DOI:00.0000/000

Given the Internet's bottlenecks, how can we build fast, scalable, content-delivery systems?

BY TOM LEIGHTON

Improving Performance on the Internet

WHEN IT COMES to achieving performance, reliability, and scalability for commercial-grade Web applications, where is the biggest bottleneck? In many cases today, we see that the limiting bottleneck is the *middle mile*, or the time data spends traveling back and forth across the Internet, between origin server and end user.

This wasn't always the case. A decade ago, the *last mile* was a likely culprit, with users constrained to sluggish dial-up modem access speeds. But recent high levels of global broadband penetration—more than 300 million subscribers worldwide—have not only made the last-mile bottleneck history, they have also increased pressure on the rest of the Internet infrastructure to keep pace.[5]

Today, the *first mile*—that is, origin infrastructure—tends to get most of the attention when it comes to designing Web applications. This is the portion of the problem that falls most within an application architect's control. Achieving good first-mile performance and reliability is now a fairly well-understood and tractable problem. From the *end user*'s point of view, however, a robust first mile is necessary, but not sufficient, for achieving strong application performance and reliability.

This is where the middle mile comes in. Difficult to tame and often ignored, the Internet's nebulous middle mile injects latency bottlenecks, throughput constraints, and reliability problems into the Web application performance equation. Indeed, the term *middle mile* is itself a misnomer in that it refers to a heterogeneous infrastructure that is owned by many competing entities and typically spans hundreds or thousands of miles.

This article highlights the most serious challenges the middle mile presents today and offers a look at the approaches to overcoming these challenges and improving performance on the Internet.

Stuck in the Middle

While we often refer to the Internet as a single entity, it is actually composed of 13,000 different competing networks, each providing access to some small subset of end users. Internet capacity has evolved over the years, shaped by market economics. Money flows into the networks from the first and last miles, as companies pay for hosting and end users pay for access. First- and last-mile capacity has grown 20- and 50-fold, respectively, over the past five to 10 years.

On the other hand, the Internet's middle mile—made up of the peering and transit points where networks trade traffic—is literally a no man's land. Here, economically, there is very little incentive to build out capacity. If anything, networks want to minimize traffic coming into their networks that they don't get paid for. As a result, peering points are often overburdened, causing packet loss and service degradation.

The fragile economic model of peering can have even more serious consequences. In March 2008, for example, two major network providers, Cogent and Telia, de-peered over a business dispute. For more than a week, customers from Cogent lost access to Telia and the networks connected to it, and vice versa, meaning that Cogent and Telia end users could not reach certain Web sites at all.

Other reliability issues plague the middle mile as well. Internet outages have causes as varied as transoceanic cable cuts, power outages, and DDoS (distributed denial-of-service) attacks. In February 2008, for example, communications were severely disrupted in Southeast Asia and the Middle East when a series of undersea cables were cut. According to TeleGeography, the cuts reduced bandwidth connectivity between Europe and the Middle East by 75%.[8]

Internet protocols such as BGP (Border Gateway Protocol, the Internet's primary internetwork routing algorithm) are just as susceptible as the physical network infrastructure. For example, in February 2008, when Pakistan tried to block access to YouTube from within the country by broadcasting a more specific BGP route, it accidentally caused a near-global YouTube blackout, underscoring the vulnerability of BGP to human error (as well as foul play).[2]

The prevalence of these Internet reliability and peering-point problems means that the longer data must travel through the middle mile, the more it is subject to congestion, packet loss, and poor performance. These middle-mile problems are further exacerbated by current trends—most notably the increase in last-mile capacity and demand. Broadband adoption continues to rise, in terms of both penetration and speed, as ISPs invest in last-mile infrastructure. AT&T just spent approximately $6.5 billion to roll out its U-verse service, while Verizon is spending $23 billion to wire 18 million homes with FiOS (Fiber-optic Service) by 2010.[6,7] Comcast also recently announced it plans to offer speeds of up to 100Mbps within a year.[3]

Demand drives this last-mile boom: Pew Internet's 2008 report shows that one-third of U.S. broadband users have chosen to pay more for faster connections.[4] Akamai Technologies' data, shown in Figure 1, reveals that 59% of its global users have broadband connections (with speeds greater than 2 Mbps), and 19% of global users have "high broadband" connections greater than 5Mbps—fast enough to support DVD-quality content.[2] The high-broadband numbers represent a 19% increase in just three months.

Figure 1: Broadband Penetration by Country.

Broadband

Ranking	Country	% > 2 Mbps
—	Global	59%
1	South Korea	90%
2	Belgium	90%
3	Japan	87%
4	Hong Kong	87%
5	Switzerland	85%
6	Slovakia	83%
7	Norway	82%
8	Denmark	79%
9	Netherlands	77%
10	Sweden	75%
…		
20.	United States	71%

Fast Broadband

Ranking	Country	% > 5 Mbps
—	Global	19%
1	South Korea	64%
2	Japan	52%
3	Hong Kong	37%
4	Sweden	32%
5	Belgium	26%
6	United States	26%
7	Romania	22%
8	Netherlands	22%
9	Canada	18%
10	Denmark	18%

Source: Akamai's State of the Internet Report, 02 2008

A Question of Scale

Along with the greater demand and availability of broadband comes a rise in user expectations for faster sites, richer media, and highly interactive applications. The increased traffic loads and performance requirements in turn put greater pressure on the Internet's internal infrastructure—the middle mile. In fact, the fast-rising popularity of video has sparked debate about whether the Internet can scale to meet the demand.

Consider, for example, delivering a TV-quality stream (2Mbps) to an audience of 50 million viewers, approximately the audience size of a popular TV show. The scenario produces aggregate bandwidth requirements of 100Tbps. This is a reasonable vision for the near term—the next two to five years—but it is orders of magnitude larger than the biggest online events today, leading to skepticism about the Internet's ability to handle such demand. Moreover, these numbers are just for a single TV-quality show. If hundreds of millions of end users were to download Blu-ray-quality movies regularly over the Internet, the resulting traffic load would go up by an additional one or two orders of magnitude.

Another interesting side effect of the growth in video and rich media file sizes is that the distance between server and end user becomes critical to end-user performance. This is the result of a somewhat counterintuitive phenomenon that we call the Fat File Paradox: given that data packets can traverse networks at close to the speed of light, why does it takes so long for a "fat file" to cross the country, even if the network is not congested?

It turns out that because of the way the underlying network protocols work, latency and throughput are directly coupled. TCP, for example, allows only small amounts of data to be sent at a time (that is, the TCP window) before having to pause and wait for acknowledgments from the receiving end. This means that throughput is effectively throttled by network round-trip time (latency), which can become the bottleneck for file download speeds and video viewing quality.

Packet loss further complicates the problem, since these protocols back off and send even less data before waiting for acknowledgment if packet loss is detected. Longer distances increase the chance of congestion and packet loss to the further detriment of throughput.

Figure 2 illustrates the effect of distance (between server and end user) on throughput and download times. Five or 10 years ago, dial-up modem speeds would have been the bottleneck on these files, but as we look at the Internet today and into the future, middle-mile distance becomes the bottleneck.

Four Approaches to Content Delivery

Given these bottlenecks and scalability challenges, how does one achieve the levels of performance and reliability required for effective delivery of content and applications over the Internet? There are four main approaches to distributing content servers in a content-delivery architecture: centralized hosting, "big data center" CDNs (content-delivery networks), highly distributed CDNs, and peer-to-peer networks.

*Centralized Hosting.*Traditionally architected Web sites use one or a small number of collocation sites to host content. Commercial-scale sites generally have at least two geographically dispersed mirror locations to provide additional performance (by being closer to different groups of end users), reliability (by providing redundancy), and scalability (through greater capacity).

This approach is a good start, and for small sites catering to a localized audience it may be enough. The performance and reliability fall short of expectations for commercial-grade sites and applications, however, as the end-user experience is at the mercy of the unreliable Internet and its middle-mile bottlenecks.

There are other challenges as well: site mirroring is complex and costly, as is managing capacity. Traffic levels fluctuate tremendously, so the need to provision for peak traffic levels means that expensive infrastructure will sit underutilized most of the time. In addition, accurately predicting traffic demand is extremely difficult, and a centralized hosting model does not provide the flexibility to handle unexpected surges.

Figure 2: Effect of Distance on Throughput and Download Times.

Distance from Server to User	Network Latency	Typical Packet Loss	Throughput (quality)	4GB DVD Download Time
Local: <100 mi.	1.6 ms	0.6%	44 Mbs (HDTV)	12 min.
Regional: 500–1,000 mi.	16 ms	0.7%	4 Mbs (not quite DVD)	2.2 hrs.
Cross-continent: ~3,000 mi.	48 ms	1.0%	1 Mbs (not quite TV)	8.2 hrs.
Multi-continent: ~6,000 mi.	96 ms	1.4%	0.4 Mbs (poor)	20 hrs.

"Big Data Center" CDNs. Content-delivery networks offer improved scalability by offloading the delivery of cacheable content from the origin server onto a larger, shared network. One common CDN approach can be described as "big data center" architecture—caching and delivering customer content from perhaps a couple dozen high-capacity data centers connected to major backbones.

Although this approach offers some performance benefit and economies of scale over centralized hosting, the potential improvements are limited because the CDN's servers are still far away from most users and still deliver content from the wrong side of the middle-mile bottlenecks.

It may seem counterintuitive that having a presence in a couple dozen major backbones isn't enough to achieve commercial-grade performance. In fact, even the largest of those networks controls very little end-user access traffic. For example, the top 30 networks *combined* deliver only 50% of end-user traffic, and it drops off quickly from there, with a very long tail distribution over the Internet's 13,000 networks. Even with connectivity to all the biggest backbones, data must travel through the morass of the middle mile to reach most of the Internet's 1.4 billion users.

A quick back-of-the-envelope calculation shows that this type of architecture hits a wall in terms of scalability as we move toward a video world. Consider a generous forward projection on such an architecture—say, 50 high-capacity data centers, each with 30 outbound connections, 10Gbps each. This gives an upper bound of 15Tbps total capacity for this type of network, far short of the 100Tbps needed to support video in the near term.

Highly Distributed CDNs. Another approach to content delivery is to leverage a very highly distributed network—one with servers in thousands of networks, rather than dozens. On the surface, this architecture may appear quite similar to the "big data center" CDN. In reality, however, it is a fundamentally different approach to content-server placement, with a difference of two orders of magnitude in the degree of distribution.

By putting servers within end-user ISPs, for example, a highly distributed CDN delivers content from the right side of the middle-mile bottlenecks, eliminating peering, connectivity, routing, and distance problems, and reducing the number of Internet components depended on for success.

Moreover, this architecture scales. It can achieve a capacity of 100Tbps, for example, with deployments of 20 servers, each capable of delivering 1Gbps in 5,000 edge locations.

On the other hand, deploying a highly distributed CDN is costly and time consuming, and comes with its own set of challenges. Fundamentally, the network must be designed to scale efficiently from a deployment and management perspective. This necessitates development of a number of technologies, including:

- Sophisticated global-scheduling, mapping, and load-balancing algorithms
- Distributed control protocols and reliable automated monitoring and alerting systems
- Intelligent and automated failover and recovery methods
- Colossal-scale data aggregation and distribution technologies (designed to handle different trade-offs between timeliness and accuracy or completeness)
- Robust global software-deployment mechanisms
- Distributed content freshness, integrity, and management systems
- Sophisticated cache-management protocols to ensure high cache-hit ratios

These are nontrivial challenges, and we present some of our approaches later on in this article.

Peer-to-Peer Networks. Because a highly distributed architecture is critical to achieving scalability and performance in video distribution, it is natural to consider a P2P (peer-to-peer) architecture. P2P can be thought of as taking the distributed architecture to its logical extreme, theoretically providing nearly infinite scalability. Moreover, P2P offers attractive economics under current network pricing structures.

As Web sites become increasingly dynamic, personalized, and application-driven, however, the ability to accelerate uncacheable **content becomes equally critical to delivering a strong end-user experience.**

In reality, however, P2P faces some serious limitations, most notably because the total download capacity of a P2P network is throttled by its total uplink capacity. Unfortunately, for consumer broadband connections, uplink speeds tend to be much lower than downlink speeds: Comcast's standard high-speed Internet package, for example, offers 6Mbps for download but only 384Kbps for upload (one-sixteenth of download throughput).

This means that in situations such as live streaming where the number of uploaders (peers sharing content) is limited by the number of downloaders (peers requesting content), average download throughput is equivalent to the average uplink throughput and thus cannot support even mediocre Web-quality streams. Similarly, P2P fails in "flash crowd" scenarios where there is a sudden, sharp increase in demand, and the number of downloaders greatly outstrips the capacity of uploaders in the network.

Somewhat better results can be achieved with a hybrid approach, leveraging P2P as an extension of a distributed delivery network. In particular, P2P can help reduce overall distribution costs in certain situations. Because the capacity of the P2P network is limited, however, the architecture of the non-P2P portion of the network still governs overall performance and scalability.

Each of these four network architectures has its trade-offs, but ultimately, for delivering rich media to a global Web audience, a highly distributed architecture provides the only robust solution for delivering commercial-grade performance, reliability, and scale.

Application Acceleration

Historically, content-delivery solutions have focused on the offloading and delivery of static content, and thus far we have focused our conversation on the same. As Web sites become increasingly dynamic, personalized, and application-driven, however, the ability to accelerate *uncacheable* content becomes equally critical to delivering a strong end-user experience.

Ajax, Flash, and other RIA (rich Internet application) technologies work to enhance Web application responsiveness on the browser side, but ultimately, these types of applications all still require significant numbers of round-trips back to the origin server. This makes them highly susceptible to all the bottlenecks I've mentioned before: peering-point congestion, network latency, poor routing, and Internet outages.

Speeding up these round-trips is a complex problem, but many optimizations are made possible by using a highly distributed infrastructure.

Optimization 1: Reduce transport-layer overhead. Architected for reliability over efficiency, protocols such as TCP have substantial overhead. They require multiple round-trips (between the two communicating parties) to set up connections, use a slow initial rate of data exchange, and recover slowly from packet loss. In contrast, a network that uses persistent connections and optimizes parameters for efficiency (given knowledge of current network conditions) can significantly improve performance by reducing the number of round-trips needed to deliver the same set of data.

Optimization 2: Find better routes. In addition to reducing the number of round-trips needed, we would also like to reduce the time needed for each round-trip—each journey across the Internet. At first blush, this does not seem possible. All Internet data must be routed by BGP and must travel over numerous autonomous networks.

BGP is simple and scalable but not very efficient or robust. By leveraging a highly distributed network—one that offers potential intermediary servers on many different networks—you can actually speed up uncacheable communications by 30% to 50% or more, by using routes that are faster and much less congested. You can also achieve much greater communications reliability by finding alternate routes when the default routes break.

Optimization 3: Prefetch embedded content. You can do a number of additional things at the application layer to improve Web application responsiveness for end users. One is to prefetch embedded content: while an edge server is delivering an HTML page to an end user, it can also parse the HTML and retrieve all embedded content *before* it is requested by the end user's browser.

The real growth in bandwidth-intensive Web content, rich media, and Web- and IP-based applications is just beginning. The challenges presented by this growth are many.

The effectiveness of this optimization relies on having servers near end users, so that users perceive a level of application responsiveness akin to that of an application being delivered directly from a nearby server, even though, in fact, some of the embedded content is being fetched from the origin server across the long-haul Internet. Prefetching by forward caches, for example, does not provide this performance benefit because the prefetched content must still travel over the middle mile before reaching the end user. Also, note that unlike link prefetching (which can also be done), embedded content prefetching does not expend extra bandwidth resources and does not request extraneous objects that may not be requested by the end user.

With current trends toward highly personalized applications and user-generated content, there's been growth in either uncacheable or long-tail (that is, not likely to be in cache) embedded content. In these situations, prefetching makes a huge difference in the user-perceived responsiveness of a Web application.

Optimization 4: Assemble pages at the edge. The next three optimizations involve reducing the amount of content that needs to travel over the middle mile. One approach is to cache page fragments at edge servers and dynamically assemble them at the edge in response to end-user requests. Pages can be personalized (at the edge) based on characteristics including the end user's location, connection speed, cookie values, and so forth. Assembling the page at the edge not only offloads the origin server, but also results in much lower latency to the end user, as the middle mile is avoided.

Optimization 5: Use compression and delta encoding. Compression of HTML and other text-based components can reduce the amount of content traveling over the middle mile to one-tenth of the original size. The use of delta encoding, where a server sends only the *difference* between a cached HTML page and a dynamically generated version, can also greatly cut down on the amount of content that must travel over the long-haul Internet.

While these techniques are part of the HTTP/1.1 specification, browser support is unreliable. By using a highly distributed network that controls both endpoints of the middle mile, compression and delta encoding can be successfully employed regardless of the browser. In this case, performance is improved because very little data travels over the middle mile. The edge server then decompresses the content or applies the delta encoding and delivers the complete, correct content to the end user.

Optimization 6: Offload computations to the edge. The ability to distribute applications to edge servers provides the ultimate in application performance and scalability. Akamai's network enables distribution of J2EE applications to edge servers that create virtual application instances on demand, as needed. As with edge page assembly, edge computation enables complete origin server offloading, resulting in tremendous scalability and extremely low application latency for the end user.

While not every type of application is an ideal candidate for edge computation, large classes of popular applications—such as contests, product catalogs, store locators, surveys, product configurators, games, and the like—are well suited for edge computation.

Putting it All Together

Many of these techniques require a highly distributed network. Route optimization, as mentioned, depends on the availability of a vast overlay network that includes machines on many different networks. Other optimizations such as prefetching and page assembly are most effective if the delivering server is near the end user. Finally, many transport and application-layer optimizations require bi-nodal connections within the network (that is, you control both endpoints). To maximize the effect of this optimized connection, the endpoints should be as close as possible to the origin server and the end user.

Note also that these optimizations work in synergy. TCP overhead is in large part a result of a conservative approach that guarantees reliability in the face of unknown network conditions. Because route optimization gives us high-performance, congestion-free paths, it allows for a much more aggressive and efficient approach to transport-layer optimizations.

Highly Distributed Network Design

It was briefly mentioned earlier that building and managing a robust, highly distributed network is not trivial. At Akamai, we sought to build a system with extremely high reliability—no downtime, ever—and yet scalable enough to be managed by a relatively small operations staff, despite operating in a highly heterogeneous and unreliable environment. Here are some insights into the design methodology.

The fundamental assumption behind Akamai's design philosophy is that a significant number of component or other failures are occurring at all times in the network. Internet systems present numerous failure modes, such as machine failure, data-center failure, connectivity failure, software failure, and network failure—all occurring with greater frequency than one might think. As mentioned earlier, for example, there are many causes of large-scale network outages—including peering problems, transoceanic cable cuts, and major virus attacks.

Designing a scalable system that works under these conditions means embracing the failures as natural and expected events. The network should continue to work seamlessly despite these occurrences. We have identified some practical design principles that result from this philosophy, which we share here.[1]

Principle 1: Ensure significant redundancy in all systems to facilitate failover. Although this may seem obvious and simple in theory, it can be challenging in practice. Having a highly distributed network enables a great deal of redundancy, with multiple backup possibilities ready to take over if a component fails. To ensure robustness of all systems, however, you will likely need to work around the constraints of existing protocols and interactions with third-party software, as well as balancing trade-offs involving cost.

For example, the Akamai network relies heavily on DNS (Domain Name System), which has some built-in constraints that affect reliability. One example is DNS's restriction on the size of responses, which limits the number of IP addresses that we can return to a relatively static set of 13. The Generic Top Level Domain servers, which supply the critical answers to akamai.net queries, required more reliability, so we took several steps, including the use of IP Anycast.

We also designed our system to take into account DNS's use of TTLs (time to live) to fix resolutions for a period of time. Though the efficiency gained through TTL use is important, we need to make sure users aren't being sent to servers based on stale data. Our approach is to use a two-tier DNS—employing longer TTLs at a global level and shorter TTLs at a local level— allowing less of a trade-off between DNS efficiency and responsiveness to changing conditions. In addition, we have built in appropriate failover mechanisms at each level.

Principle 2: Use software logic to provide message reliability. This design principle speaks directly to scalability. Rather than building dedicated links between data centers, we use the public Internet to distribute data—including control messages, configurations, monitoring information, and customer content—throughout our network. We improve on the performance of existing Internet protocols—for example, by using multirouting and limited retransmissions with UDP (User Datagram Protocol) to achieve reliability without sacrificing latency. We also use software to route data through intermediary servers to ensure communications (as described in Optimization 2), even when major disruptions (such as cable cuts) occur.

Principle 3: Use distributed control for coordination. Again, this principle is important both for fault tolerance and scalability. One practical example is the use of leader election, where leadership evaluation can depend on many factors including machine status, connectivity to other machines in the network, and monitoring capabilities. When connectivity of a local lead server degrades, for example, a new server is automatically elected to assume the role of leader.

Principle 4: Fail cleanly and restart. Based on the previous principles, the network has already been architected to handle server failures quickly and seamlessly, so we are able to take a more aggressive approach to failing problematic servers and restarting them from a last known good state. This sharply reduces the risk of operating in a potentially corrupted state. If a given machine continues to require restarting, we simply put it into a "long sleep" mode to minimize impact to the overall network.

Principle 5: Phase software releases. After passing the quality assurance (QA) process, software is released to the live network in phases. It is first deployed to a single machine. Then, after performing the appropriate checks, it is deployed to a single region, then possibly to additional subsets of the network, and finally to the entire network. The nature of the release dictates how many phases and how long each one lasts. The previous principles, particularly use of redundancy, distributed control, and aggressive restarts, make it possible to deploy software releases frequently and safely using this phased approach.

Principle 6: Notice and proactively quarantine faults. The ability to isolate faults, particularly in a recovery-oriented computing system, is perhaps one of the most challenging problems and an area of important ongoing research. Here is one example. Consider a hypothetical situation where requests for a certain piece of content with a rare set of configuration parameters trigger a latent bug. Automatically failing the servers affected is not enough, as requests for this content will then be directed to other machines, spreading the problem. To solve this problem, our caching algorithms constrain each set of content to certain servers so as to limit the spread of fatal requests. In general, no single customer's content footprint should dominate any other customer's footprint among available servers. These constraints are dynamically determined based on current levels of demand for the content, while keeping the network safe.

Practical Results and Benefits

Besides the inherent fault-tolerance benefits, a system designed around these principles offers numerous other benefits.

Faster software rollouts. Because the network absorbs machine and regional failures without impact, Akamai is able to safely but aggressively roll out new software using the phased rollout approach. As a benchmark, we have historically implemented approximately 22 software releases and 1,000 customer configuration releases per month to our worldwide network, without disrupting our always-on services.

Minimal operations overhead. A large, highly distributed, Internet-based network can be very difficult to maintain, given its sheer size, number of network partners, heterogeneous nature, and diversity of geographies, time zones, and languages. Because the Akamai network design is based on the assumption that components will fail, however, our operations team does not need to be concerned about most failures. In addition, the team can aggressively suspend machines or data centers if it sees any slightly worrisome behavior. There is no need to rush to get components back online right away, as the network absorbs the component failures without impact to overall service.

This means that at any given time, it takes only eight to 12 operations staff members, on average, to manage our network of approximately 40,000 devices (consisting of more than 35,000 servers plus switches and other networking hardware). Even at peak times, we successfully manage this global, highly distributed network with fewer than 20 staff members.

Lower costs, easier to scale. In addition to the minimal operational staff needed to manage such a large network, this design philosophy has had several implications that have led to reduced costs and improved scalability. For example, we use commodity hardware instead of more expensive, more reliable servers. We deploy in third-party data centers instead of having our own. We use the public Internet instead of having dedicated links. We deploy in greater numbers of smaller regions—many of which host our servers for free—rather than in fewer, larger, more "reliable" data centers where congestion can be greatest.

Conclusion

Even though we've seen dramatic advances in the ubiquity and usefulness of the Internet over the past decade, the real growth in bandwidth-intensive Web content, rich media, and Web- and IP-based applications is just beginning. The challenges presented by this growth are many: as businesses move more of their critical functions online, and as consumer entertainment (games, movies, sports) shifts to the Internet from other broadcast media, the stresses placed on the Internet's middle mile will become increasingly apparent and detrimental. As such, we believe the issues raised in this article and the benefits of a highly distributed approach to content delivery will only grow in importance as we collectively work to enable the Internet to scale to the requirements of the next generation of users. C

References

1. Afergan, M., Wein, J., LaMeyer, A. Experience with some principles for building an Internet-scale reliable system. In *Proceedings of the 2nd Conference on Real, Large Distributed Systems 2.* (These principles are laid out in more detail in this 2005 research paper.)
2. Akamai Report: The State of the Internet, 2nd quarter, 2008; http://www.akamai.com/stateoftheinternet/. (These and other recent Internet reliability events are discussed in Akamai's quarterly report.)
3. Anderson, N. Comcast at CES: 100 Mbps connections coming this year. ars technica (Jan. 8, 2008); http://arstechnica.com/news.ars/post/20080108-comcast-100mbps-connections-coming-this-year.html.
4. Horrigan, J.B. Home Broadband Adoption 2008. Pew Internet and American Life Project; http://www.pewinternet.org/pdfs/PIP_Broadband_2008.pdf.
5. Internet World Statistics. Broadband Internet Statistics: Top World Countries with Highest Internet Broadband Subscribers in 2007; http://www.internetworldstats.com/dsl.htm.
6. Mehta, S. Verizon's big bet on fiber optics. *Fortune* (Feb. 22, 2007); http://money.cnn.com/magazines/fortune/fortune_archive/2007/03/05/8401289/.
7. Spangler T. AT&T: U-verse TV spending to increase. *Multichannel News* (May 8, 2007); http://www.multichannel.com/article/CA6440129.html.
8. TeleGeography. Cable cuts disrupt Internet in Middle East and India. *CommsUpdate* (Jan. 31, 2008); http://www.telegeography.com/cu/article.php?article_id=21528.

Tom Leighton co-founded Akamai Technologies in August 1998. Serving as chief scientist and as a director to the board, he is Akamai's technology visionary, as well as a key member of the executive committee setting the company's direction. He is an authority on algorithms for network applications. Leighton is a Fellow of the American Academy of Arts and Sciences, the National Academy of Science, and the National Academy of Engineering.

A previous version of this article appeared in the October 2008 issue of *ACM Queue* magazine.