A Taxonomy and Survey of Content Delivery Networks

Al-Mukaddim Khan Pathan and Rajkumar Buyya

Grid Computing and **D**istributed **S**ystems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
University of Melbourne, Parkville, VIC 3010, Australia
{apathan, raj}@csse.unimelb.edu.au

Abstract: Content Delivery Networks (CDNs) have evolved to overcome the inherent limitations of the Internet in terms of user perceived Quality of Service (QoS) when accessing Web content. A CDN replicates content from the origin server to cache servers, scattered over the globe, in order to deliver content to end-users in a reliable and timely manner from nearby optimal surrogates. Content distribution on the Internet has received considerable research attention. It combines development of high-end computing technologies with high-performance networking infrastructure and distributed replica management techniques. Therefore, our aim is to categorize and analyze the existing CDNs, and to explore the uniqueness, weaknesses, opportunities, and future directions in this field.

In this paper, we provide a comprehensive taxonomy with a broad coverage of CDNs in terms of organizational structure, content distribution mechanisms, request redirection techniques, and performance measurement methodologies. We study the existing CDNs in terms of their infrastructure, request-routing mechanisms, content replication techniques, load balancing, and cache management. We also provide an in-depth analysis and state-of-the-art survey of CDNs. Finally, we apply the taxonomy to map various CDNs. The mapping of the taxonomy to the CDNs helps in "gap" analysis in the content networking domain. It also provides a means to identify the present and future development in this field and validates the applicability and accuracy of the taxonomy.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network Topology*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing Protocols*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed databases*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed Systems*; H.3.5 [**Information Storage and Retrieval**]: Online Information service—*Web-based Services*

General Terms: Taxonomy, Survey, CDNs, Design, Performance

Additional Key Words and Phrases: Content networks, Content distribution, peer-to-peer, replica management, request-routing

1. Introduction

With the proliferation of the Internet, popular Web services often suffer congestion and bottlenecks due to large demands made on their services. Such a scenario may cause unmanageable levels of traffic flow, resulting in many requests being lost. Replicating the same content or services over several mirrored Web servers strategically placed at various locations is a method commonly used by service providers to improve performance and scalability. The user is redirected to the nearest server and this approach helps to reduce network impact on the response time of the user requests.

Content Delivery Networks (CDNs) [8][16][19] provide services that improve network performance by maximizing bandwidth, improving accessibility and maintaining correctness through content replication. They offer fast and reliable applications and services by distributing content to cache or edge servers located close to users [8]. A CDN has some combination of content-delivery, request-routing, distribution and accounting infrastructure. The content-delivery infrastructure consists of a set of edge servers (also called surrogates) that deliver copies of content to end-users. The request-routing infrastructure is responsible to directing client request to appropriate edge servers. It also interacts with the distribution infrastructure to keep an up-to-date view of the content stored in the CDN caches. The distribution infrastructure moves content from the origin server to the CDN edge servers and ensures consistency of content in the caches. The accounting infrastructure maintains logs of client accesses and records the usage of the CDN servers. This information is used for traffic reporting and usage-based billing. In practice, CDNs typically host static content including images, video, media clips, advertisements, and other embedded objects for dynamic Web content. Typical customers of a CDN are media and Internet advertisement companies, data centers, Internet Service Providers (ISPs), online music

retailers, mobile operators, consumer electronics manufacturers, and other carrier companies. Each of these customers wants to publish and deliver their content to the end-users on the Internet in a reliable and timely manner. A CDN focuses on building its network infrastructure to provide the following services and functionalities: storage and management of content; distribution of content among surrogates; cache management; delivery of static, dynamic and streaming content; backup and disaster recovery solutions; and monitoring, performance measurement and reporting.

A few studies have investigated CDNs in the recent past. Peng [19] presents an overview of CDNs. His work presents the critical issues involved in designing and implementing an effective CDN, and surveys the approaches proposed in literature to address these problems. Vakali et al. [16] present a survey of CDN architecture and popular CDN service providers. The survey is focused on understanding the CDN framework and its usefulness. They identify the characteristics and current practices in the content networking domain, and present an evolutionary pathway for CDNs, in order to exploit the current content networking trends. Dilley et al. [2] provide an insight into the overall system architecture of the leading CDN, Akamai [2][3]. They provide an overview of the existing content delivery approaches and describe the Akamai network infrastructure and its operations in detail. They also point out the technical challenges that are to be faced while constructing a global CDN like Akamai. Saroiu et al. [48] examine content delivery from the point of view of four content delivery systems: HTTP web traffic, the Akamai CDN, Gnutella [93][94] and KaZaa [95][96] peer-to-peer file sharing systems. They also present significant implications for large organizations, service providers, network infrastructure providers, and general content delivery providers. Kung et al. [47] describe a taxonomy for content networks and introduce a new class of content network that perform "semantic aggregation and content-sensitive placement" of content. They classify content networks based on their attributes in two dimensions: content aggregation and content placement. As none of these works has categorized CDNs, in this work we focus on developing a taxonomy and presenting a detailed survey of CDNs.

Our contributions –In this paper, our key contributions are to:

1. Develop a comprehensive taxonomy of CDNs that provides a complete coverage of this field in terms of organizational structure, content distribution mechanisms, request redirection techniques, and performance measurement methodologies. The main aim of the taxonomy, therefore, is to explore the unique features of CDNs from similar paradigms and to provide a basis for categorizing present and future development in this area.
2. Present a state-of-the-art survey of the existing CDNs that provides a basis for an in-depth analysis and complete understanding of the current content distribution landscape. It also gives an insight into the underlying technologies that are currently in use in the content-distribution space.
3. Map the taxonomy to the existing CDNs to demonstrate its applicability to categorize and analyze the present-day CDNs. Such a mapping helps to perform "gap" analysis in this domain. It also assists to interpret the related essential concepts of this area and validates the accuracy of the taxonomy.
4. Identify the strength, weaknesses, and opportunities in this field through the state-or-the-art investigation and propose possible future directions as growth advances in related areas through rapid deployment of new CDN services.

The rest of the paper is structured as follows: Section 2 defines the related terminologies, provides an insight into the evolution of CDNs, and highlights other aspects of it. It also identifies uniqueness of CDNs from other related distributed computing paradigms. Section 3 presents the taxonomy of CDNs in terms of four issues/factors – CDN composition, content distribution and management, request-routing, and performance measurement. Section 4 performs a detailed survey of the existing content delivery networks. Section 5 categorizes the existing CDNs by performing a mapping of the taxonomy to each CDN system and outlines the future directions in the content networking domain. Finally, Section 6 concludes the paper with a summary.

2. Overview

A *CDN* is a collection of network elements arranged for more effective delivery of content to end-users [7]. Collaboration among distributed CDN components can occur over nodes in both homogeneous and heterogeneous environments. CDNs can take various forms and structures. They can be centralized, hierarchical infrastructure under certain administrative control, or completely decentralized systems. There can also be various forms of internetworking and control sharing among different CDN entities. General considerations on designing a CDN can be found in [106]. The typical functionality of a CDN includes:

- *Request redirection and content delivery services* to direct a request to the closest suitable surrogate server using mechanisms to bypass congestion, thus overcoming flash crowds or SlashDot effects.
- *Content outsourcing and distribution services* to replicate and/or cache content to distributed surrogate servers on behalf of the origin server.
- *Content negotiation services* to meet specific needs of each individual user (or group of users).

- *Management services* to manage the network components, to handle accounting, and to monitor and report on content usage.

A CDN provides better performance through caching or replicating content over some mirrored Web servers (i.e. *surrogate servers*) strategically placed at various locations in order to deal with the sudden spike in Web content requests, which is often termed as *flash crowd* [1] or *SlashDot effect* [56]. The users are redirected to the surrogate server nearest to them. This approach helps to reduce network impact on the response time of user requests. In the context of CDNs, *content* refers to any digital data resources and it consists of two main parts: the *encoded media* and *metadata* [105]. The encoded media includes static, dynamic and continuous media data (e.g. audio, video, documents, images and Web pages). Metadata is the content description that allows identification, discovery, and management of multimedia data, and also facilitates the interpretation of multimedia data. Content can be pre-recorded or retrieved from live sources; it can be persistent or transient data within the system [105]. CDNs can be seen as a new virtual overlay to the Open Systems Interconnection (OSI) basic reference model [57]. This layer provides overlay network services relying on application layer protocols such as HTTP or RTSP for transport [26].

The three key components of a CDN architecture are – content provider, CDN provider and end-users. A *content provider* or *customer* is one who delegates the URI name space of the Web objects to be distributed. The *origin server* of the content provider holds those objects. A *CDN provider* is a proprietary organization or company that provides infrastructure facilities to content providers in order to deliver content in a timely and reliable manner. *End-users* or *clients* are the entities who access content from the content provider's website.

CDN providers use *caching* and/or *replica servers* located in different geographical locations to replicate content. CDN cache servers are also called *edge servers* or *surrogates*. In this paper, we will use these terms interchangeably. The surrogates of a CDN are called *Web cluster* as a whole. CDNs distribute content to the surrogates in such a way that all cache servers share the same content and URL. Client requests are redirected to the nearby surrogate, and a selected surrogate server delivers requested content to the end-users. Thus, transparency for users is achieved. Additionally, surrogates send accounting information for the delivered content to the accounting system of the CDN provider.

2.1. The evolution of CDNs

Over the last decades, users have witnessed the growth and maturity of the Internet. As a consequence, there has been an enormous growth in network traffic, driven by rapid acceptance of broadband access, along with increases in system complexity and content richness [27]. The over-evolving nature of the Internet brings new challenges in managing and delivering content to users. As an example, popular Web services often suffer congestion and bottleneck due to the large demands made on their services. A sudden spike in Web content requests may cause heavy workload on particular Web server(s), and as a result a *hotspot* [56] can be generated. Coping with such unexpected demand causes significant strain on a Web server. Eventually the Web servers are totally overwhelmed with the sudden increase in traffic, and the Web site holding the content becomes temporarily unavailable.

Content providers view the Web as a vehicle to bring rich content to their users. A decrease in service quality, along with high access delays mainly caused by long download times, leaves the users in frustration. Companies earn significant financial incentives from Web-based e-business. Hence, they are concerned to improve the service quality experienced by the users while accessing their Web sites. As such, the past few years have seen an evolution of technologies that aim to improve content delivery and service provisioning over the Web. When used together, the infrastructures supporting these technologies form a new type of network, which is often referred to as content network [26].

Several content networks attempt to address the performance problem through using different mechanisms to improve the Quality of Service (QoS). One approach is to modify the traditional Web architecture by improving the Web server hardware adding a high-speed processor, more memory and disk space, or maybe even a multi-processor system. This approach is not flexible [27]. Moreover, small enhancements are not possible and at some point, the complete server system might have to be replaced. Caching proxy deployment by an ISP can be beneficial for the narrow bandwidth users accessing the Internet. In order to improve performance and reduce bandwidth utilization, caching proxies are deployed close to the users. Caching proxies may also be equipped with technologies to detect a server failure and maximize efficient use of caching proxy resources. Users often configure their browsers to send their Web request through these caches rather than sending directly to origin servers. When this configuration is properly done, the user's entire browsing session goes through a specific caching proxy. Thus, the caches contain most popular content viewed by all the users of the caching proxies. A provider may also deploy different levels of local, regional, international caches at geographically distributed locations. Such arrangement is referred to as *hierarchical caching*. This may provide additional performance improvements and bandwidth savings [39].

A more scalable solution is the establishment of server farms. It is a type of content network that has been in widespread use for several years. A server farm is comprised of multiple Web servers, each of them sharing

the burden of answering requests for the same Web site [27]. It also makes use of a Layer 4-7 switch, Web switch or content switch that examines content request and dispatches them among the group of servers. A server farm can also be constructed with surrogates [63] instead of a switch. This approach is more flexible and shows better scalability [27]. Moreover, it provides the inherent benefit of fault tolerance [26]. Deployment and growth of server farms progresses with the upgrade of network links that connects the Web sites to the Internet.

Although server farms and hierarchical caching through caching proxies are useful techniques to address the Internet Web performance problem, they have limitations. In the first case, since servers are deployed near the origin server, they do little to improve the network performance due to network congestion. Caching proxies may be beneficial in this case. But they cache objects based on client demands. This may force the content providers with a popular content source to invest in large server farms, load balancing, and high bandwidth connections to keep up with the demand. To address these limitations, another type of content network has been deployed in late 1990s. This is termed as *Content Distribution Network* or *Content Delivery Network*, which is a system of computers networked together across the Internet to cooperate transparently for delivering content to end-users.

With the introduction of CDN, content providers started putting their Web sites on a CDN. Soon they realized its usefulness through receiving increased reliability and scalability without the need to maintain expensive infrastructure. Hence, several initiatives kicked off for developing infrastructure for CDNs. As a consequence, Akamai Technologies [2][3] evolved out of an MIT research effort aimed at solving the flash crowd problem. Within a couple of years, several companies became specialists in providing fast and reliable delivery of content, and CDNs became a huge market for generating large revenues. The flash crowd events [97] like the 9/11 incident in USA [98], resulted in serious caching problems for some site. This influenced the CDN providers to invest more in CDN infrastructure development, since CDNs provide desired level of protection to Web sites against flash crowds. First generation CDNs mostly focused on static or Dynamic Web documents [16][61]. On the other hand, for second generation of CDNs the focus has shifted to Video-on-Demand (VoD), audio and video streaming. But they are still in research phase and have not reached to the market yet.

With the booming of the CDN business, several standardization activities also emerged since vendors started organizing themselves. The Internet Engineering Task Force (IETF) as a official body took several initiatives through releasing RFCs (Request For Comments) [26][38][63][68]. Other than IETF, several other organizations such as Broadband Services Forum (BSF) [102], ICAP forum [103], Internet Streaming Media Alliance [104] took initiatives to develop standards for delivering broadband content, streaming rich media content – video, audio, and associated data – over the Internet. In the same breath, by 2002, large-scale ISPs started building their own CDN functionality, providing customized services. In 2004, more than 3000 companies were found to use CDNs, spending more than $20 million monthly [101]. A market analysis [98] shows that CDN providers have doubled their earnings from streaming media delivery in 2004 compared to 2003. In 2005, CDN revenue for both streaming video and Internet radio was estimated to grow at 40% [98]. A recent marketing research [100] shows that combined commercial market value for streaming audio, video, streaming audio and video advertising, download media and entertainment was estimated at between $385 million to $452 million in 2005. Considering this trend, the market was forecasted to reach $2 billion in four-year (2002-2006) total revenue in 2006, with music, sports, and entertainment subscription and download revenue for the leading content categories [132]. However, the latest report [133] from AccuStream iMedia Research reveals that since 2002, the CDN market has invested $1.65 billion to deliver streaming media (excluding storage, hosting, applications layering), and the commercial market value in 2006 would make up 36% of the $1.65 billion four-year total in media and entertainment, including content, streaming advertising, movie and music downloads and User Generated Video (UGV) distribution [134]. A detailed report on CDN market opportunities, strategies, and forecasts for the period 2004-2009, in relation to streaming media delivery can be found in [135].

2.2. Insight into CDNs

Figure 1 shows a typical content delivery environment where the replicated Web server clusters are located at the edge of the network to which the end-users are connected. A content provider (i.e. customer) can sign up with a CDN provider for service and have its content placed on the content servers. The content is replicated either on-demand when users request for it, or it can be replicated beforehand, by pushing the content to the surrogate servers. A user is served with the content from the nearby replicated Web server. Thus, the user ends up unknowingly communicating with a replicated CDN server close to it and retrieves files from that server.

CDN providers ensure the fast delivery of any digital content. They host third-party content including static content (e.g. static HTML pages, images, documents, software patches), streaming media (e.g. audio, real time video), User Generated Videos (UGV), and varying content services (e.g. directory service, e-commerce service, file transfer service). The sources of content include large enterprises, Web service providers, media companies and news broadcasters. The end-users can interact with the CDN by specifying the content/service

request through cell phone, smart phone/PDA, laptop and desktop. Figure 2 depicts the different content/services served by a CDN provider to end-users.

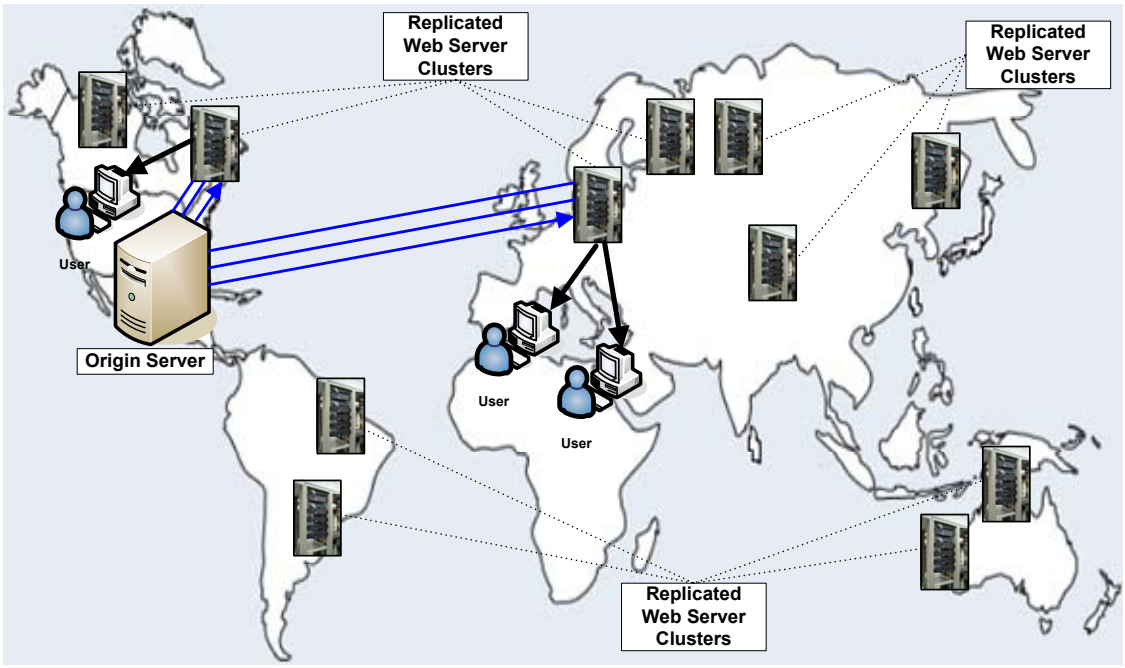

Figure 1: Abstract architecture of a Content Delivery Network (CDN)

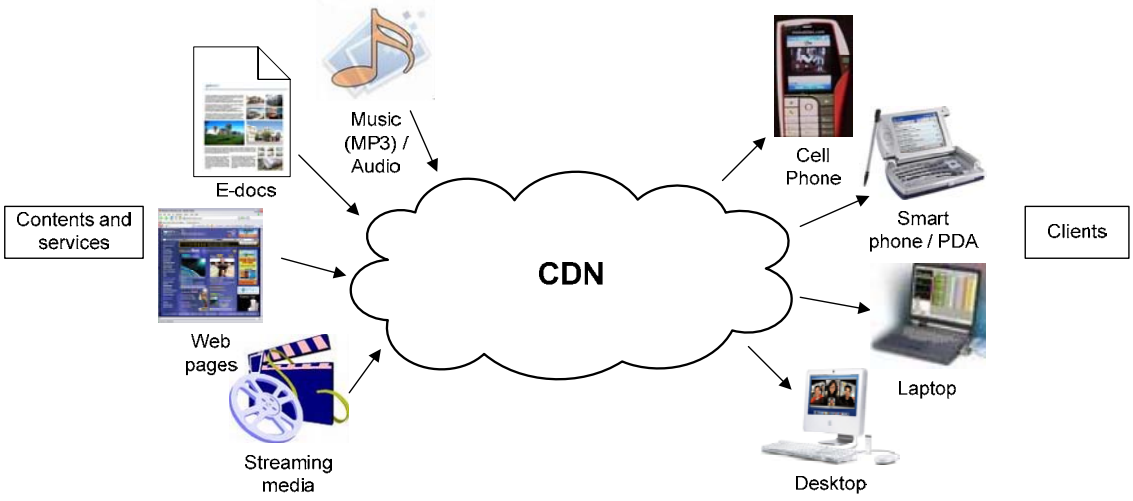

Figure 2: Content/services provided by a CDN

CDN providers charge their customers according to the content delivered (i.e. traffic) to the end-users by their surrogate servers. CDNs support an accounting mechanism that collects and tracks client usage information related to request-routing, distribution and delivery [26]. This mechanism gathers information in real time and collects it for each CDN component. This information can be used in CDNs for accounting, billing and maintenance purposes. The average cost of charging of CDN services is quite high [25], often out of reach for many small to medium enterprises (SME) or not-for-profit organizations. The most influencing factors [8] affecting the price of CDN services include:

- bandwidth cost
- variation of traffic distribution
- size of content replicated over surrogate servers
- number of surrogate servers

- reliability and stability of the whole system and security issues of outsourcing content delivery

A CDN is essentially aimed at content providers or customers who want to ensure QoS to the end-users while accessing their Web content. The analysis of present day CDNs reveals that, at the minimum, a CDN focuses on the following business goals: scalability, security, reliability, responsiveness and performance [40][48][64].

Scalability – The main business goal of a CDN is to achieve scalability. Scalability refers to the ability of the system to expand in order to handle new and large amounts of data, users and transactions without any significant decline in performance. To expand in a global scale, CDNs need to invest time and costs in provisioning additional network connections and infrastructures [40]. It includes provisioning resources dynamically to address flash crowds and varying traffic. A CDN should act as a shock absorber for traffic by automatically providing capacity-on-demand to meet the requirements of flash crowds. This capability allows a CDN to avoid costly over-provisioning of resources and to provide high performance to every user.

Security – One of the major concerns of a CDN is to provide potential security solutions for confidential and high-value content [64]. Security is the protection of content against unauthorized access and modification. Without proper security control, a CDN platform is subject to cyber fraud, distributed denial-of-service (DDoS) attacks, viruses, and other unwanted intrusions that can cripple business [40]. A CDN aims at meeting the stringent requirements of physical, network, software, data and procedural security. Once the security requirements are met, a CDN can eliminate the need for costly hardware and dedicated component to protect content and transactions. In accordance to the security issues, a CDN combat against any other potential risk concerns including denial-of-service attacks or other malicious activity that may interrupt business.

Reliability, Responsiveness and Performance – Reliability refers to when a service is available and what are the bounds on service outages that may be expected. A CDN provider can improve client access to specialized content through delivering it from multiple locations. For this a fault-tolerant network with appropriate load balancing mechanism is to be implemented [142]. Responsiveness implies, while in the face of possible outages, how soon a service would start performing the normal course of operation. Performance of a CDN is typically characterized by the response time (i.e. latency) perceived by the end-users. Slow response time is the single greatest contributor to customers' abandoning Web sites and processes [40]. The reliability and performance of a CDN is affected by the distributed content location and routing mechanism, as well as by data replication and caching strategies. Hence, a CDN employs caching and streaming to enhance performance especially for delivery of media content [48]. A CDN hosting a Web site also focuses on providing fast and reliable service since it reinforces the message that the company is reliable and customer-focused [40].

2.3. Layered architecture

The architecture of content delivery networks can be presented according to a layered approach. In Figure 3, we present the layered architecture of CDNs, which consists of the following layers: *Basic Fabric*, *Communication & Connectivity*, *CDN* and *End-user*. The layers are defined in the following as a bottom up approach.

- *Basic Fabric* is the lowest layer of a CDN. It provides the infrastructural resources for its formation. This layer consists of the distributed computational resources such as SMP, clusters, file servers, index servers, and basic network infrastructure connected by high-bandwidth network. Each of these resources runs system software such as operating system, distributed file management system, and content indexing and management systems.
- *Communication & Connectivity* layer provides the core internet protocols (e.g. TCP/UDP, FTP) as well as CDN specific internet protocols (e.g. Internet Cache Protocol (ICP), Hypertext Caching Protocol (HTCP), and Cache Array Routing Protocols (CARP), and authentication protocols such as PKI (Public Key Infrastructures), or SSL (Secure Sockets Layer) for communication, caching and delivery of content and/or services in an authenticated manner. Application specific overlay structures provide efficient search and retrieval capabilities for replicated content by maintaining distributed indexes.
- *CDN* layer consists of the core functionalities of CDN. It can be divided into three sub-layers: CDN services, CDN types and content types. A CDN provides *core services* such as surrogate selection, request-routing, caching and geographic load balancing, and *user specific services* for SLA management, resource sharing and CDN brokering. A CDN can operate within an enterprise domain, it can be for academic and/or public purpose or it can simply be used as edge servers of content and services. A CDN can also be dedicated to file sharing based on a peer-to-peer (P2P) architecture. A CDN provides all types of MIME content (e.g. text, audio, video etc) to its users.
- *End-users* are at the top of the CDN layered architecture. In this layer, we have the Web users who connect to the CDN by specifying the URL of content provider's Web site, in their Web browsers.

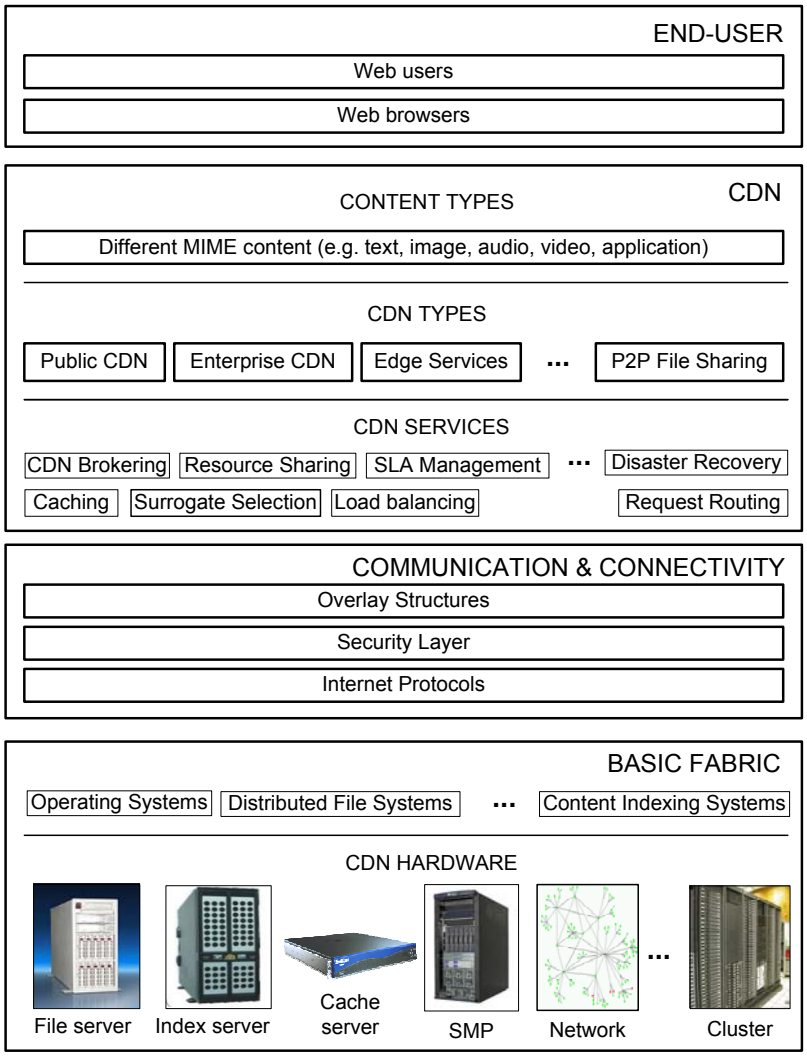

Figure 3: Layered architecture of a CDN

2.4. Related systems

Data grids, distributed databases and peer-to-peer (P2P) networks are three distributed systems that have some characteristics in common with CDNs. These three systems have been described here in terms of requirements, functionalities and characteristics.

Data grids – A data grid [58][59] is a data intensive computing environment that provides services to the users in different locations to discover, transfer, and manipulate large datasets stored in distributed repositories. At the minimum, a data grid provides two basic functionalities: a high-performance, reliable data transfer mechanism, and a scalable replica discovery and management mechanism [41]. A data grid consists of computational and storage resources in different locations connected by high-speed networks. They are especially targeted to large scientific applications such as high energy physics experiments at the Large Hadron Collidor [136], astronomy projects – Virtual Observatories [137], and protein simulation – BioGrid [138] that require analyzing huge amount of data. The data generated from an instrument, experiment, or a network of sensors is stored at a principle storage site and is transferred to other storage sites around the world on request through the data replication mechanism. Users query the local replica catalog to locate the datasets that they require. With proper rights and permissions, the required dataset is fetched from the local repository if it is present there or otherwise it is fetched from a remote repository. The data may be transmitted to a computational unit for processing. After processing, the results may be sent to a visualization facility, a shared repository, or to individual users' desktops. Data grids promote an environment for the users to analyze data, share the results with the collaborators, and maintain state information about the data seamlessly across organizational and regional boundaries. Resources in a data grid are heterogeneous and are spread over multiple administrative

domains. Presence of large datasets, sharing of distributed data collections, having the same logical namespace, and restricted distribution of data can be considered as the unique set of characteristics for data grids. Data grids also contain some application specific characteristics. The overall goal of data grids is to bring together existing distributed resources to obtain performance gain through data distribution. Data grids are created by institutions who come together to share resources on some shared goal(s) by forming a Virtual Organization (VO). On the other hand, the main goal of CDNs is to perform caching of data to enable faster access by the end-users. Moreover, all the commercial CDNs are proprietary in nature – individual companies own and operate them.

Distributed databases – A distributed database (DDB) [42][60] is a logically organized collection of data distributed across multiple physical locations. It may be stored in multiple computers located in the same physical location, or may be dispersed over a network of interconnected computers. Each computer in a distributed database system is a node. A node in a distributed database system acts as a client, server, or both depending on the situation. Each site has a degree of autonomy, is capable of executing a local query, and participates in the execution of a global query. A distributed database can be formed by splitting a single database or by federating multiple existing databases. The distribution of such a system is transparent to the users as they interact with the system as a single logical system. The transactions in a distributed database are transparent and each transaction must maintain integrity across multiple databases. Distributed databases have evolved to serve the need of large organizations that need to replace existing centralized database systems, interconnect existing databases, and to add new databases as new organizational units are added. Applications provided by DDB include distributed transaction processing, distributed query optimization, and efficient management of resources. DDBs are dedicated to integrate existing diverse databases to provide a uniform, consisting interface for query processing with increased reliability and throughput. Integration of databases in DDBs is performed by a single organization. Like DDBs, the entire network in CDNs is managed by a single authoritative entity. However, CDNs differ from DDBs in the fact that CDN cache servers do not have the autonomic property as in DDB sites. Moreover, the purpose of CDNs is content caching, while DDBs are used for query processing, optimization and management.

Peer-to-peer networks – Peer-to-peer (P2P) networks [43][55] are designed for the direct sharing of computer resources rather than requiring any intermediate and/or central authority. They are characterized as information retrieval networks that are formed by ad-hoc aggregation of resources to form a fully or partially decentralized system. Within a peer-to-peer system, each peer is autonomous and relies on other peers for resources, information, and forwarding requests. Ideally there is no central point of control in a P2P network. Therefore, the participating entities collaborate to perform tasks such as searching for other nodes, locating or caching content, routing requests, encrypting, retrieving, decrypting, and verifying content. Peer-to-peer systems are more fault-tolerant and scalable than the conventional centralized system, as they have no single point of failure. An entity in a P2P network can join or leave anytime. P2P networks are more suited to the individual content providers who are not able to access or afford the common CDN. An example of such system is BitTorrent [112], which is a popular P2P replication application. Content and file sharing P2P networks are mainly focused on creating efficient strategies to locate particular files within a group of peers, to provide reliable transfers of such files in case of high volatility, and to manage heavy traffic (i.e. flash crowds) caused by the demand for highly popular files. This is in contrast to CDNs where the main goal lies in respecting client's performance requirements rather than efficiently finding a nearby peer with the desired content. Moreover, CDNs differ from the P2P networks because the number of nodes joining and leaving the network per unit time is negligible in CDNs, whereas the rate is important in P2P networks.

3. Taxonomy

This section presents a detailed taxonomy of CDNs with respect to four different issues/factors. As shown in Figure 4, they are – *CDN composition*, *content distribution and management*, *request-routing*, and *performance measurement*. Our focus in this paper is on the categorization of various attributes/aspects of CDNs. The issues considered for the taxonomy provide a complete reflection of the properties of existing content networks. A proof against our claim has been reflected in Section 4, which illustrates a state-of-the-art survey on existing CDNs.

The first issue covers several aspects of CDNs related to organization and formation. This classifies the CDNs with respect to their structural attributes. The second issue pertains to the content distribution mechanisms in the CDNs. It describes the content distribution and management approaches of CDNs in terms of surrogate placement, content selection and delivery, content outsourcing, and organization of caches/replicas. The third issue considered relates to the request-routing algorithms and request-routing methodologies in the existing CDNs. The final issue emphasizes on the performance measurement of CDNs and looks into the performance metrics and network statistics acquisition techniques used for CDNs. Each of the issues covered in the taxonomy is an independent field, for which extensive research are to be conducted. In this paper, we also validate our taxonomy in Section 5, by performing a mapping of this taxonomy to various CDNs.

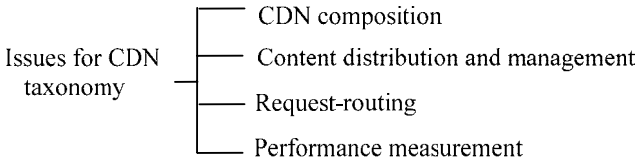

Figure 4: Issues for CDN taxonomy

3.1. CDN composition

A CDN typically incorporates dynamic information about network conditions and load on the cache servers, to redirect request and balance loads among surrogates. The analysis on the structural attributes of a CDN reveals the fact that CDN infrastructural components are closely related to each other. Moreover, the structure of a CDN varies depending on the content/services it provides to its users. Within the structure of a CDN, a set of surrogates is used to build the content-delivery infrastructure, some combinations of relationships and mechanisms are used for redirecting client requests to a surrogate and interaction protocols are used for communications among the CDN elements.

Figure 5 shows the taxonomy based on the various structural characteristics of CDNs. These characteristics are central to the composition of a CDN and they address the organization, types of servers used, relationships and interactions among CDN components, as well as the different content and services provided by the CDNs.

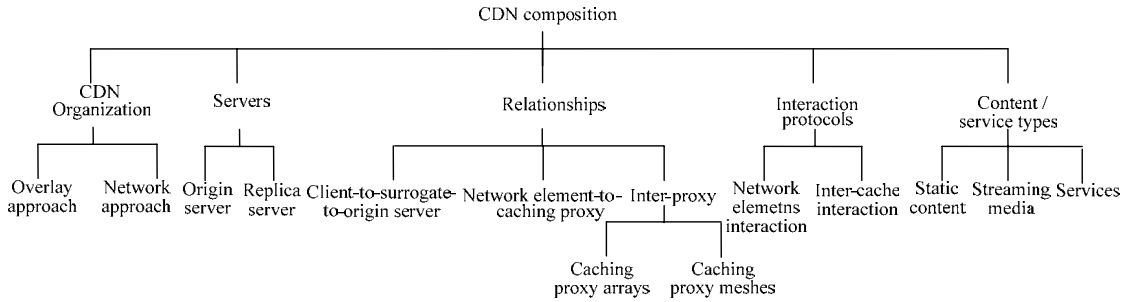

Figure 5: CDN composition taxonomy

CDN organization **–** There are two general approaches to building CDNs: overlay and network approach [61]. In the *overlay* approach, application-specific servers and caches at several places in the network handle the distribution of specific content types (e.g. web content, streaming media, and real time video). Other than providing the basic network connectivity and guaranteed QoS for specific request/traffic, the core network components such as routers and switches play no active role in content delivery. Most of the commercial CDN providers such as Akamai, AppStream, and Limelight Networks follow the overlay approach for CDN organization. These CDN providers replicate content to thousands of cache server worldwide. When content requests are received from end-users, they are redirected to the nearest CDN server, thus improving Web site response time. As the CDN providers need not to control the underlying network infrastructure elements, the management is simplified in an overlay approach and it opens opportunities for new services. In the *network* approach, the network components including routers and switches are equipped with code for identifying specific application types and for forwarding the requests based on predefined policies. Examples of this approach include devices that redirect content requests to local caches or switch traffic coming to data centers to specific servers optimized to serve specific content types. Some CDN (e.g. Akamai, Mirror Image) use both the network and overlay approach for CDN organization. In such case, a network element (e.g. switch) can act at the front end of a server farm and redirects the content request to a nearby application-specific surrogate server.

Servers **–** The servers used by a CDN are of two types – origin server and replica server. The server where the definitive version of a resource resides is called *origin server*, which is updated by the content provider. A server is called a *replica server* when it is holding a replica of a resource but may act as an authoritative reference for client responses. The origin server communicates with the distributed replica servers to update the content stored in it. A replica server in a CDN may serve as a media server, Web server or as a cache server. A media server serves any digital and encoded content. It consists of media server software. Based on client requests, a media server responds to the query with the specific video or audio clip. A Web server contains the links to the streaming media as well as other Web-based content that a CDN wants to handle. A cache server makes copies (i.e. caches) of content at the edge of the network in order to bypass the need of accessing origin server to satisfy every content request.

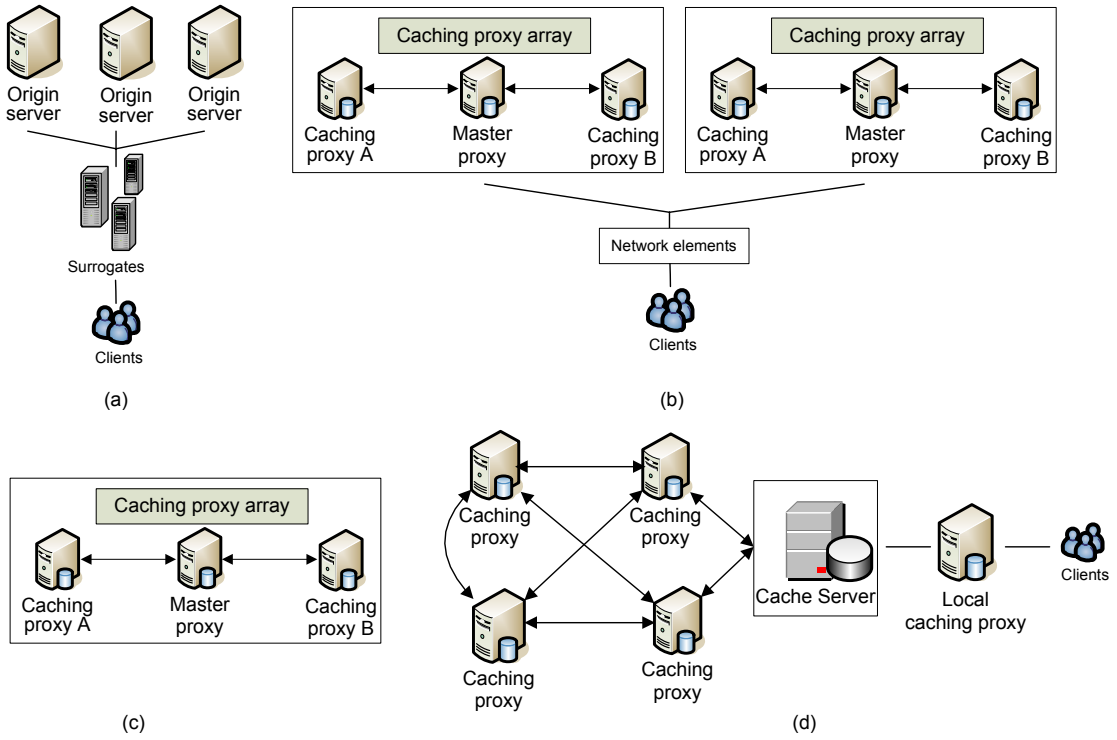

Figure 6: (a) Client-to-surrogate-to-origin server; (b) Network element-to-caching proxy; (c) Caching proxy arrays; (d) Caching proxy meshes

Relationships – The complex distributed architecture of a CDN exhibits different relationships between its constituent components. In this section, we try to identify the relationships that exist in the replication and caching environment of a CDN. The graphical representations of these relationships are shown in Figure 6. These relationships involve components such as clients, surrogates, origin server, proxy caches and other network elements. These components communicate to replicate and cache content within a CDN. Replication involves creating and maintaining duplicate copy of some content on different computer system. It typically involves 'pushing' content from the origin server to the replica servers [64]. On the other hand, caching involves storing cacheable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. A more detailed overview on Web replication and caching has been undertaken by Davison and others [63][86][87].

In a CDN environment, the basic relationship for content delivery is among the client, surrogates and origin servers. A client may communicate with surrogate server(s) for requests intended for one or more origin servers. Where a surrogate is not used, the client communicates directly with the origin server. The communication between a user and surrogate takes place in a transparent manner, as if the communication is with the intended origin server. The surrogate serves client requests from its local cache or acts as a gateway to the origin server. The relationship among client, surrogates and the origin server is shown in Figure 6(a).

As discussed earlier, CDNs can be formed using a network approach, where logic is deployed in the network elements (e.g. router, switch) to forward traffic to servers/proxies that are capable of serving client requests. The relationship in this case is among the client, network element and caching servers/proxies (or proxy arrays), which is shown in Figure 6(b). Other than these relationships, caching proxies within a CDN may communicate with each other. A proxy cache is an application-layer network service for caching Web objects. Proxy caches can be simultaneously accessed and shared by many users. A key distinction between the CDN proxy caches and ISP-operated caches is that the former serve content only for certain content provider, namely CDN customers, while the latter cache content from all Web sites [89].

Based on inter-proxy communication [63], caching proxies can be arranged in such a way that proxy arrays (Figure 6(c)) and proxy meshes (Figure 6(d)) are formed. A caching proxy array is a tightly-coupled arrangement of caching proxies. In a caching proxy array, an authoritative proxy acts as a master to communicate with other caching proxies. A user agent can have relationship with such an array of proxies. A caching proxy mesh is a loosely-coupled arrangement of caching proxies. Unlike the caching proxy arrays, proxy meshes are created when the caching proxies have one-to one relationship with other proxies. Within a caching proxy mesh, communication can happen at the same level between peers, and with one or more parents

[63]. A cache server acts as a gateway to such a proxy mesh and forwards client requests coming from client's local proxy.

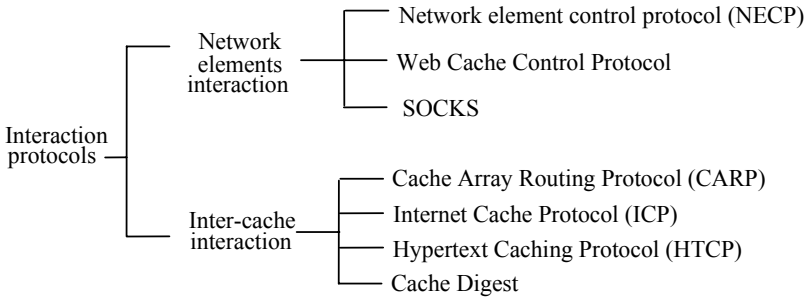

Figure 7: Various interaction protocols

Interaction protocols – Based on the communication relationships described earlier, we can identify the interaction protocols that are used for interaction among CDN elements. Such interactions can be broadly classified into two types: interaction among network elements and interaction between caches. Figure 7 shows various interaction protocols that are used in a CDN for interaction among CDN elements. Examples of protocols for network element interaction are *Network Element Control Protocol (NECP)* [35], *Web Cache Coordination Protocol* [35] and *SOCKS* [36]. On the other hand, *Cache Array Routing Protocol (CARP)* [31], *Internet Cache Protocol (ICP)* [28], *Hypertext Caching protocol (HTCP)* [37], and *Cache Digest* [65] are the examples of inter-cache interaction protocols. These protocols are briefly described here:

NECP: The Network Element Control Protocol (NECP) [35] is a lightweight protocol for signaling between servers and the network elements that forward traffic to them. The network elements consist of a range of devices, including content-aware switches and load-balancing routers. NECP allows network elements to perform load balancing across a farm of servers and redirection to interception proxies. However, it does not dictate any specific load balancing policy. Rather, this protocol provides methods for network elements to learn about server capabilities, availability and hints as to which flows can and cannot be served. Hence, network elements gather necessary information to make load balancing decisions. Thus, it avoids the use of proprietary and mutually incompatible protocols for this purpose. NECP is intended for use in a wide variety of server applications, including for origin servers, proxies, and interception proxies. It uses TCP as the transport protocol. When a server is initialized, it establishes a TCP connection to the network elements using a well-known port number. Messages can then be sent bi-directionally between the server and network element. All NECP messages consist of a fixed-length header containing the total data length and variable length data. Most messages consist of a request followed by a reply or acknowledgement. Receiving a positive acknowledgement implies the recording of some state in a peer. This state can be assumed to remain in that peer until the state expires or the peer crashes. In other words, this protocol uses a 'hard state' model. Application level KEEPALIVE messages are used to detect a dead peer in such communications. When a node detects that its peer has been crashed, it assumes that all the states in that peer need to be reinstalled after the peer is revived.

WCCP: The Web Cache Coordination Protocol (WCCP) [35] specifies interaction between one or more routers and one or more Web-caches. It runs between a router functioning as a redirecting network element and interception proxies. The purpose of such interaction is to establish and maintain the transparent redirection of selected types of traffic flow through a group of routers. The selected traffic is redirected to a group of Web-caches in order to increase resource utilization and to minimize response time. WCCP allows one or more proxies to register with a single router to receive redirected traffic. This traffic includes user requests to view pages and graphics on World Wide Web servers, whether internal or external to the network, and the replies to those requests. This protocol allows one of the proxies, the designated proxy, to dictate to the router how redirected traffic is distributed across the caching proxy array. WCCP provides the means to negotiate the specific method used to distribute load among Web caches. It also provides methods to transport traffic between router and cache.

SOCKS: The SOCKS protocol is designed to provide a framework for client-server applications in both the TCP and UDP domains to conveniently and securely use the services of a network firewall [36]. The protocol is conceptually a "shim-layer" between the application layer and the transport layer, and as such does not provide network-layer gateway services, such as forwarding of ICMP messages. When used in conjunction with a firewall, SOCKS provides an authenticated tunnel between the caching proxy and the firewall. In order to implement SOCKS protocol, TCP-based client applications are recompiled so that they can use the appropriate encapsulation routines in SOCKS library. When connecting to a cacheable content behind firewall, a TCP-based client has to open a TCP connection to the SOCKS port on the SOCKS server system. Upon successful establishment of the connection, a client negotiates for the suitable method for authentication, authenticates with

the chosen method and sends a relay request. SOCKS server in turn establishes the requested connection or rejects it based on the evaluation result of the connection request.

CARP: The Cache Array Routing Protocol (CARP) [31] is a distributed caching protocol based on a known list of loosely coupled proxy servers and a hash function for dividing URL space among those proxies. An HTTP client implementing CARP can route requests to any member of the Proxy Array. The proxy array membership table is defined as a plain ASCII text file retrieved from an Array Configuration URL. The hash function and the routing algorithm of CARP take a member proxy defined in the proxy array membership table, and make an on-the-fly determination about the proxy array member which should be the proper container for a cached version of a resource pointed to by a URL. Since requests are sorted through the proxies, duplication of cache content is eliminated and global cache hit rates are improved. Downstream agents can then access a cached resource by forwarding the proxied HTTP request [66] for the resource to the appropriate proxy array member.

ICP: The Internet Cache Protocol (ICP) [28] is a lightweight message format used for inter-cache communication. Caches exchange ICP queries and replies to gather information to use in selecting the most appropriate location in order to retrieve an object. Other than functioning as an object location protocol, ICP messages can also be used for cache selection. ICP is a widely deployed protocol. Although, Web caches use HTTP [66] for the transfer of object data, most of the caching proxy implementation supports it in some form. It is used in a caching proxy mesh to locate specific Web objects in neighboring caches. One cache sends an ICP query to its neighbors and the neighbors respond with an ICP reply indicating a 'HIT' or a 'MISS'. Failure to receive a reply from the neighbors within a short period of time implies that the network path is either congested or broken. Usually, ICP is implemented on top of UDP [67] in order to provide important features to Web caching applications. Since UDP is an uncorrected network transport protocol, an estimate of network congestion and availability may be calculated by ICP loss. This sort of loss measurement together with the round-trip-time provides a way to load balancing among caches.

HTCP: The Hypertext Caching Protocol (HTCP) [37] is a protocol for discovering HTTP caches, cached data, managing sets of HTTP caches and monitoring cache activity. HTCP is compatible with HTTP 1.0, which permits headers to be included in a request and/or a response. This is in contrast with ICP, which was designed for HTTP 0.9. HTTP 0.9 allows specifying only a URI in the request and offers only a body in the response. Hence, only the URI without any headers is used in ICP for cached content description. Moreover, it also does not support the possibility of multiple compatible bodies for the same URI. On the other hand, HTCP permits full request and response headers to be used in cache management. HTCP also expands the domain of cache management to include monitoring a remote cache's additions and deletions, requesting immediate deletions, and sending hints about Web objects such as the third party locations of cacheable objects or the measured uncacheability or unavailability of Web objects. HTCP messages may be sent over UDP [67], or TCP. HTCP agents must not be isolated from network failure and delays. An HTCP agent should be prepared to act in useful ways in the absence of response or in case of lost or damaged responses.

Cache Digest: Cache Digest [65] is an exchange protocol and data format. Cache digests provide a solution to the problems of response time and congestion associated with other inter-cache communication protocols such as ICP [28] and HTCP [37]. They support peering between cache servers without a request-response exchange taking place. Instead, other servers who peer with it fetch a summary of the content of the server (i.e. the Digest). When using cache digests it is possible to accurately determine whether a particular server caches a given URL. It is currently performed via HTTP. A peer answering a request for its digest will specify an expiry time for that digest by using the HTTP Expires header. The requesting cache thus knows when it should request a fresh copy of that peer's digest. In addition to HTTP, Cache Digests could be exchanged via FTP. Although the main use of Cache Digests is to share summaries of which URLs are cached by a given server, it can be extended to cover other data sources. Cache Digests can be a very powerful mechanism to eliminate redundancy and making better use of Internet server and bandwidth resources.

Content/service types – CDN providers host third-party content for fast delivery of any digital content, including – static content, streaming media (e.g. audio, real time video) and varying content services (e.g. directory service, e-commerce service, and file transfer service). The sources of content are large enterprises, web service providers, media companies, and news broadcasters. Variation in content and services delivered requires the CDN to adopt application-specific characteristics, architectures and technologies. Due to this reason, some of the CDNs are dedicated for delivering particular content and/or services. Here we analyze the characteristics of the content/service types to reveal their heterogeneous nature.

Static content: Static HTML pages, images, documents, software patches, audio and/or video files fall into this category. The frequency of change for the static content is low. All CDN providers support this type of content delivery. This type of content can be cached easily and their freshness can be maintained using traditional caching technologies.

Streaming media: Streaming media delivery is challenging for CDNs. Streaming media can be live or on-demand. Live media delivery is used for live events such as sports, concerts, channel, and/or news broadcast. In

this case, content is delivered 'instantly' from the encoder to the media server, and then onto the media client. In case of on-demand delivery, the content is encoded and then is stored as streaming media files in the media servers. The content is available upon requests from the media clients. On-demand media content can include audio and/or video on-demand, movie files and music clips. Streaming servers are adopted with specialized protocols for delivery of content across the IP network.

Services: A CDN can offer its network resources to be used as a service distribution channel and thus allows the value-added services providers to make their application as an Internet infrastructure service. When the edge servers host the software of value-added services for content delivery, they may behave like transcoding proxy servers, remote callout servers, or surrogate servers [53]. These servers also demonstrate capability for processing and special hosting of the value-added Internet infrastructure services. Services provided by CDNs can be directory, Web storage, file transfer, and e-commerce services. Directory services are provided by the CDN for accessing the database servers. Users query for certain data is directed to the database servers and the results of frequent queries are cached at the edge servers of the CDN. Web storage service provided by the CDN is meant for storing content at the edge servers and is essentially based on the same techniques used for static content delivery. File transfer services facilitate the worldwide distribution of software, virus definitions, movies on-demand, highly detailed medical images etc. All these contents are static by nature. Web services technologies are adopted by a CDN for their maintenance and delivery. E-commerce is highly popular for business transactions through the Web. Shopping carts for e-commerce services can be stored and maintained at the edge servers of the CDN and online transactions (e.g. third-party verification, credit card transactions) can be performed at the edge of CDNs. To facilitate this service, CDN edge servers should be enabled with dynamic content caching for e-commerce sites.

3.2. Content distribution and management

Content distribution and management is strategically vital in a CDN for efficient content delivery and for overall performance. Content distribution includes – the *placement of surrogates* to some strategic positions so that the edge servers are close to the clients; *content selection and delivery* based on the type and frequency of specific user requests and *content outsourcing* to decide which outsourcing methodology to follow. Content management is largely dependent on the techniques for *cache organization* (i.e. caching techniques, cache maintenance and cache update). The content distribution and management taxonomy is shown in Figure 8.

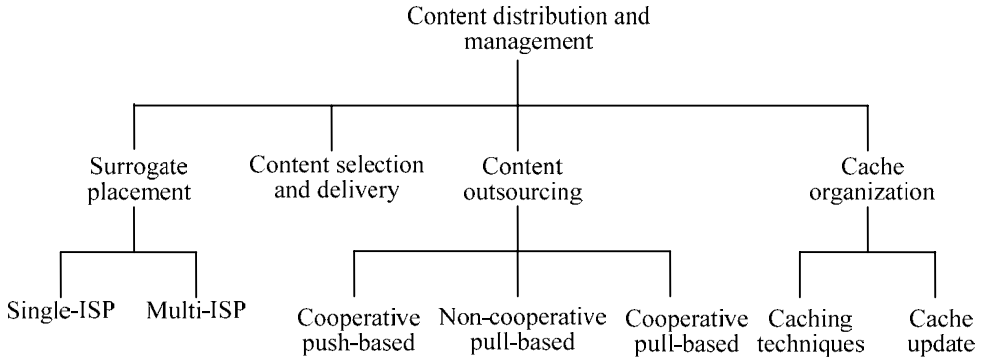

Figure 8: Content distribution and management taxonomy

Surrogate placement – Since location of surrogate servers is closely related to the content delivery process, it puts extra emphasis on the issue of choosing the best location for each surrogate. The goal of optimal surrogate placement is to reduce user perceived latency for accessing content and to minimize the overall network bandwidth consumption for transferring replicated content from servers to clients. The optimization of both of these metrics results in reduced infrastructure and communication cost for the CDN provider. Therefore, optimal placement of surrogate servers enables a CDN to provide high quality services and low CDN prices [62].

In this context, some theoretical approaches such as *minimum k-center problem* [9], *k-hierarchically well-separated trees (k-HST)* [9][10] have been proposed. These approaches model the server placement problem as the *center placement problem* which is defined as follows: for the placement of a given number of centers, minimize the maximum distance between a node and the nearest center. *k-HST* algorithm solves the server placement problem according to graph theory. In this approach, the network is represented as a graph $G(V,E)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of links. The algorithm consists of two phases. In the first phase, a node is arbitrarily selected from the complete graph (parent partition) and all the nodes which are within a random radius from this node form a new partition (child partition). The radius of the child partition is a factor of k smaller than the diameter of the parent partition. This process continues until each of the nodes is in a

partition of its own. Thus the graph is recursively partitioned and a tree of partitions is obtained with the root node being the entire network and the leaf nodes being individual nodes in the network. In the second phase, a virtual node is assigned to each of the partitions at each level. Each virtual node in a parent partition becomes the parent of the virtual nodes in the child partitions and together the virtual nodes form a tree. Afterwards, a greedy strategy is applied to find the number of centers needed for the resulted *k*-HST tree when the maximum center-node distance is bounded by *D*. On the other hand, the *minimum K-center problem* is NP-complete [69]. It can be described as follows: (1) Given a graph *G(V, E)* with all its edges arranged in non-decreasing order of edge cost *c: c(e1)* $\leq$ *c(e2)* $\leq \ldots \leq$ *c(em)*, construct a set of square graphs G^2_1, G^2_2,…., G^2_m. Each square graph of *G*, denoted by G^2 is the graph containing nodes *V* and edges *(u, v)* wherever there is a path between *u* and *v* in *G*. (2) Compute the maximal independent set M_i for each G^2_i. An independent set of G^2 is a set of nodes in G that are at least three hops apart in *G* and a maximal independent set *M* is defined as an independent set V' such that all nodes in $V - V'$ are at most one hop away from nodes in V'. (3) Find smallest *i* such that $M_i \leq K$, which is defined as *j*. (4) Finally, M_j is the set of *K* center.

Due to the computational complexity of these algorithms, some heuristics such as *Greedy replica placement* [11] and *Topology-informed placement strategy* [13] have been developed. These suboptimal algorithms take into account the existing information from CDN, such as workload patterns and the network topology. They provide sufficient solutions with lower computation cost. The *greedy algorithm* chooses *M* servers among *N* potential sites. In first iteration, the cost associated with each site is computed in the first iteration. It is assumed that access from all clients converges to the site under consideration. Hence, the lowest-cost site is chosen. In the second iteration, the greedy algorithm searches for a second site (yielding the next lowest cost) in conjunction with the site already chosen. The iteration continues until *M* servers have been chosen. The greedy algorithm works well even with imperfect input data. But it requires the knowledge of the clients locations in the network and all pair wise inter-node distances. In *topology-informed placement strategy*, servers are placed on candidate hosts in descending order of outdegrees (i.e. the number of other nodes connected to a node). Here the assumption is that nodes with more outdegrees can reach more nodes with smaller latency. This approach uses Autonomous Systems (AS) topologies where each node represents a single AS and node link corresponds to BGP peering. In an improved topology-informed placement strategy [70] router-level Internet topology is used instead of AS-level topology. In this approach, each LAN associated with a router is a potential site to place a server, rather than each AS being a site.

Also some other server placement algorithms like *Hot Spot* [12] and *Tree-based* [14] replica placement are also used in this context. The *hotspot* algorithm places replicas near the clients generating greatest load. It sorts the *N* potential sites according to the amount of traffic generated surrounding them and places replicas at the top *M* sites that generate maximum traffic. The *tree-based* replica placement algorithm is based on the assumption that the underlying topologies are trees. This algorithm models the replica placement problem as a dynamic programming problem. In this approach, a tree *T* is divided into several small trees T_i and placement of *t* proxies is achieved by placing t'_i proxies in the best way in each small tree T_i, where $t = \sum_i t'_i$. Another example is Scan [15], which is a scalable replica management framework that generates replicas on demand and organizes them into an application-level multicast tree. This approach minimizes the number of replicas while meeting clients' latency constraints and servers' capacity constraints. Figure 9 shows different surrogate server placement strategies.

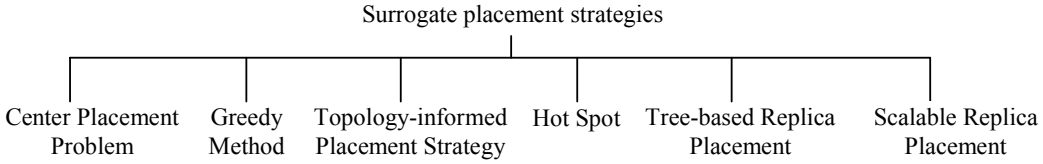

Figure 9: Surrogate placement strategies

For surrogate server placement, the CDN administrators also determine the optimal number of surrogate servers using *single-ISP* and *multi-ISP* approach [16]. In the *Single-ISP* approach, a CDN provider typically deploys at least 40 surrogate servers around the network edge to support content delivery [7]. The policy in a single-ISP approach is to put one or two surrogates in each major city within the ISP coverage. The ISP equips the surrogates with large caches. An ISP with global network can thus have extensive geographical coverage without relying on other ISPs. The drawback of this approach is that the surrogates may be placed at a distant place from the clients of the CDN provider. In *Multi-ISP* approach, the CDN provider places numerous surrogate servers at as many global ISP Points of Presence (POPs) as possible. It overcomes the problems with single-ISP approach and surrogates are placed close to the users and thus content is delivered reliably and timely from the requesting client's ISP. Some large CDN providers such as Akamai has more than 20000 servers

[2][3]. Other than the cost and complexity of setup, the main disadvantage of the multi-ISP approach is that each surrogate server receives fewer (or no) content requests which may result in idle resources and poor CDN performance [8]. Estimation of performance of these two approaches shows that single-ISP approach works better for sites with low-to-medium traffic volumes, while the multi-ISP approach is better for high-traffic sites [7].

Content selection and delivery – The efficiency of content delivery lies in the right selection of content to be delivered to the end-users. An appropriate content selection approach can assist in reduction of client download time and server load. Figure 10 shows the taxonomy of content selection and delivery techniques. Content can be delivered to the customers in full or in partial.

Full-site content selection and delivery: It is a simplistic approach where the entire set of origin server's objects is outsourced to surrogate servers. In other words, in this approach, the surrogate servers perform 'entire replication' in order to deliver the total content site to the end-users. With this approach, a content provider configures its DNS in such a way that all client requests for its Web site are resolved by a CDN server, which then delivers all of the content. The main advantage of this approach is its simplicity. However, such a solution is not feasible considering the on-going increase in the size of Web objects. Although the price of storage hardware is decreasing, sufficient storage space on the edge servers is never guaranteed to store all the content from content providers. Moreover, since the Web content is not static, the problem of updating such a huge collection of Web objects is unmanageable.

Partial site content selection and delivery: On the other hand, in partial-site content selection and delivery, surrogate servers perform 'partial replication' to deliver only embedded objects – such as web page images – from the corresponding CDN. With partial-site content delivery, a content provider modifies its content so that links to specific objects have host names in a domain for which the CDN provider is authoritative. Thus, the base HTML page is retrieved from the origin server, while embedded objects are retrieved from CDN cache servers. A partial-site approach is better than the full-site approach in the sense that the former reduces loads on the origin server and on the site's content generation infrastructure. Moreover, due to the infrequent change of embedded content, a partial-site approach exhibits better performance.

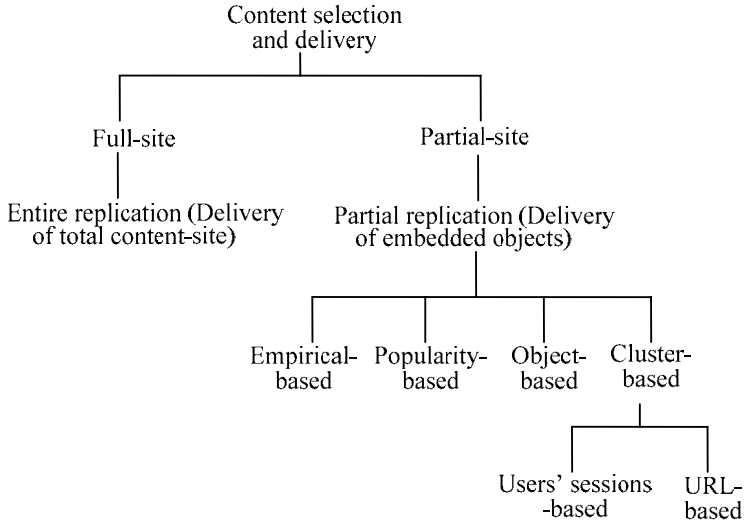

Figure 10: Taxonomy of content selection and delivery

Content selection is dependent on the suitable management strategy used for replicating Web content. Based on the approach to select embedded objects to perform replication, partial-site approach can be further divided into – *empirical*, *popularity*, *object* and *cluster*-based replication [17][18][144]. In *empirical*-based approach, the Web site administrator empirically selects the content to be replicated to the edge servers. Heuristics are used in making such an empirical decision. The main drawback of this approach lies in the uncertainty in choosing the right heuristics. In *popularity*-based approach, the most popular objects are replicated to the surrogates. This approach is time consuming and reliable objects request statistics is not guaranteed due to the popularity of each object varies considerably [18]. Moreover, such statistics are often not available for newly introduced content. In *object*-based approach, content is replicated to the surrogate servers in units of objects. This approach is greedy because each object is replicated to the surrogate server (under storage constraints) that gives the maximum performance gain [18][144]. Although such a greedy approach achieve the best performance, it suffers from high complexity to implement on real applications. In *cluster*-based approach, Web content is grouped based on either correlation or access frequency and is replicated in units of content

clusters. The clustering procedure is performed either by fixing the number of clusters or by fixing the maximum cluster diameter, since neither the number nor the diameter of the clusters can ever be known. The content clustering can be either *users' sessions-based* or *URL-based*. In *user's session-based* approach, Web log files [17] are used to cluster a set of users' navigation sessions, which show similar characteristics. This approach is beneficial because it helps to determine both the groups of users with similar browsing patterns and the groups of pages having related content. In *URL-based* approach, clustering of web content is done based on web site topology [17][18]. The most popular objects are identified from a Web site and are replicated in units of clusters where the correlation distance between every pair of URLs is based on a certain correlation metric. Experimental results show that content replication based on such clustering approaches reduce client download time and the load on servers. But these schemes suffer from the complexity involved to deploy them.

Content outsourcing – Given a set of properly placed surrogate servers in a CDN infrastructure and a chosen content for delivery, choosing an efficient content outsourcing practice is crucial. Content outsourcing is performed using either of *cooperative push-based*, *non-cooperative pull-based* and *cooperative pull-based* approaches.

Cooperative push-based: This approach is based on the pre-fetching of content to the surrogates. Content is pushed to the surrogate servers from the origin, and surrogate servers cooperate to reduce replication and update cost. In this scheme, the CDN maintains a mapping between content and surrogate servers, and each request is directed to the closest surrogate server or otherwise the request is directed to the origin server. Under this approach, greedy-global heuristic algorithm is suitable for making replication decision among cooperating surrogate servers [25]. Still it is considered as a theoretical approach since it has not been used by any CDN provider [17][18].

Non-cooperative pull-based: In this approach, client requests are directed (either using DNS redirection or URL rewriting [17]) to their closest surrogate servers. If there is a cache miss, surrogate servers pull content from the origin server. Most popular CDN providers (e.g. Akamai, Mirror Image) use this approach. The drawback of this approach is that an optimal server is not always chosen to serve content request [71]. Many CDNs use this approach since the cooperative push-based approach is still at the experimental stage [8].

Cooperative pull-based: The cooperative pull-based approach differs from the non-cooperative approach in the sense that surrogate servers cooperate with each other to get the requested content in case of cache miss. In the cooperative pull-based approach client requests are directed to the closest surrogate through DNS redirection. Using a distributed index, the surrogate servers find nearby copies of requested content and store it in the cache. The cooperative pull-based approach is reactive wherein a data object is cached only when the client requests it. An academic CDN Coral [45] has implemented the cooperative pull-based approach using a variation of Distribution Hash Table (DHT).

The optimal placement of outsourced content is another quite important content distribution issue. In the context of content outsourcing, it is crucial to determine in which surrogate servers the outsourced content should be replicated. Several works can be found in literature demonstrating the effectiveness of different replication strategies for outsourced content. Kangasharju et al. [25] have used four heuristics, namely – *random*, *popularity*, *greedy-single*, and *greedy-global* for replication of outsourced content. Tse [145] has presented a set of greedy approaches where the placement is occurred by balancing the loads and sizes of the surrogate servers. Pallis et al. [146] have presented a self-tuning, parameterless algorithm called lat-cdn for optimally placing outsourced content in CDN's surrogate servers. This algorithm uses object's latency to make replication decision. An object's latency is defined as the delay between a request for a Web object and receiving the object in its entirety. An improved algorithm, called il2p is presented in [147], which places the outsourced objects to surrogate servers with respect to the latency and load of the objects.

Cache organization – Content management is essential for CDN performance, which is mainly dependent on the cache organization approach followed by the CDN. Cache organization is in turn composed of the caching techniques used and the frequency of cache update to ensure the freshness, availability and reliability of content. Other than these two, the cache organization may also include the integration of caching policies on a CDN's infrastructure. Such integration may be useful for a CDN for effective content management. Potential performance improvement is possible in terms of perceived latency, hit ratio and byte hit ratio if replication and caching is used together in a CDN [148]. Moreover, the combination of caching with replication fortifies CDNs against flash crowd events. In this context, Stamos et al. [150] have presented a generic non-parametric heuristic method that integrates Web caching with content replication. They have developed a placement similarity approach, called SRC, for evaluating the level of integration. Another integrated approach, called Hybrid, which combines static replication and Web caching using an analytic model of LRU is presented in [149]. The Hybrid gradually fills the surrogate servers caches with static content with each iteration, as long as it contributes to the optimization of response times.

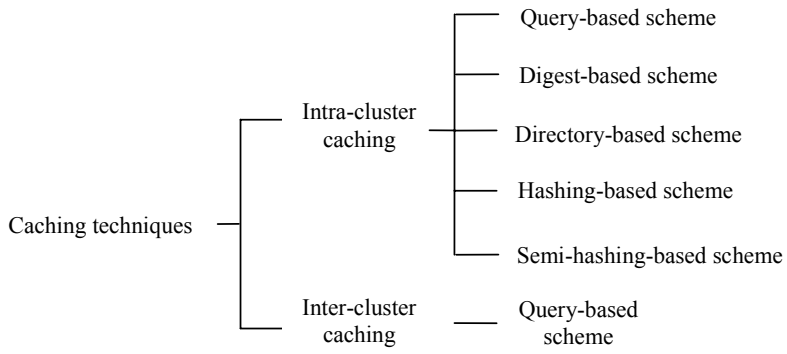

Figure 11: Caching techniques taxonomy

o *Caching techniques:* Replicating content is common and widely accepted phenomenon in large-scale distributed environments like CDNs, where content is stored in more than one location for performance and reliability reasons. Different replication strategies are suitable for different applications. A detailed survey of replication strategies in wide area distributed systems can be found in [139]. Replication in commercial CDNs is performed through caching content across the globe for high profile customers that need to deliver large volumes of data in a timely manner. Content caching in CDNs can be intra-cluster and inter-cluster basis. A taxonomy of caching techniques is shown in Figure 11.

Intra-cluster caching: For intra-cluster caching of content either of a *query-based* [28], *digest-based* [29], *directory-based* [30] or *hashing-based* scheme [31][32] can be used. In a *query-based* scheme, on a cache miss a CDN server broadcasts a query to other cooperating CDN servers. The problems with this scheme are the significant query traffic and the delay because a CDN server have to wait for the last 'miss' reply from all the cooperating surrogates before concluding that none of its peers has the requested content. Because of these drawbacks, the query-based scheme suffers from implementation overhead. The *digest-based* approach overcomes the problem of flooding queries in query-based scheme. In digest-based scheme, each of the CDN servers maintains a digest of content held by the other cooperating surrogates. The cooperating surrogates are informed about any sort of update of the content by the updating CDN server. On checking the content digest, a CDN server can take the decision to route a content request to a particular surrogate. The main drawback is that it suffers from update traffic overhead, because of the frequent exchange of the update traffic to make sure that the cooperating surrogates have correct information about each other. The *directory-based* scheme is a centralized version of the digest-based scheme. In directory-based scheme, a centralized server keeps content information of all the cooperating surrogates inside a cluster. Each CDN server only notifies the directory server when local updates occur and queries the directory server whenever there is a local cache miss. This scheme experiences potential bottleneck and single point of failure since the directory server receives update and query traffic from all cooperating surrogates. In a *hashing-based* scheme the cooperating CDN servers maintain the same hashing function. A designated CDN server holds a content based on content's URL, IP addresses of the CDN servers, and the hashing function. All requests for that particular content is directed to that designated server. Hashing-based scheme is more efficient than other schemes since it has smallest implementation overhead and highest content sharing efficiency. However, it does not scale well with local requests and multimedia content delivery since the local client requests are directed to and served by other designated CDN servers. To overcome this problem, a semi-hashing-based scheme [33][34] can be followed. Under the semi-hashing-based scheme, a local CDN server allocates a certain portion of its disk space to cache the most popular content for its local users and the remaining portion to cooperate with other CDN servers via a hashing function. Like pure hashing, semi-hashing has small implementation overhead and high content sharing efficiency. In addition, it has been found to significantly increase the local hit rate of the CDN.

Inter-cluster caching: Inter-cluster content routing is necessary when intra-cluster content routing fails. A hashing-based scheme is not appropriate for inter-cluster cooperative caching, because representative CDN servers of different clusters are normally distributed geographically. The digest-based or directory-based scheme is also not suitable for inter-cluster caching since the representative CDN servers have to maintain a huge content digest and/or directory including the content information of CDN servers in other clusters. Hence, a query-based scheme can be used for inter-cluster caching as stated in [34]. In this approach, when a cluster fails to serve a content request, it queries other neighboring cluster(s). If the content is obtainable from this neighbor, it replies with a 'hit' message or if not, it forwards the request to other neighboring clusters. All the CDN servers inside a cluster use hashing based scheme for serving content request and the representative CDN server of a cluster only queries the designated server of that cluster to serve a content request. Hence, this scheme uses the hashing-based scheme for intra-cluster content routing and the query-based scheme for inter-cluster content

routing. This approach improves performance since it limits flooding of query traffic and overcomes the problem of delays when retrieving content from remote servers through the use of a Timeout value and TTL (Time-to-Live) number with each query message.

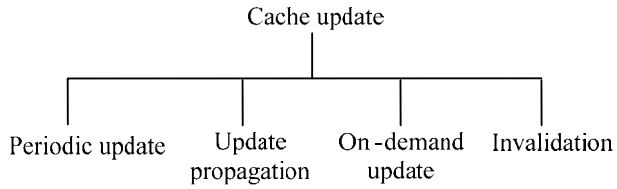

Figure 12: Cache update taxonomy

- *Cache update*: Cached objects in the surrogate servers of a CDN have associated expiration times after which they are considered stale. Ensuring the freshness of content is necessary to serve the clients with up to date information. If there are delays involved in propagating the content, a CDN provider should be aware that the content may be inconsistent and/or expired. To manage the consistency and freshness of content at replicas, CDNs deploy different cache update techniques. The taxonomy of cache update mechanisms is shown in Figure 12.

The most common cache update method is the *periodic update*. To ensure content consistency and freshness, the content provider configures its origin Web servers to provide instructions to caches about what content is cacheable, how long different content is to be considered fresh, when to check back with the origin server for updated content, and so forth [89]. With this approach, caches are updated in a regular fashion. But this approach suffers from significant levels of unnecessary traffic generated from update traffic at each interval. The *update propagation* is triggered with a change in content. It performs active content pushing to the CDN cache servers. In this mechanism, an updated version of a document is delivered to all caches whenever a change is made to the document at the origin server. For frequently changing content, this approach generates excess update traffic. *On-demand update* is a cache update mechanism where the latest copy of a document is propagated to the surrogate cache server based on prior request for that content. This approach follows a assume nothing structure and content is not updated unless it is requested. The disadvantage of this approach is the back and forth traffic between the cache and origin server in order to ensure that the delivered content is the latest. Another cache update approach is *invalidation*, in which an invalidation message is sent to all surrogate caches when a document is changed at the origin server. The surrogate caches are blocked from accessing the documents when it is being changed. Each cache needs to fetch an updated version of the document individually later. The drawback of this approach is that it does not make full use of the distribution network for content delivery and belated fetching of content by the caches may lead to inefficiency of managing consistency among cached contents.

Generally CDNs give the content provider control over freshness of content and ensure that all CDN sites are consistent. However, content providers themselves can build their own policies or use some heuristics to deploy organization specific caching policies. In the first case, content providers specify their caching policies in a format unique to the CDN provider, which propagates the rule sets to its caches. These rules specify instructions to the caches on how to maintain the freshness of content through ensuring consistency. In the later case, a content provider can apply some heuristics rather than developing complex caching policies. With this approach, some of the caching servers adaptively learn over time about the frequency of change of content at the origin server and tune their behavior accordingly.

3.3. Request-routing

A *request-routing system* is responsible for routing client requests to an appropriate surrogate server for the delivery of content. It consists of a collection of network elements to support request-routing for a single CDN. It directs client requests to the replica server 'closest' to the client. However, the closest server may not be the best surrogate server for servicing the client request [151]. Hence, a request-routing system uses a set of metrics such as network proximity, client perceived latency, distance, and replica server load in an attempt to direct users to the closest surrogate that can best serve the request. The content selection and delivery techniques (i.e. full-site and partial-site) used by a CDN have a direct impact on the design of its request-routing system. If the full-site approach is used by a CDN, the request-routing system assists to direct the client requests to the surrogate servers as they hold all the outsourced content. On the other hand, if the partial-site approach is used, the request-routing system is designed in such a way that on receiving the client request, the origin server delivers the basic content while surrogate servers deliver the embedded objects. The request-routing system in a CDN has two parts: deployment of a request-routing algorithm and use of a request-routing mechanism [62]. A request-routing algorithm is invoked on receiving a client request. It specifies how to select an edge server in

response to the given client request. On the other hand, a request-routing mechanism is a way to inform the client about the selection. Such a mechanism at first invokes a request-routing algorithm and then informs the client about the selection result it obtains.

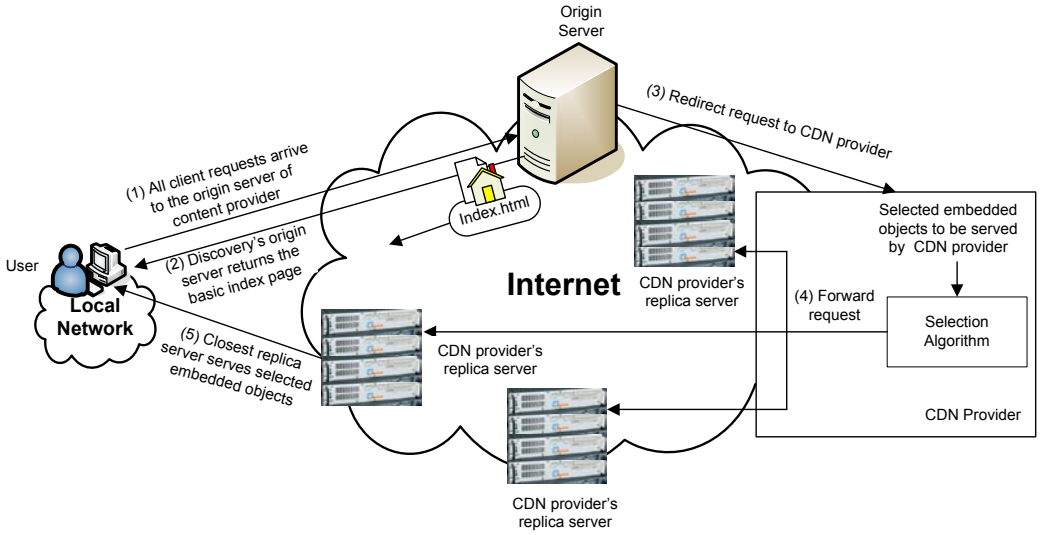

Figure 13: Request-routing in a CDN environment

Figure 13 provides a high-level view of the request-routing in a CDN environment. The interaction flows are: (1) the client requests content from the content provider by specifying its URL in the Web browser. Client's request is directed to its origin server; (2) when origin server receives a request, it makes a decision to provide only the basic content (e.g. index page of the Web site) that can be served from its origin server; (3) to serve the high bandwidth demanding and frequently asked content (e.g. embedded objects – fresh content, navigation bar, banner ads etc.), content provider's origin server redirects client's request to the CDN provider; (4) using the proprietary selection algorithm, the CDN provider selects the replica server which is 'closest' to the client, in order to serve the requested embedded objects; (5) selected replica server gets the embedded objects from the origin server, serves the client requests and caches it for subsequent request servicing.

3.3.1. Request-routing algorithms

The algorithms invoked by the request-routing mechanisms can be adaptive or non-adaptive (Figure 14). Adaptive algorithms consider the current system condition to select a cache server for content delivery. Current condition of the system is obtained by estimating some metrics like load on the replica servers or the congestion of selected network links. Non-adaptive request-routing algorithms use some heuristics for selecting a cache server rather than considering the current system condition. A non-adaptive algorithm is easy to implement, while the former is more complex. Complexity of adaptive algorithms arises from their ability to change behavior to cope with an enduring situation. A non-adaptive algorithm works efficiently when the assumptions made by the heuristics are met. On the other hand, an adaptive algorithm demonstrates high system robustness [113] in the face of events like flash crowds.

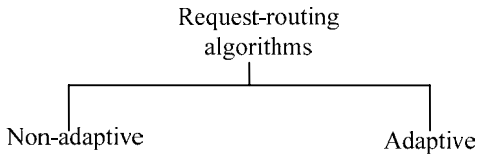

Figure 14: Taxonomy of request-routing algorithms

Non-adaptive request-routing algorithms: An example of the most common and simple non-adaptive request-routing algorithm is round-robin, which distributes all requests to the CDN cache servers and attempts to balance load among them [114]. It is assumed that all the cache servers have similar processing capability and that any of them can serve any client request. Such simple algorithms are efficient for clusters, where all the replica servers are located at the same place [115]. But the round-robin request-routing algorithm does not perform well for wide area distributed systems where the cache servers are located at distant places. In this case it does not consider the distance of the replica servers. Hence, client requests may be directed to more distant

replica servers, which cause poor performance perceived by the users. Moreover, the aim of load balancing is not fully achieved since processing different request can involve significantly different computational costs.

In another non-adaptive request-routing algorithm, all replica servers are ranked according to the predicted load on them. Such prediction is done based on the number of requests each of the servers has served so far. This algorithm takes client-server distance into account and client requests are directed to the replica servers in such a way that load is balanced among them. The assumption here is that the replica server load and the client-server distance are the most influencing factors for the efficiency of request processing [62]. Though it has been observed in [116] that deploying this algorithm can perform well for request-routing, the client perceived performance may still be poor.

Several other interesting non-adaptive request-routing algorithms are implemented in the Cisco DistributedDirector [117]. One of these algorithms considers the percentage of client requests that each replica server receives. A server receiving more requests is assumed to be more powerful. Hence, client requests are directed to the more powerful servers to achieve better resource utilization. Another algorithm defines preference of one server over another in order to delegate the former to serve client requests. The DistributedDirector also supports random request distribution to replica servers. Furthermore, some other non-adaptive algorithms can be found which considers the client's geographic location to redirect requests to the nearby replica. But this algorithm suffers from the fact that client requests may be assigned to overloaded replica servers, which may degrade client perceived performance.

Karger et al. [32] have proposed a non-adaptive algorithm which calculates a hashing function *h* from a large space of identifiers, based on the URL of the content. This hashing function is used to efficiently route client requests to a logical ring consisting of cache servers with IDs from the same space. It is assumed that the cache server having the smallest ID larger than *h* is responsible for holding the referenced data. Hence, client requests are directed to it. Variations of this algorithm has been used in context of intra-cluster caching [33][34] and P2P file sharing systems [118].

Adaptive request-routing algorithms: Globule [46] uses an adaptive request-routing algorithm that selects the replica server closest to the clients in terms of network proximity [114]. The metric estimation in Globule is based on path length which is updated periodically. The metric estimation service used in globule is passive, which does not introduce any additional traffic to the network. However, results in [119] show that the distance metric estimation procedure is not very accurate.

Andrews et al. [120] and Ardiaz et al. [121] have proposed adaptive request-routing algorithms based on client-server latency. In this approach, either client access logs or passive server-side latency measurements are taken into account, and the algorithms decide to which replica server the client requests are to be sent. Hence, they redirect a client request to a replica which has recently reported the minimal latency to the client. These algorithms are efficient since they consider latency measurements. However, they require the maintenance of central database of measurements, which limits the scalability of systems on which these algorithms are deployed.

Cisco DistributedDirector [117] has implemented an adaptive request-routing algorithm. The request-routing algorithm deployed in this system takes into account a weighted combination of three metrics, namely – inter-AS distance, intra-AS distance, and end-to-end latency. Though this algorithm is flexible since it makes use of three metrics, the deployment of an agent in each replica server for metric measurement makes it complex and costly. Moreover, the active latency measurement techniques used by this algorithm introduce additional traffic to the Internet. Furthermore, the isolation of DistributedDirector component from the replica server makes it unable to probe the servers to obtain their load information.

Akamai [2][3] uses a complex adaptive request-routing algorithm. It takes into consideration a number of metrics such as replica server load, the reliability of loads between the client and each of the replica servers, and the bandwidth that is currently available to a replica server. This algorithm is proprietary to Akamai and the technology details have not been revealed.

3.3.2. Request-routing mechanisms

Request-routing mechanisms inform the client about the selection of replica server, generated by the request-routing algorithms. Request-routing mechanisms can be classified according to several criteria. In this section we classify them according to the variety of request processing. As shown in Figure 15, they can be classified as: Global Server Load Balancing (GSLB) [27], DNS-based request-routing [16][39], HTTP redirection [16][19], URL rewriting [7], anycasting [19], and CDN peering [16] [49].

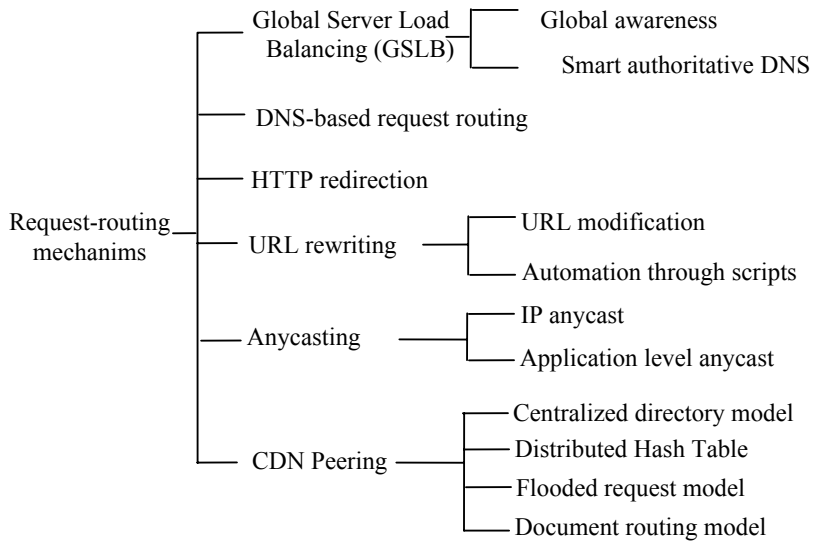

Figure 15: Taxonomy of request-routing mechanisms

These routing schemes are stated in the following:

Global Server Load Balancing (GSLB): In this approach, service nodes (which serve content to end-users) consisting of a GSLB-enabled Web switch and a number of real Web servers are distributed in several locations around the world. Two new capabilities of the service nodes allow them to support global server load balancing. The first is *global awareness* and the second is *smart authoritative DNS* [27]. In local server load balancing each service node is aware of the health and performance information of the Web servers directly attached to it. In GSLB, one service node is aware of the information in other service nodes and includes their virtual IP address in its list of servers. Hence, the Web switches making up each service node are globally aware and each knows the addresses of all the other service nodes. They also exchange performance information among the Web switches in GSLB configuration. To make use of such global awareness, the GSLB switches act as a smart authoritative DNS for certain domains. The advantage of GSLB is that since the service nodes are aware of each other, each GSLB switch can select the best surrogate server for any request. Thus this approach facilitates choosing servers not only from the pool of locally connected real servers, but also the remote service nodes. Another significant advantage of GSLB is that the network administrator can add GSLB capability to the network without adding any additional networking devices. A disadvantage of GSLB is the manual configuration of the service nodes to enable them with GSLB capability.

DNS-based request-routing: In this approach, the content distribution services rely on the modified DNS servers to perform the mapping between a surrogate server's symbolic name and its numerical IP address. It is used for full-site content selection and delivery. In DNS-based request-routing, a domain name has multiple IP addresses associated to it. When an end-user's content request comes, the DNS server of the service provider returns the IP addresses of servers holding the replica of the requested object. The client's DNS resolver chooses a server among these. To decide, the resolver may issue probes to the servers and choose based on response times to these probes. It may also collect historical information from the clients based on previous access to these servers. Both full and partial-site CDN providers use DNS redirection. The performance and effectiveness of DNS-based request-routing has been examined in a number of recent studies [22][90][91][92]. The advantage of this approach is the transparency as the services are referred to by means of their DNS names, and not their IP addresses. DNS-based approach is extremely popular because of its simplicity and independence from any actual replicated service. Since it is incorporated to the name resolution service it can be used by any Internet application [62]. In addition, the ubiquity of DNS as a directory service provides advantages during request-routing. The main disadvantage of DNS-based request-routing is that, it increases network latency because of the increase in DNS lookup times. CDN administrators typically resolve this problem by splitting CDN DNS into two levels (low-level DNS and high-level DNS) for load distribution [22]. Another disadvantage is that it does not take into account the IP address of the clients. The knowledge of Internet location of the client DNS server limits the ability of the request-routing system to determine a client's proximity to the surrogate. Another limitation of this approach is that it is not scalable and it has limited control on client identification since a DNS query does not carry the addresses of the querying client. Most significantly, DNS cannot be relied upon to control all incoming requests due to caching of DNS data at both the ISP and client level. Indeed, it can have control over as little as 5% of requests in many instances [88]. Furthermore, since clients do not access the

actual domain names that serve their requests, it leads to the absence of any alternate server to fulfill client requests in case of failure of the target surrogate server.

HTTP redirection: The approach propagates information about replica server sets in HTTP headers. HTTP protocols allow a Web server to respond to a client request with a special message that tells the client to re-submit its request to another server. HTTP redirection can be used for both full-site and partial-site content selection and delivery. This mechanism can be used to build a special Web server, which accepts client requests, chooses replica servers for them and redirects clients to those servers. It requires changes to both Web servers and clients to process extra headers. Clients must also be modified to implement replica server selection. The main advantage of this approach is flexibility and simplicity. Another advantage is that replication can be managed at fine granularity, since individual web pages are considered as a granule [19]. The most significant disadvantage of HTTP redirection is the lack of transparency. Moreover, the overhead perceived through this approach is significant since it introduces extra message round-trip into request processing as well as over HTTP.

URL rewriting or *Navigation hyperlink:* Though most CDN systems use a DNS based routing scheme, some systems use the URL rewriting. It is mainly used for partial-site content selection and delivery where embedded objects are sent as a response to client requests. In this approach, the origin server redirects the clients to different surrogate servers by rewriting the dynamically generated pages' URL links. For example, with a Web page containing an HTML file and some embedded objects, the Web server would modify references to embedded objects so that the client could fetch them from the best surrogate server. To automate this process, CDNs provide special scripts that transparently parse Web page content and replace embedded URLs [22]. URL rewriting can be pro-active or reactive. In the pro-active URL rewriting, the URLs for embedded objects of the main HTML page are formulated before the content is loaded in the origin server. In reactive approach involves rewriting the embedded URLs of a HTML page when the client request reaches the origin server. The main advantage of URL rewriting is that the clients are not bound to a single surrogate server, because the rewritten URLs contain DNS names that point to a group of surrogate servers. Also finer level of granularity can be achieved through this approach since embedded objects can be considered as granule. The disadvantages through this approach are the delay for URL-parsing and the possible bottleneck introduced by an in-path element. Another disadvantage is that content with modified reference to nearby surrogate server rather than to the origin server is non-cacheable.

Anycasting: The anycasting approach can be divided into two: IP anycasting and Application-level anycasting. IP anycasting, proposed by Partridge et al. [23], assumes that the same IP address is assigned to a set of hosts and each IP router holds a path in its routing table to the host that is closest to this router. Thus, different IP routers have paths to different hosts with the same IP address. IP anycasting can be suitable for request-routing and service location. It targets network-wide replication of the servers over potentially heterogeneous platforms. A disadvantage of IP anycasting is some part of IP address space is to be allocated for anycast address. Fei et al. [24] proposed an application level anycasting mechanism where the service consists of a set of *anycast resolvers*, which performs the *anycast domain names* to IP address mapping. Clients interact with the anycast resolvers by generating an anycast query. The resolver processes the query and replies with an anycast response. A Metric database, associated with each anycast resolver contains performance data about replica servers. The performance is estimated based on the load and the request processing capability of the servers. The overhead of the performance measurement is kept at a manageable level. The performance data can be used in the selection of a server from a group, based on user-specified performance criteria. An advantage of application level anycasting is that better flexibility can be achieved through this approach. One disadvantage of this approach is, deploying the anycasting mechanism for request-routing requires changes to the servers as well as to the clients. Hence, it may lead to increased cost considering possibly large number of servers and clients.

CDN peering: Peer-to-peer content networks are formed by symmetrical connections between host computers. Peered CDNs deliver content on each other's behalf. Thus, a CDN could expand its reach to a larger client population by using partnered CDN servers and their nearby forward proxies. A content provider usually has contracts with only one CDN and each CDN contacts other peer CDNs on the content provider's behalf [49]. Peering CDNs are more fault-tolerant as the necessary information retrieval network can be developed on the peering members themselves instead of relying on a dedicated infrastructure like traditional CDNs. To locate content in CDN peering, a *centralized directory model*, *Distributed Hash Table (DHT)*, *flooded request model* or *document routing model* can be used [27][95]. In a centralized directory model, peers contact a centralized directory where all the peers publish content that they want to share with others. When the directory receives a request it responses with the information of the peer that holds the requested content. When more than one peer matches the request, the best peer is selected based on metrics such as network proximity, highest bandwidth, least congestion and highest capacity. On receiving the response from the directory, the requesting peer contacts the peer that it has been referred to for content retrieval. The drawback of this approach is that, the centralized directory is subject to a single point of failure. Moreover, the scalability of a system based on a centralized directory is limited to the capacity of the directory. Archi [123], WAIS [124] are the examples of centralized

directory systems for retrieving FTP files located on various systems. In systems using DHTs, peers are indexed through hashing keys within a distributed system. Then a peer holding the desired content can be found through applying complex queries [126]. The advantage of this approach is the ability to perform load balancing by offloading excess loads to the less-loaded peers [127]. In the flooded request model, a request from a peer is broadcast to the peers directly connected to it. These peers in turn forwards the message to other peers directly connected to them. This process continues until the request is answered or some broadcast limit is reached. The drawback of this approach is that it generates unnecessary network traffic and hence, it requires enormous bandwidth. Thus, it suffers from scalability problem and it limits the size of the network [27]. Gnutella [93][94] is the example of a system using the flooded request model. In document routing model an authoritative peer is asked for referral to get the requested content. Each peer in the model is helpful, though they partially complete the referral information. Each referral moves to the requester closer to a peer that can satisfy the query [27]. The main advantage of this approach is that it can complete a comprehensive search within a bounded number of steps. Moreover, it shows good performance and it is scalable enough to grow significantly large. Chord [125] is the example of a system using this model.

3.4. Performance measurement

Performance measurement of a CDN is done to measure its ability to serve the customers with the desired content and/or service. Customers utilizing the service need feedback on the performance of the CDN as a whole. Performance measurement offers the ability to predict, monitor and ensure the end-to-end performance of a CDN. The measurement is achieved with a combination of hardware and software-based probes distributed around the CDN, as well as using the logs from various servers within the CDN. Typically five key metrics are used by the content providers to evaluate the performance of a CDN [7][22][72]. Those are:

- *Cache hit ratio:* It is defined as the ratio of the number of cached documents versus total documents requested. A high hit rate reflects that a CDN is using an effective cache policy to manage its caches.
- *Reserved bandwidth:* It is the measure of the bandwidth used by the origin server. It is measured in bytes and is retrieved from the origin server.
- *Latency:* It refers to the user perceived response time. Reduced latency signifies the decreases in bandwidth reserved by the origin server.
- *Surrogate server utilization:* It refers to the fraction of time during which the surrogate servers remain busy. This metric is used by the administrators to calculate CPU load, number of requests served and storage I/O usage.
- *Reliability:* Packet-loss measurements are used to determine the reliability of a CDN. High reliability indicates that a CDN incurs less packet loss and is always available to the clients.

Performance measurement can be accomplished based on internal performance measures as well as from the customer perspective. Thus, it involves the usage of internal measurement technologies and external services (e.g. MediaMetrics and Nielsen ratings). However, a CDN provider's own performance testing can be misleading, since it may perform well for a particular Web site and/or content, but perform poorly for others. To ensure reliable and efficient performance measurement, it can also be performed by independent third-party such as Keynote Systems [128] or Giga Information Group [129]. The performance measurement taxonomy is shown in Figure 16.

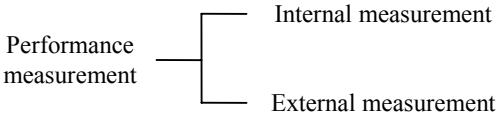

Figure 16: Performance measurement taxonomy

Internal measurement – Measurement of the content delivery services throughout the network can be performed by collecting and analyzing the logs from the caches and streaming media servers. CDN servers could be equipped with the ability to collect statistics in order to get an end-to-end measurement of its performance. In addition, deployment of probes (hardware and/or software) throughout the network and correlation of the information collected by probes with the cache and server logs can be used to measure the end-to-end performance of the CDN. Internal measurement can also be performed by using some third-party tools that can produce graphical representation of performance data.

External measurement – In addition to internal performance measurement, external measurement of performance by an independent third-party informs the CDN customers about the verified and guaranteed performance. This process is efficient since the independent performance-measuring companies support benchmarking networks of strategically located measurement computers connected through major Internet backbones in several cities. These computers measure how a particular Web site performs from the end-user's perspective, considering meaningful metrics on Web application performance in critical areas [16]. Giga

Information Group is one such evaluator that measure performance by analyzing a CDN in terms of several parameters such as scalability/availability, manageability, content type support, ease of use, cost, coverage, reporting, and viability.

3.4.1. Network statistics acquisition

For internal and external performance measurement, different network statistics acquisition techniques are deployed based on several parameters. Such techniques may involve network probing, traffic monitoring, and feedback from surrogates. Typical parameters in the network statistics acquisition process include geographical proximity, network proximity, latency, server load, and server performance as a whole. Figure 17 presents the mechanisms used by the CDNs to perform network statistics acquisition.

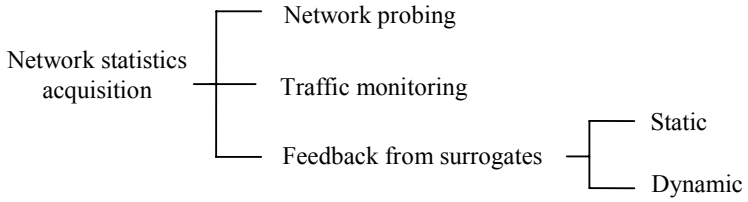

Figure 17: Network statistics acquisition techniques

Network probing: Network probing is a measurement technique where the possible requesting entities are probed in order to determine one or more metrics from each surrogate or a set of surrogates. Example of such probing technique is an ICMP ECHO message that is sent periodically from a surrogate or a set of surrogates to a potential requesting entity. Active probing techniques are sometimes not suitable and limited for some reasons. It introduces additional network latency which may be significant for small Web requests. Moreover, performing several probes to an entity often triggers intrusion-detection alerts, resulting in abuse complaints [130]. Probing sometimes may lead to an inaccurate metric as ICMP traffic can be ignored or reprioritized due to concerns of DDoS (Distributed Denial of Service) attacks. A distributed anycasting system by Freedman et al. [130] has shown that ICMP probes and TCP probes to high random ports are often dropped by firewalls and flagged as unwanted port scans.

Traffic monitoring: Traffic monitoring is a measurement technique where the traffic between the client and the surrogate is monitored to know the actual performance metrics. Once the client connects, the actual performance of the transfer is measured. This data is then fed back into the request-routing system. An example of such traffic monitoring is to watch the packet loss from a client to a surrogate or the user perceived latency by observing the TCP behavior. Response time (latency) is the simplest and mostly used distance metric, which can be estimated by monitoring of the number of packets (i.e. traffic) traveled along the route between client and the surrogate. A metric estimation system such as IDMaps [131] measures and disseminates distance information on the global Internet in terms of latency and bandwidth. This system considers two types of distance information based on timeliness — load sensitive and "raw" (where distance information is obtained considering no load on the network). The estimation of these information is performed through traffic monitoring with an update frequency on the order of days, or if necessary, hours.

Feedback from surrogates – Feedback information can be obtained by periodically probing a surrogate by issuing application specific requests (e.g. HTTP) and taking related measures. Feedback information can also be obtained from agents that are deployed in the surrogates. These agents can communicate a variety of metrics about their nodes. Methods for obtaining feedback information can be static or dynamic. Static methods select a route to minimize the number of hops or to optimize other static parameters. Dynamic probing allows computing round-trip time or other QoS parameters in 'real time' [39].

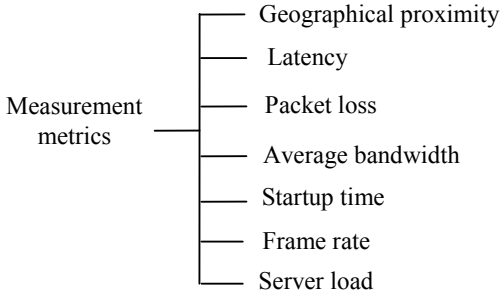

Figure 18: Metrics used for measuring network and system performance

Network statistics acquisition methods rely on a number of metrics to obtain CDN status information. Figure 18 shows the different metrics used by CDNs to measure the network and system performance. *Geographical proximity* is a measure of identifying a user's location within a certain region. It is often used to redirect all users within a certain region to the same Point of Presence (POP). The measurement of such network proximity is typically derived through probing of Border Gateway Protocol (BGP) routing tables. The end-user perceived *response time (latency)* is a useful metric to select the suitable surrogate (i.e. POP) for that user. *Packet loss* information through a network path is a measurement metric that is used to select the path with lowest error rate. *Average bandwidth*, *startup time* and *frame rate* are the metrics used to select the best path for streaming media delivery. Server load state can be computed based on metrics such as CPU load, network interface load, active connection and storage I/O load. This metric is used to select the server with the aggregated least load.

3.4.2. Performance measurement through simulation

Other than using internal and external performance measurement, researchers use simulation tools to measure a CDN's performance. Some researchers also experiment their CDN policies on real platforms such as PlanetLab [111]. The CDN simulators implemented in software are valuable tools for researchers to develop, test and diagnose a CDN's performance, since accessing real CDN traces and logs is not easy due to the proprietary nature of commercial CDNs. Such a simulation process is economical because of no involvement of actual hardware to carry out the experiments. Moreover, it is flexible because it is possible to simulate a link with any bandwidth and propagation delay and a router with any queue size and queue management policy. A simulated network environment is free of any uncontrollable factors such as unwanted external traffic, which the researchers may experience while running experiments in real network. Hence, simulation results are reproducible and easy to analyze. A wide range of network simulators [153][154] are available which can be used to simulate a CDN to measure its performance. Moreover, there are also some specific CDN simulation systems [18][25][113][152][153] that allow a (closely) realistic approach to research community and the CDNs' developers to measure a CDN's performance and experiment their CDN policies. However, the results obtained from a simulation may be misleading if a CDN simulation system does not take into account several critical factors such as the bottleneck that are likely to occur in a network, the number of sessions that can serve each network element (e.g. router, surrogate server), and the number of traversed nodes considering the TCP/IP network infrastructure.

4. Existing CDNs: A survey

In this section, we provide a state-of-the-art survey of the existing CDNs. Many commercial CDNs (e.g. Akamai, Adero, Digital Island, Mirror Image, Inktomi, Limelight Networks etc.) as well as academic CDNs (e.g. Coral, Codeen, Globule etc.) are present in the content distribution space. Table 1 and Table 2 show a list of these CDNs and a brief summary of each of them. In Section 5, these CDNs are categorized according to the taxonomy presented in Section 3.

Most or all of the operational CDNs are developed by commercial companies which are subject to consolidation over time due to acquisition and/or mergers. Hence, in the survey, we focus on studying those CDNs that have been in stable operation for a significant period of time. In this context, it is worth mentioning that many CDN-specific information such as fees charged by CDNs, existing customers of CDNs are ignored since they are highly likely to change quickly over time. Therefore, the information provided in this section is expected to be stable and up-to-date. However, for readers' understanding on how a CDN charges its customers, we provide a brief discussion on the pricing policies used for CDN services.

As mentioned earlier, a CDN charges its customers according to the content delivered (i.e. traffic) to the end-users by its surrogate servers. There are technical and business challenges in pricing CDN services [8]. However, most of the commercial CDNs do not reveal any information about the strategies used by them to charge their services. In the CDN research community, pricing of CDN services is a relatively new and unexplored area. The use of analytical models to address the optimal prices of CDN services is presented in [161]. It shows that CDN pricing polices should consider providing volume discount to content providers. Such an approach is consistent with current industry practices. It concludes that the recent trends such as decreasing bandwidth cost will have impact on CDN pricing policies and the price of CDN services will decline, while the content delivery process on a Web site will accelerate. In another work Hosanagar et al. [162] demonstrates the importance of realizing an optimal pricing strategy for CDN services under varying traffic patterns, the adoption driver of CDN services, and the drivers of profitability within CDN services. It also reveals that the pure usage-based pricing strategy used by most commercial CDNs is suboptimal in cases with varying level of traffic burstiness. Therefore, it concludes that a percentile-based pricing strategy can be used in this case that allows volume discount for content providers with high mean traffic and also additional charges for content providers with highly bursty traffic.

Table 1: Summary of existing commercial content delivery networks

CDN Name	Service Type	Coverage	Products/Solutions (If any)
Accellion www.accellion.com	Provides large file delivery service	Covers industries such as advertising/media production, healthcare, manufacturing, consumer goods, higher education etc.	Accellion Courier Secure File Transfer Appliances (SFTA) for on-demand and secure exchange of large-size files, and Accellion Backup and Recovery Services (BRS) as online desktop and server backup and recovery solution
Akamai www.akamai.com	Provides CDN service, including streaming	Covers 85% of the market. 20,000 servers in nearly 1,000 networks in 71 countries. It handles 20% of total Internet traffic today.	Edge Platform for handling static as well as dynamic content, Edge Control for managing applications, and Network Operations Control Center (NOCC)
AppStream www.appstream.com	Provides on demand software distribution and software license management tools for extended enterprises	Small business to Fortune 1,000 corporations, educational institutions and government	AppStream Software V5.0 including AppStream server, client and end-user application portal
EdgeStream www.edgestream.com	Provides disrupted video streaming applications over the public Internet	Provides video streaming over consumer cable or ADSL modem connections around the globe, even over paths that have 20 router hops between server and end-user	EdgeStream video on-demand and IPTV Streaming software for video streaming
Globix www.globix.com	Provides Internet infrastructure and network services	Consists of both a trans-Atlantic/trans-continental IP backbone as well as an optical network throughout the Northeast and mid-Atlantic regions. It has more than 1200 customers	N/A
Limelight Networks www.limelightnetworks.com	Provides distributed on-demand and live delivery of video, music, games and download	Surrogate servers located in 72 locations around the world	Limelight ContentEdge for distributed content delivery via HTTP, Limelight MediaEdge Streaming for distributed video and music delivery via streaming, and Limelight Custom CDN for custom distributed delivery solutions
Mirror Image www.mirror-image.com	Provides content delivery, streaming media, Web computing and reporting services	Surrogate servers located in 22 countries around the world.	N/A
Netli www.netli.com	Provides business quality Internet services over the NetliOne platform	Computer clusters in 13 cities around the world.	N/A
SyncCast www.synccast.com	Provides solutions for delivering digital content and related data via Internet	It covers motion picture companies, media production, music industry, and online retailers.	N/A

Table 2: Summary of existing academic content delivery networks

CDN Name	Service Type	Coverage	Products/Solutions (if any)
CoDeeN www.codeen.cs.princeton.edu	Provides caching of content and redirection of HTTP requests	CoDeeN is an academic testbed content distribution network built on top of PlanetLab	N/A
COMODIN http://lisdip.deis.unical.it/research/index.html	Provides collaborative media playback service	COMODIN is an academic streaming CDN deployed on an international testbed	N/A
Coral www.coralcdn.org	Provides content replication in proportion to the content's popularity	Coral is a free peer-to-peer CDN. During beta testing, the Coral node network is hosted on PlanetLab [111], a large scale distributed research network of 400 servers, instead of third party volunteer systems. Of those 400 servers, about 275 are currently running Coral	N/A
Globule www.globule.org	Provides replication of content, monitoring of servers and redirecting client requests to available replicas	Globule is an open source collaborative CDN	N/A

4.1. Commercial CDNs

Accellion – Accellion, Inc. a privately held company headquartered in Palo Alto, California, is the provider of on-demand secure file transfer appliance and data management solutions. Accellion customers are industries such as advertising/media production, manufacturing, healthcare, consumer goods, higher education etc. Accellion's products are built on the SeOS (SmartEdge Operating System) technology, which is a distributed file storage and transmission infrastructure for enterprise applications. The SeOS technology, enables Accellion to move, replicate, and manage large sized files. It unifies and manages multiple storage types, across geographically dispersed locations, using a range of transport and delivery protocols. Accellion Courier Secure File Transfer Appliances (SFTA) is an on-demand file transfer solution for securely exchanging files [73][74]. It sends large attachments outside of the e-mail infrastructure yet providing the convenience of e-mail to both sender and receiver. The standard workflow for transferring files using SFTA is: 1) a user sends large-sized files – including gigabyte-sized files – through a Web interface; 2) files are uploaded to the Accellion Courier Standard SFTA; 3) the receiver receives an e-mail with an embedded, secure HTTP link; 4) the recipient clicks on the link and downloads the files from Accellion Courier Standard SFTA. In an enterprise context, files are replicated between Accellion Courier Enterprise SFTAs and recipient downloads necessary files from the closest Accellion Courier Enterprise SFTA. SFTA is deployed in parallel with the existing e-mail infrastructure to take the file transfer load off the e-mail system. Accellion ensures the authorized access of data/files by the users through deployment of a LDAP (Lightweight Directory Access Protocol) infrastructure via LDAPS (LDAP over SSL/TLS) queries. Accellion SFTA can also be customized to access the authentication and address book lookup services via HTTPS (HTTP over SSL/TLS) SOAP calls to Web Services server [75]. Accellion also provides online desktop and server backup and recovery solutions through Accellion Backup and Recovery Solutions (BRS).

Akamai – Akamai technologies [2][3] evolved out of an MIT research effort aimed at solving the flash crowd problem. It is the market leader in providing content delivery services. It owns more than 20,000 servers over 1,000 networks in 71 countries [3]. Akamai's approach is based on the observation that serving Web content from a single location can present serious problems for site scalability, reliability and performance. Hence, a system is devised to serve requests from a variable number of surrogate servers at the network edge [4]. Akamai servers deliver static, dynamic content and streaming audio and video.

Akamai's infrastructure handles flash crowds by allocating more servers to sites experiencing high load, while serving all clients from nearby servers. The system directs client requests to the nearest available server likely to have the requested content. Akamai provides automatic network control through the mapping technique (i.e. the direction of request to content servers), which uses a dynamic, fault-tolerant DNS system. The mapping system resolves a hostname based on the service requested, user location and network status. It also uses DNS for network load-balancing. Akamai name servers resolve hostnames to IP addresses by mapping requests to a server. Akamai agents communicate with certain border routers as peers; the mapping system uses BGP [5]

information to determine network topology. The mapping system in Akamai combines the network topology information with live network statistics – such as traceroute data [6] – to provide a detailed, dynamic view of network structure and quality measures for different mappings.

Akamai's DNS-based load balancing system continuously monitors the state of services and their servers and networks. To monitor the entire system's health end-to-end, Akamai uses agents that simulate end-user behavior by downloading Web objects and measuring their failure rates and download times. Akamai uses this information to monitor overall system performance and to automatically detect and suspend problematic data centers or servers. Each of the content servers frequently reports its load to a monitoring application, which aggregates and publishes load reports to the local DNS server. That DNS server then determines which IP addresses (two or more) to return when resolving DNS names. If a certain server's load exceeds a certain threshold, the DNS server simultaneously assigns some of the server's allocated content to additional servers. If the server's load exceeds another threshold, the server's IP address is no longer available to clients. The server can thus shed a fraction of its load when it experiences moderate to high load. The monitoring system in Akamai also transmits data center load to the top-level DNS resolver to direct traffic away from overloaded data centers. In addition to load balancing, Akamai's monitoring system provides centralized reporting on content service for each customer and content server. This information is useful for network operational and diagnostic purposes.

AppStream – AppStream, Inc. is a private company funded by Draper Fisher Jurvetson, JK&B Capital, Goldman Sachs, Evergreen Partners, Sun Microsystems and Computer Associates. It is a provider of technology for on-demand software distribution and software license management tools for the extended enterprises. Its product is AppStream Software 5.0, which is a software distribution and license management platform. AppStream provides solutions in four key areas: self-service software distribution, software license management, remote software access, and virtual image distribution. AppStream allows users to launch an application from a browser or from a traditional desktop shortcut. With AppStream, software can be managed as service within an enterprise. Thus, the users in an enterprise are capable of streaming and caching both desktop and enterprise applications; since all application functionalities are preserved by AppStream, including interaction with peripherals and traditionally installed applications. AppStream software divides applications into segments (streamlets) required to start the application on a client desktop, delivering the streamlets to the users as needed based upon their usage behavior. Users get the flavor of using a fully installed product; while from the business perspective, AppStream provides the high functionality of a locally installed application allowing centralized access and scalability for any enterprise. The AppStream server communicates as a traditional Web application, using HTTP, with the Software Streaming Transfer Protocol (SSTP) running over HTTP for the most efficient delivery of application segments [44].

EdgeStream – EdgeStream, Inc., based in Southern California is a provider of video streaming applications over the public Internet. It provides video on-demand and IPTV streaming software to enable transportation of high bit rate video over Internet. EdgeStream developed Continuous Route Optimization (CROS) and Congestion Tunnel Through (ICTT) technologies that address the latency, packet loss, and congestion bottlenecks [76]. Embedded applications in Consumer Electronics Devices, wireless handheld devices, IP set top boxes, and advanced digital TV's can use the EdgeStream software for video streaming. EdgeStream server software consists of Content Management and Online Reporting (CMOR) Server Software Module, EdgeStream Control Server Software Module, EdgeStream Database System Module, and EdgeStream Streaming Server Module. CMOR module manages accounts, content, and all other servers in the system. It also generates Web-based real time reports for viewing statistics and transactions from a SQL database. The control module provides necessary authority to obtain the content location information along with streaming delivery management and logging functions. The database module maintains logs for accounting and billing purpose and the streaming server module is designed for load balancing and for running on standard low cost server platforms. The EdgeStream client software provides measurement of Internet connection quality on a second by second basis [77]. EdgeStream offers demonstration of its performance to the prospective customers through maintaining a streaming server network, and offers short term and long term video hosting services for a quick and cost effective roll out of video applications.

Globix – Globix Corporation is a provider of Internet infrastructure and network services. It offers a comprehensive suite of services from offering network bandwidth, to the management of Web applications, servers, databases, to security, media streaming, and collocation. Globix Internet infrastructure consists of both a trans-Atlantic/trans-continental IP backbone as well as an optical network throughout the Northeast and mid-Atlantic regions. Globix IP backbone connects the customers to the Internet via a high-capacity network, fully owned and operated by Globix. Globix provide four types of services: network Services, hosting Services, managed services, and media services. Under the managed services, Globix offers security, storage, messaging, disaster recovery, monitoring, application and database management services. Globix also provides media services to capture, store, host and distribute media content from live event production, encoding, presentation tools, and traffic analysis. Globix load balancing service distributes traffic among multiple servers, sending the request to the least loaded server in the dedicated server cluster. Unlike software-based load balancer, the

service is built with an ASIC-based hardware architecture that provides increased traffic performance. Globix also offers Globix Traffic Analyzer which is a monitoring and reporting service. It is used to measure and provide day-to-day summary reports on the physical network and server hardware, Web and application services, and backend database performance [78].

Limelight Networks – Limelight Networks is a CDN that provides distributed on-demand and live delivery of video, music, games and download. Limelight Networks has surrogate servers located in 72 locations around the world including New York, Los Angeles, San Jose, London, Amsterdam, Tokyo, and Hong Kong. Typical Limelight Networks' customers include companies who use Internet for product delivery, and deliver extreme volume of content to large audiences [79]. Limelight Networks has created a system for distributed digital media delivery to large audiences. Limelight Networks has a CDN platform that allows it to shape the CDN to meet any content provider's specific needs and environment. Limelight Networks has the following products: Limelight ContentEdge for distributed content delivery via HTTP, Limelight MediaEdge Streaming for distributed video and music delivery via streaming, and Limelight Custom CDN for custom distributed delivery solutions. Limelight MediaEdge Streaming is a distribution platform that provides high performance services for live and on-demand streaming of audio and video content over the Internet. Content providers using Limelight's streaming services use Limelight User Exchange (LUX) to track end-users' activity with real time reporting. Thus it can be used to access and improve the media streaming strategy.

Mirror Image – Mirror Image is a global network that is dedicated to provide online content, application and transaction delivery to the users around the world. Mirror Image has surrogate servers located at network peering points in 22 countries across America, Europe and Asia, where the concentration of Web traffic and users is the highest. It provides content delivery, streaming media, Web computing and reporting services [80]. Mirror Image exploits a global Content Access Point (CAP) infrastructure on top of the Internet to provide content providers, service providers and enterprises with a platform that delivers Web content to end-users. As a secure and managed high-speed layer on top of the Internet, each CAP offloads origin servers and networks by intelligently placing content at locations closer to users world-wide. Mirror Image also provides real-time monitoring and reporting through a Web-based CustomerCenter console.

Netli – Netli is a privately owned company based in Mountain View, California. It is a provider of business quality Internet. Netli has computer clusters in 13 cities around the world. Netli's services are – NetLightning® for optimizing delivery of Web applications and content; NetliOffload™ to deliver infrastructure that meets enterprise requirements; NetliView™ to provide near real-time information on the performance, availability, and usage pattern of business applications; and NetliContinuity™ to gain strategic control and management of data center resources. The NetliOne platform is a global Application Delivery Network (ADN) that ensures short response time, improved visibility and control of applications over the Internet. Netli ADN services are delivered over the NetliOne platform. NetliOne platform consists of a series of globally distributed Virtual Data Centers (VDCs) and Application Access Points (AAPs), a global DNS redirection and IP address mapping system, a high performance protocol and content optimization software, an online monitoring and reporting system, and a 24X7 network operations center [82]. The global set of Netli's VDC and AAP forms a direct bi-nodal network between origin servers and users at the edge of the Internet. Netli uses a proprietary protocol that accelerates application and content delivery bi-directionally. When a user accesses a Web application, the browser request is processed by Netli's dynamic mapping and redirection system which directs the request to a nearby Virtual Data Center (VDC). On receiving a page request, the VDC supplies the base page, simultaneously parses the base page, predicts requests for subsequent embedded objects and issues corresponding requests to the origin server. The VDC acts as the proxy for the origin server which acknowledges the browser session, and the request is forwarded to the Application Access Point (AAP) located next to the origin server. The AAP then issues the request to the origin server, obtains response and forwards it to the user through the VDC [81].

SyncCast – SyncCast is a CDN, which facilitates global load balancing, utilizing tier-1 Internet backbones. SyncCast offers solutions from application development, Web hosting and Internet connectivity to deployment and systems integration. It provides solutions for delivering digital content and related data via the Internet and other media [83]. SyncCast is designed to deliver content rapidly in a cost-effective manner. SyncCast load balances client traffic through the use of load-balancing equipments from major components such as F5, Cisco and Foundry. Thus, SyncCast allows the users to connect to the streaming servers in multiple data centers over the most efficient network path, and the users perceive better network performance. SyncCast offers P2P streaming technologies to the streaming clients. SyncCast P2P technology intelligently monitors the quality of audio/video stream to each user and it switches them to another stream source in occurrence of service quality degradation. SyncCast also provides collocation and storage services along with load balancing, firewall, database management and managed backup services.

4.2. Academic CDNs

CoDeeN – CoDeeN [84][85] is developed at Princeton University, USA. It is an HTTP-based CDN. A number of projects are related to Codeen – CoBlitz (a scalable Web-based distribution system for large files), CoDeploy (an efficient synchronization tool for PlanetLab slices), CoDNS (a fast and reliable name lookup service), CoTop (a command line activity monitoring tool for PlanetLab), CoMon (a Web-based slice monitor that monitors most PlanetLab nodes), and CoTest (a login debugging tool). CoDeeN provides caching of Web content and redirection of HTTP requests. It is an academic testbed CDN built on top of PlanetLab. This testbed consists of a network of high performance proxy servers. Each of these proxy servers acts both as request redirectors and surrogate servers. They provide fast and robust Web content delivery service through cooperation and collaboration. At the time of initial deployment, CoDeeN system was subject to the constraint of limited number of participating proxies. Recently, it is scaled to run on every available PlanetLab node. CoDeeN operates in the following way: 1) users set their internet caches to a nearby high bandwidth proxy that participates in the CoDeeN system; 2) requests to that proxy are then forwarded to an appropriate member of the system that has the file cached and that has sent recent updates showing that it is still alive. The file is forwarded to the proxy and then to the client. Thus even if the server response time is slow, as long as the content is cached on the system, serving requests to that file will be fast. It also means that the request will not be satisfied by the original server. CoDeeN can not handle dynamic content. Moreover, due to forwarding the files to the proxy and then to the client, CoDeeN is not transparent to the end-users. For rare files this system could be slightly slower than downloading the file itself. All accesses via CoDeeN are logged. These log files are monitored for performance measurement of CoDeeN system [85].

A significant application service running on top of CoDeeN is CoBlitz [166]. It is a file transfer service which distributes large files without requiring any modifications to standard Web servers and clients, since all the necessary support is located on CoDeeN itself. One of the motivations for building CoBlitz on top of CoDeeN is to ensure long duration caching so that client requested content can be served quickly even after demand for it drops. CoBlitz is publicly accessible which allows the clients to prepend the original URL with "http://coblitz.codeen.org:3125" and fetch it like any other URL. A customized DNS server maps the name coblitz.codeen.org to a nearby PlanetLab node. To deploy CoBlitz, the HTTP CDN, CoDeeN is made amenable to handling large files. This approach includes modifying large file handling to efficiently support them on CoDeeN and modifying its request-routing to enable more swarm-like behavior under heavy load. In CoBlitz, a large file is considered as a set of small files (chunks) that can be spread across the CDN. CoBlitz works if the chunks are fully cached, partially cached, or not at all cached, fetching any missing chunks from the origin as needed. Thus, while transferring large files over CoBlitz, no assumptions are made about the existence of the file on the peers.

COMODIN – COMODIN (COoperative Media On-Demand on the InterNet) [163][164] is an academic streaming CDN providing collaborative media streaming services on the current Internet infrastructure. It is currently deployed on an international distributed testbed, which is composed of three logical sub-networks, SATRD, LISDIP and ICAR-CS intranets, located at different geographical locations and connected through the Internet. The two plane-based architecture of COMODIN consists of an IP-multicast enabled streaming CDN, which represents the Base plane, and a set of distributed components providing collaborative playbacks, which represents the Collaborative plane. At the client side, the system centers on Java-based applications and applets which interface the cooperative groups of users. COMODIN uses a protocol to let a group of users to coordinate and control the playback of a common multimedia streaming session. In a shared session, if a control command (e.g. play/pause) is issued by a user, upon acceptance by the control server all users experience the same change on the shared session. The protocol is cooperative in the sense that clients try not to hinder each other. If a client detects a control request issued by another client, it will not issue further requests until it receives a reply from the server. Therefore, the focus of COMODIN is not on content delivery but on the control. Significant applications featured by the COMODIN system include Collaborative Learning on-Demand [165] and Distributed Virtual Theaters [164].

Coral – Coral [45] is a free, P2P content distribution network designed to mirror Web content. Coral is designed to use the bandwidth of volunteers to avoid slashdotting and to reduce the load on websites and other Web content providers in general. During beta testing, the Coral node network is hosted on PlanetLab [111] (a large scale distributed research network of 755 nodes at 363 sites), instead of third party volunteer systems. The source code of CoralCDN is freely available under the terms of the GNU GPL. To use CoralCDN, a content publisher has to append ".nyud.net:8090" to the hostname in a URL. Clients are redirected to the nearby Coral Web caches transparently through DNS redirection. Coral Web caches cooperate to transfer data from nearby peers whenever possible, minimizing both the load on the origin Web server and the latency perceived by the user. Coral allows nodes to nearby cached objects without redundantly querying more distant nodes. It also prevents the creation of hotspots even under degenerate loads. Coral uses an indexing abstraction called Distributed Sloppy Hash Table (DSHT), which is a variant of DHTs. Coral's architecture is based on clusters of well-connected machines. Clusters are exposed in the interface to higher-level software, and in fact form a

crucial part of the DNS redirection mechanism. Performance measurements of Coral demonstrate that it allows under-provisioned Web sites to achieve dramatically higher capacity, and its clustering provides quantitatively better performance than locality-unaware systems [45]. Thus Coral can handle large amounts of traffic with reasonable performance. But the main drawback of Coral is that it is very difficult to select replica placement. Coral also can not handle dynamically generated content. Most significantly, the use of DNS-based request-routing in Coral limits its scalability. Moreover, the DNS redirection mechanism in Coral does not consider the heterogeneity of proxies while performing request-routing. Since volunteer nodes are used in Coral, a suitable mechanism should be in place to ensure the consistency and integrity of cached data. Coral also can not prevent a significant amount of traffic to be directed to the origin server at the start of the flash crowds.

Globule – Globule [46] is an open-source collaborative CDN developed at the Vrije Universiteit in Amsterdam. Globule is available for public use under open source license. Globule aims to allow Web content providers to organize together and operate their own world-wide hosting platform. It provides replication of content, monitoring of servers and redirecting of client requests to available replicas. In Globule, a site is defined as a collection of documents that belong to one specific user (the site's owner) and a server is a process running on a machine connected to a network, which executes an instance of the Globule software. Each server is capable of hosting one or more sites and to deliver content to the clients. Globule takes inter-node latency as the proximity measure. This metric is used to optimally place replicas to the clients, and to redirect the clients to an appropriate replica server. Globule is implemented as a third-party module for the Apache HTTP Server that allows any given server to replicate its documents to other Globule servers. To replicate content, content providers only need to compile an extra module into their Apache server and edit a simple configuration file. Globule automatically replicates content and redirects clients to a nearby replica. This can improve the site's performance, maintain the site available to its clients even if some servers are down, and to a certain extent help to resist to flash crowds and the Slashdot effect. Though Globule addresses issues like client localization, replica placement and Web application replication, a suitable mechanism is yet to develop in order to enforce the fair resource usage among participants. Moreover, the reliability of Globule system and the way it reacts to flash crowd events should be improved further.

5. Discussion

In this section, we provide the categorization and mapping of our taxonomy to the existing CDNs that have been surveyed in Section 4. We also provide a critical evaluation of the existing systems while classifying them. Our discussion of the existing systems based on the taxonomy also identifies the future directions in the content-distribution space and examines the validity of the taxonomy.

5.1. Mapping of the taxonomy to existing CDNs

Table 3, Table 4, Table 5, and Table 6 represent the mapping of the taxonomy to the CDNs that have been surveyed in Section 4. This mapping is done based on the four factors/issues – CDN composition, content distribution and management, request-routing, and performance measurement that we have considered for the taxonomy.

Table 3 shows the annotation of the existing systems based on CDN composition taxonomy. As shown in the table, majority of the existing CDNs use overlay approach for CDN organization, while some of them use network approach or the both. In an overlay approach, the following relationships are common – client-to-surrogate-to-origin server and network element-to-caching proxy. Inter-proxy relationship is also common among the CDNs, which supports inter-cache interaction. While using network approach, CDN providers rely on the interaction of network elements for providing services through deploying request-routing logic to the network elements based on predefined policies. The preference for overlay approach over the network approach is because of the scope for new services integration and simplified management of underlying network infrastructure. Offering a new service in overlay approach is as simple as distributing new code to CDN servers [61]. The use of both the overlay and network approach is also common among the CDN providers like Akamai, Globix, and Mirror Image. When a CDN provider uses a combination of these two approaches for CDN formation, a network element can be used to redirect HTTP requests to a nearby application-specific surrogate server. CDN providers use origin and replica servers to perform content delivery. Most of the replica servers of the CDN providers are used as Web servers for serving Web content. Some of the providers like Akamai, EdgeStream, Limelight Networks, Mirror Image and SyncCast use their replica servers as media server for delivering streaming media and video hosting services. Replica servers can also be used for providing services like caching, large file transfer, reporting, and DNS services. From Table 3, it can also be seen that most of the CDNs are dedicated to providing particular content, since variation of services and content requires the CDNs to adopt application-specific characteristics, architectures and technologies. Most of them provide static content, while only some of them provide streaming media, broadcasting and other services.

The mapping of various CDNs to the taxonomy presented in Section 3.2, based on content distribution and management is shown in Table 4. From the table it is clear that a single-ISP approach is mostly common for determining the optimal number of surrogate servers in order to deploy them at the network edge. Only CDNs with extensive geographical coverage follow the multi-ISP approach to place numerous number of surrogate servers at many global ISP POPs. Commercial CDNs such as Akamai, Globix, and light Networks, and academic CDNs such as Coral and CoDeeN use multi-ISP approach. Though the single-ISP approach suffers from the distant placement of the surrogates with respect to the locality of the end-users, it is most commonly used approach. Such preference is caused due to the setup cost, administrative overhead and complexity associated with deploying and managing of the system in multi-ISP approach. An exception to this can be found for sites with high traffic volumes. Multi-ISP approach performs better in this context since single-ISP approach is suitable only for sites with low-to-medium traffic volumes [16]. Most of the CDNs support partial site content delivery, while both full and partial-site content delivery is also possible. CDNs prefer to support partial site content delivery because it reduces load on the origin server and on the site's content generation infrastructure. Moreover, due to the infrequent change of embedded content, partial-site approach performs better than the full-site content delivery approach. Only few CDNs – Akamai, Mirror Image and Coral to be specific, are found to support clustering of contents. The content distribution infrastructure of other CDNs does not reveal any information whether other CDNs use any scheme for content clustering. Akamai and Coral cluster content based on users' sessions. This approach is beneficial because it helps to determine both the groups of users with similar browsing patterns and the groups of pages having related content. The only CDN to use the URL-based content clustering is Mirror Image. But URL-based approach is not popular because it suffers from the complexity involved to deploy them. Content outsourcing of the CDNs mostly use non-cooperative pull-based approach because of the simplicity provided in this approach through the use of DNS redirection or URL rewriting. Cooperative push-based approach is still considered as a theoretical approach and none of the existing CDNs supports it. Cooperative pull based approach involves complex technologies (e.g. DHT) as compared to the non-cooperative approach and it is used by the CDNs following P2P architecture [8]. Moreover, it imposes a large communication overhead (in terms of number of messages exchanged) when the number of clients is large. It also does not offer high fidelity when the content changes rapidly or when the coherency requirements are stringent.

From Table 4 it is also evident that except for academic CDNs and CDNs with large geographic coverage, intra-cluster caching is mainly used for content caching. Cache update techniques used by the CDNs mainly include periodic and on-demand update. Only Coral uses invalidation for updating caches since it delivers static content which changes very infrequently. None of the CDNs use update propagation due to the network overhead experienced during the delivery of an updated version of content to all the caches with change in content. Of all the cache update mechanisms, periodic update has the greatest reach since the caches are updated in a regular fashion. Thus, it has the potential to be most effective in ensuring cache content consistency. Update propagation and invalidation are not generally applicable as steady-state control mechanisms, and they can cause control traffic to consume bandwidth and processor resources that could otherwise be used for serving content [89]. Content providers themselves may administer to deploy specific caching policies or heuristics for cache update. The use of caching policy distribution is simpler to administer but it has limited effects. On the other hand, cache heuristics are a good CDN feature for content providers who do not want to develop own caching policies. But heuristics will not deliver the same results as well-planned policy controls [89].

Table 5 maps the request-routing mechanisms of various CDNs to the request-routing taxonomy presented in Section 3.3. As can be observed in Table 5, DNS-based mechanisms are very popular for request-routing. The main reason of this popularity is its simplicity and the ubiquity of DNS as a directory service. DNS-based mechanisms mainly consist of using a specialized DNS server in the name resolution process. Among other request-routing mechanisms, HTTP redirection is also highly used in the CDNs because of the finer level of granularity on the cost of introducing an explicit binding between a client and a replica server. Flexibility and simplicity are other reasons of using HTTP redirection for request-routing in CDNs. Some CDNs like Globix, Mirror Image use GSLB for request-routing. It is advantageous since less effort is required to add GSLB capability to the network without adding any additional network devices. Among the academic CDNs, Coral exploits overlay routing techniques, where indexing abstraction for request-routing is done using DSHT. Thus, it makes use of P2P mechanism for request redirection. As we mentioned earlier, the request-routing system of a CDN is composed of a request-routing algorithm and a request-routing mechanism. The request-routing algorithms used by the CDNs are proprietary in nature. The technology details of most of them have not been revealed. Our analysis of the existing CDNs indicates that Akamai and Globule use adaptive request-routing algorithm for their request-routing system. Since reliable information on the request-routing algorithms used in existing CDNs was not found, we refrain from presenting the mapping of the taxonomy for request-routing algorithms to the existing CDNs.

Performance measurement of a CDN through some metric estimation measures its ability to serve the customers with the desired content and/or services. A CDN's performance should be evaluated in terms of

storage, cache hit ratio, bandwidth consumption, communication overhead, latency, surrogate server utilization, scalability, and reliability. The estimation of performance metrics gives an indication of system conditions and helps for efficient request-routing and load balancing in large systems. Although performance measure of a CDN provider is important for a content provider to select the most appropriate one, the proprietary nature of the providers makes it difficult to perform measurement experiments in this area. Table 6 shows the mapping of different measurement techniques used in existing CDNs to the performance measurement taxonomy presented in Section 3.4. From the table, we can see that performance measurement of a CDN is done through internal measurement technologies as well as from the customer perspective. It is evident that, most of the CDNs use internal measurement based on network probing, traffic monitoring or the like. External performance measurement of CDN providers is not common because CDNs are commercial enterprises, which are not run transparently, and there is commercial advantage to keep the performance metrics and methodologies internal. Despite this, some CDNs like Akamai allow external measurement to be performed by third-party organizations.

Table 3: CDN composition taxonomy mapping

CDN Name	CDN Type	CDN Organization	Servers	Relationships	Interaction protocols	Content/service types
Accellion	Commercial	Network approach	N/A	N/A	Network elements interaction	Large file transfer services
Akamai	Commercial	Network and overlay approach	Origin and replica servers	Client-to-surrogate-to-origin server, Network element-to-caching proxy, Inter-proxy	Network elements interaction, inter-cache interaction	Static content, streaming media, and services (network monitoring, geographic targeting)
AppStream	Commercial	Overlay approach	Streaming caching/ replica servers	Inter-proxy	Inter-cache interaction	Content delivery services for applications
EdgeStream	Commercial	Network approach	N/A	N/A	Network elements interaction	Video streaming, video hosting services
Globix	Commercial	Network and overlay approach	Origin and replica servers	Client-to-surrogate-to-origin server, Network element-to-caching proxy	Network elements interaction	Internet infrastructure and network services
Limelight Networks	Commercial	Overlay approach	Origin and replica servers	Client-to-surrogate-to-origin server, Network element-to-caching proxy	Network elements interaction	Static content, streaming media
Mirror Image	Commercial	Network and Overlay approach	Origin and replica servers	Client-to-surrogate-to-origin server, Network element-to-caching proxy	Network elements interaction	Static content, streaming media, Web computing and reporting services
Netli	Commercial	Network approach	N/A	N/A	Network elements interaction	Application delivery network services
SyncCast	Commercial	Network approach	N/A	N/A	Network elements interaction	Streaming media

CDN Name	CDN Type	CDN Organization	Servers	Relationships	Interaction protocols	Content/service types
CoDeeN	Academic	Overlay approach	Origin and replica/proxy cache servers	Client-to-surrogate-to-origin server, Network element-to-caching proxy, inter-proxy	Network elements interaction, inter-cache interaction	Static content
COMODIN	Academic	Network approach	Origin and replica servers	Client-to-surrogate-to-origin server	Network elements interaction	Streaming media
Coral	Academic	Overlay approach	Origin and replica/proxy cache servers	Client-to-surrogate-to-origin server, Network element-to-caching proxy, inter-proxy	Network elements interaction, inter-cache interaction	Static content
Globule	Academic	Overlay approach	Origin and replica servers	Client-to-surrogate-to-origin server, Network element-to-caching proxy, inter-proxy	Network elements interaction, inter-cache interaction	Static content and monitoring services

Table 4: Content distribution and management taxonomy mapping

CDN Name	Surrogate placement	Content Selection and Delivery	Content Outsourcing	Cache Organization
Accellion	Single-ISP approach	Content selection • Partial-site content delivery Content Clustering N/A	Non-cooperative pull-based	Caching technique • Inter-cluster caching Cache update N/A
Akamai	Multi-ISP approach	Content selection • Full and partial-site content delivery Content Clustering • Users' sessions based	Non-cooperative pull-based	Caching technique • Intra and inter-cluster caching Cache update • On-demand
AppStream	Single-ISP approach	Content selection • Partial-site content delivery Content Clustering N/A	Non-Cooperative pull-based	Caching technique • Intra-cluster caching Cache update • On-demand
EdgeStream	Single-ISP approach	Content selection • Partial-site content delivery Content Clustering N/A	Non-cooperative pull-based	Caching technique • Inter-cluster caching Cache update N/A
Globix	Multi-ISP approach	Content selection • Full and partial-site content delivery Content Clustering N/A	Non-cooperative pull-based	Caching technique • Intra-cluster caching Cache update • On-demand

CDN Name	Surrogate placement	Content Selection and Delivery	Content Outsourcing	Cache Organization
Limelight Networks	Multi-ISP	Content selection • Partial-site content delivery Content Clustering N/A	Non-cooperative pull-based	Caching technique • Intra-cluster caching Cache update • On-demand
Mirror Image	Multi-ISP approach	Content selection • Partial-site content delivery Content Clustering • URL based	Non-cooperative pull-based	Caching technique • Intra-cluster caching Cache update • On-demand
Netli	Single-ISP approach	Content selection • Partial-site content delivery Content Clustering N/A	Non-cooperative pull-based	Caching technique • Inter-cluster caching Cache update N/A
SyncCast	Single-ISP approach	Content selection • Full and partial-site content delivery Content Clustering N/A	Non-cooperative pull-based	Caching technique • Intra-cluster caching Cache update • On-demand
CoDeeN	Multi-ISP approach	Content selection • Partial-site content delivery Content Clustering N/A	Cooperative pull-based	Caching technique • Intra and inter-cluster caching Cache update N/A
COMODIN	Single-ISP	Content selection • Full-site content delivery Content Clustering N/A	Non-cooperative pull-based	Caching technique • Intra-cluster caching Cache update • Periodic update
Coral	Multi-ISP approach	Content selection • Full and partial-site content delivery Content Clustering • Users' sessions based	Cooperative pull-based	Caching technique • Intra and inter-cluster caching Cache update • Cache invalidation
Globule	Single-ISP approach	Content selection • Partial-site content delivery Content Clustering N/A	Cooperative pull-based	Caching technique • Intra and inter-cluster caching Cache update N/A

Table 5: Request-routing taxonomy mapping

CDN Name	Request-routing Technique
Accellion	HTTP redirection
Akamai	DNS-based request-routing
AppStream	HTTP redirection
EdgeStream	HTTP redirection
Globix	Global Server Load Balancing (GSLB) • Global awareness • Smart authoritative DNS
Limelight Networks	DNS-based request-routing

CDN Name	Request-routing Technique
Mirror Image	Global Server Load Balancing (GSLB) • Global awareness • Smart authoritative DNS
Netli	DNS-based request-routing
SyncCast	Global Server Load Balancing (GSLB) • Global awareness • Smart authoritative DNS
CoDeeN	HTTP redirection
COMODIN	DNS-based request-routing
Coral	DNS-based request-routing
Globule	HTTP redirection DNS-based request-routing

Table 6: Performance measurement taxonomy mapping

CDN Name	Performance Measurement
Accellion	N/A
Akamai	Internal measurement • Network probing • Traffic monitoring (proactive) External measurement • Performed by Giga Information group according to eight criteria
AppStream	Internal measurement • Network probing • Traffic and application monitoring
EdgeStream	Internal measurement • Traffic monitoring through Real Time Performance Monitoring Service (RPMS)
Globix	Internal measurement • Network probing • Traffic and application monitoring
Limelight Networks	N/A
Mirror Image	Internal measurement • Network probing • Traffic monitoring and reporting
Netli	Internal measurement • Traffic and application monitoring (continuous) though NetliView
SyncCast	Internal measurement • Network probing • Traffic and Streaming Media Performance Monitoring (proactive), and reporting

CDN Name	Performance Measurement
CoDeeN	Internal measurement • Traffic and system monitoring through CoTop
COMODIN	Internal measurement • Network probing • Feedback from surrogates
Coral	Internal measurement • Traffic monitoring
Globule	Internal measurement • Traffic monitoring

5.2. Future directions

The taxonomy and survey presented in this paper give a high level description of the present trends in the content networking domain. From the detailed analysis of the technologies and trends, we have found the following possibilities that are expected to drive innovation within this domain:

A unified content network – To make content services an Internet infrastructure service, vendors have implemented content service networks (CSN) [53], which act as another network infrastructure layer built upon CDNs and provide next generation of CDN services. CSN appears to be a variation of the conventional CDN. This logical separation between content and services under the 'Content Delivery/Distribution' and 'Content Services' domain, is undesirable considering the on-going trend in content networking. Hence, a unified content network, which supports the coordinated composition and delivery of content and services, is highly desirable.

Towards a research Content Network (CN) – As a part of normal system research cycle, CDN researchers want to deploy and test new services and mechanisms. In this context, to achieve the real-time performance result, user requests should be simulated to generate real content network traffic. It points to the need of an integrated research framework that allows researchers to quickly assemble CDN systems from existing components, and to experiment with low-level operating systems mechanisms [105]. The world-wide test bed, called PlanetLab [111] can be used to support part of CDN research. However, sometimes the experimental results achieved through this platform drift from reality because of the concurrent execution of several experiments. Hence, such a test bed should have the ability to provide access control in order to limit concurrent experiments to a reasonable level.

An adaptive CDN for media streaming – Hosting of on-demand media content streaming service is challenging because of the enormous network and bandwidth required to simultaneously deliver large amount of content to end-users. To avoid network congestion and to improve performance, peer-to-peer (P2P) techniques can be used to build an adaptive CDN. In such a system, content storage and workload from streaming server, network, and storage resources are offloaded to the end-users' workstations. The fundamental idea is to allow multiple subscriber peers to serve streams of the same video content simultaneously to a consuming peer rather than the traditional single-server-to-client streaming model, while allowing each peer to store only a small portion of the content. Such a solution for cost-effective media streaming using a P2P approach has been reported in the design of the Decentralized Media Streaming Infrastructure (DeMSI) [140]. Another work on open and adaptive streaming CDN through collaborative control on media streaming can be found in [143].

A mobile dynamic CDN – Mobile networks are becoming increasing popular for distributing information to a large number of highly dynamic users. In comparison to wired networks, mobile networks are distinguished by potentially much higher variability in demand due to user mobility. Content distribution techniques for mobile networks must take into account potentially very high spatial and temporal demand variations to dynamically reconfigure the system in order to minimize the total traffic over the network backbone. A model for mobile dynamic CDN should be designed to allow the access of accurate and up-to-date information and enterprise applications. Such a mobile dynamic CDN model for enterprise networks and related content management policies are presented in [155].

Dynamic content – Dynamic content includes HTML or XML pages that are generated on the fly based on user specification. The dynamic generation of Web pages can be performed with the use of scalable Web application hosting techniques such as edge computing [157], context-aware data caching [156][158], data replication [156] and content blind data caching [156]. Instead of replicating the dynamic pages generated by a Web server, these techniques aim to replicate the means of generating pages over multiple edge servers [156]. In order to manage dynamic content, a CDN provider may use such scalable techniques to accelerate the dynamic generation of Web pages. The choice of the appropriate strategy may vary depending on the characteristics of Web applications.

Web services – Now-a-days, few commercial CDNs host Web services. For instance, Akamai has deployed .NET services on its network. Mirror Image has also developed an Application Delivery Network (ADN) that hosts both .NET and J2EE applications at its surrogate servers. Several studies [159][160] have shown that the performances of Web services are relatively poor because of the requirements for processing and special hosting capability. Therefore, technologies to improve the performances of Web services are needed. To address this problem, several mechanisms/applications as described in [159][160] can be used to effectively replicate the Web services to the surrogate servers of a CDN.

Scalability and Quality of Service (QoS) – Within the structure of present day CDN business model, content providers pay the CDN providers to maximize the impact of their content. However, current trends reveal that the type of applications that will be supported by CDNs in future, will transform the current business model [105]. In future, the content providers as well as end-users will also pay to receive high quality content. In this context, scalability will be an issue to deliver high quality content, maintaining low operational costs.

Content distribution through peering – Most recently, the CDNs view content distribution services as a way to use shared networking resources to handle their peak capacity requirements, thus allowing reduced investments in their own infrastructure [89]. Thus, present trends in content networks and content networking capabilities give rise to the interest in interconnecting content networks. Several projects/works are being conducted for finding ways to peer the CDNs for better overall performance. In this context, the CDI (Content Distribution Internetworking) working group within IETF (Internet Engineering Task Force) has addressed many related issues including a model for CDI [26], architectural questions [107], distribution requirements [108], CDI scenarios [109], and CDI Authentication, Authorization and Accounting requirements [110]. Though many techniques for content internetworking and/or CDN peering has been proposed in literature [26][50][51][52], only Coral exhibits CDN peering though index abstraction of content using DSHT. Peered CDNs cooperate and collaborate to deliver content on each other's behalf, and thus reach to a large client population that one CDN cannot reach otherwise. Therefore, content networking domain will experience increased collaboration through the innovation of technologies for such peering arrangements. Such peering CDNs can be formed using a Virtual Organization (VO)-based model [49]. A VO will consist of existing CDN providers, which are self-interested and autonomous stakeholders. These entities would co-exist, cooperate and sometimes compete with one another in a virtual marketplace. One or more entities in such a virtual marketplace may sometimes realize the potential benefit of collaborating with other entities by exchanging resources. When such potential is recognized, relevant entities go through a process of forming a new VO to exploit it. Hence, the participants of VO cooperate and coordinate their activities in order to effectively mange the VO and to achieve the common goal in such a way that maximizes individual gain.

Load balancing and content replication in cooperative domain – The choice of strategy that will efficiently balance the load across surrogate servers of peered CDNs will be crucial to produce required QoS of end-users. This is a complex exercise where an integrated approach, combining proper design of networks, improved utilization of existing networks, detailed analysis of traffic flow congestion and adoption of appropriate load and resource distribution strategies, is required. Moreover, the issue of content replication and caching is critical to the success of peering arrangement of CDNs. In order to facilitate good performance for users across the globe, content must be replicated across surrogate servers of peered CDNs. Such replication mechanism will cache content on demand with respect to the locality of requests, focusing on regions where specific content is needed most. The concept of caching 'hot' content is not new, but in the context of a cooperative content delivery, there will be significant competing considerations. Modeling such replication mechanism in cooperative domain may lead to the deployment of market-based mechanisms.

Deployment of market mechanisms – Peered CDNs are dynamic in nature, where the availability of resource and request for a specific content may vary with time. The involvement of commercial CDN providers in such dynamic environment will lead to the deployment of market-based mechanisms, where demand-and-supply patterns decide the price and availability of resources. An economic model can exploit the dynamism of the market and makes the system more manageable through analyzing the emergent marketplace behavior. This also provides benefits to the involving CDNs to offer their resources and to open up many interesting new services. Deployment of the market mechanisms can be done based on a Service-Oriented Architecture (SOA). In addition, replication, resource sharing, and load balancing polices need to be guided by profit-driven utility functions that satisfy QoS requirements of the end-users. The use of such an economic model for content replication among peered CDNs can be found in [122].

Service-Oriented Architecture (SOA) – Future trends in content networking domain are expected to allow services to be composed of other services by building on standard protocols and invocation mechanisms. Thus, content networks exploit an SOA. High-level transparency within SOA is required, which could have impact on all the constituent technologies. Content management in such an SOA-based CDN is expected to be highly motivated by user preferences. Hence, a comprehensive model for managing the distributed contents and services in future CDN would be crucial to avail end-user's preferences. To address this issue, contents can be personalized to meet specific user's (or a group of users) preferences. Like Web personalization [54], user

preferences can be automatically learned from content request and usage data by using data mining techniques. Data mining over content network can exploit significant performance improvement through dealing with proper management of traffic, pricing and accounting/billing in SOA-based CDNs.

6. Summary and conclusion

In this paper, we have analyzed and categorized the infrastructural and technical attributes of Content Delivery Networks (CDNs). We have developed a comprehensive taxonomy for CDNs based on four issues: CDN composition, content distribution and management, request-routing, and performance measurement. We further build up taxonomies for each of these paradigms to classify the common trends in content networking and to provide a basis for comparison of existing CDNs. In doing so, the readers can gain an insight into the technology, services, strategies, and practices that are currently followed in this field. We have also provided a detailed survey of the existing CDNs and identified the future directions that are expected to drive innovation in this domain. Finally, we have performed mapping of the taxonomy to the existing systems. Such a mapping provides a basis to realize an in-depth understanding of the state-of-the-art technologies in content distribution space, and validates the applicability and accuracy of the taxonomy.

Recently, the CDN industry is getting consolidated as a result of acquisitions and/or mergers. During the preparation of this manuscript, we have experienced significant changes in the content distribution landscape due to this consolidation. For example, a recent news outburst [141] reveals that two commercial CDNs Akamai Technologies, Inc. and Netli, Inc. have signed a definitive agreement for Akamai to acquire Netli in a merger transaction. Along with the proliferation, formation, and consolidation of the CDN landscape, new forms of Internet content and services are coming into the picture. Consequently, content distribution, caching and replication techniques are gaining more attention in order to meet up the new technical and infrastructure requirements of the next generation CDNs. This may lead to new issues in the design and architecture of CDNs. Present trends in content networking domain indicate that better understanding and interpretation of the essential concepts in this area is necessary. Therefore, we hope that the comprehensive comparison framework based on our taxonomy and the state-of-the-art survey presented in this paper will not only serve as a tool to understand this complex area, but also will help to map the future research efforts in content networking.

Acknowledgements

We would like to acknowledge the efforts of all the developers of the commercial and academic CDNs surveyed in this paper. We thank our colleagues at the University of Melbourne – James Broberg, Marcos Assunção, Rajiv Ranjan, and Srikumar Venugopal for sharing thoughts and for making incisive comments and suggestions on this paper. We would like to express our gratitude to Athena Vakali (Aristotle University of Thessaloniki, Greece), George Pallis (Aristotle University of Thessaloniki, Greece), Carlo Mastroianni (ICAR-CNR, Italy), Giancarlo Fortino (Università della Calabria, Italy), Christian Vecchiola (University of Genova, Italy), and Vivek Pai (Princeton University, USA) for their visionary comments on various parts of the taxonomy. We are also thankful to Fahim Husain (Akamai Technologies, Inc., USA), Whitney Glockner (Netli, Inc., USA), D. Cirule (LocalMirror, Inc., USA), William Good (Mirror Image Internet, Inc., USA), and Lisa Amini (IBM T. J. Watson Research Center, USA) for providing valuable research papers, technical reports, white papers, and data sheet while preparing the manuscript.

References

[1] M. Arlitt, and T. Jin, "A Workload Characterization Study of 1998 World Cup Web Site," *IEEE Network*, pp. 30-37, May/June 2000.

[2] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally Distributed Content Delivery," *IEEE Internet Computing*, pp. 50-58, September/October 2002.

[3] Akamai Technologies, Inc., www.akamai.com, 2007

[4] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web," In *Proceedings of 29th Annual ACM Symposium on Theory of Computing*, ACM Press, NY, pp. 654-663, 1997.

[5] Y. Rekhter, and T. Li, "A Border Gateway Protocol 4," Internet Engineering Task Force RFC 1771, March 1995. www.ietf.org/rfc/rfc1771.txt

[6] G. Malkin, "Traceroute Using an IP Option," Internet Engineering Task Force RFC 1393, January 1993, www.ietf.org/rfc/rfc1393.txt

[7] F. Douglis, and M. F. Kaashoek, "Scalable Internet Services," *IEEE Internet Computing*, Vol. 5, No. 4, 2001, pp. 36-37.

[8] G. Pallis, and A. Vakali, "Insight and Perspectives for Content Delivery Networks," *Communications of the ACM*, Vol. 49, No. 1, ACM Press, NY, USA, pp. 101-106, January 2006..

[9] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "On the placement of Internet Instrumentation," In *Proceedings of IEEE INFOCOM*, Tel-Aviv, Israel, pp. 295-304, March 2000.

[10] Y. Bartal, "Probabilistic Approximation of Metric Space and its Algorithmic Applications," In *Proceedings of 37th Annual IEEE Symposium on Foundations of Computer Science*, October 1996.
[11] P. Krishnan, D. Raz, and Y. Shavitt, "The Cache Location Problem," *IEEE/ACM Transaction on Networking*, Vol. 8, No. 5, 2000.
[12] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the Placement of Web Server Replicas," In *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, USA, pp. 1587-1596, April 2001.
[13] S. Jamin, C. Jin, A. R. Kure, D. Raz, and Y. Shavitt, "Constrained Mirror Placement on the Internet," In *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, USA, April 2001.
[14] B. Li, M. J. Golin, G. F. Italiano, D. Xin, and K. Sohraby, "On the Optimal Placement of Web Proxies in the Internet," In *Proceedings of IEEE INFOCOM*, NY, USA, pp. 1282-1290, March 1999.
[15] Y. Chen, R. H. Katz, and J. D. Kubiatowicz, "Dynamic Replica Placement for Scalable Content Delivery," In *Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS 02)*, LNCS 2429, Springer-Verlag, pp. 306-318, 2002.
[16] A. Vakali, and G. Pallis, "Content Delivery Networks: Status and Trends," *IEEE Internet Computing*, IEEE Computer Society, pp. 68-74, November-December 2003.
[17] N. Fujita, Y. Ishikawa, A. Iwata, and R. Izmailov, "Coarse-grain Replica Management Strategies for Dynamic Replication of Web Contents," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Vol. 45, Issue 1, pp. 19-34, May 2004.
[18] Y. Chen, L. Qiu, W. Chen, L. Nguyen, and R. H. Katz, "Efficient and Adaptive Web Replication using Content Clustering," *IEEE Journal on Selected Areas in Communications*, Vol. 21, Issue 6, pp. 979-994, August 2003.
[19] G. Peng, "CDN: Content Distribution Network," Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, Stony Brook, NY, 2003. http://citeseer.ist.psu.edu/peng03cdn.html
[20] C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson, and D. Culler, "Using Smart Clients to Build Scalable Services," In *Proceedings of USENIX 1997 Annual Technical Conference*, Anaheim, California, USA, January 1997.
[21] M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm, "Enhancing the Web Infrastructure – from caching to replication," *IEEE Internet Computing*, pp. 18-27, March-April 1997.
[22] B. Krishnamurthy, C. Willis, and Y. Zhang, "On the Use and Performance of Content Distribution Network," In *Proceedings of 1st International Internet Measurement Workshop*, ACM Press, pp. 169-182, 2001.
[23] C. Partridge, T. Mendez, and W. Milliken, "Host Anycasting Service," Internet Engineering Task Force RFC 1546, November 1993. www.ietf.org/rfc/rfc1546.txt
[24] Z. Fei, S. Bhattacharjee, E. W. Zugura, and M. H. Ammar, "A Novel Server Selection Technique for Improving the Response Time of a Replicated Service," In *Proceedings of IEEE INFOCOM*, San Francisco, California, USA, pp. 783-791, March 1998.
[25] J. Kangasharju, J Roberts, and K. W. Ross, "Object Replication Strategies in Content Distribution Networks," *Computer Communications*, Vol. 25, No. 4, pp. 367-383, March 2002.
[26] M. Day, B. Cain, G. Tomlinson, and P. Rzewski, "A Model for Content Internetworking (CDI)," Internet Engineering Task Force RFC 3466, February 2003. www.ietf.org/rfc/rfc3466.txt
[27] M. Hofmann, and L. R. Beaumont, *Content Networking: Architecture, Protocols, and Practice*, Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 129-134, 2005.
[28] D. Wessels, and K. Claffy, "Internet Cache Protocol (ICP) version 2," Internet Engineering Task Force RFC 2186, September 1997. www.ietf.org/rfc/rfc2186.txt
[29] A. Rousskov, and D. Wessels, "Cache Digests," *Computer Networks and ISDN Systems*, Vol. 30, No. 22-3, pp. 2155-2168, November 1998.
[30] S. Gadde, M. Rabinovich, and J. Chase, "Reduce, Reuse, Recycle: An Approach to Building Large Internet Caches," In *Proceedings of 6th Workshop on Hot Topics in Operating Systems*, pp.93-98, April 1997.
[31] V. Valloppillil, and K. W. Ross, "Cache Array Routing Protocol v1.0," Internet Draft, February 1998.
[32] D. Karger, A. Sherman, A. Berkheimer, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, and Y. Yerushalmi, "Web Caching with Consistent Hashing," *Computer Networks*, Vol. 31, No. 11-16, pp. 1203-1213, 1999.
[33] J. Ni, and D. H. K. Tsang, "Large Scale Cooperative Caching and Application-level Multicast in Multimedia Content Delivery Networks," *IEEE Communications*, Vol. 43, Issue. 5, pp. 98-105, May 2005.
[34] J. Ni, D. H. K. Tsang, I. S. H. Yeung, and X. Hei, "Hierarchical Content Routing in Large-Scale Multimedia Content Delivery Network," In *Proceedings of IEEE International Conference on Communications, 2003 (ICC '03)*, Vol. 2, pp. 854-859, May 2003.
[35] M Cieslak, D Foster, G Tiwana, and R Wilson, "Web Cache Coordination Protocol Version 2," http://www.web-cache.com/Writings/Internet-Drafts/draft-wilson-wrec-wccp-v2-00.txt
[36] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones, "SOCKS Protocol Version 5," Internet Engineering Task Force RFC 1928, March 1996. www.ietf.org/rfc/rfc1928.txt
[37] P. Vixie, and D. Wessels, "Hyper Text Caching Protocol (HTCP/0.0)," Internet Engineering Task Force RFC 2756, January 2000. www.ietf.org/rfc/rfc2756.txt
[38] A. Barbir, R. Penno, R. Chen, H. Hofmann, and H. Orman, "An Architecture for Open Pluggable Edge Services (OPES)," Internet Engineering Task Force RFC 3835, August 2004. www.ietf.org/rfc/rfc3835.txt
[39] N. Bartolini, E. Casalicchio, and S. Tucci, "A Walk Through Content Delivery Networks," In *Proceedings of MASCOTS 2003*, LNCS Vol. 2965/2004, pp. 1-25, April 2004.
[40] Akamai Technologies Inc., "Akamai-The Business Internet - A Predictable Platform for Profitable E-Business," 2004.

[41] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," *Journal of Network and Computer Applications*, Vol. 23, pp. 187–200, 2001.
[42] S. Ceri, and G. Pelagatti, *Distributed Databases: Principles and Systems*, McGraw-Hill, NY, 1984.
[43] A. Oram, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly & Associates, Inc., Sebastopol, CA, 2001.
[44] AppStream, Inc., "AppStream Technology for Windows Edition," Technology Overview, www.appstream.com/downloads/AppStream-Technology-for-Windows-Edition.pdf
[45] M. J. Freedman, E. Freudenthal, and D. Mazières, "Democratizing Content Publication with Coral," In *Proceedings of 1st USENIX/ACM Symposium on Networked Systems Design and Implementation*, San Francisco, CA, March 2004.
[46] G. Pierre, and M. van Steen, "Globule: A Collaborative Content Delivery Network," *IEEE Communications*, Vol. 44, No. 8, August 2006.
[47] H. T. Kung, and C. H. Wu, "Content Networks: Taxonomy and New Approaches," *The Internet as a Large-Scale Complex System*, (Kihong Park and Walter Willinger eds.), Oxford University Press, 2002.
[48] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An Analysis of Internet Content Delivery Systems," *ACM SIGOPS Operating Systems Review*, Vol. 36, pp. 315-328, 2002.
[49] R. Buyya, A. M. K. Pathan, J. Broberg, and Z. Tari, "A Case for Peering of Content Delivery Networks," *IEEE Distributed Systems Online*, Vol. 7, No. 10, IEEE CS Press, Los Alamitos, CA, USA, October 2006.
[50] E. Turrini, "An Architecture for Content Distribution Internetworking," Technical Report UBLCS-2004-2, University of Bologna, Italy, March 2004.
[51] I. Chaudhri, "CiRouter," FastTide Whitepaper, July 2001.
[52] A. Biliris, C. Cranor, F. Douglis, M. Rabinovich, S. Sibal, O. Spatscheck, and W. Sturm, "CDN brokering," *Computer Communications*, Vol. 25, Issue 4, pp. 393-402, March 2002.
[53] W. Y. Ma, B. Shen, and J. T. Brassil, "Content Services Network: Architecture and Protocols," In *Proceedings of 6th International Workshop on Web Caching and Content Distribution (IWCW6)*, 2001.
[54] B. Mobasher, R. Cooley, and J. Srivastava, "Automatic Personalization Based on Web Usage Mining," *Communications of the ACM*, Vol. 43, No. 8, pp. 142-151, August 2000.
[55] S. Androutsellis-Theotokis, and D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies," *ACM Computing Surveys*, Vol. 36, No. 4, ACM Press, NY, USA, pp. 335-371, 2004.
[56] S. Adler, "The SlashDot Effect: An Analysis of Three Internet Publications," *Linux Gazette Issue*, Vol. 38, 1999.
[57] International Standards Organization (ISO), "Open Systems Interconnection--Basic Reference Model," ISO 7498, 1989.
[58] R. Moore, T. A. Prince, and M. Ellisman, "Data Intensive Computing and Digital Libraries," *Communications of the ACM*, Vol. 41, No. 11, ACM Press, NY, USA, pp. 56-62, 1998.
[59] S. Venugopal, R. Buyya, and K. Ramamohanarao, "A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing," *ACM Computing Surveys*, Vol. 38, No. 1, ACM Press, NY, USA, 2006.
[60] M. T. Ozsu, and P. Valduriez, *Principles of Distributed Database Systems*, Prentice-Hall, Inc., Upper Saddle River, NJ, 1999.
[61] I. Lazar, and W. Terrill, "Exploring Content Delivery Networking," *IT Professional*, Vol. 3, No. 4, pp. 47-49, 2001.
[62] S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. Van Steen, "Replication of Web Hosting Systems," *ACM Computing Surveys*, Vol. 36, No. 3, ACM Press, NY, USA, 2004.
[63] I. Cooper, I. Melve, and G. Tomlinson, "Internet Web Replication and Caching Taxonomy," Internet Engineering Task Force RFC 3040, January 2001. www.ietf.org/rfc/rfc3040.txt
[64] R. Brussee, H. Eertink, W. Huijsen, B. Hulsebosch, M. Rougoor, W. Teeuw, M. Wibbels, and H. Zandbelt, "Content Distribution Network State of the Art," Telematica Instituut, June 2001.
[65] M. Hamilton, A. Rousskov, and D. Wessels, "Cache Digest Specification – version 5," December 1998. http://www.squid-cache.org/CacheDigest/cache-digest-v5.txt
[66] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," Internet Engineering Task Force RFC 2068, January 1997. www.ietf.org/rfc/rfc2068.txt
[67] J. Postel, "User Datagram Protocol," Internet Engineering Task Force RFC 768, August 1980. www.ietf.org/rfc/rfc768.txt
[68] A. Barbir, O. Batuner, A. Beck, T. Chan, and H. Orman, "Policy, Authorization, and Enforcement Requirements of the Open Pluggable Edge Services (OPES)," Internet Engineering Task Force RFC 3838, August 2004. www.ietf.org/rfc/rfc3838.txt
[69] M. R. Garey, and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co., New York, NY, USA, 1979.
[70] P. Radoslavov, R. Govindan, and D. Estrin, "Topology-informed Internet Replica Placement," In *Proceedings of Sixth International Workshop on Web Caching and Content Distribution*, Boston, Massachusetts, June 2001.
[71] K. L. Johnson, J. F. Carr, M. S. Day, and M. F. Kaashoek, "The Measured Performance of Content Distribution Networks," *Computer Communications*, Vol. 24, No. 2, pp. 202-206, February 2001.
[72] S. Gadde, J. Chase, and M. Rabinovich, "Web Caching and Content Distribution: A View from the Interior," *Computer Communications*, Vol. 24, No. 2, pp. 222-231, 2001.
[73] Accellion, Inc., "Accellion Courier File Transfer Appliance Products Datasheet," www.accellion.com
[74] Accellion, Inc., "Large File Attachments in E-mail Servers – Five Coping Strategies," White paper, www.accellion.com
[75] Accellion, Inc., "Designing a Global Multi-Office File Transfer Architecture Using Secure File Transfer Appliance," White paper, www.accellion.com

[76] R. Chung, "Network Latency and Its Effect on Video Streaming," EdgeStream White paper, August 2004. www.edgestream.com
[77] EdgeStream, Inc., "EdgeStream Streaming Software Platform Datasheet," 2006. www.edgestream.com
[78] Globix Corporation, "Hosting and Distribution Service," Globix Datasheet, 2006. www.globix.com
[79] M. Gordon, "The Internet Streaming Media Boom: A Powerful Trend that Represents Fundamental Change," Limelight Networks White paper, 2007. www.limelightnetworks.com
[80] Mirror Image Internet, Inc., www.mirror-image.com, 2007.
[81] Netli, Inc., NetliOne Platform, White Paper, 2007. www.netli.com
[82] Netli, Inc., NetliOne Platform, Netli Datasheet, 2007. www.netli.com
[83] SyncCast, www.synccast.com, 2007.
[84] V. S. Pai, L. Wang, K. S. Park, R. Pang, and L. Peterson, "The Dark Side of the Web: An Open Proxy's View," In *Proceedings of the Second Workshop on Hot Topics in Networking (HotNets-II)*, Cambridge, MA, USA, November 2003.
[85] L. Wang, K. S. Park, R. Pang, V. S. Pai, and L. Peterson, "Reliability and Security in CoDeeN Content Distribution Network," In *Proceedings of the USENIX 2004 Annual Technical Conference*, Boston, MA, USA, June 2004.
[86] B. D. Davison, "A Web Caching Primer," *IEEE Internet Computing*, Vol. 5, No. 4, pp. 38-45, July 2001.
[87] J. Wang, "A Survey of Web Caching Schemes for the Internet," *SIGCOMM Computer Communication Review*, Vol. 29, No. 5, ACM Press, NY, USA, pp. 36-46, October 1999.
[88] V. Cardellini, E. Casalicchio, M. Colajanni, and P. S. Yu, "The State of the Art in Locally Distributed Web-Server Systems," *ACM Computing Surveys*, Vol. 34, No. 2, ACM Press, NY, USA, pp. 263-311, June 2002.
[89] P. Gayek, R. Nesbitt, H. Pearthree, A. Shaikh, and B. Snitzer, "A Web Content Serving Utility," *IBM Systems Journal*, Vol. 43, No. 1, pp. 43-63, 2004.
[90] A. Barbir, B. Cain, R. Nair, and O. Spatscheck, "Known Content Network Request-Routing Mechanisms," Internet Engineering Task Force RFC 3568, July 2003. www.ietf.org/rfc/rfc3568.txt
[91] A. Shaikh, R. Tewari, and M. Agrawal, "On the Effectiveness of DNS-Based Server Selection," In *Proceedings of IEEE INFOCOM*, Anchorage, AK, USA, pp. 1801-1810, April 2001.
[92] Z. M. Mao, C. D. Cranor, F. Boughs, M. Rabinovich, O. Spatscheck, and J. Wang, "A Precise and Efficient Evaluation of the Proximity between Web Clients and their Local DNS Servers," In *Proceedings of the USENIX 2002 Annual Technical Conference*, Monterey, CA, USA, pp. 229-242, June 2002.
[93] K. Aberer, and M. Hauswirth, "An Overview on Peer-to-Peer Information Systems," Swiss Federal Institute of Technology (EPFL), Switzerland. lsirpeople.epfl.ch/hauswirth/papers/WDAS2002.pdf
[94] Clip2 Distributed Search Solutions, "The Gnutella Protocol Specification v0.4," www.content-networking.com/papers/gnutella-protocol-04.pdf
[95] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-Peer Computing," Technical Report, HP Laboratories, Palo Alto, CA, HPL-2002-57, March 8, 2002. www.hpl.hp.com/techreports/2002/HPL-2002-57.pdf
[96] J. Lee, "An End-User Perspective on File-Sharing Systems," *Communications of the ACM*, Vo. 46, No. 2, ACM Press, NY, USA, pp. 49-53, February 2003.
[97] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites," In *Proceedings of the International World Wide Web Conference*, pp. 252-262, May 2002.
[98] AccuStream iMedia Research, "CDN Market Share: A Complete Business Analysis 2004 and 2005," Market Research Report, 2005. www.researchandmarkets.com
[99] Wikipedia, "September 11, 2001 Attacks," http://en.wikipedia.org/wiki/September_11,_2001_attack
[100] AccuStream iMedia Research, "CDN Market Dynamics, Analysis and Streaming Share: 2005-2006," Market Research Report, March 2006. www.researchandmarkets.com
[101] Internet Research Group, http://www.irg-intl.com/, 2007.
[102] Broadband Service Forum, http://broadbandservicesforum.org, 2007.
[103] ICAP Forum, http://www.i-cap.org/, 2007.
[104] Internet Streaming Media Alliance, http://www.isma.tv/, 2007.
[105] T. Plagemann, V. Goebel, A. Mauthe, L. Mathy, T. Turletti, and G. Urvoy-Keller, "From Content Distribution to Content Networks – Issues and Challenges," *Computer Communications*, Vol. 29, No. 5, pp. 551-562, 2006.
[106] D. C. Verma, *Content Distribution Networks: An Engineering Approach*, John Wiley & Sons, Inc.,, New York, 2002.
[107] M. Green, B. Cain, G. Tomlinson, S. Thomas, and P. Rzewski, "Content Internetworking Architectural Overview," Internet draft <draft-ietf-cdi-architecture-00.txt>, February 2002.
[108] L. Amini, S. Thomas, and O. Spatscheck, "Distribution Requirements for Content Internetworking," <draft-ietf-cdi-distribution-reqs-01>, December 2002.
[109] M. Day, D. Gelletti, and P. Rzewski, "Content Internetworking (CDI) Scenarios," Internet Engineering Task Force RFC 3570, July 2003. www.ietf.org/rfc/rfc3570.txt
[110] D. Gelletti, R. Nair, J. Scharber, and J. Guha, "Content Internetworking (CDI) AAA requirements," <draft-ietf-cdi-aaa-reqs-01>, June 2002.
[111] PlanetLab Consortium, "An Open Platform for Developing, Deploying, and Accessing Planetary-Scale Services," http://www.planet-lab.org/
[112] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. Al Hamra, and L. Garces-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime," In *Proceedings of 5th Annual Passive and Active Measurement Workshop (PAM'2004)*, Antibes Juan-Les-Pins, France, 2004.

[113] L. Wang, V. S. Pai, and L. Peterson, "The Effectiveness of Request Redirection on CDN Robustness," In *Proceedings of 5th Symposium on Operating Systems Design and Implementation*, Boston, MA, USA, pp. 345-360, December 2002.
[114] M. Szymaniak, G. Pierre, and M. van Steen, "Netairt: A DNS-based Redirection System for Apache," In *Proceedings of International Conference WWW/Internet*, Algrave, Portugal, 2003.
[115] V.S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, E. Nahum, "Locality-Aware Request Distribution in Cluster-Based Network Servers," *ACM SIGPLAN Notices*, Vol. 33, No. 11, ACM Press, NY, USA, pp. 205-216, 1998.
[116] A. Aggarwal, and M. Rabinovich, "Performance of Dynamic Replication Schemes for an Internet Hosting Service," Technical Report, HA6177000-981030-01-TM, AT&T Research Labs, Florham Park, NJ, USA, October 1998.
[117] K. Delgadillo, "Cisco DistributedDirector," Cisco White Paper, Cisco Systems, Inc., June 1997.
[118] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Looking Up Data in P2P Systems," Communications of the ACM, Vol. 46, No. 2, ACM Press, NY, USA, pp. 43-48, 2003.
[119] B. Huffaker, M. Fomenkov, D. J. Plummer, D. Moore and K. Claffy, "Distance Metrics in the Internet," In *Proceedings of IEEE International Telecommunications Symposium*, IEEE CS Press, Los Alamitos, CA, USA, 2002.
[120] M. Andrews, B. Shepherd, A. Srinivasan, P. Winkler, and F. Zane, "Clustering and Server Selection Using Passive Monitoring," In *Proceedings of IEEE INFOCOM*, NY, USA, 2002.
[121] O. Ardaiz, F. Freitag, and L. Navarro, "Improving the Service Time of Web Clients Using Server Redirection," *ACM SIGMETRICS Performance Evaluation Review*, Vol. 29, No. 2, ACM Press, NY, USA, pp. 39-44, 2001.
[122] A. M. K. Pathan, and R. Buyya, "Economy-based Content Replication for Peering Content Delivery Networks," IEEE TCSC Doctoral Symposium, In *Proceedings of IEEE 7th International Conference on Cluster Computing and the Grid (CCGrid 2007)*, Rio de Janeiro, Brazil, May 14-17, 2007.
[123] A. Emtage, and P. Deutsch, "Archie: An Electronic Directory Service for the Internet," In *Proceedings of the Winter Usenix Conference*, San Francisco, CA, USA, pp. 93-110, January 1992.
[124] B. Kahle, and A. Medlar, "An Information System for Corporate Users: Wide Area Information Servers," ConneXions—The Interoperability Report, 5(11), November 1991.
[125] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Transactions on Networking (TON)*, Vol. 11, No. 1, ACM Press, NY, USA, pp. 17-32, 2003.
[126] M. Harren, J. M. Hellerstein, R. Huebsch, B. T. Loo, S. Shenker, and I. Stoica, "Complex Queries in DHT-based Peer-to-Peer Networks," In *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002.
[127] J. Byers, J. Considine, and M. Mitzenmacher, "Simple Load Balancing for Distributed Hash Tables," In *Proceedings of 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, pp. 31-35, 2003.
[128] Keynote Systems—Web and Mobile Service Performance Testing Corporation, http://www.keynote.com/, 2007.
[129] The Gigaweb Corporation, http://www.gigaweb.com/, 2007.
[130] M. J. Freedman, K. Lakshminarayanan, and D. Mazières, "OASIS: Anycast for Any Service," In *Proceedings of 3rd Symposium of Networked Systems Design and Implementation (NSDI'06)*, Boston, MA, USA, 2006.
[131] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "IDMaps: A Global Internet Host Distance Estimation Service," *IEEE/ACM Transactions on Networking (TON)*, Vol. 9, No. 5, ACM Press, NY, USA, pp. 525-540, 2001.
[132] AccuStream iMedia Research, "Subscription Streaming and Download Media: Revenue and Market Share, 2003-2006," Market Research Report, October 2005. www.researchandmarkets.com
[133] AccuStream iMedia Research, "CDN Growth and Market Share Shifts: 2002-2006," Market Research Report, November 2006. www.researchandmarkets.com
[134] Technology Marketing Corporation (TMCnet) News, "AccuStream Research: CDN Market Share Shifts Highlight Four-Year $1.65 Billion Sector Growth; Flash at 21.9% Stream Share in '06," November 15, 2006. http://www.tmcnet.com/usubmit/2006/11/15/2082485.htm
[135] WinterGreen Research, Inc., "Streaming Media: Market Opportunities, Strategies, and Forecasts, 2004 to 2009," Market Research Report, February 2004. www.wintergreenresearch.com/reports/Streaming_Media_Brochure.pdf
[136] P. Lebrun, "The Large Hadron Collider, A Megascience Project," In *Proceedings of the 38th INFN Eloisatron Project Workshop on Superconducting Materials for High Energy Colliders*, Erice, Italy, 1999.
[137] A. Szalay, and J. Gray, "The World-Wide Telescope," *Science 293*, 5537, pp. 2037-2040, 2001.
[138] BioGrid Project, http://www.biogrid.jp, Japan, 2005.
[139] S. Goel, and R. Buyya, "Data Replication Strategies in Wide Area Distributed Systems," *Enterprise Service Computing: From Concept to Deployment*, Robin G. Qiu (ed), ISBN 1-599044181-2, Idea Group Inc., Hershey, PA, USA, pp. 211-241, 2006.
[140] A. Yim, and R. Buyya, "Decentralized Media Streaming Infrastructure (DeMSI): An Adaptive and High-Performance Peer-to-Peer Content Delivery Network," *Journal of Systems Architecture*, Vol. 52, Issue 12, ISSN 1383-7621, Elsevier Science, The Netherlands, pp. 737-772, December 2006.
[141] Netli, Inc., "Akamai to Acquire Netli," Press Release, February 5, 2007. http://www.netli.com/.
[142] B. Molina, C. E. Palau, and M. Esteve, "Modeling Content Delivery Networks and their Performance," *Computer Communications*, Vol. 27, Issue 15, pp. 1401-1411, September 2004.
[143] M. Esteve, G. Fortino, C. Mastroianni, C. E. Palau, and W. Russo, "Collaborative Control of Media Streaming based on a Content Distribution Network," Technical Report N. RT-ICAR-CS-06-03, Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche (ICAR-CNR), Rende (CS), Italy, April 2006.
[144] B. Wu and A. D. Kshemkalyani, "Objective-Optimal Algorithms for Long-Term Web Prefetching," *IEEE Transactions on Computers*, Vol. 55, No. 1, pp.2-17, 2006.

[145] S. S. H. Tse, "Approximate Algorithms for Document Placement in Distributed Web Servers," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 6, pp. 489-496, June 2005.
[146] G. Pallis, A. Vakali, K. Stamos, A. Sidiropoulos, D. Katsaros, and Y. Manolopoulos, "A Latency-Based Object Placement Approach in Content Distribution Networks," In *Proceedings of the 3rd latin American Web Congress (La-Web 2005)*, IEEE Press, Buenos Aires, Argentina, pp. 140-147, October 2005.
[147] G. Pallis, K. Stamos, A. Vakali, A. Sidiropoulos, D. Katsaros, and Y. Manolopoulos, "Replication-Based on Objects Load Under a Content Distribution Network," In *Proceedings of the 2nd International Workshop on Challenges in Web Information Retrieval and Integration (WIRI)*, Altanta, Georgia, USA, April 2006.
[148] K. Stamos, G. Pallis, and A. Vakali, "Integrating Caching Techniques on a Content Distribution Network," In *Proceedings of 10th East-European Conference on Advances in Databases and Information Systems (ADBIS 2006)*, Springer-Verlag, Thessaloniki, Greece, September 2006.
[149] S. Bakiras and T. Loukopoulos, "Combining Replica Placement and Caching Techniques in Content Distribution Networks," *Computer Communications*, Vol. 28, No. 9, pp. 1062-1073, June 2005.
[150] K. Stamos, G. Pallis, C. Thomos, and A. Vakali, "A Similarity-Based Approach for Integrated Web Caching and Content Replication in CDNs," In *Proceedings of 10th International Databased Engineering and Applications Symposium (IDEAS 2006)*, IEEE Press, New Delhi, India, December 2006.
[151] C. M. Chen, Y. Ling, M. Pang, W. Chen, S. Cai, Y. Suwa, O. Altintas, "Scalable Request-Routing with Next-Neighbor Laod Sharing in Multi-Server Environments," In *Proceedings of the 19th International Conference on Advanced Information Networking and Applications*, IEEE Computer Society, Washington, DC, USA, pp. 441-446, 2005.
[152] CDNSim, "A Content Distribution Network Simulator," 2007. http://oswinds.csd.auth.gr/~cdnsim/
[153] Network simulators, http://www-nrg.ee.lbl.gov/kfall/netsims.html, 2007.
[154] The network simulator – ns-2, http://www.isi.edu/nsnam/ns/, 2007.
[155] W. M. Aioffi, G. R. Mateus, J. M. de Almeida, and A. A. F. Loureiro, "Dynamic Content Distribution for Mobile Enterprise Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 23, Issue 10, pp. 2022-2031, October 2005.
[156] S. Sivasubramanian, G. Pierre, M. Van Steen, and G. Alonso, "Analysis of Caching and Replication Strategies for Web Applications," *IEEE Internet Computing*, Vol. 11, No. 1, pp. 60-66, 2007.
[157] M. Rabinovich, Z. Xiao, F. Douglis, and C. Kalmanek, "Moving Edge Side Includes to the Real Edge - the Clients," In *Proceedings of USENIX Symposium on Internet Technologies and Systems*, Sealttle, Washington, USA, March 2003.
[158] T. Buchholz, I. Hochstatter, and C. Linnhoff-Popien, "A Profit Maximizing Distribution Strategy for Context-Aware Services," In *Proceedings of 2nd International Workshop on Mobile Commerce and Services (WMCS'05)*, pp. 144-153, 2005.
[159] X. Geng, R. D. Gopal, R. Ramesh, A. B. Whinston, "Scaling Web Services with Capacity Provision Networks," *IEEE Computer*, Vol. 36, No. 11, pp. 64-72, 2003.
[160] T. Takase, M. Tatsubori, "Efficient Web Services Response Caching by Selecting Optimal Data Representation," In *Proceedings of 24th International Conference on Distributed Computing Systems (ICDCS 2004)*, pp. 188-197, 2004.
[161] K. Hosanagar, R. Krishnan, M. D. Smith, and J. Chuang, "Optimal Pricing of Content Delivery Network (CDN) Services," In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, pp. 205-214, 2004.
[162] K. Hosanagar, J. Chuang, R. Krishnan, and M. D. Smith, "Service Adoption and Pricing of Content Delivery Network (CDN) Services," Working Paper, July 2005. http://ssrn.com/abstract=590350
[163] M. Esteve, G. Fortino, C. Mastroianni, C. E. Palau, and W. Russo, "CDN-supported Collaborative Media Streaming Control," *IEEE Multimedia*, 2007.
[164] G. Fortino, C.E. Palau, W. Russo, and M. Esteve, "The COMODIN System: A CDN-based Platform for Cooperative Media On-Demand on the InterNet," In *Proceedings of the 10th International Conference on Distributed Multimedia Systems (DMS'04)*, San Francisco Bay (CA), USA, September 2004.
[165] G. Fortino and L. Nigro, "Collaborative Learning on-Demand on the Internet MBone," *Usability Evaluation of Online Learning Programs* (C. Ghaoui eds.), Idea Publishing Group, USA, 2003.
[166] K. S. Park and V. S. Pai, "Scale and Performance in the CoBlitz Large-File Distribution Service," In *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI 2006)*, San Jose, CA, USA, May 2006.