Realtime publishers

The Definitive Guide to Cloud Acceleration

Dan Sullivan

sponsored by

CDNetworks

CDNetworks
Global Cloud Acceleration

Accelerate your web performance across the globe

- Dynamic Web, Cloud Application, and Content Acceleration
- Reach 99% of the World in Milliseconds
- Only Global CDN with PoPs in Mainland China
- Multiple PoPs in Russia, India, Brazil and Emerging Markets

www.cdnetworks.com

Copyright Statement

Chapter 2: How Websites and Web Applications Work

To understand the challenges to optimizing your Web sites and Web applications, it helps to start with basics on how Web sites and Web applications work. This chapter will consider differences between static and dynamic content and techniques for optimizing each type of content. It will also discuss the limitations of these techniques.

Consider a simple example: If you were to type a URL into your browser, such as www.example.com, you would start a series of events that ultimately leads to having a Web page rendered in your browser. These events include:

- Sending the domain to a domain name service (DNS) server to map the URL to an IP address
- Routing the Web page request, possibly over a series of different Internet providers, to the Web server at the IP address provided by the DNS server
- Retrieving or generating the content of the Web page from the Web server
- Packaging the content of the Web page into a series of TCP packets that deliver the content to the client device that is making the Web page request
- Reconstructing the Web page from packets on the client device
- Rendering the Web page for the user

Retrieving a Web page might sound like a simple operation, but there are clearly multiple steps involved and each of them can introduce delays into the process.

If a DNS server is unreachable or slow to respond, the client device making the request might need to query a different DNS server. This requirement delays the start of the process to retrieve the Web page.

Traffic that moves between Internet service providers (ISPs) might be subject to congestion when routing between services or to limits on the speed with which traffic from competing ISPs is handled. This potential bottleneck can delay the transmission of data between the client device and the Web server.

The Web server itself can be the cause of delays. If the Web server is subject to heavy loads, there might be a long queue of requests waiting to be processed. If Web page requests require a substantial number of I/O operations on the Web server, the response time will be longer than if those operations could be avoided.

The way the TCP protocol functions is even a potential cause of delays. TCP guarantees delivery of packets. To meet this guarantee, TCP requires more communication steps to ensure packets are delivered. If some packets are lost or delayed, they must be retransmitted. The need for retransmitting and the delays it can introduce might be minimal on local area networks (LANs), but moving data across global-scale networks will more likely entail some lost packets. You must also consider the latency of long distance networks. The physical limitations of network speed combined with network traffic and ISP polices all influence the time required to transmit data between a client device and a Web server. The speed of the last-mile connection to the client device and the configuration of the client device can also influence the speed with which Web content is rendered.

The remainder of this chapter will delve into potential problems in more detail, in particular:

- Static content types
- Dynamic content types
- Targeted content delivery

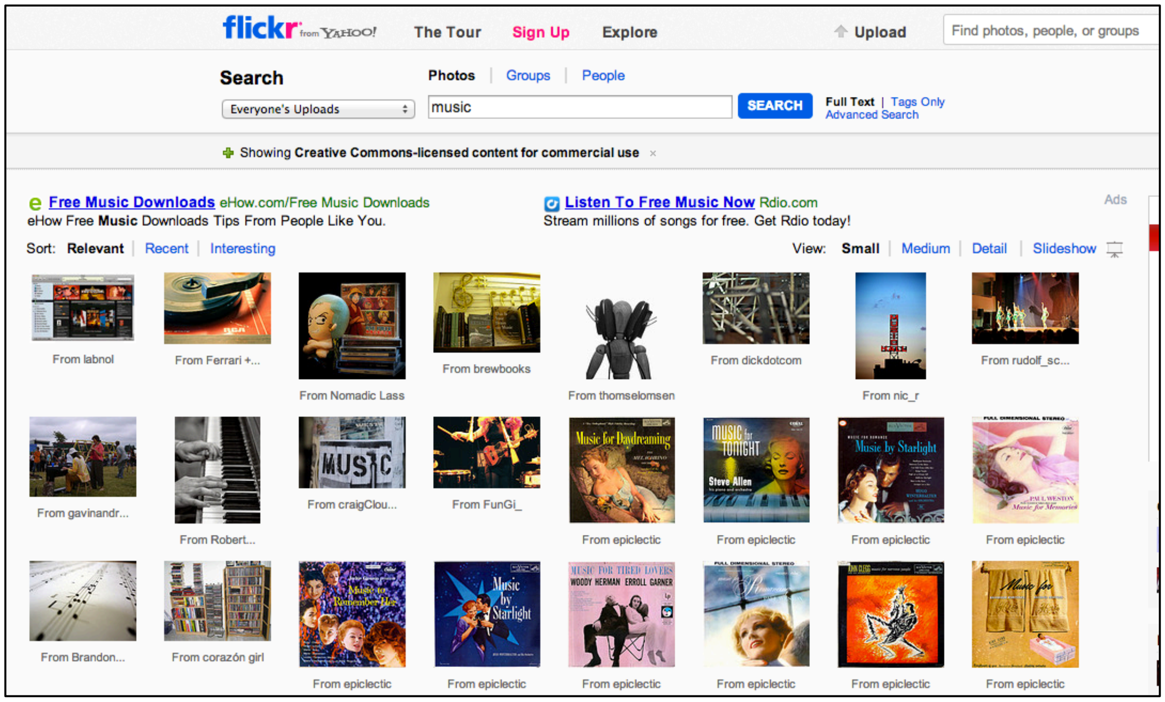
In each case, the goal will be to understand aspects of Web application design and architecture that adversely affect performance, then examine methods for addressing those adverse effects.

Optimizing the Delivery of Static Content

When the Web was created in the early 1990s, the primary purpose was to link static content across a network of servers. Tim Berners-Lee wrote the first Web browser and Web server and effectively brought hypertext to the Internet. The key components were URLs to describe Web site addresses, Hypertext Markup Language (HTML) to create Web pages with static content, and Hypertext Transfer Protocol (HTTP) for requesting and delivering content to devices.

Multiple Object Static Web Pages

It was only a matter of time before linking text documents led to more complex document configurations. It is commonplace now for Web pages to contain text, images, and multimedia content (see Figure 2.1).



Free Music Downloads eHow.com/Free Music Downloads
eHow Free **Music** Downloads Tips From People Like You.

Listen To Free Music Now Rdio.com
Stream millions of songs for free. Get Rdio today!

Figure 2.1: Web pages routinely combine text, images, video, and sound.

Web pages constructed of multiple types of content are typically built from components stored in multiple files. For example, the Web page depicted in Figure 2.1 includes 24 images. Those images are each stored in separate files on a Web server.

When the page is rendered on a client device, the client issues an HTTP GET command to download each image. Each GET command requires a connection to the Web server and I/O operations on the server to retrieve the image from persistent storage.

Note

Actually, the Web server might not have to retrieve the image from disk or other persistent storage if it is cached. I will discuss that scenario shortly.

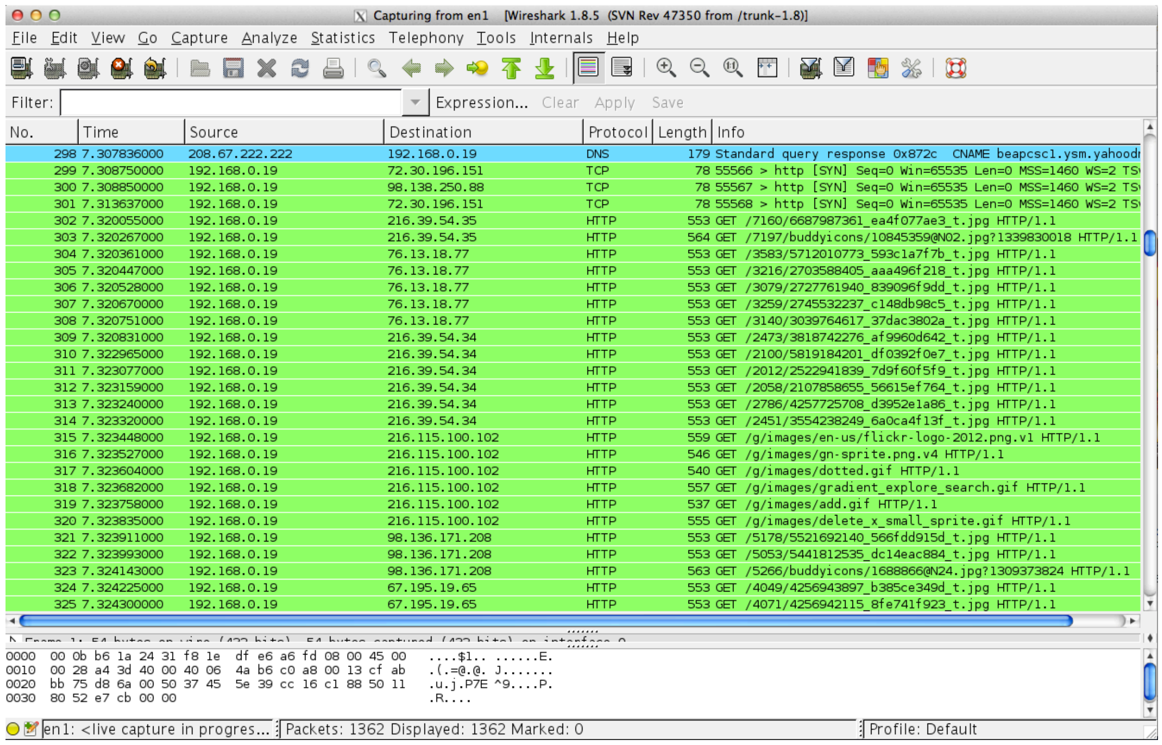
Once the server has retrieved the content for each of the 24 images, the Web server transmits each image to the client device. Each transmission is subject to some of the potential problems described earlier in the chapter, including congestion and lost packets. These problems are especially problematic for Web pages that require all or most of the content on page to be rendered before the page is useful.

Consider a Web site offering weather information. A page that allows a user to enter a location to check today's weather would be functional as soon as the search box and related text are rendered. Other images, such as ad displays, could load while the user enters text into the search box; there is little or no adverse effect on the user. If, however, a user loads a page with multiple maps and satellite images, the user might have to wait for multiple images to load to find the specific information they are seeking.

When a Web page has multiple objects that require separate connections, the number of connections required determines the time required to fully load a page. The speed of a download is determined by three factors:

- Bandwidth
- Latency
- Packet loss

Each of these factors should be taken into account when analyzing Web page load times (see Figure 2.2).


Capturing from en1 [Wireshark 1.8.5 (SVN Rev 47350 from /trunk-1.8)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
298	7.307836000	208.67.222.222	192.168.0.19	DNS	179	Standard query response 0x872c CNAME beapcsc1.ysm.yahoodr
299	7.308750000	192.168.0.19	72.30.196.151	TCP	78	55566 > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=2 TS
300	7.308850000	192.168.0.19	98.138.250.88	TCP	78	55567 > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=2 TS
301	7.313637000	192.168.0.19	72.30.196.151	TCP	78	55568 > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=2 TS
302	7.320055000	192.168.0.19	216.39.54.35	HTTP	553	GET /7160/6687987361_ea4f077ae3_t.jpg HTTP/1.1
303	7.320267000	192.168.0.19	216.39.54.35	HTTP	564	GET /7197/buddyicons/10845359@N02.jpg?1339830018 HTTP/1.1
304	7.320361000	192.168.0.19	76.13.18.77	HTTP	553	GET /3583/5712010773_593c1a7f7b_t.jpg HTTP/1.1
305	7.320447000	192.168.0.19	76.13.18.77	HTTP	553	GET /3216/2703588405_aaa496f218_t.jpg HTTP/1.1
306	7.320528000	192.168.0.19	76.13.18.77	HTTP	553	GET /3079/2727761940_839096f9dd_t.jpg HTTP/1.1
307	7.320670000	192.168.0.19	76.13.18.77	HTTP	553	GET /3259/2745532237_c148db98c5_t.jpg HTTP/1.1
308	7.320751000	192.168.0.19	76.13.18.77	HTTP	553	GET /3140/3039764617_37dac3802a_t.jpg HTTP/1.1
309	7.320831000	192.168.0.19	216.39.54.34	HTTP	553	GET /2473/3818742276_af9960d642_t.jpg HTTP/1.1
310	7.322965000	192.168.0.19	216.39.54.34	HTTP	553	GET /2100/5819184201_df0392f0e7_t.jpg HTTP/1.1
311	7.323077000	192.168.0.19	216.39.54.34	HTTP	553	GET /2012/2522941839_7d9f60f5f9_t.jpg HTTP/1.1
312	7.323159000	192.168.0.19	216.39.54.34	HTTP	553	GET /2058/2107858655_56615ef764_t.jpg HTTP/1.1
313	7.323240000	192.168.0.19	216.39.54.34	HTTP	553	GET /2786/4257725708_d3952e1a86_t.jpg HTTP/1.1
314	7.323320000	192.168.0.19	216.39.54.34	HTTP	553	GET /2451/3554238249_6a0ca4f13f_t.jpg HTTP/1.1
315	7.323448000	192.168.0.19	216.115.100.102	HTTP	559	GET /g/images/en-us/flickr-logo-2012.png.v1 HTTP/1.1
316	7.323527000	192.168.0.19	216.115.100.102	HTTP	546	GET /g/images/gn-sprite.png.v4 HTTP/1.1
317	7.323604000	192.168.0.19	216.115.100.102	HTTP	540	GET /g/images/dotted.gif HTTP/1.1
318	7.323682000	192.168.0.19	216.115.100.102	HTTP	557	GET /g/images/gradient_explore_search.gif HTTP/1.1
319	7.323758000	192.168.0.19	216.115.100.102	HTTP	537	GET /g/images/add.gif HTTP/1.1
320	7.323835000	192.168.0.19	216.115.100.102	HTTP	555	GET /g/images/delete_x_small_sprite.gif HTTP/1.1
321	7.323911000	192.168.0.19	98.136.171.208	HTTP	553	GET /5178/5521692140_566fdd915d_t.jpg HTTP/1.1
322	7.323993000	192.168.0.19	98.136.171.208	HTTP	553	GET /5053/5441812535_dc14eac884_t.jpg HTTP/1.1
323	7.324143000	192.168.0.19	98.136.171.208	HTTP	563	GET /5266/buddyicons/1688866@N24.jpg?1309373824 HTTP/1.1
324	7.324225000	192.168.0.19	67.195.19.65	HTTP	553	GET /4049/4256943897_b385ce349d_t.jpg HTTP/1.1
325	7.324300000	192.168.0.19	67.195.19.65	HTTP	553	GET /4071/4256942115_8fe741f923_t.jpg HTTP/1.1

```
0000  00 0b b6 1a 24 31 f8 1e  df e6 a6 fd 08 00 45 00   ....$1.. ......E.
0010  00 28 a4 3d 40 00 40 06  4a b6 c0 a8 00 13 cf ab   .(.=@.@. J.......
0020  bb 75 d8 6a 00 50 37 45  5e 39 cc 16 c1 88 50 11   .u.j.P7E ^9....P.
0030  80 52 e7 cb 00 00                                   .R....
```

en1: <live capture in progres... | Packets: 1362 Displayed: 1362 Marked: 0 | Profile: Default

Figure 2.2: Loading a single Web page can require multiple files and therefore multiple connections between a client and a server (Screenshot of Wireshark Version 1.8.5).

Determining Web Page Load Times

Bandwidth is a measure of the amount of data that can be transmitted over a network in a given amount of time. Network bandwidths vary widely. 10 gigabit Ethernet can be found in enterprises while tens of megabit bandwidths are common in residential services. Bandwidth is only one of three factors influencing the time required to download a Web page, so increasing bandwidth may not improve overall performance.

Latency is a measure of the amount of time required to send a packet of data from a source to a destination and then to send a packet from the destination back to the source. Latency is influenced by a number of factors, including: network traffic, the distance packets of data must travel, and network device configurations.

Note

Latency is also known as round trip time (RTT) to distinguish it from one-way latency, which is a measure of time required to send a packet between two devices on a network.

Network traffic can adversely affect latency if the volume of incoming traffic is more than network devices can process at the time. When buffers are full, a device may send a single packet to the source system to stop transmitting data. This situation can increase the RTT on that connection.

The greater the physical distance a data packet must travel, the longer it will take. In addition, longer distances usually entail additional network devices, some of which could be significantly slower than others. This setup can introduce bottlenecks in the network that limit the overall speed to the speed of the slowest segment, as Figure 2.3 illustrates.

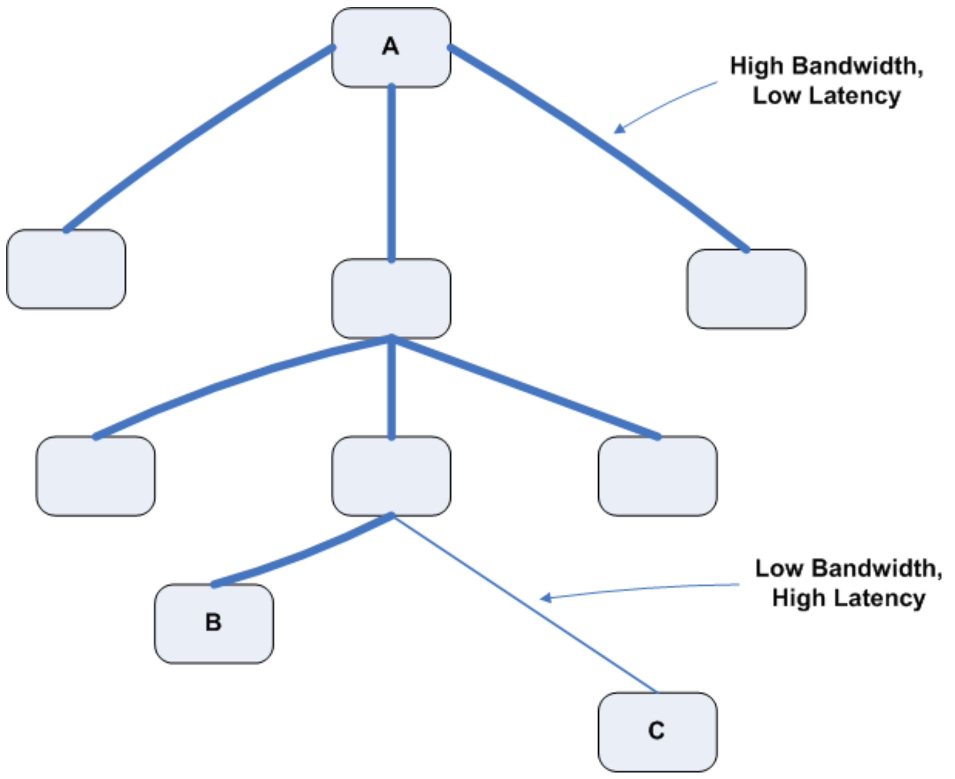


Figure 2.3: Bottlenecks in the network can slow data transmission to the speed of the slowest segment of the network.

Changes in TCP protocol configurations can affect the performance of a network. For example, high-bandwidth networks might not be taking full advantage of the bandwidth available if devices on the network have TCP protocols configured to send smaller amounts and then wait for an acknowledgment before sending more data.

Packet loss is another problem sometimes related to device configuration (see Figure 2.4). Devices that receive data from a network connection have to buffer data so that the data can be processed by an application. If the buffers become full while incoming data continues to arrive, then packets can be lost. The TCP protocol is designed to guarantee delivery of packets. Devices that send packets to a receiving device expect an acknowledgement that packets have been received. Sending devices are configured to wait a certain period of time, and if no acknowledgment is received in that time, the sender retransmits the unacknowledged packets.

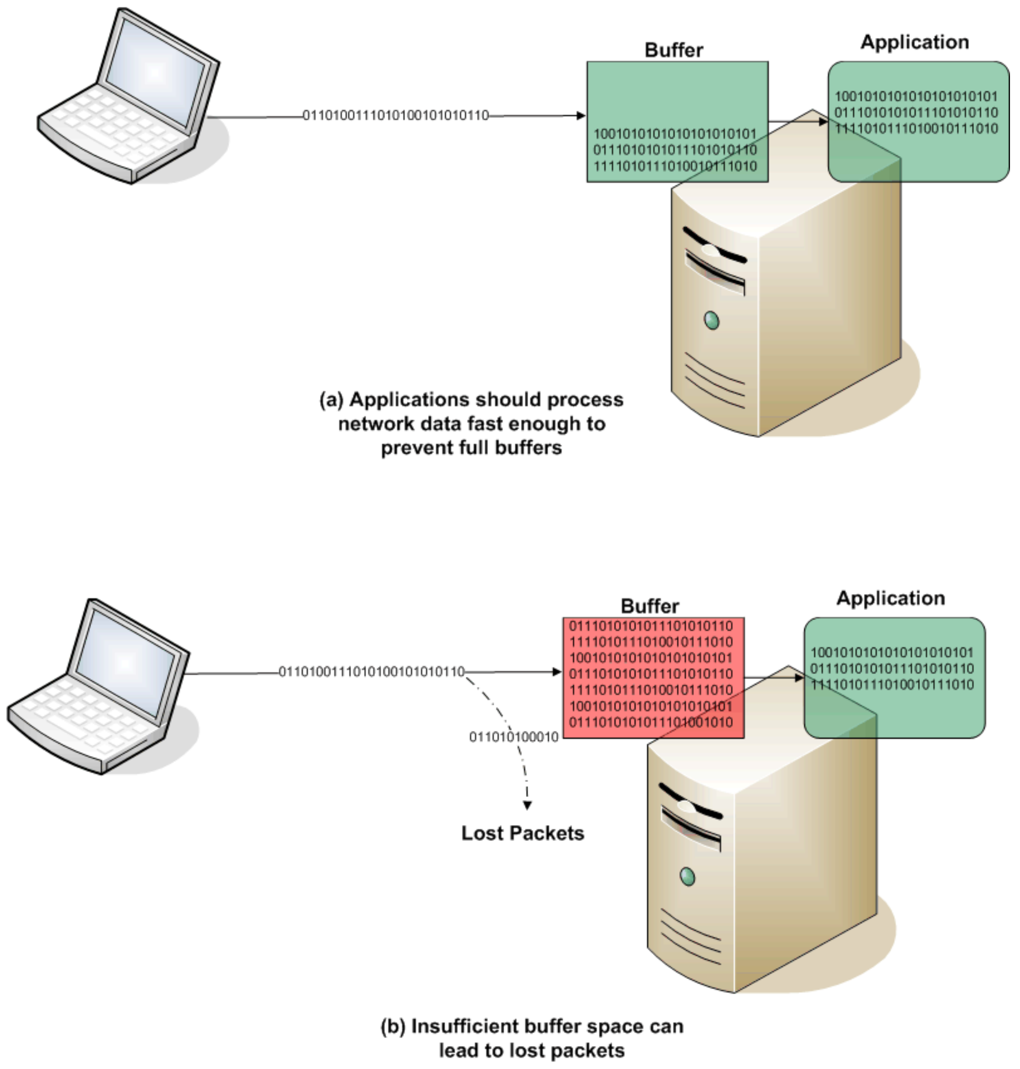

Buffer
Application
0110100111010100101010110
(a) Applications should process network data fast enough to prevent full buffers
Buffer
Application
0110100111010100101010110
011010100010
Lost Packets
(b) Insufficient buffer space can lead to lost packets

Figure 2.4: When applications cannot read from the connection's data buffer fast enough to keep up with incoming traffic, then packets can be lost. Data loss can occur for similar reasons when there is congestion on network devices between the source and destination devices.

Assessing the Scope of the Problem

The problem of slow page loading times is a product of multiple factors, including the amount of static content, network bandwidth, latency, and packet loss. Consider each factor as you try to develop a solution for slow page loading times.

Amount of Static Content

The amount of static content is driven by the business requirements that lead to the development of the Web site. A product catalog needs to provide sufficient information for customers to understand the features and value of a product and distinguish it from competitors' products. This need might lead Web site designers to include multiple images of each product in a variety of configurations. Asking a Web designer to limit the amount of content on a page to improve page load speeds is an option of last resort. You should assume the content is there for a business reason and should remain.

Distance Between Source and Destination

Bandwidth and packet loss are two network factors that are only partially in your control. Your business will have control of internal networks and enterprise WANs, but once your content is transmitted over the Internet, you are depending on the network infrastructure provided by and managed by others.

If your content is routed over an ISP network with significant congestion and packet loss, network protocols may be able to re-route traffic automatically, but you are still depending on infrastructure outside of your control. As a result, your control over the speed with which your static Web content loads is limited. The more you depend on ISPs' infrastructure, the greater the chance of encountering network problems that you have limited ability to control.

Popularity of Content

In addition to network performance problems, the speed at which you can deliver content is also influenced by the speed of Web servers. When large numbers of users are requesting content from a Web server, there might be contention and congestion for resources. For example, if every request for content results in multiple I/O operations to retrieve content from disk, the performance level of the disk will be a limiting factor in the responsiveness of your Web site. Retrieving content from disk is slower than retrieving content from RAM. One way to improve response time is to store a copy of content in RAM. This method is one type of caching technique that can improve Web site performance.

Before delving into the different types of caching, it is important to note that the benefits from all forms of caching are a function of the popularity of content. The reason is that when a piece of content is retrieved from persistent storage, a copy is saved in the cache. The next time that content is requested, it can be retrieved from the cache more quickly than it could be retrieved from disk.

Very low-popularity content is far less likely to be cached than is popular content. As a result, popular content can be delivered faster than unpopular content can. This reality is certainly better than consistently slow page load times, but it can lead to inconsistent performance in a user's experience with your Web site.

Distributed Caching and Persistent Storage

There are different ways to cache content. There are at least three types of caching that are used to improve Web application and Web site responsiveness:

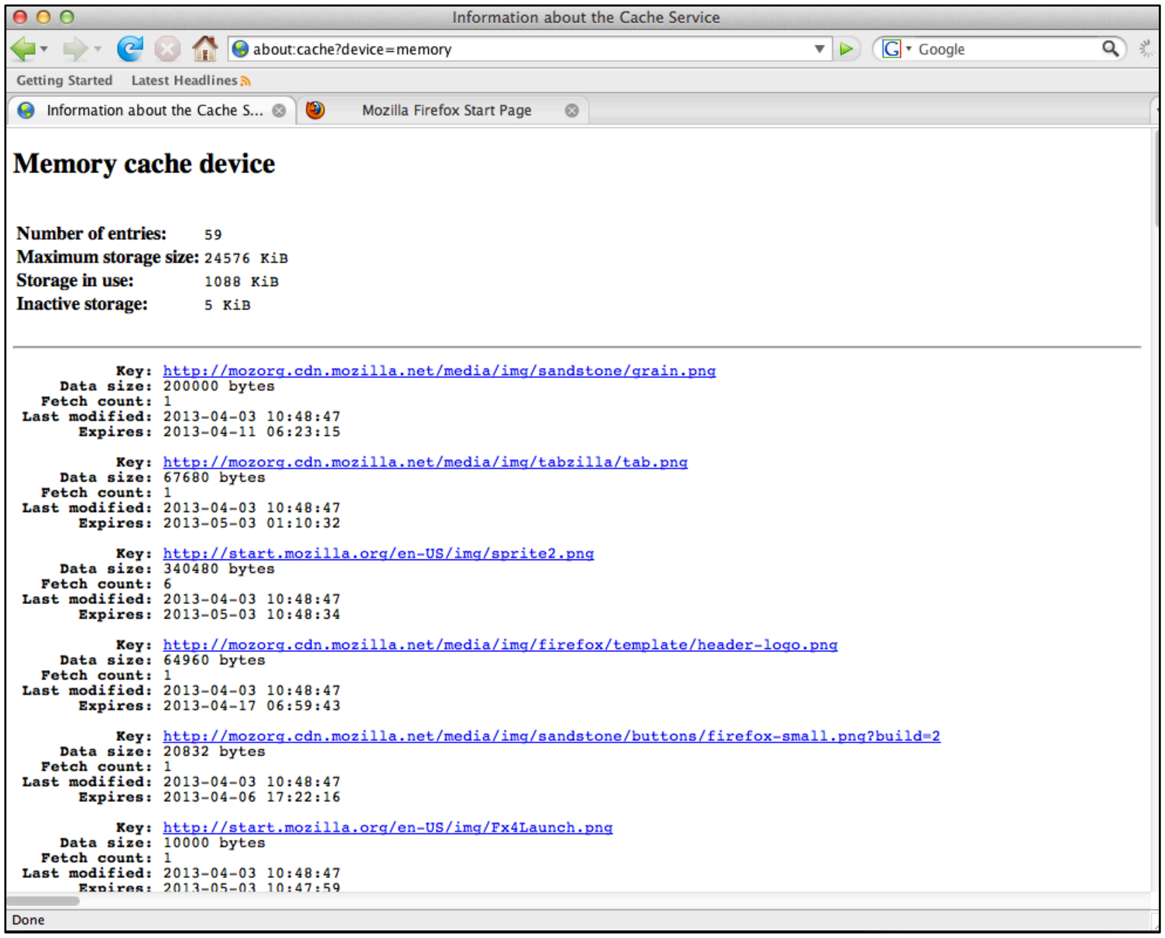
- Browser caching
- Web server caching
- Proxy caching

Each of these can improve Web site performance with static content but they work in different ways and have different advantages and limitations.

Browser Caching

Web browsers can store local copies of content on a client device. The first time content is downloaded, it can be stored in the cache. The next time it is needed, the content is loaded from cache instead of downloading from the Web site (see Figure 2.5). This setup saves the RTT to retrieve the content.

Browser caching helps with content that is used repeatedly during a session. Logo images and CSS files that may be used throughout a site need to be downloaded only once during a session.



Information about the Cache Service

about:cache?device=memory

Memory cache device

```
Number of entries:      59
Maximum storage size:   24576 KiB
Storage in use:         1088 KiB
Inactive storage:       5 KiB
```

```
          Key: http://mozorg.cdn.mozilla.net/media/img/sandstone/grain.png
    Data size: 200000 bytes
  Fetch count: 1
Last modified: 2013-04-03 10:48:47
      Expires: 2013-04-11 06:23:15

          Key: http://mozorg.cdn.mozilla.net/media/img/tabzilla/tab.png
    Data size: 67680 bytes
  Fetch count: 1
Last modified: 2013-04-03 10:48:47
      Expires: 2013-05-03 01:10:32

          Key: http://start.mozilla.org/en-US/img/sprite2.png
    Data size: 340480 bytes
  Fetch count: 6
Last modified: 2013-04-03 10:48:47
      Expires: 2013-05-03 10:48:34

          Key: http://mozorg.cdn.mozilla.net/media/img/firefox/template/header-logo.png
    Data size: 64960 bytes
  Fetch count: 1
Last modified: 2013-04-03 10:48:47
      Expires: 2013-04-17 06:59:43

          Key: http://mozorg.cdn.mozilla.net/media/img/sandstone/buttons/firefox-small.png?build=2
    Data size: 20832 bytes
  Fetch count: 1
Last modified: 2013-04-03 10:48:47
      Expires: 2013-04-06 17:22:16

          Key: http://start.mozilla.org/en-US/img/Fx4Launch.png
    Data size: 10000 bytes
  Fetch count: 1
Last modified: 2013-04-03 10:48:47
      Expires: 2013-05-03 10:47:59
```

Figure 2.5: Browser cache can improve page loading times on individual devices, especially for content reused within a Web site.

Web Server Caching

Web server caching, as the name implies, operates at the server level (see Figure 2.6). Content requested by any user can be stored in memory and made available for other users. This functionality can reduce the disk I/O load on the Web server because popular content can be retrieved from memory without requiring disk access operations.

Unlike browser caching, this form of caching does not eliminate the need for round-trip communication between the client device and the Web server. It does, however, have the additional benefit of caching content accessed by other users, which is not the case with browser-based caching.

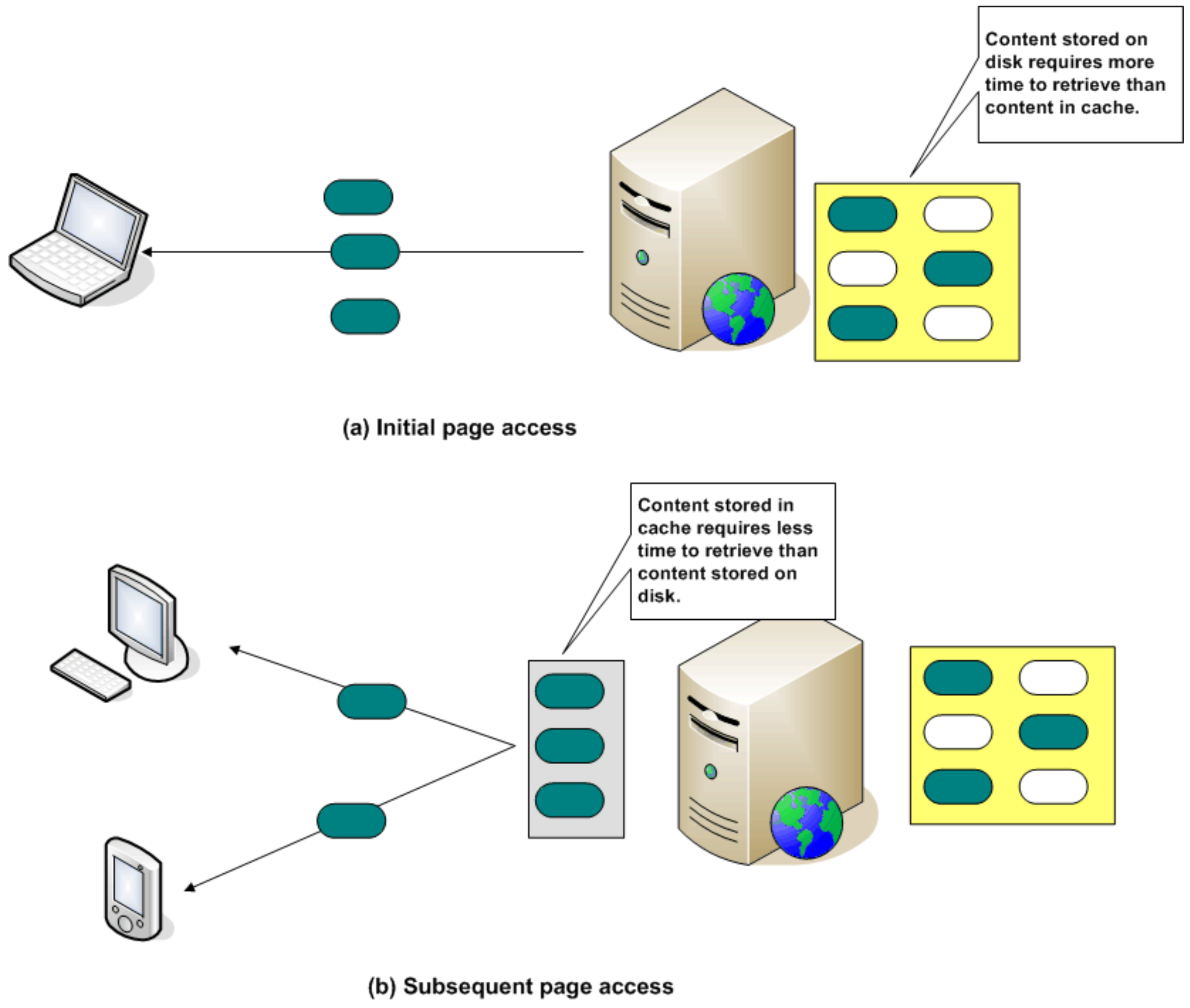

Content stored on disk requires more time to retrieve than content in cache.
(a) Initial page access
Content stored in cache requires less time to retrieve than content stored on disk.
(b) Subsequent page access

Figure 2.6: Web server caching pools the benefits of caching across multiple users. Note: Green ovals represent pages retrieved; white ovals are pages stored but not retrieved.

Proxy Caching

Proxy caching is a service offered by ISPs to reduce the time needed to load content on client devices and to reduce traffic on their own networks. As Figure 2.7 shows, proxy caching works by keeping copies of popular content on the ISP's servers. When a client device makes a request for content, the proxy cache is queried and, if the content is found, it is delivered to the client device from the ISP. This setup avoids the overhead and delay encountered when the request has to be sent to the Web site's server.

Proxy caching reduces RTT because the client needs only to wait for a response from the ISP. This type of caching has advantages for the ISP's customers. It cuts down on the load on the customer's Web site and reduces the traffic on the customer's networks. Customers have limited control over the proxy cache because the ISP is likely using the cache for multiple customers.

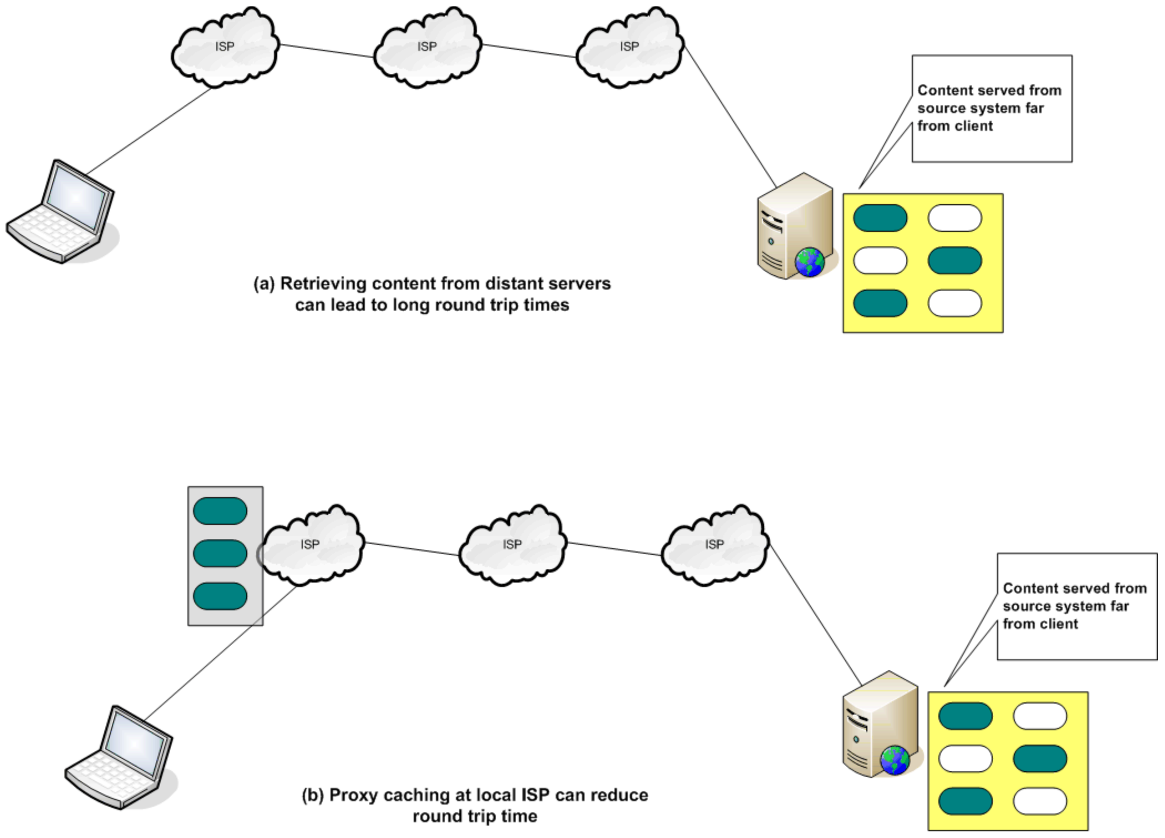

ISP
ISP
ISP
Content served from source system far from client
(a) Retrieving content from distant servers can lead to long round trip times
ISP
ISP
ISP
Content served from source system far from client
(b) Proxy caching at local ISP can reduce round trip time

Figure 2.7: Proxy caching reduces RTT by shortening the distance network traffic must traverse to respond to a request.

Caching vs. Replication

A common characteristic of Web server caching and proxy caching is that they depend on temporary storage. When servers restart, the content of the cache is lost unless it is copied to persistent storage first. This situation is not necessarily a problem from an ISP's perspective because the demand for particular content is constantly changing, at least when one considers demand across a wide range of customers. For individual businesses, however, the most popular static content may be frequently requested over extended periods of time.

Another limitation on caching is the amount of cache available. RAM is a limited resource on any server and only so much can be dedicated to caching Web content. When the cache is full, no other content can be stored there or some of the content must be deleted to make room for newer content. This requirement can be handled in several ways (see Figure 2.8): A simple strategy is to simply remove the oldest content whenever the cache is full. Old content, however, might be popular content, so a better approach in some cases is to delete the least recently used content without respect for age of the content.

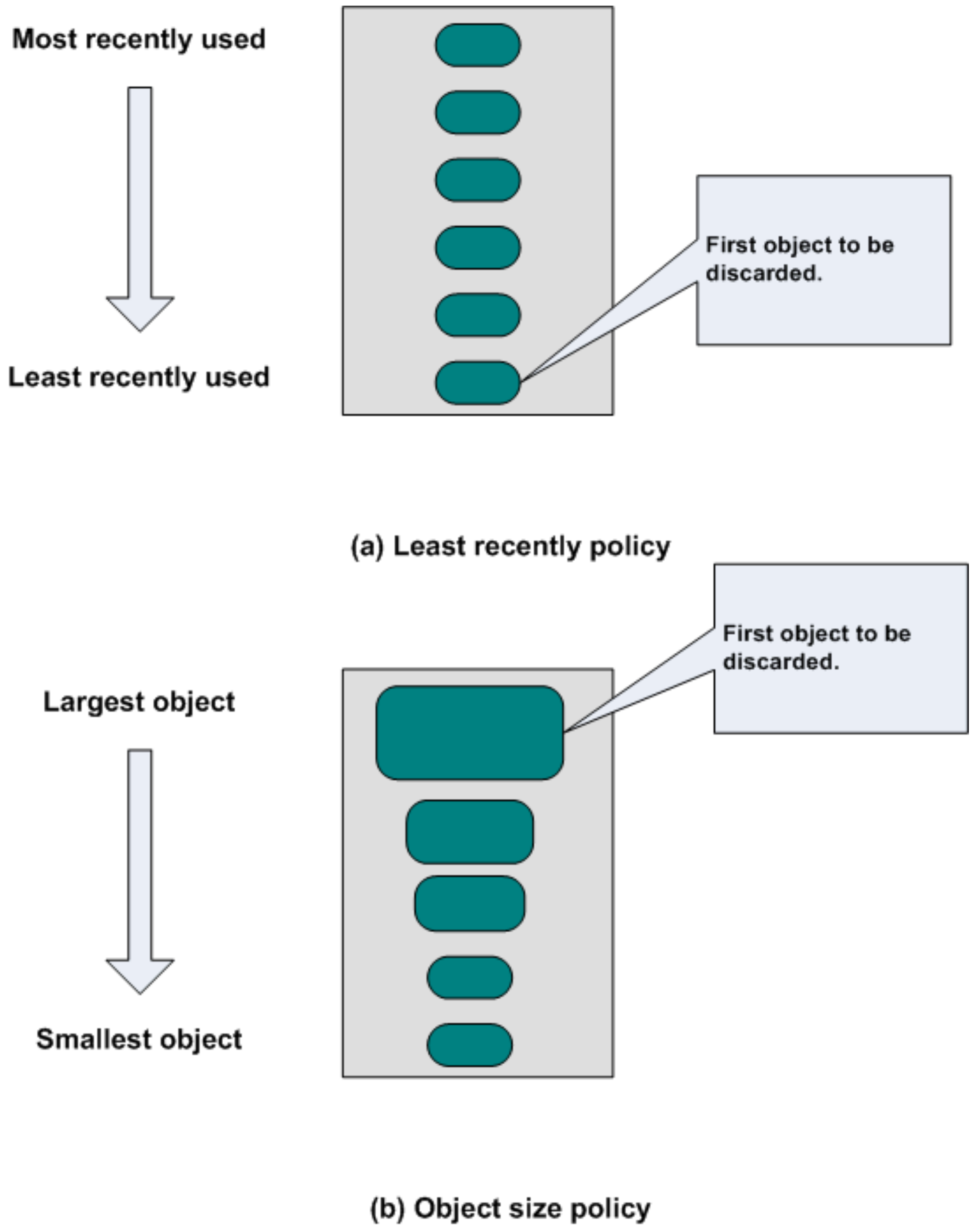


Figure 2.8: Cache content replacement policies optimize for different measures, such as frequency of use or object size.

Another strategy considers content size rather than just age or recent usage. The idea is that if more small objects are kept in the cache, more objects can be stored. In turn, this setup should increase the rate that objects are found in the cache. Size, age, and frequency of access policies can be combined to optimize different objectives with caching.

The age of objects in the cache is another way to determine when an object should be deleted from the cache. A parameter known as time to live (TTL) determines how long an object is stored in the cache before it is deleted. A cache with a 1200 second TTL, for example, would keep objects in the cache for at-most 20 minutes before deleting them.

An alternative to caching is replication, in which copies of content are stored persistently on servers distributed across the Internet. The goal is to keep copies of content closer to users to reduce the RTT need to satisfy requests for content. It also has the added benefit of reducing load on the Web server hosting the original source of the content. Key considerations with replication are the number and location of replicated servers and the frequency with which content is updated.

If a large portion of your Web traffic originates in Europe, it makes sense to replicate content to servers located there. Other factors should be considered as well. If you located a replicated server in a data center in Amsterdam, for example, would customers and business partners in Eastern Europe realize the same performance improvements as those in Western Europe? If latency and congestion are problems with some ISPs in Eastern Europe, you might want to deploy a replicated server in such as way as to minimize the traffic over that ISP.

Replicated content will need to be refreshed to keep all content consistent across servers. Frequent, incremental updates are warranted when content changes often. The type of content you are replicating can also influence your decisions about update frequency. If content contains legal or regulated information, such as forward-looking statements from a publicly traded company, it is especially important to minimize the chance that users in different parts of the world would see different versions of the content (see Figure 2.9).

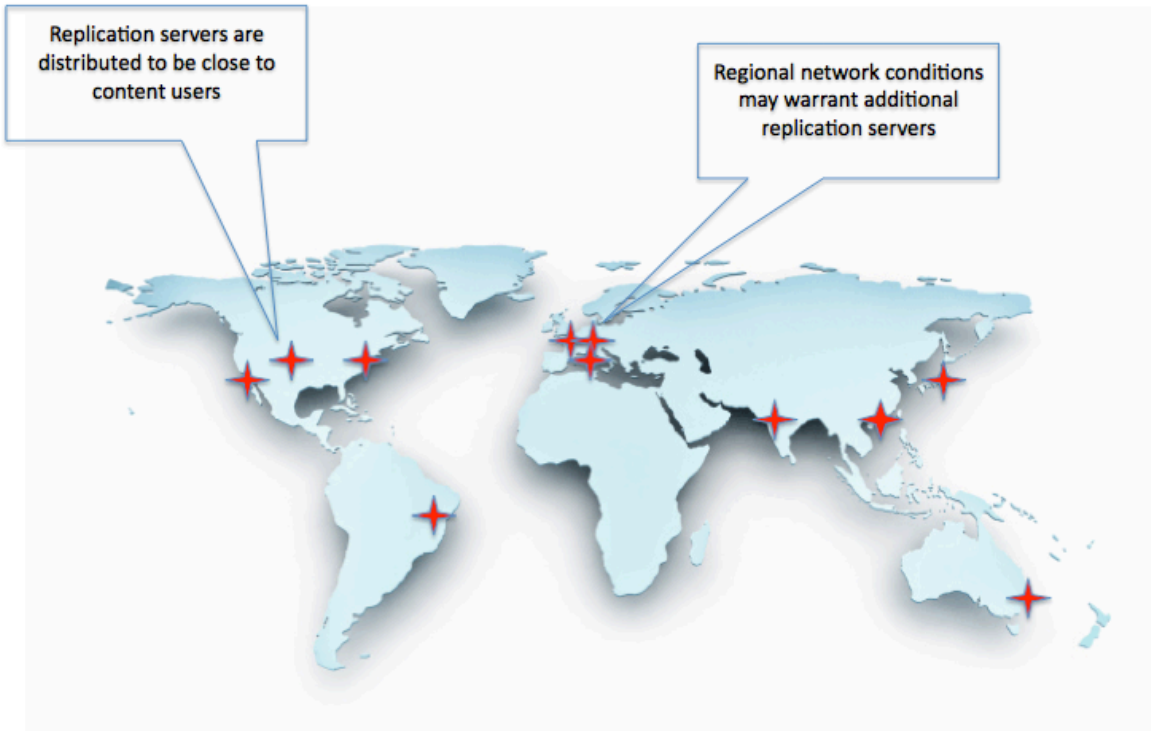

Replication servers are distributed to be close to content users
Regional network conditions may warrant additional replication servers

Figure 2.9: Origin of traffic and network conditions influence the placement and number of replicated servers.

Caching and replication can help with static content but many Web pages are dynamically generated. For dynamic content, you need to consider other methods to optimize delivery.

Optimizing the Delivery of Dynamic Content

Dynamic content is generated by a variety of applications. Customers purchasing products will receive content specific to their purchase choices. Analysts querying a business intelligence system will receive pages with data or visualizations based on their queries. Employees working with a human resources portal will have content generated according to their roles and benefits within the organization. In all of these cases, caching the content would serve no purpose.

Before considering ways to optimize dynamic content, let's explore the steps involved in generating and delivering that content, taking a purchase transaction as an example.

When a customer views a catalog of products, the content may be served from a cache or replicated site. Once the customer selects a set of products for purchase, they will begin viewing dynamically generated content. Each request must be served from the origin. As Figure 2.10 illustrates, a purchase transaction requires:

- Requests to generate a page listing the products selected for purchase
- Customized options based on the customer's profile; for example, options for free or discounted shipping
- Customized options based on the content of the shopping cart, such as related products
- Time-sensitive information such as details about delivery time estimates

This content is generated from multiple sources including product and inventory databases, application servers running business logic services, and Web servers with customized style sheets. Pieces of data from all of these sources are collected and combined on an as-needed basis.

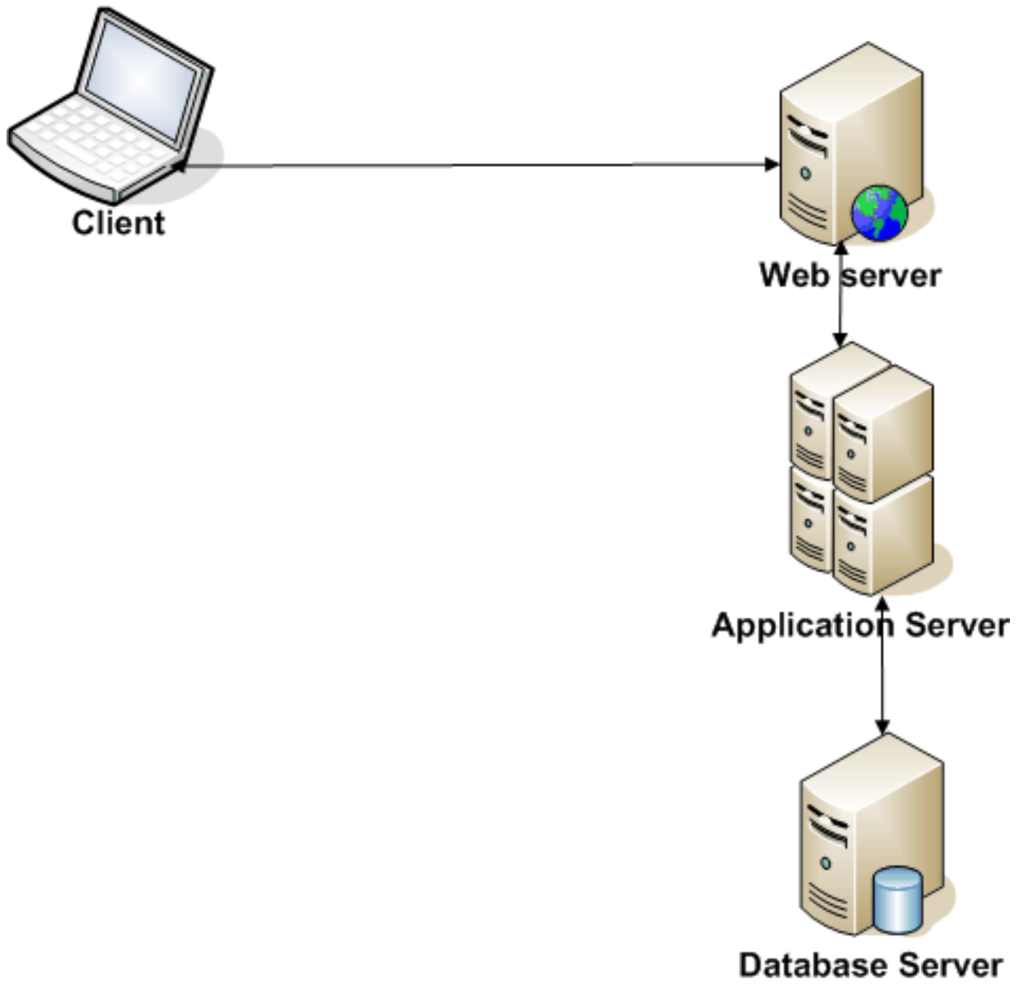


Figure 2.10: Multiple applications can contribute to the construction of a single dynamically generated Web page.

Assessing the Scope of Dynamic Content

When assessing the scope of problems related to delivering dynamic content, consider at least three factors:

- Number of requests for dynamic content
- Importance of dynamic content
- Distance to origin

The combination of these three factors can influence your strategy for addressing problems with maintaining responsive Web applications.

Number of Requests for Dynamic Content

One of the first factors to consider is how much dynamic content is being generated and retrieved by users. Monitoring network traffic is one way to detect the number of URLs associated with dynamic content and the size of the objects downloaded in response to requests for those URLs. If there is a wide variation in traffic over time, the monitoring period should be sufficiently long to capture the scope of variation. Sampling network traffic at various times can also provide sufficient information to assess the volume of dynamic traffic on your network.

A drawback of the monitoring method is that it does not capture information about expected changes in network traffic. For example, if a new customer-facing application will be brought online in two months, the volume of dynamic content from that application will not be reflected in monitoring data. In this case, it helps to have an inventory of applications generating dynamic content as well as information about planned changes and additions to that set of applications.

Importance of Dynamic Content

The importance of dynamic content is a function of the business driver behind that content. Large volumes of dynamic content may be important because of the large number of customers, clients, and collaborators who work with that content. The volume of dynamic content is not the sole measure of its importance. For example, a small group of analysts that work with dynamically generated content from a business intelligence platform may depend on that content for high-value decision making. In addition to the size of dynamic content, be sure to consider the importance of content with respect to business drivers.

Distance to Origin

The distance to origin is another factor that can significantly impact the speed with which a Web application can respond to a user. The greater the distance between a client and a server, the longer the time required to transmit data. Such is the case regardless of other network conditions, such as the rate of packet loss or congestion on the network.

Transmitting data on global scales on the Internet requires data to move over networks managed by multiple ISPs. Those ISPs may have varying policies on handling traffic traversing their networks and service level agreements (SLAs) with their own customers that result in lower priority for non-customer traffic.

When there is congestion on a network, there is a risk of losing packets. The TCP protocol expects acknowledgements for packets sent and waits a specified period of time for that acknowledgement. If an acknowledgement is not received, the packet is resent. This setup introduces a two-fold delay in the Web application: The time the TCP client waits for the acknowledgement, and the time required to send the packet again.

Understanding the distance between client devices and the system of origin is a key element in assessing the need for dynamic content optimization. When the combination of the number of requests for dynamic content, the importance of the dynamic content, and the distance to origin warrants optimized dynamic content, then it is time to consider how to reduce the request burden on the system of origin.

Reducing the Burden on the System of Origin

The system of origin is the application, or set of applications, that is responsible for generating dynamic content. These systems can be complicated combinations of multiple services each responsible for a part of dynamic content creation and related activities. Consider, for example, an application for processing sales transactions. It can include:

- A Web server for generating HTML and related content
- An application server running business logic, such as verifying customers' credit
- An inventory database providing information on the quantity and availability of products
- A sales transaction database for recording details of sales
- A cross-selling platform that offers suggestions for additional purchases related to items in the customers' shopping cart

These are all core functions related to the sales transaction. In addition to these, the system of origin may have to contend with other operations, such as encrypting and decrypting data as it is sent and received and retransmitting lost packets.

There are multiple ways to improve the performance of the system of origin:

- Web server caching can be used to reduce disk I/O operations related to the static content portion of the application
- Databases can be tuned to improve query response time and minimize the number of disk read and write operations
- Application code can be analyzed to identify time-consuming operations
- An application cluster can be expanded to distribute the workload over a larger number of servers

These, however, will not directly address problems with delivering dynamic content to distant client devices. Some additional performance tuning options include:

- TCP optimizations to reduce connection/transmission times
- Terminating SSL at the network edge to offload the system of origin
- Reducing packet loss for fewer transmission delays

These options can address networking issues not handled by the other tuning methods described earlier.

TCP Optimizations

TCP optimizations are techniques used to improve the overall performance of the TCP protocol. The protocol was developed early in the history of internetworking and was designed to provide reliable transmission over potentially unreliable networks. Today's network can realize much faster transmission speeds than those available when TCP was first designed.

A number of different kinds of optimizations are available:

- Options for changing the window size
- Selective acknowledgement
- Detection of spurious retransmission timeouts

The TCP receive window size determines how much data can be sent to a device before the sender must receive an acknowledgement. The TCP specification defines a maximum size of 64Kb for the window size. This limitation can prevent full use of a high-bandwidth network. Changes to the TCP protocols now allow for larger receiver window sizes and therefore more efficient use of high-bandwidth networks.

The selective acknowledgment optimization can help reduce the amount of data retransmitted when packets are lost. TCP was originally designed to use something called a *cumulative acknowledgement,* which provided limited information about lost packets. This functionality can leave the sender waiting additional RTT periods to find out about additional lost packets. Some senders might operate under the assumption that additional packets were lost and retransmit other packets in the segment. Doing so can lead to unnecessary retransmission of packets. Selective acknowledgement provides more information about lost packets and helps reduce unnecessary retransmission of data.

Detecting a spurious retransmission begins by retransmitting the first packet in an unacknowledged segment. The sender then monitors the following acknowledgements to detect patterns that would indicate a spurious retransmission. If a spurious retransmission is detected, the additional packets in the segment are not retransmitted. These and other TCP optimizations can help to better utilize the high-bandwidth networks available today.

Terminating SSL at Network Edge

Encrypting large amounts of data can be computationally demanding. If data is encrypted and decrypted on the same device that is generating dynamic content and responding to Web requests, the system overall responsiveness can decrease.

An alternative to encrypting and decrypting on the system of origin is to perform those operations at the network edge. This setup takes advantage of the fact that once data reaches your, or your provider's, trusted network, it can be decrypted. Similarly, instead of depending on the system of origin to encrypt data, that process can be offloaded to a network device.

Reduce Packet Loss

Reducing packet loss can improve Web application responsiveness. This reduction can be done with a combination of reducing congestion on the network and routing optimizations.

Improving Web application performance when large amounts of dynamic content are involved is challenging. A combination of techniques that reduce the load on the system of origin, optimize TCP, and reduce the distance that traffic must travel over high-congestion networks can all help address the problem.

Targeted Content Delivery

Although the focus of this chapter has been on static and dynamic content in general, it is worth noting the problems described earlier can be even more difficult when you consider the need for targeted content delivery. This delivery requirement can come in a few different forms:

- Device-specific content
- Browser-specific content
- Geography-specific content

Each of these factors can lead to larger volumes of static content and require more complex applications for generating dynamic content. Businesses should consider the need for device-, browser-, and geography-specific content as they assess their current Web applications and need for acceleration.

Summary

Maintaining consistent Web application performance for all users is a challenge. Various forms of caching can improve performance in some cases but some cases are better served by replicating content to servers closer to end users. Dynamic content requires other optimization techniques to improve overall TCP performance, reduce packet loss, and cut total RTT.