APV 10.4.3
用户手册

北京信安世纪科技股份有限公司
二零二四年

版权声明

本手册所涉及的案例均是当前情况，华耀有权利随时更改，恕不提前通知。对于本手册内容，包括但不限于隐含的商业性能和特定用途适应性说明，华耀不承担任何责任。

如因本手册的描述造成设备性能、使用和按键操作问题，进而引发相关事故并造成损失的，华耀不承担任何责任。

警告：未经华耀许可，任何人不得对华耀 APV 设备进行任何改动，否则将无权继续使用该设备。

商标声明

本手册中所涉及的产品名称仅作识别之用。手册中涉及的其他公司的注册商标或版权属各商标注册人所有，恕不逐一列明。

合格声明

华耀自主声明华耀 APV 系列产品符合 FCC 规范第 15 部分的规定。操作本设备需要满足以下条件：（1）本设备不会产生有害干扰，（2）本设备必须能够接受收到的所有干扰，包括可能导致意外操作的干扰。

关于华耀

北京华耀科技有限公司（简称：华耀）于 2003 年创建于北京，是优秀的网络功能平台解决方案提供商，也是应用交付解决方案、移动应用接入（SSL VPN）解决方案的全球领导者。华耀总部位于北京，并在北京、上海、广州、杭州、深圳等全国多地设有销售及技术支持部门，负责销售及客户支持工作。

华耀一贯秉持为用户打造敏捷灵活与安全性能兼顾的网络环境的理念。作为多年的应用交付解决方案全球领导者，华耀确保应用性能、高可靠性和安全性的同时，将应用推送到终端用户。通过华耀产品，用户可以使用任何设备、从任何地点访问云环境或企业数据中心的应用、桌面或服务。从 Web 站点、到电子商务、到企业应用、再到云服务，华耀解决方案提供了卓越的终端用户体验和可靠的安全性，全力保障企业的运营效率。

联系华耀

请通过以下方式联系华耀：

- 官方网站：https://www.arraynetworks.com.cn/
- 电子信箱：support@arraynetworks.com.cn
- 地址：北京市海淀区建枫路（南延）6 号西三旗金隅科技园 2 号楼信安大厦
- 邮编：100096
- 电话：010-68025518、400-6007878
- 电话联系时间：周一至周五早 9 点至晚 6 点

目录

第1章 引言

1.1 编写目的

本手册详细介绍了应用安全网关产品的系统各模块的功能原理，以及如何通过 CLI 命令配置具体功能，指导用户正确使用该系统。

1.2 适用对象

本手册适用对象为售前、产品实施及技术支持人员、网络管理员、以及测试人员等，假定具备以下概念知识：

- 网络拓扑
- 网络地址、路由和 DNS
- TCP/IP、SSL、HTTP/HTTPS

第2章 产品简介

2.1 产品概述

本设备能够在极大地提高企业核心应用和业务平台的可用性、性能以及安全性的同时，降低企业数据中心成本和复杂性。作为一个为企业级应用服务设计的应用交付解决方案，设备代表了新一代的应用负载均衡和性能优化产品，仅仅通过一个功能强大的系统，即可提供高度集成的应用交付功能。

本设备可以提供千兆级的七层应用吞吐效率，以及深层次的应用协议和数字证书字段的分析和内容交换。不论是大中型企业还是电信运营商，应用设备解决方案，将确保 7×24 小时的应可用性，避免系统宕机或网络故障对营业收入带来的影响，并极大提升应用效率和性能，显著提高数据中心的效率和投资回报率（Return on Investment，ROI）。

2.2 产品优势

- 保障应用的高可用性
- 确保网络的高可用性
- 突破性的提升用户访问体验
- 强化应用安全保护
- 易于使用和部署

第3章 设备安装

3.1 概述

本章将介绍设备的安装环境、准备工作和安装步骤。

3.2 安装要求

3.2.1 电源要求

设备的运行需要标准 220V 电压。

安装设备之前请关闭电源，在拆装或移动设备之前须先切断电源。

电源插座尽量不要远离设备的电源输入接口，以便为设备断电时方便切断电源。

3.2.2 防静电要求

如果设备的电源涉及到室外布线，需要采取防雷保护措施。

超过一定容限的静电会对电路乃至整机产生严重的破坏作用，因此应确保设备良好的接地以防止静电的破坏。

另外人体的静电也会导致设备内部元器件和印刷电路损坏，所以如有条件最好能够佩戴防静电手腕。

3.2.3 环境要求

3.2.3.1 基本条件

要将设备安装在一间温度和湿度都可以控制的设备房中，要注意放置处周围物质的导电性，如果湿度太大可能造成短路发生危险，如果房间太干燥的话可能引起火灾，因此必须要有一个适当的环境放置设备。

设备一般要求安装在防火、防潮、防尘、防盗的机房里，旁边有稳定可靠的电源，有条件的尽量安装到机柜上。

3.2.3.2 温度湿度及光照

建议温度-0~40℃，湿度在 5~95RH 无凝结之间。室内光线要明亮，至少保证操作时不受影响，建议在 500~750 流明/平方米。

湿度过高会影响绝缘材料的绝缘效果，也易导致设备中金属部件的锈蚀；湿度过低会引起绝缘垫片干缩，使紧固螺钉松动；干燥的环境易对设备产生静电危害；长期高温将加速各种部件的老化，使设备的可靠性大大降低，严重影响其寿命。

3.2.3.3 空气质量

由于空气中的微粒可能被从吸气孔吸入电源，长时间积累，会造成电路板上出现短路危险，因此对空气的纯度有一定要求，建议微粒的浓度小于 180 毫克/立方。对安装设备的机房来说打印机、复印机要远离安装有供电设备、电源的机柜，以免纸屑墨粉被吸入供电设备，您可以把它们放在房子的对角处。

应保证设备运行环境的清洁度，因为灰尘落在机体上可以造成静电，若当灰尘吸附在机体内时，还会使金属接插件或金属接点接触不良。

3.2.3.4 通风条件

设备的后部和侧部有散热栅，用于内外空气的流通，以达到冷却的目的，所以为确保空气流通，在设备的两侧和后面保留 100mm 的空间，不要让空气入口和出口被阻塞。

3.2.3.5 安全措施

设备安装的机房要有一定的安全措施，比如专人看管或其他有效的防盗措施，要有防雷击、闪电措施，接地良好，接地电缆直径不小于 5 毫米。地震多发区要采取防震措施，比如放置在防震的机房里机柜加装有防震固定护栏。

3.3 准备工作

3.3.1 网络资源

无论您是外部使用，还是内部使用，设备都需要连接到网络上。当然这个网络可以是私网也可以是公网。所以首先查看您现在的网络是否连接正常，并为设备在您的网络上找一个接口。一般可接在交换机或路由器上。

3.3.2 IP 地址

根据网络具体结构和需求，为设备分配公网或私网地址，所分配的 IP 地址既可以是 IPv4 地址也可以是 IPv6 地址。

注意：管理员可以给设备配置静态 IP 地址，此外也可以通过配置“**ip dhcp**”命令使设备通过 DHCP 服务器自动获取 IP 地址。

3.3.3 网线

准备标准 RJ-45 接头的网线或光纤网线来连接设备与您所在的网络。

3.3.4 网络拓扑

根据设备所在网络中的连接位置及相关外线连接情况等基本信息，绘出网络拓扑图。

3.4 安装过程

设备可以被安放到桌面上，也可以安装在机架中，请利用产品包装箱内提供的设备，按以下步骤进行安装。

注意：不要在设备上放置任何重物，以免将设备底盘压坏。另外安装时应确保入风口和风扇出口侧没有被阻塞。

3.4.1 网络连接

3.4.1.1 内网连接

将网线一端接到设备前面板的 Port1 以太网口上；将网线的另一端与连接到局域网 Hub/Switch 的普通网口上；用另一根网线将您计算机的 LAN 口连接到 Hub/Switch 上。

3.4.1.2 外线连接

将另一根网线一段连接到设备前面板的 Port2 以太网口上；将该网线的另一端同 Router/FireWall 的相关接口连接。

3.4.2 连接电源

将随机附带的电源线与设备后面板的电源接孔连接起来；将电源线的另一端插入到离设备较近的电源插座上。

3.4.3 设备上电

在按照以上步骤完成设备的安装、连接后，就可以为设备接通电源，等待系统启动成功（这里设备需要一点时间来完成它的自检）。

第4章 使用管理系统

4.1 概述

本章将介绍使用控制台连接、SSH 连接和 WebUI 接口三种方式访问设备并对其进行配置和管理。

4.2 控制台连接

如果需要通过控制台连接设备，首先将 Console 线连接到设备的 Console 接口上，然后将控制面板（如超级终端）设置如下：

- 支持 VT100 终端
- 波特率为 9600
- 数据位为 8 位
- 不设置同位
- 停止位为 1 位
- 没有控制位

通过控制台终端成功连接到设备后，使用者就可以进入到命令行模式，并对设备进行配置和管理。

4.3 SSH 连接

在 SSH 服务被启动并且 IP 地址已确定的情况下，可以对设备作出自定义配置。下面是一个使用 SSH 连接到命令行的示例。

注意：Windows、MacOS 或者 Unix 下的 SSH 软件，可以从 http://www.openssh.com 网站获得。

在工作站上运行 SSH 程序，输入如下指令，并用设备的 IP 地址作为参数。在下面的例子中，我们使用 10.3.55.251 作为设备的 IP 地址。

```
>> # ssh array@10.3.55.251
```

成功建立连接后，设备会提示管理员输入密码。系统默认的用户名为 array，口令是 admin。

```
>> # ssh array@10.3.55.251
>> # array@10.3.55.251's password:
```

注意：管理员必须保证 IP 地址配置和基础网络配置全部正确，才能通过 SSH 正确的连接到设备。

4.4 WebUI 连接

网络用户配置接口（Web User Interface，WebUI）使用户可以通过直观、友好的图形化接口，对设备进行配置和管理，达到与命令行配置同样的配置效果，大大方便了用户的使用和操作。

设备的 WebUI 功能具有以下优点：

- 通过快速响应来改善用户体验；
- 将设备的功能和性能最大化；
- 实现更简便的系统配置和管理。

注意：WebUI 功能默认是关闭的。如果想通过 WebUI 界面对设备进行配置和管理，请首先参考 5.5.4 小节“启动 WebUI”。

设备提供 WebUI 进行系统管理和配置，可以通过在浏览器的地址栏内输入设备的 IP 地址和端口号访问。

访问 WebUI，例如（8888 为 WebUI 的默认端口号）：

```
https://192.168.1.200:8888
```

按下回车，浏览器内出现欢迎界面，提示输入用户名和密码。系统默认的用户名为 array，密码是 admin。输入正确的用户名和密码后，就可以进入到 WebUI 界面。

WebUI 支持通过 Chrome、Safari 和 Edge 浏览器访问。另外浏览器分辨率应设置为至少 1024 x 768。

4.5 WebUI SSL 设置

4.5.1 SSL 客户端认证配置

在 SSL 通信中，如果对客户端身份没有认证的必要，SSL 协商流程通常只对服务器身份进行认证，即执行 SSL 单向认证。在某些对客户端身份有严苛要求的场景中，SSL 协商流程需要同时对客户端身份和服务器身份进行认证，即需要执行 SSL 双向认证。

设备支持 WebUI SSL 客户端认证功能，来满足特定应用场景中对于 SSL 双向认证的需求。同时，设备还支持 WebUI SSL 客户端认证强制模式。启用强制模式

后，WebUI 客户端必须通过客户端身份认证后才能与设备 WebUI 建立 SSL 连接。否则，WebUI 访问将失败。如果只开启客户端认证而不开启强制认证模式，客户端不提供证书，但管理员仍然能访问 WebUI。

- **CLI 配置示例**

要启用 WebUI SSL 客户端认证功能，执行以下步骤：

1. 为 WebUI 导入证书链文件。

```
Demo(config)#webui ssl import certificate ftp://10.8.6.20/cert/chain.pem
```

2. 导入客户端 CA 证书。

```
Demo(config)#webui ssl import clientca ftp://10.8.6.20/cert/webui.pem
```

3. 启用 WebUI SSL 客户端认证功能。

```
Demo(config)#webui ssl settings clientauth enable
```

4. 启用 WebUI SSL 客户端认证强制模式。

```
Demo(config)# webui ssl settings authmandatory enable
```

4.5.2 SSL 协议版本和密码套件设置

设备还支持修改 WebUI（SSL 服务器）支持的 SSL 协议版本和密码套件。

执行以下命令修改 WebUI 支持的 SSL 协议版本：

```
Demo(config)#webui ssl settings protocol TLSv11:TLSv12
```

执行以下命令修改 WebUI 支持的 SSL 密码套件：

```
Demo(config)#webui ssl settings ciphersuites
“ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:AES128-GCM-SHA256:AES256-GCM-SHA384”
```

第5章 初始设置

5.1 概述

本章将介绍设备的基本网络拓扑、用户访问级别、命令行接口结构以及其他基本配置。

5.2 基本网络拓扑

为了使您更快的了解配置策略进而将设备的功能发挥的更好，请先了解您自己的基础网络体系。本文将主要采用以下网络拓扑结构用于举例说明。

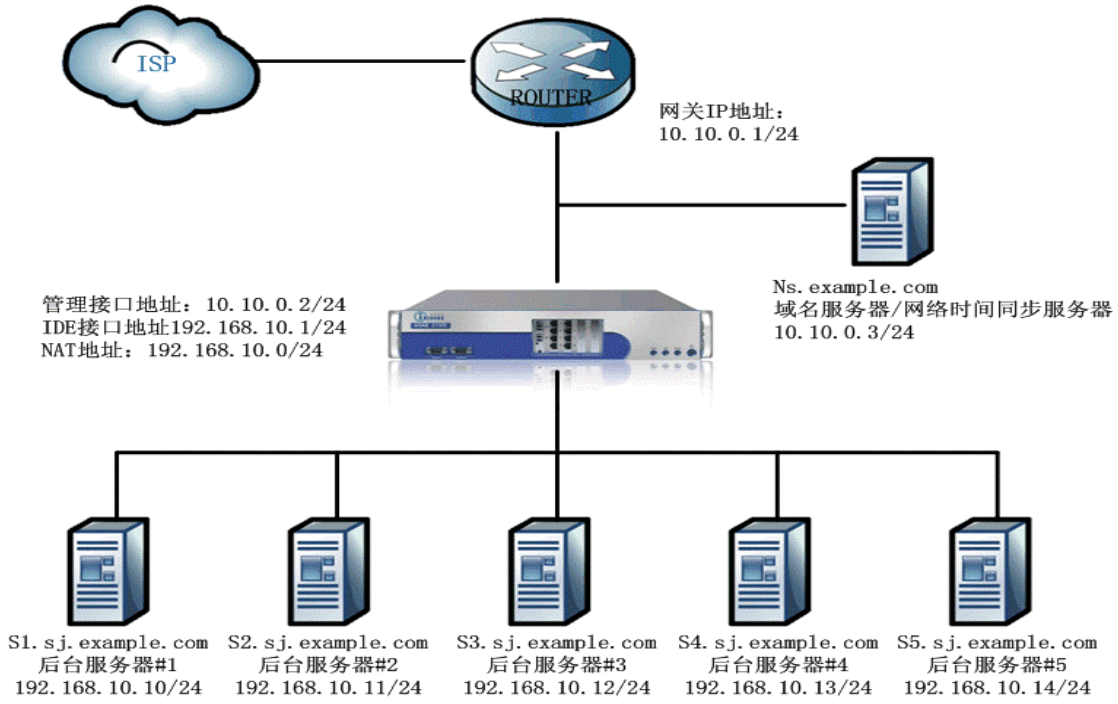

图5–1 基本网络拓扑

如果想正确的将设备连接到网络中使用，必须对设备的 Port1 接口、Port2 接口和网关等项目的 IP 地址进行正确配置。下表列出了以上网络拓扑结构中的一些关键的 IP 地址配置项。

表5–1 基本网络配置

IP 地址	描述
10.10.0.1/24	网关 IP 地址
10.10.0.2/24	管理 IP 地址
192.168.10.1/24	Port2 接口地址
192.168.10.0/24	NAT 映像地址
192.168.10.10	后台服务器#1 地址

IP 地址	描述
192.168.10.11	后台服务器#2 地址
192.168.10.12	后台服务器#3 地址
192.168.10.13	后台服务器#4 地址
192.168.10.14	后台服务器#5 地址
10.10.0.3	域名服务器/网络时间同步服务器地址

5.3 用户访问级别

使用者可以选择使用 CLI 命令行来对设备进行配置。CLI 为连接到设备的使用者提供了多级别的配置方法和连接方式。每个管理级别都有不同的命令行提示符。CLI 提示符的组成形式为：设备的主机名称，后面跟“>”或“#”或“(config)#”。在 CLI 命令行中，用户的操作权限被分为三个级别。

5.3.1 用户级别

第一个访问级别为用户（User）级别，用于解决基础网络故障诊断。处于这个级别的用户，只能使用一些基本的命令，如“**ping**”和“**traceroute**”等基本网络诊断命令，对设备的一些非关键的功能进行管理。使用者刚登录到设备时，所处的就是 User 级别。下面是用户级别的命令行提示符。

```
Demo>
```

5.3.2 权限级别

第二个访问级别为 Enable 级别。这个级别的用户可以使用的命令大多数只能用于查看信息，比如使用“**show version**”命令可以查看版本相关信息。Enable 级别的用户还具有 User 级别命令的权限。

如果想要进入到这个级别，需要执行“**enable**”命令。执行这个命令后，系统会提示用户输入正确的密码（缺省密码为空，按“回车”键即可）。如果 CLI 提示符从“Demo>”变成“Demo#”，那么说明用户已经成功进入 Enable 级别。

```
Demo>enable
Enable password:
Demo#
```

5.3.3 配置级别

最后一个访问级别为配置（Config）级别。这个级别的使用者可以使用设备上的所有命令（包括 User 级别和 Enable 级别的命令）对设备的功能进行配置和管理，也可以对现有的配置进行修改。系统默认同一时间只允许一个用户连接到 Config 级别。

如果想要进入 Config 级别，使用者需要执行下面的命令：

```
Demo#config terminal
```

执行该命令后，CLI 命令行提示符将转变成：

```
Demo(config)#
```

如果无法进入 Config 模式，可能有其它用户正处于 Config 模式。管理员可在执行“**config terminal**”命令时添加参数“force”，这样设备将会强制正处于该级别的用户退出 Config 模式，使当前用户可以进入 Config 级别进行配置。

```
Demo#config terminal force
```

此外，系统支持多登录 Config 级别功能，即同一时间允许多个管理员账户处于 Config 级别或同一管理员账户开启多个 Config 级别的 SSH 连接。管理员可以通过执行“config multiconfig enable”命令进行启用。

在每一个连接级别上，用户都可以通过输入“**?**”来获取帮助。例如，在配置级别下输入“**slb real ?**”将得到命令“**slb real**”可以使用的所有参数和协议。

```
Demo(config)#slb real ? [enter]
activation      Recovery and warm-up time of real service
disable         Remove real service from load balancing
dns             Define SLB DNS real service
enable          Activate real service for load balancing
ftp             Define SLB FTP real service
…
```

5.4 命令行接口结构

命令行接口（Command Line Interface，CLI）将设备的功能和性能发挥到极至。通过 CLI，你可以配置和控制设备的关键功能，从而可以使你的服务器获得更好的性能及连通性。

设备的软件具有很强的交互性能。使用 CLI 命令时，可以通过只输入命令单词的首字母，完成所需功能，方便快捷。其它的便捷方式如下所示。

表5–2 CLI 快捷键

CLI 快捷键	作用
^a/^e	将光标移动到行的开头/结尾
^f/^b	将光标向前/向后移动一个字符
Esc-f	将光标向前移动一个单词
Esc-b	将光标向后移动一个单词
^d	删除光标对应的字符
^k	删除游标处到行尾的内容

CLI 快捷键	作用
^u	删除输入的行

注意：使用“^”这个符号时，需要按住键盘上的 Ctrl 键，同时按“^”之后的字母。

设备 CLI 命令行中采用不同的字体和符号来代表不同含义，如下表所述：

表5–3 命令行字体及符号说明

格式	含义
粗体	命令行主体采用加粗字体表示。
斜体	命令行参数采用斜体表示。
< >	表示用“< >”括起来的参数在配置时是必选的。
[ ]	表示用“[ ]”括起来的参数在配置时是可选的。 表示“**no**”、“**show**”和“**clear**”等子命令。
{x|y|…}	表示从两个或多个选项中选取一个或多个。
[x|y|…]	表示从两个或多个选项中选取一个或者不选。

下面是一个 CLI 命令行的示例。

ip address *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname}* <*ip_address*> <*netmask*>

注意：当参数取值为字符串时，推荐将参数取值置于双引号中，以确保设备能够正确执行命令。

5.5 命令行配置示例

下面列出了进行基本网络配置所需的命令。对于具体的描述信息，请参考命令行使用手册。

表5–4 设备基本网络配置命令

配置操作	命令行
配置接口 IP 地址	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname}* <*ip_address*> <*netmask*> **ip dhcp {on|off}** <*interface_name*>
设置默认的网关 IP 地址	**ip route default** <*gateway_ip*>
查看 IP 地址配置	**ping** *{ip|hostname}* **show ip address** **show ip route**

配置操作	命令行
启动 WebUI	**webui {on|off}** **webui port** <*port*> **webui ip** <*ip_address*>
更改设备名	**hostname** <*host_name*>
保存配置	**write memory**

5.5.1 为接口配置 IP 地址

根据图 4-1 基本网络拓扑，我们现在来为设备的 Port1 接口和 Port2 接口设置对应的 IP 地址和子网掩码，应当使用以下配置命令：

```
Demo(config)#ip address port1 10.10.0.2 255.255.255.0
Demo(config)#ip address port2 3fff::bb 64
```

Port2 接口和 Port1 接口不能指定相同的 IP 地址。如果设置了相同的 IP 地址，CLI 将拒绝执行这条命令并会发出一条警告信息。

设备支持对系统接口的 MAC 地址进行更改，可以使用命令“**interface mac** <*interface_name*> <*mac_address*>”进行配置。

```
Demo(config)#interface mac port1 00:30:48:81:54:9c
```

注意：管理员必须确保终端使用者可以基于当前的 IP 地址和协议类型通过路由直接向外发送信息。

5.5.2 设置默认的网关 IP 地址

接下来我们将为设备设置默认的网关 IP 地址，使用下面的配置命令：

```
Demo(config)#ip route default 10.10.0.1
```

5.5.3 检查 IP 地址的配置

为了验证系统确实正确的连接到了网络中，可以使用“**ping**”命令测试是否可以正确连通网关和后台服务器。

用“**ping**”命令测试与网关的连通性：

```
Demo(config)#ping 10.10.0.1
PING 10.10.0.1(10.10.0.1): 56 data bytes
64 bytes from 10.10.0.1: icmp_seq=0 ttl=128 time=0.671 ms
64 bytes from 10.10.0.1: icmp_seq=1 ttl=128 time=0.580 ms
64 bytes from 10.10.0.1: icmp_seq=2 ttl=128 time=0.529 ms
64 bytes from 10.10.0.1: icmp_seq=3 ttl=128 time=0.486 ms
```

```
64 bytes from 10.10.0.1: icmp_seq=4 ttl=128 time=0.638 ms

--- 10.10.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.486/0.581/0.671/0.068 ms
```

用 **ping** 命令测试与后台服务器的连通性：

```
Demo(config)#ping 192.168.10.1
PING 192.168.10.1(192.168.10.156 data bytes
64 bytes from 192.168.10.1: icmp_seq=0 ttl=128 time=0.661 ms
64 bytes from 192.168.10.1: icmp_seq=1 ttl=128 time=0.581 ms
64 bytes from 192.168.10.1: icmp_seq=2 ttl=128 time=0.552 ms
64 bytes from 192.168.10.1: icmp_seq=3 ttl=128 time=0.484 ms
64 bytes from 192.168.10.1: icmp_seq=4 ttl=128 time=0.632 ms

--- 192.168.10.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.486/0.581/0.671/0.068 ms
```

想要确认或查看 IP 地址的配置，可以通过下列命令实现：

```
Demo(config)#show ip address
ip address "port1" 10.10.0.2 255.255.255.0
ip address "port2"    192.168.10.1 255.255.255.0

Demo(config)#show ip route
Destination        Netmask             Gateway
default                                 10.10.0.1
```

可以通过在命令前加“**no**”来删除某个已经输入的配置。例如，命令“**no ip address port1**”将删除您之前在 Port1 接口上的 IP 地址配置。

5.5.4 启动 WebUI

如果管理员希望通过 WebUI 接口连接到设备，首先需要为 WebUI 连接分配一个 IP 地址。在下面的示例里，我们使用设备的 Port1 接口的 IP 地址作为 WebUI 的默认 IP 地址，使用默认的端口 8888。

执行以下命令启用 WebUI：

```
Demo(config)#webui on
```

注意：

如果需要自定义 WebUI 使用的 IP 地址，请使用“**webui ip**”命令。

如果需要自定义 WebUI 使用的端口号，请使用“**webui port**”命令。

然后我们就可以通过 Web 浏览器，使用 WebUI 界面来管理和配置我们的设备了。使用 WebUI 的具体方法，请参考“使用管理系统”章节中的“WebUI 连接”部分的相关介绍。

5.5.5 更改主机名

如果使用了集群技术，那么在一个局部的网络中可能存在多台设备。系统允许为设备分配新的名称。使得使用者更加方便的监控每个设备，了解其性能及配置细节。一旦给设备重新命名，命令行提示符上默认的“AN”将变成新分配的名称。要想重新命名设备名称，可以执行下面的命令：

```
Demo(config)#hostname SJ-Box1
SJ-Box1(config)#
```

5.5.6 保存配置

想要保存配置，请执行下面的命令：

```
SJ-Box1(config)#write memory
```

你修改的配置已经被保存在文件中，这个文档在重新启动机器时会运行。

5.5.7 覆盖配置

对相同虚拟服务或后台服务重复执行命令时，要实现配置覆盖的效果，请执行以下命令：

```
Demo(config)#system command override on
```

该功能默认为启用。

如果需要禁用命令覆盖功能，请执行以下命令：

```
Demo(config)#system command override off
```

第6章 高级网络配置

6.1 概述

这章将介绍通过命令行对设备的 VLAN、MNET、端口转发、NAT、动态路由和 IP 地址池进行配置。

注意：IP 地址、默认路由等基本网络配置和高级网络配置通常被其它配置关联或使用。当完成这些配置后，请勿任意修改或者删除，否则将会造成不可预知的问题。如果管理员需要修改网络配置，请在修改网络配置后依次执行命令“**write memory/file**”、命令“**clear config all**”和命令“**config memory/file**”。

6.1.1 VLAN

虚拟局域网（Virtual Local Area Network，VLAN）用于在逻辑上将网络按照功能或应用划分为更小的网络，VLAN 的应用无需考虑用户网络的物理结构。每一个 VLAN 都是一个逻辑上独立的网络。以太网的 VLAN 分为以下两种：

- 以端口划分的 VLAN（Port-based VLAN）

基于交换机端口划分的 VLAN，这种 VLAN 配置简单，但通常仅限于单个交换机的环境下使用。

- 带标签的 VLAN（Tag-based VLAN）

基于标签划分的 VLAN 可以使一组来自不同物理网段的设备，相互间像在同一物理网段中一样通信。在基于标签划分的 VLAN 中，将一个称为“VLAN ID”或“卷标”的数字插入到以太网帧中，以便让网络设备依据这些信息来决定数据帧的转发。这种加入了标签的帧比普通的帧长度长四个字节，其中包含两个字节的 VLAN 协议标识符（TPID）和两个字节的 VLAN 控制信息（VCI）。

设备在它所有的接口上都支持带卷标的 VLAN。带标记的 VLAN 在以太网帧头中增加一个标记，以便交换机和路由器利用这些信息来转发数据报。

设备的 VLAN 功能支持 IPv4 和 IPv6 网络环境。可以使用“**show interface**” 命令来查看当前系统上所有基于 IPv4 和 IPv6 的 VLAN 接口配置。例如：

```
Demo(config)#show interface
……
V1(vlan1): flags=8843<UP,BROADCAST,RUNNING,SIMPLEX> mtu 1500
        inet6 fe80::230:48ff:fe93:a73e prefixlen 64 scopeid 0xb
        ether 00:30:48:93:a7:41
        media: autoselect
        status: no carrier
```

```
        vlan : 10 parent interface: port2
        webwall status: OFF
        packet drop (not permit): 0
                tcp 0           udp 0           icmp 0          ah 0            esp 0
        packet drop (deny): 0
                tcp 0           udp 0           icmp 0          ah 0            esp 0
        5 minute input rate 0 bits/sec, 0 packets/sec
        5 minute output rate 0 bits/sec, 0 packets/sec

V2(vlan2): flags=8843<UP,BROADCAST,RUNNING,SIMPLEX> mtu 1500
        inet6 fe80::230:48ff:fe93:a73e prefixlen 64 scopeid 0xc
        ether 00:30:48:93:a7:41
        media: autoselect
        status: no carrier
        vlan : 20 parent interface: port2
        webwall status: OFF
        packet drop (not permit): 0
                tcp 0           udp 0           icmp 0          ah 0            esp 0
        packet drop (deny): 0
                tcp 0           udp 0           icmp 0          ah 0            esp 0
        5 minute input rate 0 bits/sec, 0 packets/sec
        5 minute output rate 0 bits/sec, 0 packets/sec
```

6.1.2 MNET

多网段（Multi-Netting，MNET）用于在同一个物理接口上复用多个 IP 地址，在分配 IP 地址的时候，网络管理员可以选择下面这种方式：为这台服务器配置一个 IP 地址，等到新的服务器到位后再分别配置新的 IP 地址。

也可以采用另一种 IP 地址的分配方案。管理员可以将四个 IP 地址一次性分配给当前使用的那台服务器，每个 IP 地址正是未来新服务器要使用的 IP 地址。网络管理员现在就可以为未来的服务器先行建立 DNS 解析条目。这种为了给一个网络设备接口配置多个 IP 地址的过程，称为 MNET。

设备的 MNET 功能支持 IPv4 和 IPv6 网络环境。

6.1.3 端口转发

在设备上，在一个 IP 地址/端口对上的任意数据流都可以通过端口转发来转发，来自外部客户端的数据可以直接访问内部网络中的服务器。

注意：Port Forwarding 功能不支持 FTP 协议，在这种情况下，建议用户使用 SLB 功能。

举例来说，管理员在设备的 Port2 接口所连接的内网中的一台服务器上运行 SSH，想从外网的客户端建立连接。为了从外网获取内网 IP/端口对的信息，管理员需要在设备上启用端口转发。如下图所示。

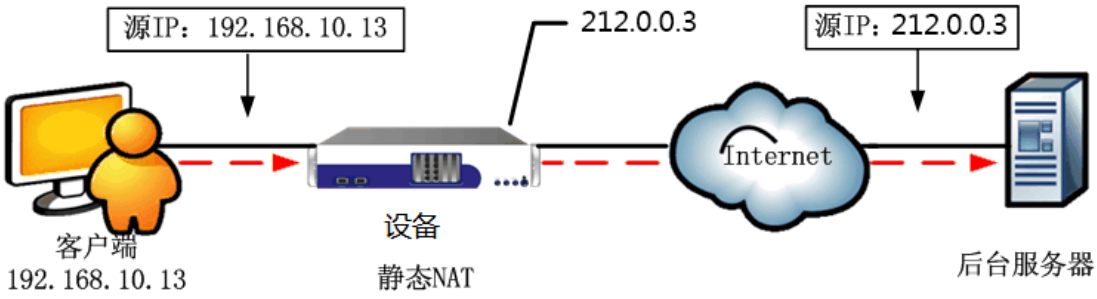

图6–1 端口转发

6.1.4 NAT

网络地址转换（Network Address Translation，NAT）将一个网络中的 IP 地址转换为另一个网络中的另一个 IP 地址。其中一个网络是内部网络，另一个是外部网络。当你想让内网中的后台服务器向外网开放，就需要使用 NAT。使用 NAT 后，从后台服务器的角度来看，所有的数据报都来自设备。

6.1.4.1 NAT 工作机制

当一台内网客户端向外网设备发起请求时，将不可避免地向那台设备发送 IP 数据报。这些数据报包含所有它们到达目的地所需要的信息。

当这些数据报通过 NAT 网关，它们将被重建为看起来像是从 NAT 网关发出的一样。NAT 网关将这些改变记录在自己的状态表中，以便这些数据报返回的时候能够得到反向重建，数据报通过防火墙的时候也不会被拒绝。

NAT 支持 PPTP 隧道穿越

设备的 NAT 功能支持 PPTP（Point-to-Point Tunneling Protocol，点到点隧道协议）隧道穿越。

PPTP 协议是一个在中间网络之间传递 PPP（Point-to-Point Protocol，点到点协议）帧的隧道机制。PPTP 协议使用 GRE 封装 PPP 数据报。一般情况下，GRE 隧道不能从 NAT 设备（设备）中穿越。为了使 NAT 功能支持 PPTP 隧道穿越，设备中使用了 PPTP NAT 编辑器。该编辑器是一个安装在 NAT 设备上的软件组件，可以实现 IP 地址、端口和 call ID 的转换服务。

6.1.4.2 设备支持四种 NAT 类型

- **静态 NAT**

一对一的进行 IP 地址转换。通过配置静态 NAT，设备将内网的一个真实 IP 转换成外网的一个虚拟 IP。对于直接来自外网虚拟 IP 的通信，数据报将被直接转发到所对应的内部真实 IP，这些数据报在转发的过程中不会改变端口号等信息。相对的，内网中客户端也可以直接通过虚拟 IP 访问外网端口。来自内网客户端的数据将使用外网虚拟 IP 作为源 IP 地址。数据报的端口号和协议信息不会被更改。静态地址转换支持 TCP、UDP 和 ICMP 协议。

在下面的例子中，使用静态 NAT，客户端的 IP 地址 192.168.10.13 将会被转换为 212.0.0.3。如下图所示。

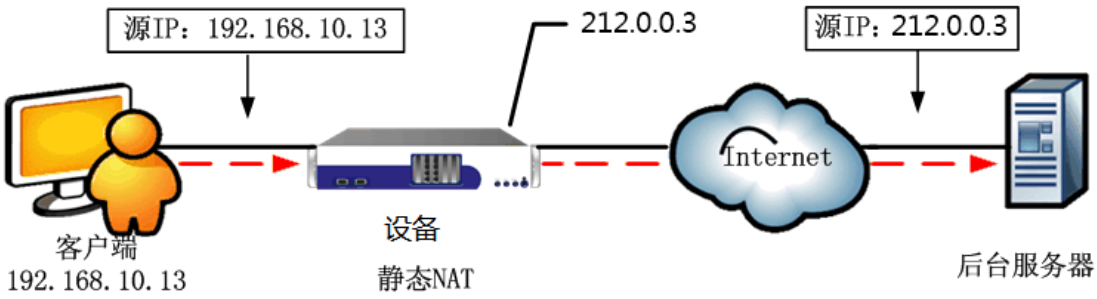

图6–2 静态 NAT

- **基于网络地址端口的 NAT 转换**

将多个内部真实 IP 地址通过多个不同的端口号绑定到一个单独的虚拟 IP 地址上。通过配置网络地址端口转换 NAT，一组内网客户端都可以直接通过设备的 Port1 接口连接到外网。

在下面的例子中，客户端的 IP 地址范围从 192.168.10.11 到 192.168.10.13 之间，这些地址都通过不同的端口号被转换成了 212.0.0.3 这个外网地址。如下图所示。

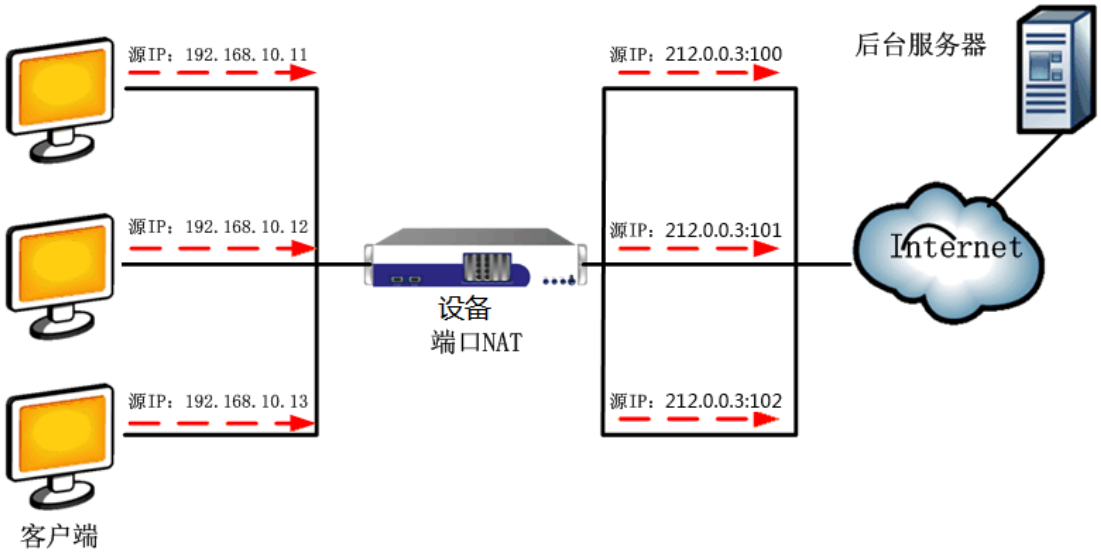

图6–3 基于网络地址端口的 NAT 转换

如果一个后台服务器的 IP 地址在网络地址端口转换 NAT 配置的网段中，静态 NAT 享有优先处理这个 IP 数据报的权利。一个已经被静态 NAT 使用的 VIP 不能被用在网络地址端口转换 NAT 的配置中，也不能被用在其它的静态 NAT 配置中。

- **基于地址池的动态 NAT 转换**

一对多或多对多的动态地址转换，将多个内部真实 IP 地址绑定到一个包含多个 IP 地址的地址池上，通过配置基于地址池的 NAT 转换，一组客户端的 IP 地址可以被转换成不同的 IP 地址连接到外网。

在下面的例子中，客户端的 IP 地址范围从 192.168.10.11 到 192.168.10.13 之间，这些地址都通过不同的端口号被转换成了地址池中 212.0.0.1 到 212.0.0.3 这个地址段，连接到外网。如下图所示。

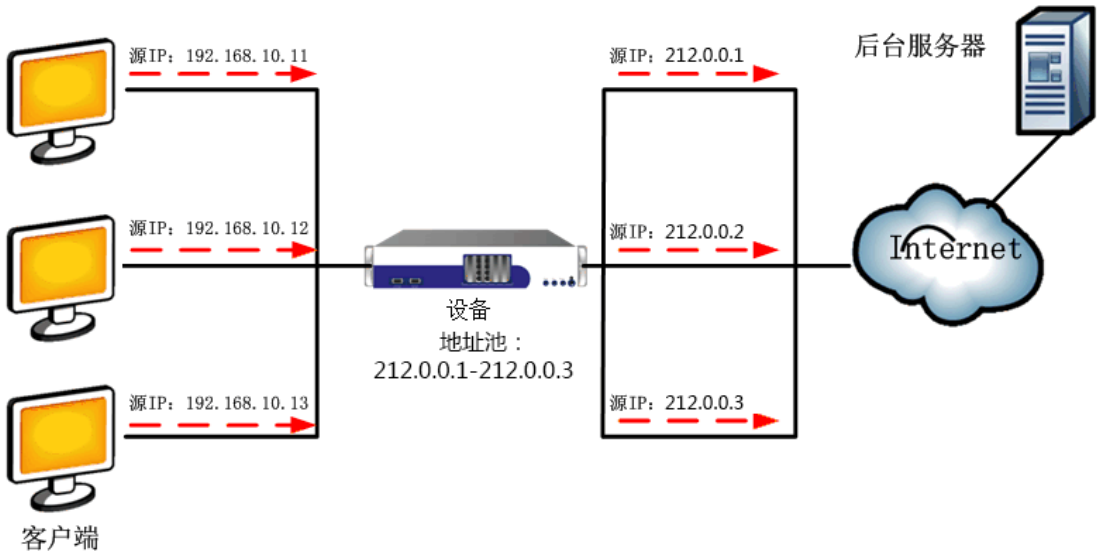

图6–4 基于地址池的动态 NAT 转换

- **基于目的 IP 的地址转换**

设备支持基于目的 IP 的动态网络地址转换。这种地址转换方式要求数据报的目的 IP 地址在指定的网段或与路由网关处在相同的网段。若网关设为预设的 0.0.0.0，则需要进行转换的目的 IP 地址或 IP 地址池应与设定的路由网关处于同一网段中。

注意：对于 IPv6 地址，只有 TCP 和 UDP 类型的数据报可以进行转换且不支持配置网关地址。

配置示例

5. 配置 IP 地址池。

```
Demo(config)#ip pool "pool1" 212.16.72.45 212.16.72.55
Demo(config)#ip pool "pool3" 3ffd::45 3ffd::46
```

6. 配置基于目的 IP 的 NAT。

```
Demo(config)#nat portdst "pool1" 212.16.72.0 255.255.255.0 500 212.16.72.101
Demo(config)#nat portdst "pool3" 3ff1:: 65 50
```

查看 NAT 配置信息与统计

以下命令用于查看 NAT 配置与统计信息。

- 查看 NAT 表

```
Demo(config)#show nat table
From 212.16.73.108(5208) through 212.16.72.53(50776) to 212.16.72.77(80)
From 212.16.73.108(5200) through 212.16.72.53(50768) to 212.16.72.77(80)
From 212.16.73.108(5880) through 212.16.72.53(51448) to 212.16.72.77(80)
From 212.16.73.108(6008) through 212.16.72.53(51576) to 212.16.72.77(80)
From 3ffd::108(42720) through 3ffd (45672) to 3ff1::77(80)
From 3ffd::108(42728) through 3ffd (45680) to 3ff1::77(80)
From 3ffd::108(43336) through 3ffd (46288) to 3ff1::77(80)
From 3ffd::108(43376) through 3ffd (46328) to 3ff1::77(80)
From 3ffd::108(43464) through 3ffd (46416) to 3ff1::77(80)
From 3ffd::108(43408) through 3ffd (46360) to 3ff1::77(80)
From 3ffd::108(43840) through 3ffd (46792) to 3ff1::77(80)
From 3ffd::108(43896) through 3ffd (46848) to 3ff1::77(80)
```

- 查看 NAT 统计信息。

```
Demo(config)#show statistics nat
nat portdst "pool1" 212.16.72.0 255.255.255.0 500 212.16.72.101
        protocol(total/current):icmp(1/1) udp(0/0) tcp(23724/251)
nat portdst "pool3" 3ff1:: 65 50
        protocol(total/current):icmp(0/0) udp(0/0) tcp(23716/246)
```

6.1.5 网站分类

网站分类是设备通过订阅 Webroot BrightCloud 的网站分类服务提供的一种动态识别网站类别的功能。该功能允许设备通过本地缓存、本地数据库和在线连接到 Webroot 服务器来查询用户访问的网站类别。如需使用在线查询功能，需确保配置了网站分类功能的设备能够访问外网。

如需启用网站分类功能，需导入许可证。该功能支持两种类型的许可证：

- 试用许可证：30 天免费试用。

- 正式许可证：有效期为 365 天。

许可证到期后，网站分类功能将不可用，Webroot 服务器会拒绝设备的访问，设备也会停止向 Webroot 服务器发送网站类别查询请求。如需获取网站分类功能的使用许可证，请联系公司技术支持并提供设备的型号和序列号信息。

6.1.6 DNS NAT

DNS NAT 功能应用于内网 DNS 服务器返回的 DNS 响应中的解析 IP 地址为不能从外部访问的内网 IP 地址的场景。启用 DNS NAT 功能后，当收到来自内网 DNS 服务器的 DNS 响应时，设备将根据静态 NAT 条目（通过命令“**nat static**”配置），将 DNS 响应中的内网解析 IP 地址转换为公网 IP 地址，然后将改写后的 DNS 响应返回给客户端。当外网用户使用公网 IP 地址访问时，系统根据静态 NAT 条目再将公网 IP 地址转换为内网服务器的 IP 地址，这样外网用户就可以访问内网服务器。默认情况下，该功能是禁用的。

6.1.7 动态路由

动态路由是一种使路由器可以自动决定和选择数据报转发路径的协议。动态路由比静态路由更为强壮。动态路由目前支持很多协议，包括 RIPv1、RIPv2、OSPFv2 和 OSPFv3 及 BGP（含多协议扩展）。

动态路由特别适合当今这种庞大的、频繁发生变化的网络环境。它通过自动收敛来适应网络拓扑的改变。并且动态路由协议会在网络中的路由器之间相互传递路由信息，来决定转发数据报的最佳路径。

6.1.8 IP 地址池

IP 地址池包含了来自同一网段的多个 IP 地址，当管理员配置 NAT 地址转换和 SLB 服务器负载均衡的时候，可以使用预先配置的 IP 地址池，以满足网络环境的需求。

对于 NAT 模块，IP 地址池可以配置在出向接口上，实现对多个出向 IP 地址的 NAT 转换。这样不仅可以提高设备的并发能力，同时可以更加充分的利用 IP 地址资源。

对于 SLB 模块，可以为后台服务组配置 IP 地址池。在 SLB 反向代理模式中，当选择了不同的后台服务组，设备可以选择不同的 IP 地址中的 IP 地址来连接到后台服务器。这样不仅提高了设备的并发连接能力，也为管理员提供了更加简便的配置方法。

设备可以支持多个 IP 地址池，每个地址池最多可以包含 256 个 IP 地址。根据设备内存的不同，所支持的 IP 地址池个数也不相同，具体如下表所示：

表6–1 最大 IP 地址池个数

系统内存	最大 IP 地址池数
4GB	32
8GB	64
16GB	128
32GB	256

IP 地址池中支持配置 IPv4 或 IPv6 地址。

设备最多支持 32 个 IP 地址池，每个地址池最多可以包含 256 个 IP 地址。

在配置 IP 地址池的时候需要注意：

- 每个 IP 地址池的名称必须是唯一的。
- 一个 IP 地址池中的 IP 地址必须在同一网段内。
- 多个 IP 地址池可以使用同一网段内的 IP 地址。
- 同一个 IP 地址可以同时添加到多个 IP 地址池中。
- IP 地址段必须由连续的 IP 地址组成，一个 IP 地址池可以添加多个的 IP 地址段。
- IP 地址池中的 IP 地址必须是合法 IP 地址：
  - 设备的接口 IP 地址所在的网段以外的 IP 地址将被视为非法。
  - 广播地址将被视为非法。
  - 主机位为 0 的 IP 地址将被视为非法。

6.1.9 VXLAN

设备支持 Virtual eXtensible Local Area Network（虚拟扩展局域网，VXLAN）功能。VXLAN 采用 MAC-over-UDP 的报文封装模式，将二层报文用三层协议进行封装，可实现二层网络在三层范围内的扩展，满足了数据中心大二层虚拟迁移的需求。VXLAN 采用 24 位的 VXLAN Network Identifier（VXLAN 网络标识符，VNI）来标识不同的 VXLAN 网络，支持多达 1600 万的租户隔离，解决了云计算中海量租户隔离的问题。

6.1.9.1 基本概念

VXLAN 网络中的基本概念包括：

- VNI：VXLAN 网络使用 VXLAN Network Identifier（VXLAN 网络标识符，VNI）来标识不同的 VXLAN 网络。属于不同 VNI 的虚拟机之间不能直接进行二层通信。

- 隧道：VXLAN 隧道是一条建立在 Underlay 网络上的虚拟通道，用来传输经过封装的 VXLAN 报文。
- VTEP：VXLAN Tunnel End Point（VXLAN 隧道端点，VTEP）是 VXLAN 隧道的起点和终点，负责原始以太网报文的 VXLAN 封装和解封装。VXLAN 隧道一端的 VTEP 将二层数据报文封装，然后发送到 VXLAN 隧道，另一端的 VTEP 接收到封装的数据报文后，对数据报文解封装。VTEP 可以是任何支持 VXLAN 的设备。设备作为 VTEP，既可以作为三层网关，也可以作为二层网关。

6.1.9.2 VXLAN 工作模式

VXLAN 隧道可以传输单播、广播和多播数据报文，支持两种工作模式：

- P2MP：在该工作模式下，在一个 VXLAN 中，管理员需要手动为设备与所有的 VXLAN 隧道端点（VXLAN Tunnel End Point，VTEP）建立隧道。
- Multicast：在该工作模式下，在一个 VXLAN 中，管理员只需要为设备建立一个到该 VXLAN 的多播隧道。

此外，VXLAN 隧道同时支持 IPv4 和 IPv6，可以在 IPv4 隧道传输 IPv6 报文，也可以在 IPv6 隧道传输 IPv4 报文。

6.1.9.3 VXLAN 学习

设备支持 VXLAN 学习功能。当启用 VXLAN 学习功能后，设备将自动学习 VTEP IP 地址和 MAC 地址间的映射关系，建立转发表。自动学习的转发表项的生存时间为 20 分钟。

6.1.10 物理接口关闭

设备支持手动关闭指定的物理接口。关闭指定物理接口后，其配置权限等信息将发生变化（详情参见命令行手册中“interface shutdown”命令）

手动关闭物理接口后，管理员可通过“**show interface**”命令查看到接口状态信息显示为“status: down (Administratively down)”。

- **配置示例**

执行以下命令关闭指定物理接口。

```
Demo(config)#interface shutdown port2
```

执行以下命令取消关闭指定物理接口。

```
Demo(config)#no interface shutdown port2
```


注意：手动关闭物理接口后，接口状态将显示为“Administratively down”。此时，系统已不再检测接口状态。如果继续拔掉接口线，接口状态依旧为“Administratively down”。

6.2 高级网络配置示例

为了更好的通过您的配置来发挥设备的最佳性能，请根据您网络的具体情况，绘制出指导配置的网络拓扑图。本节中的高级网络配置示例将基于下图。

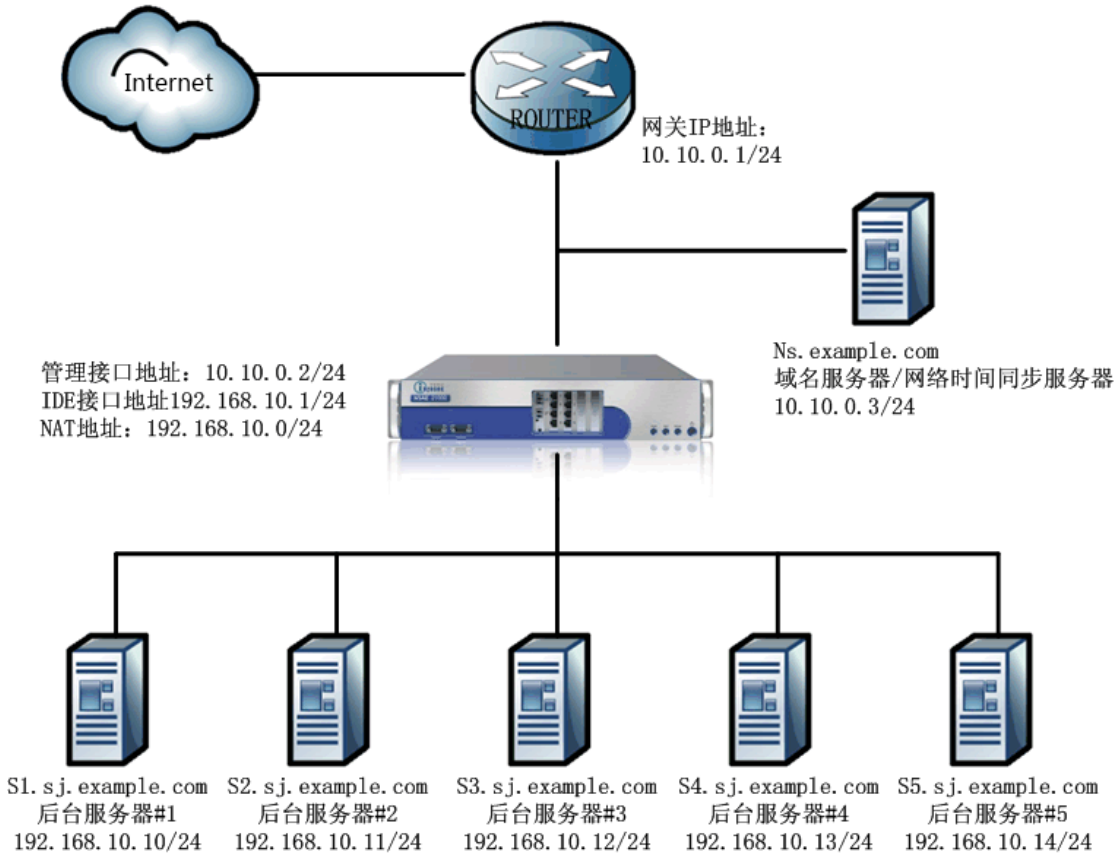

图6–5 高级网络拓扑

下表列出了以上网络拓扑图中的一些关键网络参数。

表6–2 高级网络配置

IP 地址	描述
10.10.0.1/24	网关 IP 地址
10.10.0.2/24	管理 IP 地址
192.168.10.1/24	Port2 接口地址
192.168.10.0/24	NAT 映像地址
192.168.10.10	后台服务器#1 地址
192.168.10.11	后台服务器#2 地址
192.168.10.12	后台服务器#3 地址

192.168.10.13	后台服务器#4 地址
192.168.10.14	后台服务器#5 地址
10.10.0.3	域名服务器/网络时间同步服务器地址

下表列出了 VLAN、MNET、端口转发、NAT 和动态路由的配置命令，相关命令的描述信息，请参考命令行使用手册。

表6–3 设备高级网络配置命令

配置操作	命令行
配置 VLAN	**vlan** *{system_ifname|bond_ifname}* *<user_interface_name>* *<vlan_tag>*
配置 MNET	**mnet** *{system_ifname|bond_ifname|vlan_ifname}* *<user_interface_name>*
配置端口转发	**fwd tcp** *<local_ip>* *<local_port>* *<remote_ip>* *<remote_port>* *[timeout]* **fwd udp** *<local_ip>* *<local_port>* *<remote_ip>* *<remote_port>* *[timeout]*
配置 NAT	**nat port** *{pool_name|vip}* *<source_ip>* *{netmask|prefix}* *[timeout]* *[gateway]* *[description]* **nat static** *<vip>* *<network_ip>* *[timeout]* *[gateway]* *[description]*
配置动态路由	**rip {on|off}** **rip network** *<ip_address>* *<netmask>* **ospf {on|off}** **ospf network** *<ip_address>* *<netmask>* *<area_id>*
配置 IP 地址池配置	**ip pool** *<pool_name>* *<start_ip>* *[end_ip]* **slb proxyip global** *<pool_name>* **slb proxyip group** *<group_name>* *<pool_name>*

6.2.1 VLAN 配置

在我们的例子里，我们将建立两个 VLAN：“inside-vlan 1”和“inside-vlan 2”。其中“inside-vlan 1”使用“500”这个标签，“inside-vlan 2”使用“3001”这个标签。这些卷标已经插入到以太网帧中。

1. 使用“**vlan**”命令创建 VLAN。

```
Demo(config)#vlan port2 inside-vlan1 500
Demo(config)#vlan port2 inside-vlan2 3001
```

7. 使用“**ip address**”命令为每个 VLAN 分配 IP 地址。

```
Demo(config)#ip address inside-vlan1 192.168.1.1 255.255.255.0
Demo(config)#ip address inside-vlan2 192.168.2.1 255.255.255.0
```

对于配置了 VLAN 的接口，其所连接的交换机或者路由器需要打开 Tagged VLAN 或者 Trunking 功能。

6.2.2 MNET 配置

在接口上配置 MNET 和配置 VLAN 非常类似。在我们的例子里，要在设备的 Port2 接口上配置两个 IP 地址：192.168.1.1/24 和 192.168.2.1/24。

1. 使用“**mnet**”命令为接口分配 MNET。

```
Demo(config)#mnet port2 mnet1
Demo(config)#mnet port2 mnet2
```

8. 使用“**ip address**”命令为每个 MNET 分配 IP 地址。

```
Demo(config)#ip address mnet1 192.168.1.1 255.255.255.0
Demo(config)#ip address mnet2 192.168.2.1 255.255.255.0
```

此外我们有必要参考一下与 Port2 接口相连的交换机或者路由器的文档，查阅如何在它们的接口上使用 MNET。

6.2.3 端口转发配置

在我们的例子中，将配置 TCP 协议使用端口转发。

```
Demo(config)#fwd tcp 10.10.0.2 4000 192.168.10.10 22 300
```

在设备上，端口转发应该使用一个高端端口，而不是低于 1024 的端口，因为其它的服务可能需要利用这些端口来进行监听，比如 443（SSL）和 80（HTTP）。而在内部的后台服务器上，可以选择一个低于 1024 的端口作为服务端口。使用“**fwd tcp**”命令所对应的“**show**”、“**no**”或“**clear**”命令版本可以查看和删除端口转发配置。

6.2.4 NAT 配置

在我们的配置示例当中，将使用以下命令：

```
Demo(config)#nat port 10.10.0.2 192.168.10.0 255.255.255.0 60 10.10.0.1
```

这条命令在 192.168.10.0/24 这个网络上使用了 NAT。在我们的例子里，虚拟 IP 地址 10.10.0.2 和网关地址 10.10.0.1 在同一个网段中。在“**nat port**”命令中的参数“**gateway**”，可以被配置为默认的 0.0.0.0。如果虚拟 IP 地址和网关地址不在同一网段中，那么“**nat port**”命令中的参数“**gateway**”比如设置为路由器网关的地址。

我们可以通过改变网络屏蔽，来控制指定内部网段访问外部网络。举个例子，下面的命令只允许在 192.168.10.0 至 192.168.10.128 之间的 IP 地址访问外部网络。

```
Demo(config)#nat port 10.10.0.2 192.168.10.0 255.255.255.128 60 0.0.0.0
```

如果我们想允许其余的 IP 地址（192.168.10.129-192.168.10.254）访问外部网络，需要做以下配置。

```
Demo(config)#nat port 10.10.0.2 192.168.10.129 255.255.255.128 60 0.0.0.0
```

如果我们想允许某个真实 IP 地址访问外部网络，需要配置静态 NAT。

```
Demo(config)#nat static 10.10.0.2 192.168.10.12
```

6.2.5 DNS NAT

➢ **配置目标**

在下面场景中，用户发送 DNS 解析请求“www.abc.com”。设备用于链路负载均衡（Link Load Balance，LLB）。

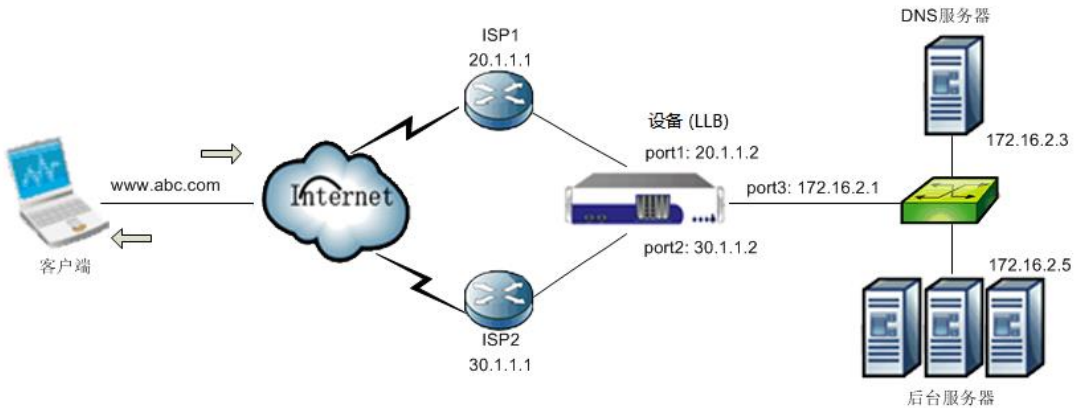

图6–6 DNS NAT 场景

配置目标如下：

- DNS NAT 功能启用后，公网 IP 地址“20.1.1.5”或“30.1.1.5”作为 DNS 响应中的解析 IP 地址返回到客户端。
- 外网用户可以使用返回的公网 IP 地址正常访问内网服务器“172.16.2.5”。

➢ **配置步骤**

1. 配置静态 NAT 条目。

```
Demo(config)#nat static 20.1.1.3 172.16.2.3
Demo(config)#nat static 30.1.1.3 172.16.2.3
Demo(config)#nat static 20.1.1.5 172.16.2.5
Demo(config)#nat static 30.1.1.5 172.16.2.5
```

2. 配置 LLB 链路。

```
Demo(config)#llb link route ISP1 20.1.1.1
Demo(config)#llb link route ISP2 30.1.1.1
```

3. 配置 LLB 链路健康检查。

```
Demo(config)#llb link health checker icmp ISP1 “20.1.1.1” 10 5 3 3
Demo(config)#llb link health checker icmp ISP2 “30.1.1.1” 10 5 3 3
```

4. 启用 DNS NAT 功能。

```
Demo(config)#nat protocol dns
```

注意：在非 LLB 场景中，只需要配置步骤 1 和 4。

6.2.6 动态路由配置

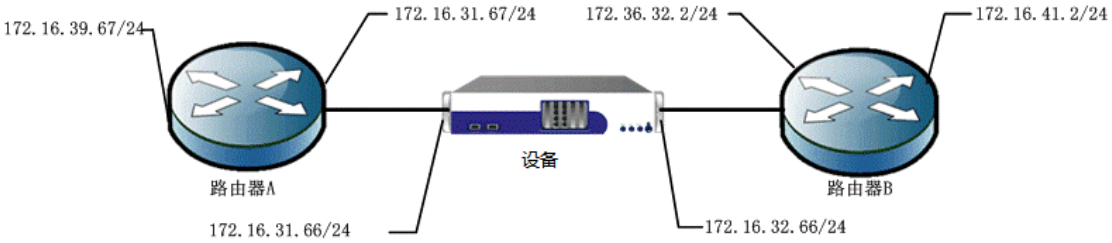

图6–7 动态路由配置

1. 配置 RIP 协议。

```
Demo(config)#rip on
Demo(config)#rip version 2
Demo(config)#rip network 172.16.31.0 255.255.255.0
Demo(config)#rip network 172.16.32.0 255.255.255.0
```

2. 配置 OSPF 协议。

```
Demo(config)#ospf on
Demo(config)#ospf network 172.16.32.0 255.255.255.0 0
Demo(config)#ospf network 172.16.31.0 255.255.255.0 0
```

在配置结束后，可以通过“**show ip route**”命令查看动态路由的配置。

```
Demo(config)#show ip route
Destination       Netmask           Gateway
RIP routes:
Destination       Netmask           Gateway
172.16.39.0       255.255.255.0     172.16.31.67
OSPF routes:
Destination       Netmask           Gateway
172.16.41.0       255.255.255.0     172.16.32.2
```

现在，关于设备的基本网络配置的介绍结束。具体的配置方法，请根据实际网络拓扑，结合上面的示例进行配置。

6.2.7 IP 地址池配置示例

NAT 地址池配置

在我们的例子中，将为 NAT 配置 IP 地址池。

1. 使用“**ip pool**”命令创建 IP 地址池。

```
Demo(config)#ip pool "pool1" 124.0.0.22 124.0.0.22
Demo(config)#ip pool "pool2" 124.0.1.22 124.0.1.22
```

2. 使用“nat port”命令将创建好的 IP 地址池作为 NAT 的地址。

```
Demo(config)#nat port "pool1" 1.1.1.0 255.255.255.0 60 124.0.0.125
Demo(config)#nat port "pool2" 1.1.1.0 255.255.255.0 60 124.0.1.125
```

SLB 地址池配置

在我们的例子中，将为 SLB 功能配置 IP 地址池。

1. 使用“**ip pool**”命令创建 IP 地址池。

```
Demo(config)#ip pool "pool2" 2.2.2.150 2.2.2.180
Demo(config)#ip pool "pool1" 2.2.2.100 2.2.2.150
```

2. 将创建好的 IP 地址池设置为 SLB 的全局 IP 地址池。

```
Demo(config)#slb proxyip global "pool2"
```

3. 为后台服务组设置可用的 IP 地址池。

```
Demo(config)#slb proxyip group "gpi" "pool1"
```

注意：后台服务组的 IP 地址池的优先级高于全局 IP 地址池。

6.2.8 VXLAN 配置示例

6.2.8.1 设备作为 VXLAN 三层网关

在下面的图中，位于 192.10.10.100/24 子网中的客户端需要与虚拟机 VM1 和 VM2 通信，设备作为 VXLAN 三层网关。

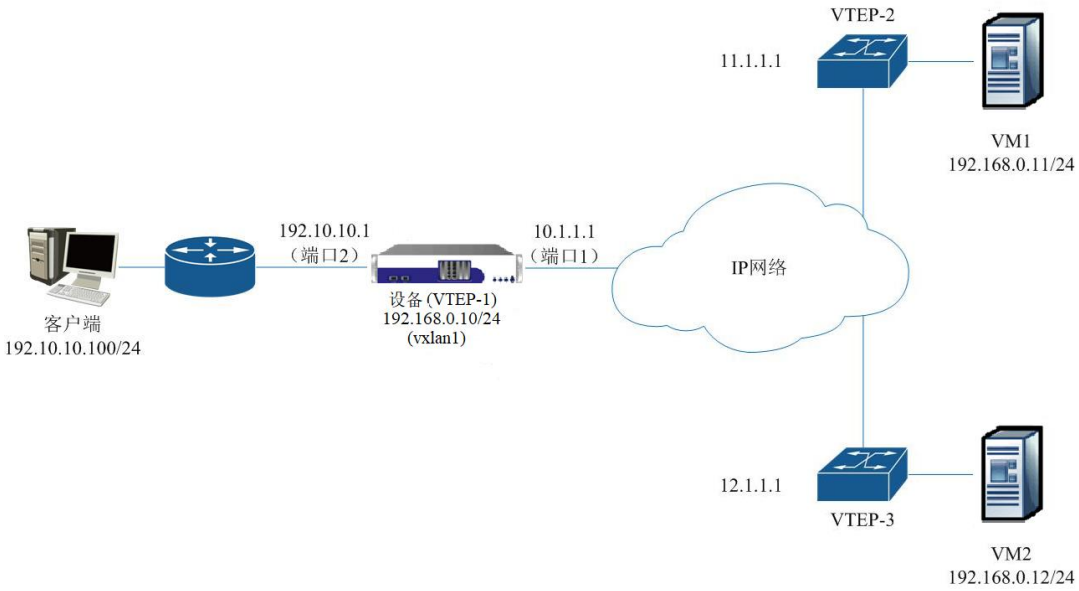

图6–8 VXLAN 三层网关

客户端与虚拟机的通信流程如下：

1. 客户端发送数据报文。
2. 设备收到数据报文后，查询路由表发现需要把数据报文转发到 VXLAN 网络中的一台设备，假设是 VM1。设备查询转发表获得 VTEP-2 的 IP 地址，对数据报文进行封装，通过 VXLAN 隧道将数据报文发送到 VTEP-2。
3. VTEP-2 将数据报文转发给 VM1。
4. VTEP-2 收到 VM1 返回的数据报文，通过 VXLAN 隧道转发给设备。
5. 设备对数据报文进行解封装，并把数据报文转发至客户端。同时，如果启动了 VXLAN 学习功能，设备还将学习 VTEP IP 地址和 VM1 MAC 地址间的映射关系，记录在转发表中或者对于已有记录进行刷新。
6. 客户端收到返回的数据报文。

- **CLI**

1. 设置 VXLAN 的工作模式。

```
Demo(config)#vxlan mode p2mp
```

2. 设置 VXLAN 隧道的目的端口号。

```
Demo(config)#vxlan port 4789
```

3. 启用 VXLAN 学习功能。

```
Demo(config)#vxlan learn on
```

4. 创建一个 VXLAN 接口并与指定的 VXLAN 网络关联。

```
Demo(config)#vxlan interface vxlan1 9000
```

5. 为 VXLAN 接口配置 IP 地址。

```
Demo(config)#ip address vxlan1 192.168.0.10 24
```

6. 创建 VXLAN 隧道。

```
Demo(config)#vxlan tunnel tunnel1 10.1.1.1 11.1.1.1
Demo(config)#vxlan tunnel tunnel2 10.1.1.1 12.1.1.1
```

7. 将 VXLAN 隧道绑定到指定的 VXLAN 接口。

```
Demo(config)#vxlan bind vxlan1 tunnel1
Demo(config)#vxlan bind vxlan1 tunnel2
```

8. 启用 VXLAN 功能。

```
Demo(config)#vxlan enable
```

6.2.8.2 设备作为 VXLAN 二层网关

在下面的图中，192.168.0.20/24 子网中的客户端需要与 VM1 和 VM2 通信，设备作为 VXLAN 二层网关。

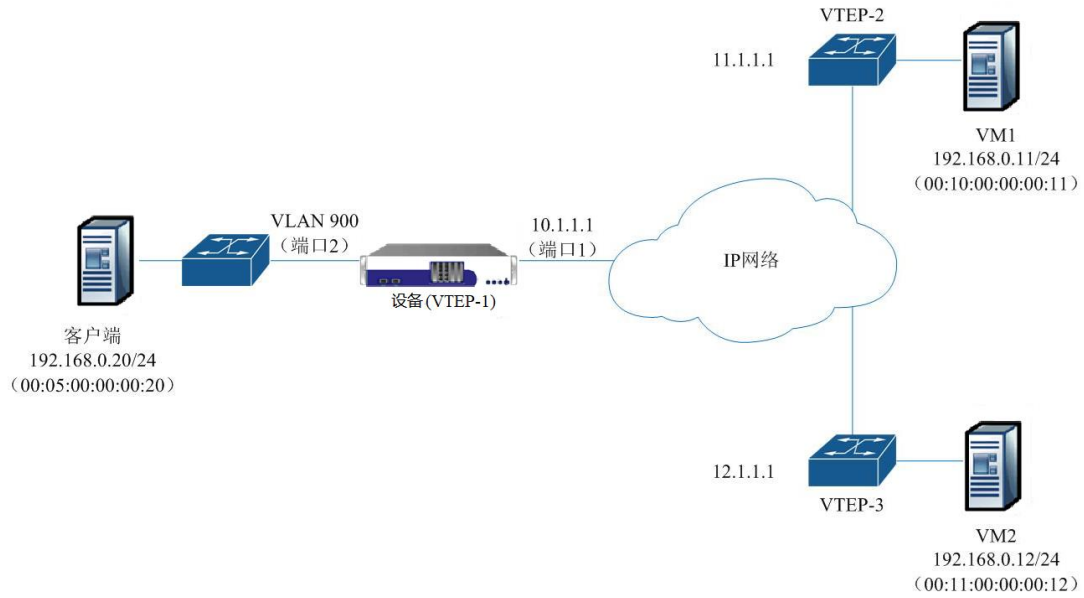

图6–9 VXLAN 二层网关

客户端与虚拟机的通信流程如下：

1. 客户端发送 VLAN 数据报文。
2. 设备收到数据报文后，发现需要将数据报文转发到 VXLAN 网络中的一台设备。假设是 VM1，目的 MAC 为 00:10:00:00:00:11。设备根据目的 MAC 地

址查询转发表，获取 VTEP-2 IP 地址，对数据报文进行封装，通过 VXLAN 隧道将数据报文发送到 VTEP-2。

3. VTEP-2 将数据报文转发给 VM1。
4. VTEP-2 收到 VM1 返回的数据报文，通过 VXLAN 隧道转发给设备。
5. 设备收到 VTEP-2 返回的 VXLAN 数据报文。设备对 VXLAN 数据报文进行解封装，并把其转发到关联的 VLAN 网络。同时，如果启动了 VXLAN 学习功能，设备还将学习 VTEP IP 地址和 VM1 MAC 地址间的映射关系，记录在转发表中或者对于已有记录进行刷新。
6. 客户端收到返回的数据报文。

注意：在第 2 步中：

- 如果收到目的 MAC 地址为多播或广播的数据报文时，设备将该数据报文封装为 VXLAN 报文。在 P2MP 模式下，通过各个 VXLAN 隧道逐一发往各个 VTEP 设备；在 Multicast 模式下，通过多播隧道发送至属于同一多播组的所有 VTEP 设备。
- 如果未查询到匹配的转发表项，设备将该数据报文封装为 VXLAN 报文。在 P2MP 模式下，通过各个 VXLAN 隧道逐一发往各个 VTEP 设备；在 Multicast 模式下，通过多播隧道发送至属于同一多播组的所有 VTEP 设备。

➢ **CLI**

1. 设置 VXLAN 的工作模式。

```
Demo(config)#vxlan mode p2mp
```

2. 设置 VXLAN 隧道的目的端口号。

```
Demo(config)#vxlan port 4789
```

3. 创建一个 VXLAN 接口并与指定的 VXLAN 网络关联。

```
Demo(config)#vxlan interface vxlan1 9000
```

4. 创建 VXLAN 隧道。

```
Demo(config)#vxlan tunnel tunnel1 10.1.1.1 11.1.1.1
Demo(config)#vxlan tunnel tunnel2 10.1.1.1 12.1.1.1
```

5. 将 VXLAN 隧道绑定到指定的 VXLAN 接口。

```
Demo(config)#vxlan bind vxlan1 tunnel1
Demo(config)#vxlan bind vxlan1 tunnel2
```

6. 创建一个 VLAN 接口。

```
Demo(config)#vlan port2 vlan1 900
```

7. 将 VLAN 接口与 VXLAN 接口关联。

```
Demo(config)#vxlan associate vxlan1 vlan1
```

8. 添加转发表项到 VTEP 转发表中。

```
Demo(config)#vxlan forwarding remote 9000 00:10:00:00:00:11 11.1.1.1
Demo(config)#vxlan forwarding remote 9000 00:11:00:00:00:12 12.1.1.1
```

9. 添加一条转发表项到 VXLAN-VLAN 转发表中。

```
Demo(config)#vxlan forwarding local 9000 00:05:00:00:00:20
```

注意：如果启用了 VXLAN 学习功能，无需配置 8 和 9。

10. 启用 VXLAN 功能。

```
Demo(config)#vxlan enable
```

第7章 链路聚合

7.1 概述

本章将介绍链路聚合（Link Aggregation）功能。链路聚合又称骨干路，可以大幅度提高网络的性能和稳定性。

7.2 链路聚合的原理

链路聚合是把两个或多个数据信道捆绑成一个单一的、高带宽的逻辑链路。所有聚合接口被看成是一个“高级”接口。如果每一个聚合链路都是不同的物理链路，它就可以成倍增加网络带宽，提高接口可靠性，提供链路冗余和容错能力。然而，链路聚合不可以与接口冗余功能一起使用。

链路聚合必须在配置 MNET 或 VLAN 之前进行配置。如果在聚合接口上做了其它配置，例如 MNET 或 VLAN，当这个聚合接口上的这些配置发生变化时，管理员必须先保存这些变化，然后重启系统，这样这些配置变化才会起作用。

设备系统最多支持 8 个聚合接口，每个聚合接口上最多可以绑定 12 个系统接口。聚合接口会检查每个系统接口是否正常工作。如果某个接口链路断开，之前由该系统接口处理的流量将转发到聚合接口中其他正常工作的系统接口上。

向聚合接口中添加系统接口时，我们可以将系统接口设置为聚合接口上的主接口或备用接口。每个聚合接口上可以设置多个主接口或备用接口。当所有主接口都无法工作时，备用接口将接替主接口开始工作。

注意：将系统接口绑定到聚合接口时，系统接口上不能配置任何 IP 地址。如果系统接口配有 IP 地址，请先将 IP 配置删除。否则，系统禁止把该接口绑定到聚合接口上。

另外，设备还支持在聚合接口上配置 MNET 或 VLAN。链路聚合的配置必须在配置多网段或虚拟局域网之前进行。

7.3 链路聚合健康检查

链路聚合健康检查用于判断聚合接口的健康状况（Up 或 Down）。链路聚合健康检查允许设备对聚合接口的每个子接口做健康检查，并将每个子接口的健康状况标记为“up”或“down”。通过配置链路聚合健康检查功能，用户可以方便地使用标记为“up”的子接口传输数据。

注意：如果聚合接口中所有子接口的健康状况都标记为“down”而 ARP 数据报和 IPv6 NS 数据报仍可正常发送，此时聚合接口依旧处于可用状态，业务流量将继续发往每个子接口。同时，请检查健康检查的目的 IP 地址是否配置正确。

7.4 链路聚合配置

7.4.1 配置向导

为了更好的通过您的配置来发挥设备的最佳性能，请根据您网络的具体情况，绘制出指导配置的网络拓扑图。本节中的链路聚合配置示例将基于下图。

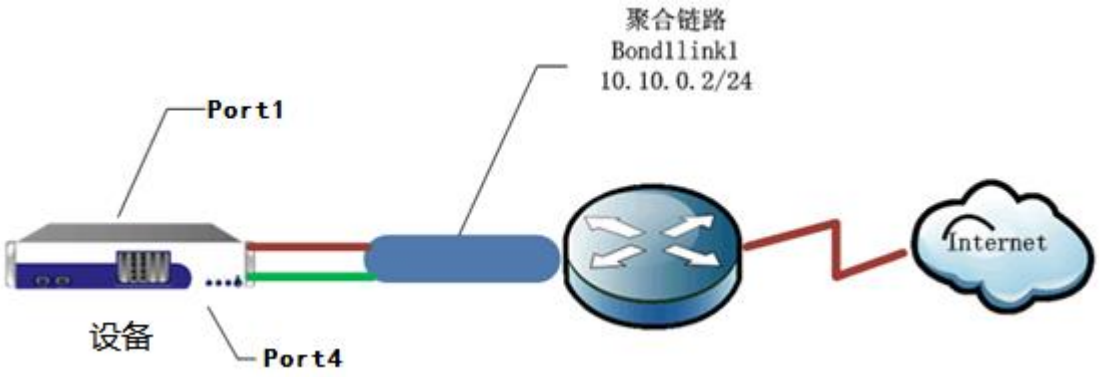

图7–1 链路聚合配置

下面列出了配置链路聚合功能所需要用到的命令，相关命令的描述信息，请参考命令行使用手册。

表7–1 链路聚合配置命令

配置操作	命令行
绑定接口	**bond interface** <*bond_name*> <*interface_name*> *[1|0]*
为绑定后的接口配置一个名称	**bond name** <*bond_id*> <*bond_name*>
为绑定后的接口配置 IP 地址	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname}* <*ip_address*> *{netmask|prefix}*
配置默认网关	**ip route default** <*gateway_ip*>
配置健康检查	**bond health** <*bond_name*> <*destination_ip*> *[interval] [timeout] [up_check_times] [down_check_times] [gateway_ip]*

7.4.2 配置示例

1. 绑定接口。

在我们的例子里，我们需要把设备的 Port1 接口和 Port4 接口绑定在一起，并且将 Port1 接口设置为主接口，将 Port4 接口设置为备用接口。

```
Demo(config)#bond interface bond1 port1 1
Demo(config)#bond interface bond1 port4 0
```

2. 为聚合后的接口分配名称。

我们可以使用“**bond name**”命令为聚合接口配置聚合名称。

```
Demo(config)#bond name bond1 link1
```

3. 为聚合接口分配 IP 地址。

```
Demo(config)#ip address link1 10.10.0.2 255.255.255.0
```

4. 配置网关地址。

```
Demo(config)#ip route default 10.10.0.1
```

为了验证设备已经成功完成了网络基本配置，你可以使用“**ping**”命令测试网关和后台服务器的连通性。如果前面的配置正确，在执行“**ping**”命令后会返回如下信息。

```
Demo(config)#ping 10.10.0.1
PING 10.10.0.1(10.10.0.1): 56 data bytes
64 bytes from 10.10.0.1: icmp_seq=0 ttl=128 time=0.671 ms
64 bytes from 10.10.0.1: icmp_seq=1 ttl=128 time=0.580 ms
64 bytes from 10.10.0.1: icmp_seq=2 ttl=128 time=0.529 ms
64 bytes from 10.10.0.1: icmp_seq=3 ttl=128 time=0.486 ms
64 bytes from 10.10.0.1: icmp_seq=4 ttl=128 time=0.638 ms

--- 10.10.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.486/0.581/0.671/0.068 ms
```

5. 配置并启用链路聚合健康检查。（可选）

```
Demo(config)#bond health link1 172.16.77.81 5 3 3 3 172.16.77.1
```

第8章 集群

8.1 概述

设备集群技术可以使一个局部站点具有高可靠性。同时，通过集群与其他功能的配合使用，还可以实现集群设备之间的负载均衡。

8.2 集群的工作原理

设备集群技术允许两台或更多的设备相互连接组成一台逻辑设备，以便为本地网站提供高可靠性和高可用性。如下图所示。

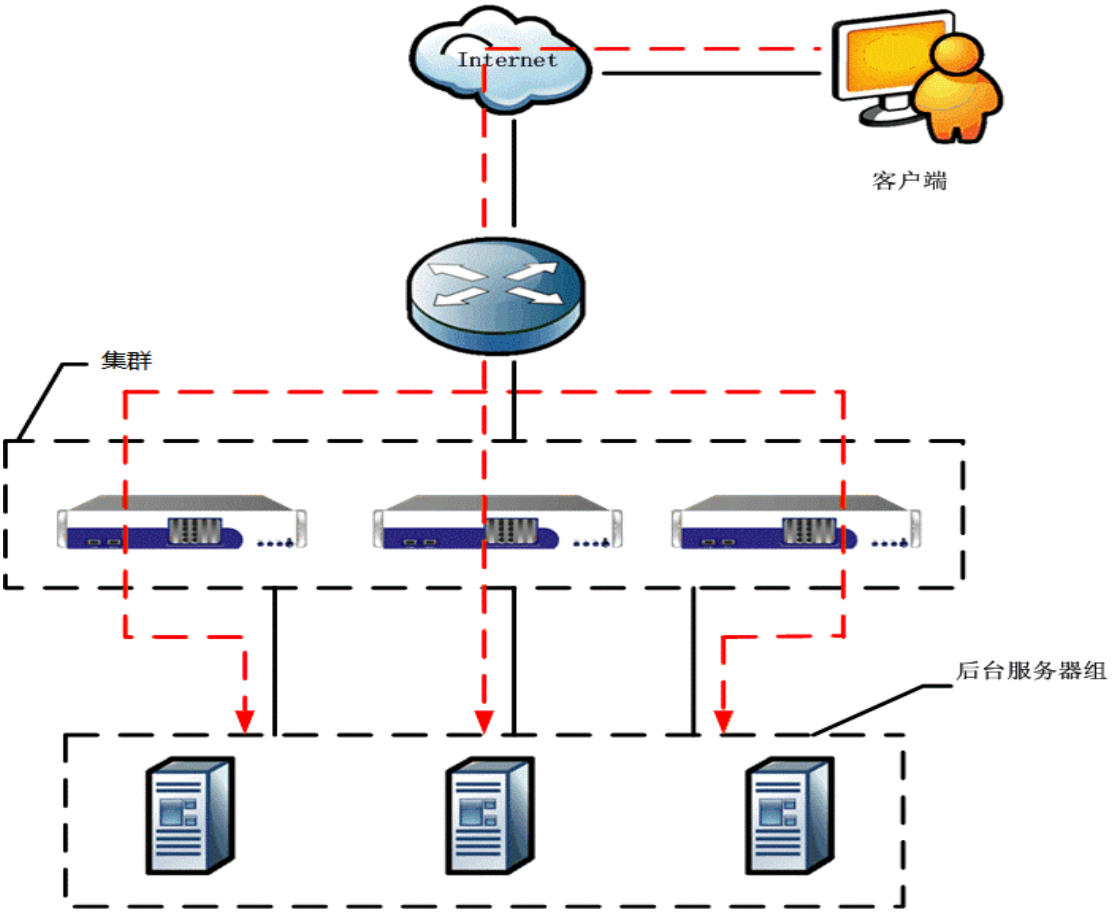

图8–1 集群技术

设备集群技术支持“主备（Active-Standby）”和“主主（Active-Active）”两种模式。

- 主备模式（Active-Standby）

在主备模式下，集群中的一台设备上的所有 VIP 都处于主控状态，这些 VIP 在集群中其它设备上都处于备份状态。该模式支持快速失效切换。

- 主主模式（Active-Active）

在主主模式下，集群中每台设备都有不同的 VIP 或集群 ID 处于主控状态。

8.2.1 快速失效切换模式

快速失效切换（Fast Failover，FFO）功能是通过使用两台设备的 USB 端口（设备主板上的快速失效切换端口）来互相探测对方的状态的。当一台设备出现关机、宕机、重启或者连接断开时，所有的流量就会被立即转到另一台设备上。与一般的集群功能比较，具有快速失效切换的集群功能可用性更高，反应时间更短。而只有“主备模式”的集群支持快速失效切换。如下图所示。

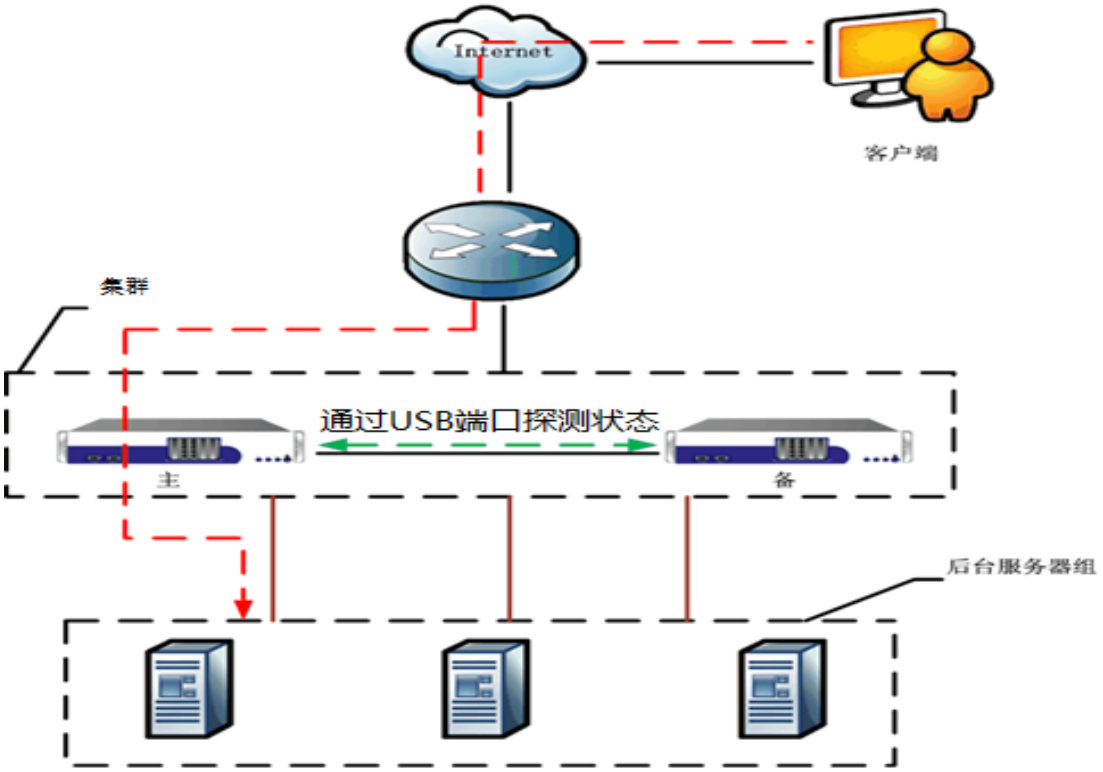

图8–2 设备集群快速失效切换模式

8.2.2 谨慎备份模式

传统的集群环境中，处于主控状态和备份状态的设备通过网络相互传递他们的状态信息。如果备份设备一段时间内没有接收到来自主控设备的 VRRP 报文，就会尝试成为主控设备。但是由于网络的不稳定性，某些意想不到的情况会导致网络中同时出现两个主控设备的情况。

而谨慎备份模式的设计就考虑到了如何避免出现双主控设备的情况。在这种模式中，系统将自动根据心跳在线数据传送的间隔时间来判断是否需要立刻进行主备状态转换。这种模式的应用更加贴近网络的实际情况，VRRP 信息的暂时丢失不会导致双主控设备这种情况的出现。

下面是谨慎备份模式的工作机制：

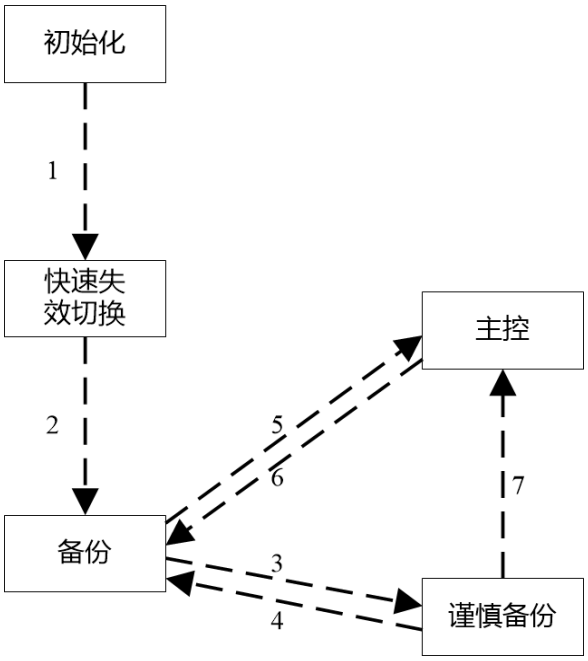

图8–3 谨慎备份模式工作机制

1. 在启用集群后，设备进入初始状态。然后为了检查心跳线的健康情况，处于初始状态的设备将切换到快速失效切换状态。

2. 设备收集心跳在线的健康信息之后，如果心跳线连接状态正常，设备将会切换到备份状态。

注意：如果心跳线连接切断，设备仍然会切换到备份状态，并且集群将会正常运行，尽管在这种情况下谨慎备份模式已经无效了。

3. 如果备份设备接收到一个具有更高优先级的 VRRP 数据报，它会切换到谨慎备份状态。

4. 在以下两种情况下，谨慎备份状态会切换到备份状态：

   - 处于谨慎备份状态的设备接收到一个具有更低优先级的 VRRP 数据报。（在成功进行状态切换之后，备份状态将会切换至主控状态）

   - 处于谨慎备份状态下的设备检查心跳线的健康状况，如果心跳线连接失败，设备将回到备份状态。

5. 在以下两种情况下，备份设备将会切换到主控状态：

- 备份状态下的设备接收到一个具有更低优先级的 VRRP 报文（开启抢占模式时）。
- 在三个连续的广播周期中（默认广播周期为 5 秒，三个广播周期即为 15 秒），备份设备没有从主控设备接收到 VRRP 报文。

6. 如果处于主控状态的设备接收到一个具有更高优先级的 VRRP 数据报，他将切换到备份状态。
7. 如果设备从心跳在线健康状况判断主控设备的网卡已经失效，处于谨慎备份状态的设备将直接切换至主控状态。

注意：所有的集群配置状态都可使用“**show cluster virtual transition**”进行追踪。

默认情况下，谨慎备份模式是关闭的。

若要配置谨慎备份模式，首先必须使用以下命令来启用谨慎备份模式。

```
Demo(config)#cluster virtual ffo on
Demo(config)#cluster virtual discreet on
```

8.2.3 集群支持 IPv6

设备集群功能目前支持 IPv6 类型的 VIP 切换。这样，设备既支持基于 IPv4 类型的 VRRP 数据包，也支持基于 IPv6 类型的 VRRP 数据包。

当在集群相应的接口上同时配置了 IPv4 地址和 IPv6 地址，或者仅配置了 IPv4 地址时，集群将使用 IPv4 类型的 VRRP 数据包在设备之间通信。当在集群相应的接口上仅配置了 IPv6 地址时，集群将使用 IPv6 类型的 VRRP 数据包在设备之间通信。

注意：VRRP 数据包在不同版本的设备之间不兼容，所以在集群中请使用相同版本的系统。

8.3 集群配置示例

8.3.1 服务器负载均衡虚拟 IP 的集群配置

使用设备集群时，我们首先要定义在集群中使用的 SLB 虚拟 IP 地址。在下面的段落中定义了我们将要使用的虚拟 IP 地址。

有关服务器负载均衡的内容，请参见“服务器负载均衡（SLB）”章节的相关介绍。

8.3.1.1 主备模式—双节点

在主备模式中，只有一个节点能够成为虚拟 IP 的拥有者，另一个节点将成为备份设备。一旦主控设备失效，备份设备将成为虚拟 IP 地址的所有者。如果在主控设备上启用了抢占模式，当失效的主控节点恢复正常时，它的主控身份也会随之恢复。除此之外，虚拟 IP 地址会一直跟着新的主控设备直到它失效。

下图描述了一个典型的主备模式的网络架构。

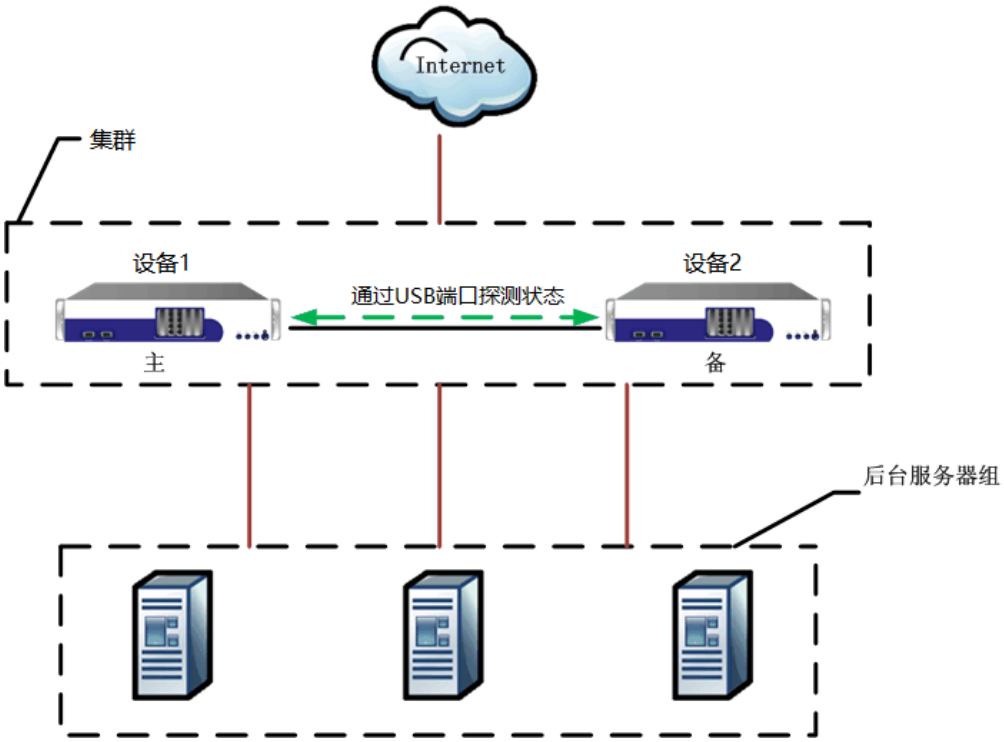

图8–4 主备模式双节点的网络架构

如上图所示，设备 1 为主控设备，负责处理虚拟 IP 地址上的 SLB 流量。设备 2 为备份设备，监听来自主控设备的广播报文。如果设备 1 失效（例如故障）并停止发送广播报文，设备 2 将从备份状态切换到主控状态。

下表列出了配置单节点主备模式集群的命令，相关命令的描述信息，请参考命令行使用手册。

表8–1 设备集群单节点主备模式配置命令

配置操作	命令行
配置 SLB	请参照本文档“服务器负载均衡（SLB）”章节的相关内容。
配置虚拟接口	**cluster virtual ifname** *<interface_name> <cluster_id>*
配置虚拟集群身份认证	**cluster virtual auth** *<interface_name> <cluster_id> {0|1} [password]*
配置抢占模式	**cluster virtual preempt** *<interface_name> <cluster_id> <mode>*
配置虚拟 IP 地址	**cluster virtual vip** *<interface_name> <cluster_id> <vip>*

配置操作	命令行
配置优先级	**cluster virtual priority** *<interface_name> <cluster_id> <priority> [synconfig_peer_name]*
启用虚拟集群	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]*
配置快速失效切换	**cluster virtual ffo interface carrier loss timeout** *<interface_timeout>*

接下来，对设备 1 和设备 2 配置主备集群模式。

1. 为设备 1 和设备 2 进行 SLB 配置。

```
Demo1(config)#slb real http "server1" 192.168.1.50 80 1000 tcp 1 1
Demo1(config)#slb real http "server2" 192.168.1.51 80 1000 tcp 1 1
Demo1(config)#slb group method "group1" rr
Demo1(config)#slb group member "group1" "server1" 1
Demo1(config)#slb group member "group1" "server2" 1
Demo1(config)#slb virtual http "vip1" 192.168.2.100 80
Demo1(config)#slb policy default "vip1" "group1"
Demo2(config)#slb real http "server1" 192.168.1.50 80 1000 tcp 1 1
Demo2(config)#slb real http "server2" 192.168.1.51 80 1000 tcp 1 1
Demo2(config)#slb group method "group1" rr
Demo2(config)#slb group member "group1" "server1" 1
Demo2(config)#slb group member "group1" "server2" 1
Demo2(config)#slb virtual http "vip1" 192.168.2.100 80
Demo2(config)#slb policy default "vip1" "group1"
```

2. 配置虚拟接口名称。

```
Demo1(config)#cluster virtual ifname "port1" 100
Demo2(config)#cluster virtual ifname "port1" 100
```

3. 配置虚拟集群身份认证。

```
Demo1(config)#cluster virtual auth port1 100 0
Demo2(config)#cluster virtual auth port1 100 0
```

4. 配置抢占模式。

现在我们为设备 1 配置抢占模式，而对于设备 2，则不需要配置抢占模式。

```
Demo1(config)#cluster virtual preempt port1 100 1
Demo2(config)#cluster virtual preempt port1 100 0
```

5. 使用“cluster virtual vip”命令为集群配置虚拟 IP 地址。

```
Demo1(config)#cluster virtual vip "port1" 100 192.168.2.100
Demo2(config)#cluster virtual vip "port1" 100 192.168.2.100
```

6. 定义优先级。

优先级决定了哪个节点将成为主控设备，优先级最高的节点将成为主控设备。由于我们希望设备 1 上的 VIP 处于主控状态，因此将该设备上 VIP 的优先级配置为 255。而对于设备 2，我们为其 VIP 配置稍低的优先级，例如 100。在双节点集群的部署环境中，允许进行如上配置。当集群中包含多个节点时，需要为每个节点上的 VIP 设置不同的优先级，以保证节点间能够正常通信和进行失效切换。我们使用下列命令来配置优先级。

```
Demo1(config)#cluster virtual priority port1 100 255
Demo2(config)#cluster virtual priority port1 100 100
```

注意：通过以上配置，设备 2 将成为备份设备，因为它的优先级比设备 1 低。

7. 开启集群。

```
Demo1(config)#cluster virtual on
Demo2(config)#cluster virtual on
```

8. 开启快速失效切换。

```
Demo1(config)#cluster virtual ffo on
Demo1(config)#cluster virtual ffo interface carrier loss timeout 1000
Demo2(config)#cluster virtual ffo on
Demo2(config)#cluster virtual ffo interface carrier loss timeout 1000
```

8.3.1.2 主主模式—双节点

在“主主模式”中，设备 1 将会成为虚拟 IP 地址 1 的主控设备，同时又会成为虚拟 IP 地址 2 的备份设备。设备 2 将成为虚拟 IP2 的主控设备，同时又会成为虚拟 IP 地址 1 的备份设备。这种配置将会提高网站的性能。

下面的例子展示了这种设计。我们需要配置两个虚拟集群 ID（Virtual Cluster ID，VCID），每个虚拟集群 ID 包含至少一个虚拟 IP 地址。如下图所示。

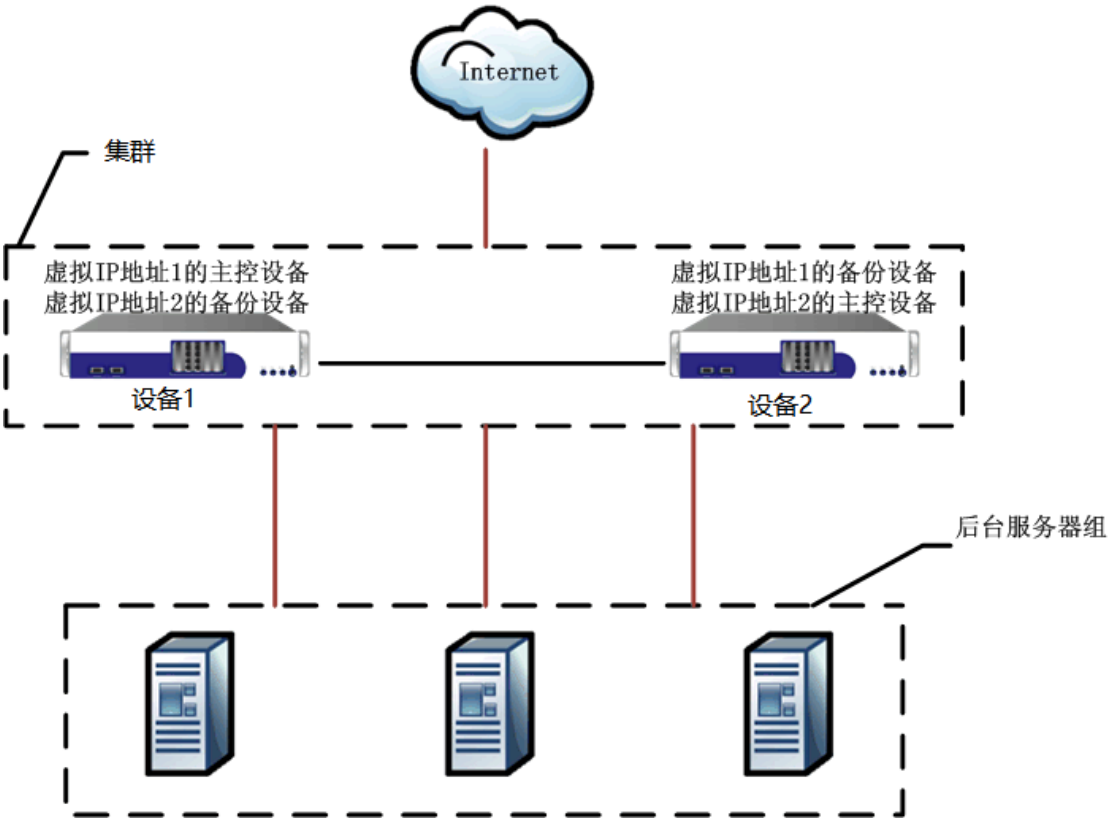

图8–5 主主模式双节点的网络架构

虚拟集群 ID1（VCID1）包含 192.168.2.100 这个虚拟 IP 地址，虚拟集群 ID2（VCID2）包含 192.168.2.101 这个虚拟 IP 地址。

下表列出了配置双节点“主主模式”集群的命令，相关命令的描述信息，请参考命令行使用手册。

表8–2 设备集群双节点主主模式配置命令

配置操作	命令行
配置 SLB	请参照本文档“服务器负载均衡（SLB）”章节的相关内容。
配置虚拟接口	**cluster virtual ifname** *<interface_name> <cluster_id>*
配置虚拟集群身份认证	**cluster virtual auth** *<interface_name> <cluster_id> {0|1} [password]*
配置抢占模式	**cluster virtual preempt** *<interface_name> <cluster_id> <mode>*
配置虚拟 IP 地址	**cluster virtual vip** *<interface_name> <cluster_id> <vip>*
配置优先级	**cluster virtual priority** *<interface_name> <cluster_id> <priority> [synconfig_peer_name]*
启用虚拟集群	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]*

我们将把设备 1 配置为虚拟 IP 地址 1 的主控设备和虚拟 IP 地址 2 的备份设备，把设备 2 配置为虚拟 IP 地址 1 的备份设备和虚拟 IP 地址 2 的主控设备。

接下来我们按照上图来配置我们的设备。

1. 为设备 1 和设备 2 进行 SLB 配置。

```
Demo1(config)#slb real http "server1" 192.168.1.50 80 1000 tcp 1 1
Demo1(config)#slb real http "server2" 192.168.1.51 80 1000 tcp 1 1
Demo1(config)#slb group method "group1" rr
Demo1(config)#slb group member "group1" "server1" 1
Demo1(config)#slb group member "group1" "server2" 1
Demo1(config)#slb virtual http "vip1" 192.168.2.100 80
Demo1(config)#slb virtual http "vip2" 192.168.2.101 80
Demo1(config)#slb policy default "vip1" "group1"
Demo1(config)#slb policy default "vip2" "group1"

Demo2(config)#slb real http "server1" 192.168.1.50 80 1000 tcp 1 1
Demo2(config)#slb real http "server2" 192.168.1.51 80 1000 tcp 1 1
Demo2(config)#slb group method "group1" rr
Demo2(config)#slb group member "group1" "server1" 1
Demo2(config)#slb group member "group1" "server2" 1
Demo2(config)#slb virtual http "vip1" 192.168.2.100 80
Demo2(config)#slb virtual http "vip2" 192.168.2.101 80
Demo2(config)#slb policy default "vip1" "group1"
Demo2(config)#slb policy default "vip2" "group1"
```

2. 为虚拟端口配置名称。

```
Demo1(config)#cluster virtual ifname "port1" 100
Demo1(config)#cluster virtual ifname "port1" 101
Demo2(config)#cluster virtual ifname "port1" 100
Demo2(config)#cluster virtual ifname "port1" 101
```

3. 配置虚拟集群认证。

我们建议您使用带有身份认证机制的集群，以避免未经批准的设备加入到集群中来。

```
Demo1(config)#cluster virtual auth port1 100 0
Demo1(config)#cluster virtual auth port1 101 0
Demo2(config)#cluster virtual auth port1 100 0
Demo2(config)#cluster virtual auth port1 101 0
```

4. 配置集群抢占模式。

```
Demo1(config)#cluster virtual preempt port1 100 1
Demo1(config)#cluster virtual preempt port1 101 0
Demo2(config)#cluster virtual preempt port1 100 0
Demo2(config)#cluster virtual preempt port1 101 1
```


5. 使用“**cluster virtual vip**”命令配置虚拟 IP。

```
Demo1(config)#cluster virtual vip "port1" 100 192.168.2.100
Demo1(config)#cluster virtual vip "port1" 101 192.168.2.101
Demo2(config)#cluster virtual vip "port1" 100 192.168.2.100
Demo2(config)#cluster virtual vip "port1" 101 192.168.2.101
```

6. 配置优先级。

集群优先级决定了哪个设备将成为主控设备，优先级最高的设备会成为主控设备。

```
Demo1(config)#cluster virtual priority port1 100 255
Demo1(config)#cluster virtual priority port1 101 100
Demo2(config)#cluster virtual priority port1 100 100
Demo2(config)#cluster virtual priority port1 101 255
```

7. 启用集群功能。

```
Demo1(config)#cluster virtual on
Demo2(config)#cluster virtual on
```

8.3.1.3 主主模式—三节点

本节将介绍如何在多个设备上配置集群。利用矩阵概念可以更好的解释这种机制：依次设置多个节点的优先级，使得一个节点失效以后网络负载能够被正确的分配到其它各个工作节点上去。当然，在极端的情况下，当集群中的三个节点中的两个失效时，其余的节点就要处理这个网站的所有负载。下图是一个典型示例。

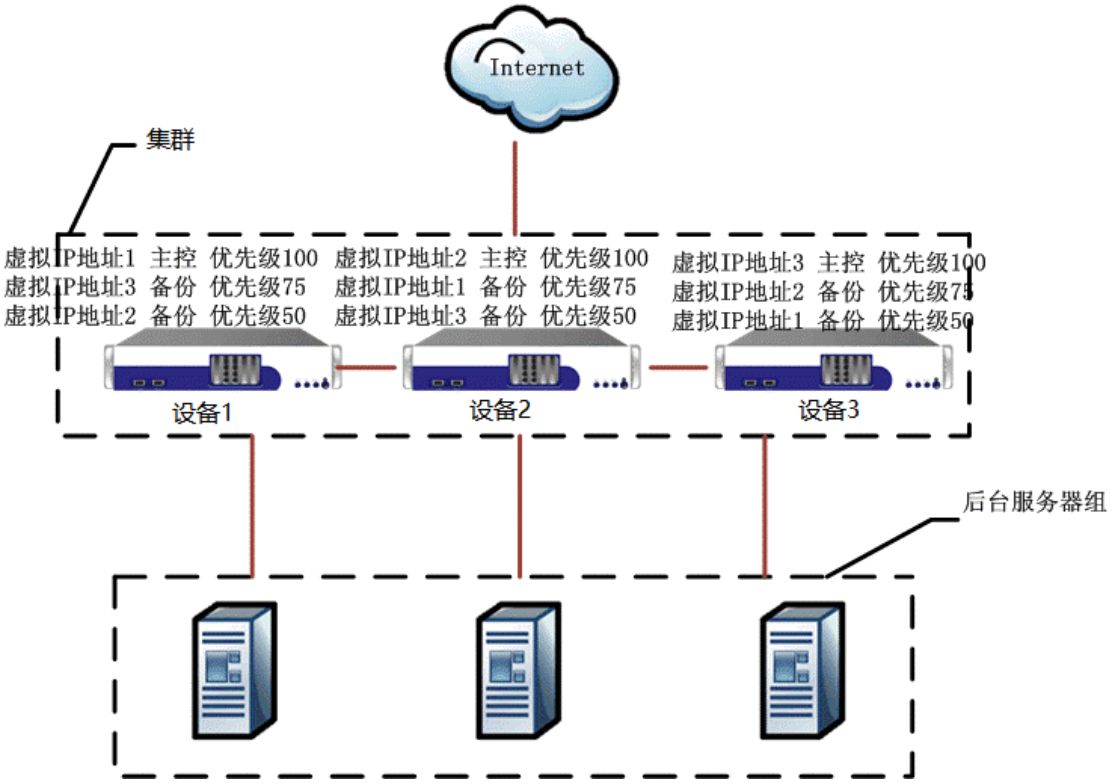

图8–6 服务器负载均衡主主模式三节点集群矩阵

下表列举了一个集群配置中的三个应用配置。这样设置的优先级使得任意一个节点失效以后，负载会分配到另外两个节点上去。

表8–3 设备集群优先级配置

节点/优先级	VIP1 10.10.0.10	VIP2 10.10.0.20	VIP3 10.10.0.30
设备 1	100	50	75
设备 2	75	100	50
设备 3	50	75	100

1. 配置设备 1。

```
Demo1(config)#cluster virtual ifname port1 1
Demo1(config)#cluster virtual auth port1 1 0
Demo1(config)#cluster virtual preempt port1 1 1
Demo1(config)#cluster virtual vip port1 1 10.10.0.10
Demo1(config)#cluster virtual priority port1 1 100
Demo1(config)#cluster virtual ifname port1 2
Demo1(config)#cluster virtual auth port1 2 0
Demo1(config)#cluster virtual preempt port1 2 0
Demo1(config)#cluster virtual vip port1 2 10.10.0.20
Demo1(config)#cluster virtual priority port1 2 50
Demo1(config)#cluster virtual ifname port1 3
Demo1(config)#cluster virtual auth port1 3 0
```

```
Demo1(config)#cluster virtual preempt port1 3 0
Demo1(config)#cluster virtual vip port1 3 10.10.0.30
Demo1(config)#cluster virtual priority port1 3 75
Demo1(config)#cluster virtual on
```

2. 配置设备 2。

```
Demo2(config)#cluster virtual ifname port1 1
Demo2(config)#cluster virtual auth port1 1 0
Demo2(config)#cluster virtual preempt port1 1 0
Demo2(config)#cluster virtual vip port1 1 10.10.0.10
Demo2(config)#cluster virtual priority port1 1 75
Demo2(config)#cluster virtual ifname port1 2
Demo2(config)#cluster virtual auth port1 2 0
Demo2(config)#cluster virtual preempt port1 2 1
Demo2(config)#cluster virtual vip port1 2 10.10.0.20
Demo2(config)#cluster virtual priority port1 2 100
Demo2(config)#cluster virtual ifname port1 3
Demo2(config)#cluster virtual auth port1 3 0
Demo2(config)#cluster virtual preempt port1 3 0
Demo2(config)#cluster virtual vip port1 3 10.10.0.30
Demo2(config)#cluster virtual priority port1 3 50
Demo2(config)#cluster virtual on
```

3. 配置设备 3。

```
Demo3(config)#cluster virtual ifname port1 1
Demo3(config)#cluster virtual auth port1 1 0
Demo3(config)#cluster virtual preempt port1 1 0
Demo3(config)#cluster virtual vip port1 1 10.10.0.10
Demo3(config)#cluster virtual priority port1 1 50
Demo3(config)#cluster virtual ifname port1 2
Demo3(config)#cluster virtual auth port1 2 0
Demo3(config)#cluster virtual preempt port1 2 0
Demo3(config)#cluster virtual vip port1 2 10.10.0.20
Demo3(config)#cluster virtual priority port1 2 75
Demo3(config)#cluster virtual ifname port1 3
Demo3(config)#cluster virtual auth port1 3 0
Demo3(config)#cluster virtual preempt port1 3 1
Demo3(config)#cluster virtual vip port1 3 10.10.0.30
Demo3(config)#cluster virtual priority port1 3 100
Demo3(config)#cluster virtual on
```

8.3.2 内部端口集群配置

内部端口集群配置同服务器负载均衡虚拟 IP 的集群有所不同。

注意：如果内网中的设备需要通过设备访问其他网络，那么我们强烈推荐使用 NAT 网络地址转换。

有两种方法可以用于建立内部端口集群。第一种是使用一个属于虚拟集群中某个设备的虚拟 IP 地址。第二种，如果我们需要在不同节点间分担网络流量，我们需要在内部端口上配置“主主”模式。下面，我们将分别介绍这两种配置方法。

8.3.2.1 主备模式（单虚拟 IP 地址）

在主备模式下，集群中的一台设备作为内网的网关。当主控设备出现意外故障时，集群中的备份设备将会切换成为主控设备。为了实现该目的，我们选取了内网中空闲 IP 地址（192.168.1.3）来作为内网的网关。

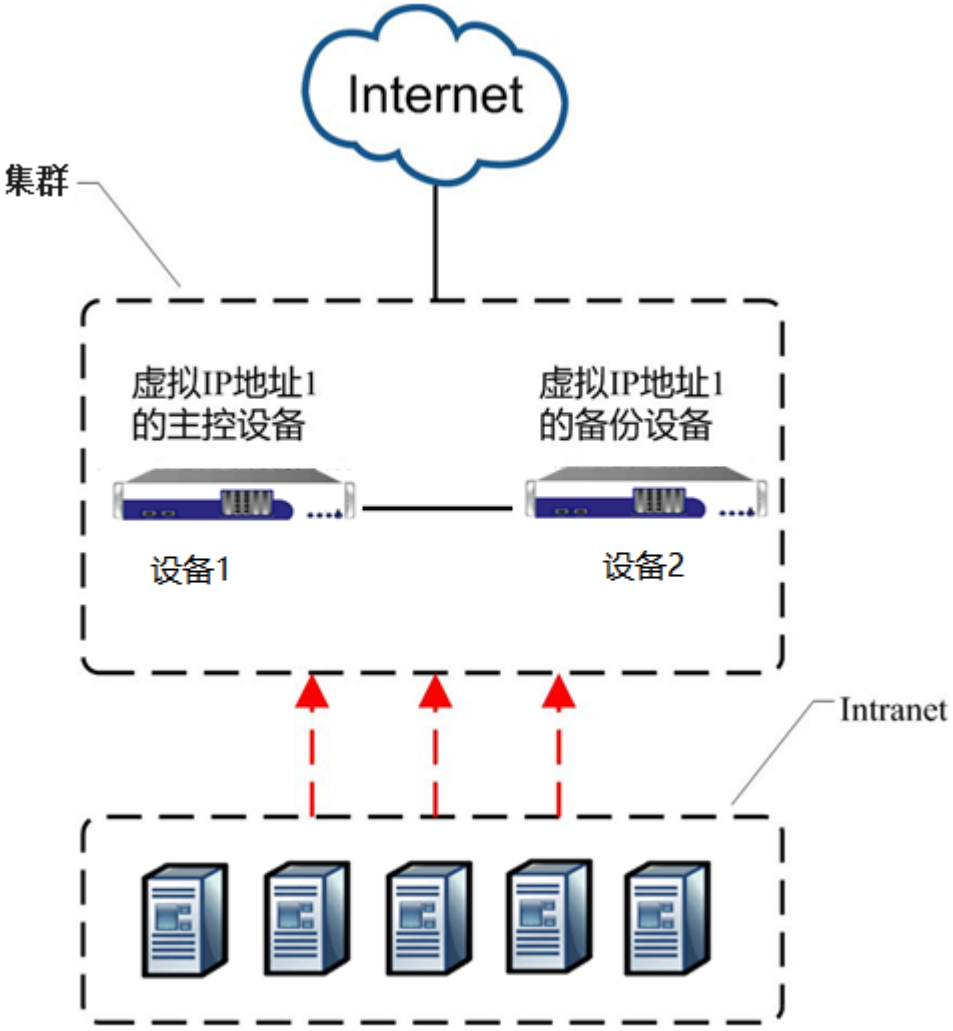

图8–7 内部端口主备模式的集群配置

下表列出了配置内部端口主备模式集群的命令。

表8–4 设备内部端口主备模式集群配置命令

配置操作	命令行
配置一个虚拟端口	**cluster virtual ifname** *<interface_name> <cluster_id>*
配置虚拟 IP 地址	**cluster virtual vip** *<interface_name> <cluster_id> <vip>*
配置优先级	**cluster virtual priority** *<interface_name> <cluster_id> <priority> [synconfig_peer_name]*
启用虚拟集群	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]*

接下来我们按照上图来配置内部端口主备模式集群。

1. 配置虚拟端口和它的集群 ID。

```
Demo1(config)#cluster virtual ifname "port2" 100
Demo2(config)#cluster virtual ifname "port2" 100
```

2. 使用“**cluster virtual vip**”命令配置虚拟 IP 地址。

```
Demo1(config)#cluster virtual vip "port2" 100 192.168.1.3
Demo2(config)#cluster virtual vip "port2" 100 192.168.1.3
```

3. 配置优先级。

集群优先级决定了哪个设备将成为主控设备，优先级最高的设备会成为主控设备。

```
Demo1(config)#cluster virtual priority port2 100 255
Demo2(config)#cluster virtual priority port2 100 100
```

4. 启用虚拟集群。

```
Demo1(config)#cluster virtual on
Demo2(config)#cluster virtual on
```

8.3.2.2 主主模式（双虚拟 IP 地址）

在“主主模式”中，我们将建立两个虚拟 IP 地址作为内网的网关。网络中一半服务器的预设路由指向虚拟 IP 地址 1，另一半指向虚拟 IP 地址 2，以此达到在设备间均衡负载的目的。

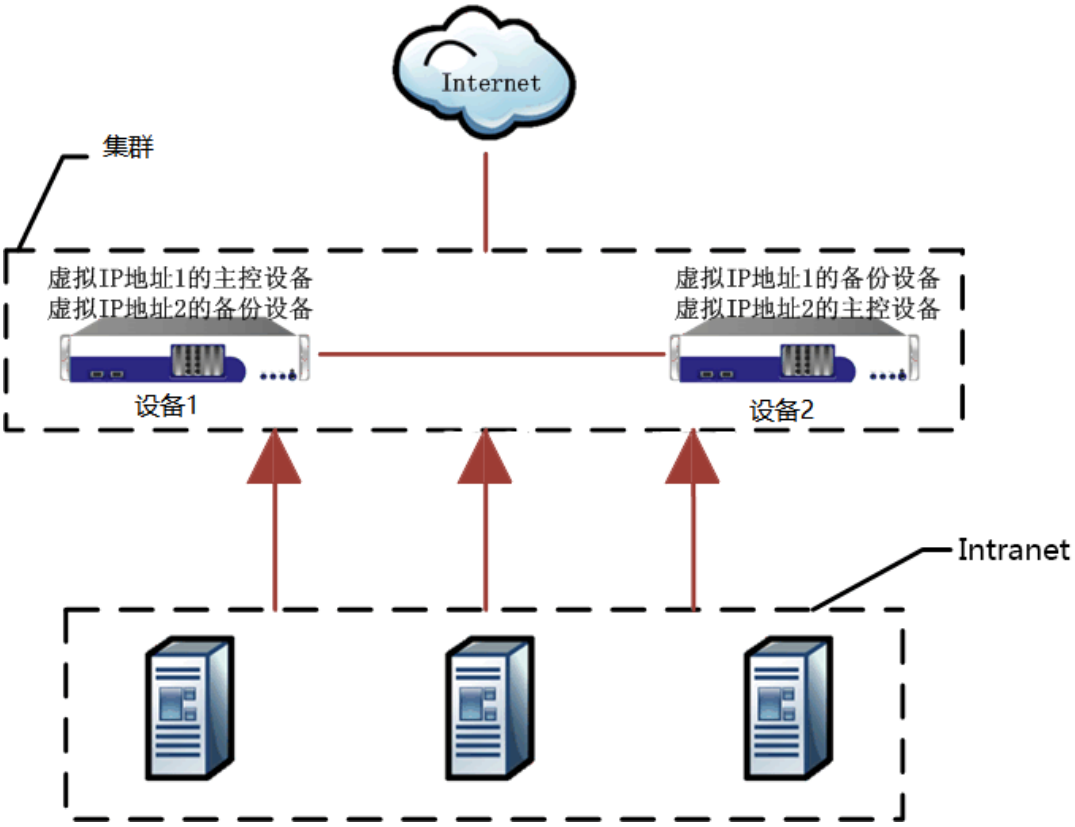

图8–8 内部端口主主模式的集群配置

下表列出了配置内部端口主主模式集群的命令。

表8–5 设备内部端口主主模式集群配置命令

配置操作	命令行
配置一个虚拟端口	**cluster virtual ifname** *<interface_name> <cluster_id>*
配置虚拟 IP 地址	**cluster virtual vip** *<interface_name> <cluster_id> <vip>*
配置优先级	**cluster virtual priority** *<interface_name> <cluster_id> <priority> [synconfig_peer_name]*
启用虚拟集群	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]*

1. 配置虚拟端口和它的集群 ID。

```
Demo1(config)#cluster virtual ifname port2 1
Demo1(config)#cluster virtual ifname port2 2
Demo2(config)#cluster virtual ifname port2 1
Demo2(config)#cluster virtual ifname port2 2
```

2. 使用“**cluster virtual vip**”命令配置虚拟 IP 地址。

```
Demo1(config)#cluster virtual vip port2 1 192.168.1.3
Demo1(config)#cluster virtual vip port2 2 192.168.1.4
Demo2(config)#cluster virtual vip port2 1 192.168.1.3
Demo2(config)#cluster virtual vip port2 2 192.168.1.4
```

3. 配置优先级。

集群优先级决定了哪个设备将成为主控设备。优先级最高的设备会成为主控设备。

```
Demo1(config)#cluster virtual priority port2 1 255
Demo1(config)#cluster virtual priority port2 2 100
Demo2(config)#cluster virtual priority port2 1 100
Demo2(config)#cluster virtual priority port2 2 255
```

4. 启用虚拟集群。

```
Demo1(config)#cluster virtual on
Demo2(config)#cluster virtual on
```

8.3.2.3 在 VLAN 内部配置集群

配置 VLAN 集群同配置内部接口的集群类似，假如你在设备上进行了相关的 VLAN 配置，你还可以对 VLAN 进行内部集群配置。

下表列出了配置 VLAN 内部集群所需要使用的命令。

表8–6 设备 VLAN 内部集群配置命令

配置操作	命令行
配置一个虚拟端口	**cluster virtual ifname** <*interface_name*> <*cluster_id*>
配置虚拟 IP 地址	**cluster virtual vip** <*interface_name*> <*cluster_id*> <*vip*>
配置优先级	**cluster virtual priority** <*interface_name*> <*cluster_id*> <*priority*> *[synconfig_peer_name]*
启用虚拟集群	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]*
查看集群配置	**show cluster virtual status**

设备 1 上的配置：

1. 在 Port2 接口上创建 vlan-1 和 vlan-2。

```
Demo1(config)#vlan port2 vlan-1 10
Demo1(config)#vlan port2 vlan-2 20
```

9. 为 vlan-1 和 vlan-2 配置 IP 地址。

```
Demo1(config)#ip address vlan-1 192.168.1.1 255.255.255.0
Demo1(config)#ip address vlan-2 192.168.2.1 255.255.255.0
```

10. 为 VLAN 创建虚拟接口。

```
Demo1(config)#cluster virtual ifname vlan-1 1
Demo1(config)#cluster virtual vip vlan-1 1 192.168.1.3
Demo1(config)#cluster virtual priority vlan-1 1 255
```

```
Demo1(config)#cluster virtual ifname vlan-2 1
Demo1(config)#cluster virtual vip vlan-2 1 192.168.2.3
Demo1(config)#cluster virtual priority vlan-2 1 100
```

11. 启用集群。

```
Demo1(config)#cluster virtual on 1
```

设备 2 上的配置：

1. 在 Port2 接口上创建 vlan-1 和 vlan-2。

```
Demo2(config)#vlan port2 vlan-1 10
Demo2(config)#vlan port2 vlan-2 20
```

12. 为 vlan-1 和 vlan-2 配置 IP 地址。

```
Demo2(config)#ip address vlan-1 192.168.1.2 255.255.255.0
Demo2(config)#ip address vlan-2 192.168.2.2 255.255.255.0
```

13. 为 VLAN 创建虚拟接口。

```
Demo2(config)#cluster virtual ifname vlan-1 1
Demo2(config)#cluster virtual vip vlan-1 1 192.168.1.3
Demo2(config)#cluster virtual priority vlan-1 1 100
Demo2(config)#cluster virtual ifname vlan-2 1
Demo2(config)#cluster virtual vip vlan-2 1 192.168.2.3
Demo2(config)#cluster virtual priority vlan-2 1 255
```

14. 启用集群。

```
Demo2(config)#cluster virtual on 1
```

最后，使用“**show cluster virtual status**”命令查看集群状态。

```
Demo2(config)#show cluster virtual status
```

注意：如果设备 1 的 Port2（Port1）接口失效了，所有设备 1 上 Port1 接口和 Port2 接口上的集群 ID 会被设置成为“初始状态”。这将保证外部，内部流量都会从相同的设备经过。

8.3.2.4 在 MNET 内部配置集群

MNET 集群同 VLAN 集群配置一样。不过需要注意 VLAN 和 MNET 不能同时配置。

下表列出了配置 VLAN 内部集群所需要使用的命令。

表8–7 设备 MNET 内部集群配置命令

配置操作	命令行
配置一个虚拟端口	**cluster virtual ifname** *<interface_name> <cluster_id>*
配置虚拟 IP 地址	**cluster virtual vip** *<interface_name> <cluster_id> <vip>*
配置优先级	**cluster virtual priority** *<interface_name> <cluster_id> <priority> [synconfig_peer_name]*
启用虚拟集群	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]*
查看集群配置	**show cluster virtual status**

在设备 1 上的配置：

1. 在 Port2 接口上配置 MNET。

```
Demo1(config)#mnet port2 mnet-1
Demo1(config)#mnet port2 mnet-2
Demo1(config)#ip address mnet-1 192.168.1.1 255.255.255.0
Demo1(config)#ip address mnet-2 192.168.2.1 255.255.255.0
```

15. 下面为集群配置 MNET 接口。

```
Demo1(config)#cluster virtual ifname mnet-1 1
Demo1(config)#cluster virtual vip mnet-1 1 192.168.1.3
Demo1(config)#cluster virtual ifname mnet-2 1
Demo1(config)#cluster virtual vip mnet-1 1 192.168.2.3
```

16. 设置虚拟集群的优先级。

```
Demo1(config)#cluster virtual priority mnet-1 1 255
Demo1(config)#cluster virtual priority mnet-2 1 100
```

17. 启用集群。

```
Demo1(config)#cluster virtual on 1
```

对于设备 2 我们作同样配置：

```
Demo2(config)#mnet port2 mnet-1
Demo2(config)#mnet port2 mnet-2
Demo2(config)#ip address mnet-1 192.168.1.2 255.255.255.0
Demo2(config)#ip address mnet-2 192.168.2.2 255.255.255.0
Demo2(config)#cluster virtual ifname mnet-1 1
Demo2(config)#cluster virtual vip mnet-1 1 192.168.1.3
Demo2(config)#cluster virtual ifname mnet-2 1
Demo2(config)#cluster virtual vip mnet-2 1 192.168.2.3
Demo1(config)#cluster virtual priority mnet-1 1 200
Demo1(config)#cluster virtual priority mnet-2 1 10
Demo2(config)#cluster virtual on 1
```

最后使用“**show cluster virtual status**”命令查看集群状态。

```
Demo2(config)#show cluster virtual status
```

第9章 高可用性（HA）

9.1 概述

随着网络应用的不断深入和发展，用户对网络和网络设备可靠性的要求越来越高。在网络规划设计时，为提高网络的可靠性，一般需要对关键节点的网络设备进行冗余备份。本文“集群”章节提及的集群功能通过 VRRP 技术来解决单点故障问题，本章将介绍设备引入的高可用性（High Availability，HA）功能，该功能不仅可以解决单点故障问题，并且提供了更多的可靠性保证策略。

HA 功能允许两台或者多台设备持续不断地交互各自的状态信息，并保持各台设备上的配置信息同步更新。当一台设备发生故障时，其它可用设备将会自动接管该节点处理的应用服务，从而保证了应用服务的高可用性。

此外，HA 功能还提供了连接同步（Stateful Session Failover，SSF）功能，保证当发生业务失效切换时，该业务上的连接也能够同步切换到新的设备上，从而避免连接中断，提升用户体验。

HA 功能的部署方式非常灵活。除了两台设备的 Active/Active 和 Active/Standby 部署场景外，HA 还支持多台设备互为备份的部署场景。

9.2 基本概念

9.2.1 HA 域和节点

HA 域是由一组提供 HA 功能的设备组成。HA 域中的设备通常称为节点。一个 HA 域支持配置多达 32 个节点。

9.2.2 浮动 IP 分组和浮动 MAC

通常来说，节点上的主备切换是通过浮动 IP 切换来实现。相同的浮动 IP 可以定义在多个节点上，但在同一时刻，只有一个节点上的浮动 IP 为 Active 状态。

为了保证切换的一致性和灵活性，在 HA 技术中，浮动 IP 的切换是按照分组进行的，即每个浮动 IP 必须加入浮动 IP 分组中才能实现状态切换。同一个分组中的所有浮动 IP 在同一时刻保持相同状态，也称为浮动 IP 分组状态。

浮动 IP 分组的状态是由分组优先级、切换模式和分组相关的健康检查结果决定的。在正确地配置了浮动 IP 分组后，HA 模块将根据配置的健康检查条件对分组的运行环境进行检查。根据检查结果，分组可以有以下两类状态：

- Active/Standby：分组通过了所关联的所有健康检查，表明分组处于可以提供服务的状态。此时，分组状态为“Active”或“Standby”。如果分组状态为“Active”，该节点将获得该分组中所有的浮动 IP 地址，并提供服务。如果

分组状态为“Standby”，该节点将提供服务备份，在发生切换时，接管服务。

- Init：初始化状态。分组没有通过所关联的任意一个健康检查，分组状态为“Init”。当分组处于“Init”状态时，表明节点无法提供分组对应的服务，即使在它节点上该分组的状态发生了切换，也无法接管服务。

注意：当分组处于“Init”状态时，需要检查分组配置或者健康检查的结果，使分组状态切换为“Active/Standby”，以提供服务或服务备份。

一个节点上可以配置多个浮动 IP 分组，各个分组的状态相互独立。如果需要同时切换节点上的所有分组，需采用“Unit_Failover”切换方式（参见“失效切换规则”小节）。

浮动 IP 地址通过“**ha group fip**”和“**ha group fiprange**”命令配置。如需详细信息，请参见命令行使用手册。

HA 功能提供浮动 MAC 功能。当分组状态发生切换时，浮动 MAC 地址将浮动到当前分组状态为“Active”的节点上。这样，分组状态切换发生后，即使提供应用服务的设备发生了变化，由于使用的 MAC 地址没有发生改变，所以客户端感知不到提供服务的设备发生了变化。位于同一浮动 IP 分组内的浮动 IP 地址（包括浮动 IP 地址段）可以绑定相同的浮动 MAC 地址，但是在不同的浮动 IP 分组之间浮动 MAC 地址不能重叠。默认情况下，系统禁用浮动 MAC 功能。在启用浮动 MAC 功能前，必须先执行“**ha off**”命令禁用 HA 功能。

在配置浮动 IP 地址时，可以选择绑定浮动 MAC 地址。为浮动 IP 分组绑定浮动 MAC 地址后，MAC 地址将被加入到浮动 IP 分组所在的物理接口的 MAC 地址列表。通过接口上的浮动 IP 地址发送的报文将使用浮动 MAC 地址作为源 MAC 地址，从接口上发出的其他报文将使用接口 MAC 地址。接口上的 MAC 地址可以在“**show interface**”输出中查看到，包括浮动 MAC 地址。

9.2.3 分组切换模式

HA 支持两种分组切换模式：非抢占（non-preempt）模式和抢占（preempt）模式。

当一个浮动 IP 分组被启用在多个节点上时：

- 在非抢占模式下：只有发生了失效切换，本节点上的分组状态才会发生切换。
- 在抢占模式下：如果本节点上设置的分组优先级高于所有对端节点上分组优先级，那么强制将本节点上的分组状态设置为“Active”。如果在此之前在某对端节点上的分组状态为“Active”，那么该状态将会被强制修改为“Standby”。

9.2.4 HA 部署场景

HA 功能提供多种部署场景。除了支持常见的两台设备的 Active/Active 和 Active/Standby 部署场景外，还支持多台设备互为备份的部署场景。

- Active/Active 部署场景是指 HA 域只包含两个节点，每个节点上都有“Active”状态的浮动 IP 分组，同时对端节点上“Active”状态的浮动 IP 分组在本节点上的状态为“Standby”。
- Active/Standby 部署场景是指 HA 域只包含两个节点，所有浮动 IP 分组的状态在一个节点上为“Active”，在另一个节点上为“Standby”。
- 多台设备互为备份的部署场景：多台设备用来提供服务和服务备份。其中，N+1 为最常见的部署场景。即 HA 域包含 N+1 个节点，在其中 N 个节点上，浮动 IP 分组的状态都为“Active”，在另外一个节点上，浮动 IP 分组的状态都为“Standby”。

9.3 可靠通信链路

HA 域中的节点可以通过多种通信链路交互各自的状态信息，从而确保通信的高可靠性。HA 节点可以通过以下三种链路进行通信：

- FFO（Fast Failover）链路
- 主链路（Primary Link）
- 备用链路（Secondary Link）

FFO 链路必须配合主链路或备用链路一起使用。FFO 链路使用专用的 FFO 线连接两台设备的专用 FFO 端口，因此 FFO 链路只能用于只有两个节点的 Active/Active 和 Active/Standby 部署场景中。默认情况下，系统不启用 FFO 链路。FFO 链路主要提供如下功能：

- 发送心跳包：在 HA 功能运行时，本节点通过 FFO 链路向对端节点发送心跳包，用于探测对端节点的运行状况。
- 用于启动时配置同步：启用 FFO 链路和启动时配置同步后，本节点将会通过 FFO 链路从对端节点同步节点和链路相关的配置信息。
- 用于快速失效切换：当一个节点失效时，对端节点通过 FFO 链路实现 VIP 的快速失效切换。

主链路和备用链路都是通过普通网线将两个节点连接起来，因此统称为网络链路。每两个节点之间有且只有一条主链路，可以有 1~31 条备用链路。默认情况下，系统启用网络链路。

在节点加入 HA 域后，节点之间将会自动建立起主链路连接。主链路主要提供如下功能：

- 发送心跳包：本节点通过主链路向所有对端节点发送心跳包，用于探测其它节点的运行状况。
- 用于启动时配置同步：在启动时配置同步模式（bootup synconfig）下，启用 HA 功能后，本节点通过主链路从对端节点同步通过“**write memory**”命令保存的配置信息。
- 用于运行时配置同步：在运行时配置同步模式（runtime synconfig）下，如果本节点的相关的白名单配置发生变化时，本节点通过主链路将白名单的配置变化同步到对端节点。本节点黑名单的配置发生变化时，黑名单的配置变化不会同步到对端节点。

备用链路只能用来向对端节点发送心跳包，是可选配置。使用时需要手动在本地和对端 HA 节点上都进行备用链路配置。需要注意的是，在两个 HA 节点之间建立一条备用链路，需要分别在两个节点上配置一个 ID 相同的备用链路。

例如，两个 HA 节点“u1”和“u2”的备用链路的 IP 地址分别为 192.168.1.1 和 192.168.1.2，如果为两个 HA 节点创建一条备用链路，分别在每个 HA 节点上做如下配置：

```
Demo(config)#ha link network secondary u1 1 192.168.1.1 65521
Demo(config)#ha link network secondary u2 1 192.168.1.2 65521
```

这样，HA 节点“u1”和“u2”之间就建立了 ID 为“1”的备用链路。

下表介绍了三种通信链路的相同点和差异点。

表9–1 HA 通信链路的异同对比

项目		FFO 链路	主链路	备用链路
不同点	连接方式	通过专用 FFO 线直连。	通过普通网线直连或者通过网络连接。	
	链路配置	不需配置。	在本地 HA 节点和对端 HA 节点都加入 HA 域后，HA 将在两个节点之间自动建立起主链路连接。	需要手动在本地和对端 HA 节点上都进行备用链路配置。
	应用场景	只能应用于 Active/Active 和 Active/Standby 部署场景。	应用于所有部署场景。	
	登录 HA 域的方式	启用 FFO 链路、网络链路和 HA 功能。	• 配置本地和对端节点的 IP 地址。 • 启用 HA 功能。	不涉及。
相同点		• 三种链路都可以用来发送心跳包。HA 节点通过发送心跳		

项目	FFO 链路	主链路	备用链路
	包交换各自的健康检查状况和分组状态信息等。 • 在 Active/Active 和 Active/Standby 部署场景下，三种链路可以互为备份。 • 如果三种链路都失效，则表明对端设备发生故障。		

9.4 失效切换规则

在 HA 域中，HA 模块会对系统状态和网络状况进行健康检查。当健康检查的结果表明节点出现故障并满足定义的分组切换条件时，要进行切换操作。通常需要重新选出最高优先级的可用节点，并将该节点上的浮动 IP 分组的状态强制修改为“Active”。为了实现该目的，HA 提供失效切换规则来控制分组状态的切换。

失效切换规则是由切换条件和切换动作组成。其中，切换条件指系统的某个软硬件的监控状态。常见的切换条件有网络接口状态、CPU 使用率等。而切换动作则是指切换条件发生时，系统执行的动作。HA 提供三种切换动作，每种动作的含义如下：

- **Group_Failover**：对指定的浮动 IP 分组执行状态切换动作。该切换动作的含义是：按照健康状况和分组优先级选择新节点，并将其上的浮动 IP 分组为“Active”状态，并接管服务。
- **Unit_Failover**：对节点上所有的浮动 IP 分组执行状态切换动作。
- **Reboot**：对节点上所有的浮动 IP 分组执行状态切换动作后，再执行重启设备操作。

HA 支持以下健康检查类型：

➢ **预定义健康检查：**

HA 预定义了基本的网络连通性检查，即 PORT_1~PORT_32，用于检查网络接口故障和节点之间网络中断等异常情况。默认情况下，当网络接口出现故障时，系统会执行 Group_Failover 切换，管理员可以根据需要修改预定义健康检查关联的失效切换动作。

注意：

- 只有 Bond 接口中所有接口的网线连接不正常时，Bond 接口上的 IP 所在的浮动 IP 分组才会执行“Group_Failover”动作。
- 对于 HA 域中为同一浮动 IP 分组提供失效切换支持的各节点，需要为该浮动 IP 分组配置相同的失效规则,包括切换规则中健康检查条件的名称。
- 对于 HA 域中为“Unit_Failover”提供失效切换支持的各节点，需要为“Unit_Failover”配置相同的失效规则，包括切换规则中健康检查条件的名称。

➢ **系统健康检查：**

HA 功能可以复用系统健康检查作为失效切换条件。管理员可以自定义以下系统健康检查条件用于 HA 节点的失效切换：

硬件类：

- CPU 温度健康检查条件
- SSL 加速卡健康检查条件
- 聚合接口健康检查条件

软件类：

- CPU 使用率健康检查条件
- 内存使用率健康检查条件
- NTP 状态健康检查条件
- 系统硬盘分区使用率健康检查条件
- 数据硬盘分区使用率健康检查条件
- SSL 加速卡的 AE 或 SE 内核使用率健康检查条件

网络环境类：

- 网关健康检查条件

在一些复杂的应用环境中，管理员需要定义一些更加复杂的失效切换规则。例如：在 Bond 接口环境中，只有所有网络端口的网线连接不正常，才执行切换动作。实际应用却要求在任意一个端口网线连接不正常，就执行切换动作。为了支持这些复杂应用，HA 功能支持配置健康检查条件组（vcondition）。vcondition 可以嵌套多个子健康检查条件，子条件之间逻辑关系可以指定为“AND”或者“OR”。对于上述例子的问题，可以通过定义多个网络端口的健康检查条件，并使用“OR”关系把它们组合成一个 vcondition，然后把 vcondition 关联切换动作。

- **自定义健康检查脚本：**

设备支持通过 SCP 或 TFTP 服务器导入自定义的健康检查脚本。自定义的健康检查脚本需通过签名认证后才生效，如需使用该功能，请联系公司技术支持获取脚本。

9.5 配置同步

HA 功能提供配置同步功能，简化了节点上的配置，并确保域中所有节点上配置信息的一致性。HA 功能支持两种配置同步方式：启动时配置同步（bootup synconfig）和运行时配置同步（runtime synconfig）。两种配置同步方式可以同时启用。

9.5.1 启动时配置同步

启动时配置同步是指节点登录到 HA 域中后，通过主链路将对端节点的配置信息同步到本地。如果启用了 FFO 链路，节点将通过 FFO 链路从对端获取 HA 节点和链路配置信息。当通过命令“**synconfig secure on**”启用了配置同步的安全模式时，在使用启动时配置同步功能前，管理员需要：

- 确保在对端节点和本地节点通过命令“**synconfig challenge**”配置了相同的 Challenge Code。
- 使用命令“**synconfig peer** <*peer_name*> <*peer_ip*>”在对端节点和本地节点上配置了同步节点，且参数“peer_name”需要同命令“**ha unit** <*unit_name*> <*ip_address*> *[port]*”中“unit_name”相同。

节点可以通过两种方式登录到域中：

- FFO Login 方式：是指节点通过 FFO 链路登录到 HA 域中。
- Network Login 方式：是指节点通过网络链路登录到 HA 域中。

如果使用 FFO Login 方式登录域并进行启动时配置同步，需要依次执行如下操作：

1. 执行“**ha link ffo on**”命令启用 FFO 链路。
2. 执行“**ha synconfig bootup on**”命令启用启动时配置同步功能。

18. 执行“**ha link network on**”命令启用网络链路。

19. 执行“**ha on**”命令启用 HA 功能。

如果使用 Network Login 方式登录域并进行启动时配置同步，需要依次执行如下操作：

1. 执行“**ha unit**”命令配置本地和对端 HA 节点信息。
2. 执行“**ha synconfig bootup on**”命令启用启动时配置同步功能。
3. 执行“**ha link network on**”命令启用网络链路。
4. 执行“**ha on**”命令启用 HA 功能。

注意：

1. 在启动时配置同步方式下，只有使用“**wirte memory**”命令保存过的配置信息才会被同步。
2. 在通过网络链路登录到 HA 域的过程中，本地节点会同步对端节点的系统时间。

9.5.2 运行时配置同步

运行时配置同步是指在 HA 功能运行的过程中，管理员在一个节点上增删或者修改相关的命令配置时，该节点能够将变化的配置信息自动同步到域中其它节点。这样能够确保 HA 域中所有节点上的配置信息相同。

注意：只有在本节点和对端节点上都启用了运行时配置同步方式，节点才会把本地配置变化同步到对端节点。如果此功能在对端节点为禁用状态，则“**ha synconfig runtime**”命令将会报错端口不可达。

9.5.3 增量配置同步

增量配置同步是指将新增配置批量同步到指定或全部对端节点。

9.6 连接同步

连接同步（Stateful Session Failover，SSF）支持最多三个 HA 节点之间的连接同步。启用 SSF 功能后，“Active”状态的浮动 IP 分组上创建的 TCP 和 UDP 连接信息会被实时更新到“Standby”状态的分组所在的节点上。当发生切换动作时，因为在分组状态为“Active”的新节点上拥有所有的连接信息，所以已经创建的 TCP 或者 UDP 连接可以继续使用。但是，如果禁用了 SSF 功能，则已经创建 TCP 和 UDP 连接将无法继续使用。

SSF 功能可以支持 TCP、UDP、FTP、IP、TCPS、HTTP 和 HTTPS 类型的 SLB 应用以及 NAT 应用。管理员可以基于单个虚拟服务来启用或禁用 SSF 功能。

注意：

- 为了保证 SSF 功能能够正常使用，请确保同一个 HA 域中的各个节点上的 HA 相关的配置相同。使用 SSF 功能时，推荐启用运行时配置同步方式。
- 在两个 HA 节点之间使用可靠的网络链路来传输 SSF 会话信息。如果该链路发生故障，那么两个节点将无法交换会话信息。如果随后发生了分组切换，已有连接可能会被重置。

9.7 HA 日志

HA 提供日志功能。默认情况下，系统禁用该功能。如果启用了 HA 功能，则日志功能同时被启用；如果禁用了 HA 功能，则日志功能同时被禁用。

设备支持八个 HA 日志级别：emerg、alert、crit、err、warning、notice、info 和 debug。管理员可以设置 HA 日志的级别。一旦设定了日志级别，低于这个级别的 HA 日志消息将被忽略，即系统不记录这些日志。默认的 HA 日志级别为 info。

HA 内部信息计入 HA 日志，浮动 IP 分组和节点变化将被计入系统日志。

9.8 配置示例

HA 功能可以部署在以下典型场景中：

- 场景 1：Active/Standby
- 场景 2：Active/Active
- 场景 3：N+1

下列章节将分别介绍三种典型场景的配置示例。

9.8.1 场景 1：Active/Standby

9.8.1.1 配置目标

Active/Standby 部署场景可以用来实现如下配置目标：

- HA 域包含两个 HA 节点，每个节点上都启用同一个浮动 IP 分组。
- 浮动 IP 分组中包含两个应用服务的 VIP 地址。
- 节点 1 提供应用服务，节点 2 提供备份。
- 使用 FFO 链路做快速失效切换。

实现如上配置目标的网络拓扑图如下图所示。

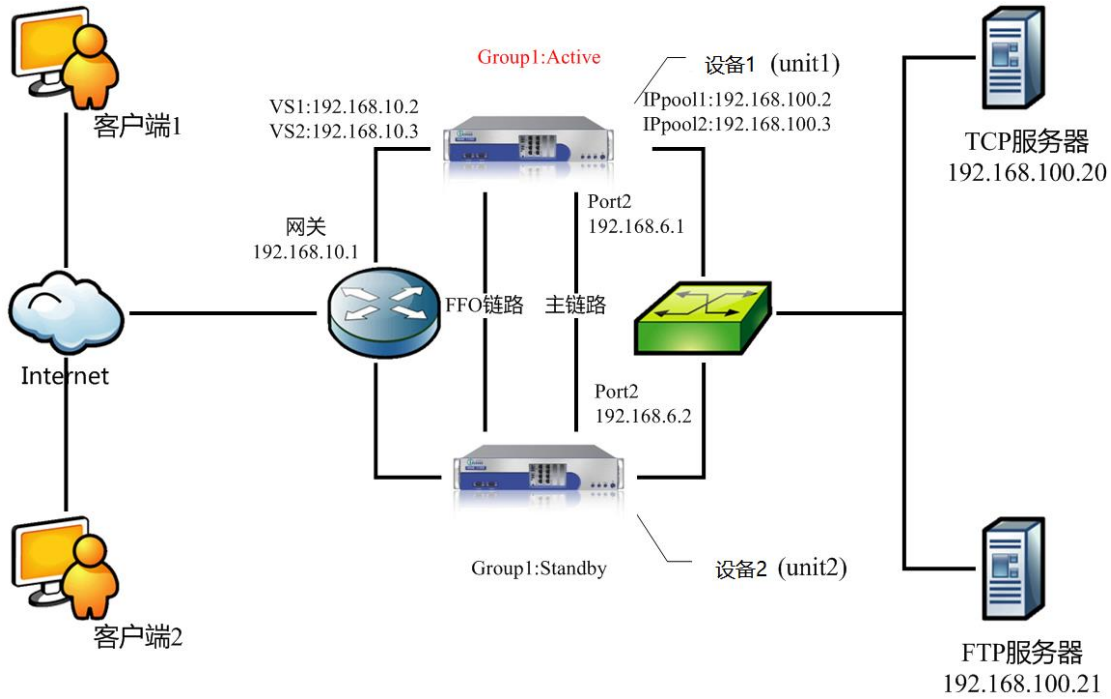

图9–1 Active/Standby 部署场景

9.8.1.2 配置示例

- **设备 1 上的配置：**

1. 执行如下命令添加 SLB 配置：

```
Demo1(config)#slb real tcp "r1" 192.168.100.20 80 65535 tcp 3 3
Demo1(config)#slb real ftp "r2" 192.168.100.21 21 65535 tcp 3 3
Demo1(config)#slb group method "slb_g1" rr
Demo1(config)#slb group method "slb_g2" rr
Demo1(config)#slb group member "slb_g1" "r1" 1 0
Demo1(config)#slb group member "slb_g2" "r2" 1 0
Demo1(config)#slb virtual tcp "v1" 192.168.10.2 80 arp 0
Demo1(config)#slb virtual ftp "v2" 192.168.10.3 21 0
Demo1(config)#slb policy default "v1" "slb_g1"
Demo1(config)#slb policy default "v2" "slb_g2"
Demo1(config)#ip pool p1 192.168.100.2 192.168.100.2
Demo1(config)#ip pool p2 192.168.100.3 192.168.100.3
Demo1(config)#slb proxyip group slb_g1 p1
Demo1(config)#slb proxyip group slb_g2 p2
```

20. 执行如下命令添加 HA 节点，同步节点和链路配置：

```
Demo1(config)#ha unit "unit1" 192.168.6.1 65521
Demo1(config)#ha unit "unit2" 192.168.6.2 65521
Demo1(config)#synconfig peer "unit1" 192.168.6.1
Demo1(config)#synconfig peer "unit2" 192.168.6.2
Demo1(config)#ha link network on
Demo1(config)#ha link ffo on
```

21. 执行如下命令添加浮动 IP 分组配置：

```
Demo1(config)#ha group id 1
Demo1(config)#ha group fip 1 192.168.10.2 port1
Demo1(config)#ha group fip 1 192.168.10.3 port1
Demo1(config)#ha group fip 1 192.168.100.2 port3
Demo1(config)#ha group fip 1 192.168.100.3 port3
Demo1(config)#ha group priority unit1 1 10
Demo1(config)#ha group priority unit2 1 5
Demo1(config)#ha group preempt on 1
Demo1(config)#ha group enable 1
```

22. （可选）执行如下命令添加健康检查条件。下面以网关和 CPU 使用率的健康检查配置为例。

```
Demo1(config)#monitor network gateway unit1 192.168.10.1 GATEWAY_1 1000 3 3
Demo1(config)#monitor network gateway unit2 192.168.10.1 GATEWAY_1 1000 3 3
Demo1(config)#monitor system cpu utilization 90 5000 3 3
Demo1(config)#monitor vcondition name vcondition1 VCONDITION_1 AND
Demo1(config)#monitor vcondition member vcondition1 GATEWAY_1
Demo1(config)#monitor vcondition member vcondition1 CPU_UTIL
```

23. （可选）执行如下命令添加失效切换规则：

```
Demo1(config)#ha decision rule vcondition1 Group_Failover 1
```

24. （可选）执行如下命令启用 SSF 功能：

```
Demo1(config)#ha ssf peer 192.168.6.2
Demo1(config)#ha ssf on
```

25. （可选）执行如下命令设置配置同步模式：

```
Demo1(config)#ha synconfig bootup on
Demo1(config)#ha synconfig runtime on
```

26. （可选）执行如下命令启用 HA 日志功能：

```
Demo1(config)#ha log on
```

27. 执行如下命令启用 HA 功能并保存 HA 的相关配置到内存中：

```
Demo1(config)#ha on
Demo1(config)#write memory
```

- **设备 2 上的配置：**

在 Active/Standby 场景中，推荐使用 FFO 链路和主链路从对端节点同步相关配置信息。

1. 执行如下命令配置同步节点：

```
Demo2(config)#synconfig peer "unit1" 192.168.6.1
Demo2(config)#synconfig peer "unit2" 192.168.6.2
```

28. 执行如下命令启用网络链路、FFO 链路和启动时配置同步模式：

```
Demo2(config)#ha link network on
Demo2(config)#ha link ffo on
Demo2(config)#ha synconfig bootup on
```

29. （可选）执行如下命令启用 SSF 功能：

```
Demo2(config)#ha ssf peer 192.168.6.1
Demo2(config)#ha ssf on
```

30. 执行如下命令启用 HA 功能：

```
Demo2(config)#ha on
```

一旦启用 HA 功能，unit2 节点（设备 2 设备）就加入到了 HA 域中。unit2 首先通过 FFO 链路从 unit1 节点（设备 1 设备）同步 HA 节点、FFO 链路和网络链路的配置信息，然后通过主链路同步其它配置。

9.8.2 场景 2：Active/Active

9.8.2.1 配置目标

Active/Active 部署场景可以用来实现如下配置目标：

- HA 域包含两个 HA 节点，提供两个浮动 IP 分组。
- 每个浮动 IP 分组中包含一个应用服务的 VIP 地址。
- 节点 1 提供分组 1 的应用服务，节点 2 提供分组 2 的应用服务。节点 1 和节点 2 相互提供备份。
- 使用 FFO 链路做快速失效切换。

实现如上配置目标的网络拓扑图如下图所示。

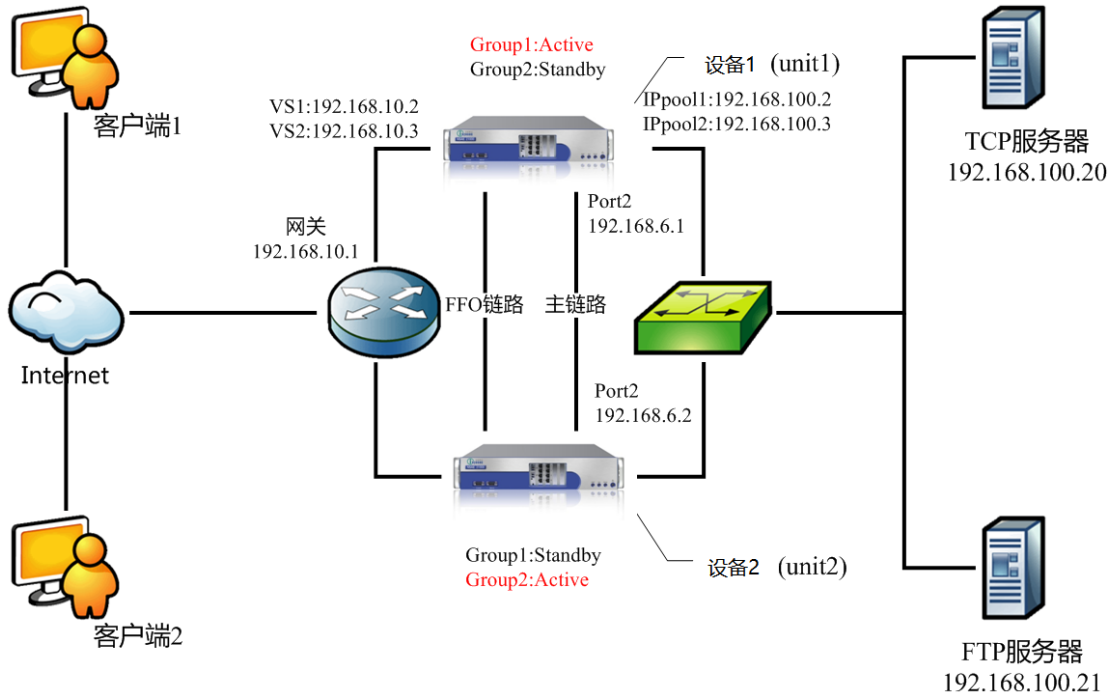

图9–2 Active/Active 部署场景

注意：在该场景下，如果为节点 1 和节点 2 上的后台服务组配置代理 IP 地址池（通过命令“slb proxyip group”），并且代理 IP 地址池中含有相同的 IP 地址，系统会从代理 IP 地址池中选择一个 IP 与后台服务通信。如果两个节点使用了同一个 IP 地址，可能会出现后台服务将响应返回给节点 1 的 SYN、ACK 包返回给备机，或者反过来，从而导致系统无法正常工作。

建议在该场景下且配置了代理 IP 地址池时：

- 节点 1 和节点 2 使用不同代理 IP 地址池时，两个代理 IP 地址池中不能有相同 IP 地址。
- 节点 1 和节点 2 共用相同的代理 IP 地址池时，节点 1 和节点 2 不要部署为 Active/Active 场景。

9.8.2.2 配置示例

➢ **设备 1 上的配置：**

1. 执行如下命令添加 SLB 配置：

```
Demo(config)#slb real tcp "r1" 192.168.100.20 80 65535 tcp 3 3
Demo(config)#slb real ftp "r2" 192.168.100.21 21 65535 tcp 3 3
Demo(config)#slb group method "slb_g1" rr
Demo(config)#slb group method "slb_g2" rr
Demo(config)#slb group member "slb_g1" "r1" 1 0
Demo(config)#slb group member "slb_g2" "r2" 1 0
Demo(config)#slb virtual tcp "v1" 192.168.10.2 80 arp 0
Demo(config)#slb virtual ftp "v2" 192.168.10.3 21 0
Demo(config)#slb policy default "v1" "slb_g1"
Demo(config)#slb policy default "v2" "slb_g2"
Demo(config)#ip pool p1 192.168.100.2 192.168.100.2
Demo(config)#ip pool p2 192.168.100.3 192.168.100.3
Demo(config)#slb proxyip group slb_g1 p1
Demo(config)#slb proxyip group slb_g2 p2
```

31. 执行如下命令添加 HA 节点，同步节点和链路配置：

```
Demo(config)#ha unit "unit1" 192.168.6.1 65521
Demo(config)#ha unit "unit2" 192.168.6.2 65521
Demo(config)#synconfig peer "unit1" 192.168.6.1
Demo(config)#synconfig peer "unit2" 192.168.6.2
Demo(config)#ha link network on
Demo(config)#ha link ffo on
```

32. 执行如下命令添加浮动 IP 分组配置：

```
Demo(config)#ha group id 1
Demo(config)#ha group fip 1 192.168.10.2 port1
Demo(config)#ha group fip 1 192.168.100.2 port3
Demo(config)#ha group priority unit1 1 10
Demo(config)#ha group priority unit2 1 5
Demo(config)#ha group preempt on 1
Demo(config)#ha group enable 1

Demo(config)#ha group id 2
Demo(config)#ha group fip 2 192.168.10.3 port1
Demo(config)#ha group fip 2 192.168.100.3 port3
Demo(config)#ha group priority unit1 2 5
Demo(config)#ha group priority unit2 2 10
Demo(config)#ha group preempt on 2
Demo(config)#ha group enable 2
```

33. （可选）执行如下命令添加健康检查条件。下面以网关和 CPU 使用率的健康检查配置为例。

```
Demo(config)#monitor network gateway unit1 192.168.10.1 GATEWAY_1 1000 3 3
Demo(config)#monitor network gateway unit2 192.168.10.1 GATEWAY_1 1000 3 3
Demo(config)#monitor system cpu utilization 90 5000 3 3
Demo(config)#monitor vcondition name vcondition1 VCONDITION_1 AND
Demo(config)#monitor vcondition member vcondition1 GATEWAY_1
Demo(config)#monitor vcondition member vcondition1 CPU_UTIL
```

34. （可选）执行如下命令添加失效切换规则：

```
Demo(config)#ha decision rule vcondition1 Unit_Failover
```

35. （可选）执行如下命令启用 SSF 功能：

```
Demo(config)#ha ssf peer 192.168.6.2
Demo(config)#ha ssf on
```

36. （可选）执行如下命令设置配置同步模式：

```
Demo(config)#ha synconfig bootup on
Demo(config)#ha synconfig runtime on
```

37. （可选）执行如下命令启用 HA 日志功能：

```
Demo(config)#ha log on
```

38. 执行如下命令启用 HA 功能并保存 HA 的相关配置到内存中：

```
Demo(config)#ha on
Demo(config)#write memory
```

- **设备 2 上的配置：**

在 Active/Active 场景中，推荐使用 FFO 链路和主链路从对端节点同步相关配置信息。

1. 执行如下命令配置同步节点：

```
Demo(config)#synconfig peer "unit1" 192.168.6.1
Demo(config)#synconfig peer "unit2" 192.168.6.2
```

39. 执行如下命令启用网络链路、FFO 链路和启动时配置同步模式：

```
Demo(config)#ha link network on
Demo(config)#ha link ffo on
Demo(config)#ha synconfig bootup on
```

40. （可选）执行如下命令启用 SSF 功能：

```
Demo(config)#ha ssf peer 192.168.6.1
Demo(config)#ha ssf on
```

41. 执行如下命令启用 HA 功能：

```
Demo(config)#ha on
```

一旦启用 HA 功能，unit2 节点（设备 2 设备）就加入到了 HA 域中。unit2 首先通过 FFO 链路从 unit1 节点（设备 1 设备）同步 HA 节点、FFO 链路和网络链路的配置信息，然后通过主链路同步其它配置。

9.8.3 场景 3：N+1

在 N+1 部署场景中，HA 域包含 N+1 个节点，在其中 N 个节点上，浮动 IP 分组的状态都为“Active”，在另外一个节点上，所有浮动 IP 分组的状态都为“Standby”。本节将介绍“3+1”部署场景的配置目标和配置示例。

9.8.3.1 配置目标

“3+1”部署场景可以用来实现如下配置目标：

- HA 域包含四个节点，提供三个浮动 IP 分组。
- 每个浮动 IP 分组中包含一个虚拟服务的 VIP 地址。
- 节点 1~3 分别提供分组 1~3 的虚拟服务，节点 4 为节点 1~3 提供备份。

实现如上配置目标的网络拓扑图如下图所示。

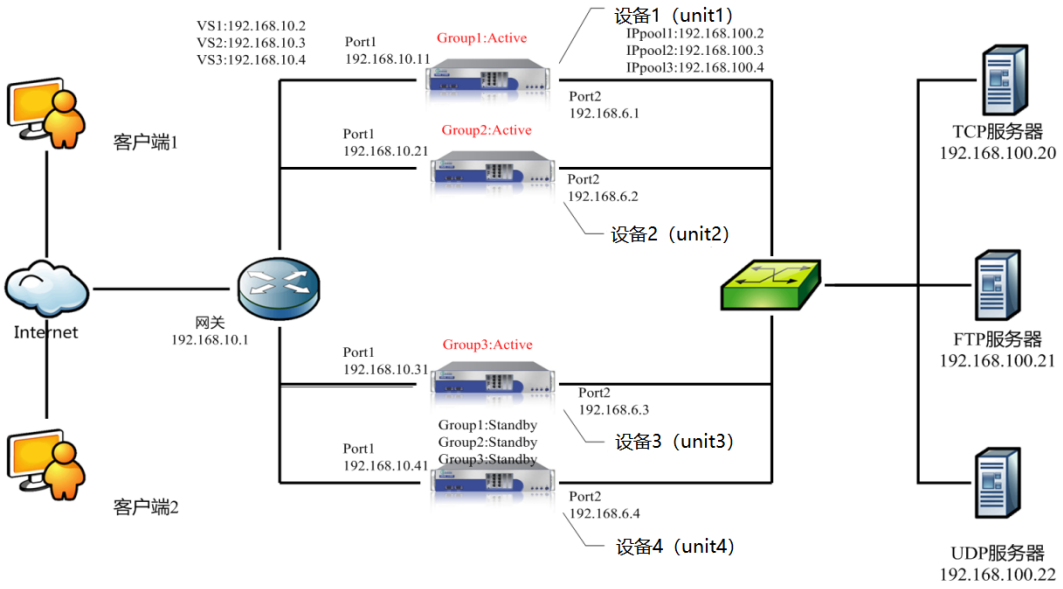

图9–3 N+1 部署场景

9.8.3.2 配置示例

- **设备 1 上的配置：**

1. 执行如下命令添加 SLB 配置：

```
Demo1(config)#slb real tcp "r1" 192.168.100.20 80 65535 tcp 3 3
Demo1(config)#slb real ftp "r2" 192.168.100.21 21 65535 tcp 3 3
Demo1(config)#slb real udp "r3" 192.168.100.22 53 65535 3 3 60 icmp
Demo1(config)#slb group method "slb_g1" rr
Demo1(config)#slb group method "slb_g2" rr
Demo1(config)#slb group method "slb_g3" rr
Demo1(config)#slb group member "slb_g1" "r1" 1 0
Demo1(config)#slb group member "slb_g2" "r2" 1 0
Demo1(config)#slb group member "slb_g3" "r3" 1 0
Demo1(config)#slb virtual tcp "v1" 192.168.10.2 80 arp 0
Demo1(config)#slb virtual ftp "v2" 192.168.10.3 21 0
Demo1(config)#slb virtual udp "v3" 192.168.10.4 53 arp 0
Demo1(config)#slb policy default "v1" "slb_g1"
Demo1(config)#slb policy default "v2" "slb_g2"
Demo1(config)#slb policy default "v3" "slb_g3"
Demo1(config)#ip pool p1 192.168.100.2 192.168.100.2
Demo1(config)#ip pool p2 192.168.100.3 192.168.100.3
Demo1(config)#ip pool p3 192.168.100.4 192.168.100.4
Demo1(config)#slb proxyip group slb_g1 p1
Demo1(config)#slb proxyip group slb_g2 p2
Demo1(config)#slb proxyip group slb_g3 p3
```

42. 执行如下命令添加 HA 节点，同步节点和链路配置：

```
Demo1(config)#ha unit "unit1" 192.168.6.1 65521
Demo1(config)#ha unit "unit2" 192.168.6.2 65521
Demo1(config)#ha unit "unit3" 192.168.6.3 65521
Demo1(config)#ha unit "unit4" 192.168.6.4 65521
Demo1(config)#synconfig peer "unit1" 192.168.6.1
Demo1(config)#synconfig peer "unit2" 192.168.6.2
Demo1(config)#synconfig peer "unit3" 192.168.6.3
Demo1(config)#synconfig peer "unit4" 192.168.6.4
Demo1(config)#ha link network secondary unit1 1 192.168.10.11
Demo1(config)#ha link network secondary unit2 1 192.168.10.21
Demo1(config)#ha link network secondary unit3 1 192.168.10.31
Demo1(config)#ha link network secondary unit4 1 192.168.10.41
Demo1(config)#ha link network on
```

43. 执行如下命令添加浮动 IP 分组配置：

```
Demo1(config)#ha group id 1
Demo1(config)#ha group fip 1 192.168.10.2 port1
Demo1(config)#ha group fip 1 192.168.100.2 port3
Demo1(config)#ha group priority unit1 1 200
Demo1(config)#ha group priority unit2 1 100
Demo1(config)#ha group priority unit3 1 50
Demo1(config)#ha group priority unit4 1 150
Demo1(config)#ha group preempt on 1
Demo1(config)#ha group enable 1

Demo1(config)#ha group id 2
Demo1(config)#ha group fip 2 192.168.10.3 port1
Demo1(config)#ha group fip 2 192.168.100.3 port3
Demo1(config)#ha group priority unit1 2 50
Demo1(config)#ha group priority unit2 2 200
Demo1(config)#ha group priority unit3 2 100
Demo1(config)#ha group priority unit4 2 150
Demo1(config)#ha group preempt on 2
Demo1(config)#ha group enable 2

Demo1(config)#ha group id 3
Demo1(config)#ha group fip 3 192.168.10.4 port1
Demo1(config)#ha group fip 3 192.168.100.4 port3
Demo1(config)#ha group priority unit1 3 100
Demo1(config)#ha group priority unit2 3 50
Demo1(config)#ha group priority unit3 3 200
```

```
Demo1(config)#ha group priority unit4 3 150
Demo1(config)#ha group preempt on 3
Demo1(config)#ha group enable 3
```

44. （可选）执行如下命令添加健康检查条件。下面以网关和 CPU 使用率的健康检查配置为例。

```
Demo1(config)#monitor network gateway unit1 192.168.10.1 GATEWAY_1 1000 3 3
Demo1(config)#monitor network gateway unit2 192.168.10.1 GATEWAY_1 1000 3 3
Demo1(config)#monitor network gateway unit3 192.168.10.1 GATEWAY_1 1000 3 3
Demo1(config)#monitor network gateway unit4 192.168.10.1 GATEWAY_1 1000 3 3
Demo1(config)#monitor system cpu utilization 90 5000 3 3
Demo1(config)#monitor vcondition name vcondition1 VCONDITION_1 AND
Demo1(config)#monitor vcondition member vcondition1 GATEWAY_1
Demo1(config)#monitor vcondition member vcondition1 CPU_UTIL
```

45. （可选）执行如下命令添加失效切换规则：

```
Demo1(config)#ha decision rule vcondition1 Unit_Failover
```

46. （可选）执行如下命令设置配置同步模式：

```
Demo1(config)#ha synconfig bootup on
Demo1(config)#ha synconfig runtime on
```

47. （可选）执行如下命令启用 HA 日志功能。

```
Demo1(config)#ha log on
```

48. 执行如下命令启用 HA 功能并保存 HA 相关配置到内存中：

```
Demo1(config)#ha on
Demo1(config)#write memory
```

- **设备 2 上的配置：**

1. 执行如下命令添加 HA 节点，同步节点和链路配置：

```
Demo2(config)#ha unit "unit1" 192.168.6.1 65521
Demo2(config)#ha unit "unit2" 192.168.6.2 65521
Demo2(config)#ha unit "unit3" 192.168.6.3 65521
Demo2(config)#ha unit "unit4" 192.168.6.4 65521
Demo2(config)#synconfig peer "unit1" 192.168.6.1
Demo2(config)#synconfig peer "unit2" 192.168.6.2
Demo2(config)#synconfig peer "unit3" 192.168.6.3
Demo2(config)#synconfig peer "unit4" 192.168.6.4
Demo2(config)#ha link network secondary unit1 1 192.168.10.11
Demo2(config)#ha link network secondary unit2 1 192.168.10.21
```

```
Demo2(config)#ha link network secondary unit3 1 192.168.10.31
Demo2(config)#ha link network secondary unit4 1 192.168.10.41
Demo2(config)#ha link network on
```

49. 执行如下命令启用启动时配置同步模式：

```
Demo2(config)#ha synconfig bootup on
```

50. 执行如下命令启用 HA 功能：

```
Demo2(config)#ha on
```

一旦启用 HA 功能，unit2 节点（设备 2 设备）就加入到了 HA 域中，并开始从 unit1 节点（设备 1 设备）同步配置。

- **设备 3 上的配置：**

1. 执行如下命令添加 HA 节点，同步节点和链路配置：

```
Demo3(config)#ha unit "unit1" 192.168.6.1 65521
Demo3(config)#ha unit "unit2" 192.168.6.2 65521
Demo3(config)#ha unit "unit3" 192.168.6.3 65521
Demo3(config)#ha unit "unit4" 192.168.6.4 65521
Demo3(config)#synconfig peer "unit1" 192.168.6.1
Demo3(config)#synconfig peer "unit2" 192.168.6.2
Demo3(config)#synconfig peer "unit3" 192.168.6.3
Demo3(config)#synconfig peer "unit4" 192.168.6.4
Demo3(config)#ha link network secondary unit1 1 192.168.10.11
Demo3(config)#ha link network secondary unit2 1 192.168.10.21
Demo3(config)#ha link network secondary unit3 1 192.168.10.31
Demo3(config)#ha link network secondary unit4 1 192.168.10.41
Demo3(config)#ha link network on
```

51. 执行如下命令启用启动时配置同步模式：

```
Demo3(config)#ha synconfig bootup on
```

52. 执行如下命令启用 HA 功能：

```
Demo3(config)#ha on
```

一旦启用 HA 功能，unit3 节点（设备 3 设备）就加入到了 HA 域中，并开始从 unit1 节点（设备 1 设备）同步配置。

- **设备 4 上的配置：**

1. 执行如下命令添加 HA 节点，同步节点和链路配置：

```
Demo4(config)#ha unit "unit1" 192.168.6.1 65521
Demo4(config)#ha unit "unit2" 192.168.6.2 65521
```

```
Demo4(config)#ha unit "unit3" 192.168.6.3 65521
Demo4(config)#ha unit "unit4" 192.168.6.4 65521
Demo4(config)#synconfig peer "unit1" 192.168.6.1
Demo4(config)#synconfig peer "unit2" 192.168.6.2
Demo4(config)#synconfig peer "unit3" 192.168.6.3
Demo4(config)#synconfig peer "unit4" 192.168.6.4
Demo4(config)#ha link network secondary unit1 1 192.168.10.11
Demo4(config)#ha link network secondary unit2 1 192.168.10.21
Demo4(config)#ha link network secondary unit3 1 192.168.10.31
Demo4(config)#ha link network secondary unit4 1 192.168.10.41
Demo4(config)#ha link network on
```

53. 执行如下命令启用启动时配置同步模式：

```
Demo4(config)#ha synconfig bootup on
```

54. 执行如下命令启用 HA 功能：

```
Demo4(config)#ha on
```

一旦启用 HA 功能，unit4 节点（设备 4 设备）就加入到了 HA 域中，并开始从 unit1 节点（设备 1 设备）同步配置。

第10章 单系统映像（SSI）

10.1 概述

SSI 是指一组设备作为单一系统对外提供服务。从逻辑上，SSI 部署中的多台设备被看作同一台设备。

SSI 功能使管理员可以通过平行部署多台设备同时提供同一个虚拟服务，从而满足用户更高的性能需求。

10.2 基本概念

SSI 与 HA 拥有相同的构架，因而具有相同的基本概念，如域、节点、浮动 IP 分组、通信链路等。

注意：SSI 功能与 HA 功能互斥，无法同时部署。

10.3 工作原理

SSI 的配置需要高交换带宽交换机（须支持 LACP 协议）的配合。SSI 支持双臂模式的服务器负载均衡，以下为其工作原理。

注意：SSI 功能只支持 SLB 的反向代理模式。

如 SSI 工作原理所示，SSI 部署中的数据流向如下：

1. 客户发出的数据包到达交换机 1；
2. 交换机 1 将数据包传送给设备（由于 SSI 域中的设备具有相同的虚拟服务地址以及浮动 MAC 的配置，数据将由交换机根据特定算法，如根据源 IP 和源 Port 进行哈希，发送给 SSI 域中的某台设备）；
3. 设备将数据包通过交换机 2 传给后台服务器；
4. 后台服务器将返回数据包发给交换机 2；
5. 交换机 2 将数据包返回给设备（由于各设备上配置的地址池不同，数据包将据此返回进行数据处理的特定设备）；
6. 设备通过交换机 1 将数据包返回给客户。

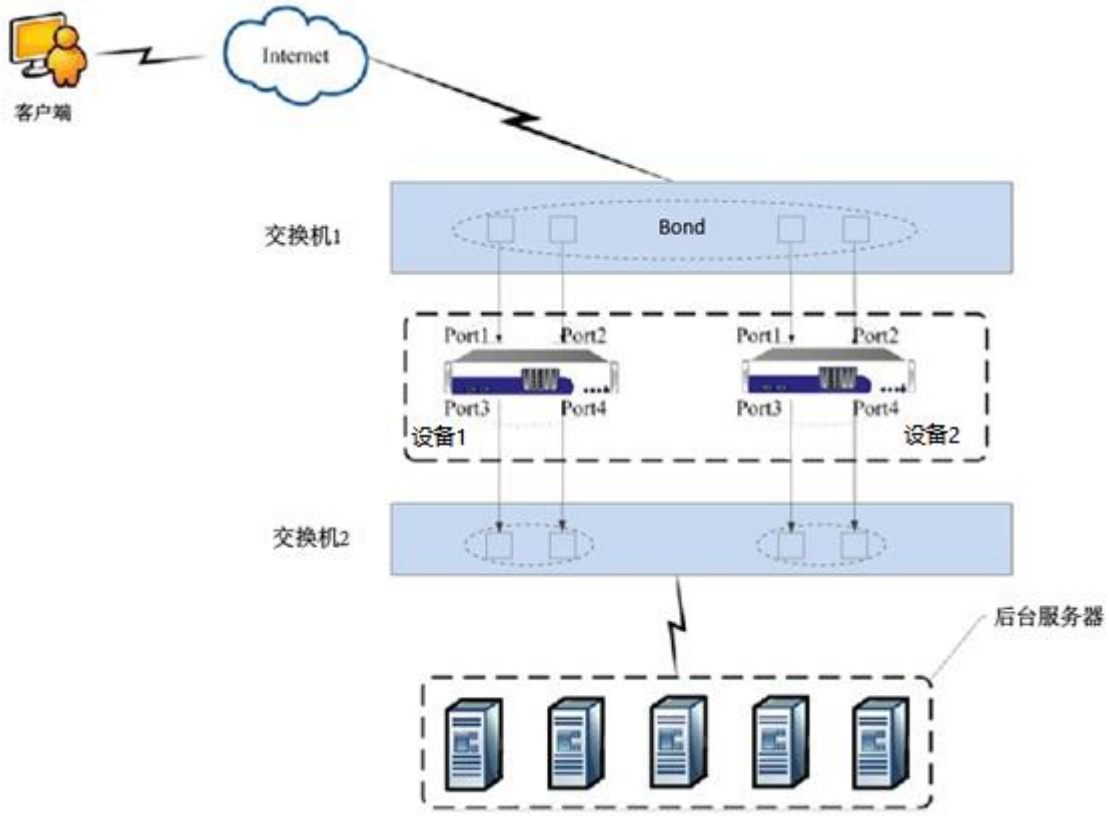

图10–1 SSI 工作原理

如上图所示，在 SSI 的配置中需要注意以下方面：

- 将交换机 1 与设备入向接口相连的接口绑定到一个聚合接口上。
- 将每台设备的入向接口绑定到一个聚合接口上。
- 将交换机 2（内网交换机）与设备 1 出向接口相连的接口绑定到一个聚合接口上。
- 将交换机 2 与设备 2 出向接口相连的接口绑定到一个聚合接口上。
- 将每台设备的出向接口绑定到一个聚合接口上。
- 给每台设备的 SLB 组配置不同的地址池作为代理 IP。

注意：SSI 各节点的入向接口均为业务接口，不能用作系统管理，如建立 SSH 连接，WebUI 连接以及系统升级。若需要对各个节点单独进行管理，则应另外指定单独的管理接口。

10.4 交换机失效切换

SSI 的交换机切换功能向与客户端通信的交换机提供备份交换机，并在当前交换机失效情况下实现交换机失效切换。当设备与交换机之间的链路异常，或者当管

理员定义的其他监控条件出现问题时，根据管理员定义的切换规则就可以实现交换机的失效切换。

交换机失效切换功能对客户端到 SSI 域之间的交换机提供的故障切换支持，使得交换机出现断电、链路断开等问题时，SSI 域各节点可以将虚拟服务自动切换到备份交换机上，保证业务的正常运行。

SSI 功能提供交换机切换功能的所需配置，如下图所示：

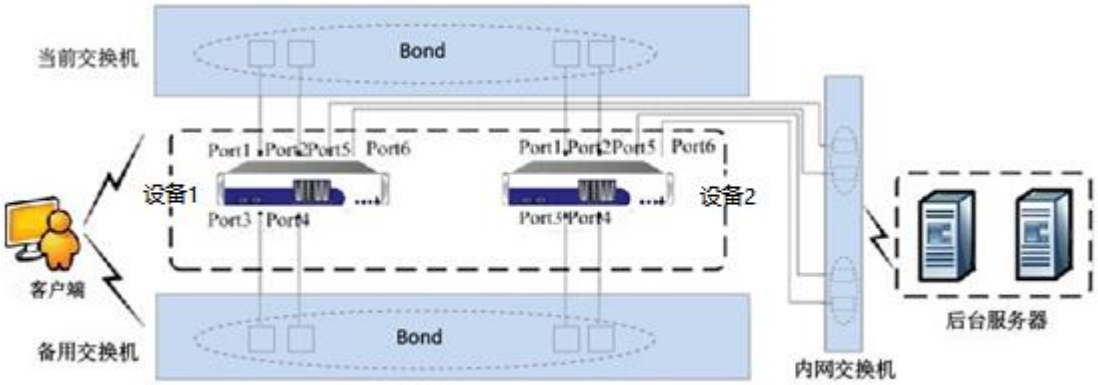

图10–2 交换机失效切换原理

交换机失效切换的配置需要注意以下方面：

- 两台交换机配置及其与设备相连接口配置是相同的，包括聚合接口配置、LACP 协议启用、流量分配机制设定等。
- 将图示中与两个交换机相连的接口分别加入两个浮动 IP 分组中，即将设备 1 的 port1、port2 和设备 2 的 port1、port2 加入 group1 中，将设备 1 的 port3、port4 和设备 2 的 port3、port4 加入 group2 中。

10.5 配置示例

10.5.1 双臂 SLB 的 SSI 配置

为了使配置达到 SSI 工作原理的效果需要进行以下配置项目：

- 对交换机进行配置；
- 对聚合接口和接口 IP 进行配置，并配置心跳线，使各节点的心跳接口处于同一个广播域内；
- 配置 SLB 功能；
- 配置 SSI 功能。

注意：SSI 功能要求每台设备的配置除代理 IP 以外完全相同。

具体配置如下：

- **交换机的配置：**

- 将设备 1 的 port1、port2 与交换机 1 连接，port3、port4 与交换机 2 连接。
- 将设备 2 的 port1、port2 与交换机 1 连接，port3、port4 与交换机 2 连接。
- 将交换机 1 上与设备 1 的 port1、port2 及设备 2 的 port1、port2 相连的接口加入同一聚合接口。
- 在交换机 1 的该聚合接口上启用 LACP 协议。
- 在交换机 1 的该聚合接口上启用流量分配（如根据源 IP 和源 Port 进行哈希）。
- 将交换机 2 上与设备 1 的 port3、port4 相连的接口加入同一聚合接口。
- 将交换机 2 上与设备 2 的 port3、port4 相连的接口加入同一聚合接口。

- **设备上的聚合接口配置和接口配置：**

```
#设备 1
Demo(config)#bond interface bond1 port1
Demo(config)#bond interface bond1 port2
Demo(config)#bond interface bond2 port3
Demo(config)#bond interface bond2 port4
Demo(config)#ip address bond1 100.8.1.1 255.255.255.0
Demo(config)#ip address bond2 192.168.1.1 255.255.255.0
#heartbeat configuration for SSI domain
Demo(config)#ip address port5 172.16.1.1 255.255.255.0

#设备 2
Demo(config)#bond interface bond1 port1
Demo(config)#bond interface bond1 port2
Demo(config)#bond interface bond2 port3
Demo(config)#bond interface bond2 port4
Demo(config)#ip address bond1 100.8.1.1 255.255.255.0
Demo(config)#ip address bond2 192.168.1.2 255.255.255.0
#heartbeat configuration for SSI domain
Demo(config)#ip address port5 172.16.1.2 255.255.255.0
```

- **设备上的 SLB 配置：**

```
#设备 1
Demo(config)#slb real tcp "r1" 192.168.1.100 80 65535 tcp 3 3
```

```
Demo(config)#slb real tcp "r2" 192.168.1.200 80 65535 tcp 3 3
Demo(config)#slb group method "slb_g1" rr
Demo(config)#slb group member "slb_g1" "r1" 1 0
Demo(config)#slb group member "slb_g1" "r2" 1 0
Demo(config)#slb virtual tcp "v1" 100.8.1.20 80 arp 0
Demo(config)#slb policy default "v1" "slb_g1"
Demo(config)#ip pool p1 192.168.1.3 192.168.1.3
Demo(config)#slb proxyip group slb_g1 p1

#设备 2
Demo(config)#slb real tcp "r1" 192.168.1.100 80 65535 tcp 3 3
Demo(config)#slb real tcp "r2" 192.168.1.200 80 65535 tcp 3 3
Demo(config)#slb group method "slb_g1" rr
Demo(config)#slb group member "slb_g1" "r1" 1 0
Demo(config)#slb group member "slb_g1" "r2" 1 0
Demo(config)#slb virtual tcp "v1" 100.8.1.20 80 arp 0
Demo(config)#slb policy default "v1" "slb_g1"
Demo(config)#ip pool p1 192.168.1.4 192.168.1.4
Demo(config)#slb proxyip group slb_g1 p1
```

- **设备上的 SSI 配置：**

```
#设备 1 & 设备 2
Demo(config)#ha ssi on
Demo(config)#ha unit 设备_1 172.16.1.1 65521
Demo(config)#ha unit 设备_2 172.16.1.2 65521

Demo(config)#ha group id 1
Demo(config)#ha group port 1 bond1 port1
Demo(config)#ha group port 1 bond1 port2
#configuring the priority of a specified floating IP group on all units in the SSI domain
Demo(config)#ha group priority ssi 1 100
Demo(config)#ha group enable 1

Demo(config)#monitor network gateway 设备_1 192.168.1.30 GATEWAY_1 1000 3 3
Demo(config)# monitor network gateway 设备_2 192.168.1.30 GATEWAY_1 1000 3 3
Demo(config)#ha decision rule "GATEWAY_1" Group_Failover 1
Demo(config)#ha on
```

10.5.2 交换机失效切换的 SSI 配置

为了使配置达到交换机失效切换原理的效果需要进行以下配置项目：

- 对交换机进行配置。

- 对聚合接口和接口 IP 进行配置，并配置心跳线，使各节点的心跳接口处于同一个广播域内。
- 配置 SLB 功能。
- 配置 SSI 功能。

具体配置如下：

- **交换机的配置：**

- 将设备 1 的 port1、port2 与当前交换机连接，port3、port4 与备用交换机连接。
- 将设备 2 的 port1、port2 与当前交换机连接，port3、port4 与备用交换机连接。
- 将设备 1 的 port5、port6 与内网交换机连接。
- 将设备 2 的 port5、port6 与内网交换机连接。
- 将当前交换机上与设备 1 的 port1、port2 及设备 2 的 port1、port2 相连的接口加入同一聚合接口。
- 将备用交换机上与设备 1 的 port3、port4 及设备 2 的 port3、port4 相连的接口加入同一聚合接口。
- 在当前交换机/备用交换机的该聚合接口上启用 LACP 协议。
- 在当前交换机/备用交换机的该聚合接口上启用流量分配。
- 将内网交换机上与设备 1 的 port5、port6 相连的接口加入同一聚合接口。
- 将内网交换机上与设备 2 的 port5、port6 相连的接口加入同一聚合接口。

- **设备上的聚合接口配置和接口配置**

```
#设备 1
Demo1(config)#bond interface bond1 port1
Demo1(config)#bond interface bond1 port2
Demo1(config)#bond interface bond1 port3
Demo1(config)#bond interface bond1 port4
Demo1(config)#bond interface bond2 port5
Demo1(config)#bond interface bond2 port6
Demo1(config)#ip address bond1 100.8.1.1 255.255.255.0
Demo1(config)#ip address bond2 192.168.1.1 255.255.255.0
#heartbeat configuration for SSI domain
Demo1(config)#ip address port7 172.16.1.1 255.255.255.0

#设备 2
Demo2(config)#bond interface bond1 port1
Demo2(config)#bond interface bond1 port2
```

```
Demo2(config)#bond interface bond1 port3
Demo2(config)#bond interface bond1 port4
Demo2(config)#bond interface bond2 port5
Demo2(config)#bond interface bond2 port6
Demo2(config)#ip address bond1 100.8.1.1 255.255.255.0
Demo2(config)#ip address bond2 192.168.1.2 255.255.255.0
#heartbeat configuration for SSI domain
Demo2(config)#ip address port7 172.16.1.2 255.255.255.0
```

- **设备上的 SLB 配置：**

```
#设备 1
Demo1(config)#slb real tcp "r1" 192.168.1.100 80 65535 tcp 3 3
Demo1(config)#slb real tcp "r2" 192.168.1.200 80 65535 tcp 3 3
Demo1(config)#slb group method "slb_g1" rr
Demo1(config)#slb group member "slb_g1" "r1" 1 0
Demo1(config)#slb group member "slb_g1" "r2" 1 0
Demo1(config)#slb virtual tcp "v1" 100.8.1.20 80 arp 0
Demo1(config)#slb policy default "v1" "slb_g1"
Demo1(config)#ip pool p1 192.168.1.3 192.168.1.3
Demo1(config)#slb proxyip group slb_g1 p1

#设备 2
Demo2(config)#slb real tcp "r1" 192.168.1.100 80 65535 tcp 3 3
Demo2(config)#slb real tcp "r2" 192.168.1.200 80 65535 tcp 3 3
Demo2(config)#slb group method "slb_g1" rr
Demo2(config)#slb group member "slb_g1" "r1" 1 0
Demo2(config)#slb group member "slb_g1" "r2" 1 0
Demo2(config)#slb virtual tcp "v1" 100.8.1.20 80 arp 0
Demo2(config)#slb policy default "v1" "slb_g1"
Demo2(config)#ip pool p1 192.168.1.4 192.168.1.4
Demo2(config)#slb proxyip group slb_g1 p1
```

- **设备上的 SSI 配置：**

```
#设备 1 &  设备 2
Demo(config)#ha ssi on
Demo(config)#ha unit  设备_1 172.16.1.1 65521
Demo(config)#ha unit  设备_2 172.16.1.2 65521

Demo(config)#ha group id 1
Demo(config)#ha group port 1 bond1 port1
Demo(config)#ha group port 1 bond1 port2
Demo(config)#ha group priority ssi 1 200
```

Demo(config)#**ha group enable 1**

Demo(config)#**ha group id 2**
Demo(config)#**ha group port 2 bond1 port3**
Demo(config)#**ha group port 2 bond1 port4**
Demo(config)#**ha group priority ssi 2 100**
Demo(config)#**ha group enable 2**

Demo(config)#**monitor network gateway 设备_1 192.168.1.30 GATEWAY_1 1000 3 3**
Demo(config)#**monitor network gateway 设备_2 192.168.1.30 GATEWAY_1 1000 3 3**
Demo(config)#**ha decision rule "GATEWAY_1" Group_Failover 1**
Demo(config)#**ha on**

第11章 服务器负载均衡（SLB）

11.1 概述

本章将介绍服务器负载均衡（Server Load Balance，SLB）的相关原理、概念和配置方法。

服务器负载均衡旨在把网络流量和负载均衡分配到服务器组中的各个服务器上。设备支持 OSI 模型中二层、三层、四层和七层的负载均衡。二层负载均衡基于网络接口。三层负载均衡基于服务器 IP 地址。四层负载均衡与 TCP 或 UDP 端口有关。七层负载均衡是基于应用层的信息，如 URL、HTTP 表头或 Cookie。最基本的服务器负载均衡配置的步骤如下：

1. 定义后台服务。

55. 定义后台服务组和负载均衡算法。

56. 将后台服务添加到后台服务组。

57. 为所有请求定义一个虚拟 IP 地址。

58. 配置负载均衡的策略将虚拟 IP 地址与后台服务组进行关联。

后台服务器、VIP 和虚拟服务这三者是在设备上配置 SLB 的基础。

- 后台服务器是部署了某些应用提供某种服务的应用服务器，负责处理来自客户端的请求。
- VIP 是一个虚拟 IP 地址，通常是可由外网用户访问的公网 IP 地址。它作为请求的入口，负责接受并向后台服务器转发来自外部用户的访问请求，并将后台服务器对请求的处理响应结果发回到客户端。
- 在四层和七层 SLB 的配置中，虚拟服务通常由一个 VIP/端口对表示。外部用户可以通过该虚拟服务获取其目标网络资源。例如，若客户端通过事先定义的 VIP 或者网站名获取其所需的资源，那么来自该客户端的请求将经由该 VIP 发送到由设备配置的后台服务器进行处理。此外，由于虚拟服务只向外公开一个虚拟 IP 地址，还可以实现对外部用户隐藏内部网络结构和后台服务器信息。

11.2 服务器负载均衡的功能原理和工作机制

设备的 SLB 服务器负载均衡由以下几个部分组成：

- 客户端
- 虚拟服务
- 虚拟 IP 地址

- 策略
- 服务组
- 负载均衡算法
- 后台服务

其中，虚拟服务、虚拟 IP 地址、策略、服务组和负载均衡算法这几个概念都是存在于设备中的虚拟概念，并非真实存在。

在一个典型的网络环境中，客户端以一个 IP 地址/端口向内网发起请求。这个 IP 地址被称为虚拟 IP 地址。这个请求会被送到设备上，设备会依照管理员的配置，根据用户请求中的虚拟 IP 地址，寻找到相应的策略和服务组。

服务组中可以包含一个或多个真实的后台服务器，而策略决定了虚拟 IP 地址同服务组的对应关系。于是设备根据策略找到相应的服务组，并将请求交给服务组处理。

服务组中既包含一个或多个真实的后台服务，同一个后台服务也可以属于多个服务组。同时又被各种负载均衡算法所定义。负载均衡算法规定了网络流量在服务组中的各个真实服务之间的分配方式。

设备根据服务组上的负载均衡算法，将用户的网络请求流量分配到后台的真实服务器上。整个过程如下图所示。

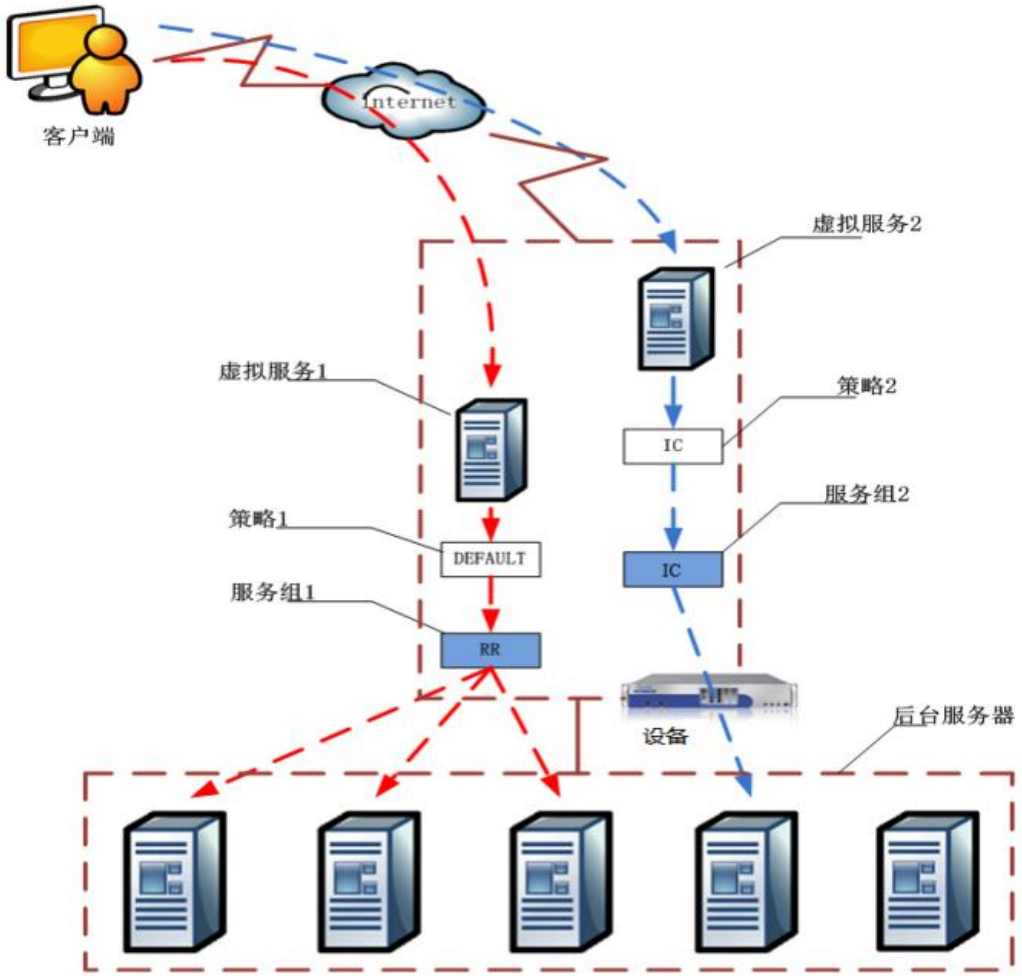

图11–1 服务器负载均衡工作原理

11.2.1 虚拟服务

虚拟服务由一个虚拟 IP 地址（VIP）和一个端口组成。客户将把请求发送到这个虚拟 IP 和端口对。之所以称为虚拟服务是因为真正的请求不是在此回应的，而是通过某种策略转发请求到后台服务组中的某个后台服务器上。换句话说，虚拟服务正是后台服务器在设备上的映像。如下图所示。

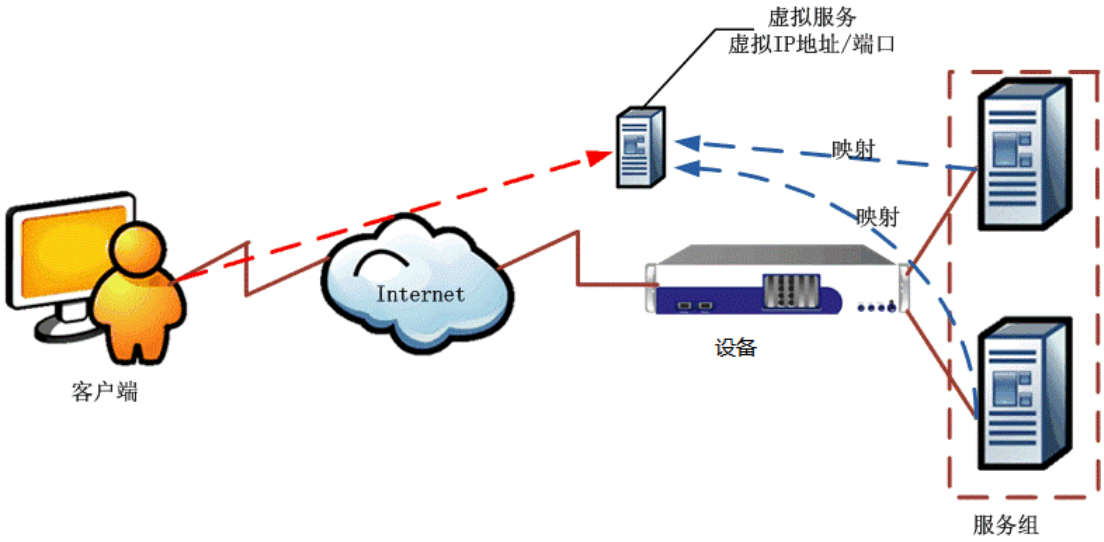

图11–2 虚拟服务

注意：虚拟 IP 地址不能和同一网卡上的系统 IP 地址相同。如果没有通过策略将虚拟服务与后台服务组绑定或者绑定的后台服务组里没有正常工作的服务，应答“503 Service Unavailable”将被发给客户表示服务不可用。

10.2.1.1 Proxy Protocol

系统已支持通过 Proxy Protocol 报文获取客户端真实 IP 地址。

TCP、TCPS、HTTP 和 HTTPS 协议类型的虚拟服务可以被可以设置为 Proxy Protocol 的报文接收者用于接收 v1 和 v2 版本的 Proxy Protocol 报文。在收到 Proxy Protocol 报文后，设备会从报文中提取客户端的源 IP 地址。对于 HTTP 或 HTTPS 类型的虚拟服务，该功能可以和命令“**http xforwardedfor on**”配合使用，将提取的客户端 IP 地址转发给后台服务。

➢ **配置示例**

- 设置虚拟服务为 Proxy Protocol 的报文接收者。

```
Demo(config)#slb virtual settings proxyprotocol vs1
```

11.2.2 后台服务组

后台服务组是一组相同类型的后台服务的集合，命中后台服务组的请求将被转发到根据设置的服务组算法选择的后台服务。

本小节介绍用来定义后台服务组、后台服务组算法、服务组成员、以及设置其他后台服务组选项。

11.2.2.1 服务器负载均衡算法

服务器负载均衡算法（SLB Methods）是在建立服务组时所定义的参数之一，不同的服务器负载均衡算法决定了请求数据在服务组内的不同后台服务器中的分配方式。依据不同的负载均衡算法，当数据到达设备时，设备会向服务组内的后台服务器做出类似轮流转发、根据探测结果做出转发决定等转发行为。

下表列出了设备支持的常用的负载均衡算法。

表11–1 设备常用负载均衡算法

负载均衡算法	描述
轮询（rr）	如果我们网络环境中，服务组 1 中有三台后台服务器，服务组 2 中有两台后台服务器，并且这两个服务组都采用轮询算法，那么通向服务组 1 的所有请求都将按照“1、2、3、1、2、3……”的顺序分配到服务组 1 中的后台服务器，通向服务组 2 的所有请求都将按“4、5、4、5”的顺序分配到服务组 2 中的后台服务器。
全局轮询（grr）	grr 在工作机制上和 rr 算法一样，使用环境不同。rr 算法适用于单 CPU 环境，grr 算法适用于多核 CPU 环境，在多核 CPU 环境下 rr 算法无法保证请求分配的准确性。
最少连接数（lc）	这种负载均衡算法将引导设备去选择服务组中当前活动连接最少的那台后台服务器。
最快回应（sr）	设备永远将下一个请求分配给拥有最短响应时间的服务器。使用这种负载均衡算法，我们可以将速度快的服务器和速度慢的服务器混合使用，并且速度快的服务器通常会被分配更多的请求。随着速度快的服务器响应时间逐渐增加，速度慢的服务器会逐渐获得新的请求。
保持 IP 地址（pi）	此方法将请求的源 IP 地址（客户 IP 地址）与某一后台服务器长期关联起来。来自此 IP 地址或网段的客户请求长期由一个后台服务器来响应。例如在电子商务中，一个客户所发起的事务中有很多中间状态，需要一直由同一后台服务器保存和处理，来实现复杂事务的连续性处理。当很多客户共享同一 IP 地址时，此方法不是很合适（比如许多客户的请求由一个总的代理服务器来提交，对于设备来说，看到的是同一个客户 IP 地址）。
保持 cookie（pc）	Persistent cookie 算法将一种 cookie 的值和一个后台服务关联起来。一旦这种关联建立起来，cookie 值一样将会一直由同一个后台服务来处理。使用基于 cookie 的方法时，需要为没有 cookie 的 HTTP 请求定义默认策略。
插入 cookie（ic）	设备动态插入 cookie 来保持客户端与后台服务器的长期对应关系。
重写 cookie（rc）	实时改写后台服务器的 cookie 来保持客户端与服务器的长期对应关系。此算法最适用于服务器已经有自己的 cookie 并且不接受任何外来的 cookie。
就近性（prox）	此算法基于 SDNS 的就近性信息，并且只能同“复位向”策略联用。这种算法将引导设备将流量导向离客户端最近的后台服务器。
SNMP（snmp）	此算法根据后台服务器的 SNMP（简单网络管理协议）的 MIB 信息来判

负载均衡算法	描述
	断服务的健康状态和可用性，例如后台服务的 CPU 和内存使用率。
嵌套 cookie（ec）	嵌套 cookie 在服务器端返回的 cookie 中插入信息，以保持后台服务器和客户端之间的连接。
哈希 query（hq）	此算法通过计算 HTTP 请求的 query 中指定的标签的哈希值保持会话的持续性。同时这个方法必须与 persistent url 策略一起使用。
QoS cookie（qc）	QoS cookie（区分服务质量的 cookie 算法）将一种 cookie 的值和一组后台服务器，而不是一个后台服务器关联起来。例如：具有[Cookie: service='GOLD']的 HTTP 请求将会被转发给"Gold"服务组来处理。而"Gold"组里的服务都是金牌级的服务，没有太大区别。使用此方法，也必须为没有 cookie 的 HTTP 请求定义默认策略。
Qos URL（qu）	Qos URL（区分服务质量的 URL 算法）利用客户请求中 URL 的一部分来做出负载均衡决策。例如：可以建立基于字符串"/german/"的算法，使得类似于 http://www.example.com/german/index.html 的请求（包含"/german/"字符串）都被分配给德语服务组，该组中的所有服务都返回德文页面。
Qos Hostname（qh）	Qos hostname 算法基于客户请求中"Host"表头来做出负载均衡决策。如果有针对不同"Host"表头的多种 HTTP 请求，对于不同 Host 的请求可以有不同的服务来处理。
最小带宽（lb）	最小带宽算法基于后台服务的带宽和权重值将流量分发至服务组中当前带宽最小的后台服务。

除了以上负载均衡算法外，本系统还支持下面算法：保持 URL（Persistent URL，pu）、保持域名（Persistent Hostname，ph）、哈希 Cookie（Hash Cookie，hc）、哈希表头(Hash Header，hh)、SSL SID（sslsid）、哈希 IP 地址（Hash IP，hi）、一致性哈希 IP 地址（Consistent Hash IP，chi）、radchu（Consistent Hash RADIUS User Name）、radchs（Consistent Hash RADIUS Session ID）、radpsun（RADIUS Persistence Session by Username）、persistence（Individual Session Persistence）、sipcid（SIP Call ID）、sipuid（SIP User ID）、sipcidps（SIP CallID Persistence Session）、sipuidps（SIP UserID Persistence Session）以及 pto（Persistent TCP Option）。

了解更多信息，请参见命令行使用手册中"服务器负载均衡"章节的相关介绍。

10.2.2.2 Proxy Protocol

系统已支持通过 Proxy Protocol 报文获取客户端真实 IP 地址。

TCP、TCPS、HTTP 和 HTTPS 协议类型的后台服务组可以被设置为 Proxy Protocol 报文发送者用于发送 Proxy Protocol 报文。系统支持发送 v1 和 v2 版本的 Proxy Protocol 报文。

- 设置后台服务组为 v2 版本的 Proxy Protocol 报文发送者。

Demo(config)#**slb group settings proxyprotocol g1 v2**

11.2.3 服务器负载均衡策略

负载均衡策略用于将虚拟服务和服务组绑定在一起。通过使用负载均衡策略，管理员可以控制 OSI 模型二至七层的负载均衡决定。虚拟服务使用策略绑定到一个服务组上，一个单个的服务组可以同时分配给不同的虚拟服务。下面列举了设备所支持的负载均衡策略。

表11–2 设备负载均衡策略

基本策略	保持策略	QoS 策略
DoH Redirect 静态 默认 备份	Persistent URL Persistent Cookie Rewrite Cookie Insert Cookie Hash URL RADIUS Username RADIUS Session ID SIP Domain SIP Call	QoS Cookie QoS Hostname QoS URL QoS Network QoS Clientport QoS Body QoS Dnsdomain QoS Dnsqtype QoS Diameter QoS Diameterappid Regex Header

不同类型的策略具有不同的优先级，通常情况下，多个设备的服务器负载均衡策略可以配置到同一个服务器负载均衡虚拟服务中，并且设备将基于最高优先级的策略来转发请求。下面列出了缺省的策略优先级顺序。

表11–3 设备负载均衡策略优先级

优先级顺序	负载均衡策略
a	DoH
b	Redirect
c	静态
d	DNSSEC
e1	QoS Clientport
	QoS Network
	Persistent URL
	Rewrite Cookie
	Insert Cookie
	Persistent Cookie
	QoS Cookie
	QoS Hostname

优先级顺序	负载均衡策略
	QoS URL
	QoS Body
	QoS Dnsdomain
	QoS Dnsqtype
	Regex
	Header
	Hash URL
e2	RADIUS Username
	RADIUS Session ID
	Filetype
	QoS Diameter
	QoS Diameterappid
	SIP Domain
	SIP Call
h	默认
i	备份

对于四层的服务器负载均衡而言，除了静态策略、默认策略和备份策略之外，只有四种策略（QoS Clientport、QoS Network、QoS Dnsdomain 和 QoS Dnsqtype）是可以使用的。考虑到功能的灵活性，缺省的策略优先级顺序是可以改变的。

11.2.3.1 策略嵌套

策略嵌套是把多个不同的策略嵌套起来共同实现服务器负载均衡。

传统的 SLB 策略机制中，策略用来关联虚拟服务与组。例如，在命令行“**slb policy qos url policy1 vs1 group1 ‘news’ 1**”中，虚拟服务“vs1”和组“group1”通过“policy1”策略进行关联。与“policy1”匹配的请求将会被直接转发到“group1”。

在策略嵌套机制中，我们引入了一个新的概念—“虚连接（vlink）”。使用者可以通过策略把虚拟服务或虚连接与组或另一个虚连接关联起来。系统将根据策略配置将请求分配给某个组或虚连接。对于发送至某个虚连接的请求，系统会根据策略配置将该请求转发至与该虚连接关联的组或另一个虚连接。只有匹配两个或更多嵌套策略时，请求才会被转发至与虚连接关联的组中。

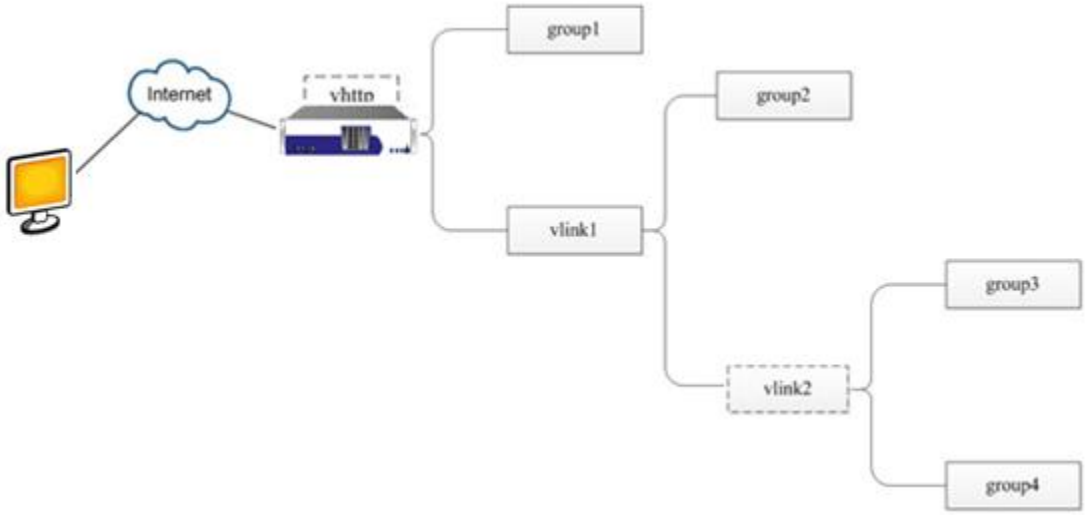

图11–3 策略嵌套

策略嵌套的限制：

- 某些策略不能被嵌套，例如：static、redirect、filetype 和 external 策略。
- 某些策略只能关联虚拟服务和组或虚连接和组，不能关联虚拟服务和虚连接或虚连接和虚连接，例如：icookie、rcookie、persistent cookie、persistent URL 和 QoS Diameter 策略。
- 为了简化配置，策略嵌套的层数应该小于或者等于三。如果嵌套的层数大于三，系统会返回 503 错误。
- 现在只支持 HTTP 和 HTTPS 协议。

注意：DoH 策略不支持策略嵌套（vlink）。

11.2.4 服务器负载均衡会话保持

设备的服务器负载均衡（SLB）模块引入一种新的基于会话 ID 的通用保持算法—Persistence 算法。Persistence 通用保持算法可独立应用于四、七层 SLB 服务，仅通过该算法即可获取会话 ID 并进行会话保持。此外，该算法也可与现有的七层负载均衡保持策略配合，先通过保持策略获取到会话 ID，进而通过该算法实现会话保持。

Persistence 算法为每个会话 ID 建立并维护一个独立的表项，以记录会话 ID 和后台服务器的对应关系，并动态处理每个会话 ID 的超时信息，确保各个会话之间互不影响。

11.2.4.1 会话 ID 的类型

Persistence 算法支持以下会话 ID 类型：

- **ip**：算法独立工作，从客户端请求中获取客户端的 IP 地址作为会话 ID。
- **ip+port**：算法独立工作，从客户端请求中获取客户端的 IP 地址和端口号作为会话 ID。
- **sslsid**：算法独立工作，从客户端请求中获取 SSL 会话 ID 作为会话 ID。
- **string**：获取指定的字符串作为会话 ID，分为以下两种情况：
  - 算法独立工作，从客户端请求或后台服务器的响应中获取指定的字符串作为会话 ID。
  - 通过负载均衡保持策略获取指定的字符串作为会话 ID。

对于 string 类型，还可以通过指定偏移量和字符串长度，获取更加精确的会话 ID。

11.2.4.2 获取会话 ID

➢ **独立获取会话 ID**

Persistence 算法可以独立工作，从客户端请求或后台服务器的响应中获取指定的字符串作为会话 ID。获取会话 ID 的方式如下：

- 从客户端请求中获取客户端的 IP 地址或 IP 地址和端口号作为会话 ID。
- 从客户端请求中获取 SSL 会话 ID 作为会话 ID。
- 从客户端请求的 HTTP URL query、HTTP cookie、HTTP header 或 HTTP body 部分获取指定的字符串作为会话 ID（通过“**slb group persistence request**”命令配置）。
- 从后台服务器的响应的 HTTP cookie、HTTP header 或 HTTP body 部分获取指定的字符串作为会话 ID。此时，也需要配置设备从客户端请求的 HTTP URL query、HTTP cookie、HTTP header 或 HTTP body 中获取指定的字符串以匹配会话 ID（通过“**slb group persistence response**”命令配置）。

注意：当配置了多个“string”类型时，设备将按照优先级从高到低依次从 HTTP URL query、HTTP cookie、HTTP header 和 HTTP body 中获取会话 ID。

➢ **从负载均衡保持策略获取会话 ID**

Persistence 算法也可以与现有的负载均衡保持策略配合使用，通过使用保持策略获取到的会话 ID，来实现会话保持。可以与 Persistence 算法配合使用的保持策略如下：

- Header
- Persistent cookie
- Persistent url
- Qos body

注意：关于会话 ID 与负载均衡保持策略和 Persistence 算法的匹配机制的细节，请联系公司技术支持获取相关文档。

关于 Persistence 算法（通过“**slb group method** <*group_name*> **persistence** ”命令配置）与现有负载均衡保持策略配合使用的更多信息，请参见本节的“Persistence 算法与保持策略配合实现会话保持”配置示例。

11.2.4.3 会话 ID 的超时管理

Persistence 算法为每个会话 ID 建立一个独立的表项，以记录会话 ID 和后台服务器的对应关系，并动态处理每个会话 ID 的超时信息，确保各个会话之间互不影响。

Persistence 算法支持以下两种对会话 ID 的动态超时管理模式：

- **idle**：表示持续未收到客户端请求的时间长度。在该时间长度内，如果持续未收到来自客户端的新的请求，设备将清除该客户端的会话 ID 信息。
- **duration**：表示客户端会话创建后的一段时间期限。该期限结束后，无论期间是否收到来自该客户端的新的请求，设备都将清除该客户端的会话 ID 信息。

注意：对会话 ID 进行动态超时管理时，必须指定会话 ID 的超时管理模式，以确保设备可以清除过期的会话 ID，记录新的会话 ID。

此外，Persistence 算法也支持对会话 ID 进行静态保持（通过“**slb group persistence session static**”命令配置）。在这种情况下，带有已定义的会话 ID 的请求将被始终发往同一后台服务器。

11.2.5 服务器负载均衡健康检查

设备为服务器负载均衡提供了四种类型的健康检查：

11.2.5.1 基本健康检查

基本健康检查也称为主健康检查，使用特定的协议格式来判断后台服务器或后台应用的健康状况（Down 或 Up）。基本健康检查里面包含了 ICMP（ping）、TCP、

TCPS、DNS、HTTP 和 HTTPS 等类型的健康检查。在单个后台服务器配置中可以指定默认的健康检查，设备在后台服务器特定的 IP/端口对上执行健康检查确定其当前状况是否可用。

11.2.5.2 附加健康检查

很多情况下，一个应用或服务的基础功能的正常工作需要多个不同类型的服务器同时运行才能够实现。例如，一个应用可能同时需要 Web 服务器、应用服务器和数据库服务器。这样，仅仅对一个服务器进行健康检查便不足以实现对该应用的健康状况进行监控。因此，除了基本健康检查之外，设备还可以配置多个不同类型的附加健康检查，并通过 AND、OR 对这些附加健康检查设置逻辑关系共同对后台服务器进行健康检查。

11.2.5.3 脚本健康检查

脚本健康检查为通过脚本的方式检查后台服务的健康检查状况。脚本健康检查可以通过两种方式定义：手动配置和文件导入。

- **手动配置**

有些应用或者非标准的协议使用请求响应序列进行通信。设备基于 TCP 或 UDP 连接提供了通用的脚本健康检查支持。它通过交换各种应用信息来判断该应用的健康状况。通用的脚本健康检查类型是“script-tcp”和“script-udp”。

脚本健康检查支持以下几种高层应用的健康检查：

FTP、SMTP、LDAP、RADIUS、POP3、DNS、TELNET 应用。

- **文件导入**

系统允许管理员从 HTTP 服务器或 FTP 服务器导入自定义健康检查脚本文件。该方式的脚本健康检查类型是“external-script”。

11.2.5.3.1 配置示例

1. 导入自定义健康检查脚本文件。

```
Demo(config)#health import extscript abc ftp://test:password@10.8.5.55/test.txt
```

2. 将指定的健康检查与自定义健康检查脚本文件绑定。

```
Demo(config)#health bind rs1 abc
```

11.2.5.4 服务组健康检查

服务组健康检查用于判断关联服务组中每一个后台服务的健康状况。服务组健康检查支持的健康检查类型和基本健康检查相同。管理员可以根据实际情况配置服

务组健康检查来获取关联服务组中每一个后台服务的健康状况，而不必为每一个后台服务单独配置健康检查，从而简化了健康检查的配置。

此外，服务组健康检查还支持开放云环境中的服务器负载均衡。关于如何在云环境中配置服务组健康检查，请联系公司技术支持获取相关文档。

注意：

- 服务组健康检查和基本健康检查的关系是“AND”。
- SSL/TLS 类型的基本健康检查、附加健康检查和服务组健康检查支持配置 SSL/TLS 协议版本和密码套件。
- 当基本健康检查为非 SSL/TLS 类型时，SSL/TLS 类型的附加健康检查将默认配置所有系统支持的 SSL/TLS 协议版本和密码套件。
- 在不配置 SSL 主机的情况下，SSL/TLS 类型的基本健康检查、附加健康检查和服务组健康检查仍然可以正常工作，ClientHello 消息将携带所有系统支持的 SSL/TLS 协议版本和密码套件。

11.2.5.5 健康检查方法

健康检查允许通过每个设备来搜集后台服务器信息，并基于这些信息做出健康诊断。这些诊断的结果将协助设备了解后台服务的情况，并决定将请求转发给哪些后台服务进行处理。

设备支持以下类型的健康检查方法：

- **ICMP 健康检查**

ICMP 健康检查通过简单的发送个 ICMP echo（ping）请求给后台服务器。如果后台服务器有 ICMP 回答，服务将被认为是好的。否则服务不可用。这种健康检查具有一定的局限性，并不能保证真正的应用服务是好用的。

- **TCP 健康检查**

TCP 健康检查简单的与后台服务器建立一个 TCP 连接。如果连接建立失败，后台服务不可用。反之，后台服务是好的。此检查与指定的端口建立连接，但是并不能保证实际服务功能上没有问题。更为有效的健康检查是通过 HTTP 请求来实现的。

对于二层服务器健康检查，TCP 健康检查方法需要在另外一台设备上设置一个健康检查反射器，由该反射器打开并监听设备的端口，对健康检查请求进行响应，从而实现对整个链路的健康检查。在这个健康检查方法中，健康检查请求会经过二层负载均衡的后台服务器，被转发到配置有反射器的设备上。若反射器能够对健康检查请求做出回应，则后台服务的健康状态被标记为“up”，否则标记为“down”。

- **TCPS 健康检查**

TCPS 检查基于 SSL 握手协议来检查能否与后台 SSL 服务器正常握手。如果 SSL 握手失败，服务不可用。成功握手说明后台服务是好的。简而言之就是与后台服务器建立 SSL 连接。

- **HTTP 和 HTTP2 健康检查**

HTTP 和 HTTP2 健康检查在建立 TCP 连接的基础上向后台服务发送预定义的 HTTP 请求。如果后台服务的回答与期望回答一致，服务是好的。否则，服务不可用。使用 HTTP 和 HTTP2 健康检查时，需要预先定义一组 HTTP 请求以及匹配的回答。系统支持配置 HTTP 和 HTTP2 类型的基本健康检查、附加健康检查以及服务组健康检查。

- **HTTPS 和 HTTPS2 健康检查**

HTTPS 和 HTTPS2 健康检查基于 SSL 握手协议，但是 HTTPS2 健康进行 SSL 握手时不检查证书。如果 SSL 握手成功，设备会给后台服务发送预定义的 HTTP 请求。如果后台服务返回的响应与期望响应一致，那么后台服务就是可用的；反之，后台服务就是不可用的。使用 HTTPS 和 HTTPS2 健康检查，用户必须提前定义好 HTTP 请求和与请求匹配的回应。对于 HTTPS 健康检查，在进行客户端认证时导入的客户端证书必须是用 DER 规则加密的。

注意：为了系统正常工作，推荐配置最多 500 条 HTTP2 类型的健康检查（包括基本、附加和服务组健康检查）。

- **Script-TCP 健康检查和 Script-UDP 健康检查**

Script-tcp 和 Script-udp 是更为细致的健康检查，不是根据一个请求/响应来判断服务好坏，而是根据一个包括多次请求与响应的握手系列来更为准确的了解后台服务情况。它们分别基于不同传输层连接：TCP 或 UDP 连接。

- **Script-TCPS 健康检查**

Script-TCPS 健康检查通过 SSL 握手协议来检查 HTTPS 后台服务是否可用。在 SSL 握手成功之后，它的工作步骤与 Script-TCP 健康检查相同。

- **DNS 健康检查**

DNS 健康检查与后台服务建立 UDP 连接并且发送一个测试 DNS 请求，然后等待后台服务的回答。如果有 DNS 回答，后台服务可用，否则后台服务不是健康的。

- **DNS-TCP 健康检查**

DNS-TCP 健康检查与后台服务建立 TCP 连接并且发送一个测试 DNS 请求，然后等待后台服务的回答。如果有 DNS 回答，后台服务可用，否则后台服务不是健康的。

- **Radius-Auth 健康检查和 Radius-Acct 健康检查**

Radius-auth 健康检查和 Radius-acct 健康检查主要用来检查 RADIUS 后台服务状况。它们都是由一系列 RADIUS 握手协议过程来完成的。Radius-auth 检查客户认证情况。Radius-acct 检查资源授权情况。

- **Diameter 健康检查**

Diameter 健康检查指设备与 Diameter 后台服务建立 TCP 连接后，发送 CER（Capabilities-Exchange-Request）报文并等待接受 CEA（Capabilities-Exchange-Answer）报文。接受正确的 CEA 报文后，系统判断后台服务状态为“up”；否则系统判断后台服务状态为“down”。

Diameter 健康检查还支持 DPR（Disconnect-Peer-Request）扩展选项。启用该功能后，设备接收到正确的CEA报文后，还将发送DPR报文。接收到正确的DPA（Disconnect-Peer-Answer）报文后，系统才判断后台服务状态为“up”；否则系统判断后台服务状态为“down”。

- **LDAP 健康检查**

设备支持对常见的LDAP服务器进行健康检查，如Windows AD、OpenLDAP以及 SunOne Directory 等，以满足用户执行绑定和搜索操作时健康检查的需要。（目前的健康检查机制仅支持绑定和搜索操作，即如果绑定、搜索成功，则将 LDAP 服务器的状态标记为“UP”，表示健康。）

LDAP 类型的健康检查只适用于 TCP 后台服务。

- **RTSP-TCP 健康检查**

RTSP 健康检查打开一个 TCP 连接，并且向后台服务器发送 RTSP“OPTIONS”请求。如果后台服务器作出应答，这个服务器将被标记为“up”，否则将被标记为“down”。

- **SIP-UDP 和 SIP-TCP 健康检查**

SIP 健康检查打开一个 udp 或者 tcp 连接，并且向后台服务器发送 SIP“OPTIONS”请求。如果后台服务器作出应答，这个服务器将被标记为“up”，否则将被标记为“down”。

- **POP3 健康检查**

POP3 健康检查通过与后台服务的一系列握手来检查后台服务的健康状况。首先，设备会与后台服务建立一个 TCP 连接。当设备收到后台服务状态为 Ready 的报文后，会向后台服务依次发送 USER 请求和 PASS 请求进行客户端身份认证。完成认证后，设备将发送 STAT 请求或自定义请求（通过“**health request**”命令配置），检查后台服务是否能返回符合预期的响应（通过“**health response**”命令配置）。收到正确响应后，设备将发送 QUIT 请求，如果后台服务返回预期响应，其状态将被标记为“up”，否则标记为“down”。

- **SNMP 健康检查**

SNMP 健康检查通过设备与后台服务的一系列握手来检查后台服务的健康状况。设备向后台服务发送一条 SNMP 的查询节点的请求，如果后台服务返回的 SNMP 响应与预期的响应一致，则它被标记为“up”；否则将被标记为“down”。

- **SMTP 健康检查**

SMTP 健康检查通过与后台服务的一系列握手来检查后台服务的健康状况。首先，设备与后台服务建立一个 TCP 连接。设备收到后台服务状态为 Ready 的报文后，依次向后台服务发送 NOOP 请求或用户自定义请求以及 QUIT 请求。如果后台服务分别返回符合预期的响应，那么后台服务将被标记为“up”，否则被标记为“down”。

- **NTP 健康检查**

NTP 健康检查与后台服务建立 UDP 连接并且发送一个 NTP 时钟同步请求，然后等待后台服务的响应。如果后台服务返回符合预期的 NTP 响应，那么后台服务状态将被标记为“up”；否则标记为“down”。

- **数据库健康检查**

数据库健康检查通过设备与后台数据库服务器的一系列握手来检查后台数据库服务器的健康状况。首先设备与后台数据库服务器建立一个 TCP 连接，然后设备将登录到数据库，并向数据库后台服务器发送一条数据库查询消息。如果后台数据库服务器返回期望的应答，则它被标记为“up”；否则将被标记为“down”。

目前，系统支持三种类型的数据库健康检查：Oracle、MySQL 和 MsSQL。MySQL 数据库健康检查包括 MySQL-DB 和 MySQL-DBS，其中，MySQL-DBS 是基于 SSL 握手协议的数据库健康检查。MsSQL 数据库健康检查包括 MsSQL-DB 和 MsSQL-DBS，其中，MsSQL-DBS 是基于 SSL 握手协议的 MsSQL 数据库健康检查。

11.2.5.6 健康检查项目和健康检查项目序列

健康检查项目是复杂的健康检查握手系列中的一个基本步骤。整个健康检查可能包含多个请求及响应来回，而一个健康检查项目只包含一个来回，即发送一个请求并且接受一个应答。健康检查项目队列就是由一系列有序的健康检查项目组成的，每个检查项目队列都具有一个名称作为标识。

注意：健康检查队列仅适用于 script-tcp 和 script-udp 类型的健康检查。

以下命令用于创建健康检查项目以及检查队列：

health checker <*checker_name*> <*request_index* ><*response_index*> *[timeout] [flag]*

health list <*list_name*>

health member *<list_name> <checker_name> [place_index]*

health app *{real_name|add_hc_name|group_hc_name} <list_name> [frequency] [hc_localip] [hc_localport]*

例如：

```
Demo(config)#health checker checker1 1 1 3 1
```

上面的命令定义了一个健康检查项目 checker1。Checker1 的请求和应答序号都是 1(请求和应答需提前定义)。健康检查的超时时间为 3 秒。

11.2.5.7 HTTP 请求和应答

缺省情况下，系统为 HTTP 健康检查预定义了一组 HTTP 请求和一组期望应答。每个请求和应答都有个序号。如果 HTTP 健康检查配置中没有特别指明请求和应答的序号，默认序号都是“0”。“**show health request**”命令显示所有预定义的请求。“**show health response**”命令显示所有预定义的应答。

下面的例子列举了 HTTP 请求和应答的输出结果。

```
Demo(config)#health on
Demo(config)#show health request
Row    Request
0      HEAD / HTTP/1.0
1      HEAD / HTTP/1.0
2      HEAD / HTTP/1.0
3      HEAD / HTTP/1.0
4      HEAD / HTTP/1.0
5      HEAD / HTTP/1.0
6      HEAD / HTTP/1.0
7      HEAD / HTTP/1.0
8      HEAD / HTTP/1.0
9      HEAD / HTTP/1.0
10     HEAD / HTTP/1.0
11     HEAD / HTTP/1.0
12     HEAD / HTTP/1.0
13     HEAD / HTTP/1.0
14     HEAD / HTTP/1.0
15     HEAD / HTTP/1.0
…
Demo(config)#show health response
Row    Response:
0      200 OK
1      200 OK
```



```
2       200 OK
3       200 OK
4       200 OK
5       200 OK
6       200 OK
7       200 OK
8       200 OK
9       200 OK
10      200 OK
11      200 OK
12      200 OK
13      200 OK
14      200 OK
15      200 OK
…
```

序号 0（0 在 HTTP 健康检查表里意味着下面 HTTP 请求将被发给后台服务）：

HEAD / HTTP/1.0

期望从后台服务得到的应答是：

200 OK

- **修改 HTTP 健康检查请求/应答**

客户可以根据后台服务器具体情况定义自己的请求和应答值。比如，请求可以是取得一个反映后台数据库服务状态的 CGI 脚本。当数据库服务正常时，此脚本正常返回 HTTP 200 OK。如果返回“404 没有找到”，则表示后台服务不正常。

缺省情况下，系统发出 HTTP HEAD 请求来做健康检查。HEAD 请求只要求后台服务返回所请求对象的 HEADER 信息。如果使用 GET 请求，期望应答将是整个请求的 Web 页面。对于很大的 Web 页面来说，这种健康检查有时也是不可忽视的额外负担。

使用下面命令来自定义健康检查的请求：

health request *<request_index> <request_string>*

例如：

1. 定义我们的 HTTP 请求。

```
Demo(config)#health request 1 "GET /cgi-bin/dbstatus.pl"
```

3. 使用“**health server**”命令绑定后台服务与请求/应答对。

```
Demo(config)#health server server2http 1 0
```


上面命令为后台服务 server2http 设置了自定义请求“1”和默认应答“0”(即“200 OK”)。

注意：健康检查应答中的“Index 0”表示返回结果为“200 OK”。

- **Web 页面的关键词检查**

HTTP 健康检查支持后台服务响应页面的关键词匹配。HTTP 健康检查模块在后台服务的应答中查找关键词。如果找到关键词，则表示该后台服务状态正常；否则该后台服务不正常。

下面是一个配置示例：

后台服务：10.3.16.188:88

Web 页面：index.txt

关键词：admin

1. 添加具有 HTTP 健康检查的后台服务。

```
Demo(config)#slb real http rs 10.3.16.188 88 1000 http 3 3
```

4. 配置 HTTP 健康检查请求和回应，在回应中指明应包含的关键词。

```
Demo(config)#health request 1 "GET /index.txt HTTP/1.0\r\n\r\n"
Demo(config)#health response 1 "admin"
```

5. 将后台服务同请求和回应对关联起来。

```
Demo(config)#health server rs 1 1
```

注意：关键词 HTTP 健康检查只能支持 ASCII 字符串匹配，不支持双字节字符串（如简体中文、繁体中文等）。

11.2.5.8 基于 TCP 协议的服务器负载均衡虚拟服务健康检查

基于TCP协议的服务器负载均衡虚拟服务健康检查为外部设备提供了一种手段，可以方便了解基于 TCP 连接的虚拟服务的可用性。

如果一个虚拟服务的所有相关联的后台服务都不可用，该虚拟服务自己也应该不可用。此时，从外部设备发起的与此虚拟服务的 TCP 连接将不被响应，这样，外部设备就可以有效探测出目标虚拟服务的可用性。

11.2.5.9 健康检查的失败切换

如果通过健康检查发现一台设备中配置的所有后台服务都是不可用的，那么所有的流量将会由其他的设备处理。这台设备配置的后台服务中只要至少有一个后台服务是可用的，如果它是抢占模式所有的流量将由它处理。

11.2.5.10 检查健康检查结果

定义好后台服务和相关的健康检查后，可以使用 CLI 命令检查后台服务的状态。这样有助于分析后台服务存在的问题，比如 IP 地址配置错误。“**show health server**”命令用来实时显示后台服务的健康状态，如下所示：

```
Demo(config)#show health server
----------------------Server Status---------------------------
real server name          status
server1http                UP
server2http                UP
server3http                UP
server4http                UP
server5http                DOWN

----------------------Health Check-----------------------------
real server name      ip                :port     status      hct      rqr     rpr     checklist
server1http           192.168.10.10     :80       UP          tcp
server2http           192.168.10.11     :80       UP          http     1       0
server3http           192.168.10.12     :80       UP          tcp
server4http           192.168.10.13     :80       UP          tcp
server5http           192.168.10.15     :80       DOWN        tcp
```

从上面结果可以看到，server5http 不健康。这时可以进行诊断，可能是它的虚拟 IP 地址定义错了，不是 192.168.10.15，而是 192.168.10.14。这样我们就知道应该做出如下改正：

```
Demo(config)#slb real http server5http 192.168.10.14
```

配置完后台服务和健康检查后，查看一下健康状态是个好习惯。这样可以保证在系统运行前，一切配置都没有问题。“**show health server**”命令就起到了这个检查报警功能。

11.2.6 透明模式、反向代理模式和三角传输模式

11.2.6.1 透明模式

设备的透明模式是指设备在转发用户请求时，透明的将客户端的连接指向到特定的服务器上，即用户的源 IP 地址对服务器是透明的，服务器可以知道哪个客户端对其进行了访问。在透明模式下，源 IP 保持不变，但源端口可能发生变化。如果要保持源端口不变，管理员设置的虚拟服务端口和后台服务端口的差值必须是设备的 CPU 核数的整数倍。如下图所示。

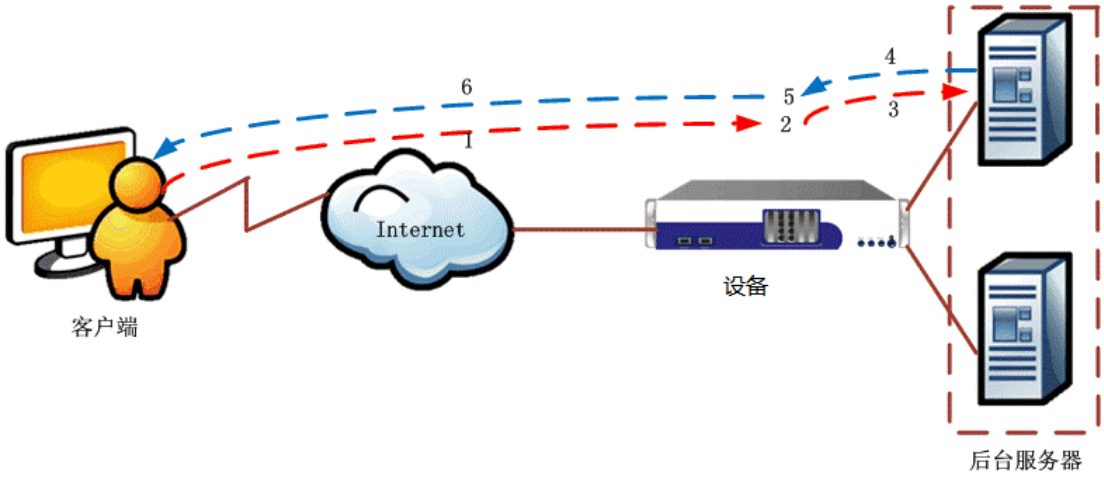

图11–4 服务器负载均衡透明模式

1. 客户端将请求发送到设备的虚拟 IP 地址上。
6. 设备通过策略和健康检查，寻找到最佳状态的服务器，将请求的目的地址换成后台服务器的 IP 地址。
7. 设备将请求发送到该服务器。
8. 服务器接到请求后，将回应发送给设备。
9. 设备将相应的源地址转换为虚拟 IP 地址。
10. 设备将该回应发送给客户端。

透明模式的优点：

服务器可以记录客户端访问的 IP。

透明模式的局限性：

- 结构/路由设计必须保障从源服务器端来的响应经过设备。
- 单臂结构下，设备对与后台服务处于同一网段的客户端，不能实现透明模式。
- 由于每个请求的 IP 地址都不同，所以无法利用连接池技术改善系统性能。

11.2.6.2 反向代理模式

使用设备的反向代理服务可以将请求转发给内部的服务器，让设备将请求均匀地转发给多台内部服务器之一上，从而达到负载均衡的目的。这种代理方式与普通的代理方式有所不同，标准代理方式是客户使用代理访问多个外部服务器，而这种代理方式是多个客户使用它访问内部服务器，因此也被称为反向代理模式。如下图所示。

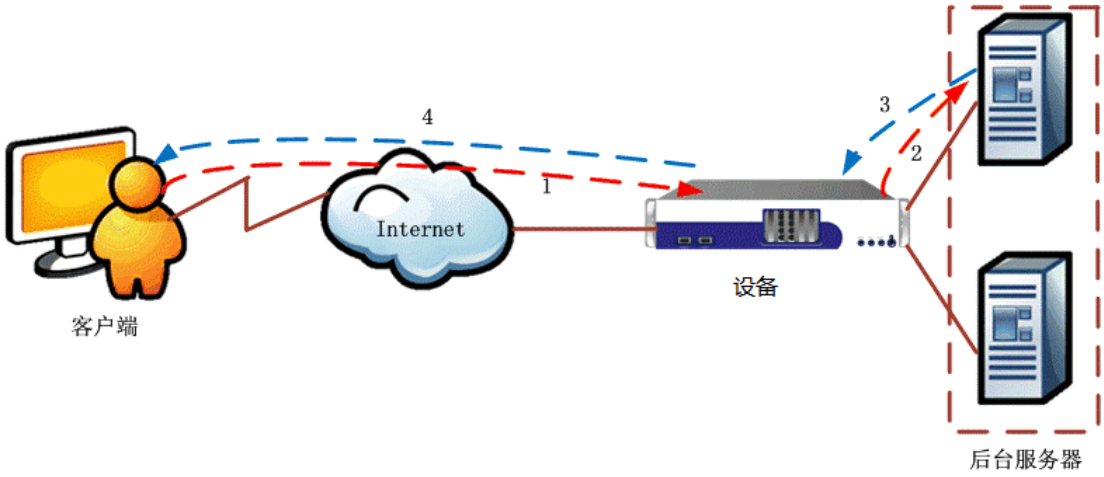

图11–5 服务器负载均衡反向代理模式

1. 客户端将请求发送到设备的虚拟 IP 地址。
2. 设备根据负载均衡策略和健康检查结果，将请求发送到最佳的后台服务器。
3. 后台服务器将回应发送给设备。
4. 设备将回应传送给客户端。

反向代理模式的优点：

- 可以采用单臂结构部署。
- 可以通过连接池技术增强系统性能。

反向代理模式的局限性：

- 服务器无法记录哪些 IP 的客户端曾经进行访问。
- 解决办法：设备可以在使用者的 HTTP 头中加入 X-Forwarded-For 字段，用于记录客户端的 IP 地址。

11.2.6.3 三角传输模式

设备的三角传输功能是专门为低入站/高出站的应用设计的，例如视频点播应用。它能以最快、最有效的方式响应请求。在这种传输模式下，客户端的请求经过设备到达后台服务，后台服务的响应直接返回客户端而不经过设备。如下图所示。

对于三角传输，管理员可以使用轮询（rr）、Persistent IP（pi）、哈希 IP（hi）、一致性哈希 IP（chi）、最少连接（lc）以及 snmp 等方法来选择一个合适后台服务。在三角传输模式下，服务器负载均衡只支持 TCP、UDP 和 IP 类型的虚拟服务。

三角传输模式的工作原理如下：

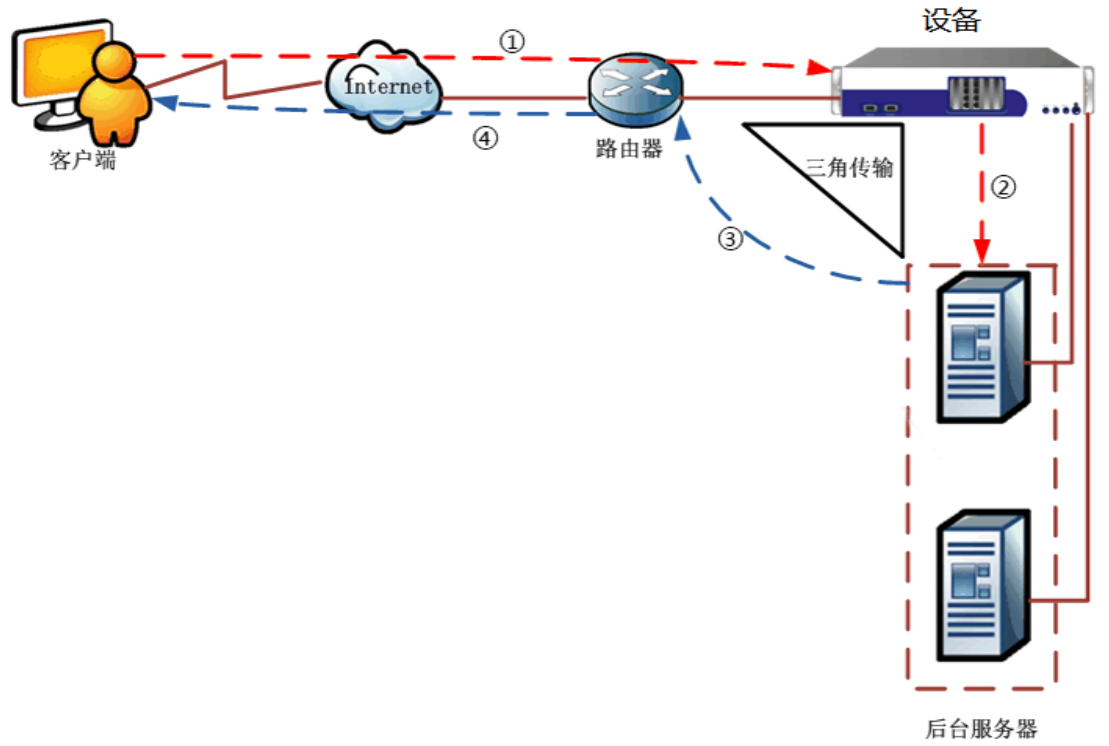

图11–6 服务器负载均衡三角传输模式

1. 客户端通过路由器向设备的虚拟 IP 地址发送一个请求。
11. 设备把这个请求转发给后台服务器。
12. 后台服务器上配置的默认网关地址正是路由器内部接口的 IP 地址，因此响应会被直接发送到路由器。
13. 路由器将回应传送给客户端。

三角传输模式的优点：

- 为数据响应提供更优化的路径。
- 为网络应用提供更快的响应速度。

注意：三角传输模式下的服务器健康检查是基于后台服务器的系统 IP 地址，而不是环回 IP 地址。这就意味着当健康检查开启时，后台服务有可能无法访问。

11.2.7 强制负载均衡（基于包的 UDP SLB）

传统的 UDP SLB 根据建立 UDP 会话的四元组转发数据报，所有源端口和源 IP 地址相同的 UDP 包将被发送到同一个后台服务器。但是，对于一些使用 UDP 协议的应用，这种方法可能会导致基于包的应用流量不能被均衡的分配到各个服务器。例如，在 Radius 应用中，通常只有很少的 Radius 代理会向 Radius 服务器发起请求，而每个代理只使用一个端口和服务器通信。结果就是，一个代理只能和一个服务器通信，而某些服务器将永远不会被使用。针对这种情况，设备提供了强制负载均衡方法（基于包的 UDP SLB），将来自同一个连接的数据报发送到不同的后台服务器，以实现更加精准的流量管理。

11.2.8 SIP 协议的负载均衡

会话初始协议（Session Initiation Protocol，SIP）是一个应用层控制协议，它可以用于在两个或更多个结点间建立，维护或终止一个网络对话。目前广泛应用于 VOIP。

SIP 服务器有三种类型：代理服务器、注册服务器和复位向服务器。每种 SIP 服务器都可以支持多种 SIP 请求或响应。大多数请求和响应都包含一个 SIP 表头，这个表头是由多个 SIP 表头字段组成的。其中一些 SIP 表头字段是用来保持会话长连接的，比如 Call-ID 和 User-ID 等表头信息。SIP 封装包具有相同的 Call-ID 或者 User-ID 的，需要被送到同一个 SIP 服务器上。如下图所示。

SIP 协议的服务器负载均衡对所有运行在 TCP 和 UDP 协议上的 SIP 流量做负载均衡。基于 SIP 协议，Call-ID 和 User-ID 表头被用于保持一个 SIP 会话，有相同表头信息的将被转发到同一个后台服务上。

SIP 协议的负载均衡仅支持反向代理模式和透明模式。出于稳定性考虑，推荐管理员为 SIP 协议的负载均衡配置透明模式。

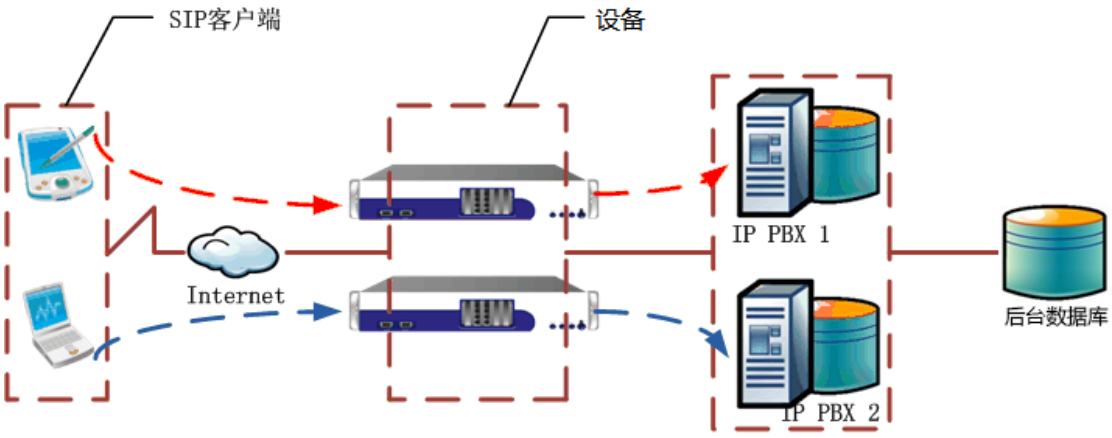

图11–7 SIP 协议的服务器负载均衡

为了使 SIP 协议服务器负载均衡更好地工作，设备实现了基本的 SIP 代理功能。代理功能允许真实服务器存放于私有网络，而不需要直接在互连网上公开。SIP 表头中与 IP 地址和端口有关的信息将被重写。被重写的 IP 地址和端口是可以配置的。

注意：SIP 代理服务器是在一个 SIP 功能网络中控制链接和 IP 地址管理的服务器。

此外，为了同步 SIP 注册信息，设备 SIP 协议服务器负载均衡支持通过广播方式同步注册请求给相同服务组组中的所有 SIP 注册服务器。

SIP 协议负载均衡支持以下负载均衡策略：

- SIP Domain 策略：通过 SIP Domain 策略关联虚拟服务和服务组后，当 SIP 请求匹配 SIP Domain 策略指定的域名和呼叫流程时，请求会被转发到与该策略关联的服务组。通过配置 SIP Domain 策略，可以将访问相同域名的 SIP 请求保持到同一个服务组。

  除了域名以外，SIP Domain 策略支持基于三种呼叫流程触发：主叫流程、被叫流程和短信、注册等其他流程。

- SIP Call 策略：SIP Call 策略支持按照域名、呼叫流程和呼叫号码触发。
  - 与 SIP Domain 策略相比，SIP Call 策略支持基于两种呼叫流程触发：主叫流程和被叫流程。
  - 呼叫号码包括主叫号码和被叫号码。其中主叫号码支持基于 PAI（逻辑主叫号码）或 FROM（源主叫号码）头域触发，可设置优先级；被叫号码支持基于请求行（逻辑被叫号码）或 To（源被叫号码）头域触发，也支持设置优先级。

SIP Call 策略需要和呼叫号码全匹配规则（通过“**slb sip callnum list import**”配置）或呼叫号码正则匹配规则（通过“**slb sip callnum regex**”命令配置）配合使用。通过 SIP Call 策略关联虚拟服务和服务组后，需为 SIP Call 策略配置呼叫号码全匹配规则或者呼叫号码正则匹配规则。两种匹配规则都支持基于主叫号码和被叫号码配置，因此每个策略最多可以配置四种匹配规则：

- 主叫号码全匹配规则
- 被叫号码全匹配规则
- 主叫号码正则匹配规则
- 被叫号码正则匹配规则

四种规则全部配置的情况下，任一规则匹配，则命中策略。

SIP Call 策略配置示例：

1. 配置 SIP TCP 虚拟服务“siptcpvs”。

```
Demo(config)#slb virtual siptcp siptcpvs 10.8.6.138
```

14. 配置后台服务组“g1”，算法为 sipcid。

```
Demo(config)#slb group method g1 sipcid
```

15. 配置 SIP TCP 后台服务“rs1”和“rs2”，并添加到服务组“g1”。

```
Demo(config)#slb real siptcp rs1 192.169.0.126
Demo(config)#slb real siptcp rs2 192.169.0.127
Demo(config)#slb group member g1 rs1
Demo(config)#slb group member g1 rs2
```

16. 配置一个主叫流程的 SIP Call 策略“p1”。

```
Demo(config)#slb policy sipcall p1 siptcpvs g1 abc.com 12 calling from to
```

17. 为策略“p1”配置被叫号码正则匹配规则。

```
Demo(config)#slb sip callnum regex p1 callee “[1][3578]\\d{9}”
```

18. 配置后台服务组“g2”，算法为 rr。

```
Demo(config)#slb group method g2 rr
```

19. 配置 SIP TCP 后台服务“rs3”，并添加到服务组“g2”。

```
Demo(config)#slb group member g2 rs3
```

20. 配置默认策略。

```
Demo(config)#slb policy default siptcpvs g2
```

基于以上配置，SIP TCP 虚拟服务会将匹配以下条件的 SIP 请求转发给服务组“g1”：

- SIP 请求属于主叫流程；
- 请求的域名为“abc.com”；
- 被叫号码匹配正则表达式“[1][3578]\\d{9}”。

服务组“g1”会按照 sipcid 算法将 SIP 主叫分配给“rs1”或“rs2”；对于所有其他的 SIP 请求，SIP TCP 虚拟服务会基于默认策略将请求转发给服务组“g2”。

11.2.9 RTSP 协议的负载均衡

实时流媒体协议（Real Time Streaming Protocol，RTSP）是一种用于控制实时媒体流的应用层协议。通常情况下，RTSP 协议包含两条通道：控制信道和媒体数据信道。控制信道为媒体数据信道中的 TCP 和 UDP 连接提供 TCP 控制。

RTSP 负载均衡用于实时传递流媒体数据，需要同时考虑到 RTSP 控制数据和媒体数据。

本版本只支持如下 RTSP 服务器负载均衡策略：“filetype”策略、默认策略（default）、备份策略（backup）和静态策略（static）。在所有被支持的策略中，“filetype”策略是专门为 RTSP 服务器负载均衡加入的。后台服务的选择是基于档扩展名称的。例如，客户端请求“rtsp://mp3.xyz.com/test.mp3”将会命中 RTSP 服务器负载均衡服务组中与“mp3”文件类型相关联的组。

RTSP 服务器负载均衡支持的负载均衡算法包括轮询（rr）、最短响应时间（sr）、snmp、保持 IP（pi）、哈希 IP（hi）、和一致性哈希 IP（chi）。特别需要注意的是，RTSP 服务器负载均衡的工作模式可以分为两种：“复位向”和“动态 NAT”。

11.2.9.1 复位向模式

在上面的复位向过程中，媒体流不会通过设备。一旦设备选定提供服务的服务器后，设备和客户端之间的连接将被断开。客户端将打开一个新的到后台服务的连接并发送媒体数据。请注意，在复位向模式下，所有的后台服务器需要有公网 IP 地址，以便让客户端通过 Internet 连接。如下图所示。

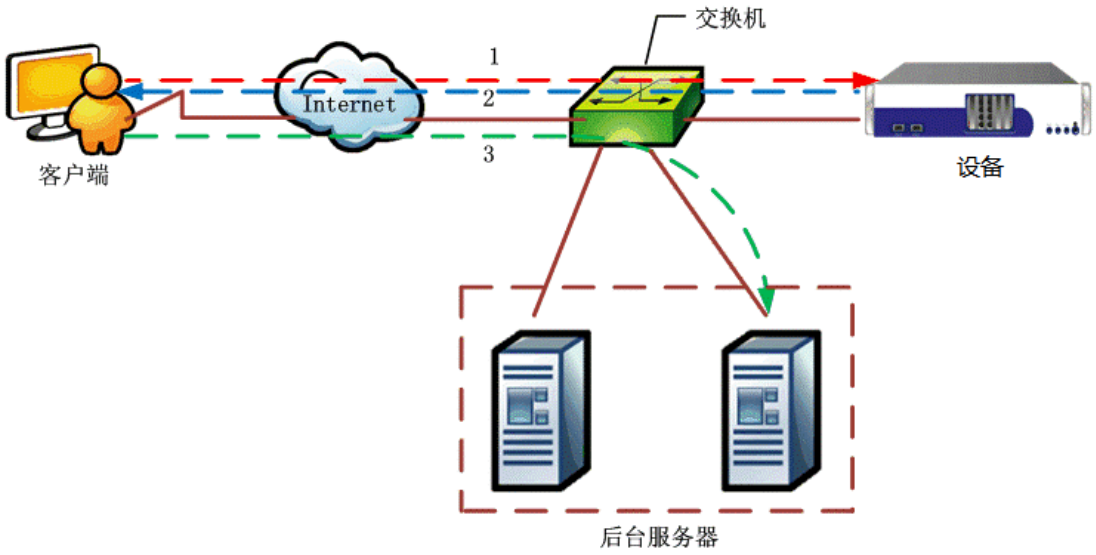

图11–8 RTSP 协议的服务器负载均衡复位向模式

在复位向模式下，RTSP 服务器负载均衡按照如下方式工作：

1. RTSP 客户端发送一个 RTSP 请求给设备。设备从客户请求中得到请求的地址并根据 URL 中的文件类型选择一个后台服务。

21. RTSP 生成一个复位向响应告诉客户端直接连接被选择的后台服务。

22. 客户端直接连接被复位向的后台服务。

在复位向模式下，客户端应该支持 RTSP/1.0 复位向方法。例如，Real Player 和 Real One Player 支持 RTSP/1.0 复位向方法，但是 Quick Time Player（7.0）不支持，所以 RTSP 服务器负载均衡支持 Real Player 和 Real One Player，但是不支持 Quick Time Player（7.0）。RTSP 服务器负载均衡不支持需要依赖 Quick Time 插件才能使用的文件类型（比如“mp4”和“mov”）。

11.2.9.2 动态 NAT 模式

在动态 NAT 模式下，后台服务没有属于自己的公网 IP 地址。控制数据和媒体数据都将通过设备传输，但是方式不同。控制数据通常由外部客户引发，由 SLB 处理。媒体数据一般由内部媒体服务器引发并通过端口映像方式传递。如下图所示。

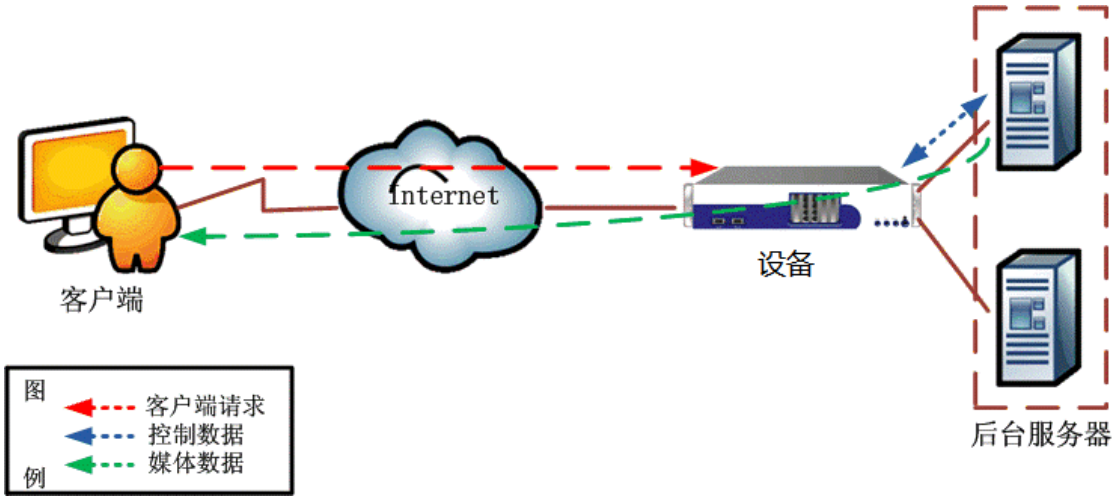

图11–9 RTSP 协议的服务器负载均衡动态 NAT 模式

在动态 NAT 模式下，RTSP 服务器负载均衡不能完全的支持文件类型策略。如果客户端的首次请求是“DESCRIBE”或者“SETUP”，设备能够使用文件类型策略找到后台服务，因为档扩展名称能够在请求中得到（类似于 Quick Time Player）。然而，对于首次请求是“OPTIONS”的，设备将会通过预设策略选择一个后台服务器，因为“OPTIONS”（类似 Real Player 或 Real One Player）请求不包括文件类型信息。除此之外，RTSP 服务器负载均衡不支持 RTSP 多播传输。

不论是复位向模式还是动态 NAT 模式，他们支持的策略和组方法的配置都是一样的。但是他们对于虚拟服务和后台服务的命名方法不太一样。

11.2.10 Tuxedo 协议的负载均衡

Tuxedo（Transactions for Unix Extended for Distributed Operations，分布式操作扩展之后的 Unix 事务系统）是一种介于客户机和服务器之间的中间件平台，用于构建可靠的大规模分布式企业应用,它使企业在简化开发和部署的同时还可以保留现有资产。Tuxedo 服务器通过两条连接来处理事务：

1. WSC（workstation client）请求与 WSL（workstation listener）建立 TCP 连接。

23. WSL 认证请求合法后，会分配一个 WSH（workstation handler）进程处理请求，并将该 WSH 进程的 IP 地址和端口号回复给 WSC。

24. WSC 请求与 WSH 建立连接，后续的请求和响应都通过该连接发送。

在 Tuxedo 负载均衡场景，设备部署于 WSC 和 WSL 之间，通过 Tuxedo 虚拟服务接收来自 WSC 的请求，然后按照配置的策略将请求转发到对应的 Tuxedo 服务组。对于每个 Tuxedo 服务组，设备都会维护一个映射表，该表记录 WSH 端口号和后台服务的对应关系：

- 当收到 WSC 的请求时，服务组会根据请求的 WSH 端口号查询映射表中是否存在对应的后台服务。如果不存在，Tuxedo 服务组会按照首次选择算法选择一个后台服务。当后台服务返回 WSH 的 IP 和端口号时，设备会将该后台服务和端口号的对应关系记录到映射表里。
- 当 WSC 请求访问的 WSH 端口号存在于映射表中时，设备会找到对应的后台服务，将请求直接转发到该后台服务。

注意：

1. 分配给每个 Tuxedo 服务器的 WSH 端口号不能重叠。
2. 所有 WSH 的 IP 地址必须设置成 Tuxedo 虚拟服务的 IP 地址。
3. 每个 WSL 只能分配本机上的 WSH 进程。

11.2.10.1 配置举例

1. 配置 Tuxedo 虚拟服务。

```
Demo(config)#slb virtual tuxedo vs1 172.16.73.155 0
```

25. 配置 Tuxedo 后台服务。

```
Demo(config)#slb real tuxedo rs1 172.16.72.145 1000 icmp 3 3
```

26. 配置 Tuxedo 服务组。

```
Demo(config)#slb group method g1 tuxedo lc 10
```

27. 配置默认策略。

Tuxedo 负载均衡支持 Static 策略、Default 策略、QoS Clientport 和 QoS Network 策略。

```
Demo(config)#slb policy default vs1 g1
```

11.2.11 WebSocket 负载均衡

WebSocket 协议为基于浏览器的应用提供了一种无需打开多个 HTTP 连接就可以和服务器进行双向通讯的机制，可以让服务器主动推送数据给客户端。它由握手和数据传输两个部分组成。在握手期间，HTTP 协议升级为 WebSocket 后 WebSocket 连接建立，后续的 WebSocket 消息都会通过该连接进行双向传递。在该流程里，设备系统可以发挥反向代理服务器的角色，在客户端和服务器之间转发基于 HTTP/1.1 的 WS（WebSocket）和 WSS（WebSocket Secure）流量。

当需要向一组服务器转发 WS 和 WSS 流量时，系统支持基于已配置的 HTTP/HTTPS 负载均衡策略和算法进行负载均衡。除了 QoS Body 策略外，所有 HTTP/HTTPS 负载均衡支持的策略和算法都同样适用于 WS 和 WSS 流量的负载均衡。

11.2.12 DNS over HTTPS（DoH）

DNS over HTTPS（DoH）是一种新的协议，通过 HTTPS 协议来实现安全的 DNS 域名解析。DoH 通过使用 HTTPS 协议加密 DoH 客户端和 DoH DNS 服务器之间的数据，从而阻止了中间人攻击，保护了用户的安全和隐私。

SLB 功能可以在不升级或改造后台 DNS 服务器情况下实现客户网络基础设施对 DoH 的支持。SLB 提供 DoH 策略，将包含传统 DNS 服务器的服务组与 HTTPS 虚拟服务关联，从而实现对外提供 DoH 服务。

- **配置示例**

1. 配置 HTTPS 虚拟服务。

```
Demo(config)#slb virtual https "vs1" 192.168.12.62 443 arp 0
```

28. 配置 DNS 后台服务和后台服务组。

```
Demo(config)#slb real dns dns_rs1 192.168.2.2
Demo(config)#slb real dns dns_rs2 192.168.2.3
Demo(config)#slb group method g1 rr
Demo(config)#slb group member g1 dns_rs1
Demo(config)#slb group member g1 dns_rs2
```

29. 配置 DoH 策略将 HTTPS 虚拟服务与 DNS 后台服务组关联。

```
Demo(config)#slb policy doh vs1 g1
```

11.2.13 DNS over TCP

DNSTCP（DNS over TCP）功能基于 TCP 协议传递 DNS 数据报文。使用 DNSTCP 可以传输长度超过 512 字节的 DNS 报文，而无需截断报文，比通过基于 UDP 协议的 DNS 传输更好地保障了数据完整性和传输可靠性。

使用 SLB 的 QoS Dnsdomain、QoS Dnsqtype、DNSSEC 和默认策略，将包含 DNSTCP 类型的后台服务的服务组与 DNSTCP 类型的虚拟服务关联，系统可以提供 DNSTCP 类型的服务器负载均衡功能。

- **配置示例**

1. 配置 DNSTCP 虚拟服务。

```
Demo(config)#slb virtual dnstcp "vdns1" 192.168.12.62 53 arp 0
```

2. 配置 DNSTCP 后台服务和后台服务组。

```
Demo(config)#slb real dnstcp "r1" 192.168.2.2 53 0 dns-tcp 3 3
Demo(config)#slb real dnstcp "r2" 192.168.2.3 53 0 dns-tcp 3 3
Demo(config)#slb group method "gdns" rr
Demo(config)#slb group member "gdns" "r1" 1 0
Demo(config)#slb group member "gdns" "r2" 1 0
```

3. 配置默认策略将 DNSTCP 虚拟服务与 DNSTCP 后台服务组关联。

```
Demo(config)#slb policy default "vdns1" "gdns"
```

11.2.14 基于 IP/MAC 的二层负载均衡

基于 IP/MAC 的二层负载均衡允许用户将网络流量通过定义在一个接口的虚拟服务分布到定义在多个接口的后台服务上，流量的源及目的 MAC 地址都会改变。以上描述的接口既可是物理接口也可是虚拟接口（如 VLAN）。

当后台服务没有可用的 IP 地址，或者后台服务不是进入流量的最终目的地时（如：每一个数据报在转发到最终的目的地之前都被病毒扫描仪检查。这里最终的目的地是一个邮件服务器或是文件服务器），设备无法基于 IP 地址对网络流量进行负载均衡。而在这种环境中，可以使用基于 IP/MAC 的二层负载均衡。

不同于高层的负载均衡（四层或是七层），二层流量负载均衡不使用传统的定义在接口（通常是 Port1 接口）上的 IP 地址或“IP 地址+端口”来选择负载均衡算法。只要用户的请求流量可被路由到已定义二层虚拟服务的设备接口，那么基于配置在多个接口上的后台服务负载均衡算法，流量可被均衡分布。在二层负载均衡的处理过程中，所有网络数据报的源 IP 和目的 IP 地址均未改变。

二层负载均衡有特定的虚拟服务、后台服务、组算法和策略定义，其中虚拟服务由 IP 地址定义，在系统内部，相关的输入接口会通过这个地址找到。而后台服务可以由 IP 地址或 MAC 地址定义。在系统内部，相关的输出接口将被找到。

二层负载均衡仅支持默认策略和备份策略，支持的算法有轮询（rr）、哈希 IP（hi）和一致性哈希 IP（chi）

通常，在应用二层负载均衡的场景中，系统将转发目的 IP 地址不是本地 IP 地址的数据包。如需要转发目的 IP 为本机 IP 地址（如 VIP、接口 IP 等）的数据包，管理员可以将这些本机 IP 配置为二层负载均衡的本机例外 IP。详情参考小节 11.3.4.3 二层负载均衡本机例外 IP 配置。

二层负载均衡的健康检查：

二层负载均衡健康检查支持所有四层和七层上支持的健康检查方法。

二层负载均衡只可以配置附加的健康检查。

OSI 模型四层基于端口范围的负载均衡可定义在二层负载均衡虚拟服务上，使系统更灵活。在端口范围内的 TCP 和 UDP 流量将被负载均衡。不在定义端口范围内的流量将不被负载均衡，将等同于正常的穿过流量被路由。

二层负载均衡的局限：

- 虚拟服务和后台服务必须定义在不同的物理或虚拟接口。这将有助于区分一个数据报命中的是二层的虚拟服务或是来自二层的后台服务。
- 每一个接口最多仅可以配置一个 IPv4 虚拟服务和一个 IPv6 虚拟服务。
- 一个二层负载均衡的真实服务仅可以关联到一个二层负载均衡组，因此一个来自二层负载均衡真实服务的数据报可被路由到确定的二层负载均衡虚拟服务。
- 基于 IP 地址或 MAC 地址的二层后台服务不可以定义在相同的接口或包含在相同的二层负载均衡组中。

若一个数据报可同时匹配二层、四/七层负载均衡或网络地址转换，那么二层负载均衡配置将是最高优先级。若二层负载均衡设置没有匹配，那么数据报将被传送到其它的模块。

11.2.15 基于 IP 的三层负载均衡

基于 IP 的三层负载均衡，将访问同一个虚拟 IP 的 TCP 和 UDP 网络流量分流到多个后台服务器。和四层负载均衡相比，三层负载均衡不考虑端口因素。目前仅支持 TCP 和 UDP 传输。

三层负载均衡只用 IP 地址定义虚拟服务和后台服务。并且三层负载均衡不用指定端口，因此比高层负载均衡适用范围更广，尤其适用于以下几类应用：

- **多端口应用**：一些会话协议需要绑定多个连接，一个初始连接，多个动态连接。
- **多协议应用**：多数流媒体传输协议使用 UDP 连接作数据传输，TCP 连接作控制信息传输。

三层载均衡只支持对后台服务做 ICMP 类型的健康检查。但是，可以根据具体应用自行定义附加健康检查。

11.2.16 基于端口段的负载均衡

基于端口段的负载均衡允许用户定义带有端口范围的虚拟服务，系统将监听指定范围内的所有端口，并将请求分布到后台服务组，这些组可以定义一个端口范围，也可以定义一个单独的端口。

用户使用基于端口段的负载均衡可以将访问动态端口的流量分配到设定的后台服务。和三层负载均衡相比，基于端口段的负载均衡适用于多端口的应用，例如 SIP，但不适用于多协议的应用，例如 RTSP。基于端口段的负载均衡支持现有四层和七层协议，因此可以使用更多智能的方法和策略达到更高效的负载均衡。

用于定义端口范围的 CLI 命令和二层负载均衡的相同。虚拟服务和后台服务都可以定义为带端口段的。并且一个虚拟服务可以带有多个端口段。而后台服务必须包括全部端口。原因是后台服务端口要由实际使用的虚拟服务端口决定。

基于端口段的负载均衡不需要特殊类型的虚拟服务和后台服务。一旦虚拟服务或后台服务被附加了端口范围，就会遵循端口段负载均衡的特定流程。

注意：端口段的虚拟服务可以映像到全端口的后台服务也可以是单端口后台服务。后面的例子中，来自多个端口的请求最终是由后台服务的一个端口处理。

与原有四/七层虚拟服务相比，基于端口段的虚拟服务优先级较低。也就是说，如果请求同时符合四/七层和端口段的虚拟服务，将会先命中四/七层虚拟服务。

注意，使用端口段负载均衡功能有一些限制：

- 使用端口段的后台服务和使用单端口的后台服务不可以在同一组。
- 端口段类型的后台服务和组只能关联端口段的虚拟服务。
- 端口段类型的虚拟服务可以关联端口段的后台服务，也可以关联单端口的后台服务。
- 同一个 IP 地址对应的端口范围不允许重迭。但是单端口的后台服务所使用的端口号可以在端口段指定的范围内。当查找虚拟服务的时候，单端口的虚拟服务优先级较高。
- FTP 类型的虚拟服务和后台服务不支持端口段负载均衡。

- 端口段后台服务由于其端口为 0，它的的健康检查类型只能是“none”或者“ICMP”。如果需要配置其他类型的健康检查，我们需要使用附加健康检查。

例如：

```
Demo(config)#slb real http rs1 10.3.0.20 0 1000 none
Demo(config)#slb real health a1 rs1 10.3.0.20 80 http
```

11.2.17 终端服务器负载均衡

设备通过使用终端服务会话目录服务（Terminal Services Session Directory Service）保持会话持续性，从而实现 RDP（Remote Desktop Protocol，远程桌面协议）流量在终端服务器群中的负载均衡。该功能可以使用户在中断了某个正在运行的应用的会话之后，重新连接到同一个正在运行的应用的同一个会话，而不论本次中断是有意而为还是网络终端故障造成的。

终端服务会话目录服务是一个数据库，主要用于保持负载均衡群中的终端服务器的会话的持续性。该数据库维护着一个用户名列表，这些用户名与会话 ID 一一关联，而会话 ID 又与负载均衡终端服务器组中的服务器相连。该数据库既可以安装在与组群中终端服务器不同的服务器上，也可以安装在终端服务器群组的一台终端服务器上。

当客户端向设备发送一个认证请求时，设备首先会把该请求发送给群组中的某台终端服务器，例如 TS1。TS1 在客户端弹出一个登录窗口，用户输入用户名和密码后，TS1 验证用户名和密码，并且向会话目录数据库查询该用户名。如果会话目录数据库发现在群组中的某台终端服务器，例如 TS2，已经有该用户名的一个会话，那么会话目录将把这些相关信息（包括 TS2 服务器的 IP 地址和端口号）发送给 TS1，TS1 则会给该客户端发送一个带有 TS2 IP 地址和端口号的 Routing Token 并中断与该客户端的连接。在此之后，该客户端再给设备发送一个带有刚才收到的 Routing Token 的认证请求。设备在收到该请求之后会根据 Routing Token 中的 IP 地址和端口号重新建立客户端与 TS2 之间的连接。

11.2.18 RADIUS 负载均衡

RADIUS（Remote Authentication Dial-In User Service，远程用户拨入认证系统）服务器负载均衡方案中，当 RADIUS 代理发往后台 RADIUS 服务器的请求数据报经过设备时，首先，设备会根据包中的 username 或者 session ID 字段使用相应的负载均衡方法（对“username”字段采用“radchu”或“radpsun”方法，对“session ID”字段采用“radchs”方法）将请求映射到某一后台 RADIUS 服务器；然后，检查与该后台服务器的连接是否存在。若连接存在，则直接将该 RADIUS 包发送到该后台服务器；否则，先创建与该后台服务器的连接并保存该连接信息（但由于设备与后台服务器之间的连接是有时限的，一旦超时，该连接将会被删除），

随后再将 RADIUS 包发送到该后台服务器。这样，设备可以在多个后台服务器之间实现对 RADIUS 请求数据报的均衡处理。

注意：相比“radchu”方法，“radpsun”方法增强了 RADIUS 会话保持功能，即相同请求再次访问服务组时可以刷新 RADIUS 会话的超时时间。

11.2.19 快速转发

快速转发是一个基于多核系统、使用多线程无锁架构实现的新的四层负载均衡模块。这个多线程无锁架构能够最大程度的发挥多核系统的优势。与传统的四层负载均衡相比，通过快速转发功能实现的四层负载均衡的性能明显大大超过了传统的四层负载均衡的性能。这个功能通过命令“**slb directfwd on <virtual_service>**”启用，通过命令“**slb directfwd off**”禁用。

快速转发功能既支持 IPv4，也支持 IPv6 类型的 TCP 报文。

由于目前这个新的多线程无锁架构中实现的功能有限，所以快速转发功能还有一些限制：

- 只能用于 TCP 类型的负载均衡，而且暂时只能支持静态、默认和备份策略。需要注意的是启用快速转发功能后，sr 算法不生效，其它的四层负载均衡算法都支持。
- 不能在不同的 MTU 环境中使用，即所有的接口都必须共享一个 MTU；否则，连接将无法传送太大的数据报。
- 不能处理 IP 分片。

注意：

- 快速转发功能使用的接口越多，其性能就越高。
- 三角传输模式下的服务器负载均衡不支持快速转发功能。

快速转发 Syncache

快速转发 Syncache 能够切实有效地防止后台服务器受到 SYN flood DOS 攻击。当快速转发 Syncache 功能启动，所有来自客户端的 SYN 包不会被直接转发给后台服务器。设备会先缓存 SYN 包中的有用信息，并给客户端发送一个 SYN-ACK 包。然后设备会收到客户端回的一个 ACK 包。最后，设备就与后台服务建立一个 TCP 连接。

注意：当快速转发 Syncache 功能打开后，客户端与后台服务之间不能协议 MTU。

11.2.20 后台服务平稳关闭和温暖上线

11.2.20.1 后台服务平稳关闭

预设情况下，当一个后台服务被禁用或者删除，设备的服务器负载均衡就不再向该后台服务发送会话请求。但是，使用和 cookie 有关的负载均衡算法和策略并且已经成功建立的会话除外，如保持 cookie（pc）、插入 cookie（ic）、重写 cookie（rc）和基于会话 ID 的通用保持方法（persistence）等，则会继续向该服务器发送请求。服务器负载均衡支持一种“平稳关闭”功能：即一个后台服务已经被关闭后，设备仍将把原先的老客户（带有老的 cookie 值）发来的 HTTP 请求转给它以保持服务连续性。但是任何新客户发来的请求将不再分配给已关闭的服务。如下图所示。

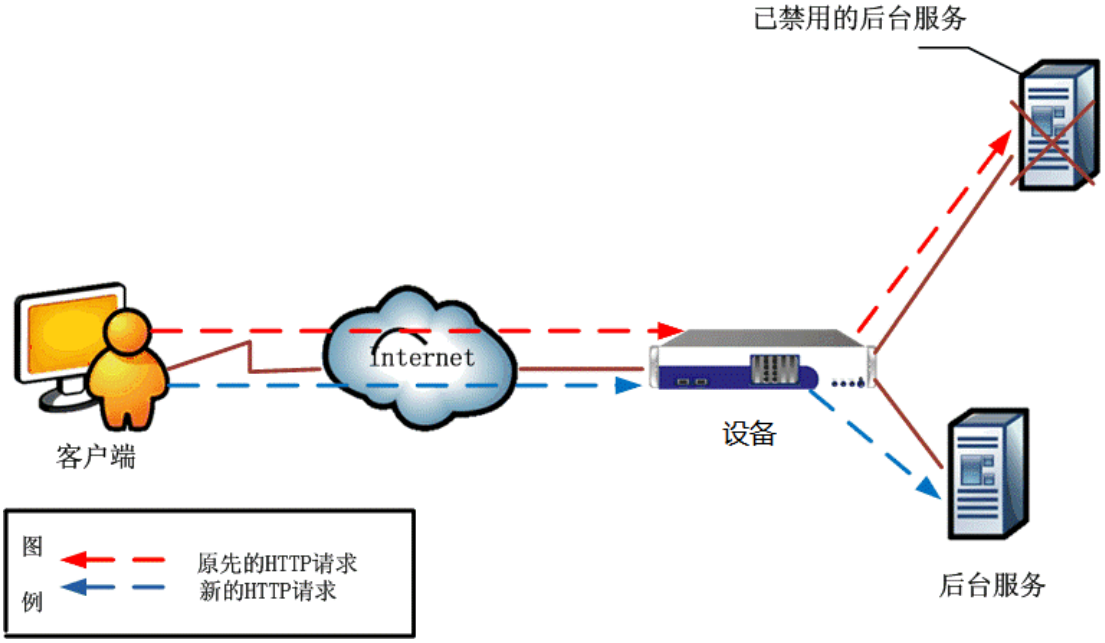

图11–10 服务器负载均衡后台服务的平稳关闭功能

当用户使用“**slb real disable**”命令禁用后台服务器后，可以使用“**show statistics slb real**”命令查看该服务器的状态：

```
Demo(config)#show statistics slb real http service
Real service service 10.8.6.42 80 DOWN INACTIVE(waiting)
        Main health check: 10.8.6.42 80 tcp DOWN
        Max Conn Count:              1000
        Current Connection Count:    4572
        Outstanding Request Count: 4215
        Total Hits:                  311
        Total Bytes In:              39431
        Total Bytes Out:             53466
```

```
        Total Packets In:           7541
        Total Packets Out:          3252
        Average Bandwidth In:       543 Kbps
        Average Bandwidth Out:      748 bps
        Average Response time:      32.000 ms
```

如上面的输出信息所示，名为“service”的后台服务器状态已经变成“INACTIVE(waiting)”，表示该服务器上还有未断开的连接，正处于平稳关闭过程中。在这个过程中，原先老的客户端的 HTTP 请求还会继续被转发至该服务器，而来自新的客户端的请求不会被转发到该服务器。

经过一段时间后，再使用“**show statistics slb real**”命令查看该服务器的状态：

```
Demo(config)#show statistics slb real http service
Real service service 192.168.10.10 80 DOWN INACTIVE(suspend)
        Main health check: 192.168.10.10 80 tcp DOWN
        Max Conn Count:             1000
        Current Connection Count:   0
        Outstanding Request Count: 0
        Total Hits:                 0
        Total Bytes In:             0
        Total Bytes Out:            0
        Total Packets In:           0
        Total Packets Out:          0
        Average Response time:      0.000 ms
```

如上面的输出信息所示，此时后台服务器“service”的状态显示为“INACTIVE(suspend)”，表示已经完全关闭。

11.2.20.2 后台服务温暖上线

后台服务器被启用后，在还没有准备好工作之前就可能收到大量的连接请求。服务器上的流量骤增，可能导致其不可用。为了解决这个问题，设备支持后台服务“温暖上线”，可以为后台服务器设置恢复时间和预热时间。

在后台服务器启动后的一段时间内，设备先不转发连接请求至该服务器，这段时间称为“恢复时间”（recovery time）。恢复时间结束后，后台服务器进入“预热时间”（warm-up time），即服务器开始处理使用者请求，但是设备只转发少量请求给它，随着时间的推移逐步增加转发的请求数量，直到达到服务器的最大连接数，表示其进入正常的工作状态。

管理员可以使用命令“**show statistics slb real**”命令查看刚刚启动的服务器的状态信息。如下所示，当后台服务器处于恢复时间时，其状态显示为“UP(softup)”，在这段时间内没有任何连接被转发到该服务器。

```
Demo(config)#show statistics slb real http service
```

```
Real service service 192.168.10.10 80 UP (softup) ACTIVE
        Main health check: 192.168.10.10 80 tcp ACTIVE
        Max Conn Count:                 1000
        Current Connection Count:   0
        Outstanding Request Count: 0
        Total Hits:                     0
        Total Bytes In:                 0
        Total Bytes Out:                0
        Total Packets In:               0
        Total Packets Out:              0
        Average Response time:          0.000 ms
```

11.2.21 后台服务组成员激活

系统支持管理员为后台服务组设置成员激活策略，控制由服务组中哪些后台服务处理流量。

后台服务组支持两种成员激活模式：

- 基于优先级的激活模式：基于服务组内后台服务的优先级降序激活允许使用数量的后台服务。
- 基于优先级子组的激活模式：按照优先级将服务组内的后台服务划分为优先级子组，最高优先级子组中的后台服务处理流量。

此外，系统支持对服务组或优先级子组的可用性进行实时监测，并在服务组或优先级子组不可用时执行流量失效切换。

基于优先级的激活模式下，如果可用的后台服务的数量小于最小限制，系统认为服务组不可用，将其流量切换到备份策略关联的服务组。

基于优先级子组的激活模式下，如果最高优先级子组中可用后台服务数量小于最小限制，系统认为该子组不可用，将流量切换到次高优先级的子组处理。当所有优先级子组都不可用时，所有发往该服务组的流量将被切换至备份策略关联的服务组。

默认情况下，服务组使用基于优先级的激活模式，允许使用所有的后台服务，不检测服务组的可用性。

➢ **配置示例**

为服务组添加后台服务，并为成员设置优先级：

```
Demo(config)#slb group member "g1" "r1" 1 3
Demo(config)#slb group member "g1" "r2" 1 3
Demo(config)#slb group member "g1" "r3" 1 3
```



```
Demo(config)#slb group member "g1" "r4" 1 2
Demo(config)#slb group member "g1" "r5" 1 2
Demo(config)#slb group member "g1" "r6" 1 2
Demo(config)#slb group member "g1" "r7" 1 1
Demo(config)#slb group member "g1" "r8" 1 1
Demo(config)#slb group member "g1" "r9" 1 1
```

为服务组配置成员激活策略：

```
Demo(config)#slb group activation g1 0 2 1
```

完成上述配置后，为服务组 g1 启用了基于优先级子组的激活模式。优先级为 3 的子组（r1，r2 和 r3）将处理该服务组的流量。

11.2.22 基于主机节点的后台服务管理

主机节点是以 IP 地址或域名信息为核心的逻辑对象，用于帮助识别并管理拥有相同 IP 地址或域名的后台服务。

拥有相同 IP 地址或域名的后台服务会被划分至一个主机节点。管理员可以启用或禁用一个指定的主机节点。当启用或禁用一个主机节点后，所有与该主机节点关联的后台服务将被启用或禁用，以此可以实现基于 IP 地址或域名的后台服务批量管理。

目前，系统支持以下两种方式创建主机节点：

- 自动创建：在创建后台服务时，系统会基于后台服务的 IP 地址或域名自动创建主机节点。自动创建的主机节点只能在后台服务删除时被自动删除。自动创建的主机节点无法保存配置。
- 手动创建：通过“**slb node name**”命令手动创建。手动创建的主机节点需要手动删除。手动创建的主机节点可以保存配置。

默认情况下，所有主机节点均为启用状态。系统支持手动配置的主机节点总数上限和后台服务上限一致。

11.3 服务器负载均衡配置示例

配置 SLB 分以下几个步骤：

1. 定义后台服务。
4. 定义一个后台服务器组以及该组上的负载均衡算法。
5. 添加后台服务到服务器组。
6. 定义虚拟服务来监听客户的请求。

7. 定义策略把虚拟服务和后台服务器组关联起来。

11.3.1 HTTP/TCP/FTP/UDP/HTTPS/TCPS/DNS 协议的负载均衡配置

这部分内容包含了一个以上的服务器负载均衡配置，首先我们将给出基本的服务器负载均衡配置示例。随后我们将介绍更多基于不同策略的示例。我们使用 HTTP 协议为例，其他协议的配置同本例类似。

11.3.1.1 配置向导

本配置示例将基于以下拓扑。

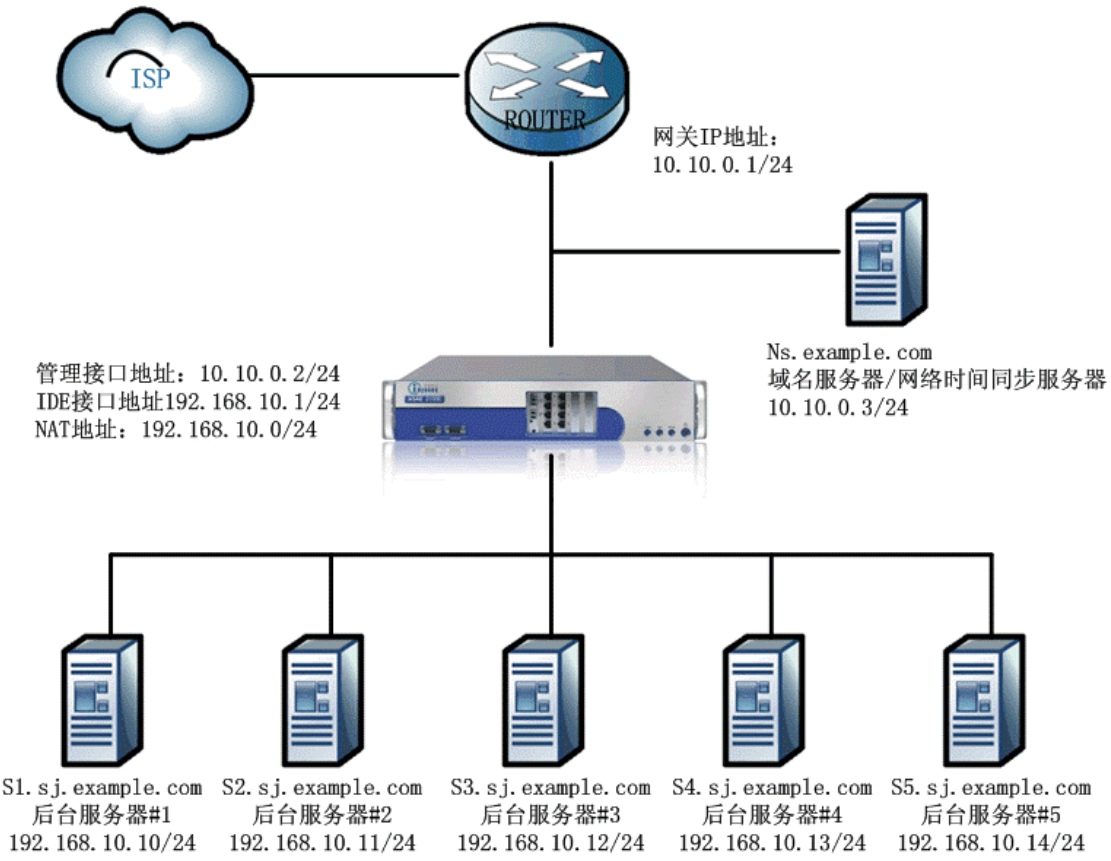

图11–11 服务器负载均衡配置网络拓扑

下面列出了配置 HTTP 协议的服务器负载均衡所需用的配置命令，相关命令的描述信息，请参考命令行使用手册。

表11–4 服务器负载均衡配置命令

配置操作	命令行
配置后台服务	**slb real tcp** *<real_service> <ip> <port> [max_connection] [hc_type] [hc_up] [hc_down]*

配置操作	命令行
	slb real ftp *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real http** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real udp** *<real_service> <ip> <port> [max_connection] [hc_up] [hc_down] [timeout] [hc_type]* **slb real https** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real tcps** *<real_service> <ip> <port> [max_connection] [hc_type] [hc_up] [hc_down]* **slb real dns** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down] [timeout]* **slb real health** *<add_hc_name> <real_service> <ip> <port> [hc_type] [hc_up] [hc_down]*
定义服务组的负载均衡算法	**slb group method** *<group_name>* **[rr|pu|sr]** **slb group method** *<group_name>* **hc** *[first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **ic** *[cookie_name] [path_attribute] [first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **rc** *[cookie_name] [offset] [first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **lc** *[threshold_granularity] [yes|no]* **slb group method** *<group_name>* **hh** *<header_name> [first_choice_method] [threshold_granularity] [prefix] [delimiter]* **slb group method** *<group_name>* **ec** *<cookie_name> [first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **lb** *[threshold_granularity]* **slb group method** *<group_name>* **pto** <tcp_option_kind> [*first_choice_method*]
将后台服务添加到服务组	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]* **slb health** *<group_hc_name> [type] [interval] [timeout] [hc_up] [hc_down] [http_method] [url_path] [expected_codes]* **slb group health** *<group_name> <group_hc_name>*
定义虚拟服务	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb virtual {tcp|tcps|udp}** *<virtual_service> <vip> <vport> [arp_support] [max_connection]* **slb virtual rtsp** *<virtual_service> <vip> [vport] [mode] [arp_support] [max_connection]* **slb virtual {ftp|ftps}** *<virtual_service> <vip> [vport] [max_connection]*

配置操作	命令行
	slb virtual ip *<virtual_service> <vip> [max_connection]* **slb virtual l2ip** *<virtual_service> <vip> [gateway_ip]*
将服务组绑定到虚拟服务上	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}* **slb policy static** *<virtual_service> <real_service>* **slb policy qos url** *<policy_name> {virtual_service|vlink_name} {group_name|vlink_name} <url_string> <precedence>* **slb policy qos cookie** *<policy_name> {virtual_service|vlink_name} {group_name|vlink_name} <policy_string> <precedence>* **slb policy persistent cookie** *<policy_name> {virtual_service|vlink_name} <group_name> <cookie_name> <precedence>* **slb policy qos hostname** *<policy_name> {virtual_service|vlink_name} {group_name|vlink_name} <host_name> <precedence>* **slb policy redirect** *<policy_name> <virtual_service> <group_name> <redirected_from_host>*

11.3.1.2 配置示例

1. 定义后台服务。

通过命令“**slb real**”来配置后台服务，如：

```
Demo(config)#slb real http service1http 192.168.10.10
```

当我们定义一个 HTTP 类型的后台服务时，如果只配置后台服务器名称和 IP 地址的话，下列配置条目将依照以下缺省值自动配置：

- 端口：80
- 最大连接数：0
- 健康检查类型：TCP
- 确定该服务健康需连续成功检查的次数：3
- 确定该服务不健康需连续失败检查的次数：3

对于第二个后台服务，我们将定义 HTTP 类型的健康检查。

```
Demo(config)#slb real http service2http 192.168.10.11 80 1000 http 3 3
```

对于其他的后台服务，我们使用下面的命令进行配置。

```
Demo(config)#slb real http service3http 192.168.10.12
Demo(config)#slb real http service4http 192.168.10.13
Demo(config)#slb real http service5http 192.168.10.15
```

至此，全部五个 HTTP 类型的后台服务都已经配置完成。若要查看已配置后台服务的相关信息，可以使用“**show slb real http** *[real_name]*”命令，参数“[real_name]”用来指定要查看的某个后台服务的名称；如果不指定该参数，则表示查看所有已配置后台服务的配置信息。

我们还可以为后台服务配置附加健康检查：

```
Demo(config)#slb real health a1 service1http 192.168.10.10 80 http
Demo(config)#slb real health a1 service1http 192.168.10.10 8080 http
```

这样，只有当主要健康检查（TCP 类型的 192.168.10.10:80）和两个附加健康检查（HTTP 类型的 192.168.10.10：80 和 HTTP 类型的 192.168.10.10:8080）全部正常时，后台服务“service1http”才会被设备识别为健康状态。

关于后台服务的维护管理，有时候我们会因为种种原因需要暂时禁用某些后台服务，在这种情况下我们可以使用“**slb real disable** <*real_name*>”命令。

```
Demo(config)#slb real disable service1http
```

这样，我们在使用“**show slb real all**”命令查看后台服务配置的时候，会发现 service1http 已经被禁用。如果想重新启用它，可以使用“**slb enable** <*real_name*>”命令。

```
Demo(config)#slb real enable service1http
```

预设情况下，当一个后台服务被禁用或者删除，设备的服务器负载均衡就不再向该后台服务发送会话请求。但是，和 cookie 有关的负载均衡算法和策略，如保持 cookie（pc）、插入 cookie（ic）、重写 cookie（rc）等，服务器负载均衡支持一种“平稳关闭”功能：即一个后台服务已经被关闭后，设备仍将把原先的老客户（带有老的 cookie 值）发来的 HTTP 请求转给它以保持服务连续性。但是任何新客户发来的请求将不再分配给已关闭的服务。如下图所示。

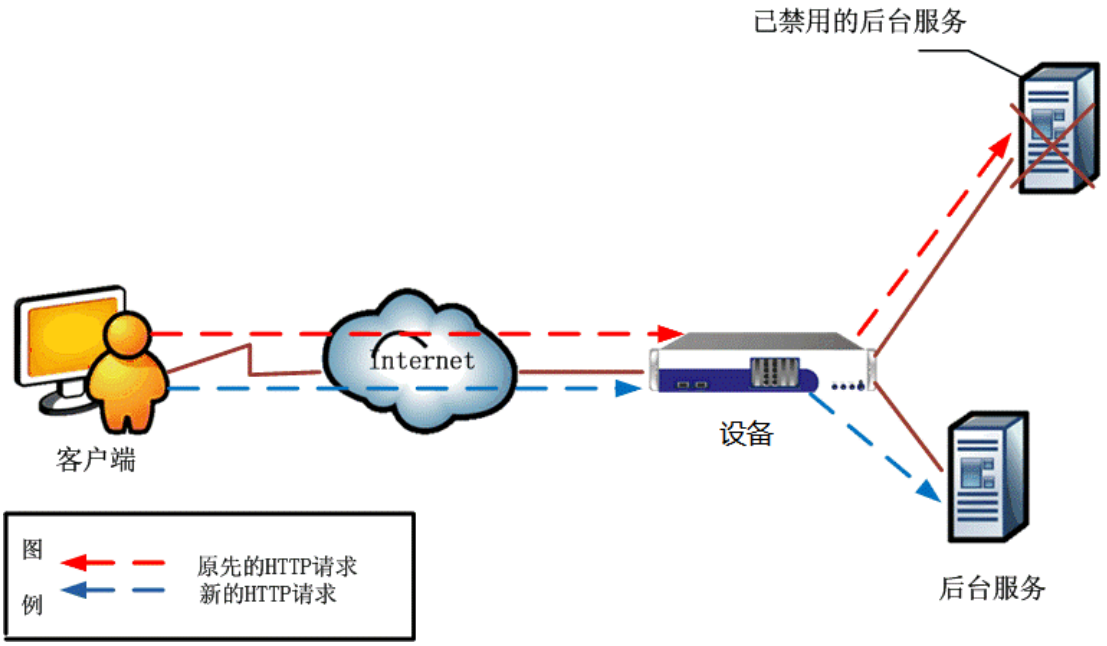

图11–12 服务器负载均衡后台服务的平稳关闭功能

```
Demo(config)#slb real http r1 10.3.0.20 80
Demo(config)#slb group method g1 pc
Demo(config)#slb group member g1 r1 "server1"
Demo(config)#slb real disable r1
```

2. 定义服务组。

现在后台服务已经被定义好了，接下来需要为他们配备服务组。我们使用“**slb group method**”命令。

下面的示例中，我们建立了一个名叫“rrgroup”的服务组，并且将它的负载均衡算法设置为轮询（rr）。

```
Demo(config)#slb group method rrgroup rr
```

3. （可选）配置服务组健康检查：

定义一个服务组健康检查条件。

```
Demo(config)#slb health check1 http 30 3 3 3 GET / "200"
```

将服务组健康检查条件与指定的服务组关联。

```
Demo(config)#slb group health rrgroup check1
```

4. 将后台服务添加到服务组中。

接下来我们使用“**slb group member**”命令将先前建立的后台服务添加到服务组中。

```
Demo(config)#slb group member rrgroup service1http
Demo(config)#slb group member rrgroup service2http
Demo(config)#slb group member rrgroup service3http
Demo(config)#slb group member rrgroup service4http
Demo(config)#slb group member rrgroup service5http
```

5. 定义虚拟服务。

虚拟服务相当于一个访问点，由一个虚拟 IP（VIP）和一个端口组成。客户将把请求发送到这个虚拟 IP 和端口对。之所以称为虚拟服务是因为真正的请求不是在此回应的，而是通过某种策略转发请求到后台服务组中的某个后台服务器上。

注意：虚拟 IP 地址不能被配置为任何管理 IP 地址。虚拟 IP 地址必须同某个系统接口配置在同一个子网之下（除了 0.0.0.0 和 noarp 的情况）。如果虚拟 IP 地址不匹配任何接口的子网，设备将不允许它进行配置。如果虚拟 IP 地址没有同任何策略绑定，客户端将会从设备接到一个“503 Service Unavailable response”错误。如果服务组中的一个后台服务器失效，客户端也将接收到 503 号错误。

```
Demo(config)#slb virtual http virtual1http 10.10.0.10 80
```

我们现在配置了一个虚拟 IP 地址 10.10.0.10，它通过 80 端口监听请求。下一步我们必须定义一个策略来引导到达设备的请求。

6. 定义策略。

最后一步是定义一个预设策略，将虚拟服务绑定到四层服务组中。

```
Demo(config)#slb policy default virtual1http rrgroup
```

这条命令将 virtual1http 的默认策略设置为“rrgroup”。我们现在建立了一个使用轮询算法的四层服务器负载均衡。我们可以通过 URL 来测试我们配置的结果：http://10.10.0.10/。

对于已配置好的虚拟服务和后台服务，系统支持直接修改其 IP 地址、端口和最大连接数，无需删除原有配置后重新添加。

注意：

1. 二层 IP 类型的虚拟服务（通过“**slb virtual l2ip**”命令配置）不支持直接修改 IP 地址。
2. 修改后台服务的 IP 地址或端口后，通过“**show statistics slb real**”命令查看的该后台服务所有“Total…”值统计将被清零；如果后台服务为 HTTP/HTTPS 类型，通过“**show statistics slb real**”命令查看的该后台服务的“HTTP response code”统计也将被清零；如果后台服务加入了服务组，通过“**show statistics slb group**”命令查看的该服务组成员命中统计也将被清零。

对于已配置好的后台服务组，在服务组未关联虚拟服务也未添加后台服务成员之前，算法可以直接修改。例如，使用 pu 算法的服务组“g1”，如果服务组内未添加任何成员，可以直接修改为任意一种系统支持的其他算法。

在服务组通过策略关联了虚拟服务或添加了后台服务成员之后，是否支持修改会因为算法类型、协议类型等因素各不相同，系统会给出相应的提示。

11.3.1.3 基于服务器负载均衡策略的配置示例

- **QoS URL**

使用 QoS URL，我们可以建立策略查看 URL 字符串，基于查看的结果来做出转发决定。在我们的下一个例子里，我们将建立两个服务组，服务组“group1”将处理所有 URL 中有关“.jpg”的请求，服务组“group2”将处理所有 URL 中有关“english”的请求。

Group 1：URL: .jpg

成员：

- S1.sj.example.com

- S2.sj.example.com

Group 2：URL: ‘english’

成员：

- S3.sj.example.com
- S4.sj.example.com

1. 建立两个服务组。

```
Demo(config)#slb group method group1 rr
Demo(config)#slb group method group2 rr
```

2. 将后台服务器添加到服务组。

```
Demo(config)#slb group member group1 service1http
Demo(config)#slb group member group1 service2http
Demo(config)#slb group member group2 service3http
Demo(config)#slb group member group2 service4http
```

3. 设置策略和每个服务组有关的 URL。

```
Demo(config)#slb policy qos url url_pol_1 virtual1http group1 ".jpg" 1
Demo(config)#slb policy qos url url_pol_2 virtual1http group2 english 2
```

4. 配置默认策略。

如果设备接收到的请求的 URL 当中既没有.jpg 也没有“english”，设备会返回一个“503 Service Unavailable”错误信息，因为它不知道该如何处理这个请求。如果网络中所有的请求的 URL 中都只包括这两个信息或其中之一的话，不会出现问题。否则，最好定义一个预设策略，告诉设备应当如何处理含有其他 URL 信息的请求。

在我们的示例中，我们让所有不包含.jpg 和“english”信息的请求包转发到“group2”。

```
Demo(config)#slb policy default virtual1http group2
```

➢ QoS cookie

使用 QoS cookie 策略，我们可以基于 cookie 名称和值等七层协议信息，来决定负载均衡的结果。下面的图显示了 QoS cookie 策略的原理，我们将要建立两个服务组，并且定义 cookie，然后建立策略使我们的网络可以基于 cookie 进行负载均衡。如下图所示。

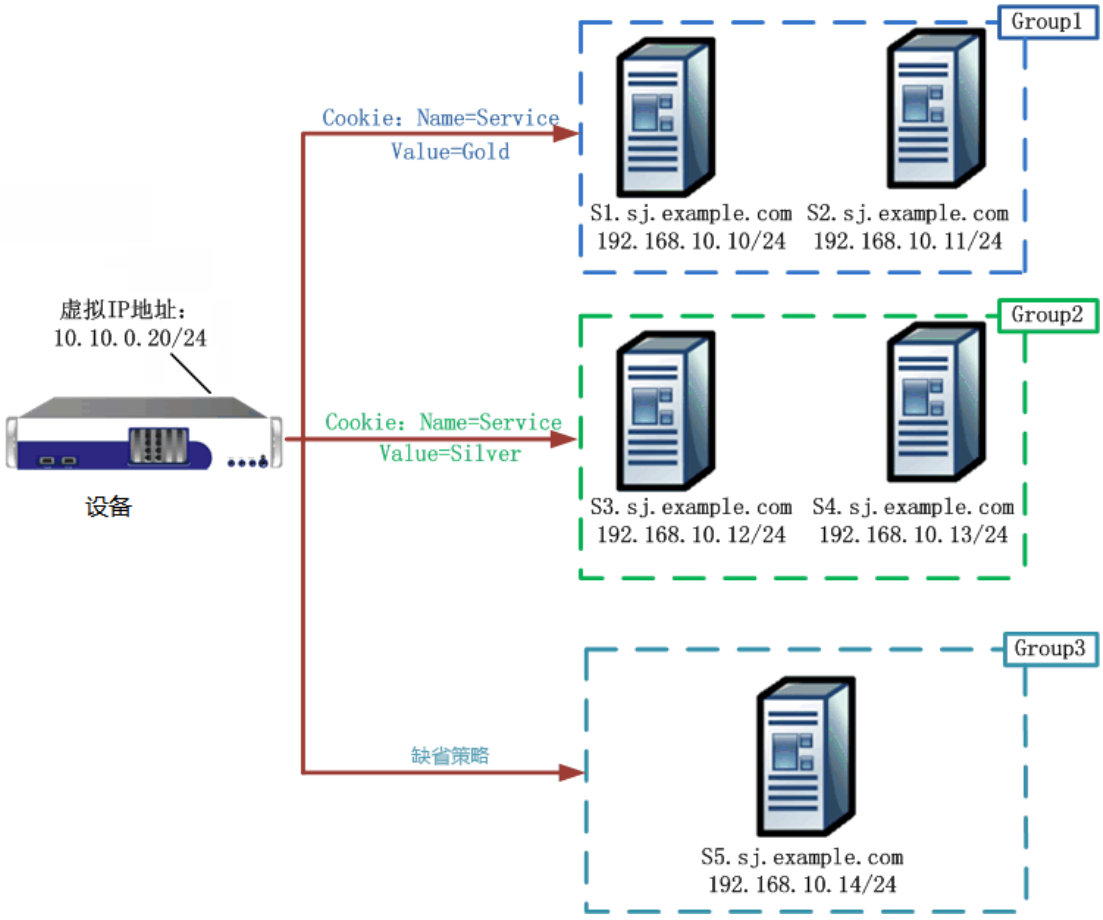

图11–13 QoS cookie 策略

因为要采用 QoS cookie，其工作在现有的后台服务组之上，我们可以用四层规则来定义组。

Group1：轮询

成员：

- S1.sj.example.com
- S2.sj.example.com

Group2：轮询

成员：

- S3.sj.example.com
- S4.sj.example.com

Group3：轮询

成员：S5.sj.example.com

命中这一策略的数据报处理如下：

1. 如果请求中有一个 cookie：名称为“Service”，值为“Gold”，这个包会被送到 group1；
2. 如果请求中有一个 cookie：名称为“Service”，值为“Silver”，这个包会被送到 group2；
3. 如果请求中没有指定的 cookie，这个包会被送到 group3。

Group3 中将完成请求的首次 cookie 载入。如果一个客户从来没有到过任一个网站，其请求中不会带有 cookie，因为 cookie 的首次安装是由服务器完成的。这样的请求不会匹配 group1 或者 group2，而会被缺省送到 group3。在客户通过 group3 访问了网站，服务器为其设置了 cookie 之后，客户再次访问网站时，浏览器会在 HTTP 请求头里加上这个 cookie。这些 cookie 可以保证我们在 group1 或者 group2 中找到合适的后台服务。

注意：如果我们没有缺省策略，服务器会返回“503 Service Unavailable”的错误信息。

现在我们已经清楚了会发生什么情况，可以开始配置这个应用。我们不需要重新定义后台服务，可以复用前面例子里所定义的后台服务。

1. 定义服务组“group1”、“group2”和“group3”。

```
Demo(config)#slb group method gold_group rr
Demo(config)#slb group method silver_group rr
Demo(config)#slb group method cookie_set_group rr
```

2. 将后台服务器添加到组。

```
Demo(config)#slb group member gold_group service1http
Demo(config)#slb group member gold_group service2http
Demo(config)#slb group member silver_group service3http
Demo(config)#slb group member silver_group service4http
Demo(config)#slb group member cookie_set_group service5http
```

3. 定义缺省策略（group3）。这个组会安装 cookie。

```
Demo(config)#slb policy default virtual1http cookie_set_group
```

4. 为 group1 和 group2 设置 QoS cookie 策略。

```
Demo(config)#slb policy qos cookie gold_policy virtual1http gold_group "service=gold" 1
Demo(config)#slb policy qos cookie silver_policy virtual1http silver_group "service=silver" 2
```

- **保持 cookie**

保持 cookies 的负载均衡，定义 cookie 名称和值，并把它们关联到指定的后台服务。这和 QoS cookie 不同的是，可以根据请求包，直接选中指定的后台服务而不是组。因此保持 cookie 的配置也是完全不同的。

group1

Server1：Cookie 名称：Service

Cookie 值：GOLD

Server2：Cookie 名称：Service

Cookie 值：SILVER

group2

Server3：无 cookie 时的匹配服务

后台服务的定义已经完成。现在继续定义 cookie、服务组和策略。

1. 定义服务组“group1”和“group2”。

```
Demo(config)#slb group method group1 pc
Demo(config)#slb group method group2 rr
```

注意：保持 cookie 策略必须同哈希 cookie（hc）或者保持 cookie（pc）这两个负载均衡策略联用。

2. 将后台服务添加到服务组。

```
Demo(config)#slb group member group1 service1http gold
Demo(config)#slb group member group1 service2http silver
Demo(config)#slb group member group2 service3http
```

3. 为服务组 group2 设置预设策略。

```
Demo(config)#slb policy default virtual1http group2
```

4. 为服务组 group1 设置保持 cookie（pc）策略。

```
Demo(config)#slb policy persistent cookie perst_pol virtual1http group1 Service 1
```

注意：一个保持 cookie 组里的所有成员必须和同一个 cookie 名称相关联。

➢ QoS 主机名

基于 QoS 主机名（qh）的负载均衡根据 URL 中的主机名来做出选择。这也被称为虚拟主机（Virtual Hosting）。其配置和 QoS cookie 类似，但要使用 qos 主机名策略来替代 qos cookie 策略。如下图所示，我们的例子中有三个组：

c-one.example.com 关联到“customer one”，c-two.example.com 关联到“customer two”。任何带有其它的主机名的请求会命中缺省策略。

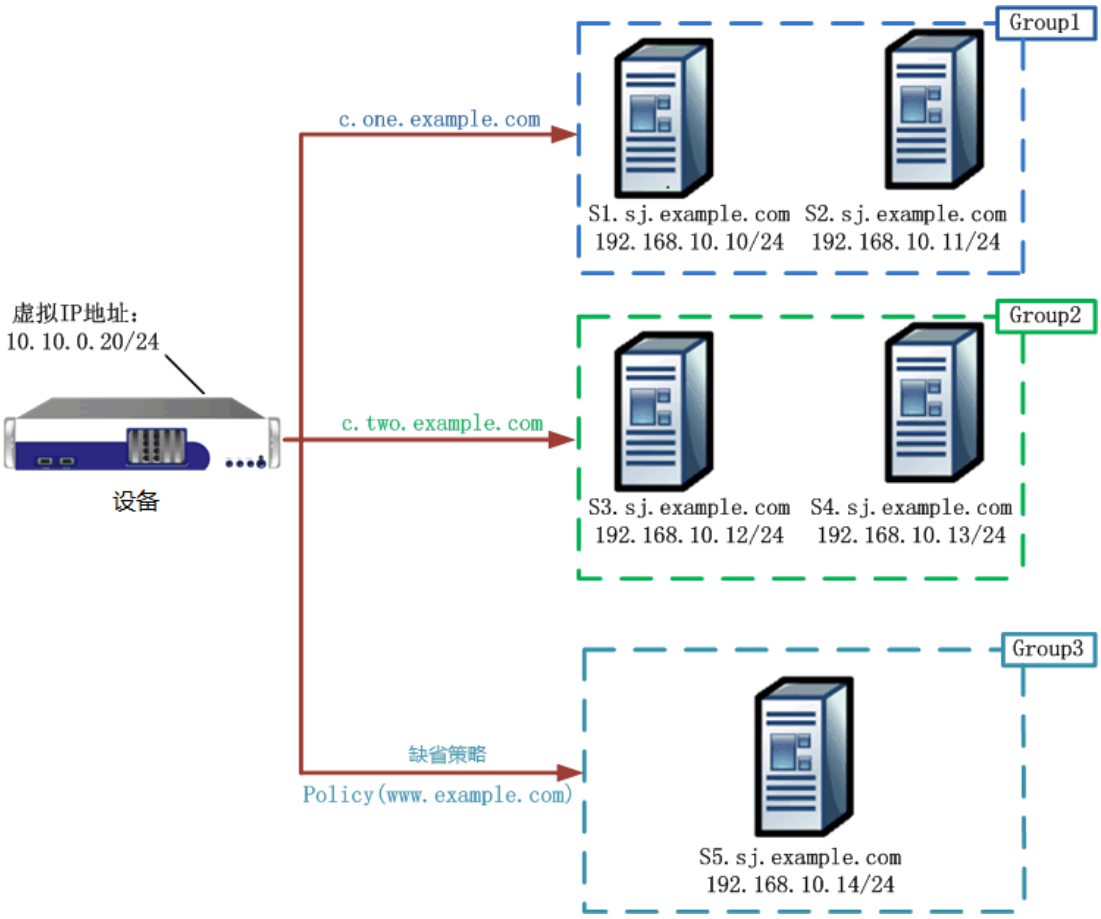

图11–14 QoS 主机名策略

Group 1：

主机名：c-one.example.com（轮询）

成员：

- S1.sj.example.com
- S2.sj.example.com

Group 2：

主机名：c-two.example.com（轮询）

成员：

- S3.sj.example.com
- S4.sj.example.com

Group 3：

任何其它主机名（缺省策略）

成员：S5.sj.example.com

1. 设置服务组。

```
Demo(config)#slb group method c_one_group rr
Demo(config)#slb group method c_two_group rr
Demo(config)#slb group method www_group rr
```

注意：QoS 主机名策略可以同任何负载均衡算法一起使用，但保持 cookie（pc）和保持 URL（pu）算法除外。

2. 将后台服务添加到服务组。

```
Demo(config)#slb group member c_one_group service1http
Demo(config)#slb group member c_one_group service2http
Demo(config)#slb group member c_two_group service3http
Demo(config)#slb group member c_two_group service4http
Demo(config)#slb group member www_group service5http
```

3. 为服务组“www_group”配置负载均衡策略。

缺省情况下，group3 处理所有主机名既不是 c-one.example.com，也不是 c-two.example.com 的请求。

```
Demo(config)#slb policy default virtual1http www_group
```

4. 为服务组“c_one_group”和“c_two_group”配置 QoS 主机名策略。

```
Demo(config)#slb policy QoS hostname c_one_pol virtual1http c_one_group
c_one.example.com 1
Demo(config)#slb policy QoS hostname c_two_pol virtual1http c_two_group
c_two.example.com 2
```

- **保持 URL**

保持 URL（pu）的负载均衡通过查找“tag=value”格式的字符串起作用。其典型应用是处理浏览器提交到服务器的窗体。这种方式中，可以通过提交给 CGI 脚本的参数值来保证后台服务的持续性。下面是窗体 URL 的例子：

```
http://www.example.com/find_user.pl?username=bob
```

这个 URL 的请求会命中我们定义的七层策略，被送到 server1。下面配置示例显示了如何配置一个保持 URL 的策略。

1. 为保持 URL 策略建立服务组。

```
Demo(config)#slb group method pu_group pu
Demo(config)#slb group method regular_group rr
```

注意：保持 URL 策略必须同保持 URL（pu）组和哈希 query（hq）组联用。

2. 将后台服务添加到服务组。

```
Demo(config)#slb group member pu_group service1http bob
Demo(config)#slb group member pu_group service2http janet
Demo(config)#slb group member pu_group service3http steve

Demo(config)#slb group member regular_group service4http
Demo(config)#slb group member regular_group service5http
```

3. 配置负载均衡策略。

```
Demo(config)#slb policy default virtual1http regular_group
Demo(config)#slb policy persistent url pol1 virtual1http pu_group username 1
```

- **插入 cookie**

插入 cookie（ic）是通过在服务器回应中插入 cookie 来长期保持客户端和服务器之间会话的一种负载均衡算法。

1. 创建一个服务组，并为其配置插入 cookie 算法。

```
Demo(config)#slb group method ic_group ic
```

注意：插入 cookie 策略必须同插入 cookie 组一起使用。

2. 为后台服务组设置插入 cookie 的属性。

```
Demo(config)#slb group option ic ic_group "expires=300 secure=yes"
```

3. 将后台服务加入到插入 cookie 服务组。

```
Demo(config)#slb group member ic_group service1http
Demo(config)#slb group member ic_group service2http
Demo(config)#slb group member ic_group service3http
Demo(config)#slb group member ic_group service4http
Demo(config)#slb group member ic_group service5http
```

4. 配置负载均衡策略，将服务组与虚拟服务关联起来。

```
Demo(config)#slb policy icookie pol1 virtual1http ic_group 1
Demo(config)#slb policy default virtual1http ic_group
```

缺省情况下，不需要指定 cookie 名称。系统会自动地随机生成一个 cookie 名称，并将该 cookie 名称保存在配置文件中。可以运行“**show slb group method**”命令来查看 cookie 名称。

```
Demo(config)#show slb group method
slb group method ic_group “yqv” 1 rr
```

如果希望自己指定 cookie 名称，可以通过“**slb group method** <*group_name*> *ic [cookie_name] [add_path] [rr|sr|lc] [threshold]*”命令进行配置。

```
Demo(config)#slb group method ic_group ic CookieExampleName
```

现在，来自后台服务的响应中会插入 cookie“CookieExampleName”，系统会用该 cookie 值来确定进行会话保持的服务器，如：

CookieExampleName=server1http

- **重写 cookie**

重写 cookie（rc）允许改写 cookie 值的一部分来确保带有 cookie 的请求会被送回同一个服务器。

注意：被改写的部分在下一个请求中不会被改回来，所以后台服务器可以见到改写后的 cookie 值。

1. 为重写 cookie 算法配置后台服务组。

```
Demo(config)#slb group method rc_group rc CookieExampleName
```

2. 将后台服务加入到服务组。

```
Demo(config)#slb group member rc_group service1http
Demo(config)#slb group member rc_group service2http
Demo(config)#slb group member rc_group service3http
Demo(config)#slb group member rc_group service4http
Demo(config)#slb group member rc_group service5http
```

3. 设置负载均衡策略。

```
Demo(config)#slb policy rcookie pol1 virtual1http rc_group 1
Demo(config)#slb policy default virtual1http rc_group
```

现在，cookie“CookieExampleName”的值会被改写为后台服务器的名称。例如：来自 server1http 的响应会被如下改写：

改写前：CookieExampleName=valueabcdefghijk

改写后：CookieExampleName=server1http!!ijk

注意：后台服务响应中的 cookie 值的长度需要大于或者等于后台服务器名称+2+4（offset，这里为 4），否则改写操作不会执行。

➢ **复位向**

复位向（redirect）是将客户对一个主机名的请求复位向到另一个主机名的策略。通过这样一种策略，所有的 HTTP 请求被均衡到不同的后台服务器，而且后续的请求全部访问新的主机名，从而保证了会话的持续性。

在我们的例子中，前往“http://www.abc.com/index.htm”主页的请求会被复位向到“http://www1.abc.com/index.htm”、“http://www2.abc.com/index.htm”或“http://www3.abc.com/index.htm”，这取决于服务组的算法。

1. 定义用于 HTTP 复位向的后台服务。

```
Demo(config)#slb real http www1.abc.com 192.168.10.10
Demo(config)#slb real http www2.abc.com 192.168.10.11
Demo(config)#slb real http www3.abc.com 192.168.10.12
```

2. 建立服务组。

```
Demo(config)#slb group method group1 rr
```

3. 将后台服务添加得到服务组。

```
Demo(config)#slb group member group1 www1.abc.com
Demo(config)#slb group member group1 www2.abc.com
Demo(config)#slb group member group1 www3.abc.com
```

4. 使用“**slb policy redirect**”命令将一个已经存在的虚拟服务同服务组联系起来。

```
Demo(config)#slb policy redirect pol1 virtual1http group1 www.abc.com
```

目前复位向策略支持下列负载均衡方法：轮询（rr）、最少连接数（lc）、最短响应时间（sr）和就近性（prox）。

注意：复位向策略的使用需要相应的域名解析支持。上面的例子中，需要支持的域名解析包括：www.abc.com、www1.abc.com、www2.abc.com 和 www3.abc.com。

11.3.2 SIP 协议的服务器负载均衡配置

下面给出 SIP SLB 的典型配置示例。

11.3.2.1 配置向导

本节中的配置示例是基于一个单臂结构，两个服务器的默认网关是设备的地址 172.16.30.170，服务器所在的子网 VLAN 30 和客户端所在的子网 VLAN 10，都通过路由器相连，路由器的地址是 172.16.30.1，如下图所示。

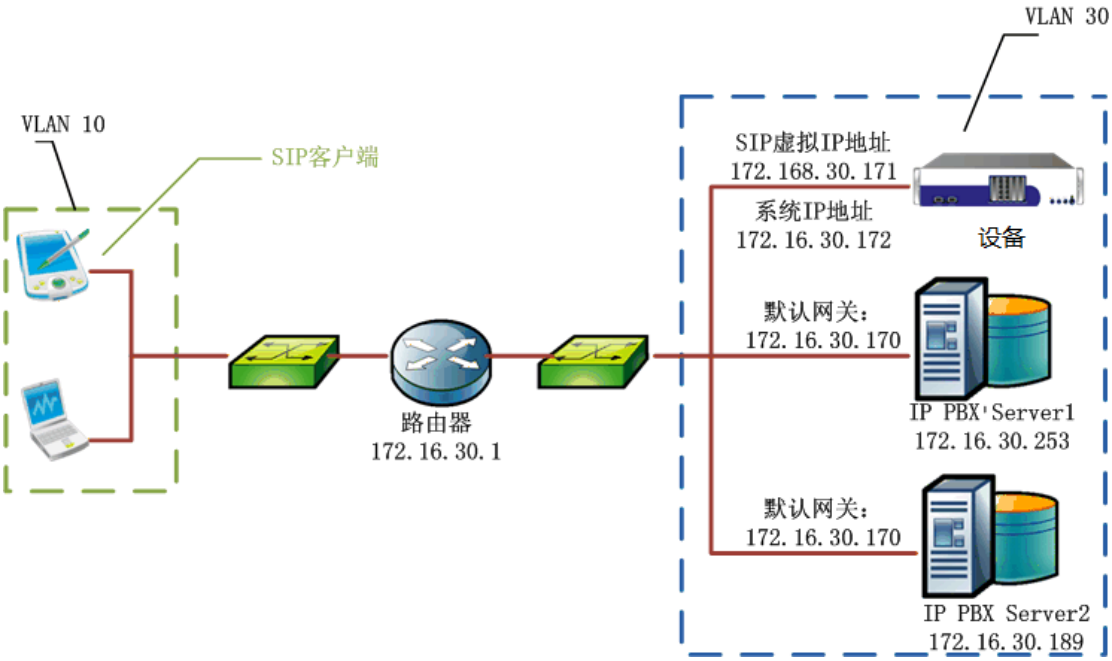

图11–15 SIP 协议的负载均衡配置

下面列出了 SIP 协议负载均衡配置所需用的命令，相关命令的描述信息，请参考命令行使用手册。

表11–5 设备 SIP 协议负载均衡配置命令

配置操作	命令行
配置后台服务	**slb real siptcp** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real sipudp** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down] [timeout]*
定义服务组的负载均衡算法	**slb group method** *<group_name>* **{sipcid|sipuid|sipcidps|sipuidps}** *[first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **{rr|pu|sr}** **slb group method** *<group_name>* **lc** [threshold_granularity] [yes|no] **slb group method** *<group_name>* **pi** *[hash_bits] [first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **hi** *[hash_bits]* **slb group method** *<group_name>* **chi** *[hash_bits]* **slb group method** *<group_name>* **snmp** *[snmp_mode] [community]*

配置操作	命令行
	[oid_count] [oid1] [oid1_weight] [oid2] [oid2_weight] [check_interval]
将后台服务添加到服务组	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]*
配置虚拟服务	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb virtual {tcp|tcps|udp}** *<virtual_service> <vip> <vport> [arp_support] [max_connection]* **slb virtual rtsp** *<virtual_service> <vip> [vport] [mode] [arp_support] [max_connection]* **slb virtual {ftp|ftps}** *<virtual_service> <vip> [vport] [max_connection]*
将服务组绑定到虚拟服务	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}* **slb policy static** *<virtual_service> <real_service>* **slb policy backup** *{virtual_service|vlink_name} {group_name|vlink_name}*

11.3.2.2 配置示例

1. 定义 SIP UDP 后台服务。

```
Demo(config)#slb real sipudp "r1" 172.16.32.253 5060 1000 sip-udp 3 3 60
Demo(config)#slb real sipudp "r2" 172.16.32.189 5060 1000 sip-udp 3 3 60
```

2. 使用“**slb group method**”命令为 SIP 负载均衡建立服务组。

```
Demo(config)#slb group method "g1" sipuid rr
```

3. 将 SIPUDP 后台服务添加到服务组。

```
Demo(config)#slb group member "g1" "r1" 1
Demo(config)#slb group member "g1" "r2" 1
```

4. 建立虚拟服务。

```
Demo(config)#slb virtual sipudp "v1" 172.16.30.171 5060
```

5. 将服务组同 SIP UDP 虚拟服务绑定。

```
Demo(config)#slb policy default "v1" "g1"
```

6. 配置 SIP 协议的 Multi-register。

如果终端服务器没有共享数据库，请打开 multi-register 功能，即 SIP 注册信息广播同步功能。

```
Demo(config)#sip multireg on
```

7. 配置 SIP 地址转换。

为了更好的控制后台服务的网络流量，你需要设置 SIP NAT 规则。

```
Demo(config)#sip nat 172.16.30.171 5060 172.16.32.253 5060 udp 60 callid
Demo(config)#sip nat 172.16.30.171 5060 172.16.32.189 5060 udp 60 callid
```

11.3.3 RTSP 协议的服务器负载均衡配置

11.3.3.1 复位向模式的配置

在我们的示例中，客户端向虚拟服务“vs_rtsp 1”（10.5.1.80）发送一个请求“rtsp://10.5.1.80/test.mp3”。设备通过策略和负载均衡算法来选择一个后台服务。在复位向模式中，设备使用后台服务器的 URL“rtsp://audio2.com:554/test.mp3”来向客户端发送响应。然后设备和客户端断开连接，客户端开始使用“audio2.com:554”的地址同后台服务通讯。在这种模式中，所有的后台服务器都必须拥有公网地址，以便同广域网中的客户端通信。如下图所示。

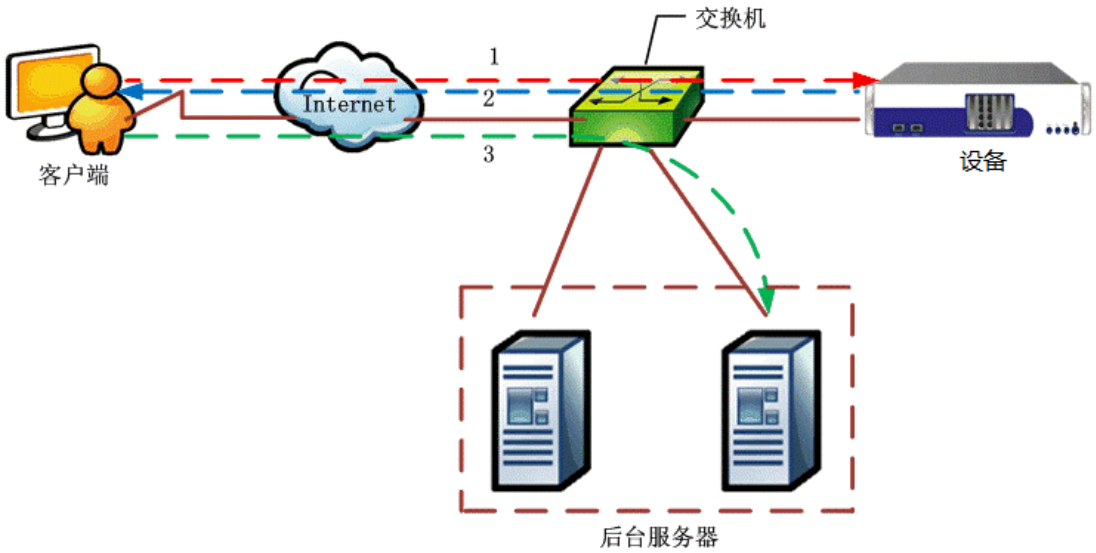

图11–16 RTSP 协议的复位向模式

下面列出了配置 RTSP 协议复位向模式负载均衡的命令，相关命令的描述信息，请参考命令行使用手册。

表11–6 设备 RTSP 复位向模式负载均衡配置命令

配置操作	命令行
配置后台服务	**slb real rtsp** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down] [timeout]*
定义服务组负载均衡算法	**slb group method** *<group_name>* **{rr|pu|sr}** **slb group method** *<group_name>* **lc** [threshold_granularity] [yes|no]

配置操作	命令行
	slb group method <*group_name*> **pi** *[hash_bits] [first_choice_method] [threshold_granularity]* **slb group method** <*group_name*> **hi** *[hash_bits]* **slb group method** <*group_name*> **chi** *[hash_bits]* **slb group method** <*group_name*> **snmp** *[snmp_mode] [community] [oid_count] [oid1] [oid1_weight] [oid2] [oid2_weight] [check_interval]*
将后台服务添加到服务组	**slb group member** <*group_name*> <*real_service*> *[weight|cookie_value|url_tag_value] [priority]*
定义虚拟服务	**slb virtual {http|https|dns|siptcp|sipudp}** <*virtual_service*> <*vip*> *[vport] [arp_support] [max_connection]* **slb virtual {tcp|tcps|udp}** <*virtual_service*> <*vip*> <*vport*> *[arp_support] [max_connection]* **slb virtual rtsp** <*virtual_service*> <*vip*> *[vport] [mode] [arp_support] [max_connection]* **slb virtual {ftp|ftps}** <*virtual_service*> <*vip*> *[vport] [max_connection]*
将服务组绑定到虚拟服务	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}* **slb policy static** <*virtual_service*> <*real_service*> **slb policy backup** *{virtual_service|vlink_name} {group_name|vlink_name}* **slb policy filetype** <*policy_name*> <*vs_name*> <*group_name* > <*filetype*>

1. 使用“**slb real rtsp**”命令定义 RTSP 后台服务。

可以通过“**slb real rtsp**”命令定义一个 RTSP 后台服务。在复位向模式下，后台服务的名称必须为“real IP[:port]”或者“domainname[:port]”

```
Demo(config)#slb real rtsp "10.5.1.90" 10.5.1.90 554 1000 rtsp-tcp 3 3
Demo(config)#slb real rtsp "10.5.1.91:554" 10.5.1.91 554
Demo(config)#slb real rtsp "audio1.com" 10.5.1.92 554
Demo(config)#slb real rtsp "audio2.com:554" 10.5.1.93 554
```

2. 定义 RTSP 服务组。

定义好后台服务以后，可以使用轮询（rr）、保持 IP（pi）、哈希 ip（hi）、一致性哈希 IP（chi）和 snmp 等负载均衡算法选定 RSTP 后台服务。

```
Demo(config)#slb group method "mp3_group" rr
Demo(config)#slb group method "song" rr
```

3. 将后台服务添加到服务组。

```
Demo(config)#slb group member "mp3_group" "10.5.1.90"
Demo(config)#slb group member "mp3_group" "10.5.1.91:554"
```

```
Demo(config)#slb group member "song" "audio1.com"
Demo(config)#slb group member "song" "audio2.com:554"
```

4. 定义一个虚拟服务。

RTSP 虚拟服务的默认模式是复位向模式，默认端口是 554。

```
Demo(config)#slb virtual rtsp "vs_rtsp1" 10.5.1.80
Demo(config)#slb virtual rtsp "vs_rtsp2" 10.5.1.81 554 "redirect"
```

5. 定义一个可以通过文件类型选择服务组的策略。

```
Demo(config)#slb policy filetype "p1" "vs_rtsp1" "mp3_group" "mp3"
Demo(config)#slb policy default "vs_rtsp1" "song"
```

11.3.3.2 动态 NAT 模式的配置

在 NAT 模式下，所有的 RTSP 控制信息将通过设备被均衡的分配到后台多个终端媒体服务器上。从媒体服务器上传出的数据报经过端口映像被发送到外部的客户端上。与复位向模式不同，后台服务器不使用公网 IP 地址。内部私有 IP 地址将被转换成设备上的公网 IP 地址。如下图所示。

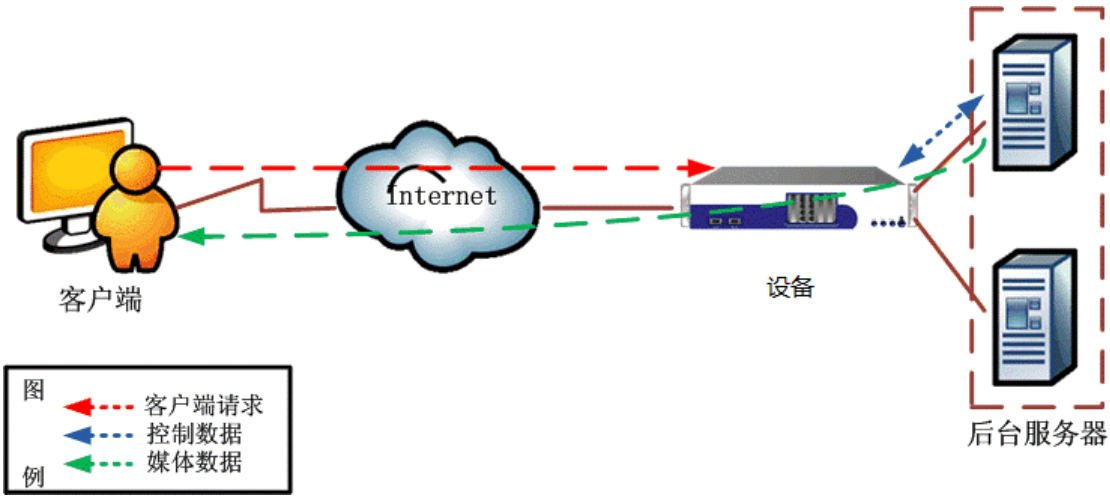

图11–17 RTSP 协议动态路由模式的负载均衡配置

下面列出了配置 RTSP 协议动态路由模式的命令，相关命令的描述信息，请参考命令行使用手册。

表11–7 设备 RTSP 协议动态路由模式配置命令

配置操作	命令行
配置后台服务	**slb real rtsp** *<real_name> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down] [timeout]*
定义服务组负载均	**slb group method** *<group_name> [rr|pu|sr]*

配置操作	命令行
衡算法	**slb group method** <*group_name*> *lc [threshold] [yes|no]* **slb group method** <*group_name*> *pi [hash_bits] [rr|sr|lc|lb] [threshold]* **slb group method** <*group_name*> *hi [hash_bits]* **slb group method** <*group_name*> *chi [hash_bits]* **slb group method** <*group_name*> *snmp [weight|cpu] [community] [oidcount] [oid1] [oidweight1] [oid2] [oidweight2] [check_interval]*
将后台服务添加到服务组	**slb group member** <*group_name*> <*real_name*> *[weight]*
定义虚拟服务	**slb virtual {http|https|dns|siptcp|sipudp}** <*virtual_name*> <*vip*> *[vport] [arp|noarp] [max_connection]* **slb virtual {tcp|tcps|udp}** <*virtual_name*> <*vip*> <*vport*> *[arp|noarp] [max_connection]* **slb virtual rtsp** <*virtual_name*> <*vip*> *[vport] [mode] [arp|noarp] [max_connection]* **slb virtual {ftp|ftps}** <*virtual_name*> <*vip*> *[vport] [max_connection]*
将服务组绑定到虚拟服务	**slb policy default** *{virtual_name|vlink_name} {group_name|vlink_name}* **slb policy static** <*virtual_name*> <*real_name*> **slb policy backup** *{virtual_name|vlink_name} {group_name|vlink_name}* **slb policy filetype** <*policy_name*> <*vs_name*> <*group*> <*filetype*>

1. 使用“**slb real rtsp**”命令定义后台服务。

```
Demo(config)#slb real rtsp "rs_rtsp1" 10.5.14.90 554 1000 rtsp-tcp 3 3 60
Demo(config)#slb real rtsp "rs_rtsp2" 10.5.14.91 554 1000 rtsp-tcp 3 3 60
```

2. 定义 RTSP 服务组。

我们可以使用轮询（rr）、保持 IP（pi）、哈希 IP（hi）、一致性哈希 IP（chi）和 snmp 负载均衡算法来配置我们的服务组。

```
Demo(config)#slb group method "grt1" rr
```

3. 将后台服务添加到服务组。

```
Demo(config)#slb group member "grt1" "rs_rtsp1" 1
Demo(config)#slb group member "grt1" "rs_rtsp2" 1
```

4. 配置 RTSP 虚拟服务。

```
Demo(config)#slb virtual rtsp "vs_rtsp1" 10.3.14.90 554 nat
Demo(config)#slb virtual rtsp "vs_rtsp2" 10.3.14.91 7070 nat
```

5. 配置负载均衡策略。

```
Demo(config)#slb policy default "vs_rtsp1" "grt1"
Demo(config)#slb policy static "vs_rtsp2" "rs_rtsp1"
```

11.3.4 基于 IP/MAC 的二层负载均衡配置

11.3.4.1 双臂模式网络中的配置

确认你当前在配置模式下，你可配置二层负载均衡对二层或三层网络设备做负载均衡，例如防火墙。防火墙产品可有不同的形式：软件产品和硬件产品。无论什么类型的防火墙，都有些性能局限。我们可以使用防火墙的负载均衡去配置多个低端或中端防火墙获取一个同高端、性能强劲的防火墙同样的吞吐量，同时具有更好的性价比。此外，防火墙的负载均衡允许使用增加服务器的数量作为防火墙的容错性，如同多个防火墙的分布式部署来提高防火墙的高可靠性。当前，我们可通过基于 IP/MAC 的二层负载均衡来支持以上类型的负载均衡。

以下的局限点适用于所有的二层、三层负载均衡：一个后台服务仅能被一个后台组包含。二层和三层的后台服务不能同时定义在相同的接口。

让我们开始设置防火墙负载均衡的网络环境。我们将描述不同的使用场景。

为了使二层负载均衡可以工作，客户端，服务器和防火墙应配置一个设备的 IP 地址作为默认网关或静态路由的下一跳地址。例如：以下的路由被分别加入到客户端、服务器、防火墙中。

表11–8 设备二层负载均衡客户端、服务器和防火墙的路由配置

配置项	具体配置
客户端路由	route add –net 172.16.167.0 172.16.162.70 -netmask 255.255.255.0
服务器路由	route add –net 172.16.162.0 172.16.167.73 -netmask 255.255.255.0
防火墙 1 路由	route add -net 172.16.167.0 172.16.165.73 -netmask 255.255.255.0 route add -net 172.16.162.0 172.16.163.70 -netmask 255.255.255.0
防火墙 2 路由	route add -net 172.16.167.0 172.16.166.73 -netmask 255.255.255.0 route add -net 172.16.162.0 172.16.164.70 -netmask 255.255.255.0

注意：此处我们假设所有系统都是类 Unix 系统。对于 Windows 系统，可使用不同版本的路由命令。

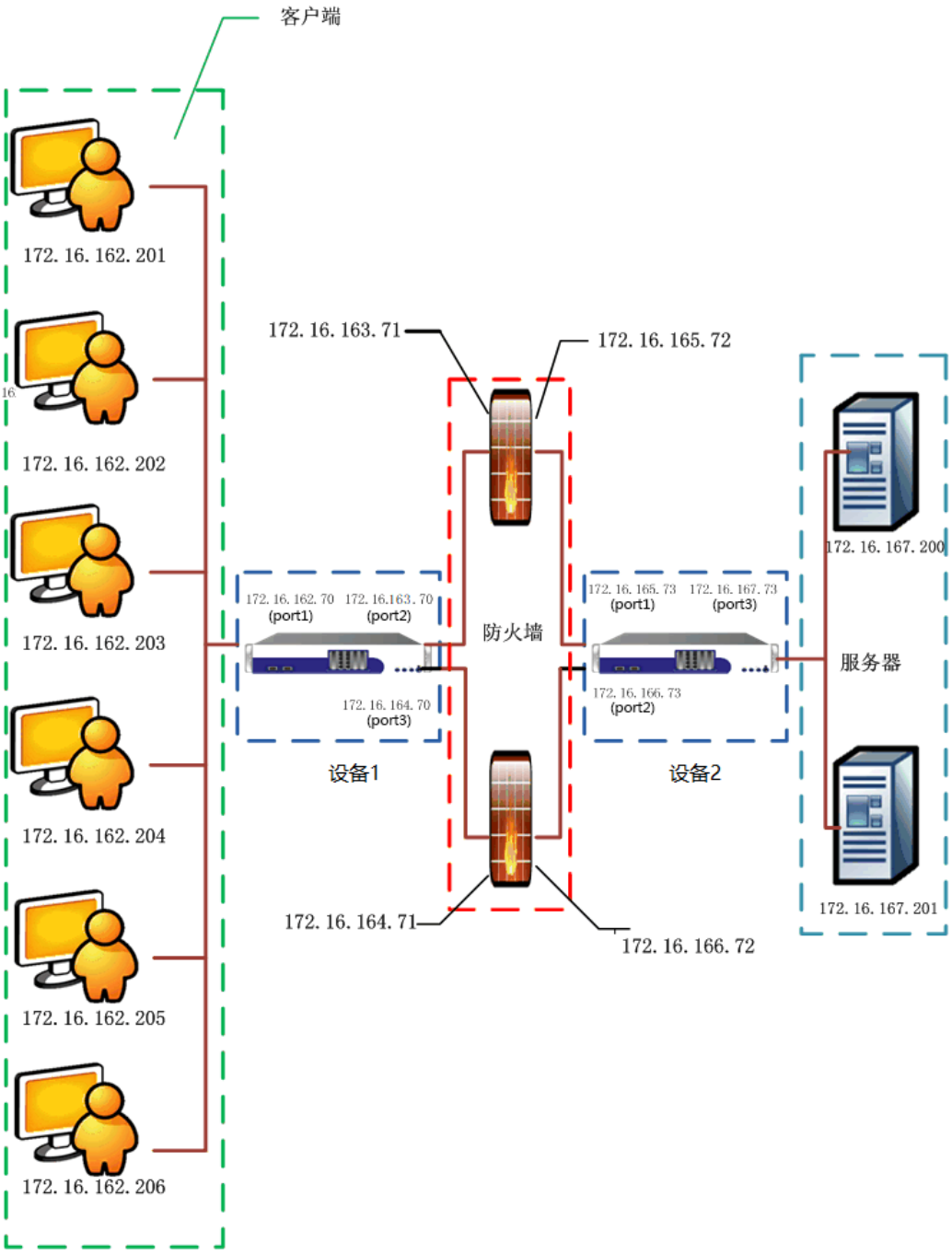

图11–18 二层基于 IP/MAC 地址的负载均衡—双臂结构

注意：一个后台服务只能属于一个服务组，三层虚拟服务和二层虚拟服务不能同时配置在一个接口上。

下面列出了用于配置二层负载均衡的命令，相关命令的描述信息，请参考命令行使用手册。

表11–9 设备二层负载均衡配置命令

配置操作	命令行
配置后台服务	**slb real l2ip** *<real_service> <real_ip>* **slb real l2mac** *<real_service> <real_mac> <output_interface>*
定义服务组负载均衡算法	**slb group method** *<group_name>* **{hi|rr|chi}** *[hash_bits]*
将后台服务添加到服务组	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]*
定义虚拟服务	**slb virtual l2ip** *<virtual_service> <vip> [gateway_ip]*
将服务组绑定到虚拟服务	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}*
定义附加健康检查	**slb real health** *<add_hc_name> <real_service> <ip> <port> [hc_type] [hc_up] [hc_down]*
设置 TCP 类型健康检查所需的反射器	**health ipreflect** *<reflector_name> <ip_address> <port> [protocol]*

下面，我们将按照上图来配置设备 1。

1. 为设备接口配置 IP 地址。

```
Demo(config)#ip address port1 172.16.162.70 255.255.255.0
Demo(config)#ip address port2 172.16.163.70 255.255.255.0
Demo(config)#ip address port3 172.16.164.70 255.255.255.0
```

2. 定义后台服务。

```
Demo(config)#slb real l2ip rs1 172.16.163.71
Demo(config)#slb real l2ip rs2 172.16.164.71
```

我们也可以使用 MAC 地址来定义后台服务。

```
Demo(config)#slb real l2mac rs1 00:e0:81:03:36:e4 port2
Demo(config)#slb real l2mac rs2 00:30:48:81:54:9c port3
```

注意：请使用你的特定系统中相应的 IP 地址命令来获取 MAC 地址。例如：如果使用 Linux，可使用命令“**ifconfig –u**”来获取网卡的 MAC 地址。

3. 定义后台服务组。

```
Demo(config)#slb group method g1 rr direct
或
Demo(config)#slb group method g1 hi direct
```

注意：使用二层负载均衡的组支持三种算法：轮询（rr）、哈希 IP（hi）和一致性哈希 IP（chi）。

当使用“**slb group method**”命令定义一个新的二层负载均衡组时，引入了一个新的参数“route|direct”。这个参数用于指定路由模式，即指定由后台服务发起的流量如何被路由出去。“route”模式是根据普通路由规则对流量进行路由处理，“direct”模式（默认模式）是从二层虚拟服务关联的网络接口处直接将流量路由出去。

4. 将后台服务添加到服务组。

```
Demo(config)#slb group member g1 rs1 1
Demo(config)#slb group member g1 rs2 1
或
Demo(config)#slb group member g1 rs1
Demo(config)#slb group member g1 rs2
```

5. 定义二层虚拟服务。

```
Demo(config)#slb virtual l2ip vs1 172.16.162.70
```

6. 定义服务器负载均衡策略。

```
Demo(config)#slb policy default vs1 g1
```

注意：二层负载均衡只支持默认策略。

7. 添加附加健康检查。

```
Demo(config)#slb real health a1 rs1 172.16.165.73 80 tcp
Demo(config)#slb real health a1 rs2 172.16.166.73 80 tcp
```

现在我们可以根据图 10-18 来配置设备 2。

1. 为设备接口配置 IP 地址。

```
Demo(config)#ip address port1 172.16.165.73 255.255.255.0
Demo(config)#ip address port2 172.16.166.73 255.255.255.0
Demo(config)#ip address port3 172.16.167.73 255.255.255.0
```

2. 定义后台服务。

```
Demo(config)#slb real l2ip rs1 172.16.165.72
Demo(config)#slb real l2ip rs2 172.16.166.72
或
Demo(config)#slb real l2mac rs1 00:e0:81:03:36:e5 port1
Demo(config)#slb real l2mac rs2 00:30:48:81:54:9d port2
```

3. 定义后台服务组。

```
Demo(config)#slb group method g1 rr direct
```

```
或
Demo(config)#slb group method g1 hi direct
```

4. 将后台服务添加到服务组。

```
Demo(config)#slb group member g1 rs1 1
Demo(config)#slb group member g1 rs2 1
或
Demo(config)#slb group member g1 rs1
Demo(config)#slb group member g1 rs2
```

5. 定义二层虚拟服务。

```
Demo(config)#slb virtual l2ip vs1 172.16.167.73
```

6. 定义服务器负载均衡策略。

```
Demo(config)#slb policy default vs1 g1
```

7. 添加附加健康检查。

```
Demo(config)#slb real health a1 rs1 172.16.163.70 80 tcp
Demo(config)#slb real health a1 rs2 172.16.164.70 80 tcp
```

8. 添加二层 SLB TCP 类型健康检查所需的反射器。

```
Demo(config)#health ipreflect aa 0.0.0.0 80 tcp
```

11.3.4.2 单臂模式网络中的配置

等同于双臂网络拓扑的应用场景，为实现网络报文转发，需要为客户端、服务器和防火墙配置新的路由。

表11–10 设备单臂模式客户端、服务器和防火墙的路由配置

配置项	具体配置
客户端路由	route add -net 172.16.167.0 172.16.162.70 -netmask 255.255.255.0
服务器路由	route ADD 172.16.162.0 MASK 255.255.255.0 172.16.167.73 route ADD 172.16.163.0 MASK 255.255.255.0 172.16.167.73 route ADD 172.16.164.0 MASK 255.255.255.0 172.16.167.73
防火墙 1 路由	route add default 172.16.163.70
防火墙 2 路由	route add default 172.16.164.70

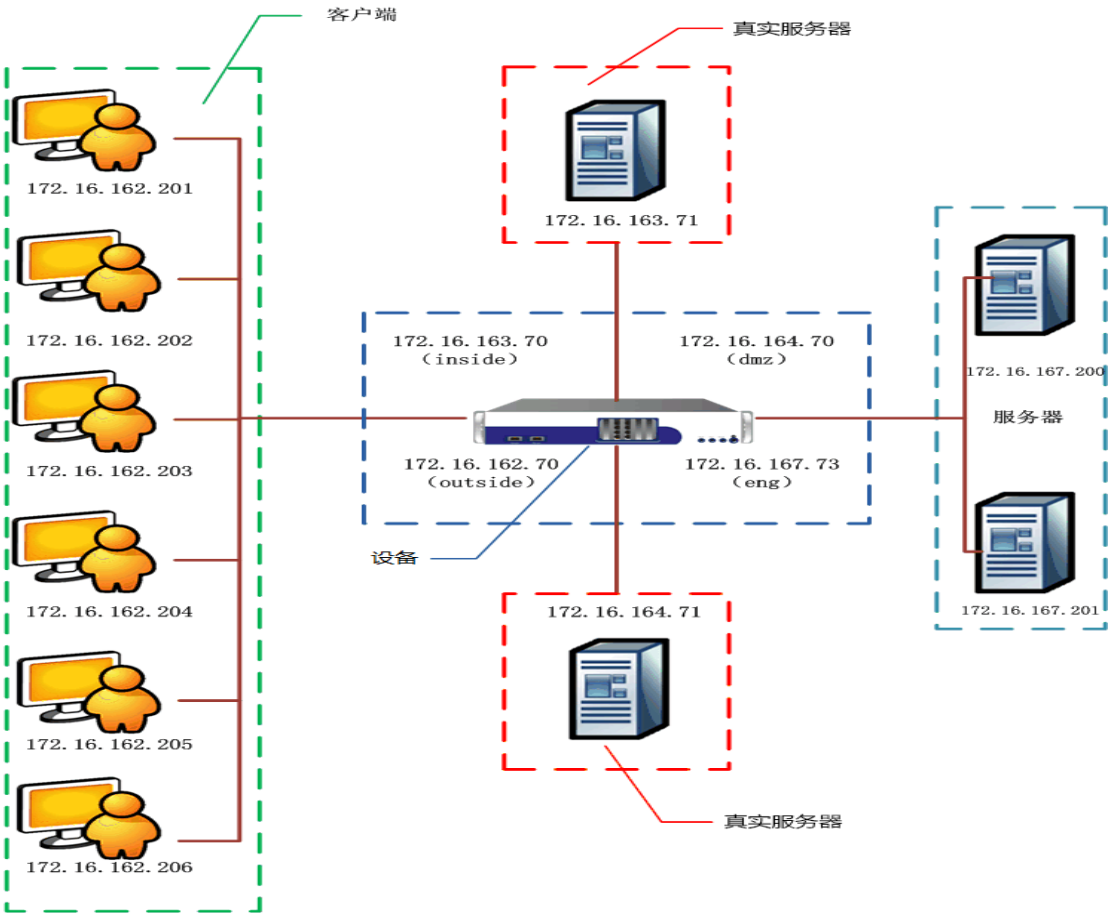

图11–19 二层基于 IP/MAC 地址的负载均衡—单臂结构

接下来我们按照上图来配置设备。

1. 为设备接口配置 IP 地址。

```
Demo(config)#ip address "port1" 172.16.162.70 255.255.255.0
Demo(config)#ip address "port2" 172.16.163.70 255.255.255.0
Demo(config)#ip address "port3" 172.16.164.70 255.255.255.0
Demo(config)#ip address "port4" 172.16.167.73 255.255.255.0
```

2. 定义后台服务。

```
Demo(config)#slb real l2ip "r1" 172.16.163.71
Demo(config)#slb real l2ip "r2" 172.16.164.71
或
Demo(config)#slb real l2mac r1 00:e0:81:03:36:e4 port2
Demo(config)#slb real l2mac r2 00:30:48:81:54:9c port3
```

3. 定义后台服务组。

```
Demo(config)#slb group method "g1" "rr" "route"
或
Demo(config)#slb group method "g1" "hi" "route"
```

4. 将后台服务添加到服务组。

```
Demo(config)#slb group member "g1" "r1" 1
Demo(config)#slb group member "g1" "r2" 1
或
Demo(config)#slb group member "g1" "r1"
Demo(config)#slb group member "g1" "r2"
```

5. 定义二层虚拟服务。

```
Demo(config)#slb virtual l2ip "v1" 172.16.162.70
Demo(config)#slb virtual l2ip "v2" 172.16.167.73
```

6. 定义服务器负载均衡策略。

```
Demo(config)#slb policy default "v1" "g1"
Demo(config)#slb policy default "v2" "g1"
```

7. 添加附加健康检查。

```
Demo(config)#slb real health a1 r1 172.16.163.71 0 icmp
Demo(config)#slb real health a1 r2 172.16.164.71 0 icmp
或
Demo(config)#slb real health a1 r1 172.16.163.70 0 icmp
Demo(config)#slb real health a1 r2 172.16.164.70 0 icmp
```

11.3.4.3 二层负载均衡本机例外 IP 配置

通常，在应用二层负载均衡的场景中，系统将转发目的 IP 地址不是本地 IP 地址的数据包。

目的 IP 和本机 IP（如 VIP、接口 IP 等）相同的数据包将略过二层负载均衡，直接进行四层或七层负载均衡处理。

设备支持本地例外 IP 配置，允许将需要进行流量编排的本机 IP（如 VIP、接口 IP 等）配置为二层负载均衡的例外 IP，从而实现二层负载均衡对其流量的处理和转发。配置本机例外 IP 后，当目的 IP 和本机 IP 相同的数据包命中 L2IP 虚拟服务时，系统将依旧对数据包进行二层负载均衡转发（如转发给安全设备），然后数据包将被返回至本设备继续做四层或七层负载均衡处理。

- **CLI 配置示例：**

为指定 L2IP 虚拟服务添加本机例外 IP。

```
Demo(config)#slb l2vs ipexception 172.16.163.71 vs1
```

11.3.5 基于 IP 的三层负载均衡配置

11.3.5.1 配置向导

下面列出了配置三层负载均衡配置所需的命令，相关命令的描述信息，请参考命令行使用手册。

表11–11 设备三层负载均衡配置命令

配置操作	命令行
配置后台服务	**slb real ip** *<real_service> <ip> [max_connection] [hc_type] [hc_up] [hc_down] [udp_timeout]*
定义服务组负载均衡算法	**lb group method** *<group_name>* **{rr|sr}** **slb group method** *<group_name>* **lc** *[threshold_granularity] [yes|no]* **slb group method** *<group_name>* **pi** *[hash_bits] [first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **hi** *[hash_bits]* **slb group method** *<group_name>* **chi** *[hash_bits]* **slb group method** *<group_name>* **snmp** *[snmp_mode] [community] [oid_count] [oid1] [oid1_weight] [oid2] [oid2_weight] [check_interval]*
将后台服务添加到服务组	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]*
定义虚拟服务	**slb virtual ip** *<virtual_service> <vip> [max_connection]*
将服务组绑定到虚拟服务	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}* **slb policy static** *<virtual_service> <real_service>* **slb policy backup** *{virtual_service|vlink_name} {group_name|vlink_name}*

11.3.5.2 配置示例

1. 建立后台服务。

```
Demo(config)#slb real ip rip0 10.3.14.10
Demo(config)#slb real ip rip1 10.3.14.20 1000 none
```

后台服务“rip0”的健康检查类型默认为“ICMP”。

2. 使用“slb group method”命令定义服务组。

```
Demo(config)#slb group method gip0
```

3. 将后台服务添加到服务组。

```
Demo(config)#slb group member "gip0" "rip0" 1
```

```
Demo(config)#slb group member "gip0" "rip1" 1
```

4. 建立虚拟服务。

```
Demo(config)#slb virtual ip vip0 10.3.14.56
```

5. 将服务组同虚拟服务绑定。

```
Demo(config)#slb policy default vip0 gip0
或
Demo(config)#slb policy static vip0 rip0
```

11.3.6 基于端口段的负载均衡配置

11.3.6.1 配置向导

下面列出了配置端口段负载均衡配置所需的命令，相关命令的描述信息，请参考命令行使用手册。

表11–12 设备端口负载均衡配置命令

配置操作	命令行
配置后台服务	**slb real tcp** *<real_service> <ip> <port> [max_connection] [hc_type] [hc_up] [hc_down]* **slb real http** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real udp** *<real_service> <ip> <port> [max_connection] [hc_up] [hc_down] [timeout] [hc_type]* **slb real https** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real tcps** *<real_service> <ip> <port> [max_connection] [hc_type] [hc_up] [hc_down]* **slb real dns** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down] [timeout]*
定义服务组负载均衡算法	**slb group method** *<group_name> [method]*
将后台服务添加到服务组	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]*
定义虚拟服务	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb virtual {tcp|tcps|udp}** *<virtual_service> <vip> <vport> [arp_support] [max_connection]* **slb virtual rtsp** *<virtual_service> <vip> [vport] [mode] [arp_support] [max_connection]*

	slb virtual {ftp|ftps} *<virtual_service> <vip> [vport] [max_connection]*
为虚拟服务定制端口段	**slb virtual portrange** *<virtual_service> <start_port> <end_port> [protocol] [port_type]*
将服务组绑定到虚拟服务	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}* **slb policy static** *<virtual_service> <real_service>* **slb policy backup** *{virtual_service|vlink_name} {group_name|vlink_name}*

11.3.6.2 配置示例

1. 定义静态或动态端口段后台服务。

当后台服务为静态时：

```
Demo(config)#slb real http rhttp0 10.3.14.10
```

将 rhttp0 的端口保持为默认的 80 端口。

```
Demo(config)#slb real http rhttp1 10.3.14.11 90
```

rhttp1 的端口已调整为 90。

如果后台服务器采用了端口段，那么健康检查的类型只能为“icmp”或者“none”。

```
Demo(config)#slb real http rhttp0 10.3.14.10 0 1000 icmp
Demo(config)#slb real http rhttp1 10.3.14.11 0 1000 none
```

2. 定义一个服务组。

```
Demo(config)#slb group method ghttp1
```

3. 将后台服务添加到服务组。

```
Demo(config)#slb group member ghttp1 rhttp0
Demo(config)#slb group member ghttp1 rhttp1
```

4. 建立虚拟服务。

```
Demo(config)#slb virtual http vhttp0 10.3.14.50 0
```

5. 为虚拟服务定义端口段。

一个负载均衡虚拟服务可以同时被配置三个端口段。

```
Demo(config)#slb virtual portrange vhttp0 80 90
Demo(config)#slb virtual portrange vhttp0 8000 9000
```

在上面的例子中，所有前往 10.3.14.50、端口号在 80~90 之间或 8000~9000 之间的数据报都将被虚拟服务“vhttp0”处理。

注意：定义端口段类型的后台服务，需要将端口指定为 0。

6. 将服务组与虚拟服务关联起来。

```
Demo(config)#slb policy default vhttp0 ghttp1
或
Demo(config)#slb policy static vhttp0 rhttp1
```

注意：端口段类型的后台服务与单端口类型的后台服务不能添加到一个组。

11.3.7 终端服务器负载均衡配置

11.3.7.1 配置向导

下面列出了配置终端服务器负载均衡配置所需的命令，相关命令的描述信息，请参考命令行使用手册。

表11–13 设备终端服务器负载均衡配置命令

配置操作	命令行
创建 RDP 后台服务	**slb real rdp** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]*
定义 RDP 后台服务组算法	**slb group method** *<group_name>* **rdprt** *[first_choice_method]*
向 RDP 组中添加后台服务	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]*
创建 RDP 虚拟服务	**slb virtual rdp** *<virtual_service> <vip> [vport] [arp_support] [max_connection]*
建立虚拟服务与后台服务组的关联关系	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}*

11.3.7.2 配置示例

1. 创建 RDP 后台服务。

RDP 后台服务预设的端口号为 3389。

```
Demo(config)#slb real rdp rs1 172.16.69.191 3389 1000 icmp 3 3
Demo(config)#slb real rdp rs2 172.16.69.192 3389 1000 icmp 3 3
```

注意：对 RDP 后台服务，只能使用 ICMP 和 TCP 类型的健康检查。

2. 创建 RDP 后台服务组。

```
Demo(config)#slb group method g1 rdprt rr
```

3. 向 RDP 后台服务组中添加后台服务。

```
Demo(config)#slb group member g1 rs1
Demo(config)#slb group member g1 rs2
```

4. 创建 RDP 虚拟服务。

RDP 虚拟服务预设的端口号为 3389。

```
Demo(config)#slb virtual rdp vs1 172.16.69.171 3389 arp 0
```

5. 关联虚拟服务与后台服务组。

```
Demo(config)#slb policy default vs1 g1
```

注意：RDP 仅支持默认的组策略（default policy）。

11.3.8 策略嵌套

11.3.8.1 配置向导

下面列出了配置策略嵌套配置所需的命令，相关命令的描述信息，请参考命令行使用手册。

表11–14 策略嵌套基本配置命令

配置操作	命令行
配置后台服务	**slb real http** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real https** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]*
定义后台服务组算法	**slb group method** *<group_name> [method]*
将后台服务添加到组	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]*
定义虚拟服务	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb virtual {tcp|tcps|udp}** *<virtual_service> <vip> <vport>*

配置操作	命令行
	[arp_support] [max_connection] **slb virtual rtsp** *<virtual_service> <vip> [vport] [mode] [arp_support] [max_connection]* **slb virtual {ftp|ftps}** *<virtual_service> <vip> [vport] [max_connection]*
建立虚连接	**slb vlink** *<vlink_name>*
将组绑定到虚拟服务或虚连接	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}* **slb policy static** *<virtual_service> <real_service>* **slb policy icookie** *<policy_name> {virtual_service|vlink_name} <group_name> <precedence>* **slb policy rcookie** *<policy_name> {virtual_service|vlink_name} <group_name> <precedence>* **slb policy persistent cookie** *<policy_name> {virtual_service|vlink_name} <group_name> <cookie_name> <precedence>* **slb policy persistent url** *<policy_name> {virtual_service|vlink_name} <group_name> <url_tag> <precedence>*

11.3.8.2 配置示例

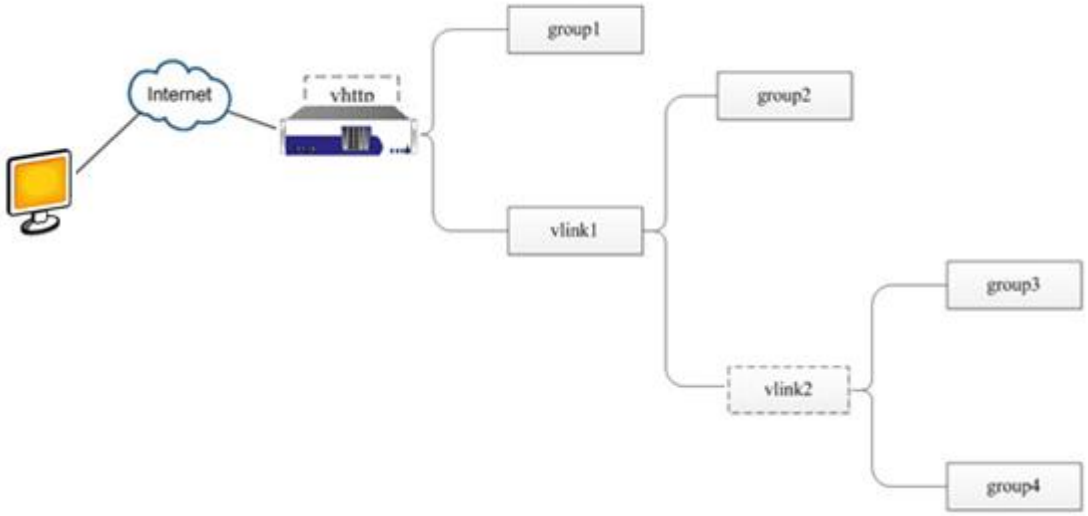

图11–20 策略嵌套

1. 建立后台服务。

```
Demo(config)#slb real http "rhttp1" 172.16.85.109 80 1000 http 3 3
Demo(config)#slb real http "rhttp2" 172.16.85.110 8081 1000 http 3 3
Demo(config)#slb real http "rhttp3" 172.16.85.111 8080 1000 http 3 3
Demo(config)#slb real http "rhttp4" 172.16.85.112 112 1000 tcp 3 3
```

```
Demo(config)#slb real http "rhttp5" 172.16.85.113 113 1000 tcp 3 3
Demo(config)#slb real http "rhttp6" 172.16.85.114 114 1000 tcp 3 3
```

2. 建立后台服务组。

```
Demo(config)#slb group method group1 rr
Demo(config)#slb group method group2 rr
Demo(config)#slb group method group3 rr
Demo(config)#slb group method group4 rr
```

3. 将后台服务添加到服务组。

```
Demo(config)#slb group member group1 rhttp1 1
Demo(config)#slb group member group1 rhttp2 2
Demo(config)#slb group member group2 rhttp3 1
Demo(config)#slb group member group2 rhttp4 2
Demo(config)#slb group member group3 rhttp5 1
Demo(config)#slb group member group4 rhttp6 1
```

4. 建立虚拟服务。

```
Demo(config)#slb virtual http vhttp 172.16.63.109 80 arp 0
```

5. 建立虚连接。

```
Demo(config)#slb vlink vlink1
Demo(config)#slb vlink vlink2
```

6. 将服务组绑定到虚拟服务或虚连接。

```
Demo(config)#slb policy qos network policy1 vhttp vlink1 192.168.2.3 255.255.255.255    1
Demo(config)#slb policy default vhttp group1
Demo(config)#slb policy qos url policy2 vlink1 vlink2 “news” 1
Demo(config)#slb policy default vlink1 group2
Demo(config)#slb policy qos url policy3 vlink2 group3 “sport” 1
Demo(config)#slb policy default vlink2 group4
```

在这个示例中，如果一个请求的源 IP 地址在子网中，它将由于匹配策略“policy1”到达虚连接“vlink1”否则该请求将被发往后台服务“group1”。

如果通过 vlink1 的请求的 URL 中包含了“news”，那这个请求将匹配策略“policy2”，被发往“vlink2”，否则该请求将被发往后台服务“group2”。

如果通过 vlink2 的请求的 URL 中含有“sports”，那么它将匹配策略“policy3”，并到达后台服务“group3”。除此之外，其他所有的请求将被转发至 group4。

11.3.9 服务器负载均衡会话保持配置

11.3.9.1 独立应用 Persistence 算法实现会话保持（ip）

➢ **配置目的：**

配置 Persistence 算法独立应用于 SLB 服务，以客户端 IP 地址作为会话 ID 实现会话保持。

在完成本例中的配置后，设备：

- 将来自客户端的第一个访问请求发送到由首次选择算法（rr）所选择的后台服务器。
- 将该客户端的 IP 地址作为会话 ID 记录到系统会话表中，作为该客户端会话的唯一标识。
- 将该客户端的后续请求发送到同一个后台服务处理。如果连续 5 分钟没有收到来自该客户端的新的请求，设备将清除该客户端的会话 ID 信息。

具体配置过程如下：

➢ **前置条件：**

- 已经定义了四层或七层的后台服务 r1 和 r2。本配置中以 HTTP 类型为例。
- 已经定义了四层或七层的虚拟服务 v1。本配置中以 HTTP 类型为例。

➢ **CLI 配置示例：**

1. 执行如下命令配置后台服务组和 Persistence 会话保持方法：

slb group method <*group_name*> **persistence** <*session_id_type*> *[rr|sr|lc] [threshold]*

例如：

Demo(config)#**slb group method g1 persistence ip rr**

2. 执行如下命令将后台服务添加到服务组：

slb group member <*group_name*> <*real_name*> *[weight]*

例如：

Demo(config)#**slb group member g1 r1 1 0** Demo(config)#**slb group member g1 r2 1 0**

3. 执行如下命令通过默认策略绑定服务组和虚拟服务：

slb policy default *{virtual_name|vlink_name} {group_name|vlink_name}*

例如：

```
Demo(config)#slb policy default v1 g1
```

4. 执行如下命令配置会话保持超时时间：

```
slb persistence timeout <timeout _ minutes> [group_name] [idle|timeout]
```

例如：

```
Demo(config)#slb persistence timeout 5 g1 idle
```

11.3.9.2 独立应用 Persistence 算法实现会话保持（string）

➢ **配置目的：**

配置 Persistence 算法独立应用于 SLB 服务，从 HTTP response cookie 中获取会话 ID 实现会话保持。

在完成本例中的配置后，设备：

- 将来自客户端的第一个访问请求发送到由首次选择算法（rr）所选择的后台服务器。
- 从该后台服务器的响应中获取“mycookie”对应的值作为会话 ID，并记录到系统会话表中。
- 当收到来自同一客户端的后续请求时，查询请求中 URL Query 的“telnum”的值是否命中会话 ID。若命中，则将该请求发送到同一个后台服务器处理。如果连续 5 分钟没有收到来自该客户端的新的请求，设备将清除该客户端的会话 ID 信息。

具体配置过程如下：

➢ **前置条件：**

- 已经定义了四层或七层的后台服务 r1 和 r2。本配置中以 HTTP 类型为例。
- 已经定义了四层或七层的虚拟服务 v1。本配置中以 HTTP 类型为例。

➢ **CLI 配置示例：**

1. 执行如下命令配置后台服务组和 Persistence 会话保持方法：

```
slb group method <group_name> persistence <session_id_type> [rr|sr|lc] [threshold]
```

例如：

```
Demo(config)#slb group method g1 persistence string rr
```

2. 执行如下命令将后台服务添加到服务组：

slb group member <*group_name*> <*real_name*> *[weight]*

例如：

Demo(config)#**slb group member g1 r1 1 0** Demo(config)#**slb group member g1 r2 1 0**

3. 执行如下命令配置设备从 HTTP 响应 cookie 中获取会话 ID：

slb group persistence request urlquery <*group_name*> <*query_name*> **slb group persistence response cookie** <*group_name*> <*cookie_name*>

例如：

Demo(config)#**slb group persistence request urlquery g1 telnum** Demo(config)#**slb group persistence response cookie g1 mycookie**

4. 执行如下命令通过默认策略绑定服务组和虚拟服务：

slb policy default *{virtual_name|vlink_name} {group_name|vlink_name}*

例如：

Demo(config)#**slb policy default v1 g1**

5. 执行如下命令配置会话保持超时时间：

slb persistence timeout <*timeout _ minutes*> *[group_name] [idle|timeout]*

例如：

Demo(config)#**slb persistence timeout 5 g1 idle**

注意：配置基于 HTTP cookie 的会话保持之前，需要在后台服务中预先设置 cookie，以保证 HTTP 响应中能够携带相应的 cookie。

11.3.9.3 Persistence 算法与 SLB 保持策略配合实现会话保持

➢ **配置目的：**

配置 Persistence 算法与七层负载均衡保持策略配合使用，算法通过保持策略获取会话 ID，进而实现会话保持。

设备上对算法和策略的配置如下表所示：

表11–15 算法和策略的配置内容

配置项	配置内容
策略	配置 Header 策略，从客户端 HTTP 请求头部的 x-up-calling-line-id 的内容中获取字符串。

配置项	配置内容
	配置默认策略。
Persistence 算法	首次选择算法为轮询（rr）。 使用 string 类型的会话 ID。 不指定是否从客户端请求或服务器响应中获取会话 ID。 指定会话 ID 的偏移和长度值。

在完成本例中的配置后：

- 如果客户端请求命中 Header 策略，设备将根据 x-up-calling-line-id 的内容和配置的会话 ID 的偏移和长度值确定并获取会话 ID。
- 如果客户端请求命中默认策略，设备将使用首次选择算法 rr 分配请求到某后台服务器，不执行会话保持。

➢ **前置条件：**

- 已经定义了四层或七层的后台服务 r1 和 r2。本配置中以 HTTP 类型为例。
- 已经定义了四层或七层的虚拟服务 v1。本配置中以 HTTP 类型为例。

➢ **CLI 配置示例：**

1. 执行如下命令配置后台服务组和 Persistence 会话保持方法：

slb group method <*group_name*> *persistence* <*session_id_type*> *[rr|sr|lc] [threshold]*

例如：

Demo(config)#**slb group method g1 persistence string rr**

2. 执行如下命令将后台服务添加到服务组：

slb group member <*group_name*> <*real_name*> *[weight]*

例如：

Demo(config)#**slb group member g1 r1 1 0**
Demo(config)#**slb group member g1 r2 1 0**

3. 执行如下命令通过 Header 策略和默认策略绑定服务组和虚拟服务：

slb policy header <*policy_name*> *{virtual_name|vlink_name} {group_name|vlink_name}* <*header_name*> <*header_string*> <*precedence*>
slb policy default *{virtual_name|vlink_name} {group_name|vlink_name}*

例如：

Demo(config)#**slb policy header p1 v1 g1 "x-up-calling-line-id" "^1" 1**
Demo(config)#**slb policy default v1 g1**

4. 执行如下命令配置通过偏移量和会话 ID 的长度确定会话 ID：

slb group persistence value *<group_name> <offset> [session_id_length]*

例如：

```
Demo(config)#slb group persistence request header g1 abc user y 0
Demo(config)#slb group persistence value g1 4 3
```

5. 执行如下命令配置会话保持超时时间：

slb persistence timeout *<timeout _ minutes> [group_name] [idle|timeout]*

例如：

```
Demo(config)#slb persistence timeout 5 g1 idle
```

11.4 服务器负载均衡一览表

表11–16 服务器负载均衡一览表

SLB 类型	优先级	虚拟服务	后台服务	健康检查类型	应用场合
七层 HTTP/HTTPS	2	IP + Port + proto (HTTP, HTTPS)	IP + Port + proto (HTTP, HTTPS)	None HTTP HTTPS TCP TCPS ICMP Additional Script	1、利用应用协议的控制信息（headers）做流量均衡，如：http headers； 2、需要设备的高速缓冲（Cache）功能时。
七层 DNS	2	IP + Port + proto (DNS)	IP + Port + proto (DNS)	None DNS ICMP Additional Script	1、DNS 请求； 2、DNS 缓存。
七层 DNS-TCP	2	IP + Port + proto (DNS-TCP)	IP + Port + proto (DNS-TCP)	None TCP DNS-TCP ICMP Additional Script-TCP	1、DNS 请求； 2、DNS 缓存。
七层 FTP	2	IP + Port + proto (FTP)	IP + Port + proto (FTP)	None TCP ICMP Additional	FTP 通讯。

SLB 类型	优先级	虚拟服务	后台服务	健康检查类型	应用场合
				Script	
七层 SIP	2	IP + Port + proto (SIP-TCP, SIP-UDP)	IP + Port + proto (SIP-TCP, SIP-UDP)	None TCP TCPS ICMP Additional Script SIP-TCP SIP-UDP	VOIP 流量的均衡。
七层 RTSP	2	IP + Port + proto (RTSP)	IP + Port + proto (RTSP)	None TCP ICMP Additional Script RTSP-TCP	实时的多媒体流量均衡。
四层	2	IP + port	IP + Port	None TCP TCPS DNS-TCP ICMP Additional Script	1、利用 TCP/UDP 包头信息做流量均衡； 2、针对特定应用服务，TCP 端口和 UDP 端口是特定的。
七层端口段	3	Layer 7 VS + Port range	Layer 7 RS Layer 7 RS (0 port)	Non-zero port RS: Layer 7 health check Zero port RS: ICMP Additional	除针对单端口的应用服务做流量均衡外，设备在网络层对多端口，动态端口映像的应用做流量均衡。
四层端口段	3	Layer 4 VS + Port range	Layer 4 RS Layer 4 RS (0 port)	Non-zero port RS: Layer 4 health check Zero port RS: ICMP Additional	设备在传输层对多端口，动态端口映像的应用做流量均衡。
三层	4	IP	IP	None ICMP Additional	除可以对多端口段的应用做流量均衡外，设备能对单个服务包含多种应用协议做流量均衡，现在仅支持单个服务包含 TCP/UDP 协议的流量均衡。
二层	1	IP + port ranges	IP, MAC	ARP Additional (only	1、后台服务器没有配置可用的网际协议地

SLB 类型	优先级	虚拟服务	后台服务	健康检查类型	应用场合
				ICMP)	址(IP Address)，因此不能通过 IP 地址对这类服务器做流量均衡； 2、后台服务器不是进入流量的最终地（例如：病毒扫描服务器首先对进入的每一个网络包进行病毒扫描然后再将它们转发到真是的目的地）。

		基本算法						基于IP地址			基于Header/Request								基于Cookie					通用
		rr	lc	snmp	sr	prox	lb	pi	hi	chi	hh	ph	chh	pu	hq	sslsid	sipcid	sipuid	rc	ec	ic	pc	hc	persistence
基本策略	静态	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
	默认	★	★	★	★	★	★	★	★	★	★	★	★	×	★	★	★	★	★	★	★	×	★	★
	备份	★	★	★	★	★	★	★	★	★	★	★	★	★	★	×	×	×	★	★	★	★	★	★
持续性策略	保持URL	×	×	×	×	×	×	×	×	×	×	×	×	★	★	×	×	×	×	×	×	×	×	★
	保持Cookie	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	★	★	★
	重写Cookie	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	★	×	×	×	×	★
	插入Cookie	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	★	×	×	★
QoS策略	QoS Cookie	★	★	★	★	★	★	★	★	★	★	★	★	×	×	×	×	×	★	★	★	★	★	★
	QoS 主机名	★	★	★	★	★	★	★	★	★	★	★	★	×	×	×	×	×	★	★	★	×	×	★
	QoS URL	★	★	★	★	★	★	★	★	★	★	★	★	×	×	×	×	×	★	★	★	×	×	★
	QoS 网络	★	★	★	★	★	★	★	★	★	★	★	★	×	×	×	×	×	★	★	★	×	×	★
	QoS客户端口	★	★	★	★	★	★	★	★	★	★	★	★	×	×	×	×	×	★	★	★	×	×	★
	Regular Expression	★	★	★	★	★	★	★	★	★	★	★	★	×	×	×	×	×	★	★	★	×	×	★
	表头	★	★	★	★	★	★	★	★	★	★	★	★	×	×	×	×	×	★	★	★	×	×	★
	QoS Body	★	★	★	★	★	★	★	★	★	★	★	★	×	×	×	×	×	★	★	★	×	×	★
其他类型策略	重定向	★	★	★	★	★	★	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
	文件类型	★	★	★	★	×	★	★	★	★	×	×	×	×	×	×	×	×	×	×	×	×	×	×
	哈希URL	★	★	★	★	★	★	★	★	★	★	★	★	×	×	×	×	×	★	★	★	×	×	★
		rr-轮询 lc-最少连接数 snmp-SNMP sr-最短响应时间 prox-就近性 lb-最小带宽						pi-保持IP hi-哈希IP chi-统一哈希IP			hh-哈希表头 ph-保持主机名 chh统一哈希表头 pu-保持URL hq-哈希query								sslsid-保持SSL SID sipcid-SIP CallID sipuip-SIP UserID					rc-重写Cookie ec-嵌套Cookie ic-插入Cookie pc-保持Cookie hc-哈希Cookie
		★策略和算法可以配合使用 ×策略和算法不可配合使用																						

注意：启用快速转发功能后，sr 算法不生效，其它的四层负载均衡算法都支持。

第12章 应用安全流量编排

12.1 概述

在传统的“糖葫芦串”网络安全架构中，所有的网络安全设备以物理串联或者逻辑串联的形式部署在互联网接入区，且按照主备部署的模式提供服务。传统的网络安全架构存在如下问题：

- 安全架构与网络架构紧耦合：新的安全需求满足需要依赖现有网络架构，改造成本高，很难满足业务变更需求，无法为不同业务提供差异化、个性化的安全策略。
- SSL 加解密成为性能瓶颈：随着加密流量的比重日益增长，安全设备需要依赖自身加解密功能才能进行七层检测和防御，不仅增加了安全设备的负担，另一方面多次的加解密导致业务访问效率低下，延时增加。
- 安全资源利用率低，扩展性差：所有流量均流经所有的安全设备，导致严重的资源浪费，另外安全资源无法实现池化，只能通过更换设备进行性能扩容。
- 安全投入成本高，维护难度大：用户通常需要购买大量的高端的设备才能满足安全和性能需求，而且后期扩容资金和维护成本高；随着串联的安全设备增加，网络故障处理成本增加。

应用安全流量编排功能旨在解决传统串行安全架构的问题和弊端，提供智能化流量编排方案，基于用户安全策略需求实现个性化、定制化流量编排，为安全架构提供资源池化、负载均衡、SSL 加解密、流量可视化、健康检查、容错等一系列功能。该功能提供如下价值：

- 提升了安全设备的利用效率，提高应用访问效率；
- 实现安全资源弹性伸缩，满足灵活扩缩容需求；
- 实现业务个性化安全策略定制，使安全等资源更好的服务于业务差异化需求。
- 实现设备安全容错，与安全业务解耦，降低故障业务的影响。

12.2 部署场景

作为流量编排设备，设备作为三层网络设备接入网络中，对上下游设备路由过来的流量（请求和响应）进行编排处理，然后再按照路由规则发出。流量编排功能支持基于用户安全策略需求实现个性化、定制化流量编排。

流量编排功能支持四种部署场景：

- **入向代理**

编排设备对外提供的虚拟服务供客户端访问，可对后端服务器做负载均衡，可处理明文或 SSL 流量。

在该场景中，流量编排功能支持：

- 对流量进行加解密
- 流量负载均衡
- 支持根据安全需求将流量转发到不同安全服务链处理。

➢ **入向透明**

编排设备作为三层网关部署，客户端访问编排设备后端的某台设备，可处理明文或 SSL 流量。

在该场景中，流量编排功能支持：

- 对流量进行加解密
- 支持根据安全需求将流量转发到不同安全服务链处理。

➢ **出向透明**

编排设备作为三层网关部署，客户端访问外部资源，由客户端完成 DNS 解析，可处理明文或 SSL 流量，可与 NAT 和出向 LLB 结合使用。

在该场景中，流量编排功能支持：

- 对流量进行加解密（通过 SSL 侦听）
- 支持根据安全需求将流量转发到不同安全服务链处理。

➢ **网桥**

编排设备作为二层网桥部署，客户端访问编排设备后端的某台设备，可处理明文或 SSL 流量。

在该场景中，流量编排功能支持：

- 对流量进行加解密。
- 支持根据安全需求将流量转发到不同安全服务链处理。

注意：当编排设备作为二层网桥部署时，不支持 L4 安全设备。

12.3 功能原理

流量编排功能主要包含流量监听器、安全服务链、安全服务、安全设备、编排策略、SSL 加解密和健康检查几个概念。

12.3.1 流量监听器

流量监听器是由一组上行链路和下行链路接口组成，作为流量编排的起点和终点。

12.3.2 安全服务链

安全服务链定义了流量需要依次通过的安全服务的路径，可以包含 L2 安全服务、L3 安全服务、L4 安全服务和 TAP 服务。流量命中安全服务链时，流量依次发往服务链中各个安全服务进行处理，例如外部 SSL 加解密、安全检查、安全审计等。

12.3.3 安全服务

安全服务是安全服务链中包含的成员，通常由一台或者多台安全设备共同提供同类的安全服务。流量命中安全服务时，支持将流量均衡负载到多台安全设备。当安全服务无可用时，支持拒绝客户端访问或者绕过该安全服务的处理。安全服务按照类型可以分为 L2 安全服务、L3 安全服务、L4 安全服务和 TAP 服务。

12.3.4 安全设备

安全设备是实际上对流量进行安全处理的设备，是组成安全服务的成员。按照部署方式，安全设备可以划分为 L2 安全设备、L3 安全设备和 L4 安全设备。

注意：TAP 安全服务只支持一个 TAP 设备，TAP 设备在定义 TAP 服务时一同配置。

12.3.5 编排策略

编排策略定义了流量需要流经的路径。系统支持针对流量监听器、安全服务链和后台服务组配置安全策略。

12.3.5.1 针对流量监听器的编排策略

此类编排策略将流量监听器接收的流量分发到特定的安全服务链或虚拟服务，或者阻断流量。

对于流量编排场景，必须为流量监听器配置编排策略，否则流量监听器接收的流量按照正常业务进行处理。

12.3.5.2 针对安全服务链的编排策略

此类编排策略将安全服务链处理后的流量分发到下一个安全服务链或虚拟服务。此类编排策略不支持阻断流量。

12.3.5.3 针对后台服务组的编排策略

此类编排策略将命中后台服务组的流量分发到其他安全服务链。此类编排策略通常用于联结上一个分发对象为虚拟服务的编排策略。

12.3.5.4 编排策略优先级

编排策略的优先级取值越大，匹配时优先级越高。

12.3.6 流量编排规则

流量编排规则定义了编排策略的命中条件。流量编排规则包括网络流量编排规则和应用流量编排规则。应用流量编排规则与网络流量编排规则关系为“或”，网络流量编排规则的优先级高于应用流量编排规则。编排策略可以关联的编排规则最大数目为 32 条（包括网络编排规则与应用编排规则）。

编排规则按照优先级匹配流程如下：

1. 如果匹配网络流量编排规则，流量直接命中该编排规则关联的编排策略。
2. 如果不匹配，系统将查看是否存在应用流量编排规则。如果不存在，系统会执行步骤 1。
    a. 如果存在，系统将检查 DPI 功能是否启用。如果未启用 DPI 功能，系统会执行步骤 1。
    b. 如果 DPI 功能启用，系统将查看流量的源 IP、源端口、目的 IP 和七层协议是否匹配应用流量编排规则。如果不匹配，系统会执行步骤 1。
    c. 如果匹配，系统将根据协议内容识别结果决定是否匹配应用流量编排规则。
3. 如果不匹配，系统将匹配下一条优先级低的编排规则。

12.3.6.1 网络流量编排规则

当流量的五元组（源 IP、源端口、目的 IP、目的端口和协议）匹配网络流量编排规则，那么流量命中编排规则关联的编排策略。

12.3.6.2 应用流量编排规则

当流量的源 IP、源端口、目的 IP 和七层协议匹配应用流量编排规则，且系统能识别协议内容时，流量会命中编排规则关联的编排策略。为了匹配七层应用流量，管理员需要启用 DPI 功能。

注意：

- 对于 HTTP/1 类型的客户端请求，必须携带 Host 头部系统才能识别出客户端请求的协议类型。
- 对于 FTP 协议类型，设备只支持识别 FTP 的控制连接，不支持识别 FTP 的数据连接。
- 对于 SMTP、POP3 及 IMAP 协议，设备首次识别流量时，需要等待 3s 再做流量识别。对于 FTP 协议，设备首次识别流量时，FTP 的控制通道需要等待 3s 再做流量识别；而 FTP 的数据通道的流量每次都需要等待 3s 才能传输。对于 FTPS 协议，FTPS 的控制通道不需要等待 3s 直接做流量识别；而 FTPS 的数据通道的流量每次都需要等待 3s 才能传输。
- 对于不同源 IP 和源端口、相同目的 IP 和目的端口的流量，设备只根据第一次命中流量编排策略的流量识别协议类型，对后续不同源 IP 和源端口、相同目的 IP 和目的端口的流量不再识别协议类型。
- 在 SSL 加密场景下配置应用流量编排规则时，需要为后台服务组配置一个默认策略，用来保证当设备未识别出流量的类型时，可以把流量回发到对应的虚拟服务进行 SSL 加密。

12.3.7 安全流量加解密

在入向部署场景，流量编排功能支持自有使用 SSL 功能对流量进行加解密，对于入向代理场景，还可支持外部 SSL 网关设备对流量进行加解密。

如果要使用外部 SSL 网关设备进行加解密，需要将外部网关定义为 L4 安全服务。

在出向部署场景，流量编排功能支持使用 SSL 侦听功能对流量进行加解密。

12.3.8 健康检查

流量编排健康检查功能支持对 L2/L3 安全设备进行健康检查。L4 安全设备的健康检查使用关联的 TCP 类型的 SLB 后台服务的健康检查配置。

L2 安全设备支持的健康检查方式为穿透方式。在穿透方式下，健康检查流量穿透安全设备，同时流经上行链路和下行链路。

L3 安全设备支持的健康检查方式为：

- 穿透方式（bidirect）：在穿透方式下，健康检查流量穿透安全设备，同时流经上行链路和下行链路。穿透方式支持的健康检查类型为 TCP。

- 非穿透方式（unidirect）：在非穿透方式下，健康检查流量被转发到安全设备，分别流经上行链路和下行链路。

非穿透方式支持的健康检查类型为 ICMP、TCP、Half-TCP、HTTP 和 HTTPS。ICMP 类型的健康检查默认对上行链路和下行链路都做健康检查；TCP、Half-TCP、HTTP 和 HTTPS 类型的健康检查默认只对上行链路做健康检查。

Half-TCP 类型的健康检查通过与安全设备建立 TCP 半连接来判断安全设备端口的健康状态，TCP 半连接仅需要向安全设备发送请求并能够收到响应即可认为设备健康。

另外，L3 安全设备还支持配置附加健康检查，允许管理员根据需要灵活地为上行链路、下行链路或双向链路配置健康检查。L3 安全设备附加健康检查的类型包括 ICMP、TCP、Half-TCP、HTTP 和 HTTPS。

12.4 配置指南

使用流量编排功能时，需要根据安全需求定义流量编排路径，并绘制编排拓扑，然后针对编排拓扑中节点逐个配置。

注意：

- 流量编排不支持 SLB 快速转发功能，在使用流量编排功能时需要关闭 SLB 快速转发功能（通过命令“**slb directfwd off**”）。
- 配置流量编排功能时，管理员必须将虚拟服务的传输模式设置为透明模式（通过命令“**system mode transparent** *[virtual_service]*”）。

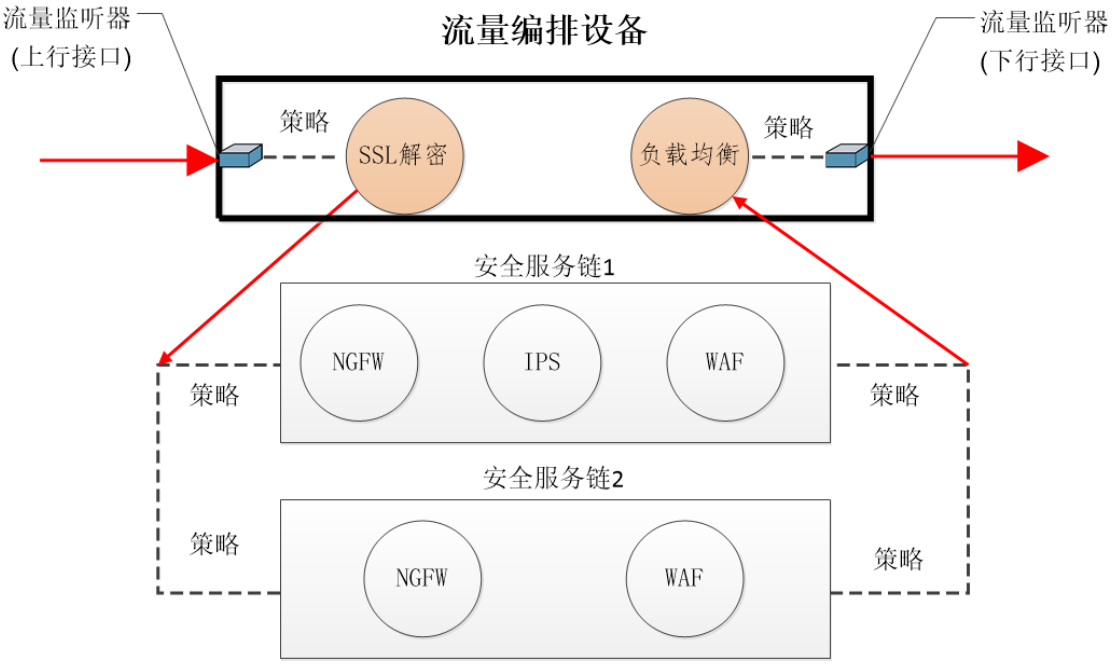

图12–1 典型流量编排拓扑

在上图中，流量编排流程为：

1. 流量编排设备收到流量后，按照流量监听器的编排策略将流量转发到 SLB 虚拟服务进行 SSL 解密，解密后流量的目的 IP 地址不变，目的端口变化。

2. 流量编排设备按照后台服务组中的不同编排策略，将流量分发到不同安全服务链处理。

3. 经过安全服务链处理后，编排设备再按照安全服务链的编排策略将流量转发到 SLB 虚拟服务进行负载均衡，然后根据 SLB 负载均衡策略和算法将流量分发到后台服务器。

在实际业务处理场景，用户可以根据实际业务需要定制流量编排流程。关于配置示例，请参考《流量编排部署指南》文档。

第13章 反向代理缓存

13.1 概述

本章描述了反向代理缓存（Reverse Proxy Cache）的概念和配置策略。反向代理缓存功能可以协助用户增强 Web 服务器的整体速度和性能，将请求频繁的数据缓存在设备的内存中，从而减少后台服务器的负荷。

13.2 反向代理缓存的原理

13.2.1 反向代理缓存的工作原理

反向代理缓存的位置一般位于 Web 服务器的前端，它可以接收来自互联网客户端的请求，并且同后台 Web 服务器一起来为这些请求提供回应，如下图所示。

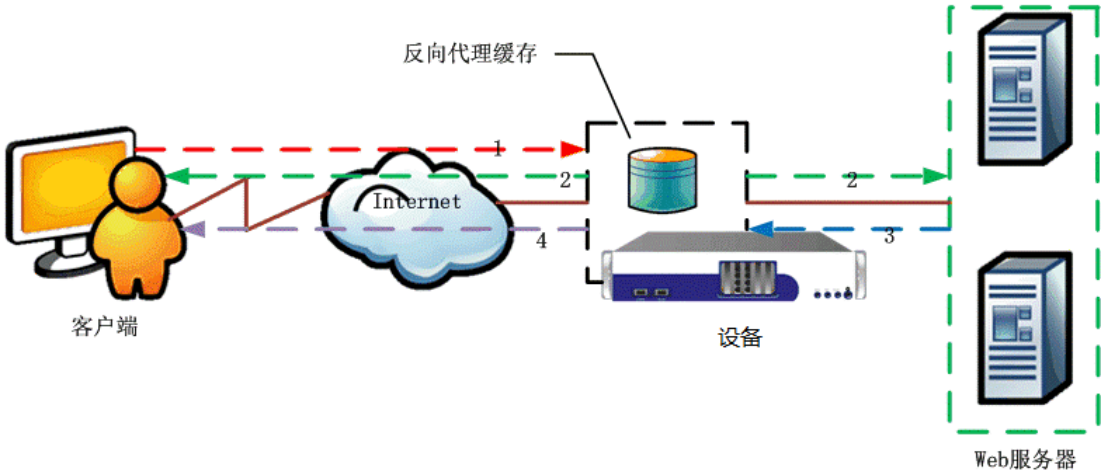

图13–1 反向代理缓存工作原理

1. 互联网上的客户端向设备发送一个请求，请求的内容是后台 Web 服务器上的一个文档。设备会将客户端的请求交给缓存模块处理。
2. 如果请求的是在设备上已经缓存了的一个尚未过期的副本，那么设备会直接将该副本发送给客户端，而不再向后台 Web 服务器转发请求。如果缓存中没有客户端所请求的内容，则该请求会被转发到后台 Web 服务器处理。
3. 后台服务器将请求的内容返回给设备。
4. 设备则会在将内容发送给客户端的同时缓存该内容。如果后续有更多关于该内容的请求，设备将用本地缓存的内容来进行响应。

预设情况下，当缓存中正在填入新的缓存项目时，设备无需等待缓存全部完成，即可以将该项目的数据实时的向客户端传送。

对于不同内存的设备，允许缓存的内容的最大字节数根据不同型号设备的系统内存而有所不同，如下表所示。

表13–1 设备反向代理缓存的内存最大字节数

设备的系统内存	缓存对象的最大字节数
2 GB	1024 KB（1 MB）
4 GB	10,240 KB（10 MB）
8 GB	20,480 KB（20 MB）
16 GB/24 GB/32 GB/64 GB	40,960 KB（40 MB）

13.2.2 设备反向代理缓存的优势

与传统的缓存功能相比，设备的反向代理缓存功能以保障设备的整体稳定和性能为前提，为客户提供了一个高效、智能和人性化的配置平台，协助客户以更加灵活的方式调整设备的缓存功能，以适应不同的 Web 网站，最终达到提升用户的 Web 网站的响应能力，降低后台服务器负载，为 Web 网站的最终用户提供完美的访问体验的目的。

设备的反向代理缓存功能具有如下优势：

- **优秀的性能保障**
  - 缓存功能作为一个单独的功能模块，当缓存功能关闭时，不会对系统造成任何影响。
  - 缓存功能严格的将自身的消耗控制在总内存的 25%以下。
  - 设备能够对已压缩和未压缩的内容进行缓存。这一特点使得设备可以将已缓存的压缩内容直接发送给客户端，无需再次压缩，从而提高了压缩内容的吞吐率。
- **注重整体稳定性**
  - 当缓存功能出现任何错误时，使用者可以随时关闭缓存功能，而不会影响到整个设备其他功能的运行。
  - 所有的缓存参数使用“缓存过滤器（Cache filter）”功能实现，减少全局控制选项，将配置中出现混乱和冲突的几率降到了最低。
- **智能监控**
  - 设备的监控功能（Monitor）会监控检测缓存的状态，当它发现任何问题时，会在适当的清理工作后重新启动缓存，而不影响客户应用。
- **灵活的配置方式**

- 命令“**show statistics cache**”输出的统计信息，将会说明使用者了解设备是如何做出缓存决定的，这有助于使用者对设备或网站进行优化调整。
- “缓存过滤器”增加了命令控制的精度，减少了全局控制参数，使用户对缓存的控制尺度更大，控制更加精确。
- 支持针对单个虚拟服务启用或关闭缓存功能。

13.2.3 缓存内容

HTTP 的流量分为两类：可缓存的内容和不可缓存的内容。内容的可缓存性是由 HTTP 头部的内容所决定的。反向代理缓存会检查请求和响应的 HTTP 头部来决定内容是否可缓存。如果对某个请求的响应被缓存了，那么对这个对象接下来的请求将由缓存来响应，而不是后台服务器。

在默认情况下，如果对象没有任何限制是否缓存的 HTTP 头部，反向代理缓存将缓存该内容。以下的是较常用的缓存控制头部（Cache_Control header），它们可以控制内容是否被缓存；如需缓存，还可以控制缓存多久。

- **Cache-Control: public**

关键词“public”表示该内容可被缓存，因为内容是公开的，而非私人的。

- **Cache-Control: private**

关键词“private”说明该内容是专门为某个使用者预留的特殊内容，不能为其他客户所用。因此不会被缓存。

- **Cache-Control: no-cache**

关键词“no-cache”表明如果此内容被缓存了，在每次使用缓存内容前，客户端必须询问服务器是否有更新。

- **Expires: Wed, 30 June 2010 14:00:00 GMT**

“Expires”头部定义了缓存内容的失效时间。如果缓存内容过期了，则必须重新向原来的服务器获得原始数据。在这个例子中，“Expires”头部规定了缓存内容将于格林威治时间 2010 年 6 月 30 日（星期三）下午两点失效。

13.2.3.1 可缓存的内容

任何缓存控制头部指出可以缓存的内容，都可以被设备缓存。如果内容中不含有缓存控制指示，该内容将被预设为是可以缓存的并且永远不会过期，直到该内容被手动从缓存中清除。如果内容中包含“Expires”头部，在过期之后，设备将在下次该内容被请求的时候，根据后台服务器的响应更新缓存中的内容。

13.2.3.2 不可缓存的内容

任何包含缓存控制头部，而且缓存头中明确禁止缓存的内容将不会被设备缓存。对含有 cookies 请求的响应通常不被缓存，除非用户使用命令允许含有 cookies 请求的响应被缓存。

13.2.4 缓存过滤器

缓存过滤器（Cache filter）可以说明用户通过简单的配置来精确的定义缓存的工作方式，如：来自特定主机的特定请求内容是否需要强制缓存、缓存的生存周期等。并且，缓存过滤器的控制参数优先级高于缓存控制头部（Cache-Control header）中相关参数的优先级。

下面是几个缓存过滤器的应用示例，更多详细的配置内容请参考本章“命令行配置示例”部分及命令行使用手册的相关介绍。

- 可以使设备缓存所有来自 www.xyz.com 主机的*.jpg 文件，并且缓存内容的生存周期为 200,000 秒。
- 可以使设备只缓存所有 JPG、GIF、BMP 等常用格式的图片文件。
- 可以设定主机上一部分缓存文件遵从缓存过滤器中的定义，另一部分遵从缓存控制头部对缓存的定义，比如该缓存内容的生存周期。
- 可以使设备在查询缓存的时候忽略请求 URL 中特定类型的查询字符串。

13.2.5 缓存内容的失效时间

在缓存运行的过程中涉及到三种缓存失效时间：

- HTTP 响应中缓存相关头部（Expires 和 Cache-Control 头部）所定义的缓存失效时间；
- 命令“**cache settings expire**”所定义的缓存失效时间；
- 命令“**cache filter rule**”中参数“ttl”所定义的缓存失效时间。

三种缓存失效时间的优先级如下：

1. 系统优先使用命令“**cache filter rule**”中的参数“ttl”所定义的缓存失效时间，如果匹配的缓存过滤规则中的参数“ttl”没有设置，则使用命令“**cache settings expire**”定义的缓存失效时间；
2. 如果缓存对象没有匹配命令“**cache filter rule**”定义的任何缓存过滤规则，则使用 HTTP 响应中缓存相关头部定义的缓存失效时间；
3. 如果 HTTP 响应头部中没有定义缓存失效时间，则使用命令“**cache settings expire**”定义的缓存失效时间。

13.2.6 缓存图片格式优化

系统支持缓存图片优化功能，通过转换图片格式的方式来压缩图片。当用户再次请求获取该图片时，系统将返回经过格式优化的图片，以此节省流量，提升页面访问速度。

目前，系统支持对 BMP、JPEG、PNG 格式的源文件进行格式优化，优化后的图片格式将被转换成 WebP 和 JPEG。

管理员可以为此功能设置待优化源图片最小值。小于该值的源图片将不再进行格式优化。管理员还可以配置图片优化例外策略，在客户端访问指定 URL 时跳过图片优化处理。当发往指定虚拟服务的 HTTP 请求中的请求 URL 匹配例外策略时，系统将不对 HTTP 响应中的图片文件进行优化。

13.3 反向代理缓存配置

13.3.1 配置向导

下面列出了配置反向代理缓存需用的命令，相关命令的描述信息，请参考命令行使用手册。

表13–2 反向代理缓存配置命令

配置操作	命令行
启用缓存	**cache {on|off}** *[virtual_service]*
查看缓存使用状态	**show cache status**
配置缓存全局失效时间配置	**cache settings expire** <*expire_time*>
设置可缓存项目的最大单个文件大小	**cache settings objectsize** <*size*>
查看缓存设置	**show cache settings**
查看缓存统计信息	**show statistics cache** *[virtual_service]*
清除缓存统计信息	**clear statistics cache** *[virtual_service]*
查看缓存内容	**show cache content** <*host_name*> <*url_regex*>
强制清除缓存内容	**clear cache content**
启用缓存过滤器	**cache filter {on|off}**
定义缓存过滤器规则	**cache filter rule** <*host_name*> <*url*> <*cache_behavior*>
查看缓存过滤器运行状态	**show cache filter status**
查看针对特定主机的缓存过滤器规则	**show cache filter hostname** <*host_name*>
查看所有的缓存过滤器规则	**show cache filter all**
查看主机同缓存过滤规则的匹配情况	**cache filter match** <*host_name*> <*url_regex*>
删除特定缓存过滤器规则	**no cache filter rule** <*host_name*> <*url*>

配置操作	命令行
清除针对特定主机的缓存过滤器规则	**clear cache filter hostname** *<host_name>*
清除所有缓存过滤器规则	**clear cache filter all**
查看缓存过滤器统计信息	**show statistics cachefilter** *<host_name> <url_regex>*
清除缓存过滤器统计信息	**clear statistics cachefilter** *[host_name]*
启用缓存图片优化功能	**cache imgopt {on|off}** *[virtual_service]*
设置可优化源图片的最小值	**cache imgopt sizelimit** *<min_size>*
设置图片优化的目标格式	**cache imgopt format** *[target_format]*
显示图片优化功能配置	**show cache imgopt settings**
配置图片优化例外策略	**cache imgopt urlexclude** *<virtual_service> <url>*
删除指定图片优化例外策略	**no cache imgopt urlexclude** *<virtual_service> <url>*
查看图片优化例外策略	**show cache imgopt urlexclude** *[virtual_service]*
删除所有图片优化例外策略	**clear cache imgopt urlexclude** *[virtual_service]*

13.3.2 配置示例

每个虚拟服务器上的缓存功能是独立运行的，在缺省情况下，缓存被禁用。当缓存被禁用时，任何内容都不会被保存在缓存中，所有请求都是通过服务器负载均衡机制去直接访问后台服务。

1. 启用缓存。

如果我们需要使用缓存功能，首先需要在相应的虚拟服务器上开启缓存功能。

在我们的例子中，我们想要在虚拟服务器 virtual_MOSS 上启用缓存

```
Demo(config)#cache on virtual_MOSS
```

当前缓存的状态可以通过“**show cache status**”命令来查看。

```
Demo(config)#show cache status
reverse proxy cache:                    enable
per-vs status " virtual_MOSS":          enable
```

2. 缓存基本设置。

现在就可以开始设备缓存功能的基本配置了，包含以下设置条目：

- 缓存中项目的生存周期；
- 可缓存项目的最大单个文件大小。

首先，我们可以使用“**show cache settings**”命令查看缓存的设置。

```
Demo#show cache settings
Cache Configuration:
        Cache Default Expiration:       82800 seconds
```

```
        Maximum Cacheable Object Size:    5120 KB
```

上面显示的所有缓存设置都是默认值，这些默认设置都已经被调整到通常情况下的最佳值，如果你的网络环境对于设备有特殊的需求，那么可以根据需要更改这些预设设置。

如果需要改变缓存条目的生存周期，我们可以使用“**cache settings expire**”命令。该命令定义了缓存中所有内容的失效时间，可以用 hh:mm:ss 或秒数的形式来确定，输入的秒数需要置于双引号内，缺省值是 82800 秒（23 小时）。缓存失效时间严格遵循 RFC2616 第 13 章第二节所定义的失效模型。在这种模型下，如果没有明确给出缓存中某个对象的失效时间，设备将使用缺省的全局失效时间。

```
Demo(config)#cache settings expire "43200"
```

改变单个缓存对象的最大字节，可以使用“**cache settings objectsize** <*size*>”命令。该命令是以千字节（KB）为单位。缺省值是 5120KB。如果发送给客户端的对象大小超过设置的最大值，该对象将不会被缓存，即使它是可缓存的。

```
Demo(config)#cache settings objectsize 1000
```

现在我们可以使用“**show cache settings**”命令来查看我们所做的缓存配置。

```
Demo(config)#show cache settings
Cache Configuration:
        Cache Default Expiration:       43200 seconds
        Maximum Cacheable Object Size:  1000 KB
```

3. 配置缓存过滤器。

首先，使用“**cache filter {on|off}**”命令启用缓存过滤器，缓存过滤器功能在默认情况下是关闭的。

```
Demo(config)#cache filter on
```

接下来，我们可以通过命令“**cache filter rule** <*host_name*> <*url*> <*cache_behavior*>”来配置缓存过滤规则，这些规则可以控制是否缓存某个对象以及缓存内容的失效时间。

在我们的例子中，我们要配置设备缓存所有从主机“www.xyz.com”发来的“.jpg”对象，并且缓存的失效时间设置为 200,000 秒。

```
Demo(config)#cache filter rule www.xyz.com ".*\.jpg" "cache=yes" "ttl=200000"
```

完成缓存过滤器规则配置后，我们可以通过“**show cache filter all**”命令查看我们配置的所有缓存过滤规则。

```
Demo(config)#show cache filter all
cache filter rule "www.xyz.com" ".*\.jpg" "cache=yes" "urlquery=yes" "ttl=200000"
cache filter rule "www.xyz.com" ".*\.bmp" "cache=yes" "urlquery=yes" "ttl=200000"
```

```
cache filter rule "www.xyz.com" ".*\.gif" "cache=yes" "urlquery=yes" "ttl=200000"
cache filter rule "www.test.com" "sample" "cache=yes" "urlquery=yes" "ttl=150000"
cache filter rule "www.test.com" ".*\.jpg" "cache=yes" "urlquery=yes" "ttl=200000"
```

4. 配置缓存图片格式优化功能。

使用以下命令配置缓存图片格式优化功能：

设置待优化图片的最小值：

```
Demo(config)#cache imgopt sizelimit 5
```

设置图片优化的目标格式：

```
Demo(config)#cache imgopt format webp
```

启用缓存图片格式优化功能：

```
Demo(config)#cache imgopt on
```

配置图片优化例外策略：

```
Demo(config)#cache imgopt urlexclude vs01 /request/image
```

5. 查看缓存配置。

如果要查看缓存的相关统计信息，请使用“**show statistics cache**”命令。

```
Demo(config)#show statis cache
Reverse Proxy Cache Global Statistics:
Basic Statistics:
    Requests received:                          3601254
    Requests with GET method:                   3601254
    Requests with HEAD method:                  0
    Requests with PURGE method:                 0
    Requests with POST method:                  0
    Number of open client connections:          115
    Number of open server connections:          115
    Requests redirected to HTTPS:               0
    Requests redirected based on regex match:   0
    Requests forwarded with rewritten url:      0
    Locations rewritten to HTTPS:               0
    Locations rewritten based on regex match:   0
    Cache skip, cache off:                      3601254
    Cache hit, reply using cache:               0
    Cache hit, reply with "Not Modified":       0
    Cache hit, reply with "Precondition Failed": 0
    Cache hit, revalidate:                      0
```

```
    Cache miss, noncacheable requests:                3601254
    Cache miss, create new entry:                     0
    Cache miss, create new entry, resp noncacheable: 0
    Hit ratio:                                        0.00%

    (Notice: the real server's time should be in sync with this machine.
             Otherwise, the time difference could expire the cachable objects
             resulting in low cache hit ratio.)

Advanced Statistics:
    Number of cache objects:                          0
    Number of cache frames:                           0
    Successful cache probes:                          0

  Why were certain requests sent to the server?
  a) We had to revalidate the cached object due to:
       Request with "no-cache":                       0
       Requset with "maxage=0":                       0
       Cached object had "no-cache":                  0
       Cache object expired:                          0

  b) We had to bypass cache for some requests because:
       Cache was filling when request was made:       0
       Revalidation failed due to IMS mismatched:     0
       Client has newer copy, cannot send from cache: 0
       Object in cache is chunked, cannot give to 1.0 client: 0
       Network memory utilization was too high:       0

  c) Request cannnot be served from cache because:
       Cache filter denied caching:                   0
       Requests with "no-store":                      0
       Requests with "authorization":                 0
       Requests with cookies:                         0
       Requests with range:                           0
       Requests non GET, non HEAD:                    0
       Requests URL too long:                         0
       Requests host too long:                        0

  d) Error occured while doing cache lookup
       Network memory shortage when cache hit (200, 304): 0
       Cache was not accessible:                      0
       Fail to send cache lookup to cache:            0
```

```
        Fail to find url and host:                          0
        Fail to parse cache specific http request headers:  0
        Fail to create a new cache object:                  0
        Noncacheble requests due to other errors:           3601254

Why were certain responses not stored in cache?
a) HTTP directive in response told us not to cache
        HTTP response code not 200, 300 or 301:             0
        Response had a "no-store":                          0
        Response had a "private":                           0
        Response had a "set-cookie":                        0
        Response had a "vary":                              0

b) The response did not meet our guidelines for cacheability
        Response noncacheable too big:                      0

 c) Error occured when trying to cache response
        Cache storage limit exceeded based on header data:  0
        Cache storage limit exceeded based on payload:      0
        Network memory shortage when storing response body: 0
        Cache object was deleted before response arrived:   0
        Fail to parse cache specific http response headers: 0
        Fail to store response headers in cache:            0
        Fail to store response body in cache:               0
        Cache object was aborted due to connection reset:   0
        Noncacheble responses due to other errors:          0
```

第14章 HTTP 内容改写

14.1 概述

本章描述了 HTTP 内容改写（HTTP Content Rewrite）功能的概念、工作原理和配置方法。

HTTP 内容改写功能旨在说明最终用户正确访问后台服务器上的网页信息，同时尽可能的降低网络等待时间，提高使用者体验。

14.2 HTTP 内容改写的原理

14.2.1 HTTP 内容改写的工作原理

当应用服务器部署在设备后端时，由于无法通过设备直接访问后台资源链接的地址，来自互联网的终端使用者不能通过 Web 浏览器正确访问应用服务器上的网页，如下图所示：

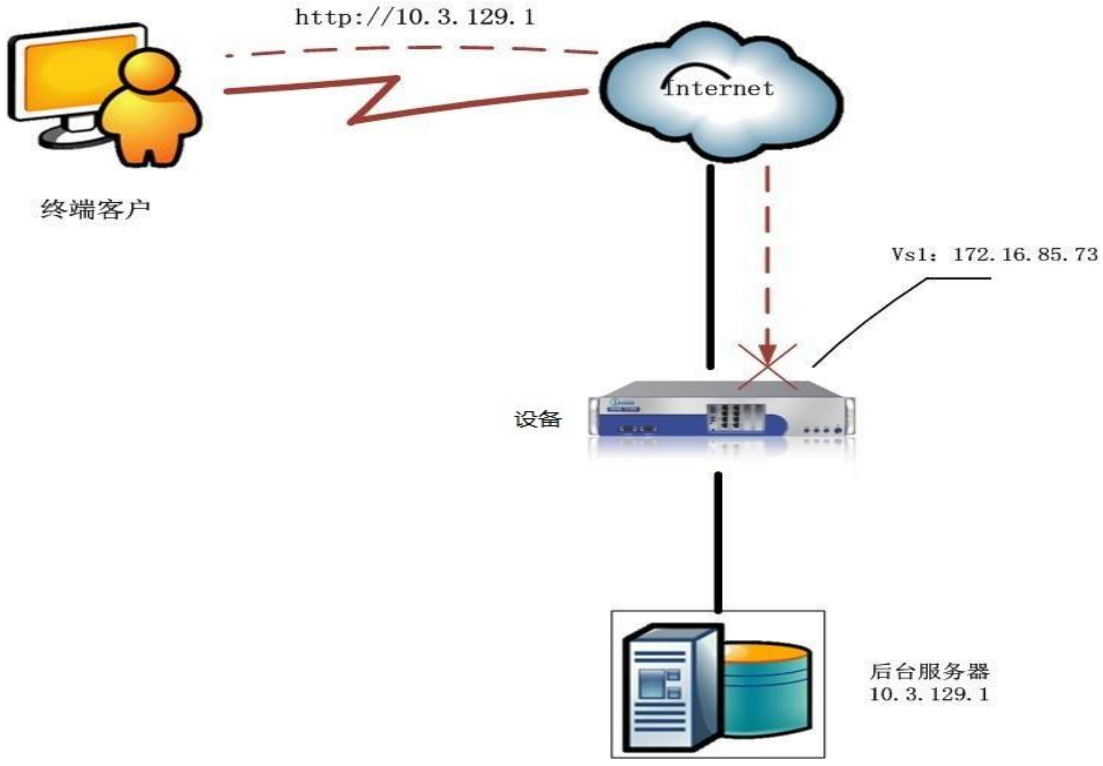

图14–1 用户无法访问内网网页

为了解决以上问题，设备提供了 HTTP 内容改写功能，在不降低终端使用者体验的前提下，通过对内网网页内容进行改写和缓存，使终端使用者通过浏览器能够获取内网网页上的内容。

设备在接收到来自后台服务器的响应数据之后，首先会读取响应数据中需要改写的内容，并将这些内容发送给设备的内容改写模块进行内容改写操作。在这个过程中，所有需要改写的文件中的地址或域名都将被映像或替换成互联网用户可以正确访问的格式。最后，设备会将这些改写后的内容返回给终端使用者，并缓存改写后的内容。如下图所示：

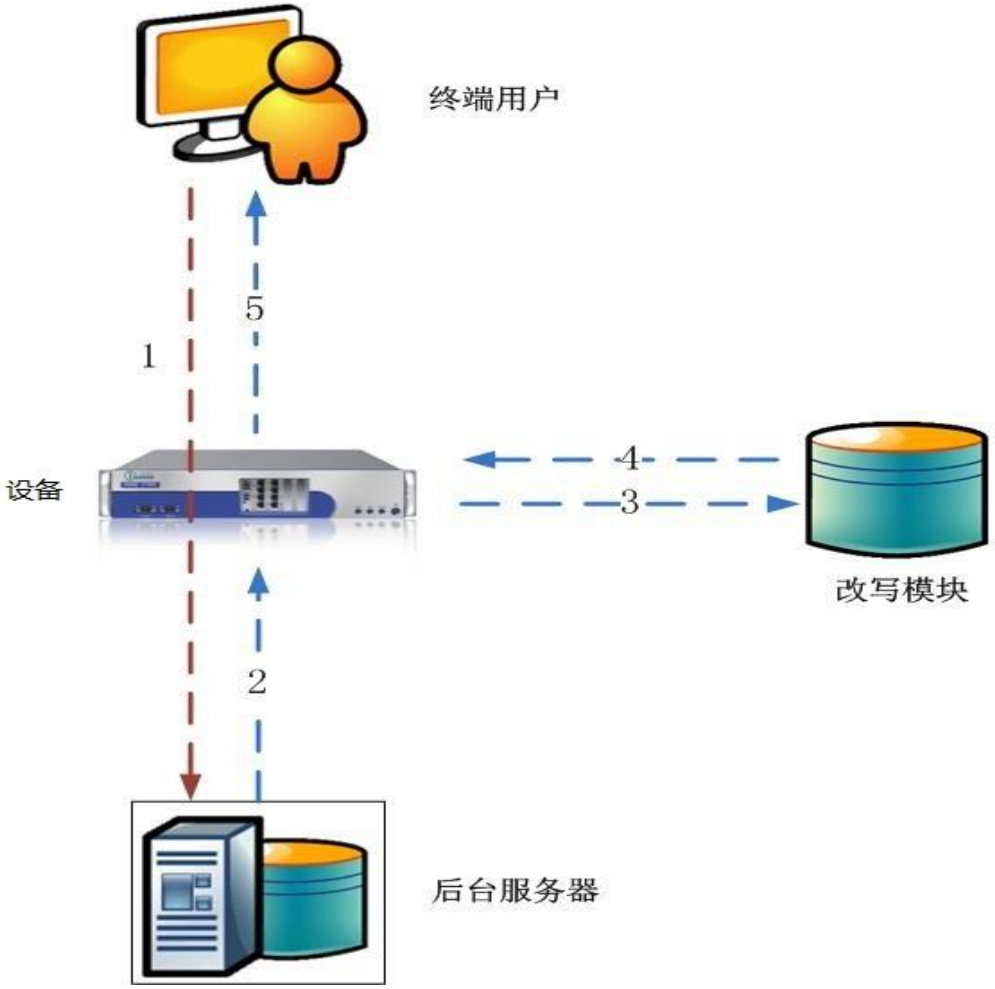

图14–2 HTTP 内容改写工作原理

1. 互联网上的终端使用者通过设备向后台服务器发送 HTTP 请求。
2. 后台服务器接收到请求之后，将响应数据发送到设备上。
3. 设备将后台服务器的响应数据发送到设备的改写模块。
4. 设备的改写模块读取后台服务器的响应数据，并对数据进行改写。
5. 设备将改写完的数据作为响应发送给终端使用者，同时对改写完的内容进行缓存。

14.2.2 HTTP 内容改写的优势

同 URL 改写方式相比，设备的 HTTP 内容改写功能在确保用户网络体验的前提下，尽可能的减少了设备与后台服务器的通信次数，从而降低了系统负载，提高了服务的响应速度。

HTTP 内容改写功能具有以下优势：

- **提升用户体验**

  HTTP 内容改写功能解决了公网用户访问企业内网页面时出现的问题，通过自定义改写策略，设备可以将内网页面上所有的内容改写成终端使用者可见的形式，从而带给用户完美的网络体验。此外，设备的 HTTP 内容改写功能还支持多字节字符，可以对中文、日文等语言进行改写操作。

- **降低对性能的影响**

  同传统的 URL 改写方式相比，使用 HTTP 内容改写功能，设备无需与后台服务器频繁的通信，从而降低了改写操作对系统性能的影响。

- **提高系统的响应速度**

  设备除了对页面内容进行改写操作以外，还可以对改写后的内容进行缓存。当下一次终端使用者再向后台服务器发送相同的页面请求时，设备将使用缓存中的改写内容进行响应，从而提高系统的整体响应速度。

- **易于维护**

  HTTP 内容改写功能通过内容改写引擎和改写模块自动实现，管理员只需要进行简单的配置，内容改写功能便会根据改写规则自动执行，无需管理员值守维护，从而为企业降低了人力成本。

14.2.3 HTTP 内容改写的工作方式

终端使用者在访问内网网页时，无法通过设备直接访问网页链接中的后台服务器 IP 地址或域名。HTTP 内容改写的任务，就是将这些无法直接访问的内网 IP 地址或域名通过映像或替换的方式，改写成终端使用者可以直接访问的 IP 地址和域名。HTTP 内容改写功能通过以下方式实现：

- **支持针对全局或单个虚拟服务开启内容改写**

设备允许定义全局的 HTTP 内容改写，也支持针对某个虚拟服务启用或关闭 HTTP 内容改写功能。

注意：

- 预设情况下，全局 HTTP 内容改写功能是禁用的，单个虚拟服务的 HTTP 内容改写功能是启用的。

- 只有启用了全局 HTTP 内容改写功能，单个虚拟服务的 HTTP 内容改写功能的启用及相关配置才会生效。

➢ 定义静态的全局改写规则

HTTP 内容改写功能支持通过定义改写规则将网页文件中的 IP 地址、域名或字符串通过映像或替换的方式，改写成新的格式。

HTTP 内容改写功能支持两种内容改写方式：

- **ProxyHTMLURLMap**

将网页文件中 HTML 标签内包含的 URL 替换成新的 URL。这种改写方式只适用于改写 HTML 和 XHTML 类型的档。例如：

替换前的档内容

```
<p><a href="10.3.129.1">10.3.129.1</a></p>
<p><a href="http://10.3.129.1/">http://10.3.129.1</a></p>
<p><a href="https://10.3.129.1/">https://10.3.129.1</a></p>
```

替换后的档内容

```
<p><a href="172.16.85.74">10.3.129.1</a></p>
<p><a href="http://172.16.85.74/">http://10.3.129.1</a></p>
<p><a href="https://172.16.85.74/">https://10.3.129.1</a></p>
```

如上面的例子所示，网页文件中只有 HTML 标签内的 URL 由“10.3.129.1”替换为“172.16.85.74”，其余内容没有被替换。

- **Substitute**

将网页文件中所有与改写规则匹配的 URL 全部替换（包括 HTML 卷标内和卷标外的 URL），例如：

替换前的档内容

```
<p><a href="10.3.129.1">10.3.129.1</a></p>
<p><a href="http://10.3.129.1/">http://10.3.129.1</a></p>
<p><a href="https://10.3.129.1/">https://10.3.129.1</a></p>
```

替换后的档内容

```
<p><a href="172.16.85.74">172.16.85.74</a></p>
<p><a href="http://172.16.85.74/">http://172.16.85.74</a></p>
<p><a href="https://172.16.85.74/">https://172.16.85.74</a></p>
```

如上面的例子所示，所有的“10.3.129.1”都被替换成了“172.16.85.74”。

注意：

- “rule”参数的所有内容都要置于双引号内。
- “ProxyHTMLURLMap”和“Substitute”要严格区分字母的大小写。
- 当“ProxyHTMLURLMap”和“Substitute”两种模式同时配置时，“ProxyHTMLURLMap”会生效。
- 更改改写规则将导致当前正在进行的内容改写操作会失败，设备将重置相应的连接。因此，建议不要在设备正在处理流量的时候更改改写规则。
- 如果启用了 HTTP 内容改写功能并配置了改写规则，响应消息中每行的长度不能超过 1MB，否则设备将会向客户端发送 RST 报文中断 TCP 连接。

➢ 定义允许改写的文件类型

HTTP 内容改写功能可以启用或关闭对指定类型的档的改写操作。该功能所支持的文件类型如下：

- text/html
- text/plain
- text/richtext
- text/xml
- application/xml
- application/xhtml+xml
- text/css
- text/javascript
- application/javascript

设备预设启用了 text/html 类型的改写操作。也就是说，在预设情况下，HTTP 内容改写功能不会改写除 text/html 以外的其他类型的档的内容。

➢ 定义基于单个虚拟服务的文件 URL 字符串

除了定义文件类型之外，HTTP 内容改写功能还允许管理员为单个虚拟服务自定义字符串来过滤出允许或拒绝进行改写操作的档，再根据之前设置的过滤规则，对允许改写的档进行改写。

在定义文件的 URL 字符串时，首先需要定义一个 URL 清单，并向这个列表中添加条件字符串。条件字符串可以是文件的扩展名、档内容或文件名的一部分。然后再通过一个允许/拒绝操作规则，将这个 URL 清单绑定到一个虚拟服务上，该虚拟服务上的任何匹配该字符串的文件都会被执行规则所定义的操作。

管理员可以向同一个 URL 清单中添加多个 URL 字符串，每个 URL 字符串之间是“或”的关系，网页档的文件名、扩展名或内容只要匹配其中任意一条字符串，该文件就会被执行规则所定义的操作。

- **定义允许改写的页面的 HTTP 回应状态代码**

HTTP 内容改写功能还支持改写包含特定 HTTP 响应状态代码的网页档。管理员可以通过命令行或 WebUI 接口增加或减少 HTTP 响应状态代码，来配置只改写某些包含特定 HTTP 响应状态代码的档，而包含其他 HTTP 响应状态代码的档不会被改写。

预设情况下，设备只对回应状态代码为“200”的网页档的改写操作，包含其他 HTTP 响应状态代码的网页档不会被改写。

14.3 HTTP 内容改写配置

14.3.1 配置向导

下图描述了一个 HTTP 内容改写配置的应用场景：

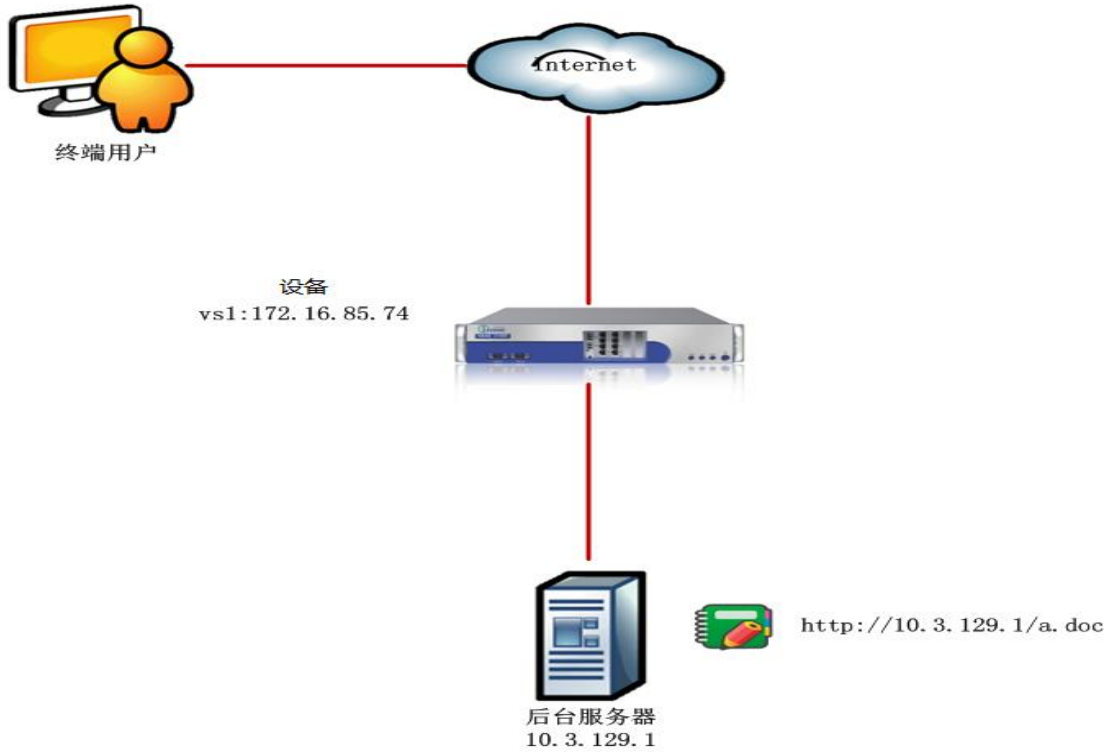

图14–3 HTTP 内容改写配置

图中，终端使用者通过设备访问后台服务器上的页面资源，由于后台服务器隐藏在设备后面，终端使用者无法从外网访问页面上的原始链接地址 http://10.3.129.1/a.doc。因此，管理员需要在设备上配置 HTTP 内容改写功能，将原始的链接地址改写成 http://172.16.85.74/a.doc。

下面列出了配置内容改写所需用的命令，相关命令的描述信息，请参考命令行使用手册。

表14–1 HTTP 内容改写配置命令

配置操作	命令行
启用 HTTP 内容改写	**http rewrite body {on|off}** *[virtual_service]*
定义 HTTP 内容改写规则	**http rewrite body rule** <*rule*> *[flags]*
定义允许改写的文件类型	**http rewrite body mimetype** <*mime_type*>
为一个 URL 改写清单添加一个条件字符串	**http rewrite body url list** <*url_list*> <*url*>
配置允许改写的 URL 字符串	**http rewrite body url permit** <*virtual_service*> <*url_list*>
配置禁止改写的 URL 字符串	**http rewrite body url deny** <*virtual_service*> <*url_list* >
定义允许改写包含某一回应状态代码的页面	**http rewrite body statuscode** <*status_code*>

14.3.2 配置示例

1. 配置 SLB 后台服务及虚拟服务，并配置服务器负载均衡算法。

```
Demo(config)#slb real http "rs1" 10.3.129.1 80 1000 tcp 3 3
Demo(config)#slb virtual http "vs1" 172.16.85.74 80 arp 0
Demo(config)#slb policy static "vs1" "rs1"
```

2. 启用 HTTP 内容改写功能。

```
Demo(config)#http rewrite body on
```

3. 查看虚拟服务的 HTTP 内容改写功能的启用状态。

```
Demo(config)#show http rewrite body status
http body rewrite:              enable
per-vs status "vs1":            enable
```

4. 定义 HTTP 内容改写的规则，将地址 10.3.129.1 改写为 172.16.85.74。

```
Demo(config)#http rewrite body rule "ProxyHTMLURLMap 10.3.129.1 172.16.85.74" -R
```

5. 查看已经定义的 HTTP 内容改写规则。

```
Demo(config)#show http rewrite body rule
http rewrite body rule "ProxyHTMLURLMap 10.3.129.1 172.16.85.74" "-R"
```

6. 定义需要改写的文件类型。

```
Demo(config)#http rewrite body mimetype xml
```

第15章 DNS 缓存

15.1 概述

本章我们将介绍 DNS 缓存（DNS Cache）的相关配置。

设备上的 DNS 服务负载均衡提供了缓存功能。每当设备收到一个 DNS 服务发送回来的 A 记录或 AAAA 记录响应，就会将它缓存下来。然后，当设备再次收到访问这条 A 记录或 AAAA 记录的客户请求时，设备会直接将缓存中的 A 记录或 AAAA 记录发送给客户端。如果缓存中没有相关记录，设备就会将 DNS 请求转发给后台服务器，并将后台服务器返回的结果保存至缓存，以待下一次的 DNS 请求。

15.2 DNS 缓存配置

15.2.1 配置向导

下面列出了配置 DNS 缓存所需的命令，相关命令的描述信息，请参考命令行使用手册。

表15–1 设备 DNS 缓存配置命令

配置操作	命令行
配置相关服务器负载均衡的设置	**slb real dns** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down] [timeout]* **slb virtual dns** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb policy static** *<virtual_service> <real_service>*
开启 DNS 配置	**dns cache {on|off}**
配置DNS缓存的失效时间	**dns cache expire** *<min_seconds> <max_seconds>*
建立DNS缓存所需的主机	**dns cache host** *<host_name> <ip>*

15.2.2 配置示例

1. 配置 DNS 缓存相关的服务器负载均衡设置。

关于服务器负载均衡设置的相关信息，请参考“服务器负载均衡（SLB）”章节的相关介绍。

```
Demo(config)#slb real dns "RS_DNS_1" 10.1.1.10 53 1000 icmp 1 1 20
```

```
Demo(config)#slb virtual dns "VS_DNS_1" 10.1.61.100 53
Demo(config)#slb policy static "VS_DNS_1" "RS_DNS_1"
```

在上面的命令中，我们先定义了一个 DNS 类型的后台服务。然后又将这个 DNS 服务以静态方式关联到了一个虚拟服务。相关命令的描述信息，请参考命令行使用手册。

2. 启用 DNS 缓存。

设备的 DNS 缓存功能默认是关闭的。如果想开启这个功能，可以使用“**dns cache {on|off}**”命令。

```
Demo(config)#dns cache on
```

3. 配置 DNS 缓存的失效时间。

```
Demo(config)#dns cache expire 1 36000
```

4. 建立 DNS 缓存所需的主机。

```
Demo(config)#dns cache host "sting" 10.1.61.200
Demo(config)#dns cache host "gunrose" 10.1.61.100
Demo(config)#dns cache host "roxxette" 10.1.61.2
Demo(config)#dns cache host "queens" 10.1.61.47
```

第16章 HTTP 压缩

16.1 概述

本章将介绍设备的 HTTP 压缩（HTTP Compression）功能。

设备支持对 HTTP 对象的在线压缩。此功能有助于优化特定网站的流量，同时终端使用者也可体验到更快的下载速度。这章描述了 HTTP 数据压缩功能的配置方法，该功能也是设备平台的一部分。

16.2 HTTP 压缩的原理

HTTP 压缩（或内容编码）是一种公开的对来自 Web 服务器去往浏览器的文本内容进行压缩的方式。HTTP 压缩使用公共域压缩算法来压缩服务器上的 XHTML、JavaScript、CSS 和其他文本文件。

在预设情况下，设备可以为所有的浏览器压缩以下的资源类型。

- 文本（text/plain）
- HTML（text/HTML）
- XML（text/XML）

而对于以下的资源类型，设备只能为特定的浏览器提供压缩功能。

- Java Scripts（application/x-javascript）
- Cascade Style Sheets（text/css, application/x-pointplus）
- PDF 文档（application/pdf）
- PPT 文档（application/powerpoint）
- XLS 文档（application/MSExcel）
- DOC（application/MSWord）

现在如果网络管理员不想压缩特定类型的文本内容，可以通过使用“**http compression policy urlexclude** <*virtual_service*> <*url*>”命令为客户端请求配置一个 URL-exclude HTTP 压缩规则来实现。如果客户端的 URL 请求符合这个规则，那么即使此时设备的 HTTP 压缩功能已经被打开，这些从 Web 服务器通往浏览器的内容也不会被压缩。

同时也并不是所有的文件类型都适合被压缩。很多文件类型已经经过压缩，例如 JPEG、GIF、PNG、视频以及一些经过打包的档内容如 ZIP、GZIP 和 BZIP2 档，这些档不会被压缩。换句话说，如果一个网站上的负载主要来源于这些文件类型，那么设备的 HTTP 压缩功能不会为这个网站的性能和负载带来太大效益。

注意：对于某些非常小的档来说，经过压缩之后的大小或许会比压缩前更大。

16.3 HTTP 压缩配置

16.3.1 配置向导

下面列出了配置 HTTP 压缩所需用的命令，相关命令的描述信息，请参考命令行使用手册。

表16–1 设备 HTTP 压缩配置命令

配置操作	命令行
启用 HTTP 压缩	**http compression {on|off}** *[virtual_service]*
查看 HTTP 压缩状态	**show http compression settings**
HTTP 压缩的高级配置	**http compression policy useragent** *<user_agent_string> <mime_type>* **http compression advanced useragent on**
设置 url-exclude 压缩规则	**http compression policy urlexclude** *<virtual_service> <url>*

16.3.2 配置示例

1. 启用 HTTP 压缩。

```
Demo(config)#http compression on
```

这个命令启用了 HTTP 压缩的默认配置。在默认配置下，设备会为以下文件类型提供 HTTP 压缩。

- 文本（text/plain）
- HTML（text/HTML）
- XML（text/XML）

2. 检查 HTTP 压缩的运行状态。

```
Demo(config)#show http compression settings
```

3. 配置 HTTP 压缩的高级功能。

如果你想为 Microsoft IE5.5 启用对 Java Script 的压缩，可以使用如下的命令。

```
Demo(config)#http compression policy useragent "MSIE 5.5" JS
```

我们还可以为以下内容配置 HTTP 压缩：

- Java Scripts（application/x-javascript）
- Cascade Style Sheets（text/css, application/x-pointplus）
- PDF 文档（application/pdf）
- PPT 文档（application/powerpoint）
- XLS 文档（application/MSExcel）
- DOC（application/MSWord）

并不是所有的浏览器可以处理这些资源类型的压缩形态。对以上资源类型的支持需要先检查客户浏览器（user agent）是否可以处理这些类型的数据压缩，然后应用压缩功能。为了使设备压缩功能能够为各种不同的浏览器提供更好的服务，管理员可以根据特定的用户代理和资源类型来启用不同的压缩功能。

注意：TEXT、XML 和 HTML 档默认会被压缩，因此它们不必使用“**http compression policy useragent**”命令来配置。

以下命令将为四种浏览器（user-agent）打开 Java Script 和 CSS 文件类型的压缩：IE 6、IE 7、IE 8 以及 Mozilla 5.0。

```
Demo(config)#http compression advanced useragent on
```

4. 设置 url-exclude 压缩规则。

```
Demo(config)#http compression policy urlexclude "v1" "/abc"
```

如果客户端向虚拟服务“v1”发起的请求中的 URL 包含了“/abc”关键词，那么设备即使开启了 HTTP 压缩功能，也不会对该服务器的响应内容进行压缩。

```
Demo(config)#http compression policy urlexclude "v1" "^/def"
```

如果客户端向虚拟服务“v1”发起的请求中的 URL 以“/def”开头，那么设备即使开启了 HTTP 压缩功能，也不会对该服务器的响应内容进行压缩。

```
Demo(config)#http compression policy urlexclude "v1" "/ghi.txt$"
```

如果客户端向虚拟服务“v1”发起的请求中的 URL 以“/ghi.txt”结尾，那么设备即使开启了 HTTP 压缩功能，也不会对该服务器的响应内容进行压缩。

```
Demo(config)#http compression policy urlexclude "v1" "abc*def"
```

如果客户端向虚拟服务“v1”发起的请求中的 URL 包含了“abc*def”关键词，那么设备即使开启了 HTTP 压缩功能，也不会对该服务器的响应内容进行压缩。

第17章 HTTP/HTTPS 路由

17.1 概述

本章将介绍设备的 HTTP/HTTPS 路由功能。

在 HTTP/HTTPS 路由场景中，设备以正向代理的方式转发客户端（前端应用服务器）的 HTTP/HTTPS 请求到指定的后台服务器。为了让设备成为转发 HTTP/HTTPS 请求的正向代理服务器，用户需要在前端应用服务器中手动配置需要访问的设备上的虚拟服务 IP 和端口。当 HTTP/HTTPS 请求命中虚拟服务后，设备根据 HTTP/HTTPS 请求头部携带的目的 IP 地址和端口选择对应的后台服务器（如果后台服务为 HTTPS 类型，会与后台服务器建立 SSL 连接）并转发请求。

设备针对 HTTP/HTTPS 路由功能进行了配置方式优化。管理员只需将虚拟服务与一台特殊的后台服务器（IP 和端口值为 0）进行关联，设备就能将命中该虚拟服务的 HTTP/HTTPS 请求根据头部携带的目的 IP 地址进行转发。除了这台特殊服务器，管理员无需配置其它后台服务器，这能大幅度减少管理员的配置工作量，以应对客户端访问量的弹性变化。

设备 HTTP/HTTPS 路由过程如下图所示。

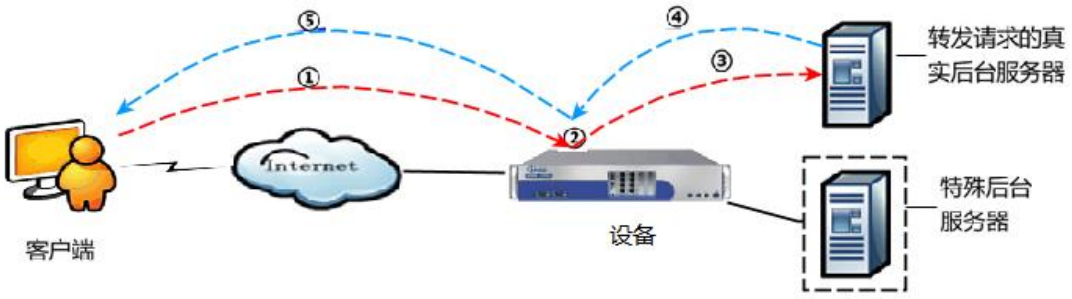

图17–1 设备 HTTP/HTTPS 路由流程

➢ 前置条件

- 客户端（前端应用服务器）已经配置了需要访问的虚拟服务的 IP 地址和端口。
- 设备的虚拟服务已经关联了一台 IP 和端口为 0 的特殊后台服务器。

➢ 路由流程

1. 客户端 HTTP/HTTPS 请求命中设备的虚拟服务。
2. 设备通过 HTTP/HTTPS 头部的 host 字段获取需要访问的后台服务器的 IP 地址和端口。

3. 设备与指定的后台服务器建立 SSL 连接（后台服务为 HTTPS 类型时）并转发请求。
4. 服务器接到请求后，将回应发送给设备。
5. 设备将该回应发送给客户端。

17.2 HTTP/HTTPS 路由配置

17.2.1 配置向导

下面列出了配置 HTTP/HTTPS 路由所需的命令，相关命令的描述信息，请参考命令行使用手册。

表17–1 设备 HTTP/HTTPS 路由配置命令

配置操作	命令行
配置虚拟服务	**slb virtual http** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb virtual https** *<virtual_service> <vip> [vport] [arp_support] [max_connection]*
配置后台服务	**slb real http** *<real_service> <ip|domain> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real https** *<real_service> <ip|domain> [port] [max_connection] [hc_type] [hc_up] [hc_down]*
配置策略	**slb policy static** *<virtual_service> <real_service>*

17.2.2 配置示例

在下面的配置示例中，我们先定义了一个 IP 和端口为 0 的特殊后台服务器。然后又将这个后台服务器以静态方式关联到了一个虚拟服务。

完成配置后，管理员无需再配置其它后台服务，设备就可以根据 HTTP/HTTPS 请求头部携带的目的 IP 和端口进行转发。

```
Demo(config)#slb real http "rs_httproute" 0 0 1000 icmp 3 3
Demo(config)#slb virtual http "vs_httproute " 10.1.61.100 53
Demo(config)#slb policy static "vs_httproute" "rs_httproute"
```

第18章 SSL 加速

18.1 概述

在我们为设备配置好服务器负载均衡和缓存之后，我们还可以配置 SSL（Secure Sockets Layer，安全套接字层）加速，来改善我们客户端通讯的安全性。SSL 加速有两种工作模式，一是对安全数据进行解密，将解密后的信息传递到后台服务器；另一种模式是将安全数据解密后，进行流量管理（服务器负载均衡、缓存等）处理，然后再对这些数据加密，将加密之后的数据传递到具有 SSL 功能的后台服务器。

设备 10.4 系统版本提供全面的 SSL 协议版本的支持，包括 SSLv3、TLSv1.0、TLSv1.1、TLSv1.2 和 TLSv1.3。除此以外，系统还支持国家商用密码局管理局制定的 SM2v1.1 协议。管理员可以根据应用需要选择配置。

18.2 SSL 加速的原理

SSL 最主要的作用是为 Web 流量提供安全访问机制。安全的含义包括保密性、信息完整性和安全认证。SSL 使用密码、数字签名和证书来保证这些元素的安全。

18.2.1 加密算法

SSL 使用加密算法保障重要信息的安全。设备可以使敏感数据在通过公共网络时，能够以较高的安全性到达目的地。数据的加密算法有两种类型：对称加密和非对称加密。

- 对称加密算法中，加密与解密使用相同的密钥，密钥由通信双方约定。密钥的传递可能被第三方截获，在实际应用中，常使用非对称加密算法来将对称密钥加密后再传递。常见的对称加密算法包括 DES、AES 等等。
- 非对称加密也称为公钥加密算法，该算法使用一对密钥，公钥和私钥。其中公钥是公开的，私钥只由一方持有，通过公钥加密的数据只能通过私钥解密。设备支持的非对称加密算法有 RSA、ECC 和 SM2 算法。

➢ **RSA 加密算法**

RSA 是最常用的非对称加密算法，其安全性的提高依赖于密钥长度的增加。随着密钥长度的逐步增加，RSA 加密算法的计算速度逐渐减慢。

➢ **ECC 加密算法**

与 RSA 算法相比，ECC 算法能够以较短的密钥长度实现与 RSA 算法同等的安全性。

➢ **SM2 加密算法**

SM2 加密算法是由中国国家商用密码局发布的基于椭圆曲线的公钥密码算法。自 SM2v1.1 版本开始，SM2 引入了双证书系统，SSL 服务器需具备两个证书：签名证书和加密证书。相应的密钥对为签名密钥对和加密密钥对。

- 签名密钥对和签名证书
- 签名证书用于 SSL 握手过程中的身份验证。

在服务器证书验证环节，服务器会在 Server Certificate 消息中携带这两种 SM2 证书，在 Server Key Exchange 消息中使用自己的签名密钥对里的私钥做数字签名，客户端会使用服务器的签名密钥对里的公钥验证签名，确认服务器的身份。

如果服务器需要验证客户端证书，客户端在 Client Certificate 消息中会携带这两种 SM2 证书，并在随后的 Client Certificate Verify 消息中使用自己的签名密钥对的私钥做数字签名，服务器会使用客户端的签名密钥对里的公钥验证签名，确认客户端的身份。

- 加密密钥对和加密证书

加密证书用于生成预主密钥。

在 ECC 密钥交换方式下，客户端在生成预主密钥后，会使用服务器的加密密钥对的公钥加密预主密钥，然后通过 Client Key Exchange 消息将预主密钥发送给服务器。服务器会用加密密钥对里的私钥进行解密，获取预主密钥的明文。

在 ECDHE 密钥交换方式下，在客户端发送 Client Key Exchange 消息给服务器后，双方会各自生成预主密钥。

18.2.2 数字签名

数字签名通过不可逆的签名算法计算得出。通过验证数字签名，可以判断信息发送方或签名者的身份，以及数据在传输过程中是否发生过篡改。SSL 虚拟主机和后台主机支持在 SSL 握手过程中协商 RSA 签名算法、RSASSA-PSS 签名算法和 ECDSA 签名算法，以及验证 RSA、RSAPSS 和 ECC 证书。

在 ECDHE（Ephemeral Elliptic Curve Diffie-Hellman）方式的密钥交换环节，服务器会通过 Server Key Exchange 消息向客户端传送临时公钥信息，该信息里会附带一个数字签名。客户端会通过验证签名判断公钥是否完整、可靠。

在客户端证书验证环节，客户端会通过 Certificate Verify 消息向服务器发送一个数字签名，目的是让服务器验证客户端身份。服务器会用客户端证书里的公钥来验证签名。

SSL 虚拟主机和后台主机支持的 RSA、RSASSA-PSS 和 ECDSA 签名算法如下表所示。

表18–1 签名算法

类型	SSL 虚拟主机	SSL 后台主机
RSA 签名算法	SHA1RSA SHA224RSA SHA256RSA SHA384RSA SHA512RSA	SHA1RSA SHA224RSA SHA256RSA SHA384RSA SHA512RSA MD5RSA
RSASSA-PSS 签名算法	rsa_pss_rsae_sha256 rsa_pss_rsae_sha384 rsa_pss_rsae_sha512 rsa_pss_pss_sha256 rsa_pss_pss_sha384 rsa_pss_pss_sha512	rsa_pss_rsae_sha256 rsa_pss_rsae_sha384 rsa_pss_rsae_sha512 rsa_pss_pss_sha256 rsa_pss_pss_sha384 rsa_pss_pss_sha512
ECDSA 签名算法	SHA1ECDSA SHA224ECDSA SHA256ECDSA SHA384ECDSA SHA512ECDSA ecdsa_secp256r1_sha256 ecdsa_secp384r1_sha384 ecdsa_secp521r1_sha512	SHA1ECDSA SHA224ECDSA SHA256ECDSA SHA384ECDSA SHA512ECDSA ecdsa_secp256r1_sha256 ecdsa_secp384r1_sha384 ecdsa_secp521r1_sha512

18.2.3 数字证书

证书包含了用于识别用户或设备的信息，这些数字文件可以证明一个公共密钥同个人或企业安全的绑定在一起。数字签名允许核实一个特定公共密钥的细节。证书有助于防止有人用虚假的密钥冒充服务器密钥。SSL 证书使用 X.509 标准来验证身份，X.509 标准证书包含了有关单位的信息，包括公共密钥和名字。一些权威的证书机构来保障证书的有效性。下面列出了 X.509 证书所包含的内容。

表18–2 X.509 证书所包含内容

名称	含义
Version	证书版本号，不同版本的证书格式不同。
Serial Number	证书序列号，具有唯一性，区别于该机构发布的其他证书。如果一个证书被撤销，其序号将被加入 CRL（Certificate Revocation List，证书撤销列表）中。
Issuer	证书发布机构，一般是权威的证书认证中心（Certificate Authority，CA）。
Valid from	证书的有效起始日期

名称	含义
Valid to	证书的有效截止日期
Subject	证书标识的个人或企业
Public key	证书持有者的公钥
Signature algorithm	用于生成数字签名的签名算法。
Thumbprint, Thumbprint algorithm	指纹以及指纹算法，指纹是根据指纹算法计算出的整个证书的哈希值，然后通过 CA 的私钥加密后得到的。

设备支持三种数字证书：RSA（包括 RSAPSS）、ECC 和 SM2 证书。

数字证书由 CA 签发，个人或企业需要通过发送证书签名请求（Certificate Signing Request，CSR）从 CA 获取证书后导入和激活证书。设备支持为虚拟主机生成 RSA、RSASSA-PSS、ECC 和 SM2 类型的 CSR。

➢ **生成 CSR**

- 如果需要申请 RSA 或 RSAPSS 证书，可以使用“**ssl csr**”命令生成 PKCSv1.5 RSA CSR 或 RSASSA-PSS CSR。
- 如果需要申请 ECC 证书，可以使用“**ssl ecc csr**”命令生成 ECC CSR。
- 如果需要申请 SM2 证书，可以使用“**ssl sm2 csr**”命令生成 SM2 CSR。

CA 会通过邮件回复需要申请的证书。获得证书后，需要通过命令导入证书。

➢ **导入证书**

- RSA、RSAPSS 和 ECC 证书通过“**ssl import certificate**”命令导入，RSAPSS 证书仅能用于支持 TLSv1.3 的 SSL 主机。如果为不支持 TLSv1.3 的 SSL 主机导入了 RSAPSS 证书，该证书不能激活，也不能用于 SSL 握手。
- 对于 SM2 证书申请，CA 会签发一个签名证书、一个加密证书和一个加密私钥，加密私钥有明文和数字信封（加密）两种形式。签名证书通过“**ssl sm2 import signcertificate**”命令导入，加密证书通过“**ssl sm2 import enccertificate**”命令导入，明文的加密私钥通过“**ssl sm2 import enckey**”命令导入，数字信封通过“**ssl sm2 import encevp**”命令导入。在导入数字信封前，必须先导入签名私钥。

➢ **激活证书**

RSA、RSAPSS、ECC 和 SM2 证书都通过“**ssl activate certificate** <*host_name*> *[certificate_index] [domain_name] [certificate_type]*”命令激活，在激活证书时，可以通过设置“certificate_type”参数一次激活所有类型的证书或仅激活指定类型的证书，其中 RSA 和 RSAPSS 的证书类型取值都为“rsa”。

每个 SSL 主机支持导入和激活的证书个数如下：

- 如果主机关联了一个或多个域名，那么可以为该主机的每个域各导入三个 RSA 证书（包括 RSAPSS 证书）和三个 ECC 证书，每个域可以各激活一个 RSA 证书或一个 RSAPSS 证书和一个 ECC 证书。
- 如果主机不关联任何域名，每个主机可以导入三个 RSA 证书（包括 RSAPSS 证书）、三个 ECC 证书和三对 SM2 证书，同时激活一个 RSA 证书或一个 RSAPSS 证书、一个 ECC 证书和一对 SM2 证书。

注意：运行 FIPS HSM 的设备不支持 ECC 和 SM2 加密算法。在 FIPS HSM 环境中，虚拟主机关联的每个域各支持激活一个 RSA 证书，每个后台主机只支持激活一个 RSA 证书。

以下步骤为后台主机“rhost”分别导入和激活一个索引为 2 的 RSA 证书和 ECC 证书的示例。

1. 导入 RSA 证书。

```
Demo(config)#ssl import certificate rhost 2
You may overwrite an existing certificate.
Type YES to continue, NO to abort: YES
Enter the certificate file in PEM format,
  use "..." on a single line, without quotes
  to terminate import
-----BEGIN CERTIFICATE-----
MIIDPDCCAqWgAwIBAgIFAK+57hMwDQYJKoZIhvcNAQELBQAwgaoxCzAJBgNVBAYTA
lVTMQswCQYDVQQIEwJDQTERMA8GA1UEBxMITWlscGl0YXMxGzAZBgNVBAoTEkFy
cmF5TmV0d29ya3MgSW5jLjEUMBIGA1UECxMLQVBWIFByb2R1Y3QxHjAcBgNVBAMTF
Xd3dy5hcnJheW5ldHdvcmtzLm5ldDEoMCYGCSqGSIb3DQEJARYZc3VwcG9ydEBhcnJheW5
ldHdvcmtzLm5ldDAeFw0xNjA3MTEwOTMwMzhaFw0yNDA5MjcwOTMwMzhaMIGUMQsw
CQYDVQQGEwJVUzEQMA4GA1UECAwHRmxvcmlkYTEQMA4GA1UEBwwHRmxvcmlkY
TEOMAwGA1UECgwFQXJyYXkxCzAJBgNVBAsMAlBPMQswCQYDVQQLDAJQTzELMA
kGA1UECwwCUE8xDjAMBgNVBAMMBXZob3N0MRowGAYJKoZIhvcNAQkBFgthYmNA
MTI2LmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMlV4+6s/0zY
hoakPUNNwGdh1HCtOyKWpXl7vw6YLxQlKIvlEel69HsfIL1hKV+YmZOuY1istLPqBdxPqLn
iyVIcttX9QpPJJYiRnaU/OjylWLKQt61yLGPrsTzrDKnv2KE9RTpe8MWEwgzy42pmW3Wptl6
Y3Xoox1k521gYVoDSfU+mqzwEFftclx0CD35v8TFbS0AP6js3iikZIddcZiMmtm87oSgQ0Q9Tn
8jvjNYXpLJ/BcEnGHFsAP1S5MFC8Wj7jNJ5wHNSPPpdePDnQ49j+E7Ah9XFY502svxbgUjEt
1zwu2CAVt4jvKTEU3tbWE+tGIjJpFtkFa4f3urUXvkCAwEAATANBgkqhkiG9w0BAQsFAAOB
gQB/CD9eCsXPllVqmCasL0XpeOF/pAgon6pgC4BVf1quZEb7C78IATQWrQJfHdYSZVQwXw
sbg6YLHnpI+SdpoNwjsrd25K6AvCmBMANtqFighIwjcbNJqINGWWxZUIquHVmyD+L/53x2
Nih4qJ5zjMfWzPbR7KbwF0cDJs1/YfMEwg==
-----END CERTIFICATE-----
```

```
...
PEM format.
Certificate import successful !!!
```

2. 导入 ECC 证书。

```
Demo(config)#ssl import certificate rhost 2
You may overwrite an existing certificate.
Type YES to continue, NO to abort: YES
Enter the certificate file in PEM format,
  use "..." on a single line, without quotes
  to terminate import
-----BEGIN CERTIFICATE-----
MIICIzCCAcqgAwIBAgIFAI88UzkwCgYIKoZIzj0EAwIwgaoxCzAJBgNVBAYTAlVTMQswC
QYDVQQIDAJDQTERMA8GA1UEBwwITWlscGl0YXMxGzAZBgNVBAoMEkFycmF5TmV
0d29ya3MgSW5jLjEUMBIGA1UECwwLQVBWIFByb2R1Y3QxHjAcBgNVBAMMFXd3dy5h
cnJheW5ldHdvcmtzLm5ldDEoMCYGCSqGSIb3DQEJARYZc3VwcG9ydEBhcnJheW5ldHdvcm
tzLm5ldDAeFw0xNjA3MTEwOTMxMjdaFw0yNDA5MjcwOTMxMjdaMIGHMQswCQYDVQ
QGEwJDTjELMAkGA1UECAwCQ04xCzAJBgNVBAcMAkNOMQswCQYDVQQKDAJDTjE
LMAkGA1UECwwCQ04xCzAJBgNVBAsMAkNOMQswCQYDVQQLDAJDTjEOMAwGA1U
EAwwFdmhvc3QxGjAYBgkqhkiG9w0BCQEWC2FiY0AxNjIuY29tMFkwEwYHKoZIzj0CAQY
IKoZIzj0DAQcDQgAECXCDImdSY1/eq400+rReCE5qLfL9VeIHygJR8lAOzFTG58stV9kBjKR
BTBL5p5tdZqFX1DbxZ0bT7+mCuBvS7jAKBggqhkjOPQQDAgNHADBEAiAfDVKFeeeyEq9
HvOmXEGEueaYDCMoVg1zm2T396BOBVQIgKZUTOqn+Kb0Nh64b9mS0Fr8mtTqps5Fl7Q/
v2YO4MqQ=
-----END CERTIFICATE-----
...
PEM format.
Certificate import successful !!!
```

3. 激活导入的 RSA 和 ECC 证书。

```
Demo(config)#ssl activate certificate rhost 2 "" all
Do you want to activate all Certificates #2? [YES/(NO)]: YES

Warning: RSA certificate chain is incomplete for rhost. Please add interca or rootca certificate.
RSA Certificate #2 is activated successfully!

Warning: ECC certificate chain is incomplete for rhost. Please add interca or rootca certificate.
ECC Certificate #2 is activated successfully!
```

18.2.3.1 客户端证书认证

客户端证书认证是指客户端依据服务器的要求（Client Certificate Request）向服务器发送自己的证书，用于证明客户端的身份。设备支持通过证书分析器（本公司专利技术）来快速验证 X.509 证书。

- 当设备作为代理客户端（SSL 后台主机）时，启用客户端认证的 SSL 后台主机会在收到后台服务发来的证书请求消息时传送证书。在该场景，如果 SSL 后台主机既存在激活的 RSA 证书，也存在激活的 ECC 证书，会按照以下原则选择发送的证书类型：
  - 如果证书请求消息里指定的证书类型包含 RSA 类型，则发送 RSA 证书。
  - 如果证书请求消息里只指定了 ECC 证书，且之前协商的密码套件为“ECDHE…”类型，则发送 ECC 证书。
  - 如果证书请求消息里只指定了 ECC 证书，但之前协商的密码套件不是“ECDHE…”类型，则重置连接。
- 当设备作为代理服务器（SSL 虚拟主机）时，启用客户端认证的 SSL 虚拟主机会向客户端发送证书请求消息，要求客户端提供证书进行认证。

18.2.4 服务器名字指示（Server Name Indication，SNI）

SNI 为 SSL 服务器上的每一个站点提供 SSL 认证。通过将多个域名与一个 SSL 虚拟主机关联而不是为每个站点配置一个 IP 地址并配置一个关联的虚拟主机，SNI 功能可以为同一个 SSL 服务器上的多个站点提供 SSL 认证，简化了配置并减少了 IP 地址的消耗。

配置示例

➢ **前置条件：**

- SSL 服务器上有两个站点 www.a.com 和 www.b.com。
- 已经为 SSL 虚拟主机导入密钥和对应的证书，且证书已被启动。

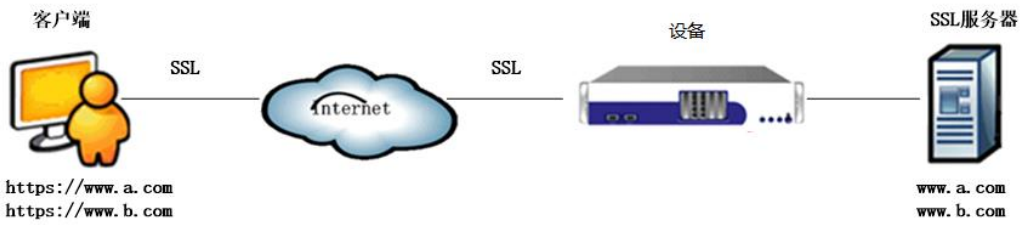

图18–1 SNI 工作原理

➢ **CLI 配置示例**

1. 配置 SLB。

添加 SLB 后台服务“rhost1”和“rhost2”，分别关联服务组“vhg1”和“vhg2”。添加虚拟服务“vshost1”。为了根据域名选择命中的后台服务，为服务组“vhg1”和“vhg2”配置 QoS 域名策略。

```
Demo(config)#slb real http "rhost1" 172.16.78.22 80 65535 tcp 3 3
Demo(config)#slb real http "rhost2" 172.16.78.23 80 65535 tcp 3 3
Demo(config)#slb group method "vhg1" rr
Demo(config)#slb group method "vhg2" rr
Demo(config)#slb group member "vhg1" "rhost1" 1 0
Demo(config)#slb group member "vhg2" "rhost2" 1 0
Demo(config)#slb virtual https "vshost1" 192.168.12.62 443 arp 0
Demo(config)#slb policy qos hostname "spa" "vshost1" "vhg1" "www.a.com" 100
Demo(config)#slb policy qos hostname "spb" "vshost1" "vhg2" "www.b.com" 200
Demo(config)#slb policy default "vshost1" "vhg1"
```

2. 创建 SSL 虚拟主机“vshost”，并关联虚拟服务“vshost1”。

```
Demo(config)#ssl host virtual vshost vshost1
```

3. 将域名 “www.a.com”和 “www.b.com”关联到 SSL 虚拟主机“vshost”。

```
Demo(config)#ssl sni vshost www.a.com
Demo(config)#ssl sni vshost www.b.com
```

4. 为域名“www.a.com”导入密钥。

关于如何获得和密钥和证书及导入密钥和证书的其他方式，请参见 18.3.2 配置示例。

```
Demo(config)#ssl import key vshost 1 www.a.com
You may overwrite an existing key file. This may require you
to purchase a new certificate. Type YES to continue: YES
Enter key, use "..." on a single line, without quotes
to terminate import
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,7B26776D2C14EB1F

WwU2hyREMjGi+6k5nq5gecnTSJSTk8q3ori+1jooOfgvRV72x/e3hGRFIysKb15J
1xeyUHFjtNJIHlEVjNg2XURXIRDO+qatI0cnvW1sYbuQodat0jGzpP/oOxGVcRsg
kTuP8xObpUou0RevvZVbfYHWsdWZnY5qhLjje1UGKUu8HwaMp3Q7OJ7pper7TOAXCkiVn
1zHgeFkB4p2lGwCHxbdNJg7ee5U2HFiSoC/8P8G64kIkEfxuMFwzkaVlia+Anf40WlBcvixZaN
ZqHkZdcfJGlqKnqnpcJWiyLd8oPr3npYu6+c1PyHiyEua67nEzK5G4KVFacW0POPLHfj/Q1yM
qYcWySyecJvtq0jxYLujGH6qckLhMMltdptCfFvZ6gfncRanNWEU+2v+nc509XKv9zJeR58Ws
```

```
Ah81AN7aChuEBj69cjL4HwOtk3nttVmpr5+cluF6mpbk2rJuZ14l+Hwc63XRAaHXV0bIQfo+rQ
BIWHkItvN4S2a0G23IC9N32Y1qTqQgoj9qxSXvw1oKmV+UcVjtt78GyT3m7pYPtNc/wdehUP
rR4Lc8hSiV3m29ZGIoNpdhVgCGw/MM65tr3dRZfYcG8cPJ4pk/3WKq9OfznlugAxo4KWebea5
1CaT4YBMO5nETHT9WesX9viiKtbhPaAIKQOd4Rwcs5FK0ETHSfBBpW0tsXw562s6QnNSm
nWuoksvqqXqCbnuNSz611z3dOhiLwHdqCqLEDNEhv231ielOk1qoUo+ad0eiM0ETrP6Zyw+j6i
9K71sSe28Zm4gVQMnSHxGyp0K+zK3cYs=
-----END RSA PRIVATE KEY-----
...
PEM format.
Enter passphrase for the private key:

Key import successful !!!
```

5. 为域名“www.a.com”导入对应的证书。

```
Demo(config)#ssl import certificate vshost 1 www.a.com
You may overwrite an existing certificate.
Type YES to continue, NO to abort: YES
Enter the certificate file in PEM format,
  use "..." on a single line, without quotes
  to terminate import
-----BEGIN CERTIFICATE-----
MIIDPDCCAqWgAwIBAgIFAK+57hMwDQYJKoZIhvcNAQELBQAwgaoxCzAJBgNVBAYTA
lVTMQswCQYDVQQIEwJDQTERMA8GA1UEBxMITWlscGl0YXMxGzAZBgNVBAoTEkFy
cmF5TmV0d29ya3MgSW5jLjEUMBIGA1UECxMLQVBWIFByb2R1Y3QxHjAcBgNVBAMTF
Xd3dy5hcnJheW5ldHdvcmtzLm5ldDEoMCYGCSqGSIb3DQEJARYZc3VwcG9ydEBhcnJheW5
ldHdvcmtzLm5ldDAeFw0xNjA3MTEwOTMwMzhaFw0yNDA5MjcwOTMwMzhaMIGUMQsw
CQYDVQQGEwJVUzEQMA4GA1UECAwHRmxvcmlkYTEQMA4GA1UEBwwHRmxvcmlkY
TEOMAwGA1UECgwFQXJyYXkxCzAJBgNVBAsMAlBPMQswCQYDVQQLDAJQTzELMA
kGA1UECwwCUE8xDjAMBgNVBAMMBXZob3N0MRowGAYJKoZIhvcNAQkBFgthYmNA
MTI2LmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMlV4+6s/0zY
hoakPUNNwGdh1HCtOyKWpXl7vw6YLxQlKIvlEel69HsfIL1hKV+YmZOuY1istLPqBdxPqLn
iyVIcttX9QpPJJYiRnaU/OjylWLKQt61yLGPrsTzrDKnv2KE9RTpe8MWEwgzy42pmW3Wptl6
Y3Xoox1k521gYVoDSfU+mqzwEFftclx0CD35v8TFbS0AP6js3iikZIddcZiMmtm87oSgQ0Q9Tn
8jvjNYXpLJ/BcEnGHFsAP1S5MFC8Wj7jNJ5wHNSPPpdePDnQ49j+E7Ah9XFY502svxbgUjEt
1zwu2CAVt4jvKTEU3tbWE+tGIjJpFtkFa4f3urUXvkCAwEAATANBgkqhkiG9w0BAQsFAAOB
gQB/CD9eCsXPllVqmCasL0XpeOF/pAgon6pgC4BVf1quZEb7C78IATQWrQJfHdYSZVQwXw
sbg6YLHnpI+SdpoNwjsrd25K6AvCmBMANtqFighIwjcbNJqINGWWxZUIquHVmyD+L/53x2
Nih4qJ5zjMfWzPbR7KbwF0cDJs1/YfMEwg==
-----END CERTIFICATE-----
...
PEM format.
Certificate import successful !!!
```

6. 为域名“www.b.com”导入密钥。

```
Demo(config)#ssl import key vshost 1 www.b.com
You may overwrite an existing key file. This may require you
to purchase a new certificate. Type YES to continue: YES
Enter key, use "..." on a single line, without quotes
to terminate import
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,7B26776D2C14EB1F

WwU2hyREMjGi+6k5nq5gecnTSJSTk8q3ori+1jooOfgvRV72x/e3hGRFIysKb15J
1xeyUHFjtNJIHlEVjNg2XURXIRDO+qatI0cnvW1sYbuQodat0jGzpP/oOxGVcRsg
kTuP8xObpUou0RevvZVbfYHWsdWZnY5qhLjje1UGKUu8HwaMp3Q7OJ7pper7TOAXCkiVn
1zHgeFkB4p2lGwCHxbdNJg7ee5U2HFiSoC/8P8G64kIkEfxuMFwzkaVlia+Anf40WlBcvixZaN
ZqHkZdcfJGlqKnqnpcJWiyLd8oPr3npYu6+c1PyHiyEua67nEzK5G4KVFacW0POPLHfj/Q1yM
qYcWySyecJvtq0jxYLujGH6qckLhMMltdptCfFvZ6gfncRanNWEU+2v+nc509XKv9zJeR58Ws
Ah81AN7aChuEBj69cjL4HwOtk3nttVmpr5+cluF6mpbk2rJuZ14l+Hwc63XRAaHXV0bIQfo+rQ
BIWHkItvN4S2a0G23IC9N32Y1qTqQgoj9qxSXvw1oKmV+UcVjtt78GyT3m7pYPtNc/wdehUP
rR4Lc8hSiV3m29ZGIoNpdhVgCGw/MM65tr3dRZfYcG8cPJ4pk/3WKq9OfznlugAxo4KWebea5
1CaT4YBMO5nETHT9WesX9viiKtbhPaAIKQOd4Rwcs5FK0ETHSfBBpW0tsXw562s6QnNSm
nWuoksvqqXqCbnuNSz611z3dOhiLwHdqCqLEDNEhv231ielOk1qoUo+ad0eiM0ETrP6Zyw+j6i
9K71sSe28Zm4gVQMnSHxGyp0K+zK3cYs=
-----END RSA PRIVATE KEY-----
...
PEM format.
Enter passphrase for the private key:

Key import successful !!!
```

7. 为域名“www.b.com”导入对应的证书。

```
Demo(config)#ssl import certificate vshost 1 www.b.com
You may overwrite an existing certificate.
Type YES to continue, NO to abort: YES
Enter the certificate file in PEM format,
  use "..." on a single line, without quotes
  to terminate import
-----BEGIN CERTIFICATE-----
MIICIzCCAcqgAwIBAgIFAI88UzkwCgYIKoZIzj0EAwIwgaoxCzAJBgNVBAYTAlVTMQswC
QYDVQQIDAJDQTERMA8GA1UEBwwITWlscGl0YXMxGzAZBgNVBAoMEkFycmF5TmV
0d29ya3MgSW5jLjEUMBIGA1UECwwLQVBWIFByb2R1Y3QxHjAcBgNVBAMMFXd3dy5h
cnJheW5ldHdvcmtzLm5ldDEoMCYGCSqGSIb3DQEJARYZc3VwcG9ydEBhcnJheW5ldHdvcm
tzLm5ldDAeFw0xNjA3MTEwOTMxMjdaFw0yNDA5MjcwOTMxMjdaMIGHMQswCQYDVQ
```

```
QGEwJDTjELMAkGA1UECAwCQ04xCzAJBgNVBAcMAkNOMQswCQYDVQQKDAJDTjE
LMAkGA1UECwwCQ04xCzAJBgNVBAsMAkNOMQswCQYDVQQLDAJDTjEOMAwGA1U
EAwwFdmhvc3QxGjAYBgkqhkiG9w0BCQEWC2FiY0AxNjIuY29tMFkwEwYHKoZIzj0CAQY
IKoZIzj0DAQcDQgAECXCDImdSY1/eq400+rReCE5qLfL9VeIHygJR8lAOzFTG58stV9kBjKR
BTBL5p5tdZqFX1DbxZ0bT7+mCuBvS7jAKBggqhkjOPQQDAgNHADBEAiAfDVKFeeeyEq9
HvOmXEGEueaYDCMoVg1zm2T396BOBVQIgKZUTOqn+Kb0Nh64b9mS0Fr8mtTqps5Fl7Q/
v2YO4MqQ=
-----END CERTIFICATE-----
...
PEM format.
Certificate import successful !!!
```

8. 为域名“www.a.com”和“www.b.com”启动证书。

```
Demo(config)#ssl activate certificate vshost 1 www.a.com
Demo(config)#ssl activate certificate vshost 1 www.b.com
```

9. 启动 SSL 虚拟主机“vshost”。

```
Demo(config)#ssl start vshost
```

18.2.5 HTTP/2 支持

当 SSL 主机配置满足指定的限制条件时，SSL 加速功能支持运行 HTTP/2 于 TLS 上。

SSL 虚拟主机必须同时满足以下条件才支持运行 HTTP/2，否则会使用 HTTP/1.1：

1. 虚拟主机关联的 SLB 虚拟服务启用了 HTTP/2。
2. 虚拟主机支持 TLSv1.2 版本。
3. 虚拟主机支持密码套件 ECDHE-RSA-AES128-GCM-SHA256、
   ECDHE-RSA-AES256-GCM-SHA384、
   ECDHE-ECDSA-AES128-GCM-SHA256、
   ECDHE-ECDSA-AES256-GCM-SHA384 中的至少一种。

SSL 后台主机必须同时满足以下条件才支持运行 HTTP/2，否则会使用 HTTP/1.1：

1. 后台主机关联的 SLB 后台服务支持并且启用了 HTTP/2。
2. 虚拟主机支持运行 HTTP/2（参见上述关于虚拟主机支持 HTTP/2 需满足的三个条件）。
3. 后台主机支持 TLSv1.2 版本。
4. 后台主机支持密码套件 ECDHE-RSA-AES128-GCM-SHA256、
   ECDHE-RSA-AES256-GCM-SHA384、

ECDHE-ECDSA-AES128-GCM-SHA256、
ECDHE-ECDSA-AES256-GCM-SHA384 中的至少一种。

注意：在 HTTP/2 运行于 TLS 之上的部署环境里，必须禁用 SSL 重协商。通过“**ssl globals renegotiation off**”命令可以全局禁用 SSL 重协商，通过“**ssl settings reneg** *<virtual_host_name>*”命令可以对单个虚拟主机禁用 SSL 重协商。

18.2.6 SSL 通道

SSL 通道能够为所有基于 TCP 协议的后台系统服务提供通用的加密通道。当流量的目的 IP 和目的端口匹配出向 SSL 通道的 IP 和端口，或者流量的源 IP 和源端口匹配入向 SSL 通道的 IP 和端口时，流量可以通过 SSL 通道。

创建 SSL 通道后，系统将基于流量方向自动生成相关 SLB 配置和 SSL 配置。具体如下：

- 流量为入向：自动生成 TCP 类型的后台服务和 TCPS 类型的虚拟服务并采用静态策略关联，自动生成 SSL 虚拟主机并且自动与虚拟服务关联。
- 流量为出向：自动生成 TCPS 类型的后台服务和 TCP 类型的虚拟服务并采用静态策略关联，自动生成 SSL 后台主机并且自动与后台服务关联。

18.3 SSL 加速配置

18.3.1 配置向导

对本章中所用到的术语解释如下：

表18–3 设备 SSL 加速配置的相关术语解释

术语名称	解释
虚拟主机	一个 SSL 虚拟主机与服务器负载均衡虚拟服务相关联。在浏览器和设备之间进行 SSL 通信时，代表一个 SSL 服务。
后台主机	一个 SSL 后台主机与服务器负载均衡后台服务相关联。在设备与后台服务器之间进行 SSL 通信时，代表一个 SSL 客户端。
源服务器	可以接受明文或加密请求的后台服务器。
明文	没有加密过的信息。
虚拟主机端口	SSL 虚拟主机的监听端口，默认为 443。
密钥（私有）	保存在设备上用于 PKI（公钥基础设施）认证的私有密钥。设备支持的密钥长度最大为 2048 位。
证书	用于认证，及帮助在浏览器与服务器之间建立安全通信。
认证中心	根据 CSR（证书签发请求）信息创建证书的机构。
受信认证中心	目前的网页浏览器中都保存有一系列知名认证中心的公钥，用于对证书

	的真实性进行校验。如果浏览器无法识别认证中心，将会提示用户。设备采取类似的方式，也维护了一份信任的认证中心列表。

在我们的例子中，使用“www.example.com”作为 SSL 虚拟主机，它与一个 IP 地址为 10.10.0.10、端口号为 443 的 SLB 虚拟服务关联。

SSL virtual Host: www.example.com

SLB virtual Host IP: 10.10.0.10

SLB virtual Host Port: 443

配置 SSL 加速的方法有两种：

- 从未配置过 SSL：先配置 SSL 虚拟主机，产生一个 CSR（证书签发请求）发送到所选择的认证中心，认证中心会发回一个签发后的证书，将该证书导入。
- 已经具有密钥和证书：可以跳过 CSR 的步骤，直接导入密钥和证书。

下面列出了配置 SSL 加速器所需的命令，相关命令的描述信息，请参考命令行使用手册。

表18–4 设备 SSL 加速器配置命令

配置操作	命令行
建立一个 SSL 虚拟主机	**slb virtual https** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb virtual tcps** *<virtual_service> <vip> <vport> [arp_support] [max_connection]* **ssl host {real|virtual}** *<host_name> [slb_service]*
为 SSL 虚拟主机导入证书和密钥	**ssl csr** *<virtual_host_name> [key_length] [certificate_index] [signature_algorithm_index] [domain_name]* **ssl ecc csr** *<virtual_host_name> [curve_name] [certificate_index] [signature_algorithm_index] [domain_name]* **ssl import certificate** *<host_name> [certificate_index] [domain_name] [tftp_ip|url] [file_name|password]* **ssl import key** *<host_name> [key_index] [domain_name] [tftp_ip|url] [file_name|password]* **ssl activate certificate** *<host_name> [certificate_index] [domain_name] [certificate_type]*
建立一个 SSL 后台主机	**slb real https** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real tcps** *<real_service> <ip> <port> [max_connection] [hc_type] [hc_up] [hc_down]* **ssl host {real|virtual}** *<host_name> [slb_service]*
SSL 虚拟主机的高级配置	**ssl stop** *<host_name>* **ssl settings ciphersuite** *<host_name> <cipher_string>*

配置操作	命令行
	ssl settings protocol *<host_name> <version>* **ssl settings reuse** *<host_name>* **ssl settings clientauth** *<host_name>* **ssl settings certfilter** *<virtual_host_name> <condition_1> [condition_2]* **ssl import rootca** *[host_name] [domain_name] [tftp_ip|url] [file_name]* **ssl settings crl offline** *<host_name> <cdp_name> <cdp_url> [domain_name] [time_interval] [delay_time]* **ssl settings crl online** *<virtual_host_name>* **ssl settings ocsp** *<virtual_host_name> <ocsp_server_url>* **ssl settings minimum** *<virtual_host_name> <cipher_strength> <redirect_url>* **ssl start** *<host_name>* **ssl import error** *<error_code> <url> [virtual_host_name]* **ssl load error** *<error_code> [virtual_host_name]*
SSL 后台主机的高级配置	**ssl settings ciphersuite** *<host_name> <cipher_string>* **ssl settings protocol** *<host_name> <version>* **ssl settings reuse** *<host_name>* **ssl settings clientauth** *<host_name>* **ssl import rootca** *[host_name] [domain_name] [tftp_ip|url] [file_name]* **ssl settings servername** *<real_host_name> <common_name>*

18.3.2 配置示例

- **创建 SSL 虚拟主机**

首先，使用“**slb virtual…**”命令建立一个 SLB TCPS 或 HTTPS 类型的虚拟服务，然后使用“**ssl host**”命令创建一个 SSL 虚拟主机，并与虚拟服务关联。

```
Demo(config)#slb virtual https virtual1https 10.10.0.10 443
Demo(config)#ssl host virtual www.example.com virtual1https
```

在上面的例子中，“virtual1https”是新创建的 SLB 虚拟服务，“www.example.com”是新创建的 SSL 虚拟主机名称。

- **为新建的 SSL 虚拟主机导入私钥和证书**

以下步骤以为虚拟主机“www.example.com”申请、导入和激活 RSA 私钥和证书为例。

1. 执行“**ssl csr**”命令为“www.example.com”生成一个 RSA CSR（如果需要申请 ECC 证书，执行“**ssl ecc csr**”命令，如果需要申请 SM2 证书，执行“**ssl sm2 csr**”命令）。

```
Demo(config)#ssl csr www.example.com
Generating keys for "www.example.com"....please wait
```

```
We will now gather some required information about your SSL virtual host.
This information will be encoded into your certificate.
TWO-character country code for your organization (eg. US) :US
State or province []:CA
location or local city []:San Jose
Organization Name :Example.com
Organizational Unit :Example.com
Organizational Unit []:
Organizational Unit []:
Do you want to use the virtual host name "www.example.com"
as the Common Name? (recommended) [Y/N]: Y
email address of administrator []:array@example.com
Do you want to add Subject Alternative Names? (recommended) [Y/N] N
-----BEGIN CERTIFICATE REQUEST-----
MIIC6jCCAdICAQAwgZUxCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTERMA8GA1U
EBwwIU2FuIEpvc2UxFDASBgNVBAoMC0V4YW1wbGUuY29tMRQwEgYDVQQLDAtFeGF
tcGxlLmNvbTEYMBYGA1UEAwwPd3d3LmV4YW1wbGUuY29tMSAwHgYJKoZIhvcNAQkB
FhFhZG1pbkBleGFtcGxlLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggE
BAKfPzHgGQA2DKh7kkzSKczUO9RkMRvrMX+MssKiVwGUpwUZ3B0YlW5gqUQ0ieYqNQ
bYr4s7G+d+zHe7NtsyUADMJwgDKK4pQaiBuxVWzQtqQqIGEZc4NqIaltJzOpzZSMqY0SvbQ
3MJqrvVgZpdObeedW5SRVjn8zyebr2j/re4tTIJpm+lj9FiFf/yVHsJdXjJrOONYfsAcaI9c8a5BLqe
PavZEvzh3p+eiCwjGflv8n48O8ub+PIGU1W230gkfRdG5D6e1yWlXFdceveunuOlfuoL2yBgHu
yxxgu+RkQd6ZqV6eACLbv47OTMr9MUGuW6TuQ0TI+T24lY8DBn/TxcCAwEAAaAPMA0G
CSqGSIb3DQEJDjEAMA0GCSqGSIb3DQEBCwUAA4IBAQCRl4Mao7hBqsqH/+kU8IQK7aq
wdujSDj5KxO5rKkSutslaqfsIbpr85nGKFqxrBxpy0lFs6NegztSV0dCc/Dt3iVaAqLEgeVmdFA9Z
bcpwHecQmeg1D200GmpsU3T2xiqM0mDc7jmRywWenCJkRWmO3EWeO9N5mbbeoOUs4Kel
KvVayMe2k9YvArSmOa3NHzyTQ1Zhqc80Q6Jg7mSw6B9et0JpKIim+3Hw12ULOdhIDijLOa8
GiDdhuL4J5FBDW0wY8Jl+YKeW7r8GDldENP1bdvWDdDkI0zHhVuwPDOuAcwWj23gT7jLo
wcNYRNIVW5RGrXHjlfb9UXKqMboJmpp+
-----END CERTIFICATE REQUEST-----

Do you want the private key to be exportable [Yes/(No)]:Yes
Enter passphrase for the private key:

Confirm passphrase for the private key:

Warning: RSA certificate chain is incomplete for www.example.com. Please add interca or rootca
certificate.
```

除了生成 RSA CSR 外，该命令同时也为 SSL 虚拟主机“www.example.com”创建了一个 RSA 密钥对和测试证书。测试证书只能用于测试目的，如果希望利用该测试证书进行测试或演示，可以直接启用该 SSL 虚拟主机：


```
Demo(config)#ssl start www.example.com
```

此时，管理员可以使用网页浏览器安全地连接到该网站。

2. 转发证书签发请求到认证中心。

从“**ssl csr**”命令的输出中，复制从“-----BEGIN CERTIFICATE REQUEST-----”到“-----END CERTIFICATE REQUEST-----”的内容，转发给 CA，CA 会回复一封包含数字证书的电子邮件。以下显示的是一个证书样例。

```
-----BEGIN CERTIFICATE-----
MIICnjCANgcANgEUMA0GCSqGSIb3DQEBBAUAMIG5MQswCQYDVQQGEwJVUzETMB
EGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2UxHDAaBgNVBAoTE0
NsaWNrQXJyYXkgTmV0d29ya3MxFDASBgNVBAsTC0RldmVsb3BtZW50MSMwIQYDVQ
QDExpkZXZlbG9wbWVudC5jbGlja2FycmF5LmNvbTEpMCcGCSqGSIb3DQEJARYaZGV2Z
WxvcG1lbnRAY2xpY2thcnJheS5jb20wHhcNMDIwMjEzMTgwMTI5WhcNMDMwMjA4MTgw
MTI5WjB0MQswCQYDVQQGEwJVUzEMMAoGA1UECBMDRE9EMQwwCgYDVQQHEw
NET08xCzAJBgNVBAoTAkRPMQswCQYDVQQLEwJETzETMBEGA1UEAxMKMTAuMTIu
MC4xNDEaMBgGCSqGSIb3DQEJARYLbWhAZGtkay5jb20wgZ8wDQYJKoZIhvcNAQEBBQ
ADgY0AMIGJAoGBAMx4r+ae4kTZggtyU047OsKUyqCt+V1MHgTPTpVxdtxYhSTSOZwYIX
gRqBEdJvs2/ua1XZRzLOCTa58VI/8I3derAPqz79WpBRsxD25rCT1rzmalfkTea3V8jHJYP6Yin
DTWKFKztxeUclkzukzPUZO6M0fI5ToXNuLEe+IwvOkfAgMBAAEwDQYJKoZIhvcNAQEEB
QADgYEAodV5O0LKUr/O0BbxOnwmyP/DkLj4bpe9XxQO6B4psDey/+xBHs6tgGKuy8spbcJ4
pQc+5KLydK1ZYcTkbxJq41K4RHM11OClXVjm3xRhqKQnjzNboExIvkZsKIBbfLkBrM1eBnE
aiYWXmsYGfxPkwdhKlQCLQgN+G3IKu2cRQLU=
-----END CERTIFICATE-----
```

注意：进行 SSL 配置时请务必谨慎操作。在导入从认证中心获得的证书之前，请一定不要删除 SSL 虚拟主机。如果清除了 SSL 信息，则只能再次发送证书签发请求到认证中心以获得另一个证书。不过，大多数认证中心允许有 30 天的试用期以便于获得另一个证书，超过了这个期限，就必须购买另一个证书。

3. 使用“**ssl import certificate**”命令为 SSL 虚拟主机导入收到的证书。

```
Demo(config)#ssl import certificate www.example.com 1
You may overwrite an existing certificate.
Type YES to continue, NO to abort: YES
Enter the certificate file in PEM format,
 use "..." on a single line, without quotes
 to terminate import
-----BEGIN CERTIFICATE-----
MIICnjCANgcANgEUMA0GCSqGSIb3DQEBBAUAMIG5MQswCQYDVQQGEwJVUzETMB
EGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2UxHDAaBgNVBAoTE0
NsaWNrQXJyYXkgTmV0d29ya3MxFDASBgNVBAsTC0RldmVsb3BtZW50MSMwIQYDVQ
QDExpkZXZlbG9wbWVudC5jbGlja2FycmF5LmNvbTEpMCcGCSqGSIb3DQEJARYaZGV2Z
```

```
WxvcG1lbnRAY2xpY2thcnJheS5jb20wHhcNMDIwMjEzMTgwMTI5WhcNMDMwMjA4MTgw
MTI5WjB0MQswCQYDVQQGEwJVUzEMMAoGA1UECBMDRE9EMQwwCgYDVQQHEw
NET08xCzAJBgNVBAoTAkRPMQswCQYDVQQLEwJETzETMBEGA1UEAxMKMTAuMTIu
MC4xNDEaMBgGCSqGSIb3DQEJARYLbWhAZGtkay5jb20wgZ8wDQYJKoZIhvcNAQEBBQ
ADgY0AMIGJAoGBAMx4r+ae4kTZggtyU047OsKUyqCt+V1MHgTPTpVxdtxYhSTSOZwYIX
gRqBEdJvs2/ua1XZRzLOCTa58VI/8I3derAPqz79WpBRsxD25rCT1rzmalfkTea3V8jHJYP6Yin
DTWKFKztxeUclkzukzPUZO6M0fI5ToXNuLEe+IwvOkfAgMBAAEwDQYJKoZIhvcNAQEEB
QADgYEAodV5O0LKUr/O0BbxOnwmyP/DkLj4bpe9XxQO6B4psDey/+xBHs6tgGKuy8spbcJ4
pQc+5KLydK1ZYcTkbxJq41K4RHM11OClXVjm3xRhqKQnjzNboExIvkZsKIBbfLkBrM1eBnE
aiYWXmsYGfxPkwdhKlQCLQgN+G3IKu2cRQLU=
-----END CERTIFICATE-----
```

也可以从远程 TFTP 服务器导入证书。使用这种方法时，需要指定 TFTP 服务器的 IP 地址和证书文件的名称：

```
Demo(config)#ssl import certificate www.example.com 1 10.10.13.82 example.crt
You may overwrite an existing certificate.
Type YES to continue, NO to abort: YES
```

注意：如果已经存在可以使用的密钥和证书文件（无需通过 CSR 申请证书），可以使用“**ssl import key**”命令和“**ssl import certificate**”命令为 SSL 虚拟主机直接导入密钥和证书，导入时既可以使用手动粘贴文件内容的方法，也可以通过远程 TFTP、FTP、SFTP、HTTP、HTTPS 和 SCP 服务器导入，但有两点需要注意：

- 必须先导入密钥然后再导入证书。
- 非 PEM 格式的密钥和证书，必须通过 TFTP、FTP、SFTP、HTTP、HTTPS 和 SCP 方式导入，不能使用手动粘贴文件内容的方法。

4. 使用“**ssl activate certificate**”命令激活证书。

```
Demo(config)#ssl activate certificate www.example.com 1
```

注意：在激活证书时，系统会检查证书链。如果根证书和中级证书在全局受信 CA 文件或中级 CA 文件中无法找到，系统将会打印一条警告信息提示证书链不完整。可以分别使用命令“**ssl import rootca**”和“**ssl import interca**”来导入根证书和中级证书。

5. 启用 SSL 虚拟主机。

```
Demo(config)#ssl start www.example.com
```

至此已经完成了对“www.example.com”的 SSL 加速配置，客户端可以通过 HTTPS 正常访问。

- **从 IIS 和 NS IPlanet 网页服务器中导入证书和密钥**

IIS

如果使用的是 Microsoft IIS 服务器，设备允许通过 TFTP 机制从 IIS4/5 中导入证书。IIS 在同一个.PFX 格式的档中存储 SSL 密钥和证书。先将该文件放到 TFTP 服务器的根目录下，以<vhostname>.crt 命名，然后使用“**ssl import certificate**”命令将档导入到设备中。本命令需要输入 TFTP 服务器的 IP 地址做为参数。

```
Demo(config)#ssl import certificate www.example.com 1 10.10.0.3
```

该命令将下载名为<vhostname>.crt 的档，在我们的例子中是从 TFTP 服务器 10.10.0.3 上下载 www.example.com.crt。

成功导入证书后，可以使用“**ssl activate certificate**”命令启动这个证书。

```
SJ-Box1(config)#ssl activate certificate www.example.com 1
```

从 TFTP 服务器上成功导入密钥和证书后，可以使用“**ssl start**”启动 SSL 子系统。

```
Demo(config)#ssl start www.example.com
Netscape/iPlanet：
```

如果使用的是 Netscape 或者 iPlanet 服务器，也可以从这些服务器上导入证书和密钥。iPlanet 服务器在/<serverroot>/alias/目录下保存密钥/证书对，<serverroot>是服务器的安装目录，在这个目录下有两个文件，<serverid-hostname>-key3.db 和<serverid-hostname>-cert7.db，需要将它们复制到 TFTP 服务器的根目录下，第一个文档以<vhostname>.key 命名，第二个文档以<vhostname>.crt 命名。请一定正确命名，否则 SSL 子系统不能正确下载它们。

现在导入证书和密钥。

```
SJ-Box1(config)#ssl import certificate www.example.com 1 10.10.0.3 www.example.com.crt
```

从 10.10.0.3 上导入证书文件“www.example.com.crt”。

```
SJ-Box1(config)#ssl import key www.example.com 1 10.10.0.3 www.example.com.key
```

从 10.10.0.3 上导入密钥文件“www.example.com.key”。

注意：在从 iPlanet 导入证书/密钥对时，必须首先导入证书，再导入密钥。

成功导入证书后，可以使用“**ssl activate certificate**”命令启动这个证书。

```
SJ-Box1(config)#ssl activate certificate www.example.com 1
```

然后启动 SSL 子系统。

```
Demo(config)#ssl start www.example.com
```

注意：在本节中我们创建了一个 SLB 虚拟服务并为它配置了 SSL。需要将该 SLB 虚拟服务关联到一个或多个 SLB 后台服务，SLB 模块就可以处理发送到 SLB 虚拟服务的请求。请参考“第 11 章 服务器负载均衡（SLB）”章节，获取“如何将后台服务关联到虚拟服务”的相关介绍。

- **建立一个 SSL 后台主机**

设备允许使用 SSL 子系统与启用了 SSL 功能的后台服务器进行通信，设备与后台服务器之间的通信将被加密。

SSL 后台主机的配置比较简单：

1. 使用“**slb real**”命令创建服务器负载均衡后台服务。
2. 使用“**ssl host**”命令定义 SSL 后台主机。

例如：

```
Demo(config)#slb real https real1https 192.168.1.20 443 tcps
Demo(config)#ssl host real www.myreal.com real1https
```

在上面的例子中，“real1https”是新创建的 SLB 后台服务，它代表一个后台服务器，IP 地址 192.168.1.20，端口号为 443，并能够处理 SSL 请求。最后一个步骤，启动 SSL 子系统：

```
Demo(config)#ssl start www.myreal.com
```

现在，设备可以与后台服务器进行 SSL 通信。

注意：在本节中我们创建了服务器负载均衡后台服务并为它配置了 SSL。需要将它关联到一个 SLB 虚拟服务上，服务器负载均衡模块就可以将流量发送到这个启用了 SSL 功能的后台服务上。请参考“第 11 章 服务器负载均衡（SLB）”章节，获取“如何将后台服务关联到虚拟服务”的相关介绍。

18.3.2.1 SSL 虚拟主机的高级配置

1. 禁用 SSL 虚拟主机。

```
Demo(config)#ssl stop www.example.com
```

如果更改 SSL 虚拟主机的配置，必须先禁用该主机。

2. 为 SSL 虚拟主机配置密码套件。

```
Demo(config)#ssl settings ciphersuite "www.example.com" "DES-CBC3-SHA"
```

下表列出了不同协议版本的 SSL 虚拟主机对 RSA、ECC（ECDHE-ECDSA..）和 SM2 密码套件的支持情况。“Y”表示支持，“N”表示不支持。

表18–5 不同协议版本的虚拟主机对密码套件的支持情况

密码套件	位数	SSL 协议 – 虚拟主机					
		SSLv3.0	TLSv1	TLSv1.1	TLSv1.2	TLSv1.3	SM2v1.1
RC4-MD5	128	Y	Y	Y	Y	N	N
RC4-SHA	128	Y	Y	Y	Y	N	N
DES-CBC-SHA	64	Y	Y	Y	N	N	N
DES-CBC3-SHA	192	Y	Y	Y	Y	N	N
EXP-RC4-MD5	40	Y	N	N	N	N	N
EXP-DES-CBC-SHA	40	Y	N	N	N	N	N
AES128-SHA	128	Y	Y	Y	Y	N	N
AES256-SHA	256	Y	Y	Y	Y	N	N
AES128-SHA256	128	N	N	N	Y	N	N
AES256-SHA256	256	N	N	N	Y	N	N
AES128-GCM-SHA256	128	N	N	N	Y	N	N
AES256-GCM-SHA384	256	N	N	N	Y	N	N
ECDHE-RSA-AES128-SHA	128	Y	Y	Y	Y	N	N
ECDHE-RSA-AES256-SHA	256	Y	Y	Y	Y	N	N
ECDHE-ECDSA-AES128-SHA	128	Y	Y	Y	Y	N	N
ECDHE-ECDSA-AES256-SHA	256	Y	Y	Y	Y	N	N
ECDHE-RSA-AES128-SHA256	128	N	N	N	Y	N	N
ECDHE-RSA-AES256-SHA384	256	N	N	N	Y	N	N
ECDHE-RSA-AES128-GCM-SHA256	128	N	N	N	Y	N	N
ECDHE-RSA-AES256-GCM-SHA384	256	N	N	N	Y	N	N
ECDHE-ECDSA-AES128-SHA256	128	N	N	N	Y	N	N

密码套件	位数	SSL 协议 – 虚拟主机					
		SSLv3.0	TLSv1	TLSv1.1	TLSv1.2	TLSv1.3	SM2v1.1
ECDHE-ECDSA-AES256-SHA384	256	N	N	N	Y	N	N
ECDHE-ECDSA-AES128-GCM-SHA256	128	N	N	N	Y	N	N
ECDHE-ECDSA-AES256-GCM-SHA384	256	N	N	N	Y	N	N
TLS-AES128-GCM-SHA256	128	N	N	N	N	Y	N
TLS-AES256-GCM-SHA384	256	N	N	N	N	Y	N
ECDHE-SM4-SM3	128	N	N	N	N	N	Y
ECC-SM4-SM3	128	N	N	N	N	N	Y
ECDHE-SM4-GCM-SM3	128	N	N	N	N	N	Y
ECC-SM4-GCM-SM3	128	N	N	N	N	N	Y

如果要为一个 SSL 虚拟主机配置多个密码套件，需要在各密码套件之间以冒号“:”分隔。

3. 为 SSL 虚拟主机配置协议版本。

```
Demo(config)#ssl settings protocol "www.example.com" "SSLv3:TLSv1:TLSv12"
```

可以设置为 SSLv3、TLSv1、TLSv11、TLSv12、TLSv13 或 SM2v11，或者同时设置其中的几个或者全部。

注意：参数值“TLSv11”代表 TLSv1.1 协议，“TLSv12”代表 TLSv1.2 协议，“TLSv13”代表 TLSv1.3 协议，SM2v11 代表 SM2v1.1 协议。

4. 为 SSL 虚拟主机配置会话复用功能。

```
Demo(config)#ssl settings reuse "www.example.com"
```

该功能默认是启用的，可以使用命令“**no ssl settings reuse**”关闭该功能。

5. 为 SSL 虚拟主机配置客户端认证。

设备支持基于 SSL 的客户端认证。如果启用该功能，在客户端可以连接到 SSL 虚拟主机之前，设备要求每个客户端提供 SSL 证书。

此外，SSL 虚拟主机还可以根据配置的证书过滤规则（使用“**ssl settings certfilter**”命令）检查客户端证书。如果客户端证书没有通过证书校验，SSL 主机将拒绝客户端访问。

```
Demo(config)#ssl settings clientauth "www.example.com"
```

注意：如果为 SSL 虚拟主机启用了 SSL 客户端认证功能，则必须提供一个受信 CA 证书，使用它对客户端证书进行校验。

```
Demo(config)#ssl import rootca "www.example.com"
```

该命令将提示用户粘贴一个 PEM 格式的受信 CA 证书。可以为 SSL 虚拟主机配置多个受信 CA。

一个 SSL 虚拟主机最多支持三条 certfilter 配置，配置之间为逻辑“或”关系。如果客户端证书匹配不到任何一条 certfilter 配置，SSL 虚拟主机将拒绝客户端访问。

例如：

```
Demo(config)#ssl settings certfilter vhost
"subject:/C=CN/O=Enterprise/OU=QA/emailAddress=admin@abc.com"
"issuer:/C=CN"
```

在本例中，只有当客户端证书同时满足以下两个条件，才能通过校验：

- “subject”字段中，“C”为“CN”，“O”为“Enterprise”，“OU”为“QA”，“emailAddress”为 admin@abc.com。
- “issuer”字段中，“C”为“CN”。

否则，客户端证书不能通过证书校验。

设备支持两种客户端认证方式：强制的和非强制的。客户端认证模式默认是强制的。在非强制模式下，当服务器发送一个证书请求到客户端时，如果客户没有匹配的证书，或者点“Cancel”取消时，服务器将会允许客户连接到限制性的网络资源，而不是放弃这个 SSL 连接。所有的网络资源如果想要公开给非认证模式的客户端，则需要执行命令“**http acl url**”。

6. 为 SSL 虚拟主机配置 CRL。

CRL（即证书撤销列表），可以通过 HTTP、FTP 或 LDAP 定期从 CRL 分发点获取 CRL 文件，CRL 分发点最多可以配置 10 条。

示例：在一个 HTTP 网页服务器上保存有 CRL 文件（list.crl），希望每隔 1 分钟获取该文件。配置命令如下：


```
Demo(config)#ssl settings crl offline www.example.com cdp1 "http://www.crldp.com/list.crl"
1
```

这样设备每隔 1 分钟会从 www.crldp.com 上下载 CRL 文件 list.crl。

也可以从 FTP 网站上下载 CRL 文件：

```
Demo(config)#ssl settings crl offline www.example.com cdp1 "ftp://ftp.crldp.com/list.crl" 1
```

也支持从 LDAP 网站上下载 CRL 文件：

```
Demo(config)#ssl settings crl offline www.example.com cdp1
"ldap://ldap.crldp.com/cn=bjhy,dc=enterprise,dc=com" 1
```

7. 为 SSL 虚拟主机配置 OCSP 在线检测证书有效性。

设备支持 OCSP（在线证书状态协议）。可以在一个 OCSP 服务器上线上校验证书。

示例：配置一个 OCSP 服务器（ocsp.crldp.com:8888）在线校验证书：

```
Demo(config)#ssl settings ocsp www.example.com "http:// ocsp.crldp.com:8888"
```

注意：OCSP 具有最高的优先级。本命令配置后，将仅通过该 OCSP 服务器对证书校验。

8. 为不具有强加密支持的客户端配置复位向。

设备提供复位向弱客户端（没有使用强加密的客户端）到另一个 URL 的功能。指定可以接受的最弱强度，任何使用比此更弱的加密算法的客户端将被复位向到另一个 URL。

比如，希望将密钥长度小于 168 位的客户端复位向到另一个不同的网站 www.example2.com。

配置命令如下：

```
Demo(config)#ssl settings minimum www.example.com 168 "http://www.example2.com"
```

9. 启用并查看 SSL 配置。

启动 SSL 虚拟主机：

```
Demo(config)#ssl start www.example.com
```

查看当前 SSL 设置：

```
Demo(config)#show ssl settings www.example.com
```

18.3.2.2 SSL 后台主机的高级配置

1. 禁用 SSL 后台主机。

```
Demo(config)#ssl stop www.myreal.com
```

如果更改 SSL 后台主机的配置，必须先禁用该主机。

2. 为 SSL 后台主机配置密码套件。

```
Demo(config)#ssl settings ciphersuite "www.myreal.com" "DES-CBC3-SHA"
```

下表列出不同协议版本的 SSL 后台主机对 RSA 密码套件、ECC 密码套件（ECDHE-ECDSA…）和 SM2 密码套件的支持情况。“Y”表示支持，“N”表示不支持。

表18–6 不同协议版本的后台主机对密码套件的支持情况

密码套件	位数	SSL 协议 – 后台主机					
		SSLv3.0	TLSv1.0	TLSv1.1	TLSv1.2	TLSv1.3	SM2v1.1
RC4-MD5	128	Y	Y	Y	Y	N	N
RC4-SHA	128	Y	Y	Y	Y	N	N
DES-CBC3-SHA	192	Y	Y	Y	Y	N	N
AES128-SHA	128	Y	Y	Y	Y	N	N
AES256-SHA	256	Y	Y	Y	Y	N	N
AES128-SHA256	128	N	N	N	Y	N	N
AES256-SHA256	256	N	N	N	Y	N	N
AES128-GCM-SHA256	128	N	N	N	Y	N	N
AES256-GCM-SHA384	256	N	N	N	Y	N	N
ECDHE-RSA-AES128-SHA	128	Y	Y	Y	Y	N	N
ECDHE-RSA-AES256-SHA	256	Y	Y	Y	Y	N	N
ECDHE-ECDSA-AES128-SHA	128	Y	Y	Y	Y	N	N
ECDHE-ECDSA-AES256-SHA	256	Y	Y	Y	Y	N	N
ECDHE-RSA-	128	N	N	N	Y	N	N

密码套件	位数	SSL 协议 – 后台主机					
		SSLv3.0	TLSv1.0	TLSv1.1	TLSv1.2	TLSv1.3	SM2v1.1
AES128-SHA256							
ECDHE-RSA-AES256-SHA384	256	N	N	N	Y	N	N
ECDHE-RSA-AES128-GCM-SHA256	128	N	N	N	Y	N	N
ECDHE-RSA-AES256-GCM-SHA384	256	N	N	N	Y	N	N
ECDHE-ECDSA-AES128-SHA256	128	N	N	N	Y	N	N
ECDHE-ECDSA-AES256-SHA384	256	N	N	N	Y	N	N
ECDHE-ECDSA-AES128-GCM-SHA256	128	N	N	N	Y	N	N
ECDHE-ECDSA-AES256-GCM-SHA384	256	N	N	N	Y	N	N
TLS-AES128-GCM-SHA256	128	N	N	N	N	Y	N
TLS-AES256-GCM-SHA384	256	N	N	N	N	Y	N
ECC-SM4-SM3	128	N	N	N	N	N	Y
ECDHE-SM4-SM3	128	N	N	N	N	N	Y
ECDHE-SM4-GCM-SM3	128	N	N	N	N	N	Y
ECC-SM4-GCM-SM3	128	N	N	N	N	N	Y

3. 为 SSL 后台主机配置协议版本。

```
Demo(config)#ssl settings protocol "www.myreal.com" "SSLv3:TLSv1"
```

可以设置为 SSLv3、TLSv1、TLSv11、TLSv12、TLSv13 或 SM2v11，或者同时设置其中的两个或者全部。

4. 为 SSL 后台主机配置会话重用。

允许在设备和后台服务器之间重用 SSL 会话，该特性默认是启用的。

```
Demo(config)#ssl settings reuse www.myreal.com
```

5. 为 SSL 后台主机配置客户端认证。

在与后台服务器通信时可以使用 SSL 客户端认证功能。如果启用了该设置，在 SSL 握手过程中设备会将客户端证书提交到后台服务器。

```
Demo(config)#ssl settings clientauth www.myreal.com
```

注意：如果希望为 SSL 后台主机启用客户端认证功能，需要使用“**ssl import certificate**”和“**ssl import key**”命令导入一个证书和密钥，证书将被发送到后台服务器用于认证。这两条命令都适用于 SSL 虚拟主机和 SSL 后台主机。这两条命令的使用方法，请参考前面描述的 SSL 虚拟主机配置。

6. 配置检查后台服务器证书的通用名字。

如果希望校验后台服务器证书，需要打开对服务器证书校验的全局配置，并可以使用“**ssl settings servername**”命令配置一个名字，与后台服务器证书的通用名字相匹配。

举例来说，与后台主机 www.myreal.com 关联的后台服务器证书的通用名字是“Myreal Inc.”，可以使用如下命令配置：

```
Demo(config)#ssl settings servername www.myreal.com "Myreal Inc."
```

7. 为 SSL 后台主机导入受信 CA 证书。

SSL 子系统相当于是后台服务器的一个客户端，像常见的网页浏览器一样有几个根 CA 证书。如果在后台服务器上使用的是一个自签发的证书或一个由本地 CA 颁发的证书，就需要使用“**ssl import rootca**”命令导入该证书。

该证书必须是 PEM 格式的。设备会提示复制并粘贴证书内容，以“…”结束。

```
Demo(config)#ssl import rootca
```

8. 应用并查看配置。

使用“**ssl start**”命令启用 SSL 后台主机：

```
Demo(config)#ssl start www.myreal.com
```

使用“**show ssl settings**”命令查看当前的 SSL 配置：

```
Demo(config)#show ssl settings www.myreal.com
```

9. 删除一个已配置的 SSL 后台主机：

```
Demo(config)#clear ssl www.myreal.com
```

第19章 SSL 侦听

通过 SSL 侦听功能，设备可以对加密的 SSL 流量进行拦截和解密后发送到防火墙、IPS（Intrusion Prevention System，入侵防御系统）和 IDS（Intrusion Detection System，入侵检测系统）等安全设备进行过滤。当解密后的数据经过安全设备检查后，会被发送到设备重新加密后再发送给真正的应用服务器。这可以防止攻击者通过加密方式在报文里携带可能导致攻击、入侵和数据渗漏等安全风险的信息。设备支持与工作在二层和三层的安全设备协作实现 SSL 侦听功能，其基本工作流程如下：

1. 入口节点接收和解密来自客户端的 SSL 流量，将明文数据发送到安全设备。
2. 安全设备对明文数据进行检查，过滤掉包含安全风险的数据，将允许通过的数据发送到出口节点。
3. 出口节点将数据转发到 SSL 流量的最终目的地。

根据入口节点和出口节点的分布方式，在实际部署中，SSL 侦听可以采用集成式和分布式两种分布方式：

- 在集成式方式下，入口节点和出口节点集成在一台设备上。
- 在分布式方式下，SSL 侦听由两台设备完成，一台设备做为入口节点，另一台设备作为出口节点。

SSL 侦听通过模拟服务器证书实现。管理员需在入口节点上安装客户端信任的证书来获取和解密来自客户端的 SSL 流量。证书的具体配置需考虑实际的网络是否应用了 PKI 系统：

- 对于没有应用 PKI 系统的网络，需在入口节点上生成 CA 证书和密钥（通过“**ssli cacert…**”命令），并为客户端浏览器导入生成的 CA 证书。当入口节点获取到服务器的证书后，会通过生成的 CA 证书对服务器证书重新签名，生成一个模拟证书，通过模拟证书里的公钥对应的私钥来解密来自客户端的 SSL 流量。
- 如果网络上应用了 PKI 系统，需为入口节点和客户端导入 CA 中级证书和密钥，该中级 CA 证书必须由受客户端信任的 CA 签发。当入口节点获取到服务器的证书后，会通过导入 CA 中级证书对服务器证书重新签名，生成一个模拟证书，通过模拟证书里的公钥对应的私钥来解密来自客户端的 SSL 流量。

模拟证书会被缓存到系统中并设置计时（通过“**ssli settings certcache timeout**”命令配置），到期后将被移出缓存。如果在缓存到期前模拟证书被再次击中，将重新计算缓存时间。

19.1 域名列表

当为一个 SSL 虚拟主机 SSL 侦听功能启用时，系统默认侦听该虚拟主机的所有 SSL 流量。但是，为了保护隐私，系统不应对某些 SSL 流量进行加解密，例如与涉及隐私信息的银行和医疗等网站的通信。

为了保护隐私，系统支持域名列表功能。域名列表功能可以通过“**ssli domainlist {on|off}**”命令开启和禁用。在禁用状态下，系统默认对所有 SSL 流量进行侦听。在启用状态下，域名列表功能通过两种类型的域名列表定义需要和无需侦听的 SSL 流量：

- 旁路域名列表：访问该域名列表内域名的 SSL 流量可以无需安全设备检查以密文形式通过。
- 侦听域名列表：访问该域名列表内域名的 SSL 流量需要经过解密和安全设备检查才可以通过。

应用了域名列表的虚拟主机将按照域名列表类型执行相应的旁路或侦听操作，每个虚拟主机只能应用一种类型的域名列表。例如：

1. 创建旁路域名列表“list_1”。

```
Demo(config)#ssli domainlist list list_1 bypass
```

2. 添加域名字符串项目“abc*”和“cde*”到“list_1”。

```
Demo(config)#ssli domainlist item list_1 “abc*”
Demo(config)#ssli domainlist item list_1 “cde*”
```

3. 应用“list_1”到虚拟主机“vhost”。

```
Demo(config)#ssli domainlist apply list vhost list_1
```

为了方便配置，域名列表功能还支持通过“**ssli domainlist apply item**”命令为虚拟主机单独应用一个旁路或侦听类型的域名字符串。同样，每个虚拟主机只能应用一种类型的域名字符串。例如，如果只需要侦听访问域名“xyz*”的 SSL 流量，可以执行以下命令将该域名作为侦听域名字符串应用于虚拟主机，无需将该域名添加到侦听域名列表后再将域名列表应用于虚拟主机。

```
Demo(config)#ssli domainlist apply item vhost2 "xyz*" intercept
```

注意：当入口节点获取服务器证书时，如果服务器请求客户端认证或 SSL 握手失败，SSL 流量也会被允许以密文形式通过。

19.2 URL 过滤

如“16.1 域名列表”一节所述，SSL 侦听模块支持手动添加侦听类型的域名列表来定义需要侦听的 SSL 流量。管理员也可以通过添加旁路类型的域名列表来定义需要透明传输的 SSL 流量。该方法适用于客户端访问的网站类型较少，不需要大量人工配置的应用场景。

在访问站点较多的应用场景，通过域名列表功能进行流量过滤需要管理员手动配置大量的域名列表。另外，管理员无法全面的了解所有的网站，某些网站的访问可能被不必要的侦听或旁路。在这种情况下，建议使用网站分类功能来实现智能的 URL 过滤功能。设备通过该功能支持识别 82 种网站类别，关于这些网站类别的详细说明，请参考该链接：
http://www.brightcloud.com/tools/change-request-url-categorization.php

网站分类功能的默认设置为对所有虚拟主机禁用，管理员可以通过“**ssli webclassify {on|off}**”命令控制该功能在每个虚拟主机上的启用状态。

19.2.1 过滤策略

通过网站分类功能，管理员无需为每个需要侦听或旁路的网站访问定义流量过滤策略。只需要定义哪一类网站需要侦听或旁路，例如可以对访问金融服务和健康服务的网站的流量定义旁路策略来保护用户隐私。

与域名列表功能相似，网站分类功能支持定义以下任意一种过滤策略进行区别化的 SSL 处理：

- 侦听策略：所有访问该类别内网站的流量都必须解密并发给安全设备检查，访问未定义类型网站的流量将被透明转发。管理员可以使用“**ssli webclassify url intercept**”命令定义侦听类型的网站类别。
- 旁路策略：所有访问该类别内网站的流量都可以透明传输，无需解密和安全设备检查，所有访问未定义类型网站的流量将被侦听。管理员可以使用“**ssli webclassify url bypass**”命令定义旁路类型的网站类别。

在同一个虚拟主机上，侦听类型和旁路类型的网站类别不能同时定义，即每个虚拟主机上只能定义一种类型的网站类别。如果虚拟主机上同时配置了域名列表和网站分类功能，对于收到的请求或响应时的处理原则如下：

- 虚拟主机会优先匹配域名列表功能里配置的域名列表。
  - 如果成功匹配到域名列表，则按照域名列表的控制类型（侦听或旁路）采取相应的处理方式。
  - 如果域名列表中未找到匹配项，会通过网站分类功能查找网站类别。
- 通过网站分类功能查找网络类别时，虚拟主机会先查询本地缓存和数据库。

- 如果通过本地缓存或数据库确定了网站类别，则按照网络类别过滤策略的控制类型（侦听或旁路）采取相应的处理方式。
- 如果本地缓存或数据库中没有网址的类别信息，系统会连接到 Webroot 服务器，在线请求查询网址的类别，并将查询到的网址类别保存到缓存中。
- 如果 Webroot 服务器也无法识别网站类别，系统默认的处理方式为侦听。该默认行为可以使用“**ssli webclassify defaction**”命令修改。

19.2.2 使用许可证

如需使用网站分类功能实现智能的 URL 过滤，请联系公司客户支持并提供设备型号和序列号信息来获取使用许可证。该功能支持两种类型的使用许可证：

- 免费试用许可证：30 天免费试用。
- 正式许可证：有效期为 365 天。

使用许可证到期后，网站分类功能将不可用，Webroot 服务器会拒绝设备的访问，设备也会主动停止向 Webroot 服务器发送网站类别查询请求。

19.2.3 配置示例

1. 导入网站分类功能许可证。

```
Demo(config)#webclassify license
41ce070c-4baa4482-02bc5237-f62e21b8-b50b177c-00000000-00000001-20171220-20180119
```

2. 启用全局网站分类功能。

```
Demo(config)#webclassify on
```

3. （可选）启用在线网站分类查询功能。该功能要求入口节点可以访问外网。

```
Demo(config)#webclassify cloud on
```

4. 为 SSL 侦听模块配置对访问无法识别网站的流量进行旁路。

```
Demo(config)#ssli webclassify defaction vhost 0
```

5. 设置医药和金融服务站点为旁路类型的网站类别。

```
Demo(config)#ssli web url bypass vhost "Health & Medicine"
Demo(config)#ssli web url bypass vhost "Financial Services"
```

6. 为 SSL 侦听虚拟主机启用网站分类功能。

```
Demo(config)#ssli webclassify on vhost
```

19.3 安全设备负载均衡

在与安全设备协作实现 SSL 侦听的过程中，设备还可以为安全设备提供负载均衡功能，从而将流量均衡地分配到多个安全设备上。安全设备负载均衡除了可以用于 SSL 侦听（正向代理拓扑），还支持反向代理拓扑环境。

在反向代理拓扑环境中，设备按照既有的 SSL 加速功能处理 SSL 流量。虚拟主机上配置有后台服务的证书和密钥，SSL 流量会在入口节点上解密后，按照负载均衡算法分配到安全设备上，然后通过出口节点发送到服务器。

在正向代理拓扑环境中，管理员需在入口节点上配置 CA 证书。入口节点会生成模拟证书来获取和解密 SSL 流量后，将流量负载均衡到安全设备。

如需了解 SSL 侦听和安全设备负载均衡在以上场景里的详细配置步骤，请联系公司客户支持获取 SSL 侦听配置指导。

第20章 External DNS 云原生的方式与容器平台对接

操作步骤：

1、在 Kuberentes 集群中部署相关资源对象

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: external-dns
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: external-dns
rules:
- apiGroups: [""]
  resources: ["services","endpoints","pods","namespaces","Secret"]
  verbs: ["get","watch","list"]
- apiGroups: ["extensions","networking.k8s.io"]
  resources: ["ingresses"]
  verbs: ["get","watch","list"]
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["list"]
---
apiVersion: v1
kind: Secret
metadata:
  name: array-credentials
  namespace: default
```

```
type: Opaque
data:
   #base64 编码
   url: aHR0cHM6Ly8xPC4yMTAuMTQuMTc0Ojk5OTc=
   username: cmVzdHFwaQ==
   password: QXJyYXPjbGljazE=
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
   name: external-dns-owner
roleRef:
   apiGroup: rbac.authorization.k8s.io
   kind: ClusterRole
   name: external-dns
subjects:
- kind: ServiceAccount
   name: external-dns
   namespace: default
---

apiVersion: apps/v1
kind: Deployment
metadata:
   name: infosec-external-dns
spec:
   strategy:
      type: Recreate
   selector:
      matchLabels:
```

```
        app: infosec-external-dns
    template:
      metadata:
        labels:
          app: infosec-external-dns
      spec:
        containers:
        - args:
          - --source=service
          - --source=ingress
          - --domain-filter=external-dns-example.com # 将使 ExternalDNS 仅看到与提供的域匹配的域，忽略不匹配的域
          - --provider=infosec
          - --policy=sync # 设置“upsert-only”将阻止 ExternalDNS 删除任何记录
          - --apv-addresses=$(ARRAY_URL)
          - --apv-username=$(ARRAY_USERNAME)
          - --apv-password=$(ARRAY_PASSWORD)
          image: "x.x.x.x:5000/external-dns:v1.1.0"
          imagePullPolicy: IfNotPresent
          env:
            # Get sensitive values from the credentials secret
            - name: ARRAY_USERNAME
              valueFrom:
                secretKeyRef:
                  name: array-credentials
                  key: username
            - name: ARRAY_PASSWORD
              valueFrom:
                secretKeyRef:
```

```
                name: array-credentials
                key: password
          - name: ARRAY_URL
            valueFrom:
              secretKeyRef:
                name: array-credentials
                key: url
        name: external-dns
        volumeMounts:
        - mountPath: /etc/kubernetes
          name: config-file
      dnsPolicy: ClusterFirst
      serviceAccount: external-dns
      serviceAccountName: external-dns
      volumes:
        - name: config-file
          hostPath:
            path: /etc/kubernetes/
```

2、创建 nginx 服务进行验证

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  annotations:
    external-dns.alpha.kubernetes.io/hostname: nginx.external-dns-example.com # 公网域名地址
    external-dns.alpha.kubernetes.io/internal-hostname: nginx-internal.external-dns-example.com # 内网域名地址
    external-dns.alpha.kubernetes.io/ttl: "600"
```

```
spec:
  type: LoadBalancer
  ports:
  - port: 80
    name: http
    targetPort: 80
  selector:
    app: nginx
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
        ports:
        - containerPort: 80
          name: http
```

服务创建后 kubectl get svc 查看

CLUSTER-IP 对应域名 nginx.external-dns-example.com

EXTERNAL-IP 对应域名 nginx-internal.external-dns-example.com

3、DNS 设备上会自动生成正确的 DNS 配置，提供 DNS 服务

第21章 RPZ 防火墙配置手册

21.1 使用场景

1.类似本地记录的查询 支持 A 和 AAAA 记录

2.支持对请求进行拦截和修改。

3.支持对应答进行拦截和修改。

21.2 配置方法

配置 RPZ zone

为 RPZ zone 配置 RPZ 记录

RPZ 记录参数如下:

RPZ 触发器

取值可以为以下几种:

Client IP Address，以.rpz-client-ip 结尾。用于匹配发起请求的客户端的 IP 地址，双栈均可支持。定义 IPv4 地址段的客户端时，如 B1.B2.B3.B4/prefix，在 RPZ 策略规则的书写规则中应当写成 prefix.B4.B3.B2.B1；定义 IPv6 地址段的客户端时，如 B1:B2:B3:B4:B5:B6:B7:B8/prefix，在 RPZ 策略规则的书写格式中应当写成 prefix.B8.B7.B6.B5.B4.B3.B2.B1。如同 IPv6 地址可以使用双冒号缩写，此处的 IPv6 地址可以相应的使用 zz 替代。如 2001::6:180/128 可写成 128.180.6.zz.2001。

如:24.0.49.101.180.rpz-client-ip，当客户端网段为 180.101.49.0/24 时，可以触发 RPZ 规则

QDNAME，正常域名，可带通配符。用于匹配发起请求包或应答包中请求域名字段的域名。

如:*.array.com，当请求或响应中包含 array.com 时，可以触发 RPZ 规则

Response IP Address，以.rpz-ip 结尾。用于匹配应答记录中的的 IP 地址，双栈均可支持。IP 地址段的编写格式与 Client IP Address 中描述的一致。

如:24.0.49.101.180.rpz-ip，当响应中包含 180.101.49.0/24 网段时，可以触发 RPZ 规则

NSDNAME，以.rpz-nsdname 结尾。用于匹配应答记录中 NS 的名字，可以在回答部分，授权部分。

如:ns.example.com.rpz-nsdname，当响应中包含 ns.example.com 的 NS 记录时，可以触发 RPZ 规则

NSDNAME，以.rpz-nsdname 结尾。用于匹配应答记录中 NS 的名字，可以在回答部分，授权部分。

如: 24.0.49.101.180.rpz-nsdname，当响应中包含 NS 记录对应的 IP 地址为 180.101.49.0/24 网段时，可以触发 RPZ 规则

RPZ 动作

NXDOMAIN, 动作为回复 NXDOMAIN 类型的应答。

NODATA，动作为回复 NODATA 类型（rcode 为 noerror 但是 answer 个数为 0）的应答。

PASSTHRU，动作为透传，即 RPZ 白名单，不走 RPZ 策略规则。

DROP，动作为丢弃，即不做应答。

TCP-ACTION，动作为将应答中的 TC 标志置 1，强制用户发起 TCP 的 DNS 请求，用于攻击防护。

LOCAL DATA，动作为回复指定预先配置好的解析结果。

第22章 安全应用访问（SAA）

为了保护特定的网络资源，很多网站都会采用认证方式来限制终端用户的访问，只有通过认证的用户才可以使用某些服务。同时，在大型网站里，终端用户的一次操作可能涉及到多个子系统的协作，如果每个子系统都要求用户重复认证，不仅会增加系统认证逻辑的复杂性，而且会降低用户体验，这就需要认证方式同时支持单点登录机制。

SAA（Secure Application Access，安全应用访问）功能提供基于 SAML、LDAP、RADIUS 和 OAuth 的 AAA 认证和授权方式和 Web SSO（Single Sign-On，单点登录认证），可以满足多种应用场景的认证需求。

22.1 AAA

AAA 是为网络上所有连接和交易提供认证、授权和计费功能一系列特性和操作的组合，通过 AAA 提供的安全架构可以控制哪些用户可以访问哪些服务，以及用户使用的资源数量，管理员可以精确控制用户访问的内容，授权特定用户访问内部资源，并实现服务收费。认证功能可以使设备通过第三方认证服务器验证用户凭据，只有通过认证的用户才可以访问受保护的服务器。授权功能可以使设备确认每个用户可以访问服务器上的哪些内容。计费功能可以使设备对服务器上每个用户的活动进行记录。

设备目前支持 AAA 特性的认证和授权功能。

22.1.1 AAA 服务器

设备使用 AAA 服务器进行终端用户的认证和授权。“AAA 服务器”是任何可以通过 AAA 实现认证或授权功能的实体。设备支持以下类型的 AAA 服务器：

- LDAP（Lightweight Directory Access Protocol，轻量级目录访问协议）
- RADIUS（Remote Authentication Dial In User Service，远程用户拨号认证系统）
- SAML SP（Security Assertion Markup Language，安全断言标记语言）
- OAuth（Open Authorization，开放授权）

22.1.1.1 SAML

SAML 定义了一种基于 XML 的框架，在该框架内，各个实体之间创建并交换认证、授权和属性信息。在设备上，SAML 认证基于 SAML 2.0（Security Assertion Markup Language，安全断言标记语言）标准实现。在 SAML 框架下主要有主体（Subject）、IdP（Identity Provider，身份提供者）和 SP（Service Provider，服务提供者）三个实体。

注意：IdP 和 SP 的时区和日期、时间设置必须相同（至少在分钟级别）。

22.1.1.1.1 元数据

元数据文件定义了实体之间如何共享必要的配置信息。在开始使用 SAML 进行认证前，SP 和 IdP 需分别导入对方的元数据文件。当元数据文件有更新时，管理员必须为实体导入新的元数据文件。元数据文件主要包含以下信息：

- 实体的标识 ID
- 实体的证书，用于签名和加密

- SSO、SLO（Single Logout，单点注销）、ACS（Assertion Consumer Service，断言消费服务）、ARS（Artifact Resolution Service，组件解析服务）等服务的 URL 地址和协议绑定类型。协议绑定类型定义了 SAML 消息在 SP 和 IdP 之间传输时使用的传输层协议。在设备上，SLO 服务支持 HTTP Redirect 和 HTTP POST 协议绑定类型，ACS 服务支持 HTTP POST 和 HTTP Artifact 协议绑定类型。
  - HTTP Redirect：利用 HTTP 重定向消息（即 302 响应）传输 SAML 消息。
  - HTTP POST：利用 Base64 编码的 HTML 内容传输 SAML 消息。
  - HTTP Artifact：利用 HTML 内容或 URL 查询字符串传输 SAML 消息的索引。
- 联系人、属性要求等其他信息

22.1.1.1.2 SSO 流程

在基于 SAML 的 SSO 认证方式中，设备作为 SP 服务器，利用 IdP 服务器提供的用户身份信息进行用户认证，双方之间通过基于 SAML 标准交换信息。下图显示了基于 SAML 的 SSO 的基本流程：

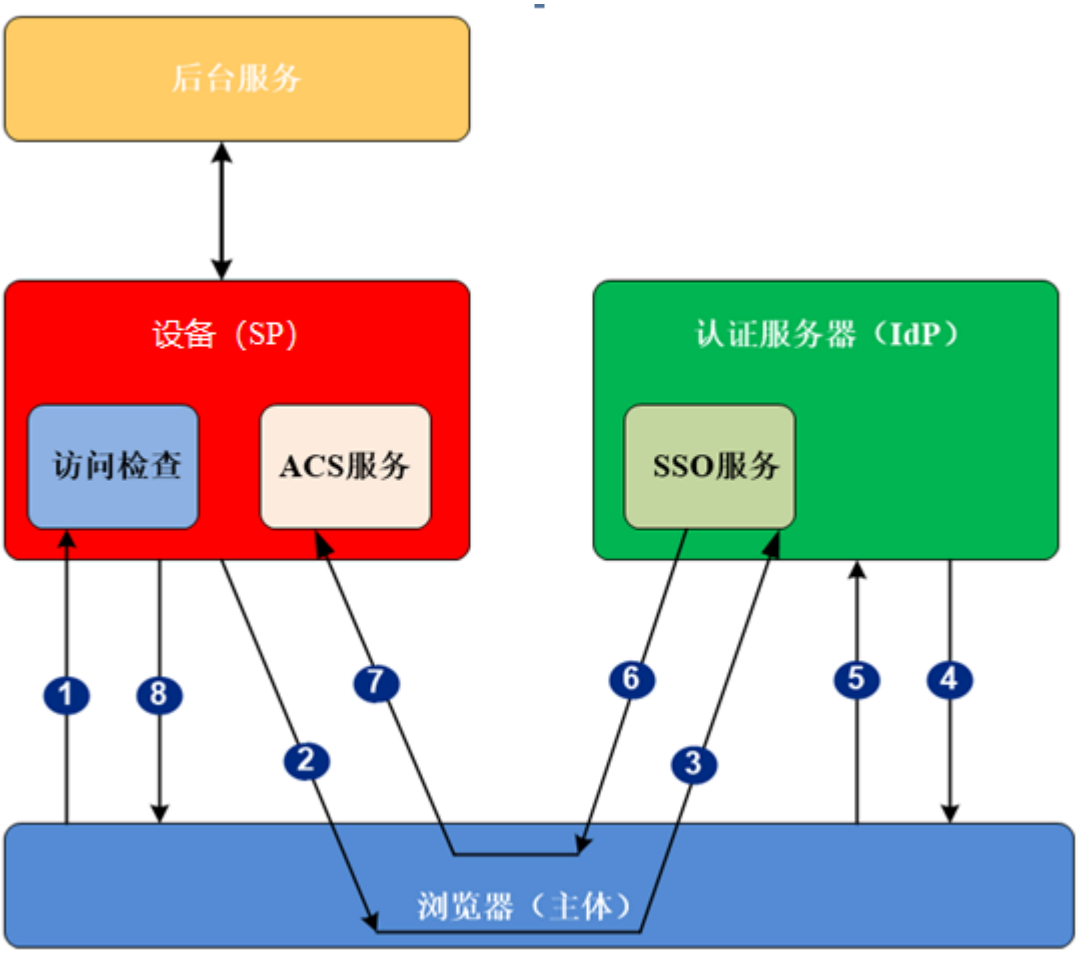

图22–1 SSO 流程

1. 客户端通过浏览器访问 SP，请求获取后台服务上的资源。
2. SP 创建一个 SAML 认证请求，重定向客户端到 IdP 服务器请求认证，请求里会指定 ACS 模块支持的协议绑定类型（通过“**aaa samlsp sp acs**”命令配置）。
3. 客户端将 SAML 认证请求发给 IdP。
4. IdP 向客户端返回认证页面，要求客户端输入登录凭据。
5. 客户端输入登录凭据。
6. IdP 验证登录凭据后返回一个 SAML 响应给客户端，其中包含关于该用户的身份断言，例如“This user is John Doe, he has an email address of john.doe@example.com, and he was authenticated into this system using a password mechanism.”。
7. 客户端浏览器将 SAML 响应发送给 SP（ACS 模块）。
8. SP 从 SAML 响应中提取出用户名（通过“**aaa samlsp idp attributes**”命令配置），确定客户端已经经过认证，返回资源给客户端。

基于以上流程的认证后，如果客户端后续访问其他资源，设备会基于已有的认证会话直接返回资源，无需客户端再次输入密码。

22.1.1.1.3 SLO 流程

通过 SAML SSO，一个终端用户可以通过一个认证环境访问多个站点，这样一个用户会在所有站点上都拥有登录会话。相比之下，SLO 的作用是回退所有这些会话的 SAML SSO 流程，将该用户在所有已登录站点上的登录会话同时注销。例如，一个终端用户通过一个 IdP 同时登录了多个 SP 站点，当该用户从其中一个 SP 站点注销时，所有其他站点上的登录也将同时被注销。

下图显示了基于 SAML 的 SLO 的基本流程：

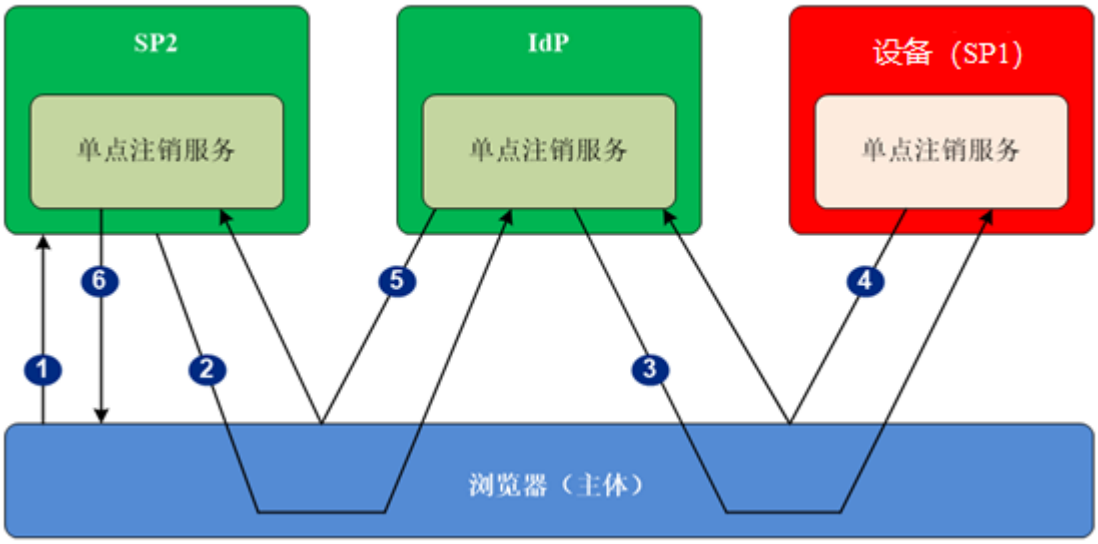

图22–2 SLO 流程

1. 客户端在 SP2 上发起注销请求。
2. SP2 创建 SAML 注销请求，重定向客户端到 IdP 请求注销。
3. IdP 创建 SAML 注销请求，根据 SP1 元数据文件定义的协议绑定类型将请求通过客户端的浏览器发送给 SP1 的 SLO 模块。
4. SP1 创建 SAML 注销响应，通过浏览器告知 IdP 客户端的登录会话已注销。
5. IdP 通过客户端浏览器回复注销响应给 SP2，告知 SP2 客户端已经在其他 SP 上注销。
6. SP2 回复注销响应给客户端，客户端从 SP2 注销。

22.1.1.1.4 配置示例

配置场景如下图所示：

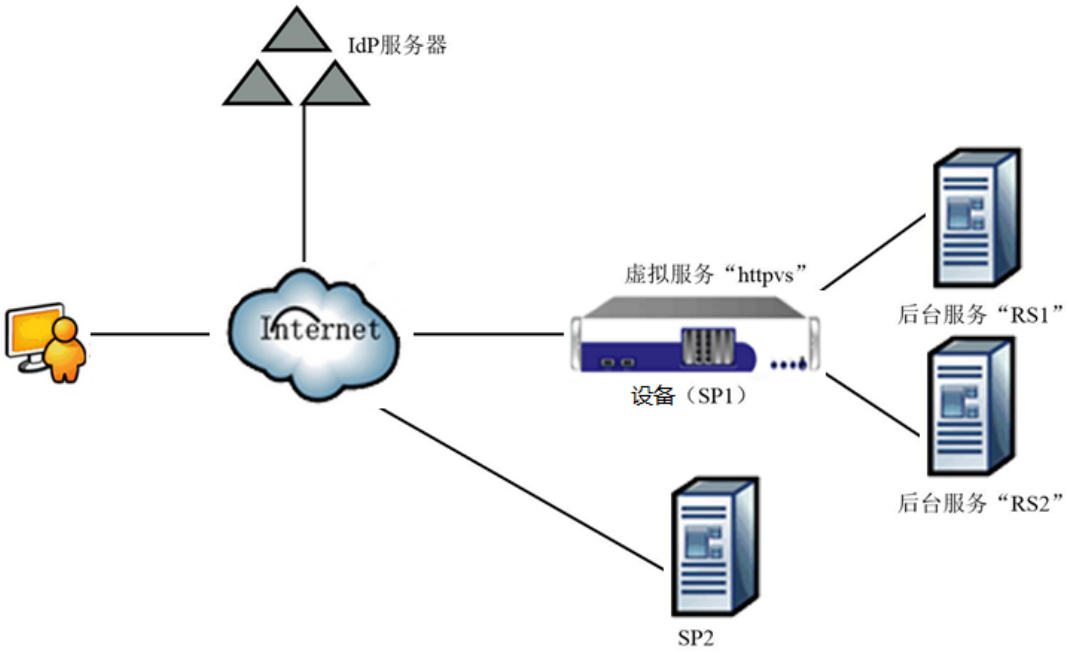

图22–3 配置示例

配置要求为设备可以通过 IdP 服务器对访问“RS1”和“RS2”的客户端进行 SAML 认证。

- **SLB 和 SSL 配置**

SLB 配置部分需完成虚拟服务、后台服务、策略和算法等基本配置，具体配置方法请参考第 11 章服务器负载均衡（SLB）。

SSL 配置部分需完成 SSL 主机和证书相关配置，具体配置方法请参考第 18 章 SSL 加速。

- **SAML 认证配置**

1. 配置一个 SAML SP 类型的 AAA 服务器。

```
Demo(config)#aaa server samlsp saml_sp
```

2. 将 AAA 服务器“saml_sp”设置为 AAA 方案“samlsp_method”。

```
Demo(config)# aaa method server samlsp_method saml_sp
```

3. 将“samlsp_method”与虚拟服务“httpsvs”关联。

```
Demo(config)#aaa method bind samlsp_method httpsvs
```

4. 为 SAML SP 服务器导入 IdP 服务器的元数据文件。

```
Demo(config)#aaa samlsp idp metadata saml_sp “http://idp.metadata/file”
```

5. 配置 SAML SP 服务器获取用户身份信息的属性名称。

```
Demo(config)#aaa samlsp idp attributes saml_sp “subject.nameid”
```

6. 配置 SAML SP 服务器的 ACS 服务支持的协议绑定类型。

```
Demo(config)#aaa samlsp sp acs saml_sp post
```

7. 配置 SAML SP 服务器的 SLO 服务支持的协议绑定类型。

```
Demo(config)#aaa samlsp sp slo saml_sp redirect
```

8. 为 IdP 服务器导入 SAML SP 服务器的元数据文件。

9. SAML SP 服务器的元数据文件的下载地址可以通过“**show aaa samlsp metadata**”命令查看，例如：

```
Demo(config)#show aaa samlsp sp metadata saml_sp
https://10.8.6.140/prx/000/http/hostlocal/saml2/server1/module.php/saml/sp/metadata.php/server1
```

10. 添加一个用户角色“staff”。

```
Demo(config)#aaa role name staff
```

11. 为“staff”添加限制规则“work”。

```
Demo(config)#aaa role qualification staff work
```

12. 为用户角色“staff”的限制规则添加限定条件，例如每个月第一天登录的所有用户会获得用户角色“staff”。

```
Demo(config)#aaa role condition staff work “LOGINDAY IS 1”
```

13. 将用户角色“staff”与虚拟服务“httpsvs”关联。

```
Demo(config)#aaa role bind staff httpsvs
```

14. 启用 AAA 功能。

```
Demo(config)#aaa on httpsvs
```

- **（可选）会话管理配置**

1. 设置“httpsvs”上允许的最大会话数。

```
Demo(config)#aaa session virtual limit httpsvs 20000
```

2. 设置“httpsvs”上单个用户允许的最大会话数。

```
Demo(config)#aaa session virtual maxperuser httpsvs 200
```

3. 启用会话重用。

```
Demo(config)#aaa session virtual reuse on
```

4. 设置闲置会话的超时时间。

```
Demo(config)#aaa session virtual timeout idle httpsvs 3600
```

5. 设置会话的有效期。

```
Demo(config)#aaa session virtual timeout lifetime httpsvs 3600
```

22.1.1.2 LDAP

在设备上，LDAP 认证和授权方式支持所有的 LDAPv3 协议，例如 OpenLDAP 和 Active Directory。为了防止单点故障，每个 LDAP 服务器可配置最多三个 LDAP 主机，管理员可以为每个主机配置基于 TLS 协议的认证和授权方式。

LDAP 支持静态和动态 LDAP 绑定模式，两种模式不能同时配置。

在动态绑定模式下，系统会优先从 LDAP 服务器获取用户的 DN。启用动态 LDAP 绑定模式后，系统会向 LDAP 服务器发送一个包含终端用户的用户名和密码的绑定请求，随后发送一个搜索请求用于获取终端用户的 LDAP 记录项，搜索请求携带通过“aaa ldap searchfilter”命令配置的搜索过滤字符串。从 LDAP 记录项获取到用户的 DN 后，系统会将该 DN 和终端用户的密码通过另一个绑定请求一起发送给 LDAP 服务器。终端用户通过认证后，LDAP 记录项会被再次用来进行授权。

在静态绑定模式下，系统会使用“<dn_prefix><USER><dn_suffix>”字符串来构建用户的 DN，其中“<USER>”表示用于登录站点所用的用户名，“<dn_prefix>”和“<dn_suffix>”对于访问同一个站点的所有用户必须是相同的。启用静态 LDAP 绑定模式后，系统会将“<dn_prefix><USER><dn_suffix>”字符串和终端用户的密码通过一个绑定请求一起发送给 LDAP 服务器。终端用户通过认证后，系统会向 LDAP 服务器发送一个搜索请求来获取终端用户的 LDAP 记录项，搜索请求携带通过“aaa ldap searchfilter”命令配置的搜索过滤字符串。获取到的 LDAP 记录项会用于后续的用户授权。

22.1.1.3 RADIUS

RADIUS 认证和授权方式支持为每个 RADIUS 服务器配置最多三个主机来提供冗余。RADIUS 请求采用非阻塞和超时机制来确保传输效率。

22.1.1.4 OAuth

OAuth 允许用户授权第三方应用或网站访问他们存储在其他的服务提供者上的资源，而不需要将用户名和密码提供给这些应用或网站。OAuth 2.0 认证过程中主要有四类角色：

- 资源所有者：指终端用户。
- 资源服务器：指存放受保护资源的服务提供商。访问这些资源，需要获得访问令牌。
- 客户端：代表向受保护资源进行资源请求的第三方应用程序。

- 授权服务器：在验证资源所有者的身份并获得授权成功后，会发放访问令牌给客户端。

设备支持用户通过 Google 和 WeChat 登录方式进行 OAuth 2.0 认证。当用户访问虚拟服务时，设备会将用户重定向到 Google 或 WeChat 的 OAuth 服务器进行认证和授权，当认证通过并且用户同意给予授权后，OAuth 客户端会向 OAuth 服务器请求访问令牌，并使用访问令牌从资源服务器获取用户信息和资源。

22.1.1.4.1 认证流程

下图显示了设备与 OAuth 服务器和资源服务器配合完成 OAuth 认证的工作流程：

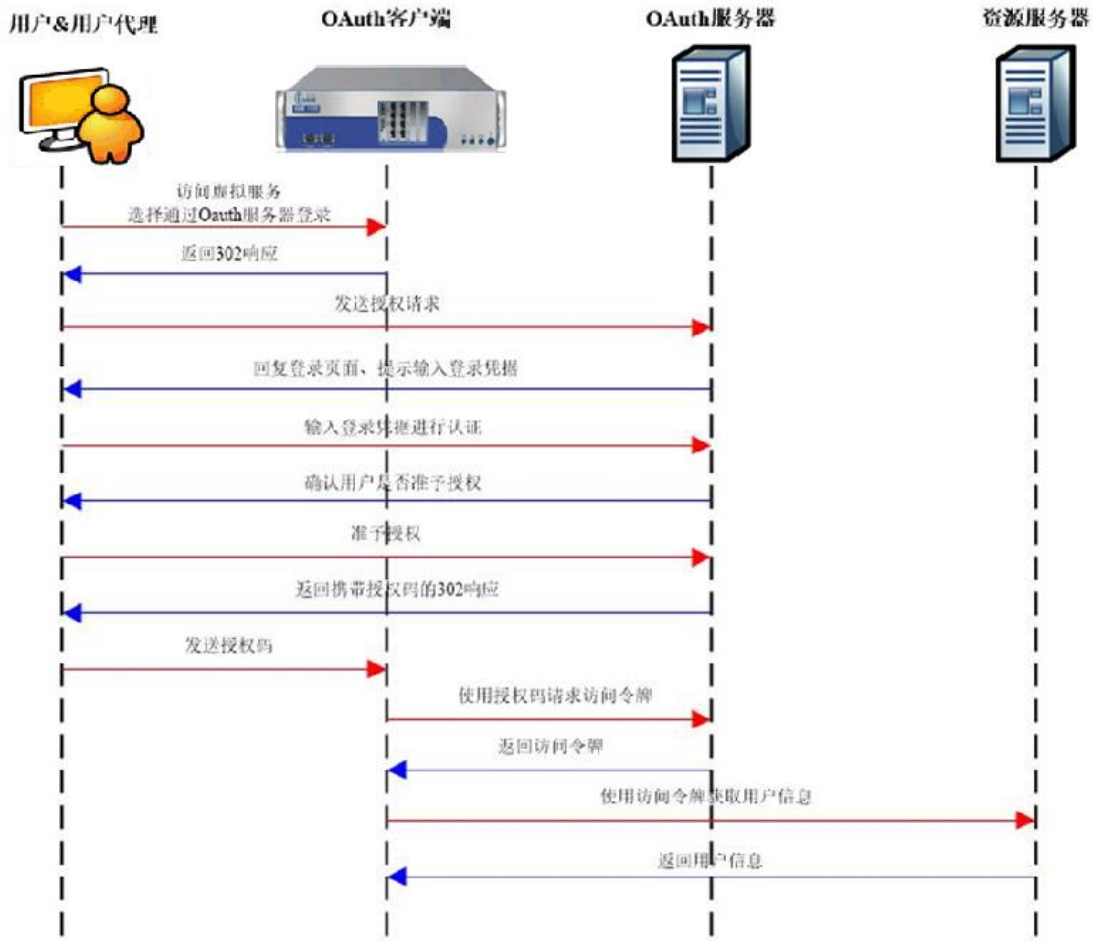

图22–4 认证流程

上述交互流程的详细说明如下：

1. 终端用户通过用户代理（浏览器）访问虚拟服务，并选择使用第三方的 OAuth 服务器登录。
2. OAuth 客户端（设备集成）返回 302 响应，重定向用户代理到 OAuth 服务器。
3. 用户代理向 OAuth 服务器发送授权请求。
4. OAuth 服务器校验授权请求后返回登录页面要求用户认证。

5. 用户输入正确的登录凭据。
6. 如果用户通过了认证，OAuth 服务器会提示用户是否授权给 OAuth 客户端。
7. 用户准予授权给 OAuth 客户端。
8. OAuth 服务器返回 302 响应，响应中包含授权码。
9. 用户代理将授权码发送给 OAuth 客户端。
10. OAuth 客户端将授权码发送给 OAuth 服务器请求访问令牌。
11. OAuth 服务器认证 OAuth 客户端后，将发放的访问令牌通过访问令牌响应回复给 OAuth 客户端。
12. OAuth 客户端使用访问令牌向资源服务器请求用户信息。
13. 资源服务器返回用户信息给 OAuth 客户端，包括用户 ID、email 账号、昵称和头像。

OAuth 2.0 要求 OAuth 客户端自己向 OAuth 服务器认证，因此管理员需要注册 OAuth 客户端来获取客户端 ID 和密码，并在 OAuth 服务器的服务提供商的开发者平台上注册重定向 URL。如果使用的是 Google OAuth 服务器，重定向 URL 的格式必须是“https://<virtual_service_domain_name>/prx/000/http/hostlocal/oauth_code”。如果使用的是微信的 OAuth 服务器，重定向 URL 必须是虚拟服务的域名。关于如何注册 OAuth 客户端和重定向 URL 的更多信息，请联系 OAuth 服务器的服务提供商。

注意：

1. OAuth 认证不支持多因素或多步骤认证。
2. 采用 OAuth 认证方式的 AAA 方案不受 AAA 方案排序功能的控制，会始终显示在终端用户的登录页面上，无法隐藏。
3. 要登录到使用第三方 OAuth 服务器的虚拟服务，终端用户在问虚拟服务时必须使用重定向 URL 中的域名。

22.1.1.4.2 高级配置

OAuth 认证支持以下高级配置：

- **认证后用户注册**

通过配置 OAuth 认证后用户注册功能，OAuth 认证功能可以将获取到 OAuth 用户名与保存在 AAA 服务器上的现有公司账户绑定。

启用该功能后，通过 OAuth 认证的用户会被要求注册，注册过程中，用户需要向 AAA 方案（通过“**aaa method register**”配置）中指定的认证服务器认证自己。用户通过这个认证后，设备会将已获取到的 OAuth 用户 ID 与用于注册的用

户名关联。用于注册的用户名（而不是已获取到的 OAuth 用户名）将用于后续的授权。

禁用 OAuth 认证后注册功能后，后续的授权中会使用获取到的 OAuth 用户名（如果是 Google OAuth 服务器，此处的 OAuth 用户名为 email 账号。如果是微信的 OAuth 服务器，此处的用户名为昵称）。因此，位于 OAuth 服务器所属的 AAA 方案中的授权服务器的账号的用户名必须和获取到的 OAuth 用户名相同，否则授权会失败。

- **使用邮箱账号前缀作为用户名**

当使用 Google OAuth 服务器时，如果禁用 OAuth 认证后用户注册功能，可以启用该选项来使用 email 账号作为授权中的 OAuth 用户名。例如获取到的 email 账号为“test@gmail.com”，那么在后续的授权中只有“test”会被用作用户名。

- **认证后授权过滤器**

通过使用认证后授权过滤器，OAuth 认证功能可以确保只有合法的公司用户才能在通过 OAuth 认证后进入授权流程。只有当 OAuth 用户名（如果是 Google OAuth 服务器，此处的 OAuth 用户名为 email 账号。如果是微信的 OAuth 服务器，此处的用户名为昵称）匹配指定的 OAuth 认证后授权过滤器时系统才会对用户提示授权。

例如配置认证后授权过滤器为“@xxxyyy.net”，那么只有当获取到的 OAuth 用户名为“test@xxxyyy.net”时，系统才会对该用户执行授权流程。

22.1.2 AAA 方案

设备通过 AAA 方案指定用于认证的 AAA 服务器和用于授权的 AAA 服务器，例如管理员可以定义一种使用 LDAP 进行认证、使用 RADIUS 进行授权的 AAA 方案。

22.1.2.1 多因素认证

如果需要对用户实施更加严格的安全检查，提高虚拟服务的安全防护级别，管理员可以在一个 AAA 方案中配置多个认证服务器来支持多因素认证（用户名相同、密码不同）。在多因素认证方式下，用户只有在通过了所有认证服务器的认证后才能成功登录到虚拟服务。

一个 AAA 方案中最多可以配置三个认证服务器，三个认证服务器可以是相同类型也可以是不同类型，但是 SAML SP 类型的服务器不可以和其他类型的同时定义。

22.1.2.2 AAA 方案排序

当应用场景内存在多个 AAA 服务器，而且每个 AAA 服务器上都存储了用户的登录凭据时，管理员可以为一个虚拟服务定义多种 AAA 方案，用户在登录虚拟服务前可以根据提示选择采用哪种 AAA 方案进行认证。如果不需要用户了解虚拟服务都支持哪些 AAA 方案，可以采用 AAA 方案排序功能。通过该功能，设备会在登录页面隐藏 AAA 方案选项，当用户输入登录凭据后，设备会按照 AAA 方案的优先级由高到低进行选择，直到认证通过。

22.1.2.3 认证

在认证过程中，设备会从授权服务器获取授权信息，目前可获取的授权信息为用户所属的群组信息，获取的授权信息还会用于用户角色的分配。

22.1.3 用户角色

设备基于用户角色的限制规则为通过认证的用户授权资源。通过用户角色，可以实现更加灵活、准确和精细的用户身份识别。管理员可以为每个用户角色定义一条或多条限制规则，每个限制规则可以包含最多 32 条限制规则，限定条件包括登录时间、用户名、群组名、源 IP 地址以及使用的 AAA 方案等。

在通过虚拟服务访问资源之前，用户必须获取至少一个用户角色，否则设备会注销用户的登录并要求用户重新登录虚拟服务。一个用户可能同时匹配多条限制规则，但是只有当用户匹配限制规则里的所有限定条件时才能获取一个角色。不匹配任何限制规则的用户无法获取用户角色，因此也无法通过设备获取任何资源。管理员也可以定义一个限制规则内不包含任何限定条件的用户角色，所有通过认证的用户都会被分配这个角色。

22.1.4 会话管理

当用户通过认证后，系统会为该用户创建一个登录会话，记录会话 ID、用户名、密码等信息，管理员可以通过“**show aaa session active**”命令查看当前会话。同时，系统还支持以下配置管理认证会话：

- 会话数限制

系统支持设置每个虚拟服务上总会话数限制和单个用户的最大会话数。如果当前活动会话数量已达到指定的限制，系统不会再创建新的会话。

- 会话超时时间

闲置会话如果长时间不关闭，会一直占用连接资源，管理员可以设置闲置会话超时时间及时释放资源。

- 对于处于活动状态的会话，系统也支持设置会话有效期来控制会话时长。

- 会话重用功能

会话重用功能允许用户名相同的用户共享一个会话，如果共享会话中的任一用户关闭连接，所有其他用户必须重新登录。

22.2 Web SSO

除了 SAML SSO 外，设备还支持 Web SSO 功能。在一个多系统的应用环境中，通过部署 Web SSO 功能，用户同样只需要登录一次就可以访问所有相互信任的应用系统。设备需要依赖于 SAML、LDAP、RADIUS 或者 OAuth 认证功能实现 Web SSO 支持。在需要部署 Web SSO 的应用场景，必须先部署 SAML、LDAP、RADIUS 或者 OAuth 认证功能。

注意：

1. 当基于 SAML 部署和实现 Web SSO 时，“**aaa samlsp idp attributes**”命令中的“username”参数必须设置为“uid”。另外，管理员必须确保设备能从 IdP 服务器获取到用户的登录凭据。本公司 NetGate 系列设备可以作为 IdP 服务器与设备协作实现 Web SSO 支持，如需更多信息，请联系公司技术支持。
2. 当基于 OAuth 部署和实现 Web SSO 时，“**aaa oauth registration**”命令的“save_password”参数值必须设置为“save”。

在设备上，Web SSO 功能可以基于以下方式实现：

- NTLM（NT LAN Manager）认证方式
- HTTP 基本认证方式
- HTTP POST SSO 规则

22.2.1 NTLM 认证方式

NTLM 是一种在认证过程中保护用户名和密码的认证协议，采用“质询-应答”方式的认证机制。在不直接提供密码的情况下，客户端能够间接证明自己知道密码，因此能够防止密码被他人获取。

注意：在 NTLM 认证方式下，后台服务长连接功能必须为启用状态，管理员可以通过“**http serverpersist on**”命令启用该功能。

设备和后台服务之间通过 NTLM 认证实现客户端 SSO 登录的流程如下：

1. 客户端通过设备访问后台服务上的资源。
2. 后台服务回复 401 未授权响应，要求客户端进行 NTLM 认证。

3. 设备代替客户端提交 NTLM 协商消息（Type-1 消息），和后台服务协商需要认证的主体、需要使用的安全服务等信息，并告知后台服务自己支持的协议内容、加密等级等。
4. 后台服务回复质询消息（Type-2 消息），消息中包含选择的加密等级、安全服务和一个随机数 challenge。
5. 设备使用客户端的密码和收到的随机数生成一个随机数，通过认证消息（Type-3 消息）发送给后台服务。
6. 后台服务会经过几乎相同的运算计算出一个认证消息，然后和收到的认证消息对比，发现二者匹配，因此确认客户端掌握正确的密码，认证成功。

22.2.2 HTTP 基本认证方式

在 HTTP 基本认证方式下，服务器通过检验登录凭据（用户名和密码）进行认证。设备和后台服务之间通过 HTTP 基本认证实现客户端 SSO 登录的流程如下：

1. 客户端通过设备访问后台服务上的资源。
2. 后台服务回复 401 未授权响应，要求客户端进行基本认证。
3. 设备代替客户端提交认证请求，请求里包含客户端的用户名和密码。
4. 后台服务验证客户端的登录凭据，返回请求的资源。

22.2.3 HTTP POST 规则

HTTP POST SSO 规则定义了登录凭据的提交地址以及如何提交。如果终端用户请求访问的 URL 地址匹配 HTTP POST SSO 规则，设备会发起 SSO POST 流程。

22.2.4 配置示例

该示例为设备配置基于 SAML 认证功能的 Web SSO。

- **配置要求**

配置设备支持基于 HTTP POST SSO 规则的 Web SSO 功能。

- **配置步骤**

1. 配置 SAML SSO。

除了“**aaa samlsp idp attributes**”命令的配置外，SAML SSO 的配置步骤与“22.1.1.1.4 配置示例”小节相同。在配置 SAML SSO 用于 Web SSO 支持的场景，该命令中的“username”参数必须设置为“uid”，另外管理员可以选配“password”参数，例如：

```
Demo(config)#aaa samlsp idp attributes saml_sp uid password
```


2. 启用 Web SSO 认证功能。

```
Demo(config)#aaa sso on
```

3. 配置 HTTP POST SSO 规则。

```
Demo(config)#aaa sso post httpsvs abc.com “http://abc.com/picture” username password
```

基于以上配置，如果终端用户访问的 URL 地址为“http://abc.com/picture”，设备将代替客户端提交登录凭据到该地址。

第23章 Webagent

Webagent 发挥网络代理服务器的功能，负责代理内网客户端访问外网应用，并可以对客户端的访问行为进行控制。在设备上，Webagent 功能通过 Webagent 服务代理 HTTP 和 HTTPS 客户端请求，适用于 IPv4 和 IPv6 环境，并支持 DNS 缓存功能。

23.1 Webagent 服务

管理员可以通过配置Webagent服务和DNS服务器来实现设备作为网络代理服务器的功能。下图描述了一个基本的 Webagent 服务配置场景（客户端需要将设备设置为代理服务器）。

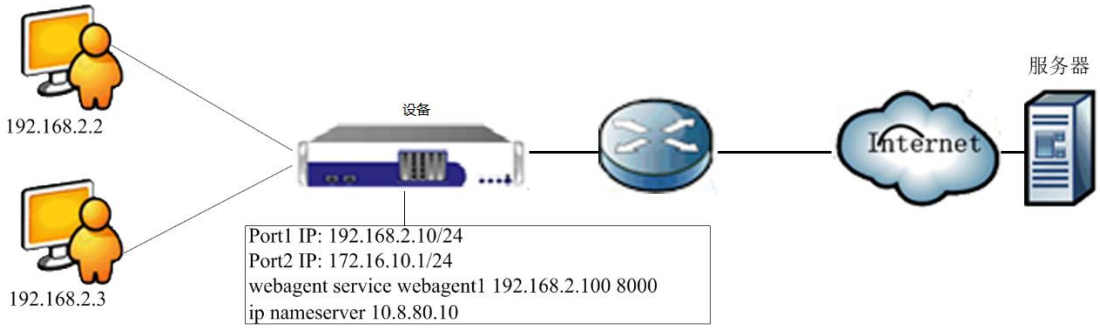

图23–1 设备作为 Webagent

23.2 Webagent 访问控制

设备支持配置以下几种 Webagent 访问控制规则：

- 允许访问指定主机名的请求通过
- 禁止访问指定主机名的请求通过
- Webagent 访问控制规则必须加入到访问组后才可以应用于 Webagent 服务。这些规则可以同时配置，系统支持配置最多 1024 条 Webagent 访问控制规则和 1024 个访问组。默认情况下，如果一个请求同时命中允许访问控制规则和禁止访问控制规则，前者生效。

示例：

1. 配置 Webagent 服务。

```
Demo(config)#webagent service webagent1 192.168.2.100 8000
```

2. 配置 Webagent 允许访问控制规则，允许访问的主机名匹配“xyz*”和“abc*”的请求通过。

```
Demo(config)#webagent acl permit http host item_1 “xyz*”
```

```
Demo(config)#web agent acl permit http host item_2 “abc*”
```

3. 配置一个 Webagent 访问组。

```
Demo(config)#webagent acl group name group_1
```

4. 添加“item_1”和“item_2”到“group_1”。

```
Demo(config)#webagent acl group member group_1 item_1
Demo(config)#webagent acl group member group_1 item_2
```

5. 为 Webagent 服务应用访问组。

```
Demo(config)#webagent acl group apply webagent_1 group_1
```

23.3 DNS 缓存

Webagent 服务的 DNS 缓存功能支持缓存 A 和 AAAA 类型的资源记录。

默认情况下，系统会对 DNS 服务器返回的资源记录实施动态缓存，缓存时间为 60 分钟。在缓存到期前，如果有客户端再次查询该域名，系统会直接返回缓存中的资源记录，不需要再请求 DNS 服务器解析域名。同时，系统会重新开始计算该资源记录的缓存时间。系统支持动态缓存最多 256 个 DNS 域名，每个域名可以缓存 8 条 A 类型和 8 条 AAAA 类型的资源记录。当缓存的动态域名个数达到 256 个时，如果有新的域名的资源记录需要缓存，系统会用新的域名替换最先缓存的域名。

对于客户端访问频繁的域名，管理员可以通过“**webagent dns host**”命令添加静态资源记录。静态资源记录一旦添加到缓存中则永不超时，只能手动删除。系统支持配置最多 256 个静态 DNS 域名，每个域名可以添加 8 条 A 类型和 8 条 AAAA 类型的资源记录。

设备支持修改 Webagent DNS 缓存记录的最短和最长过期时间。

第24章 服务质量（QoS）

24.1 概述

这章介绍如何配置服务质量（Quality of Service，QoS）功能。QoS 功能可以说明管理员更好的控制网络带宽，使他们可以同时从技术以及商业的角度去管理网络。

24.2 QoS 原理

网络服务质量是保证关健应用的高性能的标准和机制。通过 QoS 机制，网络管理员可以有效的利用现有的资源，同时在不扩大网络的基础上保证了服务的质量。

QoS 使得管理员能够通过队列机制和数据报过滤策略管理 TCP、UDP 和 ICMP 流。

24.2.1 队列机制

设备采用基于队列的 QoS。队列是由一组网络数据报缓存组成的。当队列中的一个数据报被处理后，一个新的等待处理的数据报将会被放在队列的末尾。

每一个队列都与一个特定的网络接口绑定，控制这个网络接口的进出的网络流量。设备 QoS 队列是树状管理结构。在一个树的顶端，首先要定义一个根队列来管理进出网络接口的流量。在根队列下面，又可以定义多个子队列。子队列下面还可以定义它的子队列。每一个接口至多只能有两个队列树，一个是为入站流量工作的，一个是为出站流量工作的。

每一个队列都有带宽限制和数据报处理优先级。

24.2.2 数据报过滤规则

QoS 过滤规则是一个把特定的网络流量与 QoS 队列相关联的策略。

在过滤规则中，网络流量由以下五个参数指定：源子网 IP 地址、源端口、目的子网 IP 地址、目的端口以及协议。使用者可以选择基于应用或是基于链接的 QoS 控制。基于应用的过滤规则主要依据已经定义的 TCP 或 UDP 端口，而基于链接的过滤规则则主要依据源或目的地址。

24.2.3 带宽管理

带宽管理是通过一系列的 QoS 过滤规则实现的。通过这些过滤规则，设备把特定的网络流量绑定到已经定义的有带宽限制的 QoS 队列中。QoS 过滤规则有助于设备恰当地分配带宽，从而满足不同应用和链接的需求。

为了更加灵活的控制带宽，在树状结构的 QoS 队列中使用“BORROW/UNBORROW”策略。当一个队列被设置为“BORROW”，那么它就可以通过从它的父队列借用带宽来增加它自己的带宽。如果它的父队列没有多余的带宽可以借用给它，它可以向它的父队列的父队列借用带宽，依此类推，直到向根队列借用。

24.2.4 优先级控制

优先级控制是通过给 QoS 队列设置不同的优先级来实现的。来自不同应用或链接的数据报首先根据 QoS 过滤规则进行分类，然后再分配到已经定义的队列中，从而分配到不同队列中的数据报也就有了与各自队列相同的的优先级。

当网络变的拥挤的时候，这个优先级机制能够发挥至关重要的作用。当流量达到一个顶峰时，等待处理的数据报数量超过了最大的队列缓存，这时，就会出现丢包现象。在这种情况下，如果使用了优先级机制，一个队列中优先级高的数据报就可以优先被处理，优先级低的数据报可能被丢掉。用户可以给紧急重要的应用分配高的优先级，这样就能保证这些重要的应用的数据报不会被丢。

24.3 QoS 配置

24.3.1 配置向导

下面列出了配置 QoS 所需的命令，相关命令的描述信息，请参考命令行使用手册。

表24–1 设备 QoS 配置命令

配置操作	命令行
配置 QoS 接口	**qos interface** *<interface_name> [direction] [bandwidth]*
定义 QoS 队列	**qos queue root** *<queue_name> <interface_name> [direction] [bandwidth] [priority] [borrow] [default]* **qos queue sub** *<queue_name> <parent_queue> [bandwidth] [priority] [borrow] [default]*
定义 QoS 过滤规则	**qos filter** *<filter_name> <queue_name> < src_addr> <smask> <sport> <dst_addr> <dmask> <dport> <proto> [priority]*
启用 QoS	**qos enable** *<interface_name> [direction]*

24.3.2 配置示例

1. 指定需要配置 QoS 功能的接口。这个接口可以是系统接口、VLAN 接口或者聚合接口。

```
Demo(config)#qos interface port1 OUT 5Mb
Demo(config)#qos interface port1 IN 5Mb
```

2. 创建出向 QoS 队列。

```
Demo(config)#qos queue root qr_oall port1 OUT 5Mb 3
Demo(config)#qos queue sub qs_ossh qr_oall 2Mb 3 UNBORROW NONDEFAULT
Demo(config)#qos queue sub qs_oftp qr_oall 512kb 2 UNBORROW NONDEFAULT
Demo(config)#qos queue sub qs_odeflt qr_oall 8kb 3 UNBORROW DEFAULT
```

3. 创建入向 QoS 队列。

```
Demo(config)#qos queue root qr_iall port1 IN 5Mb 3
Demo(config)#qos queue sub qs_issh qr_iall 2Mb 3 BORROW NONDEFAULT
Demo(config)#qos queue sub qs_iftp qr_iall 2Mb 2 BORROW NONDEFAULT
Demo(config)#qos queue sub qs_ideflt qr_iall 8kb 3 BORROW DEFAULT
```

4. 定义 QoS 过滤规则。

```
Demo(config)#qos filter fltr_ftp_o qs_oftp 0.0.0.0 0.0.0.0 0 10.3.54.40 255.255.255.255 0 tcp 2
Demo(config)#qos filter fltr_ftp_i qs_iftp 10.3.54.40 255.255.255.255 0 0.0.0.0 0.0.0.0 0 tcp 2
Demo(config)#qos filter fltr_ssh_o qs_ossh 0.0.0.0 0.0.0.0 22 0.0.0.0 0.0.0.0 0 tcp 3
Demo(config)#qos filter fltr_ssh_i qs_issh 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 22 tcp 3
```

5. 启用 QoS 功能。

```
Demo(config)#qos enable port1 OUT
Demo(config)#qos enable port1 IN
```

第25章 链路负载均衡（LLB）

25.1 概述

本章描述了设备在出入两个方向上的链路负载均衡（Link Load Balance，LLB）应用，包括：

- 单设备和两个 ISP
- 双设备和两个 ISP

25.2 链路负载均衡原理

链路负载均衡能够使 TCP/IP 数据流在多个 ISP 之间实施负载均衡，最多支持 128 个 ISP。链路负载均衡可以分为出口链路负载均衡和入口链路负载均衡，支持算法包括轮询（rr）、加权轮询（wrr）、最短响应时间（sr）、动态探测（dd）、哈希 IP（hi）、哈希 IP 和端口（hip）、最小带宽使用率（lb）和最小连接数（lc）。链路负载均衡也包含 ISP 链路错误检测功能，它通过默认的网关以及用户指定的检测点来检测链路状态。

设备通过逻辑端口和对端 MAC 地址来区分不同的链路。各个链路的流量统计信息也是基于逻辑端口和对端 MAC 地址来收集的。

25.2.1 出口链路负载均衡

在拥有多个默认的网关时，出口链路负载均衡（Outbound LLB）优化了链路的利用，它的基本功能是把数据流分发到多个上游路由器。例如，假设可以同时从两个 ISP（ISP1、ISP2）连到 Internet，ISP1 分配的网段是 100.1.1.0/24，ISP2 分配的网段是 200.1.1.0/24，你可以把它们分别配置到设备的网卡上，出口链路负载均衡允许出去的数据流在 ISP1 和 ISP2 之间负载均衡。分配到 ISP1 的连接被地址转换成 ISP1 的地址范围，分配到 ISP2 的连接被转换成 ISP2 的地址范围，因此，回来的响应数据也会从相应的 ISP 流入。如果其中一个 ISP 的链路不通，所有的数据流将不会分配到这个 ISP。

设备 LLB 功能可以对基于 TCP 和 UDP 协议的数据流量进行很好的链路负载均衡处理，并且能同样对基于 IP 协议或 IPsec、GRE 等高级协议的数据包进行链路负载均衡。

25.2.2 入口链路负载均衡

入口链路负载平衡（Inbound LLB）为客户访问内部服务提供负载均衡。通过各个 ISP 提供的 IP 地址，内部服务对外部的客户是可访问的。举个例子，网络环境和上面的例子相同，如果外部客户访问的 ISP1 提供的地址，那么它们所有的

数据将会从 ISP1 的骨干网络到达内部提供服务的后台主机，同理，如果外部客户访问的 ISP2 提供的地址，那么它们所有的数据将会从 ISP2 的骨干网络到达内部提供服务的后台主机。入口链路负载均衡可以把分别属于 ISP1 和 ISP2 的两个 IP 地址对应到一个设备或一个后台服务，设备上的 DNS 服务会对已配置的域名进行解析，解析的结果包含 ISP1 或 ISP2 的 IP 地址，它们都对应着同一个设备或同一个后台服务。如果其中一个 ISP 的链路不通，DNS 服务器将不会把该 ISP 的 IP 放到解析结果中，因此，客户得到的 IP 总是可用的。

25.2.3 链路负载均衡健康检查

健康检查可验证 ISP 网络是否连通，除了可以通过 ARP 广播包和使用 ICMP Ping 工具测试指定的地址之外，设备还提供了另外两种链路健康检查功能，分别是基于 TCP 的健康检查和基于 DNS 的健康检查。

ARP 数据报可以检查直接相连的上游 ISP 路由器，而 Ping 除了具备检查直接相连的上游 ISP 路由器功能外，还可以检查从直接相连的上游 ISP 路由器到用户自定义的 IP 地址是否连通。用户可以配置多个被检查的 IP 址，如果其中一个是连通的，那么至少可以证明从设备的接口到相连的上游 ISP 路由器是连通的。健康检查保证负载均衡的数据会被分配到一个连通的链路。

25.2.4 LLB 远程站点可访问性检查

链路负载均衡健康检查提供对链路的健康检查，而 LLB 远程站点可访问性检查使得对远程站点的可访问性检查成为可能。远程站点指的是一个远程网络，管理员可以指定一个远程网络中的目标 IP 地址用于远程站点的可访问性检查。

LLB 远程站点可访问性检查支持基于 ICMP/TCP 的远程站点检查。

只有出口链路负载均衡算法为“rr”或“wrr”的出向通信才能进行 LLB 远程站点可访问性检查。

25.2.5 链路负载均衡算法

出口链路负载均衡支持以下负载均衡算法：

- 轮询（rr）
- 加权轮询（wrr）
- 最短响应时间（sr）
- 动态探测（dd）
- 哈希 IP（hi）
- 哈希 IP 和端口（hip）

- 最小带宽使用率（lb）
- 最小连接数（lc）

入口链路负载均衡支持三种负载均衡算法：

- 轮询（rr）
- 加权轮询（wrr）
- 就近性算法（proximity）

轮询算法（rr）

默认的调度算法，它按轮询的方式把每个新的会话分配到各个 ISP 的网关上。

加权轮询算法（wrr）

原理与轮询算法类似，不过它给每个网关分配一个权值，权值高的网关比权值低的网关有更多的机会获得新的会话，这是使得更多的流量被分配到带宽更大的 ISP 网关上。

最短响应时间算法（sr）

拥有最短响应时间的链路将会获得下一个新的会话。最短响应时间是根据每个 TCP 连接的握手过程计算得到的。如果都是 UDP 数据流或者一些长连接的 TCP 数据流，最短响应时间是不精确的，拥有大量的 TCP 连接建立才能得到更精确最短响应时间。

注意：

- 如果系统中既没有 SLB 流量也没有 NAT 流量，LLB sr 算法将不能正常工作。
- sr 算法不能用来对 IP 分片、非 TCP/UDP 报文和重组的 UDP 报文进行链路负载均衡。

动态探测算法（dd）

对所有可用的 ISP 路径进行 proximity 计算。通过平行探测算法，一个客户端请求将会同时通过不同的 ISP 路径发送到同一个目的地 B。当客户收到第一个响应时，传输这个最快响应的 ISP 路径就被选为这个请求的最优路径，同时其他 ISP 链接都会被断开。所有其他到目的地 B 的出口流量，设备都会选择这个最优 ISP 路径。

哈希 IP 算法（hi）

该算法通过对 IP 地址进行哈希运算，增加权重高的链路被路由到的几率，使得更多的流量被分配到权重高的链路上。当通过哈希运算首次选中的链路不通时，系统将在所有可用链路中重新进行哈希运算，重新选择权重较高的链路。使用哈希 IP 算法时，可以禁用 IPflow 功能。

哈希 IP 和端口算法（hip）

该算法通过对 IP 地址和端口进行哈希运算，增加权重高的链路被路由到的几率，使得更多的流量被分配到权重高的链路上。当通过哈希运算首次选中的链路不通时，系统将在所有可用链路中重新进行哈希运算，重新选择权重较高的链路。使用哈希 IP 和端口算法时，可以禁用 IPflow 功能。

最小带宽使用率（lb）

该算法会选择出向链路中带宽使用率最小的链路来转发出向流量。使用该算法时，需要使用“**llb link route**”命令指定链路允许的最大带宽。如果链路未指定允许的最大带宽，则该算法不会选择此链路。如果所有链路都没有指定允许的最大带宽，则使用 wrr 算法。

最小连接数（lc）

该算法会选择连接数最少的出向链路来转发出向流量。lc 算法建议在所有 LLB 链路都配置 NAT 且 LLB 链路的出向流量都能被 NAT 的情况下使用。如果所有 LLB 链路都没有配置 NAT，lc 算法实际为 wrr 算法。其他情况下不建议使用该算法。

就近性算法（proximity）

该算法会将距离客户端最近的服务器的 IP 地址作为响应发送给客户端。当接收到来自客户端的请求时，设备首先会通过反向查找，找到与该 DNS 请求源地址所匹配的路由，然后根据路由网关将相应的域名服务器 IP 地址（A 记录）返回给客户端。

25.2.6 策略路由

链路负载均衡策略（Eroute）提供一种允许管理员根据源/目的 IP、服务种类（邮件、FTP、Web 等等）直接指定出口数据流向的策略。这种策略是基于 Eroute 路由的，与普通的路由不一样。在选路时，它除了利用目的 IP 地址外，还利用源地址和源/目的端口以及协议类型进行路由。举个例子，应用策略路由可以保证所有 AOL 实时通信软件的数据流都分配到同一条链路上。如果通信客户端在请求数据报中使用不同的目的 IP 地址，并且这些请求被发送到不同的链路上，通信服务器有可能无法确认客户端，从而导致登录失败。配置策略路由可以阻止这类问题，它的配置命令示例如下：

```
Demo(config)#ip eroute aol_route 1500 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 5190 tcp gateway_ip
```

设备最多支持 5000 条“ip eroute”配置。

IP 域（IP region）

设备 Eroute 路由支持 IP 域。管理员可以通过 HTTP、FTP 或本地文档的方式导入预先定义好的 IP 域表，然后使用命令“**ipregion route**”调用导入的 IP 域表，实现一次性配置大量的 Eroute，无需手动进行配置。管理员也可以通过 FTP 或本地文档的方式从设备上汇出 IP 域表。

导入 IP 域表文档时，设备会对文档内容进行检查（不检查文件类型）。因此，为了确保可以成功导入 IP 域表文档，管理员自定义的 IP 域表文档内容要严格按照规定的格式包含以下信息项：

- IP subnet（IP 子网；采用 CIDR 格式）
- Country name（国家名称；可选，最长 7 个字节）
- Brief description（描述信息；可选，最长 63 个字节）

这些信息项必须用“Tab”键分隔。

例如：

```
27.8.0.0/13      CN   China Unicom Chongqing Province network
27.36.0.0/14     CN   China Unicom Guangdong Province network
```

注意：系统默认存在“predefined_cernet”、“predefined_cnc”和“predefined_ct”三个预定义 IP 域表。管理员也可以导入自定义的 IP 域表，建议不要使用和预定义域表同样的名称。通过 IP 域配置的路由和 proximity 规则在系统中作为一个整体存在，管理员不能修改或删除其中的单个路由或规则。

25.2.7 链路负载均衡会话的超时

当使用 IP flow（源 IP 地址和目的 IP 地址）功能时，如果一个数据流已经分配到一个 ISP 链路，那么其后面的所有具有相同的源/目的 IP 地址的数据流都将会被分配到这个 ISP 链路。如果 IP flow 在一段时间内都是空闲的，那么它会超时并且被删除。这样，后面具有相同的源/目的 IP 地址的数据流会根据负载均衡算法重新分配。

25.2.8 路由优先级

管理员需要提供一种必要的策略为出口的数据流选一条基于 IP 及协议类型的路由。设备支持多种路由，其中 Eroute 路由的优先权比默认路由和静态路由的优先权高，默认路由的优先权为 1，静态路由的优先权取决于网络屏蔽，在 101 至 132 之间，比如屏蔽为 24 则优先权是 124，屏蔽为 32 则优先权为 132。接口路由优先权为 2000，在访问本地子网段地址时自动生成的路由叫做直连路由（droute），直连路由的优先权为 2000。

下表列出了各种路由的优先级：

表25–1 路由优先级

路由名称	优先级
EROUTE-P（优先 EROUTE）	2001-2999
IROUTE, DROUTE（接口路由，直连路由）	2000

路由名称	优先级
RTS（RTS 路由）	1999
EROUTE-N（正常 EROUTE 路由）	1001-1999
IPFLOW（IPFLOW 路由）	1000-1999（默认 1000）
STATIC ROUTE（静态路由）	101-132 (IPv4) 101-228 (IPv6)
DYNAMIC ROUTE（动态路由）	101-132 (IPv4) 101-228 (IPv6)
LLB LINK ROUTE（LLB 链路路由）	2
DEFAULT ROUTE（默认路由）	1

25.2.9 链路带宽管理

为了实现更好的带宽管理，设备支持为对链路带宽设定最大阈值。

当系统为出向数据流量选择最优链路时，不仅考虑链路所配置的路由策略，同时还会考虑各条链路当前的负载情况，即：若当前链路的出向负载已经达到管理员设定的最大带宽阈值，设备会根据路由的优先级从高到低进行路由匹配查找，首先查找相同优先级路由中是否存在可用链路。如果所有可用链路均达到其最大带宽阈值，则会继续查找低优先级路由中是否存在可用链路。如果所有匹配路由的网关都不可用（DOWN）或达到管理员设定的最大带宽阈值，则出向数据流量还会继续使用当前的链路。

同时，设备还能够为出向链路带宽设定优先级。当出向数据流量所匹配到的路由的优先级大于链路带宽的优先级时，则该数据流量将由该路由所在的链路转发。

有了链路带宽管理功能，管理员不再需要为多条链路配置相同优先级的 Eroute 路由，这大大提高了带宽管理和配置的效率和灵活性。

注意：

- 如果流量匹配到 RTS 或 IPflow 路由，则流量将直接经由该路由进行转发，无论其所在链路的带宽是否达到最大阈值。
- 如果系统配置了一条全 0 的 Eroute 路由，则设备不会从优先级低于 1000 的匹配路由中查找可用的链路。

25.2.10 DNS 代理

DNS 代理功能可以为出向 DNS 查询请求提供链路负载均衡。通过使用 DNS 代理功能，系统可以实现将 DNS 查询请求均衡地分配到各条 ISP 链路上的 DNS 服务器上。

DNS 代理功能原理如下：

- 在收到内网客户端发送 DNS 查询请求时，设备根据负载均衡算法从所有 ISP 链路的 DNS 服务器中选择一个 DNS 服务器，并将 DNS 查询请求的目的 IP 地址修改为该 DNS 服务器的 IP 地址。
- 设备将 DNS 查询请求转发到该 ISP 链路的 DNS 服务器。
- DNS 代理功能支持如下负载均衡算法：
- rr：按轮询的方式把 DNS 查询请求转发到各条 ISP 链路的 DNS 服务器。
- wrr：给每一个 DNS 服务器分配一个权重值，权重值高的 DNS 服务器将收到更多的 DNS 查询请求。

注意：如果为各个 DNS 服务器指定相同的“weight”或者没有为其指定参数“weight”，则使用 rr 算法选择 DNS 服务器，否则使用 wrr 算法。

配置示例：

```
Demo(config)# llb link route link1 100.10.1.1 1 0.0.0.0 0Mbps
Demo(config)# llb link route link2 200.20.1.1 1 0.0.0.0 0Mbps

Demo(config)#llb dnsproxy server server1 100.10.1.3 1
Demo(config)#llb dnsproxy server server2 200.20.1.3 1
```

此外，DNS 代理功能还支持使用域名定制策略和 DNS network 策略来为出向的 DNS 查询请求选择 DNS 服务器。域名定制策略的优先级高于 DNS network 策略，DNS Network 策略的优先级高于负载均衡算法。具体请参见 25.2.10.1 DNS Network 策略和 25.2.10.2 域名定制策略。

25.2.10.1 DNS Network 策略

DNS network 策略用于为源 IP 地址属于特定网段的 DNS 查询请求选择一个 DNS 服务器。当 DNS 查询请求的源 IP 地址属于一条 DNS network 策略指定的网段时，系统将为其选择该策略指定的 DNS 服务器。如果一个 DNS 查询请求同时匹配多条 DNS network 策略，系统将优先选择子网掩码最长的 DNS network 策略。

配置示例：

```
Demo(config)#llb dnsproxy network 10.0.0.0 255.0.0.0 dns_server1
Demo(config)#llb dnsproxy network 10.0.1.0 255.255.255.0 dns_server2
```

执行以上配置后，对来自源 IP 地址为 10.0.1.3 的 DNS 查询请求，系统将为其选择名称为“dns_server2”的 DNS 服务器。

25.2.10.2 域名定制策略

系统支持三种域名定制策略：

- 静态域名策略：系统为特定域名的 DNS 查询请求固定地选择指定的 DNS 服务器。
- “Bypass”域名策略：系统不对指定域名的 DNS 查询请求应用 DNS 代理功能。
- “Persistent 域名策略”：系统根据源 IP 地址为指定域名的 DNS 查询请求持续地选择指定的 DNS 服务器。当 DNS 查询请求首次匹配上域名的 persistent 策略时，系统根据 DNS network 策略（如果配置并匹配）或负载均衡算法为 DNS 查询请求选择一个 DNS 服务器，并记录源 IP 地址和选择的 DNS 服务器的对应关系。当后续来自同一个 IP 地址的 DNS 查询请求匹配该策略，系统将持续地为请求选择同一个 DNS 服务器。

配置示例：

```
Demo(config)#llb dnsproxy domain "*.edu" edu_dns
```

执行以上配置后，对匹配域名“*.edu”的 DNS 查询请求，系统将为其固定选择名称为“edu_dns”的 DNS 服务器。

```
Demo(config)#llb dnsproxy domain www.abc.com bypass
```

执行以上配置后，对匹配域名“www.abc.com”的 DNS 查询请求，系统将不对其应用 DNS 代理功能。

```
Demo(config)#llb dnsproxy domain www.abc.com persistent
```

执行以上配置后，对匹配域名“www.abc.com”的 DNS 查询请求，系统将根据 persistent 策略为其选择 DNS 服务器。

25.2.10.3 链路带宽检查

DNS 代理功能还支持链路带宽检查。当一条 LLB 链路上的流量达到顶峰时，与该 LLB 链路关联的 DNS 服务器将不再被选中，直到该 LLB 链路出现空余带宽。系统将根据负载均衡算法从为 DNS 代理功能配置的其他可用 DNS 服务器中选择一个 DNS 服务器。

配置示例：

```
Demo(config)#llb dnsproxy link link1 server1
```

25.2.10.4 链路健康检查

DNS 代理功能支持链路健康检查。链路健康检查允许设备通过向目的主机发送 DNS 解析请求，来检测指定链路的健康状况。

配置示例：


```
Demo(config)#llb link route link1 100.10.1.1 1 0.0.0.0 0Mbps
Demo(config)#llb dnsproxy server server1 100.10.1.3 1
Demo(config)#llb dnsproxy link link1 server1
Demo(config)#llb link health checker dns link1 "100.10.1.3" www.test.com
Demo(config)#llb link health on
```

25.2.10.5 配置示例

配置目标：

内网发出的 DNS 查询请求可以均衡地分配到链路 ISP1 和 ISP2 上。

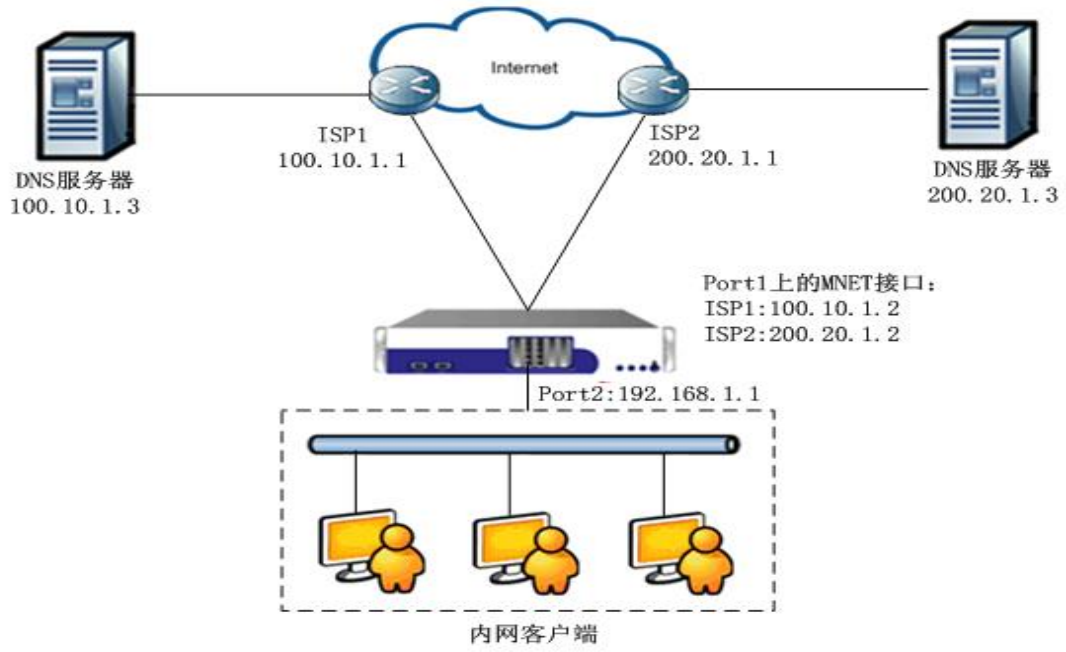

图25–1 DNS 代理

1. 为 LLB 链路 ISP1 和 ISP2 配置接口 IP 地址。

```
Demo(config)#mnet port1 ISP1
Demo(config)#mnet port1 ISP2
Demo(config)#ip address ISP1 100.10.1.2 255.255.255.0
Demo(config)#ip address ISP2 200.20.1.2 255.255.255.0
```

2. 为连接内网的接口配置 IP 地址。

```
Demo(config)#ip address port2 192.168.1.1 255.255.255.0
```

3. 配置 LLB 链路 ISP1 和 ISP2。

```
Demo(config)#llb link route ISP1 100.10.1.1 1 0.0.0.0 0Mbps
Demo(config)#llb link route ISP2 200.20.1.1 1 0.0.0.0 0Mbps
```


4. 为 DNS 代理功能配置 DNS 服务器。

```
Demo(config)#llb dnsproxy server server1 100.10.1.3 1
Demo(config)#llb dnsproxy server server2 200.20.1.3 1
```

5. 启用 DNS 代理功能。

```
Demo(config)#llb dnsproxy on
```

25.2.11 链路负载均衡支持 IPv6

设备的链路负载均衡功能提供了对 IPv6 地址的广泛支持，其策略路由、入向和出向链路负载均衡、链路健康检查和 IP 域功能都支持基于 IPv6 的网络环境。在策略路由中，源 IP、目的 IP 和网关 IP 都可以为 IPv6 地址，且策略路由所支持的 IP 域也可以配置 IPv6 地址，但同一个 IP 域表中只可配置 IPv4 或 IPv6 地址。对于出向链路负载均衡，仅基于路由的链路负载均衡可以支持 IPv6 地址，而基于 NAT 的链路负载均衡不能支持。

25.2.12 链路负载均衡支持 IP 地址组

IP 地址组功能允许管理员将出向链路相同的多个地址段添加到一个 IP 地址组中。为该 IP 地址组配置一条路由后，该路由便对 IP 地址组内的所有地址段生效。

对于 IP 地址段变化的情况，管理员也只需对于 IP 地址组进行管理，而无需关心具体的路由配置，从而简化配置，提高了易用性。

IP 地址组功能允许管理员为 Eroute 关联 IP 地址组。当修改 IP 地址组内的地址段时，对应的 Eroute 规则会立即生效。系统最多支持 1024 个 IP 地址组。

- 对于关联了地址组的 Eroute，按优先级从高到底排序。如果优先级相同，按配置的先后排序；如果优先级和五元组都相同，按照 LLB 算法进行匹配。
- 如果关联了地址组的 Eroute 和非关联地址组的 Eroute 优先级相同，非关联地址组的 Eroute 优先级高。
- IP 地址组功能允许管理员为基于源和基于目的的网络地址转换（NAT）关联 IP 地址组。当修改 IP 地址组内的地址段时，对应的 NAT 规则会立即生效。
- NAT 规则的匹配顺序为：静态 NAT>基于目的 NAT（未关联 IP 地址组）>基于源的 NAT（未关联 IP 地址组）>基于目的 NAT（关联 IP 地址组）>基于源的 NAT（关联 IP 地址组）。
- 对于关联了 IP 地址组的 NAT 规则，按照 IP 地址组关联的顺序进行排序。

25.3 链路负载均衡配置示例

25.3.1 出口链路负载均衡配置（一台设备）

在这个配置实例中，设备将要在网络中为两个 ISP 承担起出口链路负载均衡的任务。

如果这台设备停止工作，那么整个网络的通信就会中断。由于存在这种风险，我们推荐您按照“出口链路负载均衡配置（两台设备）”小节的方法在网络中配置两台设备。

25.3.1.1 配置向导

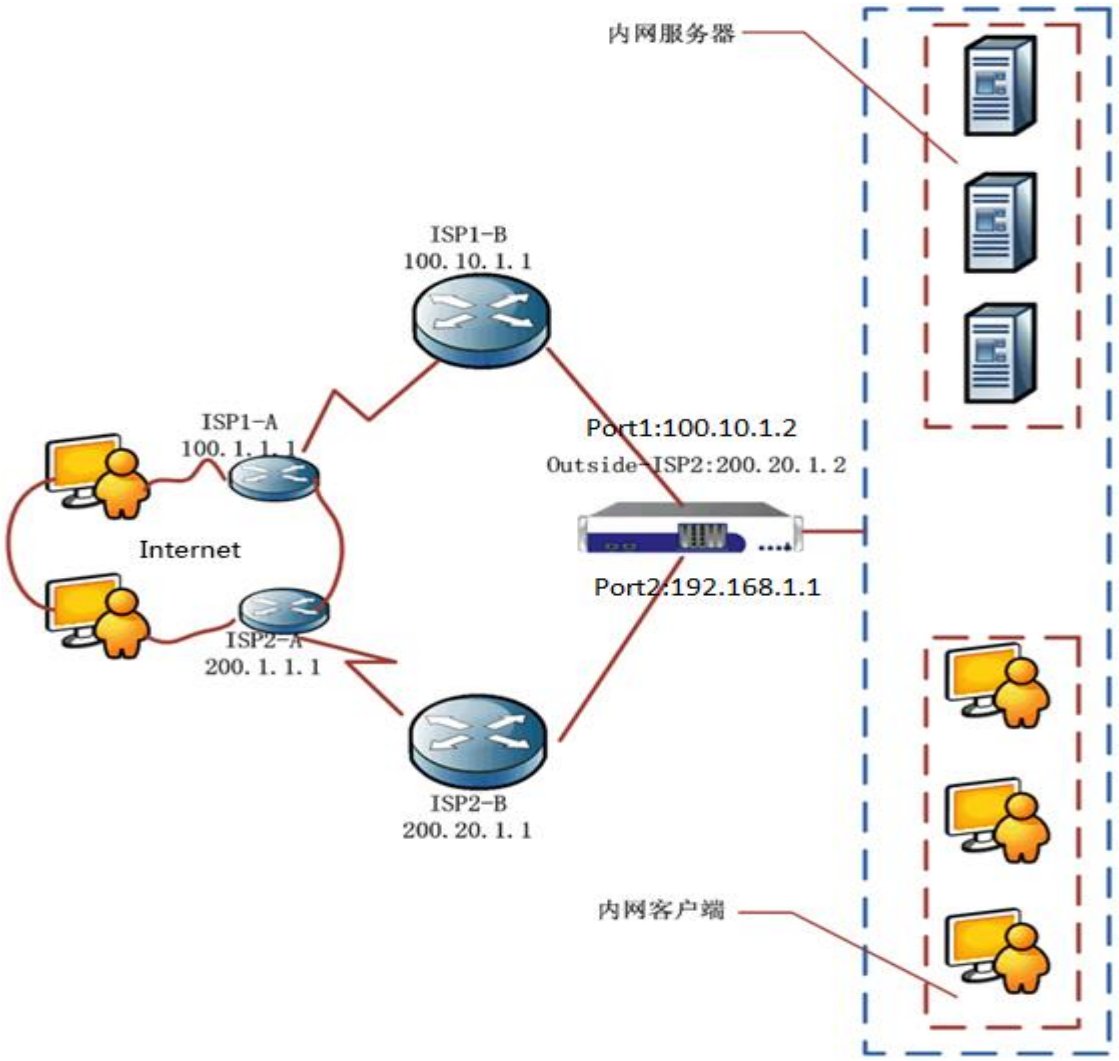

图25–2 出口链路负载均衡（一台设备）

下面列出了配置出口链路负载均衡所需的命令，相关命令的描述信息，请参考命令行使用手册。

表25–2 设备出口链路负载均衡配置命令

配置操作	命令行
配置接口 IP 地址	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix} [overlap]*
配置 MNET	**mnet** *{system_ifname|bond_ifname} <user_interface_name>*
配置链路负载均衡健康检查	**llb link route** *<link_name> <route_ip> [weight] [hc_srcip] [bandwidth_threshold]* **llb link health {on|off}** **llb link health checker icmp** *<link_name> <host> [hc_interval] [timeout] [hc_up] [hc_down]*
配置 LLB 远程站点可访问性检查	**llb rsite checker** *<checker_name> <tcp|icmp> <port> [interval] [rsite_up] [rsite_down]* **llb rsite net** *<rsite_name> <network_ip> <netmask|prefix> <checker_name> [check_ip]* **llb rsite health {on|off}**
配置出口链路负载均衡算法	**llb method outbound {rr|wrr|sr|hi|hip|lb|lc}** **llb method outbound dd** *[netmask] [prefix] [detecting_mode]*
配置 Eroute 和管理链路带宽	**ip eroute** *<name> <priority> <srcip> {srcmask|prefix} <srcport> <dsthost> {dstmask|prefix} <dstport> <protocol> <gateway_ip> [weight]* **llb link route** *<link_name> <route_ip> [weight] [hc_srcip] [bandwidth_threshold]* **llb link bw_priority** *<priority>*
配置 NAT	**nat port** *{pool_name|vip} <source_ip> {netmask|prefix} [timeout] [gateway] [description]*
启用 IPflow 和 RTS	**ip ipflow {on|off}** **ip rts off** **ip rts on** *[all|gateway]*

25.3.1.2 配置示例

1. 配置接口 IP 地址。

Port1 接口需要拥有一个属于 ISP1 的 IP 地址，如果需要配置附加的 IP 地址到 Port1 接口，就必须定义和配置 MNET，例如：创建一个名为“outside_isp2”的 MNET，并且把属于 ISP2 的 IP 分配给它。

```
Demo(config)#ip address port1 100.10.1.2 255.255.255.0
Demo(config)#mnet port1 outside_isp2
Demo(config)#ip address outside_isp2 200.20.1.2 255.255.255.0
```

现在，可以配置 Port2 接口的 IP 地址了。

```
Demo(config)#ip address port2 192.168.1.1 255.255.255.0
```

2. 配置链路。

配置链路 ISP1：

```
Demo(config)#llb link route ISP1 100.10.1.1
```

配置链路 ISP2：

```
Demo(config)#llb link route ISP2 200.20.1.1
```

3. 配置链路负载均衡健康检查（可选）。

ISP 的链路健康检查是检查从本地的 ISP 路由器（即设备主机）到 ISP 远端路由器之间是否连通。

配置链路 ISP1 的健康检查：

```
Demo(config)#llb link health checker icmp ISP1 "100.10.1.1" 10 5 3 3
```

这里支持同时配置多条链路的健康检查。假设 100.1.1.2，100.1.1.3，100.1.1.4 是链路 ISP1 的另外三个远端路由器。

```
Demo(config)#llb link health checker icmp ISP1 "100.1.1.2" 10 5 3 3
Demo(config)#llb link health checker icmp ISP1 "100.1.1.3" 10 5 3 3
Demo(config)#llb link health checker icmp ISP1 "100.1.1.4" 10 5 3 3
```

只有链路 ISP1 的所有的健康检查为不通时，才会认为链路 ISP1 是不通的。

如果发现某个 ISP 的链路的不稳定，假设是链路 ISP1，那么可以通过以下命令手动禁用链路 ISP1， 执行命令 **llb link disable ISP1** 可以禁用链路 ISP1，这样所有的出口数据流都不会从链路 ISP1 流出。重新启用链路，执行命令 **llb link enable ISP1**。

配置链路 ISP2 的健康检查：

```
Demo(config)#llb link health checker icmp ISP2 "200.20.1.1" 10 5 3 3
```

这里，我们还需要启用健康检查：

```
Demo(config)#llb link health on
```

4. 配置 LLB 远程站点可访问性检查（可选）。

LLB 远程站点可访问性检查用于检查本地设备是否能够访问远程站点。配置 LLB 远程站点可访问性检查。

```
Demo(config)#llb rsite checker TCP1 tcp 80 15 3 3
```

将已配置的 LLB 远程站点可访问性坚持关联到远程站点。


```
Demo(config)#llb rsite net Beijing 10.10.1.0 255.255.255.0 TCP1 10.10.1.8
```

启用 LLB 远程站点可访问性检查。

```
Demo(config)#llb rsite health on
```

5. 配置出口链路负载均衡算法（可选）。

支持以下出口链路负载均衡算法：

- 轮询算法（rr）
- 加权轮询算法（wrr）
- 最短响应时间算法（sr）
- 动态探测算法（dd）
- 哈希 IP 算法（hi）
- 哈希 IP 和端口（hip）
- 最小带宽使用率（lb）
- 最小连接数（lc）

默认的算法是轮询算法。

如果希望使用加权轮询算法，可以使用下面的命令：

```
Demo(config)#llb method outbound wrr
```

6. 配置 Eroute 和管理链路带宽。

为使不同的流量通过不同链路，配置如下 Eroute。

```
Demo(config)#ip eroute "er1" 1600 192.168.1.0 255.255.255.0 0 0.0.0.0 0.0.0.0 0 any
100.10.1.1 1
Demo(config)#ip eroute "er2" 1400 192.168.1.0 255.255.255.0 0 0.0.0.0 0.0.0.0 0 any
200.20.1.1 1
```

为使未匹配如上 Eroute 配置的流量通过链路 ISP1，配置如下 Eroute。

```
Demo(config)#ip eroute "er3" 1001 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 0 any 100.10.1.1 1
```

如果有必要，请按如下方式更新 LLB 链路的带宽限制。

```
Demo(config)#llb link route ISP1 100.10.1.1 1 100.10.1.2 500Mbps
Demo(config)#llb link route ISP2 200.20.1.1 2 200.20.1.2 300Mbps
```

用户可以配置链路带宽的优先级，用于确定配置的链路带宽限制是否对相应的 LLB 链路生效。

```
Demo(config)#llb link bw_priority 1500
```

因为 Eroute“er1”的优先级高于带宽的优先级，所以该 Eroute 指定的网关不受 ISP1 的链路带宽限制。反之，Eroute“er2”指定的网关受 ISP2 的链路带宽限制。

7. 为出口链路负载均衡配置 NAT。

假设根据负载均衡算法为一个新的会话选中了某个 ISP，那么与这个 ISP VIP 相对应的 NAT 规则必须已经被配置，因为出口数据流必须使用这个配置。

ISP1 对应的 NAT 规则：

```
Demo(config)#nat port 100.10.1.10 192.168.1.0 255.255.255.0
```

ISP2 对应的 NAT 规则：

```
Demo(config)#nat port 200.20.1.10 192.168.1.0 255.255.255.0
```

8. 其他必须的配置。

执行以下命令保证来自同一连接的数据报会使用同一条 NAT 规则被分配到同一条链路上。默认情况下，系统禁用 IPflow 功能。

```
Demo(config)#ip ipflow on
```

为了保证一个回应数据报（比如 ICMP 回应包）被分配到它对应的请求数据报（比如 ICMP 请求包）所在的链路上，RTS（沿原路返回）功能必须启用。默认情况下，系统禁用 RTS 功能。

```
Demo(config)#ip rts on
```

25.3.2 出口链路负载均衡配置（两台设备）

在这个配置实例中，两台设备将要在网络中为两个 ISP 承担起出口链路负载均衡的任务。这种配置方法可以实现物理冗余，只有当两台设备同时故障时，网络才会不可用。因此我们推荐这种配置方法。

25.3.2.1 配置向导

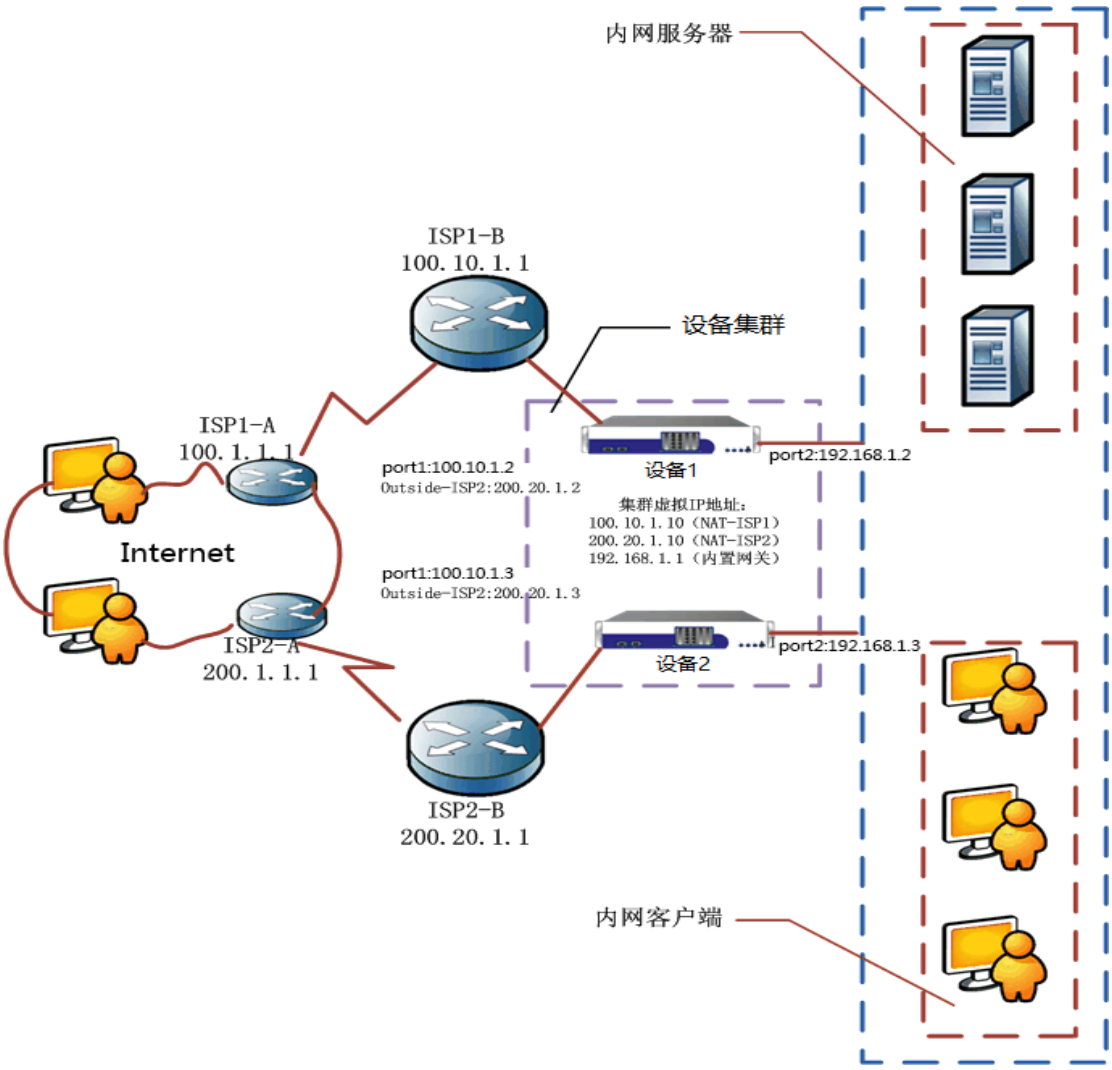

图25–3 出口链路负载均衡（两台设备）

下面列出了配置出口链路负载均衡所需的命令，相关命令的描述信息，请参考命令行使用手册。

表25–3 设备出口链路负载均衡配置命令

配置操作	命令行
配置接口 IP 地址	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix} [overlap]*
配置 MNET	**mnet** *{system_ifname|bond_ifname} <user_interface_name>*
配置一个集群虚拟路由	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]* **cluster virtual ifname** *<interface_name> <cluster_id>* **cluster virtual vip** *<interface_name> <cluster_id> <vip>* **cluster virtual priority** *<interface_name> <cluster_id> <priority> [synconfig_peer_name]*

配置操作	命令行
配置链路负载均衡健康检查	**llb link route** *<link_name> <route_ip> [weight] [hc_srcip] [bandwidth_threshold]* **llb link health {on|off}**
配置 LLB 远程站点可访问性检查	**llb rsite checker** *<checker_name> <tcp|icmp> <port> [interval] [rsite_up] [rsite_down]* **llb rsite net** *<rsite_name> <network_ip> <netmask|prefix> <checker_name> [check_ip]* **llb rsite health {on|off}**
为 NAT 流量配置集群虚拟 IP 地址	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]* **cluster virtual ifname** *<interface_name> <cluster_id>* **cluster virtual vip** *<interface_name> <cluster_id> <vip>* **cluster virtual priority** *<interface_name> <cluster_id> <priority> [synconfig_peer_name]*
配置 Eroute 和管理链路带宽	**ip eroute** *<name> <priority> <srcip> {srcmask|prefix} <srcport> <dsthost> {dstmask|prefix} <dstport> <proto> <gatewayip> [weight]* **llb link route** *<link_name> <route_ip> [weight] [hc_srcip] [bandwidth_threshold]* **llb link bw_priority** *<priority>*
配置 NAT	**nat port** *{pool_name|vip} <source_ip> {netmask|prefix} [timeout] [gateway] [description]*
启用 IPflow 和 RTS	**ip ipflow {on|off}** **ip rts off** **ip rts on** *[all|gateway]*

25.3.2.2 配置示例

按照下面的步骤来使用设备集群配置出口链路负载均衡。为了增强网络的冗余能力，我们增加了一台设备，因此需要做一些额外的配置。这个示例是以理解前一节所叙述的内容为基础的。一些附加的配置，如链路负载均衡的默认算法为轮询、默认失效时间等于 60 秒这些默认设置，将不会在下面的示例中进行提示。

1. 配置接口 IP 地址。

我们需要为我们的两个设备配置 IP 地址。相同的 MNET 名称也许会同时在两个设备的设置中使用。

(设备 1)，Port1 和 Port2 接口的配置：

```
Demo1(config)#ip address port1 100.10.1.2 255.255.255.0
Demo1(config)#mnet port1 outside_isp2
Demo1(config)#ip address outside_isp2 200.20.1.2 255.255.255.0
Demo1(config)#ip address port2 192.168.1.2 255.255.255.0
```

(设备 2)，Port1 和 Port2 接口的配置：

```
Demo2(config)#ip address port1 100.10.1.3 255.255.255.0
Demo2(config)#mnet port1 outside_isp2
Demo2(config)#ip address outside_isp2 200.20.1.3 255.255.255.0
Demo2(config)#ip address port2 192.168.1.3 255.255.255.0
```

2. 配置出口链路所需要的虚拟路由集群。

出口链路上的数据必须转发到设备上的一个地址，为了防止网关的物理地址失效，我们需要配置一个虚拟集群地址。

设备 1：将这个设备配置为虚拟路由集群的主控设备，以便它能够处理出口链路数据。因此我们需要将它的优先级配置为比第二台设备的优先级高。

```
Demo1(config)#cluster virtual ifname port2 1
Demo1(config)#cluster virtual vip port2 1 192.168.1.1
Demo1(config)#cluster virtual priority port2 1 200
Demo1(config)#cluster virtual on 1 port2
```

设备 2：将第二台设备配置为虚拟路由集群中的备份设备，以便在主控设备失效的时候，这台设备能够立即接替它来工作。因此我们需要为这台设备配置一个较低的优先级。

```
Demo2(config)#cluster virtual ifname port2 1
Demo2(config)#cluster virtual vip port2 1 192.168.1.1
Demo2(config)#cluster virtual priority port2 1 100
Demo2(config)#cluster virtual on 1 port2
```

3. 配置链路。

两台设备需要进行同样的设置。

```
Demo1(config)#llb link route ISP1 100.10.1.1
Demo1(config)#llb link route ISP2 200.20.1.1

Demo2(config)#llb link route ISP1 100.10.1.1 1
Demo2(config)#llb link route ISP2 200.20.1.1 2
```

我们还可以为链路配置健康检查来监控网络状态，并且启用健康检查。

```
Demo1(config)#llb link health checker icmp ISP1 "100.10.1.1" 10 5 3 3
Demo1(config)#llb link health checker icmp ISP2 "200.20.1.1" 10 5 3 3
Demo1(config)#llb link health on

Demo2(config)#llb link health checker icmp ISP1 "100.10.1.1" 10 5 3 3
Demo2(config)#llb link health checker icmp ISP2 "200.20.1.1" 10 5 3 3
Demo2(config)#llb link health on
```

还可以配置远程站点可访问检查来监控远程站点的可访问性。

```
Demo1(config)#llb rsite checker TCP1 tcp 80 15 3 3
Demo1(config)#llb rsite net Beijing 10.10.1.0 255.255.255.0 TCP1 10.10.1.8
Demo1(config)#llb rsite health on

Demo2(config)#llb rsite checker TCP1 tcp 80 15 3 3
Demo2(config)#llb rsite net Beijing 10.10.1.0 255.255.255.0 TCP1 10.10.1.8
Demo2(config)#llb rsite health on
```

4. 配置 NAT 地址转换所用的集群虚拟 IP 地址。

设备 1：配置到每个 ISP 的地址转换虚拟 IP 地址，并且配置一个比设备 2 设备高的优先级。

```
Demo1(config)#cluster virtual ifname port1 1
Demo1(config)#cluster virtual vip port1 1 100.10.1.10
Demo1(config)#cluster virtual prio port1 1 200
Demo1(config)#cluster virtual on 1 port1
Demo1(config)#cluster virtual ifname outside-isp2 1
Demo1(config)#cluster virtual vip outside-isp2 1 200.20.1.10
Demo1(config)#cluster virtual prio outside-isp2 1 200
Demo1(config)#cluster virtual on 1 outside-isp2
```

设备 2：配置到每个 ISP 的地址转换虚拟 IP 地址，并且配置一个比设备 1 设备低的优先级。

```
Demo2(config)#cluster virtual ifname port1 1
Demo2(config)#cluster virtual vip port1 1 100.10.1.10
Demo2(config)#cluster virtual prio port1 1 100
Demo2(config)#cluster virtual on 1 port1
Demo2(config)#cluster virtual ifname outside-isp2 1
Demo2(config)#cluster virtual vip outside-isp2 1 200.20.1.10
Demo2(config)#cluster virtual prio outside-isp2 1 100
Demo2(config)#cluster virtual on 1 outside-isp2
```

5. 配置 Eroute 和管理链路带宽。

为使不同的流量通过不同链路，配置如下 Eroute。

```
Demo1(config)#ip eroute "er1" 1600 192.168.1.0 255.255.255.0 0 0.0.0.0 0.0.0.0 0 any
100.10.1.1 1
Demo1(config)#ip eroute "er2" 1400 192.168.1.0 255.255.255.0 0 0.0.0.0 0.0.0.0 0 any
200.20.1.1 1
Demo2(config)#ip eroute "er1" 1600 192.168.1.0 255.255.255.0 0 0.0.0.0 0.0.0.0 0 any
100.10.1.1 1
```

```
Demo2(config)#ip eroute "er2" 1400 192.168.1.0 255.255.255.0 0 0.0.0.0 0.0.0.0 0 any
200.20.1.1 1
```

为使未匹配如上 Eroute 配置的流量通过链路 ISP1，配置如下 Eroute。

```
Demo1(config)#ip eroute "er3" 1001 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 0 any 100.10.1.1 1
Demo2(config)#ip eroute "er3" 1001 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 0 any 100.10.1.1 1
```

如果有必要，请按如下方式更新 LLB 链路的带宽限制。

```
Demo1(config)#llb link route ISP1 100.10.1.1 1 100.10.1.2 500Mbps
Demo1(config)#llb link route ISP2 200.20.1.1 2 200.20.1.2 300Mbps
Demo2(config)#llb link route ISP1 100.10.1.1 1 100.10.1.2 500Mbps
Demo2(config)#llb link route ISP2 200.20.1.1 2 200.20.1.2 300Mbps
```

用户可以配置链路带宽的优先级，用于确定配置的链路带宽限制是否对相应的 LLB 链路生效。

```
Demo1(config)#llb link bw_priority 1500
Demo2(config)#llb link bw_priority 1500
```

因为 Eroute“er1”的优先级高于带宽的优先级，所以该 Eroute 指定的网关不受 ISP1 的链路带宽限制。反之，Eroute“er2”指定的网关受 ISP2 的链路带宽限制。

6. 配置出口链路负载均衡会话的地址转换。

```
Demo1(config)#nat port 100.10.1.10 192.168.1.0 255.255.255.0
Demo1(config)#nat port 200.20.1.10 192.168.1.0 255.255.255.0

Demo2(config)#nat port 100.10.1.10 192.168.1.0 255.255.255.0
Demo2(config)#nat port 200.20.1.10 192.168.1.0 255.255.255.0
```

7. 其他必须的配置。

执行以下命令保证来自同一连接的数据报会使用同一条 NAT 规则被分配到同一条链路上。默认情况下，系统禁用 IPflow 功能。

```
Demo(config)#ip ipflow on
```

为了保证一个响应数据报（例如 ICMP 响应包）被分配到它对应的请求数据报（例如 ICMP 请求包）所在的链路上，RTS 功能必须被启用。预设情况下，该功能是禁用的。

```
Demo(config)#ip rts on
```

25.3.3 入口链路负载均衡配置

在这个示例中，我们将介绍使用单个设备来实现入口链路负载均衡的方法。

25.3.3.1 配置向导

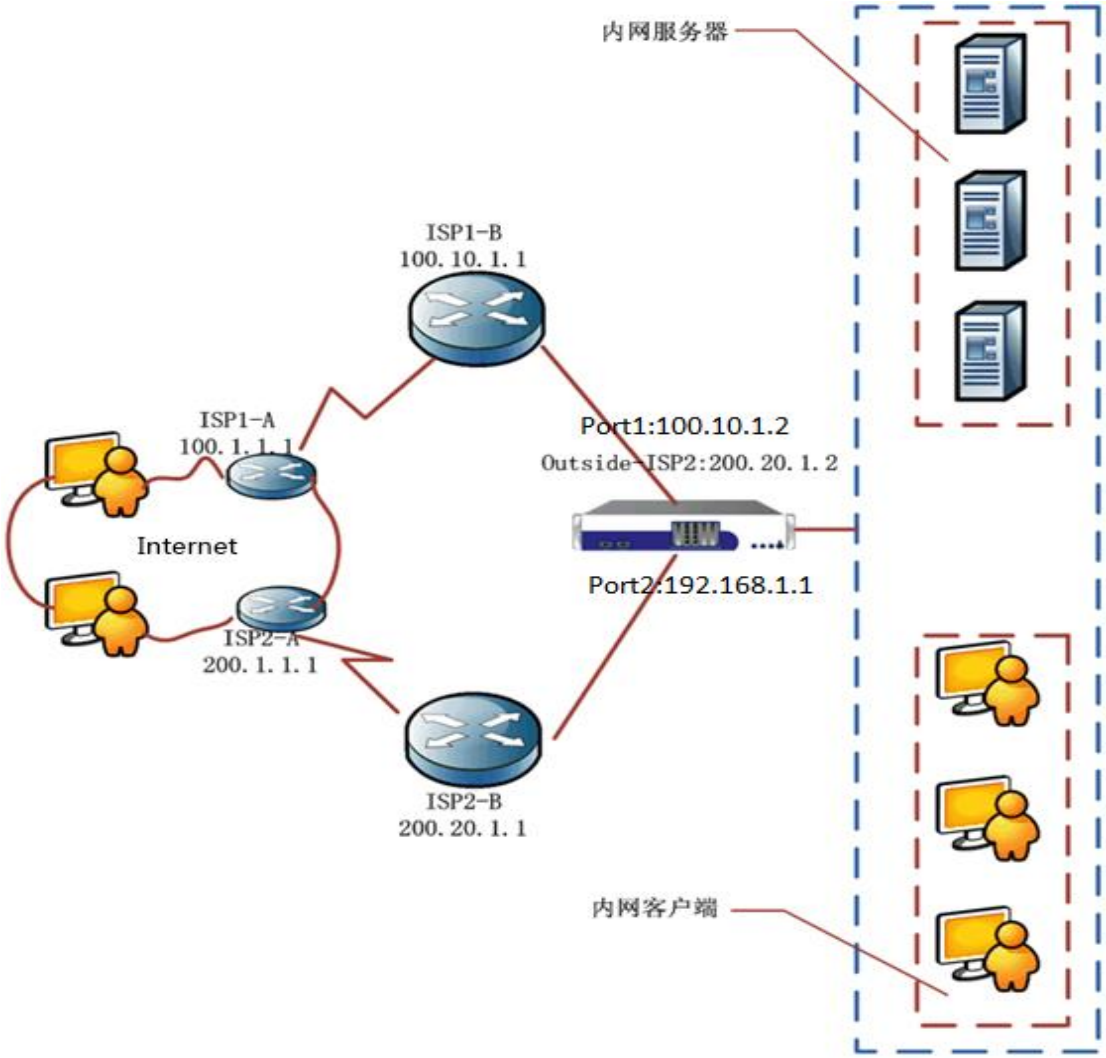

图25–4 入口链路负载均衡

下面列出了配置入口链路负载均衡所需的命令，相关命令的描述信息，请参考命令行使用手册。

表25–4 设备入口链路负载均衡配置命令

配置操作	命令行
配置接口 IP 地址	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix} [overlap]*
配置 MNET	**mnet** *{system_ifname|bond_ifname} <user_interface_name>*
配置链路负载均衡健康检查	**llb link route** *<link_name> <route_ip> [weight] [hc_srcip] [bandwidth_threshold]* **llb link health {on|off}**

配置操作	命令行
配置服务器负载均衡	**slb real http** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb virtual http** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb policy static** *<virtual_service> <real_service>*
配置链路负载均衡 DNS 和生存周期	**llb dns host** *<host_name> <ip> [weight] [port] [link_name]* **llb dns ttl** *<host_name> [seconds]*
配置负载均衡算法	**llb method inbound {rr|wrr|proximity}**
启用 IPflow 和 RTS	**ip ipflow {on|off}** **ip rts off** **ip rts on** *[all|gateway]*

25.3.3.2 配置示例

1. 配置接口 IP 地址。

Port1 接口需要拥有一个属于 ISP1 的 IP 地址，如果需要配置附加的 IP 地址到 Port1 接口，就必须定义和配置 MNET，例如：创建一个名为“outside_isp2”的 MNET，并且把属于 ISP2 的 IP 分配给它。

```
Demo(config)#ip address port1 100.10.1.2 255.255.255.0
Demo(config)#mnet port1 outside_isp2
Demo(config)#ip address outside_isp2 200.20.1.2 255.255.255.0
```

现在，可以配置 Port2 接口的 IP 地址了。

```
Demo(config)#ip address port2 192.168.1.1 255.255.255.0
```

2. 配置链路。

配置链路 ISP1：

```
Demo(config)#llb link route ISP1 100.10.1.1
```

配置链路 ISP2：

```
Demo(config)#llb link route ISP2 200.20.1.1
```

我们还可以为链路配置健康检查来监控健康网络状态。ISP 的链路健康检查是检查从本地的 ISP 路由器（即设备主机）到 ISP 远端路由器之间是否连通。

配置链路 ISP1 的健康检查：

```
Demo(config)#llb link health checker icmp ISP1 "100.10.1.1" 10 5 3 3
```

配置链路 ISP2 的健康检查：

```
Demo(config)#llb link health checker icmp ISP2 "200.20.1.1" 10 5 3 3
```

这里，我们还需要启用健康检查：

```
Demo(config)#llb link health on
```

3. 配置服务器负载均衡。

```
Demo(config)#slb virtual http vip1 100.10.1.10
Demo(config)#slb virtual http vip2 200.20.1.10
Demo(config)#slb real http server1 192.168.1.100
Demo(config)#slb policy static vip1 server1
Demo(config)#slb policy static vip2 server1
```

4. 为入口链路负载均衡配置 DNS 主机及生存周期（TTL）。

```
Demo(config)#llb dns host llb.aaaccc.com 100.10.1.10 2
Demo(config)#llb dns host llb.aaaccc.com 200.20.1.10 1
Demo(config)#llb dns ttl llb.aaaccc.com 60
```

5. 配置入口链路负载均衡算法。

```
Demo(config)#llb method inbound wrr
```

注意：如果要使用 proximity 算法，需要先进行“**ip eroute**”的相关配置。

6. 其他必须的配置。

执行以下命令保证来自同一连接的数据报会使用同一条 NAT 规则被分配到同一条链路上。默认情况下，系统禁用 IPflow 功能。

```
Demo(config)#ip ipflow on
```

为了保证一个回应数据报（比如 ICMP 回应包）被分配到它对应的请求数据报（比如 ICMP 请求包）所在的链路上，RTS（沿原路返回）功能必须启用。默认情况下，系统禁用 RTS 功能。

```
Demo(config)#ip rts on
```

第26章 全局服务器负载均衡（GSLB）

26.1 概述

全局服务器负载均衡（Global Server Load Balancing，GSLB）通过返回不同的DNS 解析结果将网络流量分配到部署在不同位置上提供相同服务的服务器上。设备采用智能 DNS（Smart DNS，SDNS）技术实现 GSLB，在返回 DNS 解析结果时做出智能决策，以实现如下目的：

- 高访问效率：SDNS 根据用户本地 DNS 服务器的地理位置和网络就近性，为其返回离用户“逻辑上”最近或响应最快的服务器的 IP 地址，提高了用户上网访问效率。
- 灾备：当 SDNS 检测到一台服务器故障时，将挑选其它可用服务器的 IP 地址返回给用户，这样就能保证用户在一台服务器故障时仍能正常访问业务，增加业务的连续性和高可用性。当 SDNS 检测到服务器恢复正常后，还可以将流量切换回该服务器。
- 负载均衡：SDNS 可以将流量分配到多个可用的服务器上。

以下章节将介绍 SDNS 的功能原理以及配置示例。

26.2 功能原理

26.2.1 SDNS 域名解析

SDNS 可以为域名提供智能和权威的解析服务。SDNS 的域名解析流程如下图所示。

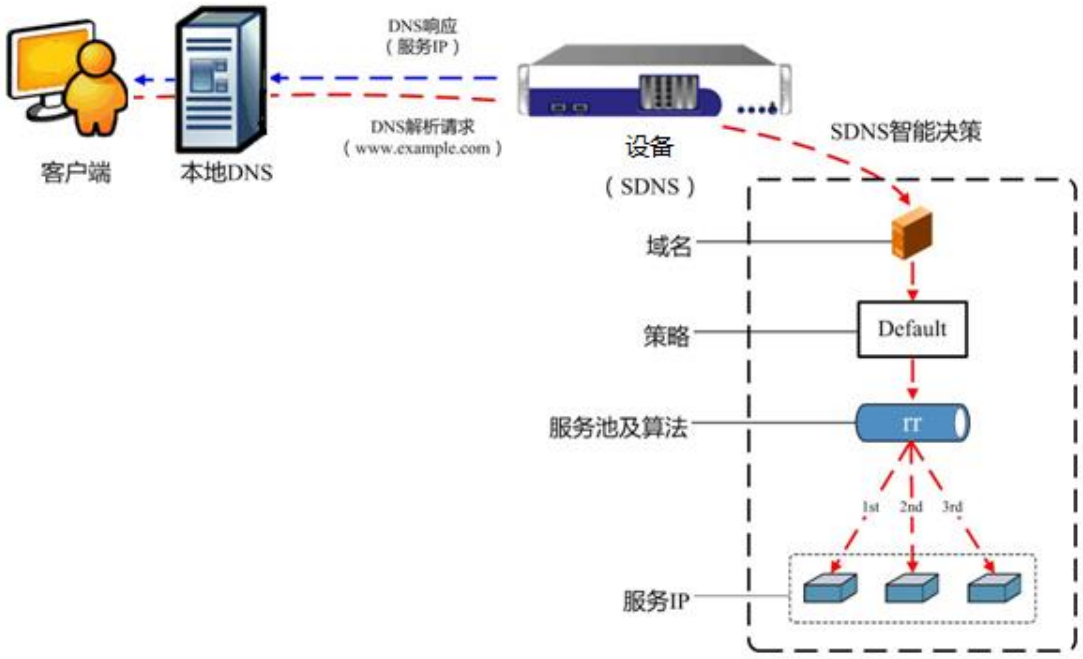

图26–1 SDNS 域名解析

SDNS 的域名解析流程详细介绍如下：

- 客户端发送 DNS 解析请求到本地 DNS 服务器以解析域名“www.example.com”。
- 经过一系列查询，本地 DNS 服务器得知 SDNS 为该域名的授权 DNS 服务器，然后将 DNS 解析请求转发给 SDNS 处理。
- SDNS 做出智能决策，挑选健康可用的服务 IP 返回给本地 DNS 服务器。
- 本地 DNS 服务器将解析结果返回给客户端。

SDNS 智能决策的流程详细介绍如下：

- SDNS 根据 DNS 解析请求的域名找到命中的 SDNS 策略。
- SDNS 根据命中的策略找到命中的 SDNS 服务池。
- SDNS 根据命中的 SDNS 服务池的算法确定命中的 SDNS 服务 IP。
- SDNS 将命中的 SDNS 服务 IP 返回给本地 DNS 服务器。

26.2.2 基本概念

26.2.2.1 SDNS 域名

SDNS 域名是 SDNS 提供域名解析服务的基本单位。在 SDNS 中，可解析的域名将被定义为 SDNS 域名，系统中最多可以定义 16,000 个 SDNS 域名。

配置示例：

```
Demo(config)#sdns host name "www.xyz.com" 60
Demo(config)#sdns host name "www.abc.com" 60
```

26.2.2.2 SDNS 监听 IP 地址

默认情况下，系统在设备的所有 IP 地址上提供 SDNS 服务，包括虚拟 IP 地址、接口 IP 地址和 HA 浮动 IP 地址，处理这些 IP 地址上收到的所有目的端口号为 53 的 DNS 请求报文。

管理员可以使用“**sdns listener**”命令配置 SDNS 监听 IP 地址，使系统只在指定的监听 IP 地址上提供 SDNS 服务。配置监听 IP 地址后，DNS 请求报文的目的 IP 地址必须匹配任意一个监听 IP 地址，否则会被拒绝访问。

26.2.2.3 SDNS 服务 IP 和服务池

在 SDNS 中，同一个 SDNS 域名可以被解析成多个地址。SDNS 允许将多个解析 IP 地址添加到一个地址池，称为“服务池”，而其中的每一个 IP 地址称为“服务 IP”。SDNS 解析域名时，根据服务池的算法确定返回给本地 DNS 服务器的 IP 地址。

服务 IP 支持 IPv4 和 IPv6，一个 SDNS 服务池的服务 IP 可以全为 IPv4 或者 IPv6 地址，但该服务池不能同时包含 IPv4 和 IPv6 服务 IP。

配置示例：

```
Demo(config)#sdns pool name "pool1" 1
Demo(config)#sdns pool name "pool2" 1
Demo(config)#sdns service ip "service1" 10.8.6.201 0
Demo(config)#sdns service ip "service2" 10.8.6.202 0
Demo(config)#sdns service ip "service3" 2012:1086::203 0
Demo(config)#sdns service ip "service4" 2012:1086::204 0
Demo(config)#sdns pool service "pool1" "service1"
Demo(config)#sdns pool service "pool1" "service2"
Demo(config)#sdns pool service "pool2" "service3"
Demo(config)#sdns pool service "pool2" "service4"
```

26.2.2.3.1 SDNS 服务池算法

SDNS 服务池算法决定了服务池的哪些服务 IP 被选中返回给本地 DNS 服务器。不同的服务池算法返回的服务 IP 不同。按照实现的效果，服务池算法可以分为：

- rr（Round Robin，轮询）
- grr（Global Round Robin，全局轮询）

- wrr（Weighted Round Robin，加权轮询）
- gwrr（Global Weighted Round Robin，全局加权轮询）
- ipo（IP Overflow，IP 优先）
- hi（Hash IP，哈希 IP）
- snmp（Simple Network Management Protocol）
- drop（丢弃）

下表描述了各种算法实现的效果。

表26–1 SDNS 服务池算法

算法	效果
rr	如果 SDNS 服务池中有 3 个 SDNS 服务 IP，参数“max_rr_count”的值为 1，并且采用轮询算法，则 SDNS 服务 IP 将按“1、2、3、1、2、3…”的顺序在 DNS 响应中被返回。该算法在多核 CPU 环境下可能会有误差。
grr	该算法与轮询算法原理相同。区别在于该算法在多核 CPU 环境下也可以严格运行。
wrr	加权轮询算法与轮询算法类似。区别在于每个 SDNS 服务都被赋予了一个权重值。只有当第一个 SDNS 服务 IP 被返回的次数达到其权重值时，第二个 SDNS 服务 IP 才会在 DNS 响应中被返回。该算法在多核 CPU 环境下可能会有误差。
gwrr	该算法与加权轮询算法原理相同。区别在于该算法在多核 CPU 环境下也可以严格加权轮询。
ipo	采用 IP 优先算法时，优先级最高的 SDNS 服务 IP 将总是在 DNS 响应中被返回。
hi	采用哈希 IP 算法时，SDNS 会根据 DNS 查询请求中的源 IP 地址计算哈希值。对于源 IP 地址相同的 DNS 请求，系统将返回同一个 SDNS 服务 IP。
snmp	采用 snmp 算法时，SDNS 会使用 SNMP 协议从每个服务 IP 收集特定 OID 的数据，根据指定的 SNMP 表达式计算每个 SDNS 服务 IP 的复合权重值（metric），并按照指定方式排序。排序最前的 SDNS 服务 IP 将在 DNS 响应中被返回。详细信息，参见 26.2.7.1.326.2.7.1.3 自定义健康检查请求/响应文件 为了能够支持复杂的健康检查，管理员需要通过导入文件的方式自定义健康检查请求和预期响应，并将健康检查模板与自定义健康检查请求和响应文件关联。这样，该模板的健康检查请求和预期响应会被替换为自定义健康检查请求和响应文件的内容。 系统支持为 HTTP/HTTPS 类型的健康检查模板关联自定义

	健康检查请求和响应文件。 配置示例:
	Demo(config)# **sdns monitor http "hc_http" "HEAD / HTTP/1.0\r\n\r\n" "200 OK" "up" 5 5 3 0.0.0.0 0 0.0.0.0 0.0.0.0** Demo(config)#**sdns health import request "request" "ftp://1.1.1.1/request.txt"** Demo(config)#**sdns health import response "response" "ftp://1.1.1.1/response.txt"** Demo(config)#**sdns health bind "hc_http" “request” “response”** Demo(config)#**sdns service ip "sv1" 10.8.6.201 80** Demo(config)#**sdns service monitor apply "sv1" "hc_http"**
	SNMP 数据收集。
drop	采用丢弃算法时，系统将直接丢弃 DNS 请求且不返回 DNS 响应。

注意：SDNS 服务池算法是针对单线程的，在多线程环境下，轮询算法和加权轮询算法的服务 IP 被命中次数会出现偏差。对于轮询算法，在单线程下，服务池中任意两个可用服务 IP 被命中的次数差值始终不超过一次，所以在多线程环境下，服务池中任意两个可用服务 IP 被命中的次数不超过线程数。

此外，SDNS 还支持为服务池设置两级服务池算法：一级（primary）算法和二级（secondary）算法。当使用一级算法没有挑选出可用的服务 IP 时，SDNS 才会使用二级算法去挑选可用的服务 IP。因此，一级算法必须配置，而二级算法为可选配置。

配置示例：

```
Demo(config)#sdns pool method primary "pool1" "ipo"
Demo(config)#sdns pool member priority "pool1" "service1" 1
Demo(config)#sdns pool member priority "pool1" "service2" 2
Demo(config)#sdns pool method secondary "pool1" "rr"

Demo(config)#sdns pool method primary "pool2" "wrr"
Demo(config)#sdns pool member weight "pool2" "service3" 3
Demo(config)#sdns pool member weight "pool2" "service4" 2
Demo(config)#sdns pool method secondary "pool2" "drop"
```

26.2.2.3.2 服务池自动失效切换

SDNS 为使用 ipo 算法的 SDNS 服务池提供自动失效切换功能。当服务池中最高优先级的服务 IP 变为不可用时，自动失效切换功能将当前选中的服务 IP 切换到该服务池中次高优先级的服务 IP。

配置示例：

```
Demo(config)#sdns pool failover "pool1" on
```

26.2.2.3.3 服务池抢占功能

SDNS 为使用 ipo 算法的 SDNS 服务池提供抢占功能。当服务池中最高优先级的服务 IP 从不可用变为可用时，抢占功能可以将该服务池当前选中的服务 IP 切换回到最高优先级的服务 IP。该功能默认是启用的。

配置示例：

```
Demo(config)#sdns pool preempt "pool1" on
```

26.2.2.3.4 服务池手动切换

SDNS 还为使用 ipo 算法的 SDNS 服务池提供手动切换方式，允许管理员将服务池当前选中的服务 IP 切换到该服务池中指定优先级的服务 IP。该功能默认是启用的。

配置示例：

```
Demo(config)#sdns pool ipo reset "pool1" 2
```

注意：

- 如果同时启用了自动失效切换和抢占功能，手动切换操作将失效。
- 手动切换操作不能当作配置被保存，在系统重启后失效。
- 当指定优先级的所有服务 IP 的状态变为“Down”时，DNS 解析失败。

26.2.2.4 SDNS 策略

SDNS 策略根据域名确定 SDNS 服务池。SDNS 支持三种类型策略：

- 区域策略：区域策略根据域名以及 DNS 解析请求所属的区域确定 SDNS 服务池。SDNS 允许为同一域名配置多个区域策略。
- 默认策略：默认策略根据域名确定 SDNS 服务池。当 SDNS 不能根据区域策略从对应的服务池挑选出可用的服务 IP 时，才使用默认策略找到对应 SDNS 服务池。
- 应急策略：应急策略根据域名确定 SDNS 应急池。当 SDNS 不能根据区域策略以及默认策略，从对应的服务池挑选出可用的服务 IP 时，SDNS 才使用应急策略找到对应的 SDNS 应急池。匹配应急策略后，无论对应的应急池中的服务 IP 是否可用，都会被返回。

SDNS 策略命中过程如下：

a. 当 DNS 解析请求的域名命中多个区域策略时，SDNS 将在优先级最高的策略对应的 SDNS 服务池中挑选服务 IP。

b. 如果不能在该 SDNS 服务池挑选出可用服务 IP，SDNS 将从默认策略对应的 SDNS 服务池中来挑选服务 IP。

c. 如果不能在默认策略对应的 SDNS 服务池中挑选出可用的服务 IP，SDNS 将从应急策略对应的 SDNS 应急池中来挑选服务 IP。

d. 如果仍不能挑选出可用的服务 IP，那么 SDNS 将会返回空 DNS 响应给本地 DNS 服务器。

关于 SDNS 区域策略的详细介绍，请参考章节 26.2.6 SDNS 区域策略。

26.2.3 SDNS CNAME 池

SDNS 还可以将域名解析为 CNAME（Canonical Name，别名）记录。SDNS CNAME 池中只能包含一个 CNAME 记录。当 SDNS 收到 CNAME 类型的 DNS 解析请求且该 CNAME 池被命中，那么 SDNS 将 CNAME 记录返回给本地 DNS 服务器。当 SDNS 收到 A 或 AAAA 类型的 DNS 解析请求且该 CNAME 池被命中时，那么 SDNS 将使用 CNAME 再次进行域名解析，然后将解析 IP 地址返回给本地 DNS 服务器。

配置示例：

```
Demo(config)#sdns service ip "web1" 10.8.6.10
Demo(config)#sdns service ip "web2" 10.8.6.20
Demo(config)#sdns pool name "pool_web" 1
Demo(config)#sdns pool method primary "pool_web" rr
Demo(config)#sdns pool service "pool_web" "web1"
Demo(config)#sdns pool service "pool_web" "web2"
Demo(config)#sdns pool cname "pool_cname" "www.web.example.com"
Demo(config)#sdns host name "www.example.com" 60
Demo(config)#sdns host name "www.web.example.com" 60
Demo(config)#sdns policy default "www.example.com" "pool_cname"
Demo(config)#sdns policy default "www.web.example.com" "pool_web"
```

完成如上配置后：

- 当 SDNS 收到“www.example.com”域名的 CNAME 类型的 DNS 解析请求时，将 CNAME“www.web.example.com”返回给本地 DNS 服务器。
- 当 SDNS 收到“www.example.com”域名的 A 或 AAAA 类型的 DNS 解析请求时，SDNS 找到域名“www.example.com”的 CNAME 记录，然后对别名“www.web.example.com”进行解析，最终将 SDNS 服务 IP“web1”或者“web2”返回给本地 DNS 服务器。

26.2.4 SDNS 应急池

SDNS 支持 SDNS 应急池。定义 SDNS 服务池时，系统会同时创建同名的应急池。系统支持为 SDNS 应急池手动添加 SDNS 服务（通过命令“**sdns pool resort service**”）。

对 SDNS 服务池进行操作将会影响到 SDNS 应急池。删除 SDNS 服务池同时会将 SDNS 应急池删除；为 SDNS 服务池添加或删除 SDNS 服务时会同时自动为 SDNS 应急池添加或删除 SDNS 服务。

SDNS 应急池只能与 SDNS 应急策略关联（通过命令“**sdns policy resort**”）。

配置示例：

```
Demo(config)#sdns host name www.abc.com 0
Demo(config)#sdns pool name pool1
Demo(config)#sdns policy resort www.abc.com pool1
Demo(config)#sdns service ip service1 192.168.83.11
Demo(config)#sdns monitor icmp m1
Demo(config)#sdns service monitor apply service1 m1
Demo(config)#sdns pool service pool1 service1
Demo(config)#show sdns pool resort service
sdns pool resort service "pool1" "service1"
```

26.2.5 SDNS 服务池回退

如果 SDNS 服务池根据池算法无法找到可用的服务 IP，SDNS 支持将该服务池处理的解析请求切换到另一个 SDNS 池进行处理，这样为 SDNS 服务池提供了失效切换机制。第二个 SDNS 池称为回退池。

一个 SDNS 服务池只能有一个回退池，但可以是多个其它服务池的回退池。多个 SDNS 服务池可以组成一个树形结构的回退方式，如下图所示。

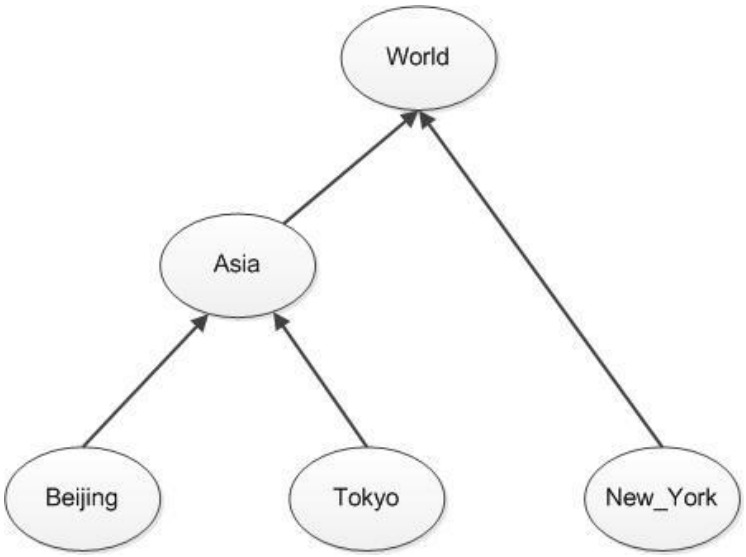

图26–2 服务池回退（树形）

配置示例：

```
Demo(config)#sdns pool name "Beijing" 1
Demo(config)#sdns pool name "Tokyo" 1
Demo(config)#sdns pool name "New_York" 1
Demo(config)#sdns pool name "Asia" 1
Demo(config)#sdns pool name "World" 1
Demo(config)#sdns pool fallback "Beijing" "Asia"
Demo(config)#sdns pool fallback "Tokyo" "Asia"
Demo(config)#sdns pool fallback "New_York" "World"
Demo(config)#sdns pool fallback "Asia" "World"
```

完成如上配置后：

- 服务池“Beijing”和“Tokyo”在不可用时将回退到服务池“Asia”；
- 服务池“New_York”在不可用时将回退到服务池“World”；
- 服务池“Asia”在不可用时将回退到服务池“World”；

注意：

- SDNS 不允许多个 SDNS 服务池组成环形回退方式。
- SDNS 服务池可以回退到 SDNS CNAME 池，但不允许 SDNS CNAME 池回退到 SDNS 服务池。

26.2.6 SDNS 区域策略

SDNS 从逻辑上将来自不同地方 DNS 解析请求划分在不同的区域，称为 SDNS 区域。管理员可以根据地理位置划分 SDNS 区域，例如 United States、Japan 和 China，也可以根据实际需要划分 SDNS 区域，例如网络服务运营商（ISP）。

管理员只能为一个域名配置一条默认策略，但可以为一个域名配置多条区域策略，将来自于不同区域的 DNS 解析请求关联到不同的 SDNS 服务池。

区域策略根据域名和 DNS 解析请求所属的区域确定 SDNS 服务池。可以为同一域名配置多个区域策略，不同的区域对应不同 SDNS 服务池。SDNS 就近性规则用来确定来自指定网段的 DNS 解析请求所属的区域。当收到 DNS 解析请求时，SDNS 采用最长前缀匹配的原则找到命中的就近性规则，进而确定 DNS 请求归属的区域。当收到的 DNS 解析请求的域名匹配多个区域策略时，SDNS 使用区域策略的优先级来确定 DNS 解析请求最终命中的区域策略。该域名的 DNS 解析请求将由最高优先级区域策略的 SDNS 服务池处理。

SDNS 允许管理员配置静态就近性规则，关于 SDNS 静态就近性规则的详细介绍，请参考章节 23.2.8 SDNS 静态就近性规则。

此外，SDNS 支持使用动态就近性探测系统（Dynamic Proximity System，DPS）生成的动态就近性规则。关于 DPS 的详细介绍，请参考章节 23.2.9 SDNS 静态就近性。

配置示例：

```
Demo(config)#sdns host name "www.xyz.com" 1
Demo(config)#sdns pool name "pool_beijing" 1
Demo(config)#sdns pool name "pool_tokyo" 1
Demo(config)#sdns pool name "pool_newyork" 1
Demo(config)#sdns region name "Beijing"
Demo(config)#sdns region name "Tokyo"
Demo(config)#sdns region name "New_York"
Demo(config)#sdns proximity 211.100.0.0 255.255.0.0 "Beijing"
Demo(config)#sdns proximity 210.10.0.0 255.255.0.0 "Tokyo"
Demo(config)#sdns proximity 216.100.0.0 255.255.0.0 "New_York"
Demo(config)#sdns policy region "policy_beijing" "www.xyz.com" "pool_beijing" "Beijing"
0
Demo(config)#sdns policy region "policy_tokyo" "www.xyz.com" "pool_tokyo" "Tokyo" 0
Demo(config)#sdns policy region "policy_newyork" "www.xyz.com" "pool_newyork"
"New_York" 0
```

完成如上配置后：

- 来自 211.100.0.0/16 网段的域名“www.xyz.com”的 DNS 解析请求将会命中区域策略“policy_beijing”；
- 来自 210.10.0.0/16 网段的域名“www.xyz.com”的 DNS 解析请求将会命中区域策略“policy_tokyo”；
- 来自 216.100.0.0/16 网段的域名“www.xyz.com”的 DNS 解析请求将会命中区域策略“policy_newyork”。

注意：系统现在支持 edns-client-subnet（ECS）选项，该选项携带了发起 DNS 查询的网络和可以缓存后续响应的网络的信息。通过支持 ECS，系统可以获取发起者的真实地址并将请求发给最优的服务器。ECS 与 SDNS 区域策略一起可以提供更好的解析服务，并改善用户体验。

26.2.7 SDNS Monitor

SDNS monitor 可以用于对 SDNS 服务进行健康检查或者向 SDNS 服务收集 SNMP 数据。要使用 SDNS monitor，管理员必须先定义健康检查模板或数据收集模板，然后将模板与一个 SDNS 服务或 SDNS 服务池关联起来生成 monitor 实例。系统将根据 monitor 实例执行健康检查或者数据收集。

SDNS 支持 ICMP、TCP、UDP、HTTP 和 HTTPS 类型的健康检查模板以及 SNMP 类型的数据收集模板。系统支持最多 1000 个健康检查和数据收集模板。健康检查模板支持 IPv4 和 IPv6 类型。

当模板与一个 SDNS 服务关联起来后，系统将生成一个 monitor 实例；当模板与一个 SDNS 服务池关联起来后，系统将为该服务池中的每个 SDNS 服务生成一个 monitor 实例。基于健康检查模板生成的 monitor 实例称为健康检查实例，基于 SNMP 数据收集模板生成的 monitor 实例称为 SNMP 数据收集实例。系统支持最多 5000 个 monitor 实例，最多 1024 个 SNMP 数据收集实例。

26.2.7.1 SDNS 健康检查

SDNS 提供健康检查功能，对 SDNS 服务池中的服务 IP 进行健康检查。当服务 IP 不可用时，SDNS 将只从该服务池中可用的服务 IP 中挑选服务 IP 来响应 DNS 解析请求。这样确保了返回给本地 DNS 的 IP 地址都是健康可用的，增加业务的连续性和高可用性。

26.2.7.1.1 健康检查类型

SDNS 提供以下几种类型的健康检查方法：

- **ICMP 健康检查**

ICMP 健康检查通过发送 ICMP Echo Request（ping）给服务 IP 来判断服务 IP 的可用性。如果从服务 IP 收到 ICMP Echo Response，则认为该服务 IP 可用；否则，认为该服务 IP 不可用。

- **TCP 健康检查**

TCP 健康检查通过尝试与服务 IP 建立一个 TCP 连接来判断服务 IP 的可用性。如果连接建立成功，则认为该服务 IP 可用；否则，认为该服务 IP 不可用。

- **UDP 健康检查**

1. 系统向服务 IP 发送 ICMP Echo 请求。如果从服务 IP 收到 ICMP Echo 响应，则系统会进入到下一步；否则，系统将认为该服务 IP 不可用。
2. 系统向服务 IP 的指定端口发送 UDP 健康检查报文。如果系统成功发送 UDP 健康检查报文，且未收到 ICMP 不可达响应，系统将认为该服务 IP 可用；否则，系统将认为该服务 IP 不可用。

- **HTTP 和 HTTP2 健康检查**

HTTP 和 HTTP2 健康检查在建立 TCP 连接的基础上向后台服务发送预定义的 HTTP 请求。如果后台服务的回答与期望回答一致，服务是好的。否则，服务不可用。使用 HTTP 和 HTTP2 健康检查时，需要预先定义一组 HTTP 请求以及匹配的回答。

- **HTTPS 和 HTTPS2 健康检查**

HTTPS 和 HTTPS2 健康检查基于 SSL 握手协议，但是 HTTPS2 健康进行 SSL 握手时不检查证书。如果 SSL 握手成功，设备会给后台服务发送预定义的 HTTP 请求。如果后台服务返回的响应与期望响应一致，那么后台服务就是可用的；反之，后台服务就是不可用的。使用 HTTPS 和 HTTPS2 健康检查，用户必须提前定义好 HTTP 请求和与请求匹配的回应。对于 HTTPS 健康检查，在进行客户端认证时导入的客户端证书必须是用 DER 规则加密的。

注意：出于性能考虑，建议配置的 HTTP2 类型的健康检查（包括基本、附加和服务组健康检查）不超过 500 条。

26.2.7.1.2 健康检查模板与实例

要使用 SDNS 健康检查功能，管理员需要先定义健康检查模板，然后将模板关联到指定的服务 IP 或者服务池。

健康检查模板由如下元素组成：

- **健康检查的类型**：可以为 ICMP、TCP、UDP、HTTP 或 HTTPS。
- **请求行**：预先设定的 HTTP 请求的请求行，内容包括 HTTP 请求方法、URL 路径和 HTTP 协议版本。仅适用于 HTTP 和 HTTPS 类型的健康检查模板。默认值为“HEAD / HTTP/1.0\r\n\r\n”。
- **期望的响应**：期望的 HTTP 响应。仅适用于 HTTP 和 HTTPS 类型的健康检查模板。默认值为“200 OK”。
- **结果标识（flag）**：收到期望的 HTTP 响应时判定的健康检查结果。仅适用于 HTTP 和 HTTPS 类型的健康检查模板。默认值为“up”。
- **健康检查间隔**：发送健康检查包的间隔。
- **健康检查超时时间**：健康检查包的超时时间。超时时间不应大于健康检查间隔。

- **最大尝试次数**：判断目的地址健康状况的最大连续健康检查次数。例如，3 表示如果连续三次健康检查结果为 Up，则认为目的地址的状态为 Up；如果连续三次健康检查结果为 Down 时，则认为目的地址的状态为 Down。
- **目的地址**：健康检查对象的 IP 地址。通常该参数设为空，当健康检查模板关联到服务 IP 或者服务池时，在生成的健康检查实例中，目的地址会替换为对应的服务 IP。
- **目的端口**：健康检查对象的端口。仅适用于 TCP、UDP、HTTP 或 HTTPS 类型的健康检查模板。通常该参数设为 0，当健康检查模板关联到服务 IP 或者服务池时，在生成的健康检查实例中，目的端口会替换为对应的服务 IP 的健康检查端口。TCP、UDP、HTTP 或 HTTPS 类型的健康检查模板的端口和服务 IP 的健康检查端口不能都设置为 0。
- **源地址**：健康检查包的源地址。
- **网关地址**：指定在多链路网络环境中转发健康检查包的网关 IP 地址。

将健康检查模板关联到指定的服务 IP 后，SDNS 将会生成健康检查实例，模板中目的地址将会替换成服务 IP，然后对服务 IP 发送健康检查包。如果将模板关联到服务池，SDNS 为服务池中每个成员（服务 IP）批量生成健康检查实例。管理员可以根据需要选择将健康检查模板关联到服务 IP 或者服务池。

如果多个健康检查模板关联到服务 IP，则该服务 IP 拥有多个健康检查实例；如果多个健康检查模板关联到服务池，则服务池中每个服务 IP 拥有多个健康检查实例。SDNS 支持为服务 IP 指定其健康检查实例之间的关系。

- “AND”：当服务 IP 的所有健康检查实例的结果为 Up 时，该服务 IP 状态才被置为 Up。
- “OR”：当服务 IP 的任何一个健康检查实例的结果为 Up 时，该服务 IP 状态就被置为 Up。
- “混合关系”：当服务 IP 的所有健康检查实例满足“**sdns pool health mixrelation**”命令中的混合关系表达式时，该服务 IP 的状态就备置为 UP。

服务 IP 的健康检查实例之间的默认关系为“AND”。

配置示例：

```
Demo(config)#sdns monitor icmp "hc_icmp" 5 5 3 0.0.0.0 0.0.0.0 0.0.0.0
Demo(config)#sdns monitor tcp "hc_tcp" 5 5 3 0.0.0.0 1000 0.0.0.0 0.0.0.0
Demo(config)#sdns monitor http "hc_http" "HEAD / HTTP/1.0\r\n\r\n" "200 OK" "up" 5 5 3 0.0.0.0 0 0.0.0.0 0.0.0.0
Demo(config)#sdns service ip "sv1" 10.8.6.201 80
Demo(config)#sdns service ip "sv2" 10.8.6.202 80
Demo(config)#sdns service ip "sv3" 10.8.6.203 80
Demo(config)#sdns service ip "sv4" 10.8.6.204 80
```

```
Demo(config)#sdns pool name "pool1" 1
Demo(config)#sdns pool service "pool1" "sv1"
Demo(config)#sdns pool service "pool1" "sv2"
Demo(config)#sdns pool service "pool1" "sv3"
Demo(config)#sdns pool service "pool1" "sv4"
Demo(config)#sdns pool monitor apply "pool1" "hc_icmp"
Demo(config)#sdns pool monitor apply "pool1" "hc_tcp"
Demo(config)#sdns pool monitor apply "pool1" "hc_http"
```

26.2.7.1.3 自定义健康检查请求/响应文件

为了能够支持复杂的健康检查，管理员需要通过导入文件的方式自定义健康检查请求和预期响应，并将健康检查模板与自定义健康检查请求和响应文件关联。这样，该模板的健康检查请求和预期响应会被替换为自定义健康检查请求和响应文件的内容。

系统支持为 HTTP/HTTPS 类型的健康检查模板关联自定义健康检查请求和响应文件。

配置示例:

```
Demo(config)# sdns monitor http "hc_http" "HEAD / HTTP/1.0\r\n\r\n" "200 OK" "up" 5
5 3 0.0.0.0 0 0.0.0.0 0.0.0.0
Demo(config)#sdns health import request "request" "ftp://1.1.1.1/request.txt"
Demo(config)#sdns health import response "response" "ftp://1.1.1.1/response.txt"
Demo(config)#sdns health bind "hc_http" “request” “response”
Demo(config)#sdns service ip "sv1" 10.8.6.201 80
Demo(config)#sdns service monitor apply "sv1" "hc_http"
```

26.2.7.2 SNMP 数据收集

SNMP 数据收集功能用于向 SDNS 服务收集特定 SNMP OID 的数据。SDNS 服务池的“snmp”算法需要依赖于 SNMP 数据收集功能。SDNS 根据服务池关联的 SNMP 表达式来计算每个 SDNS 服务的复合权重值（metric）。SNMP 表达式指定了需要根据哪些 SNMP OID 的数据来计算复合权重值（metric）。要从每个 SDNS 服务收集这些 SNMP OID 的数据，管理员需要配置 SNMP 数据收集功能。

为使“snmp”算法能够正常工作，管理员需要做如下配置：

- 定义要收集数据的 SNMP OID。
- 定义计算 SDNS 服务的复合权重值（metric）的 SNMP 表达式。
- 定义使用“snmp”算法的 SDNS 服务池及其关联的 SNMP 表达式。
- 为 SNMP OID 定义 SNMP 类型的数据收集模板。
- 将 SNMP 数据收集模板与 SDNS 服务池关联生成 SNMP 数据收集实例。

26.2.7.2.1 SNMP OID 和 SNMP 表达式

每个 SNMP 表达式最少使用一个 SNMP OID 的数据，最多使用三个 SNMP OID 的数据。SNMP 表达式的格式为：

OID1 数据 × OID1 权重 + OID2 数据 × OID2 权重 + OID3 数据 × OID3 权重

定义 SNMP 表达式前，管理员需要先定义使用的 SNMP OID。管理员可以定义最多八个 SNMP OID。

系统收集到 SNMP OID 的数据后，SDNS 将根据 SNMP 表达式计算每个 SDNS 服务的复合权重值，并按照为 SDNS 服务池配置的排序方式进行排序。

SDNS 服务池支持两种排序方式：

- 升序：SDNS 服务 IP 按照复合权重值由低到高排序。
- 降序：SDNS 服务 IP 按照复合权重值由高到低排序。

26.2.7.2.2 SNMP 数据收集模板和实例

为了能够向 SDNS 服务池中的 SDNS 服务收集 SNMP OID 的数据，管理员需要为 SNMP OID 创建 SNMP 数据收集模板，并将模板与 SDNS 服务池关联。这样，系统会为 SDNS 服务池中的每个 SDNS 服务生成 SNMP 数据收集实例。

SNMP 数据收集模板由如下元素组成：

- **OID 名称**：要收集数据的 SNMP OID 的名称。
- **共享字段**：SNMP 服务器的共享字段。默认值为“public”。
- **SNMP 版本**：通信使用的 SNMP 协议版本。
- **时间间隔**：进行数据收集的时间间隔。
- **目的地址**：数据收集对象的 IP 地址。通常该参数设为空，当数据收集模板关联到服务池时，在生成的数据收集实例中，目的地址会替换为对应的服务 IP。
- **目的端口**：数据收集对象的 SNMP 端口。默认值为 161。
- **源地址**：SNMP 请求的源地址。
- **网关地址**：在多链路网络环境中转发 SNMP 请求的网关 IP 地址。

注意：SNMP 数据收集模板只能与 SDNS 服务池进行关联。

配置示例：

```
Demo(config)#sdns snmp oid "id1" ".1.3.6.1.4.1.7564.30.1.0"
Demo(config)#sdns snmp oid "id2" ".1.3.6.1.4.1.7564.4.2.0"
```

```
Demo(config)#sdns snmp oid "id3" ".1.3.6.1.4.1.7564.30.2.0"
Demo(config)#sdns snmp exp weight "expression1" "id1" 1 "id2" 2 "id3" 3
Demo(config)#sdns service ip "sv1" 10.8.6.201 0
Demo(config)#sdns service ip "sv2" 10.8.6.202 0
Demo(config)#sdns service ip "sv3" 10.8.6.203 0
Demo(config)#sdns service ip "sv4" 10.8.6.204 0
Demo(config)#sdns pool name "pool1" 1
Demo(config)#sdns pool method primary "pool1" "snmp" "expression1" "asc"
Demo(config)#sdns pool service "pool1" "sv1"
Demo(config)#sdns pool service "pool1" "sv2"
Demo(config)#sdns pool service "pool1" "sv3"
Demo(config)#sdns pool service "pool1" "sv4"
Demo(config)#sdns monitor snmp collector "dc1" "id1" "public" "v2c" 60 0.0.0.0 161
0.0.0.0 0.0.0.0
Demo(config)#sdns monitor snmp collector "dc2" "id2" "public" "v2c" 60 0.0.0.0 161
0.0.0.0 0.0.0.0
Demo(config)#sdns monitor snmp collector "dc3" "id3" "public" "v2c" 60 0.0.0.0 161
0.0.0.0 0.0.0.0
Demo(config)#sdns pool monitor apply "pool1" "dc1"
Demo(config)#sdns pool monitor apply "pool1" "dc2"
Demo(config)#sdns pool monitor apply "pool1" "dc3"
```

26.2.8 SDNS 静态就近性规则

SDNS 支持 SDNS 静态就近性规则。静态就近性规则可以通过以下两种方式生成：

- 单网段方式：将指定网段与区域关联。
- 批量方式：将一个 IP 域表或者域表关系表达式和 SDNS 区域进行关联，生成 SDNS IP 域就近性规则。要使用 SDNS IP 域表就近性规则，管理员必须先导入 GeoIP 数据库。

SDNS 静态就近性规则的优先级高于 SDNS 动态就近性规则。

配置示例

下面分别介绍单网段方式和批量方式生成 SDNS 静态就近性规则的配置示例。

- 通过将指定网段与区域关联，配置 SDNS 静态就近性规则。

```
Demo(config)#sdns proximity 211.100.0.0 255.255.0.0 "Beijing"
Demo(config)#sdns proximity 210.10.0.0 255.255.0.0 "Tokyo"
Demo(config)#sdns proximity 216.100.0.0 255.255.0.0 "New_York"
```

- 生成省级 SDNS IP 域就近性规则。先将省级数据库导入系统，再基于省级数据库生成指定省的 IP 域表，最后将一个 IP 域表和 SDNS 区域进行关联。

```
Demo(config)#ipregion geoip import state
"ftp://192.168.5.55/home/GeoLite2-City-CSV_20160503.zip"
Demo(config)#ipregion geoip convert state china all
Demo(config)#sdns ipregion proximity "predefined_table" "region1"
```

- 生成自定义 SDNS IP 域就近性规则。先将自定义数据库导入系统，再将一个域表中的混合关系和 SDNS 区域进行关联。

```
Demo(config)#ipregion geoip import custom
"http://192.168.1.1/GeoIP_Custom.signed.tar.gz"
Demo(config)#sdns ipregion proximity "China_Beijing_CMCC_IPv4" "beijing"
Demo(config)#sdns ipregion proximity "China_Beijing_NOT_CTCC_IPv4" "beijing"
Demo(config)#sdns ipregion proximity "China_NOT_Beijing_CUCC_IPv4" "beijing"
Demo(config)#sdns ipregion proximity "China_Shaanxi_xian_CTT_IPv6" "xian"
Demo(config)#sdns ipregion proximity "China_Shaanxi_xian_NOT_CERNET_IPv6"
"xian"
Demo(config)#sdns ipregion proximity "China_Shaanxi_NOT_xian_CMCC_IPv6" "xian"
Demo(config)#sdns ipregion proximity "China_NOT_Shaanxi_xian_CTCC_IPv6" "xian"
```

管理员可以通过“**show ipregion geoip**”命令查看导入后生成的数据库，通过“**show ipregion match**”命令查看指定 IP 和域表的对应关系，通过“**show ipregion name**”命令查看已经存在的 IP 域表的名称。

26.2.9 SDNS 动态就近性探测系统

SDNS 支持使用动态就近性探测系统（Dynamic Proximity System，DPS）生成动态就近性规则。

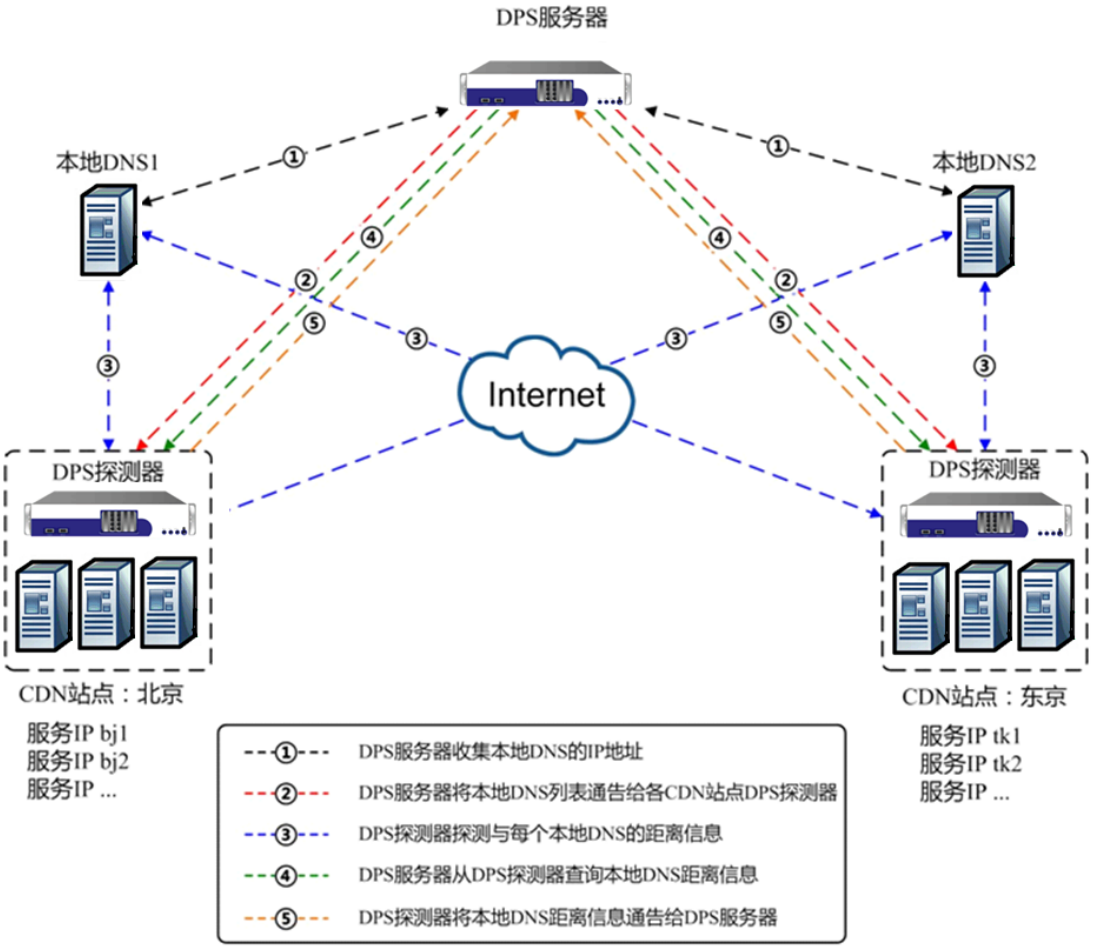

图26–3 SDNS 动态就近性探测系统

动态就近性探测系统由 DPS 服务器和 DPS 探测器组成。

- **DPS 服务器**

DPS 服务器（设备）主要提供如下功能：

- 收集本地 DNS 服务器数据，然后将本地 DNS 服务器的 IP 地址列表发送给各区域的 DPS 探测器。
- 向 DPS 探测器查询就近性探测结果。
- 根据收到的就近性探测结果，DPS 服务器生成动态就近性规则。

- **DPS 探测器**

DPS 探测器主要提供如下功能：

- 接收 DPS 服务器发送的本地 DNS 服务器的 IP 地址列表。
- 通过向本地 DNS 服务器发送请求来探测三种就近性信息：往返时间（Round Trip Time，RTT）、丢包率（Packet Loss Rate，PLR）和路由跳数（Hops）。

- 向 DPS 服务器报告就近性探测结果。

注意：DPS 探测器既可以部署在设备上，也可以部署在运行 Linux 或者 FreeBSD 操作系统的服务器上。

➢ **如何生成动态就近性规则？**

当 DPS 服务器收到某区域 DPS 探测器发送的就近性探测结果后，将根据 DPS 算法计算该 DPS 探测器（区域）与所有本地 DNS 服务器之间路径的“metric”。然后 DPS 服务器为路径的“metric”最小的本地 DNS 服务器和 SDNS 区域创建一条动态就近性规则。

SDNS 支持四种 DPS 算法：

- rtt：Metric = RTT
- plr：Metric = PLR
- hops：Metric = Hops
- mix：Metric = RTT x 权重 + PLR x 权重 + Hops x 权重

当使用 mix 算法时，管理员需要为 RTT、PLR 和 Hops 分别指定权重值。

➢ **动态就近性探测系统的工作机制**

- 本地 DNS 服务器向 DPS 服务器发送 DNS 解析请求。
- 一段时间之后 DPS 服务器就能收集到一些本地 DNS 服务器列表和它们对应的 IP 地址。
- DPS 服务器把收集到的本地 DNS 服务器的 IP 地址下发给在每个区域（CDN 站点）的 DPS 探测器。
- DPS 探测器开始探测各自区域与所有本地 DNS 之间的就近性信息。
- 当 DPS 服务器查询探测结果时，DPS 探测器就将探测到的就近性信息报告给 DPS 服务器。
- DPS 服务器根据探测结果生成动态就近性规则。

配置示例：

```
Demo(config)#sdns region name "Beijing"
Demo(config)#sdns region name "Tokyo"
Demo(config)#sdns dps interval query 1200
Demo(config)#sdns dps method rtt
Demo(config)#sdns dps expire 300
Demo(config)#sdns dps interval send 120
Demo(config)#sdns dps detector "Beijing" 211.100.1.100 44544 900 172800
```

```
Demo(config)#sdns dps detector "Tokyo" 210.10.1.100 44544 900 172800
Demo(config)#sdns dps on
```

26.2.10 全域名解析

在设备中，SDNS 负责处理 A、AAAA、CNAME 和 DNSKEY 类型的 DNS 解析请求，全域名功能负责处理所有其它类型的 DNS 解析请求。

当收到 A、AAAA、CNAME 和 DNSKEY 类型的 DNS 解析请求时，设备将这些请求交由 SDNS 处理。当收到其它类型的 DNS 解析请求时，设备将这些请求交由全域名功能处理。如果 SDNS 上没有配置 A、AAAA、CNAME 和 DNSKEY 类型的 DNS 解析请求中的域名时，设备将返回空 DNS 响应。如果全域名功能不能处理其他类型的 DNS 解析请求，设备也将返回空 DNS 响应。

如果启用递归查询功能：

- 设备将在 SDNS 上没有配置的域名的 A、AAAA、CNAME 和 DNSKEY 类型 DNS 解析请求转发给全域名功能处理。
- 设备将全域名功能不能处理的 DNS 解析请求转发给其它 DNS 服务器进行递归查询，最后把解析结果返回给本地 DNS 服务器。

默认情况下，递归查询功能是禁用的。如果要启用递归查询功能，可以执行如下命令：

```
Demo(config)#sdns recursion on
```

注意：全域名功能需要通过 WebUI 进行配置。

假设 SDNS 中只有域名“image.example.com”的 A、AAAA 或 CNAME 资源记录，为了确保对该域名的其它类型 DNS 解析请求都能返回正确的响应消息，建议在全域名功能的区域文件中为域名“example.com”中加入一条如下格式的 TXT 类型的记录：

```
image           IN          TXT          "A text string"
```

注意：“A text string”可以是任何关于这个域名的描述。

26.2.10.1 TXT 记录

TXT 资源记录是基于文本的 DNS 记录，以 TXT 的格式存储于 DNS 区域文件中，用于检索有关域名的信息。TXT 记录常被用于防止电子邮件垃圾邮件和欺骗行为或者域名所有权的验证等场景中。

要使 TXT 资源记录生效，管理员需要在 DNS 区域里面添加至少有一个 NS 记录和一个 A/AAAA 记录。并且仅当 DNS 区域内 NS/MX 记录的域名与 A/AAAA 域名相同时，才能向区域中添加 A/AAAA 记录。使用 TXT 记录功能前需要启用 SDNS 功能（通过命令“**sdns on**”）以及 SDNS 全域名功能（通过命令“**sdns fulldns on**”）。

配置示例：

- **基础配置**

1. 配置一个 SDNS ZONE。

```
Demo(config)#sdns zone name abc.com
```

2. 配置一个 NS 记录，并将其与 ZONE 绑定。

```
Demo(config)#sdns record ns ns1 abc.com www.abc.com
Demo(config)#sdns zone record abc.com ns1
```

3. 配置一个 A/AAAA 记录，并将其与 ZONE 绑定。

```
Demo(config)#sdns record ip a1 www.abc.com 1.1.1.1
Demo(config)#sdns zone record abc.com a1
```

4. 启用 SDNS 全域名功能和 SDNS 功能。

```
Demo(config)#sdns fulldns on
Demo(config)#sdns on
```

5. （可选）启用 SDNS 递归查询。

```
Demo(config)#sdns recursion on
```

- **配置 TXT 记录**

1. 配置一个 TXT 记录，并将其与 ZONE 绑定。

```
Demo(config)#sdns record txt txt1 www.abc.com “hello abc”
Demo(config)#sdns zone record abc.com txt1
```

2. 通过主机查看 TXT 记录。

```
C:\Users\Administrator>nslookup -type=txt www.abc.com
服务器:  UnKnown
Address:  192.168.93.100
www.abc.com        text =
          "hello abc"
abc.com nameserver = www.abc.com
www.abc.com        internet address = 1.1.1.1
```

26.2.11 IPv6 支持

SDNS 功能提供广泛的 IPv6 支持。

- AAAA DNS 解析：SDNS 支持处理 AAAA 类型的 DNS 解析请求。
- SDNS 服务 IP：服务 IP 可以为 IPv4 或 IPv6 地址。
- SDNS 服务池：一个 SDNS 服务池的服务 IP 可以全为 IPv4 或者 IPv6 地址，但该服务池不能同时包含 IPv4 和 IPv6 服务 IP。
- SDNS 就近性规则：就近性规则支持 IPv6。
- SDNS DPS：SDNS 支持 IPv6 DPS 探测器，支持对 IPv6 本地 DNS 服务器进行探测。
- SDNS 健康检查：SDNS 支持对 IPv4 和 IPv6 服务 IP 进行健康检查。

26.2.12 数据中心同步

通过 SDNS 数据中心同步功能，可以从逻辑上将多个位于相同或不同地理位置的 SDNS 设备组成一个同步群组，群组内的每个 SDNS 设备分别作为一个数据中心。每个群组内最多可以添加 64 个数据中心（通过“**sdns dc**”命令），数据中心会通过该命令定义的 IP 地址和其他数据中心通信。

每个数据中心通过关联 SDNS 服务来确定本地服务 IP（通过“**sdns service ip**”命令），关联后该服务 IP 和生成的健康检查实例（通过将服务 IP 与健康检查模板关联）都会被认为属于该中心，不关联任何数据中心的服务 IP 和生成的健康检查实例则不属于任何中心。各个数据中心只探测本中心和不属于任何中心的健康检查实例状态，但是会互相交换各自的健康检查实例状态，这样各个数据中心之间会共享彼此的服务 IP 健康状态，使全局的 DNS 解析保持一致。

26.2.12.1 配置同步和状态同步

数据中心之间的配置同步通过执行 CLI 命令“**sdns sync config from**”和“**sdns sync config to**”实现。以上命令不支持同步“**sdns dc**”和“**sdns ipregion proximity**”配置，因此在进行配置同步前，每个数据中心需要分别定义好本地和远端数据中心。如果使用 IP 域表功能，也需要提前在每个数据中心上分别配置好 IP 域就近性规则。

数据中心之间服务 IP 健康状态的同步机制包含主动请求和主动推送两个部分，具体原则如下：

- 各个数据中心会主动探测所有属于本中心的健康检查实例状态和不属于任何中心的本地健康检查实例状态，并生成一个健康检查项列表，包括本地和所有其他数据中心的健康检查实例 ID 和实例状态。

- 当本地服务 IP 由 Up 变成 Down，或者由 Down 变成 Up 时，本地数据中心主动将状态变化通过 PUSH 请求发送给其他数据中心。收到 PUSH 请求的设备，更新相应的实例状态。
- 各个数据中心定期（默认 3 秒）向其他数据中心发送 GET 请求获取其他中心的健康检查实例状态。收到 GET 请求的数据中心根据请求的实例列表，查找健康检查实例状态，然后通过 GET ACK 消息回复给请求方。收到 GET ACK 消息的数据中心，更新相应的健康检查实例的状态。如果响应中没有某个实例的状态，则为该实例记录一次健康检查失败。如果指定时间内（默认 2 秒）没收到 GET ACK 消息，则为所有请求过的实例分别记录一次健康检查失败。

SDNS 数据中心间进行状态同步时，主动发送 GET 请求的时间间隔和等待响应时的超时时间可以通过“**sdns sync status**”命令自定义。等待响应的超时时间不能大于发送请求的时间间隔。

系统在常规模式的基础上增加了故障切换探测模式。即当数据中心之间服务 IP 健康状态的同步失败时，本地数据中心向其他数据中心主动发送健康检查状态探测报文。

在常规模式下，系统采用原有行为，即当本地数据中心未收到其他数据中心返回的健康检查实例状态时，系统不会主动探测，直接将该数据中心的健康检查实例状态置为“Down”。

在故障切换探测模式下，当本地数据中心未收到其他数据中心返回的健康检查实例状态时，系统将会主动探测。具体分为如下两种情况：

- 如果健康检查模板设置了目的 IP 地址，并且设置的目的 IP 与 SDNS 服务 IP 不同，那么将 SDNS 服务与健康检查模板关联后，系统将会创建两个健康检查实例，分别为主实例（Primary Instance）和备实例（Instance）（可以通过命令“**show statistics sdns dc instance**”查看）。
  - 当本地数据中心探测到其他数据中心的健康检查实例状态时，主实例健康检查状态标识为探测到的状态，备实例的健康检查状态与主实例的健康检查状态相同。
  - 当本地数据中心无法探测到其他数据中心的健康检查实例状态时，主实例的健康检查状态将被标识为 Down，备实例的健康检查状态通过探测 SDNS 服务 IP 获取。
- 如果健康检查设置的目的 IP 地址与 SDNS 服务的 IP 相同时，将 SDNS 服务与健康检查模板关联后，系统仅生成一个健康检查实例，且本地数据中心获取健康检查实例状态失败时不进行主动探测。
- 如果健康检查模板未设置目的 IP 地址，将 SDNS 服务与健康检查模板关联后，系统只会生成一个健康检查实例。具体分两种情况：

- 当本地数据中心收到其他数据中心的健康检查实例状态时，直接使用此状态标识健康检查实例的状态。
- 当本地数据中心未收到其他数据中心的健康检查实例状态时，系统将主动探测健康检查实例状态。

配置示例：

设备 1

```
Demo1(config)#sdns dc dc1 192.168.93.100 60000 local

Demo1(config)#sdns dc dc2 192.168.93.200 60000 remote

Demo1(config)#sdns service ip abc 192.168.83.11 80 dc1 1
Demo1(config)#sdns monitor tcp h1 5 5 3 192.168.88.10 80 0.0.0.0 0.0.0.0
Demo1(config)#sdns service monitor apply abc h1
Demo1(config)#synconfig peer unit1 192.168.93.100
Demo1(config)#synconfig peer unit2 192.168.93.200
Demo1(config)#synconfig challenge "123"

Demo1(config)#sdns on
```

设备 2

```
Demo2(config)#clear sdns all

Demo2(config)#sdns dc dc1 192.168.93.100 60000 remote

Demo2(config)#sdns dc dc2 192.168.93.200 60000 local

Demo2(config)#synconfig peer unit1 192.168.93.100

Demo2(config)#synconfig peer unit2 192.168.93.200

Demo2(config)#synconfig challenge "123"

Demo2(config)#sdns on
```

设备 1

```
Demo1(config)#show statistics sdns dc instance

DC Name:                dc1

Ip Address:             192.168.93.100

Port:                   60000

ID:                     0

Type:                   local|master

Status:                 LISTEN
```

```
Instance Count:         2
        Instance ID:    0
        References:     1
        HC Mode:        0
        Status:         DOWN(PROBE_FAILED)
        Monitor:        h1
        Type:           TCP
        Interval:       5
        Timeout:        5
        Max retries:    3
        Address:        192.168.88.10
        Port:           80
        Source:         0.0.0.0
        Gateway:        0.0.0.0

        Instance ID:    1
        References:     1
        HC Mode:        1
        Primary ID:     0
        Status:         DOWN(SYNC_SUCCESS)
        Monitor:        h1
        Type:           TCP
        Interval:       5
        Timeout:        5
        Max retries:    3
        Address:        192.168.83.11
        Port:           80
        Source:         0.0.0.0
```

```
    Gateway:            0.0.0.0


DC Name:                dc2
Ip Address:             192.168.93.200
Port:                   60000
ID:                     1
Type:                   remote
Status:                 ESTABLISHED
Instance Count:         0
```

设备 2

```
Demo2(config)#show statistics sdns dc instance
DC Name:                dc1
Ip Address:             192.168.93.100
Port:                   60000
ID:                     0
Type:                   remote
Status:                 ESTABLISHED
Instance Count:         0


DC Name:                dc2
Ip Address:             192.168.93.200
Port:                   60000
ID:                     1
Type:                   local|master
Status:                 LISTEN
Instance Count:         0
```

设备 1

```
Demo1(config)#sdns sync config to dc2
```

设备 2

```
Demo2(config)#show statistics sdns dc instance
DC Name:                dc1
Ip Address:             192.168.93.100
Port:                   60000
ID:                     0
Type:                   remote
Status:                 ESTABLISHED
Instance Count:         2
    Instance ID:        0
    References:         1
    HC Mode:            0
    Status:             DOWN(SYNC_SUCCESS)
    Monitor:            h1
    Type:               TCP
    Interval:           5
    Timeout:            5
    Max retries:        3
    Address:            192.168.88.10
    Port:               80
    Source:             0.0.0.0
    Gateway:            0.0.0.0

    Instance ID:        1
    References:         1
    HC Mode:            1
    Primary ID:         0
    Status:             DOWN(SYNC_SUCCESS)
```

```
    Monitor:            h1
    Type:               TCP
    Interval:           5
    Timeout:            5
    Max retries:        3
    Address:            192.168.83.11
    Port:               80
    Source:             0.0.0.0
    Gateway:            0.0.0.0

DC Name:                dc2
Ip Address:             192.168.93.200
Port:                   60000
ID:                     1
Type:                   local|master
Status:                 LISTEN
Instance Count:         0
```

26.2.12.2 SDNS 运行时配置同步

启用 SDNS 运行时配置同步功能后，当一个数据中心的配置发生改变时，该数据中心通过发送 CPUSH 请求将这些配置实时同步到其他数据中心。该功能只同步功能启用后变动的配置。该功能默认是禁用的。

只有在本地和远端数据中心都启用该功能时，才能在数据中心间进行实时的配置同步。对于不需要同步的 SDNS 配置项，系统通过一个黑名单管理。启用运行时配置同步功能后，除黑名单以外的所有配置项都会被同步。

SDNS 运行时配置同步支持日志功能。该功能用于显示 SDNS 运行时配置同步过程中的信息。如果启用了 SDNS 运行时配置同步功能，则日志功能同时被启用；如果禁用了 SDNS 运行时配置同步功能，则日志功能同时被禁用。

- **SDNS 运行时配置同步**

```
Demo(config)#sdns sync config runtime on
Demo(config)#show sdns sync config
```



```
sdns sync config runtime on
Demo(config)#show sdns sync status
sdns sync status 3 2
```

- **SDNS 运行时配置同步日志分析**

```
Demo(config)#show sdns sync log
dc name: dc2(192.168.93.200)
Total failed :  2
Total loss   :  0
Total success:  65
Failed cli command:
no sdns service ip “resorts1”
Service name “resorts1” not found.
Failed tp execute “no sdns service ip” “resorts1”
no sdns service ip “resorts2”
Service name “resorts2” not found.
Failed tp execute “no sdns service ip” “resorts2”
dc name: dc22(182.16.8.222)
Total failed :  0
Total loss   :  2
Total success:  65
Loss cli command:
no sdns service ip “resorts1”
no sdns service ip “resorts2”
Demo(config)#show sdns sync log dc2 pool
192.168.63.200:sdns pool name "a1" 3
192.168.63.200:sdns pool method primary "a1" rr
192.168.63.200:sdns pool name "a2" 3
192.168.63.200:sdns pool method primary "a2" rr
192.168.63.200:sdns pool name "a3" 3
192.168.63.200:sdns pool method primary "a3" rr
```

26.2.12.3 数据中心内 HA 支持

SDNS 数据中心支持 HA 主备模式。在每个数据中心内，都可以配置一台主机和一台或多台备机。另外，数据中心的 IP 地址（通过“**sdns dc**”命令配置）需要设置为 HA 浮动 IP 地址。浮动 IP 地址所在的设备将作为数据中心的主机，并加入 SDNS 数据中心所在的同步群组，备机不属于任何中心，也不加入同步群组。主机和备机将基于以下机制探测健康检查实例的状态。当主机发生故障时，两台设备能够进行快速切换，备机将作为主机加入同步组。

主机：

- 主动探测属于本中心的健康检查实例状态，当本地服务 IP 状态发生变化时，通过 PUSH 请求主动将状态变化发送给其他数据中心。收到 PUSH 请求时，更新相应的实例状态。
- 主动探测不属于任何中心的本地健康检查实例状态。
- 定期（默认 3 秒）向其他数据中心发送 GET 请求获取其他中心的健康检查实例状态。收到 GET 请求时，回复本中心相应的健康检查实例状态。

备机：

- 主动探测不属于任何中心的本地健康检查实例的状态。
- 定期（默认 3 秒）向其他数据中心（包括本中心）发送 GET 请求获取其他中心的健康检查实例状态。

另外，当 SDNS 数据中心主机间进行 SDNS 运行时配置同步时，系统同时支持通过 HA 在 SDNS 的数据中心主机和备机间进行同步。

26.2.13 SDNS 链路

SDNS 支持为数据中心配置 SDNS 链路。通过为 SDNS 链路关联健康检查模板，管理员能够对该 SDNS 链路进行健康检查。通过将 SDNS 服务与 SDNS 链路关联，实现了对 SDNS 链路上关联的 SDNS 服务的统一管理。如果 SDNS 链路的状态为“Down”，则该 SDNS 链路关联的所有的 SDNS 服务都将不可用。

SDNS 链路支持本地数据中心探测本地数据中心的 SDNS 链路的健康状态，其他数据中心的 SDNS 链路的健康状态通过数据中心同步来实现。

此外，SDNS 支持为 SDNS 链路设置健康检查实例间的关系及混合关系。

配置示例：

```
Demo(config)#sdns link name link1 192.168.93.2 dc1
Demo(config)#sdns link service link1 service1
Demo(config)#sdns monitor apply link1 monitor1
Demo(config)#show statistics sdns link
Link Name:                    link1
Hit:                          0
Health:                       UP
Health Relation:              and
LINK Monitor Count:           1
    Monitor Name:             monitor1
    Type:                     ICMP
    Instance ID:              5
    Status:                   UP
LINK Member Count:            1
```



```
    Service Name:              service1
    Service IP:                192.168.83.71
Demo(config)#show statistics sdns dc instance
DC Name:                   dc1
Ip Address:                192.168.93.100
Port:                      60000
ID:                        0
Type:                      local|master
Status:                    LISTEN
Instance Count:            3
    Instance ID:           3
    References:            1
    HC Mode:               1
    Status:                UP
    Monitor:               h1
    Type:                  HTTP
    Request:               HEAD / HTTP/1.0\r\n\r\n
    Response:              200 OK
    Flag:                  up
    Interval:              3
    Timeout:               3
    Max retries:           1
    Address:               192.168.83.11
    Port:                  80
    Source:                0.0.0.0
    Gateway:               0.0.0.0

    Instance ID:           5
    References:            1
    HC Mode:               0
    Status:                UP
    Monitor:               monitor1
    Type:                  ICMP
    Interval:              5
    Timeout:               5
    Max retries:           3
    Address:               192.168.93.2
    Source:                0.0.0.0
    Gateway:               0.0.0.0
```

26.2.14 SDNS 加密通道

SDNS 支持通过 SSL 通道对数据中心间的通信进行加密。

26.2.14.1 配置示例

26.2.14.1.1 配置目标

- 配置两个数据中心 DC_A、DC_B；
- 配置 SSL 通道，实现 DC_A 收到的流量将加密发送到到 DC_B。

26.2.14.1.2 配置示例

下面分基本 SDNS 配置和 SSL 通道配置两部分介绍。管理员需要先完成基本配置，再配置 SSL 通道配置。

- **基本 SDNS 配置**

在 DC_A 和 DC_B 上分别定义本地和远端数据中心。

DC_A：

```
Demo(config)#sdns dc DC_A 1.1.1.1 6000 local
Demo(config)#sdns dc DC_B 2.2.2.2 6000 remote
```

DC_B：

```
Demo(config)#sdns dc DC_A 1.1.1.1 6000 remote
Demo(config)#sdns dc DC_B 2.2.2.2 6000 local
```

- **SSL 通道配置**

DC_A

```
Demo(config)#ssl channel name dca1 2.2.2.2 6000 443 out ssl_rhost1
Demo(config)#ssl import key ssl_rhost1
Demo(config)#ssl import certificate ssl_rhost1
Demo(config)#ssl import interca ssl_rhost1
Demo(config)#ssl import rootca ssl_rhost1
Demo(config)#ssl activate certificate ssl_rhost1
Demo(config)#ssl start ssl_rhost1
```

DC_B

```
Demo(config)#ssl channel name dcb1 2.2.2.2 6000 443 in ssl_vhost1
Demo(config)#ssl import key ssl_vhost1
Demo(config)#ssl import certificate ssl_vhost1
Demo(config)#ssl import interca ssl_vhost1
```



```
Demo(config)#ssl import rootca ssl_vhost1
Demo(config)#ssl activate certificate ssl_vhost1
Demo(config)#ssl start ssl_vhost1
```

完成以上配置后，在 DC_A 上同步配置到 DC_B 时（通过命令“**sdns sync config runtime on**”启用 SDNS 运行时配置同步功能），当流量命中 DC_A 的 out 方向的 SSL 通道（源 IP 端口为 1.1.1.1：6000，目的 IP 端口为 2.2.2.2：6000），在 DC_A 中处理后转发给 DC_B，目的 IP 端口为 2.2.2.2：443；当流量到达 DC_B 时，将命中 DC_B 的 in 方向的 SSL 通道，将流量从 DC_B 的 443 端口转发到 6000 端口。

26.3 配置示例

26.3.1 配置目标

假设在 CDN 网络中“北京”、“东京”和“纽约”三个站点同时对外提供“www.xyz.com”服务，且每个站点都有两台服务器同时提供服务，如下图所示。

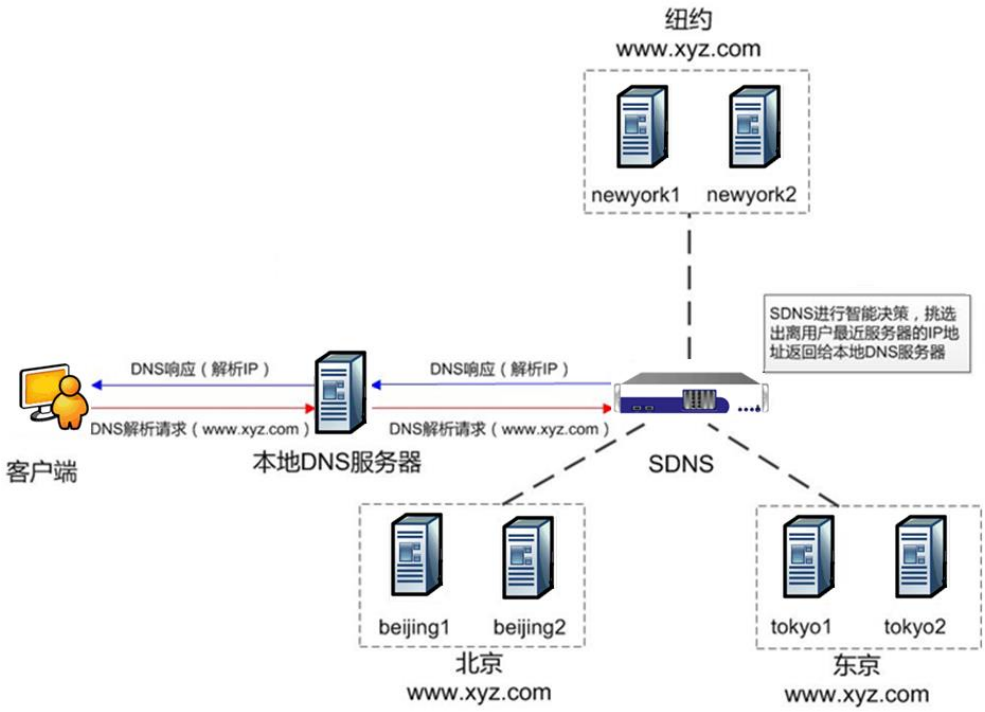

图26–4 SDNS 区域策略应用示例

配置目标如下：

- 对位置靠近“北京”站点的本地 DNS 服务器发来的 DNS 解析请求，SDNS 返回“北京”站点的服务器的 IP 地址。当“北京”站点的服务器不可用时，

SDNS 将返回“东京”站点的服务器 IP 地址。当“东京”站点的服务器也不可用时，SDNS 将返回“纽约”站点的服务器 IP 地址。

- 对位置靠近“东京”站点的本地 DNS 服务器发来的 DNS 解析请求，SDNS 返回“东京”站点的服务器的 IP 地址。当“东京”站点的服务器不可用时，SDNS 将返回“北京”站点的服务器 IP 地址。当“北京”站点的服务器也不可用时，SDNS 将返回“纽约”站点的服务器 IP 地址。
- 对位置靠近“纽约”站点的本地 DNS 服务器发来的 DNS 解析请求，SDNS 返回“纽约”站点的服务器的 IP 地址。当“纽约”站点的服务器不可用时，SDNS 将返回“北京”或“东京”站点的服务器 IP 地址。

为了实现以上目标，管理员可以：

- 将三个站点分别定义为区域“Beijing”、“Tokyo”和“New_York”
- 定义不同的区域策略将来自“Beijing”、“Tokyo”和“New_York”区域的 DNS 解析请求分别关联到服务池“pool_beijing”、“pool_tokyo”和“pool_newyork”。其中，服务池“pool_beijing”包含服务 IP“beijing1”和“beijing2”；服务池“pool_tokyo”包含服务 IP“tokyo1”和“tokyo2”；服务池“pool_newyork”包含服务 IP“newyork1”和“newyork2”。
- 为服务池“pool_beijing”、“pool_tokyo”和“pool_newyork”配置回退池。“pool_beijing”可以回退到包含“pool_beijing”和“pool_tokyo”所有服务 IP 的“pool_asia”服务池，“pool_asia”池和“pool_newyork”都可以回退到包含“pool_beijing”、“pool_tokyo”和“pool_newyork”所有服务 IP 的“pool_world”服务池。

26.3.2 配置示例

26.3.2.1 基本配置

- **添加 SDNS 域名**

例如：

```
Demo(config)#sdns host name "www.xyz.com" 60
```

- **添加 SDNS 服务 IP**

例如：

```
Demo(config)#sdns service ip "beijing1" 211.100.20.101 0
Demo(config)#sdns service ip "beijing2" 211.100.20.102 0
Demo(config)#sdns service ip "tokyo1" 210.10.50.101 0
Demo(config)#sdns service ip "tokyo2" 210.10.50.102 0
```

```
Demo(config)#sdns service ip "newyork1" 216.100.10.101 0
Demo(config)#sdns service ip "newyork2" 216.100.10.102 0
```

- **配置 SDNS 服务池及算法**

例如：

```
Demo(config)#sdns pool name "pool_beijing" 1
Demo(config)#sdns pool name "pool_tokyo" 1
Demo(config)#sdns pool name "pool_newyork" 1
Demo(config)#sdns pool method primary "pool_beijing" "ipo"
Demo(config)#sdns pool failover "pool_beijing" on
Demo(config)#sdns pool preempt "pool_beijing" on
Demo(config)#sdns pool method primary "pool_tokyo" "rr"
Demo(config)#sdns pool method primary "pool_newyork" "rr"
Demo(config)#sdns pool service "pool_beijing" "beijing1"
Demo(config)#sdns pool service "pool_beijing" "beijing2"
Demo(config)#sdns pool member priority "pool_beijing" "beijing1" 1
Demo(config)#sdns pool member priority "pool_beijing" "beijing2" 2
Demo(config)#sdns pool service "pool_tokyo" "tokyo1"
Demo(config)#sdns pool service "pool_tokyo" "tokyo2"
Demo(config)#sdns pool service "pool_newyork" "newyork1"
Demo(config)#sdns pool service "pool_newyork" "newyork2"
```

- **添加 SDNS 区域策略**

例如：

```
Demo(config)#sdns region name "Beijing"
Demo(config)#sdns region name "Tokyo"
Demo(config)#sdns region name "New_York"
Demo(config)#sdns proximity 211.100.0.0 255.255.0.0 "Beijing"
Demo(config)#sdns proximity 210.10.0.0 255.255.0.0 "Tokyo"
Demo(config)#sdns proximity 216.100.0.0 255.255.0.0 "New_York"
Demo(config)#sdns policy region "policy_beijing" "www.xyz.com" "pool_beijing" "Beijing"
0
Demo(config)#sdns policy region "policy_tokyo" "www.xyz.com" "pool_tokyo" "Tokyo" 0
Demo(config)#sdns policy region "policy_newyork" "www.xyz.com" "pool_newyork"
"New_York" 0
```

- **配置 SDNS 健康检查**

例如：

```
Demo(config)#sdns monitor icmp "hc1" 5 5 3 0.0.0.0 0.0.0.0 0.0.0.0
Demo(config)#sdns monitor tcp "hc2" 5 5 3 0.0.0.0 1000 0.0.0.0 0.0.0.0
Demo(config)#sdns service monitor apply "beijing1" "hc1"
```

```
Demo(config)#sdns service monitor apply "beijing1" "hc2"
Demo(config)#sdns service monitor apply "beijing2" "hc1"
Demo(config)#sdns service monitor apply "beijing2" "hc2"
Demo(config)#sdns service monitor apply "tokyo1" "hc1"
Demo(config)#sdns service monitor apply "tokyo1" "hc2"
Demo(config)#sdns service monitor apply "tokyo2" "hc1"
Demo(config)#sdns service monitor apply "tokyo2" "hc2"
Demo(config)#sdns service monitor apply "newyork1" "hc1"
Demo(config)#sdns service monitor apply "newyork1" "hc2"
Demo(config)#sdns service monitor apply "newyork2" "hc1"
Demo(config)#sdns service monitor apply "newyork2" "hc2"
Demo(config)#sdns service health relation "beijing1" "and"
Demo(config)#sdns service health relation "beijing2" "and"
Demo(config)#sdns service health relation "tokyo1" "and"
Demo(config)#sdns service health relation "tokyo2" "and"
Demo(config)#sdns service health relation "newyork1" "and"
Demo(config)#sdns service health relation "newyork2" "and"
```

- **为 SDNS 服务池配置回退池**

例如：

```
Demo(config)#sdns pool name "pool_asia" 1
Demo(config)#sdns pool method primary "pool_asia" "rr"
Demo(config)#sdns pool service "pool_asia" "beijing1"
Demo(config)#sdns pool service "pool_asia" "beijing2"
Demo(config)#sdns pool service "pool_asia" "tokyo1"
Demo(config)#sdns pool service "pool_asia" "tokyo2"
Demo(config)#sdns pool name "pool_world" 1
Demo(config)#sdns pool method primary "pool_world" "rr"
Demo(config)#sdns pool service "pool_world" "beijing1"
Demo(config)#sdns pool service "pool_world" "beijing2"
Demo(config)#sdns pool service "pool_world" "tokyo1"
Demo(config)#sdns pool service "pool_world" "tokyo2"
Demo(config)#sdns pool service "pool_world" "newyork1"
Demo(config)#sdns pool service "pool_world" "newyork2"
Demo(config)#sdns pool fallback "pool_beijing" "pool_asia"
Demo(config)#sdns pool fallback "pool_tokyo" "pool_asia"
Demo(config)#sdns pool fallback "pool_newyork" "pool_world"
Demo(config)#sdns pool fallback "pool_asia" "pool_world"
```

- **启用 SDNS 功能**

例如：


```
Demo(config)#sdns on
```

26.3.2.2 高级配置

基于以上基本配置，如果要实现三个站点之间的服务 IP 状态的同步，可以通过专线将三个站点相互连接，并配置 SDNS 数据中心同步功能。

1. 各个站点分别定义数据中心。

北京：

```
Demo(config)#sdns dc beijing_dc 192.168.0.136 10008 local
Demo(config)#sdns dc newyork_dc 10.0.9.12 10009 remote
Demo(config)#sdns dc tokyo_dc 172.16.9.8 12001 remote
```

纽约：

```
Demo(config)#sdns dc newyork_dc 10.0.9.12 10009 local
Demo(config)#sdns dc tokyo_dc 172.16.9.8 12001 remote
Demo(config)#sdns dc beijing_dc 192.168.0.136 10008 remote
```

东京：

```
Demo(config)#sdns dc tokyo_dc 172.16.9.8 12001 local
Demo(config)#sdns dc beijing_dc 192.168.0.136 10008 remote
Demo(config)#sdns dc newyork_dc 10.0.9.12 10009 remote
```

2. 在北京、纽约和东京中的任一站点将服务 IP 和数据中心关联。

```
Demo(config)#sdns service ip "beijing1" 211.100.20.101 0 beijing_dc
Demo(config)#sdns service ip "beijing2" 211.100.20.102 0 beijing_dc
Demo(config)#sdns service ip "tokyo1" 210.10.50.101 0 tokyo_dc
Demo(config)#sdns service ip "tokyo2" 210.10.50.102 0 tokyo_dc
Demo(config)#sdns service ip "newyork1" 216.100.10.101 0 newyork_dc
Demo(config)#sdns service ip "newyork2" 216.100.10.102 0 newyork_dc
```

3. 设置 GET 请求的发送间隔和响应超时时间，注意间隔时间必须大于或等于超时时间。

```
Demo(config)#sdns sync status 5 2
```

4. 同步配置到其他数据中心。

```
Demo(config)#sdns sync config to all
```

26.4 SDNS DNSSEC 支持

SDNS 支持 DNSSEC 功能，解决了用户在 DNS 递归查询中可能遭遇的中间人攻击，有效防止 DNS 欺骗和缓存污染。

DNSSEC 通过数字签名机制来确保 DNS 响应报文的真实性和完整性。DNSSEC 签名机制包含 ZSK（Zone Signing Key，根区签名密钥）和 KSK（Key Signing Key，根密钥签名密钥）两种密钥。在该机制中，ZSK 用于为域名的所有资源记录生成一份签名信息（RRSIG 记录），KSK 则用于对 ZSK 公钥（DNSKEY 记录）进行签名。DNS 服务器提供的签名公钥和 RRSIG 记录可以确保资源记录未经任何篡改。为了进一步验证 DNS 服务器提供的签名公钥的合法性，本地 DNS 服务器会向上一级 DNS 服务器请求 DS 记录（包含该公钥的哈希文件），这样就构建成一条完整的 DNSSEC 消息信任链。

DNSSEC 实现的流程图如下图所示：

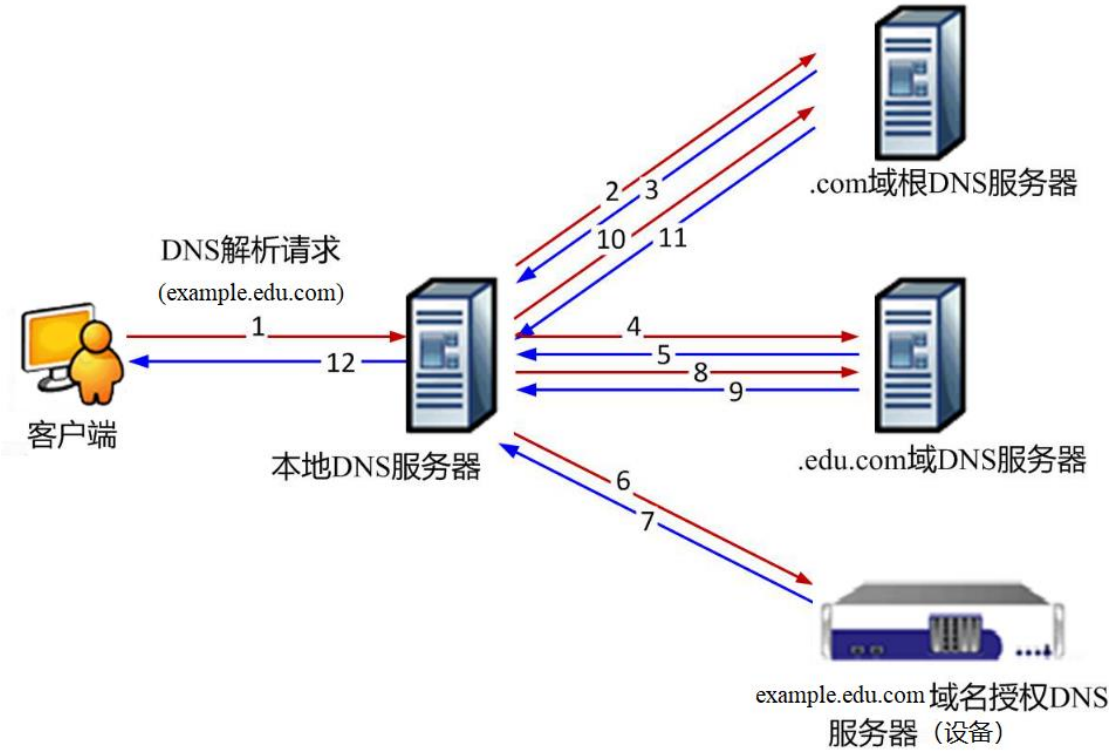

图26–5 SDNS DNSSEC 解析流程

在上图示例中，SDNS DNSSEC 实现的流程如下：

1. 客户端向本地 DNS 服务器请求解析域名 acacac.edu.com。
2. 本地 DNS 服务器向根 DNS 服务器请求该域名对应的资源记录（例如 A 记录）和根 DNS 服务器的公钥。
3. 根 DNS 名服务器将.edu.com 域服务器的 IP 地址和自己的公钥返回给本地 DNS 服务器。

4. 本地 DNS 服务器向.edu.com 域 DNS 服务器发送 A 记录请求。
5. .edu.com 域 DNS 服务器将 acacac.edu.com 域名的授权 DNS 服务器的 IP 地址返回给本地 DNS 服务器。
6. 本地 DNS 服务器向授权 DNS 服务器发送 A 记录请求和 DNSKEY 记录请求。
7. 授权 DNS 服务器将域名 acacac.edu.com 的 A 记录、RRSIG 资源记录（签名的 A 记录）和自己的公钥返回给本地 DNS 服务器。
8. 本地 DNS 服务器向.edu.com 域 DNS 服务器请求 acacac.edu.com 域名的授权 DNS 服务器的 DS 记录。DS 文件记录了 KSK 公钥的摘要，用于验证 KSK 公钥的合法性。
9. .edu.com 域 DNS 服务器将请求的 DS 记录返回给本地 DNS 服务器。
10. 本地 DNS 服务器向根 DNS 服务器请求.edu.com 域 DNS 服务器的 DS 记录，以验证.edu.com 域 DNS 服务器的 KSK 公钥的合法性。
11. 根 DNS 服务器将请求的 DS 记录返回给本地 DNS 服务器。
12. 本地 DNS 服务器把该网站的 A 资源记录返回给客户端。

为指定域名完成 SDNS 配置后，设备将成为该域名的授权 DNS 服务器。如果需要为域名启用 DNSSEC 功能，管理员还需要增加如下配置：

- 生成 ZSK 和 KSK 签名密钥
- 发布签名到对应域名根区
- 对域名资源记录执行签名操作，以生成 RRSIG 记录和 DS 文件
- 将签名密钥的 DS 文件导入到上一级 DNS 服务器
- 为 SDNS 启用 DNSSEC 功能

配置示例：

1. 为域名 acacac.edu.com 完成 SDNS 配置。

```
Demo(config)#sdns host name "acacac.edu.com" 60
Demo(config)#sdns service ip "service1" 211.100.20.101 80
Demo(config)#sdns service ip " service2" 211.100.20.102 80
Demo(config)#sdns pool name "pool1" 1
Demo(config)#sdns pool method primary "pool1" "rr"
Demo(config)#sdns pool service "pool1" " service1"
Demo(config)#sdns pool service "pool1" " service2"
Demo(config)#sdns policy default "acacac.edu.com" "pool1"
Demo(config)#sdns on
```


注意：“sdns pool name”命令中，“max_rr_count”参数只能设置为 1。

2. 为域名 acacac.edu.com 生成 ZSK 和 KSK 签名公钥。

```
Demo(config)#sdns dnssec keygen acacac.edu.com ksk
Demo(config)#sdns dnssec keygen acacac.edu.com zsk
Demo(config)#show sdns dnssec keygen
Kacacac.edu.com.+005+00942+zsk+RSASHA1+1024
Kacacac.edu.com.+005+28783+ksk+RSASHA1+1024
```

3. 发布 ZSK 和 KSK 签名公钥。

```
Demo(config)#sdns dnssec keypub acacac.edu.com ksk
"Kacacac.edu.com.+005+28783+ksk+RSASHA1+1024"
Demo(config)#sdns dnssec keypub acacac.edu.com zsk
"Kacacac.edu.com.+005+00942+zsk+RSASHA1+1024"
Demo(config)#show sdns dnssec keypub
acacac.edu.com Kacacac.edu.com.+005+00942+zsk+RSASHA1+1024
acacac.edu.com Kacacac.edu.com.+005+28783+ksk+RSASHA1+1024
```

4. 对域名 acacac.edu.com 的资源记录执行签名操作。

```
Demo(config)#sdns dnssec zonesig acacac.edu.com
Demo(config)#show sdns dnssec zonesig
signed host: acacac.edu.com
```

5. 显示 DNSSEC 的配置状态。

```
Demo(config)#show sdns dnssec status acacac.edu.com
host status flags: HOST_PUBLISH|HOST_KEY|HOST_SIGN
ttl: 60(s)
zsk key counter: 1
ksk key counter: 1
ZSK key(0):
        keyfile: Kacacac.edu.com.+005+43368+zsk+RSASHA1+1024
        keyid: 43368
        type: ZSK
        algorithm: RSASHA1
        status_flags: KEY_PUBLISH|KEY_SIGN|KEY_TIMER
        ttl: 1(hour)
        ex_time: 30(days)
        ro_time: 25(days)
KSK key(0):
        keyfile: Kacacac.edu.com.+005+07177+ksk+RSASHA1+1024
```

```
keyid: 7177
type: KSK
algorithm: RSASHA1
status_flags: KEY_PUBLISH|KEY_SIGN|NO
ttl: 1(hour)
ex_time: 30(days)
ro_time: 25(days)
```

6. 导出 acacac.edu.com 域名的 DS 文件。

```
Demo(config)#sdns dnssec export ds acacac.edu.com "ftp://test:password@10.8.5.55
/dsset-edu.com."
```

7. 将 acacac.edu.com 域名的 DS 文件导入到上一级 DNS 服务器。

注意：

- 为使 DNSSEC 功能生效，请确保上级以及根 DNS 服务器必须支持并启动了 DNSSEC 功能。
- “**sdns dnssec import ds**”命令中 FTP URL（ftp://user:password@host:port/path）的“path”需使用相对路径。

26.5 SDNS 配置备份

SDNS 配置备份功能支持在本地磁盘备份当前运行的 SDNS 配置，还支持将 SDNS 配置备份到 SCP 服务器。

➢ **备份 SDNS 配置到本地**

```
Demo(config)#sdns config write file abc
Demo(config)#show sdns config file
Running configuration backup files:
length      date/time              name                    hmac-sm3
2336         Feb 03 2023 10:23:17 abc.cfg                 4089bc58207d8945ab84a77c886
495f7c1d0f214968f79699ccf11b9f061078c
Demo(config)#show sdns config file abc
Current configuration on disk
#Last configuration change at Fri Feb  3 10:23:17 2023 by array
#version Beta.APV.10.4.3.145 build on Mon Jan 30 16:15:11 2023
#smart DNS configuration
sdns on
sdns recursion off
sdns fulldns on
.....
Demo(config)#no sdns config file abc.cfg
```

```
Type "YES" to delete this configuration file: YES
Demo(config)#clear sdns config file
Type "YES" to delete all the configuration files: YES
10 file(s) removed
```

- **备份 SDNS 配置到 SCP 服务器**

```
Demo(config)#sdns config write scp “remote_server_name” “remote_user_name” abc
Password for remote_user_name@remote_server_name:
Warning: Permanently added ‘remote_server_name’ (ED25519) to the list of known hosts.
abc                                   100% 4919KB    1.6MB/s    00:03
```

26.6 SDNS 配置加载

SDNS 配置加载功能支持从本地和 SCP 服务器加载已备份的 SDNS 配置。管理员可以使用命令“**sdns config file**”加载备份在本地的 SDNS 配置，使用命令“**sdns config scp**” 加载备份在 SCP 服务器的 SDNS 配置。

- **从本地加载 SDNS 配置**

```
Demo(config)#sdns config file abc
```

- **从 SCP 服务器加载 SDNS 配置**

```
Demo(config)#sdns config scp “remote_server_name” “remote_user_name” abc
Password for remote_user_name@remote_server_name:
Warning: Permanently added ‘remote_server_name’ (ED25519) to the list of known hosts.
abc                                   100% 4919KB    1.6MB/s    00:03
Type “Yes” to display this config before load:NO
Type “YES” to load this config:YES
```

第27章 深度报文解析（DPI）

深度报文解析（DPI，Deep Packet Inspection）是相对于传统的仅分析 IP 报文四层以下的内容而言，包括源地址、目的地址、源端口、目的端口以及协议类型。另外，由于越来越多的网络协议不再使用固定的端口进行通信，基于报文端口的协议识别存在准确性问题。DPI 除了可以分析四层以下的内容外，还增加了应用层分析，能够识别各种应用层数据的应用协议及其内容。通过设备的 DPI 功能，管理员可以为不同应用协议的流量配置单独的 LLB 链路，防止某种应用的流量因为负载过高影响其他核心业务的服务质量。

27.1 应用协议类型

通过配置 DPI 功能，设备可以识别 200 多种预定义应用协议。管理员可以使用“**show dpi protocol name predefine**”命令查看 DPI 功能支持识别的所有预定义应用协议类型。

同时，DPI 功能还可以识别基于 IP 地址、主机名和端口号定义的自定义应用协议，管理员可以使用“**show dpi protocol name custom**”命令查看为 DPI 功能配置的所有自定义应用协议。

- 基于 IP 地址：基于源 IP 网段或目的 IP 网段定义应用协议，例如将来自 10.8.6.0/24 网段的流量定义为 application1 应用的流量；将目的 IP 为 192.168.0.0/16 网段的流量定义为 application2 应用的流量。

```
Demo(config)#dpi protocol rule ip application1 10.8.6.0 24
Demo(config)#dpi protocol rule ip application2 192.168.0.0 16
```

- 基于主机名：基于目标主机的域名定义应用协议，例如将目标主机为“xyz.com”的流量定义为 application3 应用的流量。

```
Demo(config)# dpi protocol rule host application3 xyz.com
```

- 基于端口：基于应用的端口号定义应用协议，四层协议可以为 UDP 或 TCP 协议，例如将端口号为 8008 的 UDP 流量定义为 application4 应用的流量，将端口号为 9009 的 TCP 流量定义为 application5 应用的流量。

```
Demo(config)#dpi protocol rule port application4 8008 udp
Demo(config)#dpi protocol rule port application5 9009 tcp
```

27.2 配置文件

定义了 DPI 需要识别的应用协议类型后，管理员需要手动为其导入相应的配置文件（通过服务提供商获得），然后将配置文件与指定的应用协议关联。

执行以下命令导入配置文件：

dpi import profile <profile_name> <url>

执行以下命令关联配置文件：

dpi attach profile <*application_name*> <*profile_name*>

对于 Office 365 应用，除了可以手动导入配置文件外，系统还支持从地址“https://support.content.office.net/en-us/static/O365IPAddresses.xml”自动下载和更新其配置文件。

27.3 LLB 链路

DPI 通过应用层分析能够识别出流量的应用协议和内容，这为基于不同的应用类型做进一步的路由提供了依据。设备支持为通过 DPI 识别出的应用协议配置单独的 LLB 链路，这样可以防止大流量应用过度占用已有资源，影响既有业务的质量，同时可以保证大流量的应用的稳定性。

管理员可执行“**dpi apply link route**”命令将指定的应用协议与专用 LLB 路由关联，那么属于该应用的流量将只通过专用的 LLB 路由转发。

例如，通过 LLB 路由“link1”转发 Office 365 的应用流量：

Demo(config)#**dpi apply link route office365 link1**

第28章 应用安全

28.1 防火墙

28.1.1 概述

本节将主要介绍设备防火墙（WebWall）的原理、高级功能、工作流程和配置方法。

通过设备的防火墙，网络管理员可以建立允许或拒绝规则来过滤各种通过网络的数据报，设备的防火墙支持对基于 IPv4 或 IPv6 地址的 TCP、UDP 和 ICMP 协议数据包的过滤。使用访问控制列表来建立允许和拒绝规则，再将访问控制列表应用到访问组，这样就可以将访问组绑定到网络设备的相关接口上。

设备的防火墙默认允许 SLB 健康检查。关于健康检查的内容，请参考“服务器负载均衡（SLB）”章节中“服务器负载均衡健康检查”部分的相关介绍。

28.1.2 防火墙的原理

设备防火墙是一款成熟的防火墙，它的设计兼顾了速度和安全两方面的因素。设备将诸多防火墙功能集成到一台设备当中，包含很多功能，例如四至七层负载均衡、SSL 加速器、缓存、身份验证和授权等。如下图所示。

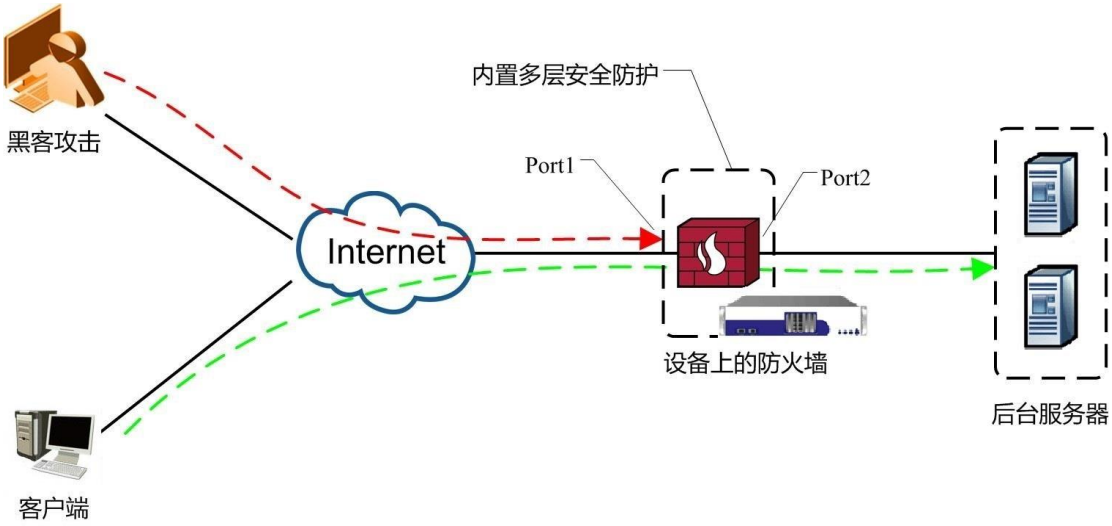

图28–1 设备防火墙

设备防火墙包含多种安全机制来防止 Web 服务器遭受攻击，包括：

- 访问控制列表过滤
- 应对 Syn-Flood、Fragmentation 和 DoS 攻击的防护

- 全状态数据报检测
- 单个数据报攻击防护

访问控制列表过滤功能利用设备的快速规则引擎严格控制着访问者的身份识别和访问权限。防火墙的访问过滤功能确保了设备运行了 1000 个以上的 ACL 规则时，不会有超过 1%的性能损失。

除了访问控制列表规律功能之外，设备防火墙还提供了全状态数据报检测和应对 Syn-Flood 攻击、碎片攻击、DoS 和单数据包攻击的机制。

设备防火墙的默认行为是拒绝，默认拒绝的含义是如果管理员没有在访问控制列表中建立任何允许通过的规则，那么任何数据报都无法通过设备。因此在我们第一次安装设备时，建议关闭防火墙，直到我们的所有配置完成。

注意：设备的防火墙功能默认是关闭的。如果要使用防火墙功能，首先需要使用命令“**webwall on**”启用该功能。

28.1.3 防火墙配置

28.1.3.1 配置场景

在设备上配置防火墙功能，确保防火墙能够按照以下规则处理管理流量和业务流量：

- 允许 IP 地址为 10.10.10.30 的客户端通过 22 端口来配置和管理设备（通过 SSH 访问）
- 允许 IP 地址为 10.10.10.30 的客户端通过 8888 端口配置和管理设备（通过 WebUI 访问）。
- 允许除 IP 地址为 10.10.10.30 以外的所有客户端通过 80 端口访问虚拟 IP 地址 10.10.0.10。
- 允许所有内网客户端 Ping 设备的 Port2 接口 IP 地址 192.168.10.1。

该配置举例所基于的网络拓扑请参见下图。

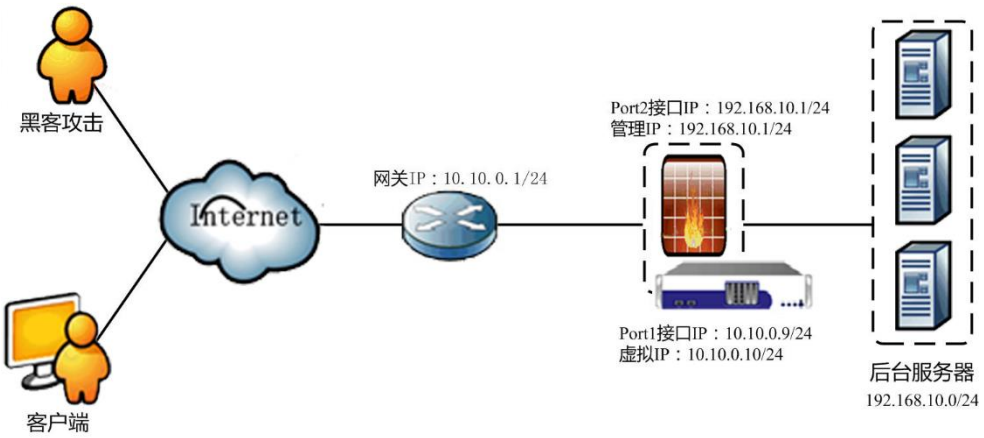

图28–2 设备防火墙配置

下表列出了配置防火墙所需要使用到的配置命令，相关命令的描述信息，请参考命令行使用手册。

表28–1 设备防火墙配置命令

配置操作	命令行
配置访问组	**accessgroup** *<accesslist_id> <interface>*
配置 ACL 访问控制规则	**accesslist permit icmp echoreply** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist permit icmp echorequest** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist permit tcp** *<source_ip> {source_mask|source_prefix} <source_port> <destination_ip> {destination_mask|destination_prefix} <destination_port> <accesslist_id>* **accesslist permit udp** *<source_ip> {source_mask|source_prefix} <source_port> <destination_ip> {destination_mask|destination_prefix} <destination_port> <accesslist_id>* **accesslist permit esp** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist permit ah** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist deny icmp echoreply** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist deny icmp echorequest** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist deny tcp** *<source_ip> {source_mask|source_prefix}*

配置操作	命令行
	<source_port> <destination_ip> {destination_mask|destination_prefix} *<destination_port> <accesslist_id>* **accesslist deny udp** *<source_ip> {source_mask|source_prefix}* *<source_port> <destination_ip> {destination_mask|destination_prefix}* *<destination_port> <accesslist_id>* **accesslist deny esp** *<source_ip> {source_mask|source_prefix}* *<destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist deny ah** *<source_ip> {source_mask|source_prefix}* *<destination_ip> {destination_mask|destination_prefix} <accesslist_id>*
启动/关闭防火墙功能	**webwall** *<interface>* **on** *[mode]* **webwall** *<interface>* **off**
检查防火墙配置	**show interface** **show accesslist** **show accessgroup**

28.1.3.2 配置步骤

- **配置访问组**

1. 配置访问组 50，用于关联各种不同类型的 ACL 访问规则。

```
Demo(config)#accessgroup 50 port2
```

2. 配置访问组 100，用于关联管理 IP 地址相关的 ACL 访问规则。

```
Demo(config)#accessgroup 100 port2
```

3. 配置访问组 150，用于关联虚拟 IP 地址相关的 ACL 访问规则。

```
Demo(config)#accessgroup 150 port1
```

- **配置 ACL 访问规则**

1. 配置允许 IP 地址为 10.10.10.30 的客户端通过 22 端口来配置和管理设备（通过 SSH 访问）

```
Demo(config)#accesslist permit tcp 10.10.10.30 255.255.255.255 0 192.168.10.1
255.255.255.255 22 100
```

2. 配置允许 IP 地址为 10.10.10.30 的客户端通过 8888 端口配置和管理设备（通过 WebUI 访问）。

```
Demo(config)#accesslist permit tcp 10.10.10.30 255.255.255.255 0 192.168.10.1
255.255.255.255 8888 100
```

3. 配置允许除 IP 地址为 10.10.10.30 以外的所有客户端通过 80 端口访问虚拟 IP 地址 10.10.0.10。

```
Demo(config)#accesslist permit tcp 0.0.0.0 0.0.0.0 0 10.10. 0.10 255.255.255.255 80 150
Demo(config)#accesslist deny tcp 10.10.10.30 255.255.255.255 0 10.10. 0.10 255.255.255.255 80 150
```

4. 配置允许所有的内网客户端 Ping 设备的 Port2 接口 IP 地址 192.168.10.1。

```
Demo(config)#accesslist permit icmp echorequest 192.168.10.0 255.255.255.255 192.168.10.1 255.255.255.255 50
Demo(config)#accesslist permit icmp echoreply 192.168.10.1 255.255.255.255 192.168.10.0 255.255.255.0 50
```

- **启用防火墙**

执行下面的命令启用防火墙。

```
Demo(config)#webwall port2 on
Demo(config)#webwall port1 on
```

启用防火墙之前，需注意：

- 如果配置了域名解析服务器，并且需要在 DNS 流量通过的接口上打开防火墙，需要配置相应的访问规则以允许访问 53 端口的流量通过。
- 如果需要在“**synconfig to**”和“**synconfig from**”命令使用的接口上打开防火墙，需要手动添加访问规则来允许访问端口 65519 的数据流通过，否则同步会失败。在 HA 应用场景，可以在本地和对端节点上分别配置“**ha synconfig bootup on**”命令来允许同步数据通过；如果不配置该命令，也可以采用手动添加访问规则的方法。

在启用防火墙后，如果需要调整防火墙配置，请注意正在使用中的 SSH 或 WebUI 访问会话可能由于配置失误而被中断。

- **检查和测试配置**

在配置 ACL 访问规则条目和访问组之后，可以使用下列命令来检查配置的结果。

```
Demo(config)#show accesslist
accesslist permit tcp 10.10.10.30 255.255.255.255 0 192.168.10.1 255.255.255.255 22 100
accesslist permit tcp 10.10.10.30 255.255.255.255 0 192.168.10.1 255.255.255.255 8888 100
accesslist permit tcp 0.0.0.0 0.0.0.0 0 10.10. 0.10 255.255.255.255 80 150
accesslist deny tcp 10.10.10.30 255.255.255.255 0 10.10. 0.10 255.255.255.255 80 150
accesslist permit icmp echorequest 192.168.10.0 255.255.255.255 192.168.10.1 255.255.255.255 50
accesslist permit icmp echoreply 192.168.10.1 255.255.255.255 192.168.10.0 255.255.255.0 50
Demo(config)#show accessgroup
accessgroup 50 port2
accessgroup 100 port2
accessgroup 150 port1
```


如果访问规则出了问题，可以简化配置。如果有多个访问组，可以尝试每次应用一个访问组来判断是哪个访问规则引起的问题。也可以把防火墙关掉，确认是否是防火墙引起的问题。

使用“**show webwall**”命令可以查看哪些接口上开启了防火墙功能，例如：

```
Demo(config)#show webwall
webwall "port2" on 0
webwall "port3" on 0
```

使用“**show interface**”命令可以查看防火墙的丢包状况，见如下示例中的斜体字体。

```
Demo(config)#show interface
port2(port2): flags=2008842<BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 00:25:90:39:97:f1
        media: autoselect
        status: no carrier
        webwall status: ON
        Hardware is i82574l
        Input queue: 0/4096 (size/max)
                total: 0 packets, good: 0 packets, 0 bytes
                broadcasts: 0, multicasts: 0
                0 64 bytes, 0 65-127 bytes,0 128-255 bytes
                0 255-511 bytes,0 512-1023 bytes,0 1024-1522 bytes
                0 input errors
                0 runts, 0 giants, 0 Jabbers, 0 CRCs
                0 Flow Control, 0 Fragments, 0 Receive errors
                0 Driver dropped, 0 Frame, 0 Lengths, 0 No Buffers
                0 overruns, Carrier extension errors: 0
        Output queue: 0/4096 (size/max)
                total: 0 packets, good:    0 packets, 0 bytes
                broadcasts: 0, multicasts: 0
                0 64 bytes, 0 65-127 bytes,0 128-255 bytes
                0 255-511 bytes,0 512-1023 bytes,0 1024-1522 bytes
                0 output errors
                0 Collisions, 0 Late collisions, 0 Deferred
                0 Single Collisions, 0 Multiple Collisions, 0 Excessive collisions
        0 lost carrier, 0 WDT reset
        packet drop (not permit): 0（通过默认的拒绝访问规则丢弃的报文数量。）
                tcp 0           udp 0           icmp 0          ah 0            esp 0
        packet drop (deny): 0（通过自定义的拒绝访问规则丢弃的报文数量。）
                tcp 0           udp 0           icmp 0          ah 0            esp 0
        5 minute input rate 0 bits/sec, 0 packets/sec
```

```
    5 minute output rate 0 bits/sec, 0 packets/sec
```

28.2 Web 应用防火墙

28.2.1 概述

Web 应用防火墙（Web Application Firewall，WAF）功能可以为 Web 站点和 Web 应用系统提供实时的安全防护。WAF 功能可以有效地缓解 Web 站点和 Web 应用系统面临地常见威胁和恶意攻击，比如 SQL 注入攻击（Structured Query Language injection，SQL injection）。

设备的 WAF 功能具备如下特点：

- HTTP/HTTPS 流量检测：WAF 功能能够完整地解析 HTTP 协议，包括报文头部、参数及载荷。此外，WAF 功能还可以与 SSL 加速功能配合解析 HTTPS 协议。
- 规则引擎：WAF 功能提供了一套易用灵活的规则引擎。管理员既可以通过应用预定义规则集识别常见的 Web 攻击，也可以通过自定义规则集识别并阻止其他攻击。另外 WAF 功能与开源的入侵检测与防护引擎 ModSecurity 兼容，管理员可以在 WAF 中参考使用 ModSecurity 的核心规则集。
- 虚拟补丁：WAF 的虚拟补丁功能可以通过在设备上修复 Web 应用漏洞来抵御攻击，相比于直接在 Web 应用上修复漏洞，更加方便快速。
- 流量日志：WAF 功能能够实时监控 HTTP 和 HTTPS 流量的信息，并记录完整的事件交互日志；WAF 功能还可以与第三方远程审计日志服务器配合使用，通过第三方远程审计日志服务器提供统计分析审计日志功能与安全事件报表功能。
- 与其他功能配合工作：WAF 功能可以与服务器负载均衡、缓存、压缩及 SSL 加速等功能配合使用。
- 灵活的部署能力：WAF 功能支持透明模式和反向代理模式。

28.2.2 基本概念

在 WAF 中，每个虚拟服务可以关联到相同或不同的 Security Profile。Security Profile 中包含一个或多个规则集，规则集中包含一条或多条规则。

28.2.2.1 规则

规则是 WAF 功能的基本指令，WAF 功能通过分析规则并根据 HTTP 和 HTTPS 流量和规则的匹配结果采取相应的动作来抵御可能的 Web 攻击。

28.2.2.1.1 规则语法

编写规则时应遵循的规则语法为：SecRule 变量 运算符 动作。下图中为一个规则的示例：

图28–3 示例

其中各部分的含义如下：

- SecRule：创建一条使用特定运算符来检查特定变量，并在变量匹配运算符时执行相应动作的规则。
- 变量：指定被检查变量的名称。每条规则必须指定一个或多个变量。变量值为各类 HTTP 参数。例如：上图中，“REQUEST_URI”为变量，指定被检查的变量为请求体。
- 运算符：描述如何对变量进行检查。运算符采用正则表达式，但 WAF 功能也提供很多可用的运算符，使用“@”即可指定要用何种运算符，例如：上图中，“@beginsWith abc”为运算符，指定检查内容以“abc”开头的 URL 地址。
- 动作：描述当变量匹配运算符时，WAF 将执行何种动作。例如：上图中，“phase:2,auditlog,deny,msg:'Failed to parse request body.'”为规则的动作，指定“规则在位置“phase:2”执行，阻断请求，在客户端浏览器网页上显示错误信息“Failed to parse request body.”，并将请求体的内容记录在审计日志中”。

WAF 对规则进行处理时，在不同时刻从 HTTP 和 HTTPS 报文中的请求头（Request Headers）、请求体（Request Body）、响应头（Response Headers）或响应体（Response Body）中提取相关信息并采取规则中定义的动作。对应的执行位置为 phase:1、phase:2、phase:3 和 phase:4。

关于 WAF 的规则及其执行位置的具体信息，请联系公司技术支持获取相关文档。

28.2.2.1.2 动作

动作是规则对匹配了规则的 HTTP 和 HTTPS 流量所采取的行为。

规则的动作包括：

- deny：当系统检测到攻击时，向客户端发送错误页面指示请求被阻断，并记录日志。
- pass：当系统检测到攻击时，仅记录日志。

- block：当系统检测到攻击时采用规则的默认动作，即规则集的动作。关于规则的默认动作的具体信息，请参考“规则集”小节中的相关介绍。

28.2.2.2 规则集

规则集包含一条或多条规则。WAF 功能将 HTTP 和 HTTPS 流量与规则集中的每条规则进行匹配，并对匹配的流量采取相应的动作。规则集包括预定义规则集和自定义规则集。预定义规则集包括 SQL 注入攻击和 XSS（Cross Site Scripting，跨站脚本攻击）。自定义规则集允许管理员使用自定义规则来抵御攻击。管理员可以根据实际情况通过设备的命令行界面或 WebUI 界面导入自定义规则集，或通过 WebUI 界面导出自定义规则集。

规则集根据优先级由高到低执行。对于优先级相同的预定义规则集和自定义规则集，执行顺序不分先后。

规则集可以为规则配置默认动作。规则集动作包括“deny”和“pass”。在预定义规则集中的所有规则的动作都为“block”，系统将执行预定义规则集的动作。对于自定义规则集，每条规则的动作可以为：“deny”，“pass”或“block”等。当规则动作为“deny”或“pass”时，系统将执行规则的动作；当规则动作为“block”时，系统执行自定义规则集的动作。

注意：如果管理员配置了指令<Location>或<LocationMatch>并将对应的自定义规则的动作配置为“block”，系统将忽略自定义规则集的动作并执行“pass”动作。

28.2.2.3 Security Profile

Security Profile 包含一个或多个规则集。通过将多个规则集添加到 Security Profile 中，并将 Security Profile 和虚拟服务关联，管理员可以为虚拟服务配置 Security Profile 以抵御各类攻击而无需将各个规则集单独关联到虚拟服务，使配置更加简单。此外，管理员可以通过控制 Security Profile 的工作模式来避免改变每一个规则集的动作，从而可以灵活的控制系统对攻击的处理方式。关于工作模式的细节，请参考“工作模式”小节中的相关介绍。

28.2.2.4 工作模式

Security Profile 的工作模式包括：

- 被动（passive）模式：当系统检测到攻击时，记录日志、但是不阻断流量并且不向客户端发送错误页面。
- 主动（active）模式：当系统检测到攻击时，记录日志、阻断流量并向客户端发送错误页面。

28.2.2.5 日志

设备已有的日志功能可以记录全部的客户端访问信息，日志的安全级别为“警告（warning）”。在此基础上，WAF 功能还提供了审计日志功能。

审计日志功能将记录 HTTP 和 HTTPS 流量的信息。审计日志可以存储在本地磁盘上，也可以发送到发送审计日志到第三方远程审计日志远程服务器。关于如何安装第三方远程审计日志服务器，请联系公司技术支持获取相关文档。

注意：审计日志不能同时存储在本地磁盘和第三方审计日志服务上。

28.2.3 配置示例

28.2.3.1 配置目标

添加用于抵御 CSRF 攻击的自定义规则集，将该规则集添加到 Security Profile，并将该 Security Profile 与虚拟服务关联，从而设备可以为该虚拟服务抵御 CSRF 攻击。

前置条件：

- 已经定义了七层的虚拟服务 vs1。关于虚拟服务的配置，请参见 SLB 章节中的相关介绍。
- 已经定义了自定义规则集 s1000。

28.2.3.2 配置示例

1. 添加 Security Profile 并指定工作模式。

例如：

```
Demo(config)#waf profile name p1
Demo(config)#waf profile mode active p1
```

2. 导入自定义规则集。

例如：

```
Demo(config)#waf import " s1000" http://192.168.10.20/c1.txt ruleset
```

3. 为 Security Profile 添加自定义规则集。

例如：

```
Demo(config)# waf profile custom "p1" " s1000" deny 1000
```

4. 关联虚拟服务和 Security Profile。

例如：

```
Demo(config)# waf profile assign p1 vs1
```

5. 启用审计日志。

例如：

```
Demo(config)# waf log audit on
```

28.3 高级访问控制

28.3.1 概述

系统支持配置高级访问控制（ACL，Access Control List）来限制网络流量，控制客户端的访问行为，从而防止大流量的恶意攻击，提高内网资源的可用性。该功能通过访问控制规则实现，其作用在于对报文的源 IP、协议、访问的域名等进行检查，根据报文是否匹配访问控制列表里规定的条件决定是否允许报文通过或采取其他处理方式。

设备的高级访问控制功能支持三种访问控制规则：ACL 规则、HTTP ACL 规则和 DNS ACL 规则，适用于 IPv4 和 IPv6 环境。

- ACL 规则限制虚拟服务上允许的每秒新建连接数（connections per second, CPS）和并发连接数（concurrent connections, CC），可应用于所有基于 TCP 协议的虚拟服务。
- HTTP ACL 规则限制虚拟服务上允许的每秒请求数（requests per second, RPS）和下载速度，仅应用于 HTTP 和 HTTPS 协议类型的虚拟服务。
- DNS ACL 规则限制虚拟服务上允许的每秒请求，仅应用于 DNS 虚拟服务和 SDNS。

注意：

- ACL 规则、HTTP ACL 规则和 DNS ACL 规则对系统的性能有一定的影响。
- 配置的 ACL 规则、HTTP ACL 规则和 DNS ACL 规则只对后续创建的连接生效，不对已有连接生效。
- 当访问使用 insert cookie 策略的 HTTP 或 HTTPS 虚拟服务时，携带设备插入的 cookie 的客户端请求不受 ACL 规则和 HTTP ACL 规则的限制。

28.3.2 ACL 规则

ACL 规则支持两种控制模式和两种控制类型。

控制模式

- “total”模式表示指定子网中的所有客户端作为一个整体受 ACL 规则的限制。
- “per-ip”模式表示指定子网中每个客户端单独受 ACL 规则的限制。

控制类型

- “cps”类型表示对每秒新建连接数进行限制。
- “concurrent”类型表示对并发连接数进行限制。

管理员在配置 ACL 规则时，可以根据需要来组合控制模式和控制类型。对于同一个子网，可以组合出四种 ACL 规则，如下表所示。

表28–2 同一个子网的四种 ACL 规则

模式　　类型	cps	concurrent
total	子网中所有客户端上的每秒新建连接数总和不能超过设定的最大值。	子网中所有客户端上的并发连接数总和不能超过设定的最大值。
per-ip	子网中每个客户端上的每秒新建连接数不能超过设定的最大值。	子网中每个客户端上的并发连接数不能超过设定的最大值。

设备支持为同一子网配置以上组合中的一种或者几种规则。如果为一个子网配置了多种规则，则这些规则同时生效。

- **子网嵌套**

ACL 规则支持子网嵌套。如果一个客户端的 IP 地址命中多个子网，该客户端受命中的最小子网的所有规则的限制，同时也受最小子网的所有上级子网的“total”规则的限制。系统最多允许嵌套三层子网。

例如：

10.8.0.0/16：同时配置“total”和“per-ip”规则

10.8.1.0/24：同时配置“total”和“per-ip”规则

则，限制 10.8.1.0/24 子网客户端的规则有：

10.8.1.0/24：“total”和“per-ip”规则

10.8.1.0/16：“total”规则

- **ACL 白名单**

管理员可以通过创建 ACL 白名单，允许其中的客户端不受 ACL 规则的限制。当 ACL 规则和白名单同时应用于指定的虚拟服务时，白名单的处理优先级高于规则的处理优先级。

如果一个客户端的 IP 地址属于某个白名单定义的子网范围，则该客户端不受任何 ACL 规则的限制。但是，该客户端的每秒新建连接数和并发连接数将分别计入所属子网的每秒新建连接总数和并发连接总数中。

例如：

规则：10.8.1.0/24；total concurrent≤10,000

白名单：10.8.1.10/32

如果 IP 地址为 10.8.1.10 客户端的并发连接数为 1000，则子网 10.8.1.0/24（不包含 10.8.1.10）允许的最大并发连接数为：9000。

- **应用范围**

ACL 规则和 ACL 白名单既可以应用于单个虚拟服务，也可以应用于全局虚拟服务。高级访问控制功能支持所有基于 TCP 协议的虚拟服务。

当访问指定虚拟服务时，客户端既要受命中的应用于该虚拟服务的规则的限制，又要受命中的应用于全局虚拟服务的规则的限制。

例如：

rule1：0.0.0.0/0，per-ip concurrent≤1000

rule2：0.0.0.0/0，per-ip concurrent≤900

rule1 应用于 vs1；rule2 应用于 global

则 IP 地址为 10.8.1.10 客户端在访问虚拟服务 vs1 时，允许的最大并发连接数为 900。

- **ACL 规则在基于 HTTP 的服务中的应用**

对于七层的 HTTP 和 HTTPS 虚拟服务，如果采用了 insert cookie 策略，设备将在返回给客户端的响应消息中插入 cookie，用于识别该客户端。当客户端再次访问虚拟服务时，如果请求中带有设备插入的 cookie，则该客户端不受任何 ACL 规则的限制。但是，该客户端的每秒连接数和并发连接数将分别计入命中的子网的每秒新建连接总数和并发连接总数中。

28.3.3 HTTP ACL 规则

HTTP ACL 规则支持对于访问 HTTP 和 HTTPS 虚拟服务的客户端的 RPS 和下载速度进行控制。RPS 控制类型的 HTTP ACL 规则被触发时，系统将会按照规则

控制客户端的访问行为或采取相应的处理方式。下载速度控制类型的 HTTP ACL 规则用于将客户端的下载速度控制在指定的范围之内。

28.3.3.1 规则类型

系统支持静态和动态两种 HTTP ACL 规则。静态 HTTP ACL 规则由管理员通过 CLI 或 WebUI 界面手动配置，动态 HTTP ACL 规则由安全模块根据系统当前是否遭受攻击而自动生成。

- **静态 HTTP ACL 规则**

静态 HTTP ACL 规则允许管理员自定义对 HTTP 流量的控制规则，管理员可以执行“**acl http rule**”命令配置静态 HTTP ACL 规则，然后通过“**acl http apply rule virtual**”命令将规则应用到一个 HTTP 或 HTTPS 虚拟服务。

acl http rule *<rule_name> <client_ip> {netmask|prefix} <acl_type> <max_limit> <condition>*
acl http apply rule virtual *<rule_name> <virtual_service>*

- **动态 HTTP ACL 规则**

系统会在 DDoS 攻击防御功能开启后生成动态 HTTP ACL 规则（通过“**ddos on**”启用）。动态 HTTP ACL 规则是对传统的 HTTP ACL 规则（静态 HTTP ACL 规则）功能的一种增强。除了自定义的静态 HTTP ACL 规则，系统在受到 HTTP DDoS 攻击时，如果 HTTP 验证未开启，系统会为可疑客户端 IP 生成一条限制其访问特定虚拟服务的动态 HTTP ACL 规则。该规则会规定 RPS 阈值为 100，下载速度的阈值为 1024 KB/s。如果匹配该 HTTP ACL 规则的客户端的 RPS 达到 100，其后续请求将被丢弃，直到该客户端的 RPS 值降到 100 以内。同时匹配该 HTTP ACL 规则的客户端的下载速度会被控制在 1024 KB/s 以内。如果系统在遭受 HTTP DDoS 攻击时开启了 HTTP 验证机制，并且可疑客户端被验证为非法，该客户端的 IP 地址会被记录到 DDoS 黑名单。

管理员可以通过“**show acl http rule**”命令查看静态 HTTP ACL 规则的配置以及系统自动生成的 HTTP ACL 规则。

注意：

- 静态 HTTP ACL 规则的优先级高于动态 HTTP ACL 规则，当动态和静态 HTTP ACL 规则被同时触发时，系统会优先执行静态 HTTP ACL 规则。
- HTTP ACL 规则只有应用到虚拟服务上才会生效

28.3.3.2 控制类型和处理方式

HTTP ACL 规则支持两种控制类型：控制 RPS（Request Per Second，每秒请求数）和控制下载速度。

1. 控制 RPS

RPS 控制类型的 HTTP ACL 规则对一个客户端或子网内所有客户端每秒内发送的 HTTP 请求数量进行统计，当 HTTP 请求数量达到指定的阈值时，系统会对后续收到的 HTTP 请求采取规则里规定的处理方式，从而防止大数量的 HTTP 请求造成服务器过载，提高服务器的可用性。

匹配条件

系统会依据 HTTP ACL 规则里规定的匹配条件来统计 RPS。当 HTTP 请求满足如下条件时，将被计入一个 HTTP ACL 规则的 RPS 计数里：

- 客户端的 IP 地址在该 HTTP ACL 规则的子网范围内。
- 请求的 URL 匹配 HTTP ACL 规则中的 URL 正则表达式。
- 使用的 HTTP 请求方法匹配 HTTP ACL 规则的 HTTP 方法设置。系统支持统计使用以下方法的 HTTP 请求：
  - ALL：表示所有的 HTTP 请求方法，即所有的 HTTP 请求都会计入 RPS 中。
  - GET、POST、HEAD、DELETE 或 PUT：只有使用指定方法的 HTTP 请求会计入 RPS 中。

处理方式

当 HTTP ACL 规则对应的 RPS 计数达到指定的阈值后，系统会对后续收到的匹配规则的客户端请求采取处理措施，系统支持两种处理方式：

- 返回错误页面。默认情况下，系统会返回系统内置的 480 错误页面。管理员也可以通过错误页面导入或错误页面重定向来定制错误页面的格式和内容：
  - 错误页面导入：通过命令“**http import error**”和“**http error**”导入一个自定义的 480 错误页面，那么系统会返回该自定义 480 错误页面。
  - 错误页面重定向：通过命令“**http redirect error**”配置一个重定向响应规则，那么系统会返回一个重定向响应，将客户端重定向到响应内指定的 URL 地址。
  - 如果错误页面导入和错误页面重定向配置同时存在，错误页面重定向配置优先生效。
- 发送 RST 报文重置连接。
- 示例：

```
Demo(config)#acl http rule rule1 10.8.6.0 24 rps 1000 “url=<regex>abc* method=GET action=errorpage”
```

- 在该例子中，位于 10.8.6.0/24 子网的客户端，在访问匹配“abc*”的 URL 地址时，每秒的 HTTP GET 请求数最多只能为 1000，对于超过该限制的后续客户端请求，系统将返回内置的 480 错误页面。

1. 控制下载速度

下载速度控制类型的 HTTP ACL 规则将每个客户端的下载速度控制在 HTTP ACL 规则里规定的阈值以内。

匹配条件

系统会依据 HTTP ACL 规则里规定的匹配条件来探测客户端的下载速度，当 HTTP 请求满足如下条件时，下载速度将受到 HTTP ACL 规则的控制：

- 客户端的 IP 地址在该 HTTP ACL 规则的子网范围内。
- 请求的 URL 匹配 HTTP ACL 规则中的 URL 正则表达式。

示例：

```
Demo(config)#acl http rule rule1 10.8.6.0 24 throughput 1000 “url=<regex>abc*”
```

在该例子中，位于 10.8.6.0/24 子网的客户端，在访问匹配“abc*”的 URL 地址时，下载速度不能超过 1000 KB/s。

28.3.3.3 子网嵌套

HTTP ACL 规则支持最多三层子网嵌套。

如果一个客户端的 IP 地址命中多个子网，而且 HTTP 请求同时匹配多个子网层对应的 HTTP ACL 规则，该客户端只受最小子网对应的 HTTP ACL 规则的限制。

例如：客户端 IP 地址为 10.8.6.11，如果请求同时匹配以下规则，系统将执行子网 10.8.6.0/24 对应的 HTTP ACL 规则的限制。

- 子网 10.0.0.0/8 对应的 HTTP ACL 规则
- 子网 10.8.0.0/16 对应的 HTTP ACL 规则
- 子网 10.8.6.0/24 对应的 HTTP ACL 规则

如果该请求不匹配子网 10.8.6.0/24 对应的任何 HTTP ACL 规则，那么系统将执行子网 10.8.0.0/16 对应的 HTTP ACL 规则。

28.3.4 DNS ACL 规则

DNS ACL 规则用于控制每秒 DNS 查询请求的个数（RPS），适用于 DNS 负载均衡和 SDNS 两种应用场景。DNS ACL 规则会规定一个 RPS 阈值，当每秒 DNS 查询请求的个数达到该阈值时，后续的 DNS 查询请求将被丢弃。

在 DNS 负载均衡场景，DNS ACL 规则控制每个 DNS 虚拟服务上的 DNS 查询请求数。

在 SDNS 场景，DNS ACL 规则控制 BIND9 域名服务器上的 DNS 查询请求数。

28.3.4.1 规则类型

系统支持静态和动态两种 DNS ACL 规则。静态 DNS ACL 规则由管理员通过 CLI 或 WebUI 界面手动配置，动态 DNS ACL 规则由系统根据当前的流量负载学习后自动生成。

- **静态 DNS ACL 规则**

静态 DNS ACL 规则允许管理员自定义对 DNS 流量的控制规则，管理员可以执行“**acl dns rule**”命令配置静态 DNS ACL 规则，然后通过“**acl dns apply rule virtual**”命令将规则应用到一个 DNS 虚拟服务，或者通过“**acl dns apply rule sdns**”命令将规则应用到全局 SDNS 或一个 SDNS 域名。

acl dns rule *<rule_name> <client_ip> {netmask|prefix} <acl_mode> <dns_type> <max_limit>*
acl dns apply rule virtual *<rule_name> <virtual_service>*
acl dns apply rule sdns *<rule_name> [host_name]*

- **动态 DNS ACL 规则**

系统会在 DDoS 攻击防御和 DNS DDoS 攻击防御功能开启后生成动态 DNS ACL 规则（通过“ddos dns on”启用）。动态 DNS ACL 规则是对传统的 DNS ACL 规则（静态 DNS ACL 规则）功能的一种增强。在网络的实际运行中，流量负荷会因为用户上网行为的影响而发生浮动，而传统的 DNS ACL 规则无法根据流量的变化动态地控制流量。

为了提供一种能基于网络流量变化而动态控制的 DNS ACL 规则，同时为了简化配置，系统支持动态 DNS ACL 规则。动态 DNS ACL 规则由系统根据当前的流量负载情况自动生成。无论管理员是否配置了静态 DNS ACL 规则，系统都会在运行过程中不间断地学习网络中的 DNS 访问流量，通过流量学习和计算动态生成 DNS ACL 规则，并根据流量的变化对生成的 DNS ACL 规则进行动态调整。

管理员可以通过“**show acl dns rule**”命令查看静态 DNS ACL 规则的配置以及系统自动生成的 DNS ACL 规则。

注意：

- 静态 DNS ACL 规则的优先级高于动态 DNS ACL 规则，当动态和静态 DNS ACL 规则被同时触发时，系统会优先执行静态 DNS ACL 规则。
- DNS ACL 规则只有应用到虚拟服务或 SDNS 上才会生效。

28.3.4.2 控制模式

控制模式表示 DNS ACL 规则在指定子网内的生效范围，有“total”和“per-ip”两种模式。

- “total”模式：指定子网中的所有客户端作为一个整体受 DNS ACL 规则的限制。
- “per-ip”模式：指定子网中每个客户端单独受 DNS ACL 规则的限制。

匹配条件

系统会依据 DNS ACL 规则里规定的匹配条件来统计 RPS。当 DNS 查询请求满足如下条件时，将被计入一个 DNS ACL 规则的 RPS 计数里：

- 客户端的 IP 地址在该 DNS ACL 规则的子网范围内。
- DNS 查询请求匹配 DNS ACL 规则里设置的 DNS 资源记录类型。设备支持统计以下类型的 DNS 资源记录。
  - ALL：表示所有 DNS 查询请求，无论请求的是哪一种 DNS 资源类型，都会计入 RPS 中。
  - A、NS、MD、MF、CNAME、SOA、MB、MG、MR、NULL、WKS、PTR、HINFO、MINFO、MX、TXT、AAAA、OTHERS、ANY：只有指定类型的 DNS 查询请求才会计入 RPS 中。

示例：

```
acl dns rule dnsrule1 192.168.0.0 16 per-ip AAAA 100
```

在该例子中，对于位于 192.168.0.0/24 子网的任何一个客户端，如果该客户端每秒发起的 AAAA 类型的 DNS 查询请求的数量达到了 100 个，那么系统会丢弃后续收到的来自该客户端的 AAAA 类型的 DNS 查询请求。

28.3.4.3 子网嵌套

DNS ACL 规则支持最多三层子网嵌套。

如果一个客户端的 IP 地址命中多个子网，而且 DNS 查询请求同时匹配多个子网层对应的 DNS ACL 规则，该客户端受最小子网对应的 DNS ACL 规则的限制，同时受最小子网的所有上级子网的“total”规则的限制。

例如：客户端 IP 地址为 10.8.6.11，如果请求同时匹配以下规则（都是基于 per-ip），那么系统将执行子网 10.8.6.0/24 对应的 DNS ACL 规则。

- 子网 10.0.0.0/8 对应的 DNS ACL 规则
- 子网 10.8.0.0/16 对应的 DNS ACL 规则
- 子网 10.8.6.0/24 对应的 DNS ACL 规则

如果该请求不匹配子网 10.8.6.0/24 对应的任何 DNS ACL 规则，那么系统将执行子网 10.8.0.0/16 对应的 DNS ACL 规则。

28.3.5 配置

28.3.5.1 配置 ACL 规则

- **配置步骤**

在配置 ACL 规则时，首先需要添加 ACL 规则，然后把规则应用于单个虚拟服务或者全局虚拟服务。

1. 执行如下命令添加 ACL 规则：

acl rule <*rule_name*> <*client_ip*> {*netmask*|*prefix*} <*acl_mode*> <*acl_type*> <*max_limit*>

例如：

Demo(config)#**acl rule rule1 61.130.10.0 255.255.255.0 total cps 1000000** Demo(config)#**acl rule rule2 61.130.10.0 255.255.255.0 total concurrent 50000** Demo(config)#**acl rule rule3 61.130.10.0 255.255.255.0 per-ip cps 10000** Demo(config)#**acl rule rule4 61.130.10.0 255.255.255.0 per-ip concurrent 500**

2. 执行如下命令应用 ACL 规则到虚拟服务：

acl apply rule virtual <*rule_name*> <*virtual_service*>

例如：

Demo(config)#**acl apply rule virtual rule1 tcp_vs1** Demo(config)#**acl apply rule virtual rule3 tcp_vs1** Demo(config)#**acl apply rule virtual rule2 http_vs1** Demo(config)#**acl apply rule virtual rule4 http_vs1**

在配置 ACL 白名单时，首先需要添加 ACL 白名单，然后把白名单应用于单个虚拟服务或者全局虚拟服务。ACL 白名单为可选配置。

3. 执行如下命令添加 ACL 白名单：

acl whitelist <*whitelist_name*> <*client_ip*> {*netmask*|*prefix*}

例如：

Demo(config)#**acl whitelist whitelist1 61.130.10.10 255.255.255.255**

4. 执行如下命令应用白名单到虚拟服务：

acl apply whitelist virtual <*whitelist_name*> <*virtual_service*>

例如：

Demo(config)#**acl apply whitelist virtual whitelist1 tcp_vs1**

```
Demo(config)#acl apply whitelist virtual whitelist1 http_vs1
```

➢ **配置结果**

完成上述配置后，设备将：

在客户端访问虚拟服务 tcp_vs1 时：

限制 61.130.10.0/24 子网中的所有客户端的每秒新建连接总数不能超过 1,000,000。

- 限制 61.130.10.0/24 子网中的每个客户端的每秒新建连接数不能超过 10,000。
- 不限制 IP 地址为 61.130.10.10 的客户端的每秒新建连接数。

在客户端访问虚拟服务 http_vs1 时：

- 限制 61.130.10.0/24 子网中的所有客户端的并发连接总数不能超过 50,000。
- 限制 61.130.10.0/24 子网中的每个客户端的并发连接数不能超过 500。
- 不限制 IP 地址为 61.130.10.10 的客户端的并发连接数。
-

28.3.5.2 配置 HTTP ACL 规则

➢ **配置步骤**

1. 执行如下命令添加 HTTP ACL 规则：

```
acl http rule <rule_name> <client_ip> {netmask|prefix} <acl_type> <max_limit> <condition>
```

例如：

```
Demo(config)#acl http rule rule1 10.8.6.0 24 rps 1000 “url=<regex>abc* method=ALL
action=reset”
Demo(config)#acl http rule rule2 192.168.0.0 16 throughput 1000 “url=<regex>xyz*”
Demo(config)#acl http rule rule3 192.168.1.0 24 throughput 100 “url=<regex>xyz*”
```

2. 执行如下命令应用 HTTP ACL 规则到虚拟服务：

```
acl http apply rule virtual <rule_name> <virtual_service>
```

例如：

```
Demo(config)#acl http apply rule virtual rule1 httpvs1
Demo(config)#acl http apply rule virtual rule2 httpvs2
Demo(config)#acl http apply rule virtual rule3 httpvs2
```

➢ **配置结果**

- 当子网 10.8.6.0/24 内的客户端通过虚拟服务 httpvs1 访问匹配“abc*”的 URL 地址时，任一客户端如果每秒请求数达到 1000，系统将在收到该客户端的后续请求时重置连接。
- 当子网 192.168.0.0/16 内的客户端通过虚拟服务 httpvs2 访问匹配“xyz*”的 URL 地址时，每个客户端的下载速度都将被控制在 1000 KB/s 以内。
- 当子网 192.168.1.0/24 内的客户端通过虚拟服务 httpvs2 访问匹配“xyz*”的 URL 地址时，虽然该子网同时命中规则 rule2 和 rule3，但是该子网内的客户端只受规则 rule3 的限制，即每个客户端的下载速度都将被控制在 100 KB/s 以内。

28.3.5.3 配置 DNS ACL 规则

➢ **配置步骤**

1. 执行如下命令添加 DNS ACL 规则：

acl dns rule <*rule_name*> <*client_ip*> *{netmask|prefix}* <*acl_mode*> <*dns_type*> <*max_limit*>

例如：

Demo(config)#**acl dns rule rule1 10.8.6.0 24 total ALL 12500** Demo(config)#**acl dns rule rule2 192.168.0.0 16 per-ip A 200**

2. 执行如下命令应用 DNS ACL 规则到虚拟服务：

acl dns apply rule virtual <*rule_name*> <*virtual_service*>

例如：

Demo(config)#**acl dns apply rule virtual rule1 dnsvs1** Demo(config)#**acl dns apply rule virtual rule2 dnsvs2**

3. 执行如下命令应用 DNS ACL 规则到 SDNS。

acl dns apply rule sdns <*rule_name*>

例如：

Demo(config)#**acl dns apply rule sdns ru1e1** Demo(config)#**acl dns apply rule sdns ru1e2**

➢ **配置结果**

在 DNS 负载均衡应用场景：

- 当子网 10.8.6.0/24 内的客户端通过虚拟服务 dnsvs1 查询 DNS 资源记录时，该子网内的所有客户端每秒发起的 DNS 查询请求总数如果达到 12,500 个，系统将丢弃后续收到的来自该子网内 DNS 查询请求。

- 当子网 192.168.0.0/16 内的客户端通过虚拟服务 dnsvs2 查询 DNS 资源记录时，该子网内任一客户端每秒发起的 A 类型 DNS 查询请求总数如果达到 200 个，系统将丢弃后续收到的来自该客户端的 A 类型的 DNS 查询请求。

在 SDNS 应用场景：

- 子网 10.8.6.0/24 内所有客户端每秒发起的 DNS 查询请求总数如果达到 12,500 个，系统会丢弃后续收到的来自该子网的 DNS 查询请求。
- 子网 192.168.0.0/16 内的任一客户端每秒发起的 A 类型 DNS 查询请求总数如果达到 200 个，系统会丢弃后续收到的来自该客户端的 A 类型的 DNS 查询请求。

28.4 DDoS 攻击防御

DDoS（Distributed Denial of Service，分布式拒绝服务）攻击已成为网络中愈发频繁的一种攻击方式，其特点是多个攻击源同时对一个对象进行攻击，最终导致服务不可用，服务失败的根本原因包括以下几种：

- 耗尽系统资源（带宽、内存、计算资源），导致设备瘫痪，例如 UDP Flood 攻击、SYN Flood 攻击。
- 破坏协议或状态信息，导致设备故障，例如 IP 分片攻击、Ping of Death 攻击。
- 利用应用弱点，导致系统异常或泄露核心信息，例如 DNS 缓存毒化。

根据 DDoS 攻击发生的网络层次，DDoS 攻击可以分类为网络层、会话层和应用层攻击。DDoS 攻击的发展趋势呈现出两个方向，一方面攻击流量急剧增加，另一方面应用层攻击爆发且日益复杂，难以探测和缓解。

设备的 DDoS 攻击防御功能针对这三个协议层的 DDoS 攻击提供攻击探测和防御机制，为服务器负载均衡业务提供更高一层的安全防护。在初始配置下，DDoS 攻击防御功能为禁用状态，管理员可以执行“**ddos {on|off}**”命令来全局启用或禁用 DDoS 攻击防御功能。

28.4.1 网络层

本节介绍常见的网络层 DDoS 攻击和设备的防御机制。

28.4.1.1 IP DDoS

➢ **IP 选项攻击**

特征：IP 选项一般用于调测或特殊的应用类型。攻击者常常通过源路由选项来利用可信客户端 IP 对服务器发起攻击。一个 IP 头部里可以包含多个选项。这些选项的类型、长度和顺序可以灵活组合，但是一些较新的系统和协议在处理这些选项时可能出现严重错误。

防御方式：系统默认不处理 IP 选项，当收到携带 IP 选项的报文时，会剥掉 IP 选项后再处理报文。

注意：对于 IP 选项攻击，系统只提供防御，不记录攻击日志。

28.4.1.2 ICMP DDoS

- **ICMP Flood**

特征：ICMP Flood 是利用大量的 ICMP 流量使目标主机过载，无法为正常请求服务，最终导致网络瘫痪。

防御方式：通过底层操作系统过滤这种攻击，使攻击无效化。

注意：对于 ICMP Flood 攻击，系统只提供防御，不记录攻击日志。

- **Ping of Death 攻击**

特征：Ping of Death 攻击的表现是发送大于 65,535 字节的 IP 数据包，某些系统在收到这些报文时，会因为无法处理而出现故障甚至重启。

防御方式：系统会丢弃所有长度超限的 IP 数据包。

注意：对于 Ping of Death 攻击，系统只提供防御，不记录攻击日志。

- **Smurf 攻击**

特征：攻击者向广播地址发送 ICMP 请求，源地址伪装成目标主机的 IP 地址，广播网络里的所有主机都会向目标主机回复 ICMP 应答，导致网络瘫痪。

防御方式：系统会丢弃目的 IP 地址是广播地址的 ICMP 请求。

注意：对于 Smurf 攻击，系统只提供防御，不记录攻击日志。

28.4.1.3 TCP DDoS

初始配置下，TCP DDoS 攻击防御会在全局启用 DDoS 攻击防御后自动生效。管理员可以通过“**ddos tcp {on|off}**”控制 TCP DDoS 攻击防御的开启状态。

开启 TCP DDoS 防御后，系统将对以下类型的 TCP DDoS 攻击实施探测和防御。

- **SYN Flood**

特征：攻击者会使用伪造的 IP 地址向目标服务器的每个端口重复发送 SYN 报文，服务器会在每个端口回复 SYN-ACK 报文，最终因为 CPU 和内存耗尽而发生故障。

防御方式：系统会检查 SYN 报文的合法性，丢弃攻击报文，并记录攻击日志。

- **PUSH/ACK Flood**

特征：ACK 和 PUSH-ACK 报文用于通信双方相互确认已接受到对方的报文。在 ACK Flood 攻击中，被攻击方会频繁收到伪造的 ACK 报文，因为要处理这些报文，被攻击方的系统资源会快速耗尽，导致业务质量下降甚至中断。

防御方式：系统对 PUSH/ACK Flood 攻击不提供防御，但会对探测到的可疑攻击源记录攻击日志。

- **FIN/RST Flood**

特征：在 RST/FIN Flood 攻击中，攻击者会以极高速率向目标服务器发送伪造的 RST 或 FIN 报文，这些报文不属于任何实际的会话，但是服务器会耗费大量资源查找匹配的会话。如果攻击者发送的 RST 或 FIP 报文命中了一个已有会话，会导致这个会话的意外中断。

防御方式：系统对 FIN/RST Flood 攻击不提供防御，但会对探测到的可疑攻击源记录攻击日志。

- **连接耗尽攻击**

特征：这种攻击的主要特点是用真实的 IP 地址向目标服务器建立大量连接，并长时间的保持连接，服务器上存在大量闲置的连接，占用服务器的连接资源，甚至无法为正常的连接请求提供服务。

防御方式：系统会缩减连接超时时间，并对探测到的可疑攻击源记录攻击日志。

28.4.1.4 UDP DDoS

UDP DDoS 攻击防御针对 UDP Flood 攻击提供防御。初始配置下，UDP DDoS 攻击防御会在全局启用 DDoS 攻击防御后自动生效。管理员可以通过“**ddos udp {on|off}**”控制 UDP DDoS 攻击防御的开启状态。

- **UDP Flood**

特征：UDP Flood 攻击是一种资源耗尽攻击。由于 UDP 是一种无状态网络协议，攻击者会随机地向目标主机的端口发送大量 UDP 报文。目标主机不停地查找目的端口对应的应用程序，因为无法找到而需要回复大量的 ICMP 目标不可达报文，导致双向网络带宽迅速消耗，严重时出现链路拥塞。

防御方式：系统对 UDP Flood 攻击不提供防御，但会对探测到的可疑攻击源记录攻击日志。

28.4.2 会话层

本节介绍常见且影响严重的会话层 DDoS 攻击类型和防御机制。

- **SSL 无效报文攻击**

特征：攻击者通过发送大量格式错误或者无法解析的 SSL 报文，导致系统进程崩溃，服务器无法提供正常服务。

防御方式：系统会通过 SSL 状态机过滤和丢弃无效报文。

注意：对于 SSL 无效报文攻击，系统只提供防御，不记录攻击日志。

- **SSL 握手超时攻击**

特征：SSL 握手超时攻击表现为攻击者会请求建立大量 TCP 连接，但在连接建立后，并不发送 SSL 握手消息或应用数据，最终耗尽服务器的 SSL 连接资源。

防御方式：系统内置了默认的 SSL 握手超时机制。在初始启动时，系统会按照默认的超时时间（10,000 ms）来检测 SSL 握手的完成时间。如果连接建立后的 10,000 ms 内未收到 SSL 握手消息或应用数据，系统会重置连接，并记录攻击日志。

随着实际的流量变化，系统会生成一个动态的超时时间，然后按照动态生成的超时时间进行探测。管理员也可以通过“**ddos ssl handshake timeout**”命令自定义 SSL 握手的超时时间，当存在自定义的超时时间时，动态生成的超时时间不生效。

- **SSL 重协商攻击**

特征：SSL 重协商攻击的特点为 SSL 握手连接建立后，攻击者通过频繁发送 SSL 重协商请求，触发新的重协商握手，导致一个 TCP 连接上存在多个 SSL 握手，消耗服务器的计算资源。

防御方式：系统内置了默认的 SSL 重协商间隔阈值。系统会按照默认的阈值（10,000 ms）来探测两个 SSL 重协商请求之间的间隔时间。如果两个重协商请求之间的时间间隔小于 10,000 ms，则重置连接，并记录攻击日志。管理员也可以通过“**ddos ssl renegotiation interval**”命令自定义 SSL 重协商的时间间隔阈值。

28.4.3 应用层

应用层最常发生和影响严重的 DDoS 攻包括 HTTP 和 DNS DDoS 攻击。

28.4.3.1 HTTP DDoS

- **HTTP 无效报文攻击**

特征：攻击者通过发送大量格式错误或无法解析的 HTTP 报文，导致系统进程崩溃，服务器无法提供正常服务。

防御方式：系统会通过 HTTP 状态机过滤和丢弃无效报文。

注意：对于 HTTP 无效报文攻击，系统只提供防御，不记录攻击日志。

- **HTTP 慢速攻击**

特征：

a. Slowloris 攻击：攻击者通过缓慢发送一个 HTTP 头部的数据报长时间保持连接，耗尽服务器的 HTTP 连接资源，使正常的 HTTP 业务连接无法建立。

b. Slowpost 攻击：类似 Slowloris 攻击，攻击者通过缓慢发送一个 POST 请求的数据报长时间保持连接。

防御方式：系统内置了默认的 HTTP 服务器响应时间阈值。在初始启动时，系统会按照默认阈值来探测 HTTP 服务器的响应时间。如果服务器的平均响应时间超过 100 ms，并且持续时间超过 30s，系统会记录一次超时，累计三次超时，则重置连接，并记录攻击日志。

随着实际的流量变化，系统会生成一个动态的阈值，然后按照动态生成的阈值进行探测。管理员也可以通过“**ddos http slow interval**”命令自定义 HTTP 响应时间阈值，当存在自定义的阈值时，动态生成的阈值不生效。

- **长表单提交和 CC（Challenge Collapsar）**

特征：长表单处理会损耗系统的大量资源。攻击者通过 GET 和 POST 等 HTTP 方法提交长表单达到攻击的目的。与长表单提交攻击相似，CC 攻击的发起者会对目标服务器发送大量 HTTP 查询请求，造成后端数据库资源的大量损耗，最终导致无法服务正常的网页请求。

防御方式：系统内置了默认的 HTTP 请求携带的 cookie 个数阈值和 URL 中包含的 query 个数阈值。初始启动时，系统会按照默认阈值探测 HTTP 请求携带的 cookie 个数和 URL 中包含的 query 个数。如果一个 HTTP 请求里携带的 cookie 或 query 个数超过 10 个，系统会对可疑客户端进行验证（“**ddos http verify on**”命令），如果非法则重置连接，并记录攻击日志。

随着实际的流量变化，系统会生成动态的阈值，然后按照动态生成的阈值进行探测。管理员也可以通过“**ddos http kvincookie**”和“**ddos http querycnt**”命令自

定义 HTTP 请求携带的 cookie 个数阈值和 query 个数阈值，当存在自定义的阈值时，动态生成的阈值不生效。

- **HTTP Flood 和 Hashdos 攻击**

特征：HTTP Flood 是一种 CPU 资源耗尽攻击，主要表现为 GET Flood 和 POST Flood，其目标是通过频繁请求大量资源，造成服务器过载，无法为正常请求提供服务。

Hashdos 也是一种损耗 CPU 计算资源的攻击方式。Hash 表的实现方式大致相同。如果一个 Hash 值对应多个 key，那么所有表项都会被存储在同一个索引位置，导致 Hash 表的性能下降。

防御方式：系统内置了 HTTP 服务器对 GET、POST、HEAD、PUT 和 DELETE 请求的响应时间阈值。初始启动时，系统会按照默认阈值探测 HTTP 服务器的响应时间。如果服务器的平均响应时间超过 100 ms，并且持续时间超过 30s，则按照以下原则处理：

系统会统计并验证 Top 10 流量客户端和服务器（“**ddos http verify on**”），将非法客户端加入 ACL 黑名单（“**ddos http acl blacklist on**”）。关于 ACL 黑名单的详细说明，请参考“21.4.6 ACL 黑名单”。如果未配置“**ddos http verify on**”，系统会依据动态 HTTP ACL 规则对控制 TopN 流量客户端的访问行为。关于 HTTP ACL 规则的详细说明，请参考“21.3.3 HTTP ACL 规则”。

随着实际流量的变化，系统会生成动态阈值，并按照动态生成的阈值进行探测。管理员也可以通过“**ddos http resptime**”命令自定义 HTTP 服务器的响应时间阈值。当系统内存在自定义的阈值时，动态生成的阈值不生效。

28.4.3.2 DNS DDoS

对于 DNS DDoS，系统提供独立控制的 DNS DDoS 攻击防御机制，管理员可以根据网络需要选择是否开启 DNS DDoS 攻击防御。DNS DDoS 攻击防御针对 DNS Query Flood 和 DNS NXDomain Flood 攻击提供防御。初始配置下，DNS DDoS 攻击防御会在全局启用 DDoS 攻击防御后自动生效。管理员可以通过“**ddos dns {on|off}**”控制 DNS DDoS 攻击防御的开启状态。

- **DNS Query Flood**

特征：DNS Query Flood 本质上是一种 UDP Flood 攻击。由于 DNS 服务器在网络运行中发挥的关键作用，DNS Query Flood 的攻击造成的影响更严重。这种攻击的特点是攻击者会向服务器发送大量的 DNS 请求，服务器最终会因为无法处理过多的请求而拒绝服务。

防御机制：系统内置了 DNS 查询响应时间的探测机制和动态 DNS ACL 规则进行防御。在运行过程中，系统会统计和计算虚拟服务对每个 DNS 查询的平均响应时间，如果发现当前响应时间比平均值高出很多，而且当前 RPS 超出阈值很高时，系统会动态生成 DNS ACL 规则，调低 RPS 阈值，直到虚拟服务对 DNS 查询的响应时间恢复到正常值，并且当前 RPS 恢复到阈值以下。

在 SDNS 场景下，系统具备足够的资源容量来抵御 A、AAAA 和 CNAME 类型的 DNS Query Flood，对于其他类型的攻击，系统会通过 DNS ACL 规则为 BIND9 服务器提供 DDoS 攻击防御。

关于 DNS ACL 规则的详细说明，请参考“21.3.4 DNS ACL 规则”。

- **DNS NXDomain Flood**

特征：DNS NXDomain Flood 与 DNS Query Flood 攻击方式类似，区别在于 DNS NXDomain 攻击者请求查询的域名是随机生成或不存在的。当 DNS 服务器发现请求无法解析时，会进行递归查询。整个解析过程会给 DNS 服务器造成过量的负担，导致 DNS 服务器崩溃。

防御方式：系统对 DNS NXDomain Flood 攻击不提供防御，但会对探测到的可疑攻击源记录攻击日志。

28.4.4 DDoS 黑名单

DDoS 黑名单记录系统探测到的最近 2000 条攻击记录，可以通过命令“show ddos blacklist”查看。当系统重启后，这些攻击记录将消失。

例如：

```
Demo(config)#show ddos blacklist
1: ssl handshake attack. 2: ssl renegotiation attack.
1001: http slowloris attack.
1002: http slowpost attack.
1003: http query count abnormal.
1004: http cookie count abnormal.
1005: http resptime abnormal.
2001: rps exceeding limit.
3001: tcp attack.
3002: udp attack.
3003: slb dns attack.
3004: gslb dns attack.
start of blacklist
 2015 Jan 16 17:15:44 172.16.77.190,44647,v155,1
 2015 Jan 16 17:16:04 172.16.77.190,44648,v155,4097
2015 Jan 16 17:17:44 172.16.77.191,44649,v157,4098
 2015 Jan 16 17:18:04 172.16.77.192,44645,v156,2
```

命令输出的含义（以“2015 Jan 16 17:15:44 172.16.77.190,44647,v155,1”为例）：

- 2015 Jan 16 17:15:44：表示攻击发生的时间
- 172.16.77.190：表示发起攻击的客户端 IP

- 44647：表示发起攻击的客户端端口号
- v155：表示虚拟服务的名称
- 1：表示攻击类型

为了便于查看和分析数据，DDoS 黑名单支持通过 WebUI 导出到本地（CSV 格式）。

28.4.5 DDoS 攻击日志

所有 DDoS 攻击都会被记录到 DDoS 攻击日志中，管理员可以使用“**show ddos record**”命令查看 DDoS 攻击日志。如果需要将 DDoS 攻击日志备份到本地，可以使用“ddos record export”命令。

28.4.6 ACL 黑名单

ACL 黑名单里记录所有被完全禁止访问的 IP 地址，如果一个 IP 地址被加入 ACL 黑名单，来自该地址的所有报文都会被丢弃。

系统支持自动将已确定为攻击者的客户端 IP 地址添加到 ACL 黑名单。管理员也可以手动配置 ACL 黑名单。

动态添加 IP 地址到黑名单的功能通过“**ddos http acl blacklist {on|off}**”命令控制。启用了该功能后，系统在探测到 HTTP DDoS 攻击并通过 HTTP 验证机制确认客户端为攻击者后，会将客户端的 IP 地址自动添加到 ACL 黑名单，60 分钟内，来自该客户端的访问流量都会被丢弃。该功能的默认设置为禁用，在禁用状态下，探测到的攻击者不会被自动加入 ACL 黑名单。

另外，管理员可以手动添加 IP 地址或子网到 ACL 黑名单或从外部 URL 地址导入一个 IP 列表文件，将 IP 列表文件应用到 ACL 黑名单。

示例：

1. 添加“192.168.0.0/24”子网到 ACL 黑名单，设置超时时间为 10 分钟。

```
Demo(config)#acl blacklist rule 192.168.0.0 24 10
```

2. 从“ftp://10.8.3.28/iplist”导入自定义的 IP 列表，并将该 IP 列表添加到 ACL 黑名单，设置 IP 列表内所有 IP 地址的超时时间为 10 分钟。

```
Demo(config)#acl blacklist import “ftp://10.8.3.28/iplist”
Demo(config)#acl blacklist ipfile apply iplist 10
```

第29章 IPv6 高级配置

29.1 概述

随着全球 IPv4 地址的耗尽，如何实现由 IPv4 网络向 IPv6 网络平滑过渡，已成为许多企业面临的难题。

为了帮助企业网络实现向 IPv6 的过渡，设备提供了对 IPv6 的广泛支持，可以在保障企业业务连续性的前提下，完成 IPv4 到 IPv6 的无缝平滑过渡。设备不仅能够将 IPv4 资源交付给 IPv6 用户，还能够将 IPv6 资源交付给 IPv4 用户，从而实现 IPv4 网络与 IPv6 网络的互联互通。而且，在单纯的 IPv6 网络环境中，设备的应用交付能力同 IPv4 网络中一样安全高效。

本章将分别介绍设备所支持的 IPv6 SLB、DNS64/NAT64、DNS46/NAT46、IPv6 NAT 和 NDP 功能。

29.2 IPv6 SLB

29.2.1 介绍

设备对 SLB 功能提供了广泛的 IPv6 支持。从服务层次上看，二层到七层 SLB 都支持 IPv6；从服务类型上看，除 RDP、SIPUDP 和 SIPTCP 之外，其它所有类型的服务均可以支持 IPv6；从策略和方法上看，所有的 SLB 策略和除 SNMP 之外其它所有方法均可以支持 IPv6。此外，对于三种不同的 SLB 部署模式，其 IPv6 支持情况如下：

- IPv6 到 IPv4（支持反向代理模式）
- IPv4 到 IPv6（支持反向代理模式）
- IPv6 到 IPv6（支持反向代理模式、透明模式和三角模式）

其中，反向代理模式既能够工作在 IPv4 和 IPv6 混合的网络环境中，也能够工作在单纯的 IPv4 或 IPv6 网络环境中。透明模式和三角模式仅能够工作在单纯的 IPv4 或 IPv6 网络环境中。

“IPv6 到 IPv4”的 SLB 部署方式的拓扑结构如下图所示：

图29–1 “IPv6 到 IPv4”SLB 拓扑

类似地，也可以采用“IPv4 到 IPv6”的 SLB 部署方式，不过虚拟服务需要配置 IPv4 地址，而后台服务则需要配置 IPv6 地址。

对于“IPv6 到 IPv6”的 SLB 部署方式，虚拟服务和后台服务都需要配置相应的 IPv6 地址。

29.2.2 配置示例

1. 执行如下命令添加基于 IPv6 的虚拟服务：

slb virtual http *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]*

例如：

Demo(config)#**slb virtual http virtualserv 3fff::251 8080 1000 tcp 3 3**

2. 执行如下命令为虚拟服务配置策略：

slb policy default *{virtual_name|vlink_name} {group_name|vlink_name}*

例如：

Demo(config)#**slb default virtualserv g1**

3. 执行如下命令添加 IPv4 后台服务和服务组：

slb real http *<real_name> <ip> [port] [max_conn]* *[http|tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns|none] [hc_up] [hc_down]* *slb group method <group_name> [algorithm]*

例如：

Demo(config)#**slb real http Serv1 172.16.65.251** Demo(config)#**slb real http Serv2 172.16.66.251** Demo(config)#**slb group method g1 rr**

29.3 DNS64 和 NAT64

29.3.1 概述

DNS64 功能支持对 A、AAAA、MX 和 PTR 类型的请求和响应进行转换，帮助 IPv6 客户端访问 IPv4 类型的网页服务器或邮件服务器。例如，在收到 IPv6 客户端发送的 DNS AAAA 请求时，DNS64 功能会将其转换为 DNS A 请求，然后把 DNS A 响应转换为 DNS AAAA 响应，再返回给 IPv6 客户端。当 IPv6 客户端使用该 IPv6 地址访问 IPv6 服务器时，NAT64（Network Address Translation IPv6 to IPv4）功能将客户端发送的 IPv6 报文转换为 IPv4 报文。在收到服务器的 IPv4 报文后，NAT64 功能将 IPv4 报文转换为 IPv6 报文。这样确保了 IPv6 客户端能与 IPv4 服务器正常通信。

DNS64 功能支持处理客户端通过同一个连接发送多个类型相同和不同 DNS 请求的情况，例如同时请求多个域名的 DNS A 和 DNS AAAA 记录。另外，DNS64 功能能够防御 DNS 请求重放攻击和响应欺骗。

DNS64 和 NAT64 功能可以分开部署在两台设备上，也可以集中部署在一台设备上。

29.3.2 优先模式

对于同时拥有 A 和 AAAA 资源记录的请求，可以配置优先模式来确定优先使用哪种资源。默认情况下，DNS64 会采用 AAAA 优先模式。

- A 优先：优先将 DNS A 请求发往 DNS A 服务器，优先选择 DNS A 响应。
- AAAA 优先：优先将 DNS AAAA 请求发往 DNS AAAA 服务器，优先选择 DNS AAAA 响应。
- 响应优先：将 DNS A 请求发往 DNS A 服务器，DNS AAAA 请求发往 DNS AAAA 服务器，优先选择先收到的响应中的资源记录。

无论采用哪种优先模式，DNS64 功能会保证返回的记录类型和请求的类型一致。

29.3.3 定时器

DNS64 功能支持优先定时器和响应定时器。

29.3.3.1 优先定时器

优先定时器只对 A 和 AAAA 优先模式生效。在这两种模式下，DNS64 会先发送一个请求，如果发送成功，就启动优先定时器。如果先发送的请求在计时器到时前没收到响应，DNS64 会发送缓存的请求。优先定时器通过“**ipv6 tune dnsnat**

priority time”命令定义，默认为 100 毫秒。自定义优先定时器，过小和过大都不合适。如果过小，优先发送的请求和缓存的请求的发送时间过于接近，尤其在网络延迟较大的环境；如果过大，客户端可能因为等待时间长而重新请求。建议的范围为 100 至 500 毫秒。

29.3.3.2 响应定时器

在 A 和 AAAA 优先模式下，响应定时器会在缓存的请求被成功发送时启动。在响应优先模式和不使用优先模式的其他情况下，例如 PTR 请求，只要请求发送成功，就会启动。如果在响应定时器到时前没有收到响应，相应的 transaction ID 记录会被删除，而不会等到连接超时后才会删除。响应定时器通过“**ipv6 tune dnsnat response timeout**”命令配置，默认为 5000 毫秒。

29.3.4 工作原理

DNS64 功能支持 A、AAAA、MX 和 PTR 类型请求和响应的转换。

- **DNS AAAA 请求和响应**

收到 DNS AAAA 请求后，DNS64 功能会先根据 DNS AAAA 请求生成一个 DNS A 请求，然后根据优先模式处理。

- A 优先：缓存 DNS AAAA 请求，将 DNS A 请求发送给 DNS A 授权服务器。
  - 如果 DNS A 请求发送成功，则启动优先计时器（默认 100 毫秒）。在计时器到时前收到 DNS A 响应，则转换成 DNS AAAA 响应再发给客户端。
  - 如果 DNS A 请求发送失败，或者在优先计时器到时前没收到响应，则将缓存的 DNS AAAA 请求发送给 DNS AAAA 授权服务器，启动响应计时器。在响应计时器到时前，如果收到 DNS AAAA 响应，就直接发给客户端；如果收不到，则删除 Transaction ID 记录。
- AAAA 优先：缓存 DNS A 请求，将 DNS AAAA 请求发送给 DNS AAAA 授权服务器。
  - 如果 DNS AAAA 请求发送成功，则启动优先计时器（默认 100 毫秒）；在计时器到时前收到 DNS AAAA 响应，则直接发给客户端。
  - 如果 DNS AAAA 请求发送失败，或者在优先计时器到时前没收到响应，则将缓存的 DNS A 请求发给 DNS A 授权服务器，启动响应定时器。在响应计时器超时前，如果收到 DNS A 响应，就转换成 DNS AAAA 响应后发给客户端；如果收不到，则删除 Transaction ID 记录。
- 响应优先：同时发送 DNS AAAA 和 DNS A 请求，分别发给 DNS AAAA 和 DNS A 域名服务器，启动响应计时器。正常情况下会先收到 DNS AAAA 响

应，无需转换即可发给客户端；如果先收到 DNS A 响应，则先转换成 DNS AAAA 响应再发给客户端。

- **DNS A 请求和响应**

在 DNS64 应用场景，DNS A 请求会被直接转发给 DNS A 授权服务器，启动响应定时器。如果在响应计时器到时前收到 DNS A 响应，则直接发给客户端。如果发送失败或者响应计时器到时前没收到响应，则删除 Transaction ID 记录。

- **MX 请求和响应**

DNS64 功能会先复制一份 MX 请求，例如 MX_DUP，然后根据优先模式发送请求和转换响应，处理原则和 DNS AAAA 请求基本相同。不同之处是收到的 MX 响应可能会同时包含 DNS A 和 DNS AAAA 记录。这种情况，在 A 优先模式下会优先选择 DNS A 记录，AAAA 优先模式下会优先选择 DNS AAAA 记录，响应优先模式下优先使用最先解析到那个，并进行相应的转换。

- **PTR 请求和响应**

对于 PTR 请求，DNS64 功能会先解析出其请求的 IP，然后根据 IP 类型进行处理，不受优先模式影响。

- 如果请求的是 IPv4 地址，会被直接发给 DNS A 授权服务器，并启动响应计时器。在计时器到时前收到的响应会被直接发给客户端。
- 如果是请求的是 IPv6 地址，但是该地址是从 IPv4 转换过来的，DNS64 功能会先进行逆向转换，再将 IPv4 的 PTR 请求发给 DNS A 授权服务器，并启动响应计时器。在响应计时器到时前，如果收到响应，会被转换回 IPv6 地址后再发给客户端。
- 如果是未经转换的 IPv6 类型 PTR 请求，会被直接发给 DNS AAAA 授权服务器，并启动响应计时器。在计时器到时前，收到的响应会被直接发给客户端。

如果发送失败，或者在响应计时器到时前没收到响应，则删除 Transaction ID 记录。

DNS64 和 NAT64 功能适用于“IPv6 to IPv4”应用场景，如下图所示。

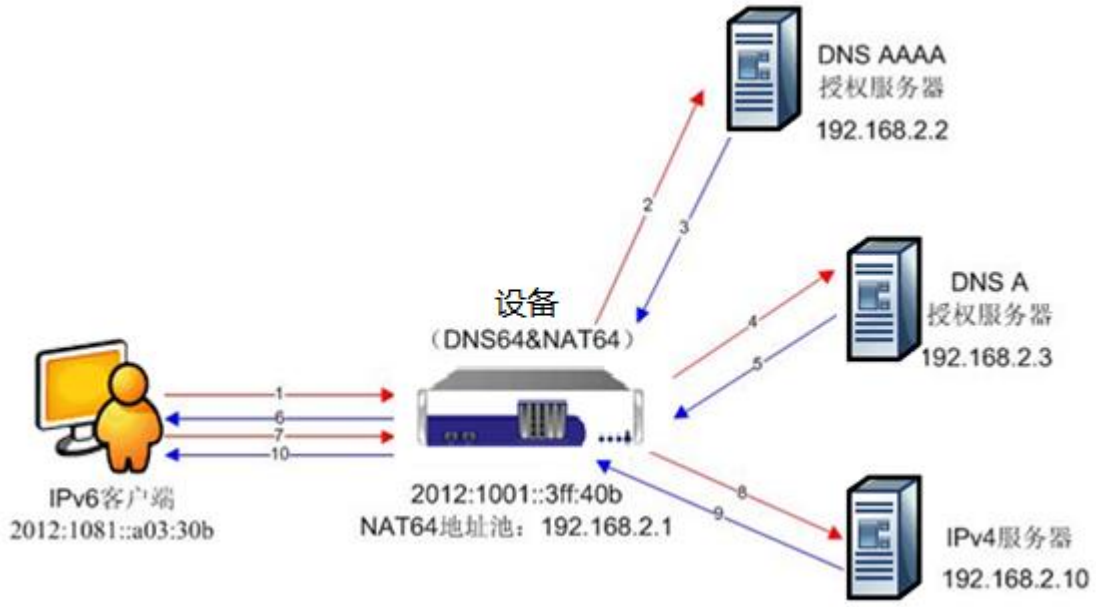

图29–2 “IPv6 to IPv4”应用场景

以 AAAA 优先模式下处理 DNS AAAA 请求为例，DNS64 和 NAT64 功能的工作流程如下所示：

1. IPv6 客户端（2012:1081::a03:30b）发送 DNS AAAA 请求到设备（2012:1001::3ff:40b），以解析域名 www.example.com。
2. 设备收到 DNS AAAA 请求后，会生成一个 DNS A 请求并缓存，然后发送 DNS AAAA 请求到该域名的 DNS AAAA 授权服务器进行解析。
3. 如果 DNS AAAA 授权服务器没有该域名的 AAAA 记录，将返回空 DNS AAAA 响应消息。此时，设备将忽略此响应消息。
4. 在发送完 DNS AAAA 请求后，设备将启动优先定时器（默认 100 毫秒）。如果在此期间没有收到有效的 DNS AAAA 响应消息，设备将发送缓存的 DNS A 请求到该域名的 DNS A 授权服务器进行解析。
5. 设备收到 DNS A 授权服务器返回的 DNS A 响应（例如，A：www.example.com －192.168.2.10）。
6. 设备通过添加配置的前缀将 DNS A 响应消息转换为 AAAA 响应消息（例如，AAAA：www.example.com－64:ff9b::192.168.2.10），然后返回转换后的 DNS AAAA 响应消息给 IPv6 客户端。
7. IPv6 客户端使用转换后的 IPv6 地址访问 www.example.com。
8. 设备将客户端发送的 IPv6 报文（src：2012:1081::a03:30b；dst：64:ff9b::192.168.2.10）转换为 IPv4 报文（src：192.168.2.1；dst：192.168.2.10），并发送给 IPv4 服务器。

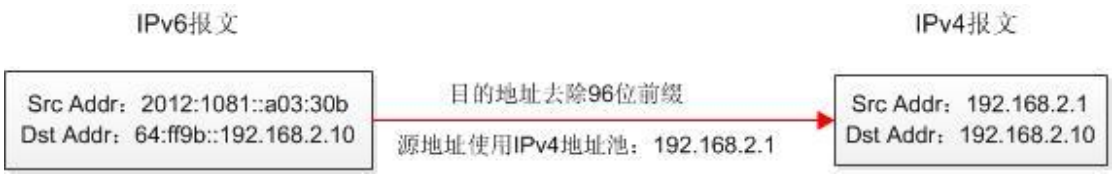

图29–3 NAT64 地址转换

9. IPv4 服务器返回 IPv4 报文（src：192.168.2.10；dst：192.168.2.1）给设备。
10. 设备将 IPv4 报文转换为 IPv6 报文（src：64:ff9b::192.168.2.10；dst：2012:1081::a03:30b），并返回给 IPv6 客户端。

29.3.5 应用说明

在整个系统中，DNS64 功能只能启用在一个 DNS 类型的虚拟服务上。该虚拟服务将作为 DNS 代理，负责把 DNS AAAA 请求转换为 DNS A 请求，然后把 DNS A 记录响应转换为 DNS AAAA 响应。

为了使 DNS64 功能能够正常工作，需要为该虚拟服务配置 default 策略和 backup 策略。设备将根据 default 策略转发 DNS AAAA 请求，根据 backup 策略转发转换后的 DNS A 请求。因此，与 default 策略关联的后台服务器应配置为可响应 AAAA 记录的 DNS 服务器，而与 backup 策略关联的后台服务器应配置为可响应 A 记录的 DNS 服务器。

注意：

1. DNS64 功能不支持巨型帧。
2. DNS64 功能只支持转换基于 UDP 的 DNS 请求和响应，不支持转换基于 TCP 的 DNS 请求和响应。

29.3.6 配置 DNS64 和 NAT64 功能

1. 执行如下命令配置地址池：

ip pool <*pool_name*> <*start_ip*> *[end_ip]*

例如：

Demo(config)#**ip pool NAT64_pool 192.168.2.1**

2. 执行如下命令完成 SLB 相关的配置：

slb real dns <*real_name*> <*ip*> <*port*> *[max_conn]*
[dns|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns|none] [hc_up] [hc_down] [timeout]
slb real enable <*real_name*>
slb group method <*group_name*> *[algorithm]*
slb group member <*group_name*> <*real_name*> *[weight|cookie|url] [priority]*

slb virtual dns *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]*
slb policy default *{virtual_name|vlink_name} {group_name|vlink_name}*
slb policy backup *{virtual_name|vlink_name} {group_name|vlink_name}*

例如：

Demo(config)#**slb real dns dns_rs1 192.168.2.2**
Demo(config)#**slb real dns dns_rs2 192.168.2.3**
Demo(config)#**slb group method g1 rr**
Demo(config)#**slb group member g1 dns_rs1**
Demo(config)#**slb group method g2 rr**
Demo(config)#**slb group member g2 dns_rs2**
Demo(config)#**slb virtual dns dns_vs1 2012:1001::3ff:40b**
Demo(config)#**slb policy default dns_vs1 g1**
Demo(config)#**slb policy backup dns_vs1 g2**
Demo(config)#**slb real enable dns_rs1**
Demo(config)#**slb real enable dns_rs2**

注意：启用 DNS64 功能的 DNS 虚拟服务只能通过 default 或 backup 策略与服务组关联。

3. 执行如下命令配置 DNS64 功能：

ipv6 dns64 on *<dns_vs_name> [priority_mode]*
ipv6 dns64 prefix *<dns64_prefix>*

例如：

Demo(config)#**ipv6 dns64 on dns_vs1 AAAA**
Demo(config)#**ipv6 dns64 prefix 64:ff9b::**

4. 执行如下命令配置 NAT64 功能：

ipv6 nat64 ippool *<ipv4_pool_name>*
ipv6 nat64 prefix *<nat64_prefix>*
ipv6 nat64 timeout *<idle_timeout>*
ipv6 nat64 on

例如：

Demo(config)#**ipv6 nat64 ippool NAT64_pool**
Demo(config)#**ipv6 nat64 prefix 64:ff9b::**
Demo(config)#**ipv6 nat64 timeout 300**
Demo(config)#**ipv6 nat64 on**

注意：

- 如果需要 DNS64 和 NAT64 功能在一台设备上配合使用，请保证参数

“dns64_prefix”和“nat64_prefix”的取值相同。

- 在将 IPv4 地址池配置给 NAT64 功能并启用了 NAT64 功能后，请勿删除该 IPv4 地址池。否则，NAT64 功能将被禁用，其相关的配置也将被删除。

29.4 DNS46 和 NAT46

29.4.1 概述

DNS46 功能支持对 A、AAAA、MX 和 PTR 类型的请求和响应进行转换，帮助 IPv4 客户端访问 IPv6 类型的网页服务器或邮件服务器。例如，在收到 IPv4 客户端发出的 DNS A 请求时，DNS46 功能会将其转换为 DNS AAAA 请求，然后将返回的 DNS AAAA 响应转换为 DNS A 记录响应，并建立一条 IPv6 地址到 IPv4 地址的映像记录。设备返回转换后的 IPv4 地址给 IPv4 客户端。当 IPv4 客户端使用该 IPv4 地址访问 IPv6 服务器时，NAT46 功能根据已经建立的映像记录将客户端发送的 IPv4 报文转换为 IPv6 报文。在收到服务器的 IPv6 报文后，NAT46 功能将 IPv6 报文转换为 IPv4 报文。这样确保了 IPv4 客户端能与 IPv6 服务器正常通信。

DNS46 功能支持处理客户端发送多个类型相同和不同 DNS 请求的情况，例如同时请求多个域名的 DNS A 和 DNS AAAA 记录。另外，DNS46 功能能够防御 DNS 请求重放攻击和响应欺骗。

因为 DNS46 和 NAT46 功能共享映像记录，所以必须部署在同一台设备上。

29.4.2 优先模式

DNS46 功能和 DNS64 功能支持同样的优先模式（参考“29.3.2 优先模式”一节）。

29.4.3 定时器

DNS46 功能和 DNS64 功能支持同样的定时器（参考“29.3.3 定时器”一节）。

29.4.4 工作原理

与 DNS64 功能相似，DNS46 功能也支持 A、AAAA、MX 和 PTR 类型请求和响应的转换，工作原理与 DNS64 相同（参考“29.3.4 工作原理”一节）。

DNS46 和 NAT46 功能适用于“IPv4 to IPv6”应用场景，如下图所示。

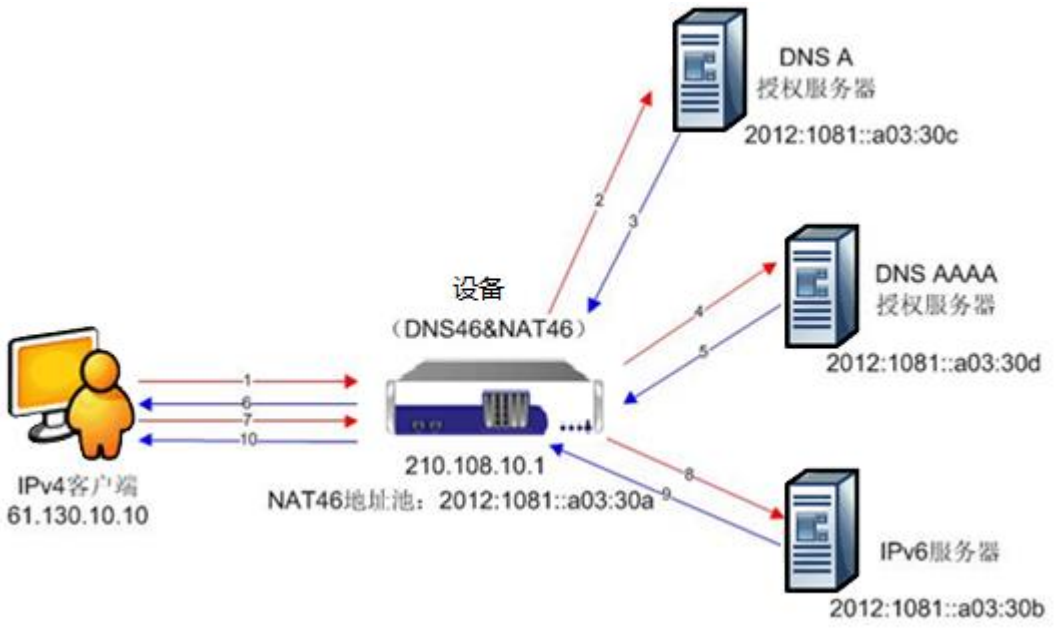

图29–4 “IPv4 to IPv6”应用场景

以 A 优先模式下处理 DNS A 请求为例，DNS46 和 NAT46 功能的工作流程如下所示：

1. IPv4 客户端（61.130.10.10）发送 DNS A 请求到设备（210.108.10.1），以解析域名 www.example.com。
2. 设备收到 DNS A 请求后，会生成一个 DNS AAAA 请求并缓存，然后发送 DNS A 请求到该域名的 DNS A 授权服务器进行解析。
3. 如果 DNS A 授权服务器没有该域名的 A 记录，将返回空 DNS A 响应消息。此时，设备将忽略此响应消息。
4. 在发送完 DNS A 请求后，设备将启动优先计时器（默认 100 毫秒）。如果在此期间没有收到有效的 DNS A 响应，设备将发送缓存的 DNS AAAA 请求到该域名的 DNS AAAA 授权服务器进行解析。
5. 设备收到 DNS AAAA 授权服务器返回的 DNS AAAA 响应消息（例如，AAAA：www.example.com－2012:1081::a03:30b）。
6. 设备根据配置的地址池（使用“**ipv6 dnsnat46 ipmap**”命令配置的用于地址映像的 IPv4 子网）将 DNS AAAA 响应消息转换为 A 响应消息（例如，A：www.example.com－210.108.10.10），然后将转换后的 DNS A 响应消息返回给 IPv4 客户端。同时，系统会自动创建一条 IPv6 地址 2012:1081::a03:30b 到 IPv4 地址 210.108.10.10 的映像记录。
7. IPv4 客户端使用转换后的 IPv4 地址访问 www.example.com。

8. 设备根据已创建的映像记录将客户端发送的 IPv4 报文（src：61.130.10.10；dst：210.108.10.10）转换为 IPv6 报文（src：2012:1081::a03:30a；dst：2012:1081::a03:30b），并发送给 IPv6 服务器。

图29–5 NAT46 地址转换

9. IPv6 服务器返回 IPv6 报文（src：2012:1081::a03:30b；dst：2012:1081::a03:30a）给设备。

10. 设备将 IPv6 报文转换为 IPv4 报文（src：210.108.10.10；dst：61.130.10.10），并返回给 IPv4 客户端。

29.4.5 应用说明

在整个系统中，DNS46 和 NAT46 功能只能启用在一个 DNS 类型的虚拟服务上。该虚拟服务将作为 DNS 代理，负责把 DNS A 请求转换为 DNS AAAA 请求，然后把 DNS AAAA 响应转换为 DNS A 响应。

为了使 DNS46 和 NAT46 功能能够正常工作，需要为该虚拟服务配置 default 策略和 backup 策略。设备将根据 default 策略转发转换后的 DNS AAAA 请求，根据 backup 策略转发 DNS A 请求。因此，与 default 策略关联的后台服务器应配置为可响应 AAAA 记录的 DNS 服务器，而与 backup 策略关联的后台服务器应配置为可响应 A 记录的 DNS 服务器。

注意：

1. DNS46 功能不支持巨型帧。
2. DNS46 功能只支持转换基于 UDP 的 DNS 请求和响应，不支持转换基于 TCP 的 DNS 请求和响应。

29.4.6 配置 DNS46 和 NAT46 功能

1. 请参考“配置 DNS64 和 NAT64 功能”小节完成地址池和 SLB 相关的 CLI 配置。

例如：

```
Demo(config)#ip pool NAT46_pool 2012:1081::a03:30a
Demo(config)#slb real dns dns_rs1 2012:1081::a03:30c
Demo(config)#slb real dns dns_rs2 2012:1081::a03:30d
Demo(config)#slb group method g1 rr
Demo(config)#slb group member g1 dns_rs1
Demo(config)#slb group method g2 rr
Demo(config)#slb group member g2 dns_rs2
Demo(config)#slb virtual dns dns_vs1 210.108.10.1
Demo(config)#slb policy default dns_vs1 g1
Demo(config)#slb policy backup dns_vs1 g2
Demo(config)#slb real enable dns_rs1
Demo(config)#slb real enable dns_rs2
```

2. 执行如下命令为指定 DNS 虚拟服务启用 DNS46 和 NAT46 功能：

ipv6 dnsnat46 on <*dns_vs_name*> *[priority_mode]*

例如：

```
Demo(config)#ipv6 dnsnat46 on dns_vs1 A
```

3. 执行如下命令配置创建映像表所使用的 IPv4 子网：

ipv6 dnsnat46 ipmap <*ipv4_address*> <*netmask*> *[timeout]*

例如：

```
Demo(config)#ipv6 dnsnat46 ipmap 192.168.2.0 255.255.255.0 600
```

4. 执行如下命令指定 NAT46 功能使用的 IPv6 地址池：

ipv6 dnsnat46 ippool <*ipv6_pool_name*>

例如：

```
Demo(config)#ipv6 dnsnat46 ippool NAT46_pool
```

注意：在将 IPv6 地址池配置给 NAT46 功能并启用了 NAT46 功能后，请勿删除该 IPv6 地址池。否则，NAT46 功能将被禁用，其相关的配置也将被删除。

5. 执行如下命令设置 NAT46 TCP 连接的空闲超时时间：

ipv6 dnsnat46 timeout <*idle_timeout*>

例如：

```
Demo(config)#ipv6 dnsnat46 timeout 300
```

29.5 NAT 支持 IPv6

29.5.1 功能介绍

设备不仅支持 NAT64 和 NAT46，还支持“IPv6 到 IPv6”的网络地址转换。它能够将一个处于内网的 IPv6 网络地址转换为公网上的 IPv6 网络地址。而且，还支持配置 IPv6 地址池。

注意：

- “IPv6 到 IPv6”网络地址转换功能仅支持 TCP 和 UDP 协议，不支持 FTP 和 ICMP 协议。
- 配置“IPv6 到 IPv6”网络地址转换功能时，命令“**nat port** *{pool_name|vip} <source_ip> {netmask|prefix} [timeout] [gateway] [description]*”不支持配置网关。

29.5.2 配置示例

执行如下命令配置 IPv6 NAT：

nat port *{pool_name|vip} <source_ip> {netmask|prefix} [timeout] [gateway] [description]*

例如：

Demo(config)#**nat port ipv6_pool 3fff::12 64**

29.6 NDP

29.6.1 NDP 简介

邻居发现协议（Neighbour Discovery Protocol，NDP）是 IPv6 的关键协议，用于获取与本地节点相连的其它节点的链路地址信息。

与 IPv4 中的地址解析协议（ARP）类似，NDP 能够实现网络层地址到链路层地址的转换。不同的是，NDP 使用 IPv6 控制信息报文（ICMPv6）和组播来实现相邻节点（同一链路上的节点）的交互管理，并在一个子网中保持网络层地址和链路层地址之间的映射。

29.6.2 配置示例

执行如下命令添加 NDP 解析项：

ipv6 ndp *<ipv6_address> <mac_address>*

例如：

```
Demo(config)#ipv6 ndp bb::55 00:21:9C:45:80:9F
```

第30章 F5 配置转换

系统提供 F5 配置转换工具用于将部分 F5 配置转换成适合 APV 使用的配置。

30.1 F5 配置转换

目前，系统支持转换的 F5 配置包括：

- TCP、TCPS、HTTP、HTTPS 和 UDP 类型的虚拟服务和后台服务 。不支持的虚拟服务类型系统将不进行转换。
- 轮询（rr）和最少连接数（lc）算法，其中 rr 为默认算法。其他不支持的算法将被转换为默认算法。
- 负载均衡策略（default），该策略为默认策略。其他不支持的策略将被转换为默认策略。
- TCP、HTTP、ICMP 和 UDP 类型的健康检查。不支持的健康检查类型系统将不进行转换。

30.2 配置转换文件

目前，系统支持转换的配置文件有以下条件：

- 文件类型为“.conf”文件或“ucs”解压后的“.im”文件。
- 文件名长度为最大不超过 255 个字符的字符串。文件名支持中文但不支持这些特殊字符：‘…’、‘；’、‘|’、‘\'’、‘\"’、‘\n’、‘<’、‘>’、‘*’、‘%’、‘&’、‘$(’、‘..’、‘“’、‘”’、‘;’、‘‘’、‘’’和‘$（’。
- 文件的编码格式仅支持 UTF-8。
- 由于系统支持的后台服务长度最大为 128 个字符，如果待转换的 F5 配置中后台服务的长度超过 128 个字符，则无法正常转换。

30.3 配置转换方式

管理员可以通过 WebUI 页面（**管理工具>配置转换>转换文件**）在线进行 F5 配置转换，也可以通过 WebUI（**管理工具>配置转换> Windows 转换工具**）下载工具 Windows 转换工具至本地进行本地转换。

第31章 ePolicy

31.1 概述

ePolicy 是公司提供的基于工具命令语言（TCL，Tools Command Language）的定制脚本扩展功能。通过该功能，用户可以在设备现有功能的基础上，通过灵活地定制脚本，开发出新的特性。例如，用户可通过编写特定的脚本，使设备能够支持更多的应用协议，精确控制双向的 IP 应用流量，或限制特定客户端对后台服务器的访问等。

31.2 ePolicy 的基本元素

ePolicy 的基本元素如下：

- 场景
- 事件
- 命令
- 命令调用规则

31.2.1 场景

场景指示 ePolicy 生效的流量场景，即数据流量的类型。

31.2.2 事件

ePolicy 采用基于事件驱动的消息响应机制。设备对每个 Client-设备-Server 连接中的各个动作（如报文到达设备等）均设置有对应的事件，在这些事件发生时，设备会根据预先配置的 ePolicy 命令对流量进行处理。

31.2.3 命令

ePolicy 通过命令指示设备在某一事件发生后对流量的处理方式，如改写数据包内容、选择后台服务器、选择服务组、查询服务组是否有可用的后台服务器等。

31.2.4 命令调用规则

命令调用规则定义了事件与命令之间的对应关系。通过命令调用规则，用户可以灵活地组合事件和命令，根据自身的业务需求直接拦截、检测、转换或重定向任意的双向 IP 应用流量。

关于事件、命令及命令调用规则的具体信息，请联系公司技术支持获取相关文档。

31.3 ePolicy 脚本

ePolicy 使用事件特定处理脚本控制设备的行为。该脚本主要有以下部分构成：

- **脚本描述**

在脚本起始位置，管理员可以通过注释的方式为脚本添加描述信息，介绍脚本的主要作用。管理员可以使用“description:”关键字为脚本添加英文描述，使用“description_cn:”关键字为脚本添加简体中文描述。

示例：

```
#description: Redirect HTTP request
#description_cn:  重定向 HTTP 请求
```

- **场景设置**

场景设置用于指定虚拟服务的网络流量类型。

对于UDP、HTTP或HTTPS类型的虚拟服务，不需要为它们添加场景设置。ePolicy可以自动解析这些虚拟服务的流量类型。

对于 TCP 或 TCPS 类型的虚拟服务，管理员需要在关联的脚本中添加场景设置，告知 ePolicy 虚拟服务处理的何种类型的流量。如果 TCP 和 TCPS 类型的虚拟服务处理不同类型的流量，使用的场景设置也不同的，如下表所示。

表31–1 场景设置示例

网络流量类型	场景设置
HTTP	message::type http
Diameter	message::type binary binary_message::length_start_offset 1 binary_message::length_end_offset 3
Generic TCP	message::type binary

- **命令调用规则：**

管理员需要根据具体需求，结合事件、命令编写命令调用规则。

关于事件处理脚本的范例信息，请联系公司技术支持获取相关文档。

31.4 ePolicy 的应用

通过 ePolicy 脚本，设备可实现如下功能：

- 负载均衡。当应用于服务器负载均衡（SLB，Server Load Balancing）功能时，ePolicy 作为服务组的负载均衡策略与负载均衡算法配合使用，实现服务器间的负载均衡

- 针对 MySQL 服务器的不同操作分别提供负载均衡。
- 对 HTTP、SOAP、XML 和 Diameter 协议的报文内容分析
- 对 Generic TCP 和 TCPS 报文进行接收、发送、解析和丢弃
- 对文本数据进行模式匹配
- TCP 连接控制
- 流量监控和统计
- HTTP 头部和体的解析和修改
- TLS 常用字段的解析
- TCP 选项解析和修改

注意：ePolicy 仅支持 MySQL 5.x 版本的服务器。

31.4.1 与 ePolicy 配合的负载均衡算法

在服务器负载均衡功能中，可与 ePolicy 配合使用的负载均衡算法如下：

- Round Robin (rr)
- Least Connection (lc)
- Shortest Response (sr)
- Persistent IP (pi)
- Hash IP (hi)
- Hash IP and port (hip)
- Consistent hash IP (chi)
- SNMP (snmp)

31.4.2 ePolicy 与现有负载均衡策略的关系

在服务器负载均衡功能中，ePolicy 的优先级高于现有负载均衡策略。当某个虚拟服务与 ePolicy 脚本关联后，系统优先根据 ePolicy 脚本进行负载均衡。如果选不出可用的后台服务，将再次尝试使用为该虚拟服务配置的负载均衡策略选择可用的后台服务。

31.5 ePolicy 配置

执行如下步骤，完成 ePolicy 的配置：

- 根据事件、命令及命令调用规则编写事件处理脚本。
- 导入事件处理脚本。
- 关联虚拟服务和事件处理脚本。

下文将以如何通过 ePolicy 实现“根据 HTTP 请求报文的方法进行服务器负载均衡”为例，讲述具体的脚本内容和配置过程。

31.5.1 编写事件处理脚本

根据事件、命令及命令调用规则编写的事件处理脚本内容如下：

```
#description: Selecting real service based on HTTP method
#description_cn:  根据 HTTP 方法选择后台服务

message::type http

when HTTP_REQUEST_HEADER {
    if { [http::method] == "GET" } {
        slb::select_server realserver_1
    } else {
        slb::select_server realserver_2
    }
}
```

其中，“http_slb.tcl”中的 realserver_1 和 realserver_2 为 SLB 配置中后台服务器的名称。

关于事件、命令及命令调用规则的具体信息，请联系公司技术支持获取相关文档。

31.5.2 导入事件处理脚本

执行如下命令导入事件处理脚本：

epolicy import script *<url> <script_name>*

例如：

```
Demo(config)#epolicy import script http://192.168.10.10/http_slb.tcl http_slb.tcl
```

31.5.3 关联虚拟服务和事件处理脚本

执行如下命令关联虚拟服务和事件处理脚本：

epolicy attach script <*vs_name*> <*script_name*>

例如：

Demo(config)#**epolicy attach script vs_epolicy http_slb.tcl**

31.5.4 配置结果

完成上述配置后：

- 方法为“GET”的 HTTP 请求报文将被发送至后台服务器 realserver_1。
- 方法为其他值的 HTTP 请求报文将被发送至后台服务器 realserver_2。

31.6 兼容 F5 iRules

系统兼容 F5 Networks 公司的部分 iRules，管理员能够直接将这些 iRules 脚本导入设备使用。系统支持的 iRules 类型的命令具体包括 HTTP、SLB、TCP、IP、persist、类（Class）、工具和 SSL。

为了支持该功能，系统添加 ePolicy 事件 RULE_INIT、HTTP_REQUEST、HTTP_REQUEST_SEND 和 HTTP_RESPONSE。

此外，为了支持 Class 类型相关的 iRules，系统添加命令“**epolicy class**”用于配置键值对。更多细节，参见命令行手册。

关于与 iRules 相关的事件和命令的详细介绍，参见 ePolicy 用户手册。

31.6.1 配置示例

31.6.1.1 SLB 基础配置

Demo(config)#**slb real http r1 10.3.0.53 81** Demo(config)#**slb group method g1** Demo(config)#**slb group member g1 r1** Demo(config)#**slb virtual http v1 10.3.0.110** Demo(config)#**slb policy default v1 g1**

31.6.1.2 使用 HTTP 命令的配置示例

- **配置步骤**

1. 导入相关 iRules 脚本。

假设脚本名称为 1.tcl，脚本内容如下：

```
#description: When an HTTP request is received, a customized response is returned to the client
#description_cn: 当收到 HTTP 请求时返回定制的响应给客户端
when HTTP_REQUEST {
    HTTP::respond 200 content {
        <html>
            <head>
                <title>Welcome</title>
            </head>
            <body>
                Welcome to this page.
            </body>
        </html>
    }
}
```

管理员可以通过以下命令导入脚本：

```
Demo(config)#epolicy import script http://192.168.10.10/1.tcl 1.tcl
```

3. 关联虚拟服务（v1）和事件处理脚本（1.tcl）。

```
Demo(config)#epolicy attach script v1 1.tcl
```

➢ **结果验证**

通过浏览器访问虚拟服务 v1，将会返回脚本中配置的内容，如图所示。

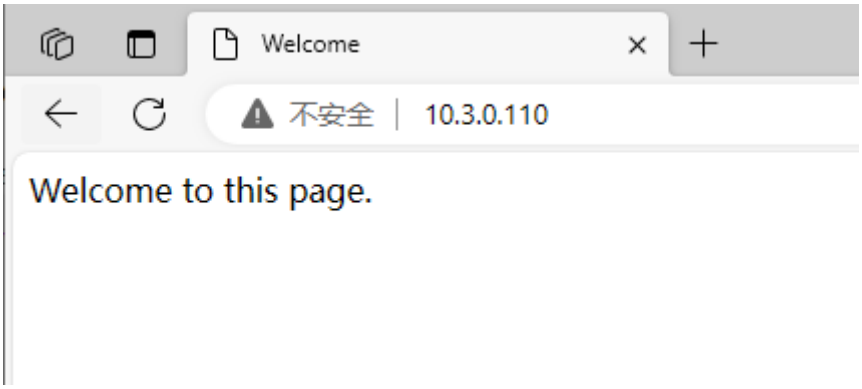

图31–1 结果验证

31.6.1.3 使用 Class 命令的配置示例

➢ **配置步骤**

1. 为 ePolicy 模块创建一个字符串类型的类“class1”，并为该类添加一个键值对“<key1 value1>”。

```
Demo(config)#epolicy class class1 key1 value1 string
```

4. 导入 Class 类型的 iRules 脚本

假设脚本名称为 2.tcl，脚本内容如下：

```
#description: When an HTTP request is received, all key-value pairs in class1 are searched for and output to the log
#description_cn: 当收到 HTTP 请求时查找 class1 中的所有键值对并输出到 log 中
when HTTP_REQUEST {
set id [class startsearch class1]
while {[class anymore class1 $id]} {
    set element [class nextelement class1 $id]
log "class element $element"
}
class donesearch class1 $id
}
```

管理员可以通过以下命令导入脚本。

```
Demo(config)#epolicy import script http://192.168.10.10/2.tcl 2.tcl
```

5. 关联虚拟服务（v1）和事件处理脚本（1.tcl）。

```
Demo(config)#epolicy attach script v1 2.tcl
```

- **结果验证**

通过浏览器访问虚拟服务 v1，通过命令“**show log buff backward**”查看日志，将会看到 ePlicy Class 相关内容。

```
Demo(config)#show log b b
start of buffer
INFO    2023 Nov 23 07:39:10 AN_SQUID_LOG 1700725150.911 3 192.168.97.99
TCP_MISS/200 990 GET / - DIRECT/10.3.0.53 -
INFO    2023 Nov 23 07:39:10 ePolicy: log_command: Script log: class element key1 value1
INFO    2023 Nov 23 07:39:10 AN_SQUID_LOG 1700725150.631 5 192.168.97.99
TCP_MISS/200 990 GET / - DIRECT/10.3.0.53 -
INFO    2023 Nov 23 07:39:10 ePolicy: log_command: Script log: class element key1 value1
```

第32章 日志

32.1 概述

本章将介绍设备的日志（Logging）功能。

设备的日志记录机制遵从 Syslog 的原则。日志子系统负责记录系统错误信息和代理（Proxy）服务中的 HTTP 访问信息。Syslog 是 Unix 平台中的一个通用程序，在 Windows 平台也存在 Syslog 的应用。在 Unix 平台上使用 Syslogd 命令来启动 Syslog 功能，Syslog 监听 UDP514 端口，负责接收并存储本机和非本地的日志信息。设备现在已经支持向三个非本地日志服务器传送日志信息。

32.2 日志原理

32.2.1 Syslog 机制

Syslog 是一种可以将警告和通知信息在网络上传送的协议。

设备 Syslog 日志有八种日志信息的安全级别：紧急（emerg）、报警（alert）、关键（crit）、错误（err）、警告（warning）、提示（notice）、信息（info）和测试（debug），并且支持从 LOCAL0 到 LOCAL7 的八个设备。用户可以使用日志信息查看管理工具来查看内部信息、选择传输协议、设置 syslog 的源、目的端口，设置设备的警告等级等等。

32.2.2 RFC 5424 Syslog

RFC5424 定义了 Syslog 的标准格式。设备支持 RFC 5424 syslog 功能。当启用 RFC 5424 syslog 功能后，系统将会生成 RFC 5424 标准格式的系统日志。RFC 5424 标准日志格式为“<PRI>VER TIMESTAMP HOSTNAME APPNAME PROCID MSGID STRUCTURED-DATA MSG-CONTENT”。（暂不支持 PROCID 和 STRUCTURED-DATA 字段，这两个字段默认显示为“-”。）默认情况下，系统禁用 RFC 5424 syslog 功能。只有执行“**log on**”和“**log rfc5424 on**”命令后，RFC 5424 syslog 功能才能生效。

32.2.3 HTTP 访问日志

HTTP 访问日志是以一种特定格式用来记录每一次 HTTP 请求和响应信息的功能。

设备 HTTP 访问日志支持四种标准格式：组合、WELF（WebTrends Enhanced Log Format）、正常和扫描。用户可以使用“**log http custom**”命令来定义他们的日志格式。

注意：只有当客户端和 Web 服务器之间完成一次正常的 HTTP 通信之后，设备才会记录一条 HTTP 访问日志。

32.2.4 TCP 访问日志

TCP 访问日志用于记录四层 TCP 类型的 SLB 访问日志。当后台服务器在设置的超时时间内返回响应时，系统会根据自定义格式记录日志。当后台服务器未在设置的超时时间内返回响应时，系统将响应数据记录为自定义的内容。

管理员可以使用“**log tcp custom**”命令来自定义 TCP 访问日志的日志格式，可以使用“**log tcp response timeout**”命令来配置 TCP 访问超时时间和 TCP 访问超时后的响应数据。

注意：只有当客户端和 Web 服务器之间完成一次正常的 TCP 通信之后，设备才会记录一条 TCP 访问日志。

32.2.4.1 配置示例

1. 自定义 TCP 访问日志的日志格式。

```
Demo(config)#log tcp custom "%h"
```

2. 配置 TCP 访问超时时间和 TCP 访问超时后的响应数据。

```
Demo(config)#log tcp response timeout 100 "Response Timeout"
```

3. 查看 TCP 访问日志。

```
Demo(config)#show log buff backward
INFO      2024 Jan 02 09:09:21 response timeout
```

32.2.5 日志过滤

日志过滤通过匹配过滤字符串把日志过滤到不同的日志服务器。其中，过滤字符串是在命令“**log filter**”中定义的。

通过设备中的日志过滤功能，管理员可以只收集感兴趣的日志信息，而不用收集所有的日志信息。例如，“www.site1.com”的管理员可能只需要“www.site1.com”的 HTTP 访问日志。如果知道这些日志中都包含一个关键词串“site1.com”，管理员可以定义一个日志过滤器，该过滤器可以过滤出所有与该字符串匹配的日志。从而管理员就能获得只包含他所需要日志的日志文件。

如果一个 Syslog 日志主机上配置了多个日志过滤器，只要与其中任意一个过滤字符串匹配的日志都会被过滤到该 Syslog 日志主机。

32.2.6 本地 Syslog 主机

系统允许管理员在设备上启用一个本地 syslog 主机，用于接收和存储系统日志消息。本地 syslog 主机默认设置为禁用。

启用了本地 syslog 主机后，本地 syslog 主机开始在 IP 地址 127.0.0.1 和 UDP 端口 515 上接收系统发送的系统日志。本地 syslog 主机可以为每个日志级别存储最多 500,000 条系统日志。在本地 syslog 主机上存储的系统日志可以通过 WebUI 查看和导出。

要启用本地 syslog 主机，管理员可以 CLI 里执行“**log localhost on**”命令。或者在 WebUI 上，选择**管理工具 > 系统日志 > 日志设置**，将**本地 Syslog 主机**滑块置为**启用**，然后点击**保存修改**按钮，如下图所示。

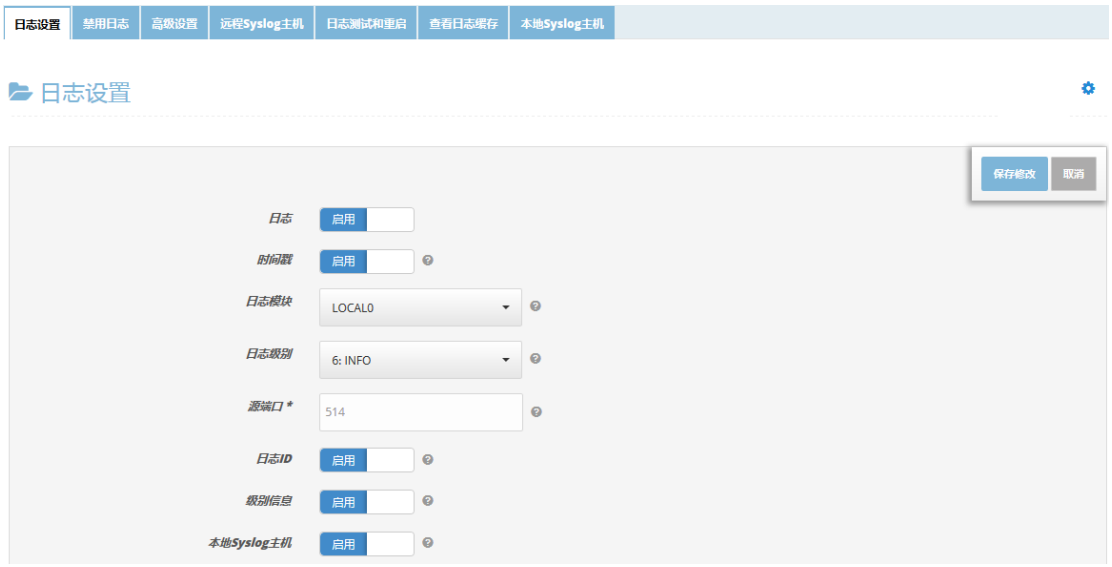

图32–1 启用本地 Syslog 主机

要查看本地 syslog 主机上存储的系统日志，点击**本地 Syslog 主机**页签，如下图所示。

图32–2 查看本地 Syslog 主机上系统日志

32.2.7 本地日志保存

系统允许管理员在设备上启用系统本地日志保存功能。启用本地日志保存功能后，系统将在硬盘上保存最近 6 个月内的系统日志。本地日志保存功能默认设置为禁用。

要获取本地保存的系统日志，管理员需要首先执行命令**“log storage on”**启用本地日志保存功能，接着执行命令**“debug log storage on”**启用生成本地日志压缩文件的功能并执行命令**“debug snapshot system”**生成名称为“fastlog_backup.tar.gz”的本地日志压缩文件。最后，管理员可以通过将本地日志压缩文件导出到远程 FTP 或 SCP 服务器，从而获取本地保存的系统日志。

32.3 日志配置

32.3.1 配置向导

下面列出了配置日志所需用的命令，相关命令的描述信息，请参考命令行使用手册。

表32–1 设备的日志配置命令

配置操作	命令行
启用日志	**log** *{on|off}*
启用 RFC 5424 Syslog	**log rfc5424 {on|off}**
配置远程主机	**log host** <*host_ip*> *[port] [udp|tcp] [host_id]*

设置日志级别	**log level** *<level>*
更改日志设备	**log facility** *<facility>*
设置 HTTP 访问日志格式	**log http** *{squid|common|combined|welf} [vip|novip] [host|nohost]* **log http custom** *<format>*
配置日志过滤	**log filter** *<host_id> <filter_id> <filter_string>*

32.3.2 配置示例

1. 启用设备的日志功能。

设备的日志功能默认是关闭的。

```
Demo(config)#log on
```

2. 启用设备的 RFC 5424 syslog 功能。

```
Demo(config)#log rfc5424 on
```

3. 设置用户接受日志信息的远程主机。

命令“**log host**”用于配置日志服务器来接收系统所产生的日志信息。日志服务器的 IP 地址应该使用 dotted IP 格式来配置，日志服务器的端口是可选的，默认端口是 514。Syslog 所使用的传输层协议可采用 UDP 和 TCP，默认使用 UDP。

```
Demo(config)#log host 10.2.37.1 514 udp1
```

4. 为主机配置日志过滤。

日志主机上最多可以配置三条日志过滤条目，并且过滤条目不能配置在 ID 为 0 的日志主机上。在下列命令运行后，只有符合过滤字符串的日志可以到达日志主机。

```
Demo(config)#log filter 1 1 "index"
```

5. 调整日志记录的级别。

设置某一日志记录的级别后，那些低于该级别的日志信息将会被系统忽略。默认的日志记录的级别为信息（info）。

```
Demo(config)#log level err
```

6. 自定义设备日志的设备。

命令“**log facility**”用于修改记录 Syslog 的自定义设备。系统为用户自定义设备预留从 LOCAL0 到 LOCAL7 一共八个设备。系统默认的设备为 LOCAL0。

```
Demo(config)#log facility LOCAL0
```

7. 设置 HTTP 访问信息的格式。

```
Demo(config)#log http squid
```

设备采用 Squid 标准格式（WELF，WebTrends Enhaced Log Format）记录 HTTP 的访问信息，当然用户也可以通过命令自定义记录这些信息的格式。

8. 生成测试信息。

命令“**log test**”用于以 emerg 的日志级别生成一条测试日志信息。

```
Demo(config)#log test
```

9. 查看和调整日志信息和配置。

命令“**show log buff** *{forward|backward} [match_str]*”用于查看日志缓冲区的日志信息，参数“backward”和“forward”分别用于查看最近产生的日志和最先产生的日志信息。

```
Demo(config)#show log buffer backward
start of buffer
<128>1 2012-07-17T06:35:26Z AN - - 100021002 - test message
```

命令“**clear log buff**”用于清空日志缓冲区的日志信息。

```
Demo(config)#clear log buffer
```

32.4 应用可视化

在系统的运行过程中，实时把握系统的运行状态对于及时的排障、保证系统的不间断服务具有重要意义。然而，了解系统的运行状态需要收集大量的数据。出于扩展性、安全性等因素，IT 的部署往往会采用分布式的、容错的架构，增加了数据收集的难度，对于收集到的大量数据，也很难提取出有价值的信息。

设备基于 WebUI 和 ELK 软件两种平台提供 SLB TCP、SLB HTTP、SSL、LLB 和 SDNS 业务的应用可视化。WebUI 和 ELK 平台将分散的系统资源、配置和应用数据进行集中记录、存储和挖掘后，将分析后的数据以直观、可视的方式展示出来。运营维护人员可以进行可视化的应用管理，随时了解应用交付情况，快速有效地获取系统状态以及定位问题。

32.4.1 WebUI 平台

WebUI 平台提供的应用可视化能力侧重于系统资源、配置以及性能方面的监控。

32.4.1.1 系统仪表盘

在 WebUI 首页（系统仪表盘），通过图形展示系统运行时间、系统状态、应用状态、网络状态、系统负载、网络负载以及连接状态等数据。另外，为 SLB、LLB、GSLB 每个模块提供单独的监控页面。

- 在 SLB 的监控页面，可以查看到虚拟服务状态、后台服务状态以及虚拟服务的入向字节数排名前 10。
- 在 LLB 的监控页面，可以查看到链路状态和链路的入向字节数排名前 10。
- 在 GSLB 的监控页面，可以查看到 SDNS 主机状态和 SDNS 请求数排名前 10。

32.4.1.2 SLB 监控中心

管理员可以通过**服务器负载均衡 > 监控中心**进入 SLB 监控中心。在 SLB 监控中心，可通过图形查看到更多详细的信息，包括：

- 所有虚拟服务的状态（Up/Down）
- 所有后台服务的状态（Up/Down）
- 虚拟服务入向/出向总字节数、总连接数排名
- 后台服务入向/出向总字节数、平均响应时间、总命中数排名
- 所有虚拟服务的总并发连接数、平均入向/出向带宽、总缓存命中率、数据总压缩比例
- 所有 SSL 主机的平均入向/出向带宽
- 所有 SSL 虚拟主机的平均入向/出向带宽
- 所有 SSL 后台主机的平均入向/出向带宽
- 每个虚拟服务的策略命中排名、关联的后台服务负载排名、并发连接数、每秒新建连接数、平均入向/出向带宽、每秒请求数、客户端连接平均 RTT 等等。在关联的后台服务负载排名图表中，如果一个后台服务关联了多个虚拟服务，显示的是所有虚拟服务的负载总数。
- 每个后台服务的并发连接数、每秒新建连接数、平均入向/出向带宽、每秒请求数、平均响应时间、返回的 HTTP 响应码数量、健康状态等等
- 每个 SSL 虚拟主机的并发连接数、每秒新建连接数、平均入向/出向带宽
- 每个 SSL 后台主机的并发连接数、每秒新建连接数、平均入向/出向带宽

32.4.2 ELK 平台

ELK 是开源软件“Elasticsearch, Logstash and Kibana”的缩写。Elasticsearch 是基于 Lucene 的分布式全文搜索引擎。Logstash 用于收集、分析和过滤日志。Kibana 负责汇集、分析和搜索重要的日志，为 Elasticsearch 和 Logstash 和提供 Web 形式的日志分析界面。管理员可以通过

https://www.elastic.co/cn/subscriptions 了解开源的 ELK 平台和其他订阅套餐的特点和支持的数据收集、管理等功能。

为了实施应用可视化，系统对日志模块进行了功能优化，优化后的日志模块支持独立控制 SLB TCP、SLB HTTP、LLB、SSL 和 SDNS 日志可视化机制。管理员可以通过“**log appvisual**”命令为 SLB TCP、SLB HTTP、LLB、SSL 和 SDNS 模块启用可视化日志功能，使系统收集到 ELK 服务器需要的日志信息。通过将 ELK 服务器设置为设备的日志服务器，设备会将收集到的日志发送给 ELK 服务器。

启用 SLB TCP、SLB HTTP、LLB、SSL、SDNS 模块的可视化日志功能后，设备将在 ELK 平台上实现以下应用服务的可视化管理和监控：

32.4.2.1 SLB HTTP

- HTTP 请求类型分布
- URL 访问排名
- 同一服务组内不同服务器的 HTTP 响应延迟
- 单个 URL 响应延迟
- 访问请求总数
- URL 或服务器延迟平均趋势
- URL 或服务器最大或最小延迟
- 大流量 URL 分布
- URL 访问失败率分析
- 不同大小的响应分布
- HTTP 协议版本分布
- 访问者位置追踪
- HTTP 响应码分布和趋势
- HTTP 响应错误率统计
- 请求路线分布
- 浏览器类型分布

32.4.2.2 SLB TCP

- 虚拟服务当前连接数
- 虚拟服务每秒新建连接数（CPS）
- 虚拟服务平均入向/出向带宽

- 虚拟服务平均每秒入向/出向报文数（PPS）
- 客户端握手时延 RTT
- 后台服务当前连接数
- 后台服务每秒新建连接数（CPS）
- 后台服务平均入向/出向带宽
- 后台服务平均每秒入向/出向报文数（PPS）
- 服务端握手时延（基于健康检查）

32.4.2.3 SDNS

- 请求的地理分布
- 响应的地理分布
- 解析失败统计
- 智能分析统计
- 非智能分析统计
- 总请求数
- 解析失败数
- 总响应数
- 请求和响应的域名排名前 30
- 解析总数
- 响应总数

32.4.2.4 SSL

- SSL 客户端排名前 20
- SSL 密码套件排名前 20
- SSL 版本信息
- SSL 握手总计数
- SSL SNI 信息
- SSL 握手失败原因
- SSL 握手成功、失败事件分布

32.4.2.5 IPV6

- 总请求数
- IPv6 客户端总数
- 实时请求总数
- IPv6/IPv4 请求比例

32.4.2.6 LLB

- 链路健康状态（Up 或 Down）
- 链路总数
- 各链路带宽（总带宽、上行带宽、下行带宽）
- 单个链路带宽使用率
- 应用流量占比分析（基于深度报文检测）
- 链路状态监控（正常、繁忙、离线、无数据）

ELK 可以基于物理机和 VMware、KVM、华耀 AVX 虚拟化平台部署。关于 ELK 平台的具体部署方式，请联系客户支持获取 ELK 部署指导。

第33章 系统管理

33.1 管理工具

33.1.1 概述

本章将介绍如何使用设备的管理工具，包括如何下载新的软件，如何重启设备，如何将现有配置恢复到一个已保存的配置以及如何恢复到出厂预设状态等内容。

本章介绍的这些配置将关系到您的设备的运行情况及它与外部网络的关联。在 Web 接口上点击“管理工具”目录，您在那里可以找到一系列子目录允许您更改管理员口令、配置同步、设置 SNMP traps 以及定义重启策略等。另外，您也可以通过命令行来设置这些功能。

33.1.2 管理工具配置

33.1.2.1 配置向导

下面列出了配置管理工具所需的相关命令，相关命令的描述信息，请参考命令行使用手册。

表33–1 设备配置管理工具配置命令

配置操作	命令行
配置外部认证	**admin aaa {on|off}** **admin aaa method** *[radius|tac_x]* **admin aaa server** *<server_id> <host_name|ip_address> <port> <secret>*
系统关闭及重启	**system shutdown** *[halt|poweroff]* **system reboot** *[interactive|noninteractive]*
配置文件维护	**clear config file** **clear config secondary** **clear config primary** **clear config all** **clear config factorydefault** **clear config timeout** **write memory** **write file** *<file_name>* **write net tftp** *<ip_tftp> <file_name>* **write net scp** *{remote_server_ip|name} <user_name> <config_file_name>*

	config memory **config net** *<tftp_server_ip> <config_file_name>* **config file** *<file_name>*
软件升级	**system update** *<url> [immediate|deferred] [partition_number]*
配置同步	**synconfig peer** *<peer_name> <peer_ip>* **synconfig challenge** *<code>* **synconfig to** *<name>* **synconfig from** *<name>*
SDNS 同步	**synconfig peer** *<peer_name> <peer_ip>* **synconfig challenge** *<code>* **synconfig sdns to** *<peer_name>*
时间同步	**ntp {on|off}** **ntp server** *<ip> [version]* **show ntp** **clear ntp**
管理 XML-RPC 功能	**xmlrpc {on|off}** *[https|http]* **xmlrpc port** *<port>* **show xmlrpc** **clear xmlrpc**
远程管理	**ssh remote** *"user@hostname"* **telnet** *"host port"*
安全增强模式	**webui semode {on|off}** **webui semode custom import** *<url>* **webui semode custom export** *<url>*

33.1.2.2 配置示例

33.1.2.2.1 配置外部认证

如果您有外部认证服务器（RADIUS/TACACS+），您可以使用这些服务器进行 SSH/WebUI 的登录验证。当您输入的用户名在系统中不存在，并且“**admin aaa**”命令设置为打开时，外部认证将被启用。执行命令如下：

```
Demo(config)#admin aaa on
Demo(config)#admin aaa method RADIUS
Demo(config)#admin aaa server es01 "10.1.1.1" 1812 radiussecret
Demo(config)#admin aaa server es02 radius_host 1812 radiussecret
```

33.1.2.2.2 配置系统维护

使用“**quit**”命令，您可以退出命令行模式。如果您想终止设备的所有网络交互，可以使用“**system shutdown**”命令。

```
Demo(config)#system shutdown
```

运行该命令后，设备会显示一条警告信息，并询问您是否真的要关闭设备。输入“YES”回车，设备就开始关闭。在 60 秒后，使用者可以关闭设备了。

有时更改系统功能配置后，您可能需要重新启动设备。使用“**system reboot**”命令即可重启设备。

```
Demo(config)#system reboot
```

33.1.2.2.3 配置文件维护

如果您想测试一些新的配置项，但又不想覆盖现有的配置，系统为您提供了配置文件的维护命令。通常，您可以使用“**write memory**”命令将运行配置保存到磁盘上。您还可以使用“**config file**”命令将现有的配置保存到指定的文件中。您还可以通过 TFTP 方式来导入或导出配置。

当您使用“**write memory**”命令时，请记住，设备重新启动后将会装载该命令所保存的配置。如果您更改配置项后想清除当前正在运行的配置，请使用“**clear config**”命令。

```
Demo(config)#clear config all
```

现在设备已恢复到出厂的默认设置了。

在任何时候，如果您想导入以前保存的配置，您首先需要如上文所述的那样清除当前的运行配置。清除完成后，您可以导入新的配置。设备为您提供了三个可以保存配置的地方。

- “memory”：当设备重启时会加载该文件作为当前配置。
- “file”：可以保存各种不同的配置到磁盘文件上。
- “net”：可以保存配置文件到网络上的其它远程路径。

下面是保存和加载配置文件的三类命令。

```
Demo(config)#write net tftp 10.10.0.3 default_config
```

如果想将配置文件重新导入并覆盖正在运行的配置文件，需要运行以下命令。

```
Demo(config)#config memory
Demo(config)#config file new_lb
Demo(config)#config net tftp 10.10.0.3 default_config
```

如果您想查看某个特定的配置文件（将内容显示到屏幕），请在“**show config file**”命令后增加文件名：

```
Demo(config)#show config file new_lb
```

当设备运行时加载一个已保存的配置时，请注意，保存的配置项是被合并到当前运行的配置项中。因此，您通常需要首先清除设备上某些相应的配置，然后再导入新的配置。例如，当前已经定义了五个后台服务器，然后执行“**config net tftp**

10.10.0.3 default_config”命令，如果将导入的配置文件中也定义了五个相同名称的后台服务器，那么您将得到一个错误提示，因为后台服务器名称不能重复。

33.1.2.2.4 软件升级程序

如果想查看当前设备的软件版本，请使用“**show version**”命令。

如果您想升级到新的版本，请使用以下步骤。

首先，请联系客户支持获得软件和文档库的访问权。请联系您的客户支持或发送 Email 到 support@arraynetworks.com.cn。

当您收到口令或者客户支持的确认后，您就可以进行升级了，您可以首先从华耀公司的网站上下载软件包。您可以把下载的软件包放到本地的 Web 服务器或匿名的 FTP 服务器上。

建议您使用串口终端进行升级。当您连接到终端后，您可以使用“**system update**”命令来更新。当前更新支持 HTTP 和 FTP 两种方式。两种方式的命令行相同，只是 URL 不同。

```
Demo(config)#system update
http://10.3.0.120/newapv/build/ArrayOS-Rel_APV_10_4_3_15.array
```

注意：

- 如果您使用域名方式，如：system-update http://s5.sj.example.com 时，请确保设备上正确的配置了域名解析，您可以使用“**ip nameserver**”命令为“s5”主机定义域名解析服务器或者使用“**ip host**”命令为“s5”主机定义本地解析，否则您再下载映象时将收到一个错误信息。
- 请勿在四分区系统硬盘上安装仅支持双分区的软件版本。
- 若 SSH 连接在安装包下载过程中断开，则升级失败；若 SSH 连接在安装包下载完成后断开，则升级成功。

接下来系统将关闭所有的负载均衡特性开始下载升级软件包，并进行确认，确认软件包来自华耀公司后开始安装。如果映象档有任何问题，将停止升级并在屏幕上给出错误提示，反之，您会获得升级成功的提示信息，并且设备将重新启动。重新启动后，您可以使用“**show version**”命令确认升级结果。

软件许可

设备的一些功能受到软件许可密钥的控制。如果您需要这些功能，请与客户支持联系：support@arraynetworks.com.cn 来获得新的软件许可密钥。

33.1.2.2.5 配置同步

设备的配置同步功能允许管理员在多台设备之间传递配置信息。配置同步功能的一系列命令允许您管理和配置在同一网络或在不同网络上通过 NAT 连通的多台设备。您可以从一台设备上将配置信息传递到另一台设备上。使用配置同步功能，

您可以迅速建立起主从冗余（Active-Standby）配置。下面的部分将为您介绍如何使用此功能。

假设两台设备的 IP 地址分别为 192.168.1.1 和 192.168.1.2，如果需要将设备 1 的配置同步到设备 2，需要做如下配置：

1. 在设备 1 上开启配置同步。

```
Demo1(config)#synconfig challenge synpassword
Demo1(config)#synconfig peer machine1 192.168.1.1
Demo1(config)#synconfig peer machine2 192.168.1.2
```

2. 在设备 2 上开启配置同步。

```
Demo2(config)#synconfig challenge synpassword
Demo2(config)#synconfig peer machine1 192.168.1.1
Demo2(config)#synconfig peer machine2 192.168.1.2
```

3. 将配置从设备 1 同步到设备 2。

```
Demo1(config)#synconfig to machine2
```

注意：如果使用“**synconfig**”命令进行同步操作的接口防火墙被打开，您可能需要增加相应的防火墙访问规则使得数据报可以在设备（设备和同步节点）65519 服务端口上往来传送。

33.1.2.2.6 SDNS 配置同步

设备的 SDNS 配置同步功能允许管理员在同一网络或在不同网络上通过 NAT 连通的多台设备之间传递除 SDNS 成员配置信息之外的 SDNS 配置信息和 BIND 9 域文件。下面的部分将为您介绍如何使用此功能。

1. 在本地设备上进行如下配置：

```
Demo1(config)#synconfig challenge synpassword
Demo1(config)#synconfig peer peerlocal 172.16.83.180
Demo1(config)#synconfig peer peerremote 172.16.83.120
```

2. 在远程设备上进行如下配置：

```
Demo2(config)#synconfig challenge synpassword
Demo2(config)#synconfig peer peerlocal 172.16.83.180
Demo2(config)#synconfig peer peerremote 172.16.83.120
```

3. 将 SDNS 配置从本地设备同步到远程设备上。

```
Demo1(config)#synconfig sdns to peerremote
```

33.1.2.2.7 NTP 时间同步

NTP（Network Time Protocol，网络时间协议）时间同步器的主要用来定期同步设备的系统时间与指定 NTP 服务器时间。

开启 NTP 时间同步器之后，每隔约 15 分钟，设备的系统时钟会自动与 NTP 服务器时间同步一次。

注意：

- 在启用 NTP 时间同步器以后，请不要再手动设置设备的系统时间。
- 设备只应用作 NTP 客户端，而非 NTP 服务器。

在系统配置多台 NTP 服务器的情况下，设备会根据收到来自每个 NTP 服务器的时间信息包内容计算出传递时间数据的延迟，然后进行比较，选择一个延迟最小的 NTP 服务器进行时间同步。

1. 配置 NTP 服务器。

```
Demo1(config)#ntp server 207.46.197.32 4
```

2. 启用 NTP 时间同步器。

```
Demo1(config)#ntp on
```

用户可以使用“**show ntp**”命令查看当前 NTP 配置。

```
Demo1(config)#show ntp
ntp server 172.16.85.12 4
ntp server 172.16.85.60 4
ntp on 0
time since restart:         20
time since reset:           20
packets received:           2
packets processed:          1
current version:            1
previous version:           0
declined:                   0
access denied:              0
bad length or format:       0
bad authentication:         0
rate exceeded:              0

        remote                  local          st poll reach    delay      offset       disp
==============================================================================
=
=172.16.85.12       172.16.85.80       16     64       0 0.00000    0.000000 3.99217
```

```
*172.16.85.60      172.16.85.80      3    64      1 0.00189   0.000050 2.81735
```

以下解释了输出信息中的内容：

表33–2 设备管理工具输出信息

输出信息	含义
Time since restart	上次系统重启后运行的时间，以小时为单位。
Time since reset	统计信息重置和系统统计监控文件更新之后运行的时间。这个信息是专门为繁忙服务器而收集的，例如操作系统为 NIST、USNO 的服务器，并能为它们早期探测到 clogging 攻击。
Packets received	收到的数据报的总数。
Packets processed	上一个数据报发出之后收到的响应包数目。
Current version	与当前 NTP 版本兼容的数据报的数目。
Previous version	与之前 NTP 版本兼容的数据报的数目。
Bad version	与任何一个 NTP 版本都不兼容的数据报的数目。
Declined	因主机上已经存在 NTP 程序而拒绝日期设置请求的次数。
Access denied	由于某种原因被拒绝访问的数据报的数目。
Bad length or format	具有无效长度、格式或者端口号的数据报的数目。
Bad authentication	未通过验证的数据报的数目。
Rate exceeded	由于速率限制而被丢弃的数目包。

33.1.2.2.8 RESTful API

RESTful 应用程序编程接口（Application Programming Interface，API）是除了 CLI、WebUI 和 XML-RPC 外的另一种管理方法。

通过 RESTful API，管理员可以通过 HTTP 或 HTTPS 协议发送 RESTful API 请求到设备从而对系统资源执行创建、读取、更新和删除操作。系统将执行这些操作并通过 RESTful API 响应返回结果。为保证 RESTful API 调用的安全性，管理员应该在 RESTful API 请求中包含有效的凭据。否则，系统将拒绝 RESTful API 请求的访问。

每一个可被管理的系统资源都用 URL 标识，格式为“http(s)://management_ip:port/path[?query_name=value]”。查看设备支持的所有 RESTful API，请参见《设备 RESTful API 参考手册》。

管理员可以使用安装在 Web 浏览器上的 RESTful API 客户端或者使用编程工具，例如 Java、Python 或者 Perl 发送 RESTful API 请求。详细信息请参见《设备 RESTful API 用户手册》。

使用 RESTful API 前，必须先启用 RESTful API 服务。为了增强系统的安全性，管理员可以配置 RESTful API 源 IP 地址限制规则，用于控制可以访问 RESTful API 服务的源地址范围。

此外，为了增强 RESTful API 服务的安全性，管理员可以自定义 SSL 设置：

- 设置 SSL 协议版本
- 设置 SSL 密码套件
- 导入 PEM 格式的证书链
- 导入客户端 CA 证书
- 启用或禁用客户端认证
- 启用或禁用客户端认证强制模式

➢ **CLI 配置示例**

要设置 RESTful API 服务的监听 IP 地址，执行如下命令：

```
Demo(config)#restapi ip 192.168.1.100
```

要配置 RESTful API 源 IP 地址限制规则，执行如下命令：

```
Demo(config)#restapi source 192.168.0.0 255.255.0.0
```

要启用 RESTful API 服务并指定使用的协议和监听的端口号，执行如下命令：

```
Demo(config)#restapi on https 9997
```

完成上述配置后，RESTful API 服务将启用，并通过 HTTPS 协议监听 IP 192.168.1.100 和端口 9997。

要创建一个具有 API 访问权限的管理员账户，执行如下命令。

```
Demo(config)#user rest password1 api
```

为 RESTful API 服务设置 SSL 协议版本，执行如下命令：

```
Demo(config)#restapi ssl settings protocol "TLSv12"
```

为 RESTful API 服务设置 SSL 密码套件，执行如下命令：

```
Demo(config)#restapi ssl settings ciphersuites
"ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:AES128-GCM-SHA256:AES256-GCM-SHA384"
```

为 RESTful API 服务导入证书链文件，执行如下命令：

```
Demo(config)#restapi ssl import certificate ftp://10.8.6.20/cert/chain.pem
```

为 RESTful API 服务导入客户端 CA 证书，执行如下命令：

```
Demo(config)#restapi ssl import clientca ftp://10.8.6.20/cert/client.pem
```

为 RESTful API 服务启用 SSL 客户端认证功能，执行如下命令：

Demo(config)#**restapi ssl settings clientauth enable**

为 RESTful API 服务启用 SSL 客户端认证强制模式，执行如下命令：

Demo(config)#**restapi ssl settings authmandatory enable**

33.1.2.2.9 XML RPC

XML RPC 允许客户端在远程使用 CLI 命令管理设备。从某种意义上说，这是一种 WebUI 管理方式的辅助。

XML RPC 是一种互联网上的远程程序呼叫协议，它使用 HTTP 协议作为传输机制，使用 XML 语言作为编码。

如下图所示，客户端发送了一个 HTTP 请求到设备，XML RPC 信息相当于这个 HTTP 请求的主体部分，包含了需要运行的命令和参数。设备对 XML RPC 信息进行译码并提取相关的命令予以执行。最后，将执行的结果以 XML 代码的形式返回给客户端。

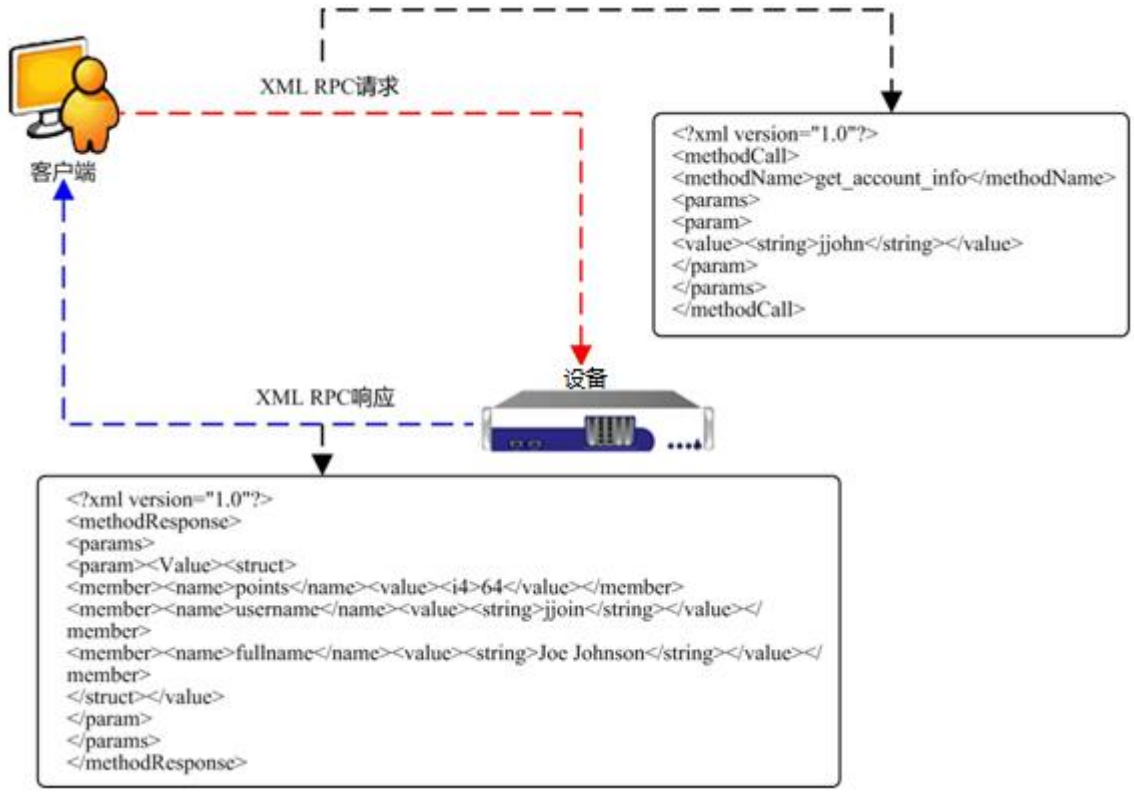

图33–1 XML RPC 工作机制

为了开启客户端与设备之间的通信，我们必须在客户端上建立一个名为“demo_xmlrpc.pl”的脚本文件。建立这个脚本的命令是：

demo_xmlrpc.pl –d <*address*> **-p** <*port*> **-f** <*data_file*>

在这个命令中，address 参数是设备的 IP 地址。port 参数是需要监听的 HTTP 服务器的端口。data_file 参数是 XML RPC 消息的文件名。

XML RPC 消息被 XML 语言格式化之后，包含了一个 methodCall 标记。

下面是一个主体包含了 XML RPC 消息的 HTTP 请求：

```
POST  /cgi-bin/xmlrpc_server  HTTP/1.1
Content-Type: text/xml
Content-Length: xxx

<?xml version='1.0' ?>
<methodCall>
<methodName>slb_real</methodName>
<params>
 <param>
  <value>
   <struct>
    <member>
     <name>enable_passwd</name>
     <value>
      <string>****</string>
     </value>
    </member>
    <member>
     <name>protocol</name>
     <value>
      <string>http</string>
     </value>
    </member>
    <member>
     <name>name</name>
     <value>
      <string>array</string>
     </value>
    </member>
    <member>
     <name>ip</name>
     <value>
      <string>10.1.1.1</string>
     </value>
    </member>
    <member>
     <name>port</name>
```

```
        <value>
          <int>80</int>
        </value>
      </member>
      <member>
        <name>maxconns</name>
        <value>
          <int>1000</int>
        </value>
      </member>
      <member>
        <name>hctype</name>
        <value>
          <string>tcp</string>
        </value>
      </member>
      <member>
        <name>hcup</name>
        <value>
          <int>1</int>
        </value>
      </member>
      <member>
        <name>hcdown</name>
        <value>
          <int>1</int>
        </value>
      </member>
    </struct>
  </value>
 </param>
</params>
</methodCall>
```

在这个例子中，前三行是 HTTP 请求的表头，其余部分是 HTTP 请求的主体。

```
POST   /cgi-bin/xmlrpc_server   HTTP/1.1
Content-Type: text/xml
Content-Length: xxx
```

XML RPC 消息的前三行中，“**slb_real**”是命令“**slb real** <*protocol*> <*name*> <*ip*> *[port] [maxconns] [hctype] [hcup] [hcdown]*”的 XML RPC 格式。

```
<?xml version='1.0' ?>
```

```
<methodCall>
<methodName>slb_real</methodName>
```

接下来的部分进入了特权用户模式并且提供了密码，这样相关的命令就可以在特权模式下运行了。“enable_password”是其中的关键词。而密码的值则放在了<string>标签中。特权模式密码会包含在每一个 XML RPC 消息中。

```
<member>
        <name>enable_passwd</name>
        <value>
          <string>****</string>
        </value>
</member>
```

在这里，“protocol”参数指定了“**slb_real**”命令的方法，其中“protocol”是关键词，它的值被放在了<string>标签中。

```
<member>
        <name>protocol</name>
        <value>
          <string>http</string>
        </value>
</member>
```

在这个例子中，“**slb_real**”命令的参数包括了协议、名称、IP、端口、最大连接数、健康检查类型和健康检查的开关等。其中，协议、名称和 IP 是必需的，而端口、最大连接数、健康检查类型和健康检查的开关是可选项。

注意：在一个 HTTP 请求中，一次可以包含多个 XML RPC 信息。

如果这个请求成功，设备会返回一个如下格式的 HTTP 响应：

```
<?xml version=’1.0’ ?>
<methodResponse>
 <params>
  <param>
   <value>
    <string>xmlrpc command successful</string>
   </value>
  </param>
 </params>
</methodResponse>
```

如果 XML RPC 消息中的命令包含了“**show**”命令，那么输出信息会取代“xmlrpc command successful”这句话的位置。如果出现任何错误，错误的信息也会在这个位置显示。

我们可以使用下列命令来配置 XML RPC 功能。

1. 开启 XML RPC。

```
Demo1(config)#xml on https
```

2. 配置 XML RPC 所监听的端口号。

```
Demo1(config)#xml port 9999
```

33.1.2.2.10 远程管理

设备支持使用者使用 Telnet 和 SSH 协议连接到其他设备上进行远程管理工作，协助使用者远程排查其他设备上的问题和故障。

在设备上运行命令“**telnet** *"host port"*”来使用 Telnet 功能，如下所示：

```
Demo#telnet “‘172.16.2.182 -4’”
Trying 172.16.2.182...
Connected to 172.16.2.182 -4.
Escape character is '^]'.
Trying SRA secure login:
User (root): array
Password:
[ SRA accepts you ].....succeed
```

在设备上运行命令“**ssh remote** *“user@hostname”*”来使用 SSH 功能，如下所示：

```
Demo#ssh remote "root@172.16.85.240"
root@172.16.85.240's password:
Linux libh-server1 2.6.32-22-generic #33-Ubuntu SMP Wed Apr 28 13:27:30 UTC 2010 i686
GNU/Linux

Welcome to Ylmf_OS!
 * Information:    http://www.ylmf.com/

0 packages can be updated.
0 updates are security updates.

Last login: Wed Apr 20 00:39:35 2011 from 10.3.46.1
root@libh-server1:~#
```

33.1.2.2.11 仪表盘（Flight Deck）

设备的监控功能可以提供很多实用的统计信息来对设备性能和网络活动进行统计。它的图形接口可以使您很方便的在一张图表中查看各种统计信息，对设备状态进行监控。这个图形接口被称作 Flight Deck。您可以通过点击 WebUI 接口首页的“仪表盘”标签来查看仪表盘信息。

Fight Deck 在启动后，可以在浏览器窗口中显示很多网络实时信息。在窗口上方，您可以看到有关 server health（服务健康）、request rate（请求率）、cache hits（缓存命中率）和 system usage（系统使用率）等信息。窗口的左面会显示 TCP、HTTP 和 SSL 的连接数信息。这三个连接数累加起来作为“**show memory**”命令中“TCP pcb”项目的输出结果。有时，一次客户请求中会包含一对 TCP 连接，例如，一次服务器负载均衡的客户请求通常会产生两个连接，一个从客户端到设备，一个从设备到服务器。

Flight Deck 的中间部分由两个可配置的图表组成。您只需要从下拉菜单中进行选择，就可以实时跟踪想了解的项目信息。

每个图表都存在有两个下拉菜单。第一个菜单，被称作“Graph Type”（类型），包含了一系列统计项目的名称。注意：两个图表的菜单内容是相同的。第二个菜单，被称作“Interval”（间隔），用来控制水平轴显示的时间粒度，同时也是设备刷新图表的间隔时间。它的默认值是 5 秒，设备每 5 秒更新一次图表信息，水平轴的时间显示为 5 的倍数。

对有些统计，需要较小的时间间隔。例如，每隔 30 秒统计处理的数据报个数。反之，有时统计需要较长的时间间隔。例如，为了了解使用者的登录情况，可能您需要每小时统计一次并发会话的个数。

需要注意的是，如果您要查看统计信息，必须首先启动 SNMP。您可以通过 WebUI 接口的“管理工具”特性链接下的“图形->SNMP”页面启动 SNMP。

注意：为安全起见，强烈建议您在启动 SNMP 功能后修改默认的 SNMP 团体名，防止系统信息被非法窃取。

下面是在图表中可选择的统计信息选项及其含义。

表33–3 Flight Deck 统计信息选项及含义

统计信息选项		含义
TCP	Active Opens	CLICKTCP 打开的连接数（从 CLOSED 状态直接转到 SYN-SENT 状态的数目）。
	Passive Opens	CLICKTCP 接收的连接数（从 LISTEN 状态直接转到 SYN-RCVD 状态的数目）。
	Open Failures	CLICKTCP 的连接数（从 SYN-SENT 或 SYN-RCVD 状态直接转到 CLOSED 状态的数目加上从 SYN-RCVD 状态直接转到 LISTEN 状态的数目）。

统计信息选项		含义
	Established Conns	CLICKTCP 的连接数（当前状态是 ESTABLISHED 或者 CLOSE-WAIT）。
	Resets Received	CLICKTCP 的连接数（从 ESTABLISHED 或 CLOSE-WAIT 状态直接转到 CLOSED 状态）。
	Resets Sent	CLICKTCP 中包含 RST 标志的分包的个数。
	Retransmits	重发的分包总数，也就是 CLICKTCP 中包含有一个或多个已经发送的分包的总个数。
	Packet Errors	接收错误的分包总数（如：错误的 CLICKTCP 校验）。
IP	Packets Received	接收的 IP 包总数。
	Packets Sent	发送的 IP 包总数。
	Bytes Received	接收的字节数。
	Bytes Sent	发送的字节数。
	Header Errors	错误的 IP 包总数。
	Unknown Protocol	未知协议的 IP 包总数。
	No Route Out	无法路由的 IP 包总数。
UDP	Packets Received	收到的 UDP 包总数。
	Packets Sent	发送的 UDP 包总数。
	Invalid Ports	包含无效端口号的 UDP 包总数。
	Packet Errors	错误的 UDP 包总数。
ICMP	Messages In	收到的 ICMP 消息总数。
	Errors In	收到的 ICMP 错误消息总数。
	Unreachable In	收到的 ICMP 无法到达消息总数。
	Echoes In	收到的 ICMP 探测请求（echoes in）消息总数。
	Echo Replies In	收到的 ICMP 探测响应（replies in）消息总数。
	Messages Out	发出的 ICMP 消息总数。
	Errors Out	发出的 ICMP 错误消息总数。
	Unreachable Out	发出的 ICMP 不可到达消息总数。
	Echoes Out	发出的探测请求（echoes in）消息总数。
	Echo Replies Out	发出的 ICMP 探测响应(replies in)消息总数。
CPU	% CPU Utilization	当您选择此选项后，图表中显示的是设备 CPU 的使用百分比。如果这个百分比是“75”，也意味着空闲（Idle）CPU 为“25”。
Proxy-Cache	% HTTP/HTTPS requests	每秒钟 HTTP/HTTPS 请求个数。
	% Established Client Connections	与客户端建立的连接个数。
	% Established Server Connections	与服务器建立的连接个数。
Compression	% Bytes Received	收到的压缩字节总数。
	% Bytes Sent	发送的压缩字节总数。

33.1.2.2.12 IPv6 外链转换工具

设备的 IPv6 外链转换工具用于处理 IPv6 站点中包含 IPv4 外链的问题，即 IPv6 天窗问题。IPv6 外链转换工具能够解析出指定域名的内容中包含的需转换的 IPv4 链接，并自动生成对应的 SLB 配置文件来完成 IPv4 到 IPv6 链接的转换。管理员可以对生成的 SLB 配置进行修改并应用到设备中。

目前，管理员可以通过 WebUI 页面（**管理工具 > IPv6 > IPv6 工具 > 在线转换**）在线使用 IPv6 外链转换工具，也可以通过 WebUI（**管理工具 > IPv6 > IPv6 工具 > Windows 安装包下载**）下载工具至本地进行使用。关于此功能使用方法，请参阅《IPv6 外链转换工具使用指南》。

33.1.2.2.13 TCP 重置错误码统计

系统提供 TCP 重置错误码统计，用于帮助管理员进行故障排查。

目前，管理员可以通过 WebUI 页面（**管理工具 > 问题排查 > TCP 重置错误码统计**）查看 TCP 重置错误码统计。

关于 TCP 重置错误码的解释，请参考《重置错误码参考指南》（点击顶部栏中**文档**按钮，点击**重置错误码参考指南**链接）。

33.1.2.2.14 安全增强模式

系统允许用户使用 WebUI 安全增强模式来对 WebUI 的流量进行 WAF 防护，并对相关漏洞及时添加或调整规则防护。

要使用 WebUI 安全增强模式，管理员需要导入自定义攻击签名配置文件。另外，系统支持将自定义攻击签名相关配置导出到外部 FTP 服务器或者将生成的日志导出到外部 FTP 服务器。

- **配置示例**

1. 在设备上运行“**webui semode on**”命令启用 WebUI 安全增强模式。

```
Demo1(config)#webui semode on
```

30. 在设备上运行“**webui semode custom import** <*url*>”命令为 WebUI 安全增强功能导入自定义攻击签名配置文件。

```
Demo1(config)#webui semode custom import ftp://1.1.1.1/test/se_custom_rule.conf
```

31. （可选）在设备上运行“**webui semode custom export** <*url*>”命令为 WebUI 安全增强功能将自定义攻击签名相关配置导出到外部 FTP 服务器。

```
Demo1(config)# webui semode custom export ftp://1.1.1.1/test/
```

32. （可选）在设备上运行“**webui semode log export** <*url*>”命令为 WebUI 安全增强功能生成的日志导出到外部 FTP 服务器。

```
Demo1(config)# webui semode log export ftp://1.1.1.1/test/
```


33.2 管理员设置和权限管理

33.2.1 概述

设备允许创建系统管理员，并为该管理员指定对设备的访问控制级别（Enable 级别和 Config 级别）。如果需要对管理员的设备配置和操作权限进行更加精确的控制，可以使用基于角色的权限管理功能控制管理员可以执行的 CLI 命令。

33.2.2 系统管理员

设备允许创建两类管理员：Enable 级别和 Config 级别管理员。Enable 级别管理员只能执行 Enable 和 User 级别允许的所有命令。Config 级别管理员可以执行 Config、Enable 和 User 级别允许的所有命令。

33.2.3 基于角色的权限管理

基于角色的权限管理功能通过为管理员分配角色控制管理员可以执行的 CLI 命令，实现了更加灵活的管理员权限控制。

一个管理员可以分配一个或多个角色，多个角色之间是逻辑“或”关系。如果任何一个角色允许执行某条 CLI 命令，则允许该管理员执行该命令；如果所有的角色都不允许执行某条 CLI 命令，则不允许该管理员执行该命令。如果没有为管理员分配角色，则该管理员可以执行配置的访问级别允许的所有命令。

角色是一组权限规则的集合。权限规则由规则字符串和操作权限组成。

- **规则字符串**

规则字符串定义了一个或一组命令行配置。在实际配置时，支持以下三种形式：

- 完整形式：slb real http ‘r1’ 172.16.2.250 80 1 tcp 3 3
- 包含部分参数的不完整形式：slb real http ‘r1’
- 包含部分命令行主体的不完整形式：slb real

配置规则字符串时，请注意如下限制：

- 命令行主体部分不允许节略。
- 命令行参数部分区分大小写。
- 整个规则字符串需要置于双引号内。如果字符串内部需要使用双引号，则用单引号来代替。
- 命令行中如果包含多个连续空格，将被视为一个空格。

系统从首字母开始，将管理员输入的 CLI 命令行与定义的规则字符串进行比较。如果 CLI 命令行与该规则字符串相同或包含该规则字符串，则认为 CLI 命令行匹配该规则，将受该规则的控制。

- **操作权限**

操作权限包含“permit”和“deny”两种。根据操作权限的不同，权限规则可以分为“permit”规则和“deny”规则，分别用于控制允许或不允许某个角色执行一个或一组命令。如果没有为某个角色配置任何“permit”规则，则系统不允许该角色执行所有访问级别允许的 CLI 命令。

当 CLI 命令同时匹配上了“permit”和“deny”规则时，系统采取“deny”规则优先的策略。

例如：

role1：

- “permit”规则：“no slb group”
- “deny”规则：“no slb group method”

则系统将不允许 role1 执行所有以“no slb group method”开头的命令，但允许执行其他以“no slb group”开头的 CLI 命令。

role2：

- “permit”规则：“no slb group method”
- “deny”规则：“no slb group”

因为“deny”规则优先，系统将不允许 role2 执行任何以“no slb group”开头的 CLI 命令，尽管“permit”规则允许执行“no slb group method”开头的命令。

如果希望禁止某个角色执行与某个功能（如 LLB）相关的配置、显示和删除操作，应为该角色配置如下权限规则：

- “deny”规则：“llb”
- “deny”规则：“no llb”
- “deny”规则：“show llb”
- “deny”规则：“clear llb”

注意：

- 系统提供两种预定义的角色“SLB”和“NETWORK”，可以执行“**show role predefined**”命令查看两种预定义角色的权限。
- WebUI 的查看操作依赖于 CLI 中的“**show…**”命令，因此，在 WebUI 上，为某个管理员分配特定角色后，必须再为这个角色配置一条“Permit”过滤规则，

允许该角色执行“**show…**”命令。

33.2.4 三权分立功能

三权分立功能可以满足不同应用场景下对操作权限区分的需求。为了支持该功能，系统提供了三种预定义的三权分立角色，并为这三种角色制定了预定义规则。三种预定义的三权分立角色为管理员角色（role_administrator）、操作员角色（role_operator）和审计员角色（role_auditor）。

启用该功能后，系统将自动创建三个用户，即管理员（Administrator）、操作员（Operator）和审计员（Auditor），并自动将其与相应的预定义角色绑定，默认密码为 admin。其他新创建的用户只有绑定角色后才可以执行命令行。

禁用该功能后，在该功能启用时创建的所有用户都将无法通过 SSH 和 WebUI 方式登录设备。

三个管理员账户权限如下：

- 管理员：有权限修改（除其他管理员外）所有用户的密码，且具有除日志操作以外的所有命令的执行权限。
- 操作员：只能修改其自己的账户密码。具有除日志、角色、用户和密码卡管理员操作以外的所有命令的执行权限。
- 审计员：只能修改自己的账户密码。具有“show”相关命令和部分日志操作的执行权限。

关于三个管理员账户操作权限的更多详细信息，请通过命令“**show role name**”、“**show role permit**”和“**show role deny**”查看。

33.2.5 WebUI 双因素认证登录

身份认证涉及三方面要素，即需记忆的身份认证内容（如密码）、认证设备（如 USB Key）和本身特征（如指纹）。由于传统的帐号密码登录方式只涉及单一的认证要素，存在被盗或泄漏的风险。为增强 WebUI 客户端登录的安全性，系统支持 WebUI 双因素认证，以此来提升设备安全访问的强度。

目前，系统已支持使用第三方 USB Key 设备作为认证设备来实现双因素认证。要为特定用户启用 WebUI 双因素认证功能，管理员必须在 USB Key 设备中导入证书，并将证书与用户账号进行绑定，最后再为此用户启用 WebUI 双因素认证功能。下次登录设备 WebUI 时，该用户必须输入帐号密码和 USB Key 的 PIN 码向认证服务器请求身份认证信息才能登录设备。

注意：

- 该功能目前支持龙脉 USB Key 设备。

- 系统目前支持使用 SM2 算法加密的公钥证书。

➢ **启用 USB Key 认证的配置流程**

1. 安装 USB Key 管理工具并导入证书。

   在使用 USB Key 访问设备前，管理员需要安装相应的 USB Key 管理工具（认证客户端软件 GM3000）。管理员需要通过该管理工具导入证书，修改初始 PIN 码。

2. 安装 WebUI 插件。

   打开 WebUI 登录界面，根据提示安装认证插件。该插件用于实现浏览器和 USB Key 管理工具的交互。安装该插件后，将能在 WebUI 中查看导入 USB Key 设备的证书。

3. 绑定证书与用户。

   安装完 WebUI 插件后，在**系统>用户管理>系统管理员>WebUI 双因素认证**页面点击**绑定证书**。在弹出的**绑定证书**窗口，选择证书，然后点击**绑定证书**。

4. 启用 WebUI 双因素认证功能。

   在**系统>用户管理>系统管理员>WebUI 双因素认证**页面（见上图）将 **WebUI 双因素认证**滑块置为**启用**。启用该功能后，指定用户在下次登录 WebUI 时，将必须使用 USB Key 通过双因素认证身份信息。

注意：如果是 UOS 系统用户，需要在浏览器中完成以下配置：

- 在浏览器中导入 rootca 证书。
- 启用 TSL1.0 和 TSL1.1。

➢ **用户通过 USB Key 访问 WebUI 流程**

完成上述配置并启用双因素认证后，用户访问 WebUI 流程如下：

1. 启动 USB Key 管理工具。
2. 将 USB Key 插入要访问 WebUI 的本地设备。
3. 打开 WebUI 登录界面，输入用户帐号密码和 USB Key 的 PIN 码，确认登录。

33.2.6 配置示例

33.2.6.1 创建系统管理员

执行如下命令添加管理员，并配置其访问控制级别：

user <*user_name*> <*password*> *[level]*

例如：

```
Demo(config)#user admin1 abcabc config
```

33.2.6.2 配置基于角色的权限管理

1. 执行如下命令添加角色：

```
role name <role_name>
```

例如：

```
Demo(config)#role name role1
```

2. 执行如下命令为角色设置命令行执行权限：

```
role deny <role_name> <filter_string>
role permit <role_name> <filter_string>
```

例如：

```
Demo(config)#role deny role1 "clear config"
Demo(config)#role permit role1 "show running config"
```

3. 执行如下命令为管理员账户分配角色：

```
role user <user_name> <role_name>
```

例如：

```
Demo(config)#role user admin1 role1
```

33.3 分区管理

分区管理功能实现了多用户环境下 SLB/SSL 业务隔离，一台物理设备可以同时服务多个租户，不同租户的用户管理系统和业务分发系统相互独立，这样可以有效降低服务的运维成本，提高资源利用率。

分区管理功能包括两种模式：

- 系统模式：在该模式下，管理员可以创建分区及分区管理员账户，指定系统接口加入分区，为分区配置相关网络。
- 分区模式：完成系统模式配置后，分区管理员可以登录相关分区进行分区配置和管理。

系统支持的分区总数取决于系统内存，最多可支持 1024 个分区，即一台物理设备可以同时为最多 1024 个租户提供相互独立的 SLB 和 SSL 业务。

33.3.1 分区管理员

分区管理员账户需要由系统管理员创建并关联到分区，每个分区需要单独分配至少一个分区管理员以及 IP 地址、接口、VLAN 等其他资源，每个分区内部只能使用分配的资源。一个分区管理员只能关联到一个分区，一个分区可以拥有多个分区管理员。所有分区的管理员总数最多不超过 2048 个。

分区管理员账户通过“**segment user**”命令创建，创建时可以同时配置账户密码和访问级别：

- enable 级别：可以运行分区内所有“enable”级别下的 CLI 命令。
- config 级别：可以运行分区内所有的全部 CLI 配置命令。
- api 级别：可以访问分区内基于 API 的 Web 服务，但是不能运行任何 CLI 命令。

通过创建的分区管理员账号登录设备即可进入该账号关联的分区进行配置和管理。分区管理员账号的密码可以在系统模式下通过“**segment passwd**”命令修改，也可以在分区模式下通过“**password**”命令修改。

注意：如果需要修改已分配的分区资源，例如分区名称、分区管理员、IP 地址等资源配置，需在修改后在分区和系统模式下都通过“**write memory**”命令进行配置保存。

33.3.2 分区支持的功能

分区拥有独立的用于配置 SLB 和 SSL 的 CLI 命令，所有分区支持的 CLI 命令相同。对于 config 级别的分区管理员账号，在 config 模式下输入“？”可查看分区内支持的所有 CLI 命令。

通过分区管理功能，每个分区可以配置自己的默认路由和静态路由。系统会基于每个分区配置的默认路由和静态路由生成 Eroute 路由，系统管理员可以通过“**show segment ip eroute**”命令查看每个分区的 Eroute 路由。

另外，通过配置 NAT 转换规则，分区管理功能允许分区之间的网段或 IP 地址重叠。系统管理员可以通过“**segment nat**”命令为每个分区配置 NAT 转换规则。

注意：

- 分区内的健康检查、ping 等非业务流量需要通过分区内的静态路由或直连路由转发。
- 除后台服务外，在分区内，系统不支持与 URL 地址为域名而不是 IP 地址的外部服务器协作，例如通过 FTP 服务器导入或备份文件、通过 LDAP 服务器下载 CRL 文件、向邮件服务器发送告警邮件、通过证书里的 URL 地址进行证书校验

等。

33.4 文件系统救援

文件系统救援只能解决因为文件系统的损坏而引起的问题，不能解决因为磁盘损坏而引起的问题。当前分区无法启动时，可以切换到对端分区，在对端分区执行救援命令来恢复当前分区。文件系统救援分为两种模式：修复和重建模式。建议先执行修复模式，如果修复不成功，再执行重建模式。

- 修复模式：检测并修复文件系统受损的分区。
- 重建模式：格式化文件系统受损的分区并构建目录结构。如果文件系统受损的分区的数据极其重要，慎重执行此模式。

文件系统救援具体执行步骤如下：

1. 切换到正常分区；
2. 执行“**system rescue file backup**”命令对文件系统受损的分区的文件进行备份（视文件系统损坏程度而异，执行该命令备份文件不一定成功）；
3. 执行“**system rescue partition** *repair*”命令，修复成功，则切换到原分区尝试进入系统；如果仍旧无法进入系统或者修复失败，则继续执行以下步骤；
4. 执行“**system rescue partition** *rebuild*”命令，格式化文件系统受损的分区；
5. 步骤 4 执行成功后，执行“**system update**”命令升级到原来的版本，系统启动后进入原分区系统；
6. 如需恢复之前的配置，执行“**system rescue file restore**”命令恢复之前的文件。

注意：两种救援模式都有丢失数据的风险，实施文件系统救援前，请先联系公司技术支持。

第34章 一键巡检

34.1 概述

一键巡检功能旨在巡检设备的运行情况，在设备出现异常情况时提示管理员尽快采取相应措施。配置一键巡检功能后，当指定巡检项达到为其设置的指定的阈值后，系统在一键巡检任务结束时会生成巡检报告，用于管理员查看设备各巡检项运行状态。

系统支持的巡检项包括硬件基本信息、硬件性能、软件基本信息、安全基线配置、网关监控配置、冗余架构情况、配置容量情况和业务状态等。

根据巡检内容的不同，一键巡检功能支持的巡检类型包括日常巡检（初级）、应急巡检（中级）和深度巡检（高级）。其中日常巡检的巡检项最少，深度巡检的巡检项最多。管理员可以根据需要配置合适的巡检类型。

34.2 阈值和风险提示

一键巡检功能通过为巡检项设置阈值等级来提示管理员采取相应的措施。阈值等级分为：

- 第一阈值：超过该值时表示该设备或该服务需要被关注。
- 第二阈值：超过该值时需要人为采取措施。

一键巡检功能支持的风险级别包括：

- 正常（normal）：表示该巡检项正常。
- 优化（optimization）：表示该巡检项的配置需要优化，例如没有配置 NTP 服务器。
- 一般（info）：表示该巡检项可能提示风险，例如不是永久许可证，某些版本可能会存在漏洞等。
- 关注（notice）：表示该巡检项需要引起注意，例如 CPU、内存等资源使用率超过第一阈值。
- 严重（critical）：表示该巡检项需要人为采取措施，例如设备上生成加速卡或电源风扇（虚拟机取不到的话显示 N/A）等的告警日志，或者设备上 CPU 或内存等资源使用率超过第二阈值。
- 致命（error）：表示该巡检项存在严重问题，例如设备上进程状态异常。

34.3 配置示例

34.3.1 日常巡检

在 WebUI 上，选择**监控中心**>**一键巡检**，点击**日常巡检**，即可进行日常巡检。

34.3.2 应急巡检

在 WebUI 上，选择**监控中心**>**一键巡检**，点击**应急巡检**，即可进行应急巡检。

34.3.3 深度巡检

在 WebUI 上，选择**监控中心**>**一键巡检**，点击**深度巡检**，即可进行深度巡检。

34.4 查看巡检报告

在 WebUI 上，选择**监控中心**>**一键巡检**，点击要查看的巡检结果操作栏中的**详情**按钮，即可查看巡检报告详细信息。

34.5 下载巡检报告

在 WebUI 上，选择**监控中心**>**一键巡检**，点击要下载的巡检结果操作栏中的**详情**按钮，在巡检报告详细信息页面点击下载 PDF 巡检报告。

34.6 删除巡检报告

在 WebUI 上，选择**监控中心**>**一键巡检**，点击要删除的巡检结果操作栏中的**删除**按钮，即可删除巡检报告。

附录I 缩略语

缩写	全称	中文
AAA	Authentication, Authorization & Accounting	验证、授权和统计
ACL	Access Control List	访问控制列表
ADC	Application Delivery Controller	应用交付控制器
API	Application Programming Interface	应用程序编程接口
ARP	Address Resolution Protocol	地址解析协议
ASCII	American Standard Code for Information Interchange	美国信息交换标准码
ASN.1	Abstract Syntax Notation One	抽象语法标记
ATCP	TCP	TCP
BGP	Border Gateway Protocol	边界网关协议
CA	Certificate Authority	认证中心
CDN	Content Distribution Network	内容分发网络
CDP	CRL Distribution Point	CRL 分发点
CGI	Common Gateway Interface	公共网关接口
CLI	Command Line Interface	命令行接口
CNAME	Canonical Name	别名
CPS	Connections Per Second	每秒连接数
CPU	Central Processing Unit	中央处理器
CRC	Cyclic Redundancy Check	循环冗余校验
CRL	Certificate Revocation List	证书撤销列表
CRS	Core Rule Set	核心规则集
CSR	Certificate Signing Request	证书签发请求
DMZ	DeMilitarized Zone	隔离区
DNS	Domain Name System	域名服务系统
DoS	Denial Of Service	拒绝服务攻击
DPS	Dynamic Proximity System	动态邻接系统
FFO	Fast Failover	快速失效切换
FIFO	First-In First-Out	先入先出(法)
FTP	File Transfer Protocol	文件传输协议
FTPS	FTP over SSL	安全文件传输协议
GMT	Greenwich Mean Time	格林威治标准时间
GRE	Generic Routing Encapsulation	通用路由封装
GSLB	Global Server Load Balance (also known as SDNS)	全局服务器负载均衡
HA	High Availability	高可用性
HC	Health Check	健康检查
HTML	HyperText Markup Language	超文本标记语言
HTTP	HyperText Transfer Protocol	超文本传输协议

缩写	全称	中文
HTTPS	HyperText Transfer Protocol over Secure Sockets Layer	安全超文本传输协议
ICMP	Internet Control Message Protocol	因特网控制报文协议
ICMPv6	Internet Control Message Protocol version 6	因特网控制报文协议第六版
IEEE	Institute of Electrical and Electronics Engineers	美国电气和电子工程师学会
IETF	Internet Engineering Task Force	因特网工程任务组
IGMP	Internet Group Management Protocol	Internet 组管理协议
IIS	Internet Information Server	因特网信息服务器
IMS	Information Management System	信息管理系统
IP	Internet Protocol	因特网协议
IPMI	Intelligent Platform Management Interface	智能平台管理接口
ISP	Internet Service Provider	因特网服务供货商
LACP	Link Aggregation Control Protocol	链路汇聚控制协议
LAN	Local Area Network	局域网
LDAP	Lightweight Directory Access Protocol	轻量级目录访问协议
LED	Light Emitting Diode	发光二极管显示器
LLB	Link Load Balancing	链路负载均衡
Local DNS	Local Domain Name System	本地域名服务
MAC	Media Access Control	媒体访问控制
MIB	Management Information Base	管理信息库
MIME	Multipurpose Internet Mail Extensions	多用途因特网邮件扩展
MNET	Multi-Netting	多地址端口
MTU	Maximum Transmission Unit	最大传输单元
NAT	Network Address Translation	网络地址转换
NDP	Neighbor Discovery Protocol	邻居发现协议
NIC	Network Interface Card	网络适配器
NMS	Network Management Station	网络管理站
NTP	Network Time Protocol	网络时间协议
NUMA	Non-uniform Memory Access	非一致性内存访问
OCSP	Online Certificate Status Protocol	在线证书状态协议
OID	Object Identifier	对象标识符
OSI	Open System Interconnection	开放式系统互连模型
OSPF	Open Shortest Path First	开放式最短路径优先协议
OSPFv2	Open Shortest Path First version 2	开放式最短路径优先协议第二版
OSPFv3	Open Shortest Path First version 3	开放式最短路径优先协议第三版
OWA	Outlook Web Access	基于微软 Hosted Exchange 技术的托管邮局的一项 Web 访问功能
PCI	Peripheral Component Interface	外围组件接口

缩写	全称	中文
PEM	Privacy Enhanced Mail	增强保密的邮件
PHY	Physical Layer	物理层
PKI	Public Key Infrastructure	公钥基础架构
PLR	Packet Loss Rate	丢包率
POP3	Post Office Protocol - Version 3	邮局协议-版本 3
PPP	Point-to-Point Protocol	点到点协议
PPTP	Point-to-Point Tunneling Protocol	点到点隧道协议
PST	Pacific Standard Time	太平洋标准时间
QoS	Quality of Service	服务质量
RADIUS	Remote Authentication Dial-In User Service	远程用户拨入认证系统
RAM	Random Access Memory	随机存取内存
RDP	Remote Desktop Protocol	远程桌面协议
RFC	Request For Comments	请求注解
RHI	Route Health Injection	路由健康注入
RIP	Routing Information Protocol	路由信息协议
RIPv1	Routing Information Protocol version 1	路由信息协议第一版
RIPv2	Routing Information Protocol version 2	路由信息协议第二版
RIPng	RIP next generation	下一代路由选择协议
RPS	Requests Per Second	每秒请求数
RTS	Return to Sender	原链路返回
RTSP	Real Time Streaming Protocol	实时流媒体协议
RTT	Round Trip Time	往返时间
SCP	Session Control Protocol	会话控制协议
SDNS	Smart DNS (also known as GSLB)	智能 DNS 解析
SDR	Sensor Data Records	传感器数据记录
SEL	System Event Log	系统事件日志
SIP	Session Initiation Protocol	会话初始化协定
SLB	Server Load Balancing	服务器负载均衡
SMTP	Simple Mail Transfer Protocol	简单邮件传输协议
SNI	Server Name Indication	服务器名字指示
SNMP	Simple Network Management Protocol	简单网络管理协议
SOAP	Simple Object Access Protocol	简单对象访问协议
SQL	Structured Query Language	结构化查询语言
SSF	Stateful Session Failover	链接同步
SSH	Secure Shell Protocol	安全外壳协议
SSI	Single System Image	单系统映像
SSL	Secure Sockets Layer	安全套接字层
SSLv3	Secure Sockets Layer version 3	安全套接字层第三版
TACACS	Terminal Access Controller Access Control System	终端访问控制器访问控制系统
TCI	Tag Control Information	标记控制信息
TCL	Tools Command Language	工具命令语言

缩写	全称	中文
TCP	Transmission Control Protocol	传输控制协议
TCPS	TCP with SSL	安全 TCP
TELNET	Terminal Emulation Protocol in a TCP/IP Environment	TCP/IP 终端模拟协议
TFTP	Trivial File Transfer Protocol	简单文件传输协议
TLS	Transport Layer Security Protocol	传输层安全协议
TPID	Tag Protocol Identifier	标签协议标识
TTL	Time to Live	生存时间
UDP	User Datagram Protocol	用户数据报协议
URL	Uniform Resource Locator	统一资源定位符
VCID	Virtual Cluster ID	虚拟集群标识
VIP	Virtual IP	虚拟 IP 地址
VLAN	Virtual Local Area Network	虚拟局域网
VOD	Video On Demand	视频点播
VoIP	Voice over Internet Protocol	基于 IP 的语音传输
VRRP	Virtual Router Redundancy Protocol	虚拟路由冗余协议
VXLAN	Virtual eXtensible Local Area Network	虚拟扩展局域网
VNI	VXLAN Network Identifier	VXLAN 网络标识符
VTEP	VXLAN Tunnel End Point	VXLAN 隧道端点
WebUI	Web User Interface	Web 用户接口
WELF	WebTrends Enhanced Log Format	国际通行的防火墙日志规范格式
XSS	Cross Site Scripting	跨站脚本攻击

附录II XML RPC 方法

一般 XML RPC 方法				
方法名称	**命令行**	**{参数名称，参数类型}**	**可选参数**	**注意事项**
os_cli_enable	所有 Enable 模式下的命令	{num,int}, {cli_string0,string}, {cli_string1,string}, {cli_string2,string}, {cli_string3, string} ……	num	如果没有设置参数“num”值，那么其默认值为 1，则必须配置“cli_string0”。CLI 命令的名称必须从“cli_string0”到“cli_string{n-1}”结束。如果中间的CLI命令丢失，XML RPC 系统将忽略这个问题，并且不会报错。
os_cli_config	所有 Config 模式下的命令			
os_cli_config_with_input	所有 Config 模式下的命令	{cli_string, string}, {num, int}, {input_string0, string}, {input_string1, string} ...	num, input_string0, input_string1, ...	如果要调用交互式 CLI 命令（例如输入“YES”来继续执行命令），则必须使用该方法。本方法一次只能执行一条 CLI 命令。如果没有设置参数“num”则其默认值为 1。参数“input_string”从“input_string0”开始到“input_string(num-1)”，如果中间丢失“input_string(n)”，则其默认值为空。如果 CLI 要求在此处必须输入一个有效的值，那么此次调用可能挂起或返回错误。本方法一次只能执行一条 CLI 命令。