Master of Science in Internetworking

Department of Electrical and Computer Engineering

Project Title:

A Distributed Content Delivery Network Architecture with Advanced Edge Routers

Supervisor:

Juned Noonari

Provided By:

Catheren Salamma Joseph Kollamkalam

Fall 2020- Winter 2021

Table of Contents

Table of Figures

Abstract

Content delivery networks help improve any website's user experience while reducing the distribution cost of the content providers. A CDN is a collection of network elements spanning the Internet, where content is replicated through mirrored Web servers located at the edge of the Internet service provider's network. The number of Internet users is exponentially rising over the last few decades. This demands a better architecture of CDN networks that can proactively respond to the end-user requests efficiently. Implementation or testbed emulation of a CDN architecture requires a tremendous investment. Therefore, all the research and improvement in CDN is implemented and deployed in industry research and development majorly. All the major CDN companies, including Akamai is investing immensely in the research of developing advanced edge routers. This project is a humble attempt to design a distributed CDN architecture with advanced edge servers.

This report explains the various components, architectures, and technologies prevalent in the CDN industry. It also throws some light on the challenges in the design of CDN networks. I have analyzed various research papers that suggested CDN architectures. After an intense literature survey of various proposed CDN architectures, I am proposing an architecture that includes the latest caching technologies that can be used primarily by streaming providers. This project's future scope involves the simulation of this architecture to conduct the performance evaluation and compare it with the other architectures involved.

Acknowledgment

Throughout my project, I have received constant guidance and support.

I would like to extend my gratitude to the University of Alberta for providing me with this opportunity and their support in providing necessary resources during this project.

I would like to thank my mentor *Mr. Juned Noonari* for his constant support and guidance throughout the project, whose expertise was invaluable. Juned, I would like to thank you for your patient support throughout the project.

I would also take this opportunity to thank our director *Prof. Mike Macgregor* for his valuable guidance. There has never been a time he said "No" to any of our requirements.

I would like to thank our program coordinator *Shahnawaz Mir* for his responsiveness to all my queries regarding the project.

Last but not least, I thank almighty God, my parents, friends, and all the professors for being there during this difficult time of the pandemic. It would not have been possible without all your mental support.

Chapter 1 Introduction

According to Cisco Visual Networking Index: Forecast and Trends, 2017–2022, the internet traffic worldwide will increase up to 22% by 2022. These trends are alarming, and various content providers should ensure that the infrastructure is set up for uninterrupted content distribution.

How is a video or file stored somewhere in the world available to our fingertips in a couple of seconds? A well-planned network of caching servers strategically placed over vast geography is responsible for this fantastic technology. This network is called CDN.

Consider an example where a user located in Australia wants content in the servers of Dallas, US. If the content is to be picked up from Dallas during each request, it will create a considerable delay and an increase in the traffic, as shown in the following figure.

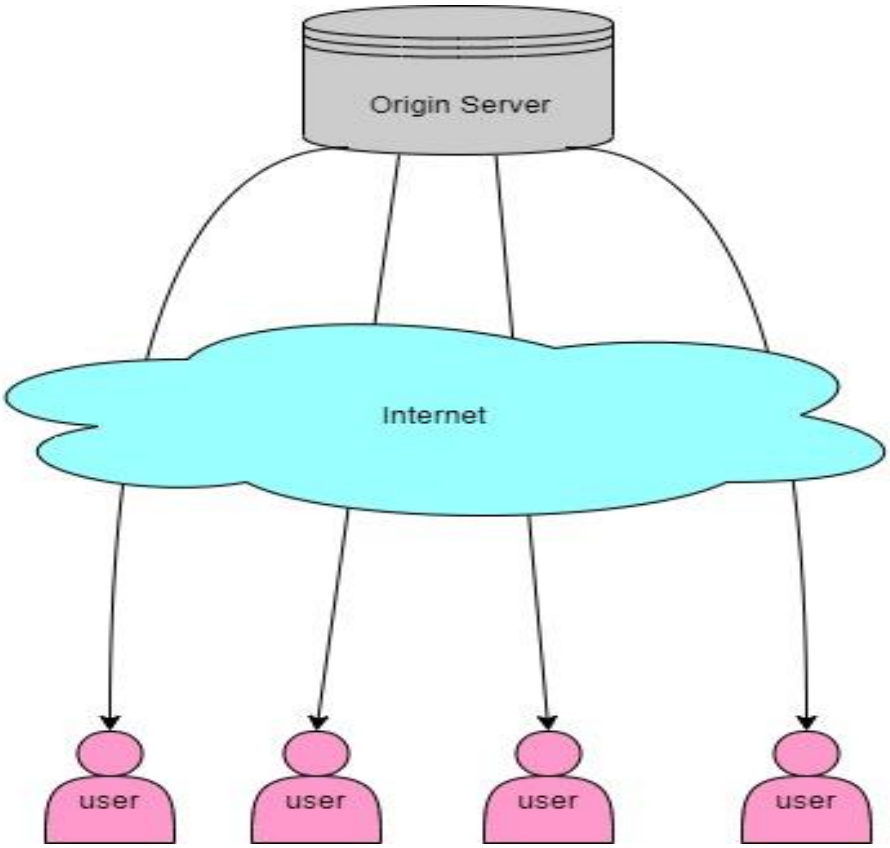


Figure 1. Content delivery without CDN.

Also, there is a probability of server crashes that can occur due to the flash crowding problem during a surge in website visits as well. That is where a CDN comes into the picture. A CDN stores the content as closer to the end-user as possible so that it can reduce the latency and international traffic across the internet, as shown in the following figure. A CDN minimizes the bandwidth usage by caching the content in edge servers.

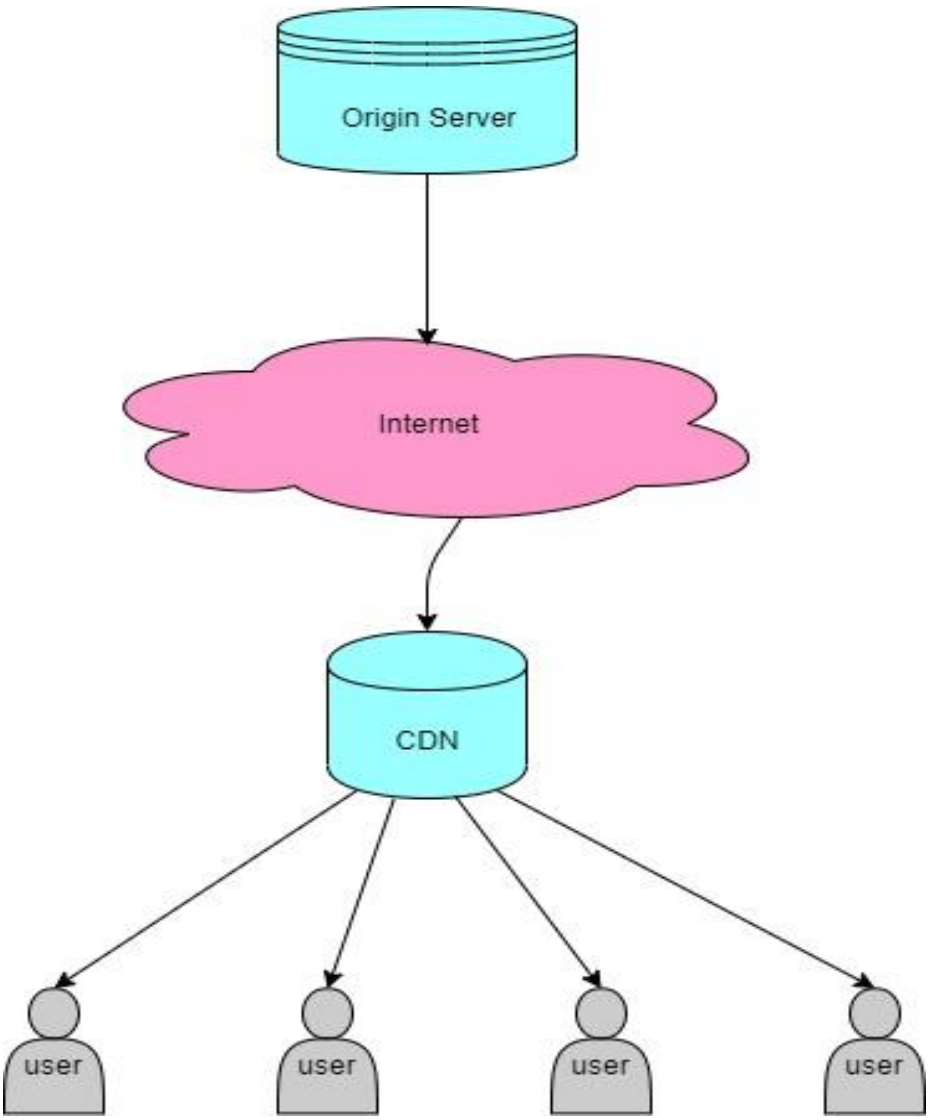


Figure 2. Content delivery with CDN.

These networks cache or replicate the data from the origin server to the edge servers to be easily accessible to the end-users. Hence, A CDN is a vast distributed network of devices and cache servers that helps in delivering the content from the content provider to many cache servers that are geographically spread across the globe. The main intention of a Content Delivery Network is

to reduce the response time of a client request by picking the content from the edge server than the origin server.

The main advantages of CDNs are their scalability, accessibility, increased uptime, improved security, and bandwidth efficiency. Also, CDNs can successfully protect flash crowds and DDOS attacks.

The ever-evolving demands of the user for effortless delivery and quality of service resulted in the need for new architectures to improve the user experience while reducing the cost of investment. Though CDNs were initially generated to deliver files like images, it is prominent nowadays in delivering dynamic content like adaptive video streaming.

There exist various CDN providers that host content owned by third parties that are cached at various geographic locations. These contents are distributed according to the various client requests from the server that can provide the lowest latency and best output.

Hence, designing a complete CDN architecture calls for research on various factors like deciding on the optimal location of the edge server, what content must be cached at each edge server, and to which router the request must be routed for the optimal delivery of the content. The traffic of various networks and availability of the content can be determined to understand the best edge server that can serve any given request.

1.1 Content Delivery at the Edges:

Each client's requests traveling through various subnets created congestion in the internet networks. The content was mirrored to various edge servers, and the content was picked up from these distributed locations to avoid this issue.

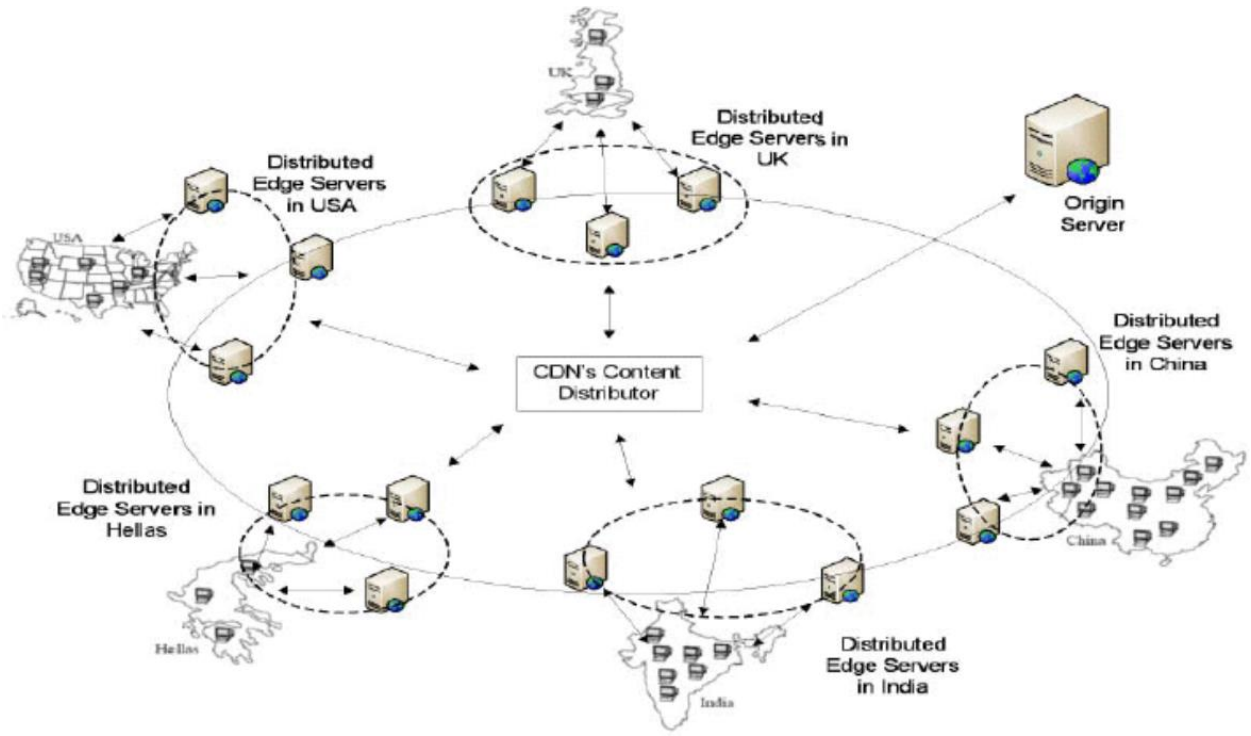


Figure 3. The architecture of content delivery at edges. [1]

All the content required for clients in a particular area should be available locally without having the requests to pass through the various networks. Proxy caching and mirroring can be used to ensure that that content can be delivered as fast as possible to the clients. Nevertheless, this requires an efficient request routing algorithm and cache prediction system for scaling down the delay considerably.

1.2 Evolution of CDNs:

CDNs were first used after a flash-crowd problem that occurred after the 9/11 terrorist attack. Akamai Technologies were released after this incident by MIT. By the end of the 20th Century, CDN technology was widely used by various websites to provide their users with high-quality content. During this time frame, CDNs were mainly used for delivering static content. These first stage static CDNs improved the performance of the websites by caching the static HTML files. These first-gen CDNs were expensive and were not affordable for smaller companies. [2]

The second-generation CDNs included both static and dynamic content that provided a better experience to the users. Though these CDNs were cheaper than the first generation CDNs, the usage of CDNs was still prevalent only among business sectors.

Later, the demand for online video streaming was peaked, during which more commercial CDNs were released. As per the studies in 2005, there was an estimated growth of 40 % for CDN revenue in internet traffic, including online audio and video services. By 2011, Amazon released cloud-based CDNs that enable their various data centers to serve users with their requests over the internet.

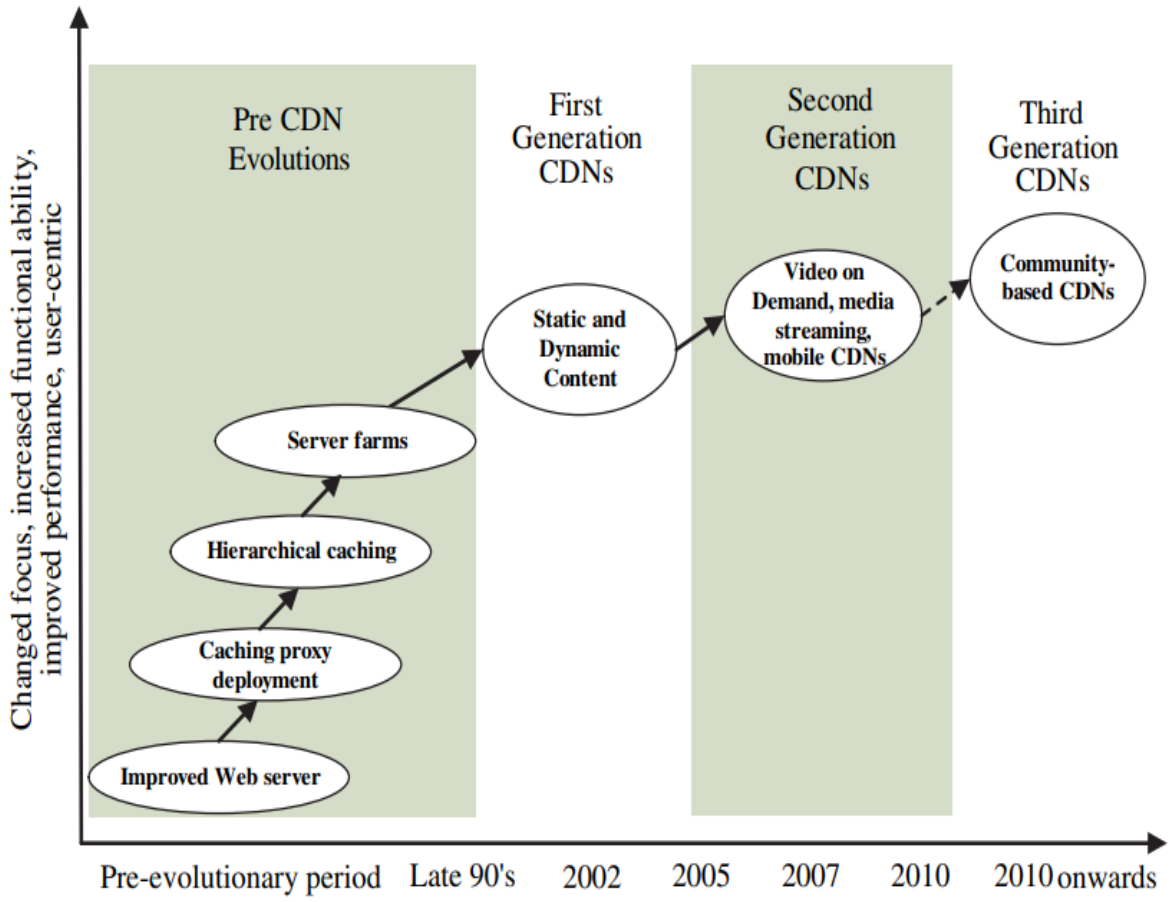


Figure 4. Evolution of CDNs[3].

As per the research, more than 50% of the internet traffic is through CDNs nowadays. In today's world, it is imperative to fulfill the customer requirements for the success of any service provider, especially for video distribution services like YouTube, Netflix, and Amazon Prime, where subscriptions are dependant on the quality of the content provided without latency.

1.3 Services Provided by CDNs:

CDN is an architecture that distributes or delivers the content that is owned by third parties. There are various types of content like static, dynamic, and streaming content that is distributed by CDNs. CDNs are adapting to various techniques, architectures, and technologies to distribute these contents to users effectively efficiently.

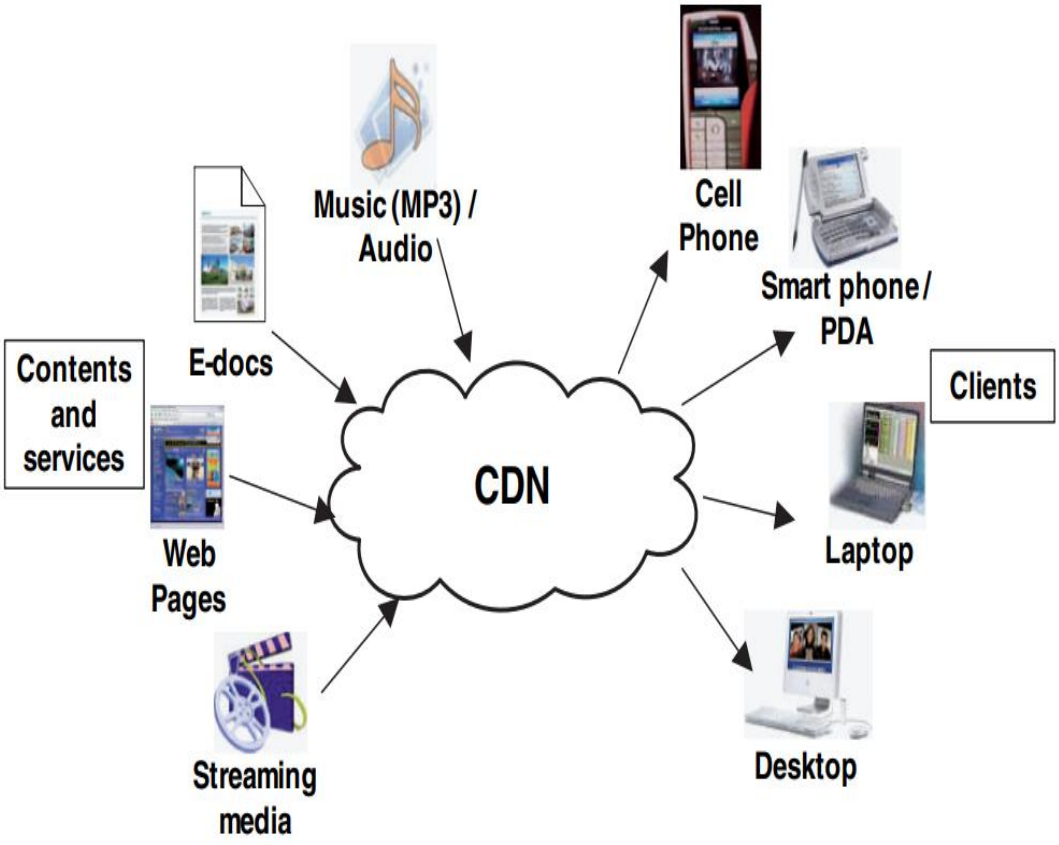


Figure 5. Services provided by a CDN. [2]

- **Static content**: Any file or content that is unlikely to change during time and is displayed the same for various clients from different parts of the world is referred to as static content. It is one of the simplest kinds of content that is distributed by a CDN. It is easy to cache static content on CDN, and conventional CDN architectures would suffice. HTML, JavaScript, CSS files, and PDF are some of the examples of static files. Most of the user-driven websites are dynamic content.

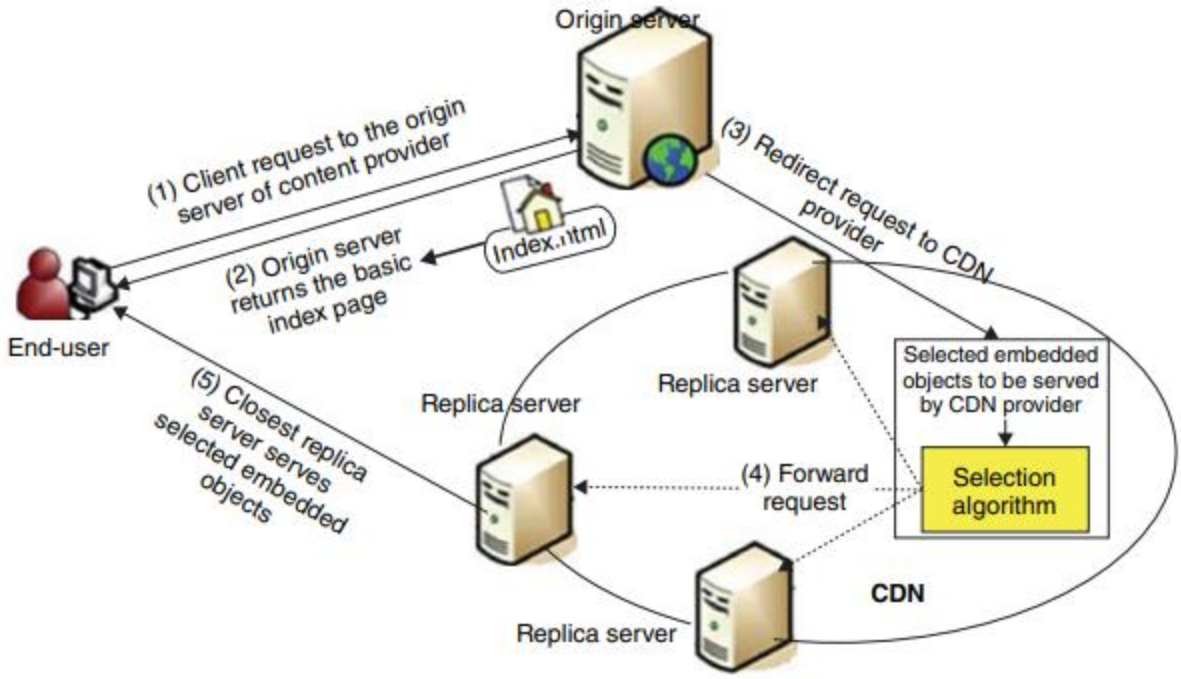


Figure 6. Static Content Delivery[4]

- **Dynamic Content**: Any file or content that varies based on the person, location, the device it is requested from is called dynamic content. Examples of dynamic content are social networking sites and shopping websites. Dynamic content, as the name suggests, varies according to the user sessions' behavior, data, and characteristics.

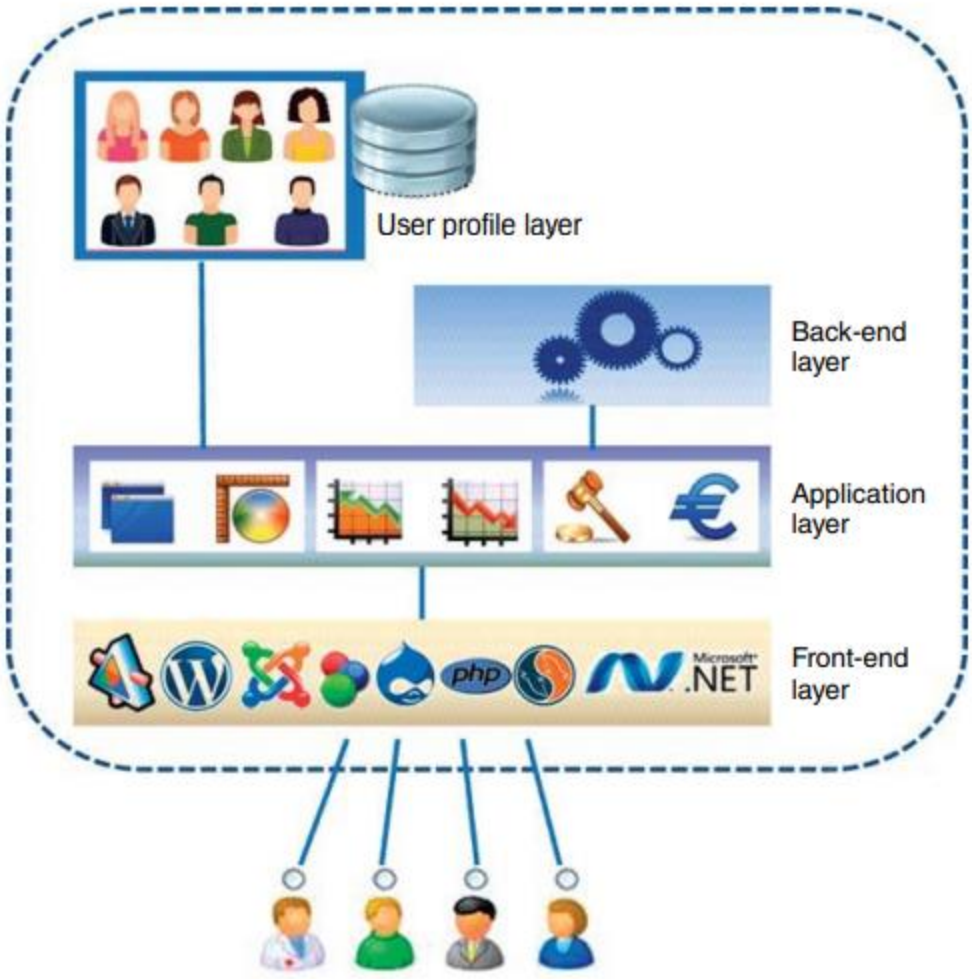


Figure 7. Dynamic Delivery Representation[4].

1.4 Types of CDNs

Based on the retrieval of content, [5][6]

- Push CDN: Any content that is being updated at the origin server is pushed to various edge servers, and this content is cached at the edge before even being requested by the client. The CDN keeps a log to map the content available at various edge servers and thus increases the efficiency. Whenever a client makes a request, it is redirected to the nearest edge server with the requested content. Otherwise, it is redirected to the origin server. The content is deleted when a newer version of the same content is being pushed from the origin server.[7] The content owner should ensure that the latest content is being distributed to the origin server from which it is further distributed. The main advantage

of the CDN is that the internet traffic is meager due to the reduced number of updates occurring in the network. The content at the edge is updated only when there is a change in the existing content.

- Pull CDN: All the client requests are redirected to the nearest edge server through a request routing technique. If it is a cache miss, the request is further directed to either the closest surrogate or the origin server, depending on the architecture of the CDN, and the content is pulled from the corresponding server. Hence, the request routing technique chosen for this CDN is exceptionally significant, and we must ensure that the request routing algorithm is effective.
  Since the content is only pulled as per the need basis, there might be a delay in delivering the content to the client.

Based on the architecture,

- Hierarchical architecture of CDN: The hierarchical architecture of CDN is designed in the form of a tree where the surrogate servers are distributed over various layers. By using caching of the content to various edge servers, the content is distributed to various geographical locations. It reduces the delay in providing content and inter-region internet traffic, also reduces the security risks to the original content. The disadvantage of this architecture is that if the content is not available at a particular server, it can query the further request only to the servers that are present in the layer above or to its parent. Each layer in the hierarchy can create an additional delay to serve the client request. As the tree structure converges towards the origin, there are often bottlenecks and long queues for the service at cache servers. Since the content is served through a linear line of service, it is often stored redundantly through various layers, and this causes loss of memory.
  According to the conventional CDNs, there are three main components as End Users, CDN providers, and Origin Server. The Origin Server stores the content and owns the copyright of the content stored. The CDN provider owns the cache servers that store the copies of the content in various geographical locations. The end-users request the content from the origin server using the internet as a platform.

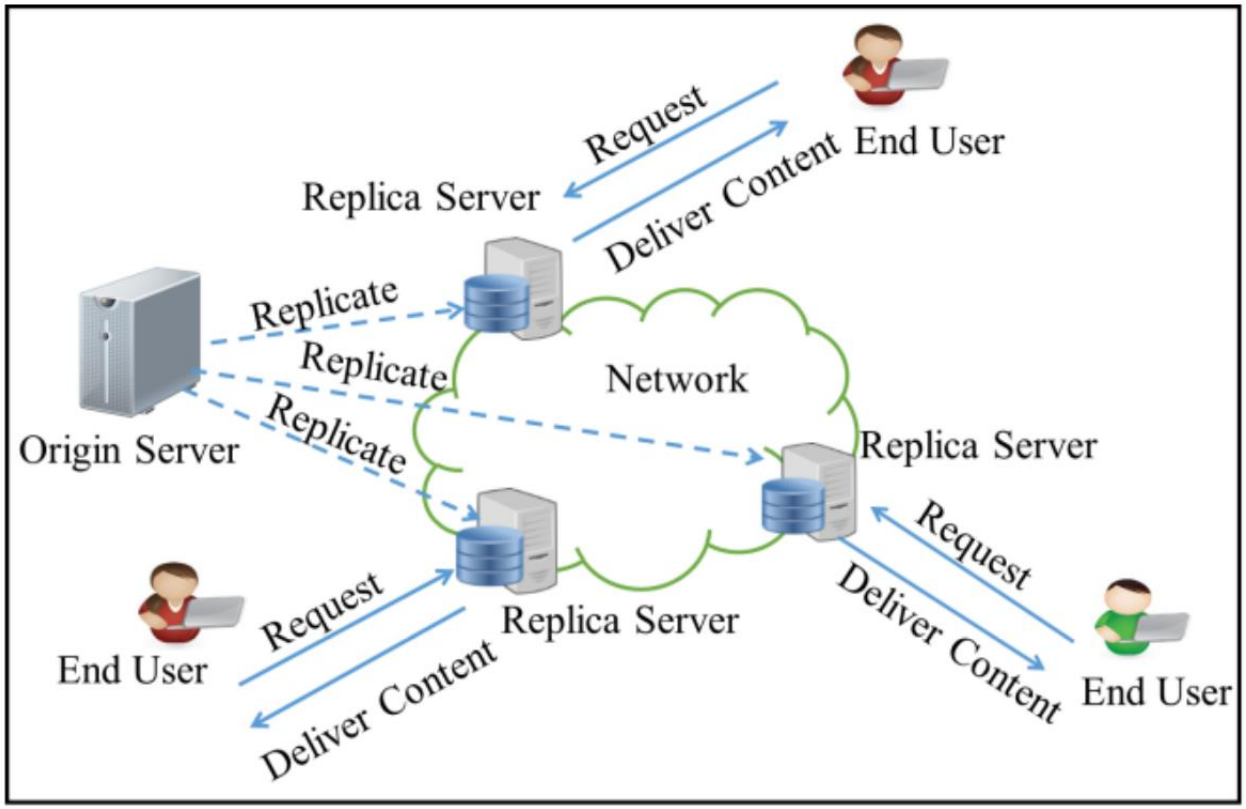


Figure 8. Hierarchical architecture of CDN[8].

- Distributed CDN: The above-mentioned disadvantages of the hierarchical CDN resulted in the development of the distributed CDN. In this architecture, caches at the higher level only maintain the metadata content that is stored by the lower caches. Rather than following a hierarchical design, the user requests are redirected to the nearest edge server. It is further redirected to the cache server that in turn checks if the content is available in any of the servers that are present beneath it. This way, sibling sharing is also conducted in the distributed architecture of CDN. The architecture consists of a local server that maintains a log of all the content that is available at the surrogate servers under its network region. This local server redirects the client request to the best surrogate server that can serve the request. Hash functions are used to map the requests to the corresponding content.[9]

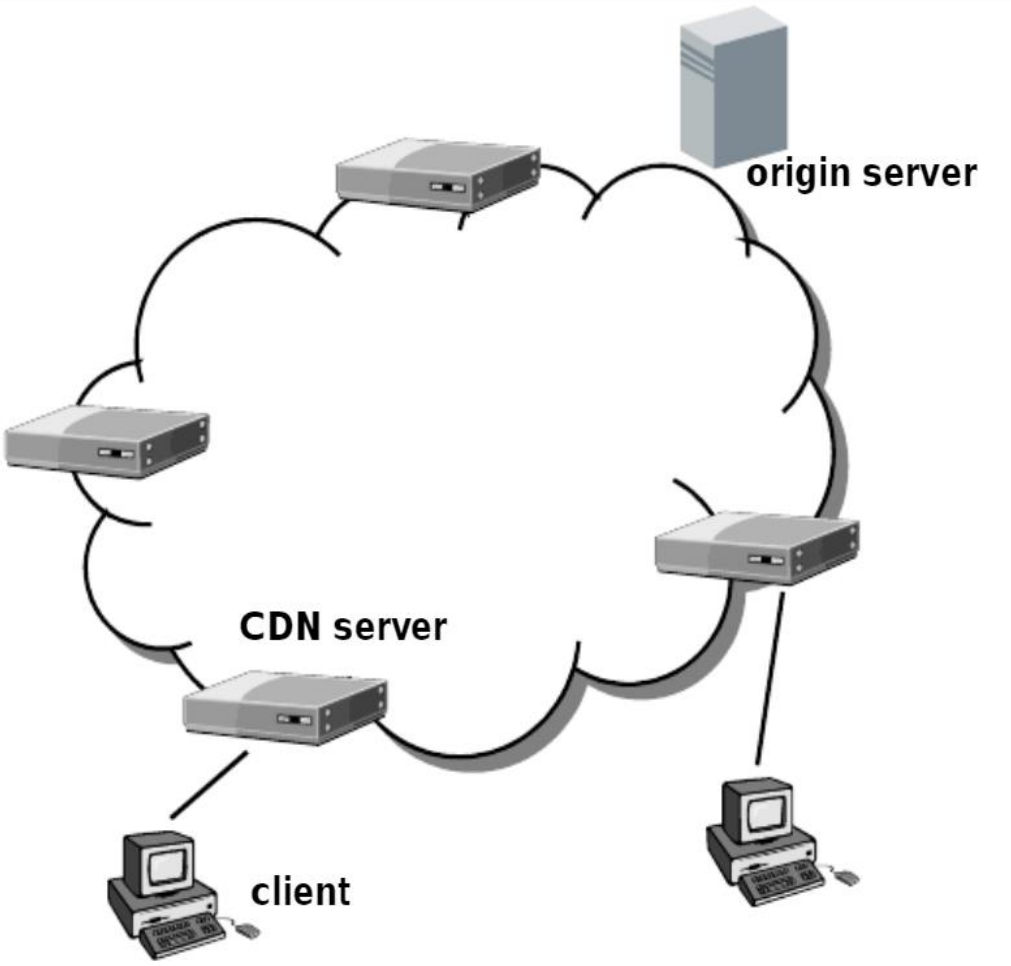


Figure 9. A Distributed CDN Architecture[10].

This architecture saves storage space by reducing the redundancy of the content stored at various surrogate servers. This architecture has a reduced latency than the hierarchical architecture.

At the same time, if the server that is responsible for redirecting the requests break down, the entire network will not work. Also, there are not many copies of the same content stored, and thus if there is a loss of data, the data must be retrieved from the origin servers. [11]

- Cloud CDN: As the number of users increased and the content type kept evolving, a CDN provider can store the content for more remote locations in cloud servers based on the requirement of the user. Also, the performance of the CDN depends on the geographical availability of the infrastructure. Thus, a cloud CDN is an economical solution that enables the content providers to store data in different servers according to certain factors

like Quality of Service (QoS) preferences and budget. Thus, moving to the cloud provides a vast storage facility without the initial capital investment for the infrastructure. Issues like flash crowds are often managed well due to the distributed location of the cloud servers. The content provider pays the CDN provider for delivery services, and the CDN provider pays the cloud services for the leasing of extra servers.

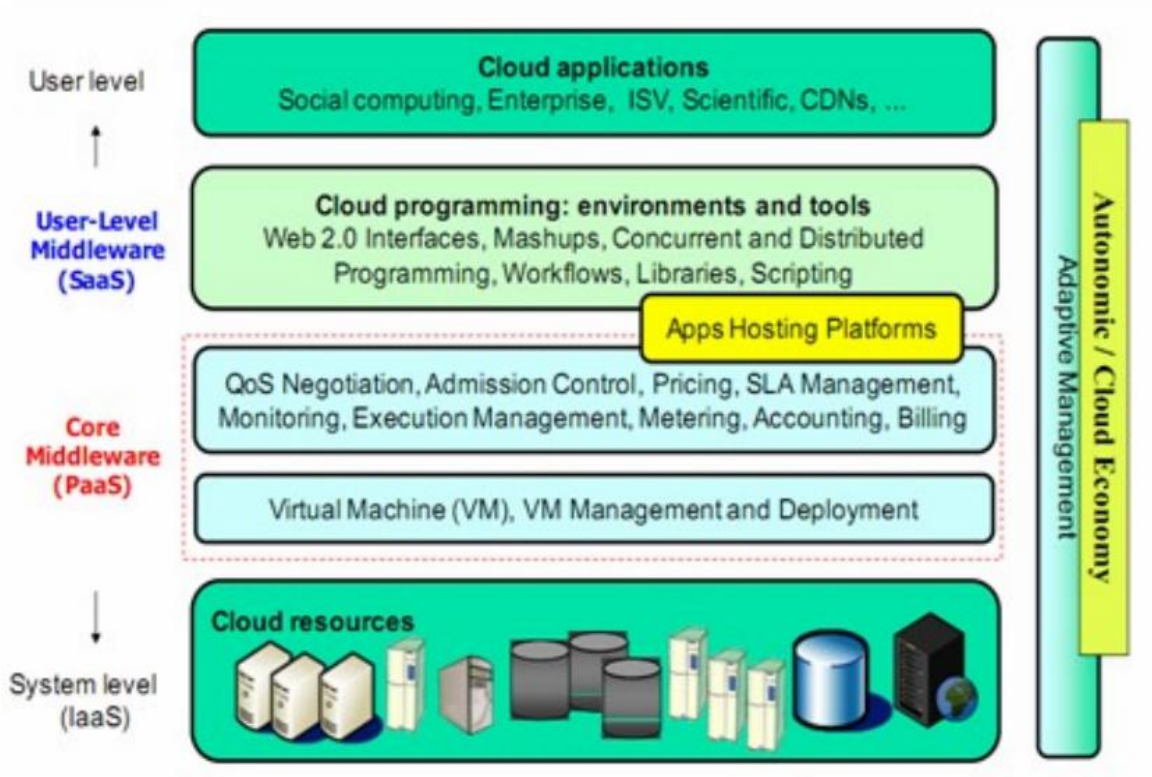


Figure 10. Cloud Computing Architecture[12]

Cloud is available to the users in the following ways:

- Infrastructure as a Service (IaaS): A service where on-demand infrastructure like servers and storage are offered to clients.
- Platform as a Service (PaaS): A higher-level service that offers a cloud environment including the middleware to the clients as per their need.
- Service as a Service (SaaS): This allows the vendors to use cloud-based software through a virtual network. The users need not install any applications or systems in their system. The applications are stored in a remote network that can be accessed through the internet[13].

Cloud provides the following features:

Cloud CDNs enable the clients to use the content as a pay-as-you-go model. Hence, it is economical than investing in the hardware and software that is necessary for the users to be part of CDN.

The Cloud-based CDNs can reduce the latency by renting out resources of a cloud provider to areas where the CDN provider is not available. Incorporating Cloud CDN can help various providers to reach remote locations. For instance, we can make use of existing cloud providers in a remote location to provide service rather than setting up an entire infrastructure in the area.

Two significant factors that should be taken into consideration to create a cloud-based CDN; are resource allocation and request routing. Estimating the resource required and Routing the user requests to the requested content.

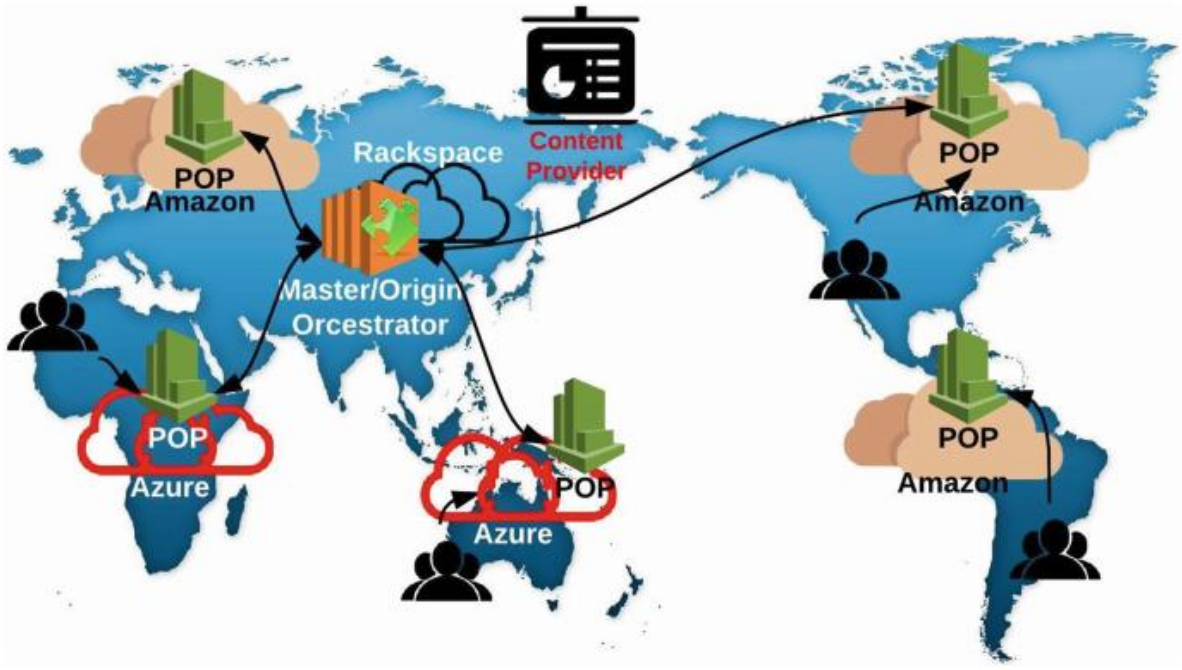


Figure 11. Cloud CDN Architecture[14].

- Peer to peer CDN: The clients can also provide and use content in this type of CDN. Hence, the content available increases as the number of users increases. The setup charges for the original contributor are low since there is no additional setup required, unlike other CDN architectures. This is implemented using a directory that contains the

list of all the content that is available, a hashing table that can be used to map the corresponding content with various client requests, and a request routing algorithm.
Any new client that is joining the system adds its own IP address and the list of content it possesses to the directory. When a peer who is already a member wants to add any content or delete any content that is already available, it informs the directory server. The peers are mapped using hashing keys to serve various client requests. This can provide load balancing and reduce the load from highly loaded peers. Whenever a peer is looking for content, the request is broadcasted to its immediate neighbors. If the content is available, it is served to the request initiator; else, it is further broadcasted to its neighbors. This process is continued till the hop count is reached or the request is timed out.

- Hybrid CDN(HCDN): As the name states, HCDN is the combination of a CDN backbone layer and a peer-to-peer CDN architecture. The CDN backbone can act as a directory server that stores the content to provide the clients. The peer edge server acts as both server and client who provides and requests the content.
  The content is distributed using the CDN backbone, and it is cached using the P2P edge servers. This way, it is easily deployable and accessible by the users. The content sharing among the peers reduces the traffic to the CDN backbone, thus reducing the load on the edge servers. [11]

1.5 Research Issues

Individual company R&D activities dominate the development in the CDN industry. The focus has been on creating new technological innovations within the industry's current trends. The following are some of the research issues involved in CDNs:

Cloud-based CDNs enhance the traditional CDN model by adding cloud-based functionalities and improving capabilities to deliver services that are not limited to Web applications but also include storage, raw computing, or access to a variety of specialized services. The integration creates new research challenges, including improving scalability, robustness, usability, performance, security, and privacy of applications that integrate with Big Data[15].

Mobile CDNs are a promising subject for future research as they explore content popularity in relation to user movement. Each end-user request is unique in terms of the information

requested, the time of the request, and the location from which the request was transmitted. When it comes to this research topic, the focus is on developing dynamic content replication mechanisms that cache content on-demand with respect to the location of requests and user activity[16].

Origin storage, when integrated with a CDN's service portfolio, can help a website store its content securely while using a CDN to reduce latency. CDNs can make use of this service for the storage of extended library content, PVR system management, and asset tracking. The challenges to building a CDN origin storage service include ensuring continuous data and service availability, guaranteeing superior throughput for uninterrupted content delivery, and facilitating automated storage provisioning and service workflow orchestration[17].

Standard IP anycasting was not considered viable for early CDN solutions. This is due to the lack of Internet Routing knowledge and the lack of importance of IP anycasting. Despite the emergence of route control mechanisms coupled with external intelligence to allow dynamic route selections, a load-aware anycast CDN architecture, and recent anycast-based measurement work, it is still possible to redirect requests using anycast IP. Further research work can be conducted to see if IP anycasting is applicable in real CDN deployments.

Chapter 2 Literature Review

As the Internet grows larger and larger, the content that is being distributed to users is multiplying exponentially. To seemingly increase the quality of the service, the content is continuously being integrated with more features like personalization, privacy, and security. Hence, to efficiently deliver these services to the users, it requires to design a great deal of intelligence to the network. A CDN is a complex network that incorporates different technologies at various levels to deliver these contents efficiently. [4]

As this project is aimed at proposing a design of an entire CDN architecture that is efficient and cheaper for small-scale content owners, it demanded learning and research of a wide range of papers. The theoretical papers that are referred to restrict themselves to a particular specific parameter often and provides a study on that feature. This project is hence a humble attempt to bring together all the aspects of a complex CDN architecture in one paper. It is necessary to know the current technologies in practice to design a new architecture.

According to George Pallis and Athena Vakali [18], the major components of a CDN network are:

- Surrogate servers or cache servers.
- Routers and network components that direct the requests.
- A mechanism that maintains logs and accounting.

The authors identified the location of surrogate placement, the selection of content to be outsourced, and the method used for outsourcing as the significant issues that result in various cost constraints to the CDN providers.

The optimal location of the surrogate server placement can result in the reduction of the number of servers required and the size of content to be cached. Thus, it can directly affect increasing the quality of service (QoS) and reducing the cost considerably. This paper supports the findings of [5] to state that greedy algorithms are the best followed by tree algorithms to place surrogate servers efficiently. For the selection of the content to be replicated, this paper recommends clustering of the content groups based on the user's

session details or URL request. Based on the user's session request, users with similar interest in contents can be grouped to improve the overall experience as well as provides insights on what similar contents can be requested in the future. The author further suggests grouping the popular content in an area helps in improving the performance of any website. The paper further infers that cooperative pull-based is the best content outsourcing technique for an optimal cache hit rate. [18]

CDN prices are comparatively high, and the solutions suggested to reduce the prices are web caching, surrogate server cache segmentation, and content personalization by employing data mining techniques. Web prefetching is the process of caching various contents before it is being requested by the users to reduce the latency. It supports a long-term prefetching algorithm that is more likely to be requested over the long term. This prefetching algorithm is called the threshold algorithm, which addresses both object access frequency and objects update frequency. The object is cached only if the frequency of these parameters combined is still over a given threshold. This algorithm is not good for short-term caching. Surrogate cache server segmentation is the technique of logically segmenting the memory into various compartments so that content stored is more easily accessible and hence reduces the accessing costs. [19]

Web personalization is based on majorly two factors:[20]

- The offline component involves preparing the data and specific usage mining tasks. The data preparation involves acquiring the user session details with a Unique Resource Identifier (URI) corresponding to each user. The usage mining tasks involve acquiring any patterns that are involved in the views or user clusters that might help to personalize the content effectively.
- The online component uses these patterns to provide personalized content based on the user navigation of the various content. These personalized contents can be anything from recommended links or targeted advertisements.
  However, this paper does not discuss how user security or privacy is maintained while storing their navigational activity.

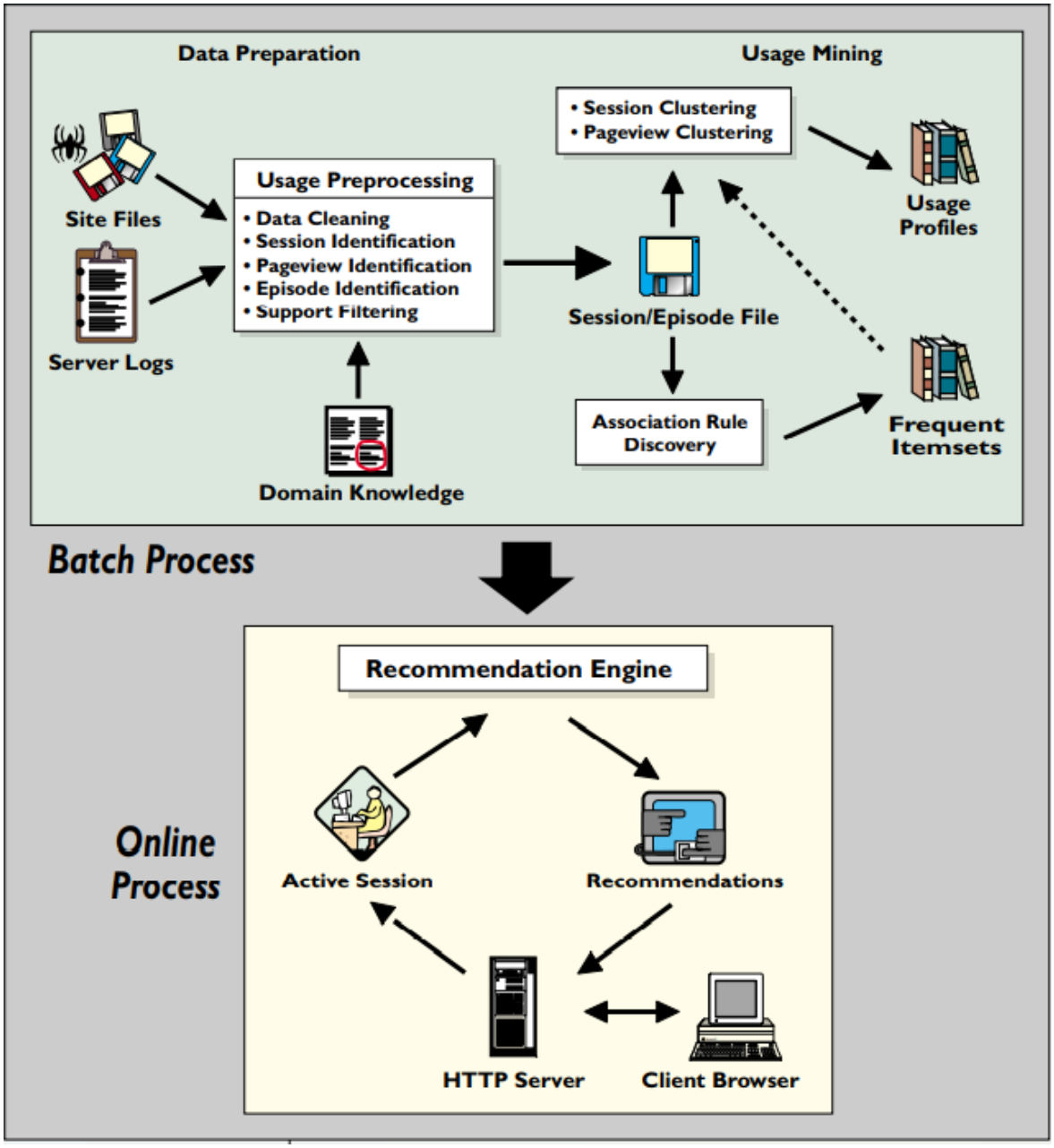


Figure 12. The architecture of a user personalization based on web activity*[20]*.

A software-driven CDN (SCDN) architecture is suggested by the authors of [21] to take advantage of the benefits of software-defined networks. In CDNs, the content provider hosts the cache servers. Nevertheless, in SCDN, the infrastructure provider hosts the cache servers near the subscriber line multiplexer. Hence, this provides control over the content pricing and caching. [21]

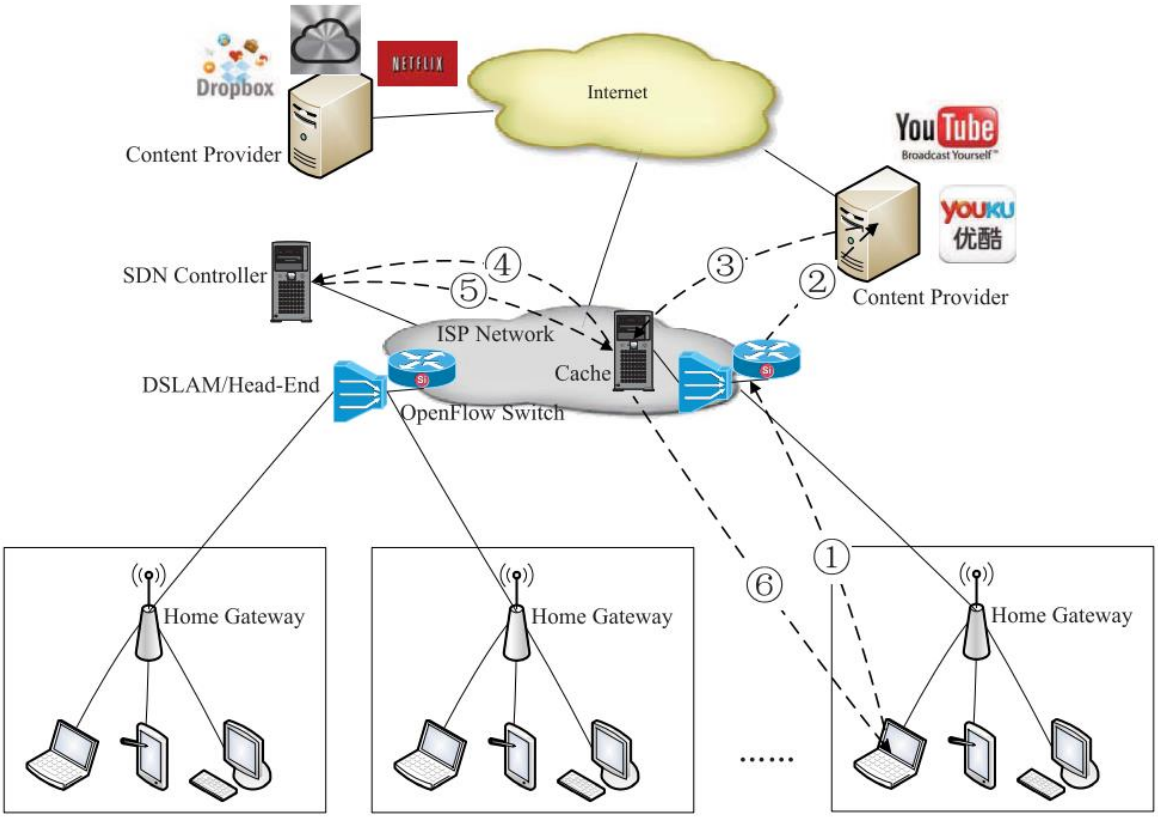


Figure 13. The suggested SCDN architecture.

In this architecture, cache servers are set up by the infrastructure provider and are directly connected to the subscriber line. In this architecture, when a customer has a content request, it is redirected to the cache server just like the standard CDN. If the content is present, the cache server addresses the request by providing the content to the user. If not, the request is redirected to the content provider. Once the content is received, the cache server must update the content request frequency and other information to the SDN controller. The decision if content should be cached is based on the request frequency is made by the cache controller. Also, the pricing is also decided by the SDN controller. The paper claims that a better QoS can be achieved by incorporating this architecture, and the IP can better reduce the charges of caching and distribution. This solution is an unconventional architecture and requires many changes in the current infrastructure. Thus, massive investment is required to set up the network according to this architecture.

Another architecture suggested for live video streaming is the hybrid CDN-P2P network. This architecture is suggested because the caching of CDN servers alone through the entire network results in expensive CDN services. The architecture suggested by this paper is as follows:

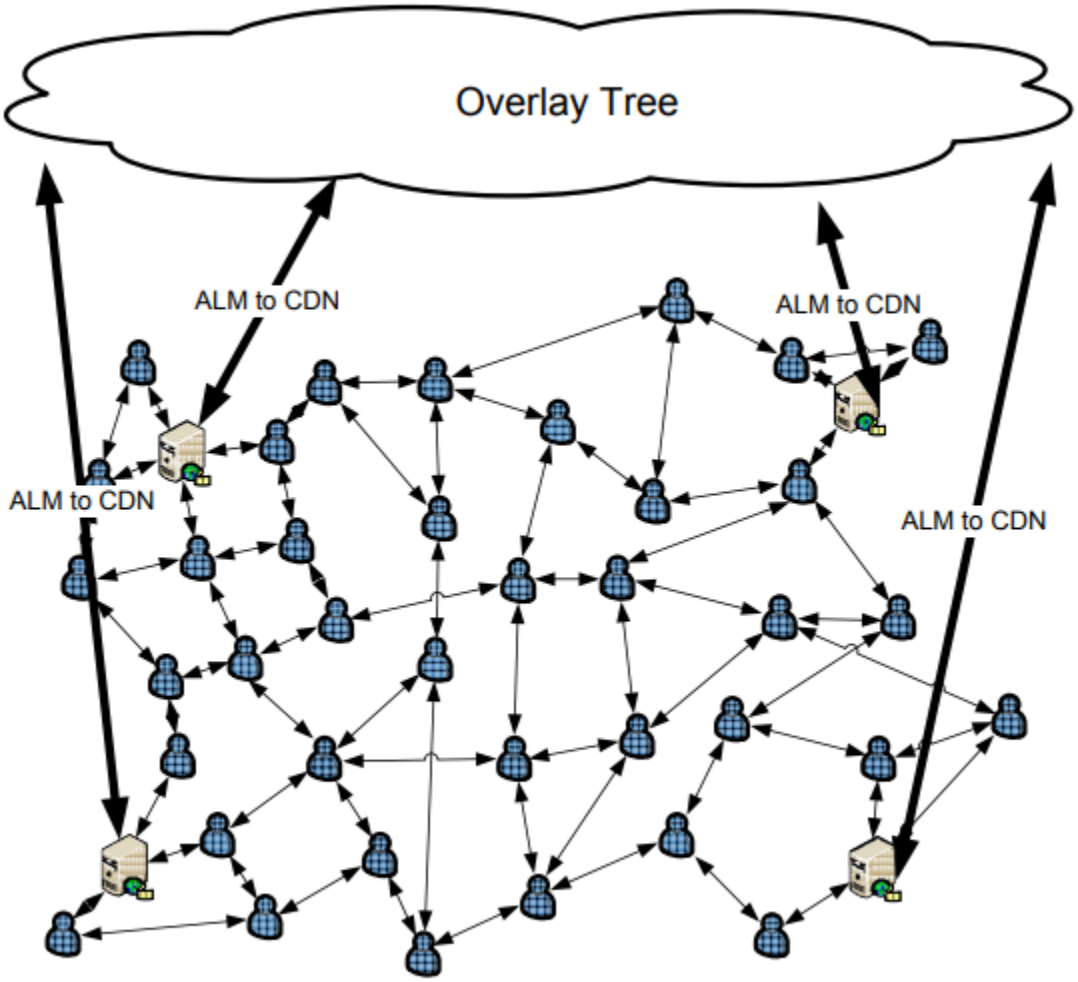


Figure 14. A CDN-P2P architecture[22].

The architecture suggests a source node encodes a live video and shares it with its neighbors. The neighbors request the videos based on the list they have received. A tracker manages all the connections within the network, monitoring all the new connections and the nodes that left the network. This architecture claims to have a higher QoS compared to the other related networks. Nevertheless, it also has a higher delay for starting up compared to the other networks. [22]

A review of content distribution network architectures was conducted by the authors of [11], where they discussed various CDN architectures and suggested that a hybrid network with CDN and P2P is suggested as a better architecture. The discussed architecture, as shown in the following figure, consists of CDN servers as the backbone of the network [23].

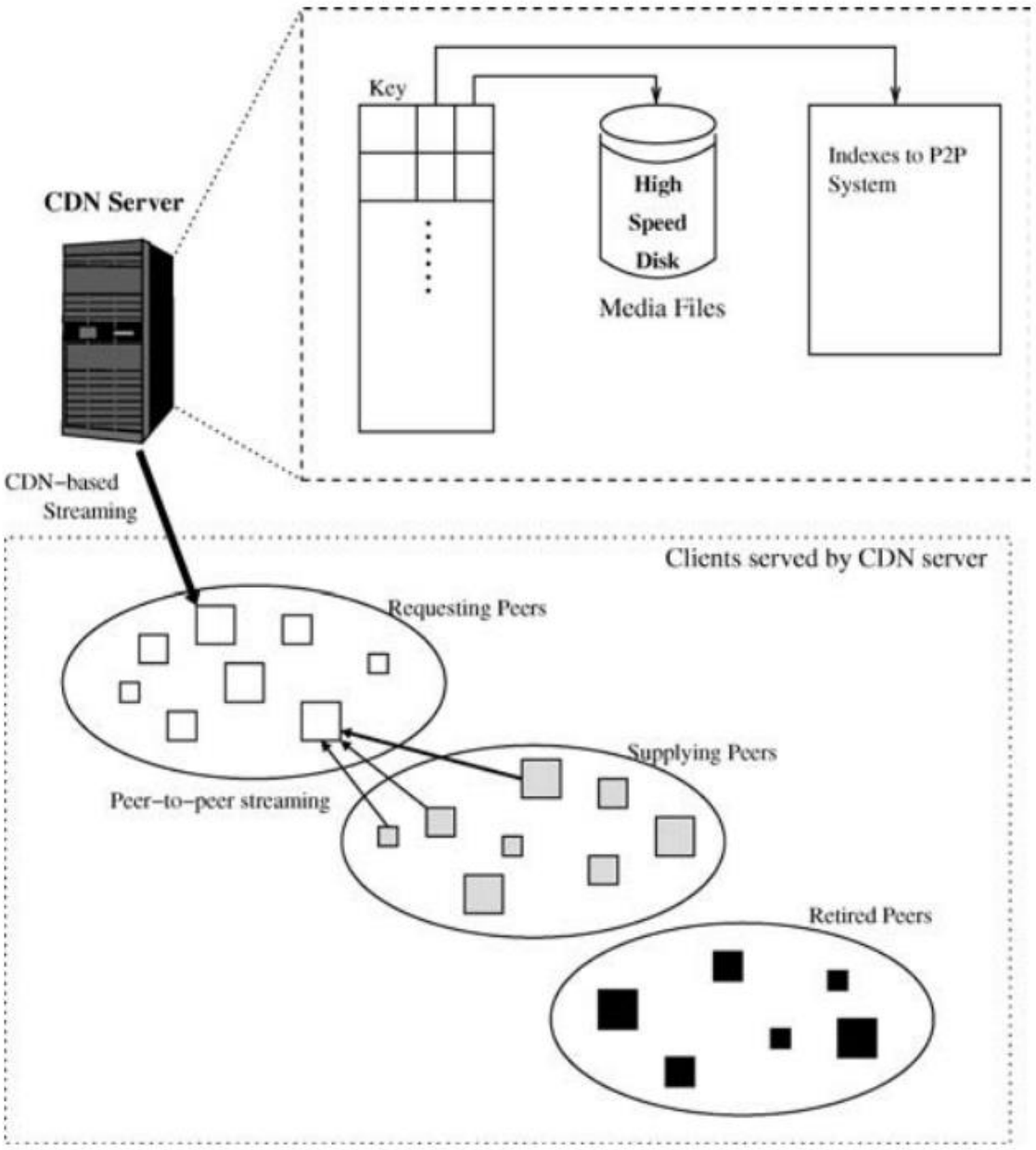


Figure 15. A cost-effective HCDN Architecture[23].

These CDN servers behave as a streaming server and index server for the P2P server. These servers can be used to cache content for the P2P servers based on the available network

bandwidth. It consists of a P2P edge layer closer to the clients that support P2P protocols. These servers can handle most traffic and reduce the pressure on backbone networks. All the content is sent to the clients through the CDN first and then cached by the edge servers. When an end-user requests content, the CDN server will search for the content within the P2P server using a tracker. This combined CDN P2P architecture is claimed to provide better performance than the conventional P2P network. [22]

In the paper of [24], the authors suggest a design of MetaCDN architecture that takes advantage of multiple cloud providers to provide a solution of low cost and better performance to the content providers. Storing the content at the cloud providers reduces the chances of flash crowds in the network. The MetaCDN is designed to provide a CDN system that uses multiple cloud systems to provide a cheaper alternative CDN to the content providers and clients. To achieve this, an extensive study of various cloud providers is conducted by the authors to identify the prices for storage space in the industry. MetaCDN is designed to be provided to the users as a web portal with a backend database to store the user accounts and other details. The users can log in to the CDN using the web portal and can start deploying the content onto various storage services that are available with MetaCDN. The difficulty in integrating different content providers to one platform is managed by the connectors. MetaCDN provides an integrated platform to small-scale content providers who would like to distribute their content to the end-user. This is done by the allocator that suggests to the user which cloud can provide the best service. A QoS monitor tracks the performance of the cloud servers. The Load Redirector is responsible for redirecting the end-users to the most efficient or best server.

A general block diagram of MetaCDN is as follows:

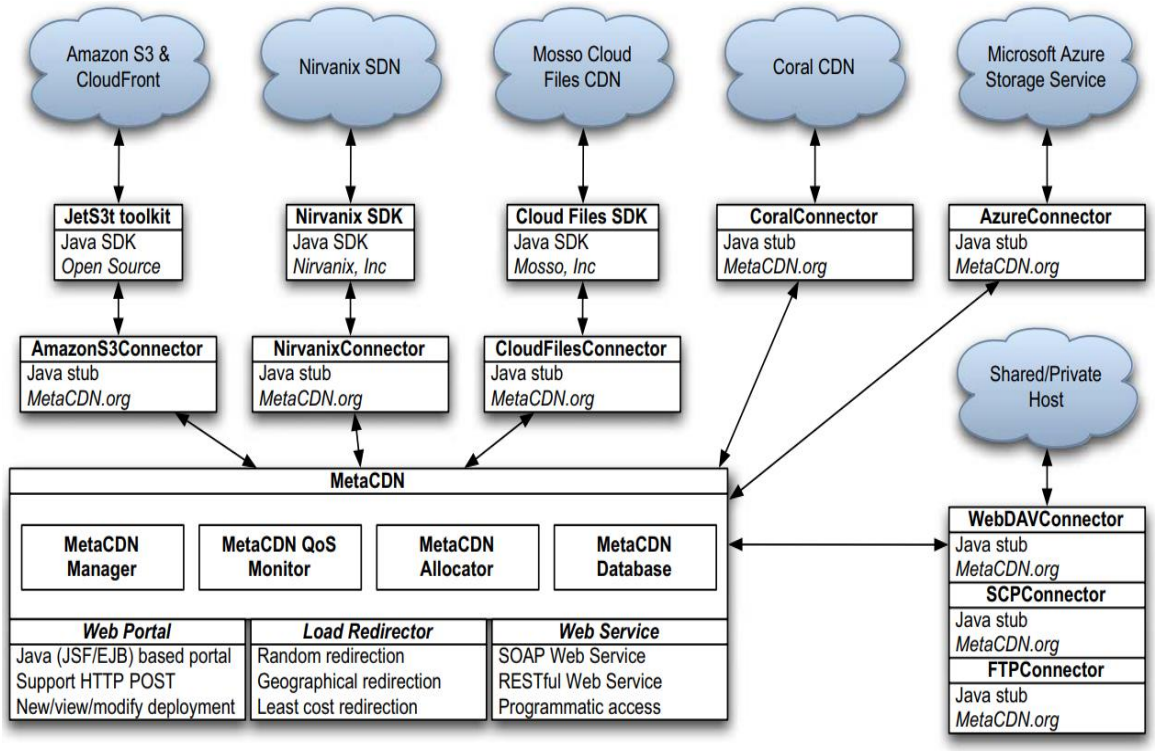


Figure 16. Block diagram of MetaCDN[24].

The authors tested the deployment using various cloud service providers like Amazon S3 and Cloudfront, Nirvanix SDN, Mosso Cloud Files CDN, Coral CDN, and Microsoft Azure Storage Service using test files of various sizes like 1 KB, 1 MB, 10 MB, and 100 MB[24].

MetaCDN claims to achieve good performance in the areas where the density of the cloud servers is high. However, at the same time, thus the performance of this architecture is totally dependant on the density of a third-party service provider.

Another cloud-based CDN architecture is proposed by the authors of [12] for mobile Wireless Applications Protocol(WAP) users. This architecture proposes edge points that are a combination of cloud servers and cache servers spread in a distributed manner. P2P technology is adapted to distribute content to edge servers. The authors claim this system an ideal alternative for telecom service providers because of the reduced delay and broader bandwidth achieved through the architecture.

The authors of [1] suggest a cooperative push-based architecture based on a cloud-based system to reduce the storage cost and improve resource utilization. The ultimate goal of the project was to deliver an efficient content retrieval method where only the partial content is being cached to

the servers before the content is being requested by the clients, thus reducing the latency, server loads and improving the network traffic. The major problem for designing a content delivery network that is identified by the authors is to difficulty in ensuring content distribution and management.

The problems identified by the authors on the existing designs are as follows:

- Lack of efficient solutions to identify an optimal cache server to serve the client request.
- Lack of communication between the servers in serving the client request on time.
- The lack of efficiency since the entire content is being cached at every level of cache server.

The solution suggested by the author is through the unconventional method of cooperative push-based architecture, which is still in the development stages and has not been tested for more extensive networks. This architecture proposes to replicate the partial contents to the cache servers by prefetching the content as soon as it is updated or inserted.

The advantages of the proposed algorithm are as follows:

- The content is cached partially before the request is made, and the cache servers cooperate to deliver the content on time. Thus, when there is a high rate of requests for new content, it avoids the formation of a bottleneck.
- The latency is considerably reduced if the cooperative push-based content caching is implemented efficiently. The testing is currently implemented only for one edge server group, and further implementation is in the development phase.

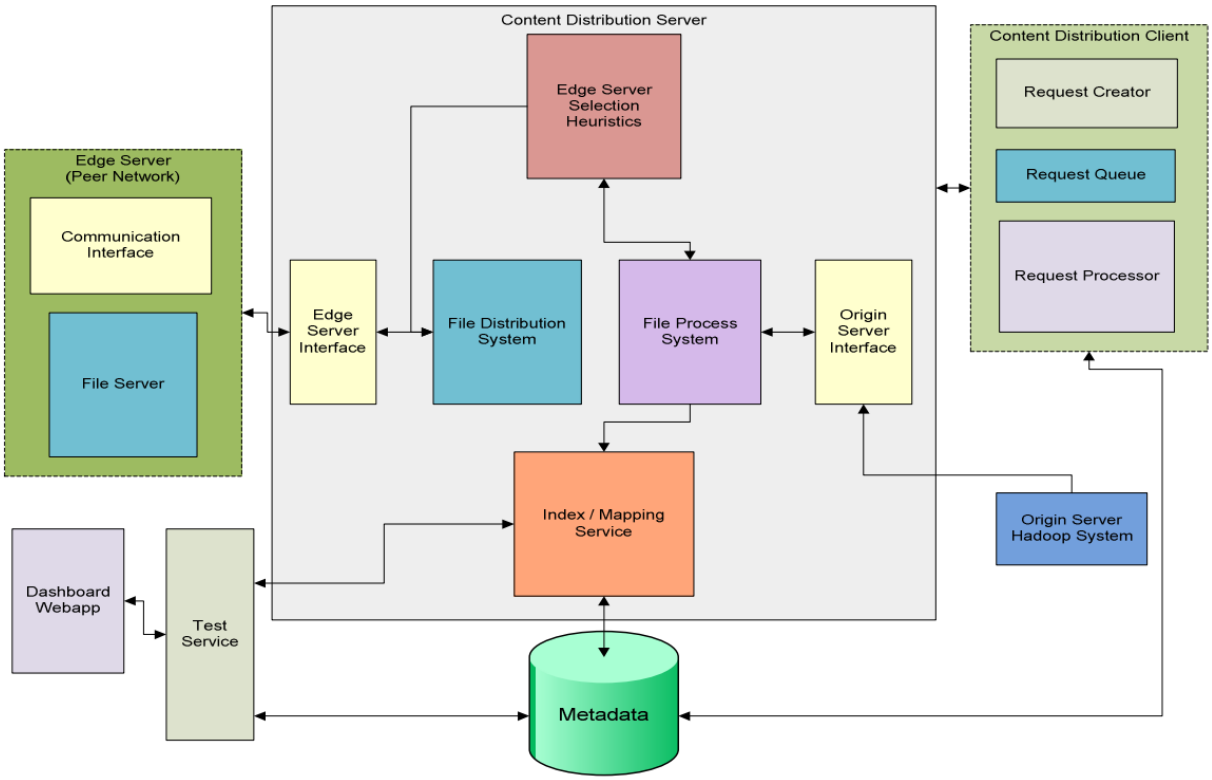


Figure 17. CDN with Cooperative Push Based Architecture[1]

The cloud storage at the origin was set up using a Hadoop framework. The videos in origin are broken down into smaller fragments by the Application Programming Interfaces (APIs). All the edge servers where the document is to be pushed are identified using a heuristic server selection algorithm.

Another notable feature proposed by the algorithm is storing the files in a Scalable Coded Video (SVC) in the origin server. This is a standard that enables the provider to encode once and can be decoded to various formats according to the bandwidth available as well as the computing capacity of the users.

A daemon named Cloud Scanner is responsible for pushing the new content to the edge servers from the origin server, as shown in the following figure.

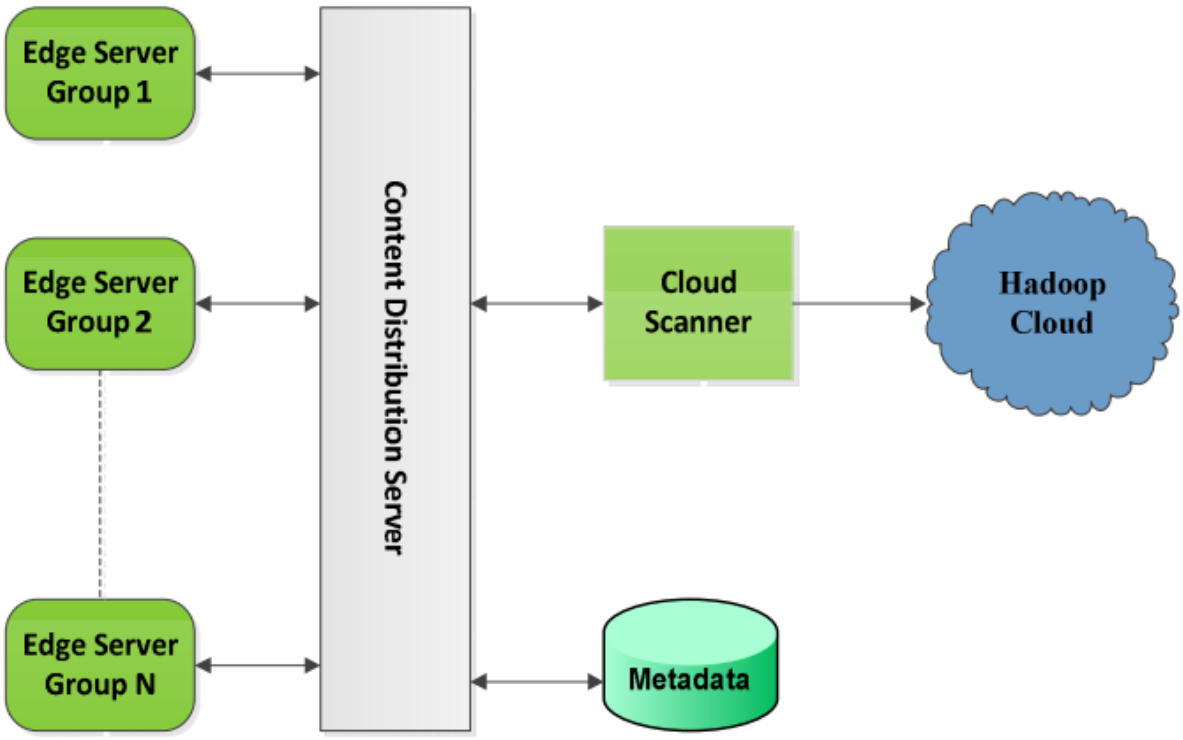


Figure 18. Implementation using Cloud Scanner [1]

At the same time, the algorithm used for determining the cache servers where the content is required is not mentioned in the paper. This might be because the test was done on a single edge server group, and hence it was out of the scope of the experiment. Identifying the edge servers where content must be cached depends on various factors like the region, languages, and popularity rate. This can be extremely challenging to be implemented in a real-world scenario.

The other cloud-based architecture referred to is by the authors of [25], who suggested a low-cost and a fault-tolerant CDN based on the cloud. This architecture was proposed to solve the challenges faced by the content providers due to the high charges incurred by the commercial CDNs and the limited capacity of the surrogate.

The problems that are addressed by the authors are as follows:

- Load balancing – An inefficient load balancing system can cause network congestion, delay in traffic, and break down of the request routing system. Hence, awareness of the load and locality is an inevitable factor to consider while designing a load balancing

technique. The health of the network and the server has to be considered before routing a client request to a particular server.

- Network Latency – In order to reduce the latency of the content, the surrogate servers are kept as close as the users.

The proposed architecture is based on a Master-Slave architecture similar to the one proposed in this report. The content provider can store the contents in the cloud that acts as an origin server.

The following is the architecture of the proposed CDN:

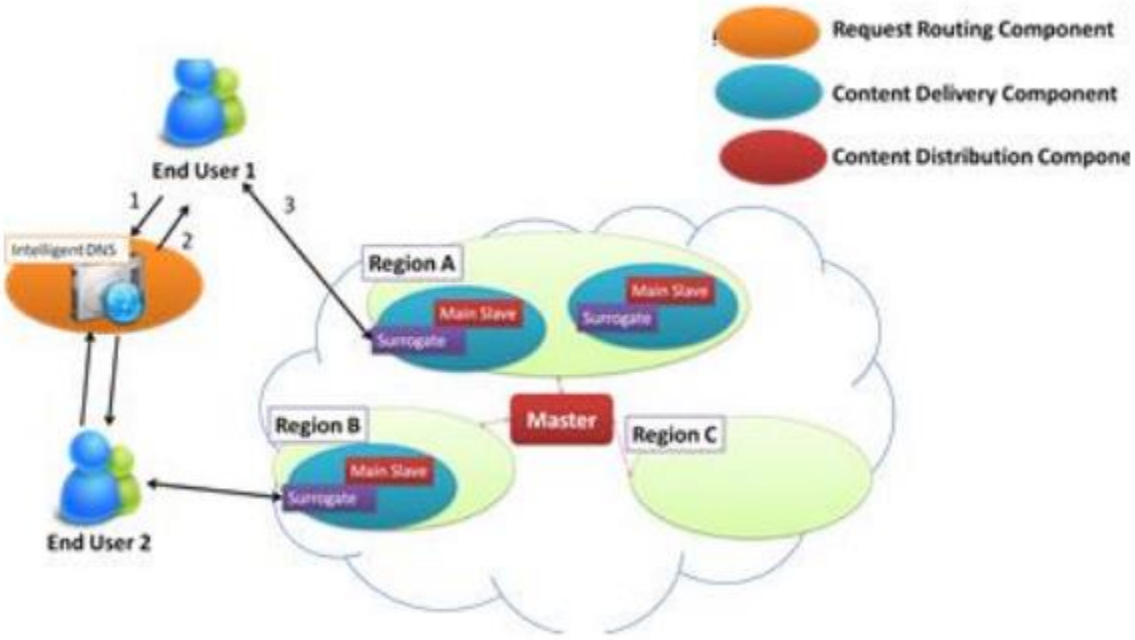


Figure 19. CCDN [25]

The major components of the architecture are as follows:

- Request Routing System – This component is responsible for redirecting the client request to the best surrogate available. A DNS-based request routing is adapted in this architecture that is claimed to reduce the access time and improve the quality of the service.
- Content Delivery System – This system is responsible for delivering the content to the end-users through a group of cache servers. If the content is present in the cache server, it

is delivered back to the client. If it is not present, the content is copied from the cloud server through the main slave, as shown in the figure. The content is cached to the surrogate as well as being delivered to the client. A regional load balancing is implemented where the load balancer redirects the request to only one of the surrogates using a hashing technique. This is claimed to have a better caching technique with a lower rate of duplication and thus saves the cache space. Also, the system ensures fault tolerance by making sure the surrogate is alive before redirecting the request to a particular surrogate. Also, the metadata cache can be used as an index to find where the content is stored. This architecture also suggests a backup server for the main slave in order to ensure fault-tolerance.

The framework of the content delivery system is shown below:

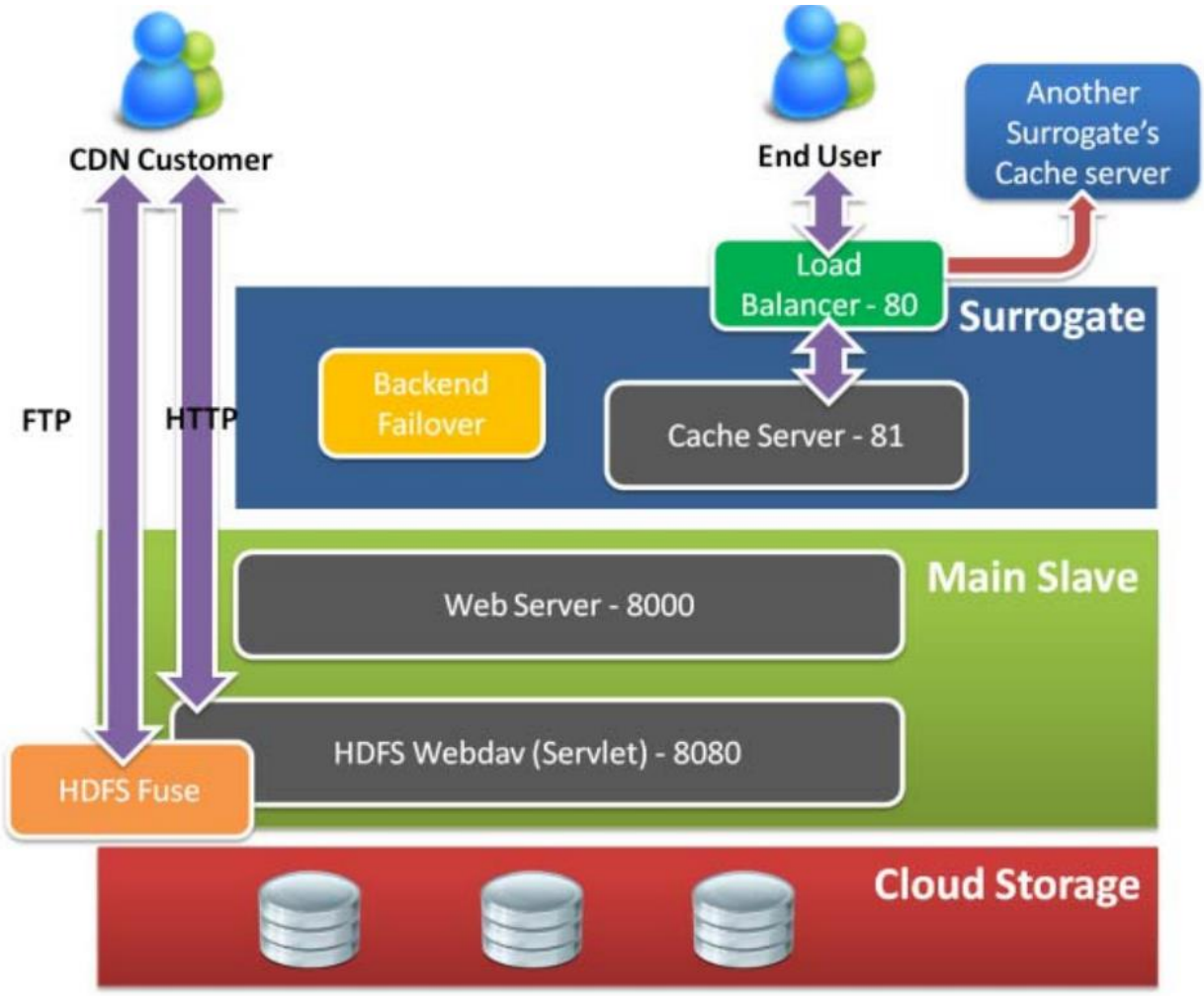


Figure 20. A framework of Content Delivery System[25]

The origin server is on the cloud in this architecture, and hence any client can upload the client to a specific domain where they are given access. They can select the region where they want to deploy the content as well, thus ensuring region awareness. This concept is a unique addition to the paper and can be incorporated into other networks as well.

The authors of [26] have suggested a novel scheme of combining CDN and Cloud Computing for video streaming services and other applications where network traffic is high.

The major factors that are considered in this paper are as follows:

- Cache Hit Ratio – It is defined as the ratio of the difference between the total requests and the missing requests to the total number of requests.
  HR = (TR – MR)/TR
  where HR is the Cache Hit Ratio
  TR is the total number of requests made by the client to a particular surrogate
  MR is the number of missed requests that were redirected to the origin from a surrogate.
  The higher the hit rate, the better the CDN.
- Latency – The delay in the time it takes for the CDN to serve a request to the end-user is considered as latency. It is optimum to achieve the least latency possible to ensure a better user experience.
- Reserved Network Bandwidth – This is the network bandwidth used by the origin server at any point in time. If the usage of the Reserved Network Bandwidth is high, it means there are more cache misses in the surrogate server and the CDN is not working efficiently. Hence, a small Network Bandwidth is desirable.

The content provider stores all the content in the cloud instead of its own infrastructure. The master server distributes the content to various surrogates or edge servers and decides how many copies of the content has to be made. The proposed architecture has the following features:

- The discussion on if the PUSH or PULL method has to be incorporated was left as an open-ended question.

- The authors suggest that the CDN should not be following a strict hierarchical method and instead select a distributed network where every server is cooperating to fetch content.
- A global load-balancing algorithm is suggested where the content should be distributed according to various factors like the geographical location of the end-user, a load of the nodes that are on the way to the destination, and the network traffic through the path.
- A local load-balancing algorithm ensures that no devices are overloaded on the path to the end-user. Hence, they infer that a load balancing algorithm that supports real-time monitoring should be chosen.
- For a video streaming service, the entire video need not be cached at the initial stages. So, the authors suggest that only the prefix of the video has to be cached at the surrogates. The remainder of the video can be cached while the video is played by the end-user. Hence, the authors suggest a mechanism called fragmenting, where the video is divided into blocks of a fixed size.

In their paper on replica server placement algorithms [26], the authors have pointed out that CDNs in the future must be scalable in order to tackle the ever-growing content density. Hence, to enhance scalability, resource efficiency, and QoS, the authors proposed a CDN that incorporates cloud computing and Network Functions Virtualization (NFV). NFV is a method of virtualizing the entire network components like routers, switches, firewalls, and load balancers without the need for the actual installation of the products. This leads to a substantial reduction of initial investment cost and human effort. NFV, according to the author, can bring a great deal of agility to the CDNs by providing highly customized services.

The other factor that is being addressed in this paper is replica server placement and content placement. Replica server placement is considered an NP-hard problem. Server placement in traditional CDNs is generally an offline procedure where real-time end-user demand is not considered. Replica server placement in cloud-based and NVF are virtual. In this paper, the authors present a detailed summary of replica server placement algorithms in various forms of CDNs.

The NFV architecture that can virtualize CDN entities like cache nodes, request routers, and load balancers is first proposed by the European Telecommunications Standards Institute (ETSI) [27]. Significant developments in this field have been evolved when features like value-added services and transcoding features are incorporated in recent times. A few other NFV based CDN architectures can be found in these papers [28][29][30]. This architecture might contain a management and articulation console to incorporate this dynamic behavior of the NFV networks. These dynamic chaining can be implemented using an SDN controller.

The following is an architecture of NFV based use case:

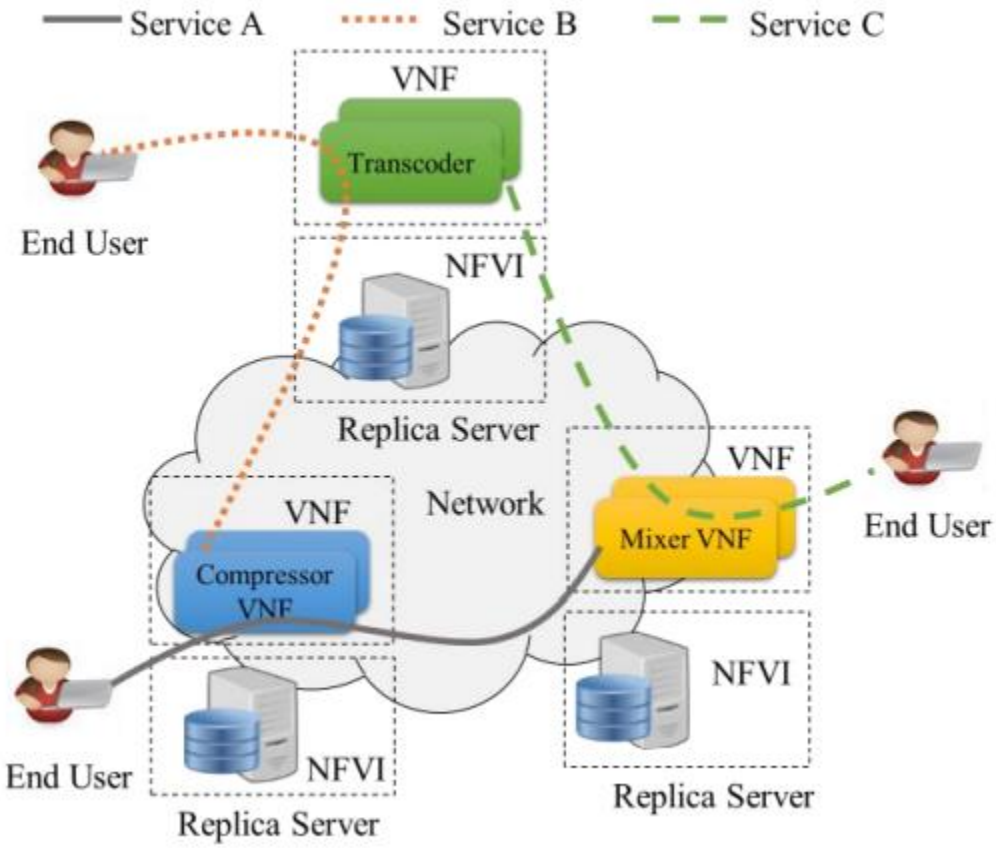


Figure 21. NFV based CDN[8].

A replica server placement problem should consider the following parameters as inputs:

- Cost Parameters – While considering the cost involved in cache server placement, both the deployment charges and the delivery of the content from the cache to the end-user should be considered.
    - The deployment cost is the initial cost that is involved in setting up the infrastructure either by buying the hardware or leasing the servers from a cloud provider.
    - The delivery cost is the cost involved in delivering the content from the cache server to the consumers. It can be calculated by including the cost per bandwidth unit that is spent over the request.
    - The update cost is the cost incurred in updating the content from the origin server to the cache servers or transferring data from one cache server to another. It can be calculated using the unit bandwidth cost and the rate at which cache servers are updated after a timeout.
- Network Parameters – A replica server placement algorithm can be dependent on the various parameters that involve the location of the customers, the location of the other network entities like data centers, and the communication links between them. Network parameters that are discussed in the paper are as follows:
    - Topology – The replica server placement algorithm varies with the network topology that is being used in a CDN. In a flat network, the servers can be kept in the form of a full mesh where various devices are spread apart. In a hierarchical network, the replica servers are kept in a tree topology where the origin server is the root.
    - Network Parameters – Network parameters like latency, available bandwidth, hop count, and link quality are significant factors that help in designing the position of the replica servers.

      Latency is the time taken by the content to reach the end-user after a request is made. This can be calculated by obtaining the round-trip travel time. Also, Global Network Positioning can be used to identify the latency. GNP is calculated based on the distance between the end-user and the coordinate of the server.

A detailed taxonomy of various replica server placement algorithms for both traditional and evolving CDN is provided in this paper, as shown in the following figure:

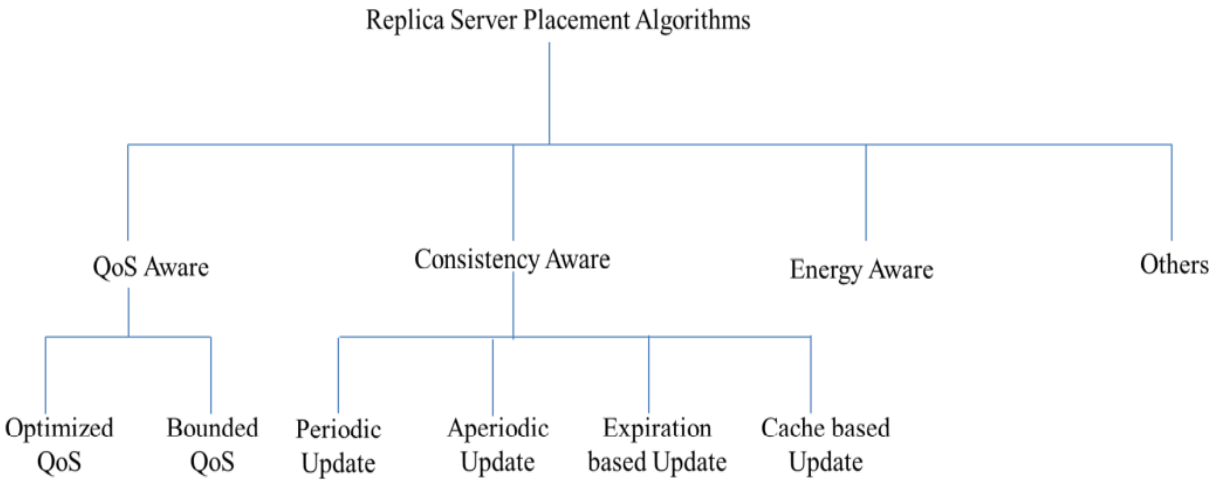


Figure 22. Taxonomy of Server Placement Algorithms on traditional CDNs

In their paper on replica placement [31], the authors have tried to understand the optimal location to place a cache server to minimize the latency and network overhead. The method adopted by the authors is to place several replica servers at various locations of a given topology to identify the optimal location to reduce the overhead. Various client-replica assignments were assumed to evaluate the efficiency of the server placements as follows:

- Client-replica assignment – It is assumed that each client selects the nearest cache server irrespective of other factors like latency or RTT. Also, for simplicity, it is assumed that there is no limit of connections to a cache server from the client.
- Replica placement – Various replica placement mentioned below are considered.
  A greedy replacement of evaluating all the locations in topology and choose a location where the network overhead is the least is considered.
  A fanout placement suggests that the nodes with the largest fanout are the closest to all other nodes and, therefore, a better choice of location.
  A max AS, max router fanout placement is where a router with the largest fanout within an AS is considered to be the closest to the rest of the nodes in the topology.
  A max AS min router fanout placement is where a router with the smallest fanout within

an AS is considered to be the closest to all the nodes in the topology. This is considered just to include all the scenarios though sounds unpractical.

A random placement of replicas is considered within the topology.

All the above scenarios were analyzed to infer the following:

- Metric Space – The average latency and the network overhead are the two parameters analyzed in the paper. It is assumed that the latency is directly proportional to the hops between the client and the server.
- Hence the average latency is given by the following equation:

$$AvgLatency = \frac{\sum_{Clients} No.of\ hops\ between\ the\ client\ and\ the\ server}{Total\ no.of\ clients}$$

The entire network overhead is given by the following equation:

- It is also assumed that the network overhead is directly proportional to the number of hops between the clients and the cache server. Moreover, this is proportional to the number of clients.

Thus, network overhead is given by the following equation:

$$Network\ Overhead = \sum_{Clients} No.of\ hops\ between\ the\ client\ and\ the\ server$$

- Hence, the relation between Average latency and the network overhead can be given as follows:

$$AvgLatency = \frac{Network\ Overhead}{Total\ no.of\ clients}$$

The paper further compares the relative latency and the relative overhead with standard and greedy placements.

As per their experiments, they found out that fanout placements performed better than greedy placement methods. One of the advantages of fanout placements over greedy placement is that fanout does not require the knowledge of client locations to identify the ideal location of the server. This is a significant benefit because client locations are a dynamic parameter and change from time to time.

One of the most critical parameters of a CDN is the caching feature that has to be efficiently implemented. In their paper [32], the author's Dean Povey and John Harrison, conduct a simulation experiment to identify the applicability of different caching mechanisms.

The primary caching mechanism that was prevalently used in CDNs was hierarchical. The copies of content in this method were often stored in the form of a tree. Following is a diagram illustrating a hierarchical caching system.

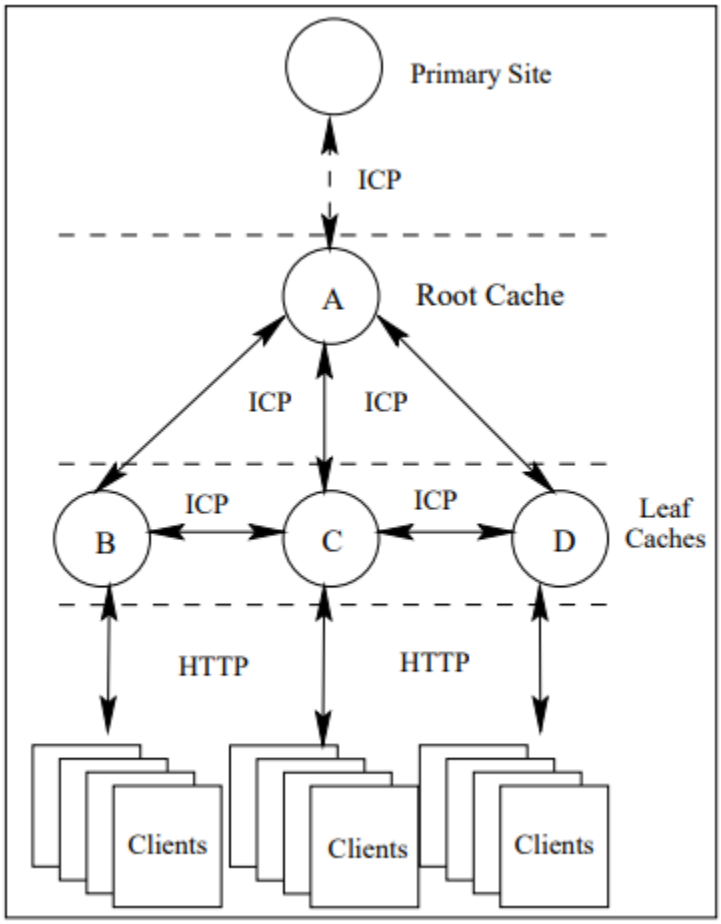


Figure 23. A Hierarchical Caching System[32]

When a client requests content, a request is made to a leaf node using the HTTP protocol. If the content is not present at the leaf node, the request is sent to its parent node. The content is retrieved from the first node that responds with a cache hit. If the content is not present in any of the nodes, the request is redirected to the root node, where all the content is present.

The advantages of the hierarchical content caching mechanism as identified by the authors are as follows:

- The network bandwidth can be reduced by the decrease in the recursive transfer of information at various levels of the network.
- When there is a heavy load to the root node, load balancing can be conducted to reduce the traffic in the network.

Following are the disadvantages of the hierarchical caching mechanism:

- The cache memory required at the higher levels of the tree is significantly high.
- All the information regarding the nodes that are below or at the same levels has to be maintained by cache servers that are acting as parent nodes.
- Research shows that the cache hit rate of the nodes at the root is way lower than the ones at the lower levels. This displays a significant loss of disk space because these nodes store more content than the ones at the bottom levels of the tree.

The hierarchical approach was improved using a distributed architecture of caching where upper-level caches store only details about the contents of the lower caches. This does not require upper-level caches to have large memory space. The distributed caching technique is a recursive method where the cache contents can be searched quickly. Hence, if there is a request for a particular content from the client and it is redirected to the root cache, it directs a URL that points to the location of the content in the network.

Advantages of the distributed caching technique are as follows:

- The upper-level nodes need not store all the data that is stored in the lower nodes. Preferably, upper nodes should contain information about the location of the content, and it has to redirect using the pointer towards the content. This can save a lot of disk space in the upper-level nodes.
- The hierarchical caching requires constant querying of its siblings to identify the contents that are cached at those nodes. Nevertheless, this problem is solved in the distributed caching technique, where the content details are updated using frequent advertisements.

The authors of [33] discuss various difficulties in designing and implementing a CDN network. The infrastructure cost that is involved in setting up a CDN is exceptionally high. The configurations are too complicated since various parameters are involved, like replica placement, architecture, and load balancing mechanisms. Hence, the author suggests the usage of simulation software to design and implement the CDN network. Implementing a simulated model creates another variant of challenges like resource management and assumptions based on real-world scenarios. The lack of a roadmap in these CDN simulators can bring more challenges for a designer. Hence, the author is discussing various implementation methodology and processes for caching mechanisms in a CDN framework.

Web caching and replication are the two most used methods in content delivery. The intent of both these procedures is to reduce the latency and bring the content as close to the end-users. Proxy servers are generally deployed by the Internet Service Providers where the CDN providers have a contract with them to cache the content provided by their network. Proxy servers dramatically reduce the bandwidth consumption, latency, traffic, and congestion in the network. Also, the eviction of stale data is determined by the cache eviction policy. Each content is assigned a priority value called cache utility value, where the content with the least value is evicted first to create space for new content.

The author also discusses the disadvantages of implementing a proxy server. Any delay or failure in updating the proxy server will result in users receiving older content that directly affects the quality of the service.

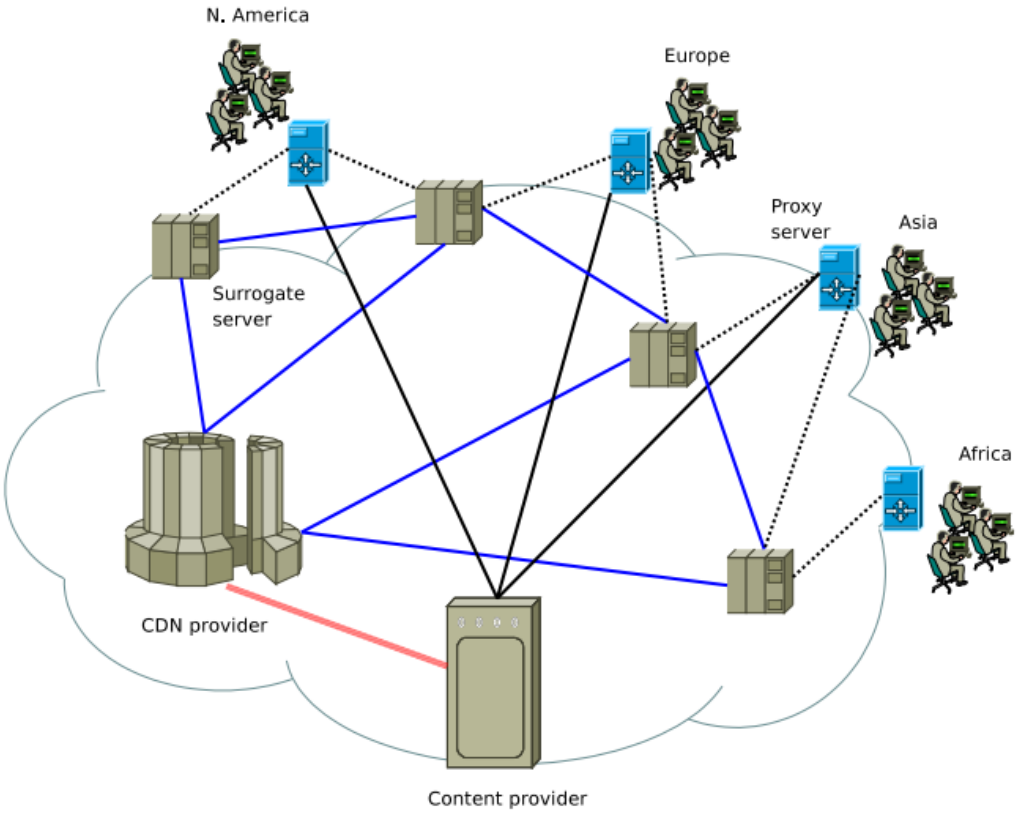


Figure 24. Content delivery using proxy caching [33]

Unlike proxy caching, replication of the content is lesser dynamic in nature. The content remains in the cache for more longer duration. Thus, inefficient management can cause a lot of stagnant to be stored in the cache servers. This results in more cache miss in the system. Hence, the author suggests a method where the cache server can be partitioned to store both static and dynamic content, as shown in the following figure.

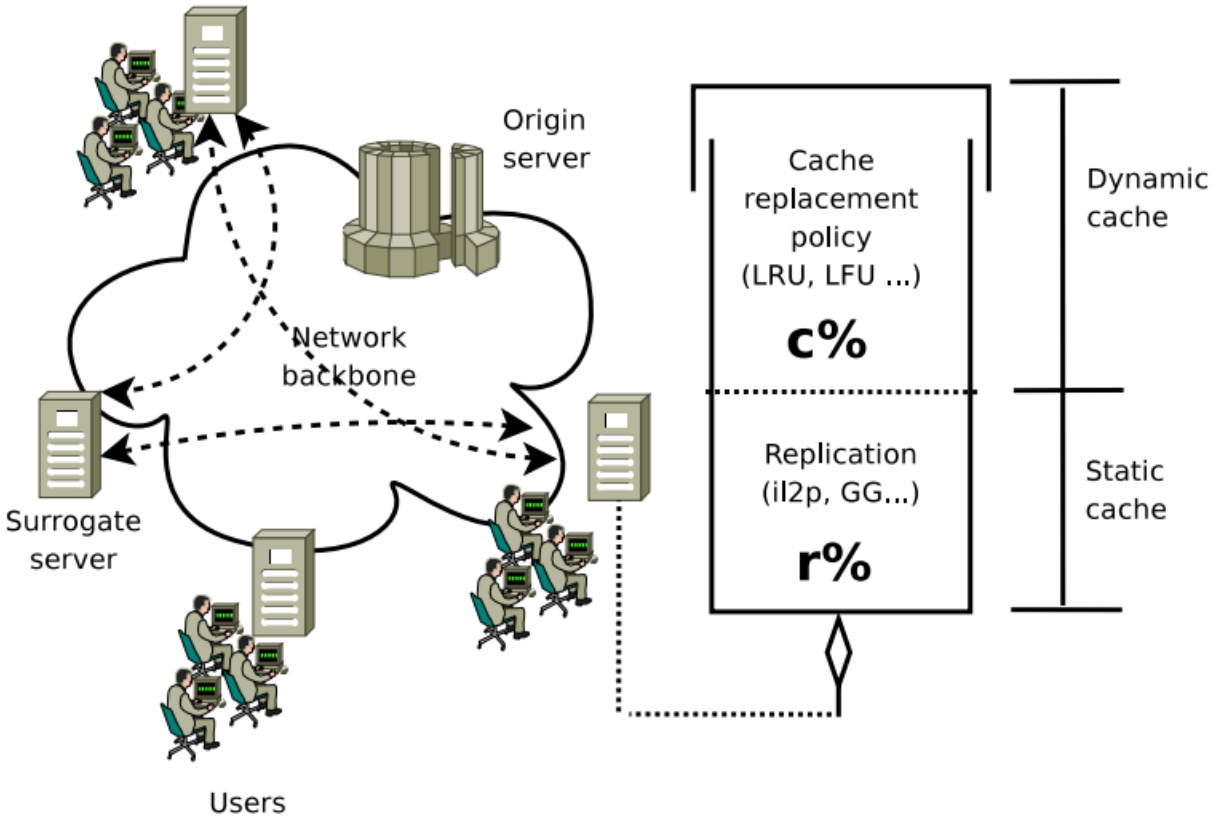


Figure 25. Partition of Cache for Static and Dynamic Content[33].

The challenge in such an implementation is to decide what fraction of the cache is to be used for static and dynamic content. Various heuristic approaches were suggested to resolve this issue of partitioning the cache server.

The dynamic content itself can be divided into three as given below:

- A content that requires a periodic update from the origin server. The changes in these websites are predictable, and a periodic update is necessary.
- A content that is updated only as per the need basis. The updates of these content depend on the user pattern and trending content.
- The unpredictable content is the most challenging in accordance with caching. These contents often end up as a cache miss most of the time.

The content is often fragmented into different pieces and cached to improve the speed of cache retrieval. Various Akamai networks incorporated a fragment caching methodology suing the Edge Slide Includes (ESI) that is accepted by the world wide web [34][33].

The cache performance is directly proportional to the RAM of the server. A cache mechanism involving search-insert-delete-update operation should be implemented in every caching system.

The Cisco Annual Internet Report (2018–2023) white paper predicts that more than 82% of the internet traffic will be videos, and it is expected to rise to 90% by 2025. Hence, it is important to cache the videos to the edge servers to reduce the latency.

In their paper[35], the authors propose a caching policy called PrefCache that is claimed to improve the performance of the edge servers by using a joint preference learning. This algorithm learns the common preference of all the users in a specific area rather than individual users. To achieve this, all the videos are divided into specific categories based on the type, duration, and other similar parameters. The preferences of the users are learned by using an unsupervised model. This algorithm maintains a continuous connection with the content providers to collect information and work independently. Also, the model is compressed using a tree-structured algorithm. To efficiently adapt when the user's preferences change, this algorithm evicts older caches.

Consider a TV series that is being watched by a user. Though the video is not being continuously being watched by the user, it is likely that the user will continue watching the upcoming episodes. Hence, clustering videos based on such parameters helped the author build a model that will help cache the video more effectively.

Some of the key observations made in the paper regarding the user preferences are as follows:

- User preferences of certain topics remain more static than the preferences based on individual objects. The authors identified that the popularity of the top 10 videos fell dramatically compared to the popularity of the top 10 topics that were divided into groups. A graph of the learning is given below:

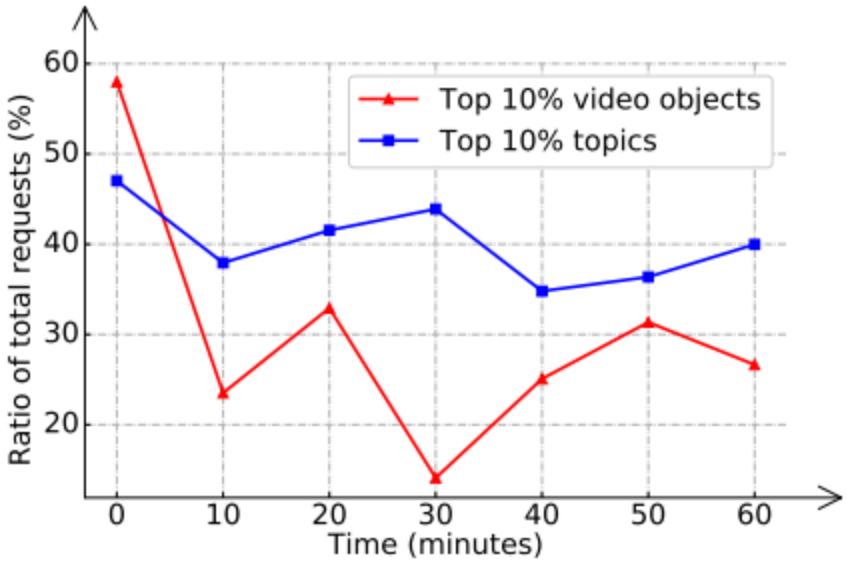


Figure 26. The change in popularity for video topics and individual videos[35].

- Caching videos based on topics generated a better performance of the cache than caching individual videos.

The overview of this preference-based caching algorithm is as follows:

- Information of user preferences is collected by the data collector that is present at the edge nodes and not from the users. Each video contains some information like the category, duration in their metadata that is collected by the edge server. The edge server picks the most used videos that belong to these categories from the content provider and caches them.
- The modeling of preference using a learning tree that helps in compressing the branching process that helps in modeling the preferences efficiently.
  The tree divides into specific categories like preferred, unpreferred, and unsure. This helps in caching items that fall in this category.

An overview of the preference-based cache system is given below:

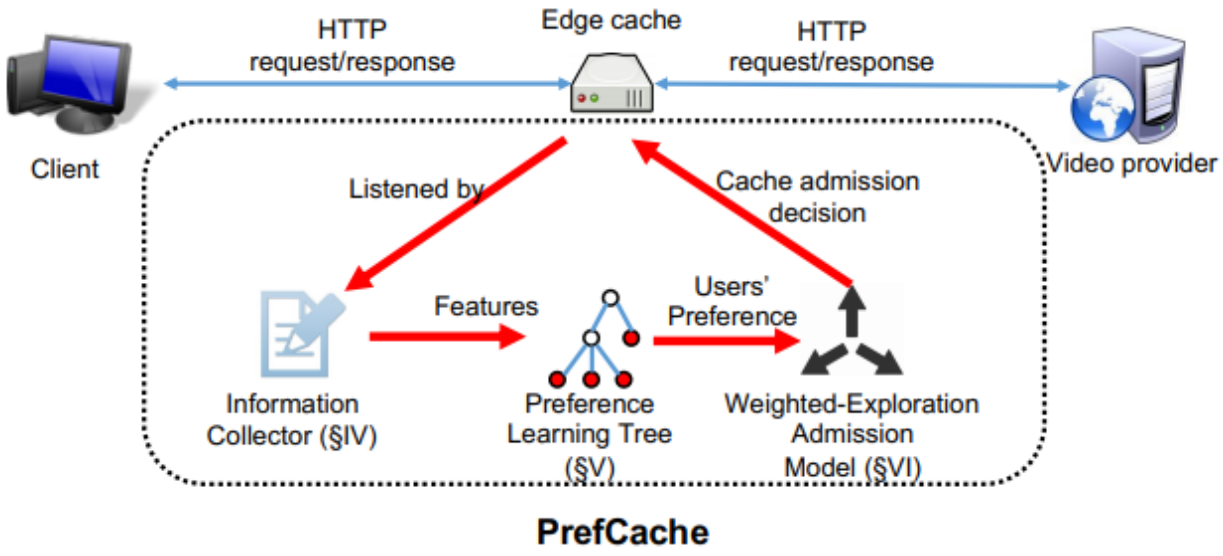


Figure 27. Overview of the preference-based caching system.[35]

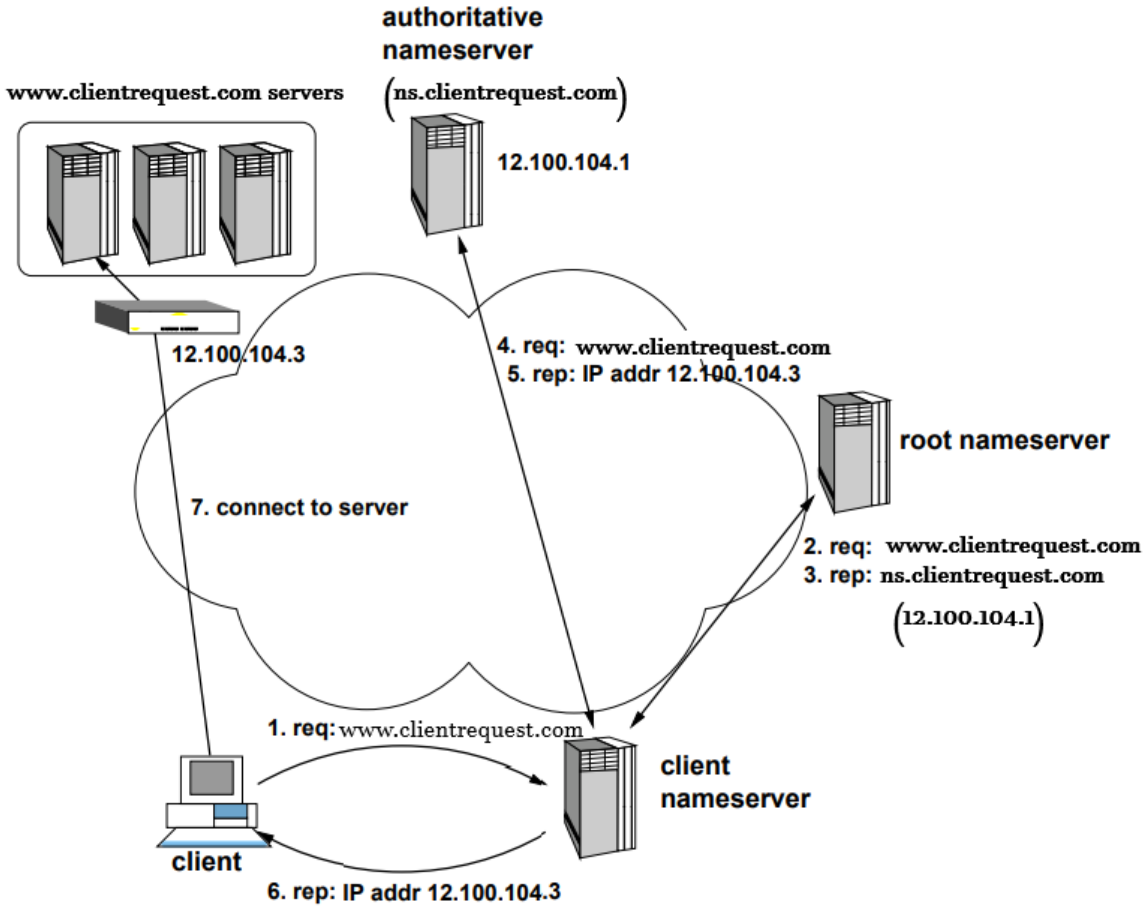
In the paper [36], a research of an efficient resource reservation of cloud services and the various factors like rent and user experience is considered to develop a heuristic approach to it. The authors designed a design that is practical for large-scale cloud-based CDN devices to attain the highest quality of user experience at a very minimal cost. The significant problems that are addressed are reservation of the resources and request routing algorithm. The users are generally spread across a wide geographical area, and to improve their user experience, it is necessary to spread the information across various data centers closer to the users. The client requests can be ever-changing and highly dynamic over time. They designed an approach that decides what contents are to be cached and where that can be cached. Also, this decides how much bandwidth and infrastructure that should be reserved at each data center. An online algorithm is suggested to route requests online to address the efficiency of user experience. Extensive simulations were conducted to study the performance of the theoretical proposal.

In this paper[36] on improving the efficiency of DNS based server selection, the authors address the drawbacks of this server selection. The main issues addressed in the paper are the analysis of the impact of low DNS cache lifetimes on the latency and measurement of client and server name proximity.

The argument the author is trying to put across is that the DNS local resolver and the end-user are topologically distant and, on average, eight hops apart. Hence, the author claims that the

resolution suggested by the local name server can create an unwanted delay in providing the responses to the end-user.

The following diagram describes the performance of the basic DNS operation to find a client resolution to a website called www.clientrequest.com:



The working of DNS is as follows:

- The client uses a resolver that makes a query to its local DNS resolver or name server.
- The local name server tries to resolve the request that is made for [www.clientrequest.com,](http://www.clientrequest.com) as shown in the image.
- The local DNS server sends the request to the root server, but it receives the IP address of the authoritative name server of the sub-domain ns.clientrequest.com.

- The local DNS server then sends the query to the authoritative server to receive the IP address of the requested server.[36]

This paper suggests addressing the problem of client-server proximity of DNS by making certain modifications like carrying the client information in the additional records of the message. The format of a DNS general query packet and the modification in the resource record is shown in the figure below:

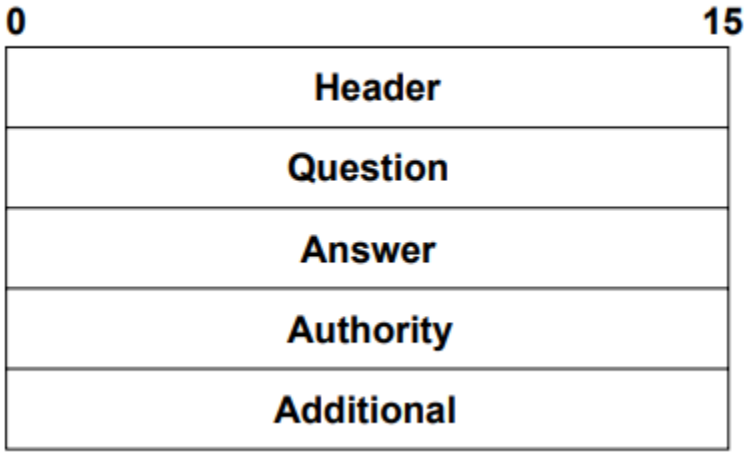


Figure 28. DNS Packet format.

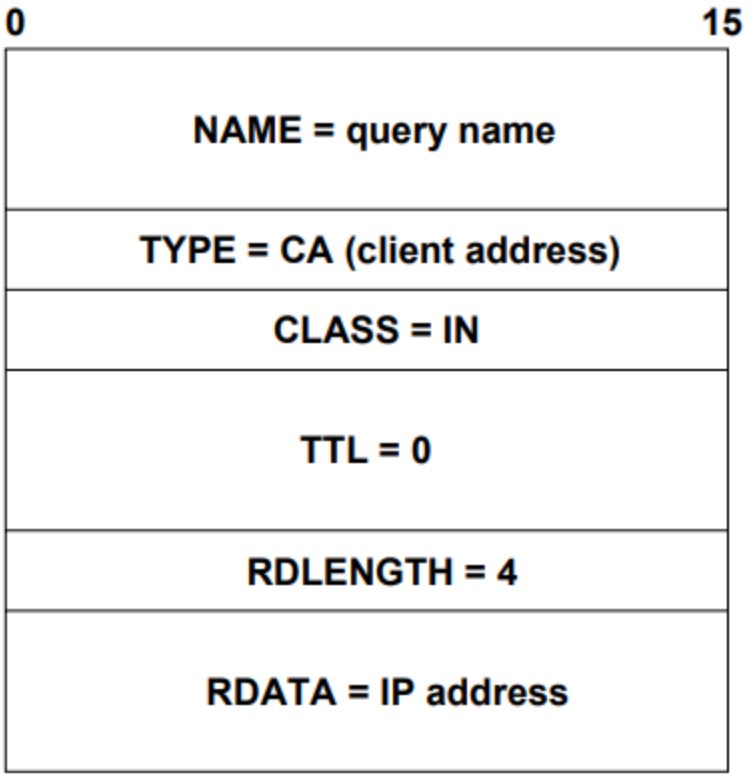


Figure 29. Resource record of the additional field of a DNS packet.

The disadvantage of the suggested architecture is that it is not always applicable since some of the DNS packet client addresses are not accurate when the client networks are connected through various firewalls or proxies.

In this paper [36], the author suggests a new mechanism that can be included in the request routing implemented in a CDN. The request routing mechanism has two parts, as per the authors. They are requesting routing algorithm and informing the user of the selection of the server. The authors suggest a change in the conventional request routing algorithm where the request is redirected to the origin server when the cache server does not have the corresponding content.

In this algorithm, a different monitoring server is present that keeps on checking the health and data status of all the cache servers that are involved in the CDN network. Hence, the request is redirected to a cache server based on the health status, which contains various information like the availability of the cache server.

The authors of [36] discuss the potential costs and benefits of long-term prefetching in a CDN. The authors claim that long-term prefetching can help to increase the cache hit rates. Prefetching can be of two types: short-term and long-term.
In short-term models, content is cached based on the recent user history, whereas in the long-term model, the content to be cached is decided based on a global access pattern. Hence, the authors suggested a CDN architecture of cache servers that follow a hierarchy. The cache servers at a higher level contain contents that can efficiently improve the global architecture, whereas the lower caches store content for the short-term.
The challenges addressed by the authors on long-term prefetching are:

- Statical analysis – The various access patterns of contents should be analyzed to identify the content that is more likely to be requested more.
- Object selection – Based on the above analysis, a set of objects should be selected that can maximize the cache hit rate while minimizing the costs.
- Sharing updates – As per the selected contents, updates should be shared to all the cache servers that are required to cache the content. A cooperative push-based system is used in our proposed architecture that can help in updating the information to all the cache servers.

The suggested algorithm in this paper considers the parameters such as object selection frequency and the update frequency and chooses only the content objects that are above a certain threshold after considering these parameters.

Though this technique improved the cache hit rate, it increased the network bandwidth and disk space costs considerably.

Using the available bandwidth is also one of the most important factors while designing a CDN network. Hence, the author of [37] proposed a newer technique for on-demand delivery for streaming videos. The idea implemented is reserving a portion of the bandwidth between the client and server in a way it is not affecting the quality of the video. Hence the video that is received will be encoded in a smaller bandwidth than the available client-server bandwidth. Some newer methods that use lesser bandwidth to merge the data transmissions are suggested. The arguments the authors put forth are that

without bandwidth skimming, it is not possible to stream merging. Stream merging can reduce the bandwidth to the order of the number of customers.

Due to the surge in the popularity of video traffic on the internet, it is necessary to learn the video popularity that will help the caching devices to prefetch the right content that will improve the hit rate significantly. Hence, the authors of [38] suggested a data analyzing solution that can provide insights to decision-making in caching.

This is a cloud-based solution that can support Analytics as a service (AaaS) architecture functionality. The architecture consists of CDN providers and cloud PaaS. The AaaS is integrated into the PaaS and executed. A few REST-based APIs are used at both ends that will help in the communication involved between the architectures.

The system architecture of the proposed architecture is as shown in the following figure:

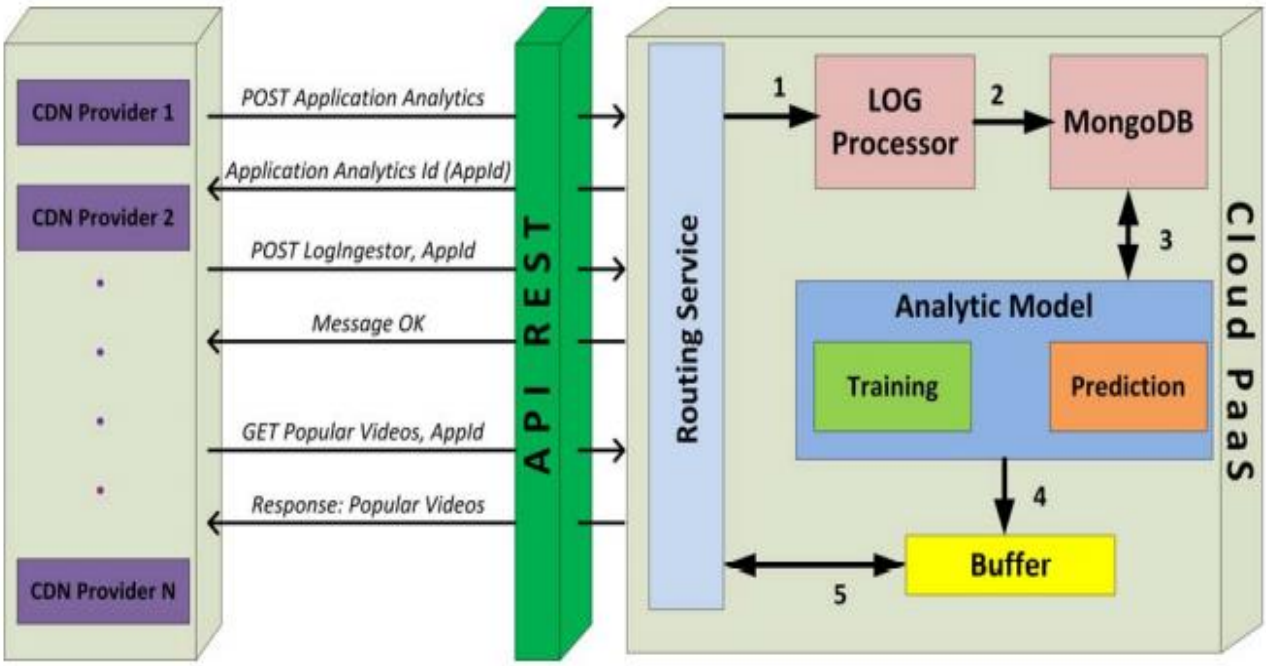


Figure 30. Analytics as a Service (AaaS) system overview.*[38]*

The architecture interfaces the CDN providers and the cloud platform using Representation State Transfer (REST) architecture. The primary services provided by the architecture are training service and the prediction algorithm that calculates the popularity of the videos in an ad-hoc manner.

The authors of [39] propose a hierarchical content delivery architecture with a two-layer overlay in the network. The authors discuss the various downsides of flat network topology and furthermore suggested a hierarchical topology that can be used in large CDN networks. To design the network, they have grouped devices into clusters either by manual or automatic configuration. Also, request routing techniques can be either transparent or non-transparent with switches and routers connected. The caching approaches that can be adapted are proactive and reactive caching. However, in this paper, they have adopted a cooperative reactive caching technique to implement this topology. There is a lot of traffic update packets present in the network that creates a network overhead in the architecture.

The suggested hierarchical content routing for multimedia is as follows:

- Servers are grouped into a two-layer cluster that is logically connected.
- Any client request is often addressed by the local CDN server.
- If the content is not available in the local CDN server, it is redirected to the CDN server inside the cluster.
- If that misses, the request is redirected to a CDN server in the nearest cluster.
- Else, it is forwarded to the origin server.

The following is the overlay of the suggested CDN:

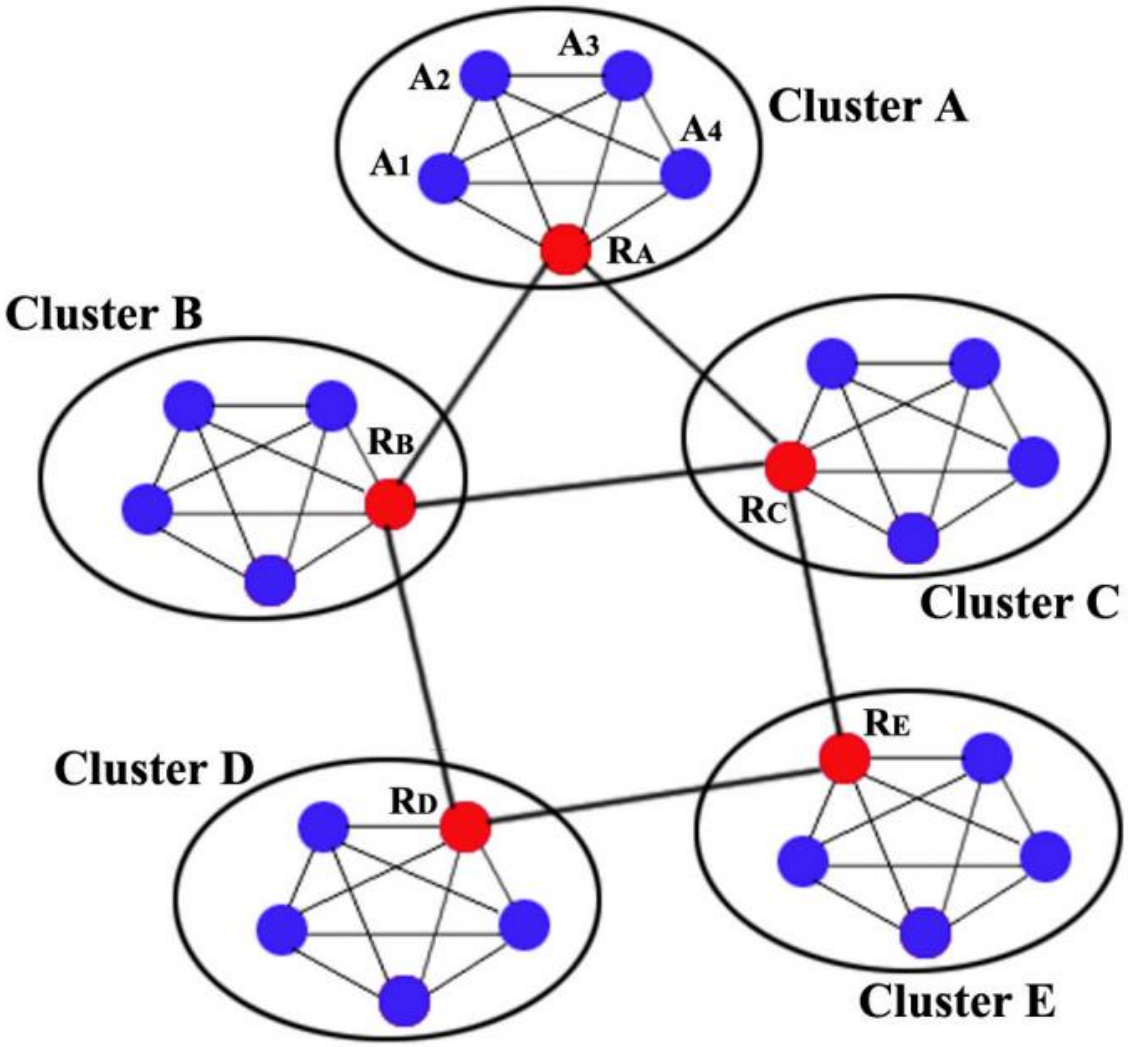


Figure 31. Network overlay of the suggested network.

2.1 Security in CDN

As CDN networks cache content over various geographic locations, the security of this content is of concern. Hence, the vulnerabilities of these networks are uncovered. According to [40], not only protection against these CDN networks can be broken into, but it also can be used to amplify the impact of the attacks.

For instance, hackers have used amplification techniques to make DDoS attacks at least twice as large as they were in the past off-late. At the same time, these attacks are more sophisticated and often use multiple attack vectors, and are opportunistic in attacking when infrastructure is

already heavily loaded. They could attack the network layer, the application layer (for the user-facing portion), or the DNS.

The following image shows DDoS attack size and frequency over time:

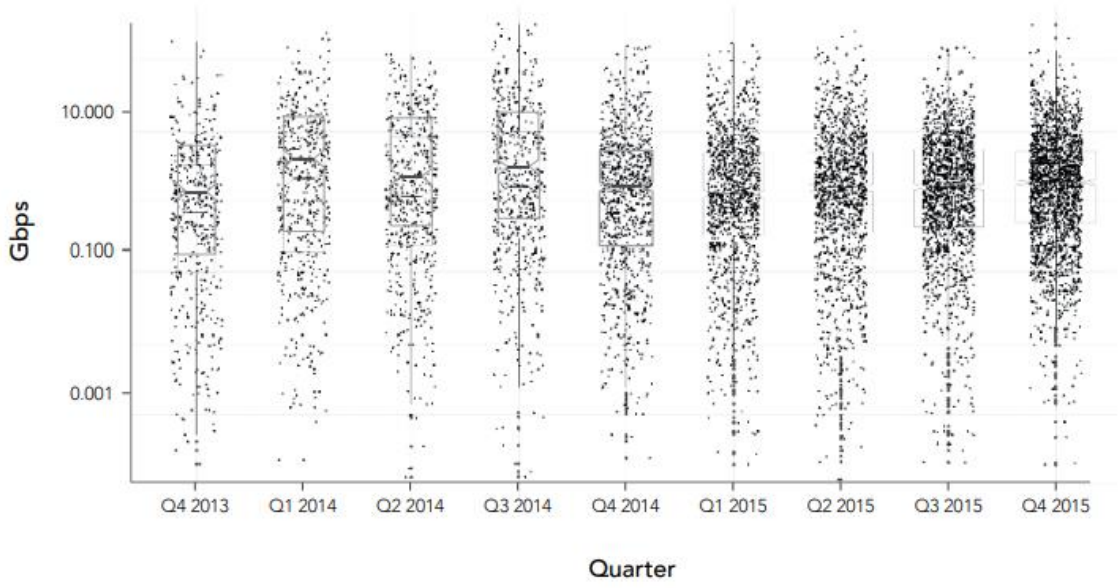


Figure 32. DDoS size and frequency[41] over time

The attack discussed in the paper is performed as follows:

- The CDN network is scanned to identify the edge server IP addresses.
- After obtaining the IP address of the edge server, a large number of HTTP requests are sent to various edge servers of a CDN.
- The edge server is manipulated to redirect the requests to the origin server. This can be done by corrupting the URL of the requested content by adding random strings to it.
- Once the origin server creates a TCP connection to the end-user, the attacker drops the connection to the edge server. Thus, the attacker saturates the origin server while not spending its own bandwidth.
- The origin server will not suspect anything since the request is from within the network by an edge server.

The mitigation to this attack is discussed in the paper as follows:

- This attack majorly relies on random string generation that allowed the attacker to break through the protection configured in the edge servers.
  To avoid this, a CDN provider can block all the requests with string arguments and return an error to the end-user for such a request. Akamai network allows their requests to be dropped if the request contains random strings.
- The other problem with this kind of attack is that the origin server is not able to identify the client as an attacker because the requests are made from within the network. To mitigate this issue, the edge server can share the client's IP address with the origin server. Hence, the origin server can identify if more than one request is sent through similar IP addresses. The attacker can still mask the real IP address of the attacker.

In this paper [40], various security threats towards CDN are discussed, along with the mechanisms that can be adapted to meet the security requirements. An analysis of the cost of achieving low and high security in CDNs are conducted.

The security that should be provided by the CDN network along with the CIA (Confidentiality, Integrity, and Availability) and AAA (Authentication, Authorization, and Accounting) according to the authors are Authentication, Access Control, and Non-repudiation. However, the most critical parameters considered by the authors are Confidentiality, Integrity, Availability, and Theft of Service Prevention.

The sub-requirements of confidentiality are as follows:

- Confidentiality of the contract between the user and content provider should be ensured.
- The content and the metadata should be protected and is known only by the user and the owner.
- The user data like username, billing address, and SSN should be kept private.
- The resources that are used to deliver the content, such as protocols, IP address space, should be kept confidential.

For a content provider, confidentiality is a business requirement, whereas, for the customers, the privacy of their request history and personal information is of utmost priority.

The following figure shows the mechanisms that can be implemented to ensure confidentiality:

Confidentiality Sub-Requirement	Mechanism
Authentication (C1, C2)	Token Password Biometrics Reputation mechanisms Digital signatures (IP) Address based
Authorization (C2)	Access Control List (File, Network) Time based authorization
Content storage & transport confidentiality (C2)	Encryption (IPSec, SSL/TLS, other) Non-encryption (Secure tunneling) Aggregation of distributed data
User data storage & transport confidentiality (C2, C3)	Encryption (3DES, AES, other) Aggregation of distributed data Software security
Network resources confidentiality (C4)	Firewalls Anonymization NATs

Figure 33. Mechanisms for ensuring confidentiality[40].

The integrity of any content can be ensured when both parties know for sure that the content is not tampered with and the data is intact.

The sub-requirements of Integrity are as follows:

- The integrity of the data has to be maintained by ensuring that the requested content by the user is delivered.
- The source of the data is the provider of the data.
- The integrity of the features and services should be maintained as per the agreement with the user.

The following figure shows the mechanisms that can be implemented to ensure integrity:

Integrity Sub-Requirement	Mechanism
Data integrity (I1)	Message Integrity Code (MIC) Message Authentication Code (MAC)
Source integrity (I2)	Source digital signatures Reputation mechanisms
Service integrity (I3)	Software integrity

Figure 34. Mechanisms for ensuring integrity[40].

The Message Integrity Code (MIC) and Message Authentication Codes (MAC) are mechanisms that are used to ensure the integrity of the data. These are embedded with the information at the sender. A MAC contains a secret key that is known by the sender and the receiver and a checksum to ensure integrity. A MIC contains only a checksum.

The sub-requirements of Availability are as follows:

- The content should be available at the specified QoS as per the agreement.
- The network should be resilient to small scale attacks to ensure the trust of customers.
- The CDN should be resilient to flooding attacks as it is the most common.

The following figure shows the mechanisms that can be implemented to ensure availability:

Availability Sub-Requirement	Mechanism
DoS prevention (A1, A2)	Patching Antivirus IPS/IDS/Firewall
DDoS prevention (A1, A3)	Scrubbers Ingress filtering Traceback Rate Limiting Over-provisioning Residual Risk

Figure 35. Mechanisms to ensure availability in CDN[40].

The quality of service can be affected by the denial of service and distributed denial of service. These security issues can be controlled by applying security patches and antivirus at the server level. Devices like Intrusion Prevention Systems (IPS) and Intrusion Detection Systems (IDS) can be installed to ensure that the network is secure.

Theft of Service is when someone is not allowed to access content without paying for the subscription.

The sub-requirements of Theft of Service prevention are as follows:

- Anyone who accesses the content is authenticated and authorized.
- The users should not be able to share any content if they are not entitled to it.

The following figure shows the mechanisms that can be implemented to ensure availability:

TS Prevention Sub-Requirement	Mechanism
Authentication (TS1) Authorization (TS2)	Similar to authentication mechanisms in confidentiality
DRM (TS3) Monitoring (TS3)	Restricted coding Watermarking

Figure 36. Mechanisms for ensuring protection against theft of service[40].

As per the studies conducted, the authors inferred that content architecture with a hierarchy could implement security features effectively.

Chapter 3 Major components of CDNs

3.1 Origin Server

An origin server is a group of powerful servers responsible for storing all the content of the website. The origin server might store different copies or versions of the contents in various quality levels to support various client requirements based on their device types.

3.2 Edge Server

The edge server is responsible for storing the content from the origin server to reduce the latency at the client-side. The edge servers are planned and placed at various geographical regions according to the density of the client locations. Whenever an edge server receives a request from a client, it checks in its own cache for the content. If it is not available, it is pulled from the origin server and stored in the cache for easy access for further client requests. Some of the files are prefetched if the demand for particular content is high in certain geographical regions. For example, Netflix shows a preview of the top viewed video in the country amongst the suggestions. Trailers of these videos are prefetched and stored so that the traffic congestion is reduced in that area. To successfully prefetch the contents, the user preferences should be learned using analyses of their viewing history of the client. Prefetching techniques using statistical data mining algorithms are used to determine the content to be prefetched. This involves synthesizing and analyzing historical information such as access logs. Since the location of the edge server is directly dependant on the quality of the service, many server location algorithms have been suggested.

3.3 Request Routing Algorithms

A CDN requires to know which the best server is to provide a certain requested content. A request routing algorithm is used to redirect a client request to the edge server that can provide the best service. Request routing can be based on various factors like network latency, network proximity, edge server location, the availability of the content. There are various types of request routing algorithms such as DNS based routing Algorithms, Transport layer request routing, and Application layer request routing.[42]

Often many parameters should be considered to evaluate the redirection technique to choose the best edge server for any user request. However, continuously gathering the related information

through continually monitoring the network is not a viable solution. Hence, other techniques like the active and passive collection of statistics and feedback from the surrogate servers are desirable. Monitoring techniques can be conducted to calculate various networks performance parameters such as loss rates of the links, delays in the queues, and bandwidth available. End-to-end measurement of the network parameters can be done through various tomography algorithms like Estimation Maximisation (EM) algorithm or Minimum Variance Weighted Average (MVWA) algorithm[43]. The parameters that should be essentially calculated include packet loss, delay or latency, average bandwidth, and frame rate in the case of streaming providers. One way to calculate this is by placing probes at the edges near clients to calculate the output at their end. Geographical proximity can be measured by calculating the forward path from the client to the closest surrogate. At the same time, the least loaded surrogate must be chosen to ensure load balancing in the network.

Passive Estimation of the Performance: This can be performed when the data is sent to and from the surrogate servers. The right approach can be the calculation of the data loss from a client to the edge server during TCP/IP communication. These measurements can be implemented using adaptive applications to improve the efficiency further and automate the process.

Active probe checking: This can be achieved through ICMP Echo requests that are sent periodically. However, this is not a desirable solution since it increases the traffic unwantedly and it can cause security alerts in various networks, and it can be blocked[2].

3.3.1 DNS Based Request Routing

A Domain Name Server resolves a URL to the IP address of the server where the content is stored. A DNS-based request routing technique is too ubiquitous and can be applied to all the IP address-based applications irrespective of the transport layer protocol used[44].

A DNS server can give a set of IP addresses of various cache servers that can respond to a particular request. The overall efficiency of a CDN is dependant on its ability to provide the requested content to the client with the least latency by directing the client requests to the most appropriate server. (Using Mobility Support for Request-Routing in IPv6 CDNs). All the client requests are received by the request receiver that identifies the local name server from which the request came from. It analyses the local name server address and identifies the closest edge server that can address the request. The response contains a list of edge servers with the closest

server mentioned first. All the servers are consulted in a round-robin fashion to check if the content is available or if the server is free. If the content is unavailable or is busy, the next server in the lookup response is consulted. The DNS TTL for these responses is kept short (around 20 seconds) to update to the changes in the edge server traffic and load. One of the disadvantages of DNS-based request routing is that the route requests come from the local domain server and not the client. Hence, the selection of the CDN server is based on the local domain server address and not the client address. This may cause proximity issues if the local network of the client is extensive. Also, it might provide the same name lookup for different clients in the same network, causing high-density traffic at the edge server. The DNS lookup response provides the list of the closest edge servers even though it is down or unavailable. This can result in delays or errors during the delivery of the content. Policy-based routing can be inefficient in DNS routing, and it might let unauthorized users access the content. Hence, DNS routing can be ideal for web browsing and can be used for small-scale content streaming services. For extensive scale content streaming services, the DNS routing can deliver discrepancies due to inefficiency in Digital Rights Management and delay in delivering video contents to the clients[44].

Woking of DNS-based request routing: When a client initiates a name lookup in the local DNS server, it should return the address of the surrogate near the client. However, if the local DNS cache does not have the address of the desired edge server, it forwards the request to the DNS root server. This is further forwarded to the DNS authoritative server that further returns the address of the edge servers near the client based on the routing techniques and load balancing mechanisms. The client then requests the content from the edge server[45].

Often, more than one address can be returned by the authoritative DNS server to the client. These addresses can be used to route the content in a round-robin method to ensure load balancing and better reliability.

3.3.2 Transport Layer Request Routing

This method checks the first packet of a client's request to select the cache server. The inspection of the packets provides information about the client's IP address, port information, and transport layer protocol used. The acquired data can be used with user-defined policies to determine the selection of the most appropriate edge server.

3.3.3 Application-Layer Request Routing

This request routing inspects the client requests more deeply than the transport layer header. This provides a better selection control over the edge server due to a better understanding of the requested objects along with the client's IP address. In DNS routing, only the local server address of the server is known.

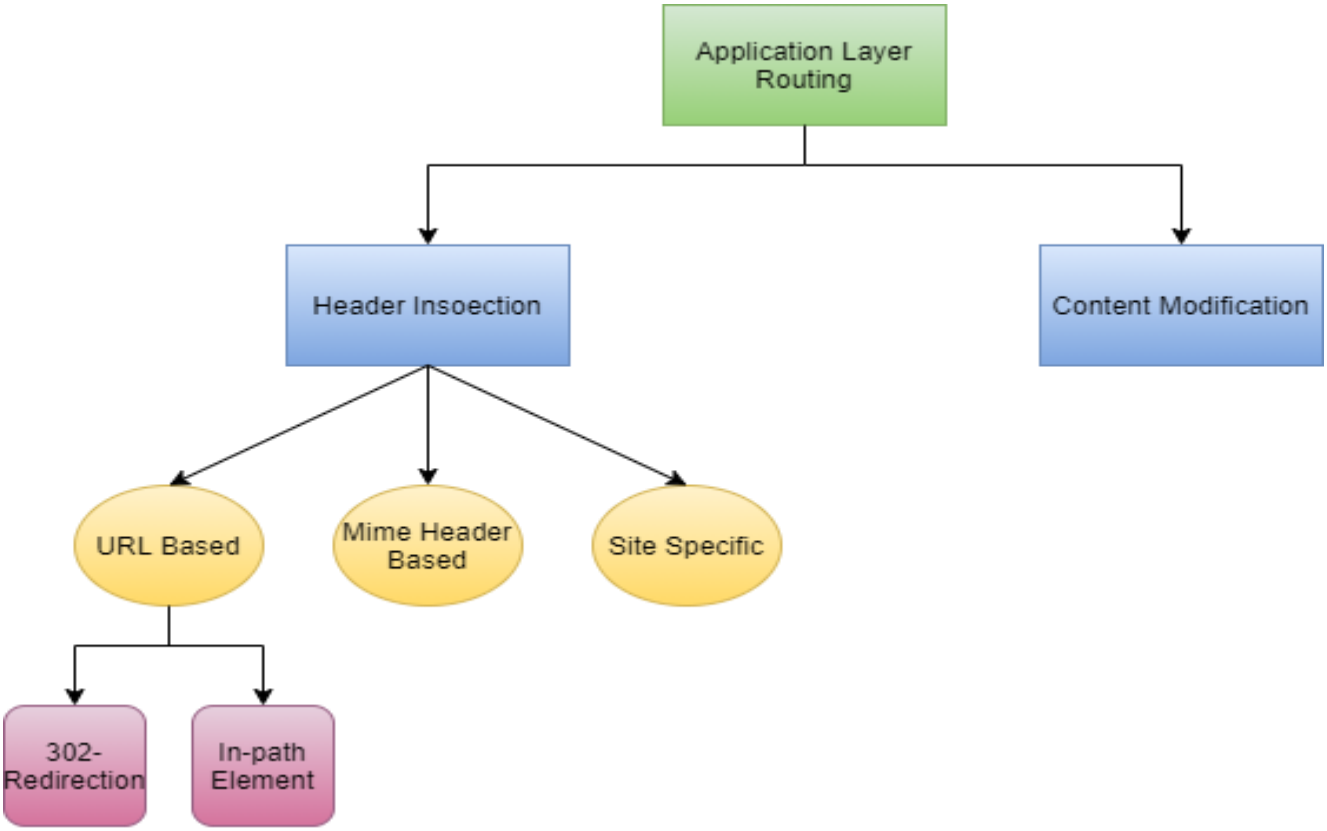


Figure 37. Various Application Layer Request Routing Methods.

Types of application-based request routing include header inspection and content modification. The client's request is redirected to a cache as per the information gained from the client request header in the header inspection method. Meanwhile, a redirect message is sent back to the client. The disadvantage of this technique is the increased latency due to the added overhead of additional messages. Header inspection can be done based on the Universal Resource Locator (URL), mime header, or site details. 302 redirection and in-path element are the different types of URL-based request routings, as shown in the preceding figure.

Content Modification: The content provider directly provides information about the best cache server without the use of a routing device in this method. The decision on the best cache can be based on a per-object request or based on a set of metrics.

Optimizing the Request-Routing Algorithms:

Request routing is the process of finding the best edge server that can serve the client request most efficiently. (ref: On the Optimization of Request Routing for Content Delivery)

Some of the factors that can optimize the request routing are as follows:

- Selection of the redirection methodologies mentioned above as per the application or the webserver.
- We must ensure that the server is not overloaded by checking it with access controls and load balancing.
- Enhanced peering capability to share content with the neighboring servers if there is a cache miss.
- YouTube uses DNS based redirection techniques and finds the optimum server considering the lowest RTT with the POP. Most of the RTT is directly dependant on the geographical distance between the devices.

3.3.4 Load Balancing Techniques

A load balancer distributes traffic across various servers to improve the client experience. A load-balancing algorithm decides which server within a server farm is best suited to address the request. Due to the increased number of internet users, any website or streaming provider should handle thousands of client requests simultaneously. To address this inevitable requirement, more servers are required for each server.[46]

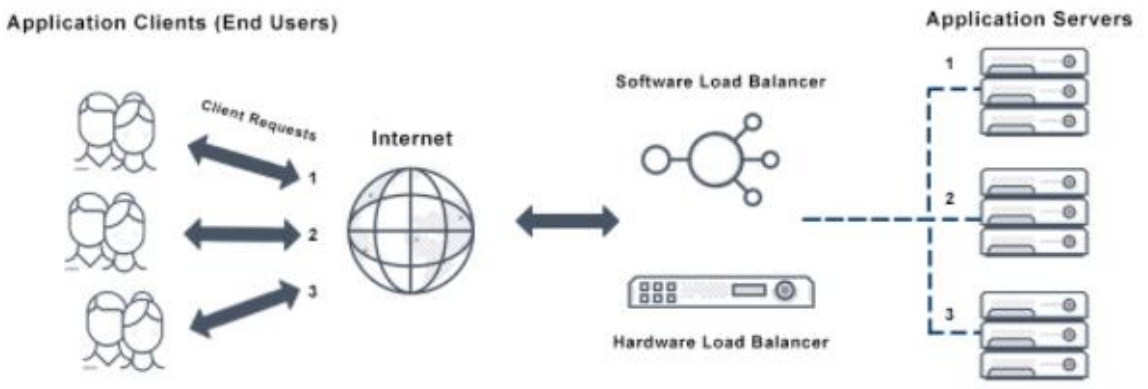


Figure 38. Illustration of Load Balancing[46]

Round-Robin: This is one of the simplest load balancing algorithms and is the most commonly used. Applications are routed to servers in turn.

This load balancing algorithm does not take into consideration the unique characteristics of the application servers. It might not be able to distribute the load among the servers effectively if it does not consider other factors like available bandwidth.

Weighted Round-Robin: This algorithm provides robust load balancing for applications running on applications servers with varying performance specifications. The administrator assigns a weight to each server based on criteria of significance.

Least Connection: This algorithm assigns the client request to the server that has the least number of active connections at the time. Weighted least connection is the algorithm where other server parameters are considered along with the number of active connections.

Weighted Response Time: This load balancer sorts out application servers based on their response time. The response time of the health check is used to calculate the server weights in applications. The server that responds the fastest to a given request is given the next request. Load balancing has the following benefits:

- Increases the scalability of a website by managing the traffic by distributing it to various servers.
- Increased uptime due to multiple numbers of servers in case of a hardware failure.
- Increased flexibility by increasing the number of multiple load-balanced servers.

- Efficiently manages any failures that occur in the hardware by redirecting the traffic to some other servers.

Chapter 4 A distributed Content Delivery Network with advanced edge routers

The diagram shows the suggested CDN architecture for content providers. The major components of the architecture are the origin server, master server, slave server, and ISP proxy cache.

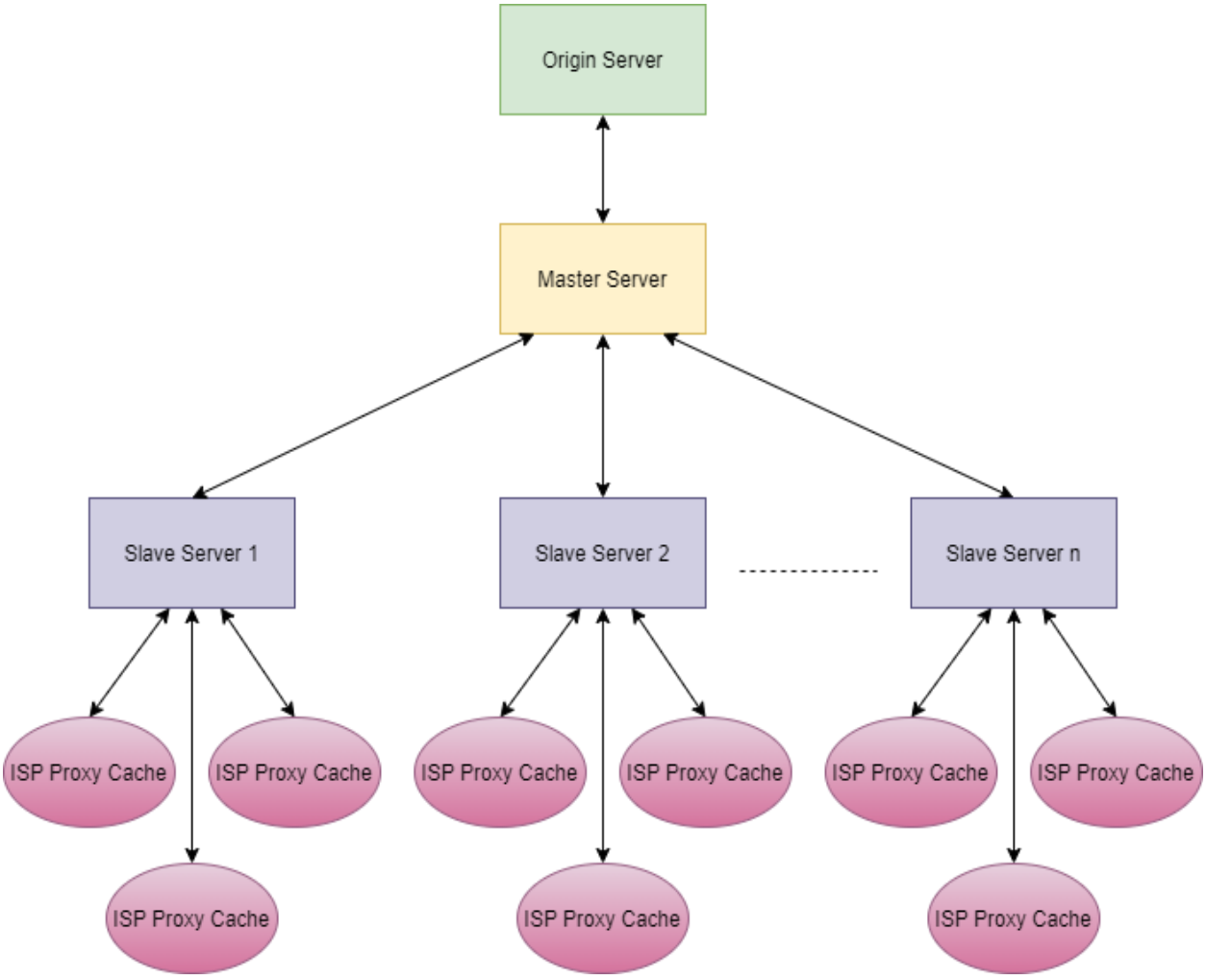


Figure 39. A distributed architecture with advanced edge routers.

4.1 Working of the Architecture

- A request routing technique that provides the optimum edge router is suggested as per the application. DNS-based routing technique can be used to find the IP address of the selected edge server.
- Usage of Proxy Caching at each ISPs can improve the cache hit rate. This is similar to the OpenConnect architecture of Netflix. This can bring the content as close to the users as well as improving the cache hit rate and reduce the RTT considerably. Any content that is stored has an expiration timer along with it. As the timer expires, the content is flushed out immediately. This can help in circulating the latest content in the entire network. The ISP proxies also should maintain a DNS resolver with the domain names and the content that is available at the proxy cache. The proxies need not store the entire video content locally, whereas they can store the initial sub-segments of the video to improve the streaming speed or throughput.
- Before any content request is sent from an ISP, this cache can be checked to see if it is available in this cache. If it is not available, the request can be sent to the edge servers. If the number of requests for particular content is more, this content can be cached. The most viewed content from a region can be prefetched for further improvement of quality of experience.
- The validity of the data is verified from the metadata of the content. The proxy server can deliver the content to the web browser.
- Proxy caching has various advantages, such as reduced bandwidth consumption and network congestion since most of the content is stored closer to the users.
- If the requested content is not present in the ISP proxy cache, the request goes to the nearest local slave server. If the content is present in that server, it is served back to the client.
- If the requested content is not present in the slave server, the request is redirected to the master server.

  The master server consists of a load balancer and a table that lists the location of all the content that is available in the slave servers in its connectivity. If the requested content is present in the table, the master server maps it to the corresponding location and redirects the request to the right slave server.

Also, if the content is present in multiple slave servers, the load balancer selects an optimum slave server for the request.

- If the content location is not present in the master's table directory, the request is sent to the origin server.

4.2 Request Redirection Techniques

The first and most important aspect of a successful CDN architecture is that it requires an efficient Request redistribution algorithm. Since this architecture is mainly for Over the Top (OTT) content providers, it can rely on DNS servers to dynamically redirect the client request to the best server available. The DNS servers can provide a list of servers where the content is available. The DNS technique is very transparent in nature while resolving the domain names. It redirects the clients to the appropriate server without even modifying any specifications in the client's hardware devices or applications. No changes to the protocols are required to adapt to DNS server selection, and it works well across any IP applications irrespective of the transport layer protocols involved. Other approaches like HTTP redirection or any other application layer protocols are too complex to implement effectively.

The best server is selected based on a round-robin fashion while considering parameters like geographic proximity and network proximity. If the routing is just based on the round-robin fashion, it often might not redirect the client to the nearest server. Hence, we suggest that the client send TCP packets to the list of addresses returned by the DNS server. This is similar to the DNS-Proxy method suggested by the authors of [47]. We can probe all the servers that are provided by the DNS server to find the one that provides the least latency. Whenever a request is received by port 53, i.e., the DNS port of the client, it is forwarded to the DNS servers to acquire the IP addresses of the same. It is practically possible to customize it to the servers that we want. The client then sends TCP SYN packets to the HTTP or HTTPS ports to these various servers. It is possible to measure the end-to-end latency from these servers by determining the round-trip time (RTT) based on the time it receives an ACK packet. In order to avoid the possibility of confusing this to a DDoS attack, the client must send a FIN packet back to the server. The CDN server with the lowest end-to-end latency is chosen for serving the client request. This technique also prevents clients from sending a request to unresponsive servers apart from identifying the server with the shortest RTT. Also, if any

DNS resolution contains only one IP address as a response, the client refrains from sending TCP packets to the server.

An analysis conducted in [47] proves that TCP packets take a shorter time than HTTP packets. One of the downsides of this probing mechanism is that it generates much traffic. Nevertheless, the client caches the best server for a particular request so that this is not conducted repeatedly and thus saves time.

A technique should be identified to analyze the available bandwidth of a server. A server that has enough bandwidth allocation capacity with the least RTT can be chosen for serving the client request. Iperf is a command-line-based program that can be used to find the available internet speed of the machine. The server with an available speed and the lowest RTT is chosen to serve the client's request.

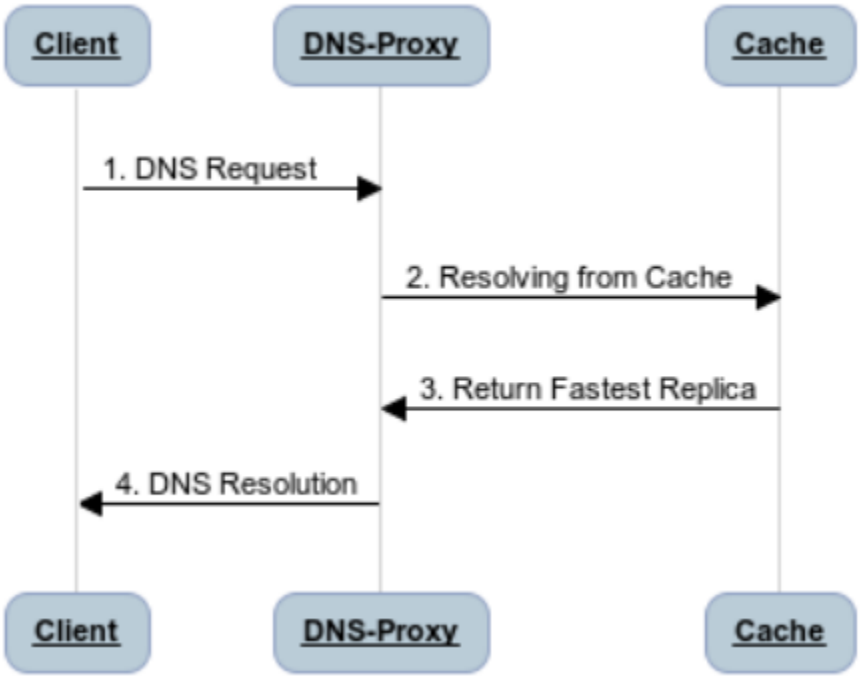


Figure 40. Request Routing in a Cached Domain[47]

4.2.1 **Algorithm for Request Routing**

1. *Start*
2. *DNS request is sent from port 53.*
3. *The list of servers is identified from the DNS response.*
4. *Starts a timer with TTL = 2s.*

i) Sends a TCP SYN packet to each of the servers.

ii) Receives TCP SYN/ACK from the servers.

5. *Probes the server for network availability using the iperf command.*
6. *The server with the best network availability and lowest RTT is chosen.*
7. *A client request is sent to the selected IP address.*

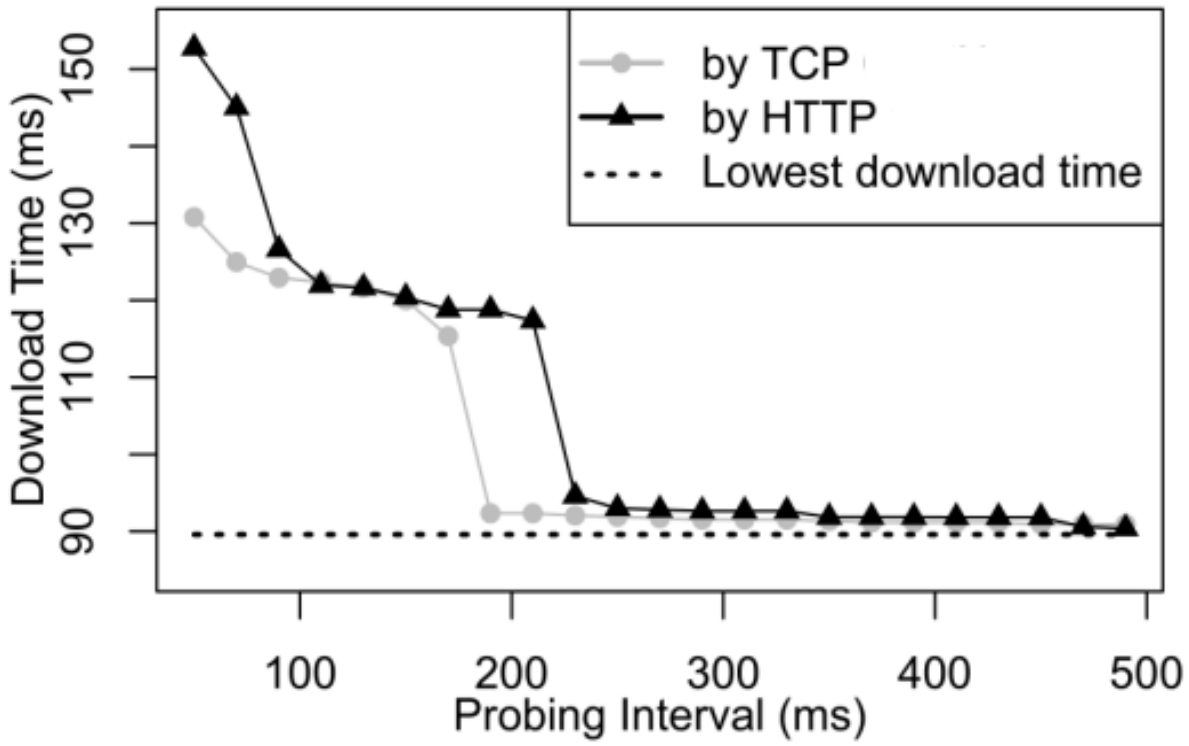


Figure 41. TCP and HTTP for Akamai CDNs[47]

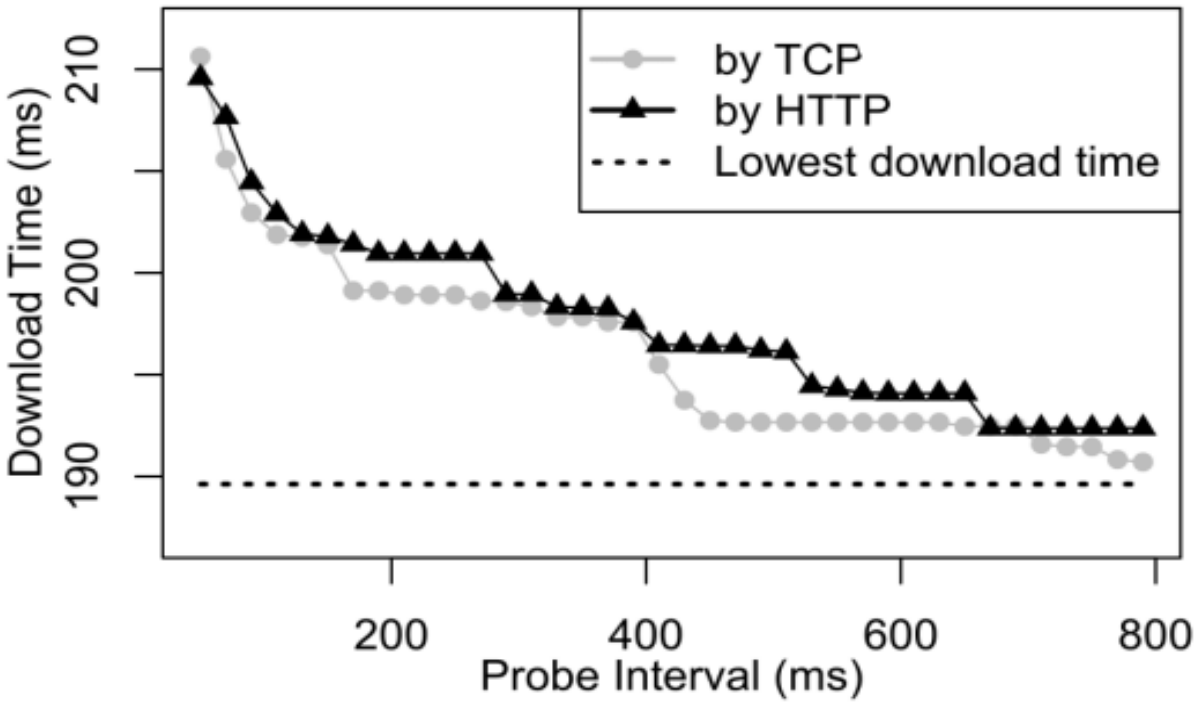


Figure 42. TCP and HTTP for Google CDNs[47]

The time to receive both HTTP and TCP were compared using download time. Both were tested on Google and Akamai CDNs. The x-axis of the images above shows the probing interval after a client request. At the same time, the y-axis shows the average download time from the CDNs.

4.3 Internet Service Provider (ISP) Proxy Caching Mechanism

An ISP Proxy Caching technology is the best option to bring the most frequently accessed content closest to the clients. Caching is one of the best ways to reduce latency and increase the response time [48]. The patterns in which users access the content can help in understanding what content must be cached within the given limited space available at the ISP caches. The caching mechanism of edge servers is more complicated than the ones suggested at the edge servers.

The proposed cache management approach suggests deploying a lower capacity cache inside the local ISP network. This is very much similar to the OpenConnect (OC) architecture of Netflix. Caching algorithms increase the hit ratio by storing the content that is most likely to be requested by the client. Our goal is to create a lightweight algorithm with less computational complexity at the local ISP networks.

A cache replacement algorithm that helps to store the most frequently visited content is favorable. Although many mechanisms such as Most Recently Used (MRU), Most Frequently Used (MFU), Least Recently Used (LRU), and Least Frequently Used (LFU) is available, LFU suits our application the most. An LFU algorithm will help us increase the cache hit ratio by taking maximum advantage of the limited storage space that is available to us. In this algorithm, the least used content is evicted whenever there is a cache overflow, and thus the most accessed content of an area is always available in the cache. Thus, we can ensure a high cache hit ratio. An LFU algorithm can be implemented using a min-heap and hashing algorithm. [49]

The primary operations that are mandatory to implement an LFU cache are:

- Insertion of new content into the cache.
- Maintaining the contents present in the cache by incrementing the demand count.
- Deleting the least frequently accessed content from the cache.

The time complexity of performing these operations is 0(Log n). [49]

4.3.1 Implementing LFU

A min-heap is implemented based on the demand count, and we want to make the algorithm as simple as possible to reduce the computational complexity at the local networks. A content is added to the cache when it is requested by the client that is a part of the ISP local network with a count of 1. Hence, at any point in time, a new content is added to the bottom of the tree. Each node is compared with its parent node to see if it is in the right place. If the parent is larger, the nodes are to be swapped. The number of swap operations is dependent on the number of levels in the tree. Due to the same reason, the time complexity of min-heap is 0(log n), where n is the number of nodes available. The average time- complexity is 0(log 1).

ISP caches do not intend to store a large number of contents; hence these operations are not expected to cost us much time. As the number of views increases, the demand count is incremented, and the object is moved away from the root. Thus, the most popular content will be farthest away from the root. Also, when a content must be removed, the root is removed in this architecture. This creates a major reorganization in the data structure. The

deleted node must be replaced by the least node value, which is generally the child node of the root. [50][49]

4.3.2 Advantages of implementing LFU.

LFU is most suitable in this scenario since caching based on usage patterns can ensure a higher hit ratio. Frequently accessed contents will never be evicted from the LFU algorithm. For example, the Google logo is being accessed by a single user multiple times. Practically, the Google logo will never be evicted from the LFU cache.

Apart from this, proactive caching based on popular content of the local region could be incorporated. Nevertheless, this will improve the cost of setting up this cache at every ISP. Hence, a simple caching method is adapted.

4.3.3 Content Personalization Techniques

The use of content personalization in CDNs enables subscribers to customize and personalize the contents they receive based on their preferences in real-time. This delivery method will make the customer's experience more satisfying. It can be used to improve the service delivery of high-speed broadband service providers or OTT providers in analyzing the methods to increase their customer experience. The basic goal of personalization systems is to deliver relevant content or services even before requested by the user[51].

Personalization involves estimating what the user truly wants to do, foreseeing what the user may be interested in, and presenting content that is relevant to the customers' requirements.

There are various methods by which we can personalize the contents to improve the content delivery. There are methods of personalizing content based on data mining where the log history of the users is used to classify users based on the interests[52]. There are also other methods that store various information of users like user-profiles and other data.

But the real challenge here is how to find the balance between content personalization and user privacy. Maintaining the privacy of the customer while providing them with customized services is a challenge that requires further research and study.

4.4 Slave Server

The edge server consists of various components like a local DNS resolver, a database that maps each of the content to the corresponding data stored, a proactive caching algorithm that learns about the popularity of the contents, a load balancing system that ensures that it prompts the master server at any point of time when it is overloaded with client requests. The challenge of edge servers lies in the complexity of management of these many client requests along with the monitoring of all the devices connected[50].

The database stores the mapping of the real location of a content and the working state of the server that stores the content. This database is updated by the master server every few minutes. This will ensure that the clients are being served with the latest content without any delay.

If the content that is being requested is available at the server, it is easier to map it using the database that provides the metadata and the memory location where it is being stored. The demand counts of each content are varied according to the number of client requests. If it is a cache miss, the request is redirected to the master server, which checks its metadata to find the location of the content and redirects the request to that server.

4.4.1 Implementing Proactive Caching

Proactive caching algorithms can predict the popularity of video and strategically cache it to the respective nodes can help reduce the latency considerably, thus improving the customer experience. If we can predict the popularity of videos according to the geographical region, it can maximize the cache hit ratio and improve the user experience. The ISP of a region carries data about what content is most prevalent in the region, which can be useful in identifying the content that must be cached. [35]

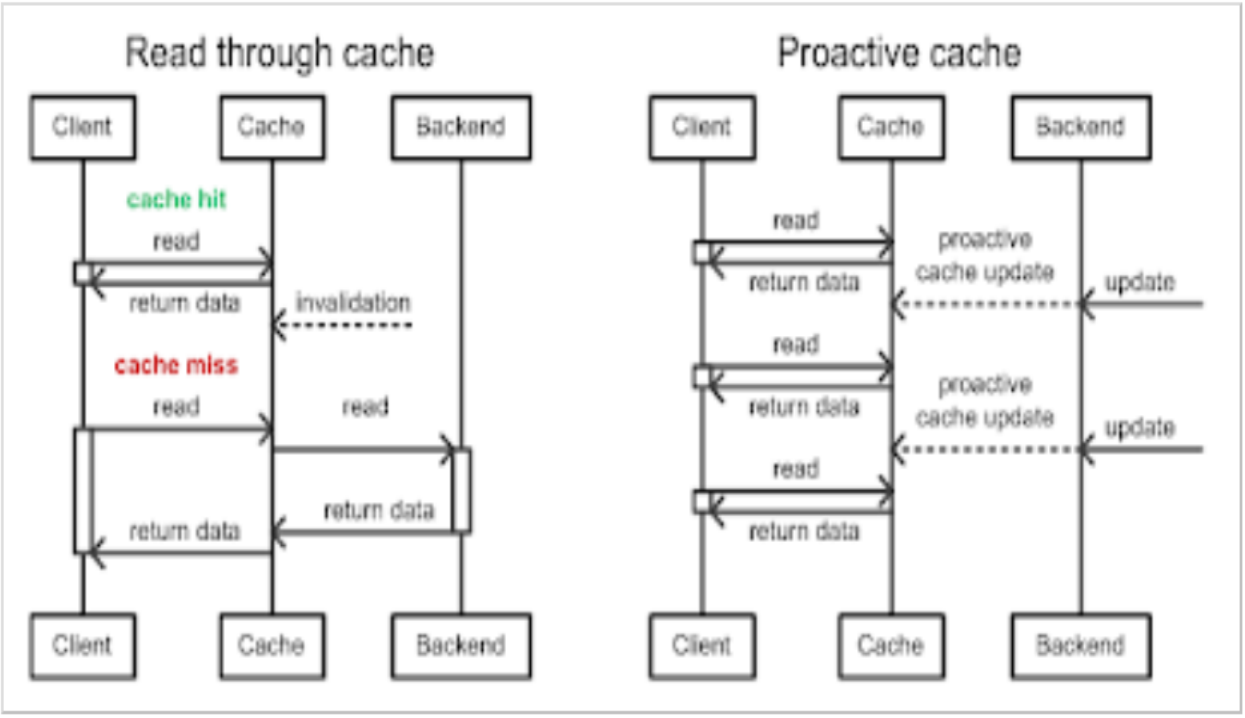


Figure 43. Proactive Caching[50]

We can decide the videos to be cached on various factors as follows:

- **Trend identification**- Various parts of the world will choose to watch or request content that is popular in their region. For example, Netflix shows the option of "Top 10 in Canada" for a subscriber in Canada. Content with regional languages can be cached as per the geography. This will, in fact, improve the user experience. We can identify the trending content in a region based on the ISP logs.
  Statistical analysis can be performed at the origin server to identify the content that is most popular in all regions.
- **Topic Modeling**- This is a method through which we can cluster various contents into a broad group. We can identify common themes for various content using this technique. Latent Dirichlet allocation (LDA) is the most common method that is used to implement topic modeling. [53]
  To evict outdated content from the cache, and LFU algorithm similar to the ISP proxy cache can be used. The entire memory of a cache cannot be used for proactive caching. At least 40% of the cache has to be dedicated to LFU caching to assure maximum hit rate.

4.4.2 Advantages of Proactive Caching

- **Increased Speed:** Once the content is cached, clients experience faster performance because the content they are downloading is stored in a geographic closer location. The underlying internet backbone can impact the speed/latency.
- **Reduced load on the origin server**: Traffic spikes can create havoc on the network. With most requests offloaded to CDN edge servers, responses from origin servers are less likely to degrade during large spikes.
- **Improved security for the origin server**: By masking the origin server and its location, the CDN can protect the origin server from DDoS attacks.

4.5 Master Server

The master server consists of a DNS resolver, a distribution system, a load balancer, and a cache that maps various contents. Every large geographic network region must contain one master server. This can be countries or continents as per the number of users in any region. The master server stores the metadata of all the content that is stored at the lower-level servers. A database can map any of the content that is present in the entire region to the corresponding location.

When a request reaches the master server, it first checks the database to find if it is present in the servers beneath it. If so, the metadata of that content with the location of the specific content will be available on the server. [9] The master server serves the requests that are missed at all the lower levels of servers. It checks the database to identify if the content is present in any of the other edge servers that If the content is available in more than one of the servers, a load balancing algorithm is implemented to find the server that can provide the best service.

The master server must monitor the connection status of all the slave servers in real-time to identify any connection errors of the servers. The load balancing of the servers should be based on the active connections at any point of time. The master server is well adaptable to all the connections and disconnections of the server, and it should have the routing information to various servers. [12]

4.6 Origin Server:

Origin server stores all the content and their metadata; thus, enables content distribution whole through the network. The origin server is updated by the content provider and updates different variations of the files that are compatible with various devices. This process is called transcoding.

All the content is distributed to the edge servers whenever there is a request from the user which cannot be served by edge servers. The origin server can use push and pull methods to distribute the content to the edge servers. A Pull CDN is easier to manage than a Push CDN. A Push CDN pushes new content to the edge servers when there is a modification in the content. A push CDN can cause more traffic in the network than a pull CDN. These two CDNs can be chosen based on the application.

The origin server should maintain logs and other accounting data about the subscribers and client information.

4.6.1 Transcoding Techniques

Once streaming videos became a mainstream industry, it was essential to store videos in various formats to adapt to the various specifications of user devices. The following image shows the need for adaptive streaming clearly:

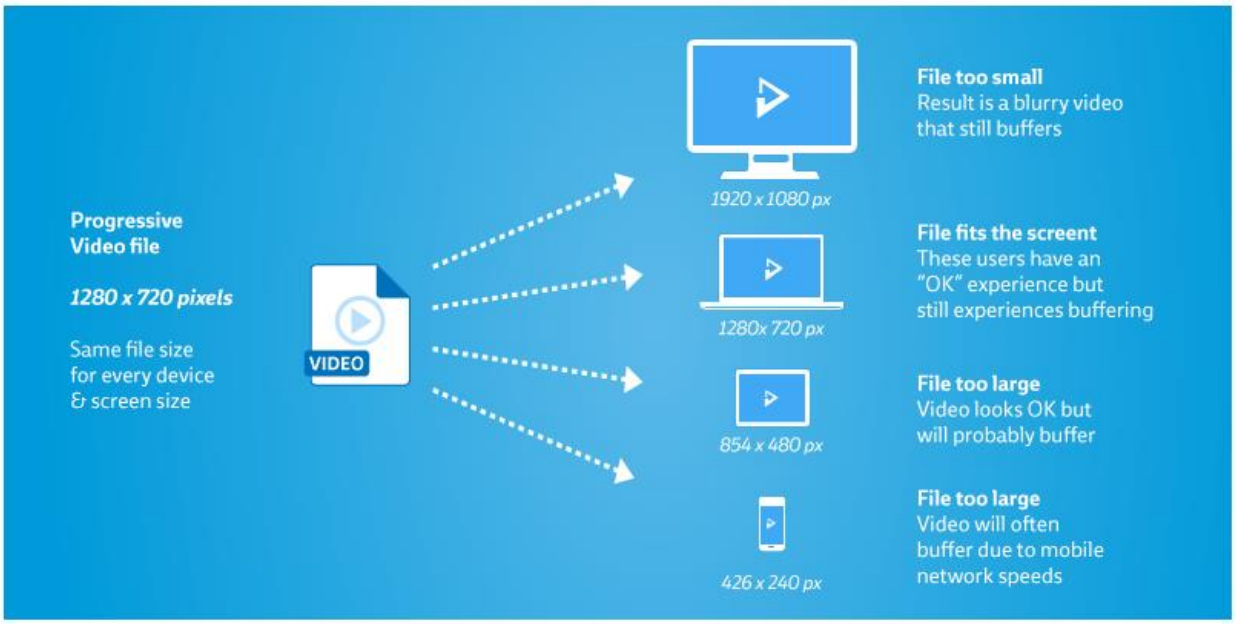


Figure 44. Progressive Video Streaming[54].

If the video stored in the same format is shared to all the various devices, like shown in the preceding image, different users will end up viewing the content either too pixelated or compressed. This will greatly affect the QoS of the viewers. Hence, transcoding is implemented where videos of various bitrates are stored in the servers to address all kinds of end-users with various terminal devices such as a personal computer, mobile phones, and TVs.

However, the current approach to transcoding in the streaming industry is to transcode all videos in all possible bit rates that result in wastage of transcoding resources, storage space, and processing time. Also, one of the concerns is that only a fraction of the transcoded video segments is ever viewed by users.

A streaming service provider must first complete the video transcoding process before the videos it streams are available to its subscribers. To support ABR streaming, a video is segmented into short clips and transcoded into different bit rates. As per the statement of Netflix, a single video should go through 120 transcoding operations before it is available to the users. Transcoding is practically a resource-intensive because it requires significant computing and storage resources.

Following are some of the techniques that can be implemented by the CDNs or content providers to reduce this computational complexity:

- Transcoding can be implemented based on priority. It is a waste of computation and storage resources to transcode the videos that are unpopular. Hence, the most popular videos are transcoded as per the number of client requests. Real-time monitoring is required to implement transcoding this way.
- Also, as per the research, it is known that the initial parts of the videos are viewed more than the second half. Thus, understanding what parts of the video are watched will help more in implementing transcoding more efficiently.

  A similar online transcoding architecture is proposed in [55], where the transcoding is implemented after the requests are made by the clients. The architecture of the proposed methodology is shown below:

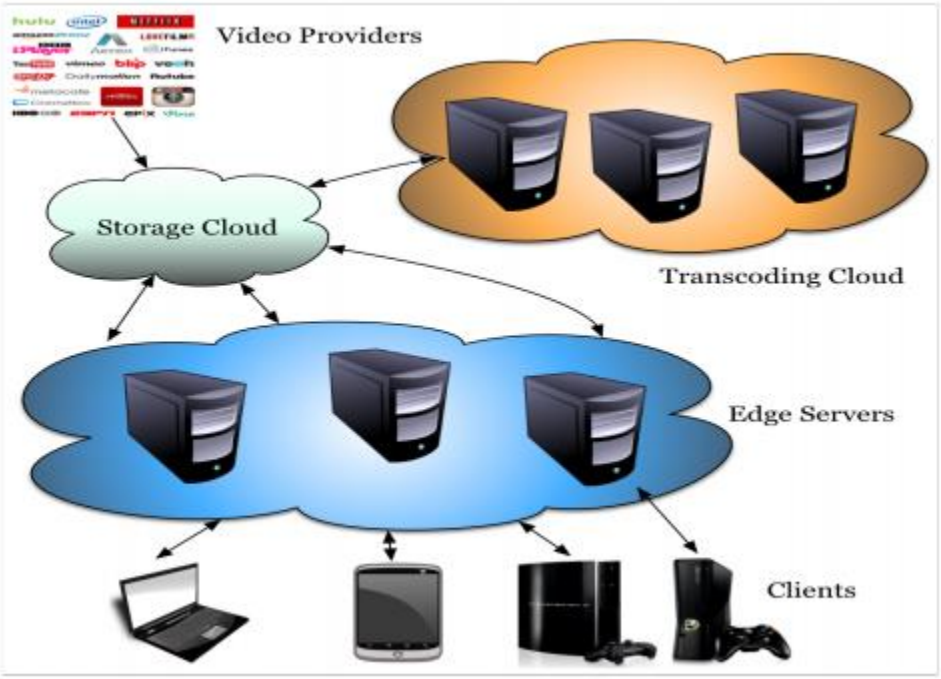


Figure 45. A real-time transcoding architecture[55].

However, as mentioned above, transcoding is an intensive process, and transcoding real-time cannot be considered a viable option.

Chapter 5 Various CDN Providers

5.1 Akamai

When you talk about CDN systems, it is unavoidable to talk about Akamai. They are one of the world's leading providers of Content Delivery Networks (CDN). The business maintains a worldwide network of servers and leases space on these servers to consumers who want their websites to run faster by delivering content from user-friendly locations. It is a network of over 275,000 servers in more than 136 countries that are deployed. These servers collect real-time information about traffic, congestion, and trouble points on approximately 1,500 of the world's networks. Every Akamai server is equipped with proprietary software that processes requests from nearby users using complex algorithms and then serves the requested content.

According to Akamai's whitepaper [41], the four key components that are required for a CDN are:

- **Highly Distributed Architecture**
  The availability of content from nearby servers to users has always been the key to providing the finest results possible. In both the geographical and network topological senses, being close to the end-user eliminates latency and prevents congested peering sites, issues with internet routing, and other middle-mile problems.
  A highly distributed architecture provides the following:
    - Better Caching Performance: A high-capacity CDN architecture is recommended to access the greatest number of users. No single network now accounts for more than 6 percent of the Internet, and the top 30 networks combined account for less than 44 percent. It takes over 600 networks to provide a full coverage of Internet connections. Despite a large number of POPs, centralized CDNs are still not within one hop from most Internet users. The "edge servers" actually sit at the Internet's core, not at the edge; as a result, delivering content to users often requires going through congested peering points and relying on BGP routing. However, since BGP is not a performance-based protocol, it is not always the best route due to its latency or outages. The distance that data needs to travel to reach its end users impacts how much latency is introduced. Because TCP is affected by

latency and packet loss with Slow Transmit, Slow Start, and Retransmit, latency can have an unexpected severe effect on performance for “chatty” web applications and high-quality video. To achieve high levels of performance, a highly distributed system, along with the ability to accurately map users to nearby servers, is essential.

  - Better Dynamic Content performance: Advantages of a highly distributed architecture helps both cacheable content and content that can not be cached due to the RTT distance. An Ecosystem needs to be highly distributed to enable Density of Content. CDNs can speed server-to-server communications using the various route and transport protocol enhancements, such as multiplexing connections or routing around BGP inefficiencies. Optimizations only work within the CDN platform, but they do not apply to the data after it is placed on an end user's device. If we examine real-world application performance, we can find that network performance is important to an application's success.
  - Better Mobile Cellular Performance: Successfully providing mobile cellular coverage is a challenge due to slower speeds and higher levels of network congestion. Deploying edge servers nearer to the user becomes even more important since a packet lost costs more delay. They deployed servers near the gateways to intelligently map users to the best ones - a significant task because the gateways are not always located in the same city, state, or even the same country as the users. Even better performance can be achieved by deploying CDN servers within the core of the mobile network. This further reduces latency to the mobile wireless user.

- **Performance Services:** The software services that are mentioned below that run on the CDN is as important as the infrastructure itself.
  - Web and Mobile Experiences: A decade ago, websites were simple and static, and web page delivery latency was of primary concern. With content caching close to end-users and intelligent server mapping, CDNs could greatly reduce latencies while improving the experience of end-users. Today, Internet connectivity remains relatively important, but the situation is far more complicated, as sites and mobile applications are becoming more dependent on the Internet, more

expansive, richer, dynamic, and more sophisticated with more third party service calls. The sizes of webpages have increased significantly since 2012 due to increased weight for images, JavaScript, CSS, and custom fonts. Even worse, Responsive Web Design sites may suffer from "over-downloading" because the site's design optimizes both desktop and mobile users at the expense of mobile users. It is estimated that in 2012, there were roughly 4,000 different mobile devices in the marketplace, and in 2015, there were more than 24,000. There are many different form factors, browsers, operating systems, and device capabilities to support. Delivering an ever-evolving and engaging customer experience to every user every time is becoming more and more complex every day. These services include advanced caching, dynamic site acceleration, front end optimization, and mobile app acceleration.

- Advanced Caching Features: While caching is a basic feature in CDN, advanced caching capabilities allow a CDN to cache more content even as sites become increasingly dynamic. Various CDN's support the ability to set TTLs and overlook various cache-control headers, but the level of functionality in cache rules and cache keys available varies. An advanced Content Delivery Network also has strong cache-control engines to service a broad range of cache behaviors through flexible, nested rules with sophisticated pattern matching. It should also have the ability to key off various request features, including cookie values, query-string, geo-location, partial URL, HTTP header values, or any combination of all of the above. This enables caching of dynamic content such as search results, API calls, product category pages, content targeted to different audience segments.
- Efficient Streaming Technology: Video delivery most commonly occurs over TCP-based HTTP, using an adaptive bitrate technology. However, as the demands for video quality continue to rise, limitations in the current state of TCP-based HTTP become apparent. Designed for reliability, TCP connections pose significant overhead when limited throughput occurs. With TCP, packet loss can lead to retransmission and cause a substantial delay that can affect the quality of the stream. UDP is designed for real-time communications and allows packets to

be dropped in case of congestion, so that stream latency is unaffected. An emerging technology combines UDP with forwarding error correction methodologies in order to get the best of both the worlds of reliability and speed, even over congested Internet routes.

- Advanced Security Features: As the volume of high-value transactions on the Internet grows, so does the threat of widespread attacks and the high expense of preventing them. In 2015, businesses around the world collectively lost an average of $7.7 million due to cybercrime[56], with U.S. businesses accounting for the largest losses annually, averaging $15.2 million.
    - Defense against DDoS attacks: By cultivating Internet-wide visibility, a highly distributed content delivery network can provide protection against even the largest of distributed denial of service attacks without affecting the performance of legitimate users. Application layer attacks are immediately dropped by the edge servers, while network layer attacks (like SYN attacks) are mitigated using signature-based filters, IP blacklists/whitelists, and adaptive rate controls. CDNs can essentially provide resilient DNS operations, granting faster DNS resolutions and protecting against DoS attacks.
    - Cloud Security: Ultimately, the use of collective intelligence and data will prove to be one of the most important weapons in the ongoing battle against hacking. Big data - which comes from across the Internet - can help identify trends, malicious actors, and other indicators, both in real-time and overtime. A CDN with massive data about various client logs can easily use these analytics to mitigate the attacks that are prevalent as well as predict and prevent the attacks before even they occur.

      Akamai provides real-time client reputation scoring capabilities, such as identifying bad IP addresses and assigning them a vulnerability index that predicts the likelihood of their participation in different types of attacks based on their past behavior. To mitigate malicious use, the vulnerability index calculation considers the severity and scale of attacks, as well as how similar clients are performing attacks. By using the real-time client reputation scores, organizations can determine the best way to handle the request.

- Management of Bots: Bots now comprise a significant amount of traffic, up to 40-60 percent of some organizations' online traffic. Sometimes bots do important business tasks, while other times, they carry out cyberattacks, steal website data, scan for vulnerabilities, perpetrate fraud, or otherwise cause harm. In many cases, bot activities decrease the QoS for human visitors. Unfortunately, effective bot management is far more complex than simply accepting or denying their requests wholesale. Organizations need the ability to identify and treat a scraper bot differently than a search-engine bot. This is another application for big-data cloud security intelligence where the potential of combining lists of known bots with analysis of bot behavior to assist in the classification of unknown bots can be realized. Customers can employ different strategies such as caching of content, serving of alternative content, sending the request to a different origin, delaying the request, or denying it altogether depending on the kind of bot it is.

An extensive study of the Akamai network is conducted by [57], where they tried to learn the working of akamai networks and identified the paths taken by the devices using PlanetLab.

Working of Akamai Network

We know that Akamai's network measurement, path selection, and cache distribution algorithms are proprietary and private; it is relatively easy to identify their redirecting mechanisms.

- Working of DNS: Akamai utilizes a hierarchy of DNS servers to redirect the Web client's request to the IP address of a nearby edge server. The translation of DNS names takes place as follows. First, the end-user contacts the organization to request a translation to get a web page from a customer. The customer's DNS server points to a canonical name (CNAME) pointing to a host in Akamai's network. A CNAME entry allows a DNS server to redirect queries to a different domain. Next, a hierarchy of DNS servers responds to the DNS request with the information to best find the customer and the content to return. In summary, the process returns two IP addresses of edge servers that are expected to provide the best performance to the end-user.

  The following image shows the working of an Akamai DNS resolution:

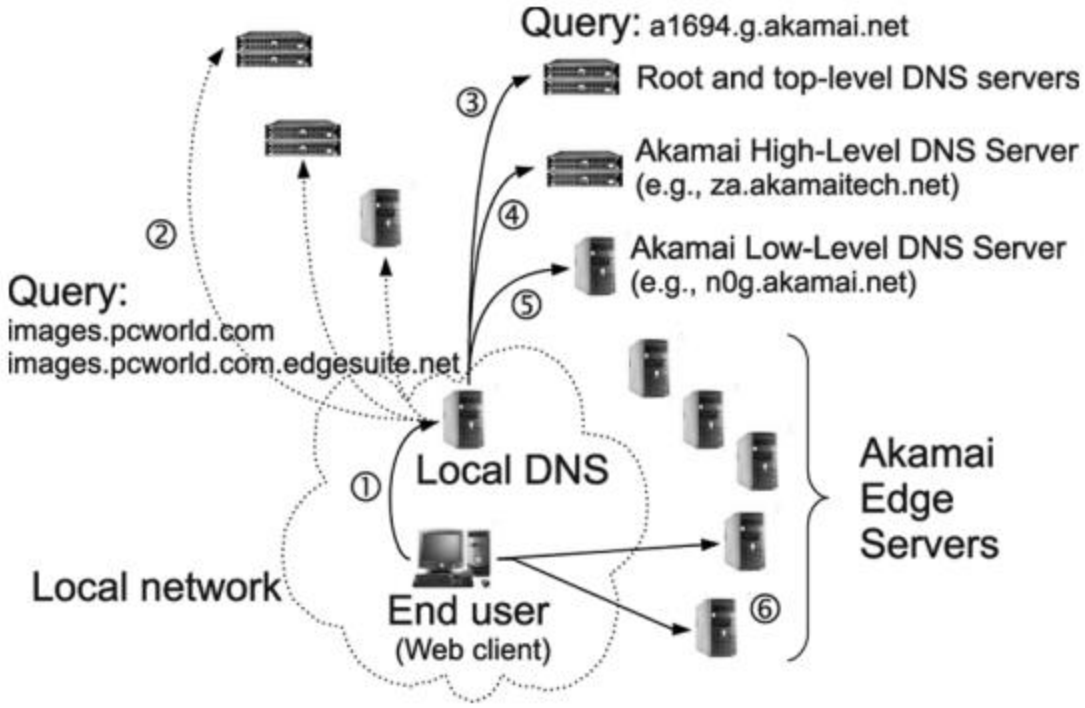


Figure 46. Akamai DNS Resolution.[57]

To maintain the accuracy of DNS records, Akamai's servers set relatively small TTL values for their entries. A TTL of 20 seconds is set for an edge router's DNS entry. This means that the Local DNS server should set up a new DNS reply every 20 seconds.

By simultaneous monitoring of network paths and frequently refreshed DNS server tables, it is inferred that Akamai-server redirections significantly performed well given the network conditions at any time. For instance, more than 70 percent of paths chosen by Akamai are among the best 10 percent of the measured network paths. Using CDN services that analyze the network and deploy servers on a global scale can significantly outperform traditional Web with load balancing servers.

5.2 Limelight

Limelight[57] is a global content delivery network (CDN) that offers services such as cloud storage and media delivery. Some typical products offered by Limelight include "Deep Insight,"

which provides customers with analytical data that can help them make good decisions. The Limelight orchestrate is one of the world's largest channel of distribution. The Limelight provides cloud storage, content restriction, security, traffic flow, and content placement devices. The cloud part of the e-learning system takes advantage of cloud-based storage services.

The network infrastructure of Limelight is spread around the globe with a QoS enabled network of about 123 points-of-presence. Each Limelight PoP has a high density of fast servers with SSDs (Solid State Disks), which results in greater origin offload, better performance, and lower bandwidth costs for Limelight customers.

The overview of Limelight Orchestrate is given below:

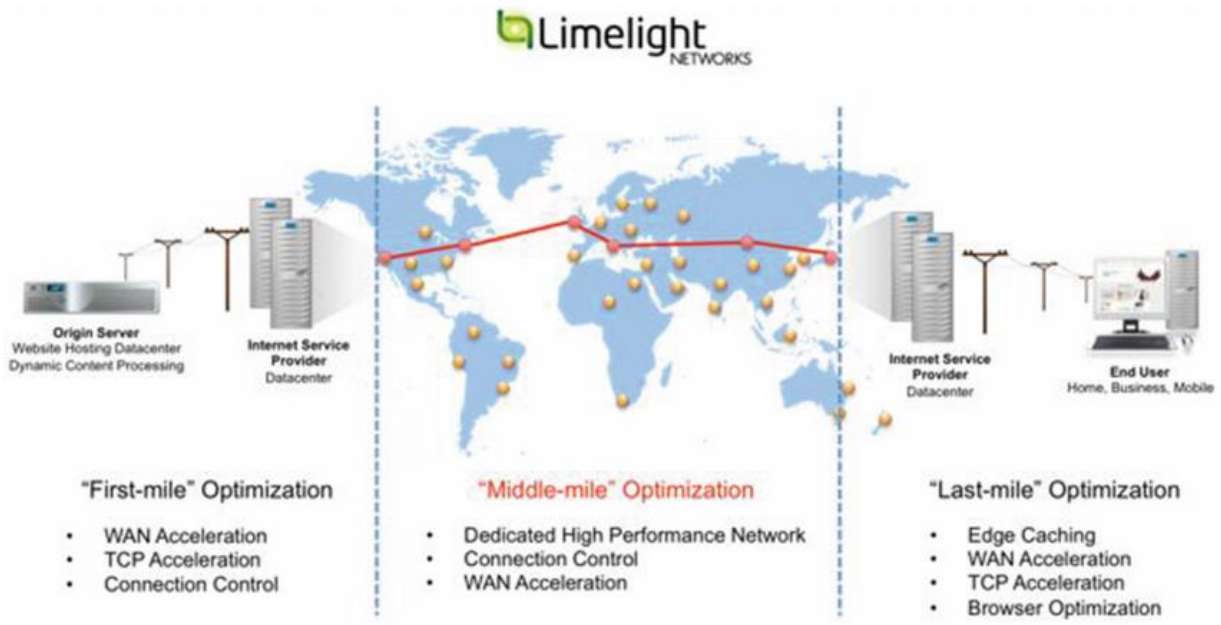


Figure 47. Limelight Orchestrate Overview[58]

Chapter 6 Case Study on Netflix

Netflix is one of the world's leading entertainment services with over 200 million paid memberships in over 190 countries, that provides TV series, documentaries, and feature films across a wide variety of genres and languages.

Netflix is based on a microservices architecture where resources and data are not shared with any other applications unless using an Application Program Interface (API). Netflix claims to have more than 600 microservices that are working towards various smaller tasks like new user registration, authentication, and customization. This architecture enabled Netflix to always work on the services and provided the flexibility to always rollback if it did not work successfully. The main benefits of having a microservices architecture are scalability, availability, and increased speed. Netflix stores all the movies into multiple formats and quality to service a broader audience. The most striking thing behind the success story of Netflix is its very own CDN, OpenConnect (OC). This CDN server is connected to the servers of the Internet Service Providers, which will eventually download the Netflix library for a particular region. A single client will be connected to multiple OC servers at the same region, and the OC that has the lowest latency serves the client. Netflix has its adaptive bitrate algorithms and CDN selection algorithms to adhere to the changing network conditions.

Designing a large-scale, fast-growing, high-availability streaming video platform with scalability challenges is difficult. The company's service used to be primarily accessed from its own data center. Netflix has begun relying on cloud computing services, content distribution networks, and other public computing systems lately. AWS, Amazon SimpleDB, S3, and Cassandra are used for file storage now. Streaming video is delivered to customers through multiple CDNs, while UltraDNS is used as its authoritative DNS servers. Netflix uses Microsoft Silverlight to render video content on the PC. The result is that Netflix manages to deploy a wide-ranging video delivery service without owning most of the infrastructure.

The authors of [59] conducted an in-depth analysis of the architecture of the Netflix platform. They identified how various video manifest files analyze how client capabilities and location

affected other factors like video quality and CDN ranking. A study on how Netflix switched CDNs while the bandwidth deteriorates over time.

The architecture was analyzed using traffic monitoring and WHOIS lookup.

The main architecture of Netflix is as follows:

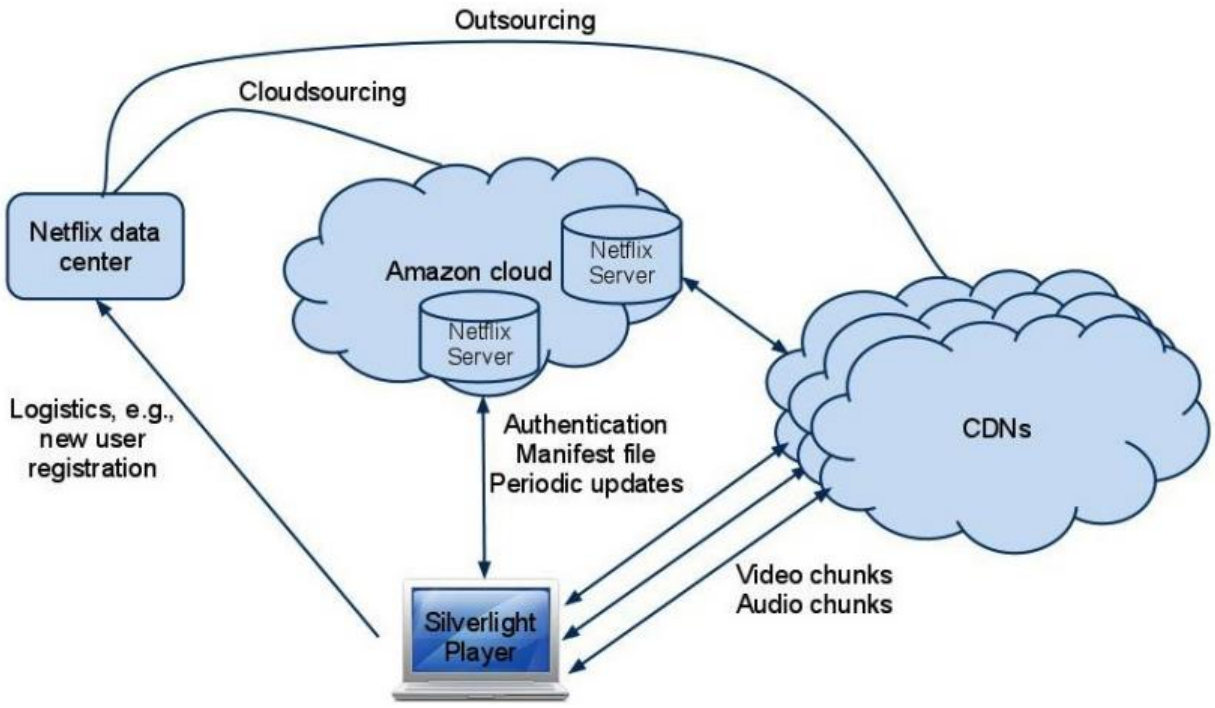


Figure 48. Bird's eye view of Netflix architecture[59]

As shown in the figure, Netflix architecture consists of 5 key components:

- Netflix Data Centres - Netflix uses its own IP addresses to redirect to www.netflix.com. This server is primarily responsible for handling and maintaining the registration and storage of user payment information. The requests are then redirected to movies.netflix.com or signup.netflix.com depending on if the user is logged in or not. This server does not contact the clients for other regular streaming.
- Amazon Cloud - All the network websites except for netfix.com, like agmoviecontrol.netflix.com, are stored in the Amazon cloud[60]. All the other key functions of a cloud service are performed in Amazon's cloud, including content storage, log recording/analysis, CDN routing, user sign-in, and mobile device support.

- CDNs - Netflix distributes videos throughout multiple CDNs to deliver the content to the end-user. The encoded videos with Digital Rights Management (DRM) are streamed from Amazon's web-services and copied to a CDN. Netflix includes three major CDNs as Akamai, LimeLight, and Level-3. All three CDNs store the same encoded content to provide uninterrupted service.
- OpenConnect Appliances (OCAs) – These devices are the building blocks of Netflix that stores encoded videos nearest to the clients and serves the request using HTTP or HTTPS protocol to the end-user devices like mobile phones, laptops, and set-top boxes. The aim of installing the OCAs is to empower ISPs to provide efficient Netflix service to their customers. These OCAs are kept as close to the customers to reduce the distance the requests have to travel during the playback of the video.
- Streaming Players - Netflix downloads, decodes, and plays movies using Silverlight. The Silverlight run-time environment is available for most of the web browsers. Netflix uses Dynamic Streaming over HTTP (DASH) to stream efficiently. Every video is available in varying quality and is divided into small segments of few seconds. A bandwidth determination algorithm is run to identify the quality of the segment that can be sent without interruption. DASH allows you to switch between the quality of the videos without much latency.

The working of Netflix is as follows:

- When the user clicks “Play,” the browser downloads Silverlight, and it starts playing the content. This small application is downloaded during every playback.
- Each video is played based on the set of instructions stored on the manifest file that is downloaded by the app. The manifest file contains DASH metadata to start adaptive video streaming. The manifest file varies from user to user, depending on the device details and network details.
- The manifest file contains various information like the list of CDNs, video URLs, etc.
- Netflix uses three CDNs, and each one is ranked to display the preferred one.
  A part of the manifest file is shown below that contains the ranks of various CDNs:

Figure 49. CDN Ranks in Manifest File[59]

- Netflix incorporates simple trick play techniques by incorporating basic features like forwarding, back, play, and pause. These are achieved by storing a set of images of various timestamps.
- Video segments are downloaded more frequently during the beginning of the video to build a sufficient buffer. Later, the download is implemented in various periodic intervals. The interval between two downloads is noticed to be two seconds.
- Often, feedbacks are sent to the server about the user experience using periodic keepalive messages and updates.

- The CDN ranking remains the same for a user account irrespective of the movie, location, and device type. Also, the CDN rankings are different for the same movie, device, and location as well as it remains unchanged over several days. The manifest file sent for a user indicating the format of the video it can play is based on the device specifications. The bitrate at which videos are sent also varies depending on the bandwidth capabilities of the user server connection. Another interesting fact noticed[59] is that the user tends to stay with the same CDN even if the bandwidth is reduced considerably till a point where it can no longer support the video streaming. The client does not change the CDN it uses even if it has to degrade the quality of the streaming.

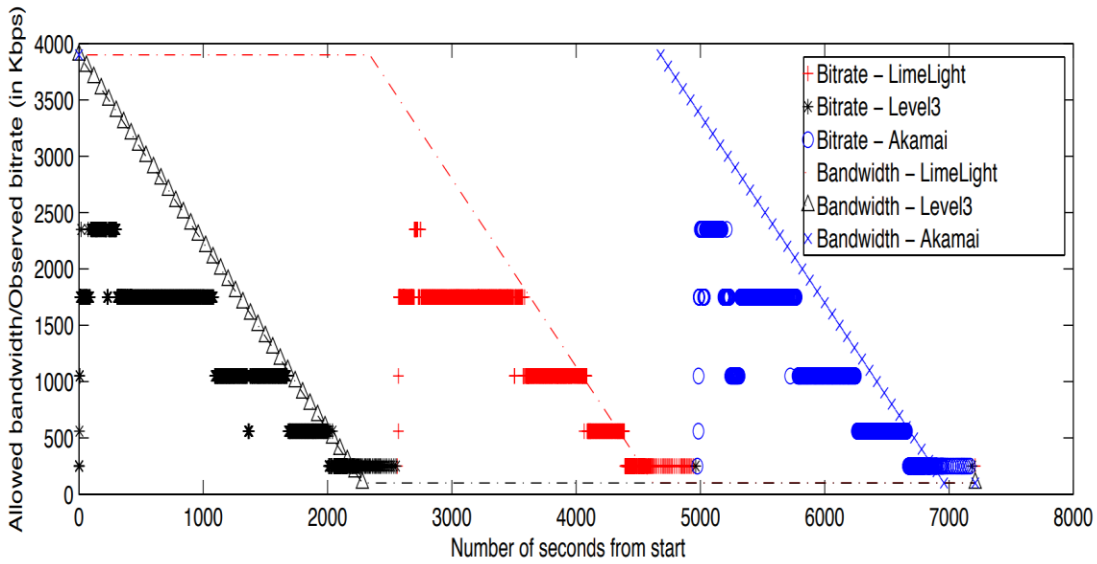


Figure 50. CDN Switching Analysis[59]

Chapter 7 Data Processing in Streaming: Trends and Techniques

According to the Market Analysis Report [61], global streaming is expected to achieve a growth rate of 21 percent from 2021 to 2028. The increasing demand for streaming applications, especially during this pandemic, calls for advanced technologies in the streaming domain.

Streaming services can be divided into various categories based on the following characteristics:

- Based on the way of communication –
    - Point-to-point: This mode of communication involves one-to-one communication. Examples of this kind of communication are video calls and one to one video streaming services. The main feature of this communication is if there is a channel for feedback or not. With a feedback channel, the sender can adapt according to the reception.
    - Multicast: This is the kind of communication where data is transmitted from one sender to many in a group distribution. IP multicast is an example of this kind of communication where data is sent in a single transmission.
    - Broadcast: The most popular among these is undoubtedly broadcast communication, where data is being transmitted from one to many. Broadcast is an efficient way to deliver information to a large audience that can be received at the same time.
- Based on the time of encoding –
    - Real-time encoding: Audios or videos can be captured in real-time and be encoded before transmission. Interactive applications like videoconferencing or online games are examples of this communication. Real-time encoding can be used in other instances where the communication is interactive. For instance, a large sports event broadcasted live.
    - Prior encoding: In various Over the Top (OTT) services, the content is encoded and stored previously. The content is distributed on-demand basis. The videos can be stored in a CDN or by the content creator.

- Based on the kind of interactions –
    - Interactive: The applications that have a time-constraint like online games and videoconferencing belongs to interactive applications. Specifically, the information has a specific time limit, and if it reaches late, it is obsolete. This is equivalent to the maximum acceptable end-to-end latency of the transmitted information of a 60-second transmission. The maximum latency for a mobile application, such as a smartphone, is 150ms.
    - Non-interactive: Non-Interactive applications have comparatively lower latency requirements, typically limited to seconds. Examples of non-interactive applications include transmissions of popular events, which may not have the tightest latency constraints.

7.1 Managing the Bandwidth Constraints

When the demand for streaming applications increased rapidly, the requirement of storing and manipulating these videos increased significantly. Also, transferring this raw format over the internet costs a lot of bandwidth. Hence, to store this tremendous quantity of videos or transmit it over the internet, techniques that are used are Compression, Multiple File Switching, and Scalable Compression.

7.1.1 Compression Techniques

Video compression is the process of encoding a video file to reduce the space used, and it is easier to transmit over the network or the Internet. Compression is the technique through which the number of bits are required to represent a digital image or video while managing the required quality. Hence, the more the compression ratio, the lesser is the requirement of the bandwidth[62].

The compression ratio is given by:

$$Compression\ Ratio\ = \frac{Size\ of\ the\ original\ video}{Size\ of\ the\ compressed\ video}$$

To get better compression ratios, some of the pixels are predicted based on other pixels, like a spatial prediction or temporal prediction. The prediction of pixels through the pixels on the same frame is called spatial prediction, whereas temporal prediction is where pixels are predicted from the previous frames.

This is possible due to the redundancy data present that can be removed with the ability to reverse the pixels. For example, consecutive frames of a video sequence typically contain the same objects and are similar in appearance. When there is spatial redundancy, it is because nearby images have similar amplitudes to stationary backgrounds. The Red, Green, and Blue colors of a given pixel are often strongly correlated.

The small changes in the video that are unnoticeable in the frames are not encoded in the video and thus are lost from the video. Hence, this is a lossy compression. The compression ratios tend to be higher in this kind of video. At the same time, there are some other techniques that are retractable or reversible. The reversed video should be identical to the initial raw video. This is a major requirement in high-quality applications. Such compression is called lossless compression.

The one main feature that is used in almost all video coding algorithms is Discrete Cosine Transform (DCT). Consider the technique, JPEG, that has been designed for the purpose of exploiting the spatial and color redundancy inherent in a single still image. Neighboring pixels in images are often highly representative of each other, while natural images have much of their energy spread across the lower frequency range. JPEG utilizes these primitives in the computation of the 2-dimensional DCT for each block. It breaks a large image into small pieces so that pixels within a small block are more likely to each other than the pixels within a larger block. The DCT compresses all the signal's energy into only a small portion of the signal's coefficients, where this small portion of the signal's coefficients is sufficient to reconstruct a true image. For each 8x8 block of DCT coefficients, several different techniques are applied, including zigzag scanning, run-length coding, and Huffman coding, which produce a compressed bitstream.

Following is an image that shows the spatial dependencies on a frame:

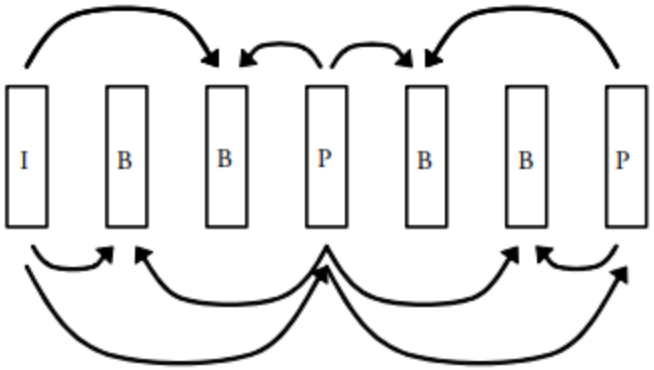


Figure 51. Spatial dependencies on a frame[63].

Some of the video compression techniques are:

- **H.261**

  The first video compression standard to gain widespread acceptance was the ITU H.261, which was designed for videoconferencing over the integrated services digital network (ISDN). H.261 was adopted as a standard in 1990. It was designed to operate at p = 1,2, ..., 30 multiples of the baseline ISDN data rate, or p x 64 kb/s. In 1993, the ITU-T initiated a standardization effort with the primary goal of videotelephony over the public switched telephone network (PSTN) (conventional analog telephone lines), where the total available data rate is only about 33.6 kb/s. The video compression portion of the standard is H.263, and its first phase was adopted in 1996. An enhanced H.263, H.263 Version 2 (V2), was finalized in 1997, and a completely new algorithm, originally referred to as H.26L, is currently being finalized as H.264/AVC.

- **MPEG**

  The Moving Pictures Group (MPEG) was established by the ISO for standards in video encoding and compression of audio. The resulting framework, commonly known as MPEG-1, attained the ability to provide even better video and sound than VHS. In a second phase, MPEG-2 is developed for digital television, for higher bit rates, and for higher resolutions. A third standard, to be called MPEG-3, was originally envisioned for higher bit-rate applications such as HDTV, but it was recognized that those applications could also be addressed within the context of MPEG-2; thus, those goals were transferred to the MPEG-2 standard. The third phase of work is designed to provide improved compression efficiency and error resilience features, as well as increased functionality,

including object-based processing, integration of both natural and synthetic content, content-based interactivity.

Video coding standards	Year developed	Publisher	Primary Intended Applications	Bit rate
H.261	1990	ITU	Video telephony and teleconferencing over ISDN	$p \times 64$ kb sec^{-1}
MPEG-1	1991	ISO/IEC	Video on digital storage media (CDROM)	1.5 Mb sec^{-1}
MPEG-2	1994	ISO/IEC	Digital television	2-20 Mb sec^{-1}
H.263	1996	ITU	Video telephony over PSTN	33.6 kb sec^{-1} and up
MPEG-4	1998	ISO/IEC	Object-based coding, synthetic content, interactivity, video streaming	Variable
MPEG-7	2001	ISO/IEC	Real-time and non-real time applications, to tag the contents and events of video streams for more intelligent processing in video management software or video analytics applications	Variable
H.264/AVC	2003	ITU-T/ ISO/IEC	Improved video compression	10's to 100's of kb sec^{-1}

Figure 52. Comparison of various video compression techniques[62]

7.1.2 Multiple File Switching

Multi-rate switching is a commonly used technique to save bandwidth where multiple copies of the same content are produced at various bitrates.

Early streaming media systems coded the exact same content at a few strategic media rates targeted for common connection speeds and allowed the user to choose the appropriate speed at the start of the session. However, in these early systems, the rate of video and audio output could only be set once at the beginning of the session. But the multi-rate switching enables dynamic rate switching within a single streaming media session. This change during a session enables better adaptation to longer-term fluctuations in available bandwidth than can be achieved by simply the client buffer.

7.1.3 Transcoding

A video transcoder selectively drops information from a high bit rate video so that a lower bitrate one can be created. Common standards for transcoding include re-quantization and spatial downscaling. Transcoding techniques vary in their ability to reduce bitrates and will vary according to the amount of bit-rate reduction necessary[63].

- Re-quantization: The first way to transcode video to a lower bitrate is by increasing the quantization step size.
- Spatial Downscaling: In spatial downscaling, we apply a lower resolution to the video, e.g., from X/Y pixels to X/2 by Y/2.

7.2 Error control Techniques

One of the major issues in video transmission is the losses involved. Losses can deteriorate the video quality to a large extent, and if not handled well, the end-product might be disastrous**.** A video streaming system is designed to prevent errors from causing data transmission errors. Following are the different types of error-correction techniques incorporated in video streaming:

- **Retransmission Technique:** In this technique, the sender notifies the receiver if a packet is received or not, and this enables the sender to resend the missing packet. This approach easily utilizes the available bandwidth by only resending lost packets, and it is easily modified to account for changes in channel conditions. The disadvantage of this technique is that this induces an additional delay in communication. It also requires an additional channel for feedback.
- **Forward Error Correction (FEC) Technique:** The purpose of FEC is to create redundant information and to recover from errors using this information. To overcome packet losses in a packet network, one typically uses block codes that take K packets and output N–K copies of the packets. In cases where any K of the N packets are received correctly, the original data could be reconstructed. On the other hand, adding additional information increases the available bandwidth through a multiple of N.

7.3 Adaptive Bitrate Streaming

Adaptive Streaming (AS) allows content providers to customize their content to deliver to various types of devices. It divides a media file into fragments, called a chunk or segment. Each chunk is encoded at multiple rates to satisfy various device and network requirements.

All video chunks are stored together in the video library and are organized using the Media Presentation Description (MPD) file format. An MPD is an XML metadata file that provides detailed information about a chunk. It calculates the rate by which video data is delivered and the duration of each frame in seconds.

Chapter 8 Simulation Tools

The only way we can analyze the performance of such huge CDNs is by testing it for its throughput and how it behaves at high traffic and low traffic. Such intensive and huge devices cannot be built just for testing purposes and work on a trial-and-error method. Hence, we have various simulators that can be used to realize through network modeling.

Several methods through which network protocols can be investigated, and network performance can be evaluated are:

- Mathematical modeling and analysis.
- A time or event-based simulation.
- Hybrid simulation (both analysis and simulation)
- Emulation using testbeds.

Mathematical modeling and statistical analysis can provide answers to difficult problems quickly. The method is faster than simulation but inaccurate in some areas. Analytical models are not readily available to cover all instances. Some of these models are inaccurate and have limited accuracy. The modeling difficulties can be greatly alleviated when the networking protocol is simple and easy to predict. Sometimes it is necessary to reduce a general model to a representative analytical path, to reduce analytical difficulties.

Network simulation provides a way to model the behavior of different devices in a network. It is the typical method in large-scale simulation studies, and it is one of the most expensive. Discrete Event Simulation (DES) is used to model in more realistic ways and therefore has a broad application. DES is an incredibly detailed model of the network and the activity in the network. The method can be time-consuming, but it is very effective when needed for larger simulations. It can take multiple days or extended periods of time to complete. However, simulation can provide a single-node system or a network of queues with accurate results.

One best way to overcome these mathematical analysis and simulation issues is by combining the advantages of both methods and overcoming the disadvantages of both. This concentrated hybrid method is typically known as hybrid simulation.

In the realm of network simulators, there are many that are popular, including OPNET, NS, OMNeT, and other simulators. Opnet provides both explicit DES and hybrid simulation, which include the ability to co-simulate and hybridize.

Typically, test-bed emulation involves the implementation of hardware into the real-world, but on a much smaller scale. Since test-bed emulation considers both protocols and real-world situations, it provides the best way to benchmark the feasibility of the algorithms and their protocols and their accuracy to real-world situations. Technology-enhanced research serves to demonstrate new networking concepts. The disadvantage is it will also involve other real-world difficulties and some unexpected engineering problems that can be completely irrelevant to the studied algorithms and protocols but can be significant to the overall emulation results. In addition, the hosting costs for a testbed may be substantial. Test trials cannot accurately simulate large systems.

8.1 OPNET Simulator

OPNET[64] stands for Optimized Network Engineering Tools. OPNET Technologies, Inc. was founded in 1986. This powerful toolset is used to create and test large network environments. OPNET Modeler is an event-driven simulation tool with a user-friendly GUI, supported by object-oriented modeling and debugging. OPNET Modeler is designed to provide a discrete event simulation tool that supports the simulation of hybrid models and analytical models.

The network simulation can be segmented into time-driven and event-driven simulations. In a time-clocked simulation, simulation progresses at a steady rate across a set period of time. Events are implemented within the time slot while the rest of the simulation is progressing. The following is the flow-chart of a time-based simulation:

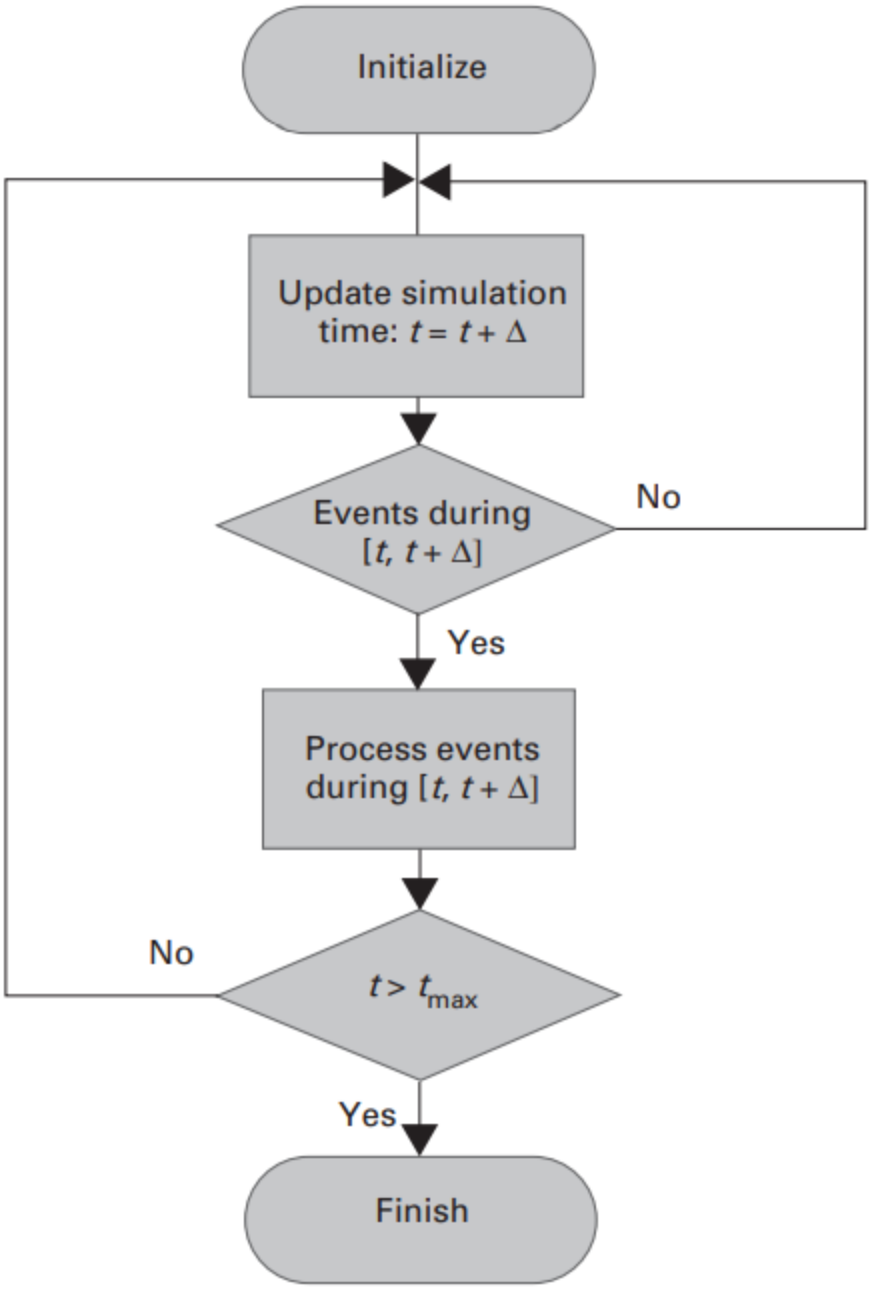


Figure 53. Flowchart of time-based simulation[64]

In discrete event simulation, events are executed in progress on schedule. Simulation time is updated as upcoming events are executed. The following is the flow-chart of a time-based simulation:

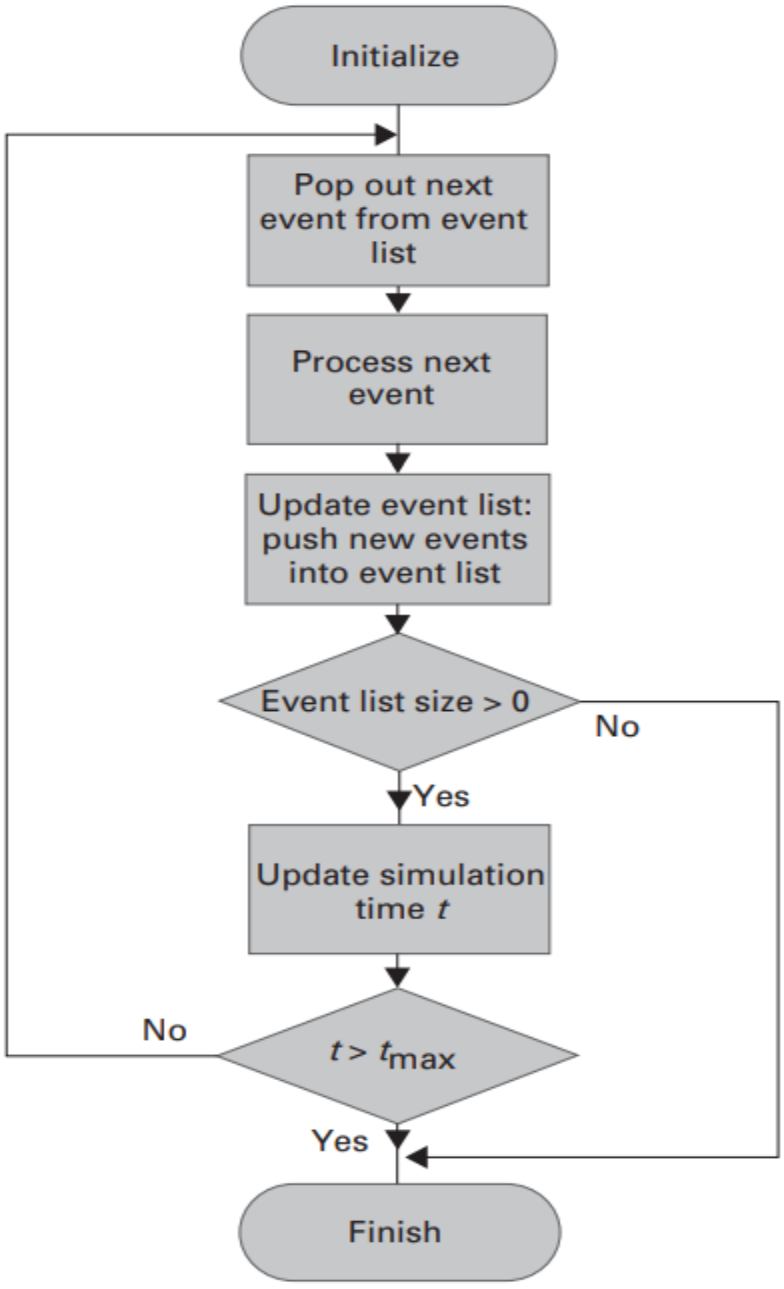


Figure 54. Flowchart of Event-based Simulation[64]

Chapter 9 Conclusion

A CDN is a vast distributed interconnected network of complex devices and cache servers that incorporates multiple algorithms to improve the webpages and the end-user experience. CDNs can be found in many areas, including educational institutions, advertisement companies, data centers, ISPs, streaming providers, mobile carriers, and huge organizations. With technological advancements, the CDN industry is changing along with the introduction of new content types and services. It demands better architectures and solutions that can bring user experience to the epitome while saving infrastructure investments and bandwidth.

This project proposes a distributed CDN architecture with advanced edge servers that can be used to save bandwidth and network infrastructure costs. This project further provides an overview of various CDN architectures, technology trends, advancements in the research and technology, along with the challenges in the research fields. The proposed architecture can be simulated in the future to identify the real-life behavior and can be compared to verify if it matches the expected output.

References

[1] P. Panchal, N. M. Ramaswamy, X. Su, and Y. Dong, "Content delivery networks in the cloud with distributed edge servers," *Proc. - 2013 IEEE Int. Conf. High Perform. Comput. Commun. HPCC 2013 2013 IEEE Int. Conf. Embed. Ubiquitous Comput. EUC 2013*, pp. 526–532, 2014, doi: 10.1109/HPCC.and.EUC.2013.81.

[2] N. Bartolini, E. Casalicchio, and S. Tucci, "A walk through content delivery networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2965, no. January, pp. 1–25, 2004, doi: 10.1007/978-3-540-24663-3_1.

[3] *Content Delivery Networks*, vol. 9 LNEE. 2008.

[4] M. Pathan, R. K Sitaraman, and D. Robinson, *Advanced Content Delivery, Streaming, and Cloud Services*. 2014.

[5] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," *Proc. - IEEE INFOCOM*, vol. 3, pp. 1587–1596, 2001, doi: 10.1109/INFCOM.2001.916655.

[6] N. Fujita, Y. Ishikawa, A. Iwata, and R. Izmailov, "Coarse-grain replica management strategies for dynamic replication of Web contents," *Comput. Networks*, vol. 45, no. 1, pp. 19–34, 2004, doi: 10.1016/j.comnet.2004.02.006.

[7] "https://www.belugacdn.com/push-cdn." .

[8] J. Sahoo, M. A. Salahuddin, R. Glitho, H. Elbiaze, and W. Ajib, "Replica Server Management," 2016.

[9] J. P. Mulerikkal and I. Khalil, "An architecture for distributed content delivery network," *ICON 2007 - Proc. 2007 15th IEEE Int. Conf. Networks*, pp. 359–364, 2007, doi: 10.1109/ICON.2007.4444113.

[10] A. Iyengar, E. Nahum, A. Shaikh, and R. Tewari, *Enhancing Web Performance*, no. September. 2002.

[11] Y. Tang, X. Li, Y. Liu, C. Liu, and Y. Xu, “Review of content distribution network architectures,” *Proc. 2013 3rd Int. Conf. Comput. Sci. Netw. Technol. ICCSNT 2013*, pp. 777–782, 2014, doi: 10.1109/ICCSNT.2013.6967223.

[12] Y. Wang, X. Wen, Y. Sun, Z. Zhao, and T. Yang, “The content delivery network system based on cloud storage,” *Proc. - 2011 Int. Conf. Netw. Comput. Inf. Secur. NCIS 2011*, vol. 1, pp. 98–102, 2011, doi: 10.1109/NCIS.2011.28.

[13] “IBM acquires Instana, a leading enterprise observability platform.” https://www.ibm.com/cloud.

[14] M. Wang *et al.*, “An overview of cloud based content delivery networks: Research dimensions and state-of-the-Art,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9070, pp. 131–158, 2015, doi: 10.1007/978-3-662-46703-9_6.

[15] L. Han, M. Punceva, B. Nath, S. Muthukrishnan, and L. Iftode, “SocialCDN: Caching techniques for distributed social networks,” *2012 IEEE 12th Int. Conf. Peer-to-Peer Comput. P2P 2012*, pp. 191–202, 2012, doi: 10.1109/P2P.2012.6335799.

[16] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web caching and zipf-like distributions: Evidence and implications,” *Proc. - IEEE INFOCOM*, vol. 1, pp. 126–134, 1999, doi: 10.1109/INFCOM.1999.749260.

[17] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst, “Patterns of cascading behavior in large blog graphs,” *Proc. 7th SIAM Int. Conf. Data Min.*, pp. 551–556, 2007, doi: 10.1137/1.9781611972771.60.

[18] G. Pallis and A. Vakali, “Insight and perspectives for Content Delivery Networks,” *Commun. ACM*, vol. 49, no. 1, pp. 101–106, 2006, doi: 10.1145/1107458.1107462.

[19] A. Venkataramani, P. Yalagandula, R. Kokku, S. Sharif, and M. Dahlin, “The potential costs and benefits of long-term prefetching for content distribution,” *Comput. Commun.*, vol. 25, no. 4, pp. 367–375, 2002, doi: 10.1016/S0140-3664(01)00408-X.

[20] B. Mobasher, R. Cooley, and J. Srivastava, “Web usage mining can help improve the scalability, accuracy, and flexibility of recommender systems.,” *Commun. ACM*, vol. 43,

no. 8, pp. 142–151, 2000.

[21] J. Duan *et al.*, “SCDN: A Novel Software-Driven CDN for Better Content Pricing and Caching,” *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 704–707, 2018, doi: 10.1109/LCOMM.2018.2803808.

[22] S. M. Y. Seyyedi and B. Akbari, “Hybrid CDN-P2P architectures for live video streaming: Comparative study of connected and unconnected meshes,” *2011 Int. Symp. Comput. Networks Distrib. Syst. CNDS 2011*, pp. 175–180, 2011, doi: 10.1109/CNDS.2011.5764567.

[23] D. Xu, S. S. Kulkarni, C. Rosenberg, and H. K. Chai, “Analysis of a CDN-P2P hybrid architecture for cost-effective streaming media distribution,” *Multimed. Syst.*, vol. 11, no. 4, pp. 383–399, 2006, doi: 10.1007/s00530-006-0015-3.

[24] J. Broberg, R. Buyya, and Z. Tari, “MetaCDN: Harnessing ‘Storage Clouds’ for high performance content delivery,” *J. Netw. Comput. Appl.*, vol. 32, no. 5, pp. 1012–1022, 2009, doi: 10.1016/j.jnca.2009.03.004.

[25] C. F. Lin, M. C. Leu, C. W. Chang, and S. M. Yuan, “The study and methods for cloud based CDN,” *Proc. - 2011 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov. CyberC 2011*, pp. 469–475, 2011, doi: 10.1109/CyberC.2011.82.

[26] L. Ling, X. Ma, and Y. Huang, “CDN cloud: A novel scheme for combining CDN and cloud computing,” *Proc. 2013 2nd Int. Conf. Meas. Inf. Control. ICMIC 2013*, vol. 1, pp. 687–690, 2013, doi: 10.1109/MIC.2013.6758055.

[27] E. G. N. 001 V1.1.1, “NFV-Use Cases,” *IEEE Netw.*, vol. 1, no. 5, pp. 1–50, 2013, [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4626228.

[28] C. Makaya, D. Freimuth, D. Wood, and S. Calo, “Policy-based NFV management and orchestration,” *2015 IEEE Conf. Netw. Funct. Virtualization Softw. Defin. Network, NFV-SDN 2015*, pp. 128–134, 2016, doi: 10.1109/NFV-SDN.2015.7387417.

[29] N. Herbaut, D. Negru, G. Xilouris, and Y. Chen, “Migrating to a NFV-based Home Gateway: Introducing a Surrogate vNF approach,” *2015 Int. Conf. Netw. Futur. NOF 2015*, 2015, doi: 10.1109/NOF.2015.7333284.

[30] T. Kim and B. Lee, “Scalable CDN service PoC over distributed cloud management platform,” *Int. Conf. ICT Converg.*, pp. 832–833, 2014, doi: 10.1109/ICTC.2014.6983304.

[31] P. Radoslavov, R. Govindan, and D. Estrin, “Topology-informed Internet replica placement,” *Comput. Commun.*, vol. 25, no. 4, pp. 384–392, 2002, doi: 10.1016/S0140-3664(01)00410-8.

[32] D. Povey and J. Harrison, “A Distributed Internet Cache,” *Proc. 20th Aust. Comput. Sci. Conf.*, no. August 1999, pp. 5–7, 1997.

[33] K. Stamos, G. Pallis, and A. Vakali, “Caching techniques on CDN simulated frameworks,” *Lect. Notes Electr. Eng.*, vol. 9 LNEE, no. June 2014, pp. 127–153, 2008, doi: 10.1007/978-3-540-77887-5_5.

[34] U. Edge and S. Includes, “EdgeSuite 5 . 0 : ESI Developer ’ s Guide,” 2004.

[35] Y. Guan, X. Zhang, and Z. Guo, “PrefCache: Edge Cache Admission with User Preference Learning for Video Content Distribution,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8215, no. c, pp. 1–1, 2020, doi: 10.1109/tcsvt.2020.3006388.

[36] Q. Fan, Y. Jiang, H. Yin, Y. Lyu, H. Huang, and X. Zhang, “Resource reservation and request routing for a cloud-based content delivery network,” *Proc. - 13th IEEE Int. Conf. Serv. Syst. Eng. SOSE 2019, 10th Int. Work. Jt. Cloud Comput. JCC 2019 2019 IEEE Int. Work. Cloud Comput. Robot. Syst. CCRS 2019*, pp. 281–286, 2019, doi: 10.1109/SOSE.2019.00048.

[37] C. Videoondemand, D. Eager, M. Vernon, and J. Zahorjan, “Bandwidth Skimming : A Technique for,” vol. 3969, no. January, pp. 206–215, 2000.

[38] M. Aloui, H. Elbiaze, R. Glitho, and S. Yangui, “Analytics as a Service Architecture for Cloud-based CDN : Case of Video Popularity Prediction,” pp. 5–8, 2018.

[39] J. Ni, D. H. K. Tsang, I. S. H. Yeung, and X. Hei, “Hierarchical content routing in large-scale multimedia content delivery network,” *IEEE Int. Conf. Commun.*, vol. 2, no. April 2014, pp. 854–859, 2003, doi: 10.1109/icc.2003.1204454.

[40] S. Triukose, Z. Al-Qudah, and M. Rabinovich, “Content delivery networks: Protection or

threat?," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5789 LNCS, no. September, pp. 371–389, 2009, doi: 10.1007/978-3-642-04444-1_23.

[41] Akamai, "Content Delivery for an Evolving Internet Choosing the Right CDN for Today and Tomorrow," 2019.

[42] A. Barbir, B. Cain, O. Spatscheck, and R. Nair, "Known content network (CN) request-routing mechanisms," *RFC Editor*, 2003. https://tools.ietf.org/html/ rfc3568.

[43] Y. Tsang, M. Coates, and R. D. Nowak, "Network delay tomography," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 2125–2136, 2003, doi: 10.1109/TSP.2003.814520.

[44] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of DNS-based server selection," *Proc. - IEEE INFOCOM*, vol. 3, pp. 1801–1810, 2001, doi: 10.1109/INFCOM.2001.916678.

[45] M. Kabir, E. Manning, and G. Shoja, "Request-routing trends and techniques in content distribution network," *Proc. Int. ...*, 2002.

[46] "Round-robin Load Balancing." https://avinetworks.com/glossary/round-robin-load-balancing/.

[47] U. Goel, M. P. Wittie, and M. Steiner, "Faster web through client-assisted CDN server selection," *Proc. - Int. Conf. Comput. Commun. Networks, ICCCN*, vol. 2015-Octob, 2015, doi: 10.1109/ICCCN.2015.7288411.

[48] J. Shim, P. Scheuermann, and R. Vingralek, "Proxy cache algorithms: design, implementation, and performance," *IEEE Trans. Knowl. Data Eng.*, vol. 11, no. 4, pp. 549–562, 1999, doi: 10.1109/69.790804.

[49] P. K. Shah, "An O (1) algorithm for implementing the LFU cache eviction scheme," *Tech. Rep.*, no. 1, pp. 1–8, 2010, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.479.5221&rep=rep1&type=pdf.

[50] "lets-build-a-min-heap." https://randerson112358.medium.com/lets-build-a-min-heap-4d863cac6521.

[51] R. Chik and Z. Ismail, “Implementing content personalization strategy for Malaysian broadband service providers to enhance subscriber satis faction,” *2013 5th Int. Conf. Inf. Commun. Technol. Muslim World, ICT4M 2013*, 2013, doi: 10.1109/ICT4M.2013.6518916.

[52] B. K. Malviya and J. Agrawal, “A study on web usage mining theory and applications,” *Proc. - 2015 5th Int. Conf. Commun. Syst. Netw. Technol. CSNT 2015*, no. April 2015, pp. 935–939, 2015, doi: 10.1109/CSNT.2015.247.

[53] A. Lobzhanidze and W. Zeng, “Proactive caching of online video by mining mainstream media,” *Proc. - IEEE Int. Conf. Multimed. Expo*, 2013, doi: 10.1109/ICME.2013.6607441.

[54] “Adaptive Streaming.” https://bitmovin.com/adaptive-streaming/.

[55] D. K. Krishnappa, M. Zink, and R. K. Sitaraman, “Optimizing the video transcoding workflow in content delivery networks,” *Proc. 6th ACM Multimed. Syst. Conf. MMSys 2015*, pp. 37–48, 2015, doi: 10.1145/2713168.2713175.

[56] “Cyber Security Reports.” http://www8.hp.com/us/en/software-solutions/ponemon-cyber-security-report/.

[57] A.-J. Su, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante, “Drafting behind Akamai (travelocity-based detouring),” vol. 17, no. 6, p. 435, 2006, doi: 10.1145/1159913.1159962.

[58] “Limelight Orchestrate CDN.” http://www.limelight.com/services/orchestrate-content-delivery.

[59] V. K. Adhikari *et al.*, “Unreeling netflix: Understanding and improving multi-CDN movie delivery,” *Proc. - IEEE INFOCOM*, pp. 1620–1628, 2012, doi: 10.1109/INFCOM.2012.6195531.

[60] “Netflix on AWS.” https://aws.amazon.com/solutions/case-studies/netflix/ (accessed Jan. 20, 2002).

[61] “Video Streaming Market Size, Share & Trends Analysis Report By Streaming Type, By

Solution, By Platform, By Service, By Revenue Model, By Deployment Type, By User, By Region, And Segment Forecasts, 2021 - 2028," 2021. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/video-streaming-market.

[62] M. Abomhara, O. O. Khalifa, O. Zakana, A. A. Zaidan, B. B. Zaidan, and A. Rame, "Video compression techniques: An overview," *J. Appl. Sci.*, vol. 10, no. 16, pp. 1834–1840, 2010, doi: 10.3923/jas.2010.1834.1840.

[63] J. Y. B. Lee, *Scalable Continuous Media Streaming Systems: Architecture, Design, Analysis and Implementation*. 2005.

[64] Z. Lu and H. Yang, *Unlocking the power of OPNET modeler*, vol. 9780521198. 2012.