Overlay Networks: An Akamai Perspective

Ramesh K. Sitaraman[1,2], Mangesh Kasbekar[1], Woody Lichtenstein[1], and Manish Jain[1]

[1]Akamai Technologies Inc*
[2]University of Massachusetts, Amherst

1 Introduction

The Internet is transforming every aspect of communication in human society by enabling a wide range of applications for business, commerce, entertainment, news, and social interaction. Modern and future distributed applications require high reliability, performance, security, and scalability, and yet need to be developed rapidly and sustainably at low operating costs. For instance, major e-commerce sites require at least "four nines" (99.99%) of reliability, allowing no more than a few minutes of downtime per month. As another example, the future migration of high-quality television to the Internet would require massive scalability to flawlessly transport tens of petabits per second of data to global audiences around the world.

However, the Internet was never architected to provide the stringent requirements of such modern and futuristic distributed applications. It was created as a heterogeneous network of networks, and its design enables various entities to interact with each other in a "best effort" fashion. Guarantees on high performance, availability, scalability and security are not inherently provided on the Internet in accordance with its best effort design principle. Today's Internet is a vast patchwork of more than 13,000 autonomous networks that often compete for business. Failures and performance degradation in transporting information across this patchwork are routine occurrences.

So, how would we bridge the gap between what modern Internet-based services need and what the Internet actually provides? A complete clean-

*Email: ramesh@cs.umass.edu, {mkasbeka,wlichten,majain}@akamai.com

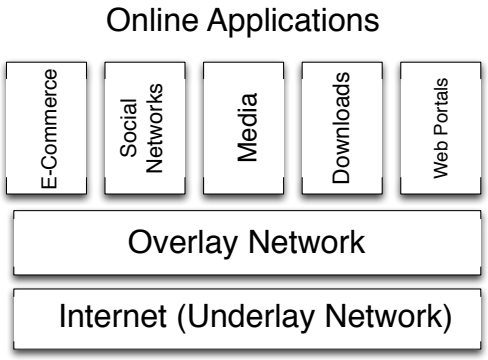


Figure 1: An overlay network is built on top of the public Internet to provide the stringent requirements that rich Internet-based services need.

slate redesign of the Internet is appealing, but would be hard to implement given the wide-spread adoption of the current technology. A novel idea to bridge the gap is *overlay networks*, or just *overlays* for short. The fundamental idea of overlays is rooted in the age-old computing paradigm of *virtualization* that states that if you do not have what you want you can *virtually* create what you want with what you have. This principle is at foundation of a number of computing innovations over the decades. For instance, virtualization is why a computer with finite and fragmented storage can be made to appear to the programmer as if it had a single, contiguous *virtual* memory space. Or, a linux server can be made to emulate a *virtual* machine to provide the abstraction of a Windows desktop to an user. Along the same lines, we can build a *virtual* network (the overlay) over the existing Internet (the underlay) to provide the stringent requirements of modern Internet-based services (cf. Figure 1). Overlays use the functional primitives that the underlay has to offer. In turn, the overlay provides richer functionality to services that are built on top of it.

1.1 Types of overlays

Different overlays offer different enhanced functionalities for online services. As such, there are as many types of overlays as there are service requirements. However, our aim is not to be comprehensive. Rather, we only review the following three types of overlays that are illustrative and important for meeting the needs of Internet-based services:

1. The ubiquitous *caching overlay* that aims to deliver web sites, on-

demand videos, music downloads, software downloads, and other forms of online content. Such overlays are applicable for content that does not change over extended periods of time and is hence *cacheable*. The key benefits that a caching overlay provides are greater availability, performance, origin offload, and scalability (cf. Section 3).

2. The *routing overlay* that provides wide-area communication with more reliability, lesser latency, and greater throughput than the public Internet can. Such overlays could be used to deliver dynamic web content or live stream content that normally cannot be cached (cf. Section 4).

3. The *security overlay* that increases the security and mitigates distributed denial of service (DDoS) attacks on web sites and other online services (cf. Section 5). Such overlays are at the core of some of the most sought after Akamai services, while at the same time they are a classic example of the overlay philosophy for enhancing the underlay by providing new functionality.

As we discuss the architecture and techniques of each type of overlay, it is important to remember that these overlays are often utilized together as a part of a single solution. For instance, an e-commerce application provider would use the caching overlay for the static web content such as images, the routing overlay for the dynamic content such as the html, and in addition use the security overlay to ensure security and business continuity.

2 Background

First, we provide background information on the Internet (the underlay) and what it can or *cannot* provide an overlay designer. In fact, the shortcomings of the Internet architecture are the very reason why overlays are required. Next, we trace the evolution of overlays from both an industry and academic research perspective. Finally, we describe the high-level architecture of the overlays we study in greater detail later. For a more comprehensive treatment of fundamental overlay architecture, the reader is referred to [15].

2.1 Deficiencies of the Internet

The Internet is not a single entity but a network of thousands of autonomous networks. No single network dominates the Internet traffic with the largest controlling less than 5% of the access traffic. This implies that most users accessing a popular website that is centrally hosted must traverse multiple

networks to obtain that content. The networks that make up the Internet are autonomous business entities and often compete with each other. Not surprisingly, the end-to-end communication on the Internet is governed largely by business rules rather than on a notion of maximizing the performance perceived by the user.

We outline the major shortcomings of Internet that make it unsuitable for *directly* supporting the stringent requirements of Internet-based services *without an overlay*. As we will see in the succeeding sections, an appropriately architected overlay can alleviate some of the shortcomings below.

(1) Outages. Partial network outages are common on the Internet caused by misconfigured core routers, DDoS attacks, cable cuts, power disruptions, natural calamities, and de-peering due to a business conflict. For instance, a number of major incidents have occurred recently with an important submarine cable system (the SE-ME-WE-4) leading to outages for millions of users in the Middle East, South, and South East Asia several times in each year between 2008 and 2013 [16]. More recently, a power outage in a Virginia data center took down a number of online services [4]. As another example, Sprint and Cogent decided to de-peer resulting in a partial loss of connectivity for over 3500 networks in 2008 [25].

(2) Congestion. When the capacity of routers and links on the Internet are insufficient to meet the traffic demand, congestion occurs resulting in packet loss. Packet loss manifests itself as performance problems perceived by the user, including slow downloads of web pages and freezing of videos during playback. Peering points where individual networks exchange packets are particularly susceptible to packet loss. Part of the reason for the peering bottleneck is economics. Networks are incentivized to provision surplus capacity on the "first mile" that serves their hosting customers and the "last mile" that serves their paying end-users. However, there is no significant economic incentive to exchange traffic with other potentially competing networks over peering points in the "middle mile". Shared mediums such as wireless links are also particularly susceptible to packet loss.

(3) Lack of scalability. Online services require provisioning server and network resources to meet the demand of users at all times, even during unexpected periods of peak demand and flash crowds. Without the existence of overlays, an enterprise may deploy their online services in a centralized fashion within a single data center and expect to serve their users from that centralized origin infrastructure. However, such a centralized solution falls significantly short of meeting the requirements of mission-critical services. The data center itself is a single point of failure where outages or congestion can adversely impact the availability and performance of the service.

Further, the enterprise would have to provision resources for peak demand that is significantly higher than the average. Such over-provisioning for peak means wasting money on infrastructure that is seldom used. On the other hand, under-provisioning to save cost has dire consequences as well, as the service may be unavailable at critical moments of high demand, leading to loss of revenue. To some degree, one can alleviate some of these issues by mirroring the origin in multiple data centers or by multihoming on multiple networks [3]. However, these solutions increase the cost and complexity of the origin infrastructure and by themselves are unlikely to provide the stringent requirements of modern online services. Therefore, an overlay solution that allows for massive replication and automatic scaling is required to meet the user demand that could vary significantly over time.

(4) Slow adaptability. Online services and their requirements evolve rapidly. However, the fundamental architecture and protocols of the Internet are slow to change or accommodate new primitives. For instance, the rollout of a new version of Internet Protocol (IP) called IPv6 was first proposed in 1998, but has just started to gain some adoption about fifteen years later. IPv6 traffic currently accounts for around 1% of the Internet traffic [10]. The large investment by networks and enterprises in the current Internet technology is often a barrier to change. The complexity in business relations between networks is also a stumbling block. In some cases, the highly decentralized architecture of the Internet means that a large number of autonomous networks must all adopt the change to reap its full benefits, which rarely happens. However, unlike the Internet architecture, an overlay is often owned and operated by a single centralized entity and can be deployed rapidly with no changes to the underlying Internet. Thus, overlays are an attractive alternative for adapting to the fast changing requirements of online services in a way that the Internet underlay cannot.

(5) Lack of security. Modern online services require protection from catastrophic events such as distributed denial of service (DDoS) attacks. A recent report by Arbor Networks found that the average size of a DDoS was 1.77 Gbps in the first quarter of 2013, an increase of almost 20% from a year ago [12]. Aside from the size, the number of DDoS attacks are also growing year after year. Each attack could cause a significant amount of revenue loss if the online business, such as an e-commerce site, is unable withstand the attack and becomes unavailable to users. The Internet architecture provides no protection against DDoS attack modes such as syn floods and packet floods. For each individual business to provision for additional server and bandwidth capacity to tackle the possibility of a DDoS attack can be prohibitively expensive, making the overlay approach to defend against such

attacks a more attractive option.

2.2 A Brief History of Overlay Networks

The idea of building one network (overlay) over another one (underlay) is at least a few decades old. In fact, the early Internet was itself initially built as an overlay on top of the telephone network that was the predominant network of the day. Much of the early overlays were build to provide functionality that the underlay natively lacked. A classic example is Mbone [26] that can be viewed as an overlay on the Internet providing multicast functionality. Further, the Internet was not the only underlay studied in the context of overlays. In fact, the concept of overlays found simultaneous and in some cases earlier development in the domain of interconnection networks for large parallel computers[1]. For instance, early work from the late 1980's [27] showed how to effectively emulate one type of network (say, a 2-D mesh) as an overlay on a different type of underlay network (say, a butterfly).

Besides creating new functionality, the potential for improving reliability and performance by building a virtual overlay network also has a long history. In the domain of parallel networks, work from the early 1990's showed that it is theoretically possible to build an overlay that provides an abstraction of a failure-free network over a failure-prone underlay network of the same size and type without significant loss in performance [9, 23, 28]. Subsequently, in the late 1990's, overlays to enhance the availability, performance, and scalability of the Internet came to prominence both within industry and academia, even as the Internet became crucially important for business, commerce, and entertainment. Seminal research advances were made in academia with systems such as RON [5] that showed that routing overlays can effectively use the path diversity of the Internet to improve both end-to-end availability and performance. Among the earliest to appear in industry were the caching overlay and the routing overlay. By the end of the 1990's, companies such as Akamai had full fledged offerings of caching overlays for delivering web and on-demand videos [11] and routing overlays that relied on the path diversity of the Internet to provide higher quality live streaming [6, 22].

Another major development over the past decade is the emergence of peer-to-peer (P2P) overlays that use (non-dedicated) computers of the end-users themselves to form overlays that can be used for downloading content. Early systems that used P2P principles include the now-defunct services

[1]Rather than use the terms overlay and underlay, the terms guest and host network respectively were used in the parallel network literature.

such as KaZaa [13] and Gnutella [14], and innovative experimental systems such as Chord [17], Content Addressable Networks [19], Tapestry [20], and Pastry [21]. While pure P2P systems have had less adoption among enterprise customers who demand higher levels of availability, performance, and content control than such systems typically offer, hybrid approaches that combine P2P principles with a dedicated overlay infrastructure are widely used in services such as Akamai's client-side delivery [1]. While P2P overlays are an important type of overlay, we do not describe them in greater detail here. Rather all the three types of overlays described here use a dedicated server infrastructure owned and operated by the overlay provider, rather than computers belonging to users.

2.3 Overlay Architecture

The overlays that we study in detail are used to deliver content, applications, and services to users on behalf of content providers[2]. Content providers include news channels, e-commerce sites, social networks, download services, web portals, banks, credit-card companies, and authentication services. The end-users access content, applications, and services from across the globe from a variety of devices including cell phones, tablets, desktops, using a variety of client software including browsers, media players, and download managers.

An overlay capable of delivering content, applications, and services to a global audience is a large distributed system consisting of hundreds of thousands of globally deployed servers that run sophisticated algorithms. Independent of the specific type of overlay, they share a similar system-level architecture as shown in Figure 2. However, the detailed design and implementation of each system component differs depending on the precise functionality that the overlay provides. The content, application or service is hosted by the content provider in one or at most a few *origin* locations on the Internet. Users who access the content, application, or service interact directly with a wide deployment of *edge servers* of the overlay, rather than directly with the origin. For instance, a large-scale Akamai overlay consists of over hundred thousand edge servers that are physically located in over 80 countries and in over 1150 networks around the world. Unlike the origin that is located in the core of the Internet, the edge servers are located at the "edges" of the Internet. The edge servers are deployed very close to users in

[2]For simplicity, we use the term content provider to denote traditional content owners such as web sites, application providers such as a software-as-a-service provider, and service providers such as an authentication service provider.

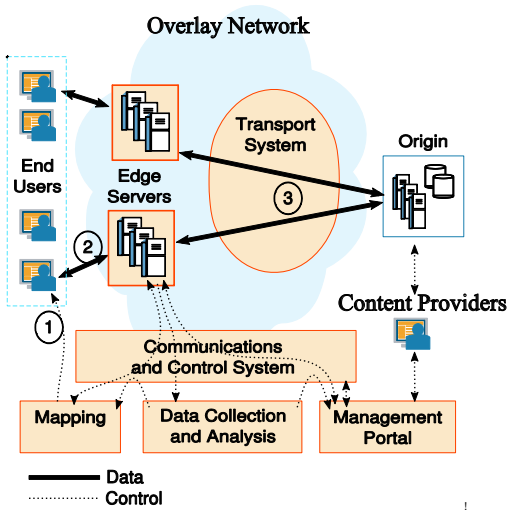


Figure 2: The system-level architecture of an overlay network [15].

a network sense so as to provide low latencies, low packet loss, and higher bandwidth, resulting in better performance. To understand how the different systems of an overlay interact to deliver content to the user, it is instructive to consider the simple example of a user entering a URL into his/her browser and receiving a web page through the overlay. The important control or data flow at each step is shown with arrows in Figure 2.

- The domain name of the URL is translated by the *mapping system* into the IP address of an edge-server that can serve the content (arrow 1). There are a number of techniques for implementing the domain translation. A simple but rudimentary mechanism is to assign the same IP address to all the edge servers, and rely on network-level anycast for determining the "right" edge server for a given end-user. A more robust and scalable mechanism is to use the domain name system (DNS) for the translation and is used in Akamai's overlays. The mapping system bases its answers on large amounts of historical and current data that have been collected and processed regarding the global Internet and server conditions. This data is used to choose an edge server that is located "close" to the end user, so as to maximize performance.
- The browser sends the request to the chosen edge server that is now

responsible for serving the requested content (arrow 2). The edge server may be able to serve the requested content from its local cache. Or it may need communicate with an *origin* server to first obtain the content that is then placed in its local cache and served to the user.

- The *transport system* is used to transfer the required content between the origin and the edge server (arrow 3). The transport system is at the heart of the overlay, which moves content over the long-haul Internet with high reliability and performance.
- The *origin* is typically operated by the content provider and consists of web servers, application servers, databases and other backend infrastructure that serve as the source of the online content, application, or service. In the case of live streaming, the origin also consists of encoders and media servers that originate the stream and transmit it to the overlay network via "entry points" [22].

The different types of overlays that we study implement each of these systems differently in accordance with the differing requirements of the overlay. We will delve into those differences in the succeeding sections.

3 Caching Overlays

The caching overlay is used for content that can be cached for some period time. Canonical examples include static objects such as an embedded image on a web page, a movie, a music file, a software download, or a virus update. Dynamic objects that do not change very often, such as a weather map that is updated once an hour, can also benefit from a caching overlay. The benefits of the overlay include availability, performance, and origin offload, each of which we address in turn.

3.1 Architecture

The high-level architecture of the caching overlay can be described with reference to Figure 2. The mapping system directs each user to the closest server in a network sense, identified by using recent knowledge of the network latency and packet loss rate along the path between the user and the edge servers. The edge servers provide the functionality of caching HTTP/HTTPS proxy servers. If the content requested by the user is found in local cache, the edge server serves it to the user. If the content is not found in its own local cache or in any other edge server in its cluster, then the edge server uses the transport system to download the content from

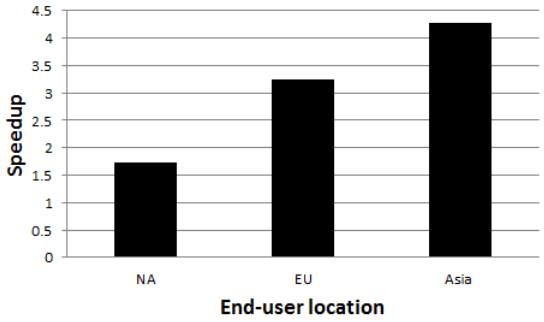


Figure 3: Performance benefits of a caching overlay

the origin. The transport system may itself have another layer of servers called *parent servers* that are capable of caching content. In this case, the edge server requests a parent server for the content. If the parent server has the content, it is served to the edge server. Otherwise, the parent server requests the content from origin, caches the content itself, and then forward the content to the edge server. In some cases, one could have multiple layers of parent caches that are requested before the request is forwarded to origin. A transport system with one or more layers of parent servers is called a *cache hierarchy*.

3.2 Performance benefits

It is easy to see why the caching overlay improves performance of cacheable web content. The *edge hit rate* is defined to be the probability of finding the requested content in the edge server without having to download it from a parent server or the origin. If the edge hit rate is high, instead of traversing a major part of the Internet to get content from the origin servers, users fetch their required content from a nearby edge server that is reachable on a path that is known to have low latency and low packet loss. For a typical popular website, the edge hit rate can be very high at 90+%, resulting in almost all user requests being served from an edge server's cache leading to significant performance improvements.

To better quantify the performance benefits for cacheable content, we used a *performance testing platform* that uses a large collection of "agents" installed on end-user desktop machines located around the world. These agents are capable of performing periodic downloads of web pages and reporting fine-grain performance measurements about each download. For our

experiment, we used 30 agents located in Asia, Europe, and North America, each with broadband connectivity to the Internet. The agents did hourly downloads of two versions of a cacheable 32 KB file. The first version of the file was delivered directly from our origin servers in Dallas where we hosted the file. Thus, this version did not use the caching overlay. The second version of the file used an Akamai caching overlay to deliver the file.

A natural measure of the performance benefits of the overlay is *speedup* that is defined to be ratio of the time to download the file directly from the origin to the time to download the same file using the overlay. In Figure 3, we show the speedup aggregated by the continent where the user (i.e, agent) was located. The caching overlay provides large speedups between 1.7 to 4.3 in all continents. Further, users further away from the Dallas origin experience larger speedups. The reason is that the download time using the overlay remains roughly the same independent of geography, since the caching overlay can find an edge server close to the user in all geographies. However, the download time from the origin deteriorates as users move further away from it.

3.3 Origin offload benefits

An important benefit of using a caching overlay is the decrease in traffic served by the origin and is measured by a metric called *origin offload* that equals the ratio of the volume of traffic served by the origin without the overlay to the volume of traffic served by the origin with the overlay. When an overlay is used, only the traffic due to cache misses at the edge servers is served by origin. If cache hit rates are high, as they are with popular web content, the origin offload could be anywhere from 10 to 100. Note that origin offload is very beneficial to the content provider as they only have to provision their origin to serve a small fraction of the traffic that they would have had to serve without the overlay, resulting in a large decrease in server, bandwidth, co-location, and operational expenses. However, not all web content is popular and most web sites have a "long tail" of less popular content. Take for instance an e-commerce site. While a few product pages may be "hot" and will yield a high edge hit rate and a high origin offload, most of the other product pages may at best be "warm" or even "cold", yielding much smaller edge hit rates and much less origin offload. Providing better origin offload for "warm" and "'cold" traffic requires more sophisticated architectures that we describe below.

3.3.1 Cache hierarchy

The basic idea in a cache hierarchy is to add a layer of *parent servers* that serve the requests that experience cache misses at the edge servers. If an edge server fails to find the requested content in its cache, it forwards the request to a chosen parent server. If the parent server has the content in its cache, it serves it to the edge server. Otherwise, the parent obtains the content from origin. Adding a layer of parent servers decreases the traffic to the origin, increasing the origin offload. Typically the requests that are cache misses at the edge are "funneled" to a smaller number of parent servers, so as to increase the hit rate at the parent, thereby increasing the origin offload.

Cache hierarchy is relatively easy to implement as a parent server uses similar software components as an edge server for caching and serving the content. Further, the mapping system can be used to find a parent server that is "proximal" to the edge server. Finding a parent close to the edge server decreases the parent-to-edge download and improves performance.

To illustrate the origin offload benefit, Figure 4 shows the traffic and hit rates over a two month period for a content provider with a mixture of popular and less popular pages. The content provider uses a cache hierarchy and has an edge hit rate in excess of 85%. The requests that "miss" at the edge are for less popular content and the probability that such a request can be served from the parent's cache, known as the *parent hit rate*, is over 70% hit rate. Cumulatively, the *overlay hit rate* which is the probability that a request is served from a parent or edge server of the overlay is 96%, i.e., only 4% of the total requests are served from origin. Note that without the cache hierarchy about 15% of the requests would have been served by origin.

In Figure 4, we plot the *edge traffic* that is served to user, the *midgress traffic* that is served by parents to edges, and the *origin traffic* that is served by the origin to the parents. Observe that without the cache hierarchy, all midgress traffic would be served from the origin. Thus, the origin offload without the cache hierarchy is simply the ratio of the edge traffic to the midgress traffic is only 6.7. While the origin offload with the cache hierarchy is the ratio of the edge traffic to the origin traffic that is much larger at about 24.3. Thus, a cache hierarchy significantly increases the offload factor.

Note that while we have described a cache hierarchy with a single layer of parent servers, one can easily extend the hierarchy to consist of multiple levels of parents where a lower-level parent fetches from a higher-level parent, and the parent at the highest level fetches from origin. While multiple levels of cache hierarchy can further increase the origin offload, it can be detrimental to performance for the fraction of requests that get forwarded

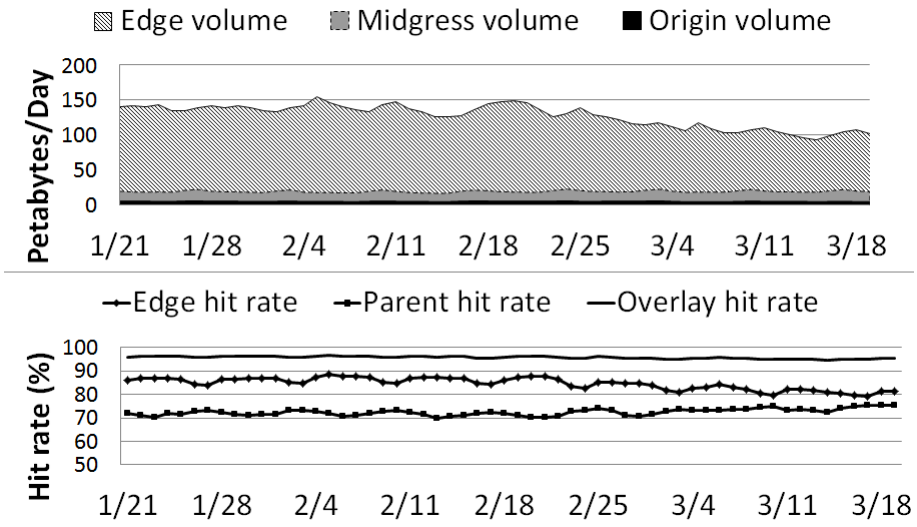


Figure 4: Traffic volumes and cache hit rates for a content provider with a mix of popular and less popular content.

to multiple parents before a response.

3.3.2 Dealing with unpopular content

In recent years, with social-networking sites that carry user-generated content such as photos, and videos, the amount of unpopular "cold" content on the Internet has exploded. For such types of content, providing origin offload can be challenging since there is a large footprint of objects that are each accessed only a few times, requiring an "offload oriented" overlay to provide higher overlay hit rates. While popular content can be served by making several copies of content and placing them close to the end-users, to achieve high origin offload for unpopular content we must make a very small number of copies of the content, thus using the available cache space to cache the most number of unique objects within the overlay.

Since any given piece of content is available at fewer locations in the overlay, a content location service is needed as part of the caching overlay. It can be implemented as a distributed hash table or with a more advanced directory service. The content placement and location techniques can be designed to automatically adapt to content popularity so that several copies are made of the popular content for enhancing performance, while the number of copies of unpopular content is kept small to conserve storage space.

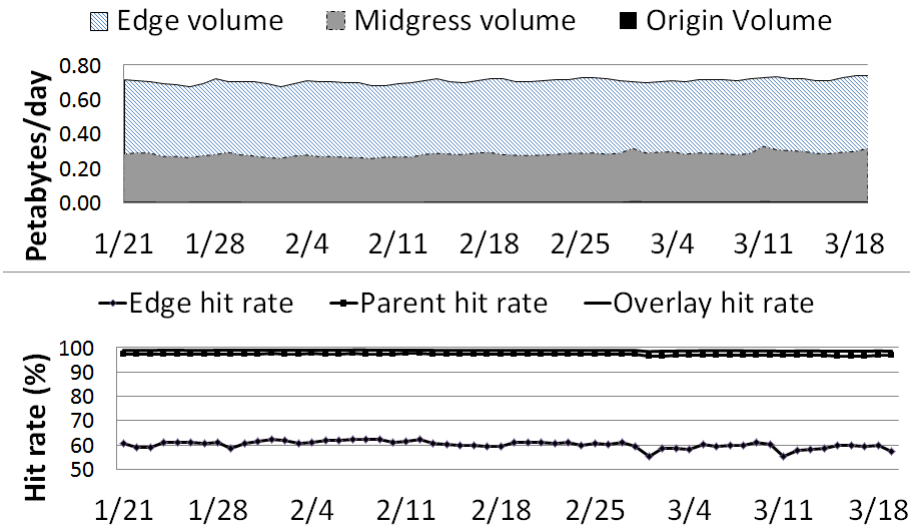


Figure 5: Traffic volumes and cache-hit rates for a social networking site with predominantly unpopular content. The overlay uses more sophisticated content location techniques to increase the overlay hit rate and origin offload.

Figure 5 shows the hit rates and traffic for unpopular social networking content using more sophisticated content location techniques. It can be seen that the content has relatively poor edge hit rate of 60.2% in comparison with the more popular content depicted in Figure 4. However, the content location techniques provide an extremely high overlay hit rate of 98.9%, i.e., only 1.1% of the total request is served by origin. The origin offload is only 2.5 without the content location techniques, but increases to 89 with it.

4 Routing Overlays

Not all content on the Internet is cacheable for long periods of time. For Internet-based applications such as shopping, banking, and gaming, the downloaded content is dynamic and uncacheable in the sense of being generated based on the user's interaction with the application in real-time. Such requests and responses therefore must traverse the Internet between the user and the origin. Another key example of dynamic content are live streams that cannot be cached and must be routed in real-time from the origin that is the source of the stream to the users who are watching it. Despite the

fact that caching cannot be used, a routing overlay improves performance and availability by discovering better "overlay paths" from the origin to the user.

4.1 Architecture

The architecture of a routing overlay is shown in Figure 6. An *overlay construction algorithm* is used to compute a set of *overlay paths* that each edge server can use to reach each of the origins. The overlay construction algorithm takes as input real-time latency, loss and available bandwidth of the Internet paths connecting all the overlay nodes, and determines a ranked set of alternate overlay paths for requests and responses that travel between each edge server and each origin. The overlay paths are chosen to provide multiple high-performance paths for origin-to-edge communication and is frequently updated to account for real-time changes in the Internet. Once the overlay is constructed, each edge server receives a real-time feed of the set of overlay paths that it could use to connect to each relevant origin in the world.

The overlay paths are used for routing communication as follows. When a request arrives at an edge server, it deduces which origin serves the requested content, and forwards the request along a chosen overlay path(s) to that origin. Each intermediate overlay node acts as a forwarder of the request to the next node in the path towards the origin. The response from the origin traverses the same path in the opposite direction to arrive at the edge server, which is then forwarded to the end-user. Note that the overlay construction process provides multiple alternate paths for each edge server and each origin. In addition, the edge server always has the option of the using the *direct path* from the edge to origin that passes through no other intermediate node. The choice of which of these alternate paths to use can depend on real-time testing of the different path options. In some overlays, each edge-to-origin communication uses a single chosen path. However, other routing overlays use multiple overlay paths simultaneously. In this case, the content can be encoded and sent across multiple paths that is then decoded by the edge server and sent to the end-user. Using multiple paths is useful if one of the chosen overlay paths experiences transient packet loss. The data lost in one path can be recovered from packets received on the other path. A more detailed account of multi-path techniques for loss recovery in the context of live streaming can be found in [22].

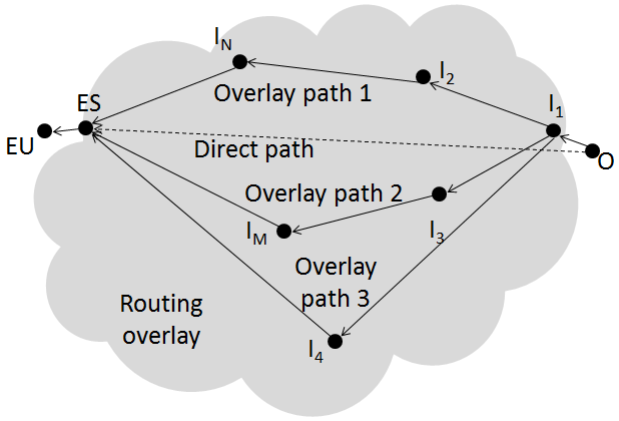


Figure 6: An end-user (EU) downloads dynamic content from an edge server (ES). The edge server in turn can download directly from origin (O) or through a set of alternate overlay paths constructed between the edge server (ES) and a reverse proxy (I_1).

4.1.1 Formulating overlay construction as multi-commodity flow

The overlay construction algorithm constructs overlay paths between each origin (O) and each edge server (ES) by solving a multi-commodity flow problem [8] on an appropriately defined flow network. The nodes of the flow network are the origins, intermediate nodes, and edge servers. The edges of the flow network are defined by fully connecting the intermediate nodes with each other, connecting each origin to all intermediate nodes, and also connecting each edge server to all intermediate nodes. The traffic from each origin O to each edge server ES is represented as a distinct commodity $\langle O, ES \rangle$ that must be routed on the flow network. The demand of each commodity $\langle O, ES \rangle$ is an estimate of the traffic that needs to be sent from origin O to edge server ES. Demand needs to be measured over short time windows, e.g. 10 seconds, since demand (i.e. traffic) can change rapidly over time. The capacity of each link in the flow network is an estimate of the maximum traffic that can be sent across that link. Finally, the cost of each link of the flow network can be modeled so that "good" links have lower cost, where goodness can be defined using a function that captures the objective of the routing overlay. Solving the multi-commodity flow problem on the appropriately defined flow network yields a set of low cost paths for

each commodity that correspond to the required overlay paths.

The power of the multi-commodity flow formulation is the ability to define link costs in different ways to construct different types of routing overlays. For instance, a routing overlay for live streaming may define link costs in a different manner than a routing overlay for dynamic web content, resulting in different overlay networks being constructed. We list a few ways of defining the link costs below.

(1) Latency versus bandwidth price. If path latency is considered to be the link cost, then the solution finds the fastest overlay paths from O to ES. If bandwidth price is considered to be the link cost, then the solution finds the cheapest overlay paths from O to ES. Combining the two types of costs could allow one to find different solutions, e.g. finding the fastest overlay routes while avoiding links that are too expensive, or finding the cheapest overlay paths while avoiding paths that are too slow.

(2) Throughput. In the context of web content delivery, two different notions of performance apply. Minimizing latency is important when delivering small-size responses. Maximizing throughput is important for large responses. These two are closely related since steady-state TCP throughput is inversely proportional to the round trip time (RTT) and the square root of the packet loss rate. If the loss rate is close enough to zero, it is possible to sustain high throughput over a long connection if TCP buffers are sufficiently large. In general, overlays with different properties can be obtained by weighting path metrics such as latency and loss differently in the derivation of the link costs.

(3) TCP performance. A vast majority of Internet traffic is served over the connection-oriented transport protocol TCP. Establishing a new TCP connection penalizes performance in two ways. Establishing a new connection requires additional round trips, adding to the latency. Further, new connections have smaller congestion windows which impacts the number of packets you can have "in flight". An overlay routing algorithm that maximizes performance after accounting for TCP penalties must also attempt to reuse existing TCP connections rather than start new ones. Thus, the overlay paths must remain "sticky" over longer periods of time and must change only when the cost parameters have changed sufficiently to overcome the potential penalties for a new TCP connection.

4.1.2 Selecting the reverse proxy

An important aspect of overlay construction is selecting the reverse proxy (node I_1 of Figure 6) for each origin. The overlay construction algorithm

typically chooses a reverse proxy close to or even co-located with the origin. It is important that the "first hop" from the origin to the reverse proxy has the smallest possible latency and loss, since the first hop is shared by all overlay paths from the origin to the edge server. Further, while the overlay provider can ensure persistent TCP connections between any two nodes of the overlay, the first hop is partially dependent on the origin that is controlled by the content provider. For this reason, the first hop is more likely to lack persistent TCP connections and is more likely to incur TCP penalties. Making the first hop as low latency as possible reduces the potential penalty for establishing a new TCP connection. Similarly, while TCP protocol optimizations that speedup downloads are implemented between any two overlay nodes, such are not guaranteed to be available on the first hop. For these reasons, choosing a reverse proxy close the origin is often desirable.

4.1.3 Fast algorithms for overlay construction

Computing the overlay paths efficiently requires heuristics since the multi-commodity flow problem as formulated in Section 4.1.1 is in general NP-hard. However, the overlays need to be kept updated in real-time responding to the ever-varying latency and loss conditions. Thus, using a scalable algorithm for solving the overlay problem is an important requirement. One approach is to use an optimal lagrangian relaxation scheme that routes flow through the lowest cost paths as determined by a modified Floyd-Warshall All-Pairs-Shortest-Path (APSP) algorithm. APSP itself can be modified to take advantage of the structure of the flow network that has a nearly fully connected "middle" with origins and edge servers attached to the middle.

Besides lagrangian relaxation, a different approach is to write a mixed integer program (MIP) whose constraints can be "relaxed" to create linear program. The fractional solution for the linear program can then rounded using generalized assignment problem (GAP) rounding techniques to produce an integral solution that corresponds to choosing the overlay paths. This approach yields provable guarantees for the approximate solution and is described in the context of constructing routing overlays for live streaming in [6, 7].

4.1.4 Performance benefits

To illustrate the benefits, we measured the performance of a routing overlay during a recent large-scale Internet outage when a submarine communications cable system (called SEA-ME-WE 4) that links Europe with the

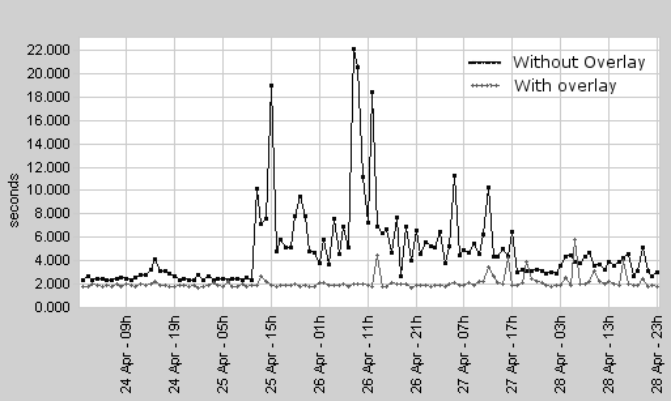
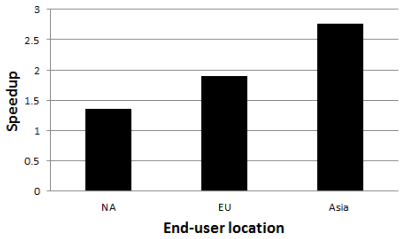


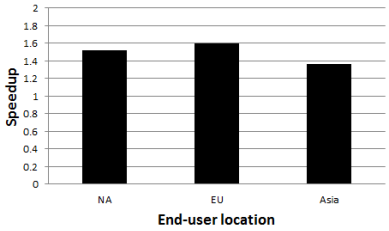
Figure 7: Performance of the routing overlay during a cable cut.

Middle East and South Asia was cut on Wednesday, 14th April 2010 in the Mediterranean sea and became inoperable due to a shunt fault approximately 1,886 kilometers from Alexandria towards Palermo, Italy in the S4 section of the cable. It underwent further repairs from 25 April to 29 April to fix the affected fibre pair in the Mediterranean sea. The repair work affected several cable systems, severely impacting Internet connectivity in many regions across the Middle East, Africa and Asia.

Figure 7 shows the download time experienced by end-users in Asia using a routing overlay during the outage, in comparison with the download time experienced by the same end-users without the overlay. For this experiment, we used a dynamic (i.e., uncacheable) web page approximately 70KB in size, including all page objects. The web page was measured using agents located in India, Malaysia and Singapore. The agents downloaded two versions of the web page - one directly from the origin in Boston, and another through the routing overlay. It can be seen that during the SE-ME-WE 4 disruption, the performance of the download directly from the Boston origin to Asia suffered severe slowdowns. However, downloads for the same web page from the same Boston origin to the same Asian end-users using the routing overlay experienced minimal performance degradation. The significantly greater performance is due to the ability of the routing overlay to find alternate paths that avoid the failed links between different parts of Asia to the Boston origin. However, without the benefit of the routing overlay, the direct Internet path that exhibits significant degradation due to the cable cut must be used for the downloads. The routing overlay speeded up the downloads for Asian end-users by over 8 times at the peak of the outage.



(a) A large-sized overlay using a single optimized path.



(b) A medium-sized overlay using multiple optimized paths and error correcting codes for loss recovery.

Figure 8: A routing overlay provides significant speedups by choosing better performing paths from the origin to the end-user.

Even when there is no major Internet outage, the routing overlay provides a significant performance benefit by discovering and using better performing overlay paths for communication. In Figure 8(a), we show the performance benefit of a large routing overlay that uses a single optimized path chosen from a set of alternate overlay paths to the origin. For the experiment, we stored a dynamic (uncacheable) file of size 38KB in an origin in Dallas and tested it from agents around the world. Speedup is the ratio of the download time of the file downloaded directly from the origin to the download time of the same file downloaded using the routing overlay. Speedup was significant for all geographies, though the speedups increases as the end-user moves further away from the origin. The performance benefits seen are due to a combination of the routing overlay finding a path with shorter latency between the origin and the edge server, the use of an optimized TCP between the overlay nodes, and a reduction in TCP penalties. The net effects of these benefits are expected to be higher on longer haul paths, resulting in greater speedups for users further away from the origin.

In Figure 8(b), we show the performance benefits for a different smaller routing overlay that uses multiple optimized paths for each communication and uses network coding across the paths for loss recovery. For the experiment, we stored a complete dynamic (uncacheable) webpage 88 KB in size in an origin server located in Japan. The page was tested by agents around the world. The speedups from all geographies are significant with the continent of origin (Asia) having the smallest speedup. However, the speedup of this routing overlay is generally smaller than that of Figure 8(a) largely due to that fact that smaller overlays have fewer choices for alternate paths

and the edge servers are less proximal to the user.

5 Security overlays

An Internet-based service needs to defend itself from distributed denial of service (DDoS) attacks that aim to take it out of commission and hackers who want to steal sensitive user information from the service. The Internet architecture by itself does not provide security, and it is desirable that security be automatically provided by the platform on which Internet services are offered. There are good reasons why security should be dealt with in the underlying platform and not as part of every individual website or service. Defending against DDoS and hacker attacks requires a vast amount of spare capacity and up-to-date expertise in security vulnerabilities. If a website experiences DDoS attacks few days each year, then maintaining the excess capacity all year round is a costly proposition. Short-duration DoS attacks happen frequently against some website or the other and maintaining excess capacity and security expertise in a shared platform is more cost-effective than individual measures taken by the owner of every Internet-based service.

For web sites and services that already use the caching and/or routing overlays, most Internet traffic is already being handled by the servers of the overlay. If the security policy needs user requests to be examined, performing that task at the first edge server to receive the request is more efficient from a performance and workload perspective, than having to forward it someplace else. Thus, for Internet-based services that already use a caching and routing overlay, a security overlay is a synergistic addition.

5.1 Architecture

The design of a security overlay is not just about having spare capacity and security features, but also about how best to use the capacity and the features to protect against a security attack at short notice. A security overlay incorporates several architectural elements described below.

(1) Shared spare capacity: The overlay has a very large number of servers, each with high-capacity network connections. The operating model of the overlay is flexible enough to divert any fraction of traffic to any server and increase the network bandwidth capacity at some locations on-demand as needed. The spare capacity that can be made available should be large enough that it is unlikely to be overwhelmed even by the largest DDoS attacks that are likely to occur. Thus the spare capacity can be used to hold a high-volume attack away from the origin servers. Having the excess

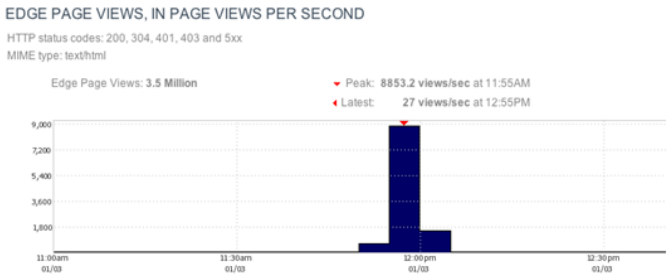
capacity shared across many content providers is a cost effective way of reserving a large pool of resources as attack capacity.

(2) Shared expertise and lower costs: Hackers often exploit new and known vulnerabilities in operating systems and applications. In the scenario without a shared security overlay, every Internet-based service has to look after its own security by keeping current with new and known vulnerabilities and individually updating their security measures. However, with a security overlay, a team of security experts of the overlay staff keeps the overlay substantially free of known and new vulnerabilities, thus providing a high level of defense to small and large Internet-based services alike. Due to the shared nature, the costs are lower and more affordable for content providers.

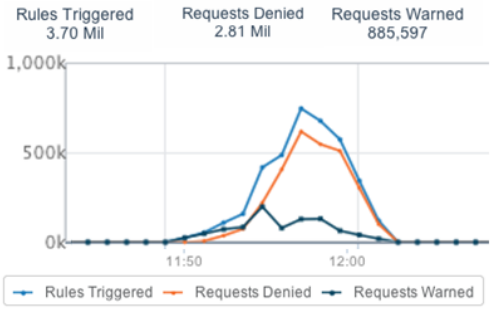
(3) Advanced security features: The overlay is designed to offer detection and defense against a whole spectrum of attacks, such as SYN and connection flooding attacks, brute force DoS attacks that generate hundreds of Gbps of traffic, malicious request injection attempts, and attempts to hack into servers. At the network layer, the overlay servers have a hardened networking stack that deals with low-level attacks and hacking attempts efficiently, and can mitigate simple attacks from turning into Gbps of traffic by denying the attackers requests. The overlay also incorporates a web-application firewall to examine request-response sequences to detect and filter harmful exchanges. With the help of the firewall, several advanced mitigations could be deployed to counter the attacks. For instance, if a certain URL is being used by hackers from a certain part of the world, then a firewall rule could be constructed to block that URL when requested from that part of the world. Such an action could be used to thwart the attackers, while leaving the site open for legitimate users.

(4) Shielding the origin: The overlay can offer another simple yet powerful feature for origin protection against hackers. The origin of an Internet-based service is more vulnerable to hacker attacks if its ip addresses are externally known to users. However, when using a security overlay, the origin can be configured so that it receives only traffic known to originate from the servers of the overlay that use a small range of ip addresses. This allows the origin's administrator to entirely firewall-off the origin from users not in the specified ip address range, thus easily identifying and blocking any traffic that arrives at the origin from outside the secure overlay. Additional protective measures such as authenticating all origin-to-overlay communication using a shared secret can further thwart the attackers.

(5) Control design: Even though the security overlay is a shared infrastructure, it should have features that allow an individual content provider to maneuver in response to an ongoing attack. In particular, controls must



(a) Traffic spike due to the DDoS attack.



(b) Rules are triggered at the edge servers to filter out the attack traffic.

Figure 9: The security overlay filters out DDoS traffic at the edge before it reaches the origin.

be provided for individual content providers to change the security configurations, firewall rules, filters, etc. as pertaining to their own site at any time and at short notice after an attack is detected.

5.2 Security benefits

We present some statistics collected during a recent DDoS attack on a content provider who uses a security overlay. Figure 9(a) shows a sudden increase in the traffic of the content provider's website due to the DDoS attack. The web site that is normally accessed at a rate less than 50 pages/second was accessed at a much higher rate of 9000 pages/sec during the attack. Figure 9(b) shows firewall rules getting triggered in response to the attack and denying over 90% of the attackers' requests, and protecting the origin from the significant surge of traffic.

6 Conclusion

In this chapter, we reviewed the rationale for building overlays as a means for providing availability, performance, scalability, and security for Internet-based services. While overlays are a powerful technique for building a variety of new functionality on top of the Internet, we focused three key types of overlays. The caching and routing overlays are perhaps the most ubiquitous and form a critical part of the Internet infrastructure. The security overlay is a novel and emerging tool to defend against DDoS attacks and other security threats that are rapidly becoming more sizable and more frequent. The future requirements of Internet-based services are hard to predict, even as modern trends like social networking were largely unforeseen during the turn of the century. The promise that overlays hold for the future is their ability to bridge the gap between what the vanilla Internet offers and what future Internet services may need. From that perspective, overlays hold the keys to the rapid evolution of Internet services, even as the underlying Internet architecture is slow to change.

References

[1] Akamai NetSession Interface (Client Side Delivery) Overview `http://www.akamai.com/client/`

[2] M. Adler, R.K. Sitaraman, and H. Venkataramani. Algorithms for optimizing the bandwidth cost of content delivery. *Computer Networks*, 55(18): 4007-4020, 2011.

[3] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman. A Measurement-Based Analysis of Multihoming. Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM), August 2003.

[4] *Amazon Web Services suffers partial outage.* `http://www.zdnet.com/blog/btl/amazon-web-services-suffers-partial-outage/79981`.

[5] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, Robert Morris. "Resilient Overlay Networks," Proc. 18th ACM SOSP, Banff, Canada, October 2001.

[6] K. Andreev, B. M. Maggs, A. Meyerson, and R. Sitaraman. Designing Overlay Multicast Networks for Streaming. Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), June 2003.

[7] Andreev, Konstantin, Bruce M. Maggs, Adam Meyerson, Jevan Saks, and Ramesh K. Sitaraman. Algorithms for constructing overlay networks for live streaming. arXiv preprint arXiv:1109.4114, 2011.

[8] Assad, Arjang A. Multicommodity network flow – a survey. Networks 8.1: 37-91, 1978.

[9] R. J. Cole, B. M. Maggs, and R. K. Sitaraman. Multi-scale self-simulation:A technique for reconfiguring processor arrays with faults. Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC), pp. 561–572, May 1993

[10] Dhamdhere, Amogh, Matthew Luckie, Bradley Huffaker, Kc Claffy, Ahmed Elmokashfi, and Emile Aben. Measuring the Deployment of IPv6: Topology, Routing and Performance. ACM Internet Measurement Conference (IMC), November 2012.

[11] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally distributed content delivery. IEEE Internet Computing, September 2002, pp. 50–58.

[12] eSecurity Planet. DDoS Attacks: Growing, but How Much?. `http://goo.gl/HhFkt`.

[13] KaZaa. `http://en.wikipedia.org/wiki/Kazaa`.

[14] Gnutella. `http://en.wikipedia.org/wiki/Gnutella`.

[15] E. Nygren, R.K. Sitaraman, and J. Sun. The Akamai Network: A platform for high-performance Internet applications. *ACM SIGOPS Operating Systems Review*, 44(3):2–19, 2010.

[16] SE-ME-WE 4 disruptions. `http://goo.gl/kW3bE`.

[17] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Computer Communication Review, Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Volume 31 Issue 4, August 2001.

[18] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, Scott Shenker. Making gnutella-like P2P systems scalable. Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, August 2003

[19] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Schenker. A scalable content-addressable network. ACM SIGCOMM Computer Communication Review, Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Volume 31 Issue 4, August 2001.

[20] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. Tapestry: A Resilient Global-scale Overlay for Service Deployment. IEEE Journal on Selected Areas in Communications, January 2004, Vol. 22, No. 1.

[21] A. Rowstron and P. Druschel. "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.

[22] L. Kontothanassis, R. Sitaraman, J. Wein, D. Hong, R. Kleinberg, B. Mancuso, D. Shaw, and, D. Stodolsky. "A Transport Layer for Live Streaming in a Content Delivery Network". Proceedings of the IEEE, Special issue on evolution of internet technologies, pages 1408–1419, Volume 92, Issue 9, August 2004.

[23] F.T. Leighton, B.M. Maggs, and R. Sitaraman. "On the fault tolerance of some popular bounded-degree networks," Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS), pp. 542–552, October 1992..

[24] *Submarine cable disruptions of 2008.* `http://en.wikipedia.org/wiki/2008_submarine_cable_disruption`.

[25] Renesys Blog. Wrestling With the Zombie: Sprint Depeers Cogent, Internet Partitioned. `http://www.renesys.com/blog/2008/10/wrestling-with-the-zombie-spri.shtml`

[26] Eriksson, Hans. "The multicast backbone." Communications of the ACM pp. 54-60, 1994.

[27] Koch, Richard R., F. Thomson Leighton, Bruce M. Maggs, Satish B. Rao, Arnold L. Rosenberg, and Eric J. Schwabe. "Work-preserving emulations of fixed-connection networks." ACM Symposium on Theory of Computing (STOC), 1989.

[28] Ramesh K. Sitaraman. "Communication and Fault Tolerance in Parallel Computers", Ph.D Thesis, Princeton University, 1993.

[29] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker and J. Zahorjan. Detour: A Case for Informed Internet Routing and Transport. IEEE Micro, Vol. 19, No. 1, January/February 1999.