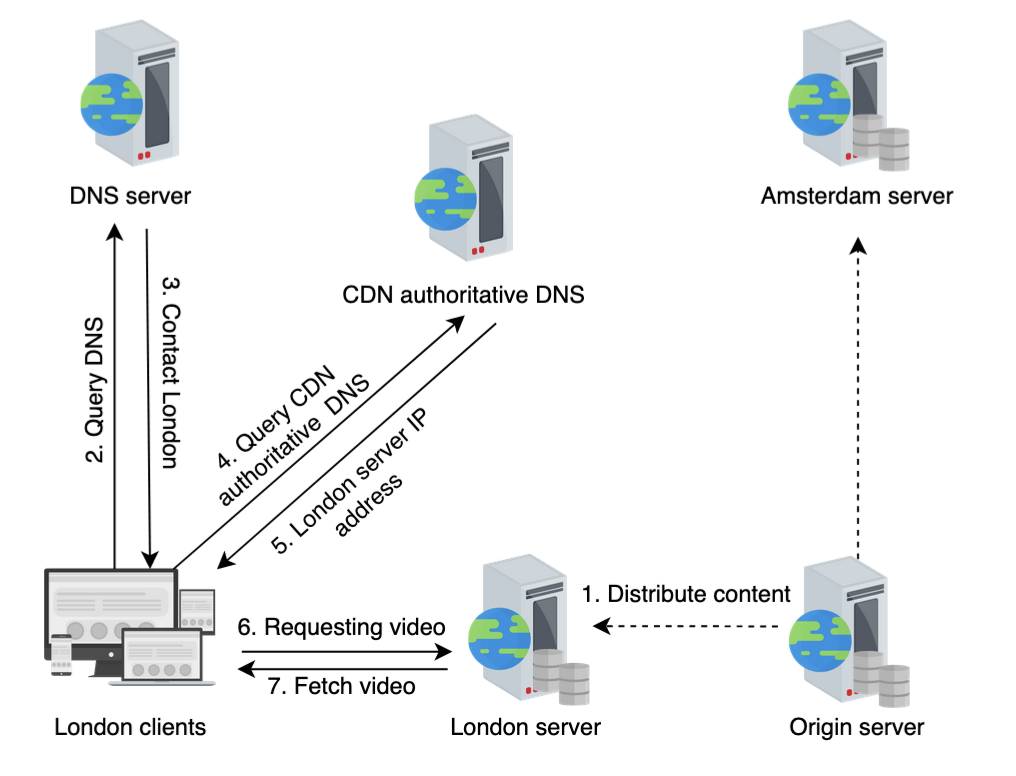
**CDN**

1. **Need:**
   1. **Reduce latency**. Latency – transmission delays (bandwidth), propagation delays(distance), nodal processing delays(CPU on servers), queuing delays (network congestion) . Therefore data transmission at long distances -high latency. Real time applications: <200 ms, VOIP: <150 ms, video streaming: <1 s. Reducing latency by reducing distance.
   2. **Data Intensive Applications:** 
      1. Long distance – packet travels across multiple ISPs, if some low MTU links are encountered, throughput can reduce. This can hurt data intensive applications like streaming apps. Since path to the data only includes the local ISP, there is not much of an issue in serving large number of users of data intensive applications in the specific area.
      2. The problem multiplies because origin servers have to provide same data multiple times to multiple users. With CDN, origin data center provides content to CDNs only once. So users don’t have to download their copy of the data from the origin centers.
   3. **Scarcity of Data center resources:** If data center goes offline, CDNs can act as cache and still serve traffic. Increases availability
2. **Functional Requirements:**
   1. **Retrieve:** Get the content from origin servers – push or pull
   2. **Search:** Search for the cached content within the CDN infra
   3. **Update:** If the content is coming from origin server but if we run a script in CDN, update that content within peer proxy servers in a CDN POP. Typically, each CDN PoP has a large number of cache servers.
   4. **Delete:** Delete the cached content after some time
3. **Non-functional Requirements:**
   1. **Performance:** Reduce latency. Serve data mostly from RAM. Long tail content stored in SSDs of the proxy servers. Still better than a network I/O. Also if one layer doesn’t have content, the next layer of proxy servers can be used (again the parent layer could store the long tail content).
   2. **Availability:** 24/7. Many peer proxy servers in a POP which have redundant data and load balanced. Protection against attacks like DDoS
   3. **Scalability:** Same performance when number of users increase i.e. CDN design should be able to scale horizontally. Scrubber servers to prevent DDoS attacks.
   4. **Security:** Protection to hosted content against attacks.
4. **CDN Components:**

Diagram

Description automatically generated

* **Routing System** receives feedback from various components – where the content is placed, how many requests for a particular content, load for proxy servers, URL namespace for various contents.
  1. One such way of routing requests is **via DNS redirection** which includes DNS load balancing. The client request to resolve the content, for example “url/video” hits the authoritative DNS server which redirects it to the location specific authoritative DNS server for the CDN: fastly.content.london.net. This corresponds to URL record on the DNS. Difference between CNAME and URL record is that with URL record the address also changes (HTTP 301 code is returned) but with CNAME, the recursive resolution continues. The request then reaches the authoritative DNS server of the CDN which provides the IP address of the proxy server.

****

Since the load at proxy servers changes over time, the content provider needs to make appropriate changes in the DNS to make the DNS redirection effective.

* 1. **Anycast:** All the edge servers have the same IP. The client reaches the nearest proxy server using BGP.
* **Scrubber Servers:** Used to prevent against DDoS
* **Proxy Servers:** Store hot content in RAM and cold content in SSD. Provide accounting information to the Management System.
* **Management System:** Resource usage statistics. Stores metrics like latency, downtime, server load etc.
* **Distribution System:** Sends feedback to the routing system i.e. which content is stored on which proxy server. This information is used by the routing system to maybe update the DNS entries, for example.
* Edge proxy server periodically sends accounting information to the management system. Management system then sends feedback to the routing system. Using this feedback, the routing system can determine if certain proxy servers are under a lot of stress and then update the load balancing logic for the proxy servers which can include updating the DNS entries. Think in terms of auto traffic steering at Paypal. The management system could be EdgeMonitor sending feedback to the request routing system (NS1) to update weights for CDNs or taking them out completely.

1. **Caching Strategies:** 
   1. **Push:** Content sent to the proxy servers from the origin server itself. Advantages – No delay as the cached content is already available and the CDN doesn’t need to request a copy of the data. Although from the second request onwards, you don’t see any delay with pull model as well. Disadvantages – More effort from the website owner to create the necessary upload process. Not suitable for dynamic content which changes frequently as this will strain the origin servers to upload the data which may not be needed by the users in first place as the data may soon evolve to a new state.
      1. To be consistent with the origin servers, **leases** are used. Time duration during which origin server agrees to notify CDNs for a change in data. Proxy server sends lease renewal messages when the leases expire.
   2. **Pull:** CDN itself pulls the data from the origin servers. CDN keeps the files for some time and then delete them. Disadvantages – if the data is changed and a previous version is cached, when do you refresh it? So you might see stale data. With push model you have flexibility as you can specify which specific files to cache, with pull model CDN may cache every files on the specified URL. Advantages – suitable for dynamic content as the CDN pulls the content from the origin server as and when required.
      1. To be consistent with the origin servers, CDNs use periodic pulling. May consume unnecessary bandwidth if the data changes infrequently. **Time to Refresh**. Another alternative to TTR is TTL. Each object sent by origin servers has a **time to live** assigned by the origin servers. TTL saves some useless requests from CDNs.
2. **How do you cache dynamic content since it’s changing?**
   1. Dynamic content generated by some scripts running on the proxy server based on parameters like location, time of the day, third party APIs specific to the location. We can cache the content generated using these scripts i.e. dynamic content caching.
   2. ESI(Edge Side Include) Markup Language – What part of web page contents dynamic content and what contains static content. So fetching a full web page is not required and dynamic content is generated at the edge server. Less data transmission between CDN and origin servers.
3. **Multi-Tier CDN Architecture.** Data needs to be distributed from the origin server to the proxy servers. Multi tier CDN architecture reduces the burden of data distribution on the origin server

**Diagram

Description automatically generated**

Multi Tier may also help with the long tail data. The long tail data or less popular data items can be moved to the parent proxy servers,

1. What determines proximity to the proxy server –
   1. **Network Distance** ( function of the length of the network path and bandwidth limits along the network path).
   2. **Request Load**
2. **Deployment of Proxy Servers – where and how many to place.**
   1. **Placement**
      1. **On premise:** Small data centers near to the major ISPs. Google uses split TCP. Persistent TCP connections with long TCP windows from ISP infrastructure to their primary data centers. User’s TCP connection terminates at the ISP infrastructure and forwarded on already established connection. Reduces latency as three way handshake to a far away host is avoided.
      2. **Off premise:** In ISP networks. Companies with less data may choose this. Beneficial to ISPs as well as it cuts the external bandwidth that ISPs pay for.
   2. CDN as a service: Akamai, Fastly etc.
   3. Specialised CDNs: Netflix (Open Connect Appliance - OCA). Reduces cost over time (cost per byte streamed).

Diagram

Description automatically generated