A Walk through Content Delivery Networks

Novella Bartolini[1⋆], Emiliano Casalicchio[2], and Salvatore Tucci[2]

[1] Università di Roma "La Sapienza", Via Salaria 113 - 00198 Roma, Italy,
novella@dsi.uniroma1.it
[2] Università di Roma "Tor Vergata", Via del Politecnico, 1 - 00133 Roma, Italy,
casalicchio@ing.uniroma2.it, tucci@uniroma2.it

Abstract. Content Delivery Networks (CDN) aim at overcoming the inherent limitations of the Internet. The main concept at the basis of this technology is the delivery at edge points of the network, in proximity to the request areas, to improve the user's perceived performance while limiting the costs. This paper focuses on the main research areas in the field of CDN, pointing out the motivations, and analyzing the existing strategies for replica placement and management, server measurement, best fit replica selection and request redirection.

1 Introduction

The commercial success of the Internet and e-services, together with the exploding use of complex media content online has paved the way for the birth and growing interest in Content Delivery Networks (CDN). Internet traffic often encounters performance difficulties characteristic of a non dedicated, best effort environment. The user's urgent request for guarantees on quality of service have brought about the need to study and develop new network architectures and technologies to improve the user's perceived performance while limiting the costs paid by providers. Many solutions have been proposed to alleviate the bottleneck problems and the most promising are based on the awareness of the content that has to be delivered. The traditional *"content-blind"* network infrastructures are not sufficient to ensure quality of service to all users in a dynamic and ever increasing traffic situation. New protocols and integrated solutions must be in place both on the network and on the server side to distribute, locate and download contents through the Internet.

The enhancement of computer networks by means of a content aware overlay creates the new architectural paradigm of the CDN. Today's CDN act upon the traditional network protocol stack at various levels, relying on dynamic and proactive content caching and on automatic application deployment and migration at the edge of the network, in proximity to the final users. Content replicas in a CDN are geographically distributed, to enable fast and reliable delivery to any end-user location: through CDN services, up-to-date content, can be retrieved by end-users locally rather than remotely.

⋆ The work of Novella Bartolini has been funded by the WEB-MINDS project supported by the Italian MIUR under the FIRB program

M.C. Calzarossa and E. Gelenbe (Eds.): MASCOTS 2003, LNCS 2965, pp. 1–25, 2004.

CDNs were born to distribute heavily requested contents from popular web servers, most of all image files. Nowadays, a CDN supports the delivery of any type of dynamic content, including various forms of interactive media streaming. CDN providers are companies devoted to hosting in their servers the content of third-party content providers, to mirroring or replicating such contents on several servers spread over the world, and to transparently redirecting the customers requests to the 'best replica' (e.g. the closest replica, or the one from which the customer would access content at the lowest latency). Designing a complete solution for CDN therefore requires addressing a number of technical issues: which kind of content should be hosted (if any) at a given CDN server (replica placement), how the content must be kept updated, which is the 'best replica' for a given customer, which mechanisms must be in place to transparently redirect the user to such replica. A proper placement of replica servers shortens the path from servers to clients thus lowering the risk of encountering bottlenecks in the non-dedicated environment of the Internet. A request redirection mechanism is provided at the access routers level to ensure that the best suited replica is selected to answer any given request of possibly different types of services with different quality of service agreements. The CDN architecture also relies on a measurement activity that is performed by cooperative access routers to evaluate the traffic conditions and the computational capacity and availability of each replica capable of serving the given request. Successfully implemented, a CDN can accelerate end user access to content, reduce network traffic, and reduce content provider hardware requirements.

This paper explores architectures, technologies and research issues in content delivery networks [53]. In section 2 we describe the core features of a CDN, discussing the motivations and how content delivery can alleviate internet performance problems. In section 3 we examine types of content and services that can beneficiate from content delivery techniques. Section 4 describes the architecture and working principles of a CDN. A detailed discussion on replica placement and management is provided in section 5. Section 6 introduces the problem of how measures can be taken to select the replica that can better fulfil an incoming request, while request redirection mechanisms are described and compared in section 7. Section 8 concludes the paper.

Let us point out that this paper is related to other papers contained in this volume. Issues related to QoS are discussed in several papers in this volume, including [8] [31] [42], while content delivery is related to peer-to-peer networking which is discussed in [39]. Content is often of multimedia nature, such as augmented reality [30] which will have high bandwidth and significant QoS needs, and the type of tools described in [32] can contribute simpler evaluation tools which are applicable to the systems we discuss.

2 Motivations for Content Delivery

Internet users commonly get frustrated by low performances and may decide to abandon a web site or to disconnect a multimedia session when experiencing

performance difficulties, causing revenue to be lost. Though centralized models are still in place in the Internet today, these architectures are poor in terms of adaptivity and scalability.

If a provider establishes a content server in a single physical location from which it disseminates data, services, and information to all its users, the single server is likely to become overloaded and its links can easily be saturated. The speed at which users can access the site could become unpredictably higher than the maximum request rate the server and its links can tolerate. Since it is impossible, with this approach, to adapt to the exponential growth of the Internet traffic, the centralized model of content serving is inherently unscalable, incapable of adaptivity and produces performance losses when traffic bursts occur. Though this leads to the conclusion that a certain amount of servers must be adopted, a cluster of servers (also known as *server farm*, that is a multi-server localized architecture, is not necessarily a solution yet.

The server computational and link capacity is only the first source of performance difficulties that may be encountered while downloading content over the Internet. There are many other possible congestion causes that may lead to unacceptable user perceived quality. The non dedicated, best effort nature of the Internet is the inborn limit to the possibility of having any sort of performance guarantee while delivering content over it.

The Internet is a network of heterogeneous networks composed of thousands of different autonomous systems ranging from large backbone providers to small local ISPs. The autonomous systems connect to each other creating the global Internet. The communication between two networks is achieved through the connection of border routers in a peering session. Two peer routers periodically exchange routing information and forward the received packets to carry each packet to its correct destination. This structure of the Internet as an interconnection of individual networks is the key to its scalability but is not sufficient to guarantee that a quickly growing number of users, services and traffic do not create bottlenecks that, if left unaddressed, can slow down performance. Bottlenecks may occur at many points in the core Internet and most of all in correspondence to peering points and backbones.

The network capacity is determined by the capacity of its cables and routers and although cable capacity is not an issue, the strongest limit to the backbone capacity comes from the packet-forwarding hardware and software of the routers. Once a peering point has been installed, traffic may have grown beyond expectations, resulting in a saturated link, typically because a network provider purchases just enough capacity to handle current traffic levels, to maximize the link utilization. The practice of running links at full capacity is one of the major causes of traffic bottlenecks showing very high utilization but also high rates of packet loss and high latency. Further the capacity of long backbones cannot always be adapted to the sudden and fast increases of the Internet traffic.

2.1 Move the Content to the Edges: An Approach to Improve the Internet Performance

The current centralized or partially distributed model of Internet content distribution requires that all user requests and responses travel several subnetworks and, therefore, traverse many possibly congested links. The first solution adopted to distribute the content trough the Internet consisted in mirroring. This technique statically replicates the web content in many locations across the Internet. Users manually select, from a list of servers, the best suited replica. The replica selection mechanism was automated and became transparent to the end-users with the introduction of the distributed web server systems [11][10].

With the introduction of proxy caching techniques to disseminate the content across the Internet, the bottlenecks at the server level and at the peering points were considerably reduced, though not ensuring a complete controllability of those systems by the content provider due to the absence of an intelligent and automated layer to perform server measures and request redirection. Proxy caching is only a semi-transparent mechanisms: the users, aware of the presence of a proxy server in their network, can/must configure their browser to use it, while the ISPs transparently manage their proxy caches. Large ISP proxy caches may also transparently cooperate with each other in a semi hierarchical structure. Proxy caches may experiment performance losses becoming themselves a bottleneck if there are frequent cache misses or cache inconsistencies. Besides this, proxy caches serve all requests independently of the required content, and do not prioritize users and QoS requirements.

In a CDN, by moving the content from multiple servers located at the edge of the Internet, a much more scalable model of distributing information and services to end-users is obtained, that is the so called edge delivery. In other words, a user would be able to find all requested content on a server within its home network. In this solution the requested content doesn't cross all the network before reaching its final destination, but only traverses the network part between the edge and the end-user. Further, cooperative access routers can be endowed with measure and server selection capabilities to perform a tradeoff solution between load balancing among the available servers and choosing the best suited replica to fulfil the agreements on quality of service.

2.2 The Features of a CDN

The design of a CDN requires, together with the distribution of replica servers at the edge of the network, a set of supporting services and capabilities. In order to be efficient for a significant number of users and for a considerably wide area, the edge servers must be deployed in thousands of networks, at different geographically spread locations. Optimal performance and reliability depend on the granularity of the distribution of the edge servers. The establishment of a CDN requires the design of some important features.

– **Replica placement mechanisms** are needed to decide the replica server locations and to adaptively fill them with the proper content prior to the

request arrival (pre-fetching). Thus servers are not filled upon request like in traditional proxy caching, but are pro-actively updated, causing a one time offloading overhead that is not repeated for every access to the origin server. Adaptivity in replica placement is required to cope with changing traffic condition and is not related to a pull behavior like in traditional caching.
- **Content update mechanisms** must be provided to automatically check the host site for changes and retrieve updated content for delivery to the edges of the network, thus ensuring content freshness. Standard mechanisms adopted in proxy caching do not guarantee content freshness since content stored on standard cache servers does not change as the source content changes.
- **Active measurement mechanisms** must be added to cooperative access routers to have immediate access to a real-time picture of the Internet traffic, in order to recognize the fastest route from the requesting users to the replica servers in any type of traffic situations, especially in presence of "flash crowds", that is sudden heavy demand, expected or not, for a single site. A measurement activity is at the basis of the replica selection mechanism.
- **Replica selection mechanisms** must be added to cooperative access routers to accurately locate the closest and most available edge server from which the end users can retrieve the required content. A robust service must also keep its servers from getting overloaded by means of access control and load balancing.
- **Re-routing mechanisms** must be able to quickly re-route content requests in response to traffic bursts and congestion as revealed by the measurement activity.

Also, the CDN infrastructure, must allow the service providers to access directly the caches and control their consistency and to get the statistics information about the accesses to the site, available from the cooperative access routers.

3 Types of Content and Services in a CDN

CDN providers host third party contents to fasten the delivery of any type of digital content, e.g. audio/video streaming media, html pages, images, formatted documents or applications. The content sources could be media companies, large enterprises, broadcasters, web/Internet service provider. Due to the heterogeneous nature of the content to be delivered, various architectures and technologies can be adopted to design and develop a CDN. We now analyze the characteristics of the content and of the applications that most likely take advantages of a CDN architecture.

- **Static web based services**. Used to access static content (static html pages, images, document, software patches, audio and/or video files) or content that change with low frequency or timely (volatile web pages, stock quote exchange). All CDN provider (Akamai Inc., Speedera Inc., AT&T inc., Globix Inc. just to mention some) support this type of content delivery. This

type of content can easily be cached and its freshness maintained at the edge using traditional content caching technologies.
- **Web storage services**. Essentially, this application can be based on the same techniques used for static content delivery. Additional features to manage logging and secure file transfer should be added. This type of application can require processing at the origin site or at the edge.
- **File transfer services**. World wide software distribution (patch, virus definition, etc.), e-learning material from an enterprise to all their global employees, movies-on-demand from a large media company, highly detailed medical images that are shared between doctors and hospitals, etc. All these content types are essentially static and can be maintained using the same techniques adopted for static web services.
- **E-commerce services**. The semantic of the query used in browsing a product catalogue is not complex, so frequent query results can be successfully cached using traditional DB query caching techniques[29][33]. Shopping charts can be stored and maintained at the replica server and also orders and credit card transactions can be processed at the edge: this requires trusted transaction-enabled replica servers. In [9] the authors propose a framework for enabling dynamic content caching for e-commerce site.
- **Web application**. Web transactions, data processing, database access, calendars, work schedules, all these services are typically characterized by an application logic that elaborates the client requests producing as results a dynamic web page. A partial solution to the employment of a CDN infrastructure in presence of dynamic pages is to fill the replica servers with the content that most frequently composes the dynamically generated web pages, and maintaining the application and its processing activity that produces the dynamic pages at the origin server. Another approach is to replicate both the application (or a portion of it) and the content at the edge server. In this way all the content generation process (application logic and content retrieval) are handled by the replica server thus offloading the origin server.
- **Directory services**. Used for access to database servers. For example, in the case of a LDAP server, frequent query results or a subsets of directories can be cached at the edge. Traditional DB query caching techniques [29] may be adopted.
- **Live or on-demand streaming**. In this case the edge server must have streaming capability. See section 3.1 for details.

Streaming media and application delivery are a challenge in CDN. A more detailed description of the solutions adopted for media streaming and dynamic contents can be found in the following subsections.

3.1 Streaming Media Content

Streaming media can be *live* and *on-demand*, thus a CDN needs to be able to deliver media in both these two modes. *Live* means that the content is delivered "instantly" from the encoder to the media server, and then onto the media client.

This is typically used for live events such as concerts or broadcasts. The end-to-end delay is at a minimum 20 seconds with today's technologies, so "live mode" is effectively "semi real-time". In *on-demand*, the content is encoded and then stored as streaming media files on media servers. The content is then available for request by media clients. This is typically used for content such as video or audio clips for later replay, e.g., video-on-demand, music clips, etc. A specialized server, called a media server, usually serves the digitalized and encoded content. The media server generally consists of media server software that runs on a general-purpose server. When a media client wishes to request a certain content, the media server responds to the query with the specific video or audio clip. The current product implementations of streaming servers are generally proprietary and demand that the encoder, server, and player all belong to the same vendor. Streaming servers also use specialized protocols (such as RTSP, RTP and MMS) for delivery of the content across the IP network. In [55] a typical streaming media CDN architecture is described. In streaming media CDNs a replica server must have, at least, the additional functionalities listed below.

- The ability to serve *live content* such as newscasts, concerts, or meetings etc. either in Multicast or Unicast mode.
- Support for delivery of stored or *on-demand content* such as training, archived meetings, news clips, etc.
- *Caching capability* of streaming media. Caching a large media file is unproductive, so typically media files are split in segment. Neighbor replica must be capable to share and exchange segment to minimize the network load and cache occupancy.
- *Peering capability* to exchange and retrieve content from the neighbor streaming cache in case of cache miss. Streaming cache node can be organized in a hierarchy.
- *Media transcoding functionality*, to adapt media streams for different client capabilities, e.g., low quality/bandwidth content to dial-up users, high quality/bandwidth to xDSL users.
- *Streaming session handoff* capability. The typically long life of a streaming session, in presence of user's mobility, causes the need for midstream handovers of streaming session between replica servers [5,49].

3.2 Web Application

Accessing dynamic content and other computer applications is one of the major challenges in CDN. CDN supporting this kind of content and services are also called Application Content Delivery Networks (ACDN). Some providers like AppStream Inc. and PIVIA Inc., implement ACDN using the so called "fat client" solution: the application is partitioned in "streamlets" or special applets and sent to the client. The client receives enough code to start the application and execute it, the other parts of the application are sent on demand. These solution use patented and proprietary technologies. Another approach is to migrate the application to the edge server using general utility such as Ajasent[1]

and vMatrix[4]. However application replication may be expensive especially if performed on demand. A completely different solution is to automatically deploy the application at the replica server. In [47] the authors define an ACDN architecture relying on standard technologies such as HTTP protocol, web servers, CGI/FastCGI scripts or servlets. Rabinovich et al. define the additional capabilities of an ACDN in terms of: an *application distribution framework* capable to dynamically deploy the application at the edge and to keep the replica consistent, a *content placement mechanism* to decide where and when to deploy the application, a *request distribution mechanism* aware of the location of the involved applications.

4 Content Delivery Networks Architecture

The main goal of server replication in a CDN is to avoid large amounts of data repeatedly traversing possibly congested links on the Internet. As Figure 1 shows, there are a variety of ways and scale (local area or wide area networks) in which content networks may be implemented. Local solutions are web clusters, that typically hosts single site, and web farms, typically used to host multiple sites. Wide area solutions include: distributed web server systems, used to host single or multiple sites; cooperative proxy cache networks (a service infrastructure to reduce latency in downloading web objects) and content delivery networks [53] that are the focus of this paper.

A typical server farm is a group of servers, ranging from two to thousands, that makes use of a so-called cooperative dispatcher, working at OSI layers 4 and/or 7, to hide the distributed nature of the system, thus appearing as a single origin site. A layer 4 web switch dispatches the requests, among a group of servers, on the basis of network layer information such as IP address and TCP port. A content switch, working at the application layer, examines the content of requests and dispatches them among a group of servers. The goals of a server cluster/farm include: load-balancing of requests across all servers in the group; automatic routing of requests away from servers that fail; routing all requests

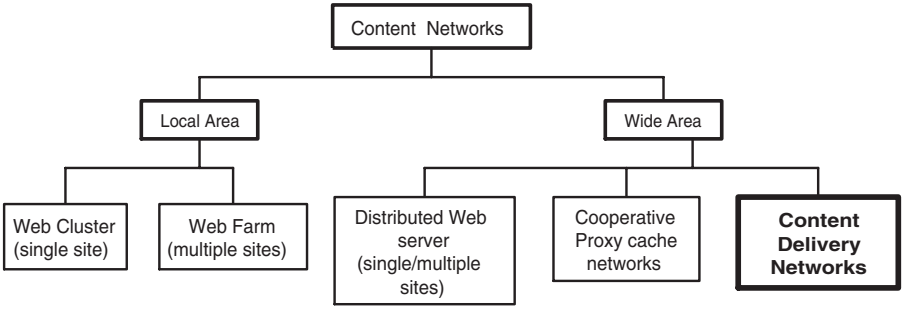

Fig. 1. Taxonomy of Content Networks

for a particular user agent's session to the same server, if necessary to preserve session state.

A type of content network that has been in use for several years is a caching proxy deployment. Such a network might typically be employed by an ISP for the benefit of narrow bandwidth users accessing the Internet. In order to improve performance and reduce bandwidth utilization, caching proxies are deployed close to the users. These users are encouraged to send their web requests through the caches rather than directly to origin servers, by configuring their browsers to do so. When this configuration is properly done, the user's entire browsing session goes through a specific caching proxy. This way the proxy cache would contain the hot portion of content that is being viewed by all the users of that caching proxy. A provider that deploys caches in many geographically locations may also deploy regional parent caches to further aggregate user requests thus creating an architecture known as hierarchical caching. This may provide additional performance improvements and bandwidth savings. Using rich parenting protocols, redundant parents may be deployed such that a failure in a primary parent is detected and a backup is used instead. Using similar parenting protocols, requests may be partitioned such that requests for certain content domains are sent to a specific primary parent. This can help to maximize the efficient use of caching proxy resources. Clients may also be able to communicate directly with multiple caching proxies.

Though certainly showing better scalability than a single origin server, both hierarchical caching and server farms have their limits. In these architectures, the replica servers are typically deployed in proximity to the origin server, therefore they do not introduce a significant improvement to the performance difficulties that are due to the network congestion. Caching proxies can improve performance difficulties due to congestion (since they are located in proximity to the final users) but they cache objects reactively to the client demand. Reactive caching based on client demand performs poorly if the requests for a given object, while numerous in aggregate, are spread among many different caching proxies.

To address these limitations, CDNs employ a solution based on proactive rather than on reactive caching, where the content is prefetched from the origin server and not cached on demand. In a CDN, multiple replicas host the same content. A request from a browser for a single content item is directed to the replica that is considered the best suited at the moment of the request arrival, and the item is served to the client in a shorter time than the one it would have taken to fetch it from its origin server. Since static information about geographic locations and network connectivity are not sufficient to choose the best replica, a CDN typically incorporates dynamic information about network conditions and load on the replicas, to redirect requests and balance the load among the servers. Operating a CDN is therefore a complex and expensive activity. For this reason a CDN is typically built and operated by a network/service provider that offers a content distribution service to several content providers.

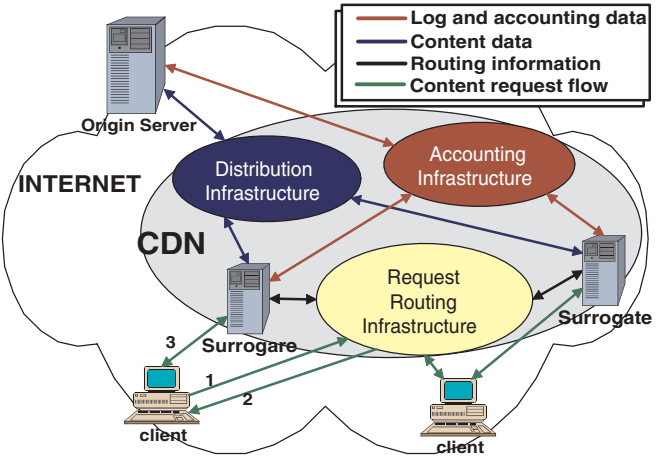

Fig. 2. Infrastructure components of a Content Delivery Network

A content delivery architecture consists of a set of **surrogate servers** that deliver copies of content to the users while combining different activities (see figure 2).

- the **request-routing** infrastructure consists of mechanisms to redirect content requests from a client to a suitable surrogate.
- the **distribution** infrastructure consists of mechanisms to move contents from the origin server to the surrogates.
- the **accounting** infrastructure tracks and collects data on request-routing, distribution, and delivery functions within the CDN creating logs and reports of distribution and delivery activities.

The **origin server** (hosting the content to be delivered) interacts with the CDN in two ways (see figure 2):

- it pushes new content to the replica servers, (the replica themselves request content updates from the origin server through the distribution infrastructure);
- it requests logs and other accounting data from the CDN or the CDN itself provides this data to the origin server through the accounting infrastructure.

The **clients** interact with the CDN through the request routing infrastructure and surrogate servers. Figure 2 shows one of the possible scenarios of interaction between the clients, the access routers, the replica servers and the origin server.

The user agent sends (1) a content request to the routing infrastructure, that redirects (2) the client request to a surrogate server, to which the client subsequently asks (3) the desired content.

5 Replica Placement and Management

5.1 Content Caching Techniques

The proactive caching infrastructure must be transparent to the end-users that must see no difference with being served directly by the central server. Proactive caching to the edges offers better delivery to the client because the content is located in their proximity. Therefore the requesting users perceive a lower latency, higher availability and lower load on the network links. Such architecture is also inherently protected from sudden burst that can be distributed among many servers so that no single device has to cope with a massive load. This close-to-the-client deployment mode is commonly known as forward proxy caching. Forward proxy implementations can reduce wide area network traffic by 30 to 50 percent (results vary based on the "cacheability" of the requested content). A Web cache monitors Internet traffic, intercepts requests for Web objects and then fulfils those requests from the set of objects it stores (*cache hit*). If the requested object is not in the cache (*cache miss*), the cache forwards the request to the origin server, that sends a copy of the object back to the cache. The cache store the object and sends it back to the requester. Caches in CDN cooperate interacting through the Internet Cache Protocol (ICP). ICP is typically used to build cache clusters or child-parent relationships in hierarchical caching[15] [44]. A cache can react to a cache miss inquiring other cooperative caches, in spite of the origin server, in order to retrieve the content from a closer location. Caching also acts as a point of control and security. Today's caches frequently include support for content filtering, anti-virus, access control and bandwidth management. Anti-virus and content filtering give users an extra level of security across the network. The access control and bandwidth management further assists in the reduction of the overall network utilization by making sure that only approved users get access to bandwidth, and that the bandwidth is being allocated in a way that ensures the best adherence to the signed agreements on quality. Caching activity in a CDN may involve different types of contents and therefore different functionalities.

Static Caching: to cache and replicate static content, such as html pages, images, documents, audio/video file etc.

Dynamic Caching: to cache and replicate dynamically generated content. This include application delivery and replication.

Streaming Media Caching: to store streaming media objects, as well as to serve streaming media to clients. Essentially, the cache acts as a streaming media server, storing media clips for later use.

Live Splitting: to cache replicated live streams, so that only one copy is pulled down from the upstream server and is then distributed to the subscribing clients.

5.2 Replica Placement

An hot topics in content delivery design is the replica placement problem: where and how the replica could be distributed across the Internet to minimize the user latency, the number of replica, and the bandwidth used for replica management?

The majority of the schemes presented in the literature tackle the problem of static replica placement that can be formulated as follows. Given a network topology, a set of CDN servers and a given request traffic pattern, decide where content has to be replicated so that some objective function is optimized while meeting constraints on the system resources. The solutions so far proposed typically try to either maximize the user perceived quality given an existing infrastructure, or to minimize the CDN infrastructure cost while meeting a specified user perceived performance. Examples of constraints taken into account are limits on the servers storage, on the servers sustainable load, on the maximum delay tolerable by the users etc.

A thorough survey of the different objective functions and constraints considered in the literature can be found in [37]. For the static case, simple efficient greedy solutions have been proposed in [46], [35] and [48]. In [46] Qiu et al. formulate the static replica placement problem as a minimum K median problem, in which K replicas have to be selected so that the sum of the distances between the users and their 'best replica' is minimized. In [35] and [48] Jamin et al. and Radoslavov et al. propose fan-out based heuristics in which replicas are placed at the nodes with the highest fan-out irrespective of the actual cost function. The rationale is that such nodes are likely to be in strategic places, closest (on average) to all other nodes, and therefore suitable for replica location. In [48] a performance evaluation based on real-world router-level topology shows that the fan-out based heuristic has behavior close to the greedy heuristic in terms of the average client latency. In both [18] and [41] the authors consider the problem of placing replicas for one origin server on a tree topology. In [40] the authors also consider very simple topology like rings and lines and tree topologies while considering the placement of intercepting proxies inside the network to reduce download time. In [36] the problem of optimally replicating objects in CDN servers is analyzed. All these solutions lack in considering the dynamics of the system (e.g. changes in the requests traffic pattern, network topology, replica sites).

In [6] a different approach is proposed and a dynamic allocation strategy is considered which explicitly takes into account the system dynamics as well as the costs of modifying the replica placement. By assuming the users requests dynamics to obey to a Markovian model a formulation of the dynamic replica placement problem as a Markovian decision process is obtained. Albeit this model may not accurately capture the user dynamics and can be numerically solved only for limited sized CDNs, it allows us to identify an optimal policy for dynamic replica placement that can be used as a benchmark for heuristics evaluation and provides insights on the formulation of a conservative placement heuristic.

The solution of replica placement for heavy contents, like video streaming, makes it impossible storing the entire content of several long streams because

it would exhaust the capacity of a conventional cache. In this case, not only the content must be replicated and distributed to replicas, but also in a way that avoids to overload servers. To address this problem, in [51,54] the authors propose a prefix caching technique whereby a proxy stores the initial frames of popular clips. Upon receiving request for the stream, the proxy initiates transmission to the client and simultaneously requests the remaining frames from the server. In addition to hiding the delay, throughput and loss effects of a weaker service model between the server and the proxy, this caching technique aids the proxies in performing work ahead smoothing into the client playback buffer, by transmitting large frames in advance of each burst. The prefix caching techniques reduces the peak and variability of the network resources requirements along the path from the proxy to the client.

5.3 Cache Consistency and Content Flow

One of the important problems in CDNs is how to manage the consistency of content at replicas with that at the origin server, especially for those documents changing dynamically. Cached objects typically have associated expiration times after which they are considered stale and must be validated with a remote server (origin or another cache) before they can be sent to a client. Sometimes, a considerable fraction of cache hits involve stale copies that turned out to be current. These validations of current objects have small message size, but nonetheless, they often induce latency comparable to cache misses. Thus the functionality of caches as latency-reducing mechanism highly depends not only on content availability but also on its freshness.

A technique to achieve cache consistency consists in pre-populating, or pushing, content to the cache before requests arrive. When automatically pushing a new, or updated, Web object to a cache, the content in the cache is guaranteed to be always fresh and there is no reason for the cache to initiate a freshness check with the side effect that this technique often generates a large amount of traffic.

In [21] the authors propose policies for populating caches to proactively validate selected objects as they become stale, and thus allow for more client requests to be processed locally. Pre-populating content takes on even more importance with broadband content. The size of rich content files can be huge and increasing every day. Compression technologies have been invented specifically for these new and emerging types of media. The load limits for servers need to be established by means of load testing of the specific environment. However, with content pre-populating a high resolution file can be pushed across low speed lines directly to the Web cache in the branch office and then serve that streaming file at the top speed of your LAN. In the traditional propagation approach the updated version of a document is delivered to all replicas whenever a change is made to the document at the origin server. It may generate significant levels of unnecessary traffic if documents are updated more frequently than accessed.

Another approach is invalidation, in which an invalidation message is sent to all replicas when a document is changed at the origin server. This approach

doesn't make full use of the distribution networks for content delivery and each replica needs to fetch an updated version individually at a later time. This can also lead to inefficiency in managing consistency at replicas.

In [27] the author propose a hybrid approach that generates less traffic than the propagation and the invalidation approach. The origin server makes the decision of using either propagation or invalidation method for each document, based on the statistics about the update frequency at the origin server and the request rated collected by replicas. They develop a technique that can reduce the burden of request rate collection at replicas and avoid the implosion problem when replicas send the statistics to the origin server.

Another main focus in CDNs research activity is how content at the origin servers have to be delivered to replicas. Two common approaches to this problem are to deliver data over N unicast channels, or over an application-level (tunnelled) multicast tree that connects the replicas [16,17]. Basically they run an auto-configuration protocol to establish a delivery structure of tunnelled topology among participating members. These approaches consist in building a mesh topology first and running the spanning tree algorithm to select a delivery tree. Intuitively, the unicast approach wastes network bandwidth and can cause congestion at bottleneck links, while application-level multicast approach is more efficient in delivery (although not as efficient as native IP multicast).

Another issue in cache management is the propagation of changes (adds and deletes of content). When content is added to the source-tree, it must also become available throughout the CDN. If there are delays involved in propagating the content, through all the CDN replicas, the content providers must be aware of that there may be periods in which contents may be inconsistent and even unavailable. For example, if a film studio wishes to make a new movie available to their global customer base, they need to know how long time it takes to make it globally available. This way, they can make sure that they do not start selling it until everyone can access it. Similarly, when the customer deletes content from their file area, it should also be expired from the caches and devices within the CDN.

6 Measurements Techniques for Request Routing

Request routing systems can use a variety of metrics in order to determine the best surrogate that can serve a client's request. The decentralized nature of the Internet makes quantitative assessment of network performance very difficult. Collecting network statistics directly on network devices (router and server) could be more expensive in terms of system performance. So typically, the acquisition of network statistics relies on the use of a combination of active network probing methods, passive traffic monitoring and feedback from surrogate servers. For deeper details of how measures can be inferred and network tomography can be done see also [7,20]. In CDN networks it is possible to combine multiple metrics using both proximity concept and surrogate feedback for best surrogate selection. Performance measurement is often a component of network management

systems and offers the ability to monitor, understand, and project end-to-end performance of the CDN. Moreover one would need to measure both the internal performance, as well as the performance from the customer perspective. Typical parameters that would be useful to measure are: packet-loss and latency for all type of content and in particular for streaming content average bandwidth, startup time and frame rate. By deploying hardware-based or software probes, strategically throughout the network, one could correlate the information collected by the probes with the cache and server logs to determine delivery and QoS statistics. The most useful place to put the probes is at the edges of the network, thus measuring the performance as perceived by the end-users throughout the CDN. Network and geographical proximity measurements can be used by the request routing system to direct users to the "closest" surrogate. Furthermore, proximity measurements can be exchanged between surrogates and the requesting entity. In many cases, proximity measurements are "one-way" in that they measure either the forward or reverse path of packets from the surrogate to the requesting entity. This is important as many paths in the Internet are asymmetric. In order to obtain a set of proximity measurements, a network may employ active probing techniques and/or passive measurement techniques. The request-routing system can use also feedback from surrogates in order to select a "least-loaded" delivery node. Feedback can be delivered from each surrogate or can be aggregated by site or by location. We now discuss in detail about passive measurement, active probing and feedback information.

Passive Measurement. Passive measurements could be obtained when a client performs data transfers to or from a surrogate. 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 passive measurement is to watch the packet loss from a client to a surrogate, or the user perceived latency by observing TCP behavior. Basically, a good mechanism is needed to ensure that not every surrogate is tested per client in order to obtain the data. In [52] the authors proposed a system based on passive measurements of the network performance to be used with adaptive applications.

Active Probing. Active probing is when past or possible requesting entities are probed using one or more techniques to determine one or more metrics from each surrogate or set of surrogates. An example of a probing technique is an ICMP ECHO request that is periodically sent from each surrogate or set of surrogates to a potential requesting entity. Active probing techniques are limited for some reasons. Measurements can only be taken periodically and should not have a perceivable load since it cannot influence the measured traffic, firewalls and NATs disallow probes, and last, probes often cause security alarms to be triggered on intrusion detection systems.

Feedback information. These information may be obtained by periodically probing a surrogate by issuing application specific requests (e.g. HTTP) and taking related measures. The problems with probing for surrogate information is that it is difficult to obtain "real-time" information and the non-real-time information are sometimes inaccurate and obsolete. Consequently,

feedback information can be obtained by agents that reside on surrogates that can communicate a variety of metrics about their nodes. There are two methods to obtain feedback information: static, in which the route that minimize the number of hops or to optimize other static parameters is selected [25][50]; dynamic probing (Real Time probing) allow to compute round trip time or other QoS parameters in "real time" [25][22]. Hybrid methods are also used to obtain other useful feedback information [25][26][43].

6.1 Metrics for Request Redirection

Replica server selection is performed with the goal to minimize some performance parameters perceived by the end-users. In this section we give a classification of the metrics that can be adopted to measure the network and systems performance in a CDN to decide where to redirect client's requests.

Geographical proximity is often used to redirect all users within a certain region to the same POP.

A measure of the *network proximity* is typically derived through active probing of the BGP routing table.

The ability to select the POP that shows the lowest latency can be obtained by enhancing the request redirection systems with active probing and passive measurement mechanisms, to maintain knowledge of *response time*.

The *server load state* can be computed, using SNMP or feedback agents on the server side, on the basis of the load state of server components (CPU, disk, memory, network interfaces) or on the basis of some aggregate performance measure, such as the server throughput or server response time.

All these measures may be relevant information to feed the server selection mechanism, combined with the knowledge of the *users identity*, that is intended to classify the user priority in accessing contents and services. As an example, paying customer may get access to better service than non-paying. The identity of paying users can be revealed by means of a cookie retrieved from the client system, or through an authentication process.

Figure 3 summarizes the above performance metrics classification.

7 Request-Redirection Mechanisms

A key challenge in CDN design is to realize efficient request-redirection service that tracks the servers where data objects are replicated and assigns each client request to a server that can offer the "best" service, this process is called server selection. Server selection algorithms include criteria like network topology, server availability and server load [13]. The ability to quickly locate replicas and perform request distribution has critical implications for the user-perceived response time. Figure 4 illustrates a high-level view of the request-redirection process: 1) the client requests a given content, e.g. a streaming file, residing at www.site.com; 2) since site.com doesn't host the requested file, but uses cdn.com as its CDN provider, the request is redirected to the cdn.com site. 3-3') By using

Metrics	*Goals*	*Measurement Techniques*
Latency	Select replica with lowest delay	Active Probing / Passive Measurement
Packet loss	Select path with lowest error rate (useful for streaming traffic)	Active Probing / Passive Measurement (TCP header info)
Network proximity	Select the shortest path	Active Probing
Avg. Bandwidth	Select the best path for streaming traffic	
Sturtup Time		
Frame Rate		
Geographical proximity	Redirect requests from a region to the same POP	IP header information, bind information
CPU load, net. interface load, active connection, storage I/O load	Select the server with the aggregated least load	feedback agents/active probing

Fig. 3. Metrics used in replica selection

some redirection algorithm, the media client gets redirected to the most appropriate replica. If the client has a CDN replica directly placed at its ISP network that is capable of guaranteeing the fulfillment of the SLA, this replica is selected (3) first (e.g. CDN cache-2), otherwise another one (e.g. CDN cache-1) is selected (3'). 4-4') The selected CDN replica serves the content to the client.

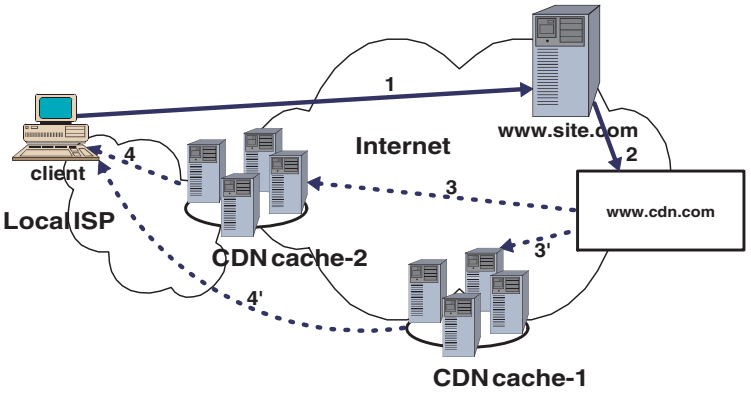

Fig. 4. The request-redirection process

Various schemes have been proposed in research literature to perform redirection at the *IP level* through some address packet rewriting scheme [24,34], at the *DNS level* through mapping of URL-name to IP-address of one of the servers

in a cluster [19,3,56], or at the *application level* [2] using the redirection features of the HTTP protocol. In [23] the authors propose a hybrid scheme based on adaptive replication of the entries of the location directory that provides the redirection service. Network level support enables client requests to be redirected to appropriate servers in a quick and efficient manner. In [12] the authors focus on an alternative architecture that integrates DNS dispatching mechanism with a HTTP redirection technique carried out by Web servers. In the following sections we describe in details the main request-redirection mechanisms.

7.1 DNS Based Request Routing

Today commercial content distribution services rely on modified Domain Name System servers to dynamically redirect clients to the appropriate content server. This is the simplest form of redirection, according to which a domain name, e.g. www.cdn.com has multiple IP records attached to it. When a client requests the IP address of the domain, any one of the IP records from the pool will be selected based on the DNS action.

The popularity of DNS based request-routing techniques is mainly due to the ubiquity of DNS as a directory service. They mainly consist in inserting a specialized DNS server in the name resolution process. This specialized server is capable of returning a different set of A, NS or CNAME records based on user defined policies, metrics, or a combination of both.

In the **single reply** approach, the DNS server returns the IP address of the best surrogate in an A record to the requesting DNS server (client site DNS server's). The IP address of the surrogate could also be a virtual IP address of the best set of surrogates for requesting DNS server. The best surrogate will be selected, in a second step, by a server switching mechanism.

In the **multiple replies** approach, the request-routing DNS server returns multiple replies such as several A records for various surrogates. Common implementations of client site DNS servers cycle through the multiple replies in a round robin fashion. The order in which the records are returned can be used to direct multiple clients using a single client site DNS server.

A **multiple-level** resolution approach is also possible according to which multiple request-routing DNS servers can be involved in a single DNS resolution, thus demanding complex decisions from a single server to multiple, more specialized, request-routing DNS servers, disseminated in different points of the Internet.

The most common mechanisms used to insert multiple request-routing DNS servers in a single DNS resolution is the use of NS and CNAME records: NS records allow the DNS server to redirect the authority of the next level domain to another request-routing DNS server; CNAME records allow the DNS server to redirect the resolution request to an entirely new domain.

There are three main drawbacks of using NS records. First the number of request-routing DNS servers are limited by the number of parts in the DNS name; second the last DNS server can determine the Time To Live (TTL) of the entire resolution process. The client will cache the returned NS record and use

it for further request resolutions until it expires. As a third drawback, a delay is added in the resolution process due to the use of multiple DNS servers.

Request-routing based on CNAME record has the advantage to redirect the resolution process to another domain, and the number of request-routing DNS servers is independent of the format of the domain name. The main disadvantage is the introduction of an additional overhead in resolving the new domain name.

The basic limitations of DNS based request-routing techniques can be summarized as follows below:

- DNS allows resolution only at the domain level. However, an ideal request resolution system should serve requests at object granularity (preserving sessions if needed).
- A short TTL of DNS entry allows to react quickly to network outages. This in return may increase the volume of requests to DNS servers. Therefore many DNS implementations do not honor the DNS TTL field.
- DNS request-routing does not take into account the IP address of the clients. Only the Internet location of the client DNS server is known: this limits the ability of the request-routing system to determine a client's proximity to the surrogate.
- Users that share a single client site DNS server will be redirected to the same set of IP addresses during the TTL interval. This might lead to overloading the surrogate during a flash crowd.

7.2 Transport Layer Request Routing

At the transport-layer, finer levels of granularity can be achieved by means of a closer inspection of client's requests. This level provides information about the client's IP address, TCP port, and other layer 4 header information. These data could be used in conjunction with other load state metrics to select the surrogate that is better suited to serve a given request. In general, the forward-flow traffic (client to newly selected surrogate) will flow through the request routing server or through a first step surrogate originally chosen by the DNS. The reverse-flow traffic (surrogate to client), which normally transfers much more data than the forward-flow, would typically take the direct path from the servant surrogate. The overhead associated with transport-layer request-routing makes it better suited for long-lived sessions such as file transfer (FTP), secure transactions (TLS, SSL), streaming (RTP, RTSP). However, transport-layer request-routing could also be used to redirect clients away from overloaded surrogates. A first course grain replica selection is operated by a DNS request router. The selected server, may operate a more accurate refinement of replica selection, based on local and server side information, using the transport-layer request-routing mechanism.

7.3 Application-Layer Request Routing

With application layer request-routing a fine-grained control, at the level of individual objects composing the multimedia content, can be achieved. The process

could be performed, in real time, when the object request reaches the content server or a switching element. In most cases the header of the client's packet contains enough information to perform request routing. Some application level protocols such as HTTP, RTSP, and SSL provide necessary information in the initial portion of the session about how the client request must be directed. Application level protocols such as HTTP and RTSP may describe the requested content by means of its URL, other redirection information come from other parts of the MIME request header such as Cookies. In many cases the URL is sufficient to disambiguate the content and suitably redirect the request.

Header Inspection. This approach is based on the inspection of the header of client requests. In a first solution, also known as *302 redirection* [28], the client's request is first resolved to a virtual surrogate that returns an application-specific code, such as the 302 in the case of HTTP or RTSP, to redirect the client to the delegate content delivery node. The application layer redirection is relatively simple to implement. Nevertheless, this approach introduces an additional latency involved in sending the redirect message back to the client. Another approach, known as the *In-Path element* considers a network element, in the forwarding path of the client's request, that provides transparent interception of the transport connection. This In-Path network element, establishes a connection with the client, examines the client's content request and performs request-routing decisions. Then, the client connection is joined to a connection with the appropriate content delivery node. Drawbacks are the delay introduced for URL-parsing and the possible bottleneck introduced by the In-Path element.

Content modification. This technique enables a content provider to take direct control over request-routing decisions without the need for specific switching devices or directory services in the path between the client and the origin server. Basically, a content provider can directly communicate to the client the best surrogate that can serve the request. Decisions about the best surrogate can be made on a per-object basis or it can depend on a set of metrics. In general, the method takes advantage of content objects that consist of basic structure that includes references to additional, embedded objects. For example, most web pages, consist of an HTML document that contains plain text together with some embedded objects (e.g. GIF, JPEG images or PDF documents) referenced using embedded HTML directives. In general embedded objects are retrieved from the origin server. A content provider can now modify references to embedded objects such that they could be fetched from the best surrogate.

Pro-active URL Rewriting. According to this scheme, a content provider formulates the embedded URLs of a main html page before the content is loaded on the origin server. In this case, URL rewriting can be done either manually or by using software tools that parse the content and replace embedded URLs. Since these scheme consists in rewriting URLs in a proactive way, it cannot take into consideration client specific information while performing request routing. However, it can be used in combination with DNS request routing to direct related DNS queries into the domain name space of

the service provider. Dynamic request routing based on client specifics are then done using the DNS approach.

Reactive URL Rewriting. This dynamic scheme consists in rewriting the embedded URLs of a html page when the client request reaches the origin server. In spite of the previous scheme, this one has the possibility to consider the identity of the client when rewriting the embedded URLs. In particular, an automated process can determine, on-demand, which surrogate would serve the requesting client best. The embedded URLs can then be rewritten to redirect the client to retrieve the objects from the surrogate that can fulfill the client request better than the other, with a consideration of the specific location and priority of the considered client.

A drawback of content modification based request-routing is that the first request, from a client to a specific site, must be served from the origin server. Besides, content that has been modified to include references to nearby surrogates rather than to the origin server should be marked as non-cacheable. To reduce this limitation, such pages can be marked to be cacheable only for a relatively short period of time. However, rewritten URLs on cached pages can cause problems, because they can get outdated and point to surrogates that are no longer available or no longer the best choice.

7.4 Anycast

This solution aims at solving the request-routing problem at the IP packet routing level. The base principle is that a group of servers providing the same service can be addressed using an anycast name and an anycast address[38,14]. A user willing to access some service, e.g., a given content, from any of the (equivalent) servers issues a request with the anycast name. This is mapped to the anycast address and the request is sent into the network with the anycast address as destination. The role of the anycast service, is to redirect these request to one of the servers, thus selecting the server which will serve the user request. Since the redirection system has the obvious goal of improving clients performance, redirection is based upon some performance criteria, e.g. minimize the user perceived response time. The redirection is therefore performed by a cooperative access router that is capable of selecting the best suited replica from the anycast table. Anycasting is a more elaborate location mechanism that targets network-wide replication of the servers over potentially heterogeneous platforms. A mechanism for request redirection that is based on the anycasting concept must allow for the maintenance of information about the servers state and performance. An anycast service can be implemented at different levels in the network protocol stack. At the network layer, anycasting mechanisms consist in associating a common IP anycast address with the group of replicated servers. The routing protocol routes datagrams to the closest server, using the routing distance metric. Standard intra-domain unicast routing protocols can accomplish this, assuming each server advertises the common IP address. In [45] the authors propose a solution to the server selection problem by introducing the idea of anycasting at the network layer. This work established the semantics of anycasting service within the

Internet. At the network level the implementation of an anycast service entails the following mechanisms:

- Anycast request interception, that maybe dealt at the network level by the edge routers. Edge routers are set to filter packets with anycast destination address.
- Anycast name to Anycast address translation mechanism.
- Server Selection that maybe dealt by an edge router module (or by an application interacting with the router). This module interacts with the measurement module and keeps and updates anycast server performance metrics.
- Anycast address to IP address translation that maybe dealt at the network level by the edge routers. Upon receiving an anycast address, edge routers perform redirection by translating them into a selected unicast address.
- Measurements collection to be used for the selection process itself.

Application layer anycast implementation is also proposed in [27,57]. The authors claim that at the application layer a better flexibility can be achieved if compared to the network layer implementation. At the application level the resolution of the anycast address is performed by means of a hierarchy of anycast resolvers that map the anycast domain name (AND) onto an IP address. To perform the mapping the resolvers maintain two types of information: 1) the list of IP addresses that form particular anycast groups, and 2) a metric database information associated with each member of the anycast group, while authoritative resolvers maintain the definitive list of IP addresses for a group, whereas local resolvers cache this information.

8 Conclusions

In this paper we have introduced the design principles at the basis of Content Delivery Networks (CDN). CDNs rely on a proactive distribution of content replicas to geographically distributed servers close to the edge of the network, in proximity to the end users. The redirection of a request to the best suited replica is performed by cooperative access routers that are capable of taking measures regarding the performance of the available replica servers, and performing the replica selection and the request redirection to the selected replica. CDNs are therefore complex systems. Their understanding and design involves knowledge in many research fields: internet traffic measurement, content caching, request rerouting at various communication protocol levels, request admission control, load balancing and more, depending on the type of content and application considered. We gave a high level description of all the mechanisms and technologies used in CDN. We first introduced the main motivations for content delivery and explored what types of content and services may beneficiate from the introduction of a CDN architecture. Then we focused on the main research problems related to the CDN design, in particular, the problems of replica placement and management, server selection and request redirection have been analyzed in more details.

References

1. Utility computing: Solutions for the next generation it infrastructure. www.ajasent.com/technology/whitepapers2.html.
2. Winddance networks corporation. http://www.winddancenet.com.
3. D. Andersen, T. Yang, V. Holmedahl, and O. Ibarra. Sweb: Toward a scalable world wide web server on multicomputers. In *Proceedings of International Parallel Processing Symposium*, April 1996.
4. A. Awadallah and M. Rosenblum. The vmatrix: A network of virtual machine monitors for dynamic content distribution, 2002.
5. N. Bartolini, E. Casalicchio, and S. Tucci. Mobility-aware admission control in content delivery networks. In *Proceedings of IEEE/ACM MASCOTS*, Orlando, Florida, October 2003.
6. N. Bartolini, F. Lo Presti, and C. Petrioli. Optimal dynamic replica placement in content delivery networks. In *Proc of the 11th IEEE Int. Conference on Networks (ICON)*, Sydney, Australia, Sept. 2003.
7. T. Bu, N. Duffield, F. Lo Presti, and D. Towsley. Network tomography on general topologies. In *Proc of ACM SIGMETRICS*, Marina del Rey, California, June 2002.
8. M. Calzarossa. Performance Evaluation of Mail Systems. In M. Calzarossa and E. Gelenbe, editors, *Performance Tools and Applications to Networked Systems*, volume 2965 of *Lecture Notes in Computer Science*. Springer, 2004.
9. K. S. Candan, W.-S. Li, Q. Luo, W.-P. Hsiung, and D. Agrawal. Enabling dynamic content caching for database-driven web sites. In *Proc. of ACM/SIGMOD Conference 2001, Santa Barbare, California, USA*, May 2001.
10. V. Cardellini, E. Casalicchio, M. Colajanni, and P. Yu. The state of the art in locally distributed web-server systems. *ACM Computing Surveys*, 34(2):263–311, 2002.
11. V. Cardellini, M. Colajanni, and P. Yu. Dynamic load balancing on web-server systems. *IEEE Internet Computing*, 3(3):28–39, May-June 1999.
12. V. Cardellini, M. Colajanni, and P. Yu. Redirection algorithms for load sharing in distributed web server systems. In *Proc. IEEE 19th Int'l Conf. on Distributed Computing Systems (ICDCS)*, 1999.
13. R. L. Carter and M. E. Crovella. On the network impact of dynamic server selection. *Computer Network*, 1999.
14. M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scalable application-level anycast for highly dynamic groups.
15. A. Chankhunthod, P. B. Dansig, C. Neerdaels, M. F. Schwartz, and K. J. Worrel. A hierarchical internet object cache. In *Proceedings of USENIX*, January 1996.
16. Y. Chawathe, S. McCanne, and E. A. Brewer. An architecture for internet content distribution as an infrstructure service. not published.
17. Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM Sigmetrics*, June 2000.
18. I. Cidon, S. Kutten, and R. Soffer. Optimal allocation of electronic content. In *Proceedings of IEEE Infocom*, 2001.
19. Cisco. Cisco's distributed director. http://www.cisco.com/warp/public/cc/pd/cxsr/dd/index.shtml.
20. M. Coates, A. O. Hero, R. Novak, and B. Yu. Internet tomography. *IEEE Signal Processing Magazine*, May 2002.
21. E. Cohen and H. Kaplan. Refreshment policies for web content caches. In *Proceedings of IEEE Infocom*, 2001.

22. M. E. Crovella and R. L. Carter. Dynamic server selection in the internet. In *Proceedings of the Third IEEE Workshop on the Architecture and implementation of High Performance Communication Subsystems, HPCS*, 1995.
23. K. Dasgupta and K. Kalpakis. Maintaining replicated redirection services in web-based information systems. In *Proceedings of the 2nd IEEE Workshop on Internet Applications*, 2001.
24. D. M. Dias, W. Kish, R. Mukherhee, and R. Tewari. A scalable and high available web server. In *Proceedings of the 41st IEEE Computer Society International Conference*, 1996.
25. S. G. Dykes, K. A. Robbins, and C. L. Jeffery. An empirical evaluation of client-side server selection algorithm. In *Proceedings of IEEE Infocom*, 2000.
26. EICE. Internet qos measurement for the server selection. Technical report, Technical Report of EICE, CQ2000-48, 2000.
27. Z. Fei, S. Bhattacharjee, E. Zegura, and M. H. Ammar. A novel server selection technique for improving the response time of a replicated service. In *Proceedings of Infocom'98*, 1998.
28. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Rfc2616: Hypertext transfer protocol – http/1.1. Rfc, June 1999.
29. D. Florescu, A. Y. Levy, and A. O. Mendelzon. Database techniques for the world-wide web: A survey. *SIGMOD Record*, 27(3):59–74, 1998.
30. E. Gelenbe, K. Hussain, and V. Kaptan. Enabling Simulation with Augmented Reality. In M. Calzarossa and E. Gelenbe, editors, *Performance Tools and Applications to Networked Systems*, volume 2965 of *Lecture Notes in Computer Science*. Springer, 2004.
31. E. Gelenbe, R. Lent, M. Gellman, P. Liu, and P. Su. CPN and QoS Driven Smart Routing in Wired and Wireless Networks. In M. Calzarossa and E. Gelenbe, editors, *Performance Tools and Applications to Networked Systems*, volume 2965 of *Lecture Notes in Computer Science*. Springer, 2004.
32. S. Gilmore, J. Hillston, and L. Kloul. PEPA Nets. In M. Calzarossa and E. Gelenbe, editors, *Performance Tools and Applications to Networked Systems*, volume 2965 of *Lecture Notes in Computer Science*. Springer, 2004.
33. R. Goldman and J. Widom. WSQ/DSQ: A practical approach for combined querying of databases and the web. In *SIGMOD Conference*, pages 285–296, 2000.
34. G. Hunt, G. Goldzmidt, R.P.King, and R. Mukherjee. Network dispatcher: A connection router for scalable internet services. In *Proceedings of 7th International World Wide Web Conference*, April 1998.
35. S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt. Constrained mirror placement on the internet. In *INFOCOM*, pages 31–40, 2001.
36. J. Kangasharju, J. Roberts, and K. W. Ross. Object replication strategies in content distribution networks. *Computer Communications*, 25(4):367–383, March 2002.
37. M. Karlsson, C. Karamanolis, and M. Mahalingam. A unified framework for evaluating replica placement algorithms. *Technical Report HPL-2002, Hewlett Packard Laboratories*.
38. D. Katabi and J. Wroclawski. A framework for scalable global IP-anycast (GIA). In *SIGCOMM*, pages 3–15, 2000.
39. A. Klemm, C. Lindemann, and O. Waldhorst. Peer–to–Peer Computing in Mobile Ad Hoc Networks. In M. Calzarossa and E. Gelenbe, editors, *Performance Tools and Applications to Networked Systems*, volume 2965 of *Lecture Notes in Computer Science*. Springer, 2004.

40. P. Krishan, D. Raz, and Y. Shavitt. The cache location problem. *IEEE/ACM Transactions on Networking*, October 2000.
41. B. Li, J. Golin, G. F. Italiano, and X. Deng. On the optimal placement of web proxies in the internet. In *Proceedings of IEEE Infocom*, 1999.
42. P. Lorenz. IP–Oriented QoS in the Next Generation Networks: Application to Wireless Networks. In M. Calzarossa and E. Gelenbe, editors, *Performance Tools and Applications to Networked Systems*, volume 2965 of *Lecture Notes in Computer Science*. Springer, 2004.
43. K. Mase, A. Tsuno, Y. Toyama, and N. Karasawa. A web server selection algorithm using qos measurement. In *Proceedings of International Conference on Communication*, 2001.
44. S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson. Adaptive web caching: Towards a new caching architecture. *Computer Network and ISDN Systems*, November 1998.
45. C. Partridge, T. Mendez, and W. Milliken. Host anycasting service. http://rfc.sunsite.dk/rfc/rfc1546.html.
46. L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In *INFOCOM*, pages 1587–1596, 2001.
47. M. Rabinovich, Z. Xiao, and A. Aggarwal. Computing on the edge: A platform for replicating internet applications. In *Proc of Int. Workshop on Web Content Caching and Distribution*, Hawthorne, NY USA, Sept. 2003.
48. P. Radoslavov, R. Govindan, and D. Estrin. Topology-informed internet replica placement. *Proceedings of WCW'01: Web Caching and Content Distribution Workshop, Boston, MA, June 2001.*
49. S. Roy, B. Shen, V. Sundaram, and R. Kumar. Application level hand-off support for mobile media transcoding sessions. In *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 95–104. ACM Press, 2002.
50. M. Sayal, Y. Breithart, P. Scheuermann, and R. Vingralek. Selection algorithms for replicated web servers. 26(3), December 1998.
51. S. Sen, J. Rexford, and D. Towsley. Proxy prefix caching for multimedia streams. In *Proceedings of IEEE Infocom*, 1999.
52. M. Stemm, R. Katz, and S. Seshan. A network measurement architecture for adaptive applications. In *Proceedings of IEEE Infocom*, 2001.
53. D. C. Verma. *Content Distribution Networks: An engineering approach.* Wiley Inter-Science, 2002.
54. B. Wang, S. Sen, M. Adler, and D. Towsley. Optimal proxy cache allocation for efficient streaming media distribution. In *Proceedings of Infocom*, 2002.
55. S. Wee, J. Apostolopoulos, W. Tan, and S. Roy. Research and design of a mobile streaming media content delivery network. In *Proceedings of IEEE International Conference on Multimedia & Expo*, 2003.
56. P. Yu and D. M. Dias. Analysis of task assignment policies in scalable distributed web-server systems. *IEEE Transaction on Parallel and Distributed Systems*, June 1998.
57. E. Zegura, M. Ammar, Z. Fei, and S. Bhattacharjee. Application-layer anycasting: a server selection architecture and use in a replicated web service. *IEEE/ACM Transaction on Networking*, 8(4), August 2000.