WebOS Switch Software

8.0 Application Guide

Part Number: 050087, Revision A, May 2000

Alteon WebSystems
Web Speed for e-Business

50 Great Oaks Boulevard
San Jose, California 95119
408-360-5500 Main
408-360-5501 Fax
www.alteonwebsystems.com

Contents

Figures

Tables

Preface

This *Application Guide* describes how to configure and use the WebOS 8.0 software included in the Alteon WebSystems family of switches.

For documentation on installing the switches physically, see the hardware installation guide for your particular switch model.

Who Should Use This Book

This *Application Guide* is intended for network installers and system administrators engaged in configuring and maintaining a network. It assumes that you are familiar with Ethernet concepts, IP addressing, the IEEE 802.1d Spanning-Tree Protocol, and SNMP configuration parameters.

What You'll Find in This Book

The chapters in this book will help you plan, implement, and administer the use of WebOS 8.0 software features. Where possible, each chapter provides a conceptual overview of a specific WebOS feature or functional area, usage examples, and, configuration instructions for implementing the feature(s) on an Alteon WebSystems switch.

Typographic Conventions

The following table describes the typographic styles used in this book.

Table 1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	This type is used for names of commands, files, and directories used within the text.	View the `readme.txt` file.
	It also depicts on-screen computer output and prompts.	`Main#`
`AaBbCc123`	This bold type appears in command examples. It shows text that must be typed in exactly as shown.	`Main#` **`sys`**
AaBbCc123	This italicized type appears in command examples as a parameter placeholder. Replace the indicated text with the appropriate real name or value when using the command.	To establish a Telnet session, enter: `host#` **`telnet`** *IP-address*
	This also shows book titles, special terms, or words to be emphasized.	Read your *User's Guide* thoroughly.
[ ]	Command items shown inside brackets are optional and can be used or excluded as the situation demands. Do not type the brackets.	`host#` **`ls`** [**`-a`**]

Contacting Alteon Networks

Use the following information to access Alteon WebSystems support and sales.

- URL for Alteon WebSystems Online:

  `http://www.alteonwebsystems.com`

  This website includes product information, software updates, release notes, and white papers. The website also includes access to Alteon WebSystems Customer Support for accounts under warranty or that are covered by a maintenance contract.

- E-mail access:

  `support@alteonwebsystems.com`

  E-mail access to Alteon WebSystems Customer Support is available to accounts that are under warranty or covered by a maintenance contract.

- Telephone access to Alteon WebSystems Customer Support:

  1-888-Alteon0 (or 1-888-258-3660)
  1-408-360-5695

  Telephone access to Alteon WebSystems Customer Support is available to accounts that are under warranty or covered by a maintenance contract. Normal business hours are 8 a.m. to 6 p.m. Pacific Standard Time.

- Telephone access to Alteon WebSystems Sales:

  1-888-Alteon2 (or 1-888-258-3662), and press 2 for Sales
  1-408-360-5600, and press 2 for Sales

  Telephone access is available for information regarding product sales and upgrades.

CHAPTER 1

Server Load Balancing

This chapter describes how to configure and use the optional Layer 4 software for Server Load Balancing (SLB). For information on activating this optional software, see your *WebOS 8.0 Command Reference*.

Overview

Benefits

SLB benefits your network in a number of ways:

- Increased efficiency for server utilization and network bandwidth

  With SLB, your Alteon WebOS switch is aware of the shared services provided by your server pool. The switch can then balance user session traffic among the available servers. For even greater control, traffic is distributed according to a variety of user-selectable rules.

  By helping to eliminate server over-utilization, important session traffic gets through more easily, reducing user competition for connections on overworked servers.

- Increased reliability of services to users

  If any server in a server pool fails, the remaining servers continue to provide access to vital applications and data. The failed server can be brought back up without interrupting access to services.

- Increased scalability of services

  As users are added and the server pool's capabilities are saturated, new servers can be added to the pool transparently.

Identifying Your Network Needs

SLB may be the right option for addressing these vital network concerns:

- A single server no longer meets the demand for its particular application.
- The connection from your LAN to your server overloads the server's capacity.
- Your NT and UNIX servers hold critical application data and must remain available even in the event of a server failure.
- Your website is being used as a way to do business and for taking orders from customers. It must not become overloaded or unavailable.
- You want to use multiple servers or hot-standby servers for maximum server uptime.
- You must be able to scale your applications to meet client and LAN request capacity.
- You can't afford to continue using an inferior load balancing technique such as DNS round robin, or a software-only system.

How SLB Works

In an average network that employs multiple servers without server load balancing, each server usually specializes in providing one or two unique services. If one of these servers provides access to applications or data which is in high demand, it can become over-utilized. Placing this kind of strain on a server can decrease the performance of the entire network as user requests are rejected by the server and then resubmitted by the user stations. Ironically, over-utilization of key servers often happens in networks where other servers are actually under-utilized.

The solution to getting the most from your servers is SLB, an optional feature on Alteon WebSystems Web switches. With this software feature, the switch is aware of the services provided by each server, and can direct user session traffic to an appropriate server based on a variety of load balancing algorithms.

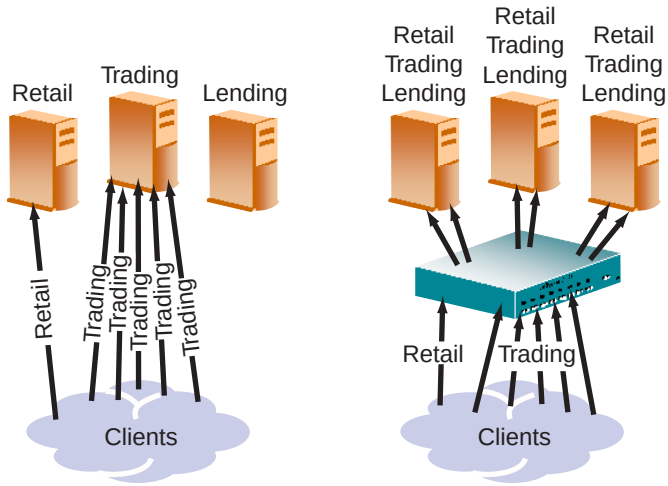


Figure 1-1 Traditional vs. SLB network configurations

To provide load balancing for any particular type of service, each server in the pool must have access to identical content, either directly (duplicated on each server) or through a back-end network (mounting the same file system or database server).

The Web switch, with SLB software, acts as a front-end to the servers, interpreting user session requests and distributing them among the available servers. To accomplish this, the switch is configured to act as a virtual server and is given a virtual IP address (or range of addresses) for each collection of services it will distribute. Depending on your switch model, there can be as many as 256 virtual servers on the switch, each distributing up to eight different services (up to a total of 2048 services).

Each virtual server is assigned a list of the real IP addresses (or range of addresses) of the real servers in the pool where its services reside. When the user stations request connections to a service, they will communicate with a virtual server on the switch. When the switch receives the request, it binds the session to the real IP address of the best available real server, and remaps the fields in each frame from virtual addresses to real addresses.

Network Topology Considerations

When deploying SLB, there are a few key aspects to consider:

- In standard SLB, all client requests to a virtual IP address and all responses from the real servers must pass through the switch. If alternate paths exist between the client and the real servers (as shown in the figure below), the Web switch can be configured to proxy requests in order to guarantee that responses use the correct path (see "Proxy IP Addresses for Complex SLB Networks" on page 36).

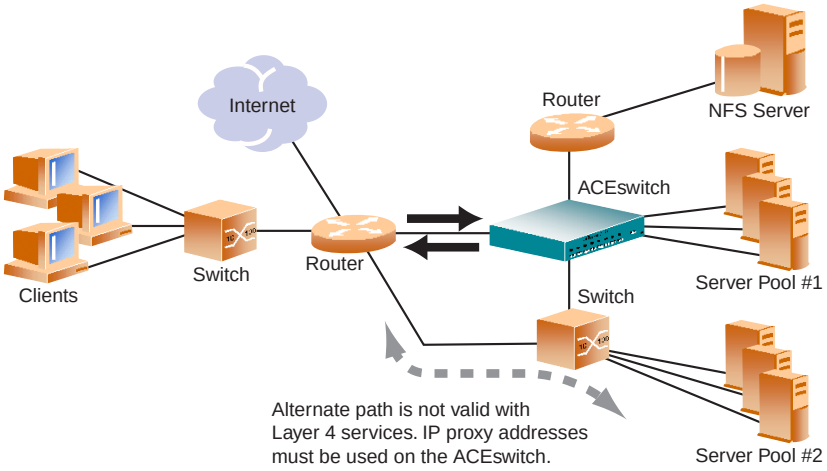


Figure 1-2 SLB Client/Server traffic routing

- Identical content must be available to each server in the same pool. Either of these methods can be used:
  - Static applications and data are duplicated on each real server in the pool.
  - Each real server in the pool has access to the same data through use of a shared file system or back-end database server.
- Some services require that a series of client requests goes to the same real server so that session-specific state data can be retained between connections. Services of this nature include Web search results, multi-page forms that the user fills in, or custom Web-based applications typically created using `cgi-bin` scripts. Connections for these types of services must be configured as "persistent" (see Chapter 7, "Persistence"), or must use the `minmisses` or `hash` metrics (see "Metrics for Real Server Groups" on page 31).

- Clients and servers can be connected through the same switch port. Each port in use on the switch can be configured to process client requests, server traffic, or both. You can enable or disable processing on a port independently for each type of Layer 4 traffic.
  - Layer 4 client processing. Ports configured to process client request traffic provide address translation from the virtual IP to the real server IP address.
  - Layer 4 server processing. Ports configured to process server responses to client requests provide address translation from the real server IP address to the virtual IP address. These ports require real servers to be connected to the Web switch, directly or through a hub, router, or another switch.

NOTE – Switch ports configured for Layer 4 client/server processing can simultaneously provide Layer 2 switching and IP Routing functions.

Consider the following network topology:

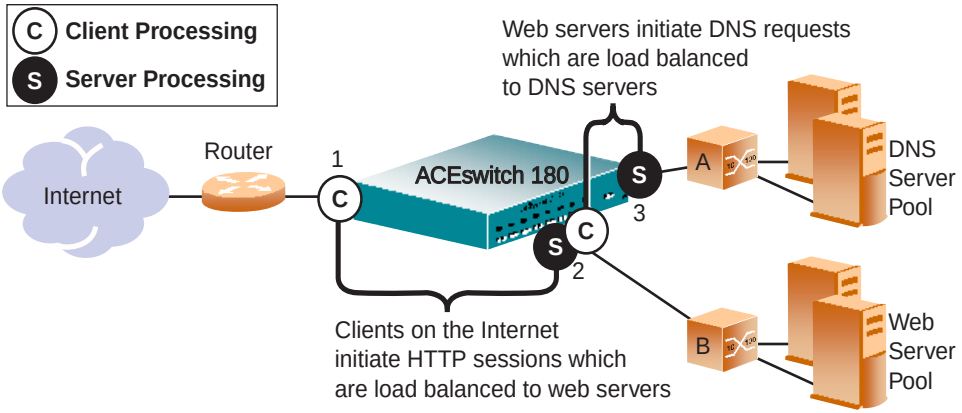


Figure 1-3 Example Network for Client/Server Port Configuration

In this figure, the switch load balances traffic to a Web server pool and to a DNS server pool. The switch port connected to the Web server pool is asked to perform both server and client processing.

Some topologies require special configuration. For example, if clients were added to switch B in the example above, these clients could not access the Web server pool using SLB services except through a proxy IP address configured on port 2 of the Alteon WebSystems Web switch.

SLB Implementation

Web Hosting Configuration

Consider a situation where customer Web sites are being hosted by a popular Web hosting company and/or Internet Service Provider (ISP). The Web content is relatively static and is kept on a single NFS server for easy administration. As the customer base increases, so does the number of simultaneous Web connection requests.

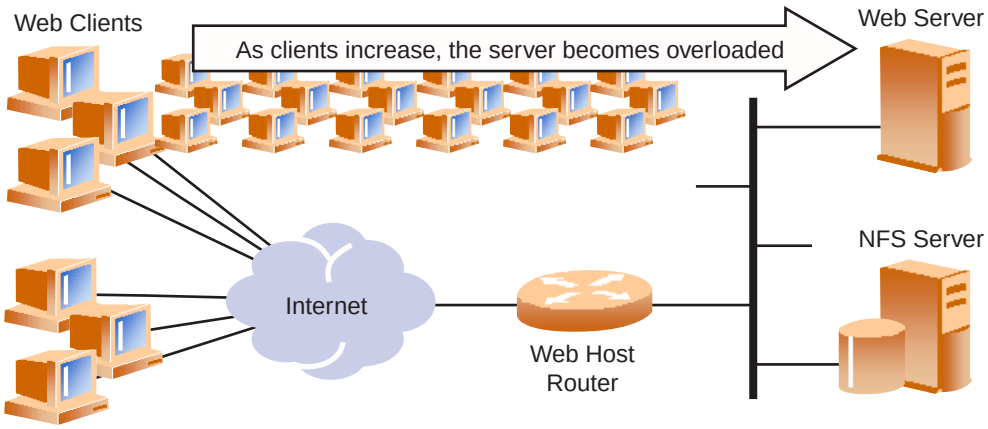


Figure 1-4 Web Hosting Configuration without VLSB

Such a company has three primary needs:

- Increased server availability
- Server performance scalable to match new customer demands
- Easy administration of network and servers

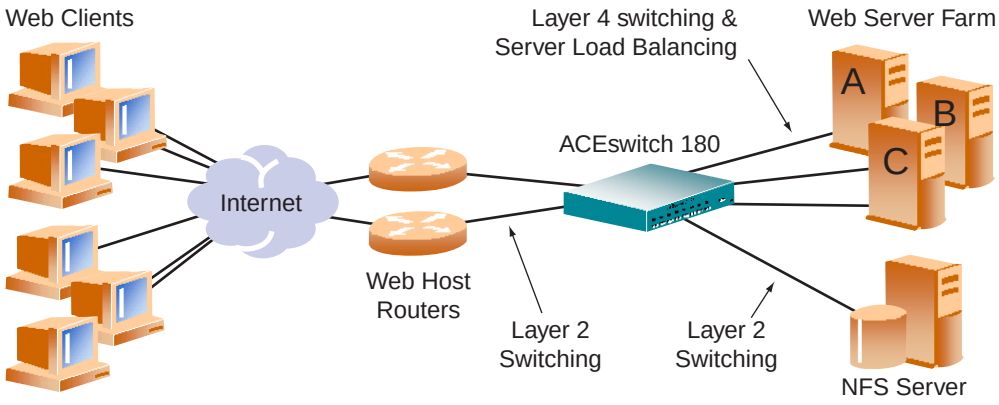


Figure 1-5 Web Hosting with SLB Solutions

Each concern about this company's site can be addressed by adding an Alteon WebSystems Web switch with optional SLB software.

- Reliability is increased by providing multiple paths from the clients to the Web switch, and by access to a pool of servers that have identical content. If one server fails, the others can take up the additional load.
- Performance is improved by balancing the Web request load across multiple servers. More servers can be added at any time to increase processing power.
- For ease of maintenance, servers can be added or removed dynamically without interrupting shared services.

Example Configuration for the Web Hosting Solution

In the following examples, many of the SLB options are left to their default values. See "Additional SLB Options" on page 30 for more options.

The following is required prior to configuration:

- You must be connected to the switch command-line interface as the administrator (see your *WebOS 8.0 Command Reference*).
- The optional SLB software must be enabled (see your *WebOS 8.0 Command Reference*).

NOTE – For details about any of the menu commands described in this example, see your *WebOS 8.0 Command Reference*.

---

1. **Assign an IP address to each of the real servers in the server pool.**

The real servers in any given real server group must have an IP route to the switch that will perform the SLB functions. This is most easily accomplished by placing the switches and servers on the same IP subnet, although advanced routing techniques can be used as long as they do not violate the topology rules outlined in "Network Topology Considerations" on page 22.

For this example, the three Web-host real servers have been given the following IP addresses on the same IP subnet:

Table 1-1 Web Host Example: Real Server IP addresses

Real Server	IP address
Server A	200.200.200.2
Server B	200.200.200.3
Server C	200.200.200.4

NOTE – An imask option can be used to define a range of IP addresses for real and virtual servers (see "IP Address Ranges Using imask" on page 30).

---

2. **Define an IP interface on the switch.**

   The switch must have an IP route to all of the real servers which receive Web switching services. For SLB, the switch uses this path to determine the level of TCP/IP reachability of the real servers.

   To configure an IP interface for this example, enter these commands from the CLI:

```
>> # /cfg/ip/if 1                          (Select IP interface #1)
>> IP Interface 1# addr 200.200.200.100   (Assign IP address for the interface)
>> IP Interface 1# ena                    (Enable IP interface #1)
```

3. **On the switch, define each real server.**

   For each real server, you must assign a real server number, specify its actual IP address, and enable the real server. For example:

```
>> IP Interface 1# /cfg/slb/real 1        (Server A is real server 1)
>> Real server 1 # rip 200.200.200.2      (Assign Server A IP address)
>> Real server 1 # ena                    (Enable real server 1)
>> Real server 1 # ../real 2              (Server B is real server 2)
>> Real server 2 # rip 200.200.200.3      (Assign Server B IP address)
>> Real server 2 # ena                    (Enable real server 2)
>> Real server 2 # ../real 3              (Server C is real server 3)
>> Real server 3 # rip 200.200.200.4      (Assign Server C IP address)
>> Real server 3 # ena                    (Enable real server 3)
```

4. **On the switch, define a real server group.**

   This combines the three real servers into one service group:

```
>> Real server 3 # /cfg/slb/group 1       (Select real server group 1)
>> Real server group 1# add 1             (Add real server 1 to group 1)
>> Real server group 1# add 2             (Add real server 2 to group 1)
>> Real server group 1# add 3             (Add real server 3 to group 1)
```

5. On the switch, define a virtual server.

All client requests will be addressed to a virtual IP address on a virtual server defined on the switch. Clients acquire the virtual IP address through normal DNS resolution. In this example, HTTP is configured as the only service running on this virtual server, and this service is associated with our real server group. For example:

```
>> Real server group 1 # /cfg/slb/virt 1          (Select virtual server 1)
>> Virtual server 1# vip 200.200.200.1           (Assign a virtual server IP address)
>> Virtual server 1# service http                (Select the HTTP service menu)
>> Virtual server 1 http Service# group 1        (Associate virtual port to real group)
>> Virtual server 1 http Service# ../ena         (Enable the virtual server)
```

NOTE – This configuration is not limited to HTTP Web service. Other TCP/IP services can be configured in a similar fashion. For a list of other well-known services and ports, see the command option information in your *WebOS 8.0 Command Reference*.

6. On the switch, define the port settings.

In this example, the following ports are being used on the Web switch:

Table 1-2 Web Host Example: Port Usage

Port	Host	L4 Processing
1	Server A, which serves SLB requests.	Server
2	Server B, which serves SLB requests.	Server
3	Server C, which serves SLB requests.	Server
4	Back-end NFS server. All three real servers get their Web content from this machine. This port does not require Web switching features.	None
5	Client router A. This connects the switch to the Internet where client requests originate.	Client
6	Client router B. This also connects the switch to the Internet where client requests originate.	Client

The ports are configured as follows:

```
>> Virtual server 1# /cfg/slb/port 1          (Select physical switch port 1)
>> SLB port 1# server ena                     (Enable server processing on port 1)
>> SLB port 1# ../port 2                      (Select physical switch port 2)
>> SLB port 2# server ene                     (Enable server processing on port 2)
>> SLB port 2# ../port 3                      (Select physical switch port 3)
>> SLB port 3# server ena                     (Enable server processing on port 3)
>> SLB port 3# ../port 5                      (Select physical switch port 5)
>> SLB port 5# client ena                     (Enable client processing on port 5)
>> SLB port 5# ../port 6                      (Select physical switch port 6)
>> SLB port 6# client ena                     (Enable client processing on port 6)
```

7. **On the switch, enable, apply, and verify the configuration.**

```
>> SLB port 6# ..                             (Select the SLB Menu)
>> Server Load Balancing# on                  (Turn Server Load Balancing on)
>> Server Load Balancing# apply               (Make your changes active)
>> Server Load Balancing# cur                 (View current settings)
```

Examine the resulting information. If any settings are incorrect, make any appropriate changes.

8. **On the switch, save your new configuration changes.**

```
>> Server Load Balancing# save                (Save for restore after reboot)
```

9. **On the switch, check the Server Load Balancing information.**

```
>> Server Load Balancing# /info/slb/dump      (View SLB information)
```

Check that all Server Load Balancing parameters are working according to expectation. If necessary, make any appropriate configuration changes and then check the information again.

Additional SLB Options

In the examples above, many of the SLB options are left to their default values. The following configuration options can be used to tune the system.

NOTE – You must `apply` any changes in order for them to take effect, and you must `save` them if you wish them remain in effect after switch reboot.

IP Address Ranges Using imask

The `imask` option lets you define a range of IP addresses for the real and virtual servers configured under SLB. By default, the `imask` setting is 255.255.255.255, which means that each real and virtual server represents a single IP address. An imask setting of 255.255.255.0 would mean that each real and virtual server represents 256 IP addresses. Consider the following example:

- A virtual server is configured with an IP address of 172.16.10.1.
- Real servers 172.16.20.1 and 172.16.30.1 are assigned to service the virtual server.
- The imask is set to 255.255.255.0.

If the client request was sent to virtual IP address 172.16.10.45, the unmasked portion of the virtual IP address (0.0.0.45) gets mapped directly to whichever real IP address is selected by the SLB algorithm. Thus, the request would be sent to either 172.16.20.45 or 172.16.30.45.

Health Checks for Real Servers

Determining health for each real server is a necessary function for SLB. By default for TCP services, the switch checks health by opening a TCP connection to each service port configured as part of each virtual service. For UDP services, the switch pings servers to determine their status.

By default, the switch checks the status of each service on each real server every two seconds. Sometimes, the real server may be too busy processing connections to respond to health checks. By default, if a service does not respond to four consecutive health checks, the switch declares the service unavailable. Both the health check interval and the number of retries can be changed:

```
>> # /cfg/slb/real <real server number>     (Select the real server)
>> Real server# inter 4                     (Check real server every 4 seconds)
>> Real server# retry 6                     (If 6 consecutive health checks fail,
                                             declare real server down)
```

More complex health-checking strategies may also be used. See Chapter 4, “Health Checking” for more details.

Metrics for Real Server Groups

Metrics are used for selecting which real server in a group will receive the next client connection. The available metrics are `minmisses` (minimum misses), `hash`, `leastconns` (least connections), and `roundrobin`, explained in detail below.

The default metric is `leastconns`. To change a real server group metric, to `minmisses` for example, enter:

```
>> # /cfg/slb/group <group number>          (Select the real server group)
>> Real server group# metric minmisses      (Use minmisses metric)
```

Minimum Misses

The `minmisses` (minimum misses) metric is optimized for Application Redirection. It uses IP address information in the client request to select a server. The specific IP address information used depends on the application:

- For Application Redirection, the client destination IP address is used. All requests for a specific IP destination address will be sent to the same server. This is particularly useful in caching applications, helping to maximize successful cache hits. Best statistical load balancing is achieved when the IP address destinations of load balanced frames are spread across a broad range of IP subnets.
- For SLB, the client source IP address and real server address are used. All requests from a specific client will be sent to the same server. This is useful for applications where client information must be retained on the server between sessions. Server load with this metric becomes most evenly balanced as the number of active clients with different source or destination addresses increases.

When selecting a server, the switch will calculate a score for each available real server based on the relevant IP address information. The server that scores the highest is assigned the connection. This metric attempts to minimize the disruption of persistency when servers are removed from service. This metric should be used only when persistence is a must.

NOTE – This metric cannot be used for Firewall Load Balancing (FWLB) since the real server IP addresses used in calculating the score for this metric are different on each side of the firewall.

Hash

The `hash` metric uses IP address information in the client request to select a server. The specific IP address information used depends on the application:

- For Application Redirection, the client destination IP address is used. All requests for a specific IP destination address will be sent to the same server. This is particularly useful for maximizing successful cache hits.
- For SLB, the client source IP address is used. All requests from a specific client will be sent to the same server. This is useful for applications where client information must be retained between sessions.
- For FWLB, both the source and destination IP address are used. This helps ensure that the two unidirectional flows of a given session to be redirected to the same firewall.

When selecting a server, a mathematical “hash” of the relevant IP address information is used as an index into the list of currently available servers. Any given IP address information will always have the same hash result, providing natural persistence as long as the server list is stable. However, if a server is added to or leaves the mix, then a different server might be assigned to a subsequent session with the same IP address information even though the original server is still available. Open connections are not cleared.

This metric provides more even load balancing than `minmisses` at any given instant. It should be used if the statistical load balancing achieved using `minmisses` is not as optimal as desired. If the load balancing statistics with `minmisses` indicate that one server is processing significantly more requests over time than other servers, consider using the `hash` metric.

Least Connections

With the `leastconns` metric, the number of connections currently open on each real server is measured in real time. The server with the fewest current connections is considered to be the best choice for the next client connection request.

This option is the most self-regulating, with the fastest servers typically getting the most connections over time, due to their ability to accept, process, and shut down connections faster than slower servers.

Round Robin

With the `roundrobin` metric, new connections are issued to each server in turn: the first real server in this group gets the first connection, the second real server gets the next connection, followed by the third real server, and so on. When all the real servers in this group have received at least one connection, the issuing process starts over with the first real server.

Weights for Real Servers

Weights can be assigned to each real server. These weights bias load balancing to give the fastest real servers a bigger share of connections during load balancing. Weight is specified as a number from 1 to 48. Each increment increases the number of connections the real server gets. By default, each real server is given a weight setting of 1. A setting of 10 would assign the server roughly 10 times the number of connections as a server with a weight of 1. To set weights, enter the following commands:

```
>> # /cfg/slb/real <real server number>      (Select the real server)
>> Real server# weight 10                    (10 times the number of connections)
```

NOTE – Weights are not applied when using the `hash` or `minmisses` metrics.

Connection Time-outs for Real Servers

In some cases, open TCP/IP sessions might not be closed properly (for example, the switch receives the `SYN` for the session, but no `FIN` is sent). If a session is inactive for 10 minutes (the default), it is released from the switch. To change the time-out period, enter the following:

```
>> # /cfg/slb/real <real server number>      (Select the real server)
>> Real server# tmout 4                      (Specify an even numbered interval)
```

The example above would change the time-out period of all connections on the designated real server to 4 minutes.

Maximum Connections for Real Servers

You can set the number of open connections each real server is allowed to handle for Server Load Balancing. To set the connection limit, enter the following:

```
>> # /cfg/slb/real <real server number>      (Select the real server)
>> Real server# maxcon 1600                  (Allow 1600 connections maximum)
```

Values average between about 500 HTTP connections for slower servers, up to 1,500 for quicker, multi-processor servers. The appropriate value also depends on the duration of each session, as well as how much CPU capacity is occupied by processing each session. Connections that use a lot of Java or `CGI` scripts for forms or searches require more server resources and thus a lower `maxcon` limit. You may wish to use a performance bench-mark tool to determine how many connections your real servers can handle.

When a server reaches its `maxcon` limit, the switch no longer sends new connections to the server. When the server drops back below the `maxcon` limit, new sessions are again allowed.

Backup/Overflow Servers

A real server can backup other real servers, and can handle overflow traffic when the maximum connection limit is reached. Each backup real server must be assigned a real server number and real IP address. It must then be enabled. Finally, the backup must be assigned to each real server it will backup. The following defines real server #4 as a backup for real servers #1 and #2:

```
>> # /cfg/slb/real 4                      (Select real server #4 as backup)
>> Real server 4 # rip 200.200.200.5      (Assign backup IP address)
>> Real server 4 # ena                    (Enable real server #4)
>> Real server 4 # ../real 1              (Select real server #1)
>> Real server 1 # backup 4               (Real server #4 is backup for #1)
>> Real server 1 # ../real 2              (Select real server #2)
>> Real server 2 # backup 4               (Real server #4 is backup for #2)
```

In a similar fashion, a backup/overflow server can be assigned to a real server group. If all real servers in a real server group fail or overflow, the backup comes online.

```
>> # /cfg/slb/group <real server group number>   (Select real server group)
>> Real server group# backup r4                 (Assign real server #4 as backup)
```

Real server groups can also use another real server group for backup/overflow:

```
>> # /cfg/slb/group <real server group number>   (Select real server group)
>> Real server group# backup g2                 (Assign group #2 as backup)
```

Mapping Virtual Ports to Real Ports

In addition to providing direct real server access in some situations (see “Port Mapping” on page 40), mapping is required when administrators choose to execute their real server processes on different TCP/UDP ports than the well known TCP/UDP ports. Otherwise, virtual server ports are mapped directly to real server ports by default and require no mapping configuration.

Port mapping is configured from the virtual server services menu. For example, to map the virtual server TCP/UDP port 80 to real server TCP/UDP port 8004, you could enter the following:

```
>> # /cfg/slb/virt 1/service 80          (Select virtual server port 80)
>> Virtual Server 1 http Service# rport 8004   (map to real port 8004)
```

NOTE – Port mapping is supported with Direct Access Mode when filtering is enabled, a proxy IP address is configured, or URL parsing is enabled on any switch port. For information about DAM, refer to “Direct Access Mode” on page 39.

Direct Server Return

Using the Direct Server Return (DSR) feature, the server can respond directly to the client. This capability is useful for sites where large amounts of data are going from servers to clients, such as with content providers or portal sites that typically have asymmetric traffic patterns.

DSR and content-intelligent Layer 7 switching cannot be performed at the same time. This is because content intelligent switching requires that all frames must go back to the switch for connection splicing.

NOTE – DSR requires that the server must be set up to receive frames that have an IP destination address that is equal to the virtual server IP address(es).

The sequence of steps that are executed in this scenario are listed and pictured below:

1. **A client request is forwarded to the Web switch.**
2. **Because only MAC addresses are substituted, the switch forwards the request to the best server, based on the configured load balancing policy.**
3. **The server responds directly to the client, bypassing the switch, using the virtual IP address as the IP source address.**

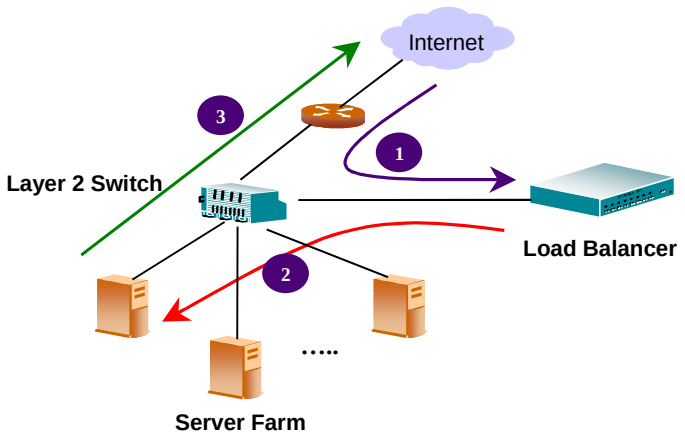


Figure 1-6 Direct Server Return

To set up direct server return, use the following commands:

```
>> # /cfg/slb/real <real server number>/submac ena
>> # /cfg/slb/virt <virtual server number>/service <service number>/nonat ena
```

Proxy IP Addresses for Complex SLB Networks

For standard SLB, all client-to-server requests to a particular virtual server and all related server-to-client responses must pass through the same Web switch.

In complex network topologies, routers and other devices can create alternate paths around the Web switch managing SLB functions (see Figure 1-2 on page 22). Under such conditions, the client switch ports can use a proxy IP address.

When the client requests services from the switch's virtual server, the client sends its own IP address for use as a return address. If a proxy IP address is configured for the client port on the switch, the switch replaces the client's source IP address with the switch's own proxy IP address before sending the request to the real server. This creates the illusion that the switch originated the request. The real server uses the switch's proxy IP address as the destination address for any response. This forces the SLB traffic to return through the proper switch, regardless of alternate paths. Once the switch receives the proxied data, it puts the original client IP address into the destination address and sends the packet to the client. This process is transparent to the client.

NOTE – Because requests appear to come from the switch proxy IP address rather than the client source IP address, use of proxy addresses can generate misleading information for network statistics or debugging.

The proxy IP address can also be used for direct access to the real servers (see "Direct Client Access to Real Servers" on page 39).

The following procedure can be used for configuring proxy IP addresses:

1. **Disable server processing on affected switch ports.**

   When implementing proxies, switch ports can be re-configured to disable server processing. Re-examining the "Example Configuration for the Web Hosting Solution" on page 26, the following revised port conditions are used:

   Table 1-3 Proxy Example: ACEswitch 180 Port Usage

Port	Host	L4 Processing
1	Server A	None
2	Server B	None
3	Server C	None
4	Back-end NFS server. All three real servers get their Web content from this machine. This port does not require Web switching.	None
5	Client router A. This connects the switch to the Internet where all client requests originate.	Client
6	Client router B. This also connects the switch to the Internet where all client requests originate.	Client

   The following commands are used to disable server processing on ports 1-3:

```
>> # /cfg/slb/port 1                    (Select switch port #1)
>> SLB port 1# server dis               (Disable server processing on port #1)
>> SLB port 1# ../port 2                (Select switch port #2)
>> SLB port 2# server dis               (Disable server processing on port #2)
>> SLB port 2# ../port 3                (Select switch port #3)
>> SLB port 3# server dis               (Disable server processing on port #3)
```

2. **Add proxy IP addresses to the client ports.**

   Each "`client`" ports requires a proxy IP address. Each proxy IP address must be unique on your network. The following shows commands used to configure client proxies for this example:

```
>> # /cfg/slb/port 5                    (Select network port #5)
>> SLB port 5# pip 200.200.200.68       (Set proxy IP address for client port #5)
>> SLB port 5# ../port 6                (Select network port #6)
>> SLB port 6# pip 200.200.200.69       (Set proxy IP address for client port #6)
```

   The proxies are transparent to the user.

3. **If the VMA feature is enabled, add proxy IP addresses for all other switch ports (except Port 9).**

   Virtual Matrix Architecture (VMA) is normally enabled on the switch. In addition to enhanced resource management, this feature eliminates many of the restrictions found in earlier versions of the WebOS. It does require, however, than when any switch port is configured with a proxy IP address, all ports must be configured with proxy IP addresses. Otherwise, if VMA is disabled, only the client port need proxy IP addresses and this step can be skipped.

   The following commands can be used for configuring the additional unique proxy IP addresses:

```
>> SLB port 6# ../port 1                  (Select network port #1)
>> SLB port 1# pip 200.200.200.70         (Set proxy IP address for port #1)
>> SLB port 1# ../port 2                  (Select network port #2)
>> SLB port 2# pip 200.200.200.71         (Set proxy IP address for port #2)
>> SLB port 2# ../port 3                  (Select network port #3)
>> SLB port 3# pip 200.200.200.72         (Set proxy IP address for port #3)
>> SLB port 3# ../port 4                  (Select network port #4)
>> SLB port 4# pip 200.200.200.73         (Set proxy IP address for port #4)
>> SLB port 4# ../port 7                  (Select network port #7)
>> SLB port 7# pip 200.200.200.74         (Set proxy IP address for port #7)
>> SLB port 7# ../port 8                  (Select network port #8)
>> SLB port 8# pip 200.200.200.75         (Set proxy IP address for port #8)
```

NOTE – Port 9 does not require a proxy IP address under VMA.

See the *WebOS 8.0 Command Reference* for more information (`/cfg/slb/adv/matrix`).

4. **Apply and save your changes.**

NOTE – Remember that you must `apply` any changes in order for them to take effect, and you must `save` them if you wish them remain in effect after switch reboot. Also, the `/info/slb` command is useful for checking the state of Server Load Balancing operations.

Direct Client Access to Real Servers

Some clients may need direct access to the real servers, to, for example, monitor a real server from a management workstation. This access can be provided in a number of ways:

- Direct Access Mode
- Multiple IP addresses on the server
- Proxy IP addresses
- Port mapping
- Management network

Direct Access Mode

When Direct Access Mode (`/cfg/slb/direct`) is enabled on a switch, any client can communicate with any real server's load-balanced service. Also, in Direct Access Mode, any number of virtual services can be configured to load balance a real service.

Traffic sent directly to real server IP addresses is excluded from load balancing decisions. The same clients may also communicate to the virtual server IP address for load balanced requests.

NOTE – When Direct Access Mode is enabled on a switch, port mapping and default gateway load balancing is supported only when filtering is enabled, a proxy IP address is configured, or URL parsing is enabled on any switch port.

Multiple IP Addresses on the Server

One way to provide both SLB access and direct access to a real server is to assign multiple IP addresses to the real server. For example, one IP address could be established exclusively for SLB, and another could be used for direct access needs.

Proxy IP Addresses

Proxy IP addresses are used primarily to eliminate SLB topology restrictions in complex networks (see "Network Topology Considerations" on page 22). Proxy IP addresses can also provide direct access to real servers.

If the switch port to the client is configured with a proxy IP address (see "Proxy IP Addresses for Complex SLB Networks" on page 36), the client can access each real server directly using the real server's IP address. This requires that the switch port connected to the real server has server and client processing disabled (see the `server` and `client` options under `/cfg/slb/port` in your *WebOS 8.0 Command Reference*).

SLB is still accessed using the virtual server IP address.

Port Mapping

When SLB is used without proxy IP addresses and without Direct Access Mode (DAM), the switch must process the server-to-client responses. If a client were to access the real server IP address and port directly, bypassing client processing, the server-to-client response could be mishandled by SLB processing as it returns through the switch, with the real server IP address getting re-mapped back to the virtual server IP address.

First, two port processes must be executed on the real server. One real server port will handle the direct traffic, and the other will handle SLB traffic. Then, the virtual server port must be mapped to the proper real server port.

In the following figure, clients can access SLB services through well-known TCP port 80 at the virtual server's IP address. This is mapped to TCP port 8000 on the real server. For direct access that bypasses the virtual server and SLB, clients can specify well-known TCP port 80 at the real server's IP address.

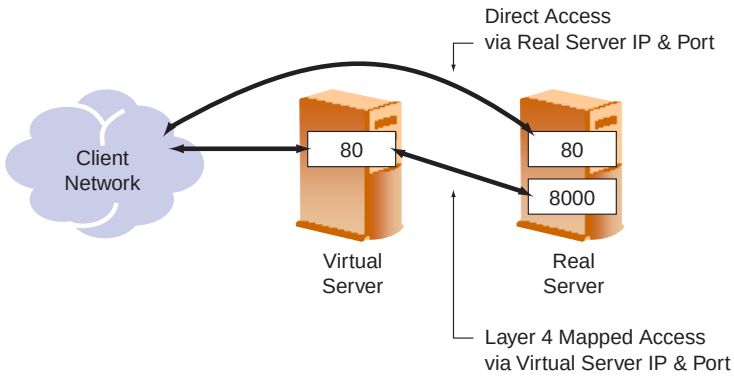


Figure 1-7 Mapped and Non-mapped server access

NOTE – Port mapping is supported with Direct Access Mode when filtering is enabled, a proxy IP address is configured, or URL parsing is enabled on any switch port.

Management Network

Typically, the management network is used by network administrators to monitor real servers and services. By configuring the `mnet` and `mmask` options of the SLB Configuration Menu (`cfg/slb`) you can access the real services being load balanced.

NOTE – Clients on the management network do not have access to SLB services and cannot access the virtual services being load balanced.

The `mnet` and `mmask` options are described below:

- `mnet`: If defined, management traffic with this source IP address will be allowed direct (non-SLB) access to the real servers. Only specify an IP address in dotted decimal notation. A range of IP addresses is produced when used with the `mmask` option
- `mmask`: This IP address mask is used with `mnet` to select management traffic which is allowed direct real server access only.

FTP Server Load Balancing

WebOS 8.0 supports load balancing of FTP servers on both public and private network for both active and passive FTP modes. To do this, the switch modifies FTP commands from both the client (for active FTP) and server (for passive FTP).

- For passive FTP SLB, the switch watches for the `PASV` command from the FTP client and modifies the “entering passive mode” command coming back from the FTP server, replacing the real IP address (RIP) with a virtual IP address (VIP), and the real server port (RPORT) with a virtual port (VPORT), so that the client will make an active open connection between the client data port and the switch, instead of the server. The switch then remaps requests to the FTP server that the control channel was bound to.
- For active FTP SLB, the switch watches for the `PORT` command from the FTP client and translates the FTP server active open connection to use *VIP*:*VPORT* instead of *RIP*:*RPORT*. The server data connection will look like it came from the switch itself.

Background

As defined in RFC 959, FTP (File Transfer Protocol) uses two channels/connections; one for control information and another for data. Each connection is unique. Unless the client requests a change, the server is always using TCP Port 21 (a well-known port) for control information and TCP Port 20 as the default data port.

FTP uses TCP for transport. After the initial three-way handshake, a connection is established. When the client requests any data information from the server, it will issue a PORT command (such as **`ls`**, **`dir`**, **`get`**, **`put`**, **`mget`** and **`mput`**) via the control port.

There are two modes of FTP operation, active and passive:

- In **Active FTP**, the FTP server initiates the data connection.
- In **Passive FTP**, the FTP client initiates the data connection. Because the client also initiates the connection to the control channel, the passive FTP mode does not pose a problem with firewalls and is the most common mode of operation.

Configuring FTP SLB

1. **Make sure there's proxy IP address (PIP) enabled on the client port(s) or Direct Access Mode (DAM) is enabled.**
2. **Make sure the virtual port for FTP is already set up for the virtual server.**
3. **Enable FTP parsing, using this command:**

```
>> # /cfg/slb/virt <virtual server number>/ftpp ena
```

4. **To make your configuration changes active, enter apply at any prompt in the CLI.**

```
>> # /cfg/slb/virt <virtual server number>/apply
```

NOTE – Remember that you must `apply` any changes in order for them to take effect, and you must `save` them if you wish them remain in effect after switch reboot.

CHAPTER 2

Filtering

This chapter provides a conceptual overview of filters and configuration examples showing how filters can be used to ensure network security and redirect traffic.

Filtering Overview

Benefits

Layer 3 (IP) and Layer 4 (application) filtering gives the network administrator a powerful tool with the following benefits:

- Filtering increases security for server networks.

  Filters can be configured to allow or deny traffic according to various IP address, protocol, and Layer 4 port criteria. This gives the administrator fine control over the types of traffic permitted through the switch. Optionally, any filter can generate `syslog` messages for increased security visibility.

- Generic Network Address Translation

  NAT can be used to map the source or destination IP address and port of private network traffic to/from an advertised network IP address and port.

- Application Redirection improves network bandwidth and provides unique network solutions.

  Filters can be created to redirect traffic to cache and application servers. Repeated client access to common web or application content across the Internet can be an inefficient use of network resources. By redirecting client requests to a local web-cache or application server, you increase the speed at which clients access the information and free up valuable network bandwidth.

Filtering Criteria

Up to 224 filters can be configured on the switch. Each filter can be set to allow, deny, redirect, or translate traffic based on any combination of the following criteria:

- Source IP Address or range
- Destination IP Address or range
- Protocol type (for example: IP, UDP, TCP, ICMP, and others)
- TCP flags
- Application, source port or range (For example: ftp, http, telnet, 31000-33000, etc.)
- Application, destination port or range (For example: ftp, http, telnet, 31000-33000, etc.)
- Inverse: activate the filter whenever the specified conditions are *not* met.

For example, you can create a single filter that blocks external Telnet traffic to your main server, except from a trusted IP address. Another filter could warn you if FTP access is attempted from a specific IP address. Another filter could redirect all incoming e-mail traffic to a server where it can be analyzed for spam. The options are nearly endless.

Below are a list of the well-known protocols and applications.

Table 2-1 Well-Known Protocol Types

Number	Protocol Name
1	icmp
2	igmp
6	tcp
17	udp
89	ospf
112	vrrp

Table 2-2 Well-Known Application Ports

Number	TCP/UDP Application	Number	TCP/UDP Application	Number	TCP/UDP Application
20	ftp-data	70	gopher	161	snmp
21	ftp	79	finger	162	snmptrap
22	ssh	80	http	179	bgp
23	telnet	109	pop2	194	irc
25	smtp	110	pop3	220	imap3
37	time	111	sunrpc	389	ldap
42	name	119	nntp	443	https
43	whois	123	ntp	520	rip
53	domain	143	imap	554	rtsp
69	tftp	144	news	1985	hsrp

Stacking Filters

Once configured, filters are assigned and enabled on a per port basis. Each filter can be used by itself or in combination with any other filter on any given switch port. The filters are numbered 1 through 224. When multiple filters are stacked together on a port, the filter's number determines its order of precedence: the filter with the lowest number is checked first. When traffic is encountered at the switch port, if the filter matches, its configured action takes place and the rest of the filters are ignored. If the filter criteria doesn't match, the next filter is tried.

As long as the filters do not overlap, you can improve filter performance by making sure that the most heavily utilized filters are applied first. For example, consider a filter system where the Internet is divided according to destination IP address:

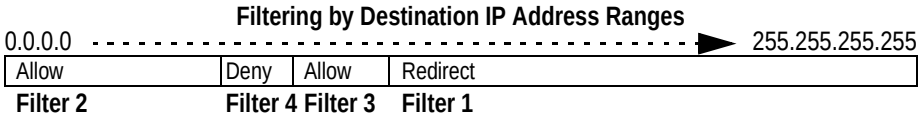


Figure 2-1 Assigning Filters according to Range of Coverage

Assuming that traffic is distributed evenly across the Internet, the largest area would be the most utilized and is assigned to filter 1. The smallest area is assigned to filter 4.

Overlapping Filters

Filters are permitted to overlap, although special care should be taken to ensure the proper order of precedence. When overlapping filters are present, the more specific filters (those that target fewer addresses or ports) should be applied before the generalized filters.

Example:

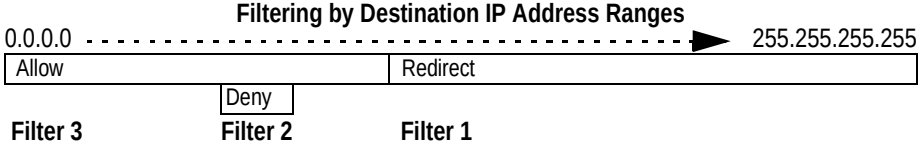


Figure 2-2 Assigning Filters to Overlapping Ranges

In this example, the "deny" filter must be processed prior to the "allow" filter. If the "allow" filter was allowed to take precedence, the "deny" filter could never be triggered.

The Default Filter

Before filtering can be enabled on any given port, a default filter should be configured. This filter handles any traffic not covered by any other filter. All the criteria in the default filter must be set to the full range possible ("`any`"). For example:

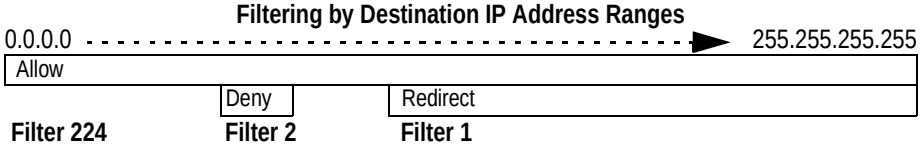


Figure 2-3 Assigning a Default Filter

In this example, filter 224 is the default filter. If no other filter acts on the traffic, filter 224 handles it. All matching criteria in filter 224 is set to the "**any**" state.

Although recommended when configuring filters for IP traffic control and redirection, default filters are not required. Using default filters can increase session performance, but takes some of the session binding resources. If you experience an unacceptable number of binding failures as shown in the Server Load Balancing Maintenance Statistics (`/stats/slb/maint`), you may wish to remove some of the default filters.

Numbering Filters

You may wish to consider numbering your filters by increments of 5 or 10 (for example: 5, 10, 15, 20, etc.). This allows for filters to be easily inserted between others in the list, if required.

Filter Logs

To provide enhanced troubleshooting and session inspection capability, packet source and destination IP addresses are included in filter log messages. Filter log messages are generated when a Layer 3/Layer 4 filter is triggered and has logging enabled. The messages are output to the console port, system host log (`syslog`), and the web-based interface message window.

Example: A network administrator has noticed a significant number of ICMP frames on one portion of the network, and wants to determine the specific sources of the ICMP messages. The administrator uses the command-line interface to create and apply the following filter:

```
>> # /cfg/slb/filt 15                    (Select filter 15)
>> Filter 15# sip any                     (From any source IP address)
>> Filter 15# dip any                     (To any destination IP address)
>> Filter 15# action allow                (Allows matching traffic to pass)
>> Filter 15# proto icmp                  (For the ICMP protocol)
>> Filter 15# log enabled                 (Create a log entry when matched)
>> Filter 15# ena                         (Enable the filter)
>> Filter 15# /cfg/slb/port 7             (Select a switch port to filter)
>> SLB port 7# add 15                     (Add the filter to the switch port)
>> SLB port 7# filt ena                   (Enable filtering on the switch port)
>> SLB port 7# apply                      (Apply the configuration changes)
>> SLB port 7# save                       (Save the configuration changes)
```

When applied to one or more switch ports, this simple filter rule will produce log messages that show when the filter is triggered, and what the IP source and destination addresses were for the ICMP frames traversing those ports.

Example: Filter log message output is shown below, displaying the filter number, port, source IP address, and destination IP address:

```
slb: filter 15 fired on port 7, 206.118.93.110 -> 20.10.1.10
```

Security Example

Consider the following sample network:

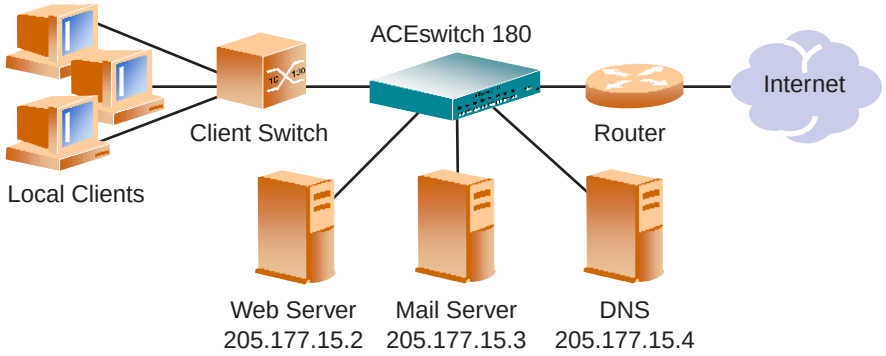


Figure 2-4 Example Security Topology

In this example, the network is made of local clients on a collector switch, a web server, a mail server, a domain name server, and a connection to the Internet. All the local devices are on the same subnet.

For best security, it is commonly considered that you should configure filters to deny all traffic except for those services you specifically wish to allow. In this example, the administrator wishes to install basic security filters to allow only the following traffic:

- External HTTP access to the local web server
- External SMTP (mail) access to the local mail server
- Local clients browsing the World Wide Web
- Local clients using Telnet to access sites outside the intranet
- Domain Name System

All other traffic will be denied and logged.

NOTE – Since IP address and port information can be manipulated by external sources, filtering does not replace the necessity for a well-constructed network firewall.

Example Configuration for the Security Solution

Prior to configuration, you must be connected to the switch command-line interface as the administrator.

In this example, *all filters will be applied only to the switch port which connects to the Internet.* If intranet restrictions were required, filters could be placed on switch ports connecting to local devices.

Also, filtering is not limited to the few protocols and TCP or UDP applications shown in this example. See the *WebOS 8.0 Command Reference* for a list of other well-known protocols and services.

1. **Assign an IP address to each of the network devices.**

   For this example, the network devices have the following IP addresses on the same IP subnet:

   Table 2-3 Web-Cache Example: Real Server IP addresses

Network Device	IP address
Local Subnet	205.177.15.0 - 205.177.15.255
Web Server	205.177.15.2
Mail Server	205.177.15.3
Domain Name Server	205.177.15.4

2. **On the switch, create a default filter that will deny and log unwanted traffic.**

   The default filter is defined as filter 224 in order to give it the lowest order of precedence:

```
>> # /cfg/slb/filt 224                (Select the default filter)
>> Filter 224# sip any                 (From any source IP addresses)
>> Filter 224# dip any                 (To any destination IP addresses)
>> Filter 224# proto any               (For any protocols)
>> Filter 224# action deny             (Deny matching traffic)
>> Filter 224# log enable              (Log matching traffic to syslog)
>> Filter 224# ena                     (Enable the default filter)
```

NOTE – When the `proto` parameter is *not* `tcp` or `udp`, then `sport` and `dport` are ignored.

3. **On the switch, create a filter that will allow external HTTP requests to reach the web server.**

   The filter must recognize and allow TCP traffic with the web-server's destination IP address and HTTP destination port:

```
>> Filter 224# ../filt 1                  (Select the menu for Filter #1)
>> Filter 1# sip any                      (From any source IP address)
>> Filter 1# dip 205.177.15.2             (To web-server dest. IP address)
>> Filter 1# dmask 255.255.255.255        (Fill mask for exact dest. address)
>> Filter 1# proto tcp                    (For TCP protocol traffic)
>> Filter 1# sport any                    (From any source port)
>> Filter 1# dport http                   (To an HTTP destination port)
>> Filter 1# action allow                 (Allow matching traffic to pass)
>> Filter 1# ena                          (Enable the filter)
```

4. **On the switch, create a pair of filters to allow incoming and outgoing mail to and from the mail server.**

   Filter 2 allows incoming mail to reach the mail server, and filter 3 allows outgoing mail to reach the Internet:

```
>> Filter 1# ../filt 2                    (Select the menu for Filter #2)
>> Filter 2# sip any                      (From any source IP address)
>> Filter 2# dip 205.177.15.3             (To mail-server dest. IP address)
>> Filter 2# dmask 255.255.255.255        (Fill mask for exact dest. address)
>> Filter 2# proto tcp                    (For TCP protocol traffic)
>> Filter 2# sport any                    (From any source port)
>> Filter 2# dport smtp                   (To a SMTP destination port)
>> Filter 2# action allow                 (Allow matching traffic to pass)
>> Filter 2# ena                          (Enable the filter)
>> Filter 2# ../filt 3                    (Select the menu for Filter #3)
>> Filter 3# sip 205.177.15.3             (From mail-server source IP address)
>> Filter 3# smask 255.255.255.255        (Fill mask for exact source address)
>> Filter 3# dip any                      (To any destination IP address)
>> Filter 3# proto tcp                    (For TCP protocol traffic)
>> Filter 3# sport smtp                   (From a smtp port)
>> Filter 3# dport any                    (To any destination port)
>> Filter 3# action allow                 (Allow matching traffic to pass)
>> Filter 3# ena                          (Enable the filter)
```

5. **On the switch, create a filter that will allow local clients to browse the web.**

   The filter must recognize and allow TCP traffic to reach the local client destination IP addresses if originating from any HTTP source port:

```
>> Filter 3# ../filt 4                    (Select the menu for Filter #4)
>> Filter 4# sip any                      (From any source IP address)
>> Filter 4# dip 205.177.15.0             (To base local network dest. address)
>> Filter 4# dmask 255.255.255.0          (For entire subnet range)
>> Filter 4# proto tcp                    (For TCP protocol traffic)
>> Filter 4# sport http                   (From any source HTTP port)
>> Filter 4# dport any                    (To any destination port)
>> Filter 4# action allow                 (Allow matching traffic to pass)
>> Filter 4# ena                          (Enable the filter)
```

6. **On the switch, create a filter that will allow local clients to Telnet anywhere outside the local intranet.**

   The filter must recognize and allow TCP traffic to reach the local client destination IP addresses if originating from a Telnet source port:

```
>> Filter 4# ../filt 5                    (Select the menu for Filter #5)
>> Filter 5# sip any                      (From any source IP address)
>> Filter 5# dip 205.177.15.0             (To base local network dest. address)
>> Filter 5# dmask 255.255.255.0          (For entire subnet range)
>> Filter 5# proto tcp                    (For TCP protocol traffic)
>> Filter 5# sport telnet                 (From a Telnet port)
>> Filter 5# dport any                    (To any destination port)
>> Filter 5# action allow                 (Allow matching traffic to pass)
>> Filter 5# ena                          (Enable the filter)
```

7. **On the switch, create a series of filters to allow Domain Name System (DNS) traffic.**

   DNS traffic requires four filters. One pair is needed for UDP traffic: incoming and outgoing. Another pair is needed for TCP traffic: incoming and outgoing.

For UDP:

```
>> Filter 5# ../filt 6                    (Select the menu for Filter #6)
>> Filter 6# sip any                      (From any source IP address)
>> Filter 6# dip 205.177.15.4             (To local DNS Server)
>> Filter 6# dmask 255.255.255.255        (Fill mask for exact dest. address)
>> Filter 6# proto udp                    (For UDP protocol traffic)
>> Filter 6# sport any                    (From any source port)
>> Filter 6# dport domain                 (To any DNS destination port)
>> Filter 6# action allow                 (Allow matching traffic to pass)
>> Filter 6# ena                          (Enable the filter)
>> Filter 6# ../filt 7                    (Select the menu for Filter #7)
>> Filter 7# sip 205.177.15.4             (From local DNS Server)
>> Filter 7# smask 255.255.255.255        (Fill mask for exact source address)
>> Filter 7# dip any                      (To any destination IP address)
>> Filter 7# proto udp                    (For UDP protocol traffic)
>> Filter 7# sport domain                 (From a DNS source port)
>> Filter 7# dport any                    (To any destination port)
>> Filter 7# action allow                 (Allow matching traffic to pass)
>> Filter 7# ena                          (Enable the filter)
```

Similarly, for TCP:

```
>> Filter 7# ../filt 8                    (Select the menu for Filter #8)
>> Filter 8# sip any                      (From any source IP address)
>> Filter 8# dip 205.177.15.4             (To local DNS Server)
>> Filter 8# dmask 255.255.255.255        (Fill mask for exact dest. address)
>> Filter 8# proto tcp                    (For TCP protocol traffic)
>> Filter 8# sport any                    (From any source port)
>> Filter 8# dport domain                 (To any DNS destination port)
>> Filter 8# action allow                 (Allow matching traffic to pass)
>> Filter 8# ena                          (Enable the filter)
>> Filter 8# ../filt 9                    (Select the menu for Filter #9)
>> Filter 9# sip 205.177.15.4             (From local DNS Server)
>> Filter 9# smask 255.255.255.255        (Fill mask for exact source address)
>> Filter 9# dip any                      (To any destination IP address)
>> Filter 9# proto tcp                    (For TCP protocol traffic)
>> Filter 9# sport domain                 (From a DNS source port)
>> Filter 9# dport any                    (To any destination port)
>> Filter 9# action allow                 (Allow matching traffic to pass)
>> Filter 9# ena                          (Enable the filter)
```

8. **On the switch, assign the filters to the switch port that connects to the Internet:**

```
>> Filter 9# ../port 5                  (Select the SLB port 5 to the Internet)
>> SLB Port 5 # add 1                   (Add filter 1 to port 5)
>> SLB Port 5 # add 2                   (Add filter 2 to port 5)
>> SLB Port 5 # add 3                   (Add filter 3 to port 5)
>> SLB Port 5 # add 4                   (Add filter 4 to port 5)
>> SLB Port 5 # add 5                   (Add filter 5 to port 5)
>> SLB Port 5 # add 6                   (Add filter 6 to port 5)
>> SLB Port 5 # add 7                   (Add filter 7 to port 5)
>> SLB Port 5 # add 8                   (Add filter 8 to port 5)
>> SLB Port 5 # add 9                   (Add filter 9 to port 5)
>> SLB Port 5 # add 224                 (Add the default filter to port 5)
>> SLB Port 5 # filt enable             (Enable filtering for port 5)
```

9. **On the switch, apply and verify the configuration.**

```
>> SLB Port 5 # ..                      (Select Server Load Balancing Menu)
>> Server Load Balancing# apply         (Make your changes active)
>> Server Load Balancing# cur           (View current settings)
```

Examine the resulting information. If any settings are incorrect, make appropriate changes.

10. **On the switch, save your new configuration changes.**

```
>> Server Load Balancing# save          (Save for restore after reboot)
```

11. **On the switch, check the Server Load Balancing information.**

```
>> Server Load Balancing# /info/slb/dump   (View SLB information)
```

Check that all Server Load Balancing parameters are working according to expectation. If necessary, make any appropriate configuration changes and then check the information again.

NOTE – Changes to filters on a given port do not take effect until the port's session information is updated (every two minutes or so). To make filter changes take effect immediately, clear the session binding table for the port (see the `/oper/slb/clear` command in the *WebOS 8.0 Command Reference*.

TCP ACK Matching for Filters

The ack filter criteria provides greater filtering flexibility. When ack is enabled, the filter matches only those frames set with the TCP ACK or RST flag.

The ack criteria appears in the WebOS Web interface, and in the command line interface on the Filter Menu (/cfg/slb/filt <*filter-number*>/adv).

Example: Consider the following network:

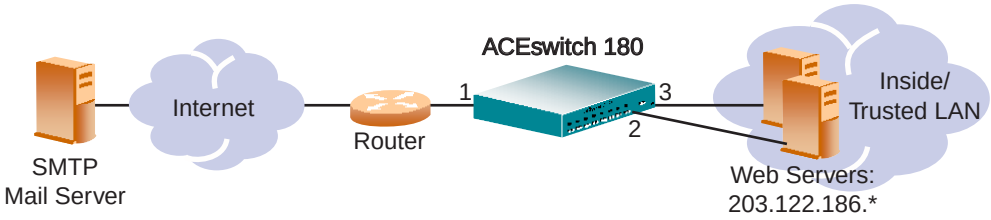


Figure 2-5 Example Filter TCP ACK Matching Network

In this network, the web servers inside the LAN must be able to transfer mail to any SMTP-based mail server out on the Internet. At the same time, we wish to prevent access to the LAN from the Internet, except for HTTP.

SMTP traffic uses well-known TCP port 25. The web servers will originate TCP sessions to the SMTP server using destination TCP port 25, and the SMTP server will acknowledge each TCP session and data transfer using source TCP port 25.

Filtering with the ACK flag closes one potential security hole. Without it, the switch would permit a TCP SYN connection request to reach any listening destination TCP port on the web servers inside the LAN, as long as it originated from TCP source port 25. The server would listen to the TCP SYN, allocate buffer space for the connection, and reply to the connect request. In some SYN attack scenarios, this could cause the server's buffer space to fill, crashing the server or at least making it unavailable.

This filter with the ACK flag requirement prevents external servers from beginning a TCP connection (with a TCP SYN) from source TCP port 25. The server will drop any frames that have the ACK flag turned off in them.

The following filters are required:

1. **One filter must allow the web servers to pass SMTP requests to the Internet.**

```
>> # /cfg/slb/filt 10                  (Select a filter for trusted SMTP requests)
>> Filter 10# sip 203.122.186.0        (From the web servers' source IP address)
>> Filter 10# smask 255.255.255.0      (For the entire subnet range)
>> Filter 10# sport any                (From any source port)
>> Filter 10# proto tcp                (For TCP traffic)
>> Filter 10# dip any                  (To any destination IP address)
>> Filter 10# dport smtp               (To well-known destination SMTP port)
>> Filter 10# action allow             (Allow matching traffic to pass)
>> Filter 10# ena                      (Enable the filter)
```

2. **One filter must allow SMTP traffic from the Internet to pass through the switch *only* if the destination is one of the web servers and the frame is an acknowledgment (ACK) of a TCP session.**

```
>> Filter 10# ../filt 15               (Select a filter for Internet SMTP ACKs)
>> Filter 15# sip any                  (From any source IP address)
>> Filter 15# sport smtp               (From well-known source SMTP port)
>> Filter 15# proto tcp                (For TCP traffic)
>> Filter 15# ack ena                  (For acknowledgments only)
>> Filter 15# dip 203.122.186.0        (To the web servers' IP address)
>> Filter 15# dmask 255.255.255.0      (To the entire subnet range)
>> Filter 15# dport any                (To any destination port)
>> Filter 15# action allow             (Allow matching traffic to pass)
>> Filter 15# ena                      (Enable the filter)
```

3. **One filter must allow trusted HTTP traffic from the Internet to pass through the switch to the web servers.**

```
>> Filter 15# ../filt 16               (Select a filter for incoming HTTP traffic)
>> Filter 16# sip any                  (From any source IP address)
>> Filter 16# sport http               (From well-known source HTTP port)
>> Filter 16# proto tcp                (For TCP traffic)
>> Filter 16# dip 203.122.186.0        (To the web servers' IP address)
>> Filter 16# dmask 255.255.255.0      (To the entire subnet range)
>> Filter 15# dport http               (To well-known destination HTTP port)
>> Filter 16# action allow             (Allow matching traffic to pass)
>> Filter 16# ena                      (Enable the filter)
```

4. **One filter must allow HTTP responses from the web servers to pass through the switch to the Internet.**

```
>> Filter 16# ../filt 17                 (Select a filter for outgoing HTTP traffic)
>> Filter 17# sip 203.122.186.0          (From the web servers' source IP address)
>> Filter 17# smask 255.255.255.0        (From the entire subnet range)
>> Filter 17# sport http                 (From well-known source HTTP port)
>> Filter 17# proto tcp                  (For TCP traffic)
>> Filter 17# dip any                    (To any destination IP address)
>> Filter 17# dport http                 (To well-known destination HTTP port)
>> Filter 17# action allow               (Allow matching traffic to pass)
>> Filter 17# ena                        (Enable the filter)
```

5. **One default filter is required to deny everything else:**

```
>> Filter 17# ../filt 224                (Select a default filter)
>> Filter 220# sip any                   (From any source IP address)
>> Filter 220# dip any                   (To any destination IP address)
>> Filter 220# action deny               (Block matching traffic)
>> Filter 220# ena                       (Enable the filter)
```

6. **Next, the filters must be applied to the appropriate switch ports.**

```
>> Filter 220# ../port 1                 (Select the Internet-side port)
>> SLB port 1# add 15                    (Add the SMTP ACK filter to the port)
>> SLB port 1# add 16                    (Add the incoming HTTPS filter)
>> SLB port 1# add 224                   (Add the default filter to the port)
>> SLB port 1# filt ena                  (Enable filtering on the port)
>> SLB port 1# ../port 2                 (Select the first web server port)
>> SLB port 2# add 10                    (Add the outgoing SMTP filter to the port)
>> SLB port 2# add 17                    (Add the outgoing HTTP filter to the port)
>> SLB port 2# add 224                   (Add the default filter to the port)
>> SLB port 2# filt ena                  (Enable filtering on the port)
>> SLB port 2# ../port 3                 (Select the other web server port)
>> SLB port 3# add 10                    (Add the outgoing SMTP filter to the port)
>> SLB port 3# add 17                    (Add the outgoing HTTP filter to the port)
>> SLB port 3# add 224                   (Add the default filter to the port)
>> SLB port 3# filt ena                  (Enable filtering on the port)
>> SLB port 3# apply                     (Apply the configuration changes)
>> SLB port 3# save                      (Save the configuration changes)
```

Network Address Translation Examples

In the following NAT examples, a company has configured its internal network with “private” IP addresses. A private network is one that is isolated from the global Internet, and is therefor free from the usual restrictions requiring the use of registered, globally unique IP addresses. Private networks can use whatever IP addresses they please, including those that are in use elsewhere on the Internet, or reserved for other purposes.

Private networks serve two main purposes. First, because private IP addresses are not valid or visible outside the private network, they can increase network security. Second, since valid, registered IP addresses are a limited resource, many companies use private IP addresses to create internal networks much larger than they could using only their official addresses.

With Network Address Translation (NAT), private networks are not required to remain isolated. NAT capabilities within the switch allow internal, private network IP addresses to be translated to valid, publicly advertised IP addresses and back again.

Internal Client Access to Internet

In this dynamic NAT example, clients on the internal private network require TCP/UDP access to the Internet:

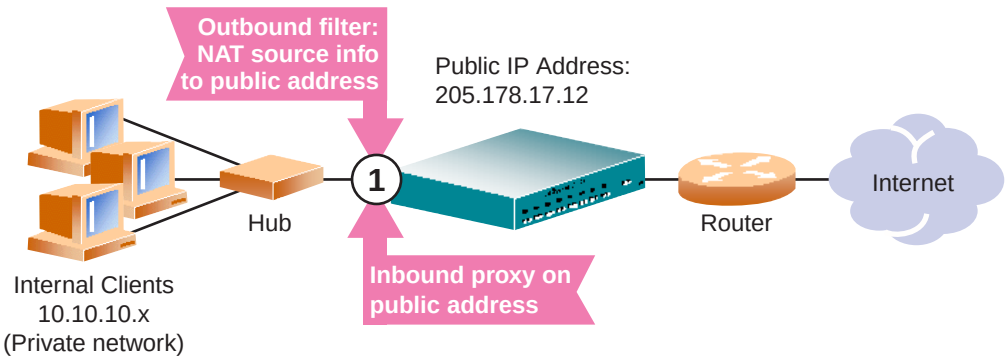


Figure 2-6 Dynamic NAT

This example requires a Network Address Translation (NAT) filter to be configured on the switch port connected to the internal clients. When the NAT filter is triggered by outbound client traffic, the internal private IP address information on the outbound packets is translated to a valid, publicly advertised IP address. In addition, the public IP address must be configured as a proxy IP address on the switch port connected to the internal clients. The proxy performs the reverse translation, restoring the private network addresses on inbound packets.

This is a “many to one” solution: multiple clients on the private subnet take advantage of a single external IP address, thus conserving valid IP addresses.

This example could be configured as follows:

NOTE – The `invert` option is only specific to this example and is not required.

```
>> # /cfg/slb/filt 14                    (Select the menu for client filter)
>> Filter 14 invert ena                  (Invert the filter logic)
>> Filter 14 dip 10.10.10.0              (If the destination is not private)
>> Filter 14 dmask 255.255.255.0         (For the entire private subnet range)
>> Filter 14 sip any                     (From any source IP address)
>> Filter 14 action nat                  (Perform NAT on matching traffic)
>> Filter 14 nat source                  (Translate source information)
>> Filter 14 adv/proxy enable            (Allow pip proxy translation)
>> Filter 14 ena                         (Enable the filter)
>> Filter 14 ../port 1                   (Select SLB port 1)
>> SLB port 1# add 14                    (Add the filter to port 1)
>> SLB port 1# pip 205.178.17.12         (Set public IP address proxy)
>> SLB port 1# filt enable               (Enable filtering on port 1)
>> SLB port 1# apply                     (Apply configuration changes)
>> SLB port 1# save                      (Save configuration changes)
```

NOTE – Dynamic NAT solutions apply only to TCP/UDP traffic. Also, filters for dynamic NAT should be placed behind any static NAT filters (next example). Dynamic filters should be given higher filter numbers.

External Client Access to Server

In this example, clients on the external Internet require access to a server on the private network:

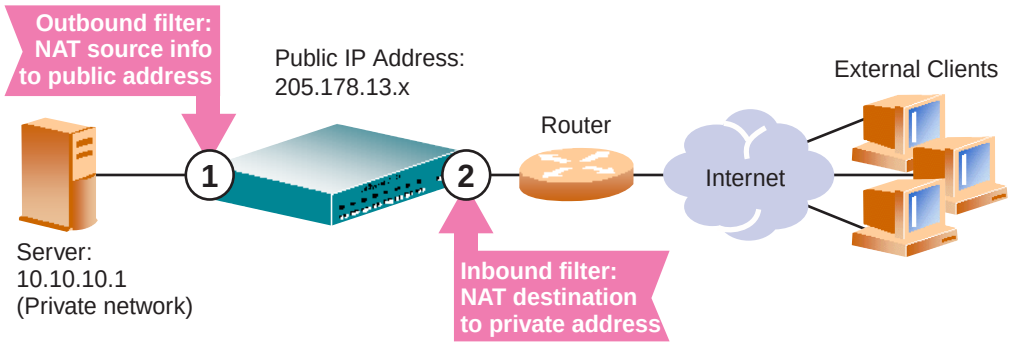


Figure 2-7 Static NAT

This static NAT (non-proxy) example requires two filters: one for the external client-side switch port, and one for the internal, server-side switch port. The client-side filter translates incoming requests for the publicly advertised server IP address to the server's internal private network address. The filter for the server-side switch port reverses the process, translating the server's private address information to a valid public address.

This could be configured as follows:

```
>> # /cfg/slb/filt 10                    (Select the menu for outbound filter)
>> Filter 10# action nat                  (Perform NAT on matching traffic)
>> Filter 10# nat source                  (Translate source information)
>> Filter 10# sip 10.10.10.0              (From the clients private IP address)
>> Filter 10# smask 255.255.255.0         (For the entire private subnet range)
>> Filter 10# dip 205.178.13.0            (To the public network address)
>> Filter 10# dmask 255.255.255.0         (For the same subnet range)
>> Filter 10# adv/proxy disable           (Override any pip proxy settings)
>> Filter 10# ena                         (Enable the filter)
>> Filter 10# ../filt 11                  (Select the menu for inbound filter)
>> Filter 11# action nat                  (Use the same settings as outbound)
>> Filter 11# nat dest                    (Reverse the translation direction)
>> Filter 11# sip 10.10.10.0              (Use the same settings as outbound)
>> Filter 11# smask 255.255.255.0         (Use the same settings as outbound)
>> Filter 11# dip 205.178.13.0            (Use the same settings as outbound)
>> Filter 11# dmask 255.255.255.0         (Use the same settings as outbound)
>> Filter 11# adv/proxy disable           (Override any pip proxy settings)
>> Filter 11# ena                         (Enable the filter)
>> Filter 11# ../port 1                   (Select server-side port)
>> SLB port 1# add 10                     (Add the outbound filter)
>> SLB port 1# filt enable                (Enable filtering on port 1)
>> SLB port 1# ../port 2                  (Select the client-side port)
>> SLB port 2# add 11                     (Add the inbound filter)
>> SLB port 2# filt enable                (Enable filtering on port 2)
>> SLB port 2# apply                      (Apply configuration changes)
>> SLB port 2# save                       (Save configuration changes)
```

Note the following important points about this configuration:

- Within each filter, the `smask` and `dmask` values are identical.
- All parameters for both filters are identical except for the NAT direction. For filter #10, `nat source` is used. For filter #11, `nat dest` is used.
- Filters for static (non-proxy) NAT should be placed ahead of dynamic NAT filters (previous example). Static filters should be given lower filter numbers.

Defining IP Address Ranges for Filters

You can specify a range of IP address for filtering both the source and/or destination IP address for traffic. When a range of IP addresses is needed, the `sip` (source) or `dip` (destination) defines the base IP address in the desired range, and the `smask` (source) or `dmask` (destination) is the mask which is applied to produce the range.

For example, to determine if a client request's destination IP address should be redirected to the cache servers attached to a particular switch, the destination IP address is masked (bit-wise AND) with the `dmask` and then compared to the `dip`.

As another example, you could configure the switch with two filters so that each would handle traffic filtering for one half of the Internet. To do this, you could define the following parameters:

Table 2-4 Filtering IP Address Ranges

Filter	Internet Address Range	dip	dmask
#1	0.0.0.0 - 127.255.255.255	0.0.0.0	128.0.0.0
#2	128.0.0.0 - 255.255.255.255	128.0.0.0	128.0.0.0

WebOS 8.0 Filtering Additions

WebOS software supports the ability to filter on all IP options, ICMP message types, and TCP flags.

In general, the switch ignores the presence of IP options when matching a frame to a filter. When configuring filters, observe the following:

- IP Options: Filtering on all options can either be enabled or disabled. Filtering a single option is not supported.
- ICMP message types: Only one message type can be set at any one time.
- TCP Flags: More than one TCP flag can be set at the same time. If there is more than one flag enabled, the flags are applied with a logical AND operator.

  Example: By setting the switch to filter `SYN` and `ACK`, the switch will filter all `SYN-ACK` frames.

Full TCP Flag Filtering

WebOS software supports packet filtering based on any or all TCP flags.

NOTE – All TCP options are disabled in the default filter configuration. What this means is that packets with TCP flags will NOT be inspected unless one or more TCP options are enabled.

Table 2-5 TCP Flags

Flag	Description
URG	Urgent
ACK	Acknowledgement
PSH	Push
RST	Reset
SYN	Synchronize
FIN	Finish

Configuring the TCP Filter

1. **Set up a filter as you would under a normal situation, configuring options such as `sip`, `smask`, `dip`, and `smask`, as needed.**

```
>> /cfg/slb/filt <filter number>/sip <IP address>
```

2. **Go to the TCP Flags Advanced Menu:**

```
>> Filter 1# adv/tcp
```

3. **Enable the flag you wish to be used to filter TCP SYN packets.**

   For example:

```
>> TCP Flags Advanced Menu# syn e
```

4. **Add the appropriate filter on the port that needs to be filtered.**

```
>> /cfg/slb/port <port number>/add <filter ID (1-224)>
```

5. **Enable filtering on the port.**

```
>> cfg/slb/port <port number>/filt ena
```

6. **To make your configuration changes active, enter apply at any prompt in the CLI.**

```
>> cfg/slb/port <port number>/apply
```

ICMP Type Filtering

WebOS supports packet filtering, based on any or all ICMP types.

NOTE – In general, the switch ignores the presence of ICMP options when matching a frame to a filter. Packets with ICMP message types will NOT be filtered out unless ICMP options are enabled.

Table 2-6 ICMP Message Types

Type #	Message Type	Description
0	echorep	ICMP echo reply
3	destun	ICMP destination unreachable
4	quench	ICMP source quench
5	redir	ICMP redirect
8	echoreq	ICMP echo request
9	rtradv	ICMP router advertisement
10	rtrsol	ICMP router solicitation
11	timex	ICMP time exceeded
12	param	ICMP parameter problem
13	timereq	ICMP timestamp request
14	timerep	ICMP timestamp reply
15	inforeq	ICMP information request
16	inforep	ICMP information reply
17	maskreq	ICMP address mask request
18	maskrep	ICMP address mask reply

Configuring the ICMP Filter

NOTE – If you want to configure ICMP type filters, you should disable the `cache` option in the Filter Advanced Menu (`cfg/slb/filt` <*filter-number*>`/adv`).

1. **Set up a filter as you would under a normal situation, configuring options such as `sip`, `smask`, `dip`, and `smask`, as needed.**

```
>> /cfg/slb/filt <filter number>/sip <IP address>
```

2. **Set protocol type to ICMP.**

```
>> /cfg/slb/filt <filter number>/proto icmp
```

3. **Go to the Filter Advanced Menu and select the ICMP option. Then, enter the option you want to be used to filter ICMP packets:**

```
>> Filter 1# adv/icmp
Current ICMP message type:      any
Enter ICMP message type or any: <message-type|number>
```

4. **Add the appropriate filter on the port that needs to be filtered.**

```
>> /cfg/slb/port <port number>/add <filter ID (1-224)>
```

5. **Enable filtering on the port.**

```
>> cfg/slb/port <port number>/filt ena
```

6. **To make your configuration changes active, enter `apply` at any prompt in the CLI.**

FTP Client NAT (Active FTP for Dynamic NAT)

Alteon WebSystems switches provide Network Address Translation (NAT) services to many clients with private IP addresses. However, on switches running WebOS 6.0, clients using active FTP cannot send a request to a remote FTP server when their client IP address is private. In WebOS 8.0, an FTP enhancement now provides the capability to perform true FTP NAT for dynamic NAT.

Because of the way FTP works in active mode, a client will send information on the control channel, information that reveals their private IP address, out to the Internet. However, the switch filter only performs NAT translation on the TCP/IP header portion of the frame, preventing a client with a private IP address from doing active FTP.

In WebOS, the switch can monitor the control channel and replace the client 's private IP address with a proxy IP (PIP) address defined on the switch. When a client in active FTP mode sends a "PORT" command to a remote FTP server, the switch will look into the data part of the frame and modify the PORT command as follows:

- The real server IP address will be replaced by a public proxy IP address, using a pool of proxy IP addresses, instead of a single one.
- The real server port will be replaced with a proxy port.

NOTE – The passive mode does not need this feature.

Configuring Active FTP Client NAT

1. **Make sure there's a proxy IP address enabled on the filter port.**
2. **Make sure there's a source NAT filter set up for the port.**
3. **Enable active FTP NAT using the following command:**

```
>> /cfg/slb/filt <filter number>/adv/ftpa e
```

4. **Apply and save the switch configuration.**

CHAPTER 3

Application Redirection

NOTE – To access Application Redirection functionality, the optional Layer 4 software must be enabled in the switch (see "Filtering and Layer 4" in Chapter 8 of the *WebOS 8.0 Command Reference*).

Web-Cache Redirection Example

For many companies, the Internet is an indispensable source for business and technical information. Much of the information brought into your company from the Internet, however, is not unique. Often, clients will access the same information many times as they return to a web-page for additional information or to explore other links.

Duplicate information may be requested more inadvertently as the myriad components that make up Internet data (pictures, buttons, frames, text, and so on) are reloaded from page to page. Add multiple clients to the picture, and the amount of repeated data that comes in through your Internet router can account for a great deal of its congestion. Redundant requests consume a considerable portion of your available bandwidth to the Internet.

Web-cache redirection can help alleviate the congestion seen at your Internet router. When Application Redirection filters are properly configured for your WebOS-powered switch, outbound client requests for Internet data are intercepted and redirected to a group of web-cache servers on your network. The web-cache servers duplicate and store inbound Internet data that has been requested by your clients. If the web-cache servers recognize a client's outbound request as one that can be filled with cached information, the web-cache servers will supply the information, rather than sending the request out across the Internet.

In addition to increasing the efficiency of your network, access to locally cached information can be granted much faster than by requesting the same information across the Internet.

Web-Cache Redirection Environment

Consider a network where client HTTP requests begin to regularly overload the Internet router.

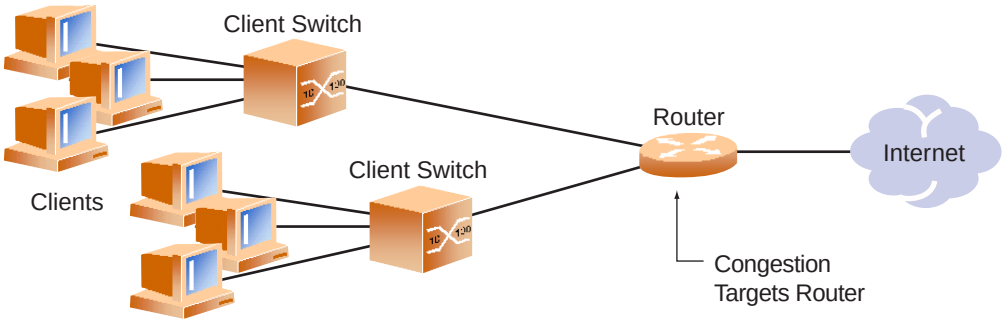


Figure 3-1 Traditional network without Web Cache Redirection

The network needs a solution that addresses the following key concerns:

- The solution must be readily scalable
- The administrator should not have to reconfigure all the clients' browsers to use Proxy Servers.

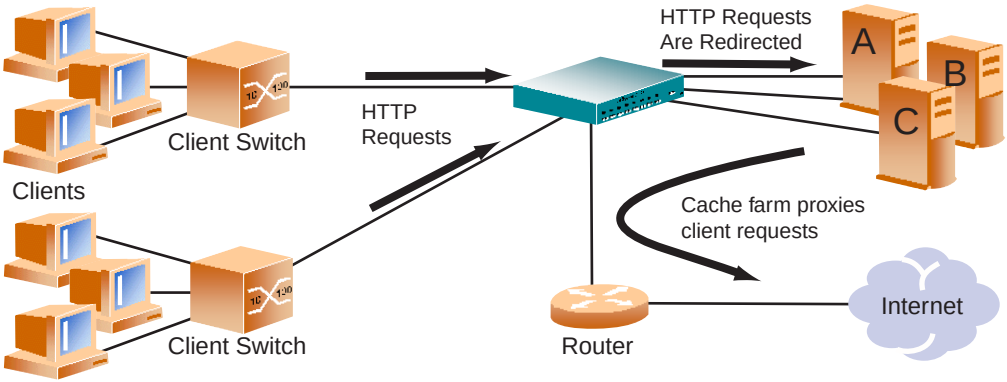


Figure 3-2 Network with Web Cache Redirection

Adding an Alteon WebSystems switch with optional Layer 4 software addresses these issues:

- Web-cache servers can be added or removed dynamically without interrupting services.
- Performance is improved by balancing the cached web request load across multiple servers. More servers can be added at any time to increase processing power.
- The proxy is transparent to the client.
- Frames that are not associated with HTTP requests are passed normally to the router.

Example Configuration for the Web-Cache Solution

The following is required prior to configuration:

- You must be connected to the switch command-line interface as the administrator.
- Optional Layer 4 software must be enabled.

NOTE – For details about the procedures above, and about any of the menu commands described in this example, see the *WebOS 8.0 Command Reference*.

In this example, an ACEswitch 180 is placed between the clients and the border gateway to the Internet. The switch will be configured to intercept all Internet bound HTTP requests (on default TCP port 80), and redirect them to the web-cache servers. The switch will distribute HTTP requests equally to the web-cache servers based on the destination IP address of the requests.

Also, filters are not limited to the few protocols and TCP or UDP applications shown in this example. See the *WebOS 8.0 Command Reference* for a list of other well-known protocols and services.

1. **Assign an IP address to each of the web-cache servers.**

   Just as with Server Load Balancing, the web-cache real servers will be assigned an IP address and placed into a real server group. The real servers must be in the same VLAN and must have an IP route to the switch that will perform the web-cache redirection. In addition, the path from the switch to the real servers must not contain a router. The router would stop HTTP requests from reaching the web-cache servers, instead directing them back out to the Internet.

   More complex network topologies can be used if configuring IP proxy addresses (see "IP Proxy Addresses for Transparent Proxies or Complex Networks" on page 74).

   For this example, the three web-cache real servers have the following IP addresses on the same IP subnet:

   Table 3-1 Web-Cache Example: Real Server IP addresses

Web Cache Server	IP address
Server A	200.200.200.2
Server B	200.200.200.3
Server C	200.200.200.4

2. **Install web-cache software on all three web-cache servers.**

3. **Full Network Address Translation (NAT) is required.**

   Install transparent proxy software on all three web-cache servers, or define proxy IP addresses on the switch (see "IP Proxy Addresses for Transparent Proxies or Complex Networks" on page 74).

4. **Define an IP interface on the switch.**

   Because, by default, the switch only re-maps destination MAC addresses, it must have an IP interface on the same subnet as the three web-cache servers.

   To configure an IP interface for this example, enter this command from the CLI:

```
>> Main# /cfg/ip/if 1                          (Select IP interface #1)
>> IP Interface 1# addr 200.200.200.100       (Assign IP address for the interface)
>> IP Interface 1# ena                         (Enable IP interface #1)
```

NOTE – The IP interface and the real servers must be in the same subnet. This example assumes that all ports and IP interfaces use default VLAN #1, requiring no special VLAN configuration for the ports or IP interface.

5. **On the switch, define each Real Server**

   For each web-cache real server, you must assign a real server number, specify its actual IP address, and enable the real server. For example:

```
>> ip# /cfg/slb/real 1                         (Server A is real server 1)
>> Real server 1 # rip 200.200.200.2           (Assign Server A IP address)
>> Real server 1 # ena                         (Enable real server 1)
>> Real server 1 # ../real 2                   (Server B is real server 2)
>> Real server 2 # rip 200.200.200.3           (Assign Server B IP address)
>> Real server 2 # ena                         (Enable real server 2)
>> Real server 2 # ../real 3                   (Server C is real server 3)
>> Real server 3 # rip 200.200.200.4           (Assign Server C IP address)
>> Real server 3 # ena                         (Enable real server 3)
```

6. **On the switch, define a Real Server Group.**

   This places the three web-cache real servers into one service group:

```
>> Real server 3 # /cfg/slb/group 1            (Select real server group 1)
>> Real server group 1 # add 1                 (Add real server 1 to group 1)
>> Real server group 1 # add 2                 (Add real server 2 to group 1)
>> Real server group 1 # add 3                 (Add real server 3 to group 1)
```

7. **On the switch, set the Real Server Group metric to `minmisses`.**

   This helps minimize web-cache misses in the event real servers fail or are taken out of service:

```
>> Real server group 1 # metrc minmisses   (Metric for minimum cache misses.)
```

8. **On the switch, verify that server processing is disabled on the ports supporting application redirection.**

NOTE – Do not use the "server" setting on a port with Application Redirection enabled. Server processing is used only with Server Load Balancing. To disable server processing on the port, use the commands on the **`/cfg/slb/port`** menu, as described in Chapter 8 of the *WebOS 8.0 Command Reference*.

---

9. **On the switch, create a filter that will intercept and redirect all client HTTP requests.**

   The filter must be able to intercept all TCP traffic for the HTTP destination port, and must redirect it to the proper port on the real server group:

```
>> SLB port 6 # /cfg/slb/filt 2          (Select the menu for Filter #2)
>> Filter 2# sip any                     (From any source IP addresses)
>> Filter 2# dip any                     (To any destination IP addresses)
>> Filter 2# proto tcp                   (For TCP protocol traffic)
>> Filter 2# sport any                   (From any source port)
>> Filter 2# dport http                  (To an HTTP destination port)
>> Filter 2# action redir                (Set the action for redirection)
>> Filter 2# rport http                  (Set the redirection port)
>> Filter 2# group 1                     (Select real server group 1)
>> Filter 2# ena                         (Enable the filter)
```

   The `rport` parameter must be configured whenever TCP/UDP protocol traffic is redirected. The `rport` parameter defines the real server TCP or UDP port to which redirected traffic will be sent. The port defined by the `rport` parameter is used when performing Layer 4 health checks of TCP services.

   Also, if transparent proxies are used for Network Address Translation (NAT) on the switch (see Step 3. on page 70), the `rport` parameter must be configured for all Application Redirection filters. Take care to use the proper port designation with `rport`: if the transparent proxy operation resides on the host, the well-known port (80, or "http") is probably required. If the transparent proxy occurs on the switch, make sure to use the service port required by the specific software package.

See "IP Proxy Addresses for Transparent Proxies or Complex Networks" on page 74 for more about IP proxy addresses.

10. **On the switch, create a default filter.**

In this case, the default filter will allow all non-cached traffic to proceed normally:

```
>> Filter 2# ../filt 224                (Select the default filter)
>> Filter 224# sip any                  (From any source IP addresses)
>> Filter 224# dip any                  (To any destination IP addresses)
>> Filter 224# proto any                (For any protocols)
>> Filter 224# action allow             (Set the action to allow traffic)
>> Filter 224# ena                      (Enable the default filter)
```

NOTE – When the `proto` parameter is not tcp or udp, then `sport` and `dport` are ignored.

11. **On the switch, assign the filters to the client ports.**

Assuming that the redirected clients are connected to physical switch ports 5 and 6, both ports are configured to use the previously created filters as follows:

```
>> Filter 224# ../port 5                (Select the SLB port 5)
>> SLB Port 5 # add 2                   (Add filter 1 to port 5)
>> SLB Port 5 # add 224                 (Add the default filter to port 5)
>> SLB Port 5 # filt enable             (Enable filtering for port 5)
>> SLB Port 5 # ../port 6               (Select the SLB port 6)
>> SLB Port 6 # add 2                   (Add filter 1 to port 6)
>> SLB Port 6 # add 224                 (Add the default filter to port 6)
>> SLB Port 6 # filt enable             (Enable filtering for port 6)
```

12. **On the switch, enable, apply, and verify the configuration.**

```
>> SLB Port 6 # ..                      (Select Server Load Balancing Menu)
>> Server Load Balancing# on            (Activate Layer 4 software services)
>> Server Load Balancing# apply         (Make your changes active)
>> Server Load Balancing# cur           (View current settings)
```

NOTE – Server Load Balancing must be turned on in order for Application Redirection to work properly. The "`on`" command is valid only if the optional Layer 4 software is enabled on your switch (see "Activating Optional Software" in the *WebOS 8.0 Command Reference*.

13. **Examine the resulting information from the "`cur`" command. If any settings are incorrect, make appropriate changes.**

14. **On the switch, save your new configuration changes.**

```
>> Server Load Balancing# save                    (Save for restore after reboot)
```

15. **On the switch, check the Server Load Balancing information.**

```
>> Server Load Balancing# /info/slb               (View SLB information)
```

Check that all Server Load Balancing parameters are working according to expectation. If necessary, make any appropriate configuration changes and then check the information again.

NOTE – Changes to filters on a given port only effect new sessions. To make filter changes take effect immediately, clear the session binding table for the port (see the `/oper/slb/clear` command in the *WebOS 8.0 Command Reference*).

IP Proxy Addresses for Transparent Proxies or Complex Networks

Transparent proxies provide the benefits listed below when used with Application Redirection. Application redirection is automatically enabled when a filter with the `redir` action is applied on a port.

- With proxies IP addresses configured on redirected ports, the switch can redirect client requests to servers located on any subnet, anywhere.
- The switch can perform transparent substitution for all source and destination addresses, including destination port remapping. This provides support for comprehensive, fully-transparent proxies. These proxies are transparent to the user. No additional client configuration is needed.

The following procedure can be used for configuring proxy IP addresses:

1. **Add proxy IP addresses to the redirection ports.**

   Each of the ports using redirection filters require proxy IP addresses to be configured. Each proxy IP address must be unique on your network. These are configured as follows:

```
>> SLB port 3# /cfg/slb/port 5              (Select network port #5)
>> SLB port 5# pip 200.200.200.68           (Set proxy IP address for port #5)
>> SLB port 5# ../proxy ena                 (Enable proxy port #5)
>> SLB port 5# ../port 6                    (Select network port #6)
>> SLB port 6# pip 200.200.200.69           (Set proxy IP address for port #6)
>> SLB port 6# ../proxy ena                 (Enable proxy port #6)
```

2. **If VMA is enabled, add proxy IP addresses for all other switch ports (except Port 9).**

   Virtual Matrix Architecture (VMA) is normally enabled on the switch. In addition to enhanced resource management, this feature eliminates many of the restrictions found in earlier versions of the WebOS. It does require, however, than when any switch port is configured with an IP proxy address, all ports must be configured with IP proxy addresses. Otherwise, if VMA is disabled, only the client port with filters need proxy IP addresses and this step can be skipped.

The following commands can be used to configure the additional unique proxy IP addresses:

```
>> SLB port 6# ../port 1                    (Select network port #1)
>> SLB port 1# pip 200.200.200.70           (Set proxy IP address for port #1)
>> SLB port 1# ../port 2                    (Select network port #2)
>> SLB port 2# pip 200.200.200.71           (Set proxy IP address for port #2)
>> SLB port 2# ../port 3                    (Select network port #3)
>> SLB port 3# pip 200.200.200.72           (Set proxy IP address for port #3)
>> SLB port 3# ../port 4                    (Select network port #4)
>> SLB port 4# pip 200.200.200.73           (Set proxy IP address for port #4)
>> SLB port 4# ../port 7                    (Select network port #7)
>> SLB port 7# pip 200.200.200.74           (Set proxy IP address for port #7)
>> SLB port 7# ../port 8                    (Select network port #8)
>> SLB port 8# pip 200.200.200.75           (Set proxy IP address for port #8)
```

NOTE – Port 9 does not require a proxy IP address with VMA enabled.

See the *WebOS 8.0 Command Reference* for more information (`/cfg/slb/adv/matrix`).

3. **Configure the Application Redirection filters.**

Once proxy IP addresses are established, you need to configure each Application Redirection filter (filter 2 in our example) with the real server TCP or UDP port to which redirected traffic will be sent. In this case, we are mapping the requests to different destination port (8080). You must also enable proxies on the real servers:

```
>> # /cfg/slb/filt 2                        (Select the menu for Filter #2)
>> Filter 2 # rport 8080                    (Set proxy redirection port)
>> Filter 2 # real 1/proxy enable           (Enable proxy on real servers)
>> Real server 1 # ../real 2/proxy enable   (Enable proxy on real servers)
>> Real server 2 # ../real 3/proxy enable   (Enable proxy on real servers)
```

NOTE – This configuration is not limited to HTTP Web service. Other TCP/IP services can be configured in a similar fashion. For example, if this had been a DNS redirect, `rport` would be sent to well-known port 53 (or the service port you want to remap to.) For a list of other well-known services and ports, see the *WebOS 8.0 Command Reference*.

4. **Apply and save your changes.**

5. **Check server statistics to verify that traffic has been redirected based on filtering criteria:**

```
>> # /info/slb/group <group#>/filter <filter#>    (View statistics for server filter)
```

Excluding Non-Cacheable Sites

Some Web sites provide content which isn't well suited for redirection to cache servers. Such sites might provide browser-based games, applications that keep real-time session information or authenticate by client IP address.

To prevent such sites from being redirected to cache-servers, create a filter which allows this specific traffic to pass normally through the switch. This filter must have a higher precedence (a lower filter number) than the Application Redirection filter.

For example, if you wished to prevent a popular web-based game site on subnet 200.10.10.* from being redirected, you could add the following to the previous example configuration:

```
>> # /cfg/slb/filt 1                    (Select the menu for Filter #1)
>> Filter 1# dip 200.10.10.0             (To the site's destination IP address)
>> Filter 1# dmask 255.255.255.0         (For entire subnet range)
>> Filter 1# sip any                     (From any source IP address)
>> Filter 1# proto tcp                   (For TCP traffic)
>> Filter 1# dport http                  (To an HTTP destination port)
>> Filter 1# sport any                   (From any source port)
>> Filter 1# action allow                (Allow matching traffic to pass)
>> Filter 1# ena                         (Enable the filter)
>> Filter 1# ../port 5                   (Select SLB port 5)
>> SLB port 5# add 1                     (Add the filter to port 5)
>> SLB port 5# ../port 6                 (Select SLB port 6)
>> SLB port 6# add 1                     (Add the filter to port 6)
>> SLB port 6# apply                     (Apply configuration changes)
>> SLB port 6# save                      (Save configuration changes)
```

Defining IP Address Ranges for Filters

You can specify a range of IP address for filtering both the source and/or destination IP address for traffic. When a range of IP addresses is needed, the `sip` (source) or `dip` (destination) defines the base IP address in the desired range, and the `smask` (source) or `dmask` (destination) is the mask which is applied to produce the range.

For example, to determine if a client request's destination IP address should be redirected to the cache servers attached to a particular switch, the destination IP address is masked (bit-wise AND) with the `dmask` and then compared to the `dip`.

As another example, you could configure the switch with two filters so that each would handle traffic filtering for one half of the Internet. To do this, you could define the following parameters:

Table 3-2 Filtering IP Address Ranges

Filter	Internet Address Range	dip	dmask
#1	0.0.0.0 - 127.255.255.255	0.0.0.0	128.0.0.0
#2	128.0.0.0 - 255.255.255.255	128.0.0.0	128.0.0.0

Additional Application Redirection Options

Application Redirection can be used in combination with other Layer 4 options such as load balancing metrics, health checks, real server group backups, and more. See "Additional SLB Options" on page 30 for details.

CHAPTER 4

Health Checking

Content-intelligent Web switches allow webmasters to customize server health checks to verify content accessibility in large Web sites. As the amount of content grows and information is distributed across different server farms, flexible, customizable content health checks are critical to ensuring end-to-end availability.

Health Check Features

WebOS 8.0 health check additions are summarized below:

- "send/expect" script-based health checks

  Dynamically verify application and content availability using "scripts." These scripts execute a sequence of tests to verify application and content availability.

- The switch can check availability of RADIUS content and SSL connections.
- Option to keep sending traffic on existing connections to a server that failed Layer 4 or content health check but reachable via ICMP.
- Ability to detect that a number of servers (*n*) in a server group are down and send a message to syslog host. The number *n* is user configurable.
- Layer 2 health check or ARP health check
  - Optional health check method for default gateway to be either ARP or Ping
  - If the default gateway that the switch is configured to use is a VRRP router, it may not answer pings according to the standard and the health check will fail.
- SSL "Hello" Health Checks

  A health check option on the Real Server Group Menu (`/cfg/slb/group/healt/sslh`) allows the switch to query the health of SSL servers.

Health-Check Parameters for Real Servers

By default, the switch checks the status of each service on each real server every two seconds. Sometimes, the real server may be too busy processing connections to respond to health checks. If a service does not respond to four consecutive health checks, the switch, by default, declares the service unavailable. Both the health check interval and the number of retries can be modified.

```
>> # /cfg/slb/real real-server-number      (Select the real server)
>> Real server# intr 4                     (Check real server every 4 seconds)
>> Real server# retry 6                    (If 6 consecutive health checks fail,
                                            declare real server down)
```

Hostname for HTTP Content Health Checks

HTTP-based health checks can include the hostname for `HOST:` headers. The `HOST:` header and health check URL are constructed from the following components:

Item	Option	Configured Under	Maximum Length
Virtual server hostname	`hname`	`/cfg/slb/virt`	9 characters
Domain name	`dname`	`/cfg/slb/virt`	35 characters
Server group health check field	`content`	`/cfg/slb/group`	34 characters

If the `HOST:` header is required, an `HTTP/1.1 GET` will occur, otherwise an `HTTP/1.0 GET` will occur.

Example 1:

```
hname   = compute
dname   = alteon.com
content = index.html
```

Health check is performed using:

```
GET /index.html HTTP/1.1
Host: compute.alteon.com
```

Example 2:

```
hname   = (none)
dname   = raleighduram.cityguru.com
content = /page/gen/?_template=alteon
```

Health check is performed using:

```
GET /page/gen/?_template=alteon HTTP/1.1
Host: raleighduram.cityguru.com
```

Example 3:

```
hname   = (none)
dname   = compute
content = index.html
```

Health check is performed using:

```
GET /index.html HTTP/1.1
Host: compute
```

Example 4:

```
hname   = (none)
dname   = (none)
content = index.html
```

Health check is performed using:

`GET /index.html HTTP/1.0` (since no `HTTP HOST:` header is required)

Example 5:

```
hname   = (none)
dname   = (none)
content = //compute/index.html
```

Health check is performed using:

```
GET /index.html HTTP/1.1
Host: compute
```

RADIUS Server Health Checking

The RADIUS (Remote Authentication Dial In User Service) protocol is used to authenticate dial-up users to remote access servers (RAS) and the client application they will use during the dial-up connection.

- RADIUS Content Health Check Enhancements
  - Include the switch IP as the NAS_IP parameter in the RADIUS content health check
  - RADIUS health check using real server port configured, that is, the `rport`.
  - Variable length RADIUS secret password. Supports less than 16 octets and up to 32 octets

RADIUS is stateless and uses UDP as its transport protocol. To support RADIUS health checking, the network administrator must configure two parameters in the switch: the `/cfg/slb/secret` value and the `content` parameter with a *username:password* value.

>> # **/cfg/slb/group** *<real server group number>*	*(Select the real server group.)*
>> # **health radius**	*(Specify the type of health checking to be performed.)*
>> # **content <***username***>:<***password***>**	*(Specify the RADIUS username:password value.)*
>> # **../adv/secret <***RADIUS coded value***>**	*(Enter up to 32 alphanumeric characters used to encrypt and decrypt password.)*

- The `secret` value is a field of up to 32 alphanumeric characters that is used by the switch to encrypt a password during the RSA Message Digest Algorithm (MD5) and by the RADIUS server to decrypt the password during verification.
- The `content` option specifies the *username:password* value that the server tries to match in its user database. In addition to verifying the user name and password, the database may specify the client(s) or port(s) to which the user is allowed access.

RADIUS Server Content Health Checks

The Alteon Web switch will provide the NAS_IP parameter while performing RADIUS content health checks. The switch uses the IP address of the IP interface that has the same subnet of the RADIUS server or the default gateway as the NAS_IP.

NOTE – Although the NAS_IP is not specified as being required in the RFC, Ascend's native RADIUS servers require this parameter.

The RADIUS health check will be performed using the configured real server port, that is, the `rport`.

IMAP Server Health Checking

IMAP (Internet Message Access Protocol) is a mail server protocol that is used between a client system and a mail server, to allow a user to retrieve and manipulate mail messages they have received.

NOTE – IMAP is not used for mail transfers between mail servers or from clients to a mail server.

IMAP servers listen to TCP port 143. To support IMAP health checking, the network administrator must configure a *username:password* value in the switch, using the `content` option on the SLB Real Server Group Menu (`/cfg/slb/group`).

>> # **`/cfg/slb/group`** *real-server-group-number*	*(Select the real server group.)*
>> # **`slb/group/health`** ***`imap`***	*(Specify the type of health checking to be performed.)*
>> # **`slb/group/content`** *username:password*	*(Specify the IMAP username:password value.)*

The `content` option specifies the *username:password* value that the server tries to match in its user database. In addition to verifying the user name and password, the database may specify the client(s) or port(s) to which the user is allowed access.

Script-Based Health Checks

Using this feature, you can configure the switch to send a series of health check requests to real servers and monitor the responses. To verify application and content availability, you can define a list of health check statements in send/expect format.

NOTE – Health check scripts can only be set up via the command-line interface.

The benefits of using script-based health checks are listed below:

- Ability to send multiple commands
- Check for any return string
- Test availability of different applications
- Test availability of multiple domains or websites

The expect string can be any string in the entire response. WebOS 8.0 supports the following capacity for a single switch:

- # bytes per script = 1024
- # scripts per switch = 8
- # health check statements (HTTP GET and expect strings) = approximately 10 to 15

Script Format

The general format for health check scripts is shown below, with a detailed example following.

```
open application_port  (e.g., 80 for HTTP, 23 for Telnet, etc.)
send request1
expect response1
send request2
expect response2
send request3
expect response3
close
```

NOTE – If you will be doing HTTP 1.1 pipelining, you'll need to individually open and close each response in the script.

Example #1: Configure the switch to check a series of web pages (HTML or dynamic CGI scripts) before it declares a real server up. A sample script is shown below:

```
/cfg/slb/group x/health script1; content: none
/cfg/slb/adv/script1

open 80
send GET /index.html HTTP/1.1\\r\\nHOST:192.192.1.1\\r\\n
expect HTTP/1.0 200
close
open 80
send GET /script.cgi HTTP/1.1\\r\\nHOST:192.192.1.1\\r\\n
expect HTTP/1.0 200
close
open 443
…
close
```

NOTE – You must type two "\"s before an "n" or "r" when entering the send string as an argument to the command "send". If the text string is instead entered after hitting <return>, then only one "\" is needed before the "n" or "r".

Example #2: **GSLB URL Health Check**

In earlier WebOS releases, each remote GSLB site's VIP was required to be a real server of the local switch. Each switch sends a health check request to the other switch's virtual servers that are configured on the local switch. The health check is successful if there is at least one real server on the remote switch that is up. If all real servers on the remote switch are down, the remote real server (a virtual server of a remote switch) will respond with an HTTP Redirect message to the health check.

Using the scriptable health check feature in WebOS 8.0, network administrators can set up health check statements to check all the sub-strings involved in all the real servers.

Site #1 with VIP_1 and the following RIPs:

- RIP_1 and RIP_2: "images"
- RIP_3 and RIP_4: "html"
- RIP_5 and RIP_6: "cgi and "bin"
- RIP_7: which is VIP_2, "any"

Site #2 with VIP_2 and the following RIPs:

- RIP_1 and RIP_2: “images”
- RIP_3 and RIP_4: “html”
- RIP_5 and RIP_6: “cgi” and “bin”
- RIP_7: which is VIP_1, “any”

A sample script is shown below:

```
/cfg/slb/group x/health script2; content: none
/cfg/slb/adv/script2

open 80
send GET /images/default.asp HTTP/1.1\r\nHOST: 192.192.1.2\\r\\n\\r\\n
expect HTTP/1.1 200
close

open 80
send GET /install/default.html HTTP/1.1\r\nHOST: 192.192.1.2\\r\\n\\r\\n
expect HTTP/1.1 200
close

open 80
send GET /script.cgi HTTP/1.1\\r\\nHOST: www.alteon.com\\r\\n\\r\\n
expect HTTP/1.1 200
close
```

Script-based health checking is intelligent in that it will only send the appropriate requests to the relevant servers. In the example above, the first GET statement will only be sent to RIP_1 and RIP_2. Going through the health check statements serially will ensure that all content are available by at least one real server on the remote site.

You should configure the remote RIP (the VIP of the remote site) to accept “any” URL requests. The purpose of the first GET is to check if RIP_1 or RIP_2 is up; that is, to check if the remote site has at least one server for “images” content. Either RIP_1 or RIP_2 will respond to the first GET health check.

If all the RIPs are down, RIP_7 (the VIP of the remote site) will respond with an HTTP Redirect (respond code 302) to the health check. Thus, the health check will fail as the expected respond code is 200. This ensures that the HTTP Redirect messages will not cause a loop.

HTTPS/SSL Health Check

The `sslh` health check option on the Real Server Group Menu (`/cfg/slb/group/healt/sslh`) allows the switch to query the health of the SSL servers by sending an SSL client "Hello" packet and then verify the contents of the server's "Hello" response. SSL health check is performed using real server port configured, that is, the `rport`

The SSL enhanced health check behavior is summarized below:

- The switch sends a SSL "Hello" packet to the SSL server.
- If it is up and running, the SSL server responds with the "Server Hello" message.
- The switch verifies various fields in the response and marks the service "UP" if the fields are OK.

During the handshake, the user and server exchange security certificates, negotiate an encryption and compression method, and establish a session ID for each session.

CHAPTER 5

Global Server Load Balancing

This chapter provides a conceptual overview of Global Server Load Balancing (GSLB) and configuration examples for performing GSLB across multiple geographic sites. The number of supported GSLB sites per switch has been increased to 64, with a total aggregate of 2048 service/site combinations.

NOTE – Both the optional Server Load Balancing and Global Server Load Balancing software keys must be enabled. See the *WebOS Command Reference* for details.

GSLB Overview

Global Server Load Balancing (GSLB), lets you balance server traffic load across multiple physical sites. This allows you to smoothly integrate the resources of a world-wide series of server sites, and to balance web content (or other services) intelligently among them. Alteon WebSystems' GSLB system takes into account individual sites' health, response time, and geographic location for a global performance perspective.

NOTE – URL-based server load balancing is compatible with GSLB. Cookie-based persistence is compatible with GSLB using Active Cookie Mode (Cookie Rewrite Mode).

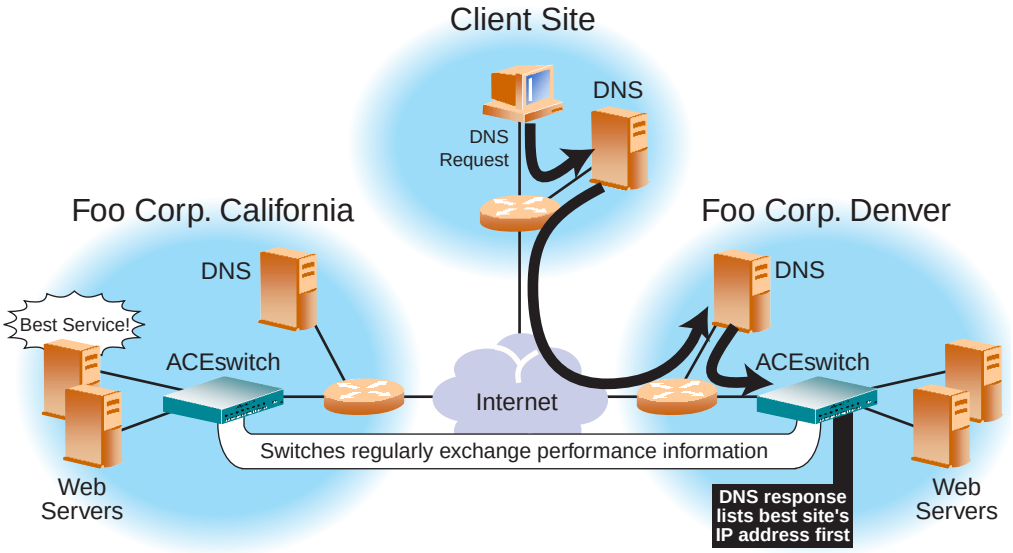
Benefits

GSLB meets the following demands for distributed network services:

- High content availability through distributed content and distributed decision making. If one site becomes disabled, the others become aware of it and take up the load.
- No latency during client connection set up. Instant site hand-off decisions can be made by any distributed switch.
- The best performing sites get a majority of traffic over a given period of time, but are not overwhelmed.
- Switches at different sites regularly exchange information through DSSP (Distributed Site State Protocol), and can trigger exchanges when any site's health status changes. This ensures that each active site has valid state knowledge and statistics.
- Takes geography into account, as well as network topology.
- Gives creative control to the administrator or web-master to build and control content by user, location, target application, and more.
- Easy to deploy, manage, and scale. Switch configuration is straight-forward. There are no complex system topologies involving routers, protocols, etcetera.
- Provides flexible design options.
- Supports all IP protocols.

How GSLB Works

Consider the following sample network:



1. Browser requests www.foocorp.com IP address from local DNS.

2. Client's DNS asks its upstream DNS, which in turn asks the next, and so on, until the address is resolved.

3. The Foo Corp. Denver DNS knows that the local ACEswitch is an authoritative name server for www.foocorp.com.

4. The switch DSLB software knows that Foo Corp. California currently provides better service, and responds with Foo Corp. California's virtual IP address listed first.

5. The client connects to Foo Corp. California for the best service.

Figure 5-1 DNS Resolution with Global Server Load Balancing

In this example, a client is using their web-browser to view the web-site for the Foo Corporation at www.foocorp.com. The Foo Corporation has two sites: one in California, and one in Denver, each with identical content and services available. Both sites have an Alteon WebSystems' Web switch configured for Global Server Load Balancing. These switches are also configured as the Authoritative Name Servers for www.foocorp.com.

When a client loads their web-browsing software and enters the URL for a website such as www.foocorp.com, a query is sent to the client's local DNS server, asking for the IP address that represents the domain name entered. If the local DNS server does not have this information cached, it will in turn ask a DNS server further upstream. Eventually, the request will reach an upstream DNS server that has this information on hand, or it will reach one of the Foo Corporation's DNS servers. The Foo Corporation's DNS server has been configured to know that the local Alteon WebSystems' Web switch with Global Server Load Balancing software is the authoritative name server for www.foocorp.com.

Each switch with GSLB software is capable of responding to the client's name resolution request. Since each switch regularly checks and communicates health and performance information with its peers, either switch can determine which site (or sites) are best able to serve the client's web-cruising needs, and can respond with a list of IP addresses for the Foo Corporation's distributed sites, prioritized by performance, geography, and other criteria.

The client's web browser will use the IP address information to open a connection to the best available site. The IP addresses represent virtual servers at any site, which are locally load balanced according to regular Server Load Balancing configuration.

If the site serving the client HTTP content suddenly experiences a failure (no healthy real servers) or becomes overloaded with traffic (all real servers reach their maximum connection limit), the switch will issue an HTTP Redirect and transparently cause the client to connect to another peer site.

The end result is that the client gets quick, reliable service with no latency and no special client-side configuration.

GSLB Configuration Example

Summary

Configuring Global Server Load Balancing is simply an extension of the configuration procedure for Server Load Balancing. The process is summarized as follows:

- Use the administrator login to connect to the switch you are configuring.
- Activate Server Load Balancing and Global Server Load Balancing software keys.
- Configure the switch at each site with basic attributes
  - Configure the switch IP interface
  - Configure the default gateways
- Configure the switch at each site to act as Domain Name System (DNS) server for each service hosted on its virtual servers. Also, configure the local DNS server to recognize the switch as the authoritative DNS server for the hosted services.
- Configure the switch at each site as usual for local Server Load Balancing.
  - Define each local real server
  - Group local real servers into real server groups
  - Define the local virtual server with its IP address, services, and real server groups
  - Define the switch port states
  - Enable Server Load Balancing
- Finally, make each switch recognize its remote peers.
  - On each switch, configure a remote real server entry for each remote service.
  - Add the remote real server entry to an appropriate real server group.
  - Enable Global Server Load Balancing

Example GSLB Configuration Procedure

Consider the following example network:

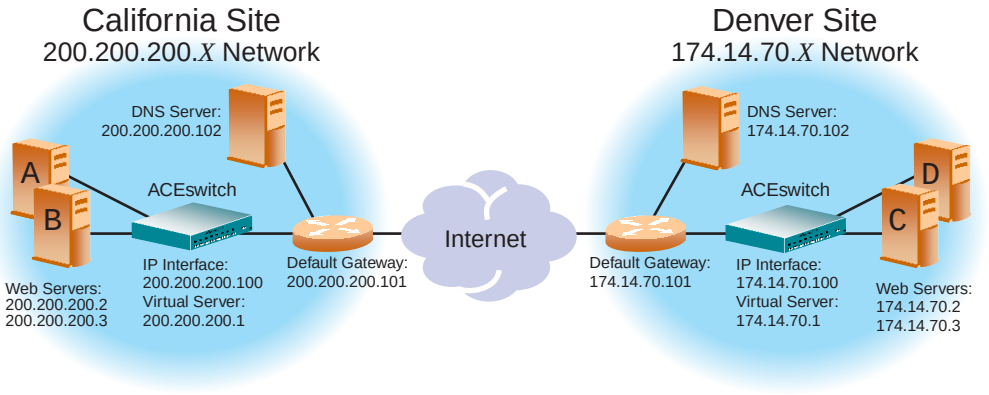


Figure 5-2 Global Server Load Balancing Example Topology

In the following examples, many of the options are left to their default values. See "Additional SLB Options" on page 30 for more options.

The following is required prior to configuration:

- You must be connected to the switch command-line interface as the administrator.
- Both of the following optional software keys must be activated:
  - Server Load Balancing
  - Global Server Load Balancing

NOTE – For details about any of the processes or menu commands described in this example, see the *WebOS 8.0 Command Reference*.

Part One: Configure the California Site with Basic System Items

1. **If the web-based interface is to be used for managing the California switch, change its service port.**

   Global Server Load Balancing uses service port 80 on the IP interface for DSSP updates. By default, the WebOS Web-based interface also uses port 80. Both services cannot use the same port. If the Web-based interface is enabled (see the `/cfg/sys/http` command in Chapter 7 of the *WebOS 8.0 Command Reference*), configure it to use a different port.

   For example, to change the web-based interface port to 8080, enter the following command:

```
>> Main# /cfg/sys                    (Select the System Menu)
>> System# wport 8080                (Set service port 8080 for web UI)
```

2. **On the California switch, define an IP interface.**

   The switch IP interface is the entity that responds when asked to resolve client DNS requests. The IP interface must have an IP route to the local real servers. The switch uses this path to determine the level of TCP/IP reachability of the real servers.

   To configure an IP interface for this example, enter these commands from the CLI:

```
>> System# /cfg/ip/if 1                    (Select IP interface #1)
>> IP Interface 1# addr 200.200.200.100    (Assign IP address for the interface)
>> IP Interface 1# ena                     (Enable IP interface #1)
```

NOTE – This example assumes that all ports and IP interfaces use default VLAN #1, requiring no special VLAN configuration for the ports or IP interface.

3. **On the California switch, define the default gateway.**

   In this example, a router at the edge of the site acts as the default gateway to the Internet. To configure the default gateway for this example, enter these commands from the CLI:

```
>> IP Interface 1# ../gw 1                      (Select default gateway #1)
>> Default gateway 1# addr 200.200.200.101     (Assign IP address for the gateway)
>> Default gateway 1# ena                       (Enable default gateway #1)
```

4. **Configure the local DNS server to recognize the local GSLB switch as the authoritative name server for the hosted services.**

   Determine the domain name which will be distributed to both sites, and the hostname for each distributed service. In this example, the California DNS server is configured to recognize 200.200.200.100 (the IP interface of the California GSLB switch) as the authoritative name server for www.foocorp.com.

Part Two: Configure the California Switch for Standard SLB

1. **Assign an IP address to each of the real servers in the local California server pool.**

   The real servers in any real server group must have an IP route to the switch that will perform the Server Load Balancing functions. This is most easily accomplished by placing the switches and servers on the same IP subnet, although advanced routing techniques can be used as long as they do not violate the topology rules outlined in "Network Topology Considerations" on page 22.

   For this example, the web-host real servers have IP addresses on the same IP subnet:

   Table 5-1 GSLB Example: California Real Server IP Addresses

Real Server	IP address
Server A	200.200.200.2
Server B	200.200.200.3

2. **On the California switch, define each local Real Server.**

   For each local real server, you must assign a real server number, specify its actual IP address, and enable the real server. For example:

```
>> Default gateway 1# /cfg/slb/real 1          (Server A is real server 1)
>> Real server 1 # rip 200.200.200.2          (Assign Server A IP address)
>> Real server 1 # ena                         (Enable real server 1)
>> Real server 1 # ../real 2                   (Server B is real server 2)
>> Real server 2 # rip 200.200.200.3          (Assign Server B IP address)
>> Real server 2 # ena                         (Enable real server 2)
```

3. **On the California switch, define a Real Server Group.**

   This combines the real servers into one service group, and sets the necessary health checking parameters. In this example, HTTP health checking is used to ensure that web content is being served. If the index.html file is not accessible on a real server during health checks, the real server will be marked as down.

   The following commands are entered:

```
>> Real server 2 # /cfg/slb/group 1            (Select real server group 1)
>> Real server group 1# add 1                  (Add real server 1 to group 1)
>> Real server group 1# add 2                  (Add real server 2 to group 1)
>> Real server group 1# health http            (Use HTTP for health checks)
>> Real server group 1# content index.html     (Set URL content for health checks)
```

4. **On the California switch, define a Virtual Server.**

   All client requests will be addressed to a virtual IP on a virtual server defined on the switch. Clients acquire the virtual IP through normal DNS resolution. HTTP uses well-known TCP port 80. In this example, HTTP is configured as the only service running on this virtual IP, and is associated with our real server group. For example:

```
>> Real server group 1 # /cfg/slb/virt 1     (Select virtual server 1)
>> Virtual server 1# vip 200.200.200.1       (Assign a virtual server IP address)
>> Virtual Server 1# service 80
>> Virtual server 1 http Service#  group 1   (Associate virtual port to real group)
>> Virtual server 1 http Service# ../ena     (Enable the virtual server)
```

NOTE – This configuration is not limited to HTTP web service. Other TCP/IP services can be configured in a similar fashion. For a list of other well-known services and ports, see the command option information in the "Configuration" chapter of the *WebOS Command Reference*.

5. **On the California switch, define the type of L4 traffic processing each port must support.**

   In this example, the following ports are being used on the ACEswitch 180:

   Table 5-2 GSLB Example: California ACEswitch 180 Port Usage

Port	Host	Layer 4 Processing
1	Server A	Server
2	Server B	Server
6	Default Gateway Router. This connects the switch to the Internet where all client requests originate.	Client

   The ports are configured as follows:

```
>> Virtual server 1# /cfg/slb/port 1      (Select physical switch port 1)
>> SLB port 1# server ena                 (Enable server processing on port 1)
>> SLB port 1# ../port 2                  (Select physical switch port 2)
>> SLB port 2# server ena                 (Enable server processing on port 2)
>> SLB port 2# ../port 6                  (Select physical switch port 6)
>> SLB port 6# client ena                 (Enable client processing on port 6)
```

6. **On the California switch, enable Server Load Balancing.**

```
>> SLB port 6# ..                         (Select the SLB Menu)
>> Server Load Balancing# on              (Turn Server Load Balancing on)
```

Part Three: Configure the California Site for GSLB

1. **On the California switch, define each remote site.**

   Add and enable the IP address for the IP interface of up to 64 remote sites. In this example, there is only one remote site: Denver, with an IP interface address of 174.14.70.100. The following commands are used:

```
>> Server Load Balancing# gslb/site 1          (Select Remote Site #1)
>> Remote site 1# prima 174.14.70.100         (Define remote interface)
>> Remote site 1# ena                         (Enable remote site #1)
```

   Each additional remote site would be configured in the same manner.

2. **On the California switch, assign each remote distributed service to a local virtual server.**

NOTE – This step can result in improper configuration if not clearly understood. Please take care to note where each configured value originates.

   In this step, we are configuring the local California site to recognize the services offered at the remote Denver site. To do this, configure one real server entry on the California switch for each virtual server located at each remote site. Since there's only one remote site (Denver) with only one virtual server, only one more local real server entry is needed at the California site.

   The new real server entry will be configured with the IP address of the remote virtual server, rather than the usual IP address of a local physical server.

   Also, the "remote" property will be enabled, and the real server entry will be added to the real server group under the local virtual server for the intended service. Finally, since the real server health checks will be headed across the Internet, the health checking interval should be increased to 30 or 60 seconds to avoid generating excess traffic. For example:

```
>> Remote site 1# /cfg/slb/real 3          (Create an entry for real server #3)
>> Real server 3# rip 174.14.70.1          (Set remote virtual server IP address)
>> Real server 3# remote enable            (Define the real server as remote)
>> Real server 3# inter 60                 (Set a high health check interval)
>> Real server 3# ena                      (Enable the real server entry)
>> Real server 3# ../group 1               (Select appropriate real server group)
>> Real server group 1# add 3              (Add real server 3 to the group 1)
```

NOTE – The IP address of the real server being added is taken from the virtual server IP address on the remote switch. Do not confuse this value with the IP interface address on the remote switch.

---

3. **On the California switch, define the domain name and hostname for each service hosted on each virtual server.**

   In this example, the domain name for the Foo Corporation is “foocorp.com,” and the hostname for the only service (HTTP) is “www.” These values are configured as follows:

```
>> Real server group 1# /cfg/slb/virt 1     (Select virtual server #1)
>> Virtual server 1# dname foocorp.com      (Define domain name)
>> Virtual server 1# service 80 hname http www(Define HTTP hostname)
```

   If other services were defined (such as FTP), additional hostname entries would be made.

4. **On the California switch, turn on Global Server Load Balancing.**

```
>> Virtual server 1# ../gslb                (Select the GSLB Menu)
>> Global SLB# on                           (Activate GSLB for the switch)
```

5. **Apply and verify the configuration.**

```
>> Global SLB# apply                        (Make your changes active)
>> Global SLB# cur                          (View current GSLB settings)
>> Global SLB# ../cur                       (View current SLB settings)
```

   Examine the resulting information. If any settings are incorrect, make and apply any appropriate changes, and then check again.

6. **Save your new configuration changes.**

```
>> Layer 4# save                            (Save for restore after reboot)
```

Part Four: Configure the Denver Site with Basic System Items

Following the same procedures as above, configuration the Denver site as follows.

1. **If the WebOS web-based interface (WBI) is to be used for managing the Denver switch, change its service port.**

```
>> Main# /cfg/sys                    (Select the System Menu)
>> System# wport 8080                (Set service port 8080 for WBI)
```

2. **On the Denver switch, define an IP interface.**

```
>> Main# /cfg/ip/if 1                (Select IP interface #1)
>> IP Interface 1# addr 174.14.70.100   (Assign IP address for the interface)
>> IP Interface 1# ena               (Enable IP interface #1)
```

3. **On the Denver switch, define the default gateway.**

```
>> IP Interface 1# ../gw 1           (Select default gateway #1)
>> Default gateway 1# addr 174.14.70.101   (Assign IP address for the gateway)
>> Default gateway 1# ena            (Enable default gateway #1)
```

4. **Configure the local DNS server to recognize the local GSLB switch as the authoritative name server for the hosted services.**

   The Denver DNS server is configured to recognize 174.14.70.100 (the IP interface of the Denver GSLB switch) as the authoritative name server for www.foocorp.com).

Part Five: Configure the Denver Switch for Standard SLB

1. **Assign an IP address to each of the real servers in the local Denver server pool.**

Table 5-3 Denver Real Server IP Addresses

Real Server	IP address
Server C	179.14.70.2
Server D	179.14.70.2

2. **On the Denver switch, define each local Real Server.**

```
>> Default gateway 1# /cfg/slb/real 1          (Server C is real server 1)
>> Real server 1 # rip 179.14.70.2             (Assign Server C IP address)
>> Real server 1 # ena                         (Enable real server 1)
>> Real server 1 # ../real 2                   (Server D is real server 2)
>> Real server 2 # rip 179.14.70.3             (Assign Server D IP address)
>> Real server 2 # ena                         (Enable real server 2)
```

3. **On the Denver switch, define a Real Server Group.**

```
>> Real server 2 # /cfg/slb/group 1            (Select real server group 1)
>> Real server group 1# add 1                  (Add real server 1 to group 1)
>> Real server group 1# add 2                  (Add real server 2 to group 1)
>> Real server group 1# health http            (Use HTTP for health checks)
>> Real server group 1# content index.html     (Set URL content for health checks)
```

4. **On the Denver switch, define a Virtual Server.**

```
>> Real server group 1 # /cfg/slb/virt 1       (Select virtual server 1)
>> Virtual server 1# vip 179.14.70.1            (Assign IP address)
>> Virtual server 1# service http              (Select the HTTP service menu)
>> Virtual server 1 http Service# group 1      (Associate virtual port to real group)
>> Virtual server 1 http Service# ../ena       (Enable the virtual server)
```

5. **On the Denver switch, define the type of L4 traffic processing each port must support.**

   In this example, the following ports are being used on the ACEswitch 180:

   Table 5-4 Web Host Example: ACEswitch 180 Port Usage

Port	Host	Layer 4 Processing
3	Server C	Server
4	Server D	Server
5	Default Gateway Router. This connects the switch to the Internet where all client requests originate.	Client

   The ports are configured as follows:

```
>> Virtual server 1# /cfg/slb/port 3          (Select physical switch port 3)
>> SLB port 3# server ena                     (Enable server processing on port 3)
>> SLB port 3# ../port 4                      (Select physical switch port 4)
>> SLB port 4# server ena                     (Enable server processing on port 4)
>> SLB port 4# ../port 5                      (Select physical switch port 5)
>> SLB port 5# client ena                     (Enable client processing on port 5)
```

6. **On the Denver switch, enable Server Load Balancing.**

```
>> SLB port 5# ..                             (Select the SLB Menu)
>> Server Load Balancing# on                  (Turn Server Load Balancing on)
```

Part Six: Configure the Denver Site for GSLB

Following the same procedures as above, here is a summary of the configuration steps for the Denver site.

1. **On the Denver switch, define each remote site.**

   Since we are now configuring the Denver site, Denver is local and California is remote. Add and enable the IP address for the IP interface of up to eight remote sites. In this example, there is only one remote site: California, with an IP interface address of 200.200.200.100. The following commands are used:

```
>> Server Load Balancing# gslb/site 1          (Select Remote Site #1)
>> Remote site 1# prima 200.200.200.100       (Define remote IP interface address)
>> Remote site 1# ena                         (Enable remote site #1)
```

2. **On the Denver switch, assign each remote distributed service to a local virtual server.**

NOTE – This step can result in improper configuration if not clearly understood. Please take care to note where each configured value originates.

In this step, we are configuring the local Denver site to recognize the services offered at the remote California site. As before, configure one real server entry on the Denver switch for each virtual server located at each remote site. Since there's only one remote site (California) with only one virtual server, only one more local real server entry is needed at the Denver site.

The new real server entry will be configured with the IP address of the remote virtual server, rather than the usual IP address of a local physical server.

Also, the "remote" property will be enabled, and the real server entry will be added the real server group under the local virtual server for the intended service. Finally, since the real server health checks will be headed across the Internet, the health checking interval should be increased to 30 or 60 seconds to avoid generating excess traffic. For example:

```
>> Remote site 1# /cfg/slb/real 3             (Create an entry for real server #3)
>> Real server 3# rip 200.200.200.1           (Set remote virtual server IP address)
>> Real server 3# remote enable               (Define the real server as remote)
>> Real server 3# inter 60                    (Set a high health check interval)
>> Real server 3# ena                         (Enable the real server entry)
>> Real server 3# ../group 1                  (Select the approp. real server group)
>> Real server group 1# add 3                 (Add real server 3 to the group 1)
```

NOTE – The IP address of the real server being added is taken from the virtual server IP address on the remote switch. Do not confuse this value with the IP interface address on the remote switch.

---

3. **On the Denver switch, define the domain name and hostname for each service hosted on each virtual server.**

   These will be the same as for the California switch: the domain name is “foocorp.com,” and the hostname for the HTTP service is “www.” These values are configured as follows:

```
>> Real server group 1# ../virt 1              (Select virtual server #1)
>> Virtual server 1# dname foocorp.com         (Define domain name)
>> Virtual server 1# service 80 hname http www(Define HTTP hostname)
```

4. **On the Denver switch, turn on Global Server Load Balancing.**

```
>> Virtual server 1# ../gslb/on                (Activate GSLB for the switch)
```

5. **Apply and verify the configuration.**

```
>> Global SLB# apply                           (Make your changes active)
>> Global SLB# cur                             (View current GSLB settings)
>> Global SLB# ../cur                          (View current SLB settings)
```

   Examine the resulting information. If any settings are incorrect, make and apply any appropriate changes, and then check again.

6. **Save your new configuration changes.**

```
>> Layer 4# save                               (Save for restore after reboot)
```

IP Proxy Addresses for Non-HTTP Application Redirects

Alteon WebSystems switches with WebOS software installed can configure GSLB remote servers to have any user request sent to them using a load balancing mechanism called *IP Proxy*.

NOTE – This feature should be used as a method of last resort for GSLB implementations, in topologies where the remote servers are usually virtual IP addresses in other Alteon WebSystems switches.

How IP Proxy Works

Example: The figure below shows two GSLB sites, each with one local virtual server (VIP 1) serviced by two real servers in Real Server Group 1. The applications being load balanced are HTTP and POP3. The network administrator wants to have any request that cannot be serviced locally to be sent to the peer site. HTTP requests will be sent to the peer site using HTTP Redirect. Any other application request will be sent to the peer site using the IP Proxy feature.

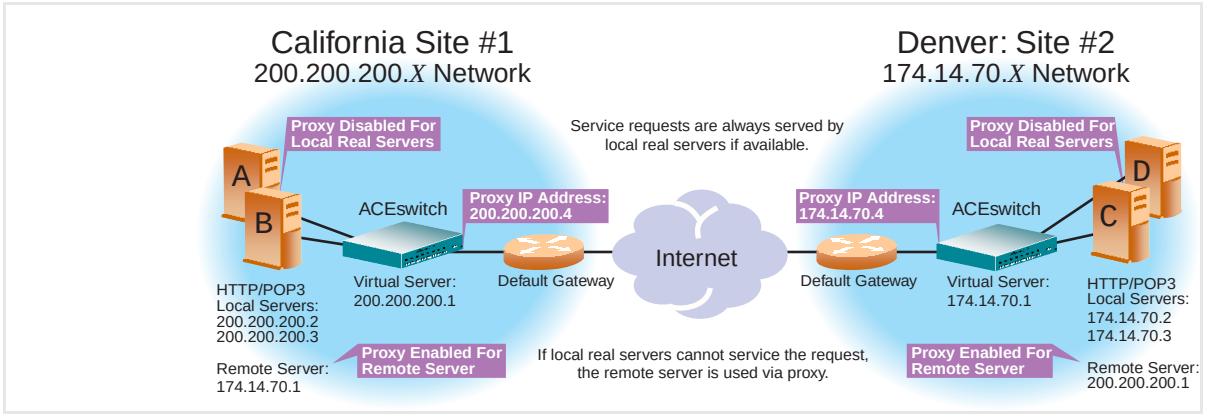


Figure 5-3 POP3 Request fulfilled via IP Proxy

When the POP3 processes at Site #1 terminate due to operator error, the following events occur to allow users' POP3 requests to be fulfilled:

1. **A user POP3 TCP SYN request is received by the virtual server at Site #1. The switch at that site determines that there are no local resources to handle the request.**

2. **The switch rewrites the request, such that it now contains a proxy IP address as the IP source address (IPSA), and the virtual server IP address at Site #2 as the IP destination address (IPDA).**

3. **The switch at Site #2 receives the TCP SYN (POP3) request to its virtual server that looks like a normal SYN frame, and thus, performs normal local load balancing mechanisms.**

4. **The TCP SYN ACK coming from Site #2's local real server IP address is sent back toward the IP address specified by the proxy IP address.**

5. **The switch at Site #2 sends the TCP SYN ACK frame towards Site #1, with Site #2's virtual server IP address as the IP source address and site #1's proxy IP address as the IP destination address.**

6. **The switch at Site #1 receives the frame and translates it, using Site #1's virtual server IP address as the IP source address and the client's IP address as the IP destination address.**

   This cycle continues for the remaining frames that are necessary to transmit the client's mail, until a FIN frame is received.

Configuring IP Proxy

In keeping with the previous example, starting on page 94, the switch at Site #1 in California is configured with switch port 6 connecting to the default gateway, and real server 3 representing the remote server in Denver. The following commands are used to configure the IP Proxy on Site #1 in California:

```
>> # /cfg/slb/port 6                          (Select port to default gateway)
>> SLB port 6# pip 200.200.200.4              (Set unique proxy IP address)
>> SLB port 6# proxy enable                   (Enable proxy for switch port 6)
>> SLB port 6# ../real 1/proxy disable        (Disable proxy for local real server)
>> Real server 1 # ../real 2/proxy disable    (Disable proxy for local real server)
>> Real server 2 # ../real 3/proxy enable     (Enable proxy for remote server)
>> Real server 3 # apply                      (Apply configuration changes)
>> Real server 3 # save                       (Save configuration changes)
```

If you want to configure IP Proxy on Site #2, the following commands are issued on the Denver switch:

```
>> # /cfg/slb/port 5                          (Select port to default gateway)
>> SLB port 5# pip 174.14.17.4                (Set unique proxy IP address)
>> SLB port 5# proxy enable                   (Enable proxy for switch port 5)
>> SLB port 5# ../real 1/proxy disable        (Disable proxy for local real server)
>> Real server 1 # ../real 2/proxy disable    (Disable proxy for local real server)
>> Real server 2 # ../real 3/proxy enable     (Enable proxy for remote server)
>> Real server 3 # apply                      (Apply configuration changes)
>> Real server 3 # save                       (Save configuration changes)
```

Basic Tests for GSLB Operation

- Execute browser request to the configured service (www.foocorp.com in the example above).
- On each switch, examine the `/info/slb` information.
- Check that all SLB parameters are working according to expectation. If necessary, make any appropriate configuration changes and then check the information again.
- On each switch, examine the following statistics:
  - `/stats/slb/gslb/virt` *<virtual server number>*
  - `/stats/slb/gslb/group` *<real server group number>*
  - `/stats/slb/maint`

GSLB Client Proximity Tables

In certain customer configurations, IANA data does not provide sufficient geographic separation of proximity information. As a result, large ISP partners cannot use their own geographic data to determine GSLB site selection based on client location. WebOS software supports client proximity tables using static “client to site” mapping. Switch managers can configure private client proximity information. The limit on the number of entries in the proximity database is 128.

The use of a static client/site database allows customizing for the user environment.

Configurable Source Network <--> Site Preference Tables,

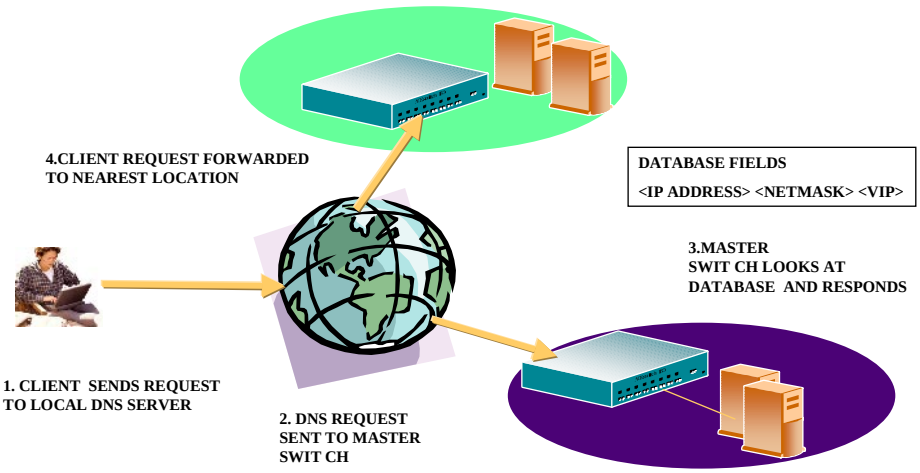


Figure 5-4 GSLB Proximity Tables: How It Works

GSLB Proximity Configuration Example

205.178.13.* prefers Site 3, Site 1

204.165.*.* prefers Site 4, Site 2

Here are the commands to configure this scenario:

```
>> #/cfg/slb/gslb/lookup/lookups ena
>> #/cfg/slb/gslb/lookup/dname alteon com
>> #/cfg/slb/gslb/lookup/network 1
                    sip 205.178.13.0
                    mask 255.255.255.0
                    vip1 IP addr of Site 3
                    vip2 IP addr of Site 1
>> #/cfg/slb/gslb/lookup/network 2
                    sip 204.165.0.0
                    mask 255.255.255.0
                    vip1 IP addr of Site 4
                    vip2 IP addr of Site 2
```

Using this configuration, the DNS request “alteon.com” from 205.178.13.* will get a DNS response with only one virtual IP address, for example, Site 1, if Site 1 has less load than Site 3.

CHAPTER 6

Content Intelligent Switching

The following table lists the primary topics described in this chapter and the page number where you'll find information about each feature.

Functionality	Features/Description	See
Content-Intelligent Server Load Balancing	URL Parsing/URL-Based WCR	page 112
	URL-Based Server Load Balancing	page 119
	Virtual Hosting	page 130
	Parsing based on browser type	page 133
	URL Hashing	page 137
	Preferential Treatment, based on Cookies	page 134
Content-Intelligent Web Cache Redirection	Cachability based on domain name	page 126
	Redirection based on domain name	

Content Switching Overview

Working with session content is much more demanding than examining TCP/IP protocol headers, for the following reasons:

- Content is non-deterministic. Content identifiers such as URLs and cookies can be of varying lengths and can appear at unpredictable locations within a request. Scanning session traffic for a specific string is far more processor-intensive than looking at a known location in a session for a specific number of bytes.
- Parsing content requests means temporarily terminating the TCP connection from a client. In other words, the Web switch must first pretend that it is the server, ask the client what it wants, examine the request, and then open a connection to an appropriate server. While this is happening, the Web switch must temporarily buffer the request, which consumes system memory. This temporary termination is called a *delayed binding*.
- With delayed binding, two independent TCP connections span a Web session: one from the client to the Web switch and the second from the Web switch to the selected server. The Web switch must modify the TCP header, including performing TCP sequence number translation and recalculating checksums on every packet that travels between the client and the server, for the duration of the session. This function, known as "TCP connection splicing," heavily tasks a Web switch, particularly when the switch must process thousands of these sessions simultaneously.

In addition to real-time traffic and connection processing, a content switch needs to monitor the servers to ensure that requests are forwarded to the best performing and healthy servers. This monitoring involves more than simple ICMP or TCP connection tests as servers continue to process network protocols while failing to retrieve any content. Furthermore, if content is segregated in different servers or server farms, the Web switch must provide a flexible, user-customizable mechanism allowing a relevant set of application and content tests to be applied to each server or server farm.

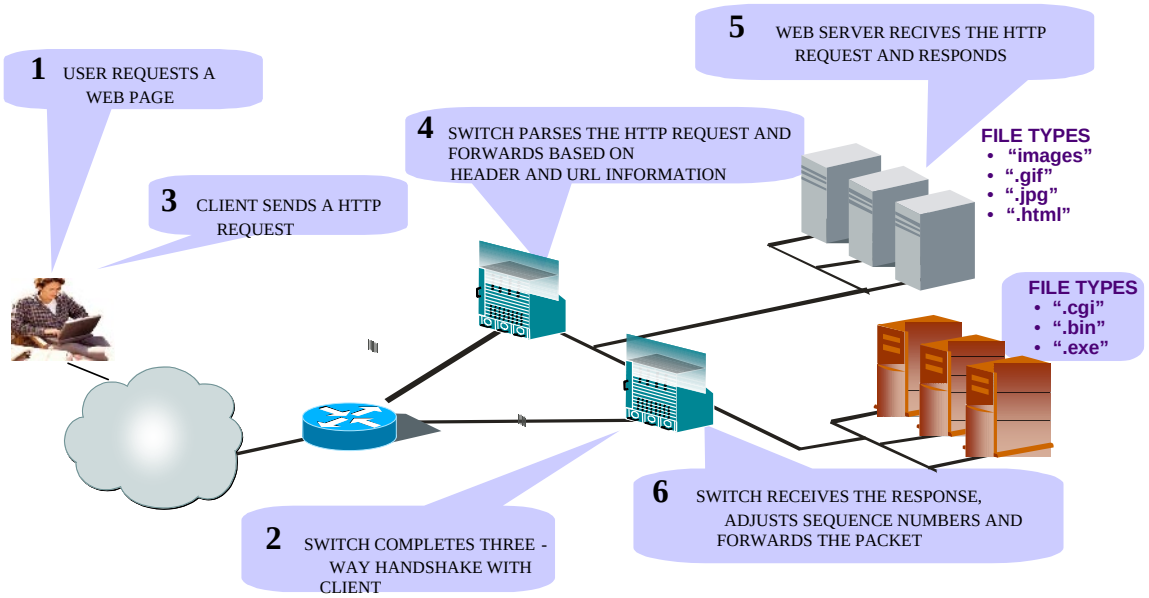


Figure 6-1 Content-Aware Load Balancing Example

To fulfill the requirements described above, a Web switch needs to perform numerous processing tasks for each incoming session, including connection setup, traffic parsing, applying server selection algorithms, splicing connections and translating session addresses, metering and controlling server bandwidth usage, processing traffic filters, collecting statistics, and so on. Not only are these functions CPU-intensive, they are executed whenever a new request arrives. In addition, the switch must also perform background functions, such as updating network topology, health-checking servers, applications and server sites, measuring server performance, and so on, on a periodic basis.

URL-Based Web Cache Redirection

By separating static and dynamic content requests via URL parsing, WebOS 8.0 enables you to send requests with specific URLs or URL sub-strings to designated cache servers. The URL-based redirection option allows you to perform cache server farm tuning and offload overhead processing from the cache servers by only sending appropriate requests to the cache server farm.

NOTE – Both HTTP 1.0 and HTTP 1.1 requests are supported.

Each request is examined and handled as described below:

- If the request is a non-GET request like HEAD, POST, PUT, or HTTP with cookies, it doesn't get sent to the cache.
- If the request is an ASP or CGI request, or a dynamically generated page, it doesn't get sent to the cache.
- If the request is a cookie, it can optionally bypass the cache.

Network administrators can configure up to 32 URL expressions, each 8 bytes long, for non-cacheable content types. Up to 128 strings (on the A180e Web switch), comprising 40 bytes each, can be used for URL sub-string matching. As each URL web request is examined, non-cacheable items are forwarded to the origin server and requests with sub-string matches are redirected to the appropriate cache server.

NOTE – The term "origin server" refers to the server originally specified in the request.

Examples of sub-strings are

- "/product" - matches URL that starts with "/product," including any information in the "/product" directory
- "product" - matches URL that has the string "product" anywhere in the entire URL

The switch is pre-configured with a list of thirteen non-cacheable items that the network administrator can add, delete, or modify via the user interface. These items are either known dynamic content file extensions or dynamic URL parameters, as described below:

- dynamic content file extensions: cgi (cgi files)
- cfm (cold fusion files), .asp (ASP files), bin (bin directory), cgi-bin (cgi-bin directory), shtml (scripted html), .htx (Microsoft HTML extension file), .exe (executable)
- dynamic URL parameters: +, !, %, =, &

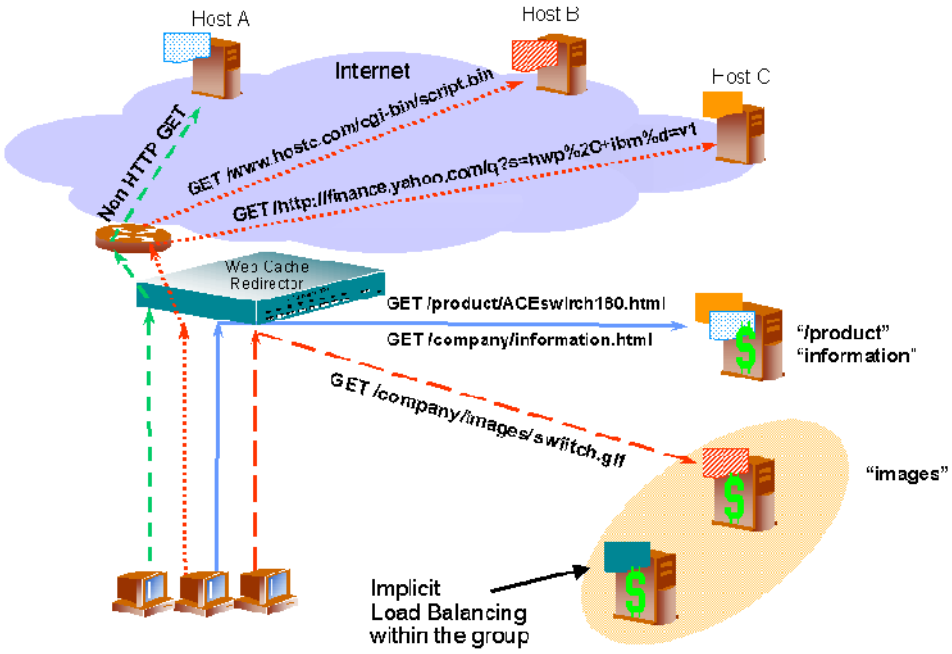


Figure 6-2 URL-Based Web Cache Redirection

Requests will be load balanced among the multiple servers matching the URL according to the metric specified for the server group (`leastConns` is the default).

Configuring URL-Based Web Cache Redirection

To configure URL-based web cache redirection, perform the following steps:

1. **Before you can configure URL-based web cache redirection, configure the switch for basic server load balancing. This includes the following tasks:**

   - Assign an IP address to each of the real servers in the server pool.
   - Define an IP interface on the switch.
   - Define each real server.

   For information on how to configure your network for server load balancing, see Chapter 2.

2. **Configure the switch to support basic web cache redirection.**

   For information on web cache redirection, refer to Chapter 4 of the *WebOS Application Guide*.

3. **Configure the parameters and file extensions that will bypass web cache redirection.**

   The switch is pre-configured with a list of thirteen non-cacheable items. These items are either known dynamic content file extensions or dynamic URL parameters:

   - dynamic content file extensions: cgi (cgi files), .cfm (cold fusion files), .asp (ASP files), bin (bin directory), cgi-bin (cgi-bin directory), .shtml (scripted html), .htx (Microsoft HTML extension file), .exe (executable)
   - dynamic URL parameters: +, !, %, =, &

   a) Add or remove expressions that should <u>not</u> be cacheable.

   ```
   >> cfg/slb/url# redir/add|remove <expression>
   ```

   b) Enable/disable ALLOW for none GETS (e.g., HEAD, POST, PUT) to origin server, as described below.

   ```
   >> cfg/slb/url# redir/urlal ena|dis
   ```

   - **`Enable`**: Switch will redirect all non-GET requests to the origin server.
   - **`Disable`**: Switch will compare all requests against the expression table to determine whether the request should be redirected to a cache server or the origin server.

c) Enable/disable cache redirection of requests that contain "`cookie:`" in the HTTP header.

```
>> cfg/slb/url# redir/cooki ena|dis
```

- **`Enable`**: Switch will redirect all requests that contain "`cookie:`" in the HTTP header to the origin server.
- **`Disable`**: Switch will compare the URL against the expression table to determine whether the request should be redirected to a cache server or the origin server.

d) Enable/disable cache redirection of requests that contain "`Cache-control:no cache`" in the HTTP 1.1 header or "`Pragma:no cache`" in the HTTP 1.0 header to the origin server.

```
>> cfg/slb/url# redir/nocache ena|dis
```

- **`Enable`**: Switch will redirect all requests that contain "Cache-control: no cache" in the HTTP 1.1 header or "Pragma:no cache" in the HTTP 1.0 header to the origin server.
- **`Disable`**: Switch will compare the URL against the expression table to determine whether the request should be redirected to a cache server or the origin server.

4. **Define the string(s) to be used for web cache server load balancing.**
Refer to the parameters listed below:

```
>> cfg/slb/url# lb/add|rem <string>
```

- **`add`**: Add string or a path.
- **`rem`**: Remove string or a path.

A default string "`any`" indicates that the particular server can handle all URL or web cache requests. A string that starts out with a backslash (`/`) such as "`/images`" indicates that, if this string is applied to a particular server, that server can only handle requests that start out with the "`/images`" string.

Example: With the "`/images`" string, the server will handle these requests:

```
/images/product/b.gif
/images/company/a.gif
/images/testing/c.jpg
```

This server will not handle these requests:

```
/company/images/b.gif
/product/images/c.gif
/testing/images/a.gif
```

A string that doesn't start out with a backslash (/) indicates that, if this string is applied to a particular server, that server can handle any requests that contain the defined string.

Example: With the "`images`" string, the server will handle these requests:

```
/images/product/b.gif
/images/company/a.gif
/images/testing/c.jpg
/company/images/b.gif
/product/images/c.gif
/testing/images/a.gif
```

If a server is configured only with the load balance string (/), it will only handle requests to the ROOT directory.

Example: With the "(/)" string, the server will handle these requests:

```
/
/index.htm
/default.asp
/index.shtm
Any files in the ROOT directory
```

For easy configuration and identification, each defined string has an ID attached, as shown in the following example:

Example: Number of entries: 6

ID	SLB String
1	any
2	.gif
3	/sales
4	/xitami
5	/manual
6	.jpg

5. **Configure the real server(s) to handle web cache redirection.**

NOTE – If you don't add a defined sub-string (or add the defined sub-string "any"), the server will handle any request.

---

a) To add a defined sub-string:

```
>> /cfg/slb/real 2/addlb <ID>
```

Where *ID* is the identification number of the defined string.

b) To remove a defined sub-string:

```
>> /cfg/slb/real 2/remlb <ID>
```

The server can have multiple defined sub-strings:

- "/images"
- "/sales"
- ".gif"

With these defined strings, this particular server can handle requests that start out with "/images" or "/sales" and any requests that contain ".gif"

6. **On the switch, define a real server group and add real servers to the real server group.**

This combines the three real servers into one server group:

```
>> Main# /cfg/slb/group 1                  (Select real server group 1)
>> Real server group 1# add 1              (Add real server 1 to group 1)
>> Real server group 1# add 2              (Add real server 2 to group 1)
>> Real server group 1# add 3              (Add real server 3 to group 1)
```

7. **Configure a filter to support basic web cache redirection.**

For instructions on how to configure filters for web-cache redirection, see "Example Configuration for the Web-Cache Solution" in Chapter 16, "Filtering," of the *WebOS 5.2 User's Guide*.

8. **Enable URL-based web cache redirection on the same filter.**

```
>> /cfg/slb/filt <filter-number>/adv/urlp ena
```

9. **On the switch, create a default filter for non-cached traffic on the client port.**

```
>> /cfg/slb/filt <filter-number> ena
```

10. **Turn on the filter to the port and add filters to the port.**

```
>> /cfg/slb/port <port-number>/filt ena
```

11. **Enable Direct Access Mode on the switch OR disable DAM and configure a proxy IP address (PIP) on the port.**

- To turn on Direct Access Mode:

```
>> /cfg/slb/adv/direct ena
```

- To turn off Direct Access Mode and configure a proxy IP (PIP) for the port:

```
>> /cfg/slb/adv/direct dis
>> /cfg/slb/port 2/pip 12.12.12.12
>> /cfg/slb/port 2/proxy ena
```

12. **On the switch, enable, apply, and verify the configuration.**

```
>> SLB port 6# ..                          (Select the SLB Menu)
>> Server Load Balancing# on               (Turn Server Load Balancing on)
>> Server Load Balancing# apply            (Make your changes active)
>> Server Load Balancing# cur              (View current settings)
```

Statistics for URL-Based WCR

To show the number of hits to the cache server or origin server, use this command:

```
>> Main#>> /stats/slb/url/redir
```

Sample Statistics:

```
Total URL based web cache redirection stats:
Total cache server hits:                  73942
Total origin server hits:                 2244
Total none-GETs hits:                     53467
Total 'Cookie: ' hits:                    729
Total no-cache hits:                      43
```

URL-Based Server Load Balancing

URL-based server load balancing allows the network administrator to optimize resource access and server tuning. Content dispersion can be optimized by basing load balancing decisions on the entire path/filename of each URL.

URL-based load balancing operates in a manner similar to URL parsing for web cache redirection, except that the switch virtual IP address (VIP) is the target of all IP/HTTP requests.

NOTE – Both HTTP 1.0 and HTTP 1.1 requests are supported.

Network administrators can configure up to 128 strings, comprising 40 bytes each, for URL matching. Each URL web request is then examined against the URL strings defined for each real server, as described under "URL-Based Web Cache Redirection" on page 6-112. URL requests will be load balanced among the multiple servers matching the URL, according to the metric specified in the server group (`leastConns` is the default).

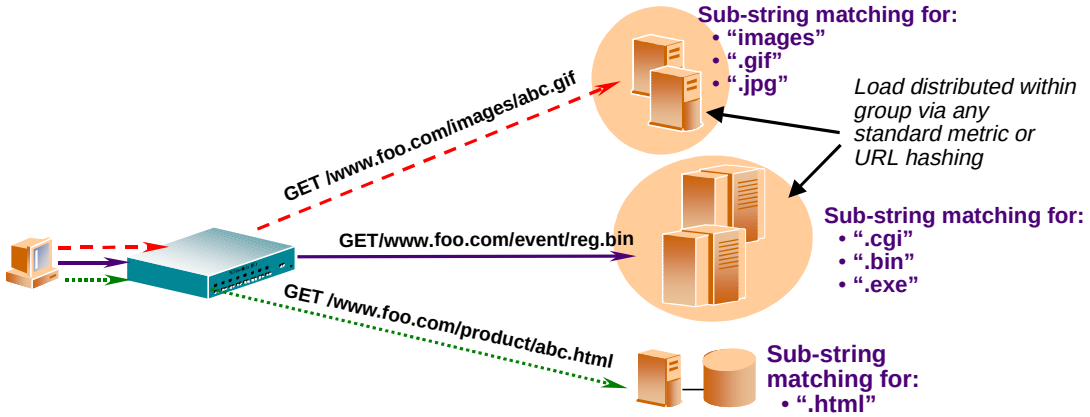


Figure 6-3 URL-Based Server Load Balancing

Example: The network administrator specifies the following criteria for load balancing:

- Requests with ".cgi" in the URL: forwarded to servers RIP1, RIP2, RIP5.
- Requests with the sub-string "images" in the URL: sent to servers RIP3, RIP4 and RIP6.
- Requests with URLs starting with the sub-string "/product:" sent to servers RIP2, RIP3 and RIP5.
- Requests containing URLs with anything else: sent to servers RIP1, RIP2, RIP3. These servers have been defined with the "any" string.

Configuring URL-Based Server Load Balancing

NOTE – When URL based SLB is used in an active/active redundant setup, use a proxy IP address (PIP) instead of Direct Access Mode (DAM) to enable the URL parsing feature.

To configure URL-based server load balancing, perform the following steps:

1. **Before you can configure URL-based load balancing, ensure that the switch has already been configured for basic server load balancing. This includes the following tasks:**
   - Assign an IP address to each of the real servers in the server pool.
   - Define an IP interface on the switch.
   - Define each real server.
   - Define a real server group and set up health checks for the group.
   - Define a virtual server on virtual port 80 (HTTP) and assign a real server group to service it.
   - Enable server load balancing on the switch.
   - Enable client processing on the port connected to the client.

   For information on how to configure your network for server load balancing, see Chapter 2.

2. **Define the string(s) to be used for URL load balancing. Refer to the information and examples given below:**

```
>> cfg/slb/url# lb/add|rem <string>
```

   - **`add`**: Add string or a path.
   - **`rem`**: Remove string or a path.

   A default string "`any`" indicates that the particular server can handle all URL or web cache requests. A string that starts out with a backslash (`/`) such as "`/images`" indicates that, if this string is applied to a particular server, that server can only handle requests that start out with the "`/images`" string.

Example: With the "`/images`" string, the server will handle these requests:

```
/images/product/b.gif
/images/company/a.gif
/images/testing/c.jpg
```

This server will not handle these requests:

```
/company/images/b.gif
/product/images/c.gif
/testing/images/a.gif
```

A string that doesn't start out with a backslash (/) indicates that, if this string is applied to a particular server, that server can handle any requests that contain the defined string.

Example: With the "`images`" string, the server will handle these requests:

```
/images/product/b.gif
/images/company/a.gif
/images/testing/c.jpg
/company/images/b.gif
/product/images/c.gif
/testing/images/a.gif
```

If a server is configured only with the load balance string (/), it will only handle requests to the ROOT directory.

Example: With the "(/)" string, the server will handle these requests:

```
/
/index.htm
/default.asp
/index.shtm
Any files in the ROOT directory
```

For easy configuration and identification, each defined string has an ID attached, as shown in the following example:

Example: Number of entries: 6

ID	SLB String
1	any
2	.gif
3	/sales
4	/xitami
5	/manual
6	.jpg

3. **Configure one or more real servers to handle URL-based load balancing.**

   a) To add a defined sub-string, use this command:

```
>> /cfg/slb/real 2/addlb ID
```

   Where *ID* is the identification number of the defined string.

NOTE – If you don't add a defined sub-string (or add the defined sub-string "`any`"), the server will handle any request.

---

   b) To remove a defined sub-string, use this command:

```
>> /cfg/slb/real 2/remlb ID
```

   The server can have multiple defined sub-strings:

   - "/images"
   - "/sales"
   - ".gif"

   With these defined strings, this particular server can handle requests that start out with "/images" or "/sales" and any requests that contain "`.gif`"

4. **On the switch, enable server load balancing.**

```
>> /cfg/slb# on                    (Turn Server Load Balancing on)
```

5. **Either enable Direct Access Mode on the switch or configure a proxy IP address (PIP) on the client port.**

   To use cookie-based preferential load balancing without DAM, you need to configure a proxy IP address on the client port.

NOTE – If Virtual Matrix Architecture (VMA) is enabled, you need to configure a proxy IP address on ports 1-8.

   On the port that you want to use for cookie-based preferential load balancing, you will enable proxy load balancing. If VMA is enabled on the switch, you can choose to configure the remaining ports with proxy IP disabled.

- To turn on Direct Access Mode:

```
>> /cfg/slb/adv/direct ena
```

- To turn off Direct Access Mode and configure a proxy IP address (PIP) on the client port:

```
>> /cfg/slb/direct dis
>> /cfg/slb# port 2/pip 12.12.12.12
>> /cfg/slb# port 2/proxy ena
```

NOTE – By enabling Direct Access Mode on the switch or, alternatively, disabling DAM and configuring a proxy IP address on the client port, port mapping for URL load balancing can be performed.

6. **Enable URL-based SLB on the virtual server(s).**

```
>> /cfg/slb/virt <virtual-server-number>/service 80/httpslb ena/urlslb
```

Statistics for URL-Based SLB

To show the number of hits to the SLB or cache server, use this command:

```
>> Main# stats/slb/url/lb
```

Sample Statistics:

ID	SLB String	Hits	ID	SLB String	Hits
1	any	73881	4	/xitami	162102
2	.gif	0	5	/manual	0
3	/sales	0	6	.jpg	0

HTTP Header Inspection

HTTP headers are used to include additional information to requests and responses. The HTTP 1.1 specification defines a total of 46 headers. HTTP headers can be general headers, request headers, response headers, and entity headers. General headers may exist in both requests and responses. Requests and response headers are specific to only requests and responses, respectively. Entity headers describe the content of the request body or the content of the response body.

Each HTTP header field consists of a name, followed immediately by a colon (":"), a single space character, and the field value. Field names are case-insensitive. Header fields can be extended over multiple lines by preceding each extra line with at least one space.

NOTE – One HTTP header is supported globally for the entire switch.

Customer applications of header inspection are listed below:

- Redirection based on domain name
- Cachability based on domain name
- Virtual hosting
- Redirection based on browser type
- Cookie-based preferential redirection

Multiple Frames Processing for Delayed Binding

In addition to the URI path, which generally is less than 300 bytes, the HTTP GET requests also include general headers and request headers. HTTP cookie request headers can be 4500 bytes in length. A single GET request can include multiple cookies.

To handle the overall length of HTTP headers, including request headers containing multiple cookies, and the Maximum Segment Size (MSS) of dial-up connections, WebOS software provides the following support:

- Parsing of HTTP GET requests for URI path matching and HTTP headers matching beyond the first frame while performing delayed binding.
- Buffering of a maximum of 4,500 bytes in total for a single GET request across multiple frames.
- Processing multiple frames from a single HTTP GET request, using a TCP stack on the Switch Processor.

HTTP Header-Based SLB

By configuring HTTP header server load balancing, you can load balance HTTP requests based on different HTTP header information, such as 'Cookie:' header for persistent load balancing, 'Host:' header for virtual hosting, and "User-Agent' for browser-smart load balancing.

NOTE – Cookie-based persistent load balancing is described in Chapter 7, "Persistence." Virtual hosting and browser-smart load balancing is discussed in this chapter.

No Cache/Cache Control for WCR

Offload cache servers from processing non-cacheable content by sending only appropriate requests to the cache server farm. When a Cache-Control header is present in a HTTP 1.1 request, it indicates a client's special request with respect to caching, such as to guarantee up-to-date data from the origin server. By enabling this feature, HTTP 1.1 GET requests with the **Cache-Control: no cache** directive in the requests are forwarded directly to the origin servers.

NOTE – For WCR, one HTTP header is supported globally for the entire switch.

In HTTP 1.0, the equivalent of the HTTP 1.1 Cache-Control: Header is the **Pragma:no-cache** headers. By enabling this feature, requests with the **Pragma: no-cache** headers are forwarded to the origin server. This allows a client to insist upon receiving an authoritative response to its requests.

Configuring HTTP Header-Based Web Cache Redirection

By configuring HTTP header WCR, we can redirect web cache requests based on different HTTP header information, such as 'Host:' header or 'User-Agent' for browser-smart load balancing.

To configure the switch to do web-cache redirection based on the 'Host:' header, perform the following procedure:

1. **Configure basic SLB.**

   Before you can configure header-based cache redirection, ensure that the switch has already been configured for basic server load balancing. SLB configuration includes the following tasks:

   - Assign an IP address to each of the real servers in the server pool
   - Define an IP interface on the switch
   - Define each real server
   - Assign servers to real server groups
   - Define virtual servers and services

   For information on how to configure your network for server load balancing, see Chapter 2.

2. **Turn on URL parsing for the filter.**

```
>> # /cfg/slb/filt 1/adv/urlp ena
```

3. **Enable header load balancing for 'Host:' header.**

```
>> # /cfg/slb/url/redir/header ena host
```

4. **Define the host names.**

```
>> # /cfg/slb/url/lb/add ".com"
>> Server Loadbalance Resource# add ".org"
>> Server Loadbalance Resource# add ".net"
```

5. **Configure the real server(s) to handle the appropriate load balance string(s).**

   To add a defined sub-string:

```
>> #/cfg/slb/real 2/addlb ID
```

   Where *ID* is the identification number of the defined string.

NOTE – If you don't add a defined sub-string (or add the defined sub-string "`any`"), the server will handle any request.

---

6. **If Host: header filtering is configured, you can configure the switch to use the host header field to determine whether requests are cacheable (or non-cacheable).**

   Example:

   If you want all domain names that end with `.net` or `.uk` NOT to go to a cache.

   a) Configure the Host header filter.

```
>> # /cfg/slb/filt 1/adv/urlp ena
```

   b) Add in expression entries:

```
>> # /cfg/slb/url/redir/add .net .uk
```

7. **You can direct the same cacheable URL request to the same cache server by configuring `minmisses` or `hash` as the metric. The switch will then use the host field in the HTTP header and the number of bytes into the URI to calculate the hash key.**

   If the host field doesn't exist and no length was specified, the switch will use the source IP address as the hash key. If host field doesn't exist, but length was specified, the switch will use all or part of the URI to calculate the hash key.

```
>> # /cfg/slb/url/redir/hash enable|disable
```

   - **Enable**: Enable hashing based on the URI and set the length of URI that will be used to hash into the cache server.
   - **Disable**: By disabling hashing based on the URI, the switch will only use the host header field to calculate the hash key.

   Example 1.Using the source IP address as the hash key:

   client1 requests http://www.yahoo.com --> cache1
   client2 requests http://www.yahoo.com --> cache2
   client3 requests http://www.yahoo.com --> cache3

   Example 2. Using the host field and/or part or all of the URI, the same URL request will go to the same cache:

   client1 requests http://www.yahoo.com --> cache1
   client2 requests http://www.yahoo.com --> cache1
   client3 requests http://www.yahoo.com --> cache1

   Example 3. If the Host field doesn't exist, but length was specified:

   client1 requests http://www.yahoo.com/sales/index.htm --> cache1
   client2 requests http://www.yahoo.com/sales/index.htm --> cache1
   client3 requests http://www.yahoo.com/sales/index.htm --> cache1

Configuring Browser-Based WCR

To configure User-Agent: header-based web-cache redirection, perform the following procedure.

1. **Before you can configure header-based cache redirection, ensure that the switch has already been configured for basic server load balancing. SLB configuration includes the following tasks:**
   - Assign an IP address to each of the real servers in the server pool.
   - Define an IP interface on the switch.
   - Define each real server.
   - Assign servers to real server groups.
   - Define virtual servers and services
2. **Turn on URL parsing for the filter.**

```
>> cfg/slb/filt 1/adv/urlp enable
```

3. **Enable header load balancing for "User-Agent:" header.**

```
>> # /cfg/slb/url/redir/header ena useragent
```

4. **Define the host names.**

```
>> # /cfg/slb/url/lb/add "Mozilla"
>> Server Loadbalance Resource# add "Internet Explorer"
>> Server Loadbalance Resource# add "Netscape"
```

5. **Configure the real server(s) to handle the appropriate load balance string(s).**

NOTE – If you don't add a defined sub-string (or add the defined sub-string "any"), the server will handle any request.

To add a defined sub-string:

```
>> /cfg/slb/real 2/addlb ID
```

Where *ID* is the identification number of the defined string.

Virtual Hosting

Increasingly, individuals and companies are interested in having a presence on the Internet in the form of a dedicated website address. They want, for example, to have a *www.site-a.com* and *www.site-b.com* instead of *www.hostsite.com/site-a* and *www.hostsite.com/site-b*.

Service providers, on the other hand, don't want to deplete the pool of unique IP addresses by dedicating an individual IP address for each home page they host. By supporting an extension in HTTP 1.0 to include the host header, WebOS 8.0 enables service providers to create a single VIP to host multiple websites per customer, each with their own hostname.

NOTE – For server load balancing, one HTTP header is supported per VIP.

The following bullets provide more detail, followed by configuration details.

- Currently, an HTTP 1.0 request sent to an origin server (NOT a proxy server) is a partial URL instead of a full URL.

  An example of the request that the origin server would see is:

  GET /product/ACE180/ HTTP/1.0
  User-agent: Mozilla/3.0
  Accept: text/html, image/gif, image/jpeg

  The GET request does not include the hostname. From the TCP/IP headers, the origin server knows its hostname, port number, and the protocol that it speaks.

- With the extension to HTTP/1.1 to include the HTTP HOST: header, the above request to retrieve the URL "/www.alteon.com/ product/ACE180" would look like the following:

  GET /product/ACE180/ HTTP/1.1
  Host: www.alteon.com
  User-agent: Mozilla/3.0
  Accept: text/html, image/gif, image/jpeg

  The Host: header carries the hostname used to yield the IP address of the site.

- Based on the Host: header, the switch will forward the request to servers representing different customers' websites.

- Network administrator needs to define a domain name as part of the 128 supported URL sub-strings.

- The switch will perform sub-string matching. That is, the sub-string "alteon.com" or "www.alteon.com" will match www.alteon.com.

Virtual Hosting Configuration Overview

The following is the sequence of events for configuring virtual hosting, based on HTTP Host: headers:

1. **Network administrator defines a domain name as part of the 128 supported URL sub-strings.**

   Both domain names “www.company-a.com” and “www.company-b.com” get resolved to the same IP address. In this example, the IP address is a virtual IP address (VIP) on the switch.

2. **“www.company-a.com” and “www.company-b.com” are defined as URL sub-strings.**

3. **Server Group 1 is configured with Servers #1 through #8.**

   Servers #1 through #4 belong to “www.company-a.com” and Servers #5 through 8 belong to “www.company-b.com.”

4. **Network administrator assigns sub-string “www.company-a.com” to Servers #1 through #4 and sub-string “www.company-b.com” to Servers #5 through #8.**

5. **Switch inspects the HTTP host header in requests received from the client.**

   - If the host header is “www.company-a.com,” the switch directs requests to one of the Servers #1 through #4.
   - If the host header is “www.company-b.com,” the switch directs requests to one of the Servers #5 through #8.

Configuring the “Host:” Header for Virtual Hosting

To configure ’Host:’ header load balancing to support virtual hosting, perform the following procedure:

1. **Before you can configure header-based load balancing, ensure that the switch has already been configured for basic server load balancing. SLB configuration includes the following tasks:**
   - Assign an IP address to each of the real servers in the server pool.
   - Define an IP interface on the switch.
   - Define each real server.
   - Assign servers to real server groups.
   - Define virtual servers and services

   For information on how to configure your network for server load balancing, see Chapter 2.

2. **Turn on URL parsing to the virtual server for virtual hosting.**

```
>> #/cfg/slb/virt 1/service 80/httpslb ena host
```

3. **Define the host names.**

```
>> # /cfg/slb/url/lb/add “www.customer1.com”
>> Server Loadbalance Resource# add “www.customer2.com”
>> Server Loadbalance Resource# add “www.customer3.com”
```

4. **Configure the real server(s) to handle the appropriate load balance string(s).**

   To add a defined sub-string:

```
>> #/cfg/slb/real 2/addlb ID
```

   Where *ID* is the identification number of the defined string.

NOTE – If you don't add a defined sub-string (or add the defined sub-string “any”), the server will handle any request.

Browser-Smart Load Balancing

By inspecting the “User-Agent” header, requests can be directed to different servers based on browser type.

Configuring Browser-Based Load Balancing

To configure “User-Agent:” header load balancing to allow the switch to perform browser-smart load balancing, perform the following procedure.

1. **Before you can configure header-based load balancing, ensure that the switch has already been configured for basic server load balancing. SLB configuration includes the following tasks:**
   - Assign an IP address to each of the real servers in the server pool.
   - Define an IP interface on the switch.
   - Define each real server.
   - Assign servers to real server groups.
   - Define virtual servers and services
2. **Turn on URL parsing to the virtual server for “User-Agent:” header..**

```
>> cfg/slb/virt 1/service 80/httpslb ena browser
```

3. **Define the host names.**

```
>> # /cfg/slb/url/lb/add “Mozilla”
>> Server Loadbalance Resource# add “Internet Explorer”
>> Server Loadbalance Resource# add “Netscape”
```

4. **Configure the real server(s) to handle the appropriate load balance string(s).**

NOTE – If you don't add a defined sub-string (or add the defined sub-string “any”), the server will handle any request.

To add a defined sub-string:

```
>> /cfg/slb/real 2/addlb ID
```

Where *ID* is the identification number of the defined string.

Cookie-Based Preferential Load Balancing

Cookies can be used to provide preferential services for customers, ensuring that certain users are offered better access to resources than other users when site resources are scarce. An example is a web server could authenticate a user via a password and then set cookies to identify them as “Gold”, “Silver” or “Bronze” customers. Using cookies, you can distinguish individuals or groups of users and place them into groups or communities that get redirected to better resources and receive better services than all other users.

Cookie-based preferential services enable the following support:

- Redirect higher priority users to a larger server or server group
- Identify a user group and redirect them to a particular server
- Serve content based on user identity
- Prioritize access to scarce resources on a web site
- Provide better services to repeat customers, based on access count

Clients to receive preferential service can be distinquished from other users by one of the following methods:

- Individual User

  Specific individual user distinguished by IP address, login authentication, or permanent HTTP cookie.

- User Communities

  Some set of users, such as “Premium Users” for service providers who pay higher membership fee than “Normal Users.” May be identified by source address range, login authentication, or permanent HTTP cookie.

- Applications

  All users using a specific application. For example, giving priority to HTTPS traffic that is performing credit card transaction versus HTTP browsing traffic.

- Content

  Users accessing specific content.

Based on one or more of the criteria above, you can load balance requests to different server groups.

Configuring Cookie-Based Preferential Load Balancing

To configure Cookie-Based Preferential Load Balancing, perform the following procedure.

1. **Before you can configure header-based load balancing, ensure that the switch has already been configured for basic server load balancing. SLB includes the following tasks:**
   - Assign an IP address to each of the real servers in the server pool.
   - Define an IP interface on the switch.
   - Define each real server.
   - Assign servers to real server groups.
   - Define virtual servers and services

   For information on how to configure your network for server load balancing, see Chapter 2.

2. **Turn on URL parsing to the virtual server.**

```
>> /cfg/slb/virt 1/service 80/httpslb 80 enable cookie sid 1 8 dis
```

3. **Define the cookie values.**

```
>> /cfg/slb/url/lb/add "Gold"
>> /cfg/slb/url/lb/add "Silver"
>> /cfg/slb/url/lb/add "Bronze"
```

   Since a session cookie does not exist in the first request of a HTTP session, we need a default server or “any” server to assign cookies to a “None” cookie HTTP request.

   Example:

   - Real Server 1: “Gold” handles gold requests.
   - Real Server 2: “Silver” handles silver request.
   - Real Server 3: “Bronze” handles bronze request.
   - Real Server 4: “any” handles any request that does not have a cookie or matching cookie.

With servers defined to handle the requests listed above, here's what happens:

- Request 1 comes in with no cookie; it is forwarded to "Real Server 4" to get cookie assigned.
- Request 2 comes in with "Gold" cookie; it will be forwarded to "Real Server 1".
- Request 3 comes in with "Silver" cookie; it will be forwarded to "Real Server 2".
- Request 4 comes in with "Bronze" cookie; it will be forwarded to "Real Server 3".
- Request 5 comes in with "Titanium" cookie; it will be forwarded to "Real Server 4", since it does not have an exact cookie match.

4. **Configure the real server(s) to handle the appropriate load balance string(s).**

To add a defined sub-string:

```
>> #/cfg/slb/real 2/addlb <ID>
```

Where *ID* is the identification number of the defined string.

NOTE – If you don't add a defined sub-string (or add the defined sub-string "any"), the server will handle any request.

URL Hashing

By default, hashing algorithms use the IP source address and/or IP destination address, depending on the application area, to determine content location. For example, Firewall Load Balancing used both IP source and destination addresses, Web Cache Redirection used only the IP destination address and Web Server Load Balancing used only the IP source address.

If URL-based WCR is enabled and the HOST header is present in the URL of a HTTP request, you can hash on the header or the URL to determine content location. All requests for *www.alteon.com*, for example, will be forwarded to the same cache server. By default, URL hashing is disabled. When enabling this option, the network administrator must also specify the number of bytes (up to 255) to be used for hashing the URL.

The applications of URL hashing for web cache redirection and server load balancing are described below.

URL Hashing for Web-Cache Redirection

Using the hashing algorithm, you can optimize “cache hits,” redirecting client requests going to the same page of an origin server to a specific cache server.

- The load-balancing algorithm must be configured to be “hash” or “minmiss”.
- Hashing is based on the URL, including the HTTP Host header (if present), up to a maximum of 255 bytes.

Example: The switch will use the string “www.alteon.com/products/ACEswitch180” for hashing the following request:

GET http://products/ACEswitch180 / HTTP/1.0

HOST:www.alteon.com

URL Hashing for Server Load Balancing

The default hashing algorithm for VIP load balancing is the IP source address. By enabling URL hashing, requests going to the same page of an origin server will be redirected to the same real server (RIP) or cache server.

- The load-balancing algorithm must be configured to be “`hash`” or “`minmiss`”.
- Hashing is based on the URL, including the HTTP Host header (if present), up to a maximum of 255 bytes.

VIP Load Balancing of Non-Transparent Caches

Customers can deploy a cluster of non-transparent proxy caches and use the VIP method to load balance requests to these cache servers.

The client's browser will be configured to send web requests to a non-transparent cache (the IP address of the VIP configured).

If hash is selected as the load-balancing algorithm, the current hashing algorithm will only use the IP Source Address for hashing in Web Server Load Balancing. Because of this, the switch may not send web requests for the same origin server to the same proxy cache server; for example, requests made from a client to http://www.alteon.com/product from different clients may get sent to different caches.

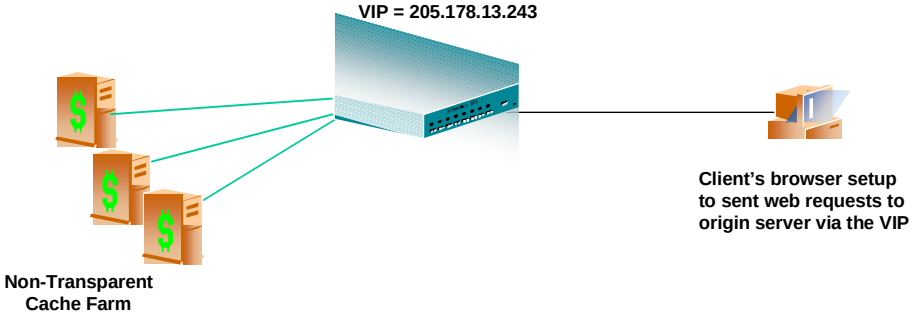


Figure 6-4 Balancing Non-Transparent Caches

Configuring URL Hashing

You can direct the same URL request to the same cache or proxy server that uses a virtual IP address to load balance proxy requests. By configuring hash or `minmisses` as the metric, the switch will use the number of bytes into the URI to calculate the hash key.

If the host field exists and the switch is configured to look into the Host: header, then the switch will also use the Host: header field to calculate the hash key.

To configure URL hashing, perform the following procedure:

1. **Before you can configure URL hashing, ensure that the switch has already been configured for basic server load balancing. SLB configuration includes the following tasks:**
   - Assign an IP address to each of the real servers in the server pool.
   - Define an IP interface on the switch.
   - Define each real server.
   - Assign servers to real server groups.
   - Define virtual servers and services

   For information on how to configure your network for server load balancing, see Chapter 2.

2. **Enable URL parsing.**

```
>> # /cfg/slb/virt 1/service 80/httpslb enable urlhash 25
```

3. **Set the metric for the real server group to `minmisses` or `hash`.**

```
>> # /cfg/slb/group 1/metric hash|minmiss
```

Exclusionary String Matching for URL SLB

URL-based SLB and WCR can match up to 128 sub-strings.

Examples of sub-strings are

- “/product”, matches URL that starts with “/product”
- “product”, matches URL that has the string “product” anywhere in the entire URL

Administrators can assign one or more sub-string to real servers. When more than one URL sub-string is assigned to a real server, requests matching any sub-string will be redirected to that real server. There is also a special sub-string known as “any” that matches all content.

WebOS supports exclusionary sub-string matching. Using this option, an administrator can define a server to accept any requests regardless of the URL, except requests with a specific sub-string. That is, an administrator can define the URL sub-string to be excluded, assign it to a real server, and have the server interpret it as an exclusion instead of an inclusion.

NOTE – Once exclusionary sub-string matching is enabled, clients cannot access the URL strings that are added to that real server. This means you cannot configure a dedicated server to receive a certain string, while at the same time have it exclude other URL strings. The exclusionary feature is enabled per server, not per string.

Example:

Sub-string #1 = cgi
Sub-string #2 = NOT cgi/form_A
Sub-string #3 = NOT cgi/form_B

When these sub-strings are assigned to a real server, the behavior is to match all cgi scripts, but exclude form_A and form_B.

Configuring Exclusionary URL Sub-String Matching

To configure exclusionary URL sub-string matching, perform the following procedure:

1. **Before you can configure URL sub-string matching, ensure that the switch has already been configured for basic server load balancing. SLB configuration includes the following tasks:**
   - Assign an IP address to each of the real servers in the server pool.
   - Define an IP interface on the switch.
   - Define each real server.
   - Assign servers to real server groups.
   - Define virtual servers and services

   For information on how to configure your network for server load balancing, see Chapter 2.

   By default, this feature is disabled. In order to enable it, you must add at least one load balancing string to the server.

```
>> # /cfg/slb/real 1/exclude enable|disable
```

Example 1:

205.178.15.49, enabled, name, weight 1, tmout 10, maxcon 200000 backup none, inter 10, retry 4, restr 8, remote disabled, proxy enabled
handle URL cookie: disabled
exclusionary string matching: enabled
2: test
real ports:
  http: vport http, group 1, httpslb
  URL hashing: disabled
  virtual server: 1, 205.178.15.45,   enabled

This server will handle any requests EXCEPT requests containing the string “test”.

Example 2:

205.178.15.49, enabled, name , weight 1, tmout 10, maxcon 200000
backup none, inter 10, retry 4, restr 8, remote disabled, proxy enabled
handle URL cookie: disabled
exclusionary string matching: enabled

2: test
3: /images
4: /product
real ports:
  http: vport http, group 1, httpslb
  URL hashing: disabled
  virtual server: 1, 205.178.15.45,   enabled

This server will handle any requests EXCEPT requests contain the string “test” OR requests that start with “/images” OR request start with “/product”.

Example 3:

205.178.15.49, enabled, name , weight 1, tmout 10, maxcon 200000
backup none, inter 10, retry 4, restr 8, remote disabled, proxy enabled
handle URL cookie: disabled
exclusionary string matching: enabled

1: any
real ports:
  http: vport http, group 1, httpslb
  URL hashing: disabled
  virtual server: 1, 205.178.15.45,   enabled

This server will NOT handle ANY requests, which is the same as disabling this server!

CHAPTER 7

Persistence

Session persistence allows you to re-establish a user's connection to a particular server. This is an important consideration for administrators of e-commerce web sites, where a server may have data associated with a specific user that is not dynamically shared with other servers at the site.

Persistence-based load balancing enables the network administrator to redirect requests from a client to the real server that initially handled the request. Persistence can be based on IP source address, HTTP cookies for HTTP requests, or SSL session ID for encrypted HTTPS requests.

IP Source Address-Based Persistence

Until recently, the only way to achieve TCP/IP session persistence was to use the source IP address as the key identifier. There are two major problems associated with session persistence based on a packet's IP source address:

- **No SLB:** Proxied clients will appear to the switch as a single IP source address and will not be able to take advantage of server load balancing on the switch. When many individual users behind a firewall use the same IP proxy source address, requests will be directed to the same server, without the benefit of load balancing the traffic across multiple servers. Persistence is supported without the capability of effectively distributing traffic load.
- **No Persistence:** When individual users share a pool of IP source addresses, persistence for any given request cannot be assured. Although each IP source address will be directed to a specific server, the address itself is randomly selected, thereby making it impossible to predict which server will receive the request. Server load balancing is supported, without true persistence for any given user.

Cookie-Based Persistence

Cookies are a mechanism for maintaining state between clients and servers. When the server receives a client request, the server issues a “cookie,” or token to the client, which the client then sends to the server on all subsequent requests. Using cookies, the server does not need to use authentication, the client IP address, or any other time-consuming mechanism to determine that the user is the same user that sent the original request.

In the simplest case, the cookie may be just a “customer ID” assigned to the user. It may be a token of trust, allowing the user to skip authentication while his or her cookie is valid. It may also be a key that associates the user with additional state data that is kept on the server, such as a shopping basket and its contents. In a more complex application, the cookie may be encoded so that it actually contains more data than just a single key or an identification number. The cookie may contain the user's preferences for a site that allow their pages to be customized.

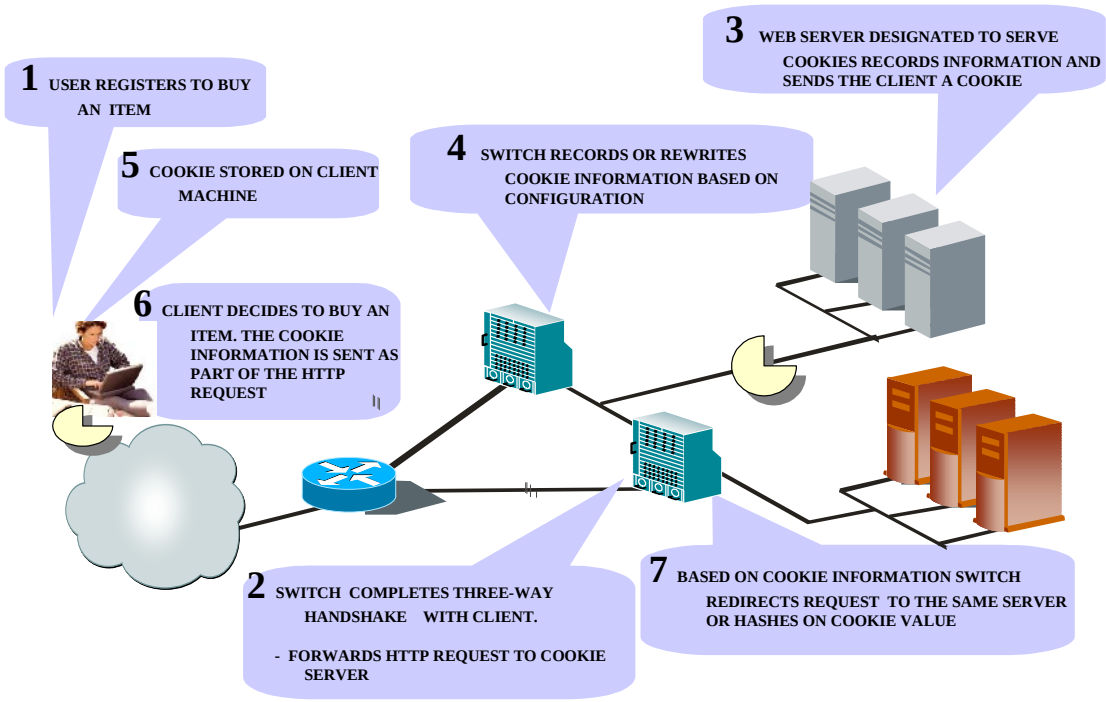


Figure 7-1 Cookie-Based Persistence - How It Works

Types of HTTP Cookies

A cookie can be defined in the HTTP header (the recommended method) or placed in the URI for hashing. On a switch running WebOS, you configure cookie-based persistence with the cookie name, offset, length, and where to match this cookie value, in the cookie header or the URI. The default is to match the cookie in the cookie header.

WebOS provides the following support:

- Cookie names of up to 20 bytes.
- Cookie values of up to 64 bytes for hashing

  This is applicable only for the passive cookie mode, using a temporary cookie. The switch hashes the cookie value to determine which server to forward the request to.
- An asterisk (*) is supported in cookie names for wildcards.

  For example, Cookie name = ASPsession*

The format of a cookie defined as an HTTP header is a "Name=Value" pair, in addition to other parameters. For example, the cookie "`SessionID=1234`" can be represented by the following:

- Cookie Header

  `Cookie:SesssionID=1234`
- Cookie within the URL

  /www.travelocity.com/Reservation/SessionID=1234

Cookies can either be permanent or temporary. A *permanent cookie* gets stored on the client's browser, as part of the response from a site's server. It will be sent by the browser when the client makes subsequent requests to the same site, even after the browser has been shut down. A *temporary cookie* is only valid for the browser session. Similar to a SSL Session-based ID, the temporary cookie expires when you shut down the browser. Based on RFC 2109, any cookie without an expiration date is a temporary cookie.

NOTE – If you will be using temporary cookies, we recommend that you use the passive cookie mode.

Examples of cookies are given below:

Cookie: `ASP_SESSIONID=POIUHKJHLKHD`

Cookie: `name=john_smith`

The first example represents an Active Server Pages (ASP) session ID. The second example represents an application-specific cookie that records the name of the client.

Modes of Operation

There are two cookie modes used to maintain session persistence; *passive* and *active* (*cookie rewrite*). Passive cookie mode works for both cookies defined in the HTTP cookie header and cookies defined in the URI. Active cookie mode (cookie rewrite mode) can only be used with cookies defined in the HTTP cookie header.

Passive Cookie Mode

In this mode, there is no special persistence cookie defined on the server. The network administrator configures the web server to embed a cookie in the server response that the switch should look for in subsequent requests from the same client. This is the recommended mode of operation when using temporary cookies.

NOTE – Passive cookie mode is not compatible with GSLB. A customer running GSLB who needs cookie-based persistence should use active cookie mode for maintaining persistence.

The following figure shows what happens in passive cookie mode.

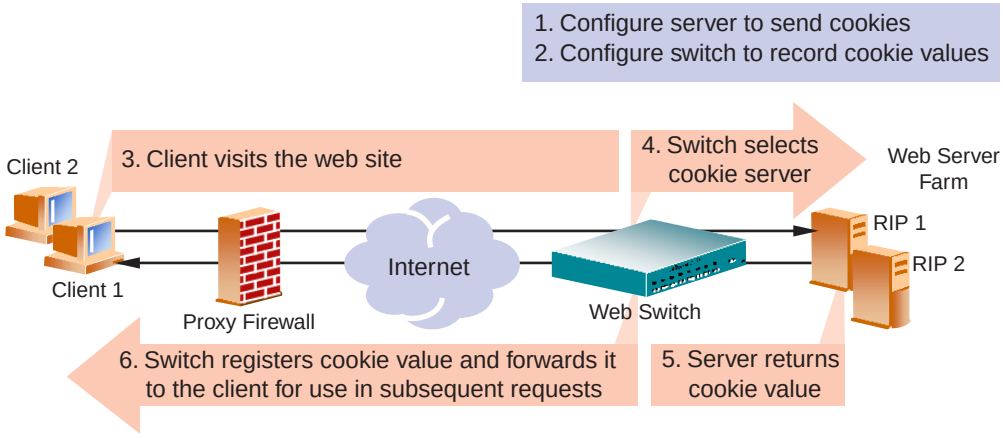


Figure 7-2 Passive Cookie Mode

Subsequent requests from Client 1 with the same cookie value will be sent to the same server (RIP 1 in this example).

Active Cookie Mode (Cookie Rewrite Mode)

In active cookie mode (or cookie rewrite mode), the switch generates the cookie value on behalf of the server, eliminating the need for a network administrator to generate cookies for each user. The server is configured to return a special persistence cookie that is pre-defined on the switch and on the server. The switch then intercepts this persistence cookie and rewrites the value to include server-specific information before sending it on to the client.

- Active cookie mode requires at least 8 bytes in the cookie header. An additional 8 bytes must be reserved if you are using cookie-based persistence with Global SLB.

NOTE – Active cookie mode (cookie rewrite mode) only works for cookies defined in the HTTP cookie header, not cookies defined in the URI. Because the switch rewrites each initial packet of a request that contains a cookie, using active cookie mode will affect switch performance.

Example: The following figure shows what happens in active cookie mode.

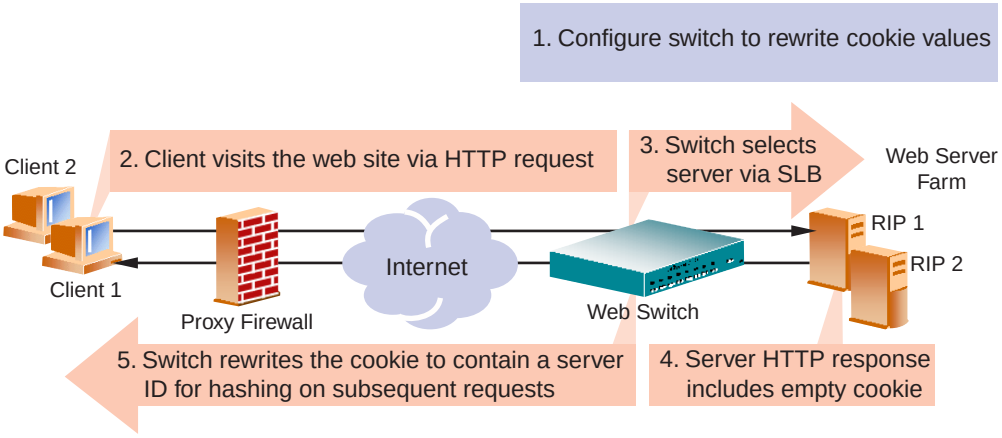


Figure 7-3 Active Cookie Mode

NOTE – When the switch rewrites the value of the cookie, the rewritten value represents the responding server; that is, the value can be used for hashing into a server ID or it can be the server IP address or server ID. The rewritten cookie value is encoded. Subsequent requests from Client #1 with the same cookie value will be sent to the same server.

Cookie Assignment Servers

Servers defined within a group can be configured to assign/issue cookies to first-time visitors to the website. These are known as *cookie assignment servers*. If there are multiple cookie assignment servers, the switch will load balance the request to these servers. Cookie assignment servers do not participate in load balancing requests that already have a cookie assigned.

Cookie assignment servers are not required for cookie-based persistence. If there are no cookie assignment servers defined, the switch assumes that either the client's request contains a cookie or that the real server that gets the first request without a cookie will issue a cookie value back to the client browser.

Cookie Values

Cookie assignment servers or any real server can return a cookie value to the client browser. The entire cookie value or part of it is used for selecting the appropriate server for directing the request. Two examples of cookies are shown below:

Example 1: `cookie: sid=1234cdb20f043243`

Example 2: `cookie: sid=0123456789abcdef;`

In the first example, the server's IP address has been embedded in the cookie value returned by the server. The value "cdb20f04" represents the IP address "205.178.15.4" in hexadecimal format. In this example, the user has defined the following for use to compute the server to redirect the request:

Cookie name = sid
Offset = 5 (the starting position of the value to be used for hashing)
Length = 8 (the number of bytes from the starting position)

Since the defined value represents a valid real server IP address, the switch will use the value directly to determine the server that will receive subsequent requests instead of using the value for hashing.

In the second example, the defined cookie value does not match a valid real server IP address and thus will be used for hashing to determine the appropriate real server that subsequent requests should be directed to.

Sequence of Events Using Cookie Assignment Server

Two scenarios for cookie-based persistence, demonstrating use of cookie assignment servers, are given below.

NOTE – When using cookie assignment servers, you must use `hash` or `minmisses` as the load balancing metric for the real server group.

1st time a HTTP client request arrives at the switch, without the specified cookie:

- With cookie assignment servers defined:
  - The switch will forward the requests to one of the cookie assignment servers to get a cookie value return to the client browser.
  - Subsequent requests from this client with this same cookie value will get hashed to an appropriate real server based on the cookie value.
- Without cookie assignment servers defined:
  - The switch will direct the request to a real server in the group, based on the load-balancing algorithm defined.
  - Subsequent requests from this client will get directed to the same server. The cookie value can be the real server IP address or it can be a cookie value previously embedded in the server response to the client that the switch will then hash to determine the server that should receive the request.

Subsequent client HTTP requests arrive at the switch, with the specified cookie:

1. **Client HTTP request sent to the switch, with the specified cookie.**
2. **The switch will use the "`offset`" and "`length`" parameters to determine which part of the cookie value should be used for determining the real server to direct the request. For the discussion below, let's call this the `persistence_cookie_value`.**
3. **If the `persistence_cookie_value` represents a valid real server IP address, the switch will redirect the request to the appropriate real server.**
4. **If the `persistence_cookie_value` does not represent a valid real server IP address, the switch will use the value to compute (hash) the server that should get this request.**

   We extended the concept of cookie by matching cookies either in the URI or the cookie header. That is, the user can configure cookie based persistence with the cookie name, offset, length and where to match this cookie value, in the URI or the cookie header itself. This is a configuration option as described below. The default is to match the cookie in the cookie header.

Assigning Server to Serve Cookies When Client Requests Don't Contain the Specified Cookie

With the `nocookie` option enabled for specific servers in a real server group, connection requests without cookies are load balanced across those real servers. Requests with specified cookies will be load balanced across the other real servers in the group.

To assign a server specifically to serve cookies when client requests don't contain the specified cookie, use this command:

```
>> # /cfg/slb/real 1/nocook ena          (Enable SLB for non-cookie requests)
```

Example:

Real Server Group 1 consists of real servers 1,2,3,4,5,6. Only real servers 5 and 6 have `/nocook` enabled.

When client requests come in for the first time and don't have the specified cookie, servers 5 and 6 will be load balanced to assign a specified cookie to the server. Subsequently, when a request from the same client comes in with the specified cookie, it will get hashed and load balanced across servers 1, 2, 3, or 4.

Using Cookie Assignment Servers: Configuration Examples

If you want to look for cookie name/value pair in the HTTP cookie header, you would configure the fourth parameter to be "*disable*."

If you want to look for cookie name/value pair in the URI, you would configure the fourth parameter to be "*enable*."

Example 1:

HTTP Header:

```
GET /product/switch/UID=12345678;ck=1234...
Host: www.alteon.com
Cookie: UID=87654321;
....
```

- If we configure the `enable|disable` parameter in **`/cfg/slb/virt 1/service 80/pbind cookie passive "cookiename" offset-length disable`** to be "*disable*," the switch will look for the name/value pair in:
  Cookie: UID=87654321;
- If we configure the `enable|disable` parameter in **`/cfg/slb/virt 1/service 80/pbind cookie passive "cookiename" offset-length enable|disable`** to be "*enable*," the switch will look for the name/value pair in:
  `/product/switch/UID=12345678;ck=1234..`

Example 2.

HTTP Header:

```
Cookie: sid=0123456789abcdef; name1=value1;...
```

- We want to use cookie name "*sid*" and "*789a*" of the cookie value as a hashing key to compute the real server. The configuration would be:
  `/cfg/slb/virt 1/service 80/pbind cookie passive "cookiename" sid 8 4 disable`

  Based on the command parameters used above, the switch will use the an offset of 8 bytes and a length of 4 bytes from that offset to determine which part of the cookie value should be used for determining the real server that should get the request.
- If we want the whole value of 'sid', the configuration would be:
  `/cfg/slb/virt 1/service 80/pbind cookie passive "cookiename" sid 1 16 disable`

Configuring Cookie-Based Persistence

When to Use Passive or Active Mode

- Temporary cookie: When implementing cookie-based persistence using temporary cookies, passive mode is recommended. You would use `leastconns` or `roundrobin` as the load balancing metric for the real server group. Using this configuration, no cookie assignment server(s) would be needed.
- Permanent cookie: Use the passive mode, with the server embedding the IP address, or use active mode.

What LB Algorithm Should I Use with Cookie-Based Persistence?

To ensure optimal load balancing, use either `leastconns` or `roundrobin` as the load balancing metric. However, if you will be using cookie assignment servers, you must use either `hash` or `minmisses` as the load balancing metric.

Using Cookie-Based Persistence with GSLB

If you will be using cookie-based persistence with Global SLB, you must use the active cookie mode and reserve 16 bytes in the cookie header.

What If Client Browser Doesn't Accept Cookies?

Under normal conditions, most browsers are configured to accept cookies. However, if a client browser is not configured to accept cookies, you must use `hash` as the load balancing metric to maintain session persistence. With cookie persistence enabled, session persistence for requests from a browser that doesn't accept cookies will be based on the source IP address. Many individual users coming from a proxy firewall will be directed to a single server, resulting in traffic being concentrated on a single server, instead of load balancing across the available real servers.

Configuration Procedure

1. **Before you can configure cookie-based persistence, you need to configure the switch for basic server load balancing. This includes the following tasks:**

   - Assign an IP address to each of the real servers in the server pool.
   - Define an IP interface on the switch.
   - Define each real server.
   - Assign servers to real server groups.
   - Define virtual servers and services

   For information on how to configure your network for server load balancing, see Chapter 2.

2. **Either enable Direct Access Mode for the switch or disable DAM and specify proxy IP (PIP) address(es) on the client port(s).**

   - Enable Direct Access Mode for the switch.

```
>> # /cfg/slb/adv/direct ena          (Enable Direct Access Mode on switch)
```

   - Disable Direct Access Mode and specify PIP address(es) on the client port(s).

NOTE – If Virtual Matrix Architecture (VMA) is enabled on the switch, you must configure a unique proxy IP address for every port.

```
>> # /cfg/slb/adv/direct disable      (Disable DAM on the switch)
>> # /cfg/slb/port 1                  (Select network port #1)
>> SLB port 1# pip 200.200.200.68     (Set proxy IP address for port #1)
```

3. **If proxy IP addresses are used, make sure server processing is disabled on the server port.**

```
>> # /cfg/slb/port 1                  (Select switch port #1)
>> SLB port 1# servr dis               (Disable server processing on port #1)
```

4. **Select the appropriate load balancing metric for the real server group.**

```
>> # /cfg/slb/group 2 metrc hash          (Select hash as server group metric)
```

- If embedding an IP address in the cookie, select `roundrobin` or `leastconns` as the metric.
- If you are NOT embedding the IP address in the cookie, select `hash` as the metric in conjunction with a cookie assignment server.

  While you may experience traffic concentration using the `hash` metric with a cookie assignment server; using a `hash` metric without a cookie assignment server will cause traffic concentration on your real servers.

5. **Enable cookie-based persistence on the virtual server service.**

```
>>/cfg/slb/virt 1/service 80/pbind cookie passive 'cookieName' off-
set-length enable|disable
>> Virtual Server 1 http Service# pbind
Enter clientip|cookie|sslid persistence mode:   cookie
Enter passive|rewrite cookie persistence mode [p/r]:    passive
Enter Cookie Name:      sid
Enter the starting point of the cookie value:   1
Enter the number of bytes to be extract:        8
Look for cookie in URI [e|d]:   ena
```

Once you specify "cookie" as the mode of persistence, you will be prompted for the following parameters:

- Cookie persistence mode: passive or rewrite (active)
- Cookie name
- Starting point of the cookie value
- Number of bytes to be extracted
- Enable/disable looking for cookie in the URI

NOTE – Cookie rewrite mode only works with cookies defined in the HTTP cookie header. When the cookie rewrite mode is selected, you will not be prompted for a value in this field.

To configure the switch to look for cookie name/value pair in the Cookie: field of the HTTP header, set the eighth (last) parameter ("Look for cookie in URI") to **`disable`**. If you want the switch to look for the cookie name/value pair in the URI, set the eighth parameter to **`enable`**.

Example 1.

HTTP Header:

```
GET /product/switch/UID=12345678;ck=1234...
Host: www.alteon.com
Cookie: UID=87654321;
```

- If you set the eighth (last) parameter to **`disable`**, the switch will look for the name/value pair in -
  `Cookie: UID=87654321;`

```
>> /cfg/slb/virt 1/service 80/pbind cookie passive UID 1 8 dis
```

- If you set the eighth (last) parameter to **`enable`**, the switch will look for the name/value pair in -
  `product/switch/UID=12345678;ck=1234..`

```
>> /cfg/slb/virt 1/service 80/pbind cookie passive UID 1 8 ena
```

Example 2. HTTP Header:

Cookie: sid=0123456789abcdef; name1=value1;...

If you want the switch to use cookie name 'sid' and '789a' of the value as a hashing key to the real server, use this command:

```
>> # /cfg/slb/virt 1/service 80/pbind cookie passive sid 8 4 dis
```

To use the whole value of "*sid*" as a hashing key to the real server, use this command:

```
>> # /cfg/slb/virt 1/service 80/pbind cookie passive sid 1 16 dis
```

You can also use wild cards in configuring cookie names for cookie-persistent load balancing, as shown in this command:

```
>> # /cfg/slb/virt 1/service 80/pbind cookie passive ASPSESSIONID* 1
16 dis
```

With this configuration, the switch will look for a cookie name that starts with "ASPSESSIONID." ASPSESSIONID123, ASPSESSIONID456, ASPSESSIONID789, will be seen by the switch as the same cookie name.

Directing Cookie Client to a Specific Server

This can be done using passive cookie mode or active cookie mode (cookie rewrite mode). The procedure for each mode is provided below.

Passive Cookie Mode

By embedding the real server's IP address (in hexadecimal format) in the cookie value, the cookie assignment server can tell the client exactly which server to go back to in subsequent connections.

Example 1:

1. **Convert the client IP address from dotted format into hexadecimal:**

   205.178.15.4 --> cdb20f04

2. **Embed the converted address into the cookie value.**

   `sid=1234cdb20f043243`

3. **Configure the switch to read in the correct cookie value.**

```
>> /cfg/slb/virt 1/service 80/pbind cookie passive sid 5 8 dis
```

Example 2:

1. **Convert the client IP address from dotted format into hexadecimal:**

   205.178.15.9 --> cdb20f09

2. **Embed the converted address into the cookie value.**

   sid=cdb20f09

3. **Configure the switch to read in the correct cookie value.**

```
>> /cfg/slb/virt 1/service 80/pbind cookie passive sid 1 8 dis
```

Example 3:

Using cookie passive mode, the switch will examine the server's 'Set-Cookie:' value and direct all subsequent connections to the server that assigned the cookie, just like SSL session ID.

Server 1 --> Set-Cookie: sid=1234567
Client 2 --> Cookie: sid=1234567

All of Client 2's traffic with the `cookie sid=1234567` will be directed to real server 1.

Active Cookie Mode (Cookie Rewrite Mode)

Example 1:

If the switch is configured to be in cookie rewrite mode with the seventh parameter (byte length) configured to be 8 or 16, the switch will rewrite the cookie value with the encrypted real server IP address (RIP) or encrypted virtual server IP address (VIP) and RIP.

Cookie-based persistence is configured on the switch as follows:

```
>> /cfg/slb/virt 1/service 80/pbind cookie rewrite sid 1 8 dis
```

Server 1 (205.178.15.4) --> Set-Cookie: sid=alteonpersistence;
The switch will rewrite: --> Set-Cookie: sid=cdb20f04rsistence;
Client --> Server 1 --> Cookie: sid=cdb20f04rsistence;

Example 2:

Cookie-based persistence is configured on the switch as follows:

```
>> /cfg/slb/virt 1/service 80/pbind cookie rewrite sid 1 16 dis
```

VIP --> (205.178.15.10)
Server 1 (205.178.15.4) --> Set-Cookie: sid=alteonpersistence;
The switch will rewrite: --> Set-Cookie: sid=cdb20f04cdb20f0ae;
Client --> Server 1 --> Cookie: sid=cdb20f04cdb20f0ae;

SSL Session ID-Based Persistence

Secure Sockets Layer (SSL) is a set of protocols built on top of TCP/IP that allow an application server and user to communicate over an encrypted HTTP session, providing authentication, non-repudiation, and security. The SSL protocol "handshake" is performed using clear text; the content data is then encrypted, using an algorithm exchanged during the "handshake," prior to being transmitted.

Using the SSL session ID, the switch forwards the request to the real server that it bound the user to during the last session. Because SSL protocol allows many TCP connections from the same client to a server to use the same session ID, key exchange needs to be done only when the session ID expires. This cuts down on CPU overhead on the server and provides a mechanism, even when the client IP address changes, to send all sessions to the same server.

NOTE – The destination port number to monitor for SSL traffic is user configurable.

How SSL Session ID-Based Persistence Works

- All SSL sessions which present the same session ID (32 random bytes chosen by the SSL server) will be directed to the same real server.

NOTE – The SSL session ID is only "visible" to the switch after the TCP 3-way handshake. In order to make a forwarding decision, the switch must terminate the TCP connection to examine the request.

- New sessions are sent to the real server based on the metric selected (`hash`, `roundrobin`, `leastconns`, or `minmisses`).
- If no session ID is presented by the client, the switch picks a real server based on the metric for the real server group and waits until a connection is established with the real server and a session ID is received.
- The session ID is stored in a session hash table. When a subsequent connection comes in with the same session ID, it is sent to the same real server. This binding is preserved even if the server changes the session ID mid-stream. A changes of session ID in the SSL protocol will cause a full handshake to happen.
- Session IDs are kept in the switch until an idle time equal to the configured real server timeout (default = 10 minutes) for the selected real server has expired.

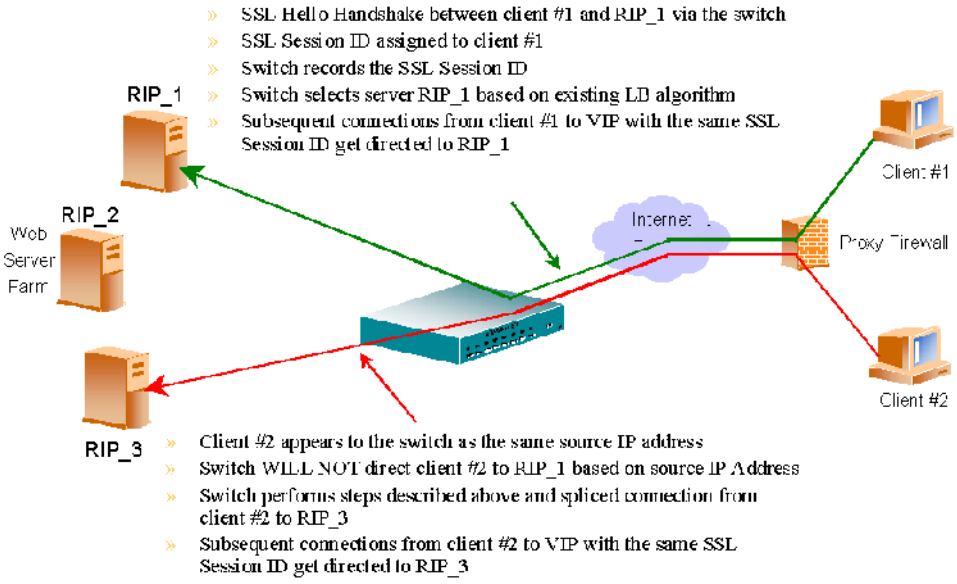


Figure 7-4 SSL Session ID-Based Persistence

Configuring SSL Session ID-Based Persistence

To configure session ID-based persistence for a real server, perform the following steps:

1. **Configure real servers and services for basic server load balancing, as indicated below:**
   - Define each real server and assign an IP address to each real server in the server pool.
   - Define a real server group and set up health checks for the group.
   - Define a virtual server on the virtual port for HTTPS (for example, 443) and assign a real server group to service it.
   - Enable server load balancing on the switch.
   - Enable client processing on the port connected to the client.

   For information on how to configure your network for server load balancing, see Chapter 2, "Virtual Server Load Balancing."

2. **If a proxy IP address is not configured on the client port, enable Direct Access Mode for real servers.**

```
>> # /cfg/slb/adv/direct ena                    (Enable Direct Access Mode on switch)
```

3. **Select the persistent binding type for the virtual port to configure session ID-based persistence.**

```
>> /cfg/slb/virt <virtual-server-number>/service <virtual-port-number> pbind
sessid
```

4. **Enable client processing on the client port.**

```
>> /cfg/slb/port <port-number>/client ena
```

5. **Enable server processing on the server port.**

```
>> /cfg/slb/port <port-number>/server ena
```

CHAPTER 8

Bandwidth Management

This feature enables website managers to allocate a certain portion of the available bandwidth for specific users or applications. Traffic classification can be based on user or application information. Bandwidth policies can be configured to set lower and upper bounds on the bandwidth allocation.

Bandwidth Management Overview

To manage bandwidth, you create one or more bandwidth management contracts. The switch uses these contracts to limit individual traffic flows. Each contract comprises the following:

- a classification policy where certain frames are grouped together
- a bandwidth policy that specifies a set of bandwidth usage limitations to be applied to these frames.

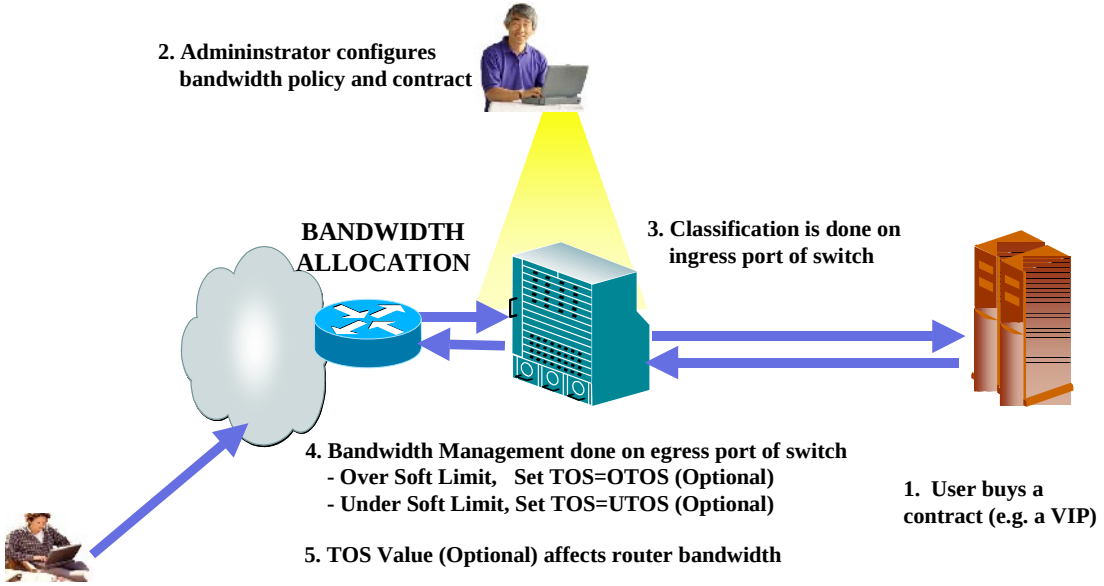


Figure 8-1 Bandwidth Management: How It Works

NOTE – At any given time, up to 256 contracts can be configured for a single AD3/A180e switch. Up to 1024 contracts can be created for a single AD4 or A184 switch.

- When Virtual Matrix Architecture (VMA) is not enabled, *bandwidth classification* is done on the *ingress* side of the switch (at the ingress port or designated port) and can be based on the following: source port, VLAN, VIP, virtual service, filters, and so on.

  When VMA is enabled, non-filter and non-SLB traffic classification get done on the "ingress port;" that is, the port on which the frame is received (not the "client port" or the "server port"). If the traffic classification is filter or SLB traffic, then the classification gets done on the "designated port".

- *Bandwidth management* is done on the *egress* port of the switch; that is, the port the frame is leaving from. However, in the case of multiple routes or trunk groups, the egress port can actually be one of several ports from the point of view of where the queues live.

  Each frame is put into a managed buffer and placed on a contract queue. According to the configured rate of the contract, the current egress rate of the ports, and the buffer size set for the contract queue, the next time that a frame is supposed to be transmitted for the contract queue is calculated and given to the scheduler. The scheduler then organizes all the frames to be sent according to their time-based ordering and meters them out to the port.

Each bandwidth management contract is assigned a bandwidth policy index and (optionally) a name. This index can be viewed using the `/cfg/bwm/cont` menu. Contracts can be enabled and disabled. The set of classifications associated with each contract can be viewed using the `/info/bwm` menu.

For frames qualifying for multiple classifications, precedence of the contracts is also specified on a per-contract basis. If no precedence is specified, the default ordering is used (see "Precedence" on page 166).

Bandwidth Policies

Bandwidth policies are bandwidth limitations that are defined for any set of frames, specifying the guaranteed bandwidth rates. A bandwidth policy is often based on a rate structure that a Web hoster or co-location provider would charge a customer for bandwidth utilization. There are three rates that are configured; a Committed Information Rate/Reserved Limit, a Soft Limit, and a Hard Limit, as described below.

A queue depth is also associated with a policy. A queue depth is the size of the queue that holds the data. It can be adjusted to accommodate delay-sensitive (e.g., audio) traffic vs. drop-sensitive traffic (e.g., FTP).

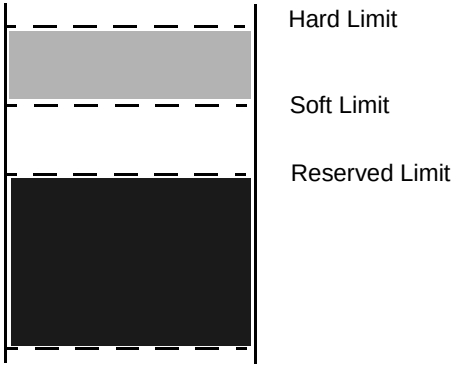


Figure 8-2 Bandwidth Rate Limits

Rate Limits

A bandwidth policy specifies three limits, listed and described in the following table:

Table 8-1 Bandwidth Rate Limits

Rate Limit	Description
Committed Information Rate (CIR) or Reserved Limit	This is a rate that a bandwidth class is always guaranteed. In configuring bandwidth management contracts, ensure that the sum of all committed information rates never exceeds the link speeds associated with ports on which the traffic is transmitted. The switch will attempt to enforce this while it is being configured. In the case where the total CIRs exceed the outbound port bandwidth, the switch will perform a graceful degradation of all traffic on the associated ports.
Soft Limit	This is the desired bandwidth rate; that is, the rate the customer has agreed to pay for on a regular basis. When output bandwidth is available, a bandwidth class will be allowed to send data at this rate. No exceptional condition will be reported when the data rate does not exceed this limit.
Hard Limit	This is a “never exceed” rate. A bandwidth class is never allowed to transmit above this rate. Typically, traffic bursts between the soft limit and the hard limit are charged for.

Bandwidth Policy Configuration

Each bandwidth policy, comprised of the reserved rate, soft and hard limits, is assigned an index. These policies can be found under the `/cfg/bwm` menu. Up to 64 bandwidth policies can be defined. Bandwidth limits are usually entered in mbps (entered as nk).

NOTE – To allow better granularities at low configured rates, any value can be entered in kbps by appending a ‘k’ to the entered number. For example, 1 Mbps can be entered as either ‘1’ or as ‘1024k’.

The following table lists the granularity of policy limits:

Table 8-2 Bandwidth Policy Limits

Bandwidth Range	Interval	Bandwidth Range	Interval
250 Kbps to 5000 Kbps	250 Kbps	50 Mbps to 150 Mbps	10 Kbps
1 Mbps to 20 Mbps	1 Mbps	150 Mbps to 500 Mbps	25 Mbps
20 Mbps to 50 Mbps	5 Mbps	500 Mbps to 1000 Mbps	50 Mbps

In addition, a queue size is associated with each policy. The queue size is measured in kilo-bytes.

Classification Policies

The frames associated with a particular bandwidth management contract are specified using the parameters listed below. All of these classifications are aimed at limiting the traffic outbound from the server farm for bandwidth measurement and control.

Server Output Bandwidth Control

- Physical Port

  All frames from a specified physical port.

- VLAN

  All frames from a specified VLAN. This means that even if VLAN translation occurs the bandwidth policy is based on the ingress VLAN.

- IP Source Address

  All frames with a specified IP source address with subnet mask.

- IP Destination Address

  All frames with a specified IP destination address with subnet mask.

- Switch Virtual Services

  The following are various Layer 4 groupings.

  - A single virtual server
  - A group of virtual servers
  - A virtual service for a particular virtual server

    Select a particular port number (virtual service) within a particular VIP.

Application Bandwidth Control

Classification policies allow bandwidth limitations to be applied to particular applications, that is, they allow applications to be identified and grouped. Classification can be based on any filtering rule, including those listed below:

- TCP Port Number

  All frames with a particular TCP port number (either source or destination).

- UDP

  All UDP frames.

- UDP Port Number

  All frames with a particular UDP port number (either source or destination).

Combinations

Combinations of classifications are limited to grouping items together into a contract. For example, if you wanted to have three different virtual servers associated with a contract, you would specify the same contract index on each of the three virtual server IP addresses. You can also combine filters in this manner.

Precedence

If a frame would qualify for different classifications, it is important to be able to specify which classification it should be associated with. There are two mechanisms to address this; a per-contract precedence value and a default ordering. If a contract does not have an assigned precedence, then the ordering is as follows:

1. Virtual Server

2. Filter

3. VLAN

4. Source Port/Default Assignment

Bandwidth Classification Configuration

Any item that is configured with a filter can be used for bandwidth management. Bandwidth classification is performed using the following menus:

- `/cfg/slb/filt` is used to configure classifications based on the IP destination address, IP source address, TCP port number, UDP, and UDP port number, or any filter rule.
- `/cfg/slb/virt` is used to configure classifications based on virtual servers.
- `/cfg/port` is used to configure classifications based on physical ports. (In case of trunking, use `/cfg/trunk`.)
- `/cfg/vlan` is used to configure classifications based on VLANs.

To associate a particular classification with a contract, enter the contract index into the "`cont`" menu option under the applicable configuration menus.

Restricting Bandwidth Usage

Data Pacing

The mechanism used to keep the individual traffic flows under control is called *data pacing*. It is based on the concept of a virtual clock and theoretical departure times (TDT). The actual calculation of the TDT is based initially on the soft limit rate. The soft limit can be thought of as a target limit for the ISP's customer. So long as bandwidth is available and the classification queue is not being filled at a rate greater than the soft limit, the TDT will be met for both incoming frames and outgoing frames and no borrowing or limitation will be necessary.

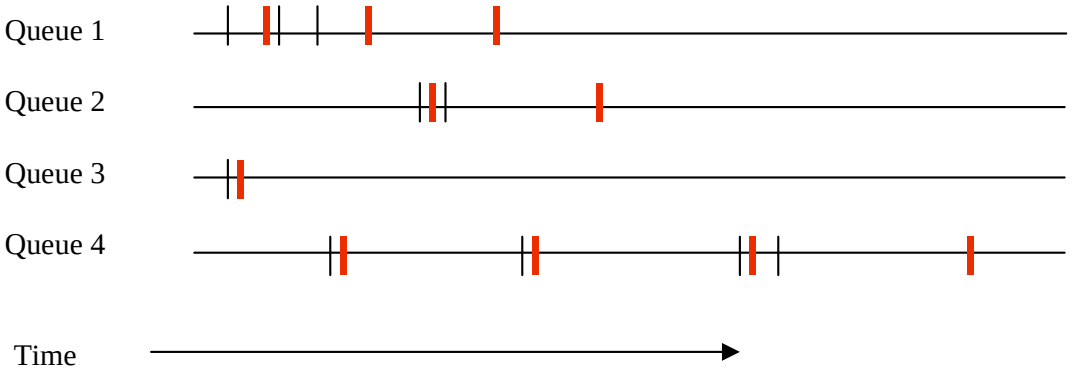


Figure 8-3 Virtual Clocks and TDT

If the data is arriving more quickly than it can be transmitted at the soft limit and there is still sufficient bandwidth, the rate is adjusted upwards based on the depth of the queue, until the rate is fast enough to reduce the queue depth or the hard limit is reached. If the data cannot be transmitted at the soft limit, then the rate is adjusted downward until the data can be transmitted or the Committed Information Rate (CIR) is hit. If the CIR is over-committed among all the contracts configured for the switch, graceful degradation will reduce each CIR until the total bandwidth allocated fits within the total bandwidth available.

Frame Discard

When packets in a contract queue have not yet been sent and the buffer size set for the queue is full, any new frames attempting to be placed in the queue will be discarded.

Bandwidth Statistics and History

Statistics are maintained in order to allow switch owners to bill for bandwidth usage. Statistics frequency and count are configurable. Statistics are kept in the individual Switch Processor (SP) and then collected every second by the MP (Management Processor). The MP then combines the statistics, as statistics for some classifications may be spread across multiple SPs.

The MP maintains some global statistics, such as total octets and a window of historical statistics. The historical statistics are kept on a configurable time per interval. When the history buffer is ready to overflow, it can be optionally e-mailed to a user for long term storage. The SMTP protocol is used for this transfer. To obtain graphs, the data must be collected and processed by an external entity through SNMP or through e-mailed logs.

Statistics Maintained

The total number of octets, octet discards, and times over the soft limit are kept for each contract. The history buffer maintains the average queue size for the time interval and the average rate for the interval.

Configuring the History Buffer

A total memory block of 128K is kept available for the history buffer. This block is used as specified in the `/cfg/bwm/stats` menu. The interval is specified and the number of intervals the switch will be able to keep, given the contract count, is calculated by the switch. This information is then stated as an output of the configuration command. History is maintained only for the contracts for which the history option is enabled. (`cfg/bwm/cont 1/hist`)

Statistics and MIBs

- For Existing Bandwidth Management Classes

  As mentioned above, the MP maintains per contract rate usage statistics. These are obtainable via a private MIB.

- When Bandwidth Management Services Are Not Enabled

  Even when bandwidth management is not enforced, the MP can still collect classification information and report it. This allows the customer to watch a network for a while before deciding how to configure it. This feature can be turned on using **`/cfg/bwm/force`**.

Packet Coloring (TOS bits)

Burst Limit

Whenever the soft limit is exceeded, optional packet coloring can be done to allow down-stream routers to use *diff-serv* mechanisms (that is, writing the TOS byte of the IP header) to delay or discard these “out of profile” frames. Frames that are not “out of profile” are marked with a different, higher priority value.

The TOS bits are set in the bandwidth policy menu. The values allowed are 0-255 and a “don’t change” value that may also be specified. There are two fields for specifying the TOS bits, the “within profile” field and the “out of profile” field. Typically the values specified should match the appropriate diff-serv specification, but could be different depending on the environment of the particular customer.

NOTE – This feature can be enabled or disabled on a per-contract basis.

Operational Keys

There are two operational keys for bandwidth management: a standard key and a demo key. The demo key automatically expires after a demo time period. These keys may only be enabled if Layer 4 services have been enabled.

Configuring Bandwidth Management

1. **Configure the switch as you normally would for virtual server load balancing (VSLB). This includes the following tasks:**
   - Assign an IP address to each of the real servers in the server pool.
   - Define an IP interface on the switch.
   - Define each real server.
   - Define a real server group.
   - Define a virtual server.
   - Define the port configuration.

   For more information about VSLB configuration, refer to Chapter 2 of the *Application Guide*.

2. **On the switch, select a bandwidth management contract and (optionally) a name for the contract.**

   Each contract must have a unique number, from 1 to 256.

```
>> Main# /cfg/bwm/cont 1                    (Select BWM Contract 1)
>> Bandwidth Management# name BigCorp       (Select contract name "BigCorp")
```

3. **(Optional) Set a precedence value for the bandwidth management contract.**

   If a frame qualifies for different classifications, you should specify which classification it should be associated with, from 1 to 256.

```
>> Bandwidth Management# prec 1             (Sets contract precedence value to 1)
```

4. **(Optional) Set a WTOS value for the bandwidth management contract.**

```
>> Bandwidth Management# wtos               (Sets contract wtos value)
```

5. **Enable the bandwidth management contract.**

```
>> Bandwidth Management# ena                (Enables this BW contract)
```

6. **Select a bandwidth management policy.**

   Each policy must have a number, from 1 to 64.

```
>> Bandwidth Management# policy 1           (Select BWM policy 1)
```

7. **Set the hard, soft, and reserved rate limits for the policy, in mbps.**

   Typically, burst rates between the soft and hard limit are charged for. Each limit must be set between 256K-1000M.

   NOTE – For rates less than 1 Mbps, append a “K” suffix to the number.

```
>> Policy 1# hard 6          (Set “never exceed” rate)
>> Policy 1# soft 5          (Set desired bandwidth rate)
>> Policy 1# resv 4          (Set committed information rate)
```

8. **(Optional) Set the TOS (Type of Service) byte value, between 0-255, for the policy underlimit and overlimit.**

   There are two parameters for specifying the TOS bits: underlimit (`utos`) and overlimit (`otos`). These TOS values are used to overwrite the TOS values of IP packets if the traffic seen for a contract is under or over the soft limit, respectively. These values have no significance to a contract if TOS overwrite is disabled in the Bandwidth Contract Menu (**`cfg/bwm/wtos dis`**).

   The administrator has to be very careful in selecting the TOS values because of their greater impact on the downstream routers.

```
>> Policy 1#  utos 204       (Set BW Policy underlimit)
>> Policy 1#  otos 192       (Set BW Policy overlimit)
```

9. **Set the buffer limit for the policy.**

   Set a parameter between 8192-128000 bytes.The buffer depth for a bandwidth management contract should be set to a multiple of the packet size

   NOTE – Keep in mind that the total buffer limit is 128K.

```
>> Policy 1#  buffer 32640   (Set BW Policy buffer limit)
```

10. **Set the classification policy for the contract.**

    Each bandwidth management contract must be assigned a classification policy. The classification can be based a filter or virtual service(s). Filters are used to create classification policies based on the IP source address, IP destination address, TCP port number, UDP, and UDP port number.

```
>> Policy 1# /cfg/slb/virt 1/cont 1          (Assign contract 1 to virtual server 1)
>> Virtual Server 1# /cfg/slb/filt 1/adv/cont 1(Assign contract 1 to filter 1)
```

11. **On the switch, enable, apply, and verify the configuration.**

NOTE – If you purchased the Bandwidth Management option, make sure you enable it by typing **/oper/swkey** and entering its software key.

```
>> Filter 1 Advanced# /cfg/bwm/on          (Turn Bandwidth Management on)
>> Bandwidth Management# apply             (Make your changes active)
>> Bandwidth Management# cur               (View current settings)
```

    Examine the resulting information. If any settings are incorrect, make any appropriate changes.

12. **On the switch, save your new configuration changes.**

```
>> Bandwidth Management# save              (Save for restore after reboot)
```

13. **On the switch, check the bandwidth management information.**

```
>> Bandwidth Management# /info/bwm <contract number>(View BWM information)
```

    Check that all bandwidth management contract parameters are set correctly. If necessary, make any appropriate configuration changes and then check the information again.

CHAPTER 9

High-Availability

In a *high-availability* network topology, no device can create a single point-of-failure for the network, or force a single point-of-failure to any other part of the network. This means that your network will remain in service despite the failure of any single device. To achieve this usually requires a redundancy for all vital network components.

Alteon WebSystems Web switches support high-availability network topologies through an enhanced implementation of the Virtual Router Redundancy Protocol (VRRP).

Our implementation of VRRP supports three modes of high-availability: active-standby, active-active, and hot-standby. The first mode, *active-standby*, is based on standard VRRP. The second and third modes, *active-active* and *hot-standby*, are based on proprietary Alteon WebSystems extensions to VRRP. Each is briefly summarized below, with a pointer to where you'll find more information.

- Active-Standby

  In an active-standby configuration, two Web switches are used. Both switches support active traffic, but are configured so that they do not support the same service simultaneously. Each switch is active for its own set of services, and behaves as a backup for the services on the other switch. If either switch fails, the remaining switch takes over processing for all services. The backup switch may forward L2 and L3 traffic, as appropriate. For a detailed description of this approach, refer to page 176.

- Active-Active

  In an active-active configuration, two Web switches provide redundancy for each other, with both active at the same time for the **same** services. For a detailed description of this approach, refer to page 177.

- Hot-Standby

  VRRP has been extended to support hot-standby failover configurations. Using this approach, the Spanning Tree Protocol is not needed to eliminate bridge loops. This speeds up failover when a switch fails. The standby switch blocks all ports configured as standby ports, whereas the master switch enables these same ports. Consequently, on a given switch, all virtual routers are either master or backup, they cannot change state individually. For a detailed description of this approach, refer to page 177.

VRRP

Virtual Router Redundancy Protocol (VRRP) support on Alteon WebSystems' Web switches provides redundancy between routers in a LAN. This is accomplished by configuring the same virtual router IP address and ID number on each participating VRRP-capable routing device. One of the virtual routers is elected as the master, based on a number of priority criteria, and assumes control of the shared virtual router IP address. If the master fails, one of the backup virtual routers will take control of the virtual router IP address and actively process traffic addresses to it.

Alteon Websystems has extended VRRP to include virtual servers as well, allowing for full active/active redundancy between its Layer 4 switches. This enables more efficient network resource allocation and supports more complex failover topologies.

Active-active redundant switch configurations increase application availability by removing single points of failure from networks. At the same time, when both switches are healthy, active-active configurations increase performance and capacity by allowing two or more Web switches to support the same interface and service.

Alteon WebSystems' active-active redundancy is based on Virtual Router Redundancy Protocol (VRRP), as defined in RFC 2338, plus Alteon-specific extensions to VRRP.

Failover Methods: An Overview

With service availability becoming a major concern on the Internet, service providers are increasingly deploying Internet traffic control devices such as Web switches in redundant configurations. Traditionally, these configurations have been *hot-standby* configurations, where one switch is active and the other is in a standby mode. A typical hot-standby configuration is shown in the figure below.

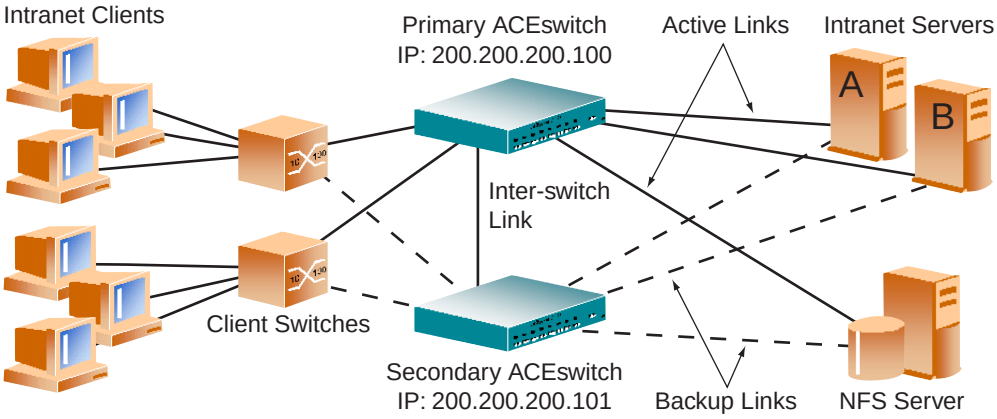


Figure 9-1 A Non-VRRP Hot-Standby Configuration

While hot-standby configurations increase site availability by removing single points of failure, service providers increasingly view them as an inefficient use of network resources because one functional web switch sits by idly until a failure calls it into action. Service providers now demand that vendors' equipment support redundant configurations where all devices can process traffic when they are healthy, increasing site throughput and decreasing user response times when no device has failed.

Alteon WebSystems' redundancy configurations are based on the Virtual Router Redundancy Protocol (VRRP) described in RFC 2338. Alteon WebSystems has developed extensions to VRRP that allow it to support Layer 4 switching services such as server load balancing, to support active operation of interfaces (at Layer 3) and services (at Layer 4) across multiple switches at the same time, and to interact with Spanning Tree to control frame path.

Alteon WebSystems switches support three approaches to providing high availability and redundancy: *active-standby*, *active-active*, and a new *hot-standby* configuration. Each is described below.

Active-Standby Redundancy

In an *active-standby configuration*, shown in Figure 9-2, both switches can support active traffic. However, services are not shared across the switches. That is, each switch can be active for some number of services, such as IP routing interfaces or load balancing VIP addresses, and act as a standby for other services on the other switch.

NOTE – In an active-standby configuration, the same service cannot be active simultaneously on both switches.

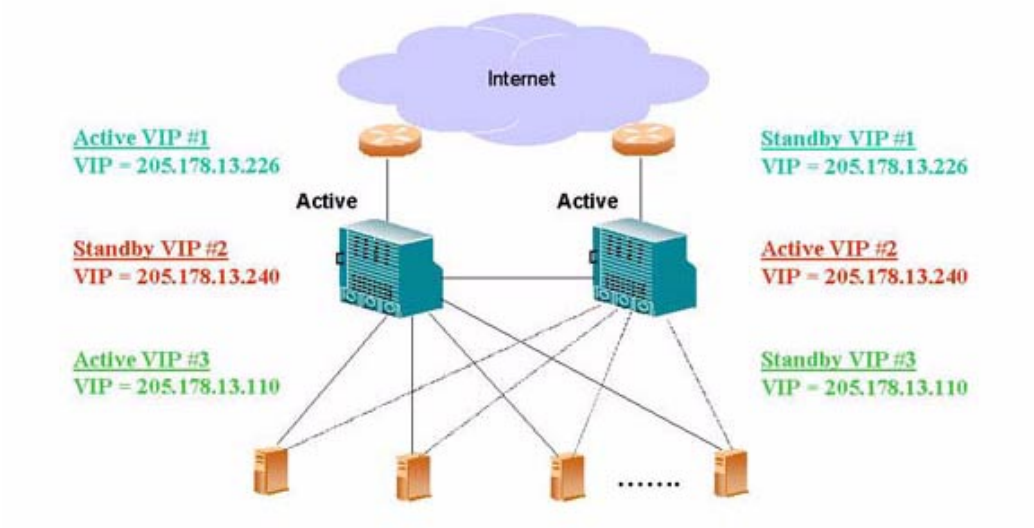


Figure 9-2 Active-Standby Redundancy

Active-Active Redundancy

In an *active-active* configuration, shown in Figure 9-3, both switches can process traffic for the same service at the same time. That is, both switches can be active simultaneously for a given IP routing interface or load balancing virtual server (VIP).

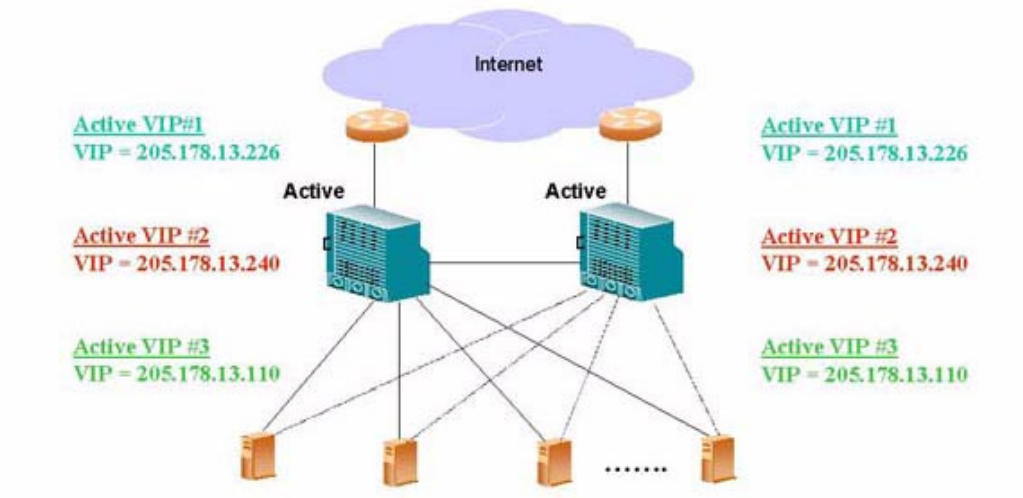


Figure 9-3 Active-Active Redundancy

In the above example, one switch is still the master router, but traffic going through the backup router (associated with the same virtual router on the switch) that is addressed to the master router will be intercepted and processed by the backup router.

Hot-Standby Redundancy

To provide as much flexibility as possible, the old hot-standby approach has been modified to eliminate the problems previously associated with it and is now based on VRRP. In a hot-standby configuration, two or more switches provide redundancy for each other. One switch is elected *master* and actively processes L4 traffic. The other switches, the backups, assume the master role should the master fail. The backups may forward L2 and L3 traffic as appropriate.

There are three components to the VRRP-based hot-standby model: the virtual router group, additional Layer 4 port states, and configuration synchronization options. Each is described below. The hot-standby model is shown in Figure 9-4.

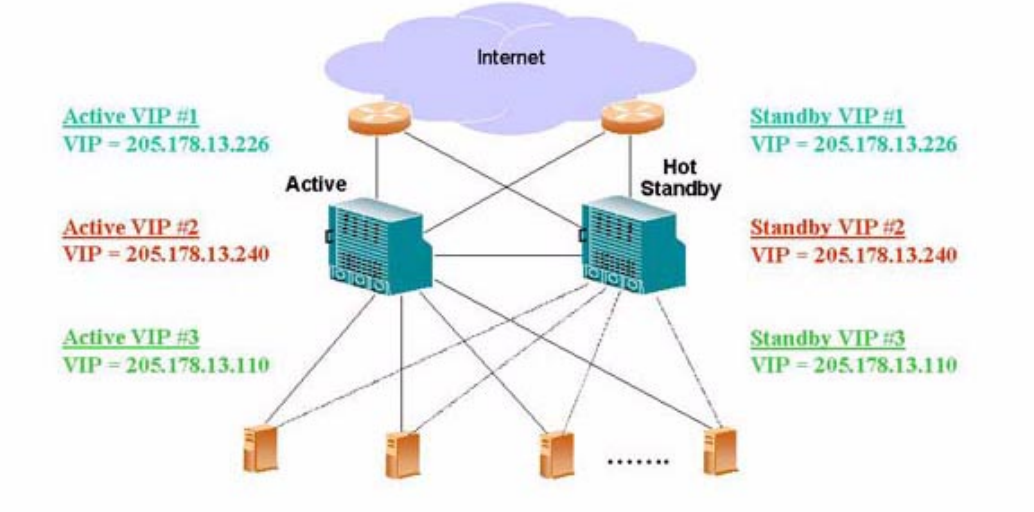


Figure 9-4 Hot Standby Redundancy

Virtual Router Group

The virtual router group ties all of the virtual routers together as a single entity and is central to the hot-standby configuration. All virtual routers on a given switch must all be either master or backup. They cannot failover individually, only as a group. Once hot-standby is globally enabled, the virtual route group must be enabled. The virtual router group aggregates all of the virtual routers as a single entity, meaning all virtual routers share the same state, master or backup. They cannot transitions from master to backup or vice versa individually, only as a whole.

If the virtual router group is master on one switch, it means the switch is master. Else the switch is backup. However, L4 processing is still enabled. If a virtual server is not a virtual router, the backup switch can still process traffic addresses to that VIP.

Filtering is also still functional. Only traffic addressed to virtual server routers is not processed.

VRRP actually contains support for virtual router groups. Each advertisement is not limited to a single virtual router IP address and can include up to 256 addresses. This means, all virtual routers are advertised in the same packet, conserving processing and buffering resources. However, the advertisements are also used to help bridges learn the virtual router MAC address. Since all of the virtual routers can have different VRIDs, we must rotate the MAC SA of the advertisement to ensure that the bridges learn all of the virtual router MAC addresses.

Hot-Standby and Inter-Switch Port States

The second part of the solution involves introducing two additional L4 port states, hot-standby and inter-switch. The forwarding states of hot-standby ports are controlled much like the forwarding states of the old hot-standby approach. Enabling hot-standby on a switch port allows the hot-standby algorithm to control the forwarding state of the port. If a switch is master, the forwarding states of the hot-standby ports are enabled. If a switch is backup, the hot-standby ports are blocked from forwarding or receiving traffic.

All hot-standby ports must have link before the virtual router group's priority can be increased from the configured value. It is automatically incremented by the "track other virtual routers" value when all hot-standby ports have link. This action allows the switches to failover when a hot-standby port loses link. Other enabled tracking features only have affect once all hot-standby ports on a switch have link.

NOTE – The VRRP hot-standby approach does NOT support single-link failover. If one hot-standby port loses link, the entire switch must become master to eliminate loss of connectivity.

The forwarding states of non-hot-standby ports are not controlled via the hot-standby algorithm. This allows the additional ports on the switches to provide added port density. The client ports on both switches should be able to process or forward traffic to the master switch.

The inter-switch port state is only a place holder. Its presence forces the user to configure a inter-switch link when hot-standby is globally enabled and prohibit the inter-switch link from also being a hot-standby link for VRRP advertisements. They must be able to reach the backup switch.

Configuration Synchronization

The final piece in configuring a high availability solution includes the addition of synchronization options to simplify the manual configuration synchronization. Configuration options have been added to refine what is synchronized and to whom and to disable synchronizing certain configurations. These include proxy IP addresses, L4 port configuration, filter configuration, and virtual router priorities.

Also, a peer menu has been added to allow the user to configure the IP addresses of the switches which should be synchronized. This provides added synchronization validation and allows the users to not have to enter the IP address of the redundant switch for each synchronization.

VRRP Overview

To give you the background necessary to understand the operation of Alteon WebSystems' redundancy configurations, this section describes VRRP operation and the Alteon-specific extensions to VRRP.

Virtual Router Redundancy Protocol (VRRP) enables redundant router configurations within a LAN. It provides alternate router paths for a host to eliminate single points of failure within a network. The router associated with a given alternate path supported by VRRP uses the same IP address and MAC address as the routers for other paths. As a result, the host's gateway information does not change, no matter what path is used. VRRP-based redundancy significantly reduces administrative overhead when compared to redundancy schemes that require hosts to be configured with multiple default gateways.

VRRP Components

Each physical router running VRRP is known as a *VRRP Router*. Two or more VRRP Routers can be configured to form a *Virtual Interface Router*. (RFC 2338 calls this entity a "Virtual Router.") In this guide, the term Virtual Interface Router is used to distinguish this type of entity from a *Virtual Server Router*, which is described in "Alteon Extensions to VRRP" on page 9-184. When the term Virtual Router is used herein, the concept applies to both Virtual Interface Routers and Virtual Server Routers. Each VRRP Router may participate in one or more Virtual Interface Routers.

A Virtual Interface Router acts as a default or next hop gateway for hosts on a LAN. Each Virtual Interface Router consists of a user-configured *Virtual Router Identifier* (VRID) and an IP address.

The VRID is used to build the *Virtual Router MAC Address*. The five highest order octets of the Virtual Router MAC Address are the standard MAC prefix (00-00-5E-00-01) defined in RFC 2338. The VRID is used to form the lowest order octet.

One, but not more than one of the VRRP Routers in a Virtual Interface Router may be configured as the IP Address Owner. This router has the Virtual Interface Router's IP address as its real interface address. This router, when up, responds to packets addressed to the Virtual Interface Router's IP address for ICMP pings, TCP connections, and so on.

There is no requirement for any VRRP Router to be the IP Address Owner. Most VRRP installations choose not to implement an IP Address Owner. For the purposes of this chapter, VRRP Routers that are not the IP Address Owner are called *Renters*.

Within each Virtual Router, one of the VRRP routers is selected to be the Virtual Router Master. See "Determining Which VRRP Router Is the Master" on page 9-182 for an explanation of the selection process.

NOTE – If the IP Address Owner is available, it will always become the Virtual Router Master.

The Virtual Router Master forwards packets sent to the Virtual Interface Router. It also responds to ARP requests sent to the Virtual Interface Router's IP address. Finally, it sends out periodic advertisements, to let other VRRP Routers know it is alive and it's priority (explained below).

Within a Virtual Router, the VRRP routers not selected to be the Master are known as Virtual Router Backups. Should the Virtual Router Master fail, one of the Virtual Router Backups becomes the Master and assumes its responsibilities.

The above points are illustrated in Figure 9-5. The Alteon switches in the diagram have been configured as VRRP Routers. They form a Virtual Interface Router.

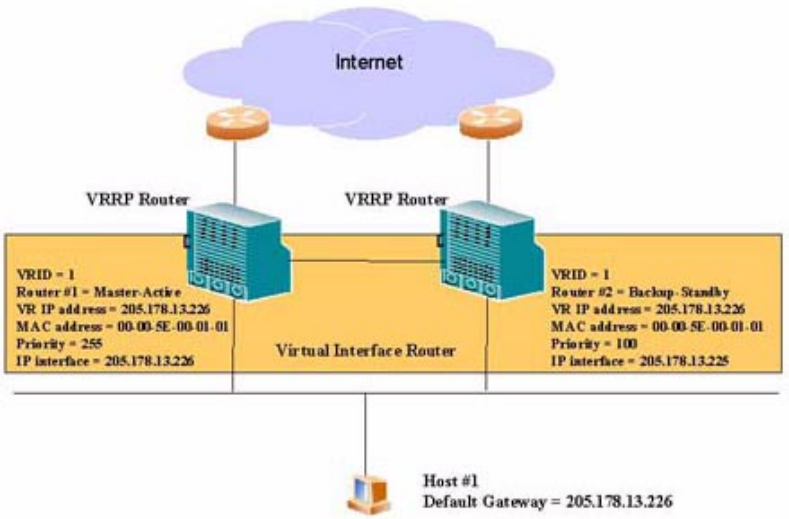


Figure 9-5 VRRP Router Example 1

The switch on the left has its real interface configured with the IP address of the Virtual Interface Router and is therefore the IP Address Owner. As a result, it is also the Virtual Router Master. The switch on the right of the figure is a Virtual Router Backup. Its real interface is configured with an IP address that is on the same subnet as that of the Virtual Interface Router but that is not the IP address of the Virtual Interface Router.

The Virtual Interface Router has been assigned a VRID = 1. Therefore, both of the VRRP Routers have a MAC address = 00-00-5E-00-01-01.

VRRP Operation

The host shown in Figure 9-5 is configured with the Virtual Interface Router's IP address as its default gateway. The Master forwards packets destined to remote subnets and responds to ARP requests. Since, in this example, the Master is also the Virtual Interface Router's IP Address Owner, it also responds to ICMP ping requests and IP datagrams destined for the Virtual Interface Router's IP address. The Backup does not forward any traffic on behalf of the Virtual Interface Router, nor does it respond to ARP requests.

If the Owner is not available, the Backup becomes the Master and takes over responsibility for packet forwarding and responding to ARP requests. However, since this switch is not the Owner, it does not have a real interface configured with the Virtual Interface Router's IP address.

Determining Which VRRP Router Is the Master

Each VRRP Router that is not an Owner is configured with a priority between 1 - 254. Per the VRRP standard, an Owner has a priority = 255. A bidding process determines which VRRP Router is or becomes the Master; that is the VRRP Router with the highest priority. Since Owners have a priority higher than the range permitted for non-Owners, the IP Address Owner, if any, is always the Master for the Virtual Interface Router, as long as it is available.

The Master periodically sends out advertisements to an IP multicast address. As long as the Backups receive these advertisements, they remain in the backup state. If a Backup does not receive an advertisement for three advertisement intervals, it initiates a bidding process to determine which VRRP router has the highest priority. That VRRP router then takes over as Master.

A backup router can stop receiving advertisements for one of two reasons - the Master can be down, or all communications links between the Master and the Backup can be down. If the Master has failed, it is clearly desirable for the Backup (or one of the Backups, if there's more than one) to become the Master.

NOTE – If the Master is healthy but communication between it and the Backup has failed, then there will be two Masters within the Virtual Router. To prevent this from happening, it is strongly recommended that redundant links be used between the switches that form a Virtual Router.

If, at any time, a Backup router determines that it has higher priority than the current Master does, it can preempt the Master, unless it is configured not to do so. In preemption, the Backup begins to send its own advertisements. The current Master will see that the Backup has higher priority and stop functioning as the Master. The Backup will then see that the Master has stopped sending advertisements and assume the role of Master.

Active-Standby Failover

The text above describes using a group of VRRP routers to form a single Virtual Interface Router. It implements a traditional hot-standby configuration. VRRP can also be used to implement active-standby configurations. In the example shown in Figure 9-6, the switch on the left is Master for the Virtual Interface Router with VRID = 1 and Backup for the Virtual Interface Router with VRID = 2. The switch on the right is Master for the Virtual Interface Router with VRID = 2 and Backup for the Virtual Interface Router with VRID = 1. In this manner, both routers can actively forward traffic at the same time, but not for the same interface.

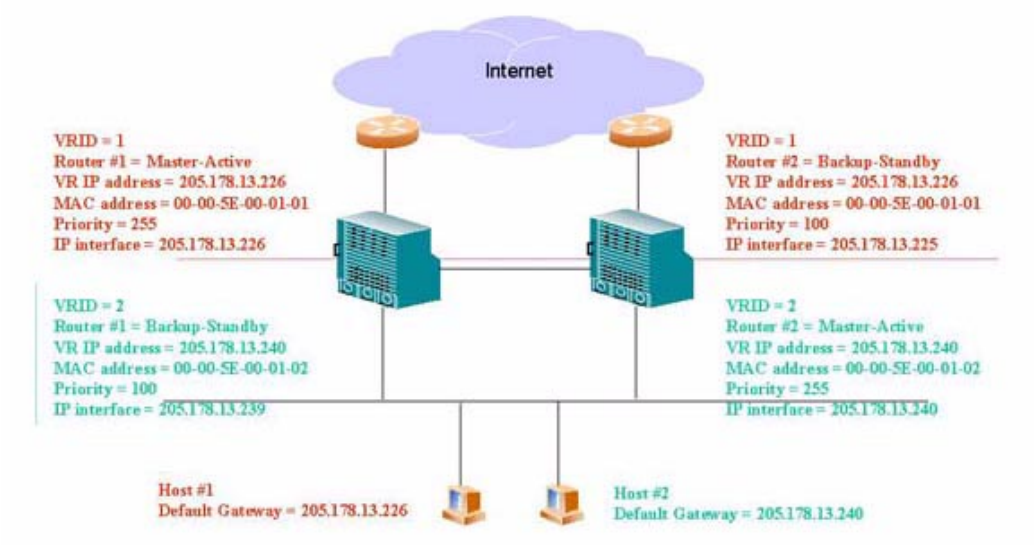


Figure 9-6 VRRP Router Example 2

Alteon Extensions to VRRP

This section describes enhancements to VRRP that are implemented in WebOS.

Virtual Server Routers

WebOS supports *Virtual Server Routers*. They extend the benefits of VRRP to VIP addresses used to perform server load balancing.

Virtual Server Routers operate for VIP addresses in much the same manner as Virtual Interface Routers operate for IP interfaces. A Master is negotiated via a bidding process, during which information about each VRRP Router's priority is exchanged. Only the Master processes packets destined for the VIP address and responds to ARP requests. The Master sends periodic advertisements. If a Backup does not receive an advertisement within a specified period, it initiates the bidding process to determine which VRRP Router takes over as Master. If, at any time, a Backup determines that it has higher priority than the current Master does, it can preempt the Master and become the Master itself, unless configured not to do so.

One difference between Virtual Server Routers and Virtual Interface Routers is that the concept of an IP Address Owner does not apply to Virtual Server Routers. All Virtual Server Routers are Renters. For a Virtual Server Router, the Master always responds to ICMP ping requests if *sharing* (see below) is not enabled. If sharing is enabled, the switch where the ping request initially enters the Virtual Server Router responds.

All Virtual Routers, whether Virtual Server Routers or Virtual Interface Routers, operate independently of one another; that is, their priority assignments, advertisements and Master negotiations are separate. For example, when you configure a VRRP Router's priority in a Virtual Server Router, you are not affecting that VRRP Router's priority in any Virtual Interface Router or another other Virtual Server Router of which it is a part. However, because of the requirement that MAC addresses be unique on a LAN, VRIDs must be unique among all Virtual Routers, whether Virtual Interface Routers or Virtual Server Routers.

Sharing/Active-Active Failover

WebOS supports *sharing* of interfaces at both Layer 3 and Layer 4, as shown in Figure 9-7. With sharing, an IP interface or a VIP address can be active simultaneously on multiple switches, enabling active-active operation.

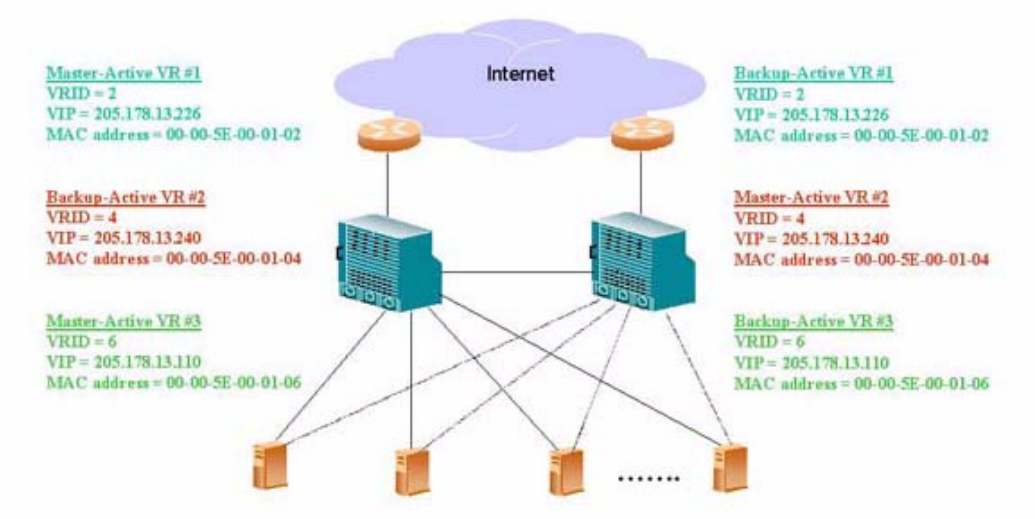


Figure 9-7 Active-Active High Availability

When sharing is used, incoming packets are processed by the switch on which they enter the Virtual Router. This is determined by external factors such as routing and Spanning Tree configuration.

NOTE – Sharing cannot be used in configurations where incoming packets have more than one entry point into the Virtual Router, such as instances where a hub is used to connect the switches.

When sharing is enabled, the Master election process still occurs. Although the process does not affect which switch processes packets that must be routed or that are destined for the VIP address, it does determine which switch sends advertisements and responds to ARPs sent to the Virtual Router's IP address.

Alteon WebSystems strongly recommends that sharing, rather than active-standby configurations, be used whenever possible. Sharing offers both better performance and fewer service interruptions in the face of fault conditions than active-standby configurations.

Tracking

WebOS supports a tracking function that dynamically modifies the priority of a VRRP Router, based on its current state. The objective of tracking is to have, whenever possible, the Master bidding processes for various Virtual Routers in a LAN converge on the same switch and to ensure that the selected switch is the one that offers optimal network performance. For tracking to have any effect on Virtual Router operation, preemption must be enabled.

Tracking only affects hot standby and active-standby configurations. It does not have any effect when sharing; that is, active-active, configurations are used.

WebOS can track the attributes listed in Table 9-1:

Table 9-1 VRRP Tracked Parameters

Parameter	Description
Number of Virtual Routers in Master mode on the switch	This is useful for making sure that traffic for any particular client/server pair is handled by the same switch, increasing routing and load balancing efficiency. This parameter influences the VRRP Router's priority in both Virtual Interface Routers and Virtual Server Routers.
Number of IP interfaces active on the switch	An IP interface is considered active when there is at least one active port on the same VLAN. This helps elect the Virtual Routers with the most available routes as the Master. This parameter influences the VRRP Router's priority in both Virtual Interface Routers and Virtual Server Routers.
Number of active ports on the same VLAN	This helps elect the Virtual Routers with the most available ports as the Master. This parameter influences the VRRP Router's priority in both Virtual Interface Routers and Virtual Server Routers.
Number of physical switch ports that have active Layer 4 processing on this switch	This helps elect the main Layer 4 switch as the Master. This parameter influences the VRRP Router's priority in both Virtual Interface Routers and Virtual Server Routers.

Table 9-1 VRRP Tracked Parameters

Parameter	Description
Number of healthy real servers behind the VIP address that is the same as the IP address of the Virtual Server Router on the switch	This helps elect the switch with the largest server pool as the Master, increasing Layer 4 efficiency. This parameter influences the VRRP Router's priority in Virtual Server Routers only.
In networks where the Hot Standby Router Protocol (HSRP) is used for establishing router failover, the number of Layer 4 client-only ports that receive HSRP advertisements	This helps elect the switch closest to the master HSRP router as the Master, optimizing routing efficiency. This parameter influences the VRRP Router's priority in both Virtual Interface Routers and Virtual Server Routers.

Each tracked parameter has a user-configurable weight associated with it. As the count associated with each tracked item increases (or decreases), so does the VRRP Router's priority, subject to the weighting associated with each tracked item. If the priority level of a Backup is greater than that of the current Master, then the Backup can assume the role of the Master.

Redundancy Configurations

Alteon WebSystems switches offer flexibility in implementing redundant configurations. This section discusses three of the more useful and easily deployed configurations. The first runs a Virtual Server Router in an active-standby configuration. The second runs one Virtual Server Router, employing an active-active configuration.

Active-Standby Virtual Server Router Configuration

Figure 9-8 shows an example configuration where two Alteon web switches are used as VRRP Routers in an active-standby configuration, implementing a Virtual Server Router. Active-standby redundancy should be used in configurations that cannot support sharing, that is, configurations where incoming packets will be seen by more than one switch, such as instances where a hub is used to connect the switches. In this configuration, when both switches are healthy, only the Master responds to packets sent to the VIP.

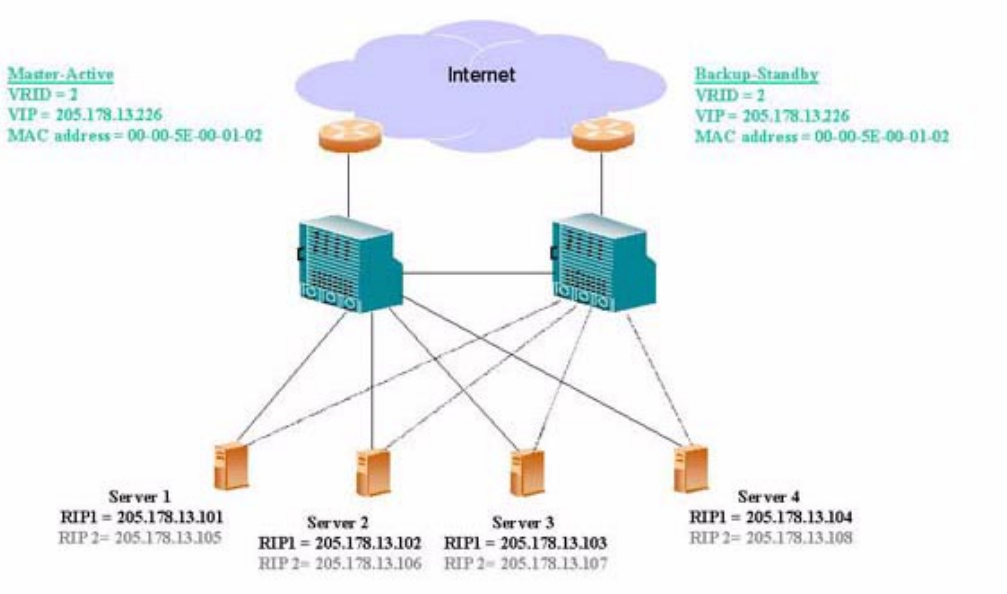


Figure 9-8 Active-Standby High Availability Configuration

Although this example shows only two switches, there is no limitation on the number of switches used in a redundant configuration. It's possible to implement an active-standby configuration across all the VRRP-capable switches in a LAN.

As in the active-active example, each VRRP-capable switch in an active-standby configuration is autonomous. Switches in a Virtual Router need not be identically configured.

To implement the active-standby example, perform the following switch configuration:

1. **Configure the appropriate Layer 2 and Layer 3 parameters on both switches.**

   This includes any required VLANs, IP interfaces, default gateways, and so on. If IP interfaces are configured, none of them should use the VIP address described in Step 4.

2. **Define all needed filters.**

   With WebOS release 5.2.13, this must be done on both switches. For later WebOS releases, the filters may be configured on one switch and pushed to the other switch in Step 5.

3. **Configure all required SLB parameters on one of the switches.**

   For the purposes of this example, assume that the switch on the left is configured in this step. Required Layer 4 parameters include a VIP = 205.178.13.226 and one real server group with four real servers, RIP = 205.178.13.101, RIP = 205.178.13.102, RIP = 205.178.13.103 and RIP = 205.178.13.104.

4. **Configure the VRRP parameters on the switch.**

   This includes the VRID = 2, the VIP = 205.178.13.226 and the priority. Enable tracking and set the parameters appropriately (refer to "Configuring Tracking" on page 203). Make sure to disable sharing.

5. **Synchronize the SLB and VRRP configurations, by pushing the configuration from the switch on the left to the one on the right.**

   Use the **`/oper/slb/sync`** command.

6. **Change the real servers in the right-hand switch's configuration to RIP = 205.178.13.105, RIP = 205.178.13.106, RIP =205.178.13.107 and RIP = 205.178.13.108.**

   Adjust the right-hand switch's priority appropriately (see "Configuring Tracking" on page 203).

   In this example, with the left-hand switch as the Master, if a link between the left-hand switch and a server fails, the server will fail health checks and be taken out of the load-balancing algorithm. Assuming that tracking is enabled and is configured to take into account the number of healthy real servers for the Virtual Router's VIP address, the left-hand switch's priority will be reduced. If it is reduced to a value lower than the right-hand switch's priority, the right-hand switch will assume the role of Master.

NOTE – In this case, all active connections being serviced by the left-hand switch's VIP will be severed.

If the link between the left-hand (Master) switch and its Internet router fails, the protocol used to distribute traffic between the routers, for example OSPF, will reroute traffic to the other router. The right-hand (Backup) switch will act as a Layer 2/3 switch and forward all traffic destined to the VIP to the left-hand switch.

If the entire left-hand (Master) switch fails, the protocol used to distribute traffic between the routers, such as OSPF, will reroute traffic to the right-hand router. The right-hand (Backup) switch/router will detect that the Master has failed because it will stop receiving advertisements. The Backup will than assume the Master's responsibility of responding to ARPs and issuing advertisements.

Active-Active VIR and VSR Configuration

Figure 9-9 shows an example configuration where two Alteon web switches are used as VRRP Routers in an active-active configuration implementing a Virtual Server Router. As noted earlier, this is the preferred redundant configuration.

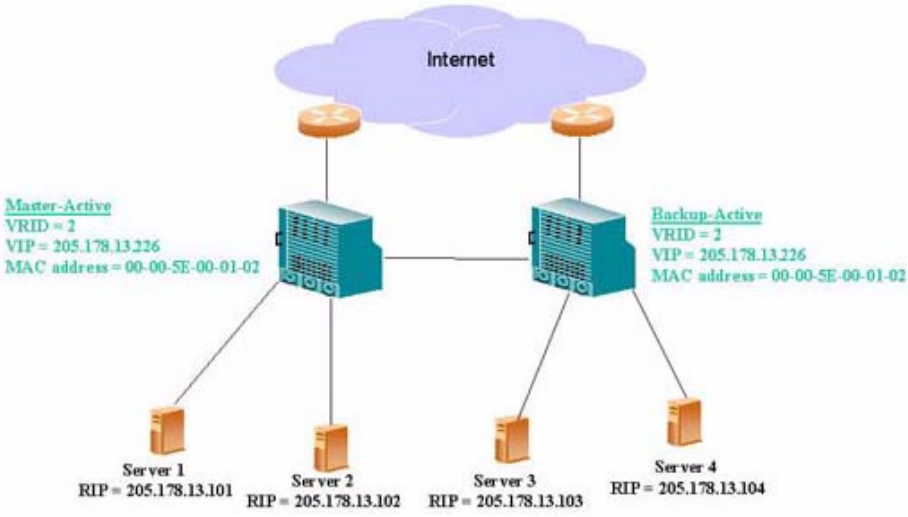


Figure 9-9 Active-Active High Availability Configuration

Although this example shows only two switches, there is no limitation on the number of switches used in a redundant configuration. It's possible to implement an active-active configuration and perform load sharing between all of the VRRP-capable switches in a LAN.

In this configuration, when both switches are healthy, both load balance packets are sent to the VIP. This results in higher capacity and performance than if the switches were used in an active-standby configuration.

The switch on which a frame enters the Virtual Server Router is the one that processes that frame. The ingress switch is determined by external factors such as routing and Spanning Tree configuration.

NOTE – Each VRRP-capable switch is autonomous. There is no requirement that the switches in a Virtual Router be identically configured. Different switch models with different numbers of ports and different enabled services may be used in a Virtual Router.

To implement this example, perform the following switch configuration procedure:

1. **Configure the appropriate Layer 2 and Layer 3 parameters on both switches.**

   This includes any required VLANs, IP interfaces, default gateways, and so on. If IP interfaces are configured, none of them should use the VIP address described in Step 4.

2. **Define all needed filters.**

   With WebOS release 5.2.13, this must be done on both switches. For later WebOS releases, the filters may be configured on one switch and pushed to the other switch in Step 5.

3. **Configure all required SLB parameters on one of the switches.**

   For the purposes of this example, assume that the switch on the left (in Figure 9-9) is configured in this step. Required Layer 4 parameters include a VIP = 205.178.13.226 and one real server group with two real servers, RIP = 205.178.13.101 and RIP = 205.178.13.102. RIP = 205.178.13.103 should be configured as a backup server to RIP = 205.178.13.101 and RIP = 205.178.13.104 should be configured as a backup server to RIP = 205.178.13.102.

NOTE – In this configuration, each server's backup is attached to the other switch. This ensures that operation will continue if all of the servers attached to a switch fail.

---

4. **Configure the VRRP parameters on the switch.**

   This includes the VRID = 2, the VIP address = 205.178.13.226 and the priority. Make sure to enable sharing.

5. **Synchronize the SLB and VRRP configurations by pushing the configuration from the switch on the left to the one on the right.**

   Use the **`/oper/slb/sync`** command.

6. **Reverse the roles of the real servers and their backups in the right switch's configuration.**

   Make RIP = 205.178.13.103 and RIP= 205.178.13.104 the real servers and RIP = 205.178.13.101 and RIP = 205.178.13.102 their backups, respectively.

   In this configuration, if a link between a switch and a server fails, the server will fail health checks and its backup (attached to the other switch) will be brought online. If a link between a switch and its Internet router fails, the protocol used to distribute traffic between the routers, for example, OSPF, will reroute traffic to the other router. Since all traffic now enters the Virtual Server Router on one switch, that switch will process all incoming connections.

   The same thing happens if an entire switch fails. If an entire switch fails and that switch is a Master, the Backup will detect this fact because it will stop receiving advertisements. In this case, the Backup will assume the Master's responsibility of responding to ARPs and issuing advertisements.

   You should think carefully before setting maxconns in this configuration. Information about maxconns is not shared between switches. Therefore, if a server is used for normal operation by one switch and is activated simultaneously as a backup by the other switch, the total number of possible connections to that server will be the sum of the maxconns limits for that server on both switches.

Active/Active Server Load Balancing Configuration

In this example, you will set up 4 virtual servers (VIPs), each load balancing 2 servers providing 1 service (for example, HTTP) per VIP.

You will be load balancing HTTP, HTTP-S, POP3, SMTP, and FTP. Each protocol will be load balanced via a different VIP. You could load balance all of these services on one VIP, but in this example we will use four distinct VIPs to illustrate the benefits of active/active failover. We'll set up one switch, dump out the configuration script (also called a text dump), edit it, and dump the configuration into the peer switch.

NOTE – Configuring the switch for active-active failover should take no longer than 15 minutes to complete. You can use either the WebOS Browser-Based Interface (BBI) or the Command Line Interface (CLI) for configuration.

Procedure #1: Background Configuration

1. **Define the IP interfaces.**

The switch will need an IP interface for each subnet that it will be connected to, so it can communicate with devices attached to it. Each interface will need to be placed in the appropriate VLAN. In our example, Interfaces #1, 2, 3, and 4 will be in VLAN 2 and Interface #5 will be in VLAN 1.

NOTE – On Alteon WebSystems switches, you are not restricted to configuring only one subnet per VLAN.

To configure the IP interfaces for this example, enter the following commands from the CLI:

```
>> Main# /cfg/ip/if 1                    (Select IP interface #1)
>> IP Interface 1# addr 10.10.10.10      (Assign IP address for the interface)
>> IP Interface 1# vlan 2                (Assign IP address for the interface)
>> IP Interface 1# ena                   (Enable IP interface #1)
```

Repeat this sequence of commands for each interface listed below:

- IF #1 10.10.10.10
- IF #2 20.10.10.10
- IF #3 30.10.10.10
- IF #4 40.10.10.10
- IF #5 200.1.1.10

2. **Define the VLANs.**

   In this configuration, we need to set up 2 VLANs: One for the outside world (the ports connected to the upstream switches - towards the routers) and one for the inside (the ports connected to the downstream switches - towards the servers).

```
>> Main# /cfg/vlan <VLAN-number>          (Globally disable STP)
>> vlan 1# add <port-number>              (Add a port to the VLAN membership)
>> vlan 1# ena                            (Enable VLAN #1)
```

   Repeat this command for the second VLAN.

   - VLAN #1 - IF #5 - physical ports connected to upstream switches have membership.
   - VLAN #2 - IFs #1,2,3,4 - physical ports connected to downstream switches have membership

3. **Disable Spanning Tree.**

   Spanning Tree can be turned off in this configuration. Reboot the switch to turn Spanning Tree off after disabling it using the following command:

```
>> Main# /cfg/stp off                     (Globally disable STP)
```

4. **Enable IP Forwarding.**

   You need to enable IP forwarding if the VIP and RIPs are on different subnets or if the switch is connected to different subnets and those subnets need to communicate through the switch (which they almost always do). If you are in doubt as to whether to enable IP forwarding or not, enable it. In our example, the VIP and RIPs are indeed on different subnets so enable this feature using the following command:

```
>> Main# /cfg/ip fwrd on                  (Enable IP forwarding)
```

Procedure #2: Server Load Balancing Configuration

1. **Define the Real Servers**

   The RIPs are defined and put into 4 groups, depending on the service they are running. Notice that RIPs #7 and 8 are on routable subnets. This is done to support passive FTP.

   For each real server, you must assign a real server number, specify its actual IP address, and enable the real server. For example:

```
>> Main# /cfg/slb/real 1                    (Server A is real server 1)
>> Real server 1 # rip 10.10.10.5/24        (Assign Server A IP address)
>> Real server 1 # ena                      (Enable real server 1)
```

   Repeat this sequence of commands for the following real servers:

   - RIP #1 10.10.10.5/24
   - RIP #2 10.10.10.6/24
   - RIP #3 20.10.10.5/24
   - RIP #4 20.10.10.6/24
   - RIP #5 30.10.10.5/24
   - RIP #6 30.10.10.6/24
   - RIP #7 200.1.1.5/24
   - RIP #8 200.1.1.6/24

2. **Define the Real Server Groups, adding the appropriate real servers.**

   This combines the three real servers into one service group:

```
>> Real server 8 # /cfg/slb/group 1         (Select real server group 1)
>> Real server group 1# add 1               (Add real server 1 to group 1)
>> Real server group 1# add 2               (Add real server 2 to group 1)
```

   Repeat this sequence of commands for the following real server groups:

   - Group #1 – Add RIP #1 and #2
   - Group #2 – Add RIP #3 and #4
   - Group #3 – Add RIP #5 and #6
   - Group #4 – Add RIP #7 and #8

3. **Define the virtual servers.**

After defining the VIPs and associating them with a real server group number, you must tell the switch which IP ports/services/sockets you want to load balance on each VIP.

```
>> Real server group 4 # /cfg/slb/virt 1     (Select virtual server 1)
>> Virtual server 1# vip 200.200.200.100     (Assign a virtual server IP address)
>> Virtual Server 1# service 80
>> Virtual server 1 http Service#  group 1   (Associate virtual port to real group)
>> Virtual server 1# ena                     (Enable the virtual server)
```

Repeat this sequence of commands for the following virtual servers:

- VIP #1 200.200.200.100 Will Load Balance HTTP (Port 80) to Group 1
- VIP #2 200.200.200.101 Will Load Balance HTTP-S (Port 443) to Group 2
- VIP #3 200.200.200.102 Will Load Balance POP/SMTP (Ports 110/25) to Group 3
- VIP #4 200.200.200.104 Will Load Balance FTP (Ports 20/21) to Group 4

4. **Define the client and server port states.**

Defining a client port states tells that port to be on the lookout for any frames destined for the VIP and load balance them if they are destined for a load balanced service. Defining a server port state tells the port to the do the re-mapping (NAT'ing) of the RIP back to the VIP. Note the following:

- The ports connected to the upstream switches (the ones connected to the routers) will need to be client port state.
- The ports connected to the downstream switches (the ones providing fan out for the servers) will need to be server port state.

Configure the ports using the following sequence of commands:

```
>> Virtual server 4# /cfg/slb/port 1      (Select physical switch port 1)
>> SLB port 1# client ena                 (Enable client processing on port 1)
>> SLB port 1# ../port 2                  (Select physical switch port 2)
>> SLB port 2# server ena                 (Enable server processing on port 2)
```

Procedure #3: Virtual Router Redundancy Configuration

1. **Configure Virtual Routers 2, 4, 6, and 8.**

   These virtual routers will have the same IP addresses as the VIP's. This is what tells the switch that these are Virtual Service Routers (VSRs). The reason we use even numbers is that the VRIDs on VSRs need to be even with Layer 3 bindings disabled (default setting). The only time you might enable layer bindings is if you are doing UDP load balancing. In this case, the VSR's VRID would need to be an odd number. In this example, Layer 3 bindings are left in their default configuration which is disabled so the VRIDs need to be even.

   Configure a virtual router using the following sequence of commands:

```
>> Virtual server 4# /cfg/vrrp/vr 2          (Select virtual router 2)
>> Virtual router 2 vrid 2                   (Set virtual router ID)
>> Virtual router 2 addr 200.200.200.100     (Assign virtual router IP address)
>> Virtual router 2 if 5                     (Assign virtual router interface)
>> Virtual router 2 ena                      (Enable virtual router 2)
```

   Repeat this sequence of commands for the following virtual routers:

   - VR #2 - VRID 2 - IF #5 (associate with IP interface #5) – Address 200.200.200.100
   - VR #4 - VRID 4 - IF #5 (associate with IP interface #5) – Address 200.200.200.101
   - VR #6 - VRID 6 - IF #5 (associate with IP interface #5) – Address 200.200.200.103
   - VR #8 - VRID 8 - IF #5 (associate with IP interface #5) – Address 200.200.200.104

2. **Configure Virtual Routers 1, 3, 5, and 7.**

   These virtual routers will act as the default gateways for the servers on each respective subnet. Because these Virtual Routers are survivable next hop/default gateways, they are called VIR's. There is no restriction as to the VRID number on a VIR.

   Configure each virtual router listed below, using the sequence of commands in Step 1.

   - VR #1 - VRID 1 - IF #1 (associate with IP interface #1) – Address 10.10.10.1
   - VR #3 - VRID 3 - IF #2 (associate with IP interface #2) – Address 20.10.10.1
   - VR #5 - VRID 5 - IF #3 (associate with IP interface #3) – Address 30.10.10.1
   - VR #7 - VRID 7 - IF #4 (associate with IP interface #4) – Address 40.10.10.1

3. **Set the renter priority for each virtual router.**

   We want Switch #1 to be the Master router, so we need to bump the default virtual router priorities, which are 100, to 101 on virtual routers 1-4 to force this switch to be the Master for these virtual routers.

   Use the following sequence of commands:

```
>> Virtual server 4# /cfg/vrrp/vr 1          (Select virtual router 1)
>> Virtual router 1 prio 101                 (Set virtual router priority)
```

   Apply this sequence of commands to the following virtual routers, assigning each a priority of 101:

   - VR #1 - Priority 101
   - VR #2 - Priority 101
   - VR #3 - Priority 101
   - VR #4 - Priority 101

4. **Configure priority tracking parameters for each Virtual Router.**

   For this example, the best parameter(s) to track on is Layer 4 ports (l4pts).

   Use the following command:

```
>> Virtual server 4# /cfg/vrrp/vr 1/track l4pts(Set priority tracking
                                                parameter for virtual router 1,
                                                electing virtual router with most
                                                available ports as the master
                                                router.)
```

   Repeat this command for the following virtual routers:

   - VR #1 - Track l4pts
   - VR #2 - Track l4pts
   - VR #3 - Track l4pts
   - VR #4 - Track l4pts
   - VR #5 - Track l4pts
   - VR #6 - Track l4pts
   - VR #7 - Track l4pts
   - VR #8 - Track l4pts

   Switch #1 configuration is done.

Configuring Switch # 2

The procedure to dump the configuration script (text dump) out of Switch #1 is described below.

If you have been using the WebOS Browser-Based Interface (BBI) to configure the switch, you need a serial cable that is a DB-9 Male to DB-9 Female, straight-through (not a null modem) cable. Connect to the switch from your computer with the cable. Open HyperTerminal (or the terminal program of your choice) and connect to the switch using the following parameters: Baud: 9600, Data Bits: 8, Parity: None, Stop Bits:1, Flow Control: None. If you are using HyperTerminal, the only thing that needs to be changed from the default settings is the Baud Rate and Flow Control. Once you are connected to the switch, start logging your session in HyperTerminal (transfer/capture text). Save the file as "Customer Name" Switch #1, then type the following command in the switch command line interface, **`cfg/dump`**. A script will be dumped out. Stop logging your session (transfer/capture text/stop).

1. **Open the text file that you just created and change the following:**
   - Delete anything above "Script Start"
   - Delete the two lines directly below "Script Start." These two lines identify the switch from which the dump was taken, as well as giving the date and time. If we leave these two lines in, it will confuse Switch #2 when we dump in the file.
   - Change the last octet in all the IP interfaces from .10 to .11. Find this in line: `/cfg/ip/if 1/addr 10.10.10.10`. Simply delete the "0" and put in a "1." Be sure to do this for all the IP interfaces, otherwise we'll have duplicate IP addresses in the network.
2. **Change the Virtual Router priorities. Virtual Routers 1-4 need to have their priority set to 100 from 101 and Virtual Routers 5-7 need to have their priorities set to 101 from 100. You can find this in line** `/cfg/vrrp/vr 1/vrid 1/if 1/prio 101`**.**
3. **Scroll to the bottom of the text file and delete anything past "Script End."**
4. **Save the changes to the text file as "Customer Name" Switch #2.**

   Go to the second switch. If there is any configuration on it, delete it by setting it back to factory settings using the following command:

```
>> Main# /boot/conf factory/reset
```

   You can tell if the switch is at factory default when you log on to it because the switch will prompt you if you want to use the step by step configuration process. When it does this respond: "No."

5. **In HyperTerminal, go to transfer/send text file and send the Switch #2 text file. The configuration will dump into the switch. Simply type "apply", then "save" after that. When you can type characters in the terminal session again, reboot the switch (`/boot/reset`).**

Virtual Router Deployment Considerations

To prevent network problems when deploying Virtual Routers, you should review the issues described in this section.

Mixing Active-Standby and Active-Active Virtual Routers

If the network environment can support sharing, enable it for all Virtual Routers in the LAN. If not, use active-standby for all Virtual Routers. Do not mix active-active and active-standby Virtual Routers in a LAN. Mixed configurations have not been tested, may result in unexpected operational characteristics, and therefore are not recommended.

VRRP Active/Active Synchronization

The old hot-standby failover required the primary and secondary switches to have identical configurations and port topology. With VRRP and active/active failover, this is optional. Each switch can be configured individually with different port topology, server load balancing, and filters. If you would rather force two active/active switches to use identical settings, you can synchronize their configuration using the following command:

`/oper/slb/sync` *IP_address*

The `sync` command copies the following settings to the switch at the specified IP interface address:

- VRRP settings
- Server Load Balancing settings (including SLB port settings)
- Filter settings (including filter port settings)

If you perform the `sync` command, you should check the configuration on the target switch to ensure that the settings are correct.

NOTE – In WebOS version 5.2.21, the `sync` command also copies IP proxy settings to the target switch. This creates duplicate IP addresses on your network. To correct this problem, you must reconfigure each IP proxy on the target switch to use a unique IP address.

VRRP, STP, and Failover Response Time

VRRP active/active failover is significantly different from the hot-standby failover method supported in previous releases. As shown in Figure 9-10, active-active configurations can introduce loops into complex LAN topologies.

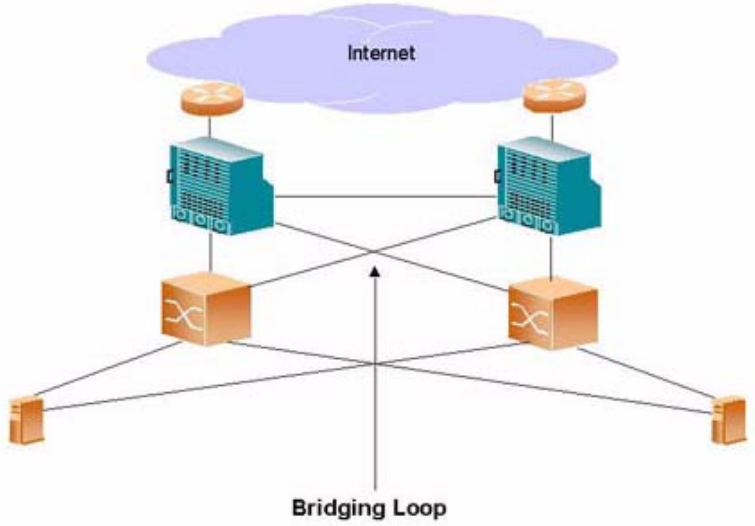


Figure 9-10 Loops in Active-Active Configuration

Using STP to Eliminate Loops

VRRP generally requires Spanning-Tree Protocol (STP) to be enabled, in order to resolve bridge loops that usually occur in cross-redundant topologies like the one shown in Figure 9-11. In this example, a number of loops are wired into the topology. STP resolves loops by blocking ports where looping is detected.

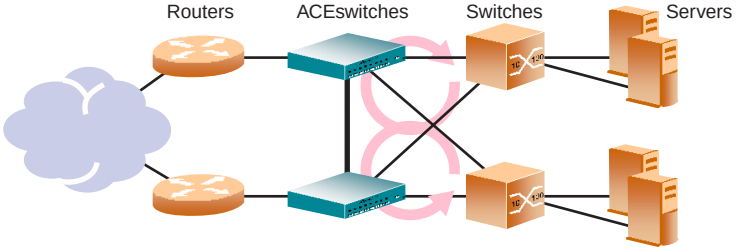


Figure 9-11 Cross-redundancy Creates Loops, but STP Resolves Them

One drawback to using STP with VRRP is the failover response time. STP could take as long as 45 seconds to re-establish alternate routes after a switch or link failure.

Using VLANs to Eliminate Loops

When using VRRP, you can decrease failover response time by using VLANs instead of STP to separate traffic into non-looping broadcast domains. An example is shown in Figure 9-12:

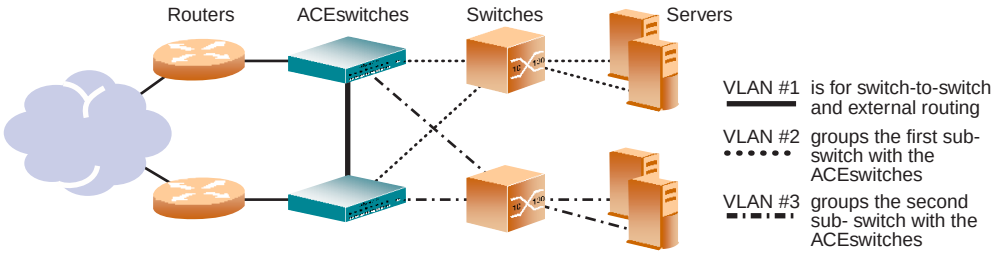


Figure 9-12 VLANs can be used to Create Non-looping Topologies.

This topology allows STP to be disabled. On the ACEswitches, IP routing allows traffic to cross VLAN boundaries. The servers use the ACEswitches as default gateways. For port failure, traffic is rerouted to the alternate path within one health-check interval (configurable between 1 and 60 seconds, with a default of 2 seconds).

VRRP Virtual Router ID Numbering

During the software upgrade process, VRRP virtual router IDs will be automatically assigned if failover is enabled on the switch. When configuring virtual routers at any point after upgrade, virtual router ID numbers (`/cfg/vrrp/vr #/vrid`) must be assigned in accordance with the following restrictions:

- The virtual router ID may be configured as ***any number*** between 1 and 255 when the virtual router IP address is not assigned the same value as a virtual server IP address.
- The virtual router ID must be configured as an ***odd number*** between 1 and 255 under the following circumstance:
  - The virtual router uses Layer 4 services (its virtual router IP address is the same as assigned to a virtual server) *and* ...
  - Layer 3 binding is turned on for the virtual server (by enabling the `layr3` option on the virtual server menu: `/cfg/slb/virt`
- The virtual router ID must be configured as an ***even number*** between 2 and 254 under the following circumstance:
  - The virtual router uses Layer 4 services (its virtual router IP address is the same as assigned to a virtual server) *and* ...
  - Layer 3 binding is turned off for the virtual server (by disabling the `layr3` option on the virtual server menu: `/cfg/slb/virt`

Configuring Tracking

Proper tracking configuration depends very much on user preferences and network environment. As a result, careful thought is required to properly implement tracking.

Consider the configuration shown in Figure 9-8 on page 188. Assume that the user wants the following behavior in their network:

- The switch on the left will be the Master router upon initial bring up.
- If the switch on the left is the Master and it has one active server fewer than the switch on the right, it remains the Master.
- The user wants this behavior because s/he believes running one server down is less disruptive than bringing a new Master online and severing all active connections in the process.
- If the switch on the left is the Master and it has two or more active servers fewer than the switch on the right, the switch on the right becomes the Master.
- If the switch on the right is the Master, it remains the Master even if servers are restored to the point on the left-hand switch where the left-hand switch has one fewer or an equal number of servers.
- If the switch on the right is the Master and it has one active server fewer than the switch on the left, the switch on the left becomes the Master.

The user can implement this behavior by configuring tracking as follows:

1. **Set the priority for the left-hand switch to the default value of 100.**
2. **Set the priority for the right-hand switch to 96.**
3. **On both switches, enable tracking based on the number of Virtual Routers in Master mode on the switch and set the value = 5.**
4. **On both switches, enable tracking based on the number of healthy real servers behind the VIP address that is the same as the IP address of the Virtual Server Router on the switch and set the value = 6.**

   Initially, the switch on the left will have a priority of 100 (base value) + 5 (since it will initially be the Master) + 24 (4 active real servers x 6 per real server) = 129. The switch on the right with have a priority of 96 (base value) + 24 (4 active real servers X 6 per real server) = 120.

   If one server attached to the left-hand switch fails, the left-hand switch's priority will be reduced by 6 to 123. Since 123 is greater than 120 (the right-hand switch's priority), the left-hand switch will remain the Master.

   If a second server attached to the left-hand switch fails, the left-hand switch's priority will be reduced by 6 more to 117. Since 117 is less than 120 (the right-hand switch's priority), the right-hand switch will become the Master. At this point, the left-hand switch's priority will fall by 5 more and the right-hand switch's will rise by 5 because the switches are tracking how many Masters they are running. So, the left-hand switch's priority will settle out at 112 and the right-hand switch's priority at 125.

When both servers are restored to the left-hand switch, that switch's priority will rise by 12 (2 healthy real servers X 6 per healthy server) to 124. Since 124 is less than 125, the right-hand switch will remain the Master.

If, at this point, a server fails on the right-hand switch, its priority will fall by 6 to 119. Since 119 is less than 124, the left-hand switch will become the Master. Its priority will settle out at 129 (since it's now the Master) while the right-hand switch's will drop by 5 more to 114.

We see from the above that the user's goals were met by the configured tracking parameters.

NOTE – There is no shortcut to setting tracking parameters. The goals must first be set and the outcomes of various configurations and scenarios analyzed to find settings that meet the goals.

Using the /oper/slb/sync Command

As noted above, each VRRP-capable switch is autonomous. Switches in a Virtual Router need not be identically configured. As a result, configurations cannot be synchronized automatically.

For user convenience, it is possible to push a configuration from one VRRP-capable switch to another using the /oper/slb/sync command. However, care must be taken when using this command to avoid unexpected results.

Using WebOS Release 5.2.13, all server load balancing and VRRP parameters can be pushed using the **`/oper/slb/sync`** command. Filter configurations are not pushed by this release. In later WebOS releases, both server load balancing and filter configurations are pushed.

Port specific parameters, such as what filters are applied and enabled on what ports, are part of what is pushed by the **`/oper/slb/sync`** command. As a result, if the **`/oper/slb/sync`** command is used, it is highly recommended that the hardware configurations and network connections of all switches in the Virtual Router be identical. That is, each switch should be the same model, have the same line cards in the same slots (if modular) and have the same ports connected to the same external network devices. Otherwise, unexpected (and unpleasant) results may occur if the **`/oper/slb/sync`** command attempts to configure a non-existent port or applies an inappropriate configuration to a port.

Using the /cfg/slb/sync Command

To synchronize the configuration between two switches, a peer must be configured on each switch. Switches being synchronized must use the same administrator password. Peers are sent SLB, FILT, and VRRP configuration updates using /**`oper/slb/synch`**.

CHAPTER 10

Secure Switch Management

To limit access to the switch's Management Processor without having to configure filters for each switch port, you can set a source IP address (or range) that will be allowed to connect to the switch IP interface through Telnet, SSH, SNMP, or the WebOS Web interface. This will also help prevent spoofing or attacks on the switch's TCP/IP stack.

The allowed management IP address range is configured using the system `mnet` and `mmask` options available on the command-line interface System Menu (`/cfg/sys`).

NOTE – The `mnet` and `mmask` commands in the `/cfg/slb` menu are used for a different purpose.

When an IP packet reaches the Management Processor, the source IP address is checked against the range of addresses defined by `mnet` and `mmask`. If the source address of the host or hosts are within this range, then they are allowed to attempt to log in. Any packet addressed to a switch IP interface with a source IP address outside this range is discarded silently.

Example: Assume that the `mnet` is set to 192.192.192.0, and the `mmask` is set to 255.255.255.128. This defines the following range of IP addresses: 192.192.192.0 to 192.192.192.127.

- A host with a source IP address of 192.192.192.21 falls within the defined range and would be allowed to access the switch Management Processor.
- A host with a source IP address of 192.192.192.192 falls outside the defined range and is not granted access. To make this source IP address valid, you would need to shift the host to an IP address within the valid range specified by the `mnet` and `mmask`, or modify the `mnet` to be 192.192.192.128 and the `mmask` to be 255.255.255.128. This would put the 192.192.192.192 host within the valid range allowed by the `mnet` and `mmask` (192.192.192.128-255).

NOTE – When the `mnet` and `mmask` Management Processor filter is applied, RIP updates received by the switch will be discarded if the source IP address of the RIP packet(s) falls outside the specified range. This can be corrected by configuring static routes.

Secure Switch Management

Secured switch management is needed for environments that perform significant management functions across the Internet. The following are some of the functions for secured management:

- Authentication of remote administrators

  Authentication is the action of determining who the administrator is; it usually involves a name and a password. The password can be either a fixed password or a challenge-response query.

- Authorization of remote administrators

  Once an administrator has been authenticated, authorization is the action of determining what that user is allowed to do. Authorization does not merely provide yes or no answers, but may also customize the service for a particular administrator.

- Encryption of management information exchanged between the remote administrator and the switch

  Examples of protocols to encrypt management information are SSH and SSL. WebOS supports the encryption of management information, using SSH, on AD4 and A184 Web switches.

Authentication and Authorization

NOTE – While authentication and authorization (AA) protocols and servers are designed to authenticate remote dial-up users, in addition to authorizing remote access capabilities to users, this overview is focussed on using the AA model to authenticate and authorize remote administrators for managing a switch.

The AA model is based on a client/server model. The "remote access server" (the switch) is a client to the back-end database server. A remote user (the remote administrator) interacts only with the remote access server, not the back-end server and database.

Two prominent "AA" protocols used to control dial-up access into networks are Cisco's TACACS+ (Terminal Access Controller Access Control System) and Livingston Enterprise's RADIUS (Remote Authentication Dial-In User Service). WebOS 8.0 supports only the RADIUS authentication method.

Components

The required components for authorization and authentication are listed below:

- A remote administrator
- The switch with authentication and authorization protocol support, acting as a client in the AA model
- A backend authentication and authorization server that performs the following functions:
  - Authenticates remote administrators
  - Checks the remote administrator's authorization to access the switch
  - Optionally, tracks and log the administrator's activity while logging on
- An AA (authorization and authorization) database that contains information about authorized administrators and their specific capabilities and privileges

Authorization and Authentication Procedure

The steps below describe the process that a remote administrator would go through to get authenticated and authorized for managing a switch, using the RADIUS AA protocol.

1. **Remote administrator connects to the switch and provides their user name and password.**
2. **Using the RADIUS protocol, the switch sends requests for authentication and authorization to the authentication/authorization server.**
3. **The authentication/authorization server checks the user name/password combination against its user ID database.**
4. **Using the RADIUS protocol, the authentication/authorization server instructs the switch to grant or deny the administrator access to the switch, based on its defined capabilities and privileges.**

RADIUS Authentication

RADIUS is an access server authentication, authorization, and accounting protocol, used to secure remote access to networks and network services against unauthorized access.

RADIUS is comprised of three components:

- A protocol with a frame format that utilizes UDP over IP: Based on RFC 2138
- A centralized server that stores all the user authorization information
- A client: in this case, the switch

The operation of RADIUS authentication and authorization protocol is similar to the "AA" model described above. The switch, acting as the RADIUS client, will communicate to the RADIUS server to authenticate and authorize a remote administrator using the protocol definitions specified in RFC 2138. Transactions between the client and RADIUS server are authenticated through the use of a shared secret, which is never sent over the network. In addition, the remote administrator passwords are sent encrypted between the RADIUS client (the switch) and the back-end RADIUS server.

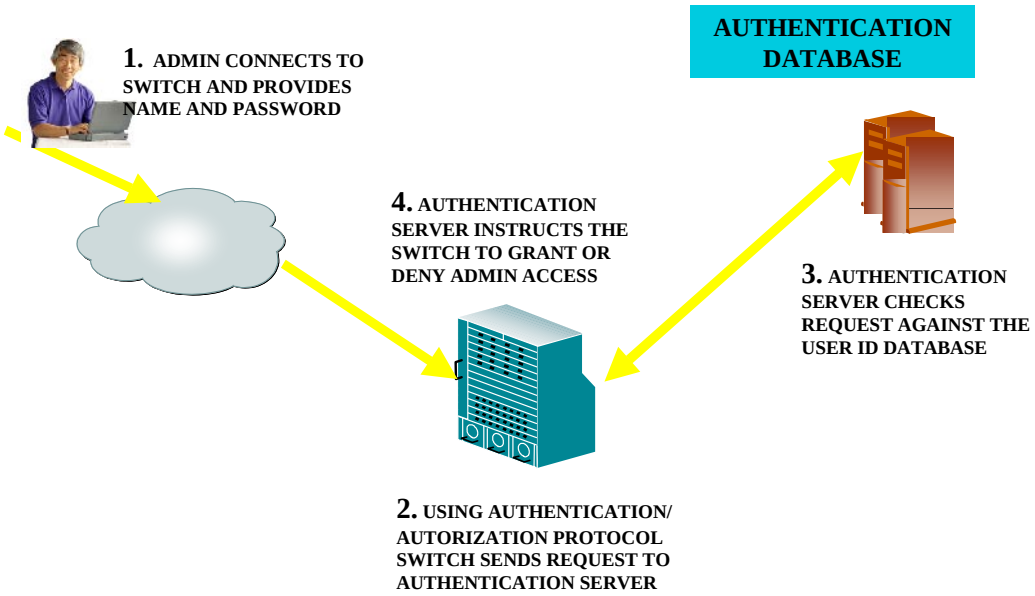


Figure 10-1 Authentication and Authorization: How It Works

RADIUS Authentication Features in WebOS

- Support of RADIUS client on the switch, based on the protocol definitions in RFC 2138.
- An option for the administrator to enable/disable support of RADIUS authentication and authorization.

  The default is to disable the use of RADIUS for authentication and authorization.
- The RADIUS secret password can be up to 32 bytes. It can be less than 16 octets.
- Support of a "secondary authentication server" so that when the primary authentication server is unreachable, the switch can send client authentication requests to the "secondary authentication server".

  Use the **`/cfg/sys/radius/cur`** command to show the currently active RADIUS authentication server.
- RADIUS server retries and the timeout value are user-configurable. The value parameters are
  - Timeout value = 1-10 seconds
  - Retries = 1-3

    The switch will timeout if it does not see response from the RADIUS server in (1-3) seconds. The switch will also automatically retry to the RADIUS servers before it declares the server is down.
- Support of user-configurable RADIUS application port. The default is 1645/udp based on RFC 2138.
- Network administrator can define privileges for one or more specific users to access the switch at the RADIUS user database. The following user accounts listed in Table 10-1 can be defined in the RADIUS server dictionary file:

Table 10-1 User Access Levels

User Account	Description and Tasks Performed	Password
User	The User has no direct responsibility for switch management. He or she can view all switch status information and statistics, but cannot make any configuration changes to the switch.	`user`
SLB Operator	The SLB Operator manages Web servers and other Internet services and their loads. In addition to being able to view all switch information and statistics, the SLB Operator can enable/disable servers using the Server Load Balancing operation menu.	`slboper`
Layer 4 Operator	The Layer 4 Operator manages traffic on the lines leading to the shared Internet services. This user currently has the same access level as the SLB operator. This level is reserved for future use, to provide access to operational commands for operators managing traffic on the line leading to the shared Internet services.	`l4oper`
Operator	The Operator manages all functions of the switch. In addition to SLB Operator functions, the Operator can reset ports or the entire switch.	`oper`
SLB Administrator	The SLB Administrator configures and manages Web servers and other Internet services and their loads. In addition to SLB Operator functions, the SLB Administrator can configure parameters on the Server Load Balancing menus, with the exception of not being able to configure filters or bandwidth management.	`slbadmin`
Layer 4 Administrator	The Layer 4 Administrator configures and manages traffic on the lines leading to the shared Internet services. In addition to SLB Administrator functions, the Layer 4 Administrator can configure all parameters on the Server Load Balancing menus, including filters and bandwidth management.	`l4admin`
Administrator	The superuser Administrator has complete access to all menus, information, and configuration commands on the switch, including the ability to change both the user and administrator passwords.	`admin`

When the user logs in, the switch authenticates their level of access by sending the RADIUS Access- Request, the "client authentication request," to the RADIUS authentication server.

If the remote user is successfully authenticated by the authentication server, the switch will verify the “privileges" of the remote user and authorize the appropriate access. When both the primary and secondary authentication servers are not reachable, the administrator has an option to allow "backdoor" access via the console only or console and telnet access. The default is “disable” for telnet access and “enable” for console access.

All user privileges other than those assigned to the “Administrator” have to be defined in the RADIUS dictionary. The file name of the dictionary is RADIUS vendor-dependent. The following user privileges are Alteon WebSystems’s proprietary definitions.

Table 10-2 Alteon WebSystems User Access Levels

User Name/Access	User-Service-Type	Value
User	*Vendor-supplied*	255
SLB Operator	*Vendor-supplied*	254
Layer 4 Operator	*Vendor-supplied*	253
Operator	*Vendor-supplied*	252
SLB Administrator	*Vendor-supplied*	251
Layer 4 Administrator	*Vendor-supplied*	250

- SecurID support, provided RADIUS server can do ACE/Server client proxy. The password is the PIN number, plus the tokencode of securID card.

Secure Shell (SSH) and Secure Copy (SCP)

Although a remote network administrator can manage the configuration of an Alteon Web switch via Telnet, this method does not provide a secure connection. Using Secure Shell (SSH) and Secure Copy (SCP), messages between a remote administrator and the switch use secure tunnels so that the data on the network is encrypted and secured.

NOTE – SSH/SCP features are supported only on the AD4 and A184 Web switches and can only be configured via the console port, using the command line interface. When SSH is enabled, SCP is also enabled.

SSH (**Secure Shell**) is a protocol that enables a remote administrator to securely log into another computer over a network to execute management commands. All the data sent over the network using SSH is encrypted and secured. Using SSH gives administrators an alternate way to manage the switch, one that provides strong security.

SCP (**Secure Copy**) is typically used to securely copy files from one machine to another. SCP uses SSH for encryption of data on the network. On an Alteon Web switch, SCP is used to download and upload the switch configuration via secure channels.

The benefits of using SSH and SCP are listed below:

- Authentication of remote administrators

  Administrator identification using NAME/PASSWORD

- Authorization of remote administrators

  Determine the permitted actions

  Customize service for individual administrators

- Encryption of management messages

  Messages between remote administrator and switch are encrypted

- Secure copy support

NOTE – The WebOS implementation of SSH is based on SSH version 1.5 and supports SSH-1.5-1.X.XX. SSH clients of other versions (especially Version 2) will not be supported. The following SSH clients have been tested:
SSH 1.2.23 and SSH 1.2.27 for Linux (freeware)
SecureCRT 3.0.2 and SecureCRT 3.0.3 (Van Dyke Technologies, Inc.)
F-Secure SSH 1.1 for Windows (Data Fellows)

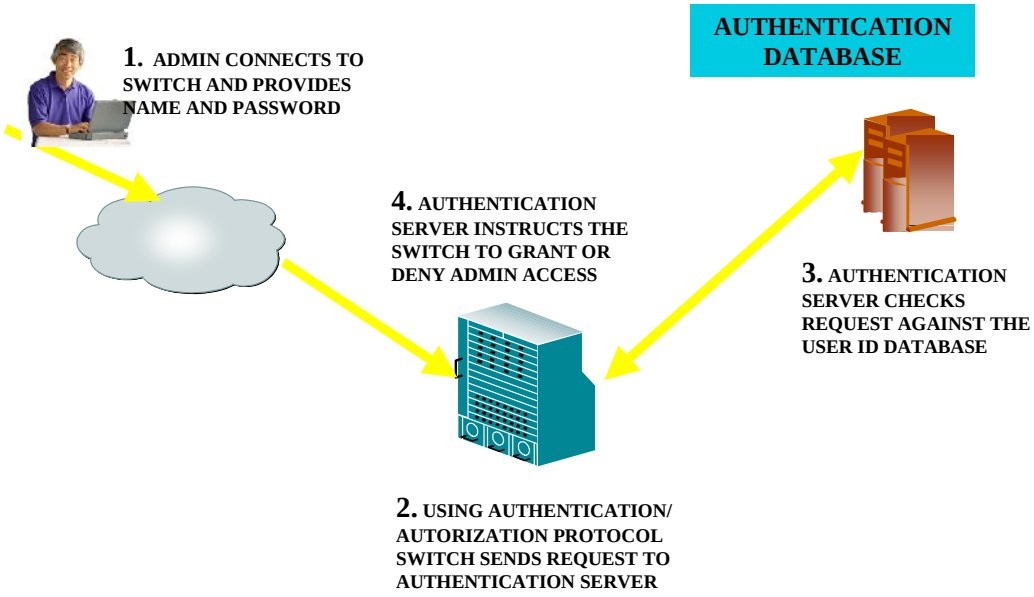


Figure 10-2 Secure Switch Management: How It Works

Encryption of Management Messages

The supported encryption and authentication methods for both SSH and SCP are listed below:

Server Host Authentication:	Client RSA-authenticates the switch in the beginning of every connection.
Key Exchange:	RSA
Encryption:	3DES-CBC, DES
User Authentication:	local password authentication, RADIUS, `SecurID` (via RADIUS, for SSH only; that is, not applied to SCP)

SCP Services

NOTE – Administrator privileges are required to perform SCP commands.

Four SCP commands are supported in this service: **`getcfg`**, **`putcfg`**, **`putcfg_apply`**, and **`putcfg_apply_save`**.

- **`getcfg`** is used to download switch's configuration to the remote host via SCP.
- **`putcfg`** is used to upload switch's configuration from a remote host to the switch; the "diff" command will be automatically executed at the end of "putcfg" to notify the remote client of the difference between the new and the current configurations.
- **`putcfg_apply`** will run the "apply" command after the "putcfg" is done
- **`putcfg_apply_save`** will save the new configuration to the flash after "putcfg_apply" is done.

The **`putcfg_apply`** and **`putcfg_apply_save`** commands are provided because extra "apply" and "save" commands are usually required after a "putcfg", however, a SCP session is not in an interactive mode at all.

RSA Host and Server Keys

To support the SSH server feature, two sets of RSA keys, host and server keys are required. The host key is 1024 bits and is used to identify the switch. The server key is 768 bits and is used to make it impossible to decipher a captured session by breaking into the switch at a later time.

When the SSH server is first enabled and applied, the switch will automatically generate the host and server keys, and will then store them into the flash.

Two commands are provided to generate these two keys manually and they are **`/cfg/sys/sshd/hkeygen`** and **`/cfg/sys/sshd/skeygen`**. Again, the host or server key will be automatically stored into flash after generated.

When the switch reboots, it will retrieve the host and server keys from the flash. If these two keys are not available in the flash and if the SSH server feature is enabled, the switch will automatically generate them during the system reboot.

The switch can also automatically regenerate the RSA server key. To set the interval of RSA server key auto-generation, use this command:

```
>> # /cfg/sys/sshd/intrval <n>
```

where *n* (number of hours) must be in the range (0..24) and a value of 0 denotes that RSA server key auto-generation is disabled. When n is greater than 0, the switch will auto-generate the RSA server key every *n* hours; however, RSA server key generation will be skipped if the switch is busy doing other key or cipher generation when the timer expires.

Radius Authentication

SSH/SCP is integrated with RADIUS authentication. After the RADIUS server is enabled in the switch, all subsequent SSH authentication requests will be re-directed to the specified RADIUS servers for authentication. The redirection is transparent to the SSH clients.

Secure ID Support

SSH/SCP can also work with SecurID, a token card-based authentication method. The use of SecurID requires the interactive mode during login which is not provided by the SSH connection.

To login using SSH without difficulties, you need to use a special username, “ace,” to log in, in order to bypass the SSH authentication. After an SSH connection is established, you will then be prompted to enter the username and password (the SecurID authentication is being performed now). You will need to provide your actual username and the token in your SecurID card as a regular Telnet user will do in order to log in.

To use SCP, you need to use SCP-only administrator’s password (that is, the `scpadm` option under the **`/cfg/sys/sshd/`** menu to bypass the checking of SecurID. Alternately, you can configure a regular administrator with a fixed password in the RADIUS server if it can be supported. A regular administrator with a fixed password in the RADIUS server can perform both SSH and SCP without additional authentication being required.

A SCP-only administrator’s password is typically used when SecurID is used. For example, it can be used in an automation program (in which the tokens of SecurID are not available) to back up (download) the switch configurations each day.

NOTE – The SCP-only administrator’s password must be different from the regular administrator’s password. If the two passwords are the same, the administrator using that password will not be allowed to login as a SSH user since the switch will recognize him as the SCP-only administrator and only allow the administrator access to SCP commands.

Configuring SSH/SCP

NOTE – SSH/SCP can only be configured via the console port, using the CLI.

To enable/disable the SSH/SCP feature, use this command:

```
>> # /cfg/sys/sshd/on
```

To set the interval of RSA server key auto-generation, use this command:

```
>> # /cfg/sys/sshd/intrval <n>
```

where *n* (number of hours) must be in the range (0..24) and a value of 0 denotes that RSA server key auto-generation is disabled. When n is greater than 0, the switch will auto-generate the RSA server key every *n* hours; however, RSA server key generation will be skipped if the switch is busy doing other key or cipher generation when the timer expires.

To enable or disable the SCP apply and save (i.e., SCP putcfg_apply and putcfg_apply_save commands), use these commands:

```
>> # /cfg/sys/sshd/ena
>> # /cfg/sys/sshd/dis
```

To view the current SSH/SCP related configuration, use this command:

```
>> # /cfg/sys/sshd/cur
```

To view the difference between the new configuration and the current configuration, use this command:

```
>> # diff
```

To apply the pending changes from the new configuration, use this command:

```
>> # apply
```

NOTE – If SSH/SCP is enabled and an **`apply`** command is issued, the switch will automatically generate the RSA host and server keys if they are not available. It will take several minutes to complete this process.

To save the current configuration to flash, use this command:

```
>> # save
```

Usually, there will be no need to manually generate the RSA host and server keys. However, you may still do so by using the following commands:

```
>> # /cfg/sys/sshd/hkeygen                    Generates the host key.
>> # /cfg/sys/sshd/skeygen                   Generates the server key.
```

NOTE – These two commands will take effect immediately without the need of an **apply** command being issued.

Some Supported Client Commands

NOTE – Up to four simultaneous Telnet/SSH/SCP connections are supported on a switch.

- To login to the switch:

  ssh <*switch_ip*>or **ssh -l** <*username*> <*switch_ip*>

- To download the switch configuration using SCP:

  scp <*switch_ip*>**:getcfg** <*local_filename*>

- To upload the configuration to the switch:

  scp <*local_filename*> <*switch_ip*>**:putcfg**

Some examples are listed below:

```
>> # ssh 205.178.15.157
>> # ssh -l dleu 205.178.15.157
>> # scp 205.178.15.157:getcfg ad4.cfg
>> # scp ad4.cfg 205.178.15.157:putcfg
```

where *205.178.15.157* is the IP address of the switch.

Please also note that **apply** and **save** commands are still needed after the last command (`scp ad4.cfg 205.178.15.157:putcfg`) is issued. Or, instead, you can use the following commands to avoid the "apply and save" issue:

```
>> # scp ad4.cfg 205.178.15.157:putcfg_apply
>> # scp ad4.cfg 205.178.15.157:putcfg_apply_save
```

CHAPTER 11

VLANs

This chapter describes network design and topology considerations for using VLANs.

Virtual Local Area Networks (*VLANs*) are commonly used to split up groups of network users into manageable broadcast domains, to create logical segmentation of workgroups, and to enforce security policies among logical segments.

Basic VLANs can be configured during initial switch configuration (see "Using the Setup Utility" in Chapter 3 of the *WebOS 8.0 Command Reference*). More comprehensive VLAN configuration can be done from the command-line interface (see "VLAN Configuration" as well as "Port Configuration" in Chapter 7 of the *WebOS Command Reference*).

VLAN ID Numbers

WebOS supports up to 246 VLANs per switch. Even though the maximum number of VLANs supported at any given time is 246, each can be identified with any number between 1 and 4094.

VLANs are defined on a per-port basis. Each port on the switch can belong to one or more VLANs, and each VLAN can have any number of switch ports in its membership. Any port that belongs to multiple VLANs, however, must have VLAN *tagging* enabled (see below).

Each port in the switch has a configurable default VLAN number, known as its *PVID*. The factory default value of all PVIDs is 1. This places all ports on the same VLAN initially, although each port's PVID is configurable to any VLAN number between 1 and 4094.

Any non-tagged frames (those with no VLAN specified) are classified with the sending port's PVID.

VLAN Tagging

WebOS software supports 802.1Q VLAN *tagging*, providing standards-based VLAN support for Ethernet systems.

Tagging places the VLAN identifier in the frame header, allowing multiple VLANs per port. When you configure multiple VLANs on a port, you must also enable tagging on that port.

Since tagging fundamentally changes the format of frames transmitted on a tagged port, you must carefully plan network designs to prevent tagged frames from being transmitted to devices that do not support 802.1Q VLAN tags.

VLANs and Spanning-Tree

When *Spanning-Tree* is enabled on the switch, it detects and eliminates logical loops in a bridged or switched network. When multiple paths exist, Spanning-Tree configures the network so that a switch uses only the most efficient path. If that path fails, Spanning-Tree automatically sets up another active path on the network to sustain network operations.

If you configure the switch with Spanning-Tree, there will be a single instance of Spanning-Tree per switch, regardless of the number of configured VLANs in an enabled state.

VLANs and the IP Interfaces

Careful consideration must be made when creating VLANs within the switch, such that communication with the switch Management Processor (MP) remains possible where it is required.

Access to the switch for remote configuration, trap messages, and other management functions can only be accomplished from stations that are on VLANs which include an IP interface to the switch (see “IP Interface Menu” in Chapter 7 of the *WebOS Command Reference*). Likewise, access to management functions can be cut off to any VLAN by excluding IP interfaces from its membership.

For example, if all IP interfaces are left on VLAN #1 (the default), and all ports are configured for VLANs other than VLAN #1, then switch management features are effectively cut off. If an IP interface is added to one of the other VLANs, the stations in that VLAN all have access to switch management features.

VLAN Topologies and Design Issues

By default, the WebOS 8.0 software has a single VLAN configured on every port. This groups all ports into the same broadcast domain. This VLAN has an 802.1Q VLAN PVID of 1. Since in this default only a single VLAN is configured per port, VLAN tagging is turned off.

Since VLANs are most commonly used to create individual broadcast domains and/or separate IP subnets, it is useful for host systems to be able to have presence on more than one VLAN simultaneously. Alteon WebSystems' Web switches and ACEnic adapters have the unique capability of being able to support multiple VLANS on a per port or per interface basis, allowing very flexible configurations.

You can configure multiple VLANs on a single ACEnic adapter, with each VLAN being configured through a logical interface and logical IP address on the host system. Each VLAN configured on the adapter must also be configured on the switch port to which it is connected. If multiple VLANs are configured on the port, tagging must be turned on.

Using this flexible multi-VLAN system, you can logically connect users and segments to a host with a single ACEnic adapter that supports many logical segments or subnets.

Example #1: Multiple VLANS with Tagging Adapters

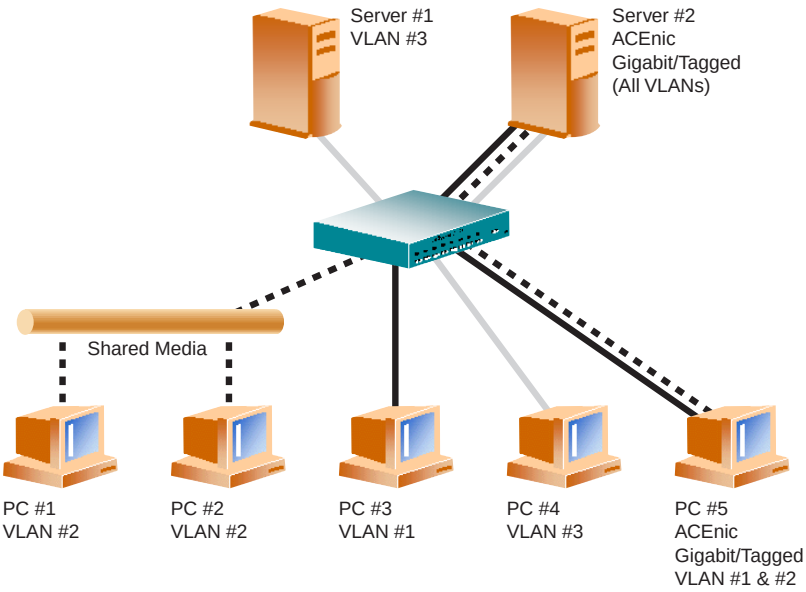


Figure 11-1 Example #1: Multiple VLANs with Tagging ACEnic Adapters

The features of this VLAN are described below:

Component	Description
WebOS Web switch	This switch is configured for three VLANs that represent three different IP subnets. Two servers and five clients are attached to the switch.
Server #1	This server is part of the VLAN #3 and only has presence in one IP subnet. The port that it is attached to is configured only for VLAN #3, so VLAN tagging is off.
Server #2	A high-use server that needs to be accessed from all VLANs and IP subnets. This server has an Alteon WebSystems ACEnic adapter installed with VLAN tagging turned on. The adapter is attached to one of the WebOS Web switch's Gigabit Ethernet ports, which is configured for VLANs #1, #2, and #3, and also has tagging turned on. Because of the VLAN tagging capabilities of both the adapter and the switch, the server is able to communicate on all three IP subnets in this network, but continues to maintain broadcast separation between all three VLANs and subnets.
PCs #1 and #2	These PCs are attached to a shared media hub that is then connected to the switch. They belong to VLAN #2, and are logically in the same IP subnet as Server #2 and PC #5. Tagging is not enabled on their switch port.
PC #3	A member of VLAN #1, this PC can only communicate with Server #2 and PC #5.
PC #4	A member of VLAN #3, this PC can only communicate with Server #1 and Server #2.
PC #5	A member of both VLAN #1 and VLAN #2, this PC has an Alteon WebSystems' ACEnic Gigabit Ethernet Adapter installed. It is able to communicate with Server #2 via VLAN #1, and to PC #1 and PC #2 via VLAN #2. The switch port to which it is connected is configured for both VLAN #1 and VLAN #2, and has tagging turned on.

NOTE – VLAN tagging is only required on ports that are connected to other Alteon WebSystems' web switches, or on ports that connect to tag-capable end-stations, such as servers with Alteon WebSystems' ACEnic Gigabit Ethernet Adapters.

Example #2: Parallel Links with VLANs

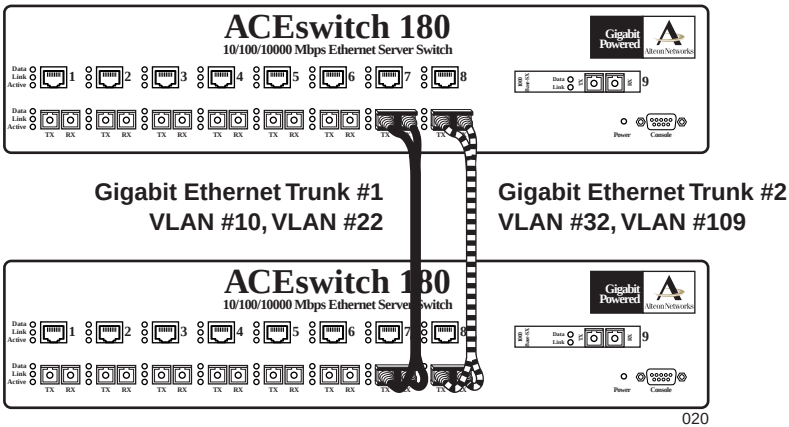


Figure 11-2 Example #2: Parallel Links with VLANs

The following items describe the features of this example:

- Example #2 shows how, through the use of VLANs, it is possible to create configurations where there are multiple links between two switches, without creating broadcast loops.
- Two Alteon WebSystems' WebOS switches are connected with two different Gigabit Ethernet links. Without VLANs, this configuration would create a broadcast loop, but the Spanning-Tree Protocol (STP) Topology Resolution process resolves parallel loop-creating links.
- With VLANs, neither switch-to-switch link shares the same VLAN and thus, are separated into their own broadcast domains.
- Ports #1 and #2 on both switches are on VLAN #10; Ports #3 and #4 on both switches are on VLAN #22. Ports #5 and #6 on both switches are on VLAN #32; and port #9 on both switches are on VLAN #109.
- It is necessary to turn off Spanning-Tree on at least one of the switch-to-switch links, or alternately turned off in both switches. Spanning-Tree executes on a per-network level, not a per-VLAN level. STP Bridge PDUs will be transmitted out both connected Gigabit Ethernet ports and be interpreted by the connected switch that there is a loop to resolve.
- Spanning-Tree is not VLAN-aware. Therefore, any VLAN configuration that might involve a parallel link from an STP perspective must be taken into account during network design. Alteon WebSystems recommends that you avoid topologies such as these, if at all possible.

CHAPTER 12

Jumbo Frames

To reduce host frame processing overhead, the Alteon WebSystems' ACEnic adapters and WebOS Web switches, both running operating software version 2.0 or greater, can receive and transmit frames that are far larger than the maximum normal Ethernet frame. By sending one Jumbo Frame instead of myriad smaller frames, the same task is accomplished with less processing.

The switches and the ACEnic adapter support Jumbo Frame sizes up to 9022 octets. These can be transmitted and received between ACEnic adapter-enabled hosts through the switch across any VLAN.

Isolating Jumbo Frame Traffic using VLANs

Jumbo Frame traffic must not be used on a VLAN where there is any device that cannot process frame sizes larger than Ethernet maximum frame size.

Additional VLANs can be configured on the adapters and switches to support non-Jumbo Frame VLANs for servers and workstations that do not support extended frame sizes. End-stations with an ACEnic adapters installed and attached to switches can communicate across both the Jumbo Frame VLANs and regular frame VLANs at the same time.

In the example illustrated in Figure 12-1 on page 12-226, the two servers can handle Jumbo Frames but the two clients cannot; therefore Jumbo Frames should only be enabled and used on the VLAN represented by the solid lines, but not for the VLAN with the dashed lines. Jumbo Frames are not supported on ports configured for half-duplex mode.

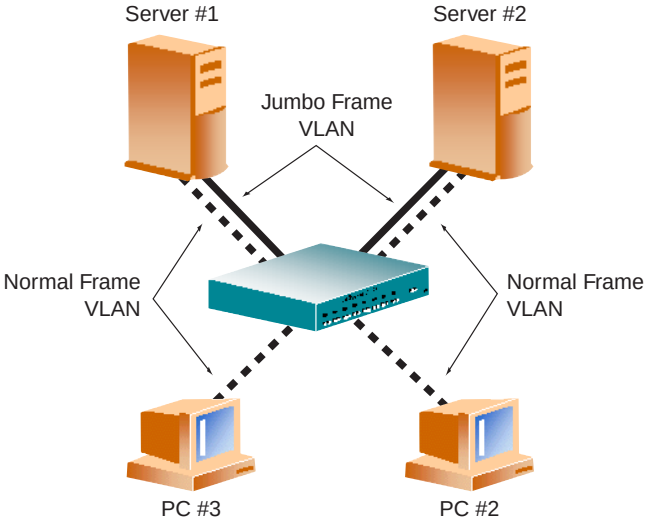


Figure 12-1 Jumbo Frame VLANs

Routing Jumbo Frames to Non-Jumbo Frame VLANs

When IP Routing is used to route traffic between VLANs, the switch will fragment jumbo UDP datagrams when routing from a Jumbo Frame VLAN to a non-Jumbo Frame VLAN. The resulting Jumbo Frame to regular frame conversion makes implementation even easier.

CHAPTER 13

IP Routing

This chapter provides configuration background and examples for using the switch to perform routing functions.

IP Routing Benefits

IP Routing allows the network administrator to seamlessly connect server IP subnets to the rest of the backbone network, using a combination of configurable IP switch interfaces and IP routing options.

The IP Routing feature enhances Alteon WebSystems' Server Switching solution in the following ways:

- It provides the ability to perform Server Load Balancing (using both Layer 3 and Layer 4 switching in combination) to server subnets which are separate from backbone subnets.
- By automatically fragmenting UDP Jumbo Frames when routing to non-Jumbo Frame VLANs or subnets, it provides another means to invisibly introduce Jumbo Frames technology into the Server Switched network.
- It provides the ability to seamlessly route IP traffic between multiple VLANs configured in the switch.

Example of Routing Between IP Subnets

The physical layout of most corporate networks has evolved over time. Classic hub/router topologies have given way to faster switched topologies, particularly now that switches are increasingly intelligent. ACElerate powered switches, in fact, are now smart enough and fast enough to perform routing functions on par with wire speed Layer 2 switching.

The combination of faster routing and switching in a single device provides another service: it allows you to build versatile topologies that account for legacy configurations.

For example, consider the following topology migration:

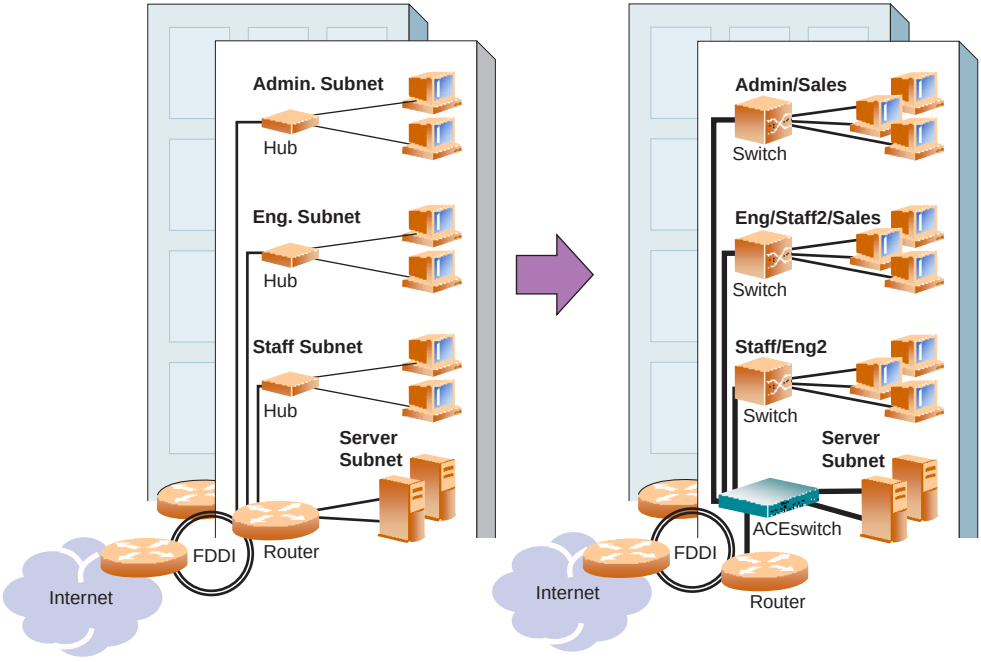


Figure 13-1 The Router Legacy Network

In this example, a corporate campus has migrated from a router-centric topology to a faster, more powerful switch-based topology. As is often the case, the legacy of network growth and redesign has left the system with a hodge-podge of illogically distributed subnets. This is a situation that switching alone cannot cure. Instead, the router is flooded with cross-subnet communication. This compromises efficiency in two ways:

- Routers can be slower than switches. The cross-subnet side trip from the switch to the router and back again adds two hops for the data, slowing throughput considerably.
- Traffic to the router increases, worsening any congestion.

Even if every end-station on the network could be moved to better logical subnets (a daunting task), competition for access to common server pools on different subnets still burdens the routers.

This problem is solved by using Alteon WebSystem's web switches with built-in IP Routing capabilities. Cross-subnet LAN traffic can now be routed within the WebOS-powered switches with wire speed Layer 2 switching performance. This not only eases the load on the router, but saves the network administrators from re-configuring each and every end-station with new IP addresses.

Take a closer look at the ACEswitch 180 in the example configuration:

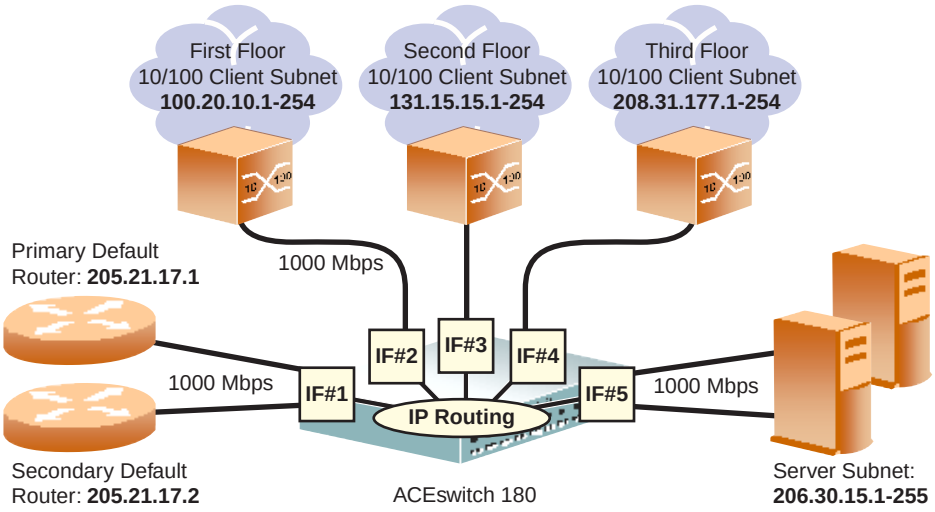


Figure 13-2 Switch-Based Routing Topology

The ACEswitch 180 connects the Gigabit Ethernet and Fast Ethernet trunks from various switched subnets throughout one building. Common servers are placed on another subnet attached to the switch. A primary and backup router are attached to the switch on yet another subnet.

Without Layer 3 IP Routing on the switch, cross-subnet communication is relayed to the default gateway (in this case, the router) for the next level of routing intelligence. The router fills in the necessary address information and sends the data back to the switch, which relays the packet to the proper destination subnet using Layer 2 switching.

With Layer 3 IP Routing in place on the Alteon WebSystems' switch, routing between different IP subnets can be accomplished entirely within the switch. This leaves the routers free to handle inbound and outbound traffic for this group of subnets.

As an added benefit, UDP Jumbo Frame traffic is automatically fragmented to regular Ethernet frame sizes when routing to non-Jumbo Frame subnets. For instance, this allows servers to communicate with each other using Jumbo Frames, and to non-Jumbo Frame devices using regular frames, all transparently to the user.

Example ACEswitch 180 Configuration for Subnet Routing

Prior to configuration, you must be connected to the switch command-line interface as the administrator (see Chapter 2 in the *WebOS Command Reference*).

NOTE – For details about any of the menu commands described in this example, see "IP Configuration" in Chapter 7 of the *WebOS Command Reference*.

---

1. **Assign an IP address (or document the existing one) for each real server, router, and client workstation.**

   In our example topology in Figure 13-2 on page 229, the following IP addresses are used:

Table 13-1 Subnet Routing Example: IP Address Assignments

Subnet	Devices	IP Addresses
#1	Primary and Secondary Default Routers	205.21.17.1 and 205.21.17.2
#2	First Floor Client Workstations	100.20.10.1-254
#3	Second Floor Client Workstations	131.15.15.1-254
#4	Third Floor Client Workstations	208.31.177.1-254
#5	Common Servers	206.30.15.1-254

2. **On the switch, assign an IP interface for each subnet attached to the switch.**

   Since there are five IP subnets connected to the switch, five IP interfaces are needed:

Table 13-2 Subnet Routing Example: IP Interface Assignments

Interface	Devices	IP Interface Address
IF #1	Primary and Secondary Default Routers	205.21.17.3
IF #2	First Floor Client Workstations	100.20.10.16
IF #3	Second Floor Client Workstations	131.15.15.1
IF #4	Third Floor Client Workstations	208.31.177.2
IF #5	Common Servers	206.30.15.200

These are configured using the following commands at the CLI:

```
>> Main# /cfg/ip/if 1                    (Select IP interface 1)
>> IP Interface 1# addr 205.21.17.3      (Assign IP address for the interface)
>> IP Interface 1# ena                   (Enable IP interface 1)
>> IP Interface 1# ../if 2               (Select IP interface 2)
>> IP Interface 2# addr 100.20.10.16     (Assign IP address for the interface)
>> IP Interface 2# ena                   (Enable IP interface 2)
>> IP Interface 2# ../if 3               (Select IP interface 3)
>> IP Interface 3# addr 131.15.15.1      (Assign IP address for the interface)
>> IP Interface 3# ena                   (Enable IP interface 3)
>> IP Interface 3# ../if 4               (Select IP interface 4)
>> IP Interface 4# addr 208.31.177.2     (Assign IP address for the interface)
>> IP Interface 4# ena                   (Enable IP interface 4)
>> IP Interface 4# ../if 5               (Select IP interface 5)
>> IP Interface 5# addr 206.30.15.200    (Assign IP address for the interface)
>> IP Interface 5# ena                   (Enable IP interface 5)
```

3. **Set each server and workstation's default gateway to point to the appropriate switch IP interface (the one in the same subnet as the server or workstation).**

4. **On the switch, configure the default gateways to point to the routers.**

   This allows the switch to send outbound traffic to the routers:

```
>> IP Interface 5# /cfg/ip/gw 1          (Select primary default gateway)
>> Default gateway 1# addr 205.21.17.1   (Point to primary router)
>> Default gateway 1# ena                (Enable primary default gateway)
>> Default gateway 1# ../gw 2            (Select secondary default gateway)
>> Default gateway 2# addr 205.21.17.2   (Point to secondary router)
>> Default gateway 2# ena                (Enable secondary default gateway)
```

5. **On the switch, enable, apply, and verify the configuration.**

```
>> Default gateway 2# ../fwrd            (Select the IP Forwarding Menu)
>> IP Forwarding# on                     (Turn IP forwarding on)
>> IP Forwarding# apply                  (Make your changes active)
>> IP Forwarding# ../cur                 (View current IP settings)
```

   Examine the resulting information. If any settings are incorrect, make any appropriate changes.

6. **On the switch, save your new configuration changes.**

```
>> IP# save                              (Save for restore after reboot)
```

Another Option: Adding VLANs to the Routing Example

The routers, servers, and clients in the example above are all in the same broadcast domain. If limiting broadcasts is desired in your network, you could use VLANs to create distinct broadcast domains. For example, you could create one VLAN for the routers, one for the servers, and one for the client trunks.

In this exercise, we are adding to the previous configuration.

1. **Determine which switch ports and IP interfaces belong to which VLANs.**

   The following table adds ports and VLANs information:

Table 13-3 Subnet Routing Example: Optional VLAN Ports

VLAN	Devices	IP Interface	Switch Port
#1	First Floor Client Workstations	3	1
	Second Floor Client Workstations	4	2
	Third Floor Client Workstations	5	3
#2	Primary Default Router	1	4
	Secondary Default Router	2	5
#3	Common Servers #1	6	6
	Common Servers #2	7	7

2. **On the switch, set the default VLAN for each port:**

```
>> # /cfg/port 1                    (Select port for First Floor)
>> Port 1# pvid 1                  (Set default to VLAN 1)
>> Port 1# ../port 2               (Select port for Second Floor)
>> Port 2# pvid 1                  (Set default to VLAN 1)
>> Port 2# ../port 3               (Select port for Third Floor)
>> Port 3# pvid 1                  (Set default to VLAN 1)
>> Port 3# ../port 4               (Select port for default router 1)
>> Port 4# pvid 2                  (Set default to VLAN 2)
>> Port 4# ../port 5               (Select port for default router 2)
>> Port 5# pvid 2                  (Set default to VLAN 2)
>> Port 5# ../port 6               (Select port for common server 1)
>> Port 6# pvid 3                  (Set default to VLAN 3)
>> Port 6# ../port 7               (Select port for common server 2)
>> Port 7# pvid 3                  (Set default to VLAN 3)
```

3. **On the switch, enable the VLANs.**

```
>> Port 7# /cfg/vlan 1                    (Select VLAN 1, the client VLAN)
>> VLAN 1# ena                            (enable VLAN 1)
>> VLAN 1# ../vlan 2                      (Select VLAN 2, the def. router VLAN)
>> VLAN 2# ena                            (enable VLAN 2)
>> VLAN 2# ../vlan 3                      (Select VLAN 3, the server VLAN)
>> VLAN 3# ena                            (enable VLAN 3)
```

4. **On the switch, add each IP interface to the appropriate VLAN.**

   Now that the ports are separated into three VLANs, the IP interface for each subnet must be placed in the appropriate VLAN. From Table 13-3 on page 13-232, the settings are made as follows:

```
>> VLAN 3# /cfg/ip/if 1                   (Select IP interface 1 for def. routers)
>> IP Interface 1# vlan 2                 (Set to VLAN 2)
>> IP Interface 1# ../if 2                (Select IP interface 2 for first floor)
>> IP Interface 2# vlan 1                 (Set to VLAN 1)
>> IP Interface 2# ../if 3                (Select IP interface 3 for second floor)
>> IP Interface 3# vlan 1                 (Set to VLAN 1)
>> IP Interface 3# ../if 4                (Select IP interface 4 for third floor)
>> IP Interface 4# vlan 1                 (Set to VLAN 1)
>> IP Interface 4# ../if 5                (Select IP interface 5 for servers)
>> IP Interface 5# vlan 3                 (Set to VLAN 3)
```

5. **On the switch, apply and verify the configuration.**

```
>> IP Interface 5# apply                  (Make your changes active)
>> IP Interface 5# /info/vlan             (View current VLAN information)
>> Information# port                      (View current port information)
```

   Examine the resulting information. If any settings are incorrect, make the appropriate changes.

6. **On the switch, save your new configuration changes.**

```
>> Information# save                      (Save for restore after reboot)
```

Defining IP Address Ranges for the Local Route Cache

The Local Route Cache lets you more efficiently use switch resources. The local network address and local network mask parameters (accessed via the `/cfg/ip/frwd/local/add` command) define a range of addresses which will be cached on the switch. The `local network address` is used to define the base IP address in the range which will be cached, and the `local network mask` is the mask which is applied to produce the range. To determine if a route should be added to the memory cache, the destination address is masked (bitwise AND) with the local network mask and checked against the local network address.

By default, the local network address and local network mask are both set to 0.0.0.0. This produces a range that includes all Internet addresses for route caching: 0.0.0.0 through 255.255.255.255.

To limit the route cache to your local hosts, you could configure the parameters as in the following examples:

Table 13-4 Local Routing Cache Address Ranges

Local Host Address Range	Local Network Address	Local Network Mask
0.0.0.0 - 127.255.255.255	0.0.0.0	128.0.0.0
128.0.0.0 - 255.255.255.255	128.0.0.0	128.0.0.0
205.32.0.0 - 205.32.255.255	205.32.0.0	255.255.0.0

NOTE – All addresses that fall outside the defined range are forwarded to the default gateway. The default gateways must be within range.

Border Gateway Protocol (BGP)

Border Gateway Protocol (BGP) is an Internet protocol that enables routers on a network to share routing information with each other and advertise information about the segments of the IP address space they can access within their network with routers on external networks. BGP allows you to decide what is the "best" route for a packet to take from your network to a destination on another network, rather than simply setting a default route from your border router(s) to your upstream provider(s). BGP is defined in RFC 1771

Internal Routing vs. External Routing

To ensure effective processing of network traffic, every router on your network needs to know how to send a packet (directly or indirectly) to any other location/destination in your network. This is referred to as *internal routing* and can be done with static routes or using active internal routing protocols such as RIP, RIPv2, and OSPF.

It is also useful to tell routers outside your network (upstream providers or *peers*) about the routes you have access to in your network. External networks (those outside your own), that are under the same administrative control are referred to as *autonomous systems* (AS) and the sharing of routing information between autonomous systems is known as *external routing*.

External BGP is used to exchange routes between different autonomous systems, while internal BGP is used to exchange routes within the same autonomous system. Internal BGP is one of the "interior routing protocols" that you can use to do "active routing" inside your network.

Typically, an AS will have one or multiple "border routers;" peer routers that exchange routes with other AS's, as well as an internal routing scheme enabling every router in that AS to get to every other router and destination within that AS.

When you *advertise* routes to border routers on other autonomous systems, you are effectively committing to carry data to the IP space represented in the route being advertised. For example, if you advertise 192.204.4.0/24, you are declaring that if another router sends you data destined for any address in 192.204.4.0/24, you know how to carry that data to its destination.

For each new route, if a peer is interested in that route (for example, if a peer would like to receive your static routes and the new route is static), an update message is sent to that peer containing the new route. For each route removed from the route table, if the route had already been sent to a peer, a update message containing the route to withdraw is sent to that peer.

For each Internet host, you must be able to send a packet out a path to that host, and that host has to have a path back to you. This means that whoever provides Internet connectivity to that host must have a path to you. Ultimately, this means that they must "hear a route" which covers the section of the IP space you're using, or you will not have connectivity to the host in question.

CHAPTER 14

Port Trunking

This chapter provides configuration background and examples for trunking multiple ports together.

Port Trunking Overview

Basics

Trunk groups can provide super-bandwidth, multi-link connections between Alteon WebSystems' WebOS switches or other trunk-capable devices. A "trunk group" is a group of ports that act together, combining their bandwidth to create a single, larger virtual link.

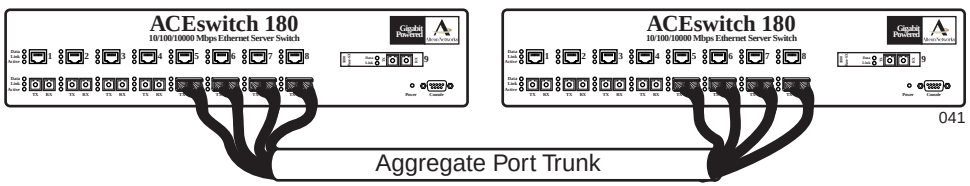


Figure 14-1 Port Trunk Group

When using port trunk groups between two ACEswitch 180 switches, for example, the network administrator can create a virtual link between the switches operating up to 6 Gigabits per second, depending on how many physical ports are combined. The switch supports up to 4 trunk groups per switch, each with 2 to 6 links.

Trunk groups are also useful for connecting an Alteon WebSystems' switch to third-party devices that support link aggregation, such as Cisco routers and switches with EtherChannel technology (*not* ISL Trunking technology), and Sun's Quad Fast Ethernet Adapter. Alteon WebSystems' trunk group technology is compatible with these devices when they are configured manually.

Statistical Load Distribution

Network traffic is statistically load balanced between the ports in a trunk group. The WebOS-powered switch uses both the Layer 2 MAC address and Layer 3 IP address information present in each transmitted frame for determining load distribution.

The addition of Layer 3 IP address examination is an important advance for traffic distribution in trunk groups. In some port trunking systems, only Layer 2 MAC addresses are considered in the distribution algorithm. Each packet's particular combination of source and destination MAC addresses results in selecting one line in the trunk group for data transmission. If there are enough Layer 2 devices feeding the trunk lines, then traffic distribution becomes relatively even. In some topologies, however, only a limited number of Layer 2 devices (such as a handful of routers and servers) feed the trunk lines. When this occurs, the limited number of MAC address combinations encountered results in a lopsided traffic distribution that can reduce the effective combined bandwidth of the trunked ports.

By adding Layer 3 IP address information to the distribution algorithm, a far wider variety of address combinations is seen. Even with just a few routers feeding the trunk, the normal source/destination IP address combinations (even within a single LAN) can be widely varied. This results in a wider statistical load distribution and maximizes the use of the combined bandwidth available to trunked ports.

Built-In Fault Tolerance

Since each trunk group is comprised of multiple physical links, the trunk group is inherently fault tolerant. As long as one connection between the switches is available, the trunk remains active.

Statistical load balancing is maintained whenever a port in a trunk group is lost or returned to service.

Port Trunking Example

In this example, three ports will be trunked between two ACEswitch 180s.

Prior to configuring each switch in this example, you must connect to the appropriate switch's command-line interface as the administrator (see Chapter 2 of the *WebOS Command Reference*).

NOTE – For details about any of the menu commands described in this example, see "Trunk Configuration" in Chapter 7 of the *WebOS Command Reference*.

1. **Connect the switch ports which will be involved in the trunk group:**

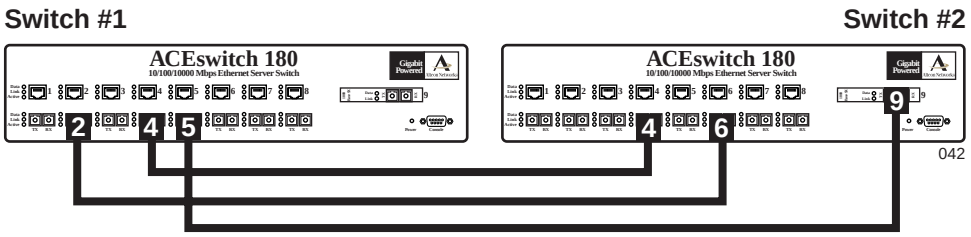


Figure 14-2 Example Port Trunk Group Configuration

2. **On Switch #1, define a Trunk Group.**

```
>> Main # /cfg/trunk 1                    (Select trunk group #1)
>> Trunk group 1# add 2                   (Add port 2 to trunk group #1)
>> Trunk group 1# add 4                   (Add port 4 to trunk group #1)
>> Trunk group 1# add 5                   (Add port 5 to trunk group #1)
>> Trunk group 1# ena                     (Enable trunk group #1)
```

3. **On Switch #1, apply and verify the configuration.**

```
>> Trunk group 1# apply                   (Make your changes active)
>> Trunk group 1# cur                     (View current trunking configuration)
```

Examine the resulting information. If any settings are incorrect, make the appropriate changes.

4. **On Switch #1, save your new configuration changes.**

```
>> Trunk group 1# save                    (Save for restore after reboot)
```

5. **On Switch #2, repeat the process.**

```
>> Main # /cfg/trunk 3                    (Select trunk group #3)
>> Trunk group 3# add 4                   (Add port 4 to trunk group #3)
>> Trunk group 3# add 6                   (Add port 6 to trunk group #3)
>> Trunk group 3# add 9                   (Add port 9 to trunk group #3)
>> Trunk group 3# ena                     (Enable trunk group #3)
>> Trunk group 3# apply                   (Make your changes active)
>> Trunk group 3# cur                     (View current trunking configuration)
>> Trunk group 3# save                    (Save for restore after reboot)
```

Switch #1 trunk group #1 is now connected to Switch #2 trunk group #3.

NOTE – In this example, both switches are Alteon WebSystems' Web switches. If a third-party device supporting link aggregation is used (such as Cisco routers and switches with EtherChannel technology, or Sun's Quad Fast Ethernet Adapter), then trunk groups on the third-party device should be configured manually. Connection problems could arise when using automatic trunk group negotiation on the third-party device.

---

6. **Examine the trunking information on each switch.**

```
>> /info/trunk                            (View trunking information)
```

Information about each port in each configured trunk group will be displayed. Make sure that trunk groups consist of the expected ports, and that each port is in the expected state.

The following restrictions apply:

- Any physical switch port can belong to no more than one trunk group.
- Up to four ports can belong to the same trunk group.
- Best performance is achieved when all ports in any given trunk group are configured for the same speed.
- Trunking from non-Alteon WebSystems' devices must comply with Cisco® EtherChannel® technology.

Glossary

RIP (Real Server)	An IP addresses that the switch load balances to when requests are made to a virtual IP address (VIP).
Real Server Group	Group of real servers that are associated with a VIP or filter.
VIP (Virtual IP Address)	An IP address that the switch owns and uses to load balance particular service requests (like HTTP) to other servers.
SIP (Source IP Address)	The source IP address of a frame.
DIP (Destination IP Address)	The destination IP address of a frame.
SPort (Source Port)	The source port (application socket; for example, HTTP-80/HTTPS-443/DNS-53).
Dport (Destination Port)	The destination port (application socket; for example, http-80/https-443/DNS-53)
Proto (Protocol)	The protocol of a frame. Can be any value represented by a 8-bit value in the IP header adherent to the IP specification (for example, TCP, UDP, OSPF, ICMP, and so on.)
NAT (Network Address Translation)	Any time an IP address is changed from one SIP or DIP to another address, network address translation can be said to have taken place. In general, half NAT is when the DIP or SIP is changed from one address to another and full NAT is when both addresses are changed from one address to another. VIP based load balancing uses half NAT by design since it NAT's the DIP (destination IP address) from the VIP (virtual IP address) to that of one of the RIP's (real servers).

Virtual Server Load Balancing

Classic load balancing. Requests destined for a virtual server IP address (VIP), which is owned by the switch, are load balanced to a real server contained in the group associated with the VIP. Network address translation is done back and forth, by the switch, as requests come and go.

Frames come to the switch destined for the VIP. The switch then replaces the VIP and with one of the real server IP addresses (RIP's), updates the relevant checksums, and forwards the frame to the server for which it's now destined. This process of replacing the destination IP (VIP) with one of the real server addresses is called half NAT. If the frames were not half NAT'ed to the address of one of the RIPs, a server would receive the frame that was destined for it's MAC address, forcing the packet up to Layer 3. The server would then drop the frame, since the packet would have the DIP of the VIP and not that of the server (RIP).

Redirection or Filter-Based Load Balancing

A type of load balancing; one that operates differently from VIP-based load balancing. With this type of load balancing, requests are transparently intercepted and "redirected" to a server group. "Transparently" meaning that requests are not specifically destined for a VIP that the switch owns. Instead, a filter is configured in the switch. This filter intercepts traffic based on certain IP header criteria and load balances it.

Filters can be configured to filter on the SIP/Range (via netmask), DIP/Range (via netmask), Protocol, SPort/Range or DPort/Range. The action on a filter can be Allow, Deny, Redirect to a Server Group, or NAT (either the SIP or DIP). When doing redirection based load balancing the DIP is NOT NAT'ed to that of one of the real servers. Therefore, redirection based load balancing is designed to be used to load balance devices that normally operate transparently in your network such as a firewall, spam filter, or transparent Web cache.

VRRP (Virtual Router Redundancy Protocol)

A protocol that acts very similarly to Cisco's proprietary HSRP address sharing protocol. The reason for both of these protocols is so devices have a next hop or default gateway that is always available. The way it works is two or more devices sharing an IP interface are either advertising or listening for advertisements. These advertisements are sent via a broadcast message to address 224.0.0.18.

With VRRP one switch is considered the master and the other the backup. The master is always advertising via the broadcasts. The backup switch is always listening for the broadcasts. Should the master stop advertising the backup will take over ownership of the VRRP IP and MAC addresses as defined by the specification. The switch announces this change in ownership to the devices around it by way of a Gratuitous ARP and advertisements. If the backup switch didn't do the Gratuitous ARP the Layer 2 devices attached to the switch would not know that the MAC address had moved in the network. For a more detailed description, refer to RFC 2338.

Virtual Router

A shared address between two devices utilizing VRRP, as defined in RFC 2338. One virtual router is associated with an IP interface. This is one of the IP interfaces that the switch is assigned. All IP interfaces on the Alteon switches must be in a VLAN. If there is more than one VLAN defined on the switch then the VRRP broadcasts will only be sent out on the VLAN for which the associated IP interface is a member of.

Priority

The value given to a Virtual Router to determine it's ranking with it's peer(s). Minimum value is 1 and maximum value is 254. Default is 100. A higher number will win out for master designation.

VRID (Virtual Router Identifier)

A value between 1 and 255 that is used by each virtual router to create it's MAC address and identify it's peer for which it is sharing this VRRP address. The VRRP MAC address as defined in the RFC is 00-00-5E-00-01-{VRID}. If you have a VRRP address that two switches are sharing, then the VRID number needs to be identical on both switches so each virtual router on each switch knows who to share with.

VIR (Virtual Interface Router)

A VRRP address that is an IP interface address shared between two or more virtual routers.

VSR (Virtual Server Router)

A VRRP address that is a shared VIP address. This is Alteon's proprietary extension to the VRRP spec. The switches must be able to share a VIPs as well as IP interfaces. If they didn't the two switches would fight for ownership of the VIP and the ARP tables in the devices around them would get very confused since they would have two ARP entries with the same IP address but different MAC addresses.

Preemption

Pre-emption will cause a Virtual Router that has a lower priority to go into backup should a peer Virtual Router start advertising with a higher priority.

Tracking

A method to increase the priority of a virtual router and thus master designation (with preemption enabled). Tracking can be very valuable in an active/active configuration.

You can track the following:

- Vrs: Virtual Routers in Master Mode (increments priority by 2 for each)
- Ifs: Active IP interfaces on the switch (increments priority by 2 for each)
- Ports: Active ports on the same VLAN (increments priority by 2 for each)
- l4pts: Active Layer 4 Ports, client or server designation (increments priority by 2 for each
- reals: healthy real servers (increments by 2 for each healthy real server)
- hsrp: HSRP announcements heard on a client designated port (increments by 10 for each)

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W