La solución de problemas de GUP, de puntos finales alternativos y de equlibrio de carga.

Contenido

Introducción

Este documento le ayudará a resolver problemas y a entender estas funciones del gatekeeper de Cisco.

- Agrupamiento del control de acceso y Protocolo de actualización del control de acceso (GUP)
- Puntos finales alternativos
- Equilibrio de carga

Refiera al portero de alto rendimiento de Cisco para toda la información necesaria sobre estas características incluyendo la descripción general de características, las plataformas admitidas, las versiones de software de Cisco IOS® necesarias y cómo configurarlas, monitorear, y mantener.

prerrequisitos

Requisitos

Quienes lean este documento deben tener conocimiento de lo siguiente:

- Conocimiento básico de la funcionalidad de gatekeeper.
- Conocimientos básicos sobre VoIP, H.323 y señalización de registro, admisión y estado (RAS).

Componentes Utilizados

La información que contiene este documento se basa en las versiones de software y hardware indicadas a continuación.

- Cisco IOS Software Release 12.3(4)T1
- Gatewayes de Cisco: Cisco AS5300, Cisco AS5400, y Cisco 3725
- Gatekeeperes de Cisco: Cisco 3725 y Cisco 2611

La información que se presenta en este documento se originó a partir de dispositivos dentro de un ambiente de laboratorio específico. Todos los dispositivos que se utilizan en este documento se pusieron en funcionamiento con una configuración verificada (predeterminada). Si la red está funcionando, asegúrese de haber comprendido el impacto que puede tener un comando antes de ejecutarlo.

Convenciones

Para obtener más información sobre las convenciones del documento, consulte Convenciones de Consejos Técnicos de Cisco.

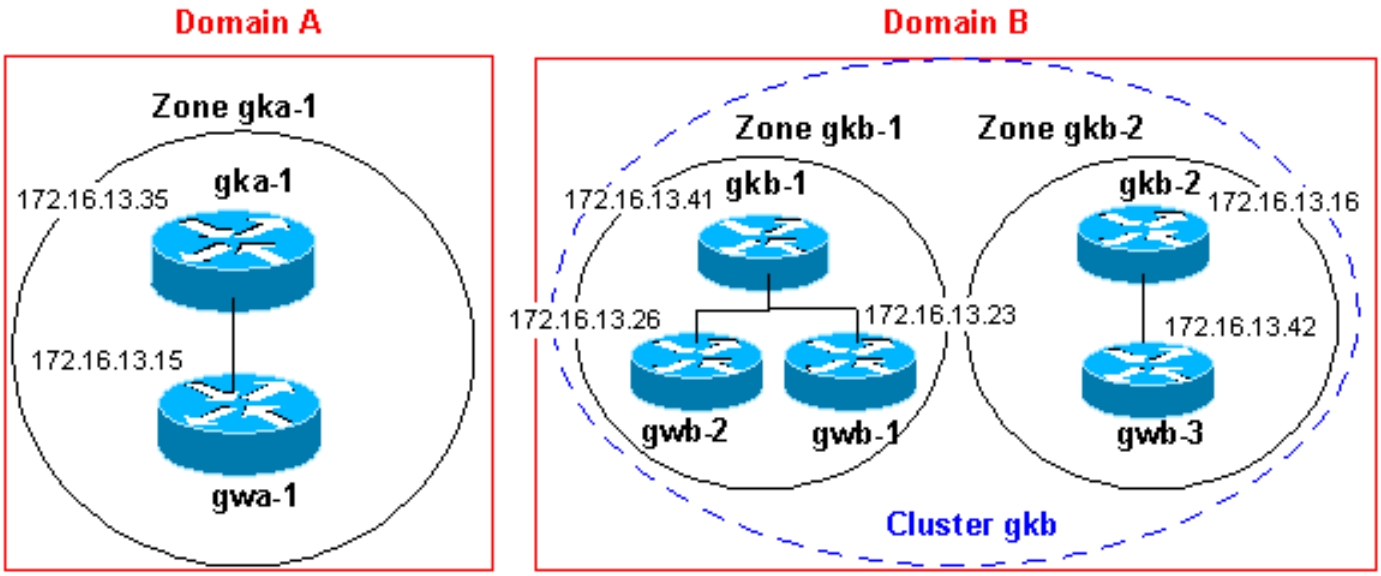
Definiciones

Término	Definición
ARQ	La solicitud de admisión (ARQ) es un mensaje RAS enviado desde un punto final de Cisco H.323 a una puerta de enlace que solicita admisión para establecer una llamada.
ACF	La Confirmación de admisión es un mensaje RAS enviado desde el gatekeeper al punto final que confirma la aceptación de una llamada.
ARJ	El rechazo de admisión (ARJ) es un mensaje RAS del control de acceso al punto final que rechaza la solicitud de admisión.
GCF	El Gatekeeper Confirm (GCF) es un mensaje RAS enviado de un portero al punto final de Cisco H.323 que confirma la detección del portero.
GR	El Gatekeeper Request (GRQ) es un mensaje RAS

Q	enviado de un punto final de Cisco H.323 para descubrir al portero.
GUP	El Gatekeeper Update Protocol se utiliza para compartir la información entre los controles de acceso en un agrupamiento y sus puntos finales y llamadas activas.
LCF	El mensaje de confirmación de ubicación, Location Confirm (LCF), es un mensaje del RAS enviado de un control de acceso a otro que confirma la Solicitud de ubicación (LRQ) e incluye la dirección IP del punto final de terminación.
LRJ	El Location Reject (LRJ) es un mensaje RAS enviado a partir de un portero a otro que rechace el LRQ.
LRQ	Location Request (Solicitud de ubicación) es un mensaje RAS de una de los controles de acceso al otro que solicita la dirección IP de un punto final de terminación remoto.
RAS	El protocolo RAS permite que un gatekeeper realice el registro, la admisión y la verificación de estado del punto final.
RCF	El registro confirma (RCF) es un mensaje RAS enviado del portero al punto final que confirma el registro.
RRJ	El Rechazo de registro (RRJ) es un mensaje RAS enviado desde el gatekeeper que rechaza la petición de registro.
RRQ	Solicitud de registro (RRQ) es un mensaje RAS enviado desde un punto final al gatekeeper que solicita registrarse con él.
URQ	La Solicitud de eliminación de registro (URQ) es un mensaje enviado desde un punto final al gatekeeper que solicita eliminar su registro con él.

Topología de laboratorio y configuraciones

Para explicar cómo las características trabajan y cómo resolver problemas, una configuración de laboratorio fue construida con esta topología:



Topología de laboratorio y configuraciones

En la siguiente tabla encontrará las configuraciones básicas de todos los gateways y gatekeepers. En casos diferentes, fue necesario realizar cierto cambio en la configuración. Se indica el cambio cuando sucede éste. Las siguientes configuraciones sólo incluyen las partes esenciales para la funcionalidad del gateway o el control de acceso para este laboratorio.

Las configuraciones del "gwb-1" y el "gwb-2" son casi similares (a excepción de la dirección IP y de H.323 ID). Por lo tanto, `gwb-1` se muestra solamente abajo.

gwa-1

```
!
controller E1 3/0
pri-group timeslots 1-2,16
!
interface Ethernet0/0
 ip address 172.16.13.15 255.255.255.224
 half-duplex h323-gateway voip interface
 h323-gateway voip id gka-1 ipaddr 172.16.13.35 1718
 h323-gateway voip
 h323-id gwa-1
 h323-gateway voip tech-prefix 1#
!
voice-port 3/0:15
!
!
dial-peer voice 5336 pots
 incoming called-number
 destination-pattern 5336
 direct-inward-dial
 port 3/0:15
 prefix 21
!
dial-peer voice 3653 voip
 incoming called-number
 destination-pattern 3653
 session target ras
 dtmf-relay h245-alphanumeric
 codec g711ulaw
!
```

```
gateway
!
ntp clock-period 17178794
ntp server 172.16.13.35
end
```

gka-1

```
!
gatekeeper
 zone local gka-1 domainA.com 172.16.13.35
 zone remote gkb domainB.com 172.16.13.41 1719
 zone prefix gkb 36*
 zone prefix gka-1 53*
 gw-type-prefix 1#* default-technology
 no shutdown
!
no scheduler
max-task-timentp master
!
end
```

gwb-1

```
!
controller E1 0
 clock source line primary
 ds0-group 0 timeslots 1-2 type r2-digital r2-compelled
!
interface Ethernet0
 ip address 172.16.13.23 255.255.255.224
 h323-gateway voip interface
 h323-gateway voip id gkb-1 ipaddr 172.16.13.41 1718
 h323-gateway voip h323-id gwb-1
 h323-gateway voip tech-prefix 2#
!
dial-peer voice 3653 pots incoming called-number
 destination-pattern 3653
 port 0:0
 prefix 21
!
dial-peer voice 5336 voip
 incoming called-number
 destination-pattern 5336
 session target ras
 dtmf-relay h245-alphanumeric
 codec g711ulaw
!
gateway
!
ntp clock-period 17179389
ntp server 172.16.13.35
end
```

gwb-3

```
!
interface Ethernet0/0
 ip address 172.16.13.42 255.255.255.224
 half-duplex h323-gateway voip interface
 h323-gateway voip id gkb-1 ipaddr 172.16.13.41 1718
 h323-gateway voip
 h323-id gwb-3
 h323-gateway voip tech-prefix 1#
!
voice-port 3/0/0
!
```

```
voice-port 3/0/1
!
dial-peer voice 3653 pots
 destination-pattern 3653
 port 3/0/0
 prefix 21
!
dial-peer voice 5336 voip
 incoming called-number
 destination-pattern 5336
 session target ras
 dtmf-relay h245-alphanumeric

 codec g711ulaw

gateway ! ntp clock-period 17179181 ntp server
172.16.13.35
!
end
```

gkb-1

```
!
gatekeeper
 zone local gkb-1 domainB.com 172.16.13.41
 zone remote gka-1 domainA.com 172.16.13.35 1719
 zone cluster local gkb gkb-1
 element gkb-2 172.16.13.16 1719
 gw-type-prefix 2#* default-technology
 no shutdown
!
ntp clock-period 17179580
ntp server 172.16.13.35
!
end
```

gkb-2

```
!
gatekeeper
 zone local gkb-2 domainB.com 172.16.13.16
 zone cluster local gkb gkb-2
 element gkb-1 172.16.13.41 1719
!
no shutdown
!
ntp clock-period 17179199
ntp server 172.16.13.35
!
end
```

Control de acceso alternativo

En las versiones anteriores a la versión 2 de Cisco H.323, cada zona sólo estaba controlada por un gatekeeper único. La versión 2 de H.323 de Cisco introduce la idea de un "gatekeeper alternativo" para proporcionar la redundancia del gatekeeper. La implementación de la opción de gatekeeper alternativo permite que múltiples gatekeepers controlen una zona. Cuando un punto final se registra con un gatekeeper, se ofrece junto a una lista de gatekeepers alternativos para la zona en la cual se registra el punto final y para la cual se especificaron reemplazantes utilizando el CLI. Si el portero falla, el punto final puede utilizar a los gatekeeperes alternativos para continuar la operación.

Se le ofrece la lista de gatekeepers alternativos al gatekeeper de Cisco a través del CLI para cada zona y se transmite hacia los puntos finales a través de los mensajes RCF (incluido Lightweight) y GRQ. Esta lista se puede también transmitir en otros mensajes, tales como ARJ o URQ, para facilitar a un portero controlado apaga.

Los gatekeeperes alternativos aprenden sobre las llamadas existentes con un intercambio de la respuesta de la petición del Interrumpir pedido (IRQ) /Information (IRR) entre los gatewayes y los porteros y no pierden de vista estas llamadas.

Un punto final que detecta el error de su portero puede recuperarse con seguridad de ese error utilizando a un gatekeeper alternativo para las peticiones futuras, incluyendo las peticiones las llamadas existentes. Los gatekeepers alternativos tienen que configurarse en un agrupamiento. Comparten la información sobre los puntos finales y las llamadas activas usando el GUP que se ejecuta en el TCP.

Protocolo Gatekeeper Update

Aquí están algunos pasos principales y advertencias del GUP. Esto también debería ayudarlo a solucionar el problema.

- Una vez que un gatekeeper que está configurado como parte de un agrupamiento está en línea, abre un puerto TCP para escuchar las conexiones entrantes para el protocolo GUP.
- Entonces anuncia su presencia enviando un mensaje GRQ en una forma periódica. El período predeterminado es 30 segundos y es configurable usando Este mensaje GRQ contiene datos no estándar para cada gatekeeper alternativo. Estos datos no estándar son un indicador a los suplentes que el GRQ no es realmente un GRQ en absoluto, pero son bastante apenas un mensaje "de anuncio". Dentro del mensaje GRQ, el portero indica el número del puerto que tiene abierto para estar atento el protocolo GUP.
- Cuando recibe un GRQ desde un nuevo control de acceso, los demás controles de acceso en el agrupamiento abren los canales TCP para ese puerto.
- Los mensajes GUP GRQ pueden ser uno de los siguientes mensajes: announcementIndication (Indicación de anuncio), announcementReject (Rechazo de anuncio), registrationIndication (Indicación de registro), unregistrationIndication (Indicación de eliminación de registro), y resourceIndication (Indicación de recurso).
- La indicación de anuncio también transporta información sobre el uso de ancho de banda para la zona. Esto permite que los gatekeepers administren adecuadamente el ancho de banda para una única zona, incluso aunque los gatekeepers se encuentren en dispositivos físicos separados.
- Para verificar si los gatekeeperes alternativos estén comunicando correctamente o no, utilice el comando show gatekeeper zone cluster. Este comando también brinda información acerca del ancho de banda para los gatekeepers alternativos.
- El portero asume que el gatekeeper alternativo ha fallado (y asume que cualquier ancho de banda previamente afectado un aparato está disponible ahora), si el portero no recibe un mensaje de anuncio en el plazo de seis períodos de anuncio, o si la conexión TCP con el portero se detecta para ser rota. Con seis períodos de anuncios cada 30 segundos, el tiempo es de tres minutos, lo que equipara lo que debería ser la duración promedio de una llamada. Luego, debería ser bastante seguro suponer que el ancho de banda se ha liberado. Después de tres minutos, este portero declara su suplente como abajo y envía una actualización para notificar todos sus puntos finales registradoes que no hay gatekeeper alternativo.

- Cuando un punto final se registra o cancela el registro con un control de acceso en un agrupamiento, ese control de acceso utiliza el mensaje registrationIndication/ unregistrationIndication para actualizar a todos los otros controles de acceso en ese agrupamiento sobre este cambio.
- Si un punto final informó un cambio de recurso con el indicador de disponibilidad de recursos (RAI) a un gatekeeper en un agrupamiento, ese gatekeeper informa sobre el cambio a todos los gatekeepers alternativos de dicho agrupamiento con el mensaje GUP resourceIndication.
- Los mensajes GUP son necesarios para que el gatekeeper en un agrupamiento tenga el conocimiento suficiente de cada uno de los puntos finales de la zona (registro, ancho de banda, llamadas activas, recursos) a fin de poder resolver las consultas en forma local.
- Cuando un punto final se conmuta a partir de un portero a un suplente, las necesidades alternas de aprender sobre las llamadas que son activas en el punto final. Cuando un gatekeeper envía una RCF para un nuevo registro, también envía una IRQ para obtener una lista de todas las llamadas que se encuentran en el punto final. Es importante asegurarse de que IRQ no llegue al punto final antes que la RCF.
- Los gatekeepers en un agrupamiento permiten que se apague el sistema aunque haya llamadas activas, siempre y cuando haya un gatekeeper alternativo definido para todas las zonas en las cuales hay llamadas activas. Si cualquier zona tiene una llamada activa y ningún gatekeeper alternativo definida, el portero rechaza el apagar.
- Los gatekeeperes alternativos validan cualquier pedido de desconexión (DRQ) para las llamadas que no eran conscientes de y pasan la información apropiada a los servidores del Authentication, Authorization, and Accounting (AAA) y del protocolo cisco gatekeeper transaction message (GKTMP). Esto sucede cuando ese punto final se mueve al gatekeeper alternativo mientras que hay llamadas activas. Además, podrán enviarse mensajes IRR que contengan información de llamada para llamadas que antes no eran conocidas. Para aquellos IRR, se crean los registros de llamadas y el ancho de banda se asigna consecuentemente.
- El control de acceso crea un mensaje de indicación de anuncio único para cada control de acceso alternativo. Si un gatekeeper alternativo recibe un mensaje que contiene un identificador de gatekeeper que no reconoce (lo cual puede suceder si el gatekeeper alternativo es un alternativo para una zona), pero no otro, esa información se ignora. Sin embargo, el gatekeeper alternativo detecta los errores en la configuración de los suplentes examinando esos mensajes y señala esos errores al usuario.
- El poder verdadero del GUP se observa cuando los direccionamientos se resuelven para una zona remota. En vez de la necesidad de la zona remota de enviar los LRQ (en orden o ráfaga) a todos los porteros, así aumentando la tara de mensajería en los links de área ancha, ahora necesita enviar esta interrogación a apenas una de los porteros en el cluster. Juntado con el nuevo telecontrol CLI del cluster de la zona, puede circular entre los porteros en el cluster y no intentar enviar el LRQ a otro portero en el cluster si recibe un rechazo.
- En el caso de que se haya movido una gateway a un gatekeeper alternativo, siempre intenta registrarse en ese gatekeeper a menos que ejecute un comando no gateway y luego un comando gateway. Cuando el gatekeeper primario del punto final está de nuevo en línea, el punto final no vuelve a registrarse en él a menos que el punto final pierda comunicación con el gatekeeper alternativo. Continúa utilizando el gatekeeper alternativo para su información de ruteo de llamada.

Comandos debug y show para agrupamientos de gatekeeper

Los debugs abajo demuestran cómo los porteros pueden unirse a un cluster y cómo comparten la información sobre sus puntos finales. los comandos show se utilizan para mostrar cómo supervisar el agrupamiento. Los debugs usados son asn1 del gatekeeper gup del debug y hacen el debug del asn1 del h225. Esto está basado en la topología y la configuración mencionada anteriormente.

Depuraciones a partir de "gkb-1" cuando fue el primero en unirse al grupo

```
Mar  1 08:15:08.348: gk_gup_listen(): listening port = 11007 !--- Opens a TCP port (here it is
11007) to listen to GUP messages. Mar 1 08:15:08.348: gk_gup_listen(): listening fd = 0 Mar 1
08:15:38.351: H225 NONSTD OUTGOING PDU ::= value GRQnonStandardInfo ::= !--- The non-standard
data that is in the GRQ. { gupAddress { ip 'AC100D29'H !--- Listening IP address 172.16.13.41.
port 11007 !--- Listening TCP port 11007. } } Mar 1 08:15:38.351: H225 NONSTD OUTGOING ENCODE
BUFFER::= 40 AC100D29 2AFF Mar 1 08:15:38.351: Mar 1 08:15:38.351: RAS OUTGOING PDU ::= value
RasMessage ::= gatekeeperRequest : !--- GRQ with the non-standard is sent out. { requestSeqNum
59 protocolIdentifier { 0 0 8 2250 0 3 } nonStandardData { nonStandardIdentifier h221NonStandard
: { t35CountryCode 181 t35Extension 0 manufacturerCode 18 } data '40AC100D292AFF'H } rasAddress
ipAddress : { ip 'AC100D29'H port 1719 } endpointType { vendor { vendor { t35CountryCode 181
t35Extension 0 manufacturerCode 18 } } mc FALSE undefinedNode FALSE } } Mar 1 08:15:38.359: RAS
OUTGOING ENCODE BUFFER::= 01 00003A06 0008914A 000340B5 00001207 40AC100D 292AFF00 AC100D29
06B72000 B5000012 00 Mar 1 08:15:38.359:
```

Depuraciones de "gkb-2" cuando se unió al agrupamiento después de "gkb-1

```
Mar  1 08:16:38.878: gk_gup_listen(): listening port = 11006 !--- Opens a TCP port (here it is
11006) to listen to GUP messages. Mar 1 08:16:38.878: gk_gup_listen(): listening fd = 0 Mar 1
08:17:08.385: RAS INCOMING ENCODE BUFFER::= 01 00003D06 0008914A 000340B5 00001207 40AC100D
292AFF00 AC100D29 06B72000 B5000012 00 Mar 1 08:17:08.385: Mar 1 08:17:08.385: RAS INCOMING PDU
::= value RasMessage ::= gatekeeperRequest : !--- GRQ message is received from gkb-1 gatekeeper
with non-standard information. { requestSeqNum 62 protocolIdentifier { 0 0 8 2250 0 3 }
nonStandardData { nonStandardIdentifier h221NonStandard : { t35CountryCode 181 t35Extension 0
manufacturerCode 18 } data '40AC100D292AFF'H } rasAddress ipAddress : { ip 'AC100D29'H !--- RAS
IP address 172.16.13.41 used gkb-1. port 1719 !--- RAS TCP port used by gkb-1. } endpointType {
vendor { vendor { t35CountryCode 181 t35Extension 0 manufacturerCode 18 } } mc FALSE
undefinedNode FALSE } } Mar 1 08:17:08.393: H225 NONSTD INCOMING ENCODE BUFFER::= 40 AC100D29
2AFF Mar 1 08:17:08.393: Mar 1 08:17:08.393: H225 NONSTD INCOMING PDU ::= value
GRQnonStandardInfo ::= !--- gkb-2 extracts the non-standard data from the GRQ. { gupAddress { ip
'AC100D29'H !--- GUP IP address 172.16.13.41 used by gkb-1. port 11007 !--- GUP TCP port 11007
used by gkb-1. } } Mar 1 08:17:08.393: check_connection: checking connection to
172.16.13.41:11007 Mar 1 08:17:08.393: gk_gup_connect(): initiating connection Mar 1
08:17:08.393: gup_connect: connecting to 172.16.13.41:11007 !--- A GUP connection is
established, and updates follow. Mar 1 08:17:08.393: gup_connect, fd = 1 Mar 1 08:17:08.401: GUP
OUTGOING PDU ::= value GUP_Information ::= !--- GUP announcement is sent to alternate GK gkb-1.
{ protocolIdentifier { 1 2 840 113548 10 0 0 2 } message announcementIndication : {
announcementInterval 30 endpointCapacity 100000 callCapacity 100000 hostName '676B622D32'H
percentMemory 8 !--- Below is information about the status of gkb-2. percentCPU 0 currentCalls 0
currentEndpoints 0 zoneInformation { { gatekeeperIdentifier {"gkb-2"} altGKIdentifier {"gkb-1"}
totalBandwidth 0 interzoneBandwidth 0 remoteBandwidth 0 } } } } Mar 1 08:17:08.405: GUP OUTGOING
ENCODE BUFFER::= 00 0A2A8648 86F70C0A 00000220 001E8001 86A08001 86A00467 6B622D32 10000000
00014200 0067006B 0062002D 00320800 67006B00 62002D00 31000000 000000 Mar 1 08:17:08.409: Mar 1
08:17:08.409: Sending GUP ANNOUNCEMENT INDICATION to 172.16.13.41 Mar 1 08:17:08.413: GUP
INCOMING ENCODE BUFFER::= 00 0A2A8648 86F70C0A 00000220 001E8001 86A08001 86A00467 6B622D31
32000000 00014200 0067006B 0062002D 00310800 67006B00 62002D00 32000000 000000 Mar 1
08:17:08.413: Mar 1 08:17:08.413: GUP INCOMING PDU ::= value GUP_Information ::= !--- GUP
announcement is received from alternate GK gkb-1. { protocolIdentifier { 1 2 840 113548 10 0 0 2
} message announcementIndication : { announcementInterval 30 endpointCapacity 100000
callCapacity 100000 hostName '676B622D31'H percentMemory 25 !--- Below is information about the
status of gkb-1. percentCPU 0 currentCalls 0 currentEndpoints 0 zoneInformation { {
gatekeeperIdentifier {"gkb-1"} altGKIdentifier {"gkb-2"} totalBandwidth 0 interzoneBandwidth 0
remoteBandwidth 0 } } } } Mar 1 08:17:08.421: Received GUP ANNOUNCEMENT INDICATION from
172.16.13.41
```

Con el comando show gatekeeper endpoint, no hay puntos finales registradoes. El resultado es el siguiente:

```
gkb-1#show gatekeeper endpoints GATEKEEPER ENDPOINT REGISTRATION
================================ CallSignalAddr Port RASSignalAddr Port Zone Name Type Flags ---
------------ ----- -------------- ----- --------- ---- ----- Total number of active
registrations = 0 gkb-2# show gatekeeper endpoints GATEKEEPER ENDPOINT REGISTRATION
================================ CallSignalAddr Port RASSignalAddr Port Zone Name Type Flags ---
------------ ----- -------------- ----- --------- ---- ----- Total number of active
registrations = 0 gkb-2#
```

Depura cuando un punto final se registra en uno de los Gatekeepers del agrupamiento

Esta depuración es tomada del gatekeeper "gkb-1". El punto final es el registro "gwb-1" en la puerta de enlace "gkb-1" con debug h225 ans1 y debug ras activados.

```
Mar  1 08:22:47.396: RAS INCOMING ENCODE BUFFER::= 00 A00AAD06
0008914A 000300AC 100D17E0 7D088001 3C050401 00205002 00006700 6B006200
2D003101 40040067 00770062 002D0031
Mar  1 08:22:47.396:
Mar  1 08:22:47.396: RAS INCOMING PDU ::= value RasMessage ::= gatekeeperRequest : !--- GRQ is
received from "gwb-1" gateway. { requestSeqNum 2734 protocolIdentifier { 0 0 8 2250 0 3 }
rasAddress ipAddress : { ip 'AC100D17'H !--- gwb-1 IP address (172.16.13.23). port 57469 !---
gwb-1 TCP port 57469. } endpointType { gateway { protocol { voice : { supportedPrefixes { {
prefix e164 : "2#" } } } } } } mc FALSE undefinedNode FALSE } gatekeeperIdentifier {"gkb-1"}
endpointAlias { h323-ID : {"gwb-1"} } } Mar 1 08:22:47.404: RAS OUTGOING PDU ::= value
RasMessage ::= gatekeeperConfirm : !--- GCF is sent back with alternate gatekeepers included. {
requestSeqNum 2734 protocolIdentifier { 0 0 8 2250 0 3 } gatekeeperIdentifier {"gkb-1"}
rasAddress ipAddress : { ip 'AC100D29'H !--- Gatekeeper gkb-1 IP address (172.16.13.41). port
1719 } alternateGatekeeper !--- List of alternate gatekeepers, here is "gkb-2" only. { {
rasAddress ipAddress : { ip 'AC100D10'H !--- Alternate gatekeeper gkb-2 IP address
(172.16.13.16) port 1719 } gatekeeperIdentifier {"gkb-2"} needToRegister TRUE priority 0 } } }
Mar 1 08:22:47.412: RAS OUTGOING ENCODE BUFFER::= 06 800AAD06 0008914A 00030800 67006B00
62002D00 3100AC10 0D2906B7 0D001401 40AC100D 1006B708 0067006B 0062002D 003280 Mar 1
08:22:47.412: Mar 1 08:22:47.432: RAS INCOMING ENCODE BUFFER::= 0E C00AAE06 0008914A 00038001
00AC100D 1706B801 00AC100D 17E07D08 80013C05 04010020 50000140 04006700 77006200 2D003108
0067006B 0062002D 003100B5 00001212 8B000200 3B010001 000180 Mar 1 08:22:47.432: Mar 1
08:22:47.436: RAS INCOMING PDU ::= value RasMessage ::= registrationRequest : !--- RRQ is
received from "gwb-1" gateway. { requestSeqNum 2735 protocolIdentifier { 0 0 8 2250 0 3 }
discoveryComplete TRUE callSignalAddress { ipAddress : { ip 'AC100D17'H !--- Gateway gwb-1 IP
address (172.16.13.23). port 1720 } } rasAddress { ipAddress : { ip 'AC100D17'H port 57469 } }
terminalType { gateway { protocol { voice : { supportedPrefixes { { prefix e164 : "2#" } } } } } }
mc FALSE undefinedNode FALSE } terminalAlias { h323-ID : {"gwb-1"} } gatekeeperIdentifier {"gkb-
1"} endpointVendor { vendor { t35CountryCode 181 t35Extension 0 manufacturerCode 18 } }
timeToLive 60 keepAlive FALSE willSupplyUUIEs FALSE maintainConnection TRUE } Mar 1
08:22:47.448: GUP OUTGOING PDU ::= value GUP_Information ::= !--- A GUP registration indicates a
message is sent to "gkb-2" to inform it !--- about the new registered endpoint. {
protocolIdentifier { 1 2 840 113548 10 0 0 2 } message registrationIndication : { version 3
callSignalAddress { ipAddress : { ip 'AC100D17'H !--- Gateway gwb-1 IP address (172.16.13.23).
port 1720 } } rasAddress { ipAddress : { ip 'AC100D17'H !--- Gateway gwb-1 IP address
(172.16.13.23). port 57469 } } terminalType { vendor { vendor { t35CountryCode 181 t35Extension
0 manufacturerCode 18 } } gateway { protocol { voice : { supportedPrefixes { { prefix e164 :
"2#*" } } } } } } mc FALSE undefinedNode FALSE } terminalAlias { h323-ID : {"gwb-1"} !--- Name/ID
of the new endpoint which has just registered. } gatekeeperIdentifier {"gkb-1"} !--- Name/ID of
the gatekeeper which the new endpoint(gwb-1) has registered to. resourceIndicator {
almostOutOfResources FALSE } } } Mar 1 08:22:47.460: GUP OUTGOING ENCODE BUFFER::= 00 0A2A8648
86F70C0A 00000232 020100AC 100D1706 B80100AC 100D17E0 7D2800B5 00001240 013C0505 01004050
10000140 04006700 77006200 2D003108 0067006B 0062002D 003100 Mar 1 08:22:47.464: Mar 1
08:22:47.464: Sending GUP REGISTRATION INDICATION to 172.16.13.16 Mar 1 08:22:47.464: RAS
OUTGOING PDU ::= value RasMessage ::= registrationConfirm : !--- RCF is sent back to "gwb-1"
gateway. { requestSeqNum 2735 protocolIdentifier { 0 0 8 2250 0 3 } callSignalAddress { }
```

```
terminalAlias { h323-ID : {"gwb-1"} } gatekeeperIdentifier {"gkb-1"} endpointIdentifier
{"61809DB800000001"} alternateGatekeeper { { rasAddress ipAddress : { ip 'AC100D10'H port 1719 }
gatekeeperIdentifier {"gkb-2"} needToRegister TRUE priority 0 } } timeToLive 60 willRespondToIRR
FALSE maintainConnection TRUE } Mar 1 08:22:47.472: RAS OUTGOING ENCODE BUFFER::= 12 C00AAE06
0008914A 00030001 40040067 00770062 002D0031 08006700 6B006200 2D00311E 00360031 00380030
00390044 00420038 00300030 00300030 00300030 00300031 0F8A1401 40AC100D 1006B708 0067006B
0062002D 00328002 003B0100 0180 Mar 1 08:22:47.472:
```

La salida antedicha contiene después de todo los gatewayes hechos salir [comando show gatekeeper endpoint](#) se registra en el cluster. Los debugs antedichos suceden para cada registro del punto final. Después de todo tres gatewayes en el cluster se registran, el **comando show gatekeeper endpoint** en ambos porteros son como sigue:

```
gkb-1#show gatekeeper endpoints GATEKEEPER ENDPOINT REGISTRATION
================================ CallSignalAddr Port RASSignalAddr Port Zone Name Type Flags ---
------------ ----- -------------- ----- --------- ---- ----- 172.16.13.23 1720 172.16.13.23
57469 gkb-1 VOIP-GW H323-ID: gwb-1 172.16.13.26 1720 172.16.13.26 49801 gkb-1 VOIP-GW H323-ID:
gwb-2 172.16.13.42 1720 172.16.13.42 57216 gkb-1 VOIP-GW A !--- A flag set. H323-ID: gwb-3 Total
number of active registrations = 3 gkb-2# show gatekeeper endpoints GATEKEEPER ENDPOINT
REGISTRATION ================================ CallSignalAddr Port RASSignalAddr Port Zone Name
Type Flags -------------- ----- -------------- ----- --------- ---- ----- 172.16.13.23 1720
172.16.13.23 57469 gkb-2 VOIP-GW A !--- A flag set. H323-ID: gwb-1 172.16.13.26 1720
172.16.13.26 49801 gkb-2 VOIP-GW A !--- A flag set. H323-ID: gwb-2 172.16.13.42 1720
172.16.13.42 57216 gkb-2 VOIP-GW H323-ID: gwb-3 Total number of active registrations = 3 !---
The "A" under the flag field means that the gatekeeper is an alternate one !--- for this
endpoint.
```

Depuraciones para los casos en que un punto final con llamada activa se mueve al control de acceso alternativo

Éstos son los debugs de un portero que comience cuando se pide la llamada y vaya hasta que sea disconnected. Algunos de los mensajes innecesarios del debug se han omitido. Estos debugs son del portero del "gkb-1". La llamada fue puesta por gwa-1 registrado hasta el "gka-1" a otro gateway (gwb-1) en el clúster de zona remota. Los debugs demuestran cómo un flujo de la llamada activa se sigue del gatekeeper primario al gatekeeper alternativo mientras que va el primario abajo.

```
Mar  2 23:59:26.714:  RecvUDP_IPSockData  successfully rcvd message of length 84
from 172.16.13.35:1719
Mar  2 23:59:26.714: RAS INCOMING ENCODE BUFFER::= 4A 80080801 01806986
40B50000 122C8286 B01100C8 C66C7D1
6 8011CC80 0D882828 5B8DF601 80140204 8073B85A 5C564004 00670077
0061002D 003100AC 100D2306 B70B800D 01400
400 67006B00 61002D00 310180
Mar  2 23:59:26.714:
Mar  2 23:59:26.714: RAS INCOMING PDU ::=

value RasMessage ::= locationRequest : !--- LRQ is received from "gka-1" gatekeeper from domain
A. { requestSeqNum 2057 destinationInfo { e164 : "3653" !--- E164 number to be resolved by the
this gatekeeper. } nonStandardData { nonStandardIdentifier h221NonStandard : { t35CountryCode
181 t35Extension 0 manufacturerCode 18 } data '8286B01100C8C66C7D168011CC800D8828285B8D...'H }
replyAddress ipAddress : { ip 'AC100D23'H port 1719 } sourceInfo { h323-ID : {"gka-1"} }
canMapAlias TRUE } Mar 2 23:59:26.722: LRQ (seq# 2057) rcvd Mar 2 23:59:26.722: H225 NONSTD
INCOMING ENCODE BUFFER::= 82 86B01100 C8C66C7D 168011CC 800D8828 285B8DF6 01801402 048073B8
5A5C5640 04006700 77006100 2D0031 Mar 2 23:59:26.722: Mar 2 23:59:26.722: H225 NONSTD INCOMING
PDU ::= !--- LRQ nonStandardInfo decoded output. value LRQnonStandardInfo ::= { ttl 6 nonstd-
callIdentifier { guid 'C8C66C7D168011CC800D8828285B8DF6'H } callingOctet3a 128 gatewaySrcInfo {
e164 : "4085272923", h323-ID : {"gwa-1"} } } parse_lrq_nonstd: LRQ Nonstd decode succeeded,
remlen = 84 Mar 2 23:59:26.726: H225 NONSTD OUTGOING PDU ::= !--- LCF nonStandardInfo reply back
to the LRQ nonStandardInfor. value LCFnonStandardInfo ::= { termAlias { h323-ID : {"gwb-1"} }
gkID {"gkb-1"} gateways { { gwType voip : NULL gwAlias { h323-ID : {"gwb-1"} !--- Gateway gwb-1
is the resolved terminating gateway sent back for the request. } sigAddress { ip 'AC100D17'H !--
```

- Gateway gwb-1 IP address (172.16.13.23). port 1720 } resources { maxDSPs 0 inUseDSPs 0 maxBChannels 0 inUseBChannels 0 activeCalls 0 bandwidth 0 inuseBandwidth 0 } } } } Mar 2 23:59:26.734: H225 NONSTD OUTGOING ENCODE BUFFER::= 00 01400400 67007700 62002D00 31080067 006B0062 002D0031 01100140 04006700 77006200 2D003100 AC100D17 06B80000 00000000 00000000 Mar 2 23:59:26.734: Mar 2 23:59:26.734: RAS OUTGOING PDU ::= value RasMessage ::= **locationConfirm** : *!-- LCF is sent back with "gwb-1" as the resolved terminating gateway.* { requestSeqNum 2057 callSignalAddress ipAddress : { ip 'AC100D17'H *!--- Resolved terminating gateway gwb-1 IP address (172.16.13.23).* port 1720 } rasAddress ipAddress : { ip 'AC100D17'H *!--- Resolved terminating gateway gwb-1 IP address (172.16.13.23).* port 51874 } nonStandardData { nonStandardIdentifier h221NonStandard : { t35CountryCode 181 t35Extension 0 manufacturerCode 18 } data '0001400400670077006200 2D0031080067006B00...'H } destinationType { gateway { protocol { voice : { supportedPrefixes { } } } } } mc FALSE undefinedNode FALSE } } Mar 2 23:59:26.742: RAS OUTGOING ENCODE BUFFER::= 4F 080800AC 100D1706 B800AC10 0D17CAA2 40B50000 1239000 1 40040067 00770062 002D0031 08006700 6B006200 2D003101 10014004 00670077 0062002D 003100AC 100D1706 B8000 000 00000000 00000010 40080880 013C0501 0000 Mar 2 23:59:26.746: Mar 2 23:59:26.746: IPSOCK_RAS_sendto: msg length 91 from 172.16.13.41:1719 to 172.16.13.35: 1719 Mar 2 23:59:26.746: **RASLib::RASSendLCF: LCF (seq# 2057) sent to 172.16.13.35** Mar 2 23:59:26.798: RecvUDP_IPSockData successfully rcvd message of length 129 from 172.16.13.23:51874 Mar 2 23:59:26.798: RAS INCOMING ENCODE BUFFER::= 27 98172700 F0003600 31003900 36003200 39003600 3800300 0 30003000 30003000 30003000 31010180 69860204 8073B85A 5C564004 00670077 0061002D 003100AC 100D0F2A FA400 500 000E40B5 00001207 80000008 800180C8 C66C7D16 8011CC80 0C882828 5B8DF645 60200180 1100C8C6 6C7D1680 11C C800D 8828285B 8DF60100 Mar 2 23:59:26.802: Mar 2 23:59:26.802: RAS INCOMING PDU ::= value RasMessage ::= **admissionRequest** : *!--- "gwb-1" sent answerCall ARQ.* { **requestSeqNum 5928** callType pointToPoint : NULL **callModel direct : NULL** endpointIdentifier {"6196296800000001"} **destinationInfo** { **e164 : "3653"** *!--- E164 number the caller is trying to reach.* } **srcInfo** { **e164 : "4085272923",** *!--- Caller information.* **h323-ID : {"gwa-1"}** } **srcCallSignalAddress ipAddress :** { **ip 'AC100D0F'H** *!--- Originating gateway (gwa-1) IP address and port.* **port 11002** } bandWidth 1280 callReferenceValue 14 *!--- Remember call reference, since it is used when the call !--- is disconnected when sending the DRQ.* nonStandardData { nonStandardIdentifier h221NonStandard : { t35CountryCode 181 t35Extension 0 manufacturerCode 18 } data '80000008800180'H } conferenceID 'C8C66C7D168011CC800C8828285B8DF6'H activeMC FALSE **answerCall TRUE** canMapAlias TRUE callIdentifier { guid 'C8C66C7D168011CC800D8828285B8DF6'H } willSupplyUUIEs FALSE } Mar 2 23:59:26.810: **ARQ (seq# 5928) rcvd** Mar 2 23:59:26.810: H225 NONSTD INCOMING ENCODE BUFFER::= 80 00000880 0180 Mar 2 23:59:26.810: Mar 2 23:59:26.810: H225 NONSTD INCOMING PDU ::= value **ARQnonStandardInfo** ::= { sourceAlias { } sourceExtAlias { } callingOctet3a 128 } parse_arq_nonstd: ARQ Nonstd decode succeeded, remlen = 129 Mar 2 23:59:26.814: RAS OUTGOING PDU ::= value RasMessage ::= **admissionConfirm** : *!--- ACF is sent back to "gwb-1".* { **requestSeqNum 5928 bandWidth 1280 callModel direct : NULL destCallSignalAddress ipAddress :** { **ip 'AC100D17'H** *!--- gwb-1 IP address (172.16.13.23).* **port 1720** } **irrFrequency 240** willRespondToIRR FALSE uuiesRequested { setup FALSE callProceeding FALSE connect FALSE alerting FALSE information FALSE releaseComplete FALSE facility FALSE progress FALSE empty FALSE } } Mar 2 23:59:26.818: RAS OUTGOING ENCODE BUFFER::= 2B 00172740 050000AC 100D1706 B800EF1A 00C00100 020000 Mar 2 23:59:26.818: Mar 2 23:59:26.818: IPSOCK_RAS_sendto: msg length 24 from 172.16.13.41:1719 to 172.16.13.23: 51874 Mar 2 23:59:26.822: RASLib::RASSendACF: **ACF (seq# 5928) sent to 172.16.13.23** Mar 2 23:59:36.046: **GUP OUTGOING PDU** ::= value GUP_Information ::= *!--- GUP update is sent out and it contains the inforomation !--- about the last call that is still active.* { protocolIdentifier { 1 2 840 113548 10 0 0 2 } message **announcementIndication** : { announcementInterval 30 endpointCapacity 46142 callCapacity 68793 hostName '676B622D31'H percentMemory 25 percentCPU 0 **currentCalls 1** currentEndpoints 2 zoneInformation { { gatekeeperIdentifier {"gkb-1"} altGKIdentifier {"gkb-2"} **totalBandwidth 1280** *!--- 1280 is 128 Kbps of total bandwidth used for the zone.* **interzoneBandwidth 1280 remoteBandwidth 1280** } } } } Mar 2 23:59:36.050: GUP OUTGOING ENCODE BUFFER::= 00 0A2A8648 86F70C0A 00000220 001E40B4 3E80010C B904676 B 622D3132 00010002 01420000 67006B00 62002D00 31080067 006B0062 002D0032 40050040 05004005 00 Mar 2 23:59:36.054: Mar 2 23:59:36.054: **Sending GUP ANNOUNCEMENT INDICATION to 172.16.13.16**

Nota: En este momento el "gkb-1" está apagan. Se permite esto (incluso si tiene una llamada activa) porque hay gatekeeper alternativo para esa zona.

Los mensajes URQ se envían a todos los puntos finales registradoes con el "gkb-1". Estos puntos finales son gatewayes del "gwb-1" y del "gwb-2". Estos gatewayes confirman el URQ devolviendo los UCF. También, gkb-1 envía a mensaje que indica la ausencia de registro GUP al gatekeeper alternativo del cluster y entonces cierra la conexión GUP.

```
Mar  2 23:59:55.914: RAS OUTGOING PDU ::= value RasMessage ::= unregistrationRequest : {
requestSeqNum 79 callSignalAddress { ipAddress : { ip 'AC100D17'H !--- UnregistrationRequest
(URQ) sent to gwb-1 (172.16.13.23). port 1720 } } } Mar 2 23:59:55.914: RAS OUTGOING ENCODE
BUFFER::= 18 00004E01 00AC100D 1706B8 Mar 2 23:59:55.914: Mar 2 23:59:55.914: IPSOCK_RAS_sendto:
msg length 12 from 172.16.13.41:1719 to 172.16.13.23: 51874 Mar 2 23:59:55.914:
RASLib::RASSendURQ: URQ (seq# 79) sent to 172.16.13.23 Mar 2 23:59:55.918: RAS OUTGOING PDU ::=
value RasMessage ::= unregistrationRequest : { requestSeqNum 80 callSignalAddress { ipAddress :
{ ip 'AC100D1A'H !--- URQ sent to gwb-2 (172.16.13.26). port 1720 } } } Mar 2 23:59:55.918: RAS
OUTGOING ENCODE BUFFER::= 18 00004F01 00AC100D 1A06B8 Mar 2 23:59:55.918: Mar 2 23:59:55.918:
IPSOCK_RAS_sendto: msg length 12 from 172.16.13.41:1719 to 172.16.13.26: 50041 Mar 2
23:59:55.918: RASLib::RASSendURQ: URQ (seq# 80) sent to 172.16.13.26 Mar 2 23:59:55.922:
RecvUDP_IPSockData successfully rcvd message of length 3 from 172.16.13.23:51874 Mar 2
23:59:55.922: RAS INCOMING ENCODE BUFFER::= 1C 004E Mar 2 23:59:55.922: Mar 2 23:59:55.922: RAS
INCOMING PDU ::= value RasMessage ::= unregistrationConfirm : { requestSeqNum 79 } Mar 2
23:59:55.922: UCF (seq# 79) rcvd Mar 2 23:59:55.926: RecvUDP_IPSockData successfully rcvd
message of length 3 from 172.16.13.26:50041 Mar 2 23:59:55.926: RAS INCOMING ENCODE BUFFER::= 1C
004F Mar 2 23:59:55.926: Mar 2 23:59:55.926: RAS INCOMING PDU ::= value RasMessage ::=
unregistrationConfirm : { requestSeqNum 80 } Mar 2 23:59:55.926: UCF (seq# 80) rcvd Mar 3
00:00:01.922: GUP OUTGOING PDU ::= value GUP_Information ::= { protocolIdentifier { 1 2 840
113548 10 0 0 2 } message unregistrationIndication : { reason explicitUnregister : NULL
callSignalAddress { ipAddress : { ip 'AC100D17'H !--- GUP UnregistrationIndication sent to
alternate gatekeeper !--- gkb-2 (172.16.13.16) in the cluster. port 1720 } } } } Mar 3
00:00:01.922: GUP OUTGOING ENCODE BUFFER::= 00 0A2A8648 86F70C0A 00000238 000100AC 100D1706 B8
Mar 3 00:00:01.926: Mar 3 00:00:01.926: Sending GUP UNREGISTRATION INDICATION to 172.16.13.16
Mar 3 00:00:01.934: gk_gup_close_connection(): closing connection to 172.16.13.16 Mar 3
00:00:01.934: gk_gup_close_listen(): closing listen
```

Aquí está el debug a partir de la "gkb-2". Los debugs que muestran que el registro del punto final movido el "gwb-1" y el "gwb-2" está omitido, puesto que parecen el registro normal. El propósito aquí es mostrar la aceptación del DRQ de la llamada activa en el "gwb-1" cuando se mueve hasta el "gkb-2".

```
Mar  3 00:00:24.307:  RecvUDP_IPSockData  successfully rcvd message of length 77
from 172.16.13.23:51874
Mar  3 00:00:24.307: RAS INCOMING ENCODE BUFFER::= 3E 172C1E00 36003100
38003400 44004300 34004300 30003000 30003000 3
0003000 300033C8 C66C7D16 8011CC80 0C882828 5B8DF600 0E21A100 1100C8C6
6C7D1680 11CC800D 8828285B 8DF60180
Mar  3 00:00:24.311:
Mar  3 00:00:24.311: RAS INCOMING PDU ::=

value RasMessage ::= disengageRequest : !--- DRQ is received with call reference 14 and normal
clearing !--- disconnect cause code. !--- This information is passed to the accounting server
and the GKTMP !--- server if configured. { requestSeqNum 5933 endpointIdentifier
{"6184DC4C00000003"} conferenceID 'C8C66C7D168011CC800C8828285B8DF6'H callReferenceValue 14
disengageReason normalDrop : NULL callIdentifier { guid 'C8C66C7D168011CC800D8828285B8DF6'H }
answeredCall TRUE } Mar 3 00:00:24.311: DRQ (seq# 5933) rcvd Mar 3 00:00:24.315: RAS OUTGOING
PDU ::= value RasMessage ::= disengageConfirm : !--- DCF is sent to "gwb-1". { requestSeqNum
5933 } Mar 3 00:00:24.315: RAS OUTGOING ENCODE BUFFER::= 40 172C Mar 3 00:00:24.315: Mar 3
00:00:24.315: IPSOCK_RAS_sendto: msg length 3 from 172.16.13.16:1719 to 172.16.13.23: 51874 Mar
3 00:00:24.315: RASLib::RASSendDCF: DCF (seq# 5933) sent to 172.16.13.23 gkb-2#
```

Gateway Fail Over de Cisco a Gatekeeper alternativo

Por abandono, los gatewayes de Cisco envían un RRQ ligero cada 45 segundos. En caso de que el portero no enviara ningún URQ al gateway (debido a un problema de ruteo quebrado, por ejemplo), los intentos del gateway (sobre la audición de un RCF o de un RRJ para su RRQ ligero) dos veces con cinco segundos entre cada uno. Si la tercera tentativa falla, considera inmediatamente al portero como muertos y se registra con el gatekeeper alternativo que utiliza el RRQ. En un escenario donde el gateway comienza el proceso de inscripción inicial con el portero, envía el GRQ para localizar la dirección IP del portero. Si hay una contestación GCF detrás, el

gateway envía el RRQ al gatekeeper primario especificado. Si por cualquier motivo el portero rechaza el pedido de inscripción, el gateway no intenta entrar en contacto a su gatekeeper alternativo. Comienza este proceso (GRQ, GCF y RRQ) encima otra vez con el gatekeeper primario.

El gateway entra en contacto solamente al gatekeeper alternativo cuando la Conectividad al gatekeeper primario se pierde y no hay contestación detrás. Si el gatekeeper primario no contesta de nuevo al mensaje GRQ cuando el gateway primero envía para descubrir al portero, después después de que tres intentos fallidos (aproximadamente cinco minutos por la tentativa), el gateway entren en contacto al gatekeeper alternativo. En una situación adonde va el gatekeeper primario abajo después de que el gateway se haya registrado con él, el gateway pierde los mensajes del Keepalives del gatekeeper primario. Después de faltar tres mensajes de keepalive consecutivos, el gateway declara al gatekeeper primario como abajo, y comienza el proceso de inscripción otra vez.

Solución de problemas con puntos finales alternativos

Un punto final de llamada puede recuperarse de una falla de configuración de la llamada al enviar un mensaje de configuración a uno de los puntos finales alternativos. La llamada puede fallar por varias razones: el gateway está inactiva y el control de acceso no está al tanto de esto en el momento de enviar la ACF o LCF, no hay recursos en el gateway y no se informó esto al control de acceso, la llamada falla debido a una configuración incorrecta en el punto final principal, etc.

Nota: El punto final de origen intenta solamente entrar en contacto a los gatekeeperes alternativos si la llamada falla antes de la etapa alerta (alerta o progreso). Si las llamadas fallan debido al usuario ocupado o a ninguna respuesta, el punto final de origen no intenta a ninguna otra suplentes.

El portero aprende sobre ése el suplente para cierto punto final cualquier por la configuración manual usando Cisco soporta un máximo de 20 suplentes para cada punto final, no importa cómo el portero los aprende.

Los problemas que necesita considerar son:

- Si el portero tiene el punto final alternativo correcto según lo deseado.
- Si el portero incluye los puntos finales alternos en su LCF o mensajes ACF RAS.
- Si el OGW intenta entrar en contacto a los suplentes en caso de que el punto final de destino principal falle.

Para mostrar cómo resolver problemas estos problemas, la misma topología que arriba se utiliza con este cambio en la configuración de control de acceso del "gkb-1" para incluir dos gatewayes alternos: el "gwb-3" y el "gwb-1" para el gateway el "gwb-2". Aquí está la configuración del portero del "gkb-1":

```
!
gatekeeper
 zone local gkb-1 domainB.com 172.16.13.41
 zone remote gka-1 domainA.com 172.16.13.35 1719
 zone cluster local gkb gkb-1
  element gkb-2 172.16.13.16 1719
 !
 gw-type-prefix 2#* default-technology
 bandwidth total zone gkb-1 512
 bandwidth session zone gkb-1 512
```

```
 no shutdown
endpoint alt-ep h323id gwb-2 172.16.13.42 !--- 172.16.13.42 is gwb-3. endpoint alt-ep h323id
gwb-2 172.16.13.23 !--- 172.16.13.23 is gwb-1. !
```

Verifique al portero tiene los puntos finales alternativos correcto

Para determinar si el control de acceso posee los puntos finales alternativos correctos, utilice el comando show gatekeeper endpoints alternates.

```
gkb-1#show gatekeeper endpoints alternates GATEKEEPER ENDPOINT REGISTRATION
================================= CallSignalAddr Port RASSignalAddr Port Zone Name Type Flags -
------------------ ------ ------------------- ------- --------------- --------- --------
172.16.13.23 1720 172.16.13.23 54670 gkb-1 VOIP-GW H323-ID: gwb-1 172.16.13.26 1720 172.16.13.26
57233 gkb-1 VOIP-GW H323-ID: gwb-2 ALT_EP: 172.16.13.42 <1720> 172.16.13.23 <1720> !--- This
shows the information about all collected endpoints. 172.16.13.42 1720 172.16.13.42 58430 gkb-1
VOIP-GW A H323-ID: gwb-3 Total number of active registrations = 3 ALL CONFIGURED ALTERNATE
ENDPOINTS !--- Only manually configured. ===================================== Endpoint H323
Id RASSignalAddr Port --------------------- ------------------- ------- gwb-2 172.16.13.42
1720 gwb-2 172.16.13.23 1720 gkb-1#
```

Verificar si el gatekeeper incluye puntos finales alternativos en sus mensajes LCF o ACF RAS

Para ver si el portero envía la dirección IP para los puntos finales alternos, usted puede girar el **asn1 del h225 del debug** y mirar el mensaje ACF o el LCF. Éste es un ejemplo de depuración tomado de "gkb-1".

```
Mar  3 04:12:47.676: H225 NONSTD OUTGOING ENCODE BUFFER::= 00 01400400
67007700 62002D00 32080067 00
6B0062 002D0031 01100140 04006700 77006200 2D003200 AC100D1A 06B80000
00000000 00000000
Mar  3 04:12:47.676:
Mar  3 04:12:47.676: RAS OUTGOING PDU ::=

value RasMessage ::= locationConfirm : { requestSeqNum 2070 callSignalAddress ipAddress : { ip
'AC100D1A'H !--- This is IP address of main destination. port 1720 } rasAddress ipAddress : { ip
'AC100D1A'H port 50041 } nonStandardData { nonStandardIdentifier h221NonStandard : {
t35CountryCode 181 t35Extension 0 manufacturerCode 18 } data
'0001400400670077006200 2D0032080067006B00...'H } destinationType { gateway { protocol { voice :
{ supportedPrefixes { } } } } mc FALSE undefinedNode FALSE } alternateEndpoints !--- Alternate
endpoints. { { callSignalAddress { ipAddress : { ip 'AC100D2A'H !--- This is the first alternate
IP address (172.16.13.42 gwb-3). port 1720 }, ipAddress : { ip 'AC100D17'H !--- This is the
second alternate IP address (172.16.13.23 gwb-1). port 1720 } } } } }
```

Verifique si OGW intenta comunicarse con alternativos en caso de que falle el punto final de destino principal

Esta sección muestra cómo reacciona la OGW cuando recibe puntos finales alternativos en su mensaje ACF. En este ejemplo, se hace que la llamada fracase cuando intenta contactar el punto final de terminación principal (gw). Las depuraciones que se deben activar aquí son debug voip ccapi inout y debug h225 asn1.

Lo primero que se ve en la depuración es el mensaje ccapi que muestra el tramo de telefonía de origen.

```
Mar  3 04:12:47.616: cc_api_call_setup_ind (vdbPtr=0x6264A60C,
callInfo={called=3653,called_oct3=0x8
0,calling=4085272923,calling_oct3=0x21,calling_oct3a=0x80,calling_xlated=false,subsc
riber_type_str=R egularLine,fdest=1,peer_tag=5336, prog_ind=0},callID=0x62155454) Mar 3
```

04:12:47.616: cc_api_call_setup_ind type 13 , prot 0 Mar 3 04:12:47.620: cc_process_call_setup_ind (event=0x6231C454) Mar 3 04:12:47.620: >>>>CCAPI handed cid 51 with tag 5336 to app "DEFAULT" Mar 3 04:12:47.620: sess_appl: ev(24=CC_EV_CALL_SETUP_IND), cid(51), disp(0) Mar 3 04:12:47.620: sess_appl: ev(SSA_EV_CALL_SETUP_IND), cid(51), disp(0) Mar 3 04:12:47.620: ssaCallSetupInd Mar 3 04:12:47.620: ccCallSetContext (callID=0x33, context=0x626EAC9C) Mar 3 04:12:47.620: ssaCallSetupInd cid(51), st(SSA_CS_MAPPING),oldst(0), ev(24)ev->e.evCallSetupIn d.nCallInfo.finalDestFlag = 1 Mar 3 04:12:47.620: ssaCallSetupInd finalDest cllng(4085272923), clled(3653) Mar 3 04:12:47.620: ssaCallSetupInd cid(51), st(SSA_CS_CALL_SETTING),oldst(0), ev(24)dpMatchPeersMo reArg result= 0 Mar 3 04:12:47.620: ssaSetupPeer cid(51) peer list: tag(3653) called number (3653) Mar 3 04:12:47.620: ssaSetupPeer cid(51), destPat(3653), matched(4), prefix(), peer(62663E7C), peer ->encapType (2) Mar 3 04:12:47.620: ccCallProceeding (callID=0x33, prog_ind=0x0) Mar 3 04:12:47.620: ccCallSetupRequest (Inbound call = 0x33, outbound peer =3653, dest=, params=0x62327730 mode=0, *callID=0x62327A98, prog_ind = 0) Mar 3 04:12:47.624: ccCallSetupRequest numbering_type 0x80 Mar 3 04:12:47.624: ccCallSetupRequest encapType 2 clid_restrict_disable 1 null_orig_clg 0 clid_transparent 0 callingNumber 4085272923 Mar 3 04:12:47.624: dest pattern 3653, called 3653, digit_strip 0 Mar 3 04:12:47.624: callingNumber=4085272923, calledNumber=3653, redirectNumber= display_info= call ing_oct3a=80 Mar 3 04:12:47.624: accountNumber=, finalDestFlag=1, guid=2d3a.ac33.16a4.11cc.8068.8828.285b.8df6 Mar 3 04:12:47.624: peer_tag=3653 Mar 3 04:12:47.624: ccIFCallSetupRequestPrivate: (vdbPtr=0x621B2360, dest=, callParams={called=3653 ,called_oct3=0x80, calling=4085272923,calling_oct3=0x21, calling_xlated=false, subscriber_type_str= RegularLine, fdest=1, voice_peer_tag=3653},mode=0x0) vdbPtr type = 1 *!--- The OGW establishes the second leg.* Mar 3 04:12:47.624: **ccIFCallSetupRequestPrivate:** (vdbPtr=0x621B2360, dest=, callParams={called=3653 , called_oct3 0x80, calling=4085272923,calling_oct3 0x21, calling_xlated=false, fdest=1, voice_pee r_tag=3653}, mode=0x0, xltrc=-5) Mar 3 04:12:47.624: ccSaveDialpeerTag (callID=0x33, dialpeer_tag=0xE45) Mar 3 04:12:47.624: ccCallSetContext (callID=0x34, context=0x626EB9A4) Mar 3 04:12:47.624: ccCallReportDigits (callID=0x33, enable=0x0) Mar 3 04:12:47.624: cc_api_call_report_digits_done (vdbPtr=0x6264A60C, callID=0x33, disp=0) Mar 3 04:12:47.624: sess_appl: ev(52=CC_EV_CALL_REPORT_DIGITS_DONE), cid(51), disp(0) Mar 3 04:12:47.624: cid(51)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_REPORT_DIGITS_DONE) oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(1)fDest(1) Mar 3 04:12:47.624: -cid2(52)st2(SSA_CS_CALL_SETTING)oldst2(SSA_CS_MAPPING) Mar 3 04:12:47.624: ssaReportDigitsDone cid(51) peer list: (empty) Mar 3 04:12:47.624: ssaReportDigitsDone callid=51 Reporting disabled. Mar 3 04:12:47.628: H225 NONSTD OUTGOING PDU ::= value ARQnonStandardInfo ::= { sourceAlias { } sourceExtAlias { } callingOctet3a 128 interfaceSpecificBillingId "ISDN-VOICE" } Mar 3 04:12:47.628: H225 NONSTD OUTGOING ENCODE BUFFER::= 80 000008A0 01800B12 4953444E 2D564F49 43 45 Mar 3 04:12:47.628: Mar 3 04:12:47.628: RAS OUTGOING PDU ::= value RasMessage ::= **admissionRequest** : *!--- ARQ is sent to the gatekeeper.* requestSeqNum 2210 callType pointToPoint : NULL callModel direct : NULL endpointIdentifier {"81206D2C00000001"} destinationInfo { e164 : "3653" } srcInfo { e164 : "4085272923", h323-ID : {"gwa-1"} } bandWidth 640 callReferenceValue 26 nonStandardData { nonStandardIdentifier h221NonStandard : { t35CountryCode 181 t35Extension 0 manufacturerCode 18 } data '80000008A001800B124953444E2D564F494345'H } conferenceID '2D3AAC3316A411CC80688828285B8DF6'H activeMC FALSE answerCall FALSE canMapAlias TRUE callIdentifier { guid '2D3AAC3316A411CC80698828285B8DF6'H } willSupplyUUIEs FALSE } Mar 3 04:12:47.636: RAS OUTGOING ENCODE BUFFER::= 27 8808A100 F0003800 31003200 30003600 44003200 43003000 30003000 30003000 30003000 31010180 69860204 8073B85A 5C564004 00670077 0061002D 00314002 80 001A40 B5000012 13800000 08A00180 0B124953 444E2D56 4F494345 2D3AAC33 16A411CC 80688828 285B8DF6 04E 02001 8011002D 3AAC3316 A411CC80 69882828 5B8DF601 00 Mar 3 04:12:47.640: Mar 3 04:12:47.656: RAS INCOMING ENCODE BUFFER::= 80 050008A1 2327 Mar 3 04:12:47.656: Mar 3 04:12:47.656: RAS INCOMING PDU ::= value RasMessage ::= requestInProgress : { requestSeqNum 2210 delay 9000 } Mar 3 04:12:47.704: RAS INCOMING ENCODE BUFFER::= 2B 0008A140 028000AC 100D1A06 B800EF1A 10C01201 1 0000200 AC100D2A 06B800AC 100D1706 B8010002 0000 Mar 3 04:12:47.704: Mar 3 04:12:47.704: RAS INCOMING PDU ::= value RasMessage ::= **admissionConfirm** : *!--- ACF is received.* { requestSeqNum 2210 bandWidth 640 callModel direct : NULL **destCallSignalAddress ipAddress** : *!--- Primary destination endpoint.* { **ip 'AC100D1A'H** port 1720 } irrFrequency 240 **alternateEndpoints** *!--- List of alternate endpoints.* **{ { callSignalAddress { ipAddress : { ip 'AC100D2A'H** *!--- 172.16.13.42.* **port 1720 }, ipAddress : { ip 'AC100D17'H** *!--- 172.16.13.23.* **port 1720 } } } }** willRespondToIRR FALSE uuiesRequested { setup FALSE callProceeding FALSE connect FALSE alerting FALSE information FALSE releaseComplete FALSE facility FALSE progress FALSE empty FALSE } } Mar 3 04:12:47.720: H225 NONSTD OUTGOING PDU ::= value H323_UU_NonStdInfo ::= { version 2 protoParam qsigNonStdInfo : { iei 4 rawMesg '04038090A31803A983816C0C2180343038353237...'H } } Mar 3 04:12:47.720: H225 NONSTD OUTGOING ENCODE BUFFER::= 60 01020001 041F0403 8090A318 03A98381 6C0C2180 34303835 32373239 32337005 80333635 33 Mar 3 04:12:47.724: Mar 3 04:12:47.724: H225.0

OUTGOING PDU ::= value H323_UserInformation ::= { h323-uu-pdu { **h323-message-body setup** : *!--- H.225 setup sent to primary endpoint.* { protocolIdentifier { 0 0 8 2250 0 2 } sourceAddress { h323-ID : {"gwa-1"} } sourceInfo { gateway { protocol { voice : { supportedPrefixes { { prefix e164 : "1#" } } } } } } mc FALSE undefinedNode FALSE } activeMC FALSE conferenceID '2D3AAC3316A411CC80688828285B8DF6'H conferenceGoal create : NULL callType pointToPoint : NULL sourceCallSignalAddress ipAddress : { ip 'AC100D0F'H port 11025 } callIdentifier { guid '2D3AAC3316A411CC80698828285B8DF6'H } fastStart { '0000000C6013800A04000100AC100D0F47F1'H, '400000060401004C6013801114000100AC100D0F...'H } mediaWaitForConnect FALSE canOverlapSend FALSE } h245Tunneling TRUE nonStandardControl { { nonStandardIdentifier h221NonStandard : { t35CountryCode 181 t35Extension 0 manufacturerCode 18 } data '6001020001041F04038090A31803A983816C0C21...'H } } } } Mar 3 04:12:47.740: H225.0 OUTGOING ENCODE BUFFER::= 20 A0060008 914A0002 01400400 67007700 61002D0 0 31088001 3C050401 00204000 2D3AAC33 16A411CC 80688828 285B8DF6 00451C07 00AC100D 0F2B1111 002D3AAC 3316A411 CC806988 28285B8D F6320212 0000000C 6013800A 04000100 AC100D0F 47F11D40 00000604 01004C60 13801114 000100AC 100D0F47 F000AC10 0D0F47F1 01000100 06A00180 2D0140B5 00001226 60010200 01041F04 0 38090A3 1803A983 816C0C21 80343038 35323732 39323370 05803336 3533 Mar 3 04:12:47.744: Mar 3 04:12:47.760: H225.0 INCOMING ENCODE BUFFER::= 25 80060008 914A0004 11001100 2D3AAC33 16A411C C 80698828 285B8DF6 10800180 Mar 3 04:12:47.760: Mar 3 04:12:47.760: H225.0 INCOMING PDU ::= value H323_UserInformation ::= { h323-uu-pdu { **h323-message-body releaseComplete** : *!--- First setup message failed.* { protocolIdentifier { 0 0 8 2250 0 4 } callIdentifier { guid '2D3AAC3316A411CC80698828285B8DF6'H } } h245Tunneling TRUE } } Mar 3 04:12:47.776: H225 NONSTD OUTGOING PDU ::= value H323_UU_NonStdInfo ::= { version 2 protoParam qsigNonStdInfo : { iei 4 rawMesg '04038090A31803A983816C0C2180343038353237...'H } } Mar 3 04:12:47.776: H225 NONSTD OUTGOING ENCODE BUFFER::= 60 01020001 041F0403 8090A318 03A98381 6C 0C2180 34303835 32373239 32337005 80333635 33 Mar 3 04:12:47.776: Mar 3 04:12:47.776: H225.0 OUTGOING PDU ::= value H323_UserInformation ::= { h323-uu-pdu { **h323-message-body setup** : *!--- Second setup sent to alternate endpoint.* { protocolIdentifier { 0 0 8 2250 0 2 } sourceAddress { h323-ID : {"gwa-1"} } sourceInfo { gateway { protocol { voice : { supportedPrefixes { { prefix e164 : "1#" } } } } } } mc FALSE undefinedNode FALSE } activeMC FALSE conferenceID '2D3AAC3316A411CC80688828285B8DF6'H conferenceGoal create : NULL callType pointToPoint : NULL sourceCallSignalAddress ipAddress : { ip 'AC100D0F'H port 11027 } callIdentifier { guid '2D3AAC3316A411CC80698828285B8DF6'H } fastStart { '0000000C6013800A04000100AC100D0F47F1'H, '400000060401004C6013801114000100AC100D0F...'H } mediaWaitForConnect FALSE canOverlapSend FALSE } h245Tunneling TRUE nonStandardControl { { nonStandardIdentifier h221NonStandard : { t35CountryCode 181 t35Extension 0 manufacturerCode 18 } data '6001020001041F04038090A31803A983816C0C21...'H } } } } Mar 3 04:12:47.796: H225.0 OUTGOING ENCODE BUFFER::= 20 A0060008 914A0002 01400400 67007700 61002D0 0 31088001 3C050401 00204000 2D3AAC33 16A411CC 80688828 285B8DF6 00451C07 00AC100D 0F2B1311 002D3AAC 3316A411 CC806988 28285B8D F6320212 0000000C 6013800A 04000100 AC100D0F 47F11D40 00000604 01004C60 13801114 000100AC 100D0F47 F000AC10 0D0F47F1 01000100 06A00180 2D0140B5 00001226 60010200 01041F04 0 38090A3 1803A983 816C0C21 80343038 35323732 39323370 05803336 3533 Mar 3 04:12:47.800: Mar 3 04:12:47.872: H225.0 INCOMING ENCODE BUFFER::= 21 80060008 914A0003 00078E11 002D3AAC 3316A41 1 CC806988 28285B8D F6390219 0000000C 60138011 14000100 AC100D17 479E00AC 100D1747 9F1D4000 00060401 004C6013 80111400 0100AC10 0D0F47F0 00AC100D 17479F01 00010008 800180 Mar 3 04:12:47.872: Mar 3 04:12:47.876: H225.0 INCOMING PDU ::= value H323_UserInformation ::= { h323-uu-pdu { **h323-message-body callProceeding** : *!--- Call proceeding received.* { protocolIdentifier { 0 0 8 2250 0 3 } destinationInfo { mc FALSE undefinedNode FALSE } callIdentifier { guid '2D3AAC3316A411CC80698828285B8DF6'H } fastStart { '0000000C6013801114000100AC100D17479E00AC...'H, '400000060401004C6013801114000100AC100D0F...'H } } h245Tunneling TRUE } } Mar 3 04:12:47.884: H225.0 OUTGOING PDU ::= value H323_UserInformation ::= { h323-uu-pdu { h323-message-body empty : NULL h245Tunneling TRUE h245Control { '0270010600088175000380138000140001000001...'H } } } Mar 3 04:12:47.884: H225.0 OUTGOING ENCODE BUFFER::= 28 10010006 C0018063 01610270 01060008 8175000 3 80138000 14000100 00010000 0100000C C0010001 00048000 104810B5 0000120C 52747044 746D6652 656C6179 00008000 16830150 80001583 01408000 12830110 80000020 C0130080 01020000 16020015 00120010 000000 Mar 3 04:12:47.888: Mar 3 04:12:47.888: H225.0 OUTGOING PDU ::= value H323_UserInformation ::= { h323-uu-pdu { h323-message-body empty : NULL h245Tunneling TRUE h245Control { '01003C4010F3'H } } } Mar 3 04:12:47.892: H225.0 OUTGOING ENCODE BUFFER::= 28 10010006 C0018008 01060100 3C4010F3 Mar 3 04:12:47.892: Mar 3 04:12:47.892: **cc_api_call_proceeding**(vdbPtr=0x621B2360, callID=0x34, prog_ind=0x0) Mar 3 04:12:47.896: sess_appl: ev(21=CC_EV_CALL_PROCEEDING), cid(52), disp(0) Mar 3 04:12:47.896: cid(52)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_PROCEEDING) oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(0)fDest(0) Mar 3 04:12:47.896: -cid2(51)st2(SSA_CS_CALL_SETTING)oldst2(SSA_CS_CALL_SETTING) Mar 3 04:12:47.896: ssaCallProc Mar 3 04:12:47.896: ccGetDialpeerTag (callID=0x33) Mar 3 04:12:47.896: ssaIgnore cid(52),

```
st(SSA_CS_CALL_SETTING),oldst(1), ev(21) Mar 3 04:12:47.900: H225.0 INCOMING ENCODE BUFFER::= 28
10010008 C0018063 01610270 01060008 8175000 6 80138000 14000100 00010000 0100000C C0010001
00048000 104810B5 0000120C 52747044 746D6652 656C6179 00008000 16830150 80001583 01408000
12830110 80000020 C0130080 01020000 16020015 00120010 000000 Mar 3 04:12:47.904: Mar 3
04:12:47.904: H225.0 INCOMING PDU ::= value H323_UserInformation ::= { h323-uu-pdu { h323-
message-body empty : NULL h245Tunneling TRUE h245Control {
'0270010600088175000680138000140001000001...'H } } } !--- Some of the unnecessary H.225 debug
messages are deleted here. Mar 3 04:12:52.116: H225.0 INCOMING ENCODE BUFFER::= 23 80060008
914A0003 000A8600 11002D3A AC3316A 4 11CC8069 8828285B 8DF60100 01000880 0180 Mar 3
04:12:52.120: Mar 3 04:12:52.120: H225.0 INCOMING PDU ::= value H323_UserInformation ::= { h323-
uu-pdu { h323-message-body alerting : !--- Alerting message received. { protocolIdentifier { 0 0
8 2250 0 3 } destinationInfo { mc FALSE undefinedNode FALSE } callIdentifier { guid
'2D3AAC3316A411CC80698828285B8DF6'H } } h245Tunneling TRUE } } Mar 3 04:12:52.124:
cc_api_call_alert(vdbPtr=0x621B2360, callID=0x34, prog_ind=0x8, sig_ind=0x1) Mar 3 04:12:52.124:
sess_appl: ev(7=CC_EV_CALL_ALERT), cid(52), disp(0) Mar 3 04:12:52.124:
cid(52)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_ALERT) oldst(SSA_CS_CALL_SETTING)cfid(-
1)csize(0)in(0)fDest(0)
```

Resolución de problemas de balance de cargas

Con la característica de la balanza de la carga, usted puede fijar al portero con cierto umbral para el número de llamadas, de memoria, de CPU, y de número de puntos finales registradoes. Una vez que se alcanza ese umbral, el portero mueve los puntos finales registradoes de Cisco H.323 a un gatekeeper alternativo o rechaza las nuevos llamadas y registros. El equilibrio de carga se habilita mediante el uso del siguiente comando gatekeeper CLI:

```
Router(config-gk)#load-balance [endpoints max-endpoints] [calls max-calls] [cpu max-%cpu][memory
max-%mem-used]
```

Cuando se alcanza el umbral, el gatekeeper usa el mensaje RRJ RAS para informarle al punto final sobre los gatekeeperes alternos y el motivo de rechazo. Cuando se recibe este mensaje, el punto final envía un nuevo RRQ al gatekeeper alternativo. Una vez que está registrado con el gatekeeper alternativo, utiliza el mensaje GUP para informarle a todos los gatekeepers en el agrupamiento acerca del nuevo punto final registrado.

Algunos de las precauciones a tener en cuenta cuando esté solucionando problemas son verificar la configuración en el gatekeeper y asegurarse de que los gatekeepers alternativos y el equilibrio de carga estén funcionando. La topología anterior se utiliza para solucionar problemas. La configuración de gatekeeper "gkb-1" se modificará para mostrar los siguientes casos:

- Cómo el portero puede rechazar una llamada cuando se resuelve un umbral.
- Cómo el control de acceso puede mover el registro de un punto final a un control de acceso alternativo cuando se alcanza un umbral.

Para hacer el debug de la característica del Equilibrio de carga, utilice la **carga del gatekeeper del debug** y **haga el debug del asn1 del h225** para ver cómo el portero reacciona cuando se resuelve el umbral.

Esta es la configuración del gatekeeper "gkb-1" que se utiliza para cubrir los dos casos mencionados anteriormente (cantidad de umbrales de llamada y cantidad de puntos extremos registrados):

```
!
gatekeeper
 zone local gkb-1 domainB.com 172.16.13.41
 zone remote gka-1 domainA.com 172.16.13.35 1719
 zone cluster local gkb gkb-1
  element gkb-2 172.16.13.16 1719
 !
```

```
 security token required-for all
 gw-type-prefix 2#* default-technology
 bandwidth total zone gkb-1 512
 bandwidth session zone gkb-1 512
 load-balance endpoints 2 calls 1 !--- maximum of 2 endpoints and call threshold is 1 no
shutdown ! !
```

Una llamada se hace a través del portero gkb-1. Mientras que esa llamada está para arriba, se hace otra llamada. La depuración capturada muestra cómo se ve la depuración del equilibrio de carga y cómo el control de acceso rechaza la segunda llamada porque se llegó al umbral. Para mostrar cómo se ejecutan varias llamadas activas utilizando el control de acceso, puede utilizar el siguiente comando:

```
gkb-1#show gatekeeper call Total number of active calls = 1. GATEKEEPER CALL INFO
==================== LocalCallID Age(secs) BW 5-29514 9 128(Kbps) Endpt(s): Alias E.164Addr src
EP: gwa-1 4085272923 Endpt(s): Alias E.164Addr dst EP: gwb-1 3653 CallSignalAddr Port
RASSignalAddr Port 172.16.13.23 1720 172.16.13.23 54670
```

Aquí se encuentra la depuración de H.225 asn 1 y la carga de gatekeeper cuando se solicita la segunda llamada.

```
Mar  3 05:04:55.354: RAS INCOMING ENCODE BUFFER::= 4A 80080501 01806986
40B50000 12298286 B0110075 7
95BF216 AB11CC80 95882828 5B8DF601 81110201 80866940 04006700 77006100
2D003100 AC100D23 06B70B80 0D
014004 0067006B 0061002D 00310180
Mar  3 05:04:55.358:
Mar  3 05:04:55.358: RAS INCOMING PDU ::=

value RasMessage ::= locationRequest : !--- LRQ is received. { requestSeqNum 2054
destinationInfo { e164 : "3653" } nonStandardData { nonStandardIdentifier h221NonStandard : {
t35CountryCode 181 t35Extension 0 manufacturerCode 18 } data
'8286B0110075795BF216AB11CC80958828285B8D...'H } replyAddress ipAddress : { ip 'AC100D23'H port
1719 } sourceInfo { h323-ID : {"gka-1"} } canMapAlias TRUE } Mar 3 05:04:55.362: H225 NONSTD
INCOMING ENCODE BUFFER::= 82 86B01100 75795BF2 16AB11CC 80958828 28 5B8DF6 01811102 01808669
40040067 00770061 002D0031 Mar 3 05:04:55.366: Mar 3 05:04:55.366: H225 NONSTD INCOMING PDU ::=
value LRQnonStandardInfo ::= { ttl 6 nonstd-callIdentifier { guid
'75795BF216AB11CC80958828285B8DF6'H } callingOctet3a 129 gatewaySrcInfo { e164 : "5336", h323-ID
: {"gwa-1"} } } Mar 3 05:04:55.366: gk_load_overloaded: Overloaded due to reaching specified
call limits !--- Number of calls threshold has met. Mar 3 05:04:55.370: RAS OUTGOING PDU ::=
value RasMessage ::= locationReject : !--- LRJ is sent. { requestSeqNum 2054 rejectReason
undefinedReason : NULL }
```

Para el segundo ejemplo, el gatekeeper "gkb-1" tiene dos puntos finales registrados. El registro para otro punto final se intenta. El portero movió el punto final, intentando registrar al gatekeeper alternativo el "gkb-2", puesto que él está en el mismo cluster. Éste es el mensaje de depuración para debug h225 asn1 y debug gatekeeper gup asn1 para este caso:

```
Mar  3 05:21:05.682: RAS INCOMING PDU ::=

value RasMessage ::= registrationRequest : !--- RRQ message is received. { requestSeqNum 4621
protocolIdentifier { 0 0 8 2250 0 3 } discoveryComplete TRUE callSignalAddress { ipAddress : {
ip 'AC100D2A'H port 1720 } } rasAddress { ipAddress : { ip 'AC100D2A'H port 49998 } }
terminalType { gateway { protocol { voice : { supportedPrefixes { { prefix e164 : "1#" } } } } }
mc FALSE undefinedNode FALSE } terminalAlias { h323-ID : {"gwb-3"} } gatekeeperIdentifier {"gkb-
1"} endpointVendor { vendor { t35CountryCode 181 t35Extension 0 manufacturerCode 18 } }
timeToLive 60 tokens { { tokenOID { 1 2 840 113548 10 1 2 1 } timeStamp 731136065 challenge
'5A70CA112E6C7A3834792BD64FF7AD2F'H random 58 generalID {"gwb-3"} } } cryptoTokens {
cryptoEPPwdHash : { alias h323-ID : {"gwb-3"} timeStamp 731136065 token { algorithmOID { 1 2 840
113549 2 5 } paramS { } hash "B1C1DAD962BEE42B1E53F368238B1D8" } } } keepAlive FALSE
willSupplyUUIEs FALSE maintainConnection TRUE } Mar 3 05:21:05.698: gk_load_overloaded:
Overloaded due to reaching specified endpoint limits !--- Endpoint threshold is met. Mar 3
05:21:05.702: RAS OUTGONG PDU ::= value RasMessage ::= registrationReject : !--- RRJ is sent. {
```

```
requestSeqNum 4621 protocolIdentifier { 0 0 8 2250 0 3 } rejectReason resourceUnavailable : NULL
!--- Reject reason. gatekeeperIdentifier {"gkb-1"} altGKInfo !--- List of alternate gatekeepers.
{ alternateGatekeeper { { rasAddress ipAddress : { ip 'AC100D10'H port 1719 }
gatekeeperIdentifier {"gkb-2"} needToRegister TRUE priority 0 } } altGKisPermanent TRUE !---
Informs the endpoint that the move is permanent. } Mar 3 05:21:05.706: RAS OUTGOING ENCODE
BUFFER::= 16 80120C06 0008914A 00038101 00080067 006B0062 0 02D0031 07001600 0140AC10 0D1006B7
08006700 6B006200 2D003280 80 Mar 3 05:21:05.706: Mar 3 05:21:05.782: Received GUP REGISTRATION
INDICATION from 172.16.13.16 !--- GUP update for the new endpoint. gkb-1#
```

Información Relacionada

- Soporte de tecnología de voz
- Soporte para productos de comunicaciones IP y por voz
- Troubleshooting de Cisco IP Telephony
- Soporte Técnico - Cisco Systems