University of Massachusetts Amherst

From the SelectedWorks of Ramesh Sitaraman

July, 2010

The Akamai Network: A Platform for High-Performance Internet Applications

Erik Nygren
Ramesh Sitaraman, *University of Massachusetts - Amherst*
Jennifer Sun

Available at: https://works.bepress.com/ramesh_sitaraman/20/

MICROSOFT-1009
Microsoft Corp. v. Sandpiper CDN, LLC
IPR2026-00190

The Akamai Network: A Platform for High-Performance Internet Applications

Erik Nygren[†] Ramesh K. Sitaraman[†‡] Jennifer Sun[†]

[†]Akamai Technologies, 8 Cambridge Center, Cambridge, MA 02142
{nygren, ramesh}@akamai.com, jennifer_sun@post.harvard.edu
[‡]Department of Computer Science, University of Massachusetts, Amherst, MA 01002
ramesh@cs.umass.edu

ABSTRACT
Comprising more than 61,000 servers located across nearly 1,000 networks in 70 countries worldwide, the Akamai platform delivers hundreds of billions of Internet interactions daily, helping thousands of enterprises boost the performance and reliability of their Internet applications. In this paper, we give an overview of the components and capabilities of this large-scale distributed computing platform, and offer some insight into its architecture, design principles, operation, and management.

Categories and Subject Descriptors
C.2.1 [**Network Architecture and Design**]: Distributed networks

C.2.4 [**Distributed Systems**]: Distributed applications, Network operating systems

General Terms
Algorithms, Management, Performance, Design, Reliability, Security, Fault Tolerance.

Keywords
Akamai, CDN, overlay networks, application acceleration, HTTP, DNS, content delivery, quality of service, streaming media

1. INTRODUCTION
The Internet is radically transforming every aspect of human society by enabling a wide range of applications for business, commerce, entertainment, news, and social networking. Yet the Internet was never architected to support the levels of performance, reliability, and scalability that modern-day commercial applications demand, creating significant technical obstacles for those who wish to transact business on the Web. Moreover, these obstacles are becoming even more challenging as current and future applications are evolving.

Akamai first pioneered the concept of Content Delivery Networks (CDNs) [18] more than a decade ago to help businesses overcome these technical hurdles. Since then, both the Web and the Akamai platform have evolved tremendously. Today, Akamai delivers 15-20% of all Web traffic worldwide and provides a broad range of commercial services beyond content delivery, including Web and IP application acceleration, EdgeComputing™, delivery of live and on-demand high-definition (HD) media, high-availability storage, analytics, and authoritative DNS services.

This paper presents a broad overview of the current Akamai platform, including a discussion of many of the key technologies and architectural approaches used to achieve its results. We hope to offer insight into the richness of the platform and the breadth of technological research and innovation needed to make a system of this scale work.

The paper is organized as follows. We first present the problem space and look at the motivations for creating such a platform. Next, an overview of the Akamai platform is followed by an examination of how it overcomes the Internet's inherent limitations for delivering web content, media streams, and dynamic applications. We present the case that a highly distributed network is the most effective architecture for these purposes, particularly as content becomes more interactive and more bandwidth hungry. We then take a more detailed look at the main components of the Akamai platform, with a focus on its design principles and fault tolerant architecture. Finally, we offer a cross-section of customer results to validate the real-world efficacy of the platform.

2. INTERNET APPLICATION REQUIREMENTS
Modern enterprise applications and services on the Internet require rigorous end-to-end system quality, as even small degradations in performance and reliability can have a considerable business impact. A single one-hour outage can cost a large e-commerce site hundreds of thousands to millions of dollars in lost revenue, for example.[1] In addition, outages can cause significant damage to brand reputation. The cost of enterprise application downtime is comparable, and may be measured in terms of both lost revenue and reduced productivity.

Application performance is also directly tied to key business metrics such as application adoption and site conversion rates. A 2009 Forrester Consulting survey found that a majority of online shoppers cited website performance as an important factor in their online store loyalty, and that 40% of consumers will wait no more than 3 seconds for a page to load before abandoning a site [19]. We can find a more concrete quantification of this effect in an Akamai study on an e-commerce website [11]. In the study, site visitors were partitioned: half were directed to the site through Akamai (providing a high-performance experience) while the other half were sent directly to the site's origin servers. Analysis showed that the users on the high-performance site were 15%

[1] For instance, a one-hour outage could cost one well-known, large online retailer $2.8 million in sales, based on 2009 revenue numbers.

more likely to complete a purchase and 9% less likely to abandon the site after viewing just one page. For B2B applications, the story is similar. In a 2009 IDC survey, customers using Akamai's enterprise application acceleration services reported annual revenue increases of $200,000 to over $3 million directly attributable to the improved performance and reliability of their applications [20].

Unfortunately, inherent limitations in the Internet's architecture make it difficult to achieve desired levels of performance natively on the Internet. Designed as a best-effort network, the Internet provides no guarantees on end-to-end reliability or performance. On the contrary, wide-area Internet communications are subject to a number of bottlenecks that adversely impact performance, including latency, packet loss, network outages, inefficient protocols, and inter-network friction.

In addition, there are serious questions as to whether the Internet can scale to accommodate the demands of online video. Even short term projections show required capacity levels that are an order of magnitude greater than what we see on the Internet today. Distributing HD-quality programming to a global audience requires tens of petabits per second of capacity—an increase of several orders of magnitude.

Bridging the technological gap between the limited capabilities of the Internet's infrastructure and the performance requirements of current and future distributed applications is thus critical to the continued growth and success of the Internet and its viability for business. We now take a closer look at why this is so challenging.

3. INTERNET DELIVERY CHALLENGES

Although often referred to as a single entity, the Internet is actually composed of thousands of different networks, each providing access to a small percentage of end users.[2] Even the largest network has only about 5% of Internet access traffic, and percentages drop off sharply from there (see Figure 1). In fact, it takes well over 650 networks to reach 90% of all access traffic. This means that centrally-hosted content must travel over multiple networks to reach its end users.

Unfortunately, inter-network data communication is neither an efficient nor reliable operation and can be adversely affected by a number of factors. The most significant include:

- **Peering point congestion**. Capacity at peering points where networks exchange traffic typically lags demand, due in large part to the economic structure of the Internet. Money flows in at the first mile (*i.e.,* website hosting) and at the last mile (*i.e.,* end users), spurring investment in first and last mile infrastructure. However, there is little economic incentive for networks to invest in the middle mile—the high-cost, zero-revenue peering points where networks are forced to cooperate with competing entities. These peering points thus become bottlenecks that cause packet loss and increase latency.

[2] According to [13], there were over 34,600 active networks (ASes) as of June 2010.

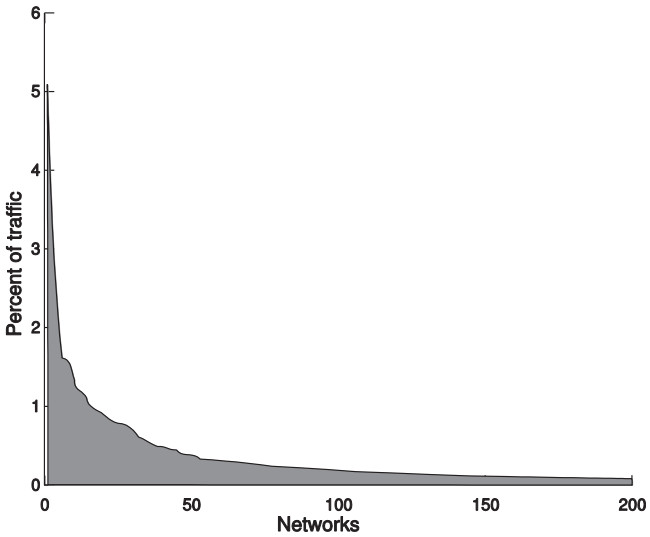

Figure 1: Percentage of access traffic from top networks

- **Inefficient routing protocols**. Although it has managed admirably for scaling a best-effort Internet, BGP has a number of well-documented limitations. It was never designed for performance: BGP bases its route calculations primarily on AS hop count, knowing nothing about the topologies, latencies, or real-time congestion of the underlying networks. In practice, it is used primarily to enforce networks' business agreements with each other rather than to provide good end-to-end performance. For example, [34] notes that several paths between locations within Asia are actually routed through peering points in the US, greatly increasing latency. In addition, when routes stop working or connectivity degrades, BGP can be slow to converge on new routes. Finally, it is well-known that BGP is vulnerable to human error as well as foul play; misconfigured or hijacked routes can quickly propagate throughout the Internet, causing route flapping, bloated paths, and even broad connectivity outages [25].
- **Unreliable networks**. Across the Internet, outages are happening all the time, caused by a wide variety of reasons—cable cuts, misconfigured routers, DDoS attacks, power outages, even earthquakes and other natural disasters. While failures vary in scope, large-scale occurrences are not uncommon. In January 2008, for example, parts of Southeast Asia and the Middle East experienced an estimated 75% reduction in bandwidth connectivity [43] when a series of undersea cables were accidentally cut. In December of the same year, another cable cut incident lead to outages for large numbers of networks in Egypt and India. In both cases the disruptions lasted for multiple days.

  Fragile peering relationships can be culprits as well. When two networks de-peer over business disputes, they can partition the Internet, such that customers from one network—as well as any networks single-homed to it—may be unable to reach customers of the other network. During the high-profile de-peering between Sprint and Cogent in October 2008, for instance, connectivity was adversely affected for an estimated 3,500 networks [35].

Finally, several high profile examples of Internet outages caused by BGP hijacking can be found in [9], such as the global YouTube blackout inadvertently caused by Pakistan in February 2008, as well as the widespread Internet outage caused by a China Telecom leak in April 2010.

- **Inefficient communications protocols**: Although it was designed for reliability and congestion-avoidance, TCP carries significant overhead and can have suboptimal performance for links with high latency or packet loss, both of which are common across the wide-area Internet. Middle mile congestion exacerbates the problem, as packet loss triggers TCP retransmissions, further slowing down communications.

  Additionally, for interactive applications, the multiple round trips required for HTTP requests can quickly add up, affecting application performance [41][40]. Most web browser also limit the number of parallel connections they make for a given host name, further limiting performance over long distances for sites that consist of many objects.

  TCP also becomes a serious performance bottleneck for video and other large files. Because it requires receiver acknowledgements for every window of data packets sent, throughput (when using standard TCP) is inversely related to network latency or round trip time (RTT). Thus, the distance between server and end user can become the overriding bottleneck in download speeds and video viewing quality. Table 1 illustrates the stark results of this effect. True HD quality streams, for example, are not possible if the server is not relatively close by.

Table 1: Effect of Distance on Throughput and Download Time

Distance (Server to User)	Network RTT	Typical Packet Loss	Throughput	4GB DVD Download Time
Local: <100 mi.	1.6 ms	0.6%	44 Mbps (high quality HDTV)	12 min.
Regional: 500–1,000 mi.	16 ms	0.7%	4 Mbps (basic HDTV)	2.2 hrs.
Cross-continent: ~3,000 mi.	48 ms	1.0%	1 Mbps (SD TV)	8.2 hrs.
Multi-continent: ~6,000 mi.	96 ms	1.4%	0.4 Mbps (poor)	20 hrs

  Although many alternate protocols and performance enhancements to TCP have been proposed in the literature ([23], [30], [45]), these tend to be very slow to make their way into use by real-world end users, as achieving common implementation across the Internet is a formidable task.

- **Scalability**. Scaling Internet applications means having enough resources available to respond to instantaneous demand, whether during planned events or unexpected periods of peak traffic. Scaling and mirroring origin infrastructure is costly and time-consuming, and it is difficult to predict capacity needs in advance. Unfortunately, underprovisioning means potentially losing business while overprovisioning means wasting money on unused infrastructure. Moreover, website demand is often very spiky, meaning that companies traditionally needed to provision for anomalous peaks like promotions, events, and attacks, investing in significant infrastructure that sits underutilized most of the time. This also has an environmental cost when underutilized infrastructure consumes significant amounts of power [33].

  Finally, it is important to note that origin scalability is only a part of the scalability challenge. End-to-end application scalability means not only ensuring that there is adequate origin server capacity, but also adequate network bandwidth available at *all* points between end users and the applications they are trying to access. As we will discuss further in Section 5.1, this is a serious problem as Internet video comes of age.

- **Application limitations and slow rate of change adoption**. Although some of the challenges the Internet faces can be partially addressed by changes to protocols and/or client software, history shows that these are all slow to change. While enterprises want to provide the best performance to their end users, they often have little or no control over the end users' software. While the benefits of some protocol changes can be seen as soon as some clients and servers adopt them, other proposed changes can be infeasible to implement as they require close to 100% client adoption to avoid breaking older clients. Most enterprises would also prefer to not to have to keep up with adapting their web infrastructure to tune performance of all of the heterogeneous client software in-use. For example, Microsoft's Internet Explorer 6 (which has considerably slower performance than later versions and doesn't work reliably with protocol optimizations such as gzip compression) was still one of the most popular browsers in use in December 2009, despite being introduced more than eight years prior [29].

4. DELIVERY NETWORK OVERVIEW

The Internet delivery challenges posed above (and in more detail in [27]) illustrate how difficult it can be for enterprises to achieve acceptable levels of performance, reliability, and cost-effective scalability in their Web operations. Most of the bottlenecks are outside the control of any given entity and are inherent to the way the Internet works—as a loosely-coordinated patchwork of heterogeneous autonomous networks.

Over a decade ago, Akamai introduced the Content Delivery Network (CDN) concept to address these challenges. Originally, CDNs improved website performance by caching static site content at the edge of the Internet, close to end users, in order to avoid middle mile bottlenecks as much as possible. Since then the technology has rapidly evolved beyond static web content delivery. Today, Akamai has application delivery networks that can accelerate entire web or IP-based applications, media delivery networks that provide HD-quality delivery of live and on-demand media, and EdgeComputing networks that deploy and execute entire Java J2EE applications in a distributed fashion.

In addition, service offerings have matured to meet additional enterprise needs, such as the ability to maintain visibility and control over their content across the distributed network. This means providing robust security, logging, SLAs, diagnostics, reporting and analytics, and management tools. Here, as with the content delivery itself, there are challenges of scale, reliability, and performance to be overcome.

4.1 Delivery Networks as Virtual Networks

Conceptually, a delivery network is a virtual network[3] built as a software layer over the actual Internet, deployed on widely distributed hardware, and tailored to meet the specific systems requirements of distributed applications and services [Figure 2]. A delivery network provides enhanced reliability, performance, scalability and security that is not achievable by directly utilizing the underlying Internet. A CDN, in the traditional sense of delivering static Web content, is one type of delivery network.

A different but complimentary approach to addressing challenges facing Internet applications is a clean-slate redesign of the Internet [32]. While a re-architecture of the Internet might be beneficial, its adoption in the real world is far from guaranteed. With hundreds of billions of dollars in sunk investments and entrenched adoption by tens of thousands of entities, the current Internet architecture will change slowly, if at all. For example, consider that IPv6—a needed incremental change—was first proposed in 1996 but is just beginning to ramp up in actual deployment nearly 15 years later.

The beauty of the virtual network approach is that it works over the existing Internet as-is, requiring no client software and no changes to the underlying networks. And, since it is built almost entirely in software, it can easily be adapted to future requirements as the Internet evolves.

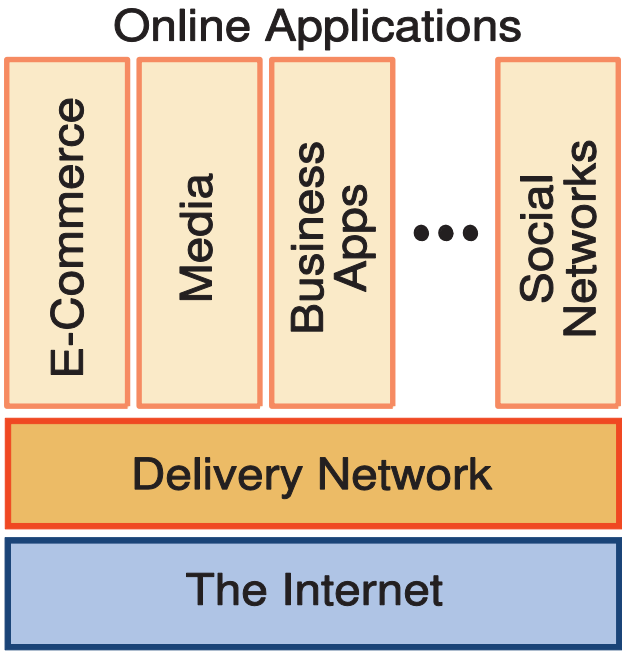

Figure 2: A delivery network is a virtual network built as a software layer over the Internet that is deployed on widely distributed hardware.

[3] The concept of building a virtual network in software to make the underlying network more reliable or higher-performing has a long history both in parallel ([28], [40]) and distributed networks [6].

4.2 Anatomy of a Delivery Network

The Akamai network is a very large distributed system consisting of tens of thousands of globally deployed servers that run sophisticated algorithms to enable the delivery of highly scalable distributed applications. We can think of it as being comprised of multiple delivery networks, each tailored to a different type of content—for example, static web content, streaming media, or dynamic applications. At a high level, these delivery networks share a similar architecture, which is shown in Figure 3, but the underlying technology and implementation of each system component may differ in order to best suit the specific type of content, streaming media, or application being delivered.

The main components of Akamai's delivery networks are as follows:

- When the user types a URL into his/her browser, the domain name of the URL is translated by the *mapping system* into the IP address of an *edge server* to serve the content (arrow 1). To assign the user to a server, the mapping system bases its answers on large amounts of historical and current data that have been collected and processed regarding global network and server conditions. This data is used to choose an edge server that is located close to the end user.
- Each edge server is part of the *edge server platform*, a large global deployment of servers located in thousands of sites around the world. These servers are responsible for processing requests from nearby users and serving the requested content (arrow 2).
- In order to respond to a request from a user, the edge server may need to request content from an origin server.[4] For instance, dynamic content on a web page that is customized for each user cannot be entirely cached by the edge platform and must be fetched from the origin. The *transport system* is used to download the required data in a reliable and efficient manner. More generally, the transport system is responsible for moving data and content over the long-haul Internet with high reliability and performance. In many cases, the transport system may also cache static content.
- The *communications and control system* is used for disseminating status information, control messages, and configuration updates in a fault-tolerant and timely fashion.
- The *data collection and analysis system* is responsible for collecting and processing data from various sources such as server logs, client logs, and network and server information. The collected data can be used for monitoring, alerting, analytics, reporting, and billing.
- Finally, the *management portal* serves two functions. First, it provides a configuration management platform that allows an enterprise customer to retain fine-grained control how their content and applications are served to the end user. These

[4] The *origin* includes the backend web servers, application servers, and databases that host the web application, and is often owned and controlled by the content or application provider rather than the operator of the delivery network. In the case of streaming media, the origin includes facilities for video capture and encoding of live events, as well as storage facilities for on-demand media.

configurations are updated across the edge platform from the management portal via the communications and control system. In addition, the management portal provides the enterprise with visibility on how their users are interacting with their applications and content, including reports on audience demographics and traffic metrics.

While all of Akamai's delivery networks incorporate the systems outlined above, the specific design of each system is influenced by application requirements. For instance, the transport system of an *application delivery network* will have a different set of requirements and a different architecture than that of a *content delivery network*. We will look at each of these system components in more detail in the upcoming sections.

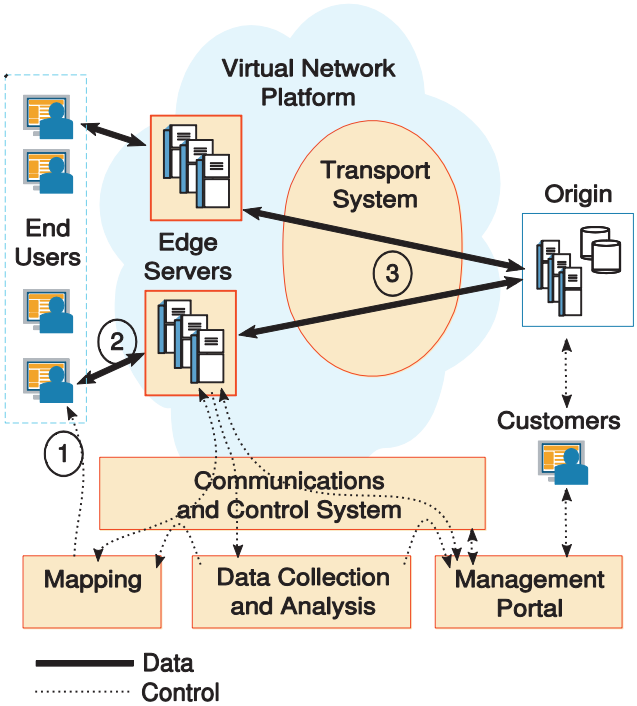

Figure 3: System components of a delivery network. To understand how these components interact, it is instructive to walk through a simple example of a user attempting to download a web page through the Akamai network.

4.3 System Design Principles

The complexity of a globally distributed delivery network brings about a unique set of challenges in architecture, operation and management—particularly in an environment as heterogeneous and unpredictable as the Internet. For example, network management and data collection needs to be scalable and fast across thousands of server clusters, many of which are located in unmanned, third-party data centers, and any number of which might be offline or experiencing bad connectivity at any given time. Configuration changes and software updates need to be rolled out across the network in a safe, quick, and consistent manner, without disrupting service. Enterprises also must be able to maintain visibility and fine-grained control over their content across the distributed platform.

To guide our design choices, we begin with the assumption that a significant number of failures (whether they be at the machine, rack, cluster, connectivity, network levels) is expected to be occurring at all times in the network. Indeed, while not standard in system design, this assumption seems natural in the context of the Internet. We have seen many reasons that Internet failures can occur in Section 3, and have observed it to be true empirically within our own network.

What this means is that we have designed our delivery networks with the philosophy that failures are normal and the delivery network must operate seamlessly despite them. Much effort is invested in designing recovery from all types of faults, including multiple concurrent faults.

This philosophy guides every level of design decision—down to the choice of which types of servers to buy: the use of robust commodity servers makes more sense in this context than more expensive servers with significant hardware redundancy. While it is still important to be able to immediately identify failing hardware (*e.g.,* via ECC memory and disk integrity checks that enable servers to automatically take themselves out of service), there are diminishing returns from building redundancy into hardware (*e.g*, dual power supplies) rather than software. Deeper implications of this philosophy are discussed at length in [1].

We now mention a few key principles that pervade our platform system design:

- **Design for reliability**. Because of the nature of our business, the goal is to attain extremely close to 100% end-to-end availability. This requires significant effort given our fundamental assumption that components will fail frequently and in unpredictable ways. We must ensure full redundancy of components (no single points of failure), build in multiple levels of fault tolerance, and use protocols such as PAXOS [26] and decentralized leader election to accommodate for the possibility of failed system components.
- **Design for scalability**. With more than 60,000 machines (and growing) across the globe, all platform components must be highly scalable. At a basic level, scaling means handling more traffic, content, and customers. This also translates into handling increasingly large volumes of resulting data that must be collected and analyzed, as well as building communications, control, and mapping systems that must support an ever-increasing number of distributed machines.
- **Limit the necessity for human management**. To a very large extent, we design the system to be autonomic. This is a corollary to the philosophy that failures are commonplace and that the system must be designed to operate in spite of them. Moreover, it is necessary in order to scale, else the human operational expense becomes too high. As such, the system must be able to respond to faults, handle shifts in load and capacity, self-tune for performance, and safely deploy software and configuration updates with minimal human intervention. (To manage its 60,000-plus machines, the Akamai network operation centers currently employ around 60 people, distributed to work 24x7x365.)
- **Design for performance**. There is continual work being done to improve the performance of the system's critical paths, not only from the perspective of improving end user

response times but for many different metrics across the platform, such as cache hit rates and network resource utilization. An added benefit to some of this work is energy efficiency; for example, kernel and other software optimizations enable greater capacity and more traffic served with fewer machines.

We will explore these principles further as we examine each of the the Akamai delivery networks in greater detail in the next sections. In Section 5 and Section 6 we outline specific challenges and solutions in the design of content, streaming media, and application delivery networks, and look at the characteristics of the transport systems, which differ for each of the delivery networks.[5] In Section 7, we provide details on the generic system components that are shared among the Akamai delivery networks, such as the edge server platform, the mapping system, the communications and control system, and the data collection and analysis system.

5. HIGH-PERFORMANCE STREAMING AND CONTENT DELIVERY NETWORKS

In this section, we focus on the architectural considerations of delivery networks for web content and streaming media. A fundamental principle for enhancing performance, reliability, and scalability for content and stream delivery is minimizing long-haul communication through the middle-mile bottleneck of the Internet—a goal made feasible only by a pervasive, distributed architecture where servers sit as "close" to end users as possible. Here, closeness may be defined in both geographic and network-topological measures; the ideal situation (from a user performance perspective) would consist of servers located within each user's own ISP and geography, thus minimizing the reliance on inter-network and long-distance communications.[6]

A key question is just how distributed such an architecture needs to be. Akamai's approach generally has been to reach out to the true edge of the Internet, deploying not only in large Tier 1 and Tier 2 data centers, but also in large numbers of end user ISPs. Rather than taking the approach of deploying massive server farms in a few dozen data centers, Akamai has deployed server clusters of varying size in thousands of locations—an approach that arguably adds complexity to system design and management. However, we made this architectural choice as we feel that it is the one that has the most efficacy.

Internet access traffic is highly fragmented across networks—the top 45 networks combined account for only half of user access traffic, and the numbers drop off dramatically from there. This means that unless a CDN is deployed in thousands of networks, a large percentage of traffic being served would still need to travel over multiple networks to reach end users. Being deployed in local ISPs is particularly critical for regions of the world with poor connectivity. More importantly, as we saw in Section 3, Table 1, because of the way TCP works, the distance between server and end user becomes a bottleneck for video throughput. If a CDN has only a few dozen server locations, the majority of users around the world would be unable to enjoy the high quality video streams their last mile broadband access would otherwise allow. Finally, being highly distributed also increases platform availability, as an outage across an entire data center (or even multiple data centers) does not need to affect delivery network performance.

For these reasons, Akamai's approach is to deploy servers as close to end users as possible, minimizing the effects of peering point congestion, latency, and network outages when delivering content. As a result, customers enjoy levels of reliability and performance that are not possible with more centralized approaches.

Finally, while peer-to-peer technologies [8] provide a highly-distributed option for serving static web content, the lack of management and control features in current implementations make them unsuitable as stand-alone solutions for enterprise-quality delivery. Akamai's enterprise customers treat the Akamai network as an extension of their own, in the sense that they expect to maintain control and visibility over their content across the network. This includes management of content freshness and correctness, fine-grained control over how different content is handled, the ability to view real-time analytics and traffic reports, and guarantees of security (including integrity, availability, and confidentiality). These capabilities are as critical to enterprise business requirements as the performance benefits themselves. The lack thereof limits the applicability of peer-to-peer content delivery solutions for most enterprise customers, although Akamai does provide a hybrid option for client-side delivery, discussed more in Section 7.5.5.

5.1 Video-grade Scalability

In addition to speed and reliability, highly distributed network architectures provide another critical advantage—that of end-to-end scalability. One goal of most CDNs, including Akamai, is to provide scalability for their customers by allowing them to leverage a larger network on-demand. This reduces the pressure on content providers to accurately predict capacity needs and enables them to gracefully absorb spikes in website demand. It also creates sizeable savings in capital and operational expenses, as sites no longer have to build out a large origin infrastructure that may sit underutilized except during popular events.

With high-throughput video, however, scalability requirements have reached new orders of magnitude. From near non-existence less than a decade ago, video now constitutes more than a third of all Internet traffic, and Cisco [14] predicts that by 2014, the percentage will increase to over 90%. Just five years old, YouTube recently announced [47] that it now receives 2 billion views per day. In addition to an increase in the number of viewers, the bitrates of streams have also been increasing significantly to support higher qualities. While video streams in the past (often displayed in small windows and watched for short periods of time) were often a few hundred Kbps, today SDTV- and HDTV-quality streams ranging between 2 to 40 Mbps are becoming prevalent as viewers watch up to full-length movies in full-screen or from set-top devices.

[5] The transport systems do share services and components, but are tailored to meet the requirements of the different types of applications they support.

[6] When long-haul communication is unavoidable, as in the case of cold content or live streaming, the transport system is architected to ensure that these communications happen with high reliability and performance.

What does this combination of increased viewership, increased bitrates, and increased viewing-duration mean in terms of capacity requirements? President Obama's inauguration set records in 2009, with Akamai serving over 7 million simultaneous streams and seeing overall traffic levels surpassing 2 Tbps. Demand continues to rise quickly, spurred by continual increase in broadband speed and penetration rates [10]. In April 2010, Akamai hit a new record peak of 3.45 Tbps on its network. At this throughput, the entire printed contents of the U.S. Library of Congress could be downloaded in under a minute. In comparison, Steve Jobs' 2001 Macworld Expo keynote, a record-setting streaming event at the time, peaked at approximately 35,500 simultaneous users and 5.3 Gbps of bandwidth—several orders of magnitude less.

In the near term (two to five years), it is reasonable to expect that throughput requirements for some single video events will reach roughly 50 to 100 Tbps (the equivalent of distributing a TV-quality stream to a large prime time audience). This is an order of magnitude larger than the biggest online events today. The functionality of video events has also been increasing to include such features as DVR-like-functionality (where some clients may pause or rewind), interactivity, advertisement insertion, and mobile device support.

At this scale, it is no longer sufficient to simply have enough server and egress bandwidth resources. One must consider the throughput of the entire path from encoders to servers to end users. The bottleneck is no longer likely to be at just the origin data center. It could be at a peering point, or a network's backhaul capacity, or an ISP's upstream connectivity—or it could be due to the network latency between server and end user, as discussed earlier in Section 3. At video scale, a data center's *nominal* egress capacity has very little to do with its real throughput to end users.

Because of the limited capacity at the Internet's various bottlenecks, even an extremely well-provisioned and well-connected data center can only expect to have no more than a few hundred Gbps of real throughput to end users. This means that a CDN or other network with even 50 well-provisioned, highly connected data centers still falls well short of achieving the 100 Tbps needed to support video's near-term growth.

On the other hand, an edge-based platform, with servers in thousands of locations, can achieve the scale needed with each location supporting just tens of Gbps of output. This reinforces the efficacy of a highly distributed architecture for achieving enterprise-grade performance, reliability, and scalability, particularly in the upcoming era where video will dominate bandwidth usage.

It is also worth noting that IP-layer multicast [16] (proposed early on as a solution for handling large streaming events) tends to not be practical in reality, both due to challenges in supporting within backbone routers without introducing security vulnerabilities, and due to an increasing set of required features such as content access control and time-shifting. This has resulted in the implementation of application-layer multicast services, as we describe in Section 5.3.2. For the drawbacks of IP-layer multicast, the reader is further referred to [12].

5.2 Streaming Performance

A web application is said to perform well if pages download quickly without errors. However, streaming performance is more multi-dimensional and complex. Akamai's research on how users experience streaming media has lead to the definition and measurement of several key metrics. A first metric is *stream availability* that measures how often a user can play streams without failures. Next, since users want the stream to start quickly, it is important to minimize *startup time*. Additionally, users want to watch the stream without interruptions or freezes. Therefore, a third metric measures the *frequency and duration of interruptions* during playback. Finally, users want to experience the media at the highest bitrate that their last-mile connectivity, desktop, or device would allow. Thus, a final important metric is the *effective bandwidth* delivered to the user. A major design goal of Akamai's stream delivery network is to optimize these metrics to provide users a high-quality experience. In addition to the key metrics above, a number of auxiliary metrics such as packet loss, jitter, frame loss, RTT, and end-to-end delay are optimized.

Akamai has built and deployed a global monitoring infrastructure that is capable of measuring stream quality metrics from a user standpoint. This infrastructure includes active measurements made by "agents" deployed around the world. Each agent is capable of simulating users by repeatedly playing streams and testing their quality. See Section 7.5.2 for additional information on monitoring for non-streaming web content.

5.3 A Transport System for Content and Streaming Media Delivery

Within each Akamai delivery network, the transport system is tasked with moving data from the origin to the edge servers in a reliable and efficient manner. The techniques used by this system are tailored to the specific requirements of the data being transported. We illustrate two techniques below, the first of which is tailored for less-frequently accessed content and the second of which can used for live streaming.

5.3.1 Tiered Distribution

With efficient caching strategies, Akamai's edge server architecture provides extremely good performance and high cache hit rates. However, for customers who have very large libraries of content, some of which may be "cold" or infrequently-accessed, Akamai's tiered distribution platform can further improve performance by reducing the number of content requests back to the origin server. With tiered distribution, a set of Akamai "parent" clusters is utilized. These clusters are typically well-provisioned clusters, chosen for their high degree of connectivity to edge clusters. When an edge cluster does not have a piece of requested content in cache, it retrieves that content from its parent cluster rather than the origin server.

By intelligent implementation of tiered distribution, we can significantly reduce request load on the origin server. Even for customers with very large content footprints, we typically see offload percentages in the high 90's [44]. This makes it particularly helpful in the case of large objects that may be subject to flash crowds. In addition, tiered distribution offers more effective use of persistent connections with the origin, as the origin needs only to manage connections with a few dozen parent clusters rather than hundreds or thousands of edge clusters. Moreover, the connections between Akamai's edge clusters and parent clusters make use of the performance-optimized transport system we will discuss in Section 6.1. Additional refinements to this approach, such as having multiple tiers, or using different sets

of parents for different content, can provide additional benefits for some types of content.

5.3.2 An Overlay Network for Live Streaming

Due to their unique requirements, many live streams are handled somewhat differently than other types of content on the Akamai network. Once a live stream is captured and encoded, the stream is sent to a cluster of Akamai servers called the *entrypoint*. To avoid having the entrypoint become a single point of failure, it is customary to send copies of the stream to additional entrypoints, with a mechanism for automatic failover if one of the entrypoints go down. Within entrypoint clusters, distributed leader election is used to tolerate machine failure.

This transport system for live streams then transports the stream's packets from the entrypoint to a subset of edge servers that require the stream. The system works in a publish-subscribe model where each entrypoint publishes the streams that it has available, and each edge server subscribes to streams that it requires. Note that the transport system must simultaneously distribute thousands of live streams from their respective entrypoints to the subset of edge servers that require the stream. To perform this task in a scalable fashion an intermediate layer of servers called *reflectors* is used. The reflectors act as intermediaries between the entrypoints and the edge clusters, where each reflector can receive one or more streams from the entrypoints and can send those streams to one or more edge clusters. Note that a reflector is capable of making multiple copies of each received stream, where each copy can be sent to a different edge cluster. This feature enables rapidly replicating a stream to a large number of edge clusters to serve a highly-popular event. In addition to the scaling benefit, the reflectors provide multiple alternate paths between each entrypoint and edge cluster. These alternate paths can be used for enhancing end-to-end quality via path optimization as described below.

The goal of the transport system is to simultaneously transmit each stream across the middle mile of the Internet with minimal failures, end-to-end packet loss, and cost. To achieve this goal, the system considers the multiple alternate paths available between entrypoints and edge server clusters and chooses the best performing paths for each stream. If no single high-quality path is available between an entry point and an edge server, the system uses multiple link-disjoint paths that utilize different reflectors as the intermediaries. When a stream is sent along multiple paths, the packet loss on any one path can be recovered from information sent along the alternate paths. The recovery is performed at the edge server and results in a "cleaner" version of the stream that is then forwarded to the user. The transport system also uses techniques such as prebursting, which provides the user's media player with a quick burst of data so that stream play can start quickly (reducing startup time). For a comprehensive description of the transport system architecture for live streams, the reader is referred to [24].

Efficient algorithms are needed for constructing overlay paths, since the optimal paths change rapidly with Internet conditions. The problem of constructing overlay paths can be stated as a complex optimization problem. Research advances on solving this problem in an efficient, near-optimal fashion using advanced algorithmic techniques such as LP-rounding can be found in [7].

6. HIGH-PERFORMANCE APPLICATION DELIVERY NETWORKS

As websites have become increasingly dynamic, the ability to improve the performance of applications and other non-cacheable content has become critical. We take two complementary approaches to this, both based on first connecting end users to nearby Akamai servers. The first approach is to speed up long-haul Internet communications by using the Akamai platform as a high-performance overlay network [5][35]. The second approach pushes application logic from the origin server out to the edge of the Internet. These approaches work in concert to improve application performance, reliability, and scalability. Some customer use cases are presented in Section 9.2. We now look at each in more detail.

6.1 A Transport System for Application Acceleration

The transport system for Akamai's application delivery network relies on the use of Akamai's highly distributed edge servers as a high-performance overlay network that makes wide-area Internet communications faster and more reliable. In particular, the communications between any two Akamai servers can be optimized to overcome the inefficiencies we discussed in Section 3 through a number of techniques including path optimization and protocol enhancements.

This transport system is applicable to many types of situations: accelerating non-cacheable customer content and applications, retrieving content (or performing freshness checks) from the

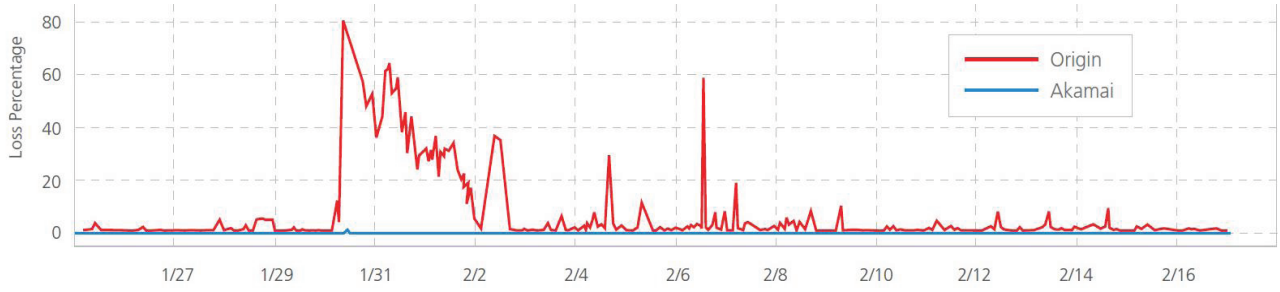

Figure 4: With real-time path optimization, Akamai helps customers avoid connectivity problems such as those depicted here, arising from the Middle East cable cuts.

origin server, and various types of communications internal to the Akamai network as well. In a typical scenario, the end user is first mapped to a nearby server. That server then uses the high-performance overlay to cross the Internet, reaching an Akamai machine near the enterprise's origin server. Typically, the Akamai server will be in the same network or even the same data center as the enterprise origin, so latencies between the two are very low.

The overlay uses several techniques to improve performance by reducing both the number of round trips and the round trip time needed for any given communication. These techniques, described below, all represent areas of ongoing performance research:

- **Path optimization**. In Section 3, we listed some of the limitations of BGP and the reasons why the routes it defines are often less than optimal. In many cases, better performance can be achieved by sending traffic via an alternate path—*i.e.* by directing it through an intermediate server on the Akamai network. Similar to approaches described in [17], [37], [36], and [38], Akamai leverages its distributed network as an Internet overlay. Internet topology and performance data from Akamai's mapping system (described in Section 7.2) are used to dynamically select potential intermediate nodes for a particular path. Then, depending on the scenario, the Akamai network may conduct races to determine which path to use, or it may send communications over multiple paths for added resiliency.

  In [34], analysis of global data collected from the Akamai network reveals that many paths, particularly in Asia, can experience a 30-50% performance improvement when using the overlay.[7] Related research [6], [38] has noted similar results, albeit in non-commercial settings across networks of much smaller scale. Note that in addition to performance improvements, the overlay also increases reliability of communications by offering alternative paths in case connectivity should become severely degraded for the direct path. Figure 4 shows how Akamai maintained high connectivity for customers during the 2008 cable cuts that caused widespread Internet outages in the Middle East, Asia, and Africa.

- **Packet loss reduction**. For latency-sensitive applications including highly interactive websites employing technologies such as AJAX or high-bitrate video streaming, TCP can introduce significant delays when retransmitting lost packets and initiating connections. The same multi-path technology used for path optimization can be used for redundancy, and when combined with forward error correction techniques in an approach similar to [31], offers significant packet loss reduction with minimal overhead and without increasing latency, even for congested paths.

- **Transport protocol optimizations**. Performance gains can also be had by overcoming some of the inefficiencies in TCP and HTTP for long distance Akamai-to-Akamai server communications. Not being constrained by client software adoption rates for internal communications, a proprietary transport-layer protocol is used between its servers to make use of such techniques as:

  - Using pools of persistent connections to eliminate connection setup and teardown overhead.
  - Using optimal TCP window sizing based on knowledge of real-time network latency conditions. For example, by increasing the initial window when throughput is known to be good, an entire communication can often be sent within the initial window, avoiding the need to wait for an acknowledgement. Larger windows can also result in increased throughput over long-distance communications.
  - Enabling intelligent retransmission after packet loss by leveraging network latency information, rather than relying on the standard TCP timeout and retransmission protocols. For example, retransmission timeouts can be set aggressively shorter when throughput is expected to be good.

  These protocol optimizations work symbiotically with the transport system's path optimizations. Much of TCP's overhead is due to its focus on reliability under uncertain network conditions. Since path optimization provides a high-performance path, the transport-layer protocol can be much more aggressive and efficient.

  Moreover, some of these optimizations can be used not only for Akamai server-to-server communications but also for direct communications to end users based on the platform's knowledge of last mile throughput and client capabilities. If the end user has broadband access, the Akamai edge server can again be aggressive in its choice of initial window sizes and retransmission timeouts.

- **Application optimizations**. There are a number of application-layer techniques that can also be used to help boost Web application responsiveness for end users. For example, while delivering an HTML page to a user, the Akamai edge server can also parse and *prefetch* any embedded content (from cache or from the origin server, as appropriate), to ensure that it is already in memory when the user's browser requests it. Thus, even if the embedded content is uncacheable or long-tail (less likely to be in cache), the user experiences responsiveness as if the site were hosted at the local edge server. Akamai edge servers can also follow embedded links and prefetch the associated content. Customer business rules dictate the when and how this should be done.

  Content compression, where appropriate, is another example of an optimization that reduces the number of TCP roundtrips from origin to edge server, as well as edge server to end user (where supported by the browser). Generally, any highly-compressible content, such as HTML, Javascript, or style sheets, that is more than a few KB in size can benefit from compression.[8] The performance benefits are particularly

[7] These improvement percentages are for small transactions. We will see later that large file transfers show substantially greater numbers.

[8] Content less than 4.2 KB in size is small enough to fit into 3 data packets, which is the default size of the initial TCP congestion window. Content this size does not benefit as much from compression as it can already be sent without any roundtrips (*i.e.,* without waiting for TCP acknowledgements

pronounced for end users with slow or high latency connections.

Additional application logic can also be implemented by edge servers, such as authentication or serving different versions of a page based on attributes of the client. More details of this are covered in Section 7.1.

Note that the path and protocol optimizations here accelerate communications in both directions and are therefore ideal for content uploads as well as downloads. Moreover, they are not limited to Web-based applications; Akamai uses similar technologies to accelerate other IP-based enterprise applications such as interactive Web conferencing, virtualized applications (*i.e.,* running over Citrix ICA or Microsoft RDP protocols), large file transfers over SFTP or SSH, and email, as well as other enterprise applications delivered via SSL VPN.

Finally, it is important to realize that the highly distributed nature of the Akamai network is key to the efficacy of the overlay network because the end points of the highly optimized long-haul tunnel are located very close to the origin and the end user. This means virtually the entire communication from origin to end user is optimized, and the brief hops on either end are extremely low latency due to the short distance. In practice, this makes for good long-distance performance—for large files, for example, origin server downloads that go over the high performance overlay can perform nearly as well as files delivered from cache because the overlay is able to deliver the file from origin to edge server as quickly as the edge server can deliver to the end user.

6.2 Distributing Applications to the Edge

While the application transport system is able to speed up communications over the wide-area Internet, the ultimate boost in performance, reliability, and scalability comes when the application itself can be distributed to the edge. Akamai introduced such capabilities on its platform nearly a decade ago with its EdgeComputing™ services, which include the capability for companies to deploy and execute request-driven Java J2EE applications or application components onto Akamai's edge servers. Akamai EdgeComputing takes cloud computing to a level where application resources are allocated not only on-demand but also near the end user. The latter piece (*i.e.,* proximity near the end user) is critical to performance yet still missing from most cloud computing services today.

Implementing a platform capable of EdgeComputing services requires overcoming a number of interesting technical challenges, including session management (and replication across machines), security sandboxing, fault management, distributed load-balancing, and resource monitoring and management, as well as providing the appropriate testing and deployment tools. Akamai's approach to these issues and general implementation are covered in some detail in [15], so we refer the interested reader there.

Not all types of applications can run entirely on the edge; those that rely heavily on large transactional databases will often require significant communication with origin infrastructure. However, many types of applications (or portions of applications) can benefit significantly from EdgeComputing. We summarize some categories of use cases from [15]:

- **Content aggregation/transformation**. These are relatively basic applications that do not require a transactional database. They simply collect content from Web services or other sources and reformat them for display (eg, using XSLT).
- **Static databases**. Product catalogs, store locators, site search, and product configurators are examples of applications that use fairly static databases and can be run entirely at the edge.
- **Data collection**. Many applications requiring forms or other user input can be handled on the edge, with data batched and sent to the origin asynchronously. For example, with a polling application, edge servers could store data locally and send results back to an origin server (or to Akamai's storage system) in a few, larger chunks rather than many individual requests. Data validation and other types of basic logic can be executed by the edge server, without origin server involvement. This approach of aggregating content can reduce origin server load by several orders of magnitude.
- **Complex applications**. Even with applications that require real-time database transactions, running presentation layer components of the application on the edge can still offer performance benefits, as origin server communications can be streamlined to include only raw data rather than full HTML pages. For example, the origin can generate a small dynamic page that references larger cacheable fragments, enabling the final HTML page to be assembled and served at the edge using Akamai's ESI (Edge Side Includes)[9] technology.

In practice, we find that EdgeComputing customers not only benefit from the uniquely high levels of performance, scalability, and fault tolerance this model offers, but also from the ability to develop and deploy their applications more quickly—with much less worry about capacity planning, provisioning infrastructure, and architecting for scalability.

7. PLATFORM COMPONENTS

Now that we have seen some of the different ways in which the Akamai platform enables the deployment and delivery of highly scalable web applications, we can take a closer look at the system components we first introduced in Section 4.2. We have already examined one of the components, the transport system, at some depth across the different types of delivery networks. We now take a closer look at the other system components. The accompanying Figure 5 provides an architectural overview of the major components we will be discussing, although this list is by no means exhaustive.

---

from the receiver), although pipelined HTTP requests make it possible to see additional benefit from compression of even the smallest of content.

[9] Similar to SSI (Server Side Includes), Akamai ESI provides a scripting language that can be executed by Akamai servers, enabling the dynamic assembly of pages at the edge. For more information, see http://www.akamai.com/html/support/esi.html.

7.1 Edge Server Platform

Akamai's edge servers are responsible for processing end user requests and serving the requested content, as well as for acting as intermediaries in our overlay network. The platform offers a rich set of functionality and content-handling features, developed over a decade of experience working with and supporting many of the most sophisticated websites and applications on the Internet. These controls not only ensure correct application behavior as experienced by the end user, but also optimize the performance of applications under different scenarios.

An important feature of the edge server platform is its tremendous configurability via *metadata configuration*, which allows enterprises to retain fine-grained control in applying the platform's various capabilities to the handling of their content.

As an example, a single virtual host (with a single DNS hostname) often contains a wide range of content with different characteristics. Some paths on the host may be configured as highly dynamic non-cacheable content that uses a customers' application-tier as an origin. At the same time, other paths may correspond to static objects served from Akamai's storage system. Authentication and other security policies may be configured for select paths while other paths may be configured to modify particular HTTP request and response headers.

Below we list several categories of edge server capabilities to give an idea of the types of functionality controlled by metadata:

- **Origin server location** and content path (which may be different from the URL path given to the end user).
- **Cache control**, including whether and how long to cache an object or part of an object. A number of different cache consistency and invalidation policies are available to suit different classes of content.
- **Cache indexing.** Customers have the ability to specify what to include in the cache index for an object—for example, whether to include a query string, ignore part of the URL path, or disregard case, in order to maximize the cacheability of their content while maintaining correctness.

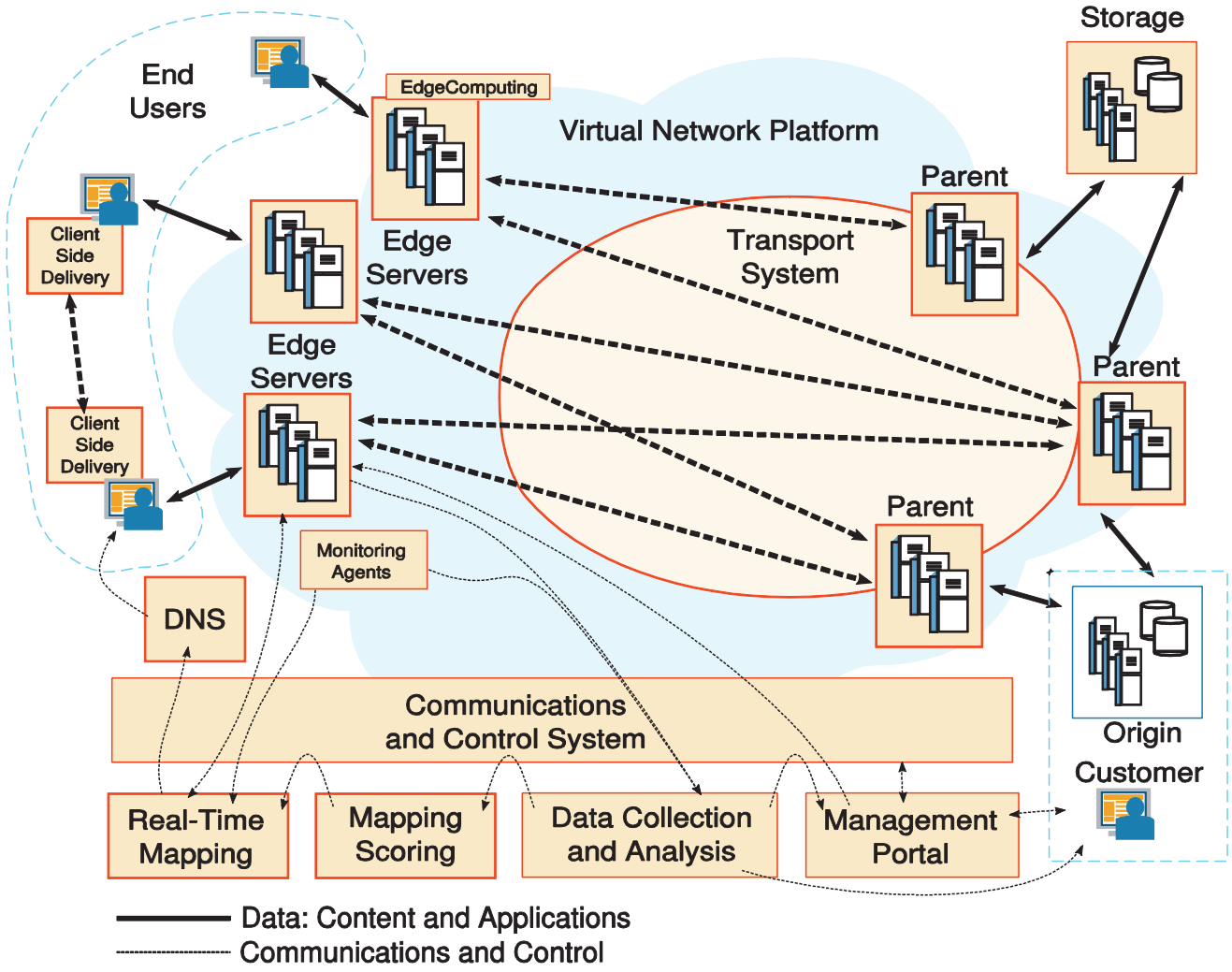

Figure 5: System components of the Akamai platform.

- **Access control**. Numerous authentication and authorization mechanisms are available to control access to or vary content. These include distributed mechanisms (such as validating cookies or client certificates at the edge) as well as centralized mechanisms that can query an origin authentication server.
- **Response to origin server failure**. In the event of origin server failure, customers may indicate whether or not to deliver (potentially stale) content from cache, to use a backup origin server, or to serve static content from the Akamai storage system. Timeout lengths and back-off parameters are configurable as well.
- **Header alteration**. Edge servers can add, delete, or rewrite HTTP request and response headers, such as those containing cookies. This can be used to pass information to origin servers and custom clients, interact with cookies, manage downstream caches, and work around varying browser behaviors, for instance.
- **EdgeComputing**. As noted above, the Akamai platform offers functionality that allows enterprises to run application logic on the edge servers. With metadata configuration, some URLs might be configured to dynamically assemble a page from fragments using Akamai ESI (Edge Side Includes), transform a response using XSLT, or pass a request off to a Java application server on the edge.
- **Performance optimization**. Numerous features have been developed to maximize performance under different conditions and for different classes of applications or content. These include major features such as tiered distribution and overlay path optimization, tuning TCP parameters on a per-connection basis, and asynchronous prefetching and refreshing of content. Many other configuration options are provided to tune performance for different applications and workloads.

The metadata system allows these features to be separately configured based on matching request and response attributes. While the simplest matches are on URL path components, file extensions, and request methods, more advanced metadata matches can change behavior based on attributes including end-user geographic location, connection speed, HTTP request and response headers, cookie values, and many others.

While the platform does support a limited number of in-band metadata features to be specified through Akamai-specific HTTP origin response headers, the primary means of metadata specification is via XML configuration files that are distributed throughout Akamai's network using the communications and control system discussed in Section 7.3. This out-of-band mechanism offers greater security and ease of integration while providing a tremendous degree of control. Metadata configuration can be set across an entire website, a portion of the site, a specific category of content, or even for individual files.

Metadata configurations are easy to update and can be pushed out to the network safely and rapidly. An end-to-end staging environment is provided to enable testing of metadata changes prior to pushing them to production, and changes to the production environment are incrementally phased out with automatic testing and monitoring in between phases.

The architecture is easily extensible, making it simple to evolve platform functionality to meet customers' changing needs. Common metadata best practices are exposed through templates that make it straightforward to configure the desired behavior without worrying about all of the details.

Edge server metadata also supports the use of variables to extend its flexibility. Variable values can be extracted from request and response attributes, transformed, and then later used. As one example, variables might be extracted out of a query string to be used as components of the page's cache key, or as part of dynamic page assembly with ESI.

As with all Akamai platform system components, there is tremendous fault tolerance built into the edge server platform, to achieve its goal of continuing to successfully handle end user requests regardless of failures—whether at the machine, data center, network, or inter-network level. We will delve into this more deeply in the next section, as the mapping system plays a key role in the fault tolerance of the edge server platform.

7.2 Mapping System

Akamai's mapping system is the platform's global traffic director: it uses historic and real-time data about the health of both the Akamai network and the Internet at large in order to create maps that are used to direct traffic on the Akamai network in a reliable, efficient, and high performance manner.

There are two main parts to this system: scoring and real-time mapping. The scoring system first creates a current, topological map capturing the state of connectivity across the entire Internet. More precisely, the map divides the Internet into equivalence classes of IP addresses and represents how (and how well) they connect to each other. This requires collecting and processing tremendous amounts of historic and real-time data—including pings, traceroutes, BGP data, logs, and IP data, collected cumulatively over the years and refreshed on a continual basis. Network latency, loss, and connectivity are monitored at a high frequency, enabling immediate response to Internet faults and changes in performance.

The real-time mapping part of the system creates the actual maps used by the Akamai platform to direct end users (identified by the IP addresses of end users and their name servers) to the best Akamai edge servers to respond to their requests. This part of the system also selects intermediates for tiered distribution and the overlay network. This assignment happens in two main steps:

- **Map to cluster**. First, a top-level map selects a preferred edge server cluster for each equivalence class of end users—assigning each small fragment of the Internet to one of the thousands of Akamai edge server locations. This mapping typically is based on a number of factors, including information from the scoring system (including topological information such as geographic and network/AS proximity between clusters and end users), real-time loss and latency, real-time capacity and demand information, class of traffic (for example, to ensure that disparate needs of latency-sensitive and large-footprint traffic are met), and real-time information about cluster health. These maps are updated roughly every minute to capture current conditions. If a connectivity problem between a cluster and a set of end users

is observed, for example, those end users will be directed to clusters that will provide them better performance. A feedback control system tracks load, capacity, and demand along multiple dimensions and ensures that the demand sent to any given cluster will not cause the load in that cluster to exceed any of its capacity targets.

- **Map to server**. Once assigned to a specific cluster, a low-level map within the cluster directs the user to a specific machine, based on factors including the likelihood of that machine to have the requested piece of content in cache. It is desirable to maintain locality within clusters (mapping requests for the same piece of content to the same machine), as this improves performance and makes efficient use of cache space. The challenge is to do so in a dynamic environment that also factors in load changes and machine failures. Akamai's pioneering research in this area began with consistent hashing in [21] and [22] over a decade ago, and has evolved significantly from that point.

When hardware and network faults are identified (such as a failing hard drive on a server), the failed edge servers are "suspended" and will finish up processing in-progress requests but will not be sent any additional end users until the fault has been resolved. A more detailed example is provided in Section 8.

The mapping system itself is a fault-tolerant distributed platform that is able to survive failures of multiple data centers without interruption. The scoring and map-to-cluster portions of the system run in multiple independent sites and leader-elect based on the current health status of each site. The map-to-server portions of the system run on multiple machines within each target cluster. All portions of the system, including monitoring agents and DNS servers, communicate via a multi-path overlay transport (described in more detail in Section 7.3) that is able to tolerate network faults.

Because the efficacy of the mapping system is important to the overall performance of Akamai's system, there is continuous research and development being performed to advance and refine it. This includes ongoing work to improve the quality of scoring data inputs, improve locality for large-footprint content, address shifts in Internet architecture, develop new methods for enhanced fault-isolation, and optimize the performance of system components to enable even faster response times.

One example of such a refinement is work that was done to enhance the servers' abilities to adjust their own capacity inputs to the mapping system based on self-monitoring of resource utilization. This enables heterogeneous hardware, running different types of applications, to be utilized more efficiently throughout the network.

The sheer volume (and frequency) of data being processed in the mapping system also presents a challenge. During an early system redesign, analysis of the initially proposed design indicated that the amount of data that would need to be collected and communicated would have exceeded the amount of end user traffic being served. Much work has been done since, and continues to be done, to reduce this data communication burden while retaining all information essential to high-quality mapping.

7.3 Communications and Control System

The Akamai platform uses several different models for internal communications, each presenting its own challenges within the context of a highly distributed Internet platform. For all of these systems, we face key challenges of scale (communicating with and controlling a network of over 60,000 machines) and reliability (particularly in communication, as some Akamai clusters have great connectivity and performance to their end users, but poor connectivity to the rest of the Internet).

We expand on a few example systems here:

- **Real-time distribution of status and control information.** Here we have small messages that need to be propagated throughout the network in real time, such as for the mapping system's inputs and outputs (load, health, connectivity, and control information). For these, we use a publish/subscribe model with multi-path tiered fan-out architecture. Publishers announce to a set of globally distributed intermediate nodes. Subscribers can subscribe to one or more of these intermediates. This multi-path architecture minimizes latency while enabling scaling and tolerance of network faults.

- **Point-to-point RPC and Web Services.** In many of the cases where systems need highly reliable and low latency point-to-point communications, such as for Web Services, we are able to utilize the Akamai high-performance overlay (described in Section 6.1) to improve reliability and performance, even in the face of network problems.

- **Dynamic configuration updates.** Many Akamai system components need to receive frequent configuration updates with low latency, sometimes as frequently as every few minutes. One example of this is the customer metadata configuration files used to configure the edge server platform, as described in Section 7.1. Key design goals here include version consistency across the network (including graceful handling of connectivity issues and machines that can fail and restart at any time), reliability and scalability of the system, and a mechanism for ensuring that propagated changes do not adversely affect the network. Our approach is to publish the data to a set of highly-available storage servers that use quorum-based replication to "accept" an update. To achieve scale and low latency in distribution, updates are then propagated throughout the network using Akamai's own content delivery services. Finally, configuration rollouts are automatically phased, with health checks performed at each step, to protect the network. Further details, including the vector-exchange-based acceptance algorithm and the index merging recovery mechanism, are covered in [39].

- **Key Management Infrastructure**. There is a strong desire to keep sensitive cryptographic keys, such as those used for SSL, from ever touching the disks of machines. Akamai's key management infrastructure performs extensive security audits of machines prior to distributing keys to software on those machines. Multiple key distribution servers (which use PAXOS to coordinate their database replication) allow machines to obtain audits and keys even in the face of network problems. Machines which have not yet received keys will automatically suspend themselves and will not be included the mapping system's maps.

- **Software and Machine Configuration Management**. Akamai's system for distributing software and system configuration information throughout its network is designed to handle the heterogeneous and distributed nature of the platform. A key requirement is that machines must converge to have the correct software and system configuration, even in the face of roll-backs and missing update steps that may have resulted from connectivity problems. As part of each machine's software and system configuration update, the desired state of the machine is constructed and then compared against the running state. The system then takes any actions necessary to make the running state match the desired state, such as updating software components. Due to the design-for-reliability principle, software updates can be made with no visible external impact—for example, a machine requiring an update will be suspended so that new requests will be directed to other servers, and the machine will then restart its services after in-progress requests have completed.

7.4 Data Collection and Analysis System

The Akamai platform also makes use of several different data collection and analysis mechanisms, all of which share the common design challenges of scalability and fault tolerance. These mechanisms include:

- **Log collection.** Customers' business requirements often dictate a need for their raw log files, and Akamai also relies on log files for billing its customers. Routinely serving well over 10 million HTTP requests per second translates into needing to process well over 100 TB of logs per day. Compressed logs are reliably aggregated to a set of clusters which have processing pipelines to validate and filter them. Attributes of selected logs are then passed on to systems that can include processing for analytics, loading into databases for historical reporting and billing, and/or delivery to customers.

- **Real-time data collection and monitoring**. Akamai's *Query* system is a distributed real-time relational database that allows for near real-time monitoring of status information across the distributed network. Status data is provided by nearly every software component in the Akamai platform in the form of table rows, and these rows are then aggregated into thousands of tables within the Query system. Query supports a standard SQL interface for enabling arbitrary, ad-hoc queries on the data, rather than having to define questions of interest ahead of time. We refer the interested reader to [36] for more information on Query's architecture and capabilities.

  Akamai makes heavy use of Query for both monitoring and alerting. For example, network-wide invariants can be expressed as SQL statements (*e.g.,* "no more than N machines should be exhibiting some system behavior within a geographic region at the same time as some other system behavior is occurring") with any returned rows resulting in alerts for further investigation within our operations centers. Other SQL statements (*e.g.,* "95th percentile of resource utilization by certain processes, broken down by type of hardware and software version") can be used to extract complex aspects of system state for trending and analysis.

- **Analytics and Reporting**. Akamai's analytics and reporting systems enable customers to view information about their site's traffic and performance. The system consumes outputs from the log collection, Query, and other systems, processing it into a format that enables multi-dimensional reporting. The most recent generation of the system uses a modified MapReduce framework to extract various attributes and submits the data into a fault-tolerant, column-oriented database system. Customers can then utilize a reporting interface to construct and issue multidimensional queries against the database to gain insight into their site traffic, user demographics, and network performance.

7.5 Additional Systems and Services

The Akamai platform includes a number of other highly scalable, high availability infrastructures that we will not discuss in detail here, though each plays an important role in the services offered to Akamai's customer base.

7.5.1 DNS

DNS is an important part of most Internet applications, being used by end users to map from host names to IP addresses. DNS is also one of the primary mechanisms for interfacing with Akamai's mapping system, communicating decisions about which end users should be assigned to which Akamai clusters and machines. As such, Akamai has deployed a globally distributed system of highly-available authoritative DNS servers, both for answering dynamic answers based on Akamai mapping decisions, as well for providing static authoritative answers for customer zones. Akamai has taken numerous measures to ensure strong fault tolerance for its DNS system, utilizing multiple mechanisms to allow the system to both scale to high request rates and to provide excellent performance around the world.

This high availability system is also made available to customers as an authoritative DNS service. With this service, DNS zone contents are securely transferred from customers' master DNS servers, which would then no longer need to be exposed to end users. The contents are distributed to Akamai's global DNS infrastructure, which then handles the customers' DNS queries.

7.5.2 Monitoring Agents

For monitoring network and website performance, Akamai has multiple globally distributed systems of monitoring agents. Various agents can perform pings, traceroutes, as well as requests via numerous Internet protocols such as HTTP. Tests are configured by both the mapping system (*e.g.*, to map out the Internet and monitor loss and latency in real-time) and by customers (*e.g.*, to measure website availability and performance, with results being fed into the analytics systems).

7.5.3 Global Traffic Manager

As customers often wish to have origin servers deployed in multiple locations for fault-tolerance and load-balancing, Akamai exposes a version of its mapping technology to its customers as a service. This Global Traffic Manager (GTM) service consists of agents at customer origin locations that monitor Internet performance and collect load information to feed into the mapping system. Answers are distributed to end users (or Akamai servers using these custom origins) via DNS, based on factors such as

load, network latency, geographic and network proximity, and customer business rules.

7.5.4 Storage

Akamai's platform includes a high-availability storage system. This system can be used as an origin server for many types of content, such as static objects, large media libraries, and backup sites. EdgeComputing applications can also use the storage system as a repository for some types of data. The storage architecture is designed to have no single point of failure; servers are deployed in clusters in multiple geographic locations, with both in-cluster and multi-cluster replication automatically provided. Multiple mechanisms are provided for uploading to the system, ranging from rsync-over-ssh to HTTPS POST.

7.5.5 Client Side Delivery

As an additional approach to improving end user performance and reducing customer cost, Akamai provides a client side delivery system [3] that can be used by customers. This includes client-side software (to be installed on end user machines) which securely communicates with distributed Akamai systems. Client-side web applications (running within the browser) can communicate locally with this client software to request content, such as for the distribution of large software packages. The client side delivery system behaves much like many peer-to-peer systems, but provides additional features required by enterprise customers. In particular, most enterprise customers care strongly about guaranteed performance and availability of their content, and they do not want to lose control or visibility over their content delivery. The client side delivery system achieves these goals: it leverages the rest of the Akamai platform to seed content, and is able to fall back to requesting content from the nearest Akamai servers when peers are not providing adequate performance. By integrating tightly with the Akamai edge server platform, client software is able to provide customers with control over and visibility into the distribution of their content, as well as guarantees about the integrity of the delivered content. As the client software communicates with an Akamai control-plane, decisions about which peers to communicate with can be made based in the Akamai mapping system's real-time understanding of Internet topology.

7.5.6 Management Portal

Akamai's web-based management portal provides customers with a high-availability configuration and management platform allowing them to maintain control over and visibility into their content, applications, and traffic. Customers can view information such as traffic reports, network performance and packet loss, end user demographics, download completion rates, media play time/buffer time, and custom-defined reports. Other portal capabilities include self-provisioning, configuration (such as managing site metadata configurations or invalidating content), diagnostics, management, alerts, and reporting. The portal system is accelerated using Akamai's application delivery network, improving its performance and reliability for customers around the world.

8. EXAMPLE: MULTI-LEVEL FAILOVER

As we mentioned in Section 4.3, we take an approach similar to recovery-oriented computing throughout our platform design—making the assumption that failures are an inevitable part of operation and the system must be able to operate regardless.

We now look briefly at a concrete application of this approach by examining how Akamai content delivery services maintain availability in a scenario of multiple component failures (more details are given in [1], based on an older version of the system). To understand how this works, we must first look at the basic flow of an HTTP request to the Akamai network.

Initially, a DNS lookup is made to resolve the Akamai hostname. The DNS resolution takes several steps:

- The first request goes to generic TLD servers, which return Akamai Top Level Name Servers (TLNS) as authorities, generally with long DNS TTLs. The Akamai TLNS are globally distributed, using a mixture of IP Anycast and large clusters.
- The next query, to an Akamai TLNS, returns delegations with shorter DNS TTLs to a number of Akamai Low Level Name Servers (LLNS). The Akamai LLNS are typically located in close network proximity to the resolving name server.
- The final query, to an Akamai LLNS, returns edge server IP addresses based on both the cluster assignment and the low level map described above. These answers have very short TTLs so that changes to the mapping assignments (such as in response to failures or shifts in demand) can be rapidly distributed to end users.

The end user browser then makes an HTTP request to the edge server IP address to receive the content. If the content is not already in cache, the edge server retrieves it from the origin server and then delivers it to the end user.

Now consider the following types of failure:

- **Machine failure**: Within an edge cluster, Akamai implements high availability techniques we have evolved from principles similar to those in TCPHA [42]. This allows for virtually seamless response to machine failures, as another machine will start responding to the IP address of the failed machine. In addition, the low level map is updated every few seconds, redirecting new requests as appropriate to accommodate for the failure.
- **Cluster failure**: When an entire cluster fails or is experiencing unreliable connectivity, the cluster assignment from mapping is rapidly updated to no longer hand out clusters that have failed or which are experiencing connectivity issues.
- **Connectivity failure**: If connectivity between the origin server and the edge degrades, the platform detects this quickly and uses its path optimization technology to find good alternate paths through intermediate nodes on the Akamai platform.

Note that even if all of the above faults occurred simultaneously, the platform would still recover quickly.

In addition to the robust platform availability that is its direct goal, there are a couple of useful byproducts to the recovery-oriented design philosophy. The first is a significant reduction in the number of operations staff needed to manage the network. Because the network is designed with the assumption that components are failing at all times, staff do not need to worry about most failures nor rush to address them. Moreover, staff can be aggressive in proactively suspending components if they have the slightest concern, since doing so will not affect the performance of the overall system. Even though the operations staff is itself distributed across multiple sites, the human staff is not in the critical path for the operation of the network.

A second benefit is the ability to roll out software updates in a rapid and non-disruptive manner, as described in Section 7.3. Again, because the failure of a number of machines or clusters does not affect the overall system, zoned software rollouts can be performed quickly and frequently without disrupting services to Akamai's customers. Some interesting metrics relating to the two benefits we cite here are presented in [1].

9. CUSTOMER BENEFITS AND RESULTS

Akamai customers can easily make use of multiple Akamai delivery networks within a single website, tailoring delivery methods to meet their application requirements. We now present a sampling of use cases which illustrate different ways in which our customers have implemented applications on the Akamai platform and the benefits they have achieved as a result. Many additional examples and case studies can be found at [2].

9.1 Customer Examples for Content and Streaming Delivery

Customer-cited benefits for content and streaming delivery include not only the revenue and brand enhancement benefits from improved performance and reliability, but also significant infrastructure cost savings, protection from DDoS attacks, and the ability to handle large flash crowds.

- **New York Post: 20X performance improvement.** The New York Post first came to Akamai after publishing an exclusive news story that generated huge traffic surge that overwhelmed its infrastructure. Akamai was able to complete provision and integrate the site within a few hours, enabling the New York Post to handle its flash crowd—while making home page download times 20 times faster as well.
- **U.S. Government: Protection against DDoS attack**. In July 2009, the U.S. government faced the largest DDoS attack in its history, with the top-targeted site receiving nearly 8 years' worth of traffic in one day. Despite the unprecedented scale of the attack, all of the U.S. government sites delivered via Akamai—including sites for the White House and 13 of the 15 Federal Cabinet level agencies—remained online, with Akamai absorbing more than 200 Gbps of attack traffic targeted at the government sites. At the same time, Akamai continued serving traffic to legitimate users and maintained a consistently high level of availability for its customers, delivering traffic at over 1 Tbps for the rest of its customer base.
- **MySpace: 6X speed up, 98% offload.** This popular social networking company called on Akamai to help it handle its virtually unprecedented pace of growth. Despite its vast footprint of personalized and user-generated content, MySpace is able to offload 98% of its traffic to Akamai (using tiered distribution). It has seen a 2.6X performance improvement to end users in the US and a 6X improvement for international users.
- **Sophos: Eliminated costly infrastructure build out.** Global security company Sophos delivers antivirus software and updates to 100 million users in 150 countries. It first began using Akamai when its London-based servers were becoming overwhelmed with each release. Now, Sophos delivers 20 times the traffic and achieves a 99.9% download success rate without any additional infrastructure. It estimates that it has saved hundreds of thousands of dollars annually and avoided a costly 25-data center deployment.
- **MTV Hope for Haiti Concert: 5.8M streams served, $61 million raised.** MTV Networks came to Akamai with a plan to hold a benefit concert for earthquake victims in Haiti within one week. They wanted to broadcast it online as well as on television, and their goal was to deliver the best possible streaming experience while handling any size audience. Akamai helped make the concert a success, delivering 100,000 concurrent streams during the event and more than 5.8 million streams throughout the weekend.

9.2 Customer Examples for Application Delivery

Akamai's application delivery services combine Akamai's performance optimization, overlay network, EdgeComputing, and content delivery capabilities to accelerate the entire range of Web and IP-based applications. We look at a sampling of customer use cases:

- **Enterprise applications.** Businesses and SaaS providers turn to Akamai to help them overcome performance and reliability challenges for their mission-critical enterprise applications. Customers typically report global performance improvements in the range of 100% to 700%.[10] An independent report by Tolly Enterprises [23], for example, tested response times seen by live users completing tasks using various applications running on a Citrix XenDesktop virtual desktop solution. Tolly found Akamai provided improved performance by 170% to 700% from different locations in Asia. For enterprises, these improvements can translate into significant dollars through increases in revenue and operational efficiency. An IDC research report [20] determined that organizations using Akamai's application acceleration services for the enterprise applications enjoyed an average annual benefit of $7 million on an average investment of $174,000.
- **Amazon EC2: Boosting cloud computing performance.** While companies are looking to services like EC2 and Google App Engine to reduce capital and operational costs, these cloud infrastructures still are lacking in terms of providing performance and reliability because content still needs to travel over the middle mile Internet to reach end users. Applications and the services they utilize (such as

[10] Many specific case studies and results can be found at http://www.akamai.com/html/solutions/web_application_accelerator.html.

storage) may also be located in different data centers. The Akamai platform works with cloud-hosted origin infrastructures the same way as any other, to improve the end user experience. Some results from EC2-hosted applications include:

- Project collaboration (SaaS) company: 110% performance improvement
- Photo and video sharing company: 290% improvement
- Professional sports organization: 300-400% improvement

- **Large file transfers.** Enterprises needing to transfer large files to customers, partners and employees across the globe see significant performance gains when leveraging Akamai's overlay network. Typical results include:
  - 5X increase in large file (2 GB) transfer throughput from Europe to the US for a data backup and recovery company
  - 4X to 5X improvements in the performance for a global semiconductor company when using SFTP to transfer large schematic design files
  - 2.3X speed ups for file transfers over a VPN between India and the US for a global publisher. In addition, a significant portion of the resource-intensive SSL encryption was offloaded to Akamai.

  Note that these performance gains are due solely to path and protocol optimizations rather than edge caching, as the latter would not be effective in point-to-point transfers and cases with very small numbers of users downloading content in any given region.

- **eCommerce:** Akamai securely enables billions of dollars in annual eCommerce revenue for its online retailers, who include 90 of the top 100 Internet retail sites. Customers [4] report significant infrastructure cost savings in addition to performance improvements that help drive site growth, enhance brand reputation, and decrease shopping cart abandonment.

- **EdgeComputing:** Sony Ericsson used Akamai EdgeComputing to avoid the costly build out of several regional data centers. They deployed a number of application components—including a phone configurator, shopping cart, and dealer locator application—to the edge, while other components ran in a centralized datacenter. The hybrid cloud strategy reduced application response time by a factor of four while increasing application availability from 92% to 100% and reducing infrastructure needs by 65%.

10. ACKNOWLEDGMENTS

All of the systems discussed here would not have been possible without over a decade of hard work by Akamai engineers and researchers. While it would not be feasible to list all of their names here, they are credited with making the Akamai platform as scalable and reliable as it is. We would also like to thank the research community for developing many protocols and algorithms utilized within our system.

11. REFERENCES

[1] Afergan, M., Wein, J., and LaMeyer, A. Experience with Some Principles for Building an Internet-Scale Reliable System. In *Proceedings of the 2nd conference on Real, Large Distributed Systems*, pp.1-6, Dec. 2005.

[2] Akamai Customer List: http://www.akamai.com/html/customers/index.html

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

[4] Akamai Online Commerce: http://www.akamai.com/dl/akamai/Akamai_Online_Commerce.pdf

[5] Andersen, D. Improving End-to-End Availability Using Overlay Networks. PhD thesis, MIT, 2005. http://www.cs.cmu.edu/~dga/papers/andersen-phd-thesis.pdf

[6] Andersen, D., Balakrishnan, H., Kaashoek, F., and Morris, R. Resilient Overlay Networks. *18th ACM SOSP*, Oct. 2001.

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

[8] Androutsellis-Theotokis, S. and Spinellis, D. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4): 335-371, 2004.

[9] Associated Press. At a glance, a look at Internet outages due to route 'hijackings'. May 2010. http://blog.taragana.com/index.php/archive/at-a-glance-a-look-at-internet-outages-due-to-route-hijackings/

[10] Belson, D. Akamai State of the Internet Report, *ACM SIGOPS Operating Systems Review*, 44(3), July 2010

[11] Case study of Akamai customer eBags: http://www.akamai.com/html/customers/case_study_ebags.html

[12] Chu, Y., Rao, S., Seshan, S. and Zhang, H. A Case for End System Multicast. *IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast*, 20(8), 2002.

[13] CIDR Report: http://www.cidr-report.org/as2.0/

[14] Cisco Systems. Cisco Visual Networking Index: Forecast and Methodology, 2009-2014. June 2010. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf

[15] Davis, A., Parikh, J., and Weihl, W. EdgeComputing: Extending Enterprise Applications to the Edge of the Internet. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, May 2004.

[16] Deering, S. Multicast Routing in Internetworks and Extended LANs. In *Proceedings of the ACM SIGCOMM*, August 1988.

[17] Detour Project: http://www.cs.washington.edu/research/networking/detour/

[18] Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. Globally Distributed Content Delivery. *IEEE Internet Computing*, 6(5):50-58, 2002.

[19] Forrester Consulting. eCommerce Web Site Performance Today: An Updated Look At Consumer Reaction To A Poor Online Shopping Experience. Aug. 17, 2009.

[20] IDC. Determining the Return on Investment of Web Application Acceleration Managed Services. Oct. 2009.

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

[22] Karger, D., Sherman, A., Berkheimer, A., Bogstad, B., Dhanidina, R., Iwamoto, K., Kim, B., Matkins, L., and Yerushalmi, Y. Web Caching with Consistent Hashing. *Computer Networks,* 31(11): 1203-1213, 1999.

[23] Katabi, D., Handley, M., and Rohrs, C. Congestion Control for High Bandwidth-Delay Product Networks. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, 2002.

[24] Kontothanassis, L., Sitaraman, R., Wein, J., Hong, D., Kleinberg, R., Mancuso, B., Shaw, D., and Stodolsky, D. A Transport Layer for Live Streaming in a Content Delivery Network. In *Proceedings of the IEEE, Special issue on evolution of Internet technologies*, 92(9):1408-1419, Aug. 2004.

[25] Kuhn, R., Kotikalapudi, S., and Montgomery, D. Border Gateway Protocol Security: Recommendations of the National Institute of Standards and Technology. Special Publication 800-54, July 2007. http://csrc.nist.gov/publications/nistpubs/800-54/SP800-54.pdf

[26] Lamport, L. The Part-Time Parliament. *ACM Transactions on Computer Systems*, 16(2):133-169, May 1998.

[27] Leighton, T. Improving Performance on the Internet. *Communications of the ACM*, 52(2):44-51, Feb. 2009.

[28] Leighton, T., Maggs, B., and Sitaraman, R. On the fault tolerance of some popular bounded-degree networks. In the *33rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 542-552, 1992.

[29] Net Market Share: http://marketshare.hitslink.com/browser-market-share.aspx?qprid=2

[30] Network Working Group RFC 3649. High Speed TCP for Large Congestion Windows, Dec 2003. http://www.ietf.org/rfc/rfc3649.txt

[31] Nguyen, T. and Zakhor, A. Path Diversity with Forward Error Correction (PDF) System for Packet Switched Networks. In *Proceedings of IEEE INFOCOM 2003*, Apr. 2003.

[32] NSF NeTS FIND Initiative: http://www.nets-find.net/

[33] Qureshi, A., Weber, R., Balakrishnan, H., Guttag, J., Maggs, B. Cutting the Electric Bill for Internet-Scale Systems. *ACM SIGCOMM*, 2009.

[34] Rahul, H., Kasbekar, M., Sitaraman, R., and Berger, A. Towards Realizing the Performance and Availability Benefits of a Global Overlay Network. *MIT CSAIL TR* 2005-070, Dec. 2005. http://hdl.handle.net/1721.1/30580

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

[36] Repantis, T., Cohen, J., Smith, S., and Wein, J. Scaling a Monitoring Infrastructure for the Akamai Network. *ACM SIGOPS Operating Systems Review*, 44(3), July 2010.

[37] Resilient Overlay Networks: http://nms.csail.mit.edu/ron/

[38] Savage, S., Collins, A., Hoffman, E., Snell, J., and Anderson, T. The End-to-End Effects of Internet Path Selection. In *Proc. ACM SIGCOMM*, pp. 289-299, Sept. 1999.

[39] Sherman, A., Lisiecki, P., Berkheimer, A., and Wein, J. ACMS: The Akamai configuration Management System. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pp. 245-258, 2005.

[40] Sitaraman, R. Communication and fault tolerance in parallel computers. PhD thesis, Princeton University, 1993.

[41] Souders, S. High-performance web sites. *Communications of the ACM*, 51(12):36-41, Dec. 2008.

[42] TCPHA: http://dragon.linux-vs.org/~dragonfly/htm/tcpha.htm

[43] TeleGeography. Cable cuts disrupt Internet in Middle East and India. *CommsUpdate*, Jan. 2008. http://www.telegeography.com/cu/article.php?article_id=21528.

[44] Testimonial of Akamai customer MySpace: http://www.akamai.com/html/customers/testimonials/myspace.html

[45] Tierney, B. TCP Tuning Guide for Distributed Applications on Wide Area Networks. *USENIX & SAGE Login*, 26(1):33-39, Feb. 2001.

[46] Tolly. Akamai IP Application Accelerator Service: Real-world Performance Benchmarking of the Citrix Virtual Desktop Enviroment. Sept. 2009. http://www.tolly.com/DocDetail.aspx?DocNumber=209121

[47] YouTube Blog, May 16, 2010. http://youtube-global.blogspot.com/2010/05/at-five-years-two-billion-views-per-day.html