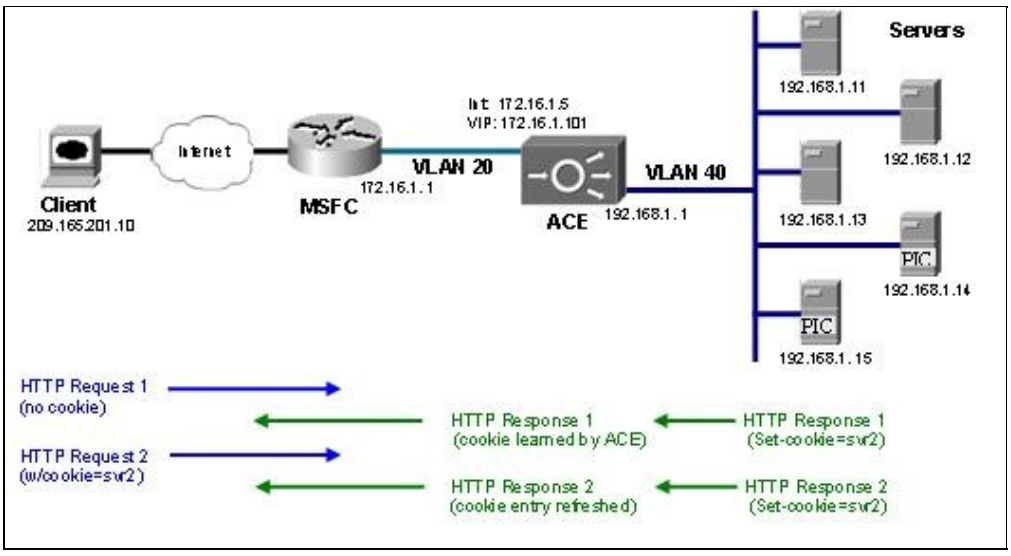
Contents

Goal

Configure basic load balancing with cookie learning where client traffic enters on one network and is directed to servers residing on a second network. Once the client has entered the site they will remain stuck to a given server based on a HTTP Cookie learned by ACE from the server.

Design

Clients will send application requests through the multilayer switch feature card (MSFC), which routes them to a virtual IP address (VIP) within the Cisco® Application Control Engine (ACE). The VIP used in this example resides in an ACE context, which is configured with a client VLAN and a server VLAN (see the figure below). Client requests will arrive at the VIP, and the ACE will check the request to see if it contains a cookie. If it does, the sticky table will be checked to see which server should receive the request. If the cookie entry has expired, or if the client does not have a cookie the ACE will pick the appropriate server to receive the request based on the requested URL. When the server responds ACE will learn the cookie value found within the HTTP Response so that upon future requests client persistence to the server will be maintained.



Within the Cisco ACE sticky resources are finite and are controlled via resource allocation. Before a context can apply session persistence using sticky groups, the context must first be given a sticky allocation. Once this is done, a sticky group is created to define parameters and the serverfarm where client requests will be sent. Recall, the load balancing action tells ACE how to handle traffic which has hit a VIP. Thus the sticky group on ACE is applied within the load balance policy-map. To enable server load-balancing with session persistence based on cookies ACE learns you need to do the following:

- Allocate sticky resources to the context
- Enable ACLs to allow data traffic through the ACE device, as it is denied by default.
- Configure the IPs of the servers (define rservers)
- Group the real servers (create a serverfarm)
- Create a sticky group
- Define the virtual IP address (VIP)
- Define how traffic is to be handled as it is received (create a policy map for load balancing)
- Apply the sticky group to the load balancing policy
- Associate a VIP to a handling action (create a multimatch policy map [a service policy])
- Create client- and server-facing interfaces
- Apply the VIP and ACL permitting client connections to the interface (apply access group and service policy to interface)

Note: For brevity only the bold steps are covered in this document. Please review the URL Load Balancing using Routed Mode document for more information on basic URL load balancing and the base configuration.

To begin the configuration, allocate sticky resources to the context you will be using. In this example a context ?routed? has already been defined. Create a resource class, allocate the desired amount of sticky entries, and apply them to the ?routed? context.

```
ACE-1/Admin# show run | begin routed
context routed
  allocate-interface vlan 10
  allocate-interface vlan 20
allocate-interface vlan 40

ACE-1/Admin(config)# resource-class sticky
ACE-1/Admin(config-resource)# limit-resource all minimum 0.00 maximum unlimited
ACE-1/Admin(config-resource)# limit-resource sticky minimum 10.00 maximum equal-to-min

ACE-1/Admin(config)# context routed
ACE-1/Admin(config-context)# member sticky
```

Note: At this time sticky resources are a fixed resource outside the other resources. Thus you must use maximum equal-to-min and you must define sticky resources even if **all** is previously used. This also applies to the Admin context.

Once the resources have been allocated a sticky group can be defined. The Cisco ACE can be configured in various ways to apply session persistence using cookies. For this example cookie learning will be used. The cookie name ACE will attempt to learn is supplied when the sticky group is created. By default ACE learns cookies and these entries have a idle timeout of 24 hours. Using the configuration below ACE will learn the value of a cookie with the name ?JSessionID?. The session will timeout after 5 minutes of no new connections/requests containing this cookie value. The ACE will use a pre-existing serverfarm named ?webfarm?.

```
ACE-1/routed(config)# sticky http-cookie JSessionID web-sticky
ACE-1/routed(config-sticky-cookie)# timeout 5
ACE-1/routed(config-sticky-cookie)# serverfarm webfarm
```

The serverfarm within the load balancing policy map must be swapped with the sticky group to apply cookie learning to new client requests. Based on this example configuration there are two possible actions for handling new client requests. Any requests for images will be sent to the ?imagefarm? serverfarm, which does not require persistence. All other requests will be sent to the web servers in the ?webfarm? where the clients will use session persistence.

```
ACE-1/routed(config)# policy-map type loadbalance http first-match slb-logic
ACE-1/routed(config-pmap-lb-c)# class class-default
```

```
ACE-1/routed(config-pmap-lb-c)# no serverfarm webfarm
ACE-1/routed(config-pmap-lb-c)# sticky-serverfarm web-sticky
```

Note: Changing from a serverfarm to a sticky group is potentially service impacting. While current connections will be allowed to finish, new connections will not be accepted during the removal of the serverfarm and applying the sticky group to the load balance policy.

When cookie learning is applied there are no entries in the http-cookie sticky database. As new connections arrive, and cookies are learned new entryies for each client will be stored in the sticky database for http-cookies. In this example there are two entries created.

```
ACE-1/routed(config-pmap-lb-c)# do show sticky database http-cookie
sticky group : web-sticky
type         : HTTP-COOKIE
timeout      : 5             timeout-activeconns : FALSE
  sticky-entry          rserver-instance      time-to-expire flags
  --------------------+---------------------+-------------+-------+
  16820511103801384579  lnx1:0                  0         -
sticky group : web-sticky
type         : HTTP-COOKIE
timeout      : 5             timeout-activeconns : FALSE
  sticky-entry          rserver-instance      time-to-expire flags
  --------------------+---------------------+-------------+-------+
  3347854103021350619   lnx2:0                  0         -
```

Note: The ?sticky-entry? is a hash of the cookie-value set by ACE for the real server. It will not appear in a sniffer trace or on the client browser. See the comment section below for determining which server a client is associated to via the cookie value.

When a client connects to the VIP they download index.html page from the web servers. At this point the client will get a cookie for the selected server to ensure all future requests will continue to use the same server. While there is no way to see how many clients have a cookie for a given server the common show commands will provide details on the incoming requests and responses and other data stats.

Recall, the Cisco ACE inserts a static entry for each server in a server farm in the sticky group. In this case there is only one sticky group and it uses cookie-insert and the serverfarm contains 2 real servers. Thus the output of the show stats sticky command displays 2 entries, despite the number of clients using the VIP.

```
ACE-1/routed# show stats sticky

+----------------------------------------+
+----------- Sticky statistics ----------+
+----------------------------------------+
 Total sticky entries reused    : 0
 prior to expiry
 Total active sticky entries    : 3
 Total active sticky conns      : 0
 Total static sticky entries    : 0
```

Related show commands

This section provides information you can use to confirm your configuration is working properly.

Certain show commands are supported by the Output Interpreter Tool (registered customers only), which allows you to view an analysis of show command output.

```
ACE-1/routed #show sticky database static
```

```
ACE-1/routed #show service-policy client-vips
ACE-1/routed #show service-policy client-vips detail
ACE-1/routed #show serverfarm
ACE-1/routed #show rserver
ACE-1/routed #show stats
```

Comments

The type of cookie ACE will insert depends on if cookie-insert is configured or cookie-insert browser-expire is configured. The sniffer traces below show the difference between the types of cookies ACE will insert.

ACE configured with ?cookie insert?

```
Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
    Set-Cookie: JsessionID=128748457485485489; path=/; expires=Thu,
16-Oct-2008 00:08:55 GMT\r\n
    Date: Fri, 29 Aug 2008 19:17:35 GMT\r\n
    Server: Apache/2.0.52 (Red Hat)\r\n
    Accept-Ranges: bytes\r\n
    Content-Length: 1112\r\n
    VirtualHost: 192.168.1.12\r\n
    Connection: close\r\n
    Content-Type: text/html; charset=UTF-8\r\n
    \r\n
```

Notice the two cookie values do not appear to be related to the sticky-value in the show sticky database static output, displayed below:

```
  sticky-entry          rserver-instance      time-to-expire flags
  --------------------+----------------------+--------------+-------+
  16820511103801384579  lnx1:0                  never         -
     <?snip?>
  3347854103021350619   lnx2:0                  never         -
```

This is because the sticky-entry is a hash of the cookie value. In order to determine the server a client is hitting one must mark the server such that it can be distinguished, possibly using a HTTP Header in the server response, or by determining the sticky-entry ACE uses for each learned cookie entry. The sticky-entry is a hash of the first 255 bytes of the cookie value. The ACE provides a command which takes the cookie value set by the server and displays the cookie entry. This provides the association between a client and a real server.

```
ACE-1/routed(config-context)# show sticky database http-cookie ?128748457485485489?

sticky group : web-sticky
type         : HTTP-COOKIE
timeout      : 5             timeout-activeconns : FALSE
  sticky-entry          rserver-instance         time-to-expire flags
  --------------------+-----------------------+--------------+-------+
  497835028048248915    lnx2:0                   254            -
```

show running-config

```
ACE-1/routed# show run
Generating configuration....
```

```
access-list everyone line 8 extended permit ip any any
access-list everyone line 16 extended permit icmp any any

rserver host lnx1
  ip address 192.168.1.11
  inservice
rserver host lnx2
  ip address 192.168.1.12
  inservice
rserver host lnx3
  ip address 192.168.1.13
  inservice
rserver host lnx4
  ip address 192.168.1.14
  inservice

serverfarm host imagefarm
  rserver lnx3
    inservice
  rserver lnx4
    inservice
serverfarm host webfarm
  rserver lnx1
    inservice
  rserver lnx2
    inservice

sticky http-cookie ACE-Insert web-sticky
  timeout 5
serverfarm webfarm

class-map type http loadbalance match-all images
  2 match http url /images/.*

class-map match-all slb-vip
  2 match virtual-address 172.16.1.101 any

policy-map type management first-match remote-access
  class class-default
    permit

policy-map type loadbalance http first-match slb
  class images
    serverfarm imagefarm
  class class-default
    sticky-serverfarm web-sticky

policy-map multi-match client-vips
  class slb-vip
    loadbalance vip inservice
    loadbalance policy slb

interface vlan 20
  description "Client Side"
  ip address 172.16.1.5 255.255.255.0
  access-group input everyone
  service-policy input client-vips
  no shutdown

interface vlan 40
  description "Default gateway of real servers"
  ip address 192.168.1.1 255.255.255.0
  service-policy input remote-access
  no shutdown
```

```
ip route 0.0.0.0 0.0.0.0 172.16.1.1
```

Related Information

Technical Support & Documentation - Cisco Systems