Ensuring Continuous Availability and Peak Performance for Web Sites and Web-Based Applications

An informational paper from CAI Networks, Inc.
(December 8, 2006)

THE PROBLEM

"*The web site is down*". These five words can induce a panic attack in even the most seasoned and level-headed network administrator. Every second of downtime costs money, potentially damages customer relationships, and inhibits new customers. And, to add insult to injury, it is not possible to contact the users to tell them what's going on because they are nameless, faceless, people located somewhere in the world ... many of whom got sick of waiting and clicked themselves over to your competition.

- Will they try again later?
- Will they try again … ever?
- How would they even know what to do if the thing that tells them that is inoperative?
- What did that problem cost? (Your job maybe?)

In today's world, where 24x7 availability is the new norm and the competition is just a mouse-click away, continuous uptime is no longer an option: it can mean the difference between life and death for online enterprises and traditional businesses alike. Even the smallest hiccup that results in a few seconds of downtime can have severe consequences for your business or organization. The cost of downtime is simply too great to not take effective measures to prevent it.

Even if you have your computer operations down to a science and you use the best and fastest web server on the market, any number of things can still go wrong: software faults, hardware faults, viruses, attacks, fires, floods, high winds that cause a tree to fall into some suspension wires, or somebody 50 miles away digging an irrigation ditch drives a backhoe into a communication line (this unfortunate author has experienced them all). Or, on an otherwise happier side, a favorable mention in the press could cause a flood of traffic to your web site, or a too-good-to-be-true sale could attract a slew of bargain-hunters, or a "Slashdot Effect" reference from another web site -- which, unhappily, could bring your server to its knees.

Such unplanned downtime and performance degradations can be hard to anticipate and accommodate; but even the things that are easy to plan for (like backup and upgrades) can mean downtime. Not that long ago, nights and weekends were available for maintenance and other tasks that required systems to be taken offline, but in today's globally connected world, your nighttime is somebody else's daytime, and your Sunday is somebody's Monday. People want to access your web site at their convenience, not yours.

Coupled with the need for web sites to be always available is the requirement that they be fast. With ever-faster Internet connection rates for businesses and consumers alike, it is no longer possible to mask a slow web site behind slow Internet connections. Thanks to technologies like ADSL and cable, even the most casual home user can reach your web site in milliseconds. If your web site does not perform as well as other web sites, it will be all too obvious.

It would not be an overstatement to say that the Internet has revolutionized computing, not just because of the Web but also because so-called Web-based technologies have transformed application delivery in

even the most stoic corporate computing environments – not only for external users but also internal ones. Web-based applications, once on the bleeding edge, are now run-of-the-mill, and offered by the likes of Oracle, Microsoft, and SAP. Increasingly, web servers are not only hosting web sites but also mission-critical applications that run the business.

THE CHALLENGE

To address heightened demands for functionality, reliability, and performance amidst ever-increasing complexity and transaction demands, CTOs, network administrators, and systems and operations managers face challenges to keep systems up and running 24x7. How to address them is not necessarily obvious. Sure, you can throw money at the problem, with redundant hardware, backup sites, and the like ... but it is easy to throw a lot of money into the wrong things. You can replicate every last bit of hardware and secure it in an underground concrete-reinforced fortress, but you need to ensure that your hot-standby/failover solution answers some basic questions:

- How do you know if the primary environment failed, and how quickly?
- How do you switch to the backup environment if the primarily environment fails?
- Is failover manual or automated process, and how long does it take?
- What happens to current users – and users trying to gain access – during failover?

A hot standby environment goes cold pretty fast if it cannot spring into action at the moment it's needed. And it seems a pity to have all that redundant gear sitting idle when it could be doubling your performance and capacity if it were active.

INTRODUCING THE "SERVER LOAD BALANCER"

Several years ago, a new type of network appliance came onto the market: the *server load balancer (SLB*). Server load balancers came into being at a time when computers (typically PCs) did not offer the capacity to host busy web sites, and so it was necessary to replicate web sites across multiple PCs to achieve scalability and performance. The server load balancer treats multiple PCs as one large virtual PC, thereby providing the capacity required to handle large volumes of traffic with peak responsiveness.

So what does load balancing have to do with ensuring the availability of web sites, preventing unplanned downtime from crashes, disasters, attacks, etc. and facilitating planned downtime for backup, maintenance, upgrades, etc.?

While performance and scalability were originally the hallmarks of server load balancers, high availability has always been a key benefit. (After all, a down system offers no performance whatsoever and can service exactly zero users.) So one of the capabilities of good server load balancers is to cope with the failure of any of the servers it is load balancing, thereby shifting the work to other servers. Server load balancers furthermore allow any server to be taken out of operation, thereby sharing the load among the remaining servers. Thereby, as a byproduct of facilitating scalability for web sites, server load balancers solve many unplanned downtime problems and also facilitate planned downtime.

Over the years, as PCs have gotten more powerful and as larger computers have become capable of hosting web sites, the high-availability and continuous-operation features of server load balancers have become more prominent. Server load balancers have gained wide acceptance and popularity although over the years, and their capabilities have evolved well beyond their modest moniker would suggest. It would be more correct to refer to today's robust load balancers as *traffic managers*, but even this description sells these products short.

Today's leading server load balancers are hybrids of various products, incorporating a number of technologies into a streamlined network appliance. Robust load balancers are not only capable of balancing the load between multiple servers and ensuring availability, they offer content-based traffic management, provide NAT functionality and SSL acceleration, some incorporate firewalls and proxy

servers, and some have switching capabilities. The modestly-named server load balancer offers a wealth of functionality, and some are Swiss Army knives among network appliances.

A CLOSER LOOK

This paper will describe server load balancer concepts and functionality, illustrate how server load balancers are beneficial, and give use-case examples. No specialized networking knowledge is necessary for the reader: server load balancers are conceptually simple and straightforward. This simplicity carries through to the setup and configuration of well-implemented load balancers, with no need to bother the network administrator (who typically looks after it) with the underlying complexity.

This paper will thereafter describe server load balancer scenarios from the perspective of CAI Network's WebMux™ network appliance, which a leading load balancer and one of the first to market.

WHO NEEDS A SERVER LOAD BALANCER?

Before looking in detail at what a server load balancer is and what it does, let's first examine the profile of companies and organizations that use them. The typical profile of server load balancer customers includes one or more of the following:

- Hosts one or more web sites, Web-based applications, email servers, FTP sites, and/or other TCP/iP-based services
- Needs to increase server capacity to handle more users/connections
- Wants to improve user performance
- Needs to ensure continuous or high availability to such services
- Does not have effective provisions in place for coping with systems problems, such as crashes, network outages, etc.
- Does not have effective provisions in place for coping with environmental problem, such as fires, floods, etc.
- Has activity peaks that put a strain on existing web servers
- Has difficulty scheduling time for server backups, upgrades, maintenance, software modifications, etc.

If your business or organization meets any of the above conditions, it is a candidate for a server load balancer.

Businesses and organizations that use load balancers do so because of a demonstrated need and the easy cost/benefit they provide, including:

- Eliminate loss of revenue due to a down web site
- Eliminate loss of revenue caused by transaction abandonment resulting from a slow web site
- Gain more revenue due to the ability to handle increasing user and traffic volumes
- Increased customer retention from high satisfaction levels based on speed, robustness, and, if instrumented, personalized content delivery

"I ALREADY DO LOAD BALANCING"

Sometimes, server load balancers get confused with other types of load balancing or failover solutions. Some software and hardware products offer load balancing and failover capabilities that on the surface may appear to provide the same type of benefits as a server load balancer, but they may not go far enough to provide adequate coverage or may have performance issues. For example, some database management systems have replication capability, where a "shadow" database can be kept synchronized

with a live database and switched to in the event that the primary database fails. While this is a useful facility that provides valuable protection, it is not the same as what a server load balancer provides, since the user connection to the application server must exist to even access the database (whether primary or shadow).

Server load balancers sit at the gate of network access, and if no traffic can get in the door and reach an operational web server or application server, the benefits of any downstream availability provisions for applications and/or databases will never be exercised.

The difference between load balancing via a server load balancer and application-based load balancing is that a server load balancer is the first to receive the transaction and it is able to allocate it to the appropriate server. Contrast this with the performance result of a possibly already-burdened server receiving the transaction, deciding it could not service it effectively, and therefore passing it on to another server: the server load balancer wins.

Server load balancers are specifically designed to handle high volumes of transactions and made near-wire-speed decisions about which server is best able to handle each one.

(Since we're now clear on what a server load balancer is and how it differs from other forms of load balancing, this paper will from now on refer to them simply as *load balancers*.)

LOAD BALANCER FEATURES AND BENEFITS

Not all load balancers are created equal: different ones have variations in features, performance, and quality. The features typically found in a load balancer, from most common to least common, include:

- Server load balancing
- Traffic management (user- and content-based)
- Failover
- Fault-tolerance
- Connection persistence
- SSL support (termination, certificate management, and acceleration)

Load balancing

We'll now examine each of these in detail.

As previously stated, a load balancer's core functionality is to balance the load between multiple web servers. For simple load balancers, this means setting up identical web servers with identical content and letting the load balancer rotate incoming transactions between them (see *Figure 1*). Better load balancers can cope with servers having different performance characteristics. The best load balancers can allocate traffic across web servers having different content, automatically directing traffic to the servers that contain the content the user is seeking. Another feature of the best load balancers is that they can dynamically target particular users to particular servers, such as premium users to the best-performing servers.

Ever go to a web page that takes a long time to load, and out of frustration you hit the Refresh button in your browser, and the page comes up immediately? That is often an indication that there is a load balancer in place that does not do an effective job of load balancing, either because of misconfiguration or missing or poorly implemented functionality. (Hitting Refresh resends the transaction, giving the load balancer another chance to service it effectively.)

Most load balancers offer a choice of algorithms that define how incoming traffic should be allocated. The most basic is *Round Robin*, where each server in turn is sent the next incoming transaction. But because some transactions take longer (sometimes a lot longer) than others to complete, a web server can get

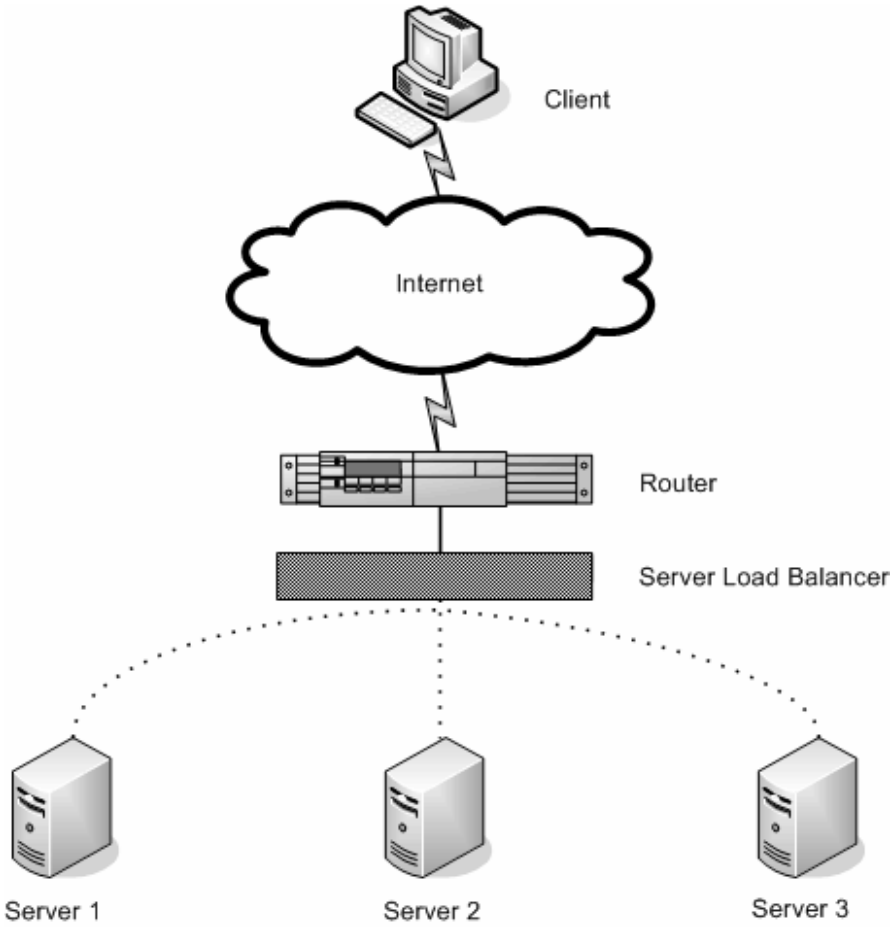

Figure 1

bogged down servicing certain transactions. More sophisticated allocation algorithms take into account real-time performance characteristics, such that the least busy servers can be sent more traffic. Some algorithms allow the allocation to be skewed based on the inherent performance characteristics of the servers, so that more powerful servers are allocated a heavier load.

These capabilities allow effective load balancing between multiple servers that are content-identical, but they do not handle allocation of transactions to servers that have disparate content, nor do they allow the targeting of transactions to be influenced by user attributes. Those two features fall into a category commonly known as *traffic management*.

Traffic management

Traffic management is a rather generic term used in networking (as, similarly, *load balancing* is a rather generic term used in computing) that in the case of server load balancers generally means more sophisticated allocation of traffic across servers. Basic load balancing is in itself a form of traffic management, albeit less robust; the distinction with traffic management is that the load balancer allocates traffic based on its characteristics: while basic load balancing treats all incoming traffic equally, traffic management examines each transaction and targets it based on a set of implementation-specific rules. (If this sounds like the load balancer is behaving like a network switch, it is: a *content switch*, as will be discussed later.)

It is important to understand the distinction between basic load balancing (in which all servers have identical content) and advanced traffic management (in which servers have different content and traffic is directed based on its relationship to the content). A technical aside is required here, as load balancer vendors use different terminology to describe this functionality (such traffic management is also commonly referred to as Layer 7 load balancing, URL-based load balancing, cookie-based load balancing, or content switching) but basically it is about the OSI layer at which traffic is being managed.

The OSI (Open Systems Interconnection) model defines seven layers among which the functions of network protocols are divided. Most load balancers perform load balancing at OSI layer 4 (Transport layer) and/or below, which can accomplish load balancing without regard to its semantic content. In order to route traffic based on content or user, OSI layer 7 (Application layer) must be used. So in considering a load balancer, be sure that if you require allocation of traffic among disparate servers that the load balancer offers OSI layer 7 support.

In content-aware (Layer 7) traffic management, a set of rules are defined that govern the allocation of transactions across multiple servers. These rules describe which servers should receive which transactions, based upon information that is known about the transactions at allocation time.

With such content switching, the amount of information that is available to use in such decision-making, and the corresponding rules definitions, depends upon the sophistication of the load balancer, but they are generally restricted to the URL and cookie reflected in the transaction. Some load balancers will examine only the cookie properties, while others will open the cookie and allow its contents to be used for making targeting decisions. The latter capability is the most flexible, since it allows the web site developer or Web-based application developer to place anything into the cookie and, in turn, configure the load balancer to recognize certain content and target the appropriate web server(s) based on that.

Both user-based and content-based targeting can be accomplished by URL and/or cookie inspection. If, for example, the task is to allocate premium users to a group of high-performing servers that are reserved for paying customers, the web site can be instrumented to authenticate the user and then insert a token into the query string which a load balancer rule will recognize and use to target those transactions to those special servers. If the task is to divert the user to particular servers that contain specific content, the web site can be instrumented to tokenize that information and pass it along in the query string, and a rule defined for the load balancer so that when a user navigates to that page he is so diverted.

Cookies can be used as an alternative to query strings, which makes it easier for the web site developer to address classes of users. With this methodology, for example, once a paying customer registers on a web site, a permanent cookie can be sent to be stored in the customer's browser so that whenever a customer accesses the site he or she can be recognized as such and targeted accordingly.

The ability to examine a cookie's content (not just its properties) allows, for example, customers to be differentiated, such that each customer can have a profile of information that is stored in the cookie. Thereby, appropriate targeting rules can be configured for the load balancer define exactly which server each transaction for each specific user should be used.

Such server targeting based on user attributes or content sought is nothing new in web site development: it is quite commonly done, via URLs, query strings, and/or cookies – for example, diverting a user to a secured payment server to finalize a shopping cart transaction. What a load balancer with sophisticated traffic management capabilities adds is the ability to divert to more than one server, and to have those servers load balanced and capable of failover. Without a load balancer in the equation, a web site would divert a user to a single server which could suffer performance problems and failures.

Have you ever tried to make an online purchase and been able to get through the shopping cart but gotten a 404 error when you reach the payment step? That was likely a case in which you were diverted to a single payment server which was down – something that would not have occurred if there was a second payment server and a load balancer.

Failover and fault tolerance

Failover and fault tolerance functionality together complete the high availability equation. *Failover* refers to the load balancer's ability to detect an inoperative web server and divert all traffic away from it. *Fault tolerance* safeguards against the failure of the load balancer itself, such that a second load balancer can take up the work if the primary load balancer fails. Both are failover operations but under different scenarios (one, the failure of a web server; the other, the failure of the load balancer). Both capabilities are required to be fully sure of continuous operation in the event of the failure of either the server or load balancer.

(Note that even if the servers and load balancers are protected by failover capability, an environmental failure would cause them both to fail, as well as all the active and standby devices within the same environment. A server load balancer can only protect against environmental failures within a limited geographical range; to safeguard against environmental failures of greater scale, multiple sites and the use of a *geographic load balancer* is called for, as described later in this paper.)

While such local failover operations may seem straightforward, they are handled differently by different load balancers. Of course, in order to initiate a failover operation the load balancer must first know that it is necessary to do so, and it is generally up to the load balancer to determine that. The method, scope, and frequency of server health checking determine how quickly and reliability the load balancer detects an inoperative server.

The criteria used to determine whether a server is able to respond to transactions is also important, since there are various failure states that a web server can experience (from a down service to a complete server crash) and not all mean the server should necessarily be considered inoperative. For example, a server hosting FTP and HTTP could have one or the other protocols fail but still are able to service transactions it expects to receive. A good load balancer will independently check the health of each protocol the web server has been defined to support, rather than simply testing whether the server is up or down (since, conversely, a server may be up but unable to do anything useful).

A good health-checking methodology should also take into account "flapping" conditions, where a failure may manifest but be of an intermittent or temporary nature. In such cases, whatever problem the server is experiencing may resolve it, and so the load balancer should continue to monitor the health of such a server and automatically bring it back into operation upon determining it is of sufficient health.

Once a load balancer decides that a failover operation is needed, it may have various scenarios for doing so based on its capabilities and configuration. Some load balancers consider all servers to be active at all times, while others provide for "hot standby" servers which are only activated should a primary server fail. For the hot standby case, there should be no requirement for one-to-one relationship between active and passive servers, so that, for example, two standby servers could cover five active servers.

Similar to failover capability, which addresses otherwise unplanned downtime cases, is the ability to facilitate planned downtime by letting the administrator explicitly take one or more servers out of operation at will, for backup, upgrade, maintenance, etc. In such controlled cases, the load balancer should ensure a graceful transition of active connections on the server(s) before they are removed from operation.

Connection persistence

Connection persistence (sometimes called *stickiness*) is an important concept in multi-server environments, which is not an issue in traditional environments having just one server. In a single-server environment, users are always connected to the same server and therefore all steps in a multi-step transaction execute on the same computer. In a multi-server environment with load balancing, each transaction step could potentially execute on a different server. This may or may not be a problem, depending upon how the web site or Web-based application is developed.

Many web sites and Web-based applications are capable of handling transactions wherein each user request may execute on a different server. Because the Internet is inherently stateless – that is, each page sent from a user's browser is effectively a new transaction, without relationship to the previous

one(s) – many web sites and Web-based applications have provisions to simulate state, and some can track this across multiple servers. For example, if the user is executing a purchasing transaction, the web site will keep track of what pages the user has viewed, what is in the user's shopping cart, etc. and hold that information in a shared database on another server. Such implementation is typically found in a three-tier architectural environment, in which the web site or application and the database reside on separate servers.

In two-tier environments, in which the web site or application and database are running on the same server, it is typical that the code will expect all connections to come into the same server. In such environments, and for web sites and applications that cannot cope with bits of a transaction operating on different servers, implementing a load balancer could prevent proper functioning of the site: if the tracking to simulate state is happening on one server and the user reaches a different server, that server would not know what came before.

In multi-server load balanced environments that require that each user's connections reach the same server (at least for the duration of a multi-step transaction or browser session), robust load balancers offer the ability to force successive connections for individual users to the same servers. In such scenarios, the load balancer does the work: it identifies the first connection for each user session and remembers which server it sent that transaction to; thereafter, whenever a connection from that same user is received, the load balancer sends it to the same server as the first connection. This load balancer feature is generally referred to as *persistence*, since a logically persistent connection for each user to a particular server is maintained.

While sending all of a user's connections to the same server may sound like it would defeat the purpose of load balancing, remember that better load balancers will dynamically test the performance of each web server, and thereby send the initial connection for other users to other servers and keep them load balanced (although perhaps marginally less effectively as it would if persistence were not required).

SSL support

If your web site or Web-based application uses SSL (*Secure Sockets Layer*), you will need a load balancer that supports SSL. How extensively you use SSL, the volume of SSL transactions you process, the SSL certificate sizes you use, and the complexity of certificate management will determine whether a load balancer with only surface SSL support will be able to do the job or whether you require one that offers more robust SSL support.

SSL is a cryptographic protocol that encrypts (and decrypts) transmitted content for security purposes. It is used selectively in web sites and Web-based applications when sensitive data, such as credit card information, is transmitted. SSL is most commonly used with HTTP to form the HTTPS protocol. (Browsers indicate when HTTPS protocol is in effect by showing a padlock icon on the status bar.)

SSL support found in better load balancers is comprised of three functions: termination, certificate management, and acceleration. In order to understand what those features are, a bit of understanding about SSL is required.

SSL involves the encryption and decryption of data based on *public keys*, which are held in the form of digital certificates. A web site may use one or more certificates, and these certificates ordinarily reside on the web server. Whenever the web server sends data via the HTTPS protocol, it must encrypt the data using the appropriate certificate; whenever it receives SSL-encrypted data, it must decrypt it using the same certificate. Such encryption and decryption is handled automatically by the Web hosting software (and, conversely, by the user's browser); the host administrator must create and maintain the certificates.

Putting a load balancer in front of the web site requires that the digital certificates instead reside within the load balancer and that the load balancer perform the encryption and decryption. This functionality is referred to as *SSL termination*, because the SSL transaction effectively terminates at the load balancer: what the load balancer passes to the web servers is clear text.

The load balancer must also maintain the certificates, and provide a user interface to manage them. This feature is called *certificate management*.

A third feature found in some load balancers (and also in standalone appliances that exist for this purpose) is *SSL acceleration*. With SSL acceleration, the performance of SSL termination is greatly increased because the encryption and decryption is done using specialized software or hardware. SSL acceleration can be highly processor-intensive since the public keys can be quite large (up to 2048 bits). Web sites that process high volumes of SSL traffic would be best served by hardware-based SSL acceleration, not only to increase SSL processing speed but also to not burden the machine's processor from performing other tasks.

Offloading SSL processing to an appliance like a load balancer frees up web server processor resources and thereby increases their effective capacity to handle traffic. Furthermore, consolidating certificates onto a single device makes their management easier: if certificates need to be managed on multiple web servers, this must be done independently for each server. For multiple web servers that use the same certificates, only one certificate, resident in the load balancer, is required.

Different load balancers offer different certificate management support: some impose only a single certificate for all SSL traffic that passes through the load balancer, which can be restrictive (especially if multiple web sites are being hosted within the same environment). Others impose restrictions on the size of the certificates that can be used, which can cause compatibility problems.

If your web site uses SSL, you should be careful to ensure that the load balancer you intend to use provides sufficient certificate management capabilities to meet your requirements. You should also ensure that the load balancer's rated SSL performance can handle the volume you expect and, importantly, that the rating is based on the same certificate size you are using or higher, since certificate size has a significant impact on SSL acceleration performance.

LOAD BALANCER PRODUCTS

There are a number of load balancer products on the market. Commercially available load balancers vary in price and features, from entry-level models that offer bare-bones functionality and can handle low to moderate traffic volumes to full-blown routers and switches that include load balancing functionality.

Price differentiation is generally based on features, capacity, performance, quality, and, importantly, network interface speed (in the range of 100 megabits to 10 gigabits). While some load balancers advertise CPU speed and memory and disk capacity, those are typically not very important factors since load balancers tend to require little storage or memory capacity, and are not CPU intensive except for performing SSL-related encryption and decryption (which is best left to a dedicated processor anyway).

Software-only load balancers

Most commercially-marketed load balancers are network appliances: that is, specialized devices designed for their particular networking purpose. There are also software-only load balancers which can run on standard PCs.

In considering whether to use a network appliance or a PC running load-balancer software for load balancing, it should be remembered that one of the main reasons for using a load balancer is to safeguard against server failures. To host the load balancer on the same type of hardware that is prone to failures, attacks, etc. is not the most reliable approach.

Better load balancers are designed to be immune to the types of failures that can befall PCs. For example, as the most common type of PC failure is a hard disk crash, a good load balancer will have no hard disk and instead use a flash disk or other means of nonvolatile storage. It will furthermore not have vulnerabilities to viruses and other maladies.

INTRODUCING WEBMUX

WebMux is a fully-featured load balancer appliance from CAI Networks, Inc., which has over a thousand installations worldwide, including several large Internet hosting companies. WebMux is especially cost effective, offering functionality found in load balancers costing many times more, and unique functionality not found in other load balancers at any price. WebMux offers strong price-performance and low cost of ownership, with a three year warranty and three years of free updates and technical support with every WebMux unit, testimony not only to WebMux's quality but also its ease of use.

WebMux was designed by CAI Network's founder, who was at the time in 1997 managing an Internet company. After contacting various networking products companies and finding nothing on the market, he decided to create his own device. WebMux emerged onto the market in the same timeframe as the earliest load balancers, thereby defining a new networking product category.

MAIN WEBMUX FEATURES AND BENFITS

This section describes WebMux's functionality in the various areas mentioned above, and how actual WebMux customers use the product and the benefits they achieve in their own environments.

Load balancing

At the heart of WebMux's feature set is load balancing of multiple servers that comprise a *server farm*. Servers in the farm can have differing performance characteristics and contain different content. WebMux (as shown in *Figure 2*) allocates transactions to the appropriate servers based on installation-specific traffic management rules that can treat all traffic agnostically or direct the traffic to specific servers based on content.

In load balancing between identical servers, WebMux uses one of the following algorithms, selected as a configuration option:

- Round-Robin: Transactions are allocated one-by-one to each server in turn
- Persistent Round-Robin: Same as Round-Robin but persistent connections are enforced
- Weighted Round-Robin: Permits a *weighting value* to be assigned to each server that reflects the server's inherent power and capacity. Servers with higher weights means they are more powerful, and are therefore allocated more transactions
- Persistent Weighted Round-Robin: Round-Robin influenced by weight and enforcing persistence (stickiness)
- Least Connections: Load balancing is determined by the number of active connections to each server, and servers with the fewest connections are allocated more transactions in turn until their connection volume is level with the other servers in the farm
- Persistent Least Connections: same as Least Connections but persistent connections are enforced
- Weighted Least Connections: same as Least Connections but weight is incorporated
- Persistent Weighted Least Connections: Least Connections influenced by weight and enforcing persistence
- Weighted Fast Response: servers with the fastest response times are allocated more transactions, with weight imposed
- Persistent Weighted Fast Response: Same as Weighted Fast Response but persistent connections are enforced

WebMux determines the performance metrics required for decision making for the Least Connections and Fast Response algorithms via a regular health check, which it performs every few seconds against all servers in the farm. This health check can be performed sequentially or concurrently for all servers, based on configuration.

WebMux, like most load balancers, performs agnostic load balancing at OSI layer 4 and, like more fully-featured load balancers, intelligent traffic management at Layer 7; but unlike competitive load balancers,

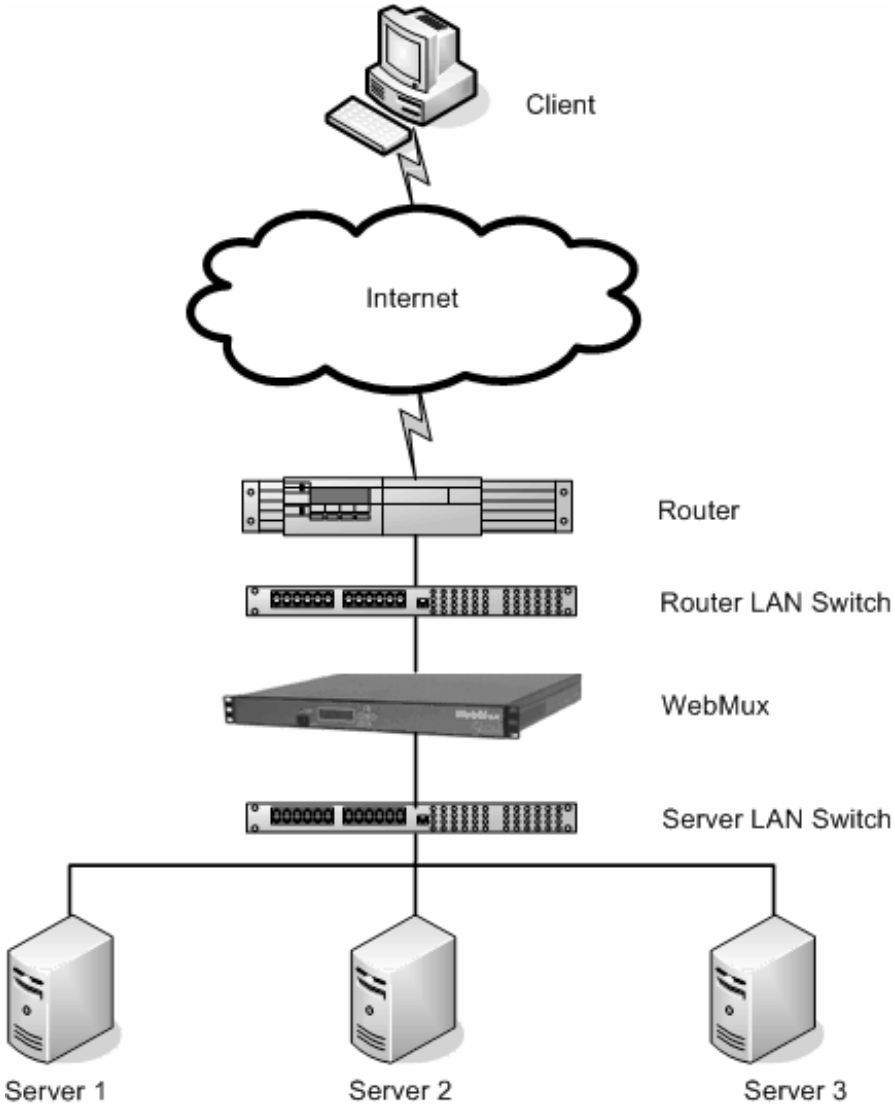

Figure 2

it can also load balance at layers 2, 3, and 5. Load balancer support of layers 2 and 3 is an exclusive and key WebMux feature, as it allows WebMux to behave as an Ethernet bridge, the advantage of which is that server IP addresses and network routes do not need to be changed to use WebMux. Layer 4 load balancing generally requires that server IP addresses need to be changed, as the load balancer assumes the IP address of the server – WebMux, by contrast, can accomplish Layer-4 load balancing without the need to change server IP addresses.

The United States Navy uses WebMux with three time servers to respond to current time requests from naval submarines and other vessels. In orchestrating naval operations, it is critical that all vessels coordinate to the same exact time, and WebMux ensures that this information can be delivered immediately and reliability on demand.

Traffic Management

WebMux can direct traffic based on content, whereby it will choose the server(s) that contain the content being sought based on a set of custom-defined rules. This permits priority to be given to certain users, by routing them to faster servers, and to host content on different non-identical servers and have WebMux route the related transactions accordingly. For performance assurance and failure protection, it is recommended that at least two servers containing each set of content are present in the server farm – WebMux will load balance them according to its normal load balancing methodologies.

For traffic management decision-making, WebMux exposes the full URL and query string, cookie properties, and cookie content to be used in rule definition. Thereby, the rule set for traffic management by WebMux can have fine granularity in that the web site or Web-based application designer can put any desired information into cookies and configure WebMux's rules to match on any cookie content. The ability to extract cookie content (not just properties) and use it for transaction routing is a feature unique to WebMux.

WebMux traffic management configuration is based on a match pattern in the host name, URL string, cookie properties, and/or cookie contents. Traffic filtering based on match patterns can be accomplished without the use of scripting: configuration is done via WebMux's Web-based GUI interface.

WebMux user- and content-oriented load balancing is performed at OSI layer 7, which effectively makes WebMux perform as a *content switch* (also known as a *Web switch* or *application switch*.). Rather than purchasing a separate content switch, WebMux can do the job in stride.

One WebMux customer, an information broker, offers a service that has a free and subscription option, whereby subscribers enjoy more functionality and better performance. This WebMux customer has a mix of servers having various content spread across them, with dedicated servers for subscription customers, and some of the content on common servers for non-paying and paying users. This customer uses WebMux's traffic management functionality to target subscribers to the dedicated servers, while non-paying users are directed to a different group of servers that offer lesser performance.

Failover and fault tolerance

WebMux's failover capability prevents unplanned downtime due to software and hardware failures, and facilitates planned downtime under administrator control.

The same health checking WebMux performs to determine performance metrics also gathers information to determine the availability of each web server and the integrity of all required protocols. WebMux's health checking methodology tests each protocol on each server that the administrator has defined needs to be operational for the server to be considered healthy. Each server can be configured independently, so, for example, the administrator can define that only HTTP and HTTPS protocols need to be checked for web site servers, while for FTP servers only FTP needs to be checked, for email servers POP and SMTP need to be checked, etc. WebMux can be configured to test all configured protocols serially or in parallel, the former of which requires less memory and is therefore the default. WebMux's health check can cope with flapping conditions, by testing each protocol over a sufficient time period.

Based on configuration, WebMux can either actively utilize all servers for maximum throughput or reserve nominated servers as "hot standbys" to be automatically brought into service in a failover event. In an all-servers-active configuration, in the event of a server failure WebMux automatically diverts traffic away from the failed server. In a hot-standby configuration, WebMux automatically diverts traffic from a failed server to its nominated standby. With WebMux, the same standby server can be designated for multiple primary servers.

WebMux can also take any server offline based on administrator intervention, in order to facilitate planned downtime for maintenance, backup, upgrade, etc., and will share the transaction load among the remaining available servers.

WebMux can furthermore safeguard against its own failure via redundancy, but allowing two WebMuxes to be run in a paired configuration, with one active and one standby (as shown in *Figure 3*). Should the primary WebMux fail, the secondary WebMux automatically takes up the work and notifies the configured administrator of the primary WebMux's failure. The two WebMuxes automatically synchronize configurations, and maintain heartbeats to detect failure.

In a failover event, WebMux, based on configuration, immediately notifies the administrator via email (which can activate a pager or other device). Notification is also given when the failed server returns online, which could happen automatically if the server recovers itself and if permitted by configuration. With the addition of an external modem, for which WebMux provides a port, more sophisticated notifications can be made.

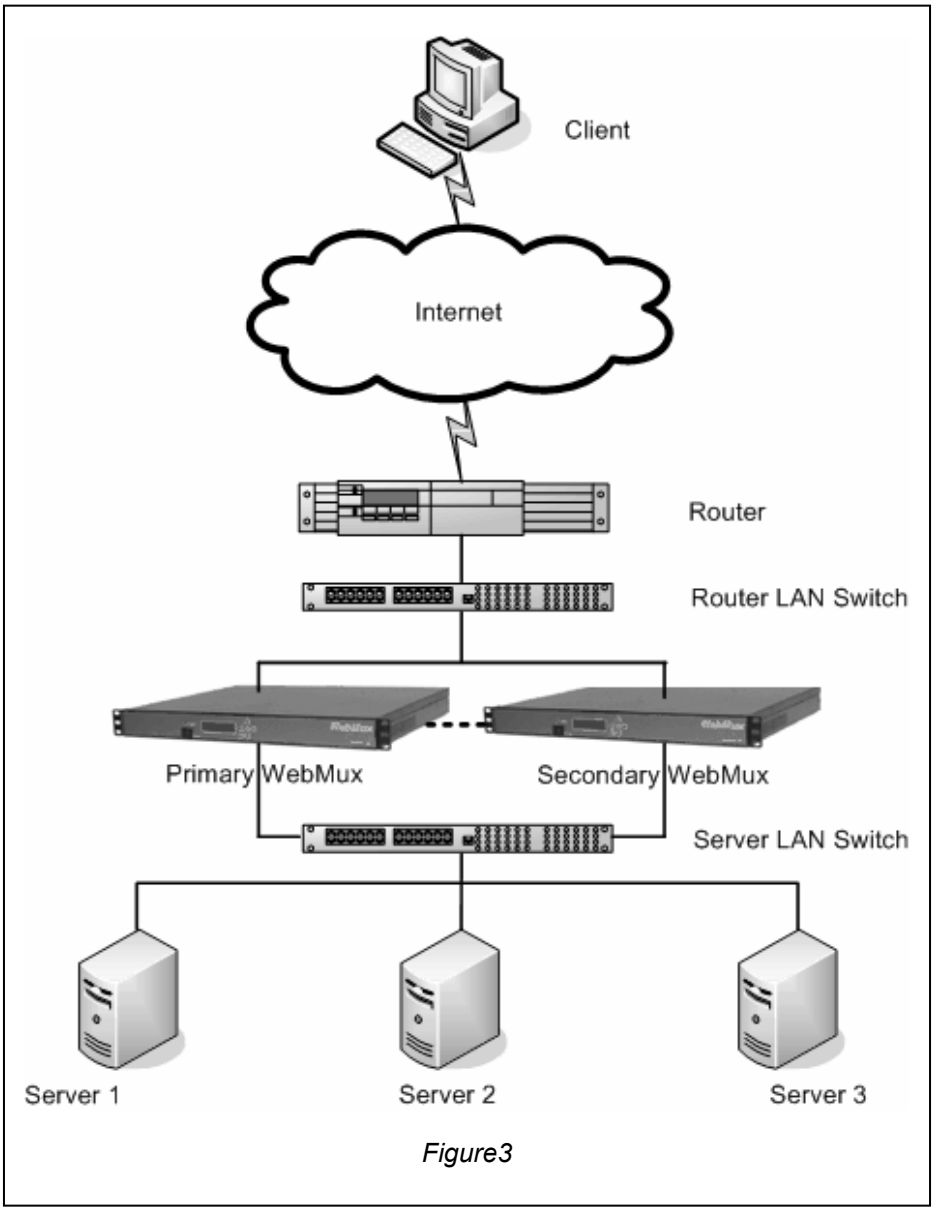

Figure3

A government agency in Sydney, Australia, uses paired WebMuxes spread out over distance between two campuses, with one WebMux and an identical server farm located at each campus. This configuration additionally provides protection against local environmental disasters, since if one campus goes offline the other automatically takes up the workload.

Connection persistence

WebMux can enforce persistent connections, whereby all transactions related to a particular browser session are connected to the same server in the farm. WebMux accomplishes persistent connections by recording the server to which each browser session is connected, and enforcing that connection with each new connection. Typical WebMux customers that take advantage of this feature include those with web sites that feature shopping carts, dynamically generated pages, those developed in BroadVision, and ASP- and JSP-based sites.

Persistence is a customer-invoked feature that is imposed at the server farm level. The time period for which persistence should be enforced for each session can be configured; typically a persistence duration of five to six minutes is imposed. The effect of the persistence duration is that all of a user's transactions during that period will be sent to the same server. Longer persistence timeouts can be specified (and often are) although this has the effect of reducing the number of concurrent connections WebMux can support, so this setting should be configured with care if very high numbers of connections are expected (WebMux's maximum concurrent connection limit exceeds 5.5 million in NAT mode – much higher in Out-of-Path mode – with 50,000 or more connections per second).

SSL support

WebMux offers robust support for SSL, either via software alone or with the assistance of an SSL-coprocessor based on an industry-leading SSL chipset. Thereby, WebMux provides not only SSL termination but also SSL acceleration, offloading CPU-intensive SSL-based encryption and decryption to a dedicated hardware processor. With this functionality, WebMux doubles as both a load balancer and SSL accelerator, eliminating the need to purchase a separate appliance for SSL acceleration.

SSL acceleration is measured in RSA bits TPS (terminations per second); WebMux offers two models of its SSL acceleration card: one that can execute 1200 TPS and the other 2400 TPS. Without hardware acceleration, WebMux can handle 200 SSL terminations per second. (These statistics are based on a certificate length of 1024 bits, which is most commonly used. Be careful when comparing performance of SSL accelerators as their quoted performance may be based on unrealistically short certificate lengths that can be processed more quickly, as less encryption and decryption is required).

WebMux also rationalizes certificate management, by consolidating digital certificates in one place – within the WebMux unit – rather than having them located on different servers. Typically, a host will impose one digital certificate for all SSL traffic entering the site, although WebMux can maintain up to 16 different certificates to be applied to one or more sites under WebMux load balancing. In this way, WebMux can streamline multi-site hosting within a common environment that utilizes multiple SSL certificates.

One WebMux customer who required SSL acceleration found WebMux to be the answer for complex certificate management, and chose WebMux primarily for that functionality. Another WebMux customer experienced such high volumes of SSL traffic after advertising a new product that without SSL acceleration they could have lost thousands of orders due to transaction abandonment, a common concern in web site hosting.

ADDITIONAL WEBMUX FEATURES AND BENEFITS

WebMux has additional features, which are not required for load balancing but which can speed implementation, ease configuration, save hardware costs, enhance security, and improve the user experience.

Firewall

Firewalls are software-based security barriers that are established to keep unwanted traffic out of the hosting environment. They are the first line of defense against hackers, attacks, and other malicious and unwelcome transactions. Firewalls can be implemented as software-only services installed on a computer, or as specialized appliances.

WebMux has a built-in firewall, which filters out common denial-of-service attacks such as TCP SYN and Ping of Death. WebMux's firewall also prevents unauthorized entry to the network by blocking all IP addresses except those associated with WebMux. WebMux's firewall will detect any possible DoS (denial of service) attack and ensure that required services are always available.

WebMux additionally has a Web-based configuration interface that can be accessed via TCP/iP from any browser, and which therefore can be accessed from anywhere in the world contingent upon configured access restrictions. WebMux's built-in firewall can prevent unauthorized users from reaching its configuration interface by allowing access from only authorized IP addresses. (An authentication check ensures that once the configuration interface is reached that only authorized users can gain access.)

Sites that do not already have a firewall, or who want additional protection, can expand the usage of WebMux's firewall. Depending on the complexity of firewall protections required, WebMux's firewall may be sufficient; in other cases not.

ADVANCED WEBMUX FEATURES AND BENEFITS

WebMux offers several advanced features to suit particular environments and purposes. Some of these features are exclusive to WebMux.

Health check customization

WebMux's health-checking process, which both determines server health and performance information, can be customized to suit particular environments. For example, in a three-tier architecture, in which an application server or ASP/JSP server depends on the health of a database server, it may be desirable in the event the database server is down to reduce the incoming traffic to the web server, suspend new traffic to the web server, or redirect all traffic to the web server. Such behaviors can be defined in CGI code, which must comply with certain requirements and be placed in the server path.

Customized health checking can also be used to dynamically adjust a server's weight to accomplish certain performance objectives, thereby causing the volume of traffic the server would be sent to be increased or decreased (in conjunction with the weights of other servers and in respect to the load balancing algorithm employed). WebMux permits virtually any server criteria (number of connections, CPU usage, memory usage, etc.) or even the ability to retrieve a specified web page to be used in evaluating web server performance.

Some WebMux customers found limitations with other load balancers, which offer only limited health-checking capabilities (such as PINGing each server to determine its health), and switched to WebMux to get a true determination of a server's ability to service traffic.

Multiple Address/Port Management

An emerging set of functionality in load balancing is that of multiple address management (MAP), which is concerned with advanced management of IP addresses and ports. MAP functionality increases flexibility, eases setup, reduces costs, and enhances the user experience. MAP functionality takes advantage of the load balancer's unique position in the network, by allowing IP addresses and ports to be related in new ways. WebMux is a MAP pioneer , being the only load balancer to thus far incorporate such functionality.

One of WebMux's MAP capabilities is the ability to host multiple IP addresses, and to multipurpose web servers within that addressing via ports. For example, say five web sites are hosted under multiple IP addresses and content for the various web sites is contained on one server per site. To provide load balancing and failover protection for those web sites would require at least doubling the number of servers (from five to ten). With MAP functionality, the load balancer can be configured to host all five IP addresses, and the content can be distributed across the existing servers and addressed by the load balancer by port, thereby eliminating the costs of adding servers.

Another WebMux MAP capability is concerned with applications that require more than one port to serve content, such as would be the case for a rich media application that uses a different port for audio, video, and control. In the normal case, should the service or protocol for one of the ports fail – for example, if the audio stopped – the video and control would still be operative but the user would experience no sound. MAP logical port binding permits the three ports to be related such that if any were to experience problems the traffic for all the associated ports would be diverted to another server.

One WebMux customer uses WebMux's multi-IP address capability to host a site accessible by several IP addresses on the same web servers without having to replicate them. This customer offers a hosted application service, where each customer is assigned a unique IP address for tracking, auditing, and billing purposes. In this configuration, WebMux is assigned multiple IP addresses and a path to each server pair on a unique port for each customer. WebMux balances the traffic so that each customer gets the performance benefit of all the servers. Without WebMux, a dedicated pair of servers would be required for each customer, greatly increasing hardware costs and maintenance work to keep all the servers up to date, and each customer's performance would be constrained to the server for that customer; furthermore, there would be no failover protection.

WEBMUX HARDWARE

WebMux is a high-quality appliance built on a hardened platform and using top-quality components. It is constructed using durable materials and has been successfully "drop tested", so it can not only survive the rigors of shipping but also any computer room manhandling.

To safeguard against hard-disk failures, WebMux instead uses a solid state flash disk, which provides durable storage and is uncrashable.

WebMux offers both single- and dual-processor models, to handle various workloads. WebMux models offer network interfaces up to 2 GB, while the next-generation high-end WebMux models will incorporate 8 GB network interfaces.

WebMux is a standard 1U rack-mountable appliance with a universal power supply, so it can be readily accommodated in any computer room anywhere in the world. WebMux's front-panel configuration keypad and LCD screen are backlit to make them visible in dark hosting centers and other lights-out environments.

IMPLEMENTING WEBMUX IN YOUR ENVIRONMENT

WebMux customers normally implement the product themselves, sometimes consulting CAI Networks technical support for advice on setup and configuration. If advice is required, it is normally about which of WebMux's three networking modes to use.

WebMux offers three networking modes to suit various network setups and customer requirements. The three modes are:

- In-Path mode (also known as NAT mode)
- Out-of-Path mode (also known as One-Leg mode)
- Transparent mode (also known as Bridge Mode)

In In-Path mode (*shown in Figure 4*), WebMux assumes the IP address of the web server it is standing in for, and the web server (and additional web servers that are added) are assigned internal IP addresses. Thereby, traffic destined for the web server by its IP address instead reaches the WebMux, and the WebMux allocates it among multiple web servers.

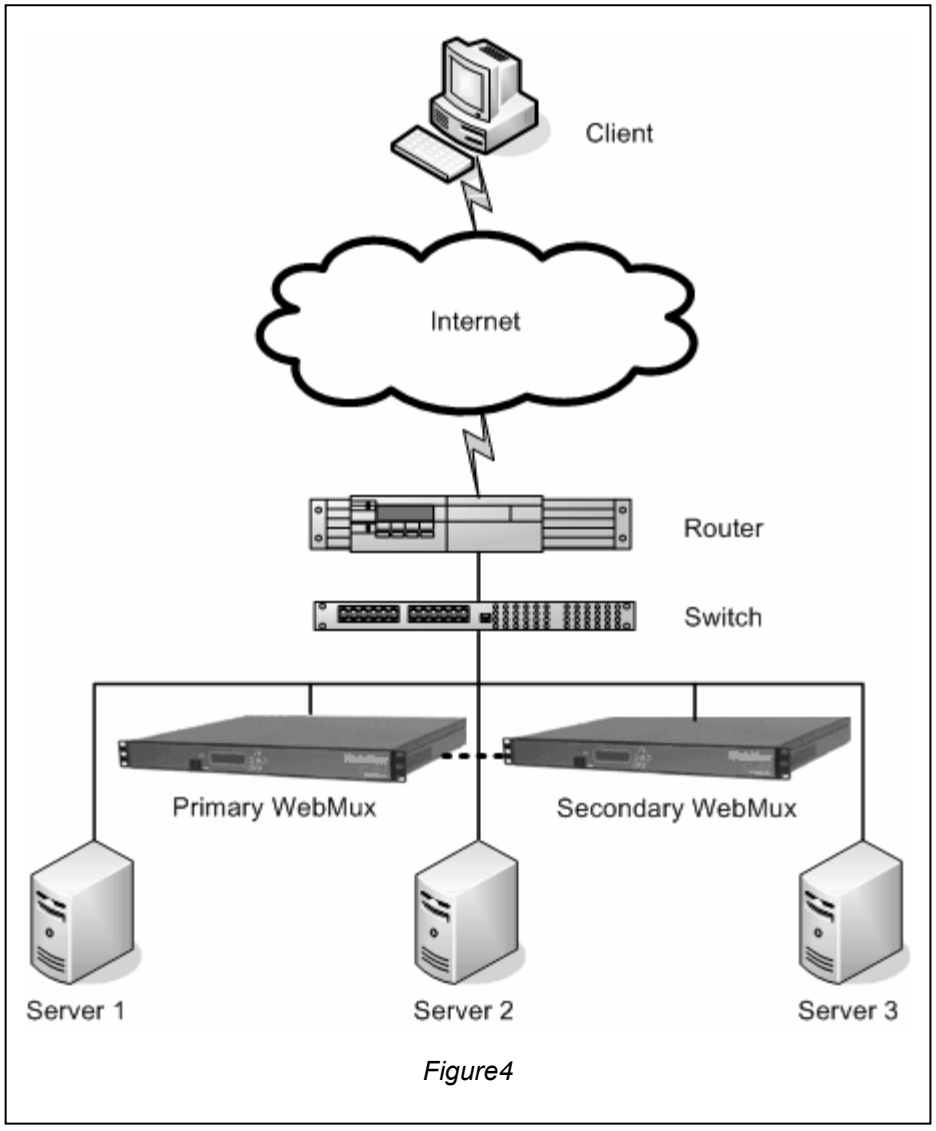

Figure4

In Out-of-Path mode (shown in *Figure 3*), WebMux is assigned a new IP address on the LAN network; existing servers can retain their IP addresses but must be equipped with loopback adapters. The firewall in front of the WebMux is configured to redirect traffic for the web server's IP address to WebMux's IP address, thereby traffic targeting the web server's IP address is reaches WebMux, which in turn allocates it among multiple web servers.

In Transparent mode, WebMux behaves as an Ethernet bridge between the servers and the router LAN: it is assigned a new IP address, and existing servers can retain their IP addresses and no loopback adapters are required, but if WebMux is run in a redundant paired configuration it must be bookended by routers that support Spanning Tree Protocol (STP).

ADDRESSING GLOBAL REQUIREMENTS

This paper has discussed addressing web server availability, performance, traffic management, and other issues in a local environment; but as described in the introduction, such challenges are not limited to the local environment. The Internet and World Wide Web are global entities, and so web site accessors are

located everywhere in the world. Increasingly, as Web technologies form the basis for even mission critical applications, and as users for even those applications become more widespread, the need to address the same concerns on a global basis mandate focus.

Many businesses and organizations have elected to host their web sites, Web-based applications, and other computing resources at multiple physical sites, either to offer improved performance for users in various geographies (by reducing latency time) or to provide immunity from disasters and other environmental problems that can befall one or more sites, or both. A further common requirement is to connect users with the most appropriate non-identical site based on one or more affinities, for example, targeting users in Spanish-speaking countries to a site implemented in Spanish (perhaps hosted in Spain).

If your business or organization has chosen to adopt a multi-site hosting strategy, or if this is something of interest, you may want to read on in this section.

While the approaches and technologies toward addressing performance, scalability, and availability in geographically distributed environments are different than for local environments, the problems and desired benefits are similar.

The general approach to load balancing, traffic management, and continuous operation assurance in a multi-site environment present a number of challenges, including:

- Ensuring high performance by directing users to the sites geographical closest to them
- Diverting users away from sites that are down
- Targeting users to the most appropriate sites based on content (as determined by locality, language, or other affinity)

A geographical load balancer and traffic manager, while providing the same types of benefits as a local load balancer/traffic manager, is quite different in its implementation: First of all, unlike a load balancer, it is not bound to a particular site, since it needs to be able to allocate transactions among various sites. It would not be sensible to add an additional layer within a site, since once a transaction reaches a particular site, even if the site was able to redirect the user to another site, the time taken to do that would have a substantial negative impact on performance; also, if the site was down, the transaction could not be routed.

So a geographical load balancer is generally implemented before the network, as a replacement DNS server – one that is intelligent enough to answer each hostname-to-IP-address question with the IP address of the most appropriate site rather than an arbitrary site, based on the user's proximity to and affinity with each available site.

CAI Networks' DnsMux™ is such a solution (deployed as in *Figure 5*). DnsMux allows each site to be defined along with targeting rules based on proximity (the geographical closeness of a user) and affinity (what users should be connected to which sites). Whenever a user's configured local DNS server requests a hostname-to-IP-address conversion, DnsMux automatically determines the originator's geographical location, checks its configuration and the current operability and performance of the various servers at the various sites under DnsMux's management, and replies with the IP address of the most appropriate site. DnsMux does not connect the user to the server; rather, the normal DNS protocols are used to accomplish that based on DnsMux's intelligent hostname-to-IP-address resolution.

Like WebMux, DnsMux can be deployed in a redundant configuration for fault-tolerance purposes, but additional DnsMuxes can be deployed to ensure the best performance. It is generally recommended to have one DnsMux per geographic site, so that the latency time required for any user's DNS request to reach the DnsMux is minimized.

Also like WebMux, DnsMux performs regular health checking of web servers to determine their availability and performance, and has a number of algorithms to choose from for allocating incoming traffic among multiple servers at a site. Since, based on DNS protocols, any DnsMux in the cluster can be called upon

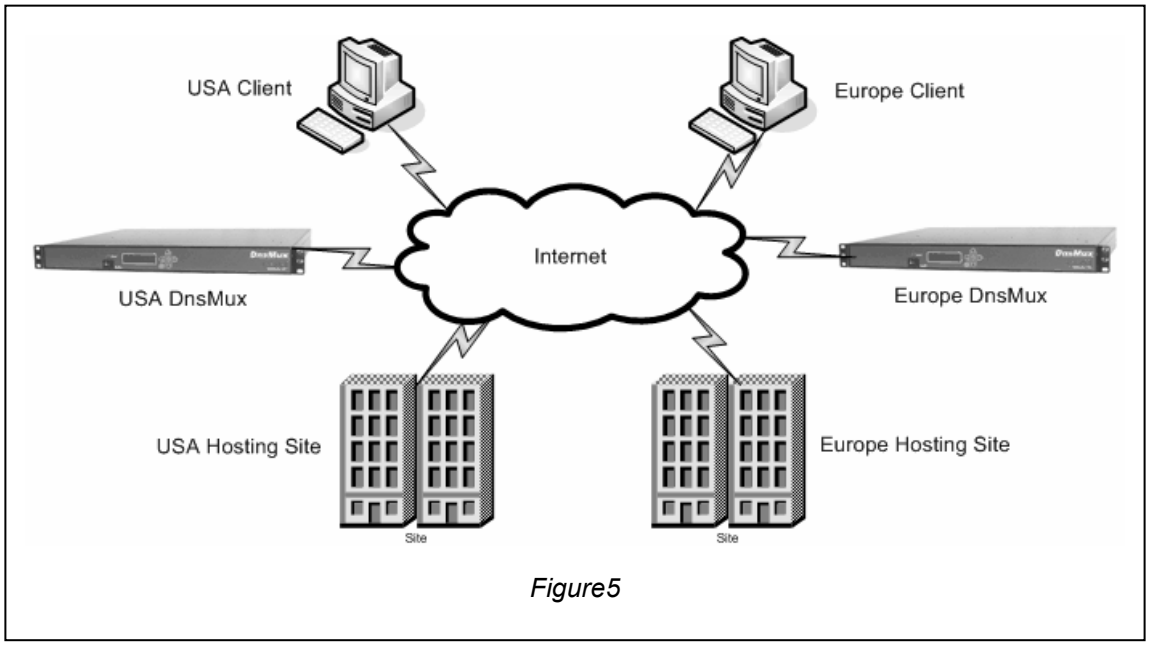

Figure5

to reply to a conversion request, all DnsMuxes in a cluster regularly health-check all web servers so that they all have performance metrics to be used in targeting. Furthermore, all DnsMuxes in the cluster maintain a heartbeat with each other to determine if any are inoperative. If so, the administrator is notified, and DNS protocols automatically try another DnsMux to service the request (and another and another until the request is replied to).

Thereby, the functionality provided by DnsMux, and other good geographic load balancing and traffic management solutions, ensure that users will automatically be connected to the best performing server based on their localities; furthermore, such proximity-based targeting can be influenced via a robust configuration interface to target users to the sites and servers that contain the content most appropriate for them.

A geographical solution is not a replacement for a local solution: in fact, the two compliment each other: the geographic solution gets the user to the best site, and the local load balancer gets the user to the best server in the site (although some geographic load balancing solutions, like DnsMux, are also capable of connecting the user to the best-performing server at the best site).

Sites that are geographically distributed will benefit from DnsMux or equivalent geographic load balancers in achieving peak performance and the most appropriate content for users, no matter where on the planet they are located.

CONCLUSION

The world of computing has come a long way since the days of the single server that ran the business. Increasing systems complexity has been paced by increasing user demands for availability and functionality. Downtime – whether planned or unplanned – is no longer acceptable. The trend toward applications built upon Web-based technology brings the challenges inherent in ensuring web site availability and performance to core applications, including mission-critical applications that form the computing basis of the business.

With the trend toward wider use of Web-based technologies, load balancers become more critical than ever, and are a need-to-have in contemporary computing environments.

Like any other reliability exercise, ensuring web site and Web-based application reliability means identifying the potential points of failure and implementing solutions to mitigate or eliminate them. While redundancy is an effective approach to addressing reliability issues, the requirements of problem detection in the primary environment and switchover to the standby environment need to be addressed. Local load balancers like WebMux are specifically designed to address the risks associated with web server uptime, while maximizing capacity, performance, and functionality. DnsMux extends these benefits globally, to ensure peak uptime, performance, and content delivery to the world.