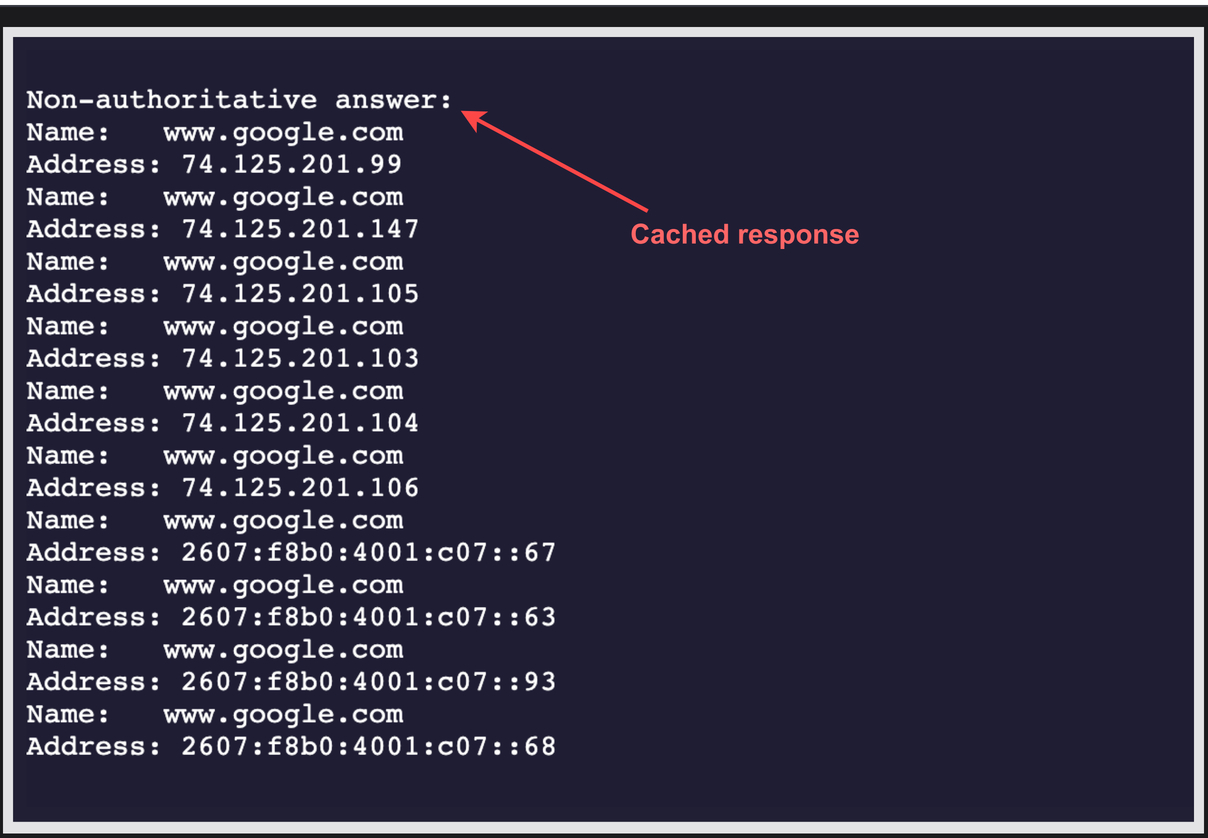
1. `nslookup www.google.com`
2. `dig www.google.com`

Terminal 1

● Terminal

```
Loading...
```

The following slide deck highlights some important aspects of `nslookup` and `dig` output.



```
Non-authoritative answer:
Name:    www.google.com
Address: 74.125.201.99
Name:    www.google.com
Address: 74.125.201.147
Name:    www.google.com
Address: 74.125.201.105
Name:    www.google.com
Address: 74.125.201.103
Name:    www.google.com
Address: 74.125.201.104
Name:    www.google.com
Address: 74.125.201.106
Name:    www.google.com
Address: 2607:f8b0:4001:c07::67
Name:    www.google.com
Address: 2607:f8b0:4001:c07::63
Name:    www.google.com
Address: 2607:f8b0:4001:c07::93
Name:    www.google.com
Address: 2607:f8b0:4001:c07::68
```

The output of nslookup www.google.com

1 of 2

Let's go through the meaning of the output:

The `nslookup` output

- The `Non-authoritative answer`, as the name suggests, is the answer provided by a server that is not the authoritative server of Google. It isn't in the list of authoritative nameservers that Google maintains. So, where does the answer come from? The answer is provided by second, third, and fourth-hand name servers configured to reply to our DNS query—for example, our university or office DNS resolver, our ISP nameserver, our ISP's ISP nameserver, and so on. In short, it can be considered as a cached version of Google's authoritative nameservers response. If we try multiple domain names, we'll realize that we receive a cached response most of the time.

- If we run the same command multiple times, we'll receive the same IP addresses list but in a different order each time. The reason for that is DNS is indirectly performing load balancing. It's an important term that we'll gain familiarity with in the coming lessons.

The `dig` output

- The `Query time: 4 msec` represents the time it takes to get a response from the DNS server. For various reasons, these numbers may be different in our case.
- The `300` value in the `ANSWER SECTION` represents the number of seconds the cache is maintained in the DNS resolver. This means that Google's ADNS keeps a TTL value of five minutes ($\frac{300\ sec}{60}$).

> **Note:** We invite you to test different services for their TTL and query times to strengthen your understanding. You may use the above terminal for this purpose.

Point to Ponder

Question

If we need DNS to tell us which IP to reach a website or service, how will we know the DNS resolver's IP address? (It seems like a chicken-and-egg problem!)