Introduction to a CDN

Learn about CDNs, and formalize the requirements for a CDN design.

We'll cover the following

Proposed solution

The solution to all the problems discussed in the previous lesson is the **content delivery network** (CDN). A CDN is a group of geographically distributed proxy servers. A **proxy server** is an intermediate server between a client and the origin server. The proxy servers are placed on the network edge. As the network edge is close to the end users, the placement of proxy servers helps quickly deliver the content to the end users by reducing latency and saving bandwidth. A CDN has added intelligence on top of being a simple proxy server.

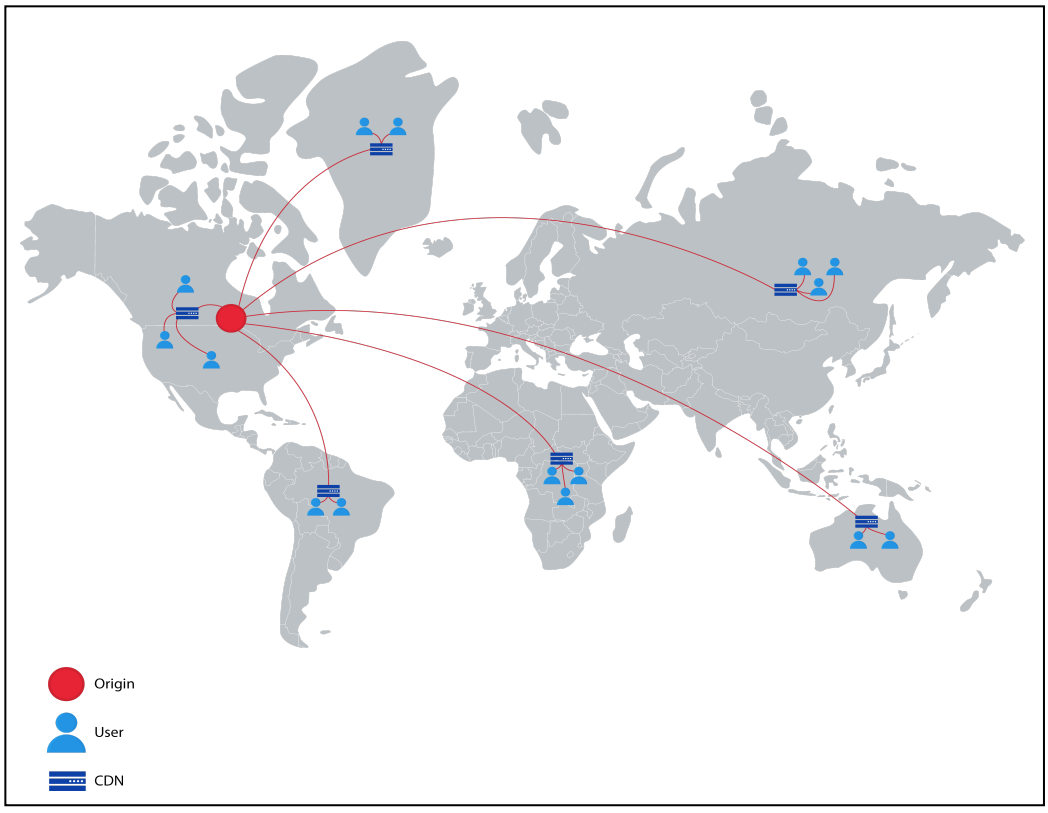
We can bring data close to the user by placing a small data center near the user and storing copies of the data there. CDN mainly stores two types of data: static and dynamic. A CDN primarily targets propagation delay by bringing the data closer to its users. CDN providers make the extra effort to have sufficient bandwidth available through the path and bring da closer to the users (possibly within their ISP). They also try to reduce transmission and queuing delays because the ISP presumably has more bandwidth available within the autonomous system.

Let's look at the different ways CDN solves the discussed problems:

- **High latency**: CDN brings the content closer to end users. Therefore, it reduces the physical distance and latency.
- **Data-intensive applications**: Since the path to the data includes only the ISP and the nearby CDN components, there's no issue in serving a large number of users through a few CDN components in a specific area. As shown below, the origin data center will have to provide the data to local CDN components only once, whereas local CDN components can provide data to different users individually. No user will have to download their own copy of data from the origin servers.

> **Note**: Various streaming protocols are used to deliver dynamic content by the CDN providers. For example, CDNsun uses the Real-time Messaging Protocol (RTMP), HTTP Live Streaming (HLS), Real-time Streaming Protocol (RTSP), and many more to deliver dynamic content.

- **Scarcity of data center resources**: A CDN is used to serve popular content. Due to this reason, most of the traffic is handled at the CDN instead of the origin servers. So, different local or distributed CDN components share the load on origin servers.



Dissemination of content to a geographically distributed CDN

> **Note:** A few well-known CDN providers are Akamai, StackPath, Cloudflare, Rackspace, Amazon CloudFront, and Google Cloud CDN.

Point to Ponder

Question

Does a CDN cache all content from the origin server?

Show Answer

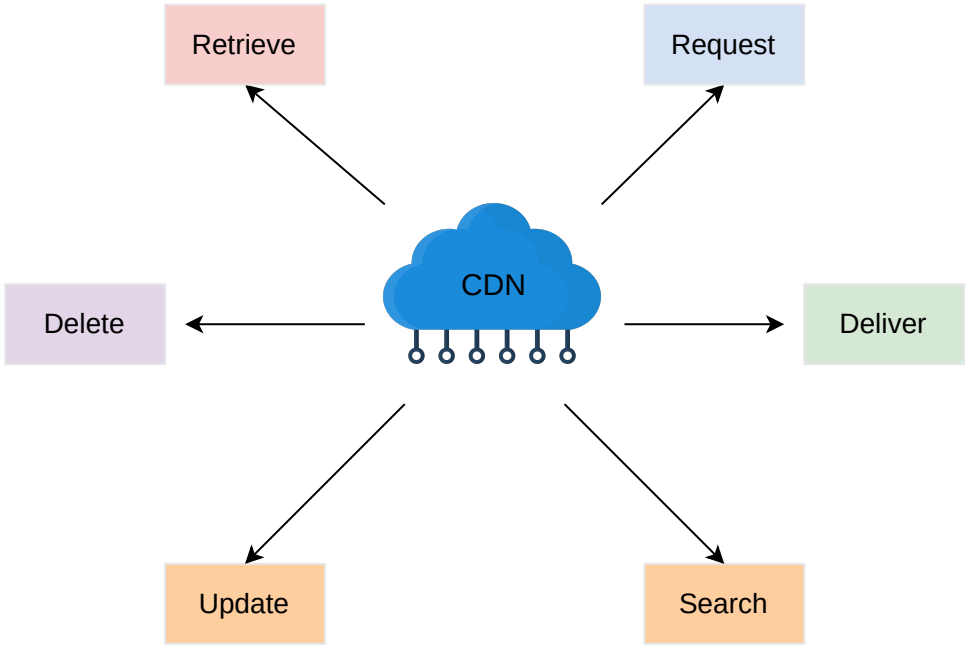
Requirements

Let's look at the functional and non-functional requirements that we expect from a CDN.

Functional requirements

The following functional requirements will be a part of our design:

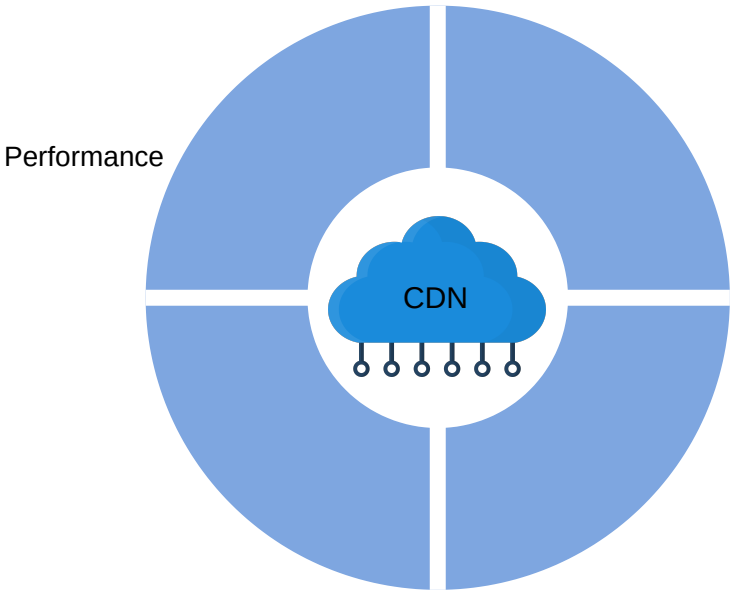
- **Retrieve**: Depending upon the type of CDN models, a CDN should be able to retrieve content from the origin servers. We'll cover CDN models in the coming lesson.
- **Request**: Content delivery from the proxy server is made upon the user's request. CDN proxy servers should be able to respond to each user's request in this regard.
- **Deliver**: In the case of the push model, the origin servers should be able to send the content to the CDN proxy servers.
- **Search**: The CDN should be able to execute a search against a user query for cached otherwise stored content within the CDN infrastructure.
- **Update**: In most cases, content comes from the origin server, but if we run a script in CDN, the CDN should be able to update the content within peer CDN proxy servers a PoP.
- **Delete**: Depending upon the type of content (static or dynamic), it should be possible to delete cached entries from the CDN servers after a certain period.



Functional requirements of a CDN

Non-functional requirements

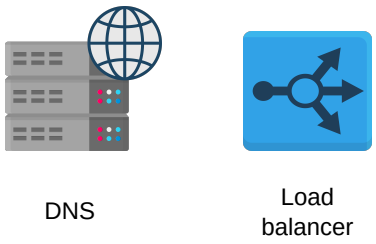
- **Performance**: Minimizing latency is one of the core missions of a CDN. The proposed design should have the minimum possible latency.
- **Availability**: CDNs are expected to be available at all times because of their effectiveness. Availability includes protection against attacks like DDoS.
- **Scalability**: An increasing number of users will request content from CDNs. Our proposed CDN design should be able to scale horizontally as the requirements increase.
- **Reliability and security**: Our CDN design should ensure no single point of failure. Apart from failures, the designed CDN must reliably handle massive traffic loads. Furthermore, CDNs should provide protection to hosted content from various attacks.



Non-functional requirements of CDN

Building blocks we will use

The design of a CDN utilizes the following building blocks:



The building blocks used in CDN design

- **DNS** is the service that maps human-friendly CDN domain names to machine-readable IP addresses. This IP address will take the users to the specified proxy server.
- **Load balancers** distribute millions of requests among the operational proxy servers.