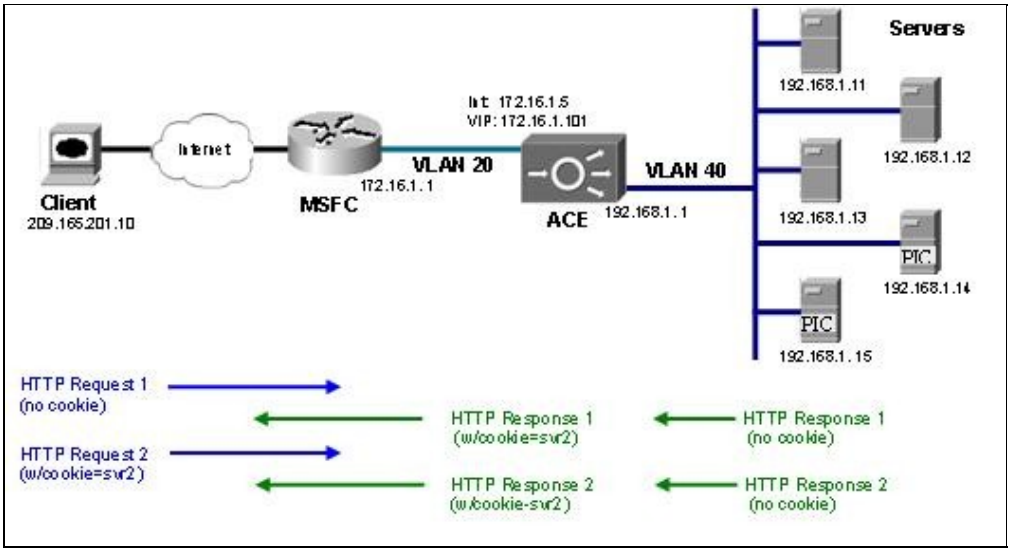
Contents

Goal

Configure basic load balancing with cookie insert 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 inserted by ACE.

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 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 insert a cookie into the HTTP Response to 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 inserts 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 is 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 insert will be used. The cookie name ACE will insert is supplied when the sticky group is created. By default ACE inserts permanent cookies which have a timeout of 24 hours. Using the configuration below ACE will insert a cookie with the name ?ACE-Insert?, it will have a timeout of 5 minutes, and use a pre-existing serverfarm named ?webfarm?.

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

Note: If session cookies (cookies that are removed when the client closes the browser) are preferred then ?cookie insert browser-expire? can be configured.

The serverfarm within the load balancing policy map must be swapped with the sticky group to apply cookie-insert sticky 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 insert is applied permanent entries are inserted into the static sticky database for each real server defined in the serverfarm. In this example there are two entries created.

```
ACE-1/routed(config-pmap-lb-c)# do show sticky database static
sticky group : web-sticky
type         : HTTP-COOKIE
timeout      : 5             timeout-activeconns : FALSE
  sticky-entry          rserver-instance      time-to-expire flags
  --------------------+----------------------+--------------+-------+
  16820511103801384579  lnx1:0                  never         -
sticky group : web-sticky
type         : HTTP-COOKIE
timeout      : 5             timeout-activeconns : FALSE
  sticky-entry          rserver-instance      time-to-expire flags
  --------------------+----------------------+--------------+-------+
  3347854103021350619   lnx2:0                  never         -
```

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    : 2
 Total active sticky conns      : 0
 Total static sticky entries    : 2
```

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: ACE-Insert=R1005880048; 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
```

ACE configured with ?cookie insert browser-expire?

```
Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
    Set-Cookie: ACE-Insert=R1005881137; path=/\r\n
    Date: Fri, 29 Aug 2008 19:31:58 GMT\r\n
    Server: Apache/2.0.52 (Red Hat)\r\n
    Accept-Ranges: bytes\r\n
    Content-Length: 353\r\n
    VirtualHost: 192.168.1.11\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 cookie value ACE uses for each server. The cookie value is a number based on the serverfarm name, rserver name, and rserver port. The following script can be used to determine the values associated with each rserver.

TCL Script for calculating Cookie Values

Note: This script works for 32-bit OS only. If you are using this on a 64-bit OS, you will need to modify it to produce a 32-bit unsigned integer.

```
[root@host cookie]# ./ace-cookie-value.tcl webfarm lnx1 0
Value: R1005880048
[root@host cookie]# ./ace-cookie-value.tcl webfarm lnx2 0
Value: R1005881137

#!/usr/bin/tclsh

##########################################################################
#                                                                        #
# Name: ace-cookie-value.tcl                                             #
#                                                                        #
# This script takes the serverfarm name, the real server name and the #
# port number used by ACE and returns the generated hash that will be #
# used for cookie insertion.                                             #
#                                                                        #
##########################################################################

if { $argc != 3 } {
    puts "[info script] <serverfarm> <realname> <port>"
    exit 0
}

set serverFarmName [lindex $argv 0]
set realServerName [lindex $argv 1]
set port [lindex $argv 2]

set hashValue 5381
set hashMultiplier 32

set cookieInsertStr "$serverFarmName:$realServerName:$port"

set len [ string length $cookieInsertStr ]

for { set ix 0 } { $ix < $len } { incr ix } {
    set hashValue [expr (($hashValue * $hashMultiplier) + $hashValue) \
        + [scan [string index $cookieInsertStr $ix] %c]]
}

puts [format "Value: R%u" $hashValue]
```

VBA Script for calculating Cookie Values

The following script was created by an ACE customer and shared for the convenience of all ACE users. We would like to sincerely thank Herve Benattar for his contribution to the ACE Doc Wiki!

```
Dim Result_Cookie As String


Function Calculating_Cookie_Name(ByVal CHAINE1 As String)
```

```
' For this script, we need to work with 32 bits unsigned integer.
' VBA does not support unsigned int of 32 bits. An integer is only 16 bits
' and a Long var is 32 bits signed. To simulate a 32 bits unsigned  integer,
' we use an double and subtract all numbers with the tops from the 32th bit

    Dim Dbl_64bits_Tmp1, Dbl_64bits_Tmp2, hashValue, hashMultiplier As Double
    Dim ix As Integer

    hashValue = 5381
    hashMultiplier = 32
    ix = 0

    Lng_Chaine = Len(CHAINE1)

For ix = 0 To (Lng_Chaine - 1) Step 1
    'MAX Value 4294967295*32+5381 = 137438958821
    Dbl_64bits_Tmp2 = hashValue * hashMultiplier + hashValue
    'MAX value 127 or 255 with Extended ASCII Codes
    Dbl_64bits_Tmp1 = CDbl(Asc(Mid(CHAINE1, ix + 1, 1)))
    'MAX 137438958821+255 = 137438959076 = 100000 00000000000000000010101111100100(2)
    hashValue = Dbl_64bits_Tmp2 + Dbl_64bits_Tmp1

Try_Again:
    Select Case hashValue
        '39th bits to 1 (Normaly none use with MAX)
        Case Is >= 545460846592#
            hashValue = hashValue - 545460846592#
            GoTo Try_Again
        '38th bits to 1
        Case Is >= 270582939648#
            hashValue = hashValue - 270582939648#
            GoTo Try_Again
        '37th  bits to 1
        Case Is >= 133143986176#
            hashValue = hashValue - 133143986176#
            GoTo Try_Again
        '36th  bits to 1
        Case Is >= 64424509440#
            hashValue = hashValue - 64424509440#
            GoTo Try_Again
        '35th  bits to 1
        Case Is >= 30064771072#
             hashValue = hashValue - 30064771072#
             GoTo Try_Again
        '34eme bits to 1
        Case Is >= 12884901888#
            hashValue = hashValue - 12884901888#
            GoTo Try_Again
        '33eme bits to 1
        Case Is >= 4294967296#
             hashValue = hashValue - 4294967296#
             GoTo Try_Again
    End Select

    'MsgBox "Value: R" & hashValue

Next

Result_Cookie = "R" & hashValue

End Function

Private Sub CommandButton1_Click()
```

```
    ActiveSheet.Range("A2").Select

    While (Not IsEmpty(ActiveCell.Value))
        serverFarmName = ActiveCell.Value
        realServerName = ActiveCell.Offset(0, 1).Value
        port = ActiveCell.Offset(0, 2).Value

        cookieInsertStr = serverFarmName & ":" & realServerName & ":" & port

        Calculating_Cookie_Name (cookieInsertStr)
        ActiveCell.Offset(0, 3) = Result_Cookie
        ActiveCell.Offset(1, 0).Select
    Wend

End Sub
```

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
  cookie insert browser-expire
  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

Hash output Examples

For the developer we have provided the following outputs as the hash is being calculated to all you to check your code against a working example.

```
[root@atl-tme-linux cookie]# ./ace-cookie-value.tcl WEB-FARM apache1 80
Value: R177660
Value: R5862849
Value: R193474083
Value: R2089677488
Value: R239880438
Value: R3621087223
Value: R3531761449
Value: R584010902
Value: R2092490640
Value: R332714481
Value: R2389643393
Value: R1548820738
Value: R3866444197
Value: R3038607021
Value: R1489783986
Value: R1918231331
Value: R3172091837
Value: R1599815573
Value: R1254306405
Final Value: R1254306405

[root@atl-tme-linux cookie]# ./ace-cookie-value.tcl WEB-FARM apache2 80
Value: R177660
Value: R5862849
```

```
Value: R193474083
Value: R2089677488
Value: R239880438
Value: R3621087223
Value: R3531761449
Value: R584010902
Value: R2092490640
Value: R332714481
Value: R2389643393
Value: R1548820738
Value: R3866444197
Value: R3038607021
Value: R1489783986
Value: R1918231332
Value: R3172091870
Value: R1599816662
Value: R1254342342
Final Value: R1254342342

[root@atl-tme-linux cookie]# ./ace-cookie-value.tcl WEB-FARM iis1 80
Value: R177660
Value: R5862849
Value: R193474083
Value: R2089677488
Value: R239880438
Value: R3621087223
Value: R3531761449
Value: R584010902
Value: R2092490640
Value: R332714489
Value: R2389643650
Value: R1548829237
Value: R3866724614
Value: R3047860736
Value: R1795156536
Value: R3405590888
Final Value: R3405590888

[root@atl-tme-linux cookie]# ./ace-cookie-value.tcl WEB-FARM iis2 80
Value: R177660
Value: R5862849
Value: R193474083
Value: R2089677488
Value: R239880438
Value: R3621087223
Value: R3531761449
Value: R584010902
Value: R2092490640
Value: R332714489
Value: R2389643650
Value: R1548829237
Value: R3866724615
Value: R3047860769
Value: R1795157625
Value: R3405626825
Final Value: R3405626825

[root@atl-tme-linux cookie]# ./ace-cookie-value.tcl Tier2  App1 80
Value: R177657
Value: R5862786
Value: R193472039
Value: R2089610105
Value: R237656779
Value: R3547706469
```

```
Value: R1110196550
Value: R2276747894
Value: R2118236582
Value: R1182330519
Value: R362201521
Value: R3362715657
Value: R3595434329
Final Value: R3595434329

[root@atl-tme-linux cookie]# ./ace-cookie-value.tcl Tier2  App2 80
Value: R177657
Value: R5862786
Value: R193472039
Value: R2089610105
Value: R237656779
Value: R3547706469
Value: R1110196550
Value: R2276747894
Value: R2118236582
Value: R1182330520
Value: R362201554
Value: R3362716746
Value: R3595470266
Final Value: R3595470266
```