Chapter 4: Discovery

Before a reliable and secure channel can be established between two processes, the client process must first discover the IP address of the remote process. The most common way to achieve this is by using the **Domain Name System (DNS)**.

DNS can be described as the phone book of the internet. It is a distributed, hierarchical, and eventually consistent key-value store that maps human-readable hostnames to machine-readable IP addresses.

The DNS Resolution Process

When you enter a URL like `www.example.com` into your browser, a multi-step resolution process begins to find the corresponding IP address.

The process is as follows:

1. **Browser Cache**: The browser first checks its local cache to see if it has resolved this hostname recently. If a valid IP address is found, it is returned immediately.
2. **DNS Resolver**: If the hostname is not in the browser's cache, the request is sent to a **DNS resolver**. This is typically a server hosted by your Internet Service Provider (ISP). The resolver also checks its own local cache first.
3. **Root Name Server**: If the resolver doesn't have the address cached, it sends the query to a **root name server**. The root server doesn't know the final IP address, but it knows who is responsible for the Top-Level Domain (TLD). It responds with the address of the name server for the `.com` TLD.
4. **TLD Name Server**: The resolver then sends a request to the `.com` TLD name server for `example.com`. This server responds with the address of the **authoritative name server** responsible for the `example.com` domain.

5. **Authoritative Name Server**: Finally, the resolver queries the authoritative name server for `www.example.com`. This server holds the actual DNS record and returns the final IP address to the resolver, which then passes it back to the browser.

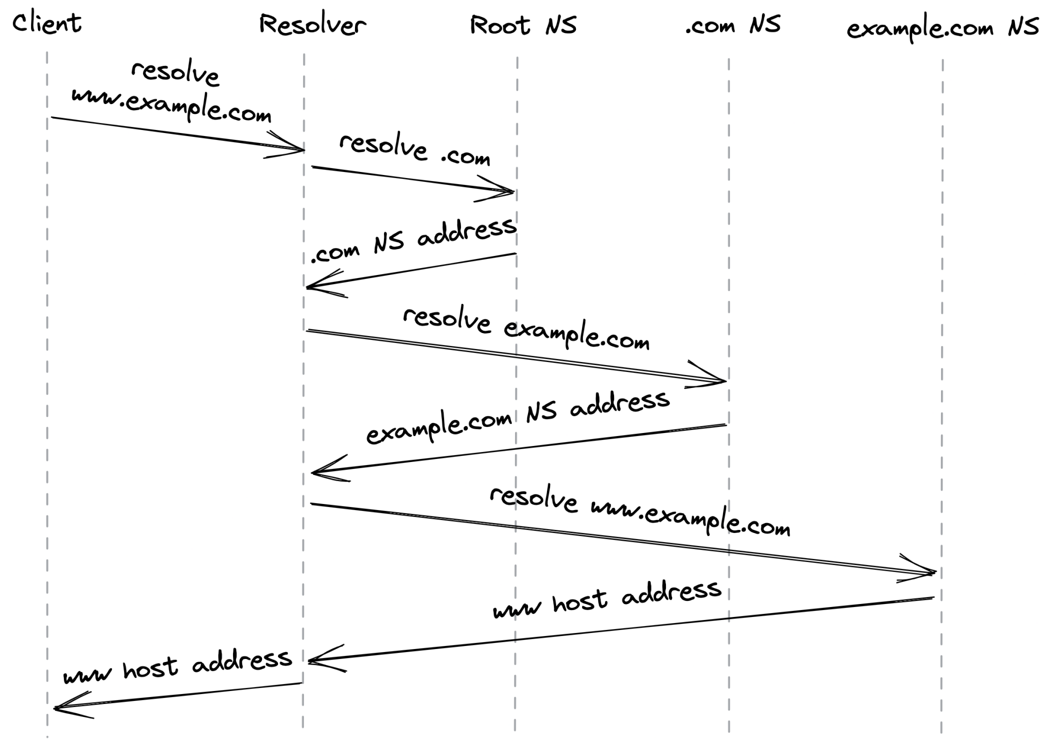

Figure 1: DNS resolution process

DNS Security

- The original DNS protocol sent messages in plain text, primarily over UDP for efficiency.
- This is insecure as it allows anyone to snoop on the transmission.
- The industry has largely moved to secure alternatives, such as **DNS over TLS**, to protect user privacy.

Caching and Time to Live (TTL)

The DNS resolution process can involve several round trips, which would be slow if every request had to go through the full lookup. To speed things up and reduce the load on name servers, DNS relies heavily on caching.

- **Caching Layers**: Caching occurs at multiple levels, including the browser, the operating system, and the DNS resolver.
- **Time to Live (TTL)**: Every DNS record has a **Time to Live (TTL)** value. This value tells caches how long they are allowed to store the record before it should be considered expired.

The TTL Trade-off

Setting the right TTL value requires making a trade-off:

- **Long TTL**: If you use a long TTL, any changes made to the DNS record will take a long time to propagate, as many clients will be using the old, cached value.
- **Short TTL**: If you use a very short TTL, you increase the load on the name servers because clients have to resolve the hostname more often. This can also increase the average response time for users. Additionally, a short TTL increases the number of clients impacted if your name server becomes unavailable.

  Note: Clients do not always strictly enforce the TTL. It's possible for a small number of clients to continue trying to connect to an old IP address long after the TTL has expired.

DNS as a Single Point of Failure

DNS can easily become a single point of failure. If your authoritative name server goes down and clients' caches expire, they will be unable to resolve your application's IP address. This means no one can connect to your service, which can lead to massive outages.

This leads to an interesting design principle known as **static stability**: a system should continue to function even when a dependency is impaired. In the case of DNS, it would be more robust for caches to serve a stale (expired) entry when they can't reach a name server, rather than failing the request entirely, since DNS entries change infrequently.