A Crash Course in DNS (Domain Name System)

BYTEBYTEGO
SEP 28, 2023 · PAID

250 2 9 Share ···

What if you woke up tomorrow and could no longer access websites by typing names like google.com or espn.com? Instead, you had to memorize and type a series of numbers just to get online - 172.217.16.206 to check Gmail or 199.181.132.250 to read sports news. Internet browsing would become extremely tedious overnight!

Luckily, this internet nightmare scenario will never happen as long as DNS keeps running smoothly in the background. DNS, short for Domain Name System, is the essential service that saves us from the huge hassle of memorizing numeric IP addresses. It efficiently matches easy-to-remember domain names with their corresponding IP addresses so we can browse seamlessly.

In this article, we explore the intricate workings of this crucial internet backbone.

DNS: The Internet’s Address Book

DNS is a distributed and hierarchical system that acts as the internet's address book. It translates domain names into IP addresses to facilitate efficient communication between devices across the globe. The primary purpose of DNS is to provide a convenient way for users to access websites and resources using easily memorable domain names, while computers and servers use IP addresses for communication behind the scenes.

The DNS is implemented as an Application layer service. It is implemented by the servers that sit at the network edge rather than routers and switches inside the network. This reflects the internet design philosophy of keeping the network core simple and putting complexity at the network’s edge.

Key Services Provided by DNS

DNS provides a number of different services to facilitate the functioning of the Internet. Here is a mind map of the services provided by DNS.

We are going to describe some of the most vital services of DNS below:

Host-to-IP address mapping

DNS maintains a database of domain names and their corresponding IP addresses. This mapping is essential for devices to locate each other on the internet.

For instance, consider the domain name "www.bytebytego.com." Host-to-IP address mapping involves determining the IP address associated with this domain, such as "172.67.21.11".

Host aliasing

DNS supports a feature commonly known as host aliasing, enabled through CNAME (Canonical Name) records. This allows a single IP address or primary domain name to be associated with multiple domain name aliases.

For example, let's consider the primary domain "bytebytego.com". You might want your website to be accessible not just via "bytebytego.com", but also through various aliases like "www.bytebytego.com", "web.bytebytego.com", and "blog.bytebytego.com". By setting up appropriate CNAME records in DNS, all these aliases can point to the primary domain "bytebytego.com". As a result, users can access your website using any of these domain names, all leading to the same destination IP address.

Email Routing

DNS plays a crucial role in email routing through MX records. These records allow a domain to specify which mail servers are responsible for receiving email messages on its behalf. This mechanism enables flexibility in email configurations.

For example, let's say the primary mail server designated to receive emails for the "bytebytego.com" domain is "mail.bytebytego.com". You might want to have specific email addresses, such as those ending in "@sales.bytebytego.com" or "@support.bytebytego.com", yet still direct all incoming mail for these addresses to the "mail.bytebytego.com" server. By configuring MX records appropriately, emails sent to any of these addresses will route to the designated primary mail server.

IP-to-host address mapping

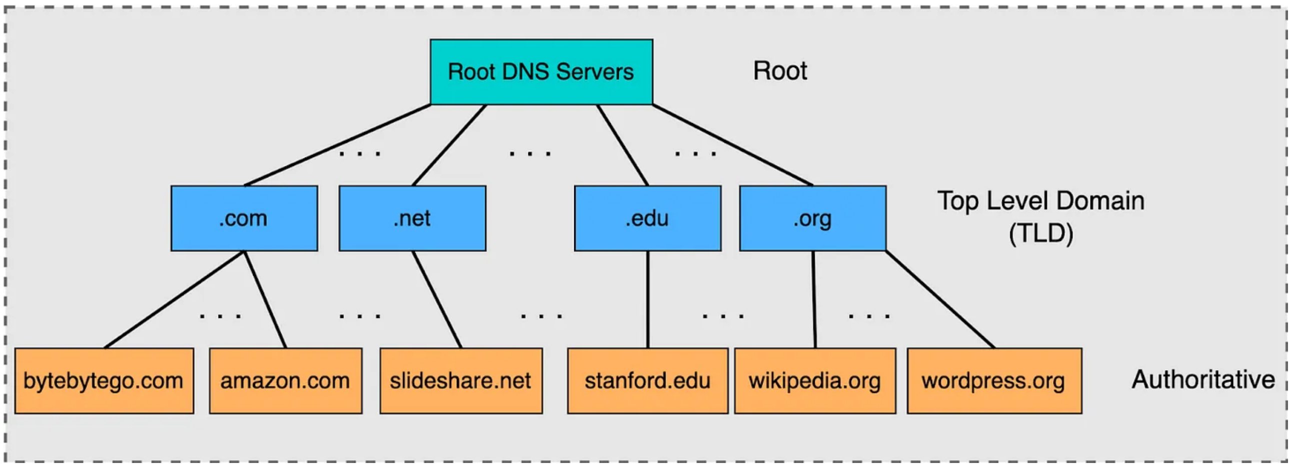
While DNS is primarily used for translating domain names to IP addresses, it can also perform reverse lookups, translating IP addresses back to domain names. This is useful for security and logging purposes.

Load balancing

DNS can distribute incoming network traffic across multiple servers by returning different IP addresses in response to the same domain name query. This helps balance the load and improve the performance and reliability of online services.

DNS Hierarchy

DNS operates as a distributed hierarchical database. The following illustration shows a high-level view of the DNS hierarchy.



Root DNS Servers

At the top of the DNS hierarchy are the root servers. The root servers are contacted when a server is not actually able to resolve a name. You can think of it as a first point of contact to get the resolution started.

Root servers act like the central nervous system of the internet, and as such, security is going to be very important. Much of the infrastructure associated with the root servers is the responsibility of ICANN (Internet Corporation for Assigned Names and Numbers). There are 13 logical root servers around the world, but each of these logical root servers is actually replicated, so corresponding to these 13 logical servers are actually close to a thousand physical servers around the world.

Top-Level Domain (TLD) DNS servers

Moving down a level from the root servers, we find the TLD (Top-Level Domain) servers and each of the servers in the TLD layer is responsible for resolving one of the addresses that have an ending like **.com**, **.edu**, **.net**, and **.org**.

The Internet Corporation for Assigned Names and Numbers (ICANN) has authority over all TLDs used on the Internet, and it delegates the responsibility of these TLDs to various organizations. For individuals or entities looking to register a new domain under these TLDs, they typically approach Domain Name Registrars, which are accredited entities interfacing with the registries to handle the registration process.

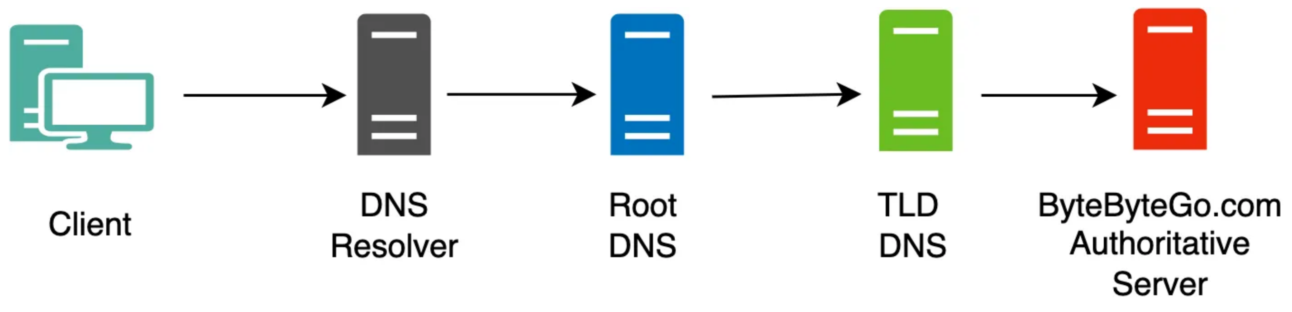
Authoritative DNS servers

Authoritative servers are the definitive source for domain name resolutions within their specific domain. They store the actual name-to-IP address mappings for a given domain. While various caching mechanisms exist across the internet to speed up domain name resolutions, it's the authoritative servers that provide the correct and final answer when queried. Domain owners or administrators configure their domain's records, but the actual infrastructure—the DNS servers—is often maintained and operated by DNS hosting providers or registrars like Cloudflare, Namecheap, GoDaddy, and others.

Recursive DNS Servers (Resolver)

Recursive servers handle DNS queries from client devices like computers and smartphones. When a device wants to resolve a domain name, it contacts these servers. Acting on behalf of the client, recursive servers traverse the DNS hierarchy, consulting various DNS servers to determine the IP address associated with a domain name. Once they obtain the answer, they return it to the client. For efficiency, recursive servers often cache responses to avoid repeatedly querying the same information.

Check out the illustration below, it shows the placement of some of the main DNS servers inside the pipeline of a DNS query.



How DNS Resolution Works

There are two main methods of query resolution in DNS:

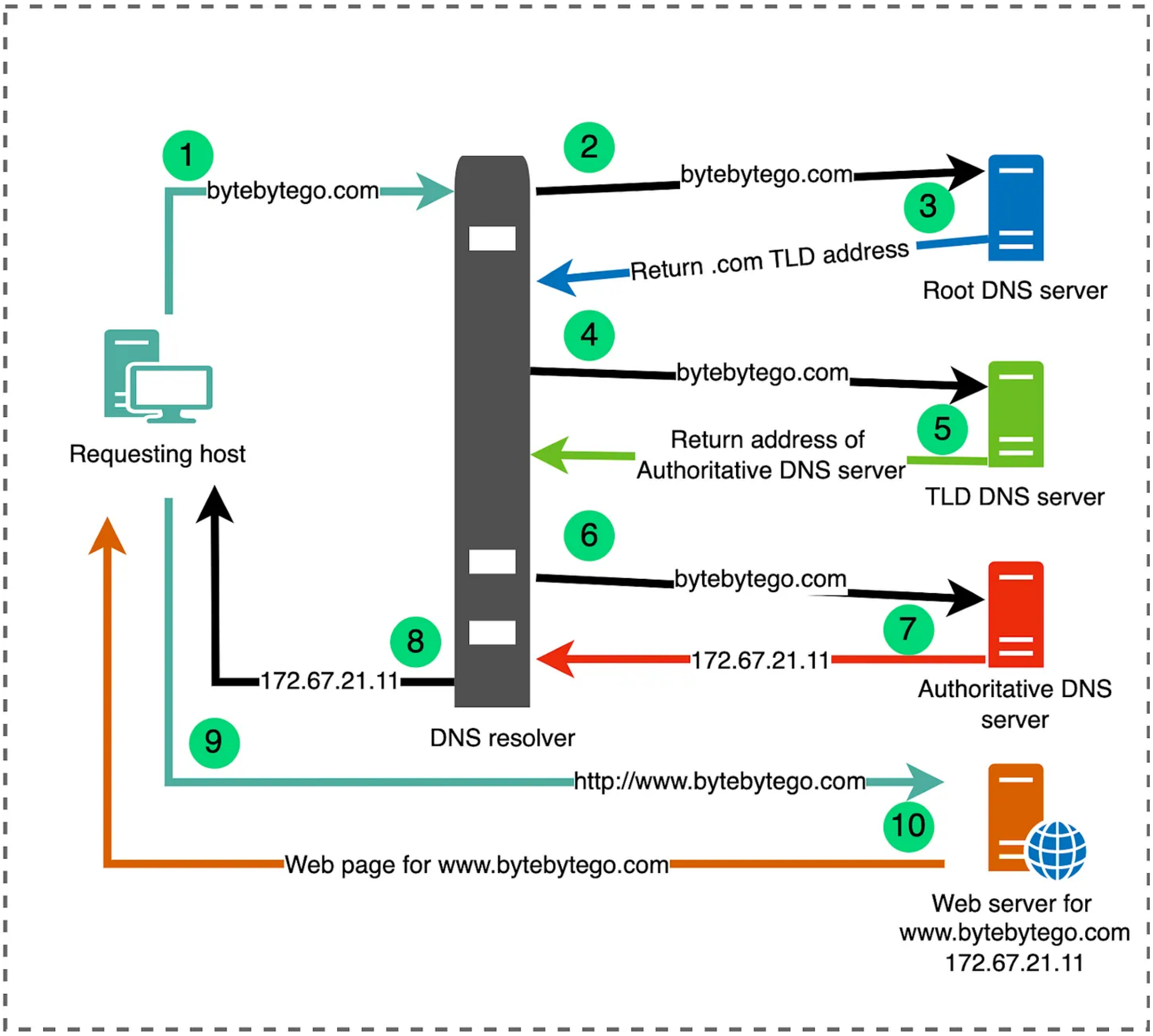
- Iterative query resolution
- Recursive query resolution

These methods define how DNS servers interact with each other to find the IP address associated with a given domain name. Here's a brief overview of each method.

Iterative Query Resolution

In iterative query resolution, the DNS server receiving the query provides referrals to the querying server, guiding it through the DNS hierarchy. The querying server actively participates in the process by sending subsequent queries based on the referrals received.

Let's try to understand the workings of iterative query resolution with the help of an example shown in the illustration below.



Suppose the requesting host is making a request to resolve the name **bytebytego.com**. Here's how this is going to unfold:

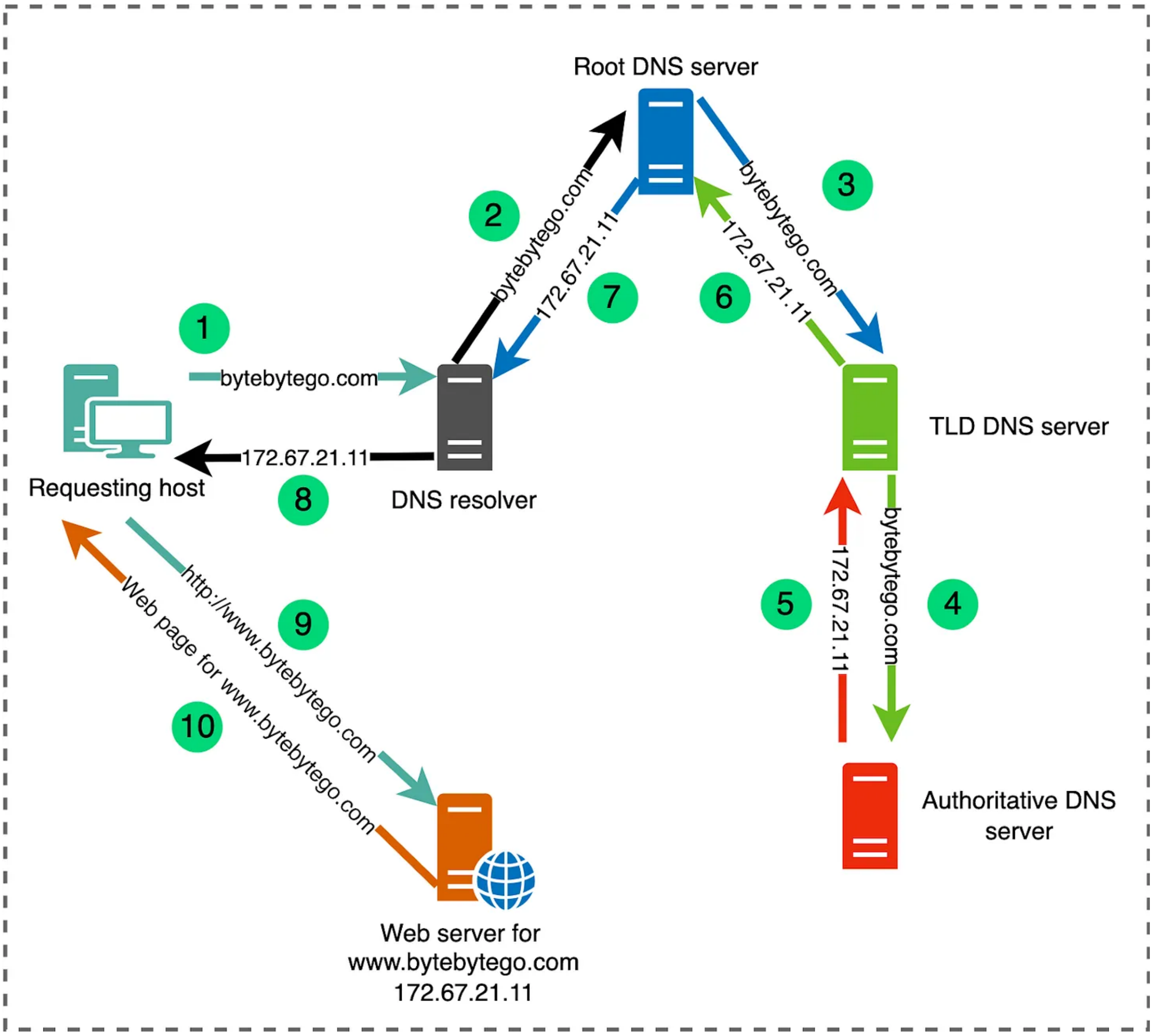
1. The host first sends a query message to the local DNS resolver. The query message contains the hostname to be translated **bytebytego.com**
2. If the resolver doesn't have the IP address in its cache, it sends a query to a root DNS server.
3. The root DNS server, recognizing the **.com** suffix, provides a referral to TLD servers responsible for **.com**
4. The resolver then sends a new query to one of these TLD servers.
5. The TLD server provides a referral to the authoritative DNS server responsible for **bytebytego.com**.
6. The resolver sends another query to the authoritative DNS server.

7. The authoritative DNS server responds with the IP address of **bytebytego.com**
8. The DNS resolver caches this IP address and then returns it to the requesting host.
9. Now, the requesting host makes an HTTP request to the IP address of the http://www.bytebytego.com web server.
10. The web server returns the webpage for www.bytebytego.com

Recursive Query Resolution

In recursive query resolution, the DNS server receiving the query takes on the responsibility of finding the IP address on behalf of the client. It may itself use iterative queries to navigate through the DNS hierarchy until it reaches the authoritative DNS server for the requested domain.

Let's use the previous example and try to resolve the IP address of bytebytego.com using the recursive query resolution as shown in the illustration below:



We have summarized the steps shown in the above illustration as follows:

- The client sends a query to the local DNS resolver.
- The resolver checks its cache for the IP address. If not found, it conducts the necessary queries—possibly iterative ones—interacting with root DNS servers, then TLD DNS servers, and finally authoritative DNS servers until it obtains the IP address.
- Once found, the resolver caches the IP address and returns it to the client.
- Subsequent queries for the same domain can be answered directly from the resolver's cache.

Both iterative and recursive query resolution methods are vital to the functioning of the DNS. Recursive queries simplify the process for clients, with the resolver managing the hierarchical queries and caching. Iterative queries involve the querying server actively following referrals to find the IP address.

Note: In practice, most client devices send queries to recursive DNS resolvers, which often use a mix of recursive and iterative queries behind the scenes to fetch and cache results, ensuring efficient operation of the global DNS system.

Caching in DNS

DNS resolution involves several steps, and to expedite repeated resolutions of the same domain names, the system leverages caching. Caching recently resolved DNS query results can drastically reduce the number of repeated queries to authoritative DNS servers.

By caching DNS records, the efficiency and speed of DNS resolution is significantly improved. It reduces the need for recursive servers to navigate the entire DNS hierarchy every time a domain is queried.

Where does DNS caching occur?

DNS records are cached at various points in the resolution process. The duration for which a record is cached is determined by its time-to-live (TTL) value.

At the local machine: Devices like computers and smartphones have local DNS caches. When you visit a website, your device's DNS resolver stores the resolved IP address along with the associated domain name in its cache. This way, if you visit the same website again or access resources on the same domain, the DNS resolution process is faster.

At the recursive DNS server: Recursive DNS servers, often maintained by ISPs and other organizations, cache DNS records. This benefits multiple users within the same network accessing the same domain, as the cached record can be provided without querying the entire DNS hierarchy again.

At the authoritative DNS server: While it might seem counterintuitive, even authoritative DNS servers (which hold the official domain records) use caching. This helps in efficiently handling high query volumes and reduces redundant processing.

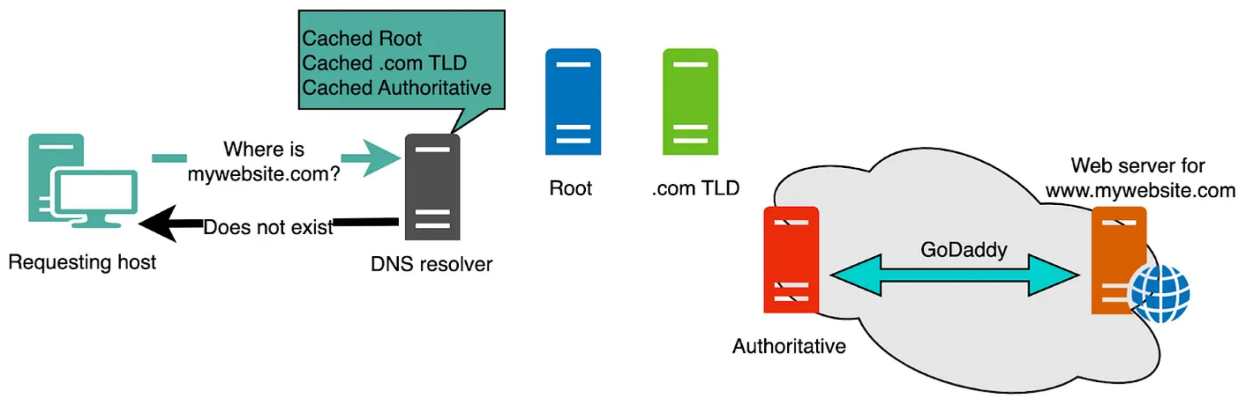
Note: Cached DNS records can become outdated if the original record changes. However, outdated cache entries are not a major concern in the DNS system. They will be automatically removed when their TTL expires.

DNS propagation

After registering or updating a domain, there might be a delay before the domain is accessible universally. This delay is called DNS Propagation.

Let's try to understand this concept with the help of an example.

Imagine you've registered a domain, mywebsite.com, with GoDaddy, a popular domain registrar. If you try to access this domain immediately after registration, you might encounter issues.



Despite updating the record at the authoritative server, many recursive DNS servers and local caches around the world might still have the old (or non-existent) record cached. Until these cached records expire (as per their TTL) and are replaced with the updated record, users might face access issues. This period of inconsistency, where different parts of the world might see different records, is DNS Propagation.

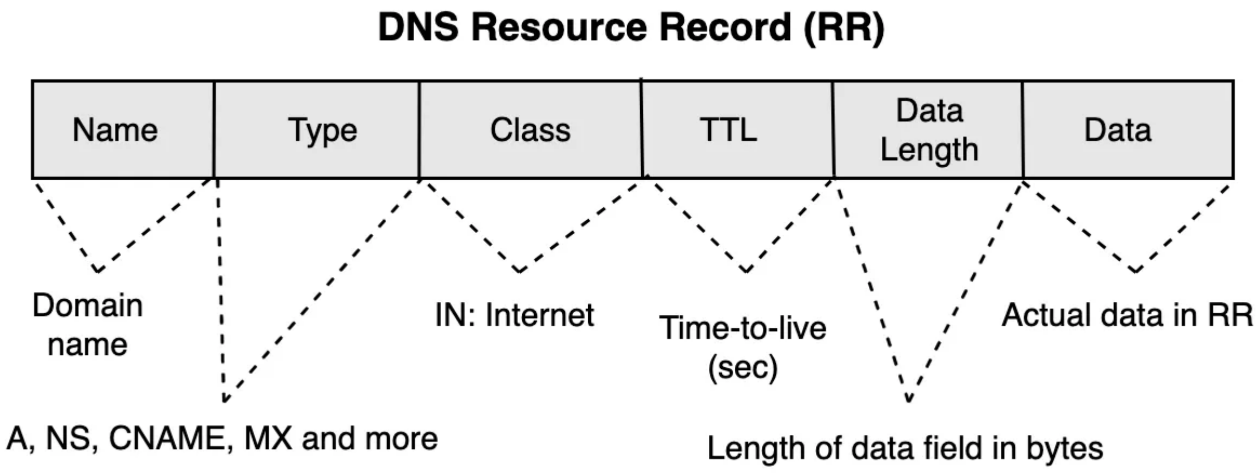
Typically, most DNS propagations complete within 24-48 hours, but the commonly cited upper limit is 72 hours. However, it's essential to set reasonable TTL values to ensure rapid propagation when changes are made.

DNS Resource Records (RRs)

Now, let's talk about how data is being stored in DNS databases.

Data in DNS is stored in the form of Resource Records (RRs). When a device queries a DNS server about a domain name, the server provides the relevant RRs, helping the device access the desired resource on the internet.

Here is the structure of a DNS resource record:



Most of the fields in the RR are self-explanatory. However, we would like to discuss the **Type** field in more detail.

There are different types of DNS resource records, but here we discuss four common RR types in DNS:

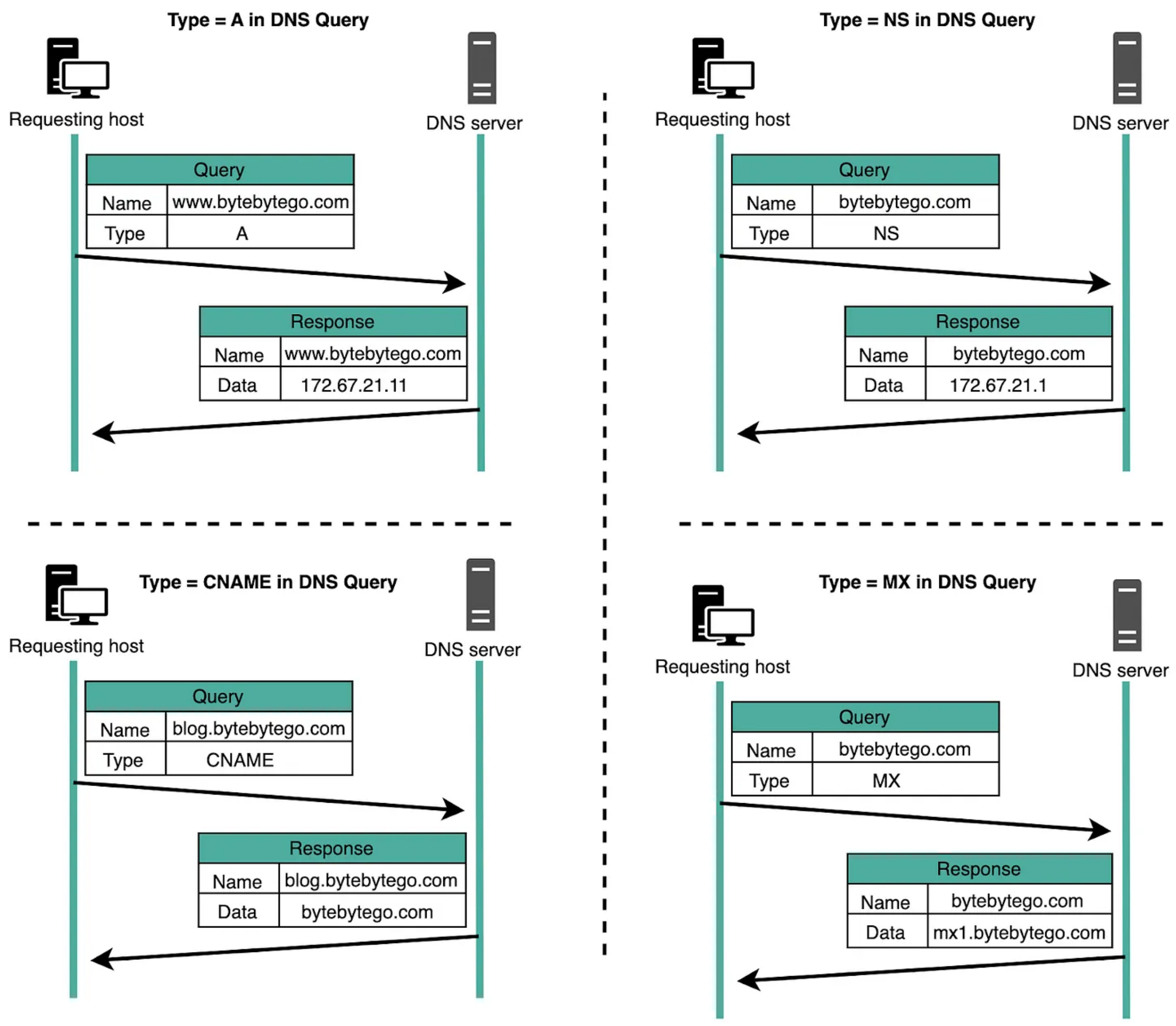
Type A record (Address record): Maps a domain name to an IPv4 address. It's used to translate human-readable domain names into numerical IP addresses as shown below.

Type **NS** record (Name Server record): Specifies the authoritative name servers for a domain. These name servers hold the official DNS records for the domain.

Type **CNAME** record (Canonical Name record): Creates an alias for a domain name, pointing it to another domain's canonical (real) name.

Type **MX** record (Mail Exchange record): Specifies the mail servers responsible for receiving email messages for a domain.

The below illustration shows these four types when used in a DNS query and their corresponding DNS responses.



Dynamic DNS

An essential extension of DNS is the Dynamic DNS (DDNS). This service allows users to automatically update the mapping between domain names and IP addresses whenever the IP address changes. The primary use-case for Dynamic DNS is to support hosts that have dynamic IP addresses, like those assigned by many residential ISPs. DDNS is invaluable for individuals and small businesses wanting to host services, websites, or devices on networks with dynamic IP addresses. It ensures consistent remote access by providing a stable domain name that always points to the current IP address, even if that address changes frequently.

DNS and Anycast

Anycast is a network addressing and routing methodology where multiple servers share the same IP address. When a client sends a request to an Anycast IP address, the network determines the closest server (in terms of routing efficiency) to serve that request. Using Anycast in DNS provides benefits such as faster response times due to reduced latency, better load distribution among servers, and enhanced redundancy. If one server fails, the network automatically reroutes requests to another available server. Many TLD name servers and major DNS service providers leverage Anycast to improve the performance, reliability, and resilience of their services.

DNS Security

Now that you understand what the DNS does, you can see how absolutely critical it is to the functioning of the internet. If the DNS stopped working, it'd be impossible to contact any host unless you knew its IP address, which means practically never. So it's critical the DNS be protected.

Below, we discuss some of the most popular security protocols to protect DNS:

DNSSEC

Domain Name System Security Extensions (DNSSEC) is a set of extensions to DNS that adds an additional layer of security by digitally signing DNS data. It helps in ensuring the authenticity and integrity of DNS responses, thus mitigating DNS spoofing and cache poisoning attacks.

The benefits of DNSSEC include:

- **Data Integrity**: It ensures that DNS data has not been altered in transit, providing data integrity.
- **Authentication**: It allows resolvers to authenticate that the DNS data they receive is from an authoritative source.
- **Trustworthiness**: It builds a chain of trust from the root zone down to individual domains, making it harder for attackers to spoof DNS data.

Note: DNSSEC authenticates and assures data integrity but doesn't encrypt DNS exchanges. For encryption, protocols like DNS over HTTPS (DoH) and DNS over TLS (DoT) are employed, as described next.

DNS-over-HTTPS (DoH)

DNS over HTTPS (DoH) is a protocol that enhances the privacy and security of DNS queries by encrypting them within the HTTPS (Hypertext Transfer Protocol Secure) protocol. With DoH, the DNS resolver uses the HTTPS protocol to send and receive DNS queries and responses.

DNS-over-TLS (DoT)

DNS over TLS (DoT) is an alternative to DoH. Like DNS over HTTPS, it enhances the privacy and security of DNS queries by encrypting them using the Transport Layer Security (TLS) protocol. With DNS over TLS, the DNS resolver establishes an encrypted TLS connection with your device to transmit DNS queries and responses.

Both of these protocols provide similar benefits in terms of protecting DNS traffic from eavesdropping and tampering. We are going to list some of the key benefits of DoH and DoT below:

Privacy: Both DoH and DoT encrypt your DNS queries, preventing third parties, including ISPs, from monitoring and intercepting your browsing history. This helps protect your online privacy.

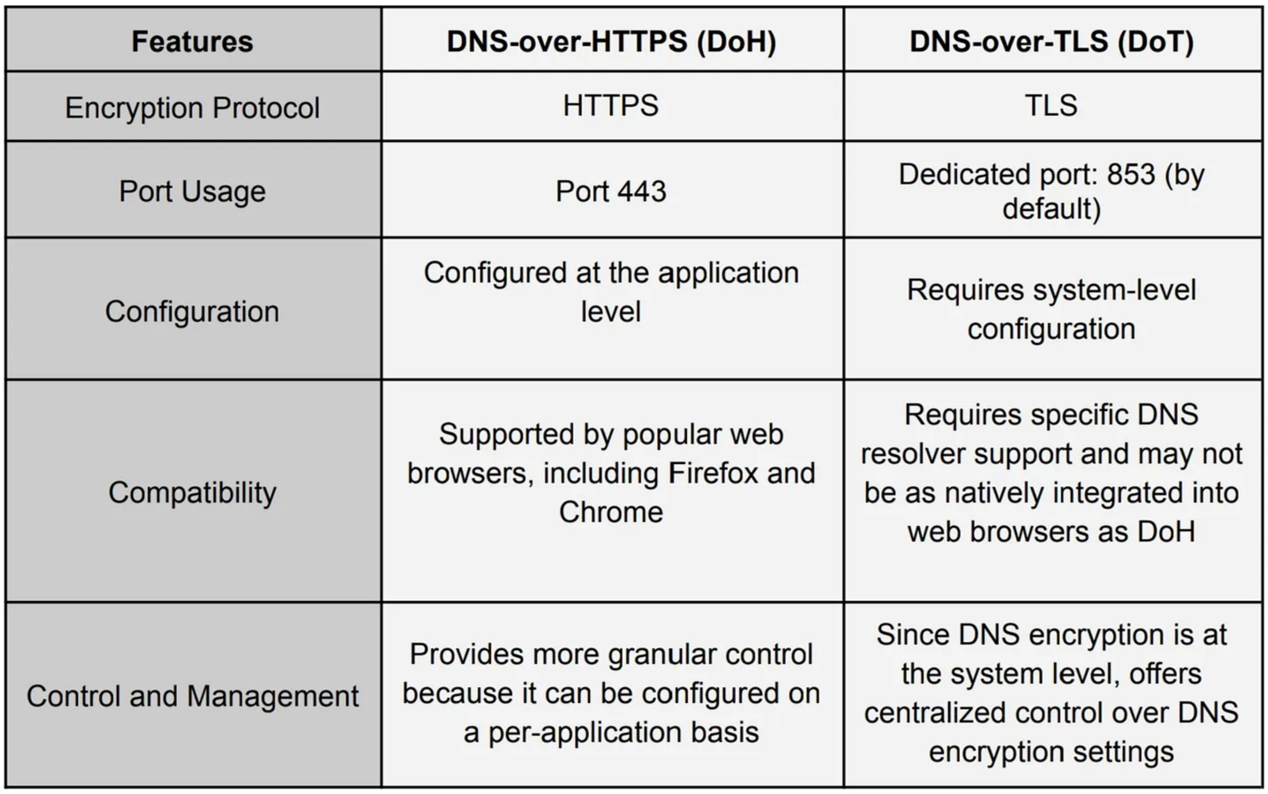
Security: Encrypting DNS queries helps prevent DNS spoofing and man-in-the-middle attacks, where malicious actors could intercept DNS traffic and redirect it to malicious websites.

Censorship Resistance: DoH and DoT can bypass some forms of DNS-based censorship or content filtering, as it makes it harder for authorities to inspect and block specific DNS requests.

Improved Trust: By leveraging the security mechanisms of HTTPS and TLS, DoH and DoT add an additional layer of trust to DNS resolution, making it more resistant to attacks and tampering.

Since both DoH and DoT provide similar benefits, the natural question is, when to use which one?

The answer to this question depends on various factors, and neither is inherently better than the other. Here are the key differences between the two protocols:



Features	DNS-over-HTTPS (DoH)	DNS-over-TLS (DoT)
Encryption Protocol	HTTPS	TLS
Port Usage	Port 443	Dedicated port: 853 (by default)
Configuration	Configured at the application level	Requires system-level configuration
Compatibility	Supported by popular web browsers, including Firefox and Chrome	Requires specific DNS resolver support and may not be as natively integrated into web browsers as DoH
Control and Management	Provides more granular control because it can be configured on a per-application basis	Since DNS encryption is at the system level, offers centralized control over DNS encryption settings

Common Security Threats to DNS

Now that we have explained popular security protocols in DNS, let's briefly describe some of the common security threats to DNS and ways to protect against them.

DNS Cache Poisoning

DNS cache poisoning occurs when malicious DNS data is introduced into the DNS resolver's cache, leading it to direct users to malicious websites. DNS cache poisoning is also known as 'DNS Spoofing'. To protect against this threat, we can:

- Implement DNSSEC (DNS Security Extensions) to add cryptographic signatures to DNS records, ensuring data integrity.
- Use DNS servers that implement query-response validation to prevent caching of malicious data.
- Regularly update and patch DNS servers to fix vulnerabilities that could be exploited for cache poisoning.

Distributed Denial of Service (DDoS) Attacks

A DDoS attack is a malicious attempt to overwhelm and disrupt the normal functioning of DNS servers by flooding them with a massive volume of traffic, rendering them unavailable. To mitigate DDoS attacks:

- Distribute DNS servers across multiple locations and use load balancing to handle traffic spikes.
- Implement rate limiting and traffic filtering rules on your DNS servers.

Man-in-the-Middle (MitM) Attacks

In a MitM attack, an attacker intercepts DNS traffic to redirect users to malicious sites or eavesdrop on their communications. To protect against MitM attacks:

- Use DNSSEC to validate the authenticity of DNS responses.
- Implement DNS over HTTPS (DoH) or DNS over TLS (DoT) to encrypt DNS traffic.
- Monitor network traffic for unusual DNS resolution patterns.

DNS Hijacking

DNS hijacking occurs when attackers gain control of a domain's DNS settings and redirect traffic to malicious servers. Protect against DNS hijacking by:

- Enabling two-factor authentication (2FA) for domain registrar accounts.
- Monitoring DNS records for unauthorized changes.

DNS security requires a multi-layered approach, including a combination of encryption, access controls, monitoring, and patch management. Regularly review and update your DNS security

practices to adapt to evolving threats.

DNS for IPv6

DNS for IPv6, often referred to as DNS6, serves a similar fundamental purpose as DNS for IPv4, but it is adapted to the IPv6 addressing scheme and offers some improvements and changes to accommodate the larger address space and other features of IPv6.

Here's how DNS for IPv6 differs from DNS for IPv4:

Address Format: IPv4 uses 32-bit addresses, while IPv6 uses 128-bit addresses. In DNS for IPv6, these longer addresses are represented using hexadecimal notation, separated by colons. For instance, an IPv6 address might look like 2001:0db8:85a3:1111:2222:3333:4444:5555.

AAAA Records: In DNS for IPv6, the primary record for mapping domain names to IPv6 addresses is the AAAA (pronounced "quad-A") record. This is analogous to the A record in DNS for IPv4. When you query a DNS server for the AAAA record of a domain, it returns the corresponding IPv6 address.

Reverse DNS: Just like in DNS for IPv4, reverse DNS lookup in DNS for IPv6 maps IPv6 addresses back to domain names. For example, if you have an IPv6 address and want to find the associated domain name, you can perform a reverse DNS lookup using a PTR record.

IPv6-Only DNS Servers: While DNS servers can support both IPv4 and IPv6 (dual-stack), there are also DNS servers designed to work exclusively with IPv6 without any reliance on IPv4. This is especially important in IPv6-only network environments.

DNSSEC for IPv6: DNSSEC (Domain Name System Security Extensions) is a security feature that is fully compatible with both DNS for IPv4 and DNS for IPv6, providing security enhancements for domain name resolution in both IP versions.

IPv6 Transport: DNS for IPv6 can use IPv6 transport exclusively for communication between DNS clients and servers, reducing the reliance on IPv4 for DNS resolution.

Summary

This wraps up our discussion of the DNS, which is a critical name resolution service for the internet. We talked about crucial functions of DNS, including mapping domain names to IP addresses, managing host aliases, mail server aliases, and reverse DNS lookups. The DNS hierarchy involves root servers, top-level domain (TLD) servers, authoritative servers, and recursive DNS servers.

Then, we discussed the two DNS resolution methods, iterative and recursive, which play a vital role in DNS functioning. DNS caching optimizes resolution speed by storing results for a set time. DNS Resource Records (RRs) store data in DNS databases, including various record types like Type A, NS, CNAME, and MX records. Lastly, we presented some of the common threats to DNS and ways to avoid them.

Keep exploring, keep learning.

Reading Material

- DNS Extensions to Support IP Version 6
- RFC 9364 DNS Security Extensions (DNSSEC)

250 Likes · 9 Restacks

2 Comments

Write a comment...

Dinh Sep 29, 2023

Thank you. The subject is very detail. I have one the question. Do you can explain for me or write a post to explain what are the stratergies of DNS when it has to lookup domain name map to ip addresses in the billion of ip address on the internet ?

LIKE REPLY SHARE

1 reply

1 more comment...