Although the volume of Web traffic on the Internet is staggering, a large percentage of that traffic is redundant-multiple users at any given site request much of the same content. This means that a significant percentage of the WAN infrastructure carries the identical content (and identical requests for it) day after day. Eliminating a significant amount of recurring telecommunications charges offers an enormous savings opportunity for enterprise and service provider customers.

Web caching performs the local storage of Web content to serve these redundant user requests more quickly, without sending the requests and the resulting content over the WAN.

Contents

Network Caching

Network caching is the technique of keeping frequently accessed information in a location close to the requester. A Web cache stores Web pages and content on a storage device that is physically or logically closer to the user-closer and faster than a Web lookup. By reducing the amount of traffic on WAN links and on overburdened Web servers, caching provides significant benefits to ISPs, enterprise networks, and end users. There are two key benefits:

- **Cost savings due to WAN bandwidth reduction** - ISPs can place cache engines at strategic points on their networks to improve response times and lower the bandwidth demand on their backbones. ISPs can station cache engines at strategic WAN access points to serve Web requests from a local disk rather than from distant or overrun Web servers.

  In enterprise networks, the dramatic reduction in bandwidth usage due to Web caching allows a lower-bandwidth (lower-cost) WAN link to serve the same user base. Alternatively, the organization can add users or add more services that use the freed bandwidth on the existing WAN link.

- **Improved productivity for end users** - The response of a local Web cache is often three times faster than the download time for the same content over the WAN. End users see dramatic improvements in response times, and the implementation is completely transparent to them.

Other benefits include:

- **Secure access control and monitoring** - The cache engine provides network administrators with a simple, secure method to enforce a site-wide access policy through URL filtering.
- **Operational logging** - Network administrators can learn which URLs receive hits, how many requests per second the cache is serving, what percentage of URLs are served from the cache, and other related operational statistics.

How Web Caching Works

Web caching works as follows:

1. A user accesses a Web page.

2. The network analyzes the request, and based on certain parameters, transparently redirects it to a local network cache.
3. If the cache does not have the Web page, it will make its own Web request to the original Web server.
4. The original Web server delivers the content to the cache, which delivers the content to the client while saving the content in its local storage. That content is now cached.
5. Later, another user requests the same Web page, and the network analyzes this request, and based on certain parameters, transparently redirects it to the local network cache.

Instead of sending the request over the Internet and Intranet, the network cache locally fulfills the request. This process accelerates the delivery of content.

The important task of ensuring that data is up-to-date is addressed in a variety of ways, depending on the design of the system.

The Benefits of Localizing Traffic Patterns

Implementing caching technology localizes traffic patterns and addresses network traffic overload problems in the following ways:

- Content is delivered to users at accelerated rates.
- WAN bandwidth usage is optimized.
- Administrators can more easily monitor traffic.

Network-Integrated Caches

The first step in creating a network-integrated cache engine is to ensure that the network supports traffic localization, which can be achieved by enabling content routing technology at the system-level, and setting specific parameters to optimize network traffic. Cisco IOS ® Web Cache Communication Protocol (WCCP) is one example of content routing technology that can be set to support traffic localization. Once the right network foundation is in place, network caches are added into strategic points within the existing network. By pairing software and hardware, Cisco creates a network-integrated cache engine. Network-integrated caches have at least the following three properties:

- Managed like networking equipment, resulting in minimized operational costs
- Designed like high-density networking hardware, resulting in better physical integration into the network infrastructure as network extensions and minimizing costs associated with leasing rack space
- Transparently inserted into the network, resulting in minimized deployment and operational costs and greater content availability

Existing Caching Solutions

The three most common types of caches on the market today are proxy servers, standalone caches, and browser-based caches.

Proxy Servers

Proxy servers are software applications that run on general-purpose hardware and operating systems. A proxy server is placed on hardware that is physically between a client application, such as a Web browser, and a Web server. The proxy acts as a gatekeeper that receives all packets destined for the Web server and examines each packet to determine if it can fulfill the requests itself; if not, it makes its own request to the

Web server. Proxy servers can also be used to filter requests, for example, to prevent its employees from accessing a specific set of Web sites.

Unfortunately, proxy servers are not optimized for caching, and do not scale under heavy network loads. In addition, because the proxy is in the path of all user traffic, two problems arise: all traffic is slowed to allow the proxy to examine each packet, and failure of the proxy software or hardware causes all users to lose network access. Expensive hardware is required to compensate for the low software performance and the lack of scalability of proxy servers.

Proxies also require configuration of each user's browser-a costly and unscalable management task for service providers and large enterprises. In addition, proxy servers that are arranged in a hierarchical fashion form an additional overlay network, contradicting any plans to strategically converge disparate networks into a single, unified network.

Standalone Caches

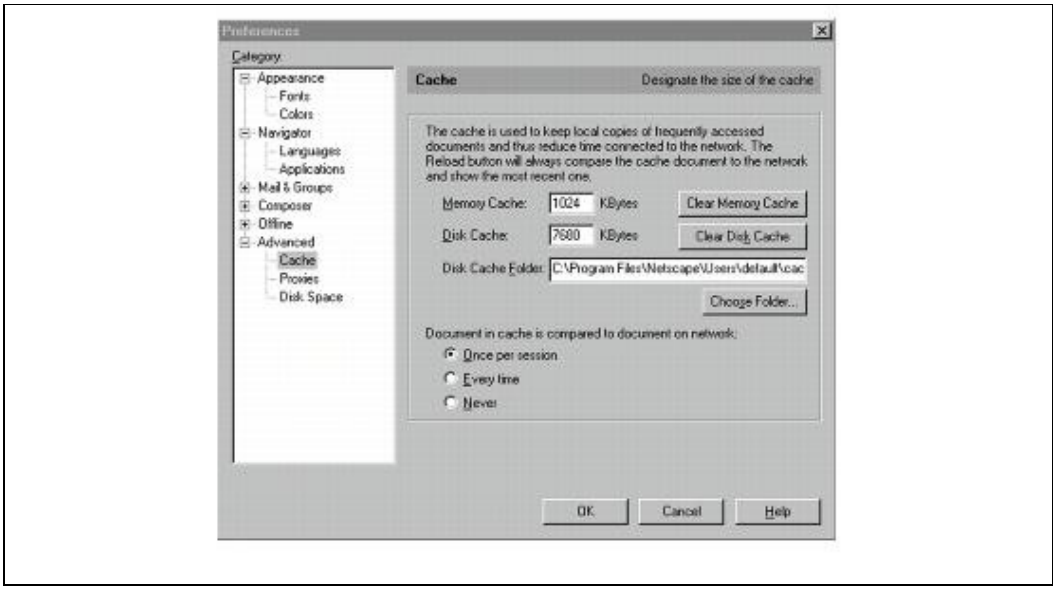
In response to the shortcomings of proxy servers, some vendors have created standalone caches. These caching-focused software applications and appliances are designed to improve performance by enhancing the caching software and eliminating other slow aspects of proxy server implementations. While this is a step in the right direction, these standalone caches are not network integrated, resulting in higher costs of ownership and making them less desirable for wide-scale deployment.

Browser-Based Client Caching

Internet browser applications allow an individual user to cache Web pages (that is, images and HTML text) on his or her local hard disk. A user can configure the amount of disk space devoted to caching.

Figure: Cache configuration window to configure the amount of disk space devoted to caching in Netscape Navigator shows the cache configuration window for Netscape Navigator.

Figure: Cache configuration window to configure the amount of disk space devoted to caching in Netscape Navigator

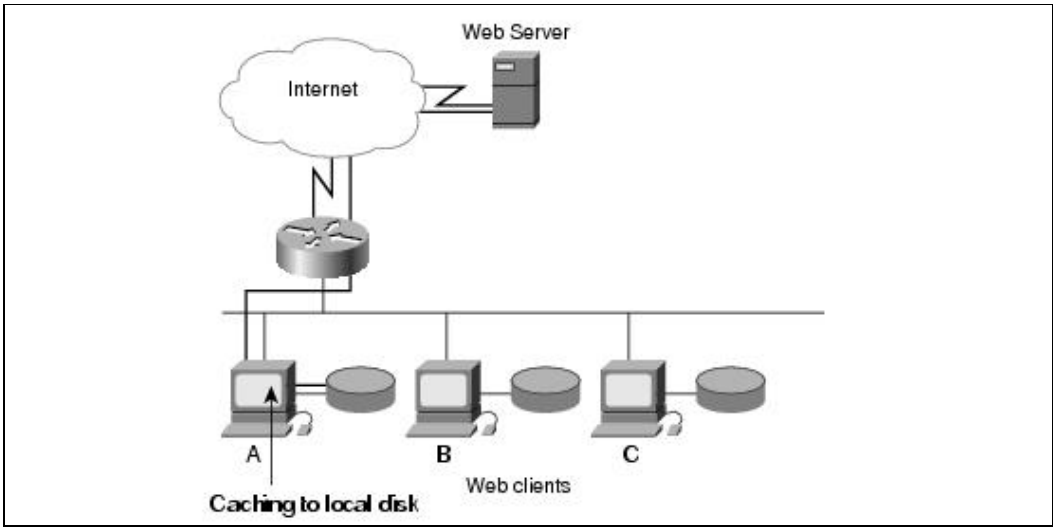


This setup is useful in cases where a user accesses a site more than once. The first time the user views a Web site, that content is saved as files in a subdirectory on that computer's hard disk. The next time the user points to this Web site, the browser gets the content from the cache without accessing the network. The user notices

that the elements of the page--especially larger Web graphics such as buttons, icons, and images appear much more quickly than they did the first time the page was opened.

This method serves this user well, but does not benefit other users on the same network who might access the same Web sites. In Figure: Benefits gained by a single node using browser caching, the fact that User A has cached a popular page has no effect on the download time of this page for Users B and C.

Figure: Benefits gained by a single node using browser caching



WCCP Network Caching

- In 1997, Cisco developed WCCP, a router-cache protocol that localizes network traffic and provides "network-intelligent" load distribution across multiple network caches for maximized download performance and content availability.
- The cache component of the Cisco caching solution comprises network-integrated caching solutions-the Cisco Cache Engine 500 Series. They are network-integrated because they:
- Provide network management capabilities already available on traditional Cisco networking gear (such as Cisco IOS CLI and RADIUS support), resulting in minimized management and operational costs.
- Are inherently designed and implemented as caching-specific networking hardware, rather than being standalone server platforms adapted as caches. Thus, the high-density Cisco Cache Engines physically integrate better into the network infrastructure as network extensions transparently insert into existing network infrastructures and adapt to unusual network conditions, resulting in minimized deployment and operational costs and greater content availability.

Network-Based Shared Caching

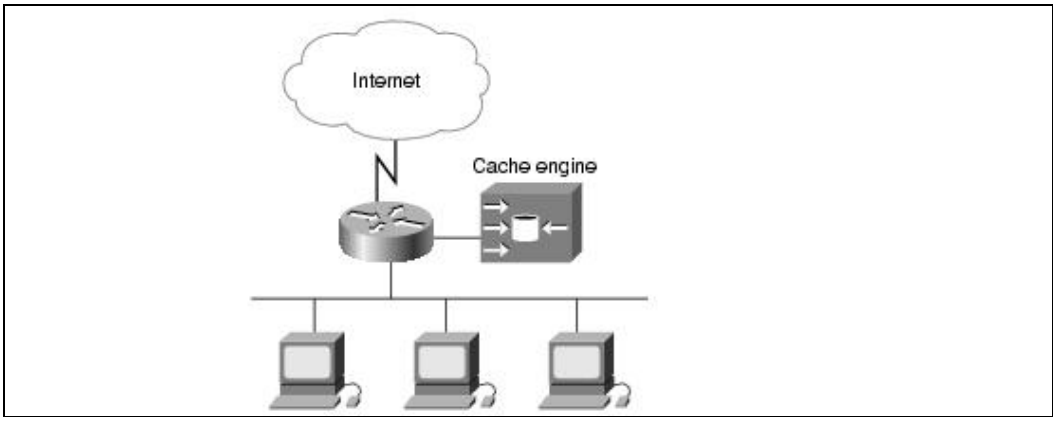
The cache engine was designed from the ground up as a loosely coupled, multinode network system optimized to provide robust shared network caching. The cache engine solution comprises the Web Cache Control Protocol (a standard feature of Cisco IOS software) and one or more Cisco cache engines that store the data in the local network.

The Web Cache Control Protocol defines the communication between the cache engine and the router. Using the Web Cache Control Protocol, the router directs only Web requests to the cache engine (rather than to the intended server). The router also determines cache engine availability, and redirects requests to new cache engines as they are added to an installation.

Figure: Cache configuration window to configure the amount of disk space devoted to caching in Netscape Nav

The Cisco cache engine is a single-purpose network appliance that stores and retrieves content using highly optimized caching and retrieval algorithms. (See the Figure: Cisco cache engine connected to a Cisco IOS router)

Figure: Cisco cache engine connected to a Cisco IOS router



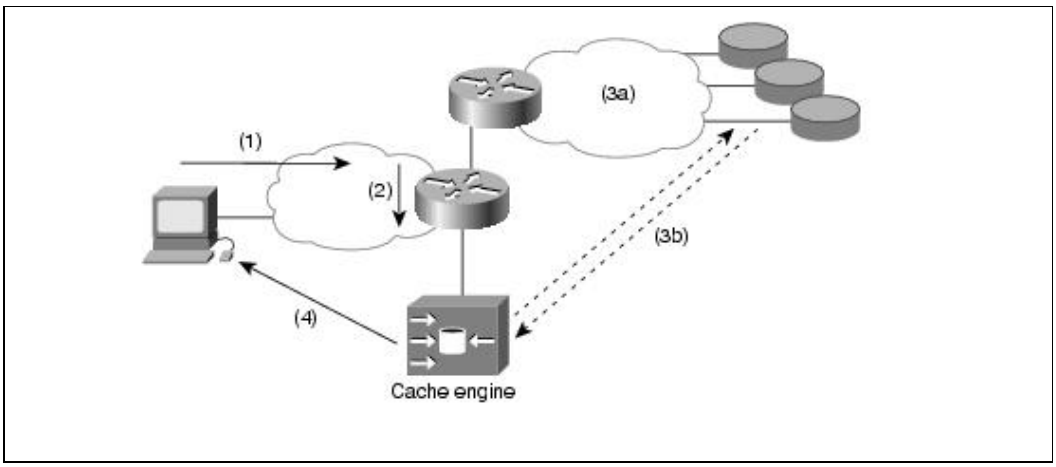
Transparent Network Caching

A cache engine transparently caches as follows:

1. A user requests a Web page from a browser.
2. The WCCP-enabled router analyzes the request, and based on TCP port number, determines if it should transparently redirect it to a cache engine.
3. If a cache engine does not have the requested content, it sets up a separate TCP connection to the end server to retrieve the content. The content returns to, and is stored on, the cache engine.
4. The cache engine sends the content to the client. Upon subsequent requests for the same content, the cache engine transparently fulfills the requests from its local storage.

A cache engine transparently caches as shown in Figure: Transparent Network Caching:

Figure: Transparent Network Caching

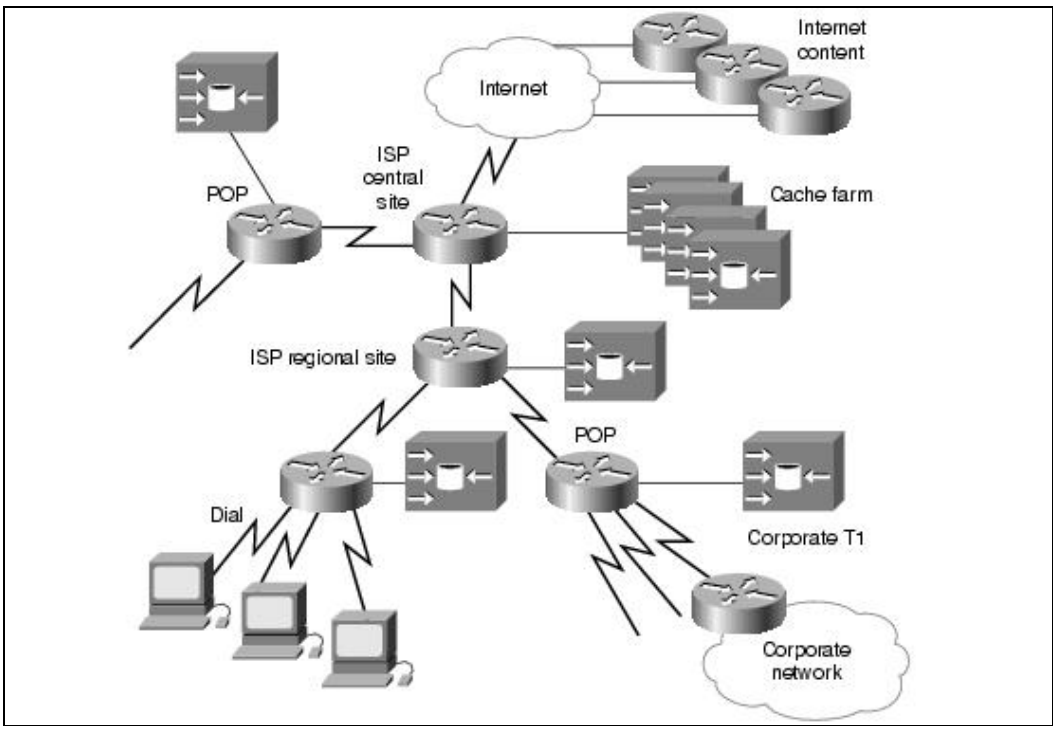


Because the WCCP router redirects packets destined for Web servers to a cache engine, the cache engine operates transparently to clients. Clients do not need to configure their browsers to point to a specific proxy server. This is a compelling feature for ISPs and large enterprises, for whom uniform browser configuration is expensive and difficult to manage. In addition, the cache engine operation is transparent to the network-the router operates entirely in its normal role for nonredirected traffic.

Hierarchical Deployment

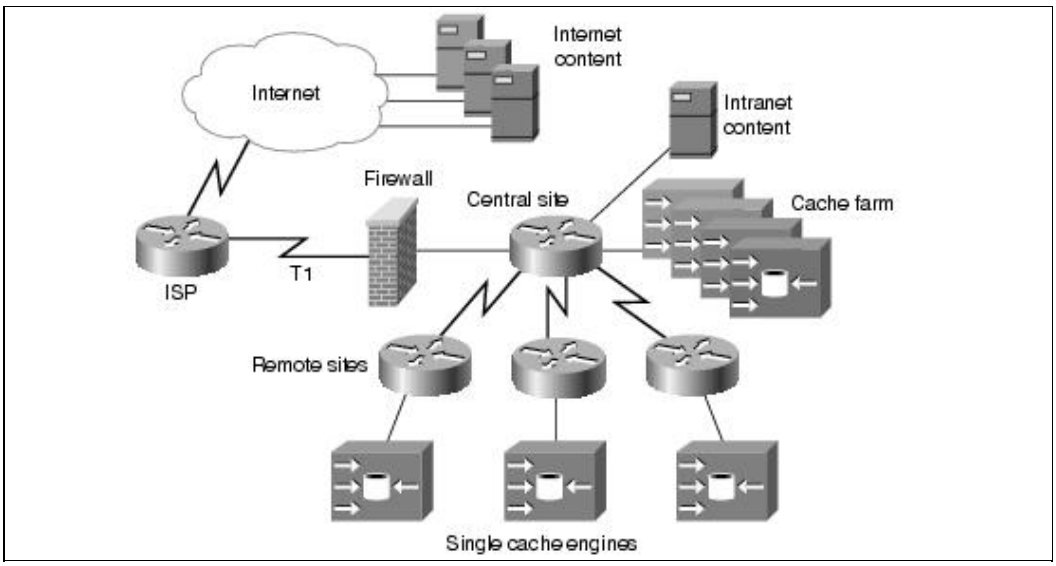
Because a Cisco Cache Engine is transparent to the client and to network operation, customers can easily place cache engines in several network locations in a hierarchical fashion. For example, if an ISP deploys a Cache Engine 590 at its main point of access to the Internet, all of its points of presence (POPs) benefit (Figure: Hierarchical Implementation of Cache Engines (ISP)). Client requests hit the Cisco Cache Engine 590 and are fulfilled from its storage. To further improve service to clients, ISPs can deploy the Cache Engine 590 or 570 at each POP. Then, when a client accesses the Internet, the request is first redirected to the POP cache. If the POP cache is unable to fulfill the request from local storage, it makes a normal Web request to the end server. Upstream, this request is redirected to the Cisco Cache Engine 590 at the main Internet access point. If the request is fulfilled by the Cisco Cache Engine 590, traffic on the main Internet access link is avoided, the origin Web servers experience lower demand, and the client experiences better network response times.

Figure: Hierarchical Implementation of Cache Engines (ISP)



Enterprise networks can apply this hierarchical-transparent architecture to benefit in the same way as shown in Figure: Hierarchical Implementation of Cache Engines (Enterprise):

Figure: Hierarchical Implementation of Cache Engines (Enterprise)



Scalable Clustering

The Cisco caching solution was designed to enable network administrators to easily cluster cache engines to scale high traffic loads. This design approach allows customers to linearly scale performance and cache storage as cache engines are added. For example, a single Cisco Cache Engine 590 can support a 45-Mbps WAN link and 144 GB of cache storage; adding a second Cisco Cache Engine 590 provides support for a 90-Mbps WAN link and 288 GB of cache storage. Up to 32 cache engines can be clustered together.

This linear scalability is achieved because of the manner in which WCCP-enabled routers redirect traffic to cache engines. WCCP-enabled routers perform a hashing function on the incoming request's destination IP address, mapping the request into one of 256 discrete buckets. Statistically, this hashing function distributes incoming requests evenly across all buckets. In addition, these buckets are evenly allocated among all cache engines in a cluster. WCCP-enabled routers ensure that a certain cache engine deterministically fulfills requests for a certain destination IP address on the Internet. Empirically, this distribution algorithm has consistently demonstrated even load distribution across a cache engine cluster. Most of the popular Web sites have multiple IP addresses, thus preventing uneven load distribution.

When the customer adds a new cache engine to the cluster, the WCCP-enabled router detects the presence of the new cache engine and reallocates the 256 buckets to accommodate the additional cache engine. For example, the simplest installation using one router and one cache engine assigns all 256 buckets to the single cache engine. If a customer adds another cache engine, the WCCP-enabled router redirects packets to the two cache engines evenly-128 buckets are allocated to each cache engine. If the customer adds a third cache engine, the WCCP-enabled router assigns 85 or 86 buckets to each of the three cache engines.

Customers can hot-insert cache engines into a fully operating cache cluster. In this situation, the WCCP-enabled router automatically reallocates the buckets evenly among all cache cluster members, including the new cache engine. Because a new cache engine will not have any content, it will incur frequent cache misses until enough content has been populated in its local storage. To alleviate this cold startup problem, the new cache engine, for an initial period, sends a message to the other cache cluster members to see if they have the requested content. If they have the content, they will send it to the new cache engine. Once the new cache engine determines it has retrieved enough content from its peers (based on configurable numbers), it will handle cache misses by directly requesting the content from the end server rather than from its peers.

Figure: Hierarchical Implementation of Cache Engines (Enterprise)

Fault Tolerance and Fail Safety

If any cache engine in a cache cluster fails, the cluster automatically heals itself. The WCCP-enabled router redistributes the failed cache engine's load evenly among the remaining cache engines. The cache cluster continues operation using one less cache engine, but operation is otherwise unaffected.

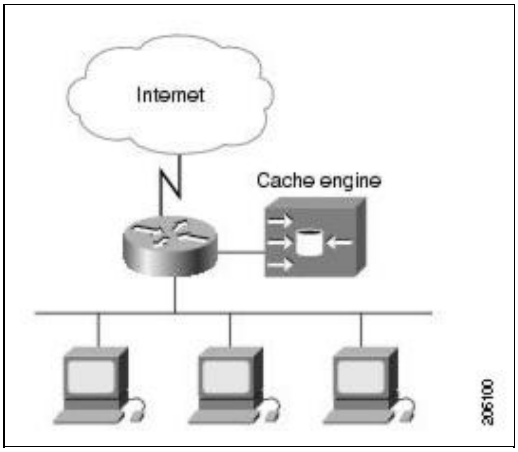
The Cisco network caching solution enables an WCCP-enabled, Multigroup Hot-Standby Router Protocol (MHSRP) router pair to share a cache engine cluster, creating a fully redundant caching system. This is referred to as WCCP multihoming. If the WCCP-enabled router fails, existing Cisco IOS fault tolerance and fail-safe mechanisms are applied. For example, a hot-standby router could dynamically take over operations, redirecting Web requests to the cache cluster.

If an entire cache cluster fails, the WCCP-enabled router automatically stops redirecting traffic to the cache cluster, sending clients' Web requests to the actual destination Web site in the traditional fashion. This loss of the entire cache cluster can appear to users as an increase in download time for Web content, but has no other significant effect. This designed-in, failsafe response is made possible because the cache cluster is not directly in line with clients' other network traffic.

WCCP Multihome Router Support

As previously mentioned, the Cisco network caching solution enables a cache engine cluster to home to multiple WCCP-enabled routers for added redundancy. Thus, Web traffic from all of the WCCP home routers will be redirected to the cache cluster. For example, a cache engine cluster that is homing to both routers in a MHSRP router pair creates a fully redundant caching system, eliminating any single points of failure (Figure: Fully Redundant Cache Engine Cluster Configuration).

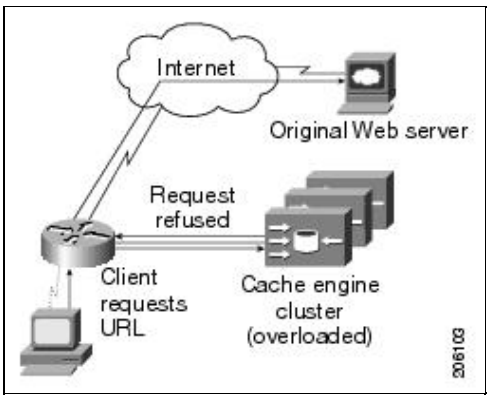
Figure: Fully Redundant Cache Engine Cluster Configuration



Overload Bypass

With a sudden Web traffic surge, a cache engine cluster could become overloaded. To gracefully handle this overload situation, each cache engine detects when it is overloaded, refuses additional requests, and forwards them to the origin Web servers. The origin Web servers respond directly to the clients because the bypassed requests were not handled by a cache engine (Figure: Overload Bypass).

Figure: Overload Bypass



The overloaded cache engine will resume accepting requests when it determines that it has the resources to do so without retriggering overload bypass in the near future. The overload bypass on/off triggers are automatically determined by CPU and file system load. In the extreme situation that the cache engine becomes so overloaded that it is unable to respond to the basic WCCP status check messages from its home router, the WCCP home router will remove the cache engine from the cluster and reallocate its buckets.

Thus, overload bypass ensures that a cache engine cluster does not introduce abnormal latencies and maintains network availability even under unusually high traffic conditions.

Dynamic Client Bypass

Some Web sites require clients to be authenticated using the client's IP address. However, when a network cache is inserted between a client and a Web server, the Web server only sees the cache's IP address and not the client's IP address.

To overcome this issue and similar situations, the Cisco Cache Engine has a dynamic client bypass feature that effectively allows clients, under certain conditions, to bypass cache engines and directly connect to origin Web servers. The result is that a Cisco Cache Engine can preserve existing source IP authentication models and pass through server error messages to clients. Because the cache engine dynamically adapts to these situations, less management is required to ensure cache transparency.

Dynamic Client Bypass Function

In Figure: Dynamic Client Bypass, a client issues a Web request, which is redirected to a cache engine. If the cache engine does not have the content, it will try to fetch the content from the origin Web server.

Figure: Dynamic Client Bypass

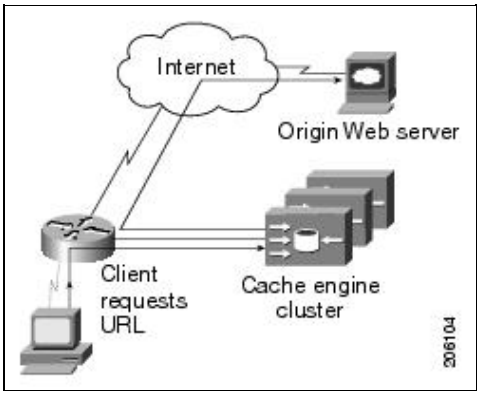
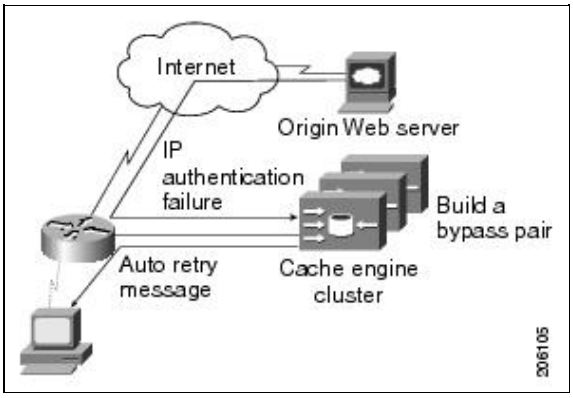


Figure: Overload Bypass

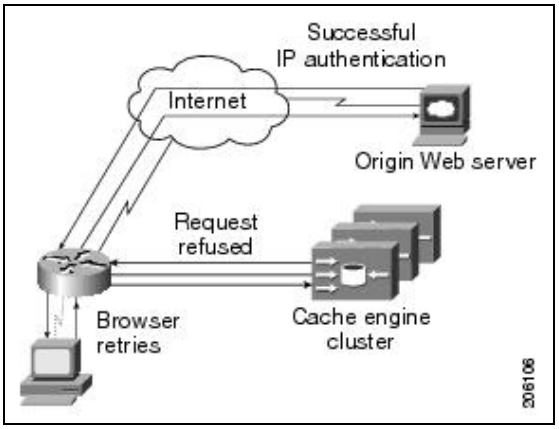
In Figure: Dynamic Client Bypass, if the server responds to the cache engine with certain HTTP error return codes (such as 401-Unauthorized request, 403-Forbidden, or 503-Service Unavailable), the cache engine will invoke the dynamic client bypass feature. The cache engine will dynamically store a client IP-destination IP address bypass pair, so that future packets with this IP address pair will bypass the cache engine. The cache engine sends an automatic HTTP retry message to the client's browser.

Figure: Dynamic Client Bypass



In Figure: Dynamic Client Bypass, when the client's browser automatically issues a reload, the request will be redirected to the cache engine. However, when the bypass table is checked and the request matches one of the table entries, the cache engine will refuse the request and send it directly to the origin Web server. Thus, the origin Web server will see the client's IP address, authenticate the client, and respond directly to the client.

Figure: Dynamic Client Bypass



Reverse Proxy Caching

Cache engines are frequently deployed nearby clients to ensure faster network response time and minimal WAN bandwidth usage. Thus, the caches are caching the clients' most frequently accessed content. In addition, cache engines can also be deployed in front of Web server farms to increase the server farm capacity and improve Web site performance. This configuration is called reverse proxy caching because the cache engines are only caching content from the servers for whom they are acting as a front-end.

This feature is particularly important when cache engines are acting as a front-end for server farms in which certain content is dramatically more popular than other content on the servers. Using reverse-proxy caching allows administrators to prevent a small number high-demand URLs from impacting overall server performance. Better yet, this means the high-demand URLs do not have to be identified, manually replicated,

or independently managed from the bulk of the URLs on the servers.

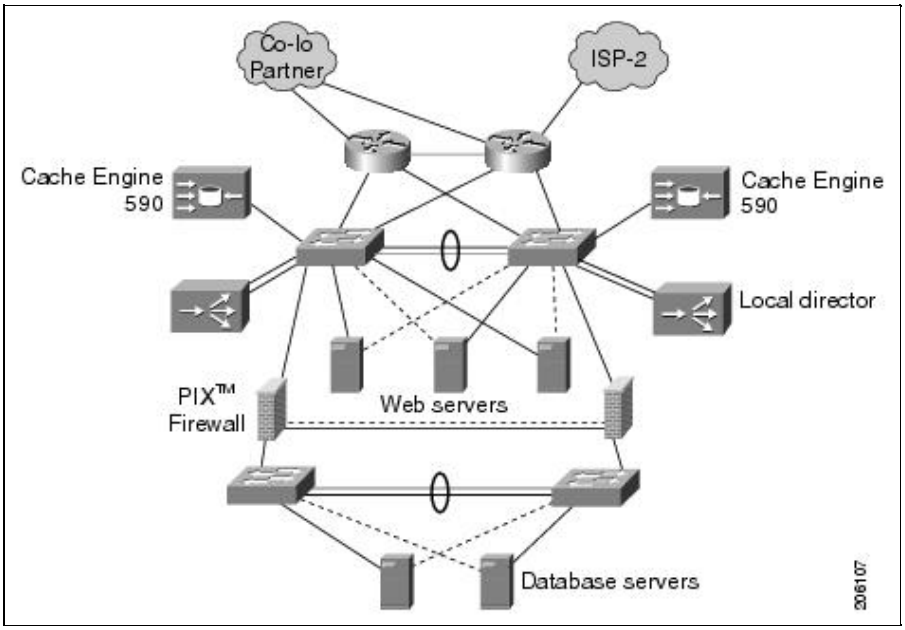
Reverse Proxy Caching Function

In Figure: Reverse Proxy Caching, each cache engine homes to WCCP-enabled routers/switches that are supporting server farms. When an incoming Web request reaches an WCCP-enabled router, the router performs a hashing function on the incoming request's source IP address and port number, mapping the request into one of 256 discrete buckets. Statistically, this hashing function distributes incoming requests evenly across all buckets. In addition, these buckets are evenly allocated among all cache engines in a cluster.

Because the hashing function is based on source IP address and port number instead of destination IP address, a given Web object could be stored in multiple cache engines in a cluster. By spreading popular content across a cache cluster, reverse proxy caching allows multiple cache engines to service requests for very popular content. Thus, additional cache engines can be added to a cluster to incrementally scale the performance of a popular site and decrease content download latency.

Note that hashing on a destination IP address could also do the reverse-proxy caching. But in this case, all requests would have the same destination IP address and would be redirected to one cache engine. If you do not need to scale beyond one cache engine act as a front-end to a server farm, then this method is sufficient.

Figure: Reverse Proxy Caching



Ensuring Fresh Content

A requirement for any caching system is the ability to ensure that users see the same content from a network cache as they would from the Web. Every Web page comprises several Web objects and each Web object has its own caching parameters, determined by content authors and HTTP standards (see the "HTTP Caching Standards" section). Thus, even a Web page with real-time objects typically has many other objects that are cacheable. Rotating ad banners and Common Gateway Interface (CGI)-generated responses are examples of objects that are typically noncacheable. Toolbars, navigation bars, GIFs, and JPEGs are examples of objects that are typically cacheable. Thus, for a given Web page, only a few dynamic objects need to be retrieved from the end server, while static objects can be fulfilled locally.

Cisco Cache Engine products deliver fresh content by obeying the HTTP caching standards and by enabling cache administrators to have control over when content should be refreshed from origin Web servers.

HTTP Caching Standards

HTTP 1.0 and 1.1 are caching standards, which specify caching parameters for each object on a Web page.

HTTP 1.0 allows content authors to enable a "Pragma: no cache" header field for any object that should not be cached and allows authors to enable content to be cached indefinitely.

HTTP 1.1 allows content authors to specify how long content is to be cached. For each object on a Web page, content authors can choose among the following caching attributes:

- Noncacheable
- OK to cache (the default setting)
- Explicit expiration date

HTTP 1.1 has a freshness revalidation mechanism called If-Modified-Since (IMS) to ensure that cached data is up to date. A cache engine will send a lightweight IMS request to the end Web server when the cache engine receives requests for cached content that has expired or IMS requests from clients where the cached content is more than a configured percentage of its maximum age. If the object has not been modified on the end server since the object was cached, the end server will return a lightweight message indicating that the cache engine can deliver its cached copy to clients. If the object has been modified on the end server since the object was cached, the end server will return this information to the cache engine. If the case of the client issuing an IMS request, and the content is less than a configured percentage of its maximum age, the cache will serve the content without checking if it is fresh.

Cache Engine Content Freshness Controls

Administrators can control the freshness of Web objects in a cache engine by configuring a parameter called the freshness factor, which determines how fast or slow content expires. When an object is stored in the cache, its time-to-live (TTL) value is calculated using the following formula: TTL value = (Current date - last modified date) * Configurable freshness factor

When an object expires, based on its TTL value, the cache engine will issue an IMS request the next time the object is requested (see "HTTP Caching Standards" section for a description of the IMS process).

If an administrator wants to adopt a conservative freshness policy, he or she can set the freshness factor to a small value (such as 0.05), so that objects expire more quickly. But the disadvantage to this approach is that IMS requests will be issued more frequently, consuming extra bandwidth. If an administrator wants to adopt a liberal freshness policy, the fresh factor can be set to a larger value, so that objects will expire more slowly and the IMS bandwidth overhead will be smaller.

Browser Freshness Controls

Finally, clients can always explicitly refresh content at any time by using the browser's reload/refresh button.

The **reload/refresh** command is a browser-triggered command to request a data refresh. A **reload/refresh** will issue a series of IMS requests asking for only data that has changed.

The **shift+reload/shift+refresh** command is an extension of the **reload/refresh** command. In correctly implemented browsers, this command always triggers a "pragma: no cache" rather than an IMS request. As a

result, cache engines are bypassed and the end server directly fulfills all content.

Summary

Much of the traffic on the Web is redundant, meaning that users in the same location often access the same content over and over. Eliminating a significant portion of recurring telecommunications offers huge savings to enterprise and service providers.

Caching is the technique of keeping frequently accessed information in a location close to the requester. The two key benefits are:

- Cost
- Improved usability

Implementing caching technology in a network accelerates content delivery, optimizes WAN bandwidth, and enables content monitoring.

Cisco has created a network-integrated cache engine by pairing system-level software and hardware.

Review Questions

Q - *On what concept is network caching based?*

A - Based on the assumption that users access the same content over and over.

Q - *What are two secondary benefits of implementing caching technology?*

A - 1. Secure access and control.

2. Operational logging-administrators can log how many hits sites receive.

Q - *Provide a brief description of network-integrated caching technology.*

A - Network-integrated caching technology combines system-level software and hardware.
Network-integrated caches must be managed like network equipment, designed like high-density hardware, and transparently inserted into the network.

Q - *How do Cisco cache engines ensure that web pages are kept up to date?*

A - By obeying HTTP caching standards that dictate which elements on a page can be cached and which cannot. Those that are not are retrieved from the source every time they are accessed.

Q - *Name an object that can be saved in cache memory, and one that cannot.*

A - Saved in cache: rotating banners, GIFs and JPEGs, toolbars, navigation bars. Noncacheable:
CGI-generated responses.