

# Spirent Server Load Balancing Testing Methodology

---

Application Note 05  
January 2004



Spirent Communications  
1175 Borregas Avenue  
Sunnyvale, CA 94089  
Phone: 408.752.7100  
Fax: 408.752.7101

[www.spirentcom.com/avalanche](http://www.spirentcom.com/avalanche)

Copyright 2004 by Spirent Communications, Inc. All rights reserved.

## Table of Contents

<b>Overview .....</b>	<b>1</b>
<b>Evolution of Server Load Balancers .....</b>	<b>2</b>
<b>Connection-based Server Load Balancing .....</b>	<b>3</b>
<b>Request-based Server Load Balancing .....</b>	<b>4</b>
<b>Request-based SLB with integrated proxy .....</b>	<b>6</b>
<b>Determining the Right SLB Platform .....</b>	<b>7</b>
<b>SLB Performance Metrics .....</b>	<b>8</b>
<b>Inadequate Measurement: A Recipe for Disaster .....</b>	<b>10</b>
<b>Realism Testing of Your Server Load Balancer using Spirent .....</b>	<b>13</b>
1. SLB Basic – Establishing a Reference Baseline Test .....	13
Objective: .....	13
Requirements: .....	13
Operational parameters to establish: .....	14
Running “SLB Basic” .....	14
2. SLB Stress – Stress-Testing Your SLB.....	16
Objective: .....	16
Operational parameters to establish: .....	16
Running “SLB Stress” .....	16
<b>Summary .....</b>	<b>21</b>
<b>Background .....</b>	<b>22</b>

## Overview

Today's enterprises are feeling the constant pressure of increased network traffic against their web servers. Industry analysts agree: Internet traffic and web-based transactions continue to grow—and this growth is only expected to continue. In response, web application and network design engineers must optimize and rescale their Web sites and other critical network infrastructures—with an eye toward efficiency and cost-effectiveness.

More often than not, enterprises turn to server load balancers (SLB) as an effective and inexpensive way to improve the performance of their web sites. Some enterprises also implement SLB technology in order to optimize the interaction of the network application. Unfortunately, enterprises often find that, once integrated into their network infrastructure, the performance of the SLB is sub-optimal or not-as-advertised, or that the SLB they've implemented doesn't integrate critical functions, such as security, secure socket layer (SSL) acceleration, or application acceleration.

In its effort to optimize a web site, SLBs can unintentionally act as an application performance bottleneck. Worse, changes to the configuration of the SLB can reduce the load balancing performance to the point of being a bottleneck for the entire web site. Even in the best-case scenario, it is difficult to rely on vendor benchmarks because the real performance of an SLB is extremely dependent on the environment. So, while SLB vendors can provide the optimization tools to enable a web site to interact with the network more efficiently and securely, they cannot tell you how *your* web site will perform in *your* particular environment—and with *your* specific business requirements.

To ensure realistic end-to-end network-operating performance and availability, web application and network design engineers must conduct their own testing based on intimate knowledge of their content and network operations. And, as SLBs are extended to include other functions—such as security, SSL acceleration, application acceleration, and firewall functionality—it is important to understand how these functions interact and their impact on overall system performance. For example, the number of SSL connections implemented can impact the total number of connections for load balancing. Further, integrated SSL can impact the performance of other functions.

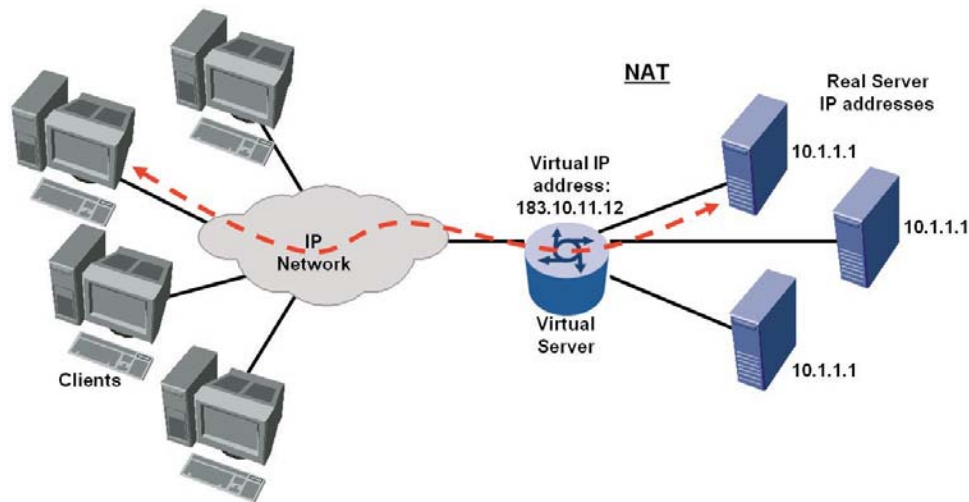
Spirent Communications, the leader in network testing and capacity planning technologies, offers a wide-range of testing methodologies to enable you to determine the real-world performance of your SLB in *your* particular environment, including SSL, firewall, and general testing methodologies.

This application note will assist you in comprehensively testing your SLB using Spirent-recommended test methodologies. By creating and running the basic, stress, and load tests discussed in this paper, as well as by calibrating your SLB, you will combine *the realism of production networks with the precision of lab testing*. In doing so, you can validate the ability of a server load balancer to properly balance network traffic for your particular Web site in your production environment.

**Note:** This application note also assumes that the reader is familiar with configuring and running basic tests using Spirent's *Avalanche™* and *Reflector™* appliances and/or the *TeraMetrics* cards in Spirent's *SmartBits* chassis.

## Evolution of Server Load Balancers

A server load balancer, or SLB, is a device that distributes connections to servers while masking the real IP addresses of the servers via a single, virtual IP address. This allows real servers to be added to, or removed from, a Web site without taking down the entire site (see Figure 1a). Most often, SLBs are added to the web site architecture with the goal of improving the site’s overall performance.



**Figure 1a:** Since the DNS name [www.mysite.com](http://www.mysite.com) is tied to the IP address 183.10.11.12, real servers can be modified without updating the DNS.

To understand how to properly evaluate SLBs, it is important to understand how the server load balancing technology has evolved to address advances in network and application infrastructure. The evolution of server load balancing requires discussion of three types of SLB technologies:



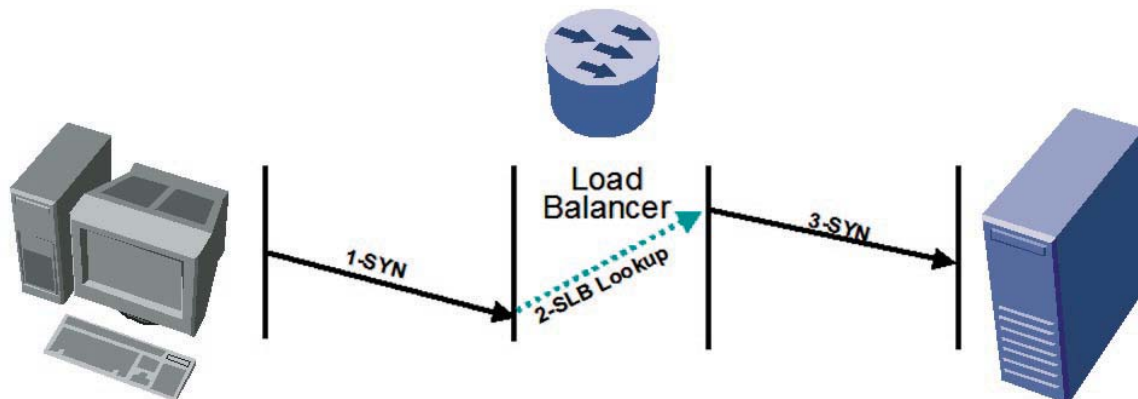
**Figure 1b:** The Evolution of SLB Technology

The remainder of this section will describe each of the three SLB technologies, along with the benefits and shortcomings of each.

## Connection-based Server Load Balancing

When first introduced to the market in mid-1990s, server load balancers distributed requests to mirrored servers with duplicate content. However, this approach often led the SLB to break applications that were not designed to be distributed. For example, when customers had Web-based “shopping carts” stored in a server, the shopping cart would be lost when the next request hit another server, leading to lost revenues and frustrated customers. Applications have since addressed this issue by storing shopping cart information in a back-end database; however, this results in a new database query for each request that lands on a new server, slowing performance. Additionally, this resource-intensive process inflicts a massive number of requests on the database server and leads to costs that can run as high as \$40,000 per database CPU<sup>1</sup> and \$160,000 for a quad-processor machine.

Enterprises later discovered that separating static web servers from specialized application servers could serve requests to each type of server many times faster, and was typically accomplished by using more CPUs in the application server than in the web server. However, many problems still existed with connection-based server load balancing, which simply distributed information based on the first received data packet.



**Figure 2:** Connection-based server load balancer operation

First, because connection-based SLBs decide how to route traffic based on the first SYN packet, which initiates the TCP connection, many connection-based Distributed Denial of Service (dDoS) attacks would get through the load balancer to the server, and since no cookie was in the packet, applications would break or servers would simply crash.

Connection-based server load balancers determine which server will see the connection based on the information in the first SYN packet. Where to route the connection is based on the following metrics:

- Round robin (IP or TCP)
- Weighted round robin
- Least TCP connections

- Fastest TCP connections
- Maximum connections
- Non-stateful SYN “rate limiting”

The SLB compares the information in the first packet against its internal information. For example, the number of connections on a given server is stored on the SLB. When a new packet arrives, the SLB makes a decision based either on stored information, such as server TCP connection resources, or on external information that can be found in the packet, such as IP address load balancing. In the latter case, all odd-numbered clients are routed to server 1 while all even-numbered clients are routed to server 2, and so on.

Second, while it is easy for a server load balancer to track information based on a single SYN packet, there are many other operations that SLBs cannot track or catch. For example, in a SYN attack, the SLB has no way of distinguishing the attack from a normal connection. Non-stateful SYN “rate limiting” blindly limits new SYN traffic to the server based on the rate or the SLB connection table size. If the connection table becomes full, the SLB reduces the rate at which it accepts connections, decreasing the amount of valid traffic the SLB can serve and impacting business-critical transactions.

Information in the HTTP request to the SLB can protect the Web site from these types of SYN attacks and enhance load balancing, including:

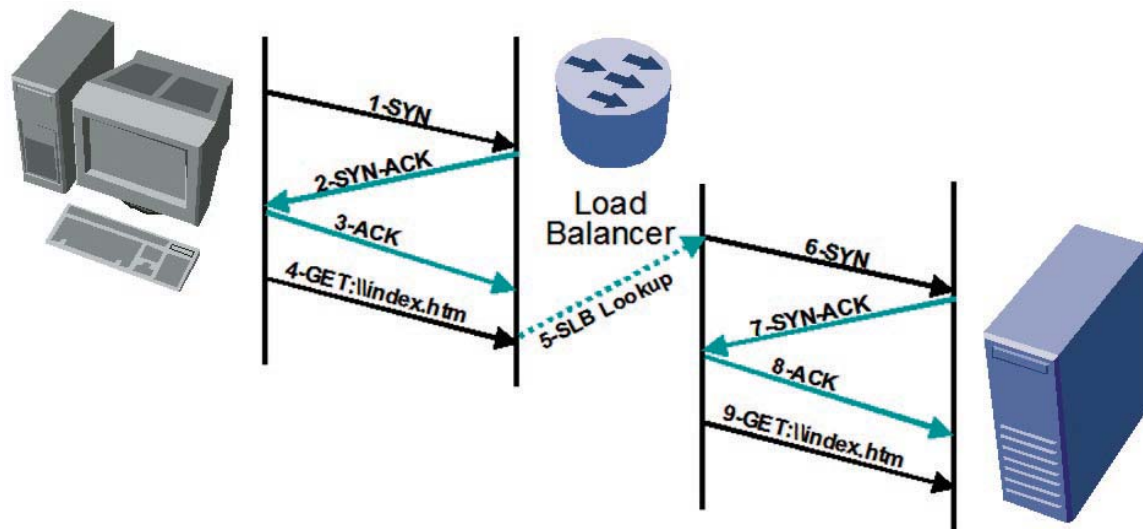
- URL – Optimizes server farms according to function by separating out the static and dynamic content.
- SSL ID – Keeps the same client connected to the same SSL server/accelerator so that the device can increase performance by taking advantage of SSL session reuse.
- Cookie - Assign a customer to a server to prevent a query to the back-end databases or for application servers that do not use databases.
- Browser type – Identifies how the Web site represents data.

The realization that HTTP request information can be used to enhance server load balancing led to the evolution of the request-based SLBs, which will be discussed next.

## **Request-based Server Load Balancing**

As discussed in the previous section, the connection-based SLB passes the SYN packet directly to the server and then expects the server to respond with a SYN-ACKnowledge. Request-based SLB technology takes this to the next step by identifying the content of the request.

In request-based server load balancing technology, the SLB can not only observe what is being requested—e.g., URL, SSL ID, cookie and/or browser type—but it can also redirect requests by terminating the TCP state referred to as “deep packet inspection” in firewall terminology. TCP termination also has the benefit of keeping users connected to the same server throughout the transaction, via sticky cookies, so that shopping carts can be kept on the appropriate server. In order to identify the content of the request, a connection-based SLB processes four times as many packets as a request-based SLB before making a load-balancing decision (see Figure 3).



**Figure 3:** Request-based server load balancer operation

#### Request-based SLB Technology

Positioned between the client and the server connection, request-based server load balancers identify the content of a request by:

1. Terminating the 1-SYN on behalf of the server.
2. The SLB then 2-SYN-ACKs the SYN.
3. The client ACKs when it receives the SYN-ACK.

(At this point, there is a TCP connection with the SLB and not the server.)

4. The client sends an HTTP GET request that contains the URL, SSL ID, cookie and browser type. The SLB can use this information to determine the destination servers.
5. The SLB uses the information to forward the request to purpose-built server groups.

Note: One specialized application server group can be tuned to handle dynamic application traffic while another server group can be tuned to handle static requests like images, HTML, etc.

In addition to TCP termination, request-based load balancers are often used to conduct:

- URL load balancing/images
- URL “304/302” redirection
- Cookie sticky (insert or passive)
- SSL Session ID sticky
- TCP connection multiplexing
- Stateful dDoS protection

As with TCP termination, each of these functions requires deeper inspection of the connection packets, with the accompanying performance hit: request-based load balancing is two to 100 times slower than connection-based load balancing. And, since more information exists in the connection tables of request-based SLBs, connection capacity can also be negatively affected. For example, an SLB supporting one million connections with connection-based load balancing may only support 64,000 connections with request-based load balancing.

Web requests also perform more slowly in request-based SLB configurations than in connection-based SLB configurations. In request-based SLBs, all connection-level attacks (e.g., Ping of Death, Teardrop, SYN Floods, Land-Based Attacks) are processed by the SLB—with the SLB acting as a firewall. For further information on this topic, please read the *Firewall Performance Testing Methodology* at <http://spirentcom.com/documents/I062.pdf>

Further exacerbating the performance problem of request-based server load balancing is how the SLB handles TCP packet recovery. Recovering packets, referred to as TCP retransmission, slows down applications while waiting for TCP timeouts. Some request-based SLBs offload TCP from servers to “optimize” the server, and then take on server-like performance characteristics—creating a new performance bottleneck rather than alleviating the problem.

To address the issues with request-based server load balancing, a new breed of server load balancer entered the market: request-based SLB with integrated proxy.

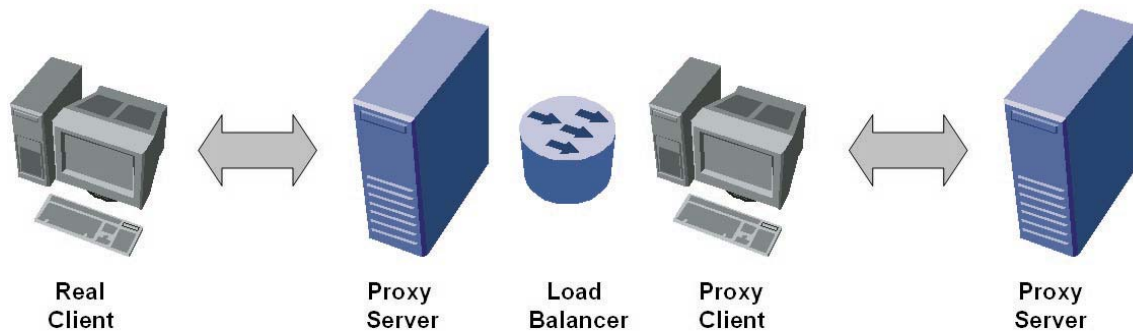
## **Request-based SLB with integrated proxy**

Request-based server load balancers with integrated proxy (or TCP retransmission) functionality were developed to overcome the shortcomings of request-based and connection-based load balancers. By adding proxy server functionality to support features such as FTP load balancing, content caching, authentication, compression and offloading TCP connections and TCP data (retransmissions), request-based SLBs have optimized the server farm and allowed enterprises to do more with fewer servers and less bandwidth.

In some cases, request-based SLBs with integrated proxies operate as the single point of authentication for the web application, allowing the SLB to optimize TCP and web access in one platform. However, this added functionality requires more processing cycles than a request-based SLB because the load balancer is now acting as a proxy web server. As previously discussed, a request-based SLB simply passes retransmissions to the server since it only cares about the request. By adding TCP retransmission, the SLB frees up the server to concentrate on other aspects of the web application. This is particularly beneficial for features such as:

- Integrated SSL offload
- TCP multiplexing and TCP buffering
- Authentication
- Content caching
- Content compression
- FTP load balancing
- Stateful application firewall

In all of these configurations, the SLB is responsible for TCP retransmission because it must fully proxy the request at the Web application (as shown in Figure 4).



**Figure 4:** Request-based server load balancer with TCP retransmissions

Since the SLB must serve all traffic, it must contain a proxy server and a proxy client to support the features listed above. When a real client issues a SYN, it lists the size of the client's TCP receive buffer. If the client is a Microsoft Windows 2000/XP client, the proxy server inside the SLB must reserve 16KB of memory for TCP retransmission, meaning each client connection requires 16KB of memory. To put this in perspective, a server load balancer claiming it can handle one million connections with SSL offload enabled would need  $16\text{KB} \times 1,000,000$  connections—or 16GB—of TCP buffer memory!

TCP retransmission is utilized when handling SSL offload requests and the SSL hardware resides within the SLB. It is also utilized if the SLB is handling web content caching and authentication. The only case where the device under test (DUT) can claim full TCP offload with retransmission and use less memory than the above calculation is when the SLB can selectively respond to ACKs. In this situation, the SLB passes the ACKs directly to the server if its TCP buffer memory is overwhelmed. When the TCP buffer memory is overwhelmed, the acceleration with retransmission reduces the number of servers. The limit is determined based on end-to-end testing, and after this has been reached, additional appliances must be added. In some cases, a 1:1 appliance to server ratio is suggested to allow the appliance and the server to have the same operating performance.

Now that you have an understanding of each of the three server load balancing technologies on the market today, and their pros and cons, you are ready to determine which technology and platform is right for your particular requirements.

## Determining the Right SLB Platform

Finding the optimal SLB solution for your network and infrastructure requires understanding the performance of every component in the equation—server load balancer, firewall, and application server—to ensure that the SLB technology does not inhibit end-to-end site performance. While integrated SLB technology offers greater manageability and incorporates newer techniques, such as buffering and authentication, critical questions must be answered to determine which SLB platform is right for you:

- In searching for an optimum solution, does the scalability of your bandwidth, SLB infrastructure and servers match? Are you paying too much for bandwidth or server maintenance? Will the SLB be slower than the servers and/or available bandwidth? How many servers can be load balanced per SLB?
- How much user application load your site can handle? How much does this change given the real world conditions of user behavior and user bandwidth applied to a real Internet with dropped packets, network bandwidth restrictions and added latency?
- Are the SLB, server platform, CPU and memory constrained to the point where, as an inline device serving multiple functions, the SLB's saturation throttles infrastructure availability and scalability?
- By consolidating multiple services, did the SLB truly optimize end-to-end performance? Did the aggregation of many features reduce performance or scalability?

Spirent offers a comprehensive methodology along with detailed testing to help you address these questions and ensure that you make the optimum investments in server load balancing equipment to offer customers the best possible experience. In addition, using Spirent to conduct performance testing enables you to inventory your IT assets and determine if and where excess exists, and if and where additional investments are required.

## SLB Performance Metrics

There are several important terms that must be defined and understood before evaluating SLB performance metrics:

- **Connection:** A single logical TCP communication channel between the client, SLB and server that is used to transfer one or more transactions or requests.
- **Session:** A single logical user communication channel between the client, SLB and server that is used to contain one or more connections, transactions or requests. One user "session" can contain many parallel or serial TCP and HTTP requests to a given site.
- **Transaction:** A single or multiple request(s) for object(s) made bi-directionally from client to server inside a connection. Both HTTP 1.0 and HTTP 1.1 can issue and receive multiple transaction/requests per TCP connection.
- **Throughput (bandwidth):** The amount of data that the physical communication channel can carry. In SLB environments, this is a collection of connections and transactions using your web object size on your web site applications. This is why testing with your own web site content is critical to understanding how to optimize performance.

Typically, vendors quote several performance metrics when evaluating an SLB implementation:

- **Connections per second:** The rate at which new TCP connections can be introduced. This metric provides an indication of how many TCP connections can connect to the website.

### Things to Watch When Testing Connections per Second

Two issues compromise the accuracy of connections per second (CPS) metrics. First, *measuring CPS, as it relates to active connections, can vary depending on the number of active connections established*. In the real world, new CPS must be measured while the SLB has thousands of established connections. These established connections take up memory. Vendors typically benchmark CPS using a minimal number of established connections. Conversely, when connection capacity is measured, it is done with a minimal number of new CPS. Second, *using CPS, as it relates to object size, can vary depending on the number of transactions per second (TPS) and the object size being requested*. This is why it is important to test the extremes of each metric and understand how vendors present this information.

In testing connections per second, vendors typically use the smallest object size (for example, eight bytes of data). When measuring throughput, vendors typically use a 512KB object. The reality is that using a combination of large and small object sizes found on your Web site provides you the most accurate representation of true site performance. Combining this with true application testing will also provide an understanding of the “application effect” against varying object sizes and more truly evaluate new equipment in the infrastructure. Typically, when there is one connection and one request, the SLB vendor is quoting the number of CPS. If the vendor is quoting the TPS number, this means there is more than one transaction per connection. In addition, both of these numbers can be specified via configuration. For example, Layer 4 CPS would be one value where Layer 7 CPS would be another lower value (due to the additional TCP processing). The danger is that there is no standard in which to measure the SLB implementation, and this is why measurement is crucial to reducing the risk of operational and design implementations. It is critical that you use one transaction (such as a single HTTP GET request) per connection as a baseline when testing CPS. Expanding this test to emulate your entire site is a final step in evaluation.

- **Transactions per second:** The rate at which the total number of new transactions can be introduced within established connections, this metric provides an indication of how the SLB will perform with multiple requests over a single connection. For example, if a Web site has 20 objects on the first home page and these 20 objects are transmitted in one TCP session and one connection, each object is a transaction. Sometimes vendors use transactions in place of connections – in this case, a single transaction will map to a single connection if using HTTP 1.0 with no keep-alive or HTTP 1.1 without persistence. The same effect can be also seen with HTTP 1.1 persistence and one URL request. Either HTTP 1.1 with persistence or HTTP 1.0 with keep-alive and multiple objects being requested will provide multiple transactions within a single TCP connection. This can help optimize network resources by avoiding having to set up a new TCP connection every time a HTTP request is needed.
- **Throughput:** The amount of bandwidth consumed using very large objects. Since bandwidth is factored by object size and TCP parameters like window size and end-end latency, most vendors benchmark with a large object that usually is not representative of your Web site. This is why focusing on Web site throughput is a design goal

that can realistically only be achieved through lab testing using average object sizes found on your Web site combined with the *Avalanche*'s ability to create measured user latency and Internet loss.

- **Number of concurrent sessions:** The number of concurrent sessions that can be supported at any single moment. This value is usually determined by how much memory is available to maintain a given session state in the system.
- **SLB CPU utilization:** In addition to the typical performance metrics, one important, and often overlooked, metric is SLB CPU load. This is critical in determining the load from processing requests and health checks to the servers. Measuring the successful transactions and comparing them to the SLB CPU load gives network operators a key indicator of when SLB device performance will negatively affect users. Since this can be probed using the avalanche and its SMTP polling capabilities during test, during production creating a real-world network operations deliverable of finding a “breaking point” of any device which can be enforced during the operation of a network. For example, when the SLB CPU is more than 75% utilized it can add several seconds to each user's page-load time. In many instances, the load time is the critical factor to understanding how the user is impacted by load on the SLB infrastructure.

## Inadequate Measurement: A Recipe for Disaster

Evaluating server load balancers is fraught with risk. The potential for risk is not just from your network architecture – if it can sufficiently handle what you want it to do – but also from increasingly sophisticated server load balancing technologies. How can these operational risks be mitigated? How do you ensure both optimized performance and availability? It is not just a matter of functionally verifying operation or that your device's CPU and memory are sufficient at some benchmark that likely doesn't mimic your network operation. Even slight SLB configuration changes can have a dangerous effect to performance and availability. Any configuration change can result in a massive performance hit that can cripple your web site.

With this said, it is still possible to realistically stress test all relevant SLB configurations to determine the following metrics:

- SLB breaking point
- Number of connections and new CPS per configuration
- New CPS while under connection load
- New CPS under dDoS attack
- New CPS while under load and dDoS attack
- Total number of concurrent connections supported by the SLB and servers
- Accuracy of the SLB load balancing algorithm
- Throughput, while stopping attacks to ensure 24/7/forever availability
- Ability to stop well-known denial-of-service attacks while under load

- Number of new CPS per configuration using multiple URL rules to block application attacks such as NIMDA and Code Red
- Ability to simulate dDoS attacks
- Number of health checks and their impact on SLB performance and scalability
- Multi-protocol load balancing consisting of HTTP, HTTPS, RTP/RTSP, telnet, DNS, and SMTP/POP3 protocols
- Failover in an active-active deployment
- Ability to ensure there are no memory-leaks for short-lived connections

**The Risks of Lab Testing vs. Real-World Conditions**

When compared with vendor datasheet numbers, there was a 60% reduction performance in connections per second for L4 connection-based SLBs, and a 90% reduction performance in connections per second for L7 request-based SLBs. The following table summarizes the risks of using lab testing and extrapolation versus real-world behavior.

Lab	Real-world	Risks
Single device	Multiple devices	Interoperability End-to-end security and availability Non-linear scalability
Controlled traffic	Complex traffic	Uncharacterized performance, security and availability under multiple-protocols
Few networking devices	End-to-end networking devices	Interaction between multiple devices
Simple network	Complex Internet behavior	Latency, packet loss (others)
Controlled load tests	Variable loads	Performance under widely-varying loads

The intricacies of *your* particular operating environment also increase the risk. Neither vendors nor third-party laboratories can recreate your network or application realism to uncover the load characteristics necessary to determine your scalability requirements.

Simply extrapolating vendor benchmarks or typical lab testing equates to inadequate measurement and a recipe for disaster. For example, vendors will use the highest performing configuration to showcase their device. Some will not quote Layer 7 performance numbers, and if they do quote these numbers will only use one URL rule to do so. Combined with the fact that your traffic and your Web content dictate your site performance, how can these adequately extrapolate the effect to HTTP (web) performance while denying (NIMDA and Code Red) application attacks using long URL list?

Spirent solutions are uniquely positioned to help you evaluate your SLB because:

- Vendor benchmarks derived from testing reflect the specific products capabilities, but do not provide insight into how the SLB will respond in your system when it is faced with your unique traffic-mix, load and end-to-end security requirements. Application level metrics are the most accurate way of knowing if a given device is throttling the infrastructure.
- Several well-known third-party labs, including Miercom, Tolly Group, VeriTest, Neophasis and NSS, use Spirent products to perform SLB testing. They are able to closely replicate your network conditions, but again, this level of testing may not adequately reflect your unique needs and the results may be different when the SLB is deployed in your system.
- By using the same testing solutions used by your SLB vendor and third-party testing labs, you can leverage the test scenarios and configurations to remove any variations added by testing frameworks. Using their test scenarios simplifies your testing strategies and can provide you with a greater degree of confidence in selecting the right firewall that suits your unique needs, thereby mitigating any new risks added by your testing solutions.

The following section illustrates how Spirent enables you to perform a comprehensive SLB evaluation by:

- Conducting a set of basic tests using the Spirent *Avalanche*, *Reflector* and *SmartBits* products with your SLB to ensure that it meets your expectations with a typical configuration.
- Conducting a set of advanced tests that ensure the end-to-end behavior of the SLB in conjunction with other devices in your IT infrastructure. These could range from security appliances such as IDS and SSL-accelerator devices, or content appliances such as caches that are individually tested by the *Avalanche* and *Reflector* products prior to being deployed in your network. Spirent recommends a methodical series of tests beginning with the individual SLB and progressing to the other infrastructure elements to reflect end-to-end requirements that closely simulate your real-life production network.
- Evaluating your SLB effectiveness with new business drivers, such as network consolidation, using technologies like Web compression, TCP offload and integrated authentication. Combine this with the need to connect your business to other business partners through the Web and there is no doubt you need to know what your business infrastructure can handle. Knowing this breaking point enables you to properly monitor your critical infrastructure.

Please note: The following tests assume you are familiar with the basic configuration, execution and result-interpretation of simple tests on *Avalanche* and *Reflector* appliances or the *SmartBits/Avalanche* platforms.

## Realism Testing of Your Server Load Balancer using Spirent

This section provides a basic SLB testing methodology using Spirent's *Avalanche* and *Reflector*, or *SmartBits TeraMetrics* and *TerametricsXD* platforms. Conducting these tests enables you to increasingly reflect your production network realism by adding more parameters to your SLB tests and includes additional devices in series with your SLB. This evaluates your SLB in a context similar to your production network, which reduces configuration and deployment risk.

The following set of tests can be run on the *Avalanche/Reflector* or *SmartBits* platforms:

- “SLB Basic” – Characterizes operating limits of the testing framework.
- “SLB Stress” – Measures the operating limits of your SLB with *connection*-based load balancing.
- “SLB Load” – Measures the operating limits of your SLB with *request*-based load balancing.

The test discussion assumes familiarity with:

- *Avalanche/Reflector* basics such as logging into the *Avalanche/Reflector* appliances (or *SmartBits/Avalanche* cards) and running the basic pre-packaged tests (such as “Echo” on *Reflector* and “SPI” on *Avalanche*).
- SLBs and networking basics (such as IP address allocation and switched networks).

### 1. SLB Basic – Establishing a Reference Baseline Test

#### Objective:

Before characterizing a SLB, the testing framework operating limits must be known. The basic testing framework consists of back-to-back *Avalanche* and *Reflector* devices connected via a switch. The switch is not mandatory; however, it is highly recommended as it simplifies network connectivity when adding a SLB to the testing framework.

Once you establish the testing equipment operating limits, inserting a SLB between the *Avalanche/Reflector* eliminates most headaches associated with the classic test problem: In case of test failure, is it the SLB or the testing framework that is causing the failure?

#### Requirements:

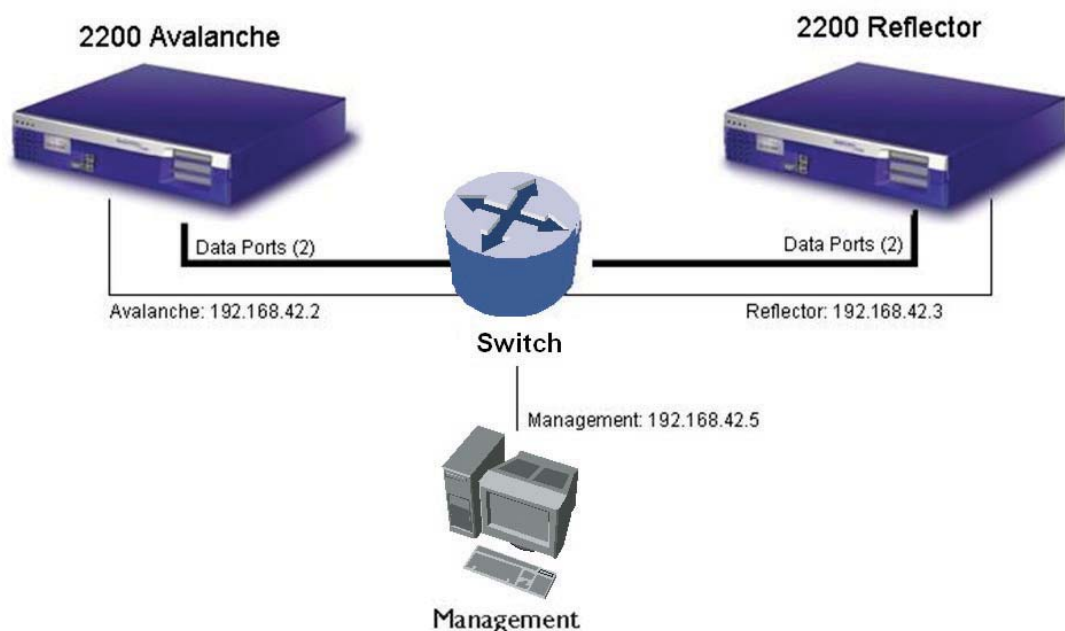
- *Avalanche* and *Reflector* (or *SmartBits/Terametrics (XD)* cards)
- Managed Layer 2-3 switch (with greater throughput than that expected from the SLB)
- Management console - any PC with Ethernet connectivity with a browser, JVM and Adobe Acrobat)

**Operational parameters to establish:**

- Connections per second (CPS) using HTTP 1.0 and HTTP 1.1
- Maximum number of concurrent connections (HTTP and HTTP 1.1)

**Running “SLB Basic”**

1. Connect your **Avalanche**, **Reflector**, management console and Layer 2-3 switch as shown in Figure 5.

**Figure 5:** SLB Basic test configuration

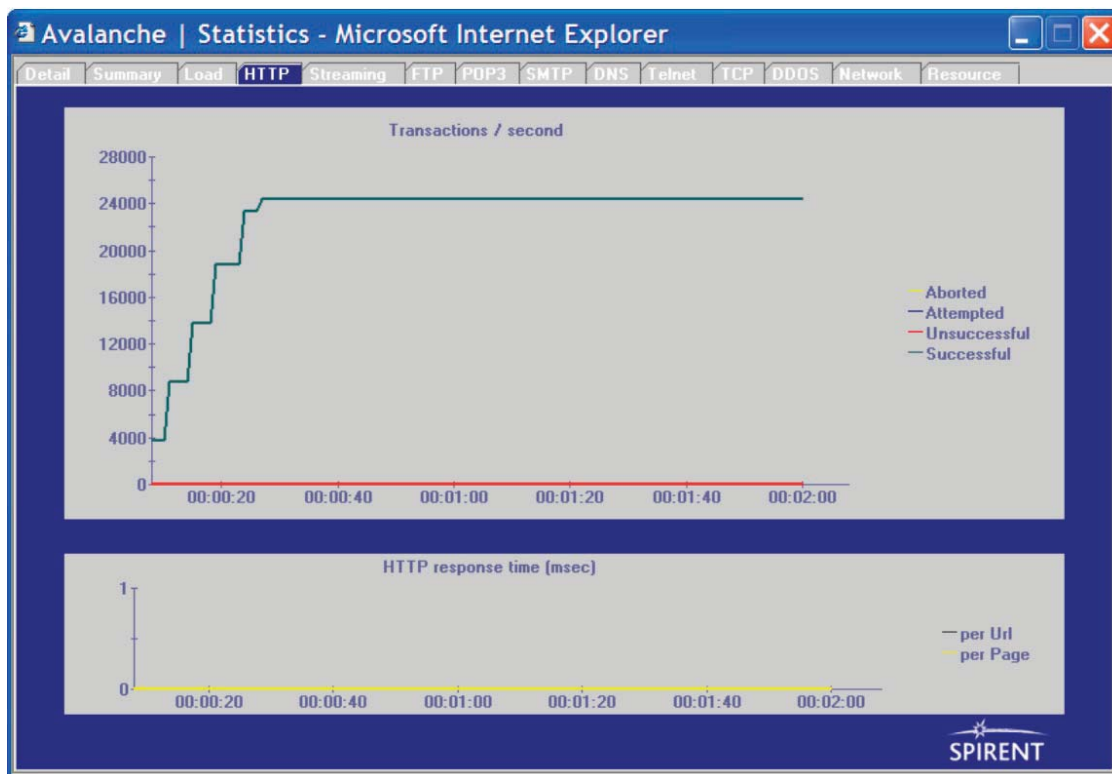
2. Ensure correct IP addressing between all devices:
  - **Avalanche** management address: 192.168.42.2 (default)
  - **Reflector** management address: 192.168.42.3 (default)
  - Management console address: 192.168.42.5 (given for this test)
3. Copy “Echo” on *Reflector* to a new test called “SLB Basic Reflector.”
  - a. Setup reflector to support HTTP 1.0 and/or HTTP 1.1.
  - b. Start SLB\_Basic\_Reflector and ensure that the test reaches steady state.
4. Configure “SLB Basic” test on *Avalanche*:
  - a. Select a pre-configured test (*SPI\_ConnsPerSec*) and copy it as “SLB Basic” test. You can now modify “SLB Basic” and save it as a new test.

- b. Setup client-side traffic to consist of two protocols: HTTP 1.0 and/or HTTP 1.1 without persistence or keep-alive set in user profiles.
- c. Configure the *Load Spec* to test the desired limits.

For example, if the SLB is benchmarked at 10,000 new connections/sec and 1,000,000 concurrent connections – and these are the parameters to be tested – run two separate sequential tests to ensure the testing framework can meet or exceed these benchmarks.

- i. Using “connections/sec” as the load-spec, run tests to determine that you can run 10% more than the SLB’s claimed CPS benchmark.
- ii. Using the SPI\_OpenConns load-spec, ensure that you can sustain over 1,000,000 connections during the “steady state” of the test run. This will require an increase in the ramp down time in the load profile to the 600 Second (10 Minutes) maximum value for the connection capacity test.

Figure 6 shows a sample output of the test results.



**Figure 6:** Sample SLB Basic equipment baseline test result

5. Check network port configurations and run “SLB Basic” on *Avalanche*.
  - a. Debug any failures that exist that might be due to improper network configurations or Ethernet connectivity between the switch/devices. Failures could include:
    - Failed requests – especially if the test is run over a long interval (48 hours) – that might uncover a memory leak
    - Timeouts for new incoming connections.
    - Servers going in and out of service during the test due to health check failures (use SNMP to help find this error, look for server event traps or server logging).
    - Resets of incoming connections while processing valid traffic.
6. Ensure that the *Avalanche/Reflector* and test-bed operating limits are higher than those of the SLB being tested.
  - a. Increase the load-spec by increments of 1,000 while maintaining “no failing tests.”
  - b. Note the operational limits of the test-bed for the specific “SLB Basic” test.

With no failing tests, you are ready to add an SLB to the test-bed for *stress* and *load* testing, building upon your “SLB Basic” test to more closely represent your production network.

## 2. SLB Stress – Stress-Testing Your SLB

### Objective:

Determine your SLB operating limits for parameters such as connections per second and open connections for a request-based configuration.

**Note:** Barring the insertion of an SLB into the test-bed, minimal changes are required in the test configuration. In contrast to “SLB Basic,” which was designed to test the operating limits of the *test equipment*, “SLB Stress” focuses on the SLB. If your SLB also bridges traffic, the only change is the VIP address in the *Avalanche* URL list. Also, a second baseline with the SLB just bridging traffic can help you understand the performance impact of no inspection on the SLB.

### Operational parameters to establish:

- Connections per second (CPS) using HTTP 1.0
- Maximum number of concurrent connections (HTTP)

### Running “SLB Stress”

1. Connect your SLB into the test-bed as shown in Figure 7.

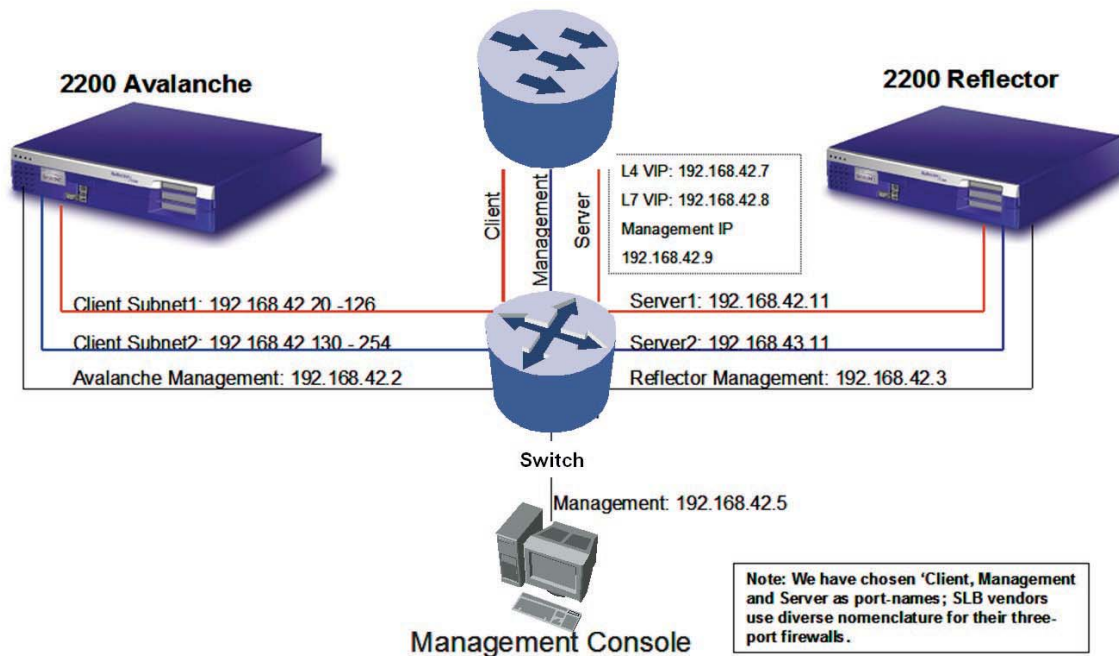


Figure 7: SLB Stress-test configuration

2. Configure appropriate network IP address:
  - a. Assign a typical three-port SLB comprising client facing ports and internal server facing ports on the *Avalanche* and *Reflector* as follows:
    - i. Reflector
      - Internal HTTP servers: 192.168.42.11, 192.168.43.11
    - i. Avalanche
      - Clients ports: 192.168.42.1-192.168.42.240 Mask 255.255.0.0

Two client-banks may be needed if more that two client ports are used on the *Avalanche*. *SLB IP addressing:*

- Management IP address: 192.168.42.9 Mask 255.255.0.0
- Request based Virtual IP address: 192.168.42.7
- Connection based Virtual IP address: 192.168.42.8
- Real server address: 192.168.42.11, 192.168.43.11
- SLB configuration (over serial port)

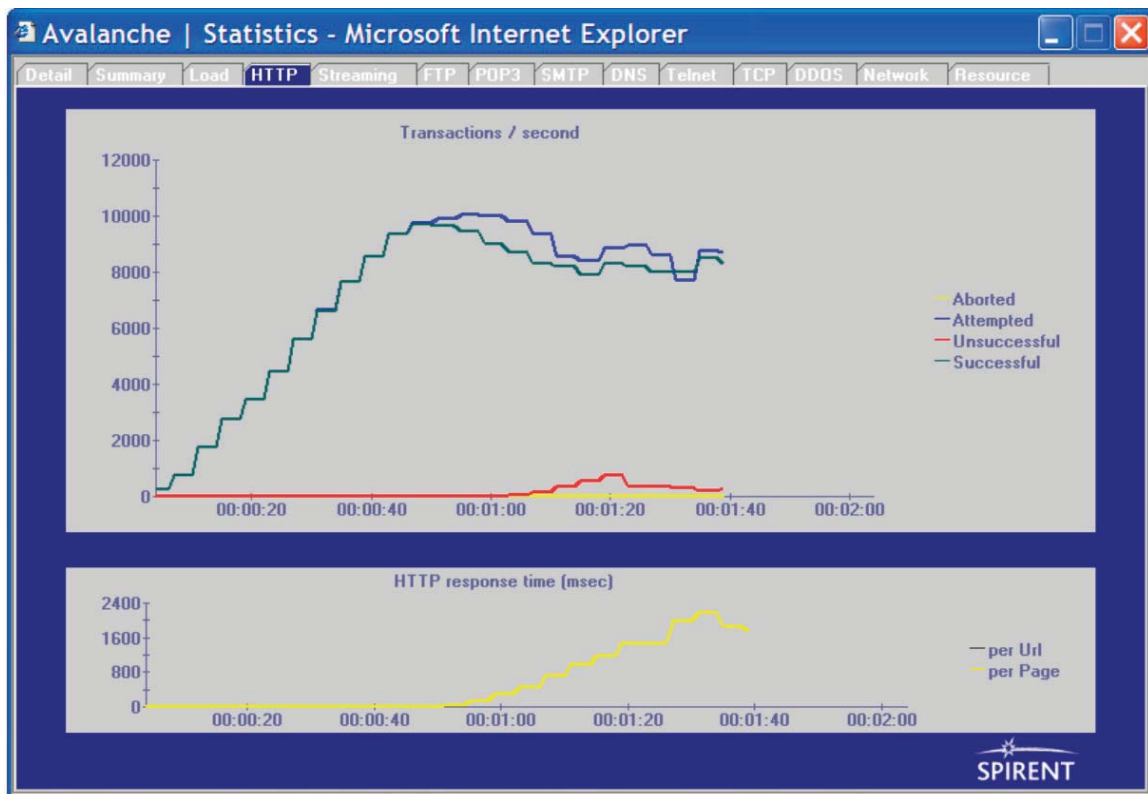
b. Configure the SLB:

Following is an *example* of the type of SLB configuration needed. It contains network IP addresses and load balancing rules, and is implemented with a simple “Connection based round robin load balancing of http” on VIP 192.168.42.7 and “Request based round robin load balancing of passive cookie sticky” rule set on VIP 192.168.42.8 (change the syntax as per your vendor’s requirements). Note that this is strictly an example to illustrate the testing methodology; each SLB vendor will have different configuration scripts and tools.

```
ip address 192.168.42.9 255.255.0.0  
virtual server ip address 192.168.42.7 cookie-passive  
real server ip address 192.168.42.11  
real server ip address 192.168.43.11  
bind VIP 192.168.42.7 to Real 192.168.42.11  
bind VIP 192.168.42.7 to Real 192.168.43.11
```

3. Run “Layer 4 load balancing test.”

Figure 8 shows a sample output of the test result.



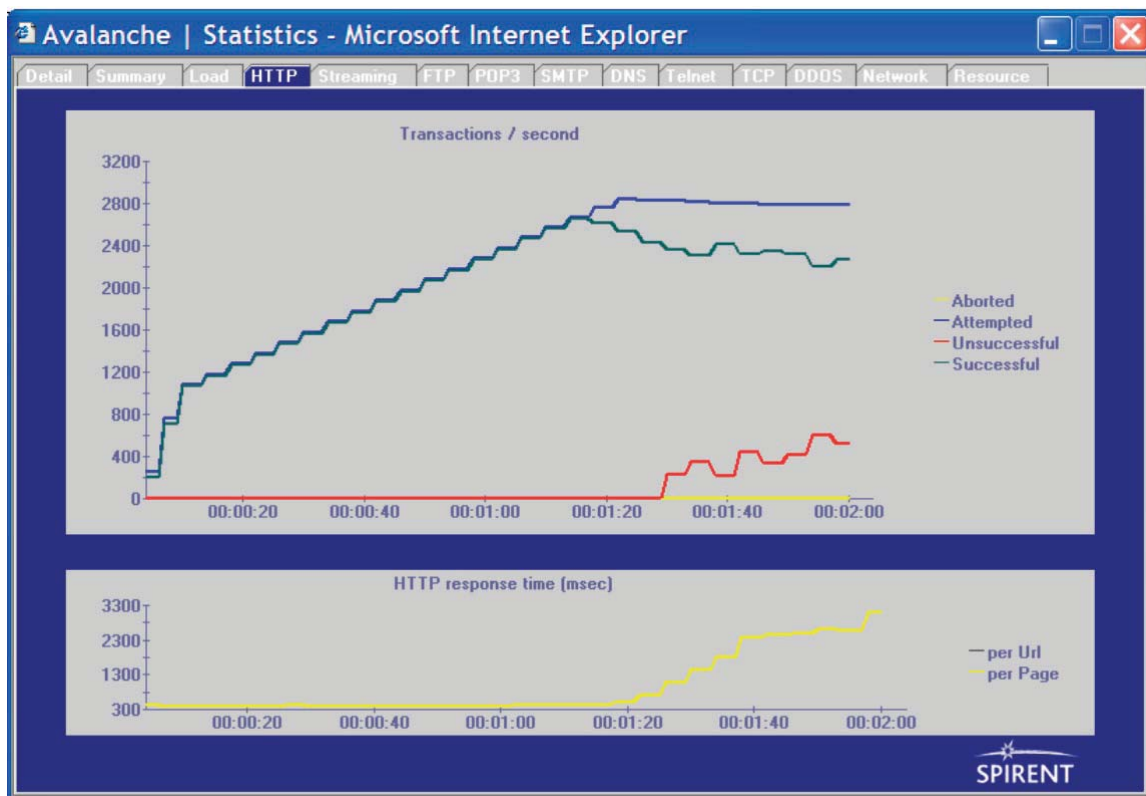
**Figure 8:** SLB Stress test Layer 4 Load Balancing performance without TCP buffering

Notice the split between successful requests and attempted requests. This is also the point where the response time climbs to over 2 seconds. The SLB adds undue latency to each request. For network management design, 30% less than this point is where the warning level should be set for the SNMP CPU MIB. The datasheet called for 25,000 CPS for this product.

4. Select and copy “SLB Basic load profile” to “SLB Stress CPS load profile.”

The only device under test (DUT) modification required for “Layer 7 Load balancing Without TCP retransmission” is to configure a passive cookie-based rule to the virtual server IP address. Optionally, for “Layer 7 Load Balancing with TCP retransmission” just configure FTP load balancing on your SLB.

5. Run “SLB Stress CPS.”
  - a. Determine SLB failure points by going through the test results and watching for a split in the TPS attempted counter (as shown in Figure 9) versus the successful transactions.



**Figure 9:** SLB Stress test Layer 4 Load Balancing performance without TCP buffering

The point where these items no longer track together is the point where the SLB is saturated and cannot dependably process any more requests. If you are also polling SMTP information from your SLB, you can set this as a critical event so that you understand the breaking point of your SLB at X% CPU utilization.

6. The only modification required for request-based testing is to change the load-balancing rule in the SLB to a “passive cookie” or “cookie insert.”
7. In Figure 9, notice the “split” where the successful transactions and the attempted transactions deviate. This is the breaking point in this particular test. Comparing this to the difference between the connection-based load balancing performance and the request-based performance gives the entire net effect to the request-based configuration change.
8. If this split does not occur because the SLB cannot keep up and failures are noticed from the start of the test:
  - a. Watch the *Reflector* statistics for successful transactions per second (HTTP) counter to find out how many successful transactions are passing through the SLB.
  - b. Reduce the load profile to meet what the SLB can handle.
  - c. Modify the load-spec to calibrate CPS.
  - d. Observe the number of requests per second getting through to the *Reflector* side.

Previously we observed 2400 CPS getting through to the **Reflector** side. This led to an adjustment in the load profile to a total test number of 3000 CPS.

9. “Layer 7 Load Balancing With TCP retransmission” test:

By adding a rule on the SLB to load balance FTP traffic, “Layer 7 Load Balancing with TCP retransmission” can be tested. The reason the SLB operates like a full proxy server is due to the implementation of FTP. Since FTP uses a different TCP connection for the control and the data connections, SLBs will enable a FTP proxy server to observe the load balancing on a given pair of connections. Because of this, the SLB will slow down by another factor of 2 – 4 times.

10. Throughput test.

A larger object can be used in the *Reflector* response by modifying the default profile on the *Reflector*. Typically, Internet Web sites use an average object size of 3kbytes. By changing the object size to 3k, the associated bandwidth the SLB can provide in this configuration is obtained. By using the average object size on your Web site, you can see if the SLB is actually providing full bandwidth as it is provided to your site.

11. Measure drop in response times due to request-based loading.

As the SLB becomes stressed with increasing load, *processing and response latencies through the SLB increase*. Since the SLB distributes load to the web servers, increased latencies equate to slower user-experience that in some cases grows to unacceptable access times for your Web sites. This too can be measured with and without request-based load balancing. Additionally, 3-5% packet loss and multiple user network access speeds (modem-DSL) can be measured to see the real-world impact of this performance degradation. Keep in mind that changes made here should be baselined without the SLB in place to provide an understanding of the performance impact.

12. In addition to CPS testing, concurrent open active connections is a key SLB benchmark parameter. To create this test:

- a. Take the “SLB Stress” test and copy it to “SLB Open.”
- b. Change the load profile to reflect *SimUsers* as the load rather than connections per second. Use the default *SimUsers* load profile as the starting number of concurrent users.
- c. Increase the number of *SimUsers* gradually until you see failing tests.

In the sample run, a moderate increase in *SimUsers* (over 45) resulted in a large number of transactions timing-out. Incomplete transactions reduce availability of the servers and can degrade performance to unacceptable levels.

As discussed earlier, you are encouraged to go beyond CPS and open connections to test relevant criteria such as throughput, using your Web object sizes, and enabling FTP load balancing on your SLB for an understanding of request-based load balancing with TCP retransmissions.

## Summary

Conducting the tests discussed in this paper enable you to increasingly reflect your production network realism by adding more parameters to your SLB tests while including additional devices in series with the SLB. Identifying and understanding the SLB’s breaking point in your configuration allows you to better program the SNMP operational breaking point and allows you to evaluate your SLB in a context similar to your production network, reducing both configuration and deployment risk.

Simply extrapolating vendor benchmarks or typical lab testing is inadequate and is a recipe for disaster. This application note demonstrates how to go beyond the typical method of extrapolating measurements and conduct a series of tests that measure server load balancer performance while under load—providing a more accurate measurement of true SLB throughput. The three most important metrics that ensure your SLB provides both scalability and availability are:

- Validating performance as applicable to your network
- Data throughput (proven Mbps/Gbps)
- Concurrent TCP connection capacity

Following a SLB testing methodology consisting of the *basic, stress and load tests* as discussed in this paper enables you to identify and mitigate risks associated with your SLB evaluations. By combining the knowledge of your network and application engineers with the capabilities of the Spirent *Avalanche, Reflector, and SmartBits* products, you can evaluate your SLB with the *realism of production networks with the precision of lab testing*. Doing so ensures you have the right server load balancing product to support your corporate network load balancing needs and helps increase both the security and availability of your network infrastructure.

Spirent encourages you to extend the basic, load and stress test-oriented methodology discussed in this paper to include additional feature testing for vendor-specific SLB features such as cache load balancing, firewall load balancing, SSL load balancing and aggregated SLB platforms that include SSL offload, Web authentication and other security functions. Contact Spirent ServiceEdge to help you extend the methodology in this paper to help you include your specific “realism testing” parameters to realize the most benefit out of your SLB and end-to-end testing methodologies.

## Background

There are numerous excellent sources for background on server load balancing and testing methodologies at the following websites:

<http://www.loadbalancing.net/white.html>

<http://computer.howstuffworks.com/question342.htm>

Benchmarking Methodology for Firewall Performance”, IETF RFC 3511, <http://www.ietf.org/rfc/rfc3511.txt?number=3511>

“Firewall Performance Testing Methodology”, Joung, P and Pochiraju, A, Spirent Communications, <http://www.spirentcom.com/documents/1062.pdf>“

AN-1 Firewall Testing Methodology using Spirent Solutions”, Hemendra Godbole, Spirent Communications “SalesNet”

---

### (Footnotes)

<sup>1</sup> <http://www.computerworld.com/databasetopics/data/software/story/0,10801,74459,00.html>