

Switching 技术笔记

红茶三杯 CCIE 学习文档

文档版本： 2.0

更新时间： 2013-04-26

文档作者： 红茶三杯

文档地址： <http://ccietea.com>

文档备注： 请关注文档版本及更新时间

1 以太网

1.1 基础知识

1. About Ethernet

Ethernet(以太网)于 20 世纪 70 年代中期,由 Xerox 公司分部 Palo Alto 研究中心(PARC)开发的。Xerox 最早发明的是一个 2Mbps 的以太网,后来又和 Intel 和 DEC 合作开发了出了 10Mbps 的以太网,俗称 (Ethernet II 或 Ethernet DIX).后来 IEEE 通过 802 委员会(802 Committee)把 Ethernet 标准化为 IEEE 802.3。它和 Ethernet II 十分相似。

在 TCP/IP 中 ,以太网的 IP 数据报文的封装格式由 RFC 894 定义 ,IEEE802.3 网络的 IP 数据报文封装由 RFC 1042 定义。当今最常使用的封装格式是 RFC894 定义的格式 , 通常称为 Ethernet II 或者 Ethernet DIX。

2. 管理 MAC 表

show mac address-table

clear mac address-table

绑定一个 mac 地址到一个接口

Switch(config)# mac address-table static 机器的 mac 接口 vlan vlan 号

要取消用 no mac address-static

1.2 以太网的数据链路层

在以太网中，针对不同的双工模式，提供不同的介质访问方法：

- 在半双工模式下采用的是 CSMA/CD 的访问方式。
- 而在全双工模式下则可以直接进行收发，不用预先判断链路的忙闲状态。

半双工和全双工是物理层的概念，而针对物理层的双工模式提供不同访问方式则是数据链路层的概念，这样就形成了以太网的一个重要特点：数据链路层和物理层是相关的。

由于以太网的物理层和数据链路层是相关的，针对物理层的不同工作模式，需要提供特定的数据链路层来访问。这给设计和应用带来了一些不便。

为此，一些组织和厂家提出把数据链路层再进行分层，分为逻辑链路控制子层（LLC）和媒体访问控制子层（MAC）。这样不同的物理层对应不同的 MAC 子层，LLC 子层则可以完全独立。如图 1-4 所示。

1. MAC 子层

MAC 子层负责如下任务：

- 提供物理链路的访问。
- 链路级的站点标识：在数据链路层识别网络上的各个站点。
- 也就是说，在该层次保留了一个站点地址，即 MAC 地址，来标识网络上的唯一——一个站点。
- 链路级的数据传输：从 LLC 子层接收数据，附加上 MAC 地址和控制信息后把数据发送到物理链路上；在这个过程中提供校验等功能。

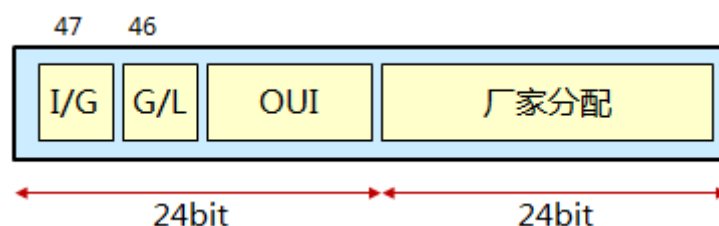
MAC 子层是物理层相关的，也就是说，不同的物理层有不同的 MAC 子层来进行访问。在以太网中，主要存在两种 MAC：

- 半双工 MAC：物理层运行模式是半双工时提供访问。
- 全双工 MAC：物理层运行模式是全双工时提供访问。

这两种 MAC 都集成在网卡中，网卡初始化的时候一般进行自动协商，根据自动协商的结果决定运行模式，然后根据运行模式选择相应的访问 MAC。

MAC 地址

MAC 地址是烧录在网卡（Network Interface Controller, NIC）的 ROM 里的



高位是 individual/group 位，当它的值为 0 时，就可以认为这个地址实际上是设备的 MAC 地址。当它的值为 1 时，就可以认为这个地址表示以太网中的广播地址或组播地址，或者表示 TR 和 FDDI 中的广播地址或功能地址。下一位是 G/L 位（也称为 U/L,这里的 U 表示全局）。当这一位设置为 0 时，就表示一个全局管理地址（由 IEEE 分配），当这一位为 1 时，就表示一个在管理上局部本地的地址（就像在 DECnet 中一样）。以太网一直使用全局唯一地址。

2. 以太网帧格式

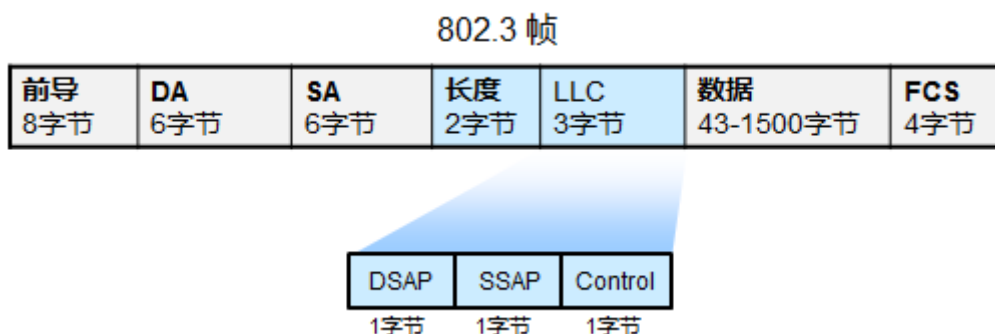
Ethernet II 帧

前导 8字节	DA 6字节	SA 6字节	类型 2字节	数据 46-1500字节	FCS 4字节
-----------	-----------	-----------	-----------	-----------------	------------

前导 (Preamble)	包括 7 个字节的前导码（一串 1、0 间隔，用于信号同步）及 1 个字节 的帧起始定界符（10101011）
类型 (Type)	802.3 使用长度字段，但 Ethernet 帧使用类型字段来识别网络层的协议。 在 EthernetII 帧中，两字节的类型字段用于标识数据字段中包含的高层协 议，也就是说，该字段告诉接收设备如何解释数据字段。 在以太网中，多种协议可以在局域网中同时共存。因此，在 Ethernet II 的 类型字段中设置相应的十六进制值提供了在局域网中支持多协议传输的机 制。 <ul style="list-style-type: none"> 类型字段取值为 0800 的帧代表 IP 协议帧。 类型字段取值为 0806 的帧代表 ARP 协议帧。 类型字段取值为 0835 的帧代表 RARP 协议帧。 类型字段取值为 8137 的帧代表 IPX 和 SPX 传输协议帧。 802.3 不能识别上层协议，且必须与专用的 LAN(比如 IPX)一起使用。
数据 (DATA)	46-1500 字节；在接口下设置 mtu xxxx，指的就是这个，并且一般不允 许手动修改。Ip mtu 指的是 ip 报文的最大值
帧校验序列 FCS (Frame check sequence)	存储 CRC 循环冗余校验的结果

PS：在以太网中，由于冲突的存在，共享介质上两台主机同时发 frame，将产生冲突。根据特定的算法，以太网中，frame 的最小长度为 64 字节。

PS：目前我们所使用到的以太网帧基本都是 Ethernet II 帧



Length	Length 字段定义了 Data 字段包含的字节数。
LLC	LLC(Logical Link Control)由目的服务访问点 DSAP(Destination Service Access Point)、源服务访问点 SSAP (Source Service Access Point) 和 Control 字段组成。

IEEE802.3 帧根据 DSAP 和 SSAP 字段的取值又可分为以下几类：

- 当 DSAP 和 SSAP 都取特定值 0xff 时，802.3 帧就变成了 Netware-ETHERNET 帧，用来承载 NetWare 类型的数据。
- 当 DSAP 和 SSAP 都取特定值 0xaa 时，802.3 帧就变成了 ETHERNET_SNAP 帧。
ETHERNET_SNAP 帧可以用于传输多种协议。因此，SNAP 可以被看作一种扩展，它允许厂商创建自己的以太网传输协议。
ETHERNET_SNAP 标准由 IEEE802.1 委员会制定 以保证 IEEE802.3 局域网和以太网之间的互操作性。
- DSAP 和 SSAP 其他的取值均为纯 IEEE802.3 帧。

3. LLC 子层

在前文的介绍中提到了 MAC 子层形成的帧结构，包括 IEEE802.3 的帧和 ETHERNET_II 帧。

在 ETHERNET_II 帧中，由 Type 字段区分上层协议，这时候就没有必要实现 LLC 子层，仅包含一个 MAC 子层。而 IEEE802.3 帧中的 LLC 子层除了定义传统的链路层服务之外，还增加了一些其他有用的特性。这些特性都由 DSAP、SSAP 和 Control 字段提供。

例如以下三种类型的点到点传输服务：

- **无连接的数据包传输服务**
目前的以太网实现就是这种服务。
- **面向连接的可靠的数据传输服务**
预先建立连接再传输数据，数据在传输过程中可靠性得到保证。
- **无连接的带确认的数据传输服务。**
该类型的数据传输服务不需要建立连接，但它在数据的传输中增加了确认机制，使可靠性大大增加。

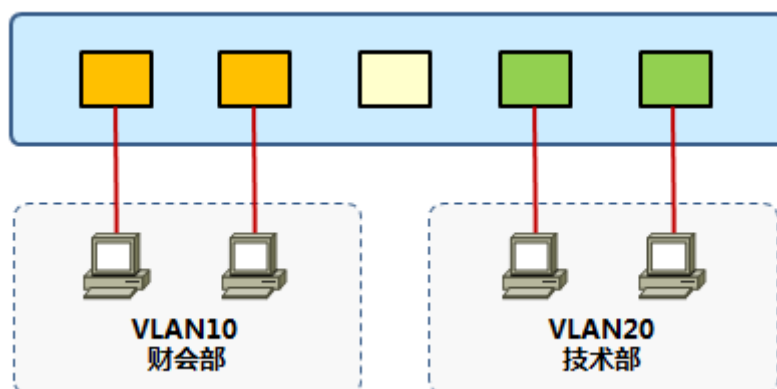
下面通过一个例子来说明 SSAP 和 DSAP 的应用。假设终端系统 A 和终端系统 B 要使用面向连接的可靠的数据传输服务，这时候会发生如下过程：

1. A 给 B 发送一个数据帧，请求建立一个面向连接的可靠连接。
2. B 接收到以后，判断自己的资源是否够用（即是否建立了太多的连接），如果够用，则返回一个确认信息，该确认信息中包含了识别该连接的 SAP 值。
3. A 接收到回应后，知道 B 已经在本地建立了跟自己的连接。A 也开辟一个 SAP 值，来表示该连接，并发一个确认给 B，连接建立。
4. A 的 LLC 子层把自己要传送的数据进行封装，其中 DSAP 字节填写的是 B 返回的 SAP，SSAP 字节填写的是自己开辟的 SAP，然后发给 MAC 子层。
5. A 的 MAC 子层加上 MAC 地址和 LENGTH 字段之后，发送到数据链路上。
6. B 的 MAC 子层接收到该数据帧之后，提交给 LLC 子层，LLC 子层根据 DSAP 字段判断出该数据帧属于的连接。
7. B 根据该连接的类型进行相应的校验和确认，通过这些校验和确认后，才向上层发送。
8. 数据传输完毕之后，A 给 B 发送一个数据帧来告诉 B 拆除连接，通信结束。

2 二层交换

2.1 VLAN

2.1.1 VLAN 基本概念



- 一个 VLAN 中所有设备都是在同一广播域内，不同的 VLAN 为不同的广播域
- VLAN 之间互相隔离，广播不能跨越 VLAN 传播，因此不同 VLAN 之间的设备一般无法互访，不同 VLAN 间需通过三层设备实现相互通信
- 一个 VLAN 一般为一个逻辑子网，由被配置为此 VLAN 成员的设备组成
- VLAN 中成员多基于交换机的端口分配，划分 VLAN 就是对交换机的接口划分
- VLAN 工作于 OSI 参考模型的第二层
- VLAN 是二层交换机的一个非常根本的工作机制

2.1.2 VLAN 基本通信原理

为了提高处理效率，交换机内部的数据帧一律都带有 VLAN Tag，以统一方式处理。当一个数据帧进入交换机端口时，如果没有带 VLAN Tag，且该端口上配置了 PVID(Port VLAN ID)，那么，该数据帧就会被标记上端口的 PVID。如果数据帧已经带有 VLAN Tag，那么，即使端口已经配置了 PVID，交换机不会再给数据帧标记 VLAN Tag。

PVID 是“端口缺省 VLAN ID”的意思，即一个端口缺省属于的 VLAN。

由于端口类型不同，交换机对帧的处理过程也不同。下面根据不同的端口类型分别介绍。

1. Access 端口处理帧的过程

Access 端口处理 VLAN 帧的过程如下：

- 1) 收到一个二层帧。
- 2) 判断帧是否有 VLAN Tag。
 - 没有 Tag，则标记上 Access 端口的 PVID，进行下一步处理。
 - 有 Tag，则比较帧的 VLAN Tag 和端口的 PVID，两者一致则进行下一步处理；否则丢弃帧。
- 3) 二层交换机根据帧的目的 MAC 地址和 VLAN ID 查找 VLAN 配置信息，决定从哪个端口把帧发送出去。
- 4) 交换机根据查到的出接口发送数据帧。
 - 当数据帧从 Access 端口发出时，交换机先剥离帧的 VLAN Tag，然后再发送出去。
 - 当数据帧从 Trunk 端口发出时，直接发送帧。
 - 当数据帧从 Hybrid 端口发出时，交换机判断 VLAN 在本端口的属性是 Untag 还是 Tag。如果是 Untag，先剥离帧的 VLAN Tag，再发送；如果是 Tag，直接发送帧。

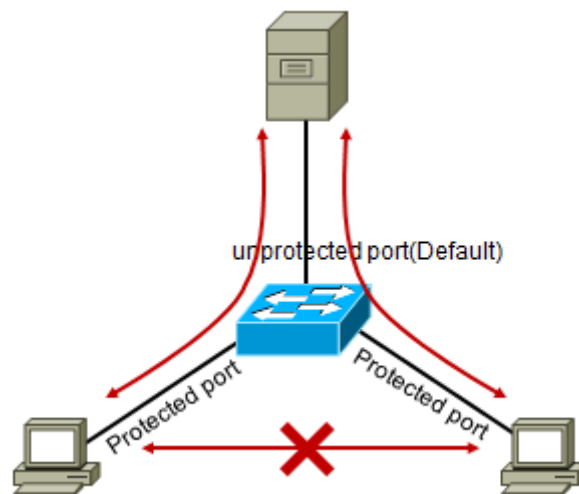
2. Trunk 端口处理帧的过程

Trunk 端口处理 VLAN 帧的过程如下：

- 1) 收到一个二层帧。

- 2) 判断帧是否有 VLAN Tag。
 - 没有 Tag，则标记上 Trunk 端口的 PVID，进行下一步处理。
 - 有 Tag，则判断该 Trunk 端口是否允许该 VLAN 帧进入。允许则进行下一步处理，否则丢弃帧。
- 3) 二层交换机根据帧的目的 MAC 地址和 VLAN ID，查找 VLAN 配置信息，决定从哪个端口把帧发送出去。
- 4) 交换机根据查到的出接口发送数据帧。
 - 当数据帧从 Access 端口发出时，交换机先剥离帧的 VLAN Tag，然后再发送出去。
 - 当数据帧从 Trunk 端口发出时，直接发送帧。
 - 当数据帧从 Hybrid 端口发出时，交换机判断 VLAN 在本端口的属性是 Untag 还是 Tag。如果是 Untag，先剥离帧的 VLAN Tag，再发送；如果是 Tag，直接发送帧。

2.1.3 Protected port



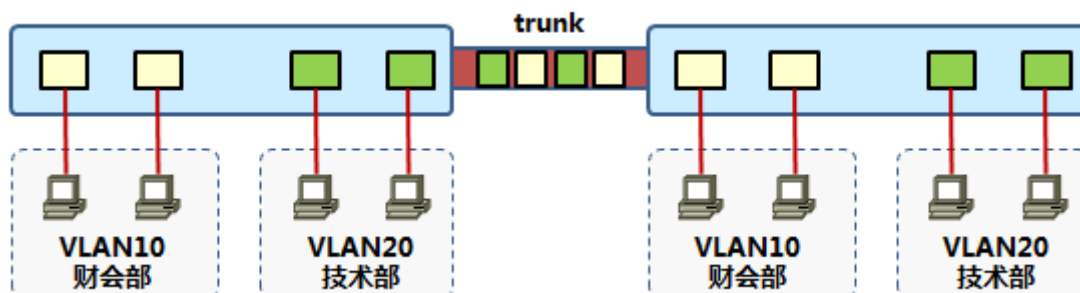
- Protected port 虽然同处一个 VLAN，但是彼此无法互相通信
- Protected port 只能与 unprotected port（默认）互相通信
- Protected port 特性无法跨交换机实现

配置非常简单：

```
Switch(config-if)# switchport protected
```

2.2 TRUNK

2.2.1 Trunk 概述



- 当一条链路，需要承载多 VLAN 信息的时候，需使用 trunk 来实现
- Trunk 两端的交换机需采用相同的干道协议
- 一般见于交换机之间或交换机与路由器、服务器之间

2.2.2 封装协议

ISL	802.1Q
CISCO私有	公有标准
采用封装的方式	采用tag的方式
在原始帧的基础上封装上新帧头及新的 FCS	在原始的帧头中插入tag字段，去掉原有 FCS，重新计算FCS

1. baby giant frame 大于标准的 MTU1500 字节，但是小于 2000 字节

对于采用 ISL 封装的，MTU=1548 (下文有解释)

对于 Dot1Q MTU=1522

2. Vlan 范围和映射

ISL 支持的 vlan 编号是 1-1005 (默认允许正常的)，802.1q 是 1-4094 (默认允许所有正常和扩展的) 所以当穿过 802.1q 和 ISL 的干道的时候就需要映射。

- (1) 单台交换机上最多允许 8 个 802.1q 到 ISL vlan 的映射
- (2) 只能映射到 ethernet 的 vlan
- (3) 该被映射的 ethernet vlan 将被阻塞

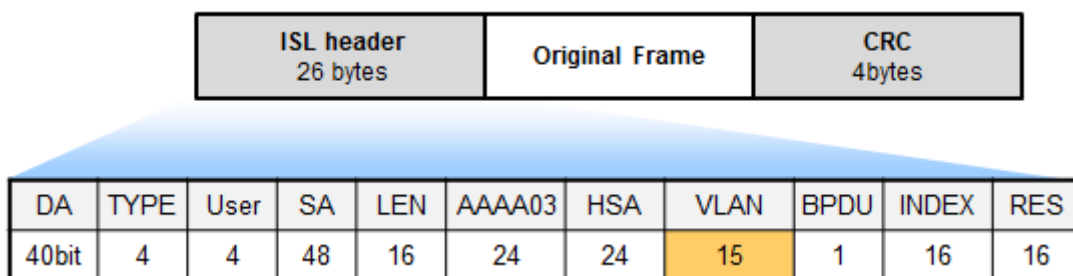
(4) 映射仅在本地有效

3. 链路聚集模式

- trunk 永久链路聚集模式，强制 trunk，发送 DTP 帧
- Nonegotiate 永久链路聚集模式，必须手动将邻居配为干道口，不发送 DTP 帧。一般用于对端设备不支持 DTP 的情况
- Desirable 主动尝试将链路成为干道（默认模式），发送 DTP 帧，如果邻接接口为 trunk、desirable、或 AUTO，那么此接口成为 Trunk。
- Auto 接口愿意成为 trunk，如果邻接接口被设置为 trunk 或 desirable，那么接口就成为 trunk
- Access 永久的 nontrunking 模式，并且与对端接口协商，使其成为 nontrunking 链路

	Dynamic Auto	Dynamic Desirable	Trunk	Access
Dynamic Auto	Access	Trunk	Trunk	Access
Dynamic Desirable	Trunk	Trunk	Trunk	Access
Trunk	Trunk	Trunk	Trunk	不建议
Access	Access	Access	不建议	Access

2.2.3 ISL



- CISCO 私有协议
- 支持 PVST
- 在原始的数据帧基础上封装上 ISL 头及新的 FCS
- 没有修改原始的数据帧，因此处理效率比 802.1Q 高
- VLAN 字段，15 个比特目前用了 10 个，那么最多支持 $2^{10} = 1024$ 个 VLAN

- “原始以太网帧”最大是 1518 个字节,1500 的 IP MTU 加上源目的 MAC 地址共 12 类型字段 2 个, CRC4 , 再加上 30 字节的 ISL 封装, 就是 1548 字节了

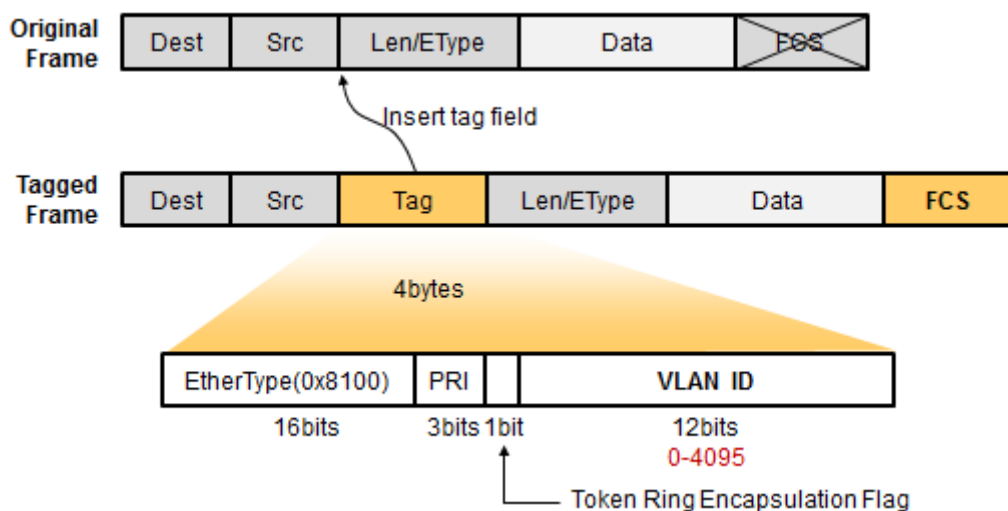
上图中几个字段 (ISL 头) 的描述如下 :

- DA 40bit 的组播地址用于标示这个 FRAME 是 ISL 的
- TYPE 标示这个帧是什么类型的, 如以太、令牌环等
- SA 发送帧的原交换机 MAC
- AAAA03 SNAP (固定值)
- VLAN 15 个比特目前用了 10 个, 那么最多支持 $2^{10}=1024$ 个 VLAN
- INDEX 这个帧的对端交换机来源端口

所以 ISL 帧最大 1548bytes (1518+26+4)

2.2.4 Dot1q

1. 帧格式



802.1Q Tag 包含 4 个字段, 其含义如下 :

- EtherType**
长度为 2 字节, 表示帧类型。取值为 0x8100 时表示 802.1Q Tag 帧。如果不支持 802.1Q 的设备收到这样的帧, 会将其丢弃。
- PRI**
Priority, 长度为 3 比特, 表示帧的优先级, 取值范围为 0~7, 值越大优先级越高。

用于当交换机阻塞时，优先发送优先级高的数据包。

- **CFI**

Canonical Format Indicator，长度为 1 比特，表示 MAC 地址是否是经典格式。

CFI 为 0 说明是经典格式，

CFI 为 1 表示为非经典格式。用于区分以太网帧、FDDI (Fiber Distributed Digital Interface) 帧和令牌环网帧。

在以太网中，CFI 的值为 0。

- **VID**

VLAN ID，长度为 12 比特，表示该帧所属的 VLAN。在 VRP 中，可配置的 VLAN ID 取值范围为 1 ~ 4094。

2. 优缺点

缺点是破坏了原始以太帧以及重新计算 FCS，ISL 是直接封装头和尾。DOT1q 公用，ISL 私有

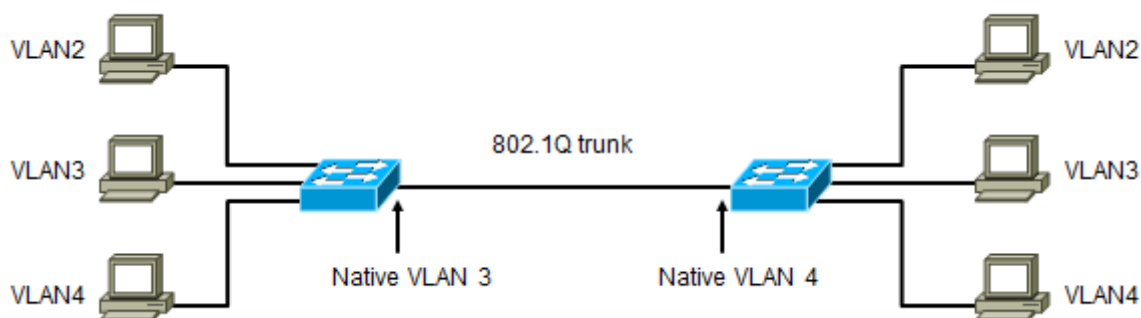
802.1Q 支持 4096 个 VLAN

最大帧：1518+4=1522

3. Native Vlan

在 802.1q 的 native vlan 是不打标签的，使用 Dot1q 的交换机把所有未被标记的 frame 转发到 native vlan 中，而 ISL 会对所有的数据帧，包括 native vlan 进行封装，因此如果收到没有封装的数据帧它会丢弃（ISL 没有 native VLAN 的概念）。

- Native VLAN 所属的帧在经过 trunk 时不打标签
- Native VLAN 在 Trunk 两端必须匹配，否则会出现 VLAN 流量互串
- 默认的 native vlan 是 vlan 1
- 建议将一个生僻的 VLAN 配置为 Native vlan



我们看上面这个图，两台交换机 trunk 两端 native vlan 不一样，会有什么问题？首先两端的 vlan2 通信肯定是没有问题的，但是 vlan3 和 vlan4 通信就有问题了，左边 vlan3 的用户发出来的数据帧从左交换机出去上 trunk，是不打标签的，但是这些数据帧到了右交换机，它会认为这些数据帧是属于 vlan4 的，这就出现问题了。

```
Switch(config-if)# switchport trunk native vlan ?
```

在 trunk 上设置 native vlan

```
Switch(config)# vlan dot1q tag native
```

上述命令将对 native vlan 也打标签

4. Vlan 范围

VLAN范围	作用
0, 4095	保留, 系统使用
1	Cisco默认vlan
2-1001	For Ethernet VLANs
1002-1005	Cisco默认为FDDI及TokenRing定义
1006-4094	只能为Ethernet使用, 在一些特殊平台被保留使用

2.2.5 DTP

- Trunk 可以手工静态配置或者通过 DTP 进行协商
- DTP 使得交换机之间能够进行 trunk 协商

trunk	永久链路聚集模式, 强制trunk, 发送DTP帧
Nonegotiate	不发送DTP帧。一般用于对端设备不支持DTP的情况; 需搭配手工指定的trunk或access模式命令
Desirable	主动尝试将链路成为干道 (默认模式), 发送DTP帧, 如果邻接接口为trunk、desirable、或AUTO, 那么此接口成为Trunk。
Auto	接口有意愿成为trunk, 但是不会主动发送DTP。如果邻接接口被设置为trunk或desirable, 那么接口就成为trunk, 会被动响应DTP协商数据帧
Access	永久的nontrunking模式, 并且与对端接口协商, 使其成为nontrunking链路

	Dynamic Auto	Dynamic Desirable	Trunk	Access
Dynamic Auto	Access	Trunk	Trunk	Access
Dynamic Desirable	Trunk	Trunk	Trunk	Access
Trunk	Trunk	Trunk	Trunk	不建议
Access	Access	Access	不建议	Access

2.2.6 Trunk 配置

```
Switch(config-if)# switchport mode access
```

- 将接口设置为 access 模式

```
Switch(config-if)# switchport mode encapsulation {dot1q | ISL}
```

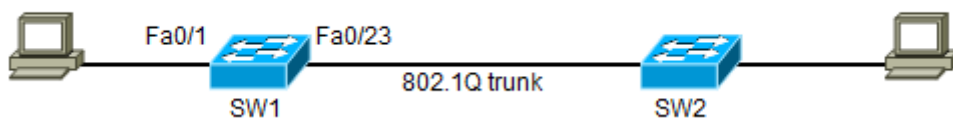
- 如果接口为 trunk，设置干道协议类型

```
Switch(config-if)# switchport mode dynamic {auto | desirable}
```

- 将接口设置为 DTP 动态协商，可选 auto 或 desirable

```
Switch(config-if)# switchport nonegotiate
```

- 将接口设置为 nonegotiate，不发送 DTP 帧，如果配置为非协商，那么就必须手工配置接口模式，为 access 或 trunk



```
SW1(config)# interface fast0/23
```

```
SW1(config-if)# switchport trunk encapsulation dot1q
```

```
SW1(config-if)# switchport mode trunk
```

```
SW1(config-if)# switchport native vlan 1
```

```
SW1(config-if)# switchport nonegotiate
```

SW1(config-if)# switchport trunk allowed vlan ?

WORD	VLAN IDs of the allowed VLANs when this port is in trunking mode
add	add VLANs to the current list
all	all VLANs
except	all VLANs except the following
none	no VLANs
remove	remove VLANs from the current list

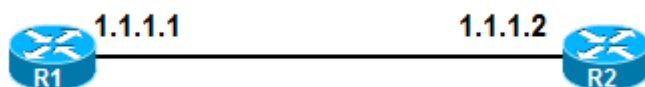
上图中,两端 PC 都属于 vlan10d 的话,如果 SW1 的 fa0/23 口将 vlan10remove 掉,那么 PC 肯定就无法通信了。

2.2.7 MTU 问题

MTU 就是最大传输单元,不同的系统对于 MTU 的设定和理解是不同的。

CISCO IOS 上, interface x 接口模式下:

- **MTU ?** 指的是二层的 MTU,这是接口 MTU,指的是不包含二层帧头的、Payload 的 MTU,这个 MTU 值一般是不能手工修改的,默认是 1500 字节。如此一来 CISCO 路由器支持的二层数据帧最大值就是 1500 的 payload 加上二层帧头及二层 FCS:目的 mac6 字节+源 mac6 字节+类型字段 2 字节+FCS4 字节,所以总的就是 1518 字节。
- **Ip mtu ?** 指的是三层的 MTU,这个值可以手工修改,但是最大值必须小于接口的二层 MTU 值也就是 1500。这个 MTU 指的是三层 IP 包的总大小,如果接口发出的包大于这个接口的 ip mtu,那么这个 IP 包将被分片



做个测试:上图中,R1 的 fa0/0 口 ip mtu 为 1500,

我们去 ping 1.1.1.2 repeat 1 size 1500,我们会发现 R1 直接将一个 ICMP 包发出去了,没有分片,报文如下:

```
Internet Protocol Version 4, Src: 1.1.1.1 (1.1.1.1), Dst: 1.1.1.2 (1.1.1.2)
  Version: 4
  Header length: 20 bytes
  + Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-E
  Total Length: 1500
  Identification: 0x0000 (0)
  + Flags: 0x00
  Fragment offset: 0
  Time to live: 255
  Protocol: ICMP (1)
  + Header checksum: 0xb21c [correct]
  Source: 1.1.1.1 (1.1.1.1)
  Destination: 1.1.1.2 (1.1.1.2)
Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0xeb07 [correct]
  Identifier (BE): 0 (0x0000)
  Identifier (LE): 0 (0x0000)
  Sequence number (BE): 0 (0x0000)
  Sequence number (LE): 0 (0x0000)
  [Response In: 5]
  + Data (1472 bytes)
```

从报文中我们可以看到，这个 IP 包的大小为 1500 字节。

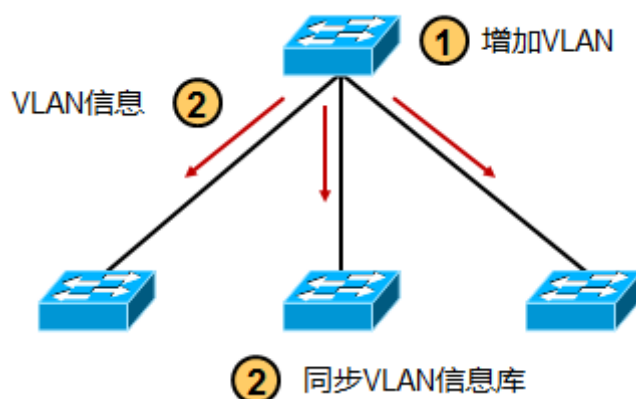
其中 IP 报头 20 字节，ICMP 报头 8 字节，ICMP data 荷载 1472 字节，刚好 1500 字节。

因此在 CISCO IOS 设备上，ping 后面跟着的 size 指的就是发出去的 IP 包整个的大小。

而在 windows PC 的 cmd 下，ping 后跟的包大小就是 ICMP data 大小，ping -l 1472，产生的包就是 1500 字节

还是上面的例子，如果我们在 R1 上，ping 1.1.1.2 repeat 1 size 1501，这个时候，R1 由于产生的这个 IP 包大于 mtu 1500，因此会被分片，然后在 R2 上，这两个分片被重组。

2.3 VTP



- 简化大型园区网中 VLAN 信息库同步的问题（同一个 VTP 管理域中）

- 只同步 VLAN 信息
- 需要交换机之间的 trunk 链路支持

1. VTP mode

Server	Client	Transparent
<ul style="list-style-type: none"> • 可创建vlan • 可修改vlan • 可删除vlan • 发送/转发信息宣告 • 同步 • VLAN信息存储于NVRAM • Catalyst交换机默认是server模式 	<ul style="list-style-type: none"> • 不能创建、修改、删除VLAN • 发送/转发信息宣告 • 同步 • VLAN信息不会存贮于NVRAM 	<ul style="list-style-type: none"> • 可创建vlan • 可修改vlan • 可删除vlan • 转发信息宣告 • 不同步 不始发 • VLAN信息存贮于NVRAM

Transparent 模式的 VTP 配置修订号始终为 0

2. VTP 的运作

- VTP 协议通过组播地址 0100-0CCC-CCCC 在 Trunk 链路上发送 VTP 通告；
- VTP Server 和 clients 通过最高的修订号来同步数据库；
- VTP 协议每隔 5 分钟发送一次 VTP 通告或者有变化时发生；

3. VTP 的配置

```
Switch(config)# vtp domain cisco
```

- **配置 VTP 域名**

```
Switch(config)# vtp mode {server | client | transparent}
```

- **配置本机 VTP 模式**

```
Switch(config)# vtp password x
```

- （可选）配置密码

```
Switch# show vtp password
```

4. VTP 的几个问题

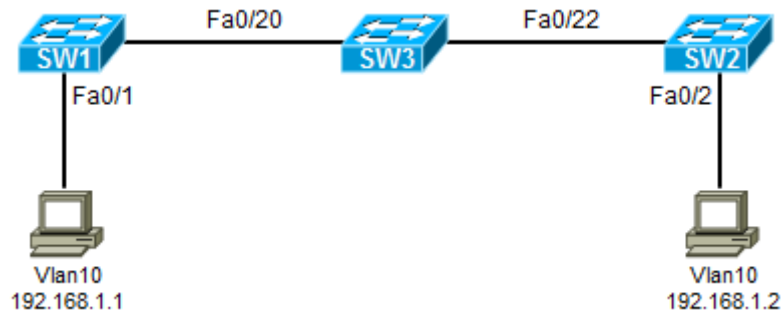


如果 VTP client 的配置修订号比 Server 的高，那么 client 也是能够将 server 的 vlan 信息覆盖掉的。



上图中，server 及 client 的配置修订号相同，但是 vlan 信息则不同，这时候就会报错，提示 md5 digest checksum mismatch

5. VTP pruning



1) 测试目的：VLAN 信息传递

SW1 及 SW2 配置为 VTP mode client；SW3 配置为 VTP mode Server

在 SW3 上创建 10、20、30 三个 VLAN

SW1 及 SW2 都能学习到，三岁小孩都能做，就不多说了

SW3#sh vtp status

VTP Version capable : 1 to 3

VTP version running : 1

VTP Domain Name : ccnp

VTP Pruning Mode : Disabled

!! 默认 VTP prunning 是关闭的

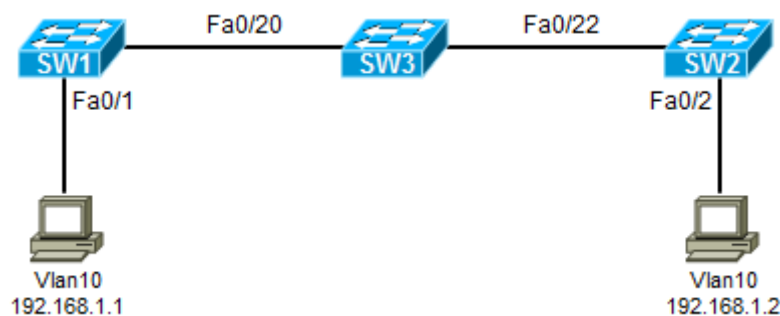
VTP Traps Generation : Disabled

```

Device ID                      : 000a.8a07.8280
Configuration last modified by 0.0.0.0 at 3-5-93 00:11:48
Local updater ID is 0.0.0.0 (no valid interface found)

Feature VLAN:
-----
VTP Operating Mode              : Server
Maximum VLANs supported locally : 1005
Number of existing VLANs       : 8
Configuration Revision          : 15
MD5 digest                     : 0x49 0x3C 0x1F 0x79 0x15 0x00 0xC7 0xAE
                                0x0E 0xDC 0xDD 0xEF 0x93 0xA5 0xB6 0x26
    
```

2) 测试目的：trunk allowed vlan



继续吧，将 SW1 及 SW2 的连接 PC 的接口划入 vlan10

完成后两个 PC 就能够互相通信了。

这时候我们在 SW1 上看一下 trunk 的状态：

SW1#sh int trunk

Port	Mode	Encapsulation	Status	Native vlan
Fa0/20	on	802.1q	trunking	1

Port	Vlans allowed on trunk
Fa0/20	1-4094

Port	Vlans allowed and active in management domain
Fa0/20	1,10,20,30

Port Vlans in spanning tree forwarding state and not pruned
Fa0/20 1,10,20,30 !!在关闭 VTP prunnig 的情况下 ,trunk 默认放行所有 VLAN 的流量 ,这在 SW2 及 SW3 上情况一样

接下去我们在 SW1 的 Fa0/20 接口上，做 switchport trunk allowed vlan remove 10，将 vlan10 的流量修剪掉，这时候 PC 之间就无法 ping 通了，再去 R1 上看一下：

SW1#sh int trunk

Port	Mode	Encapsulation	Status	Native vlan
Fa0/20	on	802.1q	trunking	1

Port Vlans allowed on trunk

Fa0/20 1-9,11-4094

Port Vlans allowed and active in management domain

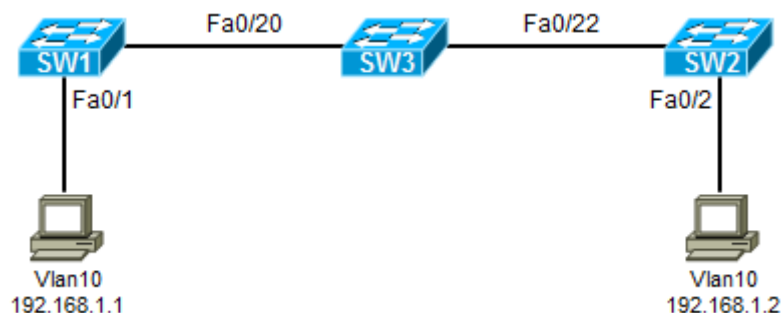
Fa0/20 1,20,30

Port Vlans in spanning tree forwarding state and not pruned

Fa0/20 1,20,30

注意，此时此刻虽然 SW1 在 fa0/20 口上将 vlan10 的流量修剪了，但是 SW3 的 fa0/20 口确实依然放行该流量的，不过不管怎样，PC 之间是无法互访了，这里只是做个演示，知道一下 allowed vlan 的作用。当然，这是手工修剪的方式。

3) 测试目的：VTP pruning



去掉步骤 2 中配置在 SW1 上的命令，还原实验环境。

接下去我们在 SW3，也就是 VTP 的 server 上开启 vtp pruning。

SW3#sh vtp st

VTP Version capable : 1 to 3

```
VTP version running          : 1
VTP Domain Name              : ccnp
VTP Pruning Mode           : Enabled
VTP Traps Generation         : Disabled
Device ID                    : 000a.8a07.8280
Configuration last modified by 0.0.0.0 at 3-5-93 02:01:49
Local updater ID is 0.0.0.0 (no valid interface found)

Feature VLAN:
-----
VTP Operating Mode           : Server
Maximum VLANs supported locally : 1005
Number of existing VLANs      : 8
Configuration Revision        : 17
MD5 digest                   : 0x73 0x52 0x60 0xE7 0x4D 0xA5 0xC7 0x4F
                               0xCA 0x3D 0x6F 0x1D 0x3F 0x23 0x03 0xBB
```

在 VTP server mode 的 SW3 上开启 VTP pruning 后，Client 的 SW1 及 SW2 都会学习到并且也开启自己的 VTP pruning。

如此一来，三台交换机都会进行 VTP 报文的交互，告知自己本地存在接入用户的 VLAN。对于本地没有用户的 VLAN，将会被自动修剪掉，我们看一下：

SW1#show int trunk

Port	Mode	Encapsulation	Status	Native vlan
Fa0/20	on	802.1q	trunking	1

Port	Vlans allowed on trunk
Fa0/20	1-4094

Port	Vlans allowed and active in management domain
Fa0/20	1,10,20,30

Port	Vlans in spanning tree forwarding state and not pruned
Fa0/20	1,10

!! 在 R1 上，Fa0/20 口修剪得只剩下了 vlan 1 和 10，因为本地只有 vlan1 和 vlan10

SW2、SW3 也是类似的情况；注意，此刻我们是没有在 trunk 口上做任何静态配置的 allowed vlan 配置，

这些都是 VTP pruning 自动协商完成的。

接下去，我们在 SW2 上，将 vlan10 的端口移除，可直接在 fa0/2 上 no switchport access vlan
如此一来 SW2 上就没有 vlan10 的用户了，sw2 会泛洪这一消息以便其他交换机知道。

SW3 收到这一消息后，知道 SW2 不再有 vlan10 的用户，也就不再需要 vlan10 的流量了，于是在自己的 FA0/22 口上将 vlan10 修剪掉：

SW3#sh int tru

Port	Mode	Encapsulation	Status	Native vlan
Fa0/20	on	802.1q	trunking	1
Fa0/22	desirable	n-isl	trunking	1

Port Vlans allowed on trunk

Fa0/20 1-4094

Fa0/22 1-4094

Port Vlans allowed and active in management domain

Fa0/20 1,10,20,30

Fa0/22 1,10,20,30

Port Vlans in spanning tree forwarding state and not pruned

Fa0/20 1,10

Fa0/22 1 !! vlan10 被修剪

最后注意：

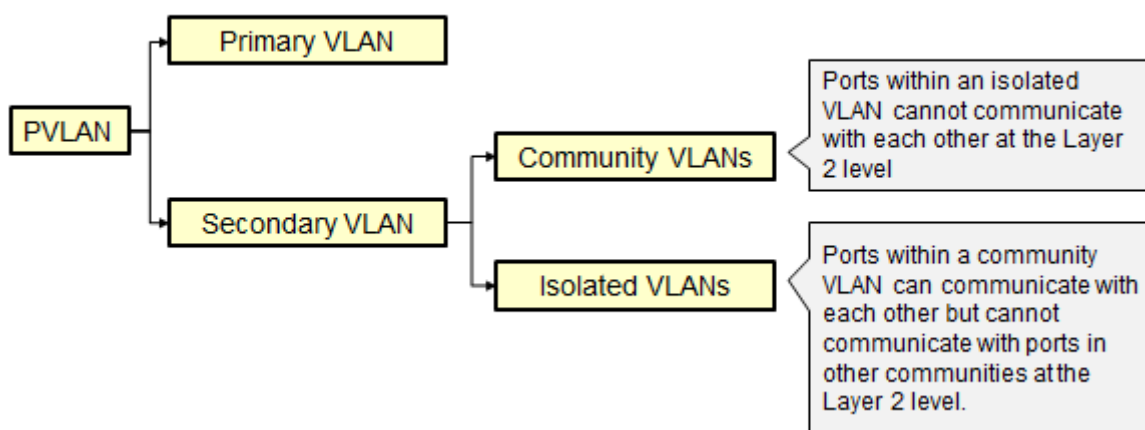
- Vtp pruning 只能在 server mode 上开启
- Server mode 上开启后，client 都会自动开启 pruning
- Vtp pruning 默认是关闭的

2.4 私有 VLAN (PVLAN)

1. 私有 VLAN 的概念

- 将一个 VLAN 划分成几个单独的 VLAN，这些 VLAN 都使用同一个 IP 网段。

- 可以提高安全性，降低子网数目，提高 IP 利用率
- 尽管网络设备处于同一个子网中，但是他们属于不同的 pVlan，pVlans 之间的通信还是必须通过默认网关来实现。
- 每个 pVLAN 包括一个**主 VLAN** 以及多个**辅助 VLAN**。所有的辅助 VLAN 都映射到主 VLAN。
- **辅助 VLAN 分为团体 VLAN 和隔离 VLAN。**
- 相同团体 VLAN 能够互相通信，但是团体 VLAN 之间必须通过设置 SVI 或者路由器接口才能通信。
- 相同隔离 VLAN 内部以及隔离 VLAN 之间都是不能够互相通信的，只能与混杂接口通信。
- **一个主 VLAN 只能有一个 Isolate vlan**
- 混杂端口能够与 pVLAN 中的任何设备通信，不管对方是处于主 VLAN，还是辅助 VLAN。



2. PVLAN 端口类型

Isolated

Communicates with only promiscuous ports

Promiscuous

Communicates with all other ports

Community

Communicates with other members of community and all promiscuous ports

3. PVLAN 注意事项

具体支持的设备，见 CISCO 官方文档。

关于 PVLAN 的配置：

<http://www.cisco.com/en/US/docs/switches/lan/catalyst4500/12.2/31sga/configuration/guide/pvlans.html>

pVLAN 必须配置在透明模式的交换机上。而且要求 VTP 版本 1 或 2。

禁止将第三层的 VLAN 接口配置为辅助 VLAN。

Etherchannel 端口不支持 pvlan

4. PVLAN 的配置

创建主 VLAN :

```
Vlan 100
Private-vlan primary
```

创建辅助 VLAN :

```
Vlan 101
Private-vlan community
Vlan 102
Private-vlan isolate
```

配置主 VLAN , 将二层辅助 VLAN 关联到主 VLAN

```
Vlan 100
Private-vlan association 101,102
```

将辅助 VLAN 映射到主 VLAN 的 SVI 接口 , 从而允许 pVLAN 入口流量的三层交换。

```
Interface vlan 100
Private-vlan mapping add 101,102
```

配置接口 :

- 如果将接口配置为主机接口

```
interface f0/1
Switchport mode private-vlan host
Switchport private-vlan host-association 100 101 //关联主 VLAN 和辅助 VLAN 到接口
Interface f0/2
Switchport mode private-vlan host
Switchport private-vlan host-association 100 102
```

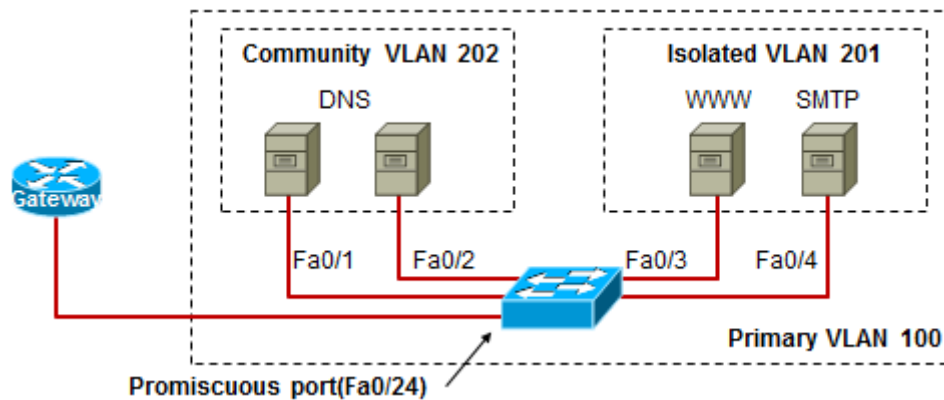
- 如果将接口配置为混杂接口

```
Interface f0/3
Switchport mode private-vlan promiscuous
Switchport private-vlan mapping add 100 101 //将端口映射到 pVLAN
```

查看及验证：

```
show pvlan mapping
```

5. PVLAN 的配置示例



- DNS、WWW、SMTP 服务器同属一个子网
- 两台 DNS 服务器同属一个 community VLAN，彼此之间能够互相通信
- WWW 及 SMTP 属于 Isolated VLAN，因此彼此之间无法互相访问；WWW 及 SMTP 服务器都只能与混杂端口，也就是与路由器通信

```
sw(config)#vtp transparent
```

```
sw(config)#vlan 201
```

```
sw(config-vlan)#private-vlan isolated
```

```
sw(config)#vlan 202
```

```
sw(config-vlan)#private-vlan community
```

```
sw(config)#vlan 100
```

```
sw(config-vlan)#private-vlan primary
```

```
sw(config-vlan)#private-vlan association 201,202
```

```
!
```

```
sw(config)#interface fa0/24
```

```
sw(config-if)#switchport mode private-vlan promiscuous
```

```
sw(config-if)#switchport private-vlan mapping 100 201,202
```

```
sw(config)#interface range fa 0/1-2
```

```
sw(config-if)#switchport mode private-vlan host
```

```
sw(config-if)#switchport private-vlan host-association 100 202
```

```
sw(config)#interface range fa 0/3-4
```

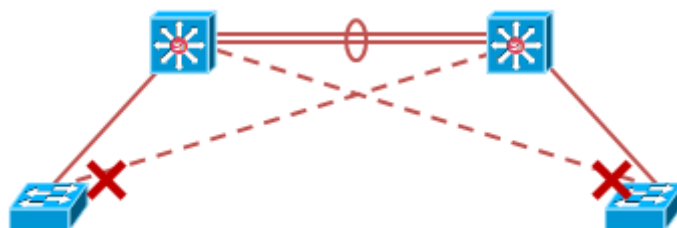
```
sw(config-if)#switchport mode private-vlan host
```



```
sw(config-if)#switchport private-vlan host-association 100 201
```

3 Spanning-tree

3.1 协议概述



- 如果接入层交换机单链路上联，则存在单链路故障另一个问题的单点故障，如果任意一个汇聚设备宕机，将直接导致下联的接入网络挂掉
- 与是接入层交换机采用双链路上联到两台汇聚设备，构成一个物理链路冗余的二层环境，解决了单链路故障问题
- 但是二层环境存在环路
- 生成树用于解决这个问题
- 通过生成树协议，在逻辑上将特定端口进行 Block，从而实现物理上存在冗余环境，而二层上又阻止环路的产生
- 当拓扑发生变更的时候，生成树协议能够探测到这些变化，并且及时自动的调整接口状态，从而适应网络拓扑的变化，实现链路冗余

3.2 协议基础

1. 生成树协议

PROTOCOL			Describe
STP (PVST)	Public	802.1 D	
PVST+	Cisco Private		Portfast,uplinkfast, backbonefast
RSTP	Public	802.1W	集成了 pvst 的功能并公有化
MST	Public	802.1S	

2. 参数

- **网桥 ID (8 字节) = 网桥优先级(2 字节)+网桥 MAC (6 字节)**
缺省优先级 32768,范围 0-65535 (前 4bit 表示优先级,后 8bit 作为 system id,为协议扩展用 , 越小越好,4096 的倍数)
- **端口 ID (2 字节) = 端口优先级(1 字节)+端口 ID(1 字节)**
缺省优先级 128 , 范围 0-255 , 越小越好
Catos 交换机端口 ID 中优先级默认 32(优先级 6 个 bit) , IOS 交换机默认 128(优先级字段 8 个比特)
- **根路径开销**
本交换机到达根交换机路径的总开销
越小越好 , 与接口带宽有关

Link Speed	Cost (New IEEE Specification)	Cost (Old IEEE Specification)
10 Gb/s	2	1
1 Gb/s	4	1
100 Mb/s	19	10
10 Mb/s	100	100

3.3 802.1D

3.3.1 BPDU 报文

BPDU 有两种类型：配置 BPDU 及 TCN

1. 配置 BPDU

在网络刚开始运行的阶段，所有交换机都会从所有端口发送 BPDU，大家都认为自己是 root，随着 BPDU 泛洪和收集，根据 BPDU 中所含信息，大家 PK 出来个结果，root 被选举出来了。在此之后由 Root 以默认 2S 为周期发送 BPDU，所有的非 root 交换机从自己的根端口收到 BPDU，再从自己的指定端口产生 bpdu 发出去。这就有点像我们从 root 倒一盆水下来，水顺着这颗无环的树从上往下不断的流。另外，被 block 的非指定端口会源源不断的收到链路上的 bpdu 并一直侦听，当其在一定时间内没有再收到 bpdu，则认为链路出现了故障，开始进入新的收敛阶段。

“When a switch receives a configuration BPDU that contains superior information (lower bridge ID, lower path cost, and so forth), it stores the information for that port. If this BPDU is received on the root port of the

switch, the switch also forwards it with an updated message to all attached LANs for which it is the designated switch.

If a switch receives a configuration BPDU that contains inferior information to that currently stored for that port, it discards the BPDU. If the switch is a designated switch for the LAN from which the inferior BPDU was received, it sends that LAN a BPDU containing the up-to-date information stored for that port. In this way, inferior information is discarded, and superior information is propagated on the network."

字节	字段	描述
2	协议	代表上层协议 (BPDU), 该值总为 0
1	版本	(802.1D 的总为 0)
1	TYPE	"配置 BPDU" 为 0x00、"TCN BPDU" 为 0x80
1	标志	LSB 最低有效位表示 TC 标志 ; MSB 最高有效位表示 TCA 标志
8	根 ID	根网桥的桥 ID
4	路径开销	到达根桥的 STP cost
8	网桥 ID	BPDU 发送桥的 ID
2	端口 ID	BPDU 发送网桥的端口 ID (优先级+端口号)
2	消息寿命 Message age	从根网桥发出 BPDU 之后的秒数 (这个 BPDU 存活了多长时间了), 每经过一个网桥都减 1 , 所以它本质上是到达根桥的跳数。
2	最大寿命 Max age	当一段时间未收到任何 BPDU , 生存期到达 MAX age 时 , 网桥认为该端口连接的链路发生故障。也可以理解为这个 BPDU 的最大寿命 , 默认 20S
2	HELLO 时间	根网桥连续发送的 BPDU 之间的时间间隔。默认 2S
2	转发延迟	在监听和学习状态所停留的时间间隔。默认 15S

Catos 交换机端口 ID 中优先级默认 32 (优先级 6 个 bit) , IOS 交换机默认 128 (优先级字段 8 个比特)

2. TCN BPDU

在网络拓扑变化的时候产生。

TYPE 字段 为 0x80

Flag 字段 LSB=TC MSB=TCA

当网络拓扑发生变化的时候 , 最先意识到变化的交换机会从根端口发送 TCN (TYPE 字段=0x80) 到上一层交换机 (朝向 ROOT 的方向) , 一直到根交换机 , 上一层交换机除了向其上一层交换机发送 TC 外 , 也回一个 TCA (MSB 置位) 的确认信息给前一个交换机。当根接收到 TC 后发送 TCA 回最开始的发送交换机并向所有网桥发送 TC (LSA 置位) 。交换机收到 ROOT 发出来的 TC 后 , 会将 MAC 地址表的老化时间减少为 15s (转发延迟) , 这个 TC 一直会持续 35s (20+15)

3.3.2 STP 的运行

1. STP 采用四部来解决二层环路：

- 1) 在一个交换网络中选举一个 root bridge
- 2) 在每个非根桥上选举一个根端口
- 3) 为每个 Segment 选举一个指定端口
- 4) 阻塞非指定端口

2. 比较原则

STP 需要网络设备相互交换消息来检测桥接环路，该消息称为**网桥协议数据单元 BPDU**，阻塞端口也会不断收到 BPDU，以保证故障发生的时候，仍然可以计算出一棵新的 STP。要理解 STP 的工作过程，非常重要的一点是要理解 BPDU 中各字段的含义，因为这些都是 STP 赖以工作的根本。

生成树构造一个无环路拓扑时，总是使用相同的 4 步来判定：

- **最低根桥 ID**
- **到根桥的最低路径成本**
- **最低的发送者网桥 ID**
- **最低的发送者端口 ID**

网桥使用这 4 步来保存各个端口接收到的“最佳”的 BPDU 的一个副本。每个 BPDU 到达时，都会按照这个 4 步判决步骤来进行检查，看它是否该端口保存的 BPDU 更优，如果是，则会更新。

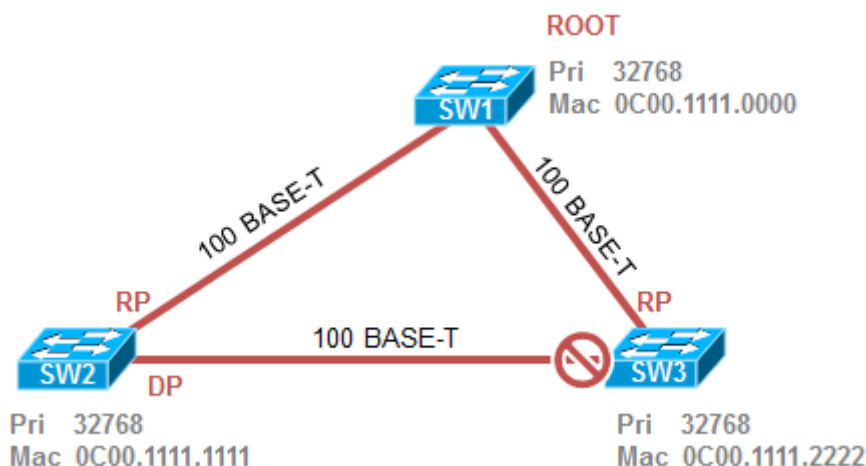
当一个网桥开始活动时，他的每个端口都是每 2s 发送一个 BPDU，而当一个端口收到一个比现在发送的更优的 BPDU，则本地端口会停止发送，如果在一段时间内（却省为 20s）后他不再收到来自邻居的更优 BPDU，则他将再次发送。因此对于 802.1D 来说，根桥会不停的向所有接口发送 BPDU，而非根桥会从自己的根端口收到 BPDU，并且向自己的指定端口去发 BPDU，非指定端口是不会发送 BPDU 的，只会侦听。

3. 注意事项：

- 根桥的角色是可抢占的
- 桥 ID 中的 MAC，是交换机的背板 MAC，端口 ID 中的 MAC 是交换机的端口 MAC，查看交换机上的所有 mac 可用
show interface | include bia
- 二层交换机的端口 mac 就在这用了：)

3.3.3 案例分析

- CASE1 :

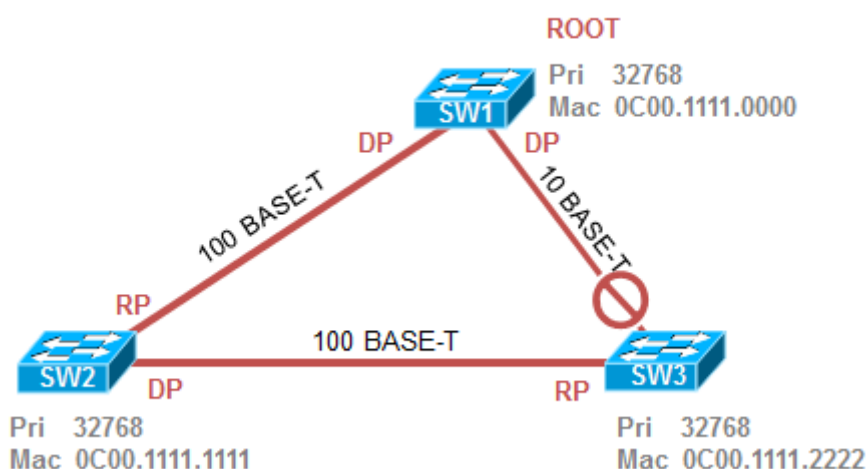


首先选 root，交换机 MAC 最小的 SW1 胜出，成为 root

其次在非根桥 SW2 及 SW3 上选择一个 root port，如图所示，因为这些端口到达 SW1 开销最小

最后在每个 segment 上选择一个指定端口 DP，sw1 是根桥，所有接口都是 DP，最后比较 SW2、SW3 之间直连链路的两个接口。SW2 会从接口上收到 SW3 发来的 BPDU，SW3 也会收到 SW2 发来的 BPDU 各自比较自己和收到 BPDU 中的桥 ID（也就是所谓的比较发送桥 ID），最终 SW2 由于 MAC 地址较小胜出。因此 Block 掉 SW3 的那个接口。

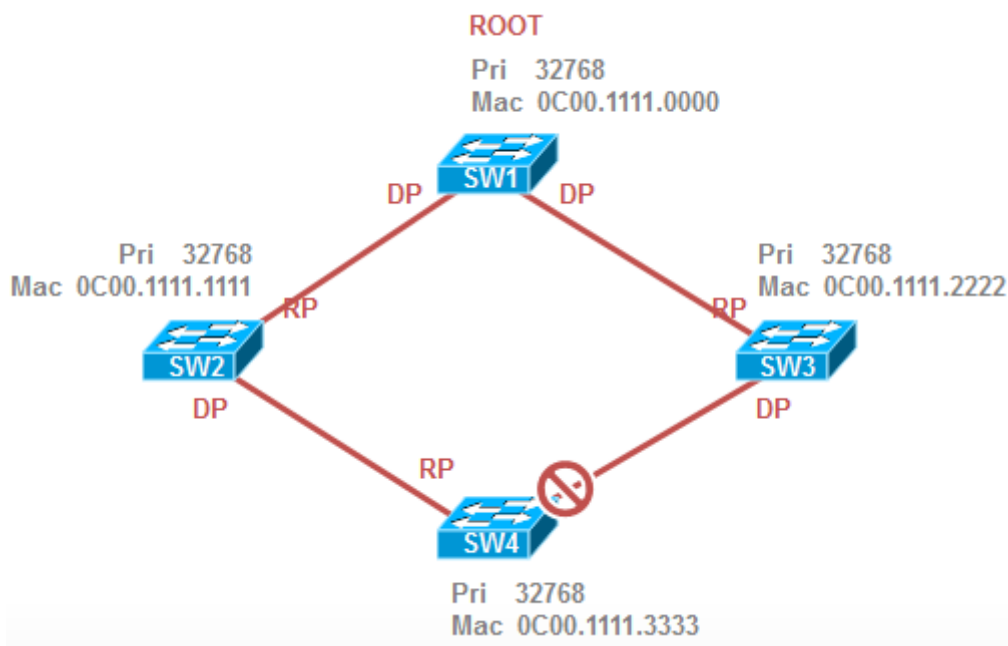
- CASE2 :



根桥的选举就不说了，接下去看 RP，SW2 两个接口，都会收到 BPDU，而上联到 root 的接口收到的 BPDU

中到 root 的开销最小，所以上联口为 RP。SW3 却不一样，由于连接到 root 的接口带宽仅为 10M，因此连接 SW2 的接口胜出为 RP。最后选择 DP。简单不赘述了。

- CASE3 :

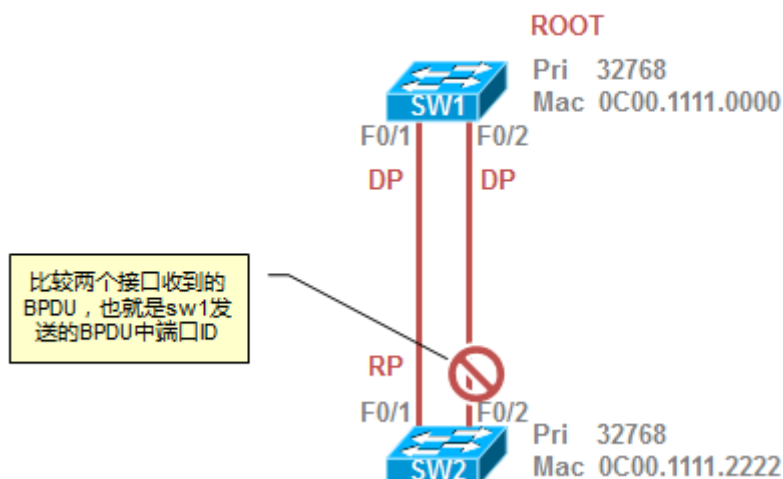


根桥的选举就不说了

根端口的选举，SW2 和 SW3 都比较简单。关键看 SW4，有两个接口，这两个接口都会收到 BPDU，首先看到 root 的开销，由于这两个 BPDU 的到 root 开销相等，加上 SW4 这两个接口的带宽也相等，因此这一步比较不出来，接着看发送者桥 ID，也就是收到的这两份 BPDU 的 BridgeID，比较后发现 SW2 的 MAC 更小 (比 SW3 小)，因此 SW4 上连接 SW2 的接口胜出，成为 RP。

最后看 DP，也比较简单如图所示。我们拿 SW3 及 SW4 之间的 segment 举例，由于 SW3 及 SW4 都会产生 BPDU，而 SW3 发给 SW4 的 BPDU 明显要优于 SW4 自己产生的从这个接口发送的 BPDU，因此最终 SW4 的接口胜出。

- CASE4 :



根桥的选举就不说了。

接下去看 RP，SW2 有两个接口，这两个接口都会收到 BPDU，都来自 SW1，因此这两个 BPDU 首先到 root 的开销都是 0，相等；其次发送者的桥 ID，由于是来自同一台交换机，因此也相等，也比较不出来；再比较这两个 BPDU 的端口 ID，也就是 SW1 上这两个接口的端口 ID，我们假设两个接口优先级相等，那么就比这两端口的接口 ID，最终选出 SW2 上的根端口 F0/1。

注意，这时候如果试图在 SW2 上将 F0/2 的接口优先级改小，也是没用的，因为看的是发送者的端口 ID。所以如果在 SW1 上，将 F0/2 的端口优先级调小。那么 SW2 上，F0/2 就会胜出成为根端口。

3.3.4 STP 端口状态

1. 写在前面的话：

由于网络设备存在固有的滞后，所以交换网络中也就存在传播延迟。基于上述原因，拓扑变更就可能发生在交换网络中的不同时间和不同的网段。如果 2 层接口直接从生成树的 block 状态切换到转发状态，就可能会出现暂时的数据环路。为了缓解这种问题，在开始转发数据帧之前，端口应当等待新的拓扑信息传播到整个交换网络中。

2. 计时器

阻塞到转发状态通常要 30-50s（默认 50s，即 20+15+15），这个时间也可以通过配置生成树计时器来调整。

Hello 时间 根网桥发送配置 BPDU 的时间间隔 缺省为 2s

转发延迟时间 侦听到学习状态，或者学习状态转换到转发状态所需要的时间 缺省为 15s。

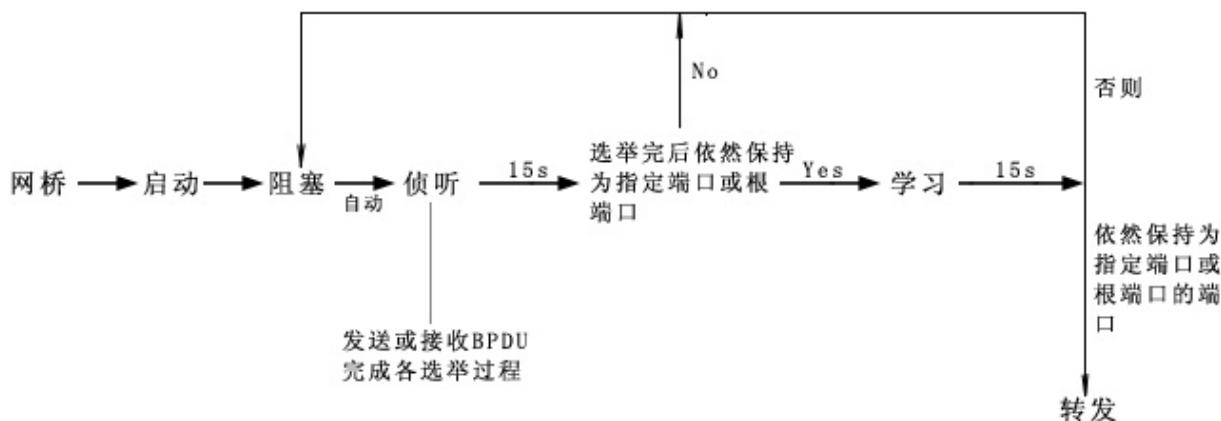
最大存活期 在丢弃 BPDU 之前，网桥用来存储 BPDU 的时间，缺省为 20s。如果连续收不到 10 个 bpdu（20s 的时间），开始进入 listening 状态

网络中的生成树拓扑依附于根网桥的计时器，root 将 BPDU 中的计时器传递给第二层的所有交换机。

3. STP 各状态如下：

Disable	不收发任何报文
Blocking	不接收也不转发帧，接收但不发送BPDU，不学习MAC地址
Listening	不接收也不转发帧，接收并且发送BPDU，不学习MAC地址
Learning	不接收也不转发帧，接收并且发送BPDU，学习MAC地址
Forwarding	接收并转发帧，接收并且发送BPDU，学习MAC地址

4. STP 端口状态转换过程



3.3.5 STP 拓扑变更

1. TCN BPDU 概述

当网络拓扑出现变更的时候，最先意识到变化的交换机将发送 TCN BPDU。

在发生以下时间时，交换机发送 TCN：

- 对于处于转发和监听状态的接口，过渡到 Block 状态（链路故障的情况）
- 端口进入转发状态，并且网桥已经拥有指定端口
- 非 root 网桥在它的指定端口收到 TCN

2. TCN BPDU

TCN BPDU 包含 3 个字段，它与配置 BPDU 除了 type 字段之外的前 3 个字段完全相同。

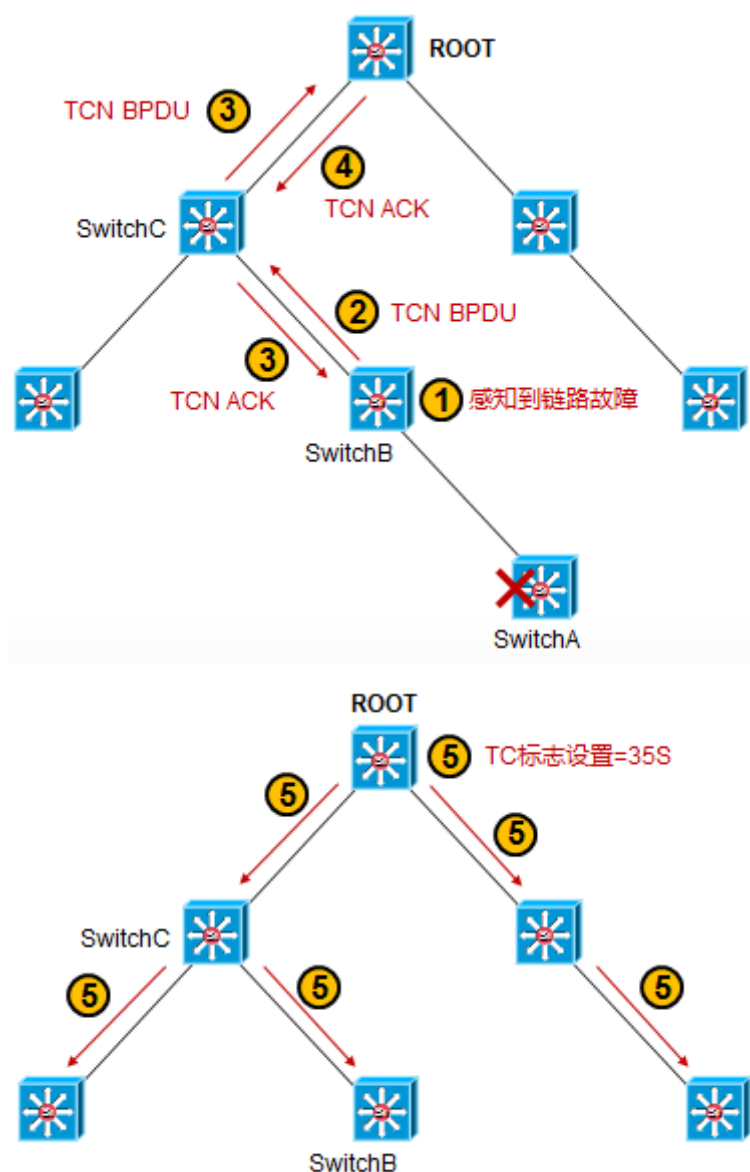
3. 拓扑变更过程

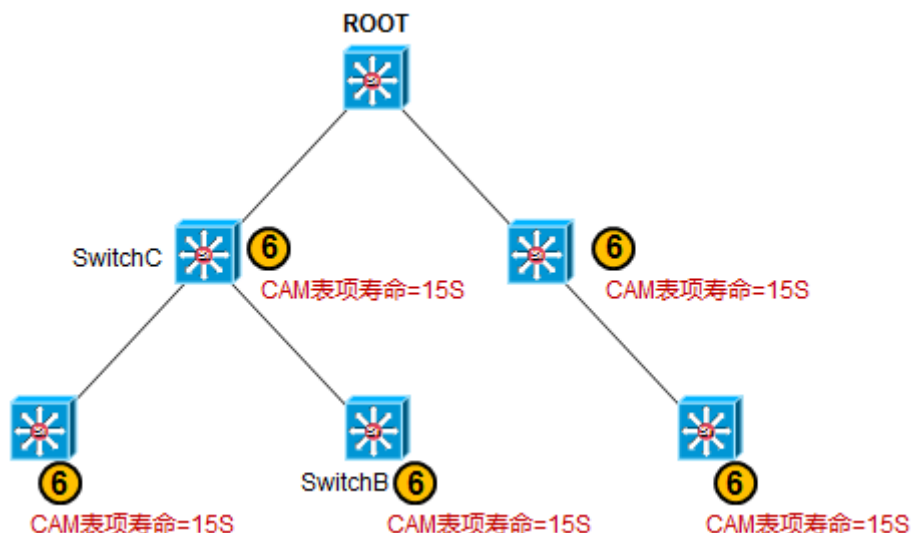
最先意识到拓扑变更的交换机发送 TCN，指定网桥接收 TCN 并且通过立即回送一个 TCA 被置位的正常 BPDU 来确认，在该网桥确认这个 TCN 之前，负责通知拓扑变更的网桥将持续发送 TCN BPDU。

接下去该指定网桥将为自己的根端口产生另外的 TCN，并且这个过程一路发到 root 为止。

一旦网桥意识到网络中发生拓扑变更的情况，它将发送 TC 被置位的配置 BPDU，网络中的每台交换机都将传递这些被置位的 BPDU，进而便于每个单独的网桥都意识到拓扑变更的情况。

4. 拓扑变更过程 范例





- 1) SwitchA 挂掉
- 2) SwitchB 最先检测到拓扑变化，于是产生 TCN BPDU 并从根端口发送出去（因为根端口是朝着根桥的方向），B 将连续发送 TCN BPDU 直到指定交换机 C 发送 TCN ACK 进行确认
- 3) SwitchB 收到这个 TCN BPDU，回送一个 TCN ACK 进行确认，同时向自己的根端口转发这个 TCN BPDU
- 4) Root 收到这个 TCN，回送一个 TCN ACK 给 C。
- 5) Root 修改自己的配置 BPDU，以此来通告整个交换网络关于拓扑变更的情况。Root 在配置 BPDU 中设置一段时间的拓扑变更（TC 标志），这段时间等于转发延迟+Max. Age，默认 35S
- 6) 当交换机收到 Root 发出的这个 TC 标志置位的配置 BPDU，它们使用转发延迟计时器（默认 15S）来更新其 MAC 地址表中的条目。也就是说条目的寿命由原来的 300S 的默认值变成 15S，这样能保证 MAC 地址条目更快速的刷新。交换机将持续这个过程，直到不再从 Root 收到 TC BPDU 消息为止。

我们会发现当拓扑变更的时候，就会产生 TCN，然而有些情况下 TCN 的过渡泛洪可能会对网络造成不必要的影响，通过在接入层交换机上、连接 PC 终端设备的接口设置为 portfast 可以在一定程度上优化网络，防止由于 PC 的开关机导致的接入交换机端口 updown 而产生过多的 TCN。

3.3.6 Spanning-tree 特性

1. Portfast

portfast 端口一旦接了设备，接口可绕过 listening 和 learning 状态直接进入 forwarding 状态。

```
Switch(config)# spanning-tree portfast default
```

(全局配置命令) 将所有非 trunk 接口激活 portfast 特性

```
Switch(config-if)# spanning-tree portfast [trunk]
```

将特定接口激活 portfast 特性

Spanning-tree portfast 特性不能直接配置在 trunk 模式的接口上，否则即使配上去了，CISCO IOS 也不生效，除非该接口变成 access 模式。如果确实需要在 trunk 接口上配置，例如该接口连接了一台支持 trunk 的服务器，那么就在 Spanning-tree portfast 命令上增加 trunk 关键字。

```
Switch(config-if)# switchport mode host
```

一条宏命令，指定接口 mode 为 access 并且开启 portfast 特性

```
Switch(config)# spanning-tree portfast bpduguard default
```

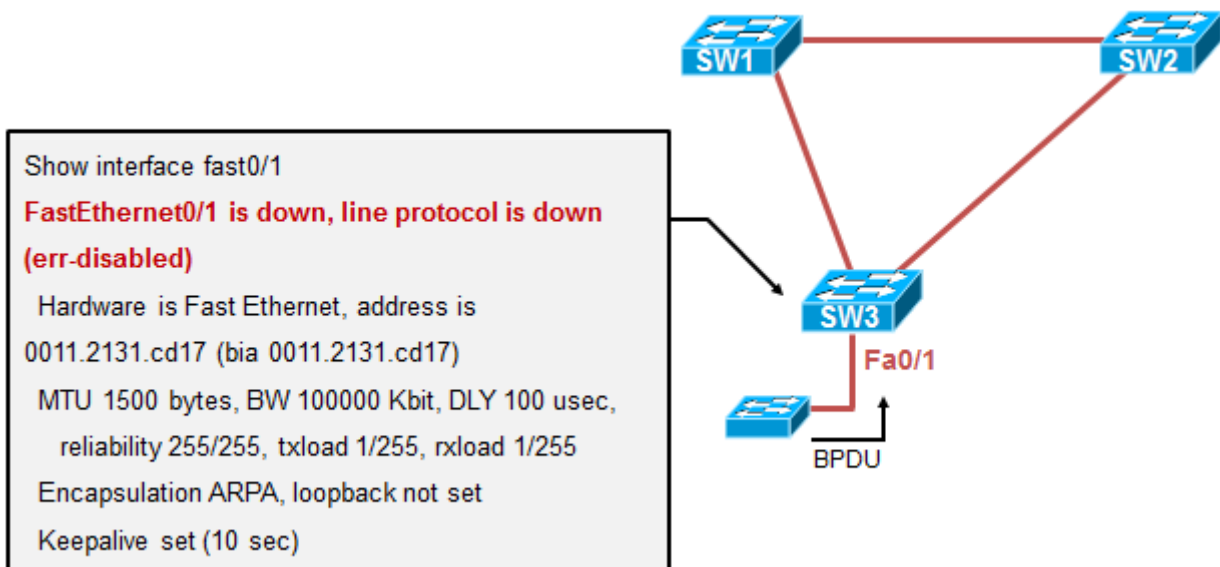
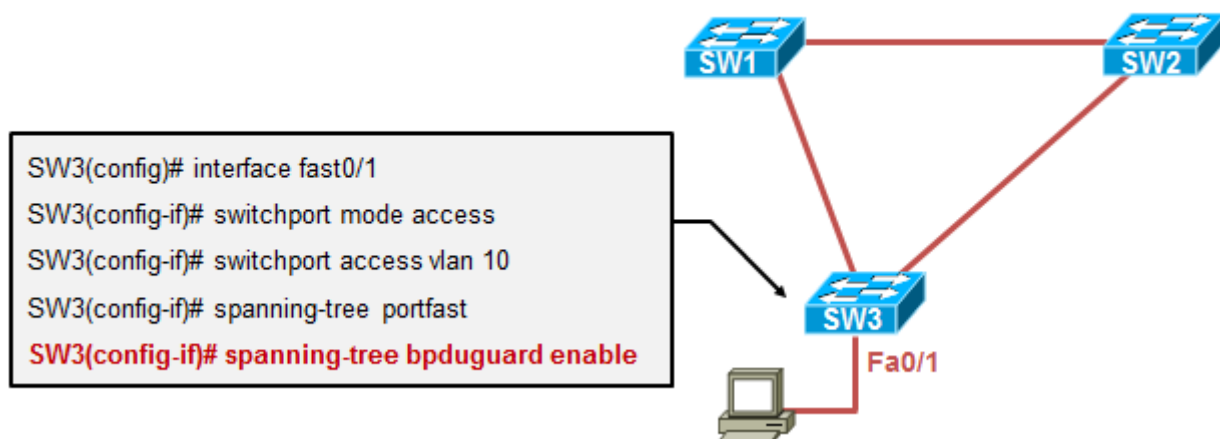
将所有激活了 portfast 的接口激活 bpduguard

注意，虽然配置了 portfast 的接口在 no shutdown 或者 link 一旦 up 的时候，会 jump to forwarding 状态，绕过 listening 和 learning，但是，这个接口建议必须是连接路由器或 PC 的。如果连的是交换机，这个接口仍然要接受 spanning-tree 的计算结果，如果计算结果是 block，那么这个接口仍然会被 block。这就是为什么你在 CISCO IOS 上敲入该条命令，IOS 会提示你，说可能会造成短暂的桥接环路。

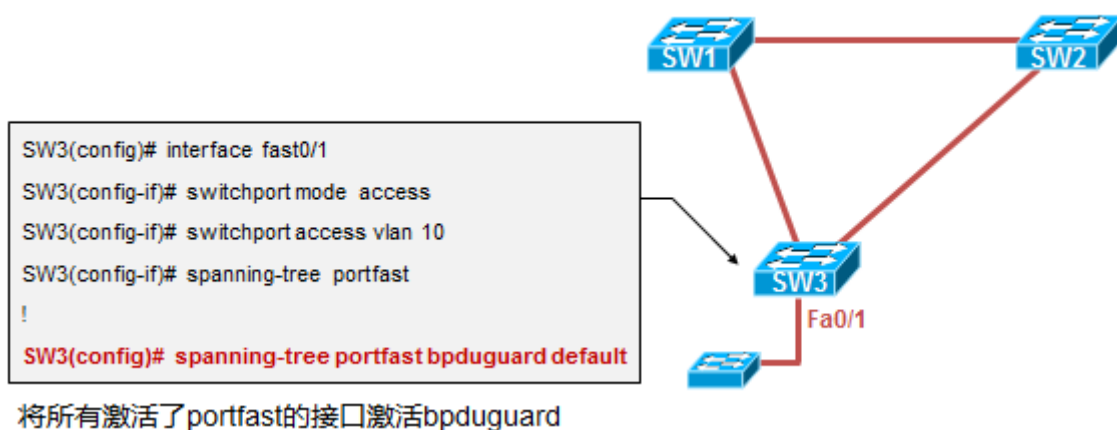
2. BPDUguard

- 该接口在收到 BPDU 报文后，会立即切换到 err-disable 状态
- 常搭配 portfast 特性在接口上一起使用，用于连接主机
- 可在接口模式上激活，也可在全局模式上配置，两者有所不同。

在接口模式下配置，该接口收到 BPDU 报文后，立即切换到 err-disable 状态：



在全局下配置，使用命令：SW3(config)# spanning-tree portfast bpduguard default，该命令会在所有激活 portfast 特性的接口上激活 bpduguard：



使用 err-disable 命令可以修改 err-disable 状态的持续时间，默认是 300S

3. BPDUFilter

可以在全局模式下配置，也可以在接口模式下配置，区别如下：

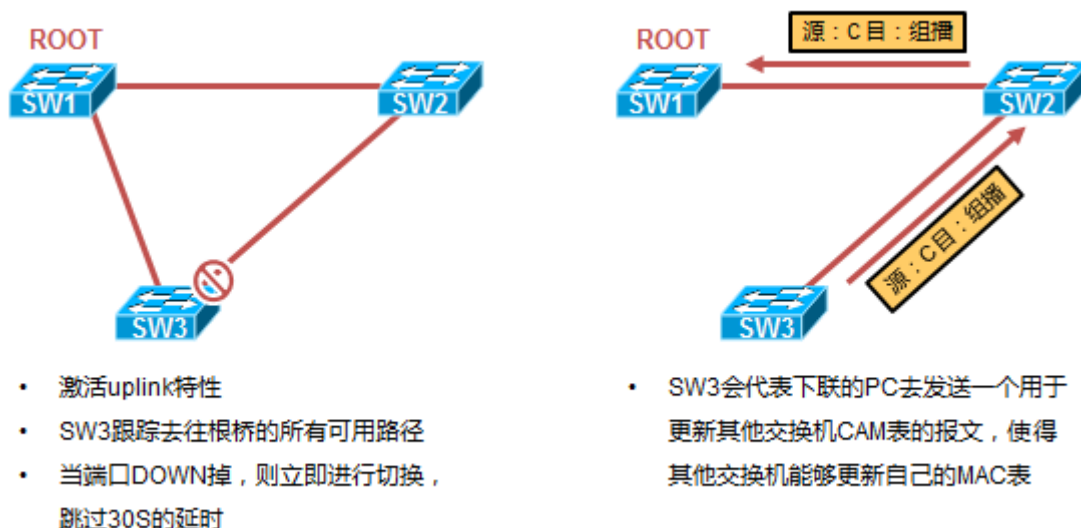
全局配置： `spanning-tree portfast bpduguard default`

- 启用了 portfast 的接口将激活 bpduguard 特性
- 上述接口在 link up 后瞬间会发送 BPDU (a few)，此后不在发送任何 BPDU
- 上述接口在收到 BPDU 后立即丢失 portfast 及 bpduguard 特性，成为一个普通的 spanning-tree 接口

接口配置： `spanning-tree bpduguard enable`

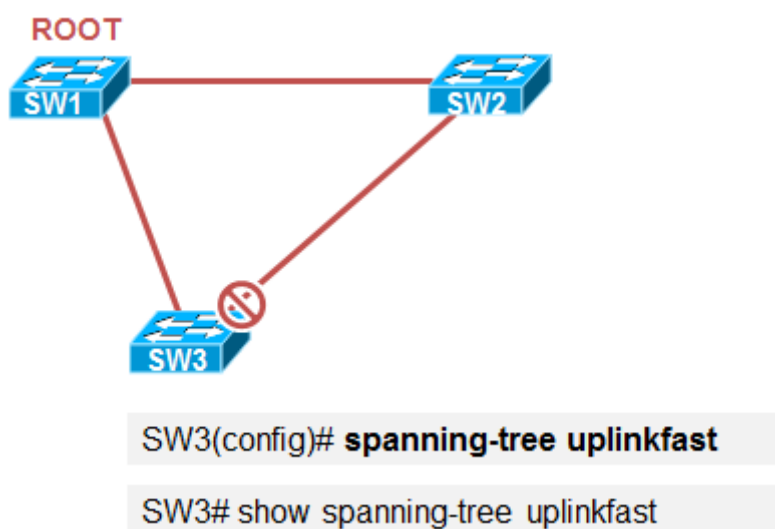
- 该接口将不会发送 BPDU，也忽略接收到的 BPDU
- Enabling BPDU filtering on an interface is the same as disabling spanning tree on it and can result in spanning-tree loops.
- 在接口上配置，不一定必须 portfast 特性，可独立实施。当然，建议搭配 portfast 特性使用。

4. UplinkFast

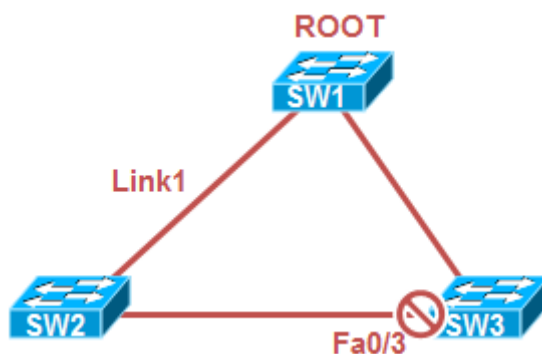


- 配置 Uplinkfast 的交换机需为末梢交换机，或者网络三层结构中的接入层交换机，不能为根桥
- 在激活 Uplinkfast 特性后，交换机会自动调整一些参数：
 - 交换机的默认桥优先级会增加到一个比缺省值更高的值（32768->49152），以使得该交换机不会成为 Root
 - 交换机的所有端口的默认 COST 值会增加 3000，以使得该交换机的端口不被选举为指定端口
 - 非默认的（手工配置的非默认）priority 与 cost 不变

另外：When the spanning tree reconfigures the new root port, other interfaces flood the network with multicast packets, one for each address that was learned on the interface. You can limit these bursts of multicast traffic by reducing the max-update-rate parameter (the default for this parameter is 150 packets per second). However, if you enter zero, station-learning frames are not generated, so the spanning-tree topology converges more slowly after a loss of connectivity.

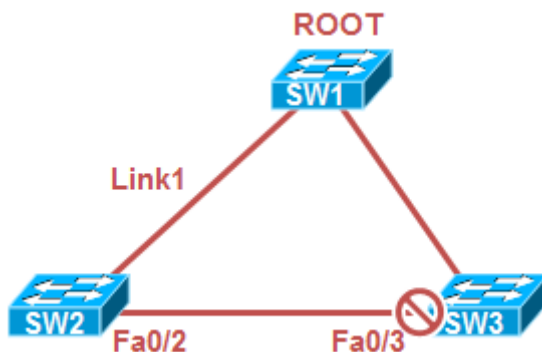


5. BackboneFast



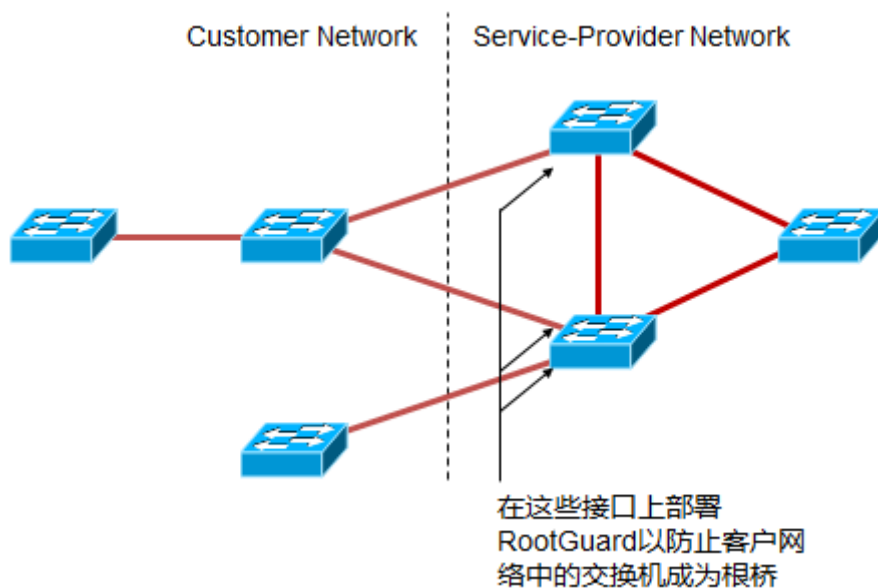
- 拓扑正常的情况下，SW3 有一个接口被 BLOCK，这个接口将不会发送 BPDU 报文
- 当 LINK1 DOWN 掉后，SW2 将无法从 ROOT 收到 BPDU，于是它认为自己的 Root，并且开始向 SW3 发送 BPDU
- SW3 收到这个 BPDU，但是发现这个 BPDU 比之前自己本地存储的 BPDU 更不优，因此忽略该 BPDU
- MAX_AGE (20S) 计时器超时后，SW3 上 Fa0/3 存储的 BPDU 老化，该端口进入侦听状态，并发送和接收 BPDU
- SW2 收到 SW3 发送的 BPDU 后，就停止发送它自己的 BPDU
- SW3 的 Fa0/3 从 Listening 到 Forwarding，需花费 20+30S

接下来看看 BackboneFast 的特性：

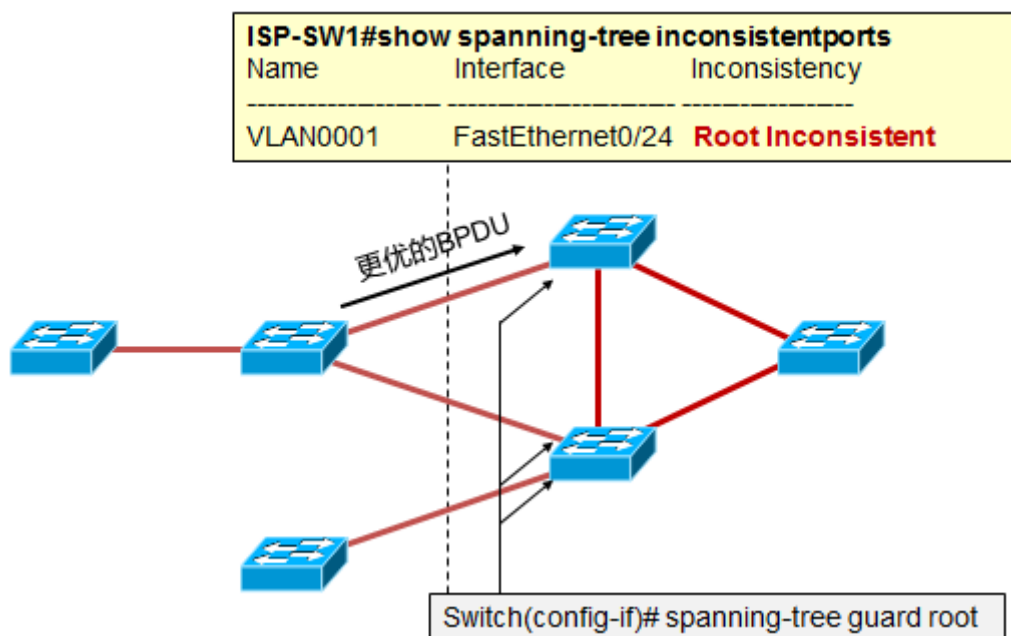


- 在部署了 Backbone Fast 后，SW3 一到收到 SW2 发送来的次优 BPDU，会即刻进行一系列步骤以重新计算根端口。SW3 会从根端口向根桥发送 RLQ 请求
- Root SW1 收到 RLQ 请求，立刻以 RLQ 响应进行回复，以告知自己仍然存活
- SW3 立即老化掉存储在 Fa0/3 上的 BPDU，端口 Fa0/3 进入 listening 状态并开始发送 BPDU
- SW2 从 SW3 收到 BPDU，经过计算得出自己的 Fa0/2 为根端口

6. RootGuard



上图中，客户与运营商网络之间是二层链路连接，如果客户的设备由于某种原因被选举为根桥，那就麻烦了，因此可以使用 rootguard 特性，在运营商连接客户的接口上部署。那么当这些接口收到来自客户的、更优的 BPDU 后，当运营商设备经过 STP 计算后得出这些接口即将成为根端口，此时设备立即将接口置为 root-inconsistent (blocked)状态。因此使用该特性，可以防止客户设备成为 Root，或者防止运营商设备上、连接客户设备的这些接口成为根端口。

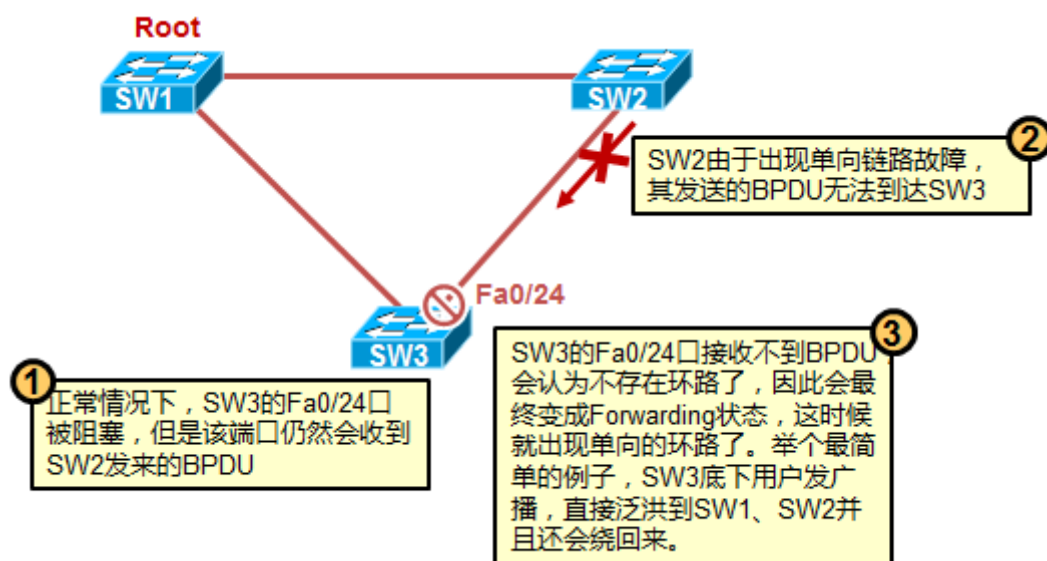


使用 show spanning-tree inconsistentports 命令查看到相关的表项，注意 inconsistent 跟 err-disable 的区别是，err-disable 会 disable 掉整个接口，而 inconsistentport 是针对特定的 VLAN 的。

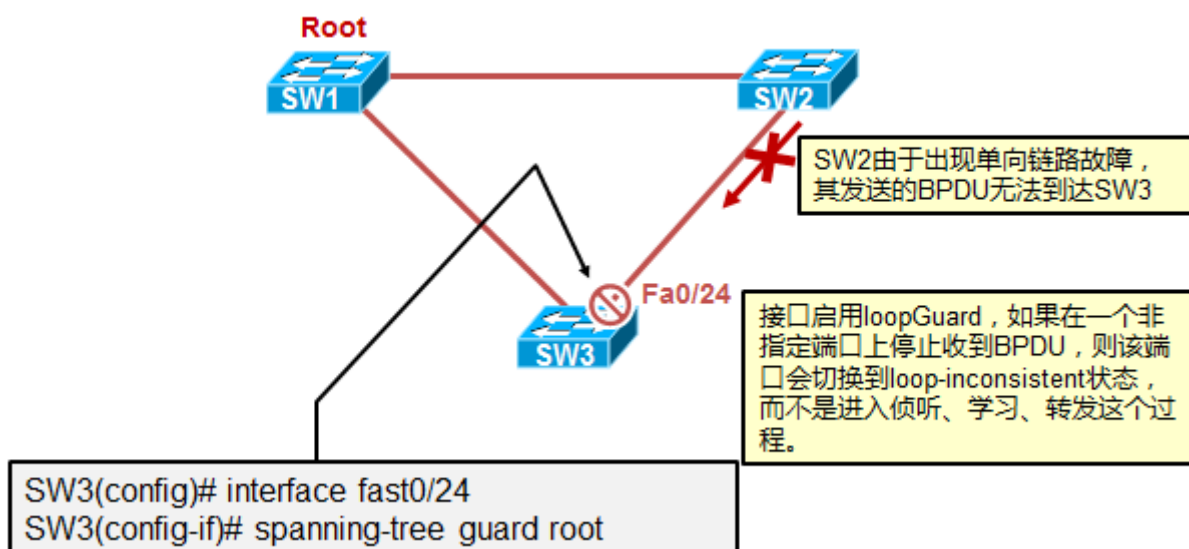
7. LoopGuard

You can use loop guard to prevent alternate or root ports from becoming designated ports because of a failure that leads to a unidirectional link. This feature is most effective when it is enabled on the entire switched network. Loop guard prevents alternate and root ports from becoming designated ports, and spanning tree does not send BPDUs on root or alternate ports.

You can enable this feature by using the **spanning-tree loopguard default** global configuration command.

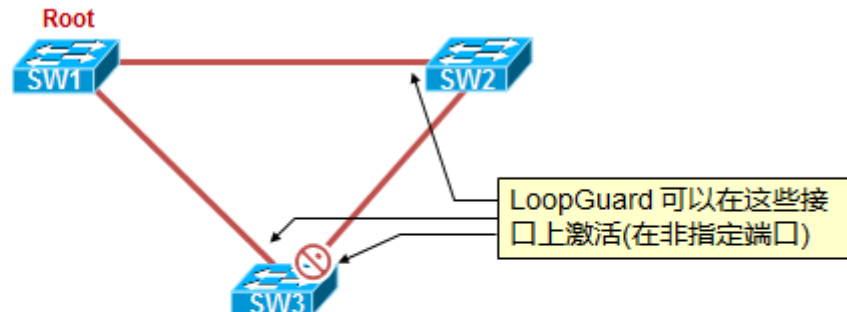


部署 loopguard 后：



注意事项：

- 与 Root Guard
 - LoopGuard 与 RootGuard 是无法同时启用的



```
SW3(config)# spanning-tree global-default loopguard enable
SW3(config-if)# spanning-tree guard root
```

8. UDLD

- 关于 UDLD

用于单向链路检测，主要用于光纤链路。需要链路两端的设备都支持 UDLD

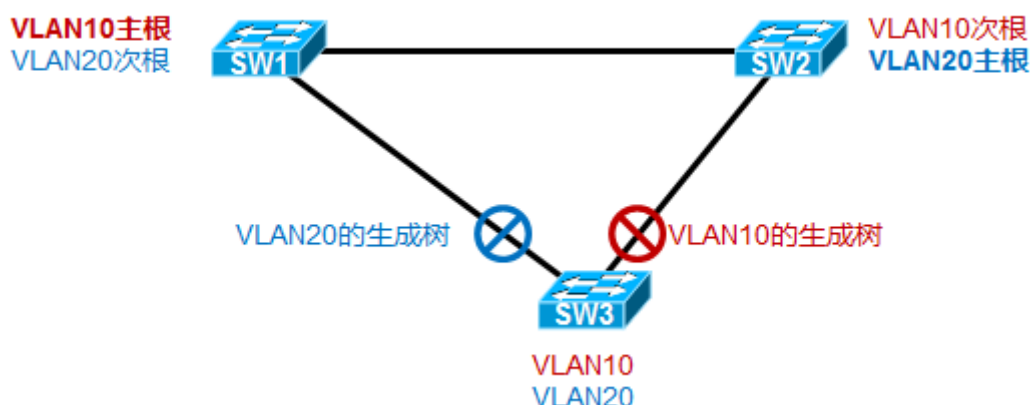
当 UDLD 检测到单向链路故障，它 administratively shuts down 掉这个接口，并且提示用户。

单向链路会引发各种问题，其中包括 spanning-tree 拓扑环路
- UDLD 操作模式

两种操作模式：normal (the default) 和 aggressive

3.4 PVST+

3.4.1 基本概念



传统的 STP，也就是 802.1D，是所有的 VLAN 共用一棵生成树，这样一来的好处是交换机不用耗费过多的资源去计算多棵生成树，但是，缺陷是，通过 block 固定的端口，导致网络中的流量只走一侧，而部分链路一丁点流量都没有，浪费了带宽。

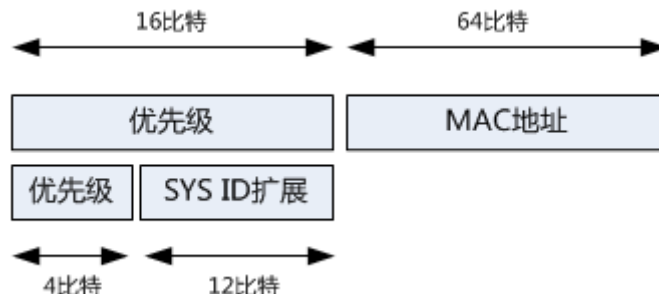
PVST+在这方面做了改进，perVLAN 的意思是，一个 VLAN 对应一棵生成树，如此一来交换机将基于 VLAN 计算生成树，我们可以通过对单独的 VLAN 生成树调节优先级等相关参数，从而影响生成树的计算，最终实现合理的链路带宽利用。例如上图，内网存在两个 VLAN，10 和 20，我们让 SW1 成为 VLAN10 的主根（全网交换机优先级最低），SW2 成为 VLAN10 的次根，SW3 优先级最高，如此一来我们将 BLOCK 掉 SW3 上连接 SW3 的上联口。相对的，VLAN20 也最特定的调节，这样，VLAN10 的用户出去将走左侧链路，VLAN20 的用户将走右侧，链路都得到了合理的运用，实现了负载均衡。

3.4.2 MAC 地址分配和缩减

- CISCO CATALYST 交换机的 MAC 地址池最多可以容纳 1024 个地址，交换机的型号决定了可用 MAC 的数目，并不是所有 catalyst 交换机都能支持到这么多个 MAC
- 这些 MAC 地址作为 VLAN 生成树中的网桥 ID 的 MAC 地址部分。不同的交换机型号支持不同的可用 MAC 地址数目。交换机依照次序分配 MAC 地址
- Show run int | include bia 能看到所有的 MAC，其中第一个 MAC 将被生成树使用，也就是 CPU 的 MAC。接下去就是每个以太网接口的 MAC。
- 我们知道交换机能够支持的 VLAN 的数据是很庞大的，如果开启 PVST+，每个 VLAN 一棵生成树，而没棵生成树都要有一个独立的标识，都需要耗费一个 MAC 的话，那么 MAC 地址池肯定是无法承受的。
- 因此需要使用到 MAC 地址缩减方案

在 PVST+中，一个 VLAN 一棵生成树，在每台交换机上，对于每一棵生成树需要有一个唯一的标示符，也就是网

桥 ID 要唯一，网桥 ID 前面说了，是由优先级和 MAC 地址构成，在 MAC 地址缩减方案中，同一台交换机的所有生成树的桥 ID 中，MAC 地址都使用自己交换机 CPU 的 MAC，同时将 16bits 的优先级进行扩展，变成 4bits 的优先级+12bits 的系统 ID，通过这个系统 ID 来识别不同的 VLAN。在 CISCO IOS 中，这个系统 ID 用的就是 VLAN ID。



如此一来，优先级就成了最高的 4bits，也就有了为什么优先级必须的 4096 的倍数（2 的 12 次方=4096）。另外系统 ID，取值 VLAN ID，例如 vlan10 的生成树，在本交换机的桥 ID 就是“4bits 的优先级+10+交换机的 MAC”SYSID 还可作为 VLAN 或 MST 实例的标识，比如 VLAN100 的系统 ID 扩展是 100

3.4.3 配置

```
Switch(config)# spanning-tree vlan-id
```

- 激活特定 VLAN 的 STP，如需关闭 STP，加 no

```
Switch(config)# spanning-tree vlan vlan-id priority pri
```

- 配置桥优先级

```
Switch(config)# spanning-tree vlan vlan-id root {primary | secondary} [diameter diameter]
```

- 设置主根、次根。用 primary 则在全网默认优先级（都是 32768）的情况下，该交换机生成树 vlan 优先级调整为 24576，如果是 secondary，则为 28672。但是如果网络中存在优先级为最低：4096 的交换机，那么这条命令也没则。也就是实际上这条命令就是检测目前网络中的其他设备的网桥优先级，并且来调整自己的优先级并保证最低，就这么简单。

Diameter 指的是一个直径值，默认 7。这个值关系到 STP 计时器。不同的 diameter 值配置上去，将导致交换机自动调整 hello 及 max age 计时器，这条命令一般在 root 上配置，这样计时器的值会通过 bpdu 传递到其他交换机。

因此其实这就是条宏命令

```
Switch(config-if)# spanning-tree [vlan vlan-id] cost cost
```

- 设置接口 cost

```
Switch(config-if)# spanning-tree [vlan vlan-id] port-priority pri
```

- 设置接口优先级，CISCO IOS 默认 128；CatOS 默认 32

```
Switch(config)# spanning-tree [vlan vlan-id] hello-time sec
```

```
Switch(config)# spanning-tree [vlan vlan-id] forward-time sec
```

```
Switch(config)# spanning-tree [vlan vlan-id] max-age sec
```

- 设置 STP 的 timer

3.5 RSTP (802.1W)

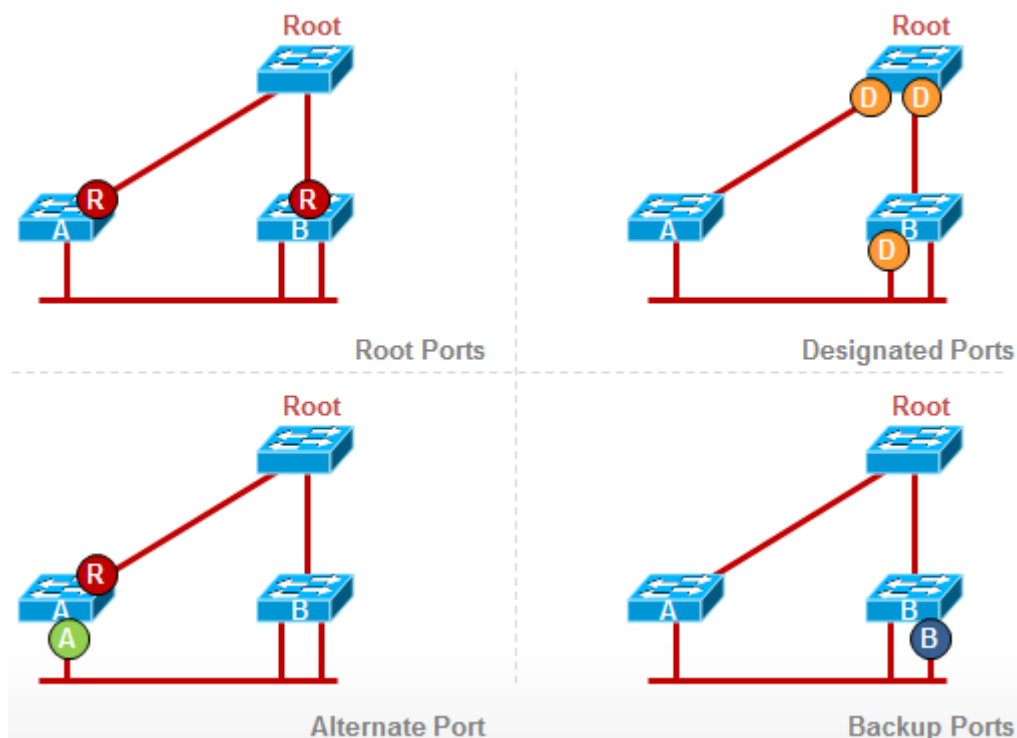
3.5.1 基础知识点

1. RSTP 端口状态

- 丢弃（也就是 802.1D 中的禁用、阻塞、侦听的合并）
- 学习
- 转发

2. 端口角色：

- **根端口：** 收到最优的 BPDU 的接口就是根端口。这是距离 Root 最近的（cost 最小）的接口。
- **指定端口：** 在每一个 segment 上选择一个指定端口，该端口将发送这个 segment 上最优的 BPDU。
- **替代端口：** 丢弃状态。本交换机除了根端口外，其他到根路径的端口，如果活跃的根本端口发生故障，替代端口将成为根端口，所以替代端口可以理解为根端口的可替代者
- **备份端口：** 丢弃状态。指定端口的备份，出现在一台交换机有两个端口连接到同一个共享介质时。
- **禁用端口**



STP 与 RSTP 的端口对比：

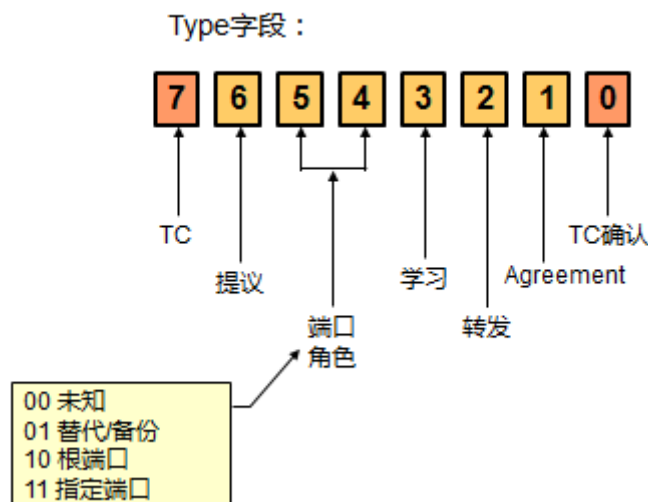
STP端口角色	STP端口状态	RSTP端口角色	RSTP端口状态
Root Port	Forwarding	Root Port	Forwarding
Designated Port	Forwarding	Designated Port	Forwarding
Nondesignated Port	Blocking	Alternative or Backup Port	Discarding
Disabled	-	Disabled	Discarding
Transition	Listening or Learning	Transition	Learning

3.5.2 BPDU 格式和操作

1. 802.1W BPDU 格式

RSTP 只在 802.1D 基础上对 BPDU 做了少量修改：

在 802.1D 中，TYPE 字段只使用了最高位和最低位，来表示 TC 和 TC 确认，RSTP 对该字段进行的扩展：



RSTP BPDUs 的协议是 2，版本是 2。

2. BPDUs 操作

在 802.1D 中，非根交换机只有从根端口收到根桥发送的 BPDUs，自己才能产生 BPDUs。而在 RSTP 中，即使非根交换机没有从根交换机处收到 BPDUs，其自己也以“hello 间隔”为周期（默认 2S）发送 BPDUs。

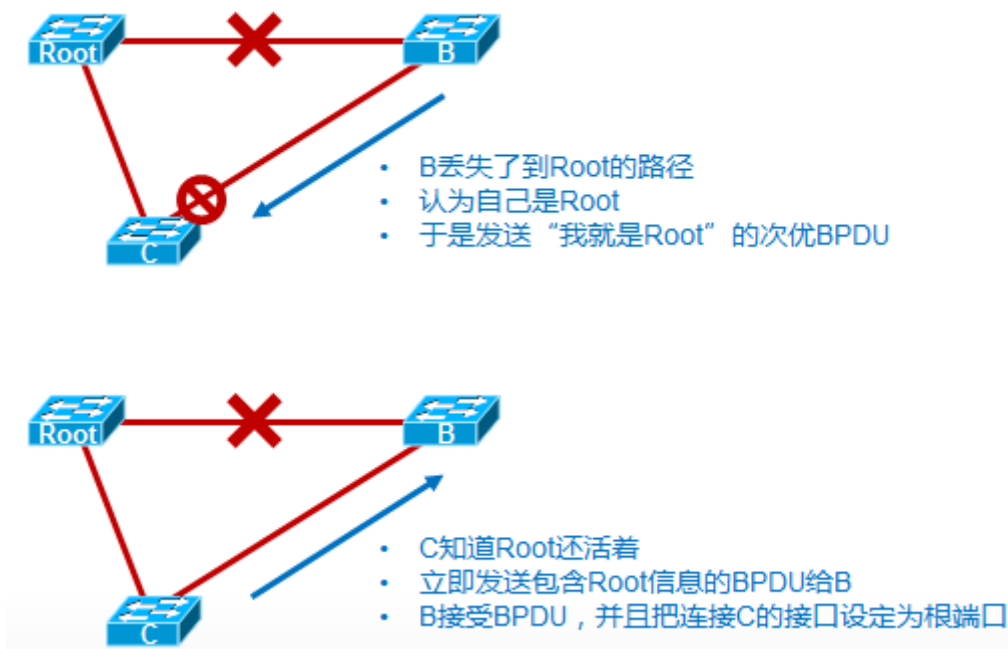
3. Faster Aging of Information

在特定的接口，如果连续三个周期没有收到 BPDUs（或者 max-age 超时），接口上的 STP 协议数据将迅速老化，如此一来，BPDUs 又有点类似交换机之间的 keep-alive 机制。这种快速老化的机制有助于对拓扑变化的快速响应。

4. Accepts Inferior BPDUs

这个机制与 CISCO 的 BackboneFast 特性非常类似。

当交换机从其他指定交换机或根桥收到次优 BPDUs，802.1D 遇到这种情况是首先忽略这些次优 BPDUs，而 RSTP 是立即接受这些次优 BPDUs 同时回传一个更优的 BPDUs。



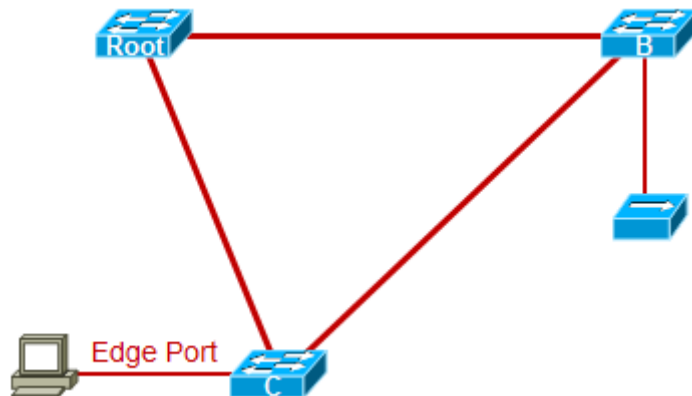
看上图，初始情况下，C 的一个接口被选举为非指定端口被 Block，B 会从指定端口发送 BPDU 给 C。我们先考虑一下 802.1D 的情况，当 Root 及 B 之间的链路故障了，由于 C 上连接 B 的接口被 BLOCK，它不发送 BPDU 给 B，因此，B 此刻认为自己是 Root，于是向 C 发送自己为 Root 的 BPDU。而由于这个时候 C 的接口上还存储着之前 B 发给自己的 BPDU，而这个 BPDU 相比与 B 后来发送给自己的 BPDU 更优，因此 C 直接忽略这些次优 BPDU，一直到 Max-Age 超时，C 的接口上存储的此前 B 发过来的 BPDU 才会老化，这时候接口进入 LST，才开始发送 BPDU，指示根为 Root。而此刻 B 才接受事实，将自己连接 C 的接口置为根端口。

那么对于 RSTP 情况就不一样了，C 在收到次优 BPDU 后，将立即回送自己的 BPDU，好让 B 了解拓扑情况。这个机制跟 BackboneFast 非常类似。

3.5.3 Rapid Transition to Forwarding State

RSTP 的一个重要的改进是端口的快速过渡。传统的 STP 算法在将一个接口过渡到 forwarding 状态之前，需要经历几个计时器。为了获得网络的快速收敛，我们可能会去调整计时器，然而这种方式有可能影响网络的稳定性。RSTP 的设计，使得我们不用依赖调整计时器，并且可以使得接口可以安全的过渡到转发状态。为了实现接口上的快速收敛，RSTP 引入了一些新的概念：

1. 边缘端口 edge ports



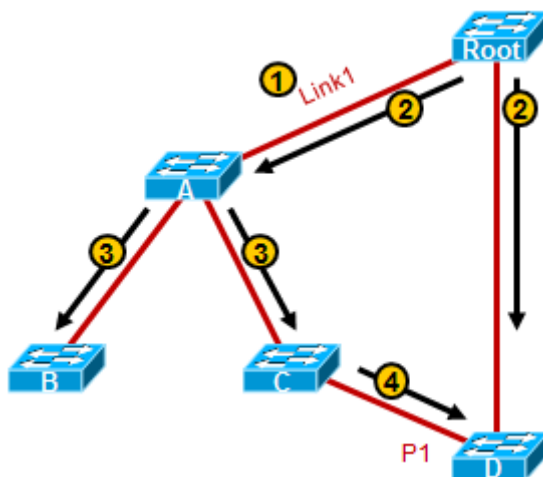
RSTP 定义的这种端口类型与 Portfast 十分类似。因为这些接口用于连接主机，所以一般不会产生环路。这些端口可以跳过 LST 或 LRN 直接过渡到转发状态。并且当这些接口 up down 的时候不会引起拓扑变更。另外，边缘端口一旦收到 BPDU，则立即丢失边缘端口的特征，变成一个普通的 spanning-tree 接口。在 catalyst 交换机上，可以用 portfast 关键字来进行手工配置。

2. 链路类型 Link types

RSTP能够在边缘端口及 point2point 链路上快速过渡。RSTP的链路类型是通过接口的双工状态自动获取的，如果接口是半双工，那么链路类型就是 shared port，如果是全双工，那么就是 point2point。当然，接口的链路类型可以通过命令修改，接口模式下：spanning-tree link-type ?

3. 802.1D 与 RSTP 的收敛对比

- 802.1D 的情形：

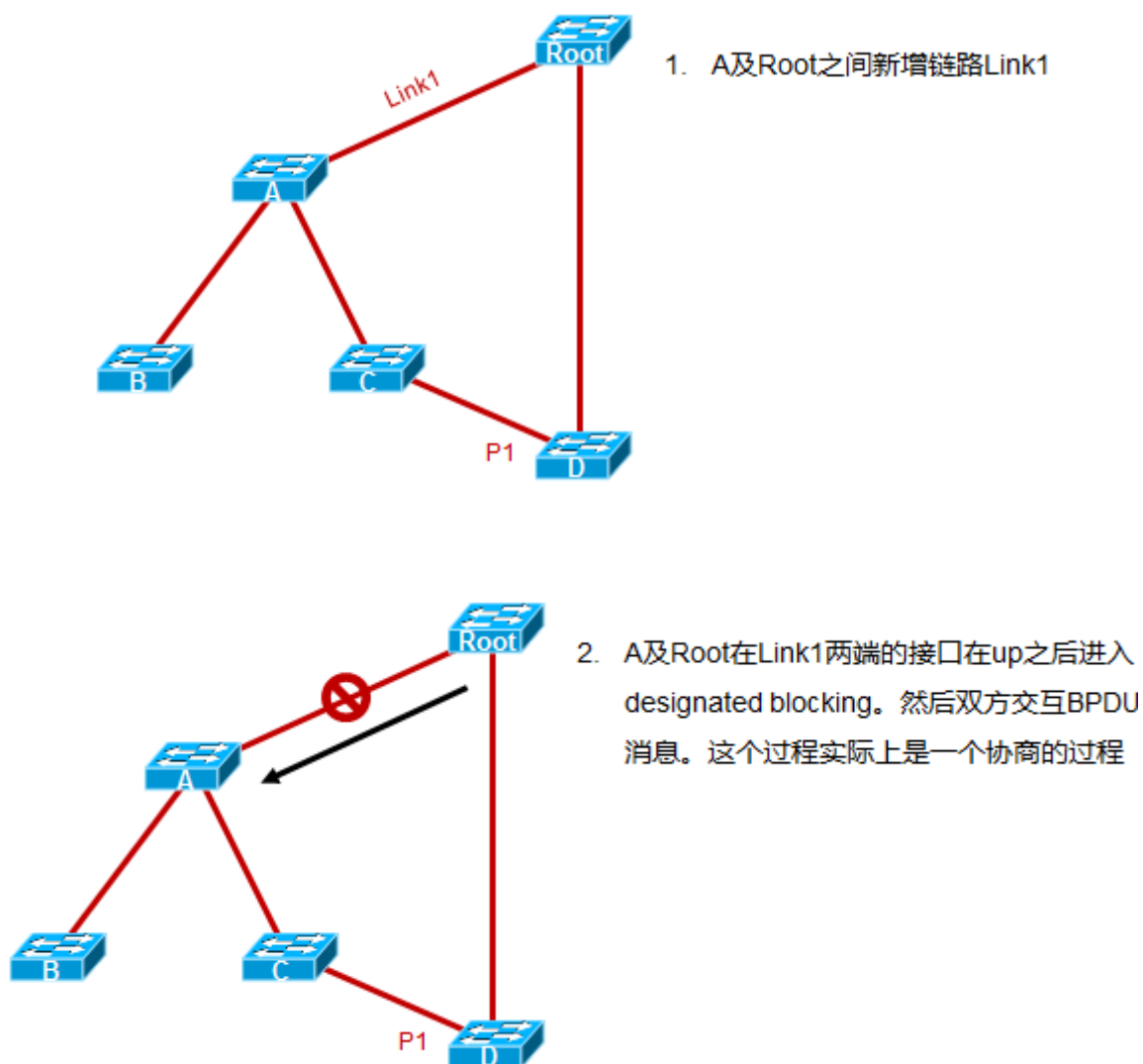


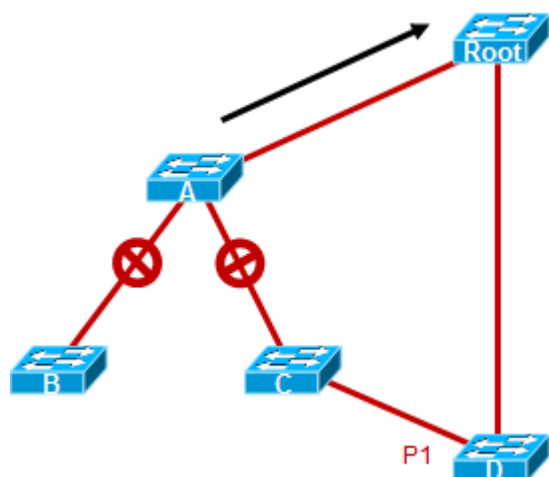
- A 及 Root 之间新增链路 Link1
- A 及 Root 在 Link1 两端的接口都进入 Listening 状态，A 将收到 Root 发出来的 BPDU
- A 将 BPDU 从自己的指定端口发送出去，BPDU 被泛洪到网络中

4. B 和 C 收到这个更优的 BPDU，继续向网络中泛洪
5. 数秒后，D 收到这个 BPDU，Block 掉端口 P1

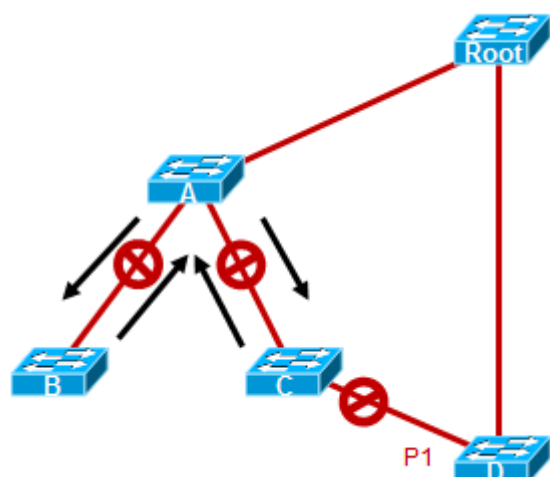
由于缺乏 feedback 机制，A 连接 Root 的接口从 Listening 到 Forwarding，需要经历 $15 \times 2s$ 的延迟。此时 A、B、C 下联的用户流量就出现问题了（因为 D 在收到更优的 BPDU 后，将 P1 口 block 了，这时候 ABC 相当于在 A 的根端口过渡到 forwarding 之前都处于网络的“隔离地带”）

• RSTP 的情形：

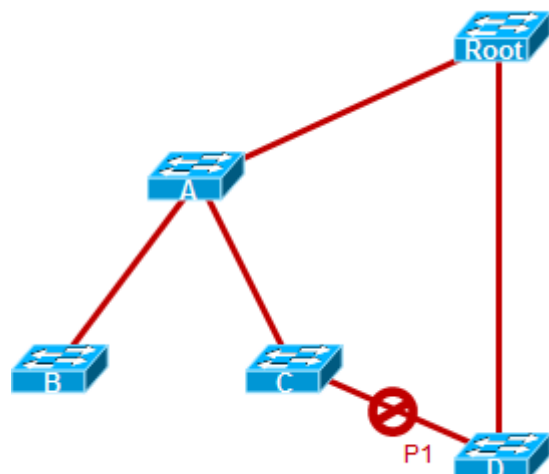




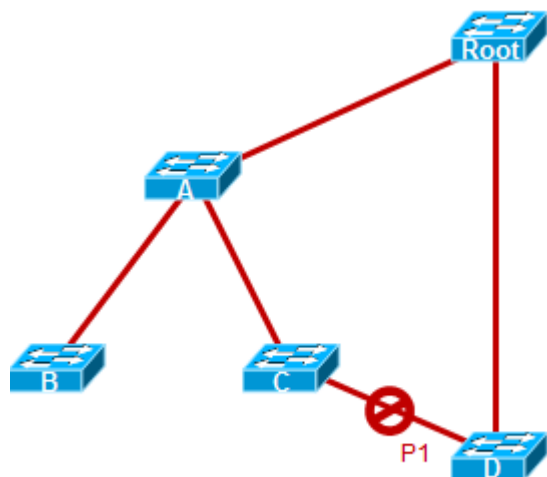
3. A在收到Root发送的BPDU后，将自己的所有非边缘端口Block（这个过程称为同步 sync），并且回送一个agreement消息给Root
4. 在此之后，Root及A在link1上的端口立即都过渡到转发状态。而网络目前是没有环路的，A往下的网络目前是切断的



5. A与B和C之间，开启新一轮的协商，BC收到A发送的BPDU后，完成同步Sync过程，将自己的非边缘端口BLOCK掉，然后都向A回送agreement消息。同时，ABC互联的接口进入转发状态。在BC同步操作过程中，B下联全是主机，因此没有端口被Block（已经完成同步）；而C要BLOCK掉连接D的端口。



7. 完成上一步之后，生成树状态如图：
8. 最终BPDU到达D，D将P1口Block掉



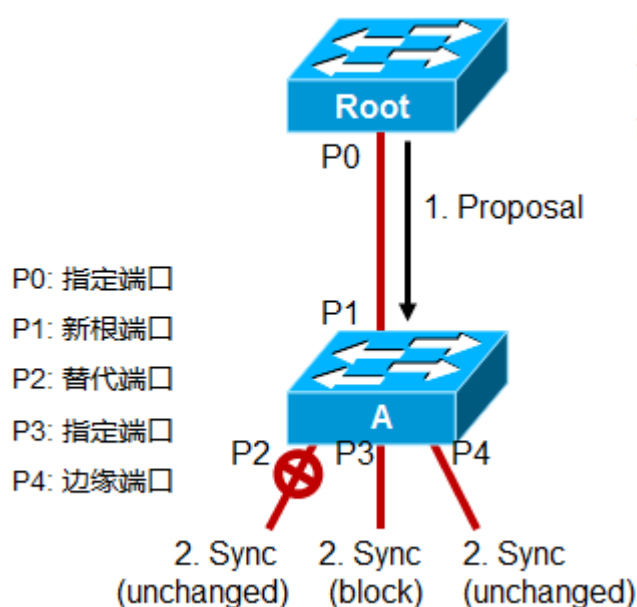
在RSTP收敛过程中，耗费的时间仅仅是BPDU从Root泛洪到网络末端的时间，不用受到任何Timer的限制，直接绕过两个转发延迟时间。因此收敛速度更快。

有两点需注意：

- 交换机之间的这种协商机制只在P2P链路上被执行
- 边缘端口的配置非常重要，如果配置不当，有可能会在同步过程中被BLOCK。

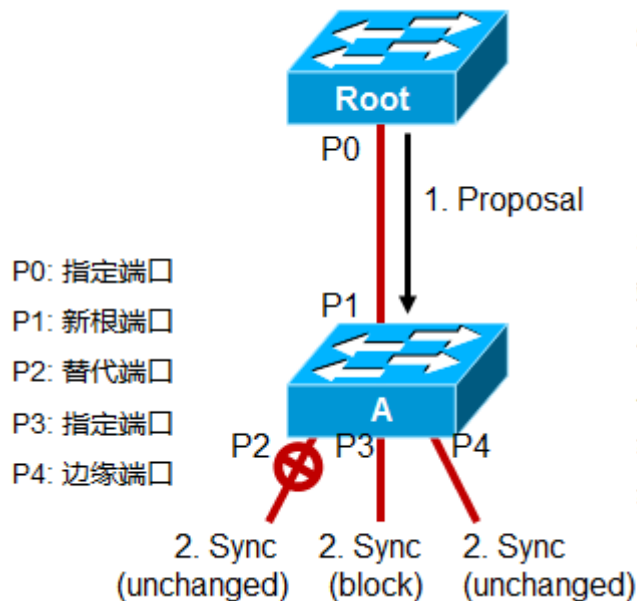
4. Proposal/Agreement Sequence

对于 802.1D 来说，当一个端口被选举为指定接口，它从 blocking 到 forwarding 至少需要 30S 的时间。然而在 RSTP 中，proposal/Agreement 机制使得接口能够在几秒内完成迅速、可靠的过渡。



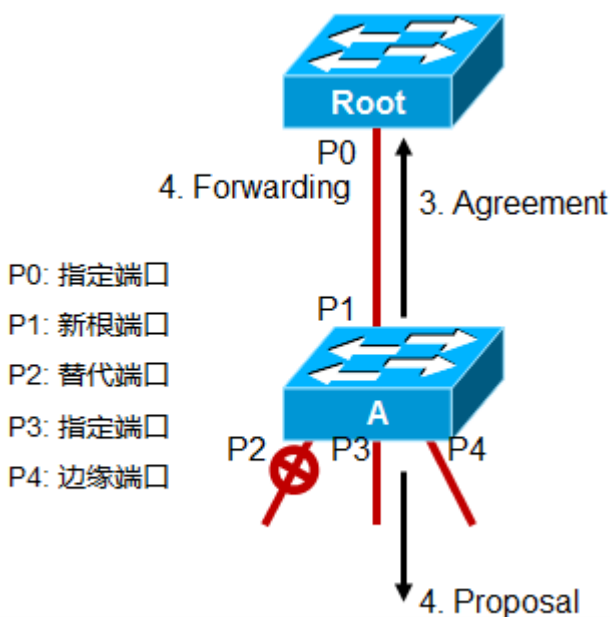
Root和A之间新增了一条链路，链路两端的接口在收到对方发送的BPDU前是designated blocking状态

1. 当一个被选举为指定端口的接口在discarding或learning状态（且只在这个状态），它在其发送的BPDU中进行proposal bit置位。这就是步骤1的P0的情况



2. A收到一个最优的BPDU，它立刻知道P1就是新的根端口。接下去A启动一个同步进程sync，A的所有端口将会促发这个进程。

当一个接口为Blocking状态或为边缘端口，则该接口已同步。而如果接口不满足上述两个条件，它将被block并且进入discarding状态已达同步。在步骤2中，P2、P3、P4都已经完成同步了。



3. 在A完成同步后，A就可以将新选出的根端口unblock并且发送一个agreement消息给Root。这个agreement消息是A的proposal消息的拷贝，但是agreement bit置位了。

4. 如此一来，P0就收到了应答，立即转为forwarding。注意这时候P3接口仍处于designated discarding状态，于是它向它的邻居网桥去发送proposal，而且也在积极等待回传的agreement以便进入forwarding状态。

小结：

- proposal agreement 的机制是非常快速的，因为它们不用受限于任何计时器，这个握手机制会迅速的蔓延到整个交换网络末梢，并且能在拓扑发生变更的时候迅速收敛。
- 如果一个 designated discarding 接口在发出 proposal 后 没有收到 agreement ,它将慢慢的过渡到 forwarding 状态，这个过程是传统的 802.1D 的 listening-learning-forwarding 过程。这种情况有可能发生在对端交换机不理解 RSTP BPDUs 或者对端交换机的端口被 block 的情况。

3.5.4 拓扑变更机制

1. 拓扑变更机制（检测）

- 在 RSTP 中，只有当非边缘端口过渡到 forwarding 状态才会触发拓扑变更。也就是说，一个端口如果丢失了连接，则不再认为是一次拓扑变更，这与 802.1D 是有区别的。
- 当一个 RSTP 交换机检测到一次拓扑变更它将：
 - 为根端口及所有的非边缘指定端口启动一个 TC while timer，timer 的值等于 2 倍的 hello-time 计时器
 - 向上述端口泛洪 MAC 表
 - 注意主要 TC while timer 在端口上计时，端口发送出去的 BPDU 就会进行 TC bit 置位，该 BPDU 也会从根端口往外发送

2. 拓扑变更机制（传递）

当一台 RSTP 交换机收到 TC bit 置位的 BPDU，它将：

- 1) 清除从所有接口学习到的 MAC 表项，除了收到 TC BPDU 的那个接口。这个动作虽然有可能导致网络中存在短暂的突发性泛洪，但是也有利于刷新 CAM 表、清除 stale 表项。
- 2) 激活 TC while timer 然后从所有的非边缘指定端口及根端口往外发送 TC 置位的 BPDU。通过这种方式，拓扑变更信息会迅速在网络中泛洪。

通过这种方式，TC 消息会被迅速的泛洪到整个网络。而不用像 802.1D 哪样，把消息传递到 root，再由 root 来同时拓扑变更。

4 三层交换

4.1 CAM 及 TCAM

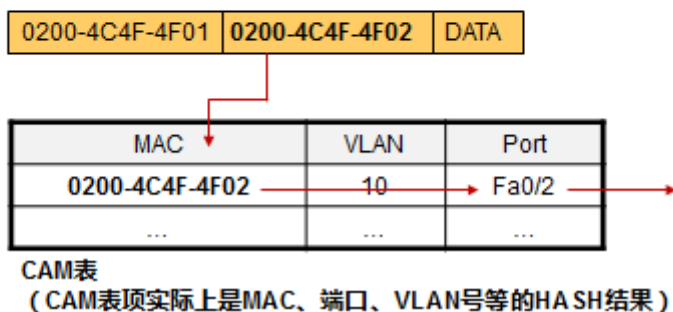
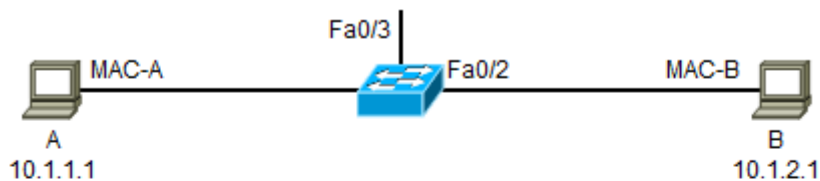
1. CAM 表

CAM 是交换机用于二层交换时所查的表

- a) Content Addressable Memory Table 内容可寻址内存
- b) 在查表时，使用二进制的 0、1 位进行匹配，并且需严格匹配，也就是目的 MAC 与 CAM 表中的 MAC 需

完全匹配

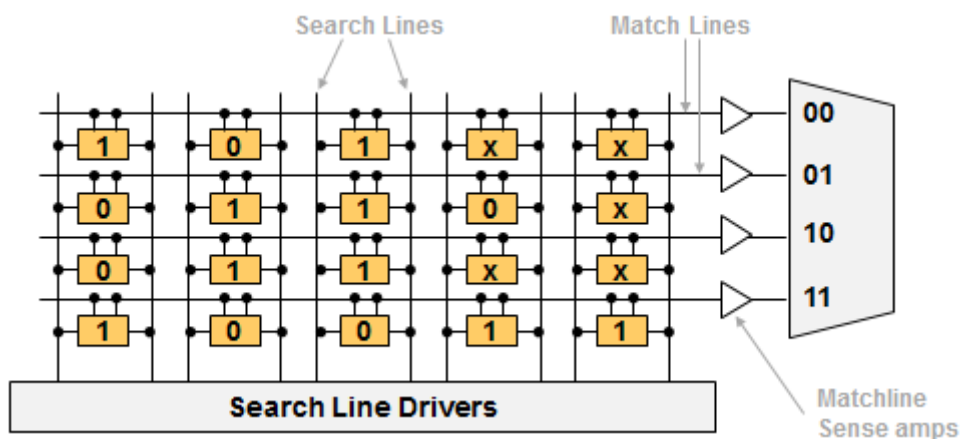
- c) 查找的结果如果发现完全匹配项，则根据返回的端口号将数据转发出去



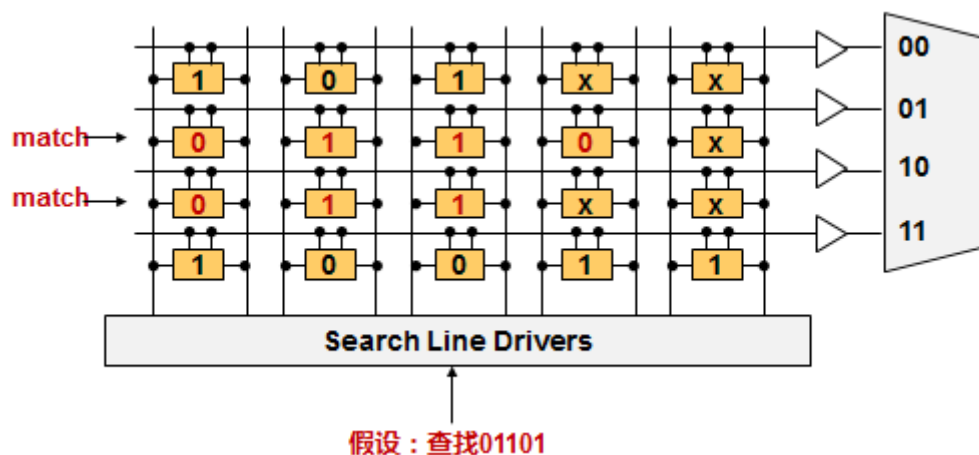
2. TCAM 表

TCAM 是路由模块或路由器用于三层转发时所查的表

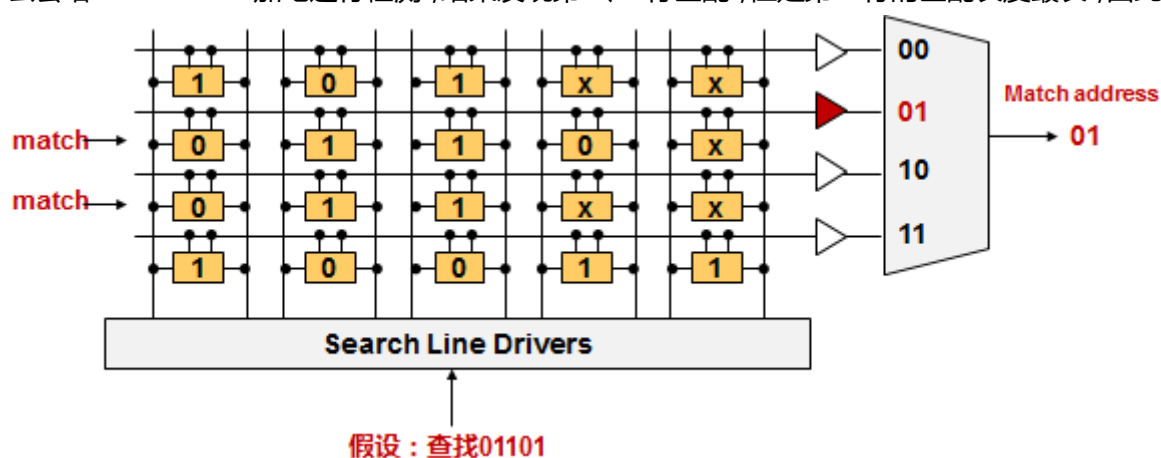
- a) 匹配表项中的 0、1、x (x 为无所谓)
b) 最长匹配原则



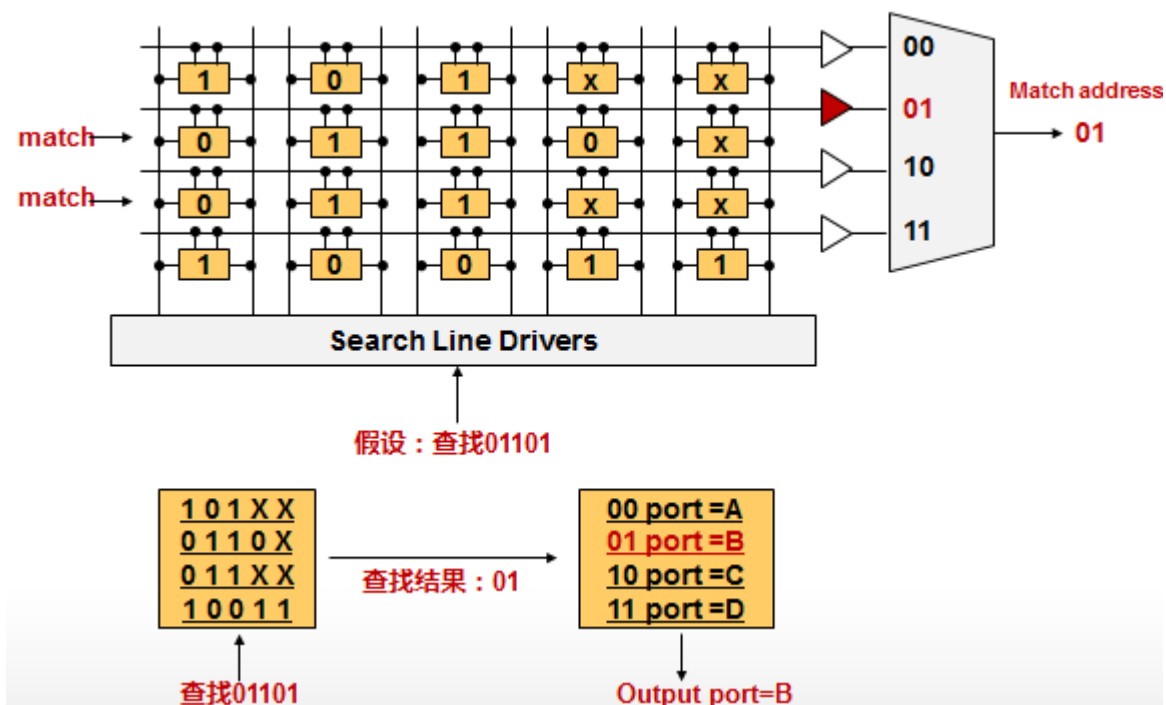
将所有路由条目+掩码放进 TCAM 表项，就是图中的“行”，二进制的 0 和 1 都要匹配，x 为无所谓。
当我查找一个目的地的时候，例如查找“01101”：



那么会给 search lines 加电进行检测，结果发现第 2、3 行匹配，但是第 2 行的匹配长度最长，因此最终结果：

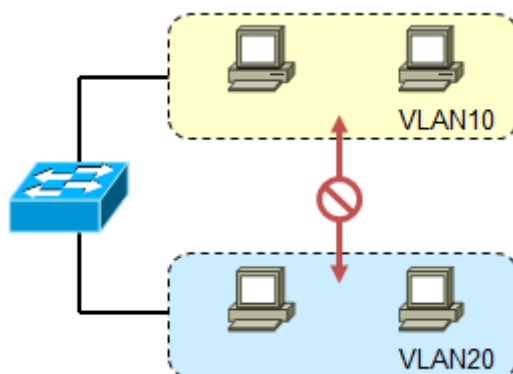


查找出来的结果，可以理解为一个指针，用于找到关联的出接口：

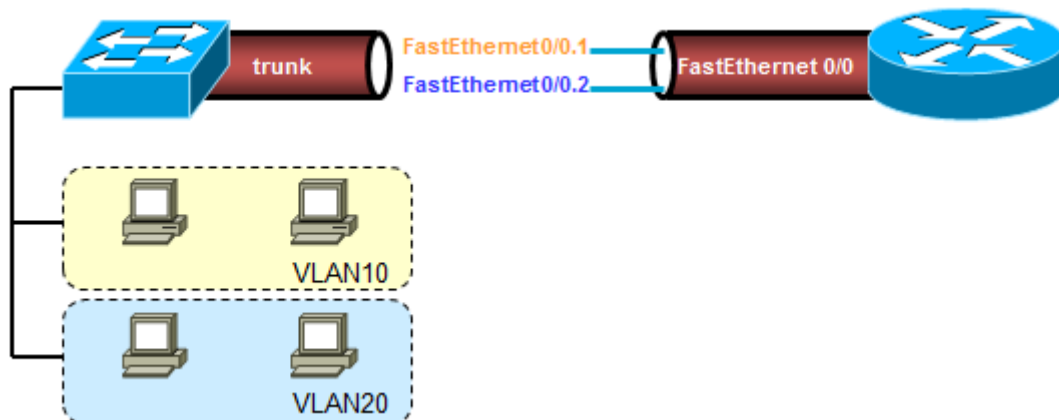


4.2 VLAN 间路由

4.2.1 单臂路由



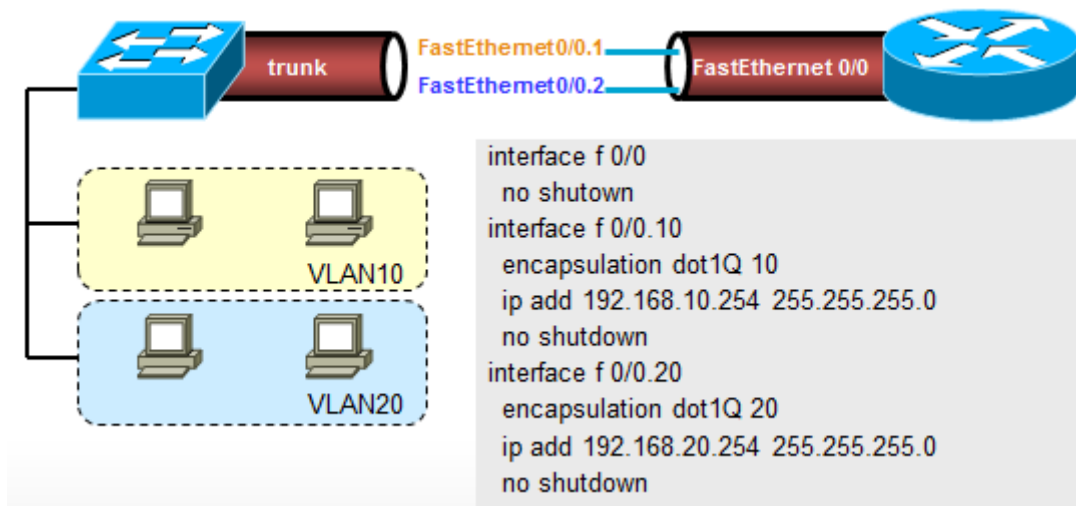
我们知道，在二层交换环境下，一个 VLAN 就是一个广播域，不同 VLAN 不同的广播域，一般也是不同的逻辑子网，而且相互隔离，互相是无法互访的，这样能起到隔绝广播的作用。但是实际网络中往往 VLAN 有互访的需求，例如同一家公司不同的部门划分在不同的 VLAN，那么如果这些部门之间有数据往来的需求呢？那么二层交换机就无法实现了，需要借助三层设备。一个最简单的方法，就是使用路由器，用单臂路由的解决方案：



所谓单臂路由，就是在路由器的以太网口上（必须是 100M 接口以上），来承载 VLAN 流量，让路由器和交换机跑一个 Trunk，使用 dot1Q 的封装，这时候，为了让路由器的以太网口支持 Dot1Q 及识别并承载 VLAN 流量，那么需对物理接口进行子接口的划分，如上图。

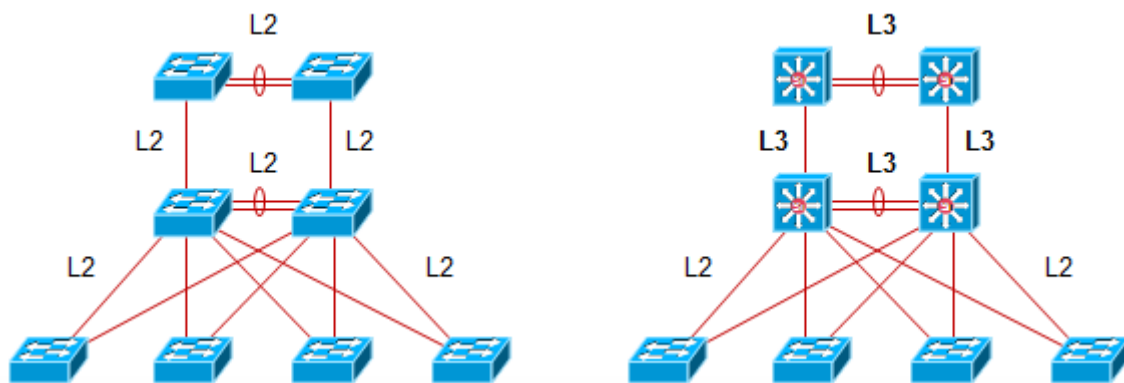
将 Fa0/0 接口划分成两个子接口，并且配置封装协议 Dot1Q，同时为流量分别打上 VLAN 的 tag，这样一来，交换机和路由器之间就起了一个 trunk。路由器的这两个子接口分别配置两个 VLAN 的网关 IP，作为 VLAN 用户的网关。

配置方式如下：



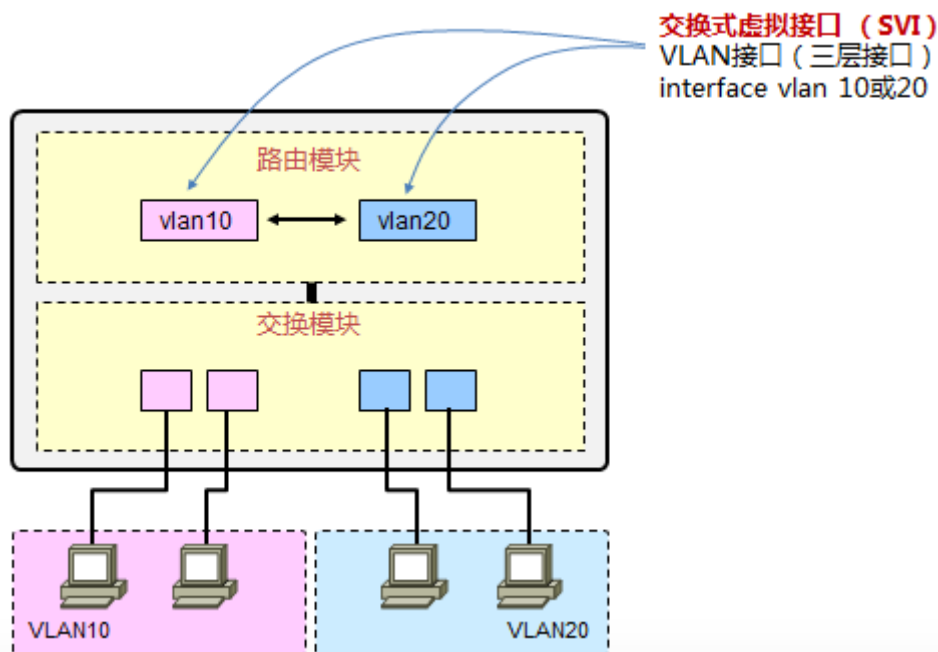
使用单臂路由确实能解决 VLAN 间数据互访的问题，但是却也存在种种弊端，例如路由器的接口，负载太大，流量需二次进出接口或链路，导致干道链路负载也过重，而且扩展性特别差。

4.2.2 路由 VS. 交换的园区网架构



- 在过去，交换是基于硬件的转发，而路由是基于软件的转发，因此园区网络更多的采用交换网络的设计
- 而如今，路由已经几乎与交换一样快，也能够基于硬件做转发，与此同时路由的设计很好的解决了交换网络的二层环路问题，以及 LAN 的隔离问题

4.2.3 Switch Virtual Interfaces (SVI)

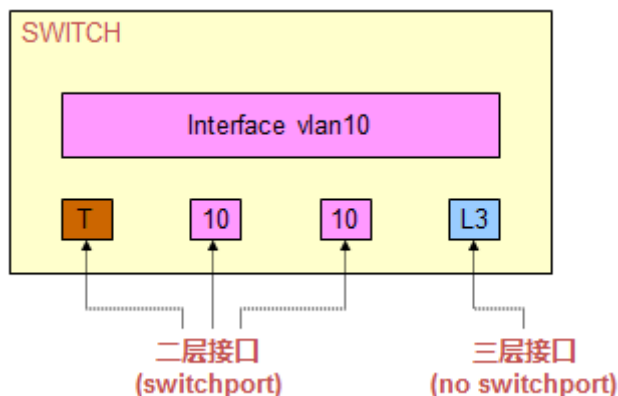


OK，在理解了单臂路由之后，我们再来看看三层交换机是如何实现 VLAN 间的数据互访的，我们从这里为切入点，开始理解并部署三层交换。我们知道二层交换机是可以实现二层交换的，它看的是数据帧，对帧头的二层信息进行读取并且根据自己的 CAM 表进行转发。而三层交换机相当于在二层交换机的基础上，多了个路由模块，于是乎它就能支持路由功能了：支持路由选择协议、支持三层数据的转发、支持 IP 路由查找、支持三层接口等等。

先来认识一下第一种三层接口：SVI 交换式虚接口，SVI 的一个逻辑接口，也就是说不是一个物理接口，当我们在交换机上创建了一个 VLAN 之后，紧接着就可以创建一个与这个 VLAN 对应的 SVI 接口，例如我们创建了 VLAN10，那么 VLAN10 对应的 SVI 接口就是 interface vlan10 或者叫 SVI10，这个 SVI10 是一个三层接口，你可以为这个 SVI 口配置 IP 地址，与 VLAN10 内的 PC 用户的 IP 地址同一网段，那么这样一来，VLAN10 内的用户就能够将网关指向这个 SVI 接口，当 VLAN10 的 PC 需要访问本网段以外的网络时他们将数据交给网关，也就是 SVI10，再由 SVI 去做路由查找及数据转发。实际上，在这个理解过程中，我们可以拿单臂路由那个模型对类比。

所以看上面这图，在三层交换机上创建了两个 VLAN：10 和 20，同时为两个 VLAN 的 SVI 分配了地址作为各自 VLAN 的用户网关，这样一来，这台交换机的路由表里就有了两个 VLAN 网段的路由。那么当两 VLAN 之间要互访时，VLAN10 的用户将数据丢给自己的网关，也就是 VLAN10 的 SVI，数据到了 SVI10 之后，三层交换机查表，发现目的地是 VLAN20 的所在网段，因此将数据从 VLAN20 扔出去，最终抵达目的地的 VLAN20 的 PC。

4.2.4 三层交换机的各类端口



总的来说，三层交换机包含两类端口：二层接口 L2 和 三层接口 L3

- 二层接口(switchport)：access 模式、trunk 模式
- 三层接口：路由接口 (no switchport 或称为 routed port)、SVI 接口

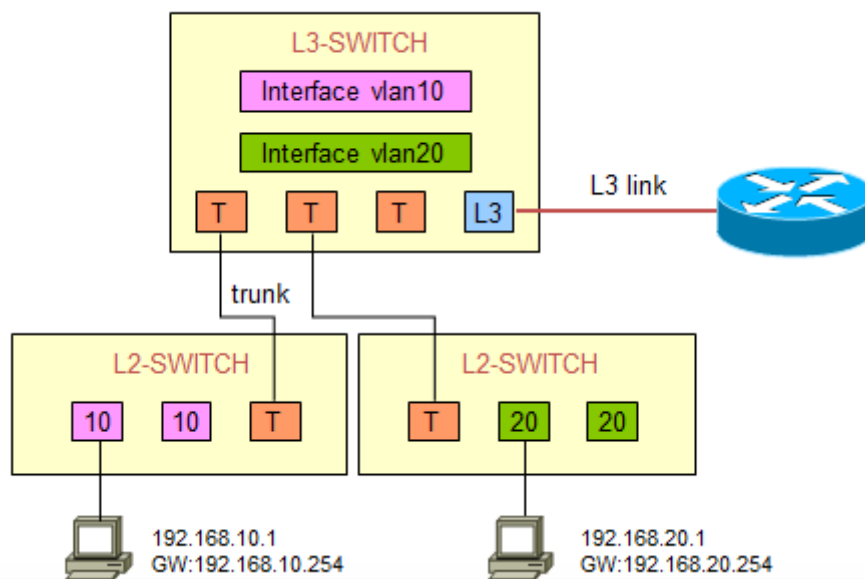
其中二层接口包含我们非常熟悉的 access mode 和 trunk mode。交换机的所有物理接口默认是二层接口，也就是 switchport。

三层接口有两种，一种是 SVI 上面我们已经讨论过了，另一种是 routed port，或者叫 no switchport。注意 SVI 口是一个虚拟接口，而 routed port 是物理接口。三层交换机支持将物理接口变成一个类似路由器物理接口的三层接口，具体的配置就是进入特定接口的配置模式后，使用 no switchport 命令，该接口就变成了一个 L3 的路由口，你可以给他配置 IP，就像操作路由器的一个以太网口那样来操作它。

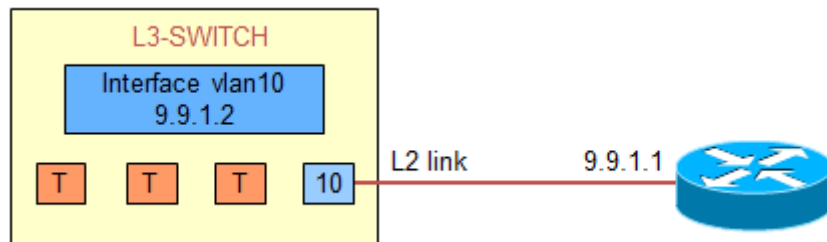
正是由于三层交换机支持这么多种类型的接口，使得三层交换网络的部署更加的灵活和可扩展。

下面我们来看一个三层交换部署的简单案例：

- 综合示例 1：



- 综合示例 2 :



4.3 三层交换机基本配置

```
Switch(config)# ip routing
```

- 开启三层交换机的路由功能

```
Switch(config)# vlan 10
```

```
Switch(config-vlan)# name Classroom
```

- 创建 VLAN

```
Switch(config)# interface vlan 10
```

```
Switch(config-if)# ip address 192.168.10.254 255.255.255.0
```

```
Switch(config-if)# no shutdown
```

- 配置 VLAN 对应的 SVI 接口

```
Switch(config)# interface fast 0/1
```

```
Switch(config-if)# no switchport
```

```
Switch(config-if)# ip address 192.168.255.1 255.255.255.0
```

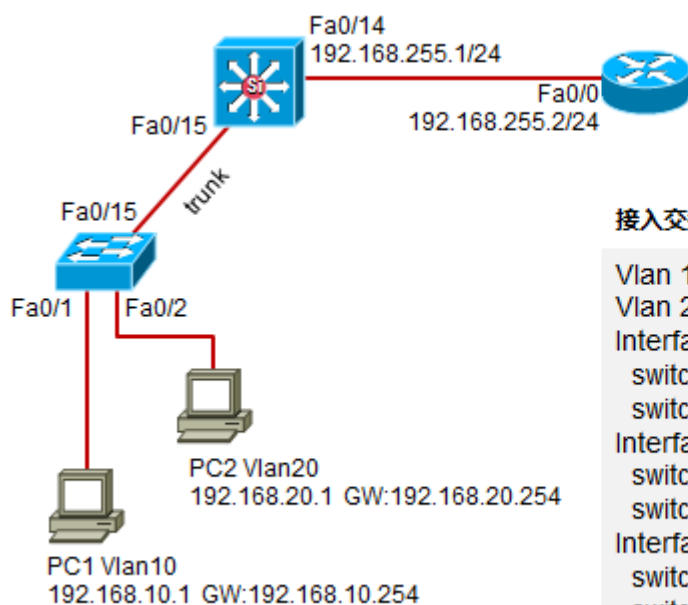
```
Switch(config-if)# no shutdown
```

- 配置 no switchport (routed port) 三层接口

```
Switch(config)# ip route 0.0.0.0 0.0.0.0 192.168.255.2
```

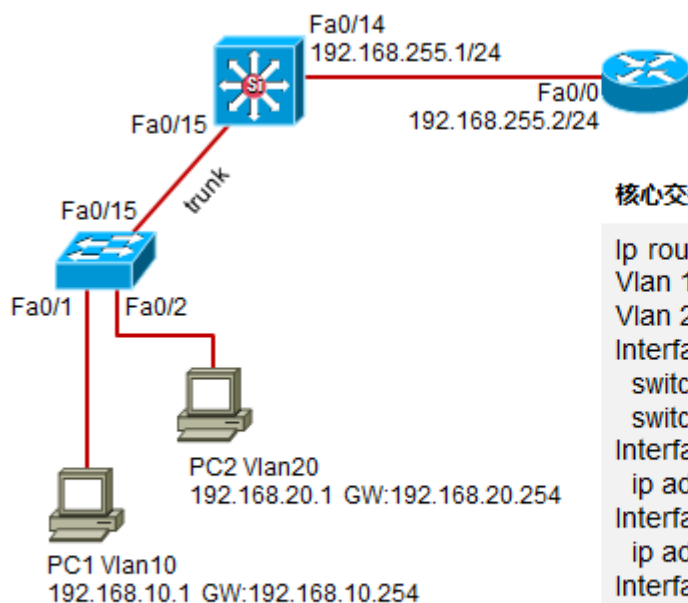
- 配置静态路由

配置示例：



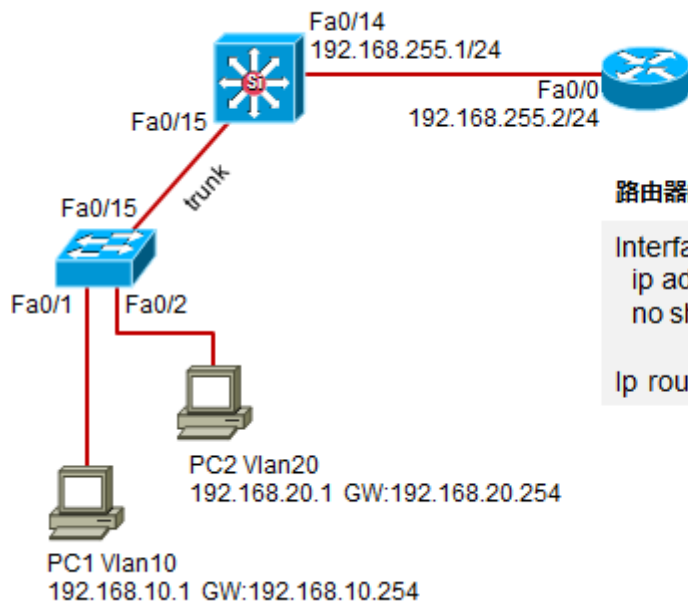
接入交换机的配置如下：

```
Vlan 10
Vlan 20
Interface fast0/1
switchport mode access
switchport access vlan 10
Interface fast0/2
switchport mode access
switchport access vlan 20
Interface fast0/15
switchport trunk encapsulation dot1q
switchport mode trunk
```



核心交换机的配置如下

```
Ip routing
Vlan 10
Vlan 20
Interface fast0/15
switchport trunk encapsulation dot1q
switchport mode trunk
Interface vlan 10
ip address 192.168.10.254 255.255.255.0
Interface vlan 20
ip address 192.168.20.254 255.255.255.0
Interface fast0/14
no switchport
ip address 192.168.255.1 255.255.255.0
Ip route 0.0.0.0 0.0.0.0 192.168.255.2
```



路由器的配置

```
Interface fast0/0
ip address 192.168.255.2 255.255.255.0
no shutdown

ip route 192.168.0.0 255.255.0.0 192.168.255.1
```

4.4 交换网络的管理

4.4.1 二层交换机的管理 VLAN

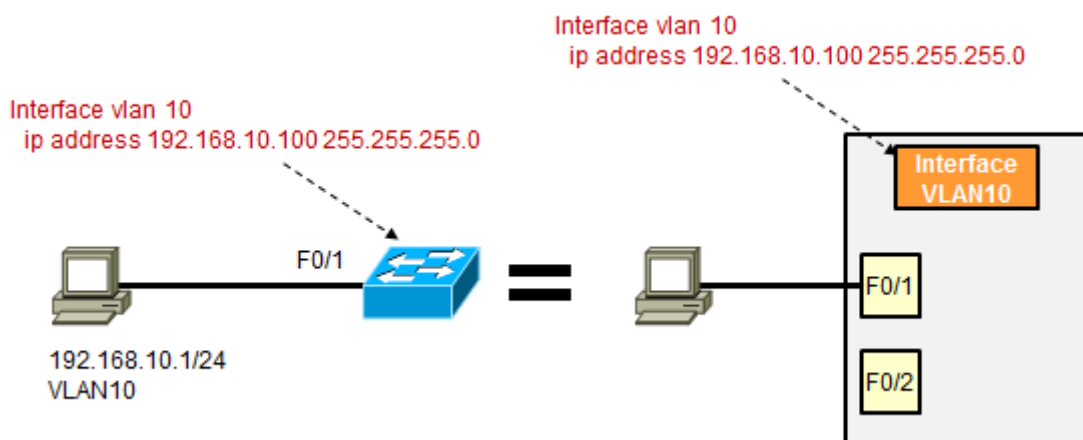
我们知道，在一个园区网中，数量最多的设备，一般而言应该是交换机（二层及三层交换机），其中又以二层交换机的数量居多，二层交换机在典型的三层网络结构中处于“接入层”，主要的任务是为终端的 PC 和用户提供接入，同时划分 VLAN 隔离广播域，再者运行 STP 来提供二层的防环机制。一个中小型的园区网，二层交换机的数量往往是上百台，这些设备在刚刚在客户现场被拆箱后，一般是由工程师现场用 Console 线缆一台台的调试的（不要惊讶，笔者的记录是一个下午的时间，个人完成近 100 台交换机的调试任务，当然，有集成商的兄弟帮忙安放设备、加点、贴标签）。这些交换机在调试好之后，就会被安装到客户现场的各个机房或者弱电间去，一切妥当后就正式上线运行了。

那么这就有一个问题，在设备上线后，如果我们要变更设备的配置、要管理这些交换机怎么办？难道要拿着笔记本电脑带着 console 线下到机房去现场调试么？这种屌丝级的方法完全不可理喻嘛，对了，可以通过 telnet 或者其他方式来远程管理。路由器上的 telnet 我们已经很熟悉了，只要是三层可达，就能 telnet 到路由器，那么接下去我们来看看，如何管理交换机。

这里我们讲的是二层交换机，大家都知道，二层交换机是无法识别三层报文的，它压根不会去看三层的 I 头，但是这不影响二层交换机自己拥有一个 IP 地址。在路由器上，我们是给路由器的物理接口配置 IP 地址，而二层

交换机，我们是在 VLAN 接口上配置 IP 地址，VLAN 接口我们也称为 SVI 交换式虚拟接口，是跟 VLAN 对应的一个逻辑的、虚拟的接口。一台二层交换机，只能够给一个 VLAN 接口分配 IP 地址。

考虑一个最简单的模型：

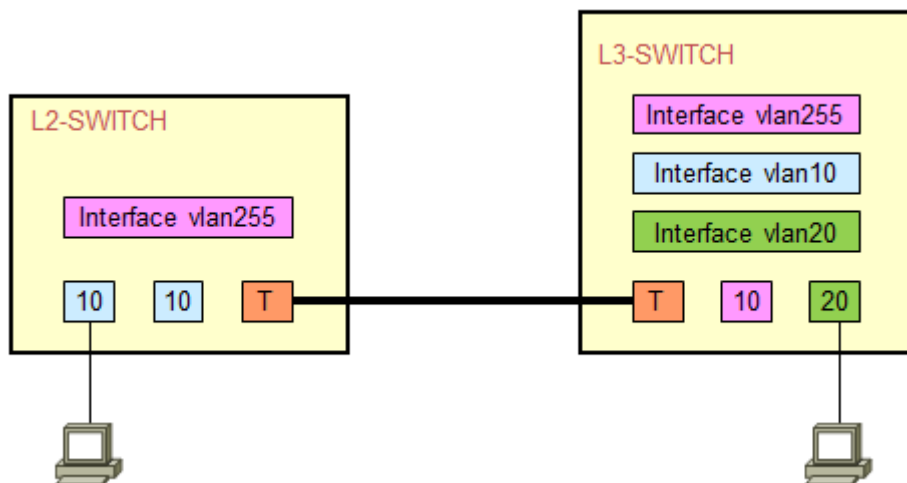


如上图左侧，PC 要想 Telnet 交换机，首先 PC 要能 ping 通交换机，其次交换机上要激活 VTY 并配置密码。那么我们在二层交换机上创建一个 VLAN10，将 F0/1 口划入 VLAN10，同时给二层交换机的 VLAN10 的逻辑接口配置一个 IP，与 PC 在同一个网段的 IP。这样一来，PC 就能访问到交换机了。可是问题来了，这样一来 PC 与交换机就在同一个网段，同一个 VLAN 了，万一下面有 PC 配置的 IP 地址与交换机有冲突那可就麻烦了，因此我们可以考虑给交换机划分一个单独的 VLAN，用于管理这些交换机，这个 VLAN 适用于整个交网络，统一的 VLAN 统一的 IP 规划，它就是管理 VLAN，因此管理 VLAN 并不是一个特定的 VLAN，更不是 VLAN1，这是许多人的误解。一般情况下，我们会使用一个较为“生僻的”VLAN ID 和 IP 编制，例如本实验中的 VLAN255，以及网段 192.168.255.0/24。

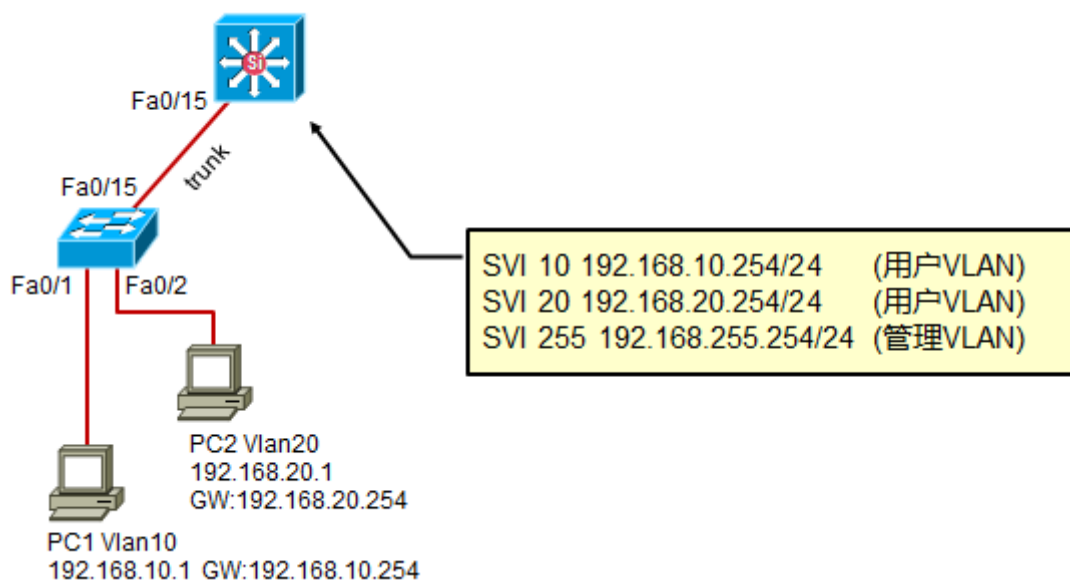
问题又来了，给交换机弄一个单独的设备管理 VLAN 固然可以起到与用户 VLAN 隔离的作用，但是这样一来用户不就无法访问到交换机了么？这就需要借助三层设备—例如路由器或者三层交换机了。与此同时，由于二层交换机没有路由功能，无法像路由器那样拥有一个 IP 路由表，因此，你还需给交换机配置一个默认网关，就像你用路由器模拟 PC 那样哦亲。

关于具体的三层交换机管理 VLAN 的实验，请见《CCNA 实验手册 By 红茶三杯》

4.4.2 三层交换机环境下的管理 VLAN



配置示例：



接入交换机的配置：

```
Switch(config)# vlan 10
Switch(config)# vlan 20
Switch(config)# vlan 255
Switch(config)# interface fast0/1
Switch(config-if)# switchport access vlan 10
Switch(config)# interface fast0/2
Switch(config-if)# switchport access vlan 20
Switch(config)# interface fast0/15
```

```
Switch(config-if)# switchport mode trunk
Switch(config)# interface vlan 255
Switch(config-if)# ip address 192.168.255.1 255.255.255.0
```

Switch(config)# ip default-gateway 192.168.255.254 !!为接入交换机自身设置的网关

核心交换机的配置：

```
Switch(config)# vlan 10
Switch(config)# vlan 20
Switch(config)# vlan 255
Switch(config)# interface fast0/15
Switch(config-if)# switchport mode trunk
Switch(config)# interface vlan 10
Switch(config-if)# ip address 192.168.10.254 255.255.255.0
Switch(config)# interface vlan 20
Switch(config-if)# ip address 192.168.20.254 255.255.255.0
Switch(config)# interface vlan 255
Switch(config-if)# ip address 192.168.255.254 255.255.255.0
```

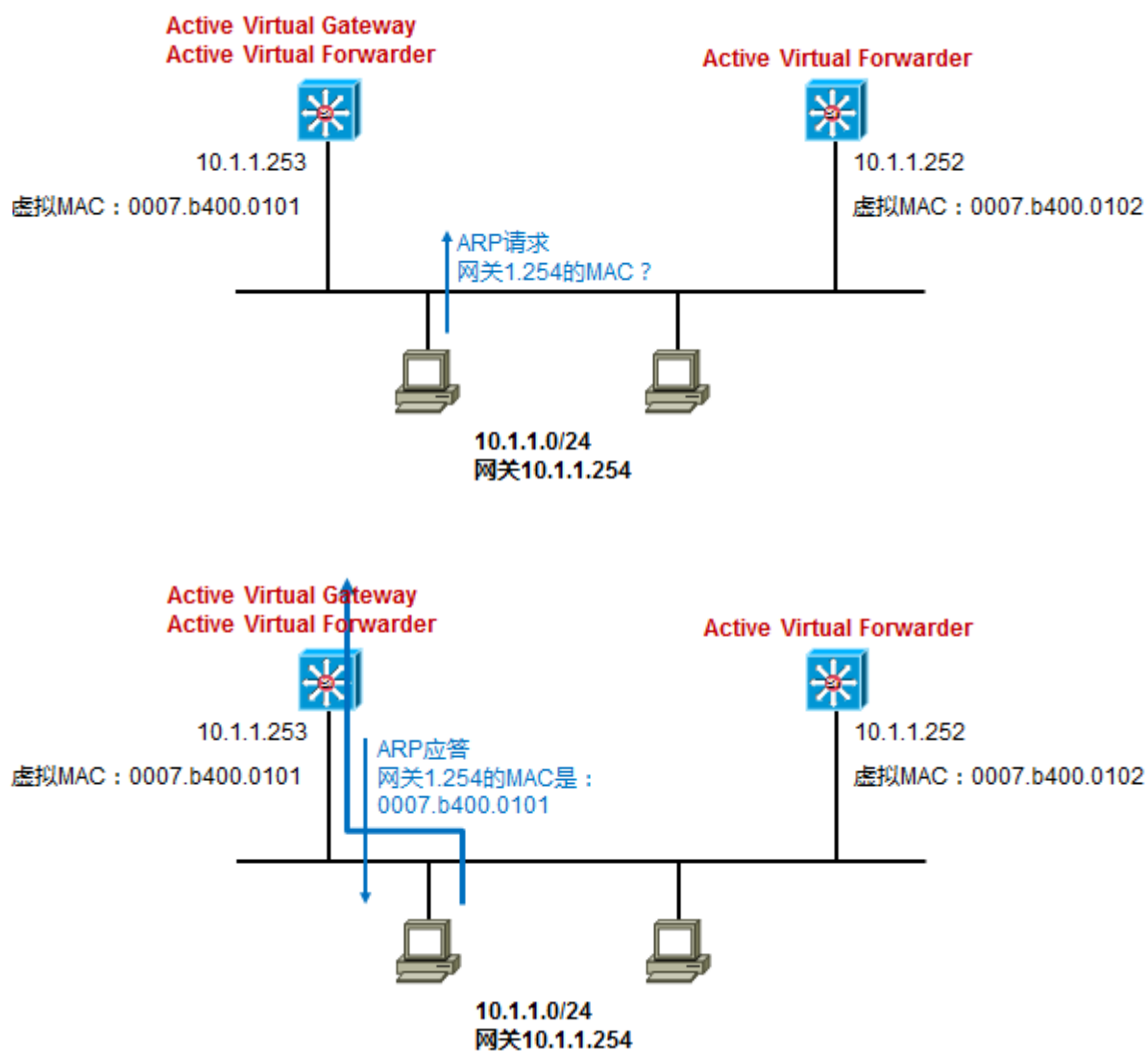
4.5 网关冗余技术

4.5.1 GLBP

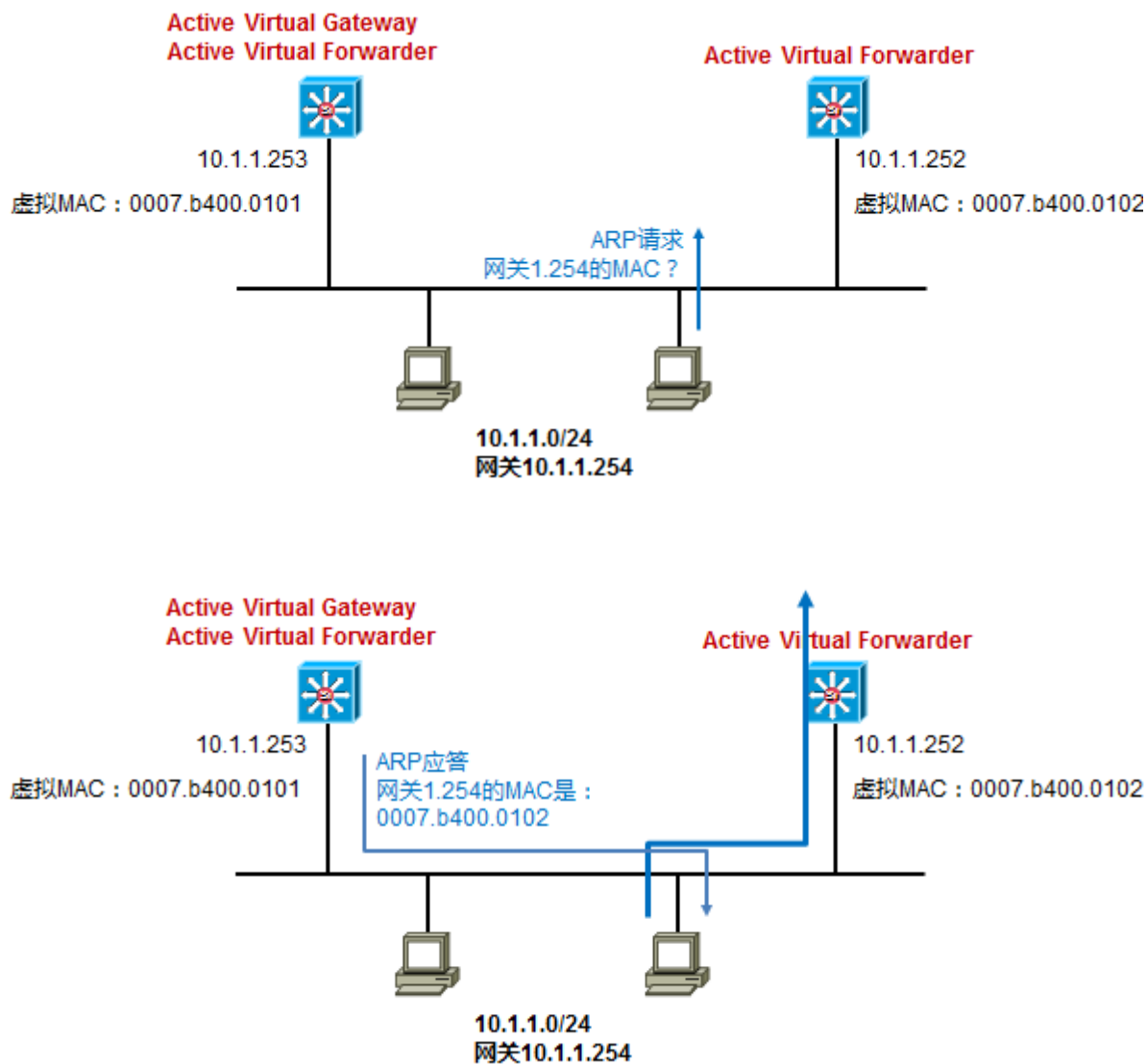
1. 协议概述

- 使得同一时间可使用多个网关，并且自动检测活跃网关。
- 每组 GLBP 最多可以有 4 台作为 ip 默认网关的成员路由器，这些网关被称为 **AVF (active virtual forwarder)**
- GLBP 自动管理虚拟 MAC 地址的分配，决定谁来负责处理转发的工作，这些功能由 **AVG(active virtual gateway 实现)**
- 因此 AVG 负责分发虚拟 MAC，AVF 负责转发数据

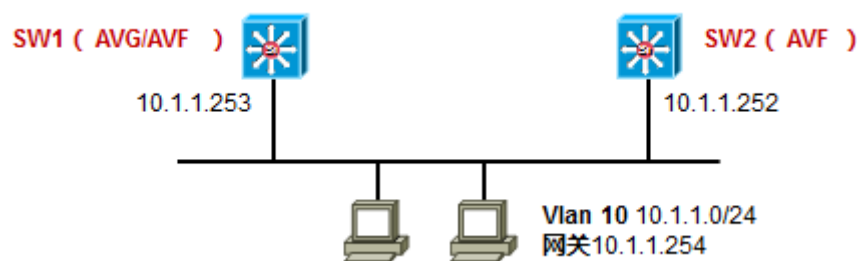
2. 工作机制



由 AVG 来负责响应 PC 对网关的 ARP 请求，同时 AVG 负责虚拟 MAC 的分配。AVG 可以根据不同的负载均衡模式灵活的对 PC 的 ARP 请求进行响应。



3. 配置及实现



```
SW1(config)# interface vlan 10
SW1(config-if)# ip address 10.1.1.253 255.255.255.0
SW1(config-if)# glbp 1 10.1.1.254
```

SW1(config-if)# glbp 1 priority 120 !! 优先级默认 100，优先级高的成为 AVG，优先级相等的情况下，最大 IP 的接口成为 AVG。默认“AVG 抢占”关闭

```
SW2(config)# interface vlan 10
SW2(config-if)# ip address 10.1.1.252 255.255.255.0
SW2(config-if)# glbp 1 10.1.1.254
```

R1#show glbp

FastEthernet0/0 - Group 1

State is Active

2 state changes, last state change 00:07:34

Virtual IP address is 10.1.1.254

Hello time 3 sec, hold time 10 sec

Next hello sent in 1.280 secs

Redirect time 600 sec, forwarder time-out 14400 sec

Preemption disabled

Active is local

Standby is 10.1.1.252, priority 100 (expires in 7.572 sec)

Priority 120 (configured)

Weighting 100 (default 100), thresholds: lower 1, upper 100

Load balancing: round-robin

Group members:

cc00.043c.0000 (10.1.1.253) local

cc01.043c.0000 (10.1.1.252)

There are 2 forwarders (1 active)

There are 2 forwarders (1 active)

Forwarder 1

State is Active

1 state change, last state change 00:07:24

MAC address is 0007.b400.0101 (default)

Owner ID is cc00.043c.0000

Redirection enabled

Preemption enabled, min delay 30 sec

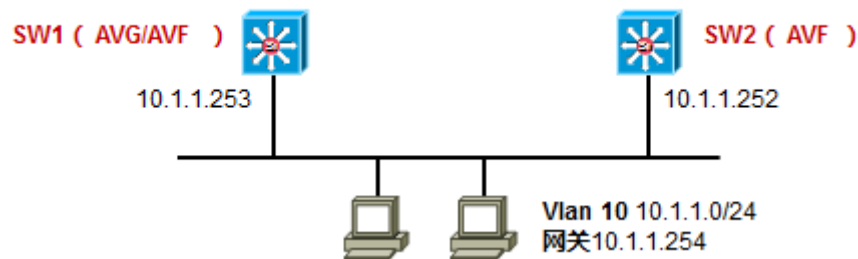
Active is local, weighting 100

```

Arp replies sent: 3
Forwarder 2
State is Listen
MAC address is 0007.b400.0102 (learnt)
Owner ID is cc01.043c.0000
Redirection enabled, 597.032 sec remaining (maximum 600 sec)
Time to live: 14397.032 sec (maximum 14400 sec)
Preemption enabled, min delay 30 sec
Active is 10.1.1.252 (primary), weighting 100 (expires in 7.028 sec)
Arp replies sent: 1
    
```

4. 负载均衡

实验一：host-dependent

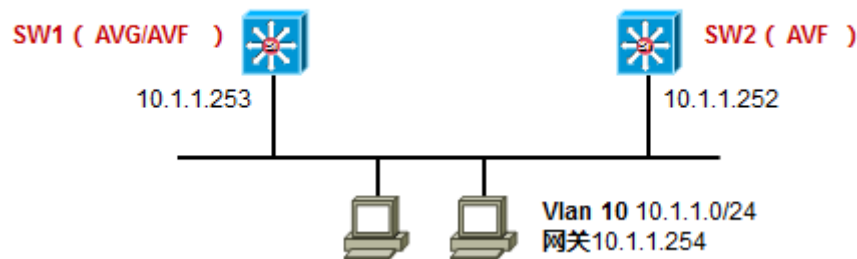


```

SW1(config)# interface vlan 10
SW1(config-if)# ip address 10.1.1.253 255.255.255.0
SW1(config-if)# glbp 1 10.1.1.254
SW1(config-if)# glbp 1 priority 120
SW1(config-if)# glbp 1 load-balancing host-dependent  !! 同一台 PC( 源 MAC ) 分配到固定的网关虚拟 MAC
    
```

- 负载均衡在 AVG 上修改即可生效，但是建议在所有路由器上都做配置
- 默认的负载均衡方式是 round-robin

实验二：weighted

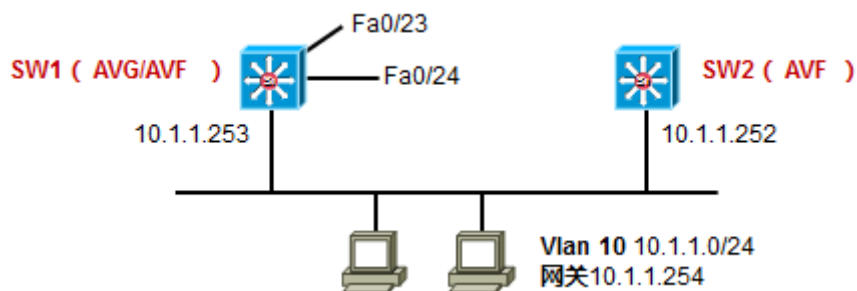


```
SW1(config)# interface vlan 10
SW1(config-if)# ip address 10.1.1.253 255.255.255.0
SW1(config-if)# glbp 1 10.1.1.254
SW1(config-if)# glbp 1 priority 120
SW1(config-if)# glbp 1 load-balancing weighted
SW1(config-if)# glbp 1 weighting 200
```

```
SW1(config)# interface vlan 10
SW1(config-if)# ip address 10.1.1.253 255.255.255.0
SW1(config-if)# glbp 1 10.1.1.254
SW1(config-if)# glbp 1 load-balancing weighted
SW1(config-if)# glbp 1 weighting 100
```

2:1

实验三：weighted 扩展



```
SW1(config)# track 10 interface fa0/23 line-protocol
SW1(config)# track 20 interface fa0/24 line-protocol
!
SW1(config)# interface vlan 10
SW1(config-if)# ip address 10.1.1.253 255.255.255.0
SW1(config-if)# glbp 1 10.1.1.254
SW1(config-if)# glbp 1 priority 120
SW1(config-if)# glbp 1 load-balancing weighted
SW1(config-if)# glbp 1 weighting 110 lower 85 upper 105
SW1(config-if)# glbp 1 weighting track 10 decrement 10
SW1(config-if)# glbp 1 weighting track 20 decrement 20
```

当weight低于85，则不再分配任何流量。只有流量再高于105，才会转发流量

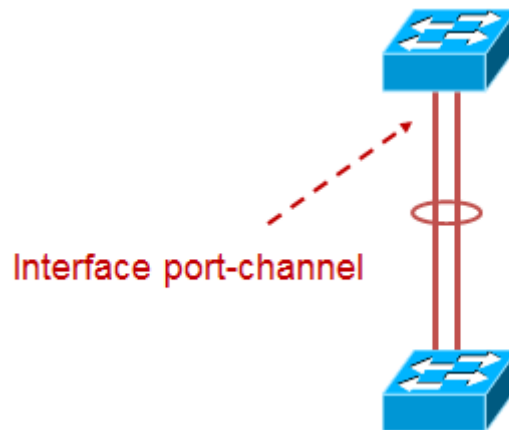
Track失败，weight减去一个值

4.5.2 HSRP

5 EtherChannel

1. 协议概述

- Solution to provide more bandwidth
- Logical aggregation of similar links
- Viewed as one logical link
- Provides load balancing and redundancy
- Supported for switch ports and routed ports



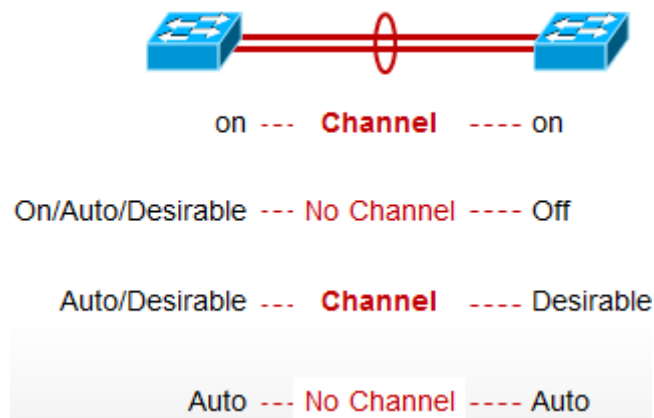
2. EtherChannel 协议

PAgP is a Cisco proprietary protocol

LACP is IEEE 802.3ad standard

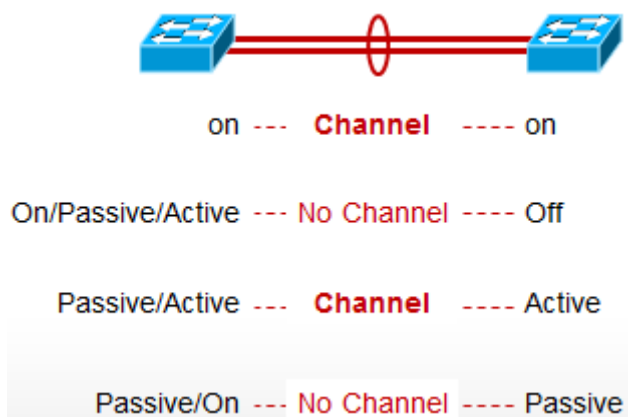
PAgP 的模式：

- On: channel member without negotiation (no protocol)
- Desirable: actively ask if the other side can/will
- Auto: passively wait for other side to ask
- Off: EtherChannel not configured on interface



LACP 的模式：

- On: channel member without negotiation (no protocol)
- Active: actively ask if the other side can/will
- Passive: passively wait for other side to ask
- Off: EtherChannel not configured on interface



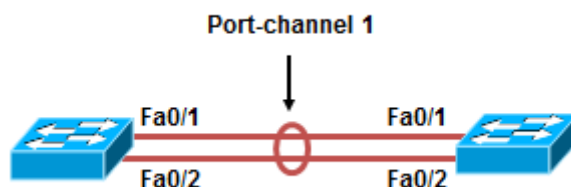
3. 实施要点

- Port-channel 接口一旦建立完成后，就形成了一个逻辑的接口，后续针对该接口的配置在 port-channel 逻辑接口中完成
- Port-channel 接口不能成为 SPAN 的目的接口
- 隶属于一个 port-channel 的物理接口需有相同的如下配置
 - 相同的 speed 和 duplex
 - 相同的接口模式 (access、trunk)
 - 如果是 trunk 模式，那么 native vlan 及 allowed vlan 需相同
 - 如果是 access 模式，所属 vlan 需相同

4. 配置及实现

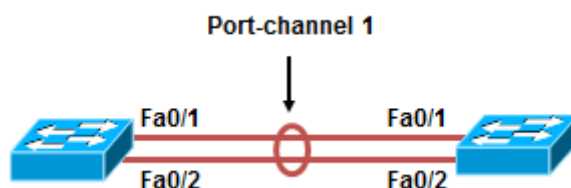
- 1) 选择用于 Channel 的端口
- 2) 选择 PAgP 或 LACP
- 3) 在接口上配置 channel-group
 - a) 设置 channel-group ID
 - b) 根据特定的协议，选择接口模式
- 4) 完成上述步骤后，逻辑的 etherchannel 接口就建立好了
可以进一步对这个逻辑的 etherchannel 接口进行配置

• 二层捆绑



```
Sw1(config)#interface range fastEthernet 0/1 – 2 //进入接口范围
Sw1(config-if-range)#switchport //将接口配置为二层接口
Sw1(config-if-range)#switchport trunk encapsulation dot1q
//trunk 封装协议为 dot1q
Sw1(config-if-range)#switchport mode trunk
//设置接口模式为 Trunk 模式
Sw1(config-if-range)#channel-protocol pagp/lacp
Sw1(config-if-range)#channel-group 1 mode desirable
//配置 etherchannel , ID 为 1 , 模式为 desirable
//查看命令：Show etherchannel 1 summary
```

• 三层捆绑

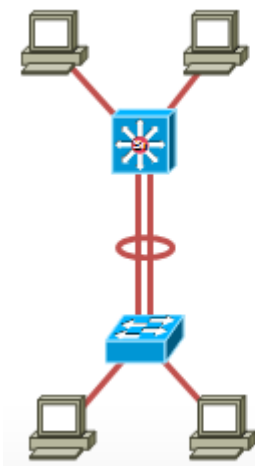


```
Sw1(config)#interface range fastEthernet 0/1 – 2
```

```
Sw1(config-if-range)#no switchport //配置 L3 接口
Sw1(config-if-range)#no ip address
Sw1(config-if-range)#channel-group 1 mode desirable
Sw1(config-if-range)#no shutdown
Sw1(config)#interface port-channel 1 //进入 port-channel 组 1 配置 IP
Sw1(config-if)#ip address 172.16.10.1 255.255.255.0
Sw1(config-if)#no shu
查看命令：Show eth 1 summary / show ip route / show ip int brief
```

5. 负载均衡

- EtherChannel 支持在同一个 port-channel 的链路中执行负载均衡
- 负载均衡动作可以基于 MAC、端口、IP (源 IP、目的 IP 或两者)
- 默认的行为：源目 IP 地址对 (src-dst-ip)



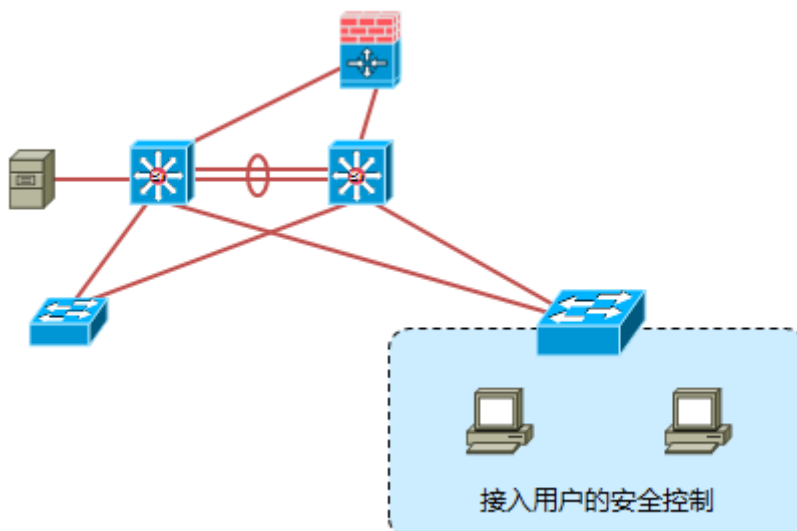
```
Switch(config)# port-channel load-balance type
```

```
Switch# show etherchannel load-balance
```

6 交换网络安全

6.1 Port-Security

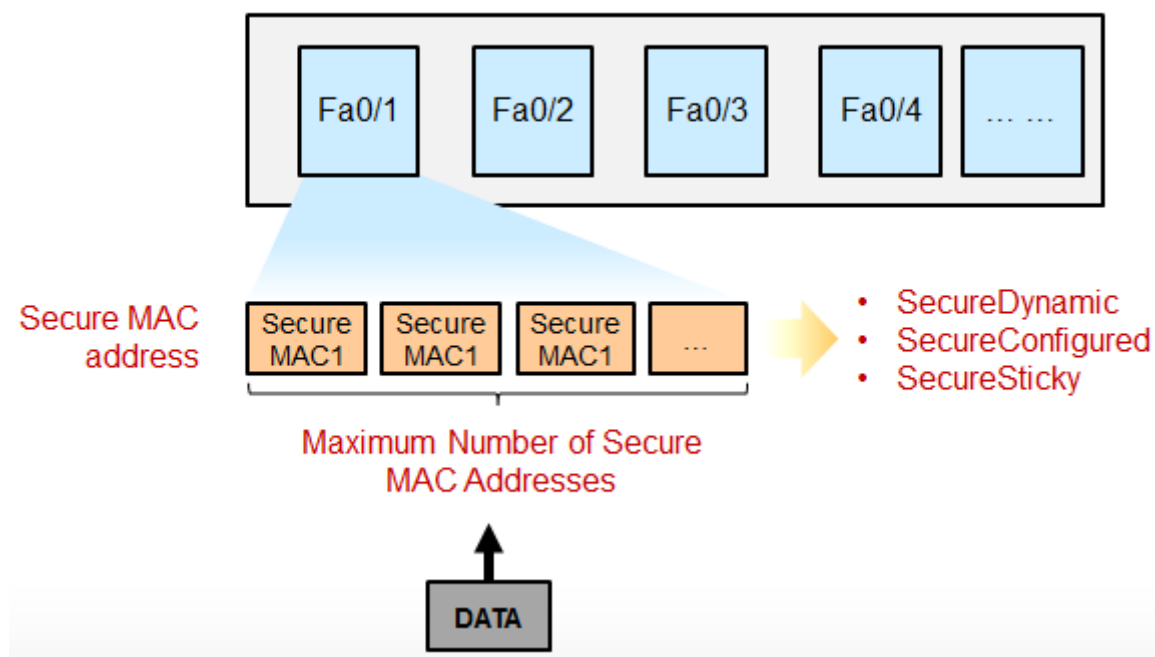
6.1.1 Port-Security 概述



在部署园区网的时候，对于交换机，我们往往有如下几种特殊的需求：

- 限制交换机每个端口下接入主机的数量（MAC 地址数量）
- 限定交换机端口下所连接的主机（根据 IP 或 MAC 地址进行过滤）
- 当出现违例时间的时候能够检测到，并可采取惩罚措施

上述需求，可通过交换机的 Port-Security 功能来实现：



6.1.2 理解 Port-Security

1. Port-Security 安全地址：secure MAC address

在接口上激活 Port-Security 后，该接口就具有了一定的安全功能，例如能够限制接口（所连接的）的最大 MAC 数量，从而限制接入的主机用户；或者限定接口所连接的特定 MAC，从而实现接入用户的限制。那么要执行过滤或者限制动作，就需要有依据，这个依据就是**安全地址** – secure MAC address。

安全地址表项可以通过让使用端口**动态学习到的 MAC (SecureDynamic)**，或者是**手工在接口下进行配置 (SecureConfigured)**，以及 **sticky MAC address (SecureSticky)** 三种方式进行配置。

当我们将接口允许的 MAC 地址数量设置为 1 并且为接口设置一个安全地址，那么这个接口将只为该 MAC 所属的 PC 服务，也就是源为该 MAC 的数据帧能够进入该接口。

2. 当以下情况发生时，激活惩罚 (violation)：

- 当一个激活了 Port-Security 的接口上，MAC 地址数量已经达到了配置的最大安全地址数量，并且又收到了一个新的数据帧，而这个数据帧的源 MAC 并不在这些安全地址中，那么启动惩罚措施
- 当在一个 Port-Security 接口上配置了某个安全地址，而这个安全地址的 MAC 又企图在同 VLAN 的另一个 Port-Security 接口上接入时，启动惩罚措施

当设置了 Port-Security 接口的最大允许 MAC 的数量后，接口关联的安全地址表项可以通过如下方式获取：

- 在接口下使用 switchport port-security mac-address 来配置静态安全地址表项
- 使用接口动态学习到的 MAC 来构成安全地址表项
- 一部分静态配置，一部分动态学习

当接口出现 up/down，则所有动态学习的 MAC 安全地址表项将清空。而静态配置的安全地址表项依然保留。

3. Port-Security 与 Sticky MAC 地址

上面我们说了，通过接口动态学习的 MAC 地址构成的安全地址表项，在接口出现 up/down 后，将会丢失这些通过动态学习到的 MAC 构成的安全地址表项，但是所有的接口都用 switchport port-security mac-address 手工来配置，工作量又太大。因此这个 sticky mac 地址，可以让我们将这些动态学习到的 MAC 变成“粘滞状态”，可以简单的理解为，我先动态的学，学到之后我再将你粘起来，形成一条“静态”（实际上是 SecureSticky）的表项。

在 up/down 现象出现后仍能保存。而在使用 wr 后，这些 sticky 安全地址将被写入 start-up config，即使设备重启也不会被丢失。

6.1.3 默认的 Port-Security 配置

Port-Security

默认关闭

默认最大允许的安全 MAC 地址数量	1
惩罚模式	shutdown (进入 err-disable 状态), 同时发送一个 SNMP trap

6.1.4 Port-Security 的部署注意事项

1. Port-Security 配置步骤

- 1) 在接口上激活 Port-Security
Port-Security 开启后, 相关参数都有默认配置, 需关注
- 2) 配置每个接口的安全地址 (Secure MAC Address)
可通过交换机动态学习、手工配置、以及 sticky 等方式创建安全地址
- 3) 配置 Port-Security 惩罚机制
默认为 shutdown, 可选的还有 protect、restrict
- 4) (可选) 配置安全地址老化时间

2. 关于被惩罚后进入 err-disable 的恢复 :

如果一个 psec 端口由于被惩罚进入了 err-disable, 可以使用如下方法来恢复接口的状态 :

- 使用全局配置命令 : err-disable recovery psecure-violation
- 手工将特定的端口 shutdown 再 no shutdown

3. 清除接口上动态学习到的安全地址表项

使用 clear port-security dynamic 命令, 将清除所有 port-security 接口上通过动态学习到的安全地址表项

使用 clear port-security sticky 命令, 将清除所有 sticky 安全地址表项

使用 clear port-security configured 命令, 将清除所有手工配置的安全地址表项

使用 clear port-security all 命令, 将清除所有安全地址表项

使用 show port-security address 来查看每个 port-security 接口下的安全地址表项

4. 关于 sticky 安全地址

Sticky 安全地址, 是允许我们将 Port-Security 接口通过动态学习到的 MAC 地址变成 “粘滞” 的安全地址, 从而不会由于接口的 up/down 丢失。然而如果我们希望在设备重启之后 这个 sticky 的安全地址表项仍然存在, 那么就需要 wr 一下。将配置写入 start-up config 文件。Sticky 安全地址也是一个简化我们管理员操作的一个很好的工具, 毕竟现在不用再一条条的手工去绑了。

5. port-security 支持 private vlan

6. port-security 支持 802.1Q tunnel 接口
7. port-security 不支持 SPAN 的目的接口
8. port-security 不支持 etherchannel 的 port-channel 接口
9. 在 CISCO IOS 12.2(33)SXH 以及后续的版本，我们可以将 port-security 及 802.1X 部署在同一个接口上。
而在此之前的软件版本：
 - 如果你试图在一个 port-security 接口上激活 802.1X 则会报错，并且 802.1X 功能无法开启
 - 如果你试图在一个 802.1X 接口上激活 port-security 则也会报错，并且 port-security 特性无法开启

10. Port-Security 支持 nonegotiating trunk 接口

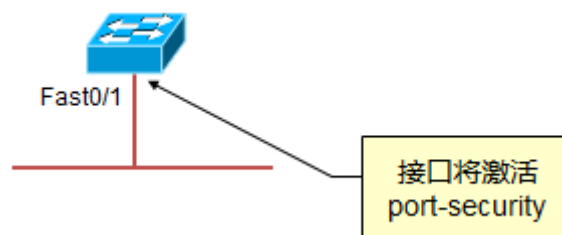
- Port-Security 支持在如下配置的 trunk 上激活


```
switchport
switchport trunk encapsulation ?
switchport mode trunk
switchport nonegotiate
```
- If you reconfigure a secure access port as a trunk, port security converts all the sticky and static secure addresses on that port that were dynamically learned in the access VLAN to sticky or static secure addresses on the native VLAN of the trunk. Port security removes all secure addresses on the voice VLAN of the access port.
- If you reconfigure a secure trunk as an access port, port security converts all sticky and static addresses learned on the native VLAN to addresses learned on the access VLAN of the access port. Port security removes all addresses learned on VLANs other than the native VLAN.

11. Flex links 和 Port-Security 互不兼容

6.1.5 Port-security 的配置

1. 激活 Port-Security (在 access 接口上)



```
Switch(config)# interface fast0/1
Switch(config-if)# switchport
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# switchport port-security
```


接口的 Port-Security 特性一旦激活后，默认的最大安全地址个数为 1，也就是说，在不进行手工配置安全地址的情况下，这个接口将使用其动态学习到的 MAC 作为安全地址，并且，这个接口相当于被该 MAC（所属的设备）独占。而且默认的 violation 是 shutdown

SW1#show port-security interface f0/1

```
Port Security           : Enabled
Port Status             : Secure-up    !!接口目前的状态是 up 的
Violation Mode          : Shutdown     !!违例后的惩罚措施，默认为 shutdown
Aging Time              : 0 mins
Aging Type              : Absolute
SecureStatic Address Aging : Disabled
Maximum MAC Addresses   : 1            !!最大安全地址个数，默认为 1
Total MAC Addresses     : 0
Configured MAC Addresses : 0           !!手工静态配置的安全 MAC 地址，这里没配
Sticky MAC Addresses    : 0           !!sticky 的安全地址，这里没有
Last Source Address:Vlan : 00b0.1111.2222:10 !!最近的一个安全地址+vlan
Security Violation Count : 0           !!该接口历史上出现过的违例次数
```

这个时候，如果另一台 PC 接入到这个端口上，那么该 port-security 接口将会收到一个新的、非安全地址表项中的 MAC 地址的数据帧，于是触发的违例动作，给接口将被 err-disable 掉。同时产生一个 snmp trap 消息，另外，接口下，Security Violation Count 将会加 1


2. 激活 Port-Security（在 trunk 接口上）



- 注意该trunk接口需为DTP非协商模式
- 同时由于默认的port-security最大允许的安全地址表项为1且violation为shutdown，因此在激活port-security前强烈建议先修改最大允许的安全地址数量

```
Switch(config)# interface fast0/24
Switch(config-if)# switchport
Switch(config-if)# switchport trunk encapsulation {dot1q | isl}
Switch(config-if)# switchport mode trunk
Switch(config-if)# switchport nonegotiate
Switch(config-if)# switchport maximum ?
Switch(config-if)# switchport port-security
```

3. Port-Security violation 惩罚措施



默认的惩罚措施是shutdown

```
Switch(config)# interface fast0/1
Switch(config-if)# switchport port-security violation {protect | restrict | shutdown}
```

- Protect 仅丢弃非法的数据帧
- Restrict 丢弃非法的数据帧，同时产生一个syslog消息
- Shutdown 将端口置为err-disable，接口不可用，同时产生一个syslog消息

默认的 violation 是 shutdown。如果是 protect，那么惩罚就会温柔些，对于非法的数据帧（例如数据帧的源 MAC 不在安全地址表项中的、且安全地址已经达到最大数），这些非法数据将仅仅被丢弃，合法数据照常转发，同时不会触发一个 syslog 消息，另外接口下的“Security Violation Count”也不会加 1。而如果是 restrict，那么非法数据被丢弃，同时触发一个 syslog 消息，再者，Security Violation Count 加 1，合法的数据照常转发。

4. 配置 Port Security Rate Limiter

（注意，在 6509 交换机，truncated switching 模式下不支持该功能）

在交换机接口上开启 Port-Security 是会耗费资源的，Port-Security 会检测每一个进入接口的数据帧，以判断流量是否合法，或者是否存在违例行为。当一个安全接口设置的 violation 为 shutdown 的时候，该接口在违

例后触发惩罚机制，进入 err-disable 状态，这样可以有效的方式有效的防止交换机由于处理违例事件导致交换机的 CPU 利用率过高。然而 protect 和 restrict 的惩罚措施，则不会将端口关闭，端口依然可用，那么这就可能导致在违例事件发生的情况下交换机的 CPU 利用率飙升（例如大量的非法数据涌入）。因此当使用 protect 和 restrict 这两种违例惩罚措施事，可以通过 Port-Security rate limiter 来防止 CPU 飙升。

```
Switch(config)# mls rate-limit layer2 port-security rate_in_pps [burst_size]
```

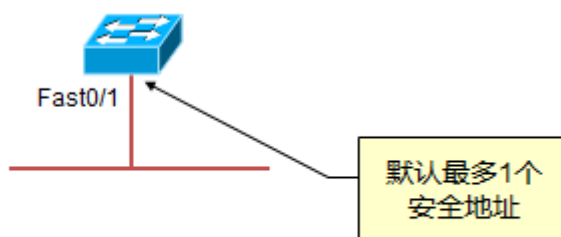
关于 rate_in_pps 参数：

- 范围是 10- 1000000
- 没有默认值
- 值设置的越低，对 CPU 的保护程度就越高，这个值对惩罚措施发生前、后都生效，当然这个值也不能设置的过低，至少要保障合法流量被处理吧。一般低于 1000 就差不多。

关于 burst-size 参数：

- 范围是 1-255
- 默认是 10，这个默认值一般就够用了。

5. 配置 Port-Security 最大允许的安全地址数量



```
Switch(config)# interface fast0/1
Switch(config-if)# switchport port-security maximum ?
```

最大安全地址数量，不同的软件平台允许的上限值有所不同，但是默认都是 1。

在 trunk 口上，前面说了，也是可以激活 port-security 的，而在 trunk 口上配置最大安全地址数量，可以基于接口配置（对所有 VLAN 生效），也可以基于 VLAN 进行配置。如下：

```
switchport port-security maximum 1
```

```
switchport port-security maximum 1 vlan 10,20,30
```

!!可以关联多个 VLAN

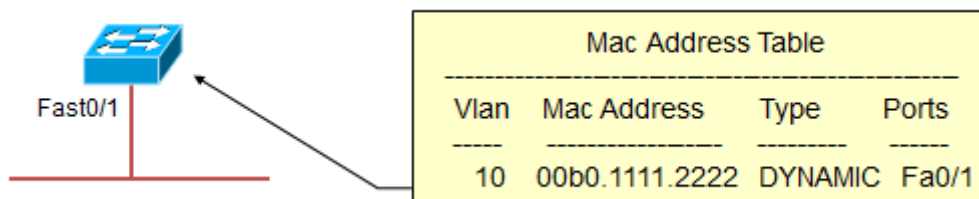
6. 在 port-security 接口上手工配置安全地址



```
Switch(config)# interface fast0/1
Switch(config-if)# switchport
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# switchport port-security
Switch(config-if)# switchport port-security maximum 3
Switch(config-if)# switchport port-security mac-address xxxx [vlan id]
```

- 上述配置中，最大安全地址数设置为 3，然后使用手工配置了一个安全地址，那么剩下 2 个，交换机可以通过动态学习的方式来构建安全地址。
- 在 trunk 接口上手工配置安全地址，可关联 vlan 关键字。如果在 trunk 接口上手工配置安全地址，没有关联 vlan 关键字，那么该安全地址将被关联到 trunk 的 native vlan 上

7. 在 port-security 接口上使用 sticky MAC 地址



```
Switch(config)# interface fast0/1
Switch(config-if)# switchport
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# switchport port-security
Switch(config-if)# switchport port-security mac-address sticky
```

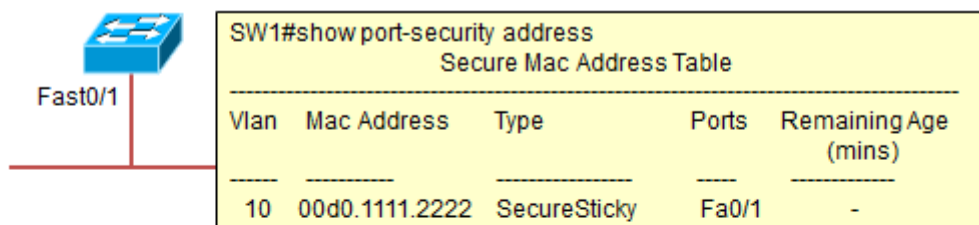
我们知道，构成安全地址表项的方式有好几种，其中一种是使用 switchport port-security mac 来手工配置，但是这种方式耗时耗力，更需要去 PC 上抄 MAC，工作成本比较高。而另一种构成安全地址的方式是让交换机使用动态学习到的 MAC，然而这些安全地址在接口一旦 up/down 后，将丢失，更别说重启设备了。因此可以使用 sticky mac 的方式，这种方式激活后，交换机将动态学习到的 MAC “粘起来”，具体的动作很简单，就是在动态学习到 MAC（例如一个 00b0.1111.2222）后，如果我激活了 sticky MAC address，则在接口下自动产生一条命令：

```
interface FastEthernet0/1
switchport access vlan 10
```

```
switchport mode access
switchport port-security
switchport port-security mac-address sticky
```

switchport port-security mac-address sticky 00b0.1111.2222 !! 自动产生

这样形成的安全地址表项（是 SecureSticky 的），即使在接口翻动，也不会丢失。在者如果 wr 保存配置，命令写入 config.text，那么设备即使重启，安全地址也不会丢失。



```
Switch(config)# interface fast0/1
Switch(config-if)# switchport
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# switchport port-security
Switch(config-if)# switchport port-security mac-address sticky
Switch(config-if)# switchport port-security mac-address sticky 00b0.1111.2222
```

当在接口上激活了 port-security mac-address sticky，那么：

- 该接口上所有通过动态学习到的 MAC，将被转成 sticky mac address 从而形成安全地址
- 接口上的静态手工配置的安全地址不会被转成 sticky mac address
- 通过 voice vlan 动态学习到的安全地址不会被转成 sticky mac address
- 命令配置后，新学习到的 MAC 地址，也是 sticky 的

当此时又敲入 no port-security mac-address sticky，则所有的 sticky 安全地址条目都变成动态的安全地址条目（SecureDynamic）

8. 配置安全地址老化时间

配置的命令比较简单：

```
Switch(config-if)# switchport port-security aging type {absolute | inactivity}
```

配置老化时间的类型，如果选择 absolute，也就是绝对时间，需要搭配 aging time 命令设定老化时间的具体值，命令一旦敲下去后，所有的通过动态学习的 MAC 构建的安全地址表项将开始以 aging time 倒计时。如果是 inactivity 关键字，则只在该动态安全地址表项不活跃的时候（譬如主机离线了或者挂掉了）才开始倒计时。

```
Switch(config-if)# switchport port-security aging time ?
```

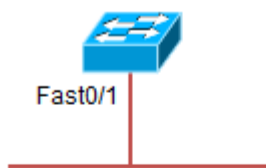
设定老化时间

```
Switch(config-if)# switchport port-security aging static
```

使用前面两条命令，老化时间是会影响那些使用静态手工配置的安全地址表项的，当然 sticky 表项也不会受影响，这些表项都是永不老化的，但是如果搭配上上面这条命令，则手工配置的安全地址表项也受限于老化时间了。

不过对于 sticky 表项，则始终不会激活 aging time，它是不会老化的。

• 示例 1：



```
Switch(config)# interface fast0/1
Switch(config-if)# switchport
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# switchport port-security
Switch(config-if)# switchport port-security maximum 3
Switch(config-if)# switchport port-security aging type absolute
Switch(config-if)# switchport port-security aging time 50
```

将安全地址老化时间设置为 50min。针对动态学习到的 MAC 构成的安全地址有效

50min 是一个绝对时间，配置完成后开始倒计时，无论该 MAC 是否依然活跃，都始终进行倒计时

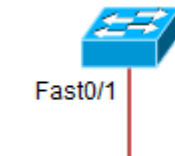
• 示例 2：



```
Switch(config)# interface fast0/1
Switch(config-if)# switchport
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# switchport port-security
Switch(config-if)# switchport port-security maximum 3
Switch(config-if)# switchport port-security aging type inactivity
Switch(config-if)# switchport port-security aging time 50
```

针对动态学习到的 MAC 构成的安全地址有效,如果该 MAC 在 50min 内一直处于失效状态(例如主机离线了),那么该安全地址在 aging time 超时后被清除

• 示例 3 :



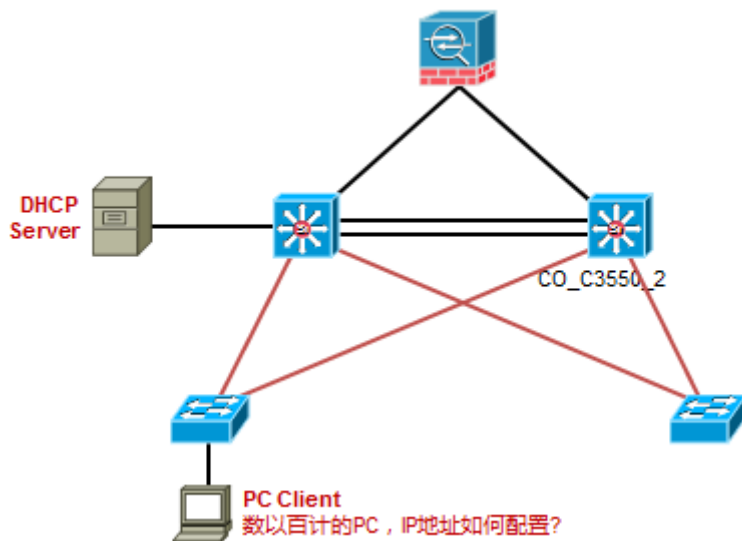
SW1#sh port-security address				
Secure Mac Address Table				
Vlan	Mac Address	Type	Ports	Remaining Age (mins)
10	00b0.9999.9999	SecureConfigured	Fa0/1	47
10	00d0.2222.2222	SecureDynamic	Fa0/1	47

注意,上述两种配置方式,对手工配置 switchport port-security mac-address 00b0.9999.9999 的安全地址无效。也就是采用上述方法配置的静态安全地址表项永不超时。

- 如果增加 switchport port-security aging static 命令,则手工静态配置的安全地址也的 aging time 也开始计时
- 注意,对于 sticky mac address,安全地址的老化时间无效

6.2 DHCP

6.2.1 DHCP 概述



- Dynamic Host Configuration Protocol
- 用于动态地址分配
- 基于 UDP 协议 端口 67 及 68
- bootPC:67 (客户端端口号); bootPS : 68 (服务端端口号)

6.2.2 协议机制

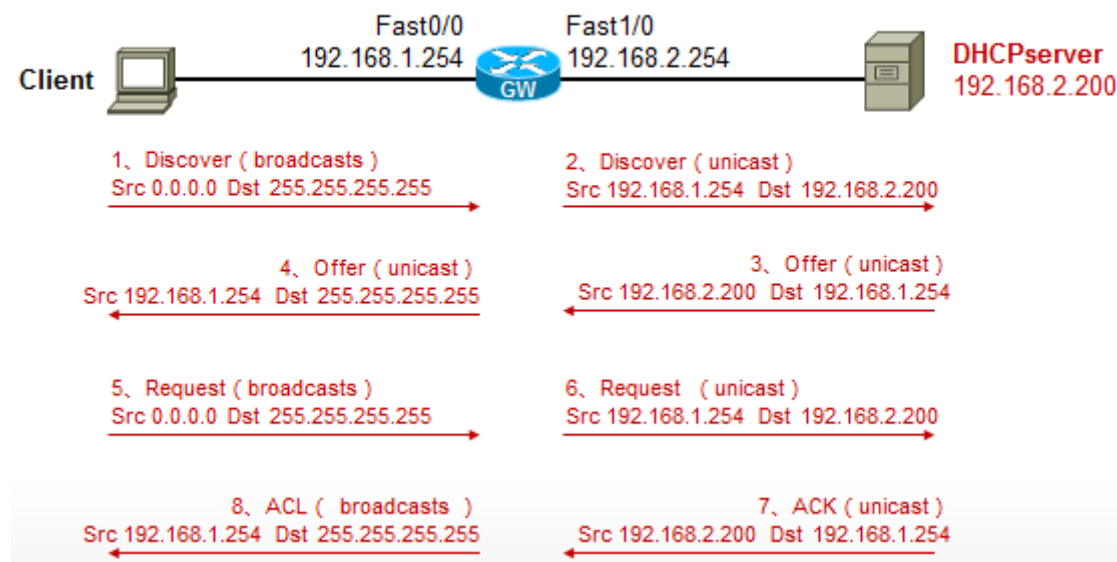
1. 报文交互过程



No. .	Time	Source	Destination	Protocol	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	DHCP Discover
2	1.984000	192.168.1.254	255.255.255.255	DHCP	DHCP offer
3	2.033000	0.0.0.0	255.255.255.255	DHCP	DHCP Request
4	2.129000	192.168.1.254	255.255.255.255	DHCP	DHCP ACK

注意，不同的产品，数据交互过程有所不同，上述过程是在 CISCO 设备（服务端）上测试的结果。

2. DHCP 中继



6.2.3 配置及实现

1. 基本配置

```
Router(config)#server dhcp
```

开启 DHCP 服务。By default, the Cisco IOS DHCP server and relay agent features are enabled on your router

```
Router(config)#ip dhcp pool [pool name]
```

定义 DHCP 地址池，一个网段对应一个地址池

```
Router(config-dhcp)#network [network address][subnet mask]
```

定义地址池关联的网段

```
Router(config-dhcp)#default-router [host address]
```

定义分配给客户端的网关 IP

```
Router(config)#ip dhcp excluded-address 172.16.1.100 172.16.1.103
```


将一个或多个地址排除在地址池之外 (如网关 IP 等), 以避免分配给客户端

2. 查看及验证

```
show ip dhcp database
```

Displays recent activity on the DHCP database

```
show ip dhcp server statistics
```

Shows count information about statistics and messages sent and received

```
show ip route dhcp
```

Displays routes added to the routing table by DHCP

```
Show ip dhcp binding
```

Show dhcp binding on DHCP server

3. 验证

```
debug dhcp detail
```

Enables debugging on the DHCP client

```
debug ip dhcp server {events | packets | linkage}
```

Enables debugging on the DHCP server

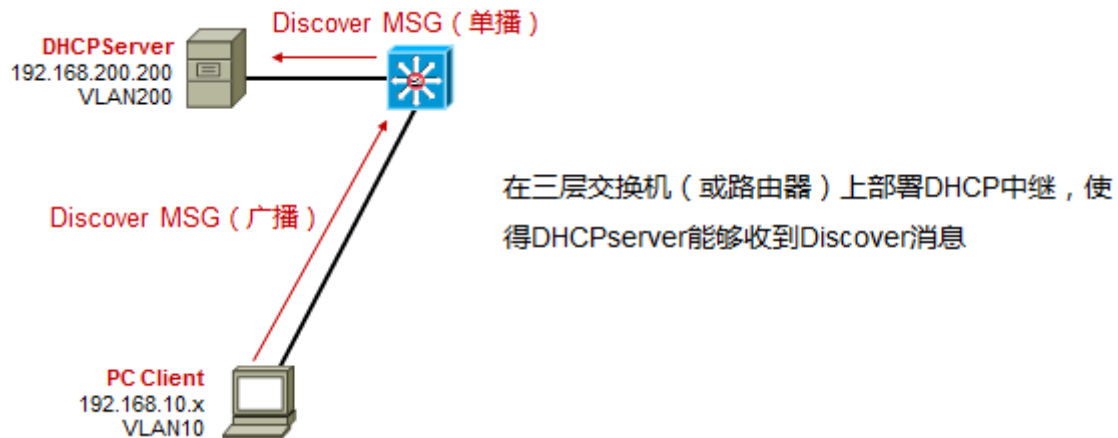
6.2.4 配置示例

1. 基础实验



2. DHCP 中继实验

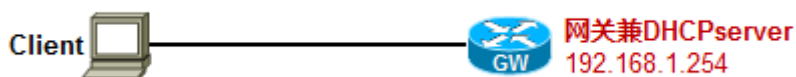




注意，中继在配置的时候：

- Ip helper-address 命令是配置在收到 client 发出的广播 discovery 报文的那个三层接口上，或者说，是“阻挡”这个广播 DHCP 报文的接口上的，例如，如果是三层交换机用 access 接口连接 PC，那么就应该配置在该 VLAN 的 SVI 接口上。
- Ip helper-address 命令只需配置在第一个收到 client 发出的广播 discovery 报文的那个三层接口上，在此之后这些广播的 DHCP 报文将被中继成单播包，就是走 IP 路由了。
- 注意中继设备需开启 DHCP 服务：service dhcp
- 注意 DHCPserver 与中继接口之间必须三层能通，因为中继后的单播包，源地址是配置 ip helper-address 的那个中继接口，所以 DHCPserver 在回包的时候，报文的目的 IP 就是这个配置 ip helper-address 的接口的接口 IP。

3. DHCP 静态地址绑定



- Manual bindings are IP addresses that have been manually mapped to the MAC addresses of hosts that are found in the DHCP database. Manual bindings are stored in NVRAM on the DHCP server. Manual
- bindings are just special address pools. There is no limit on the number of manual bindings, but you can only configure one manual binding per host pool.

配置命令如下：

```
ip dhcp pool BIND
host 192.168.1.111 255.255.255.0
client-identifier unique-identifier
```

- DHCP clients require client identifiers. The unique identification of the client is specified in dotted

hexadecimal notation, for example, 01b7.0813.8811.66, where 01 represents the Ethernet media type.

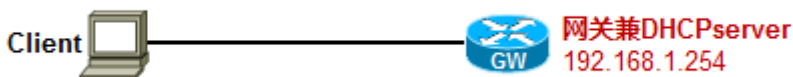
- 也可以在服务器上通过 show ip dhcp binding 抓取客户端的 client-identifier

如果客户端发出的 DHCP discovery 报文中不包含 client-identifier, 则可使用 hardware address 来识别终端设备。配置命令如下:

```
ip dhcp pool BIND
  host 192.168.1.111 255.255.255.0
  hardware-address hardware-address type ! 可选
```

上面命令中的 type 参数, 用来指示这个硬件地址的硬件类型, 如 type 为 1, 则类型为 ethernet, 如果 type 为 6, 则类型为 802, 默认是 ethernet

4. DHCP 配置 手工绑定 从网络读取绑定



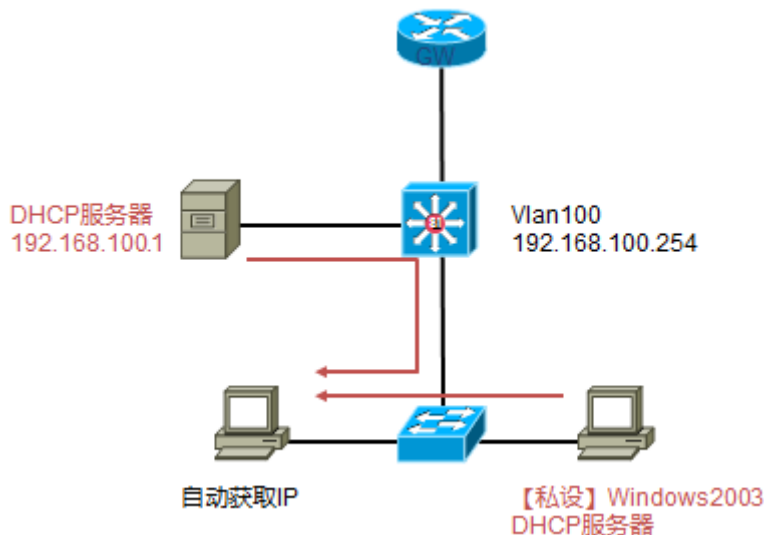
```
ip dhcp pool BIND
  origin file flash:CCIE.txt
```

CCIE.txt 格式:

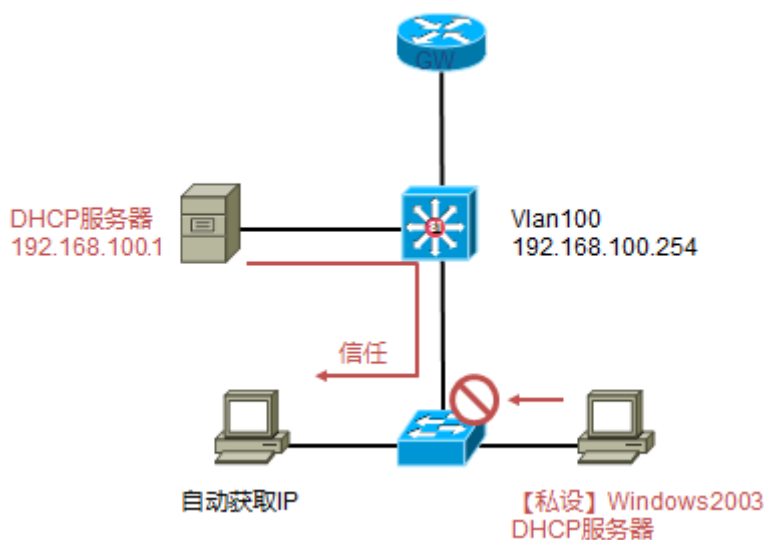
```
*time* Nov 05 2011 07:52 PM
*version* 2
!IP address      Type      Hardware address      Lease expiration
30.30.44.45 /24   id        0100.e01e.6012.89     Infinite
*end*
```

6.3 DHCPsnooping

6.3.1 机制概述



DHCP 都非常熟悉了,对于 DHCP 客户端而言,初始过程中都是通过发送广播的 DHCP discovery 消息寻找 DHCP 服务器,然而这时候如果内网中存在私设的 DHCP 服务器,那么就会对网络造成影响,例如客户端通过私设的 DHCP 服务器拿到一个非法的地址,最终导致 PC 无法上网。



在 DHCP snooping 环境中 (部署在交换机上),我们将端口视为 trust 或 untrust 两种安全级别,也就是信任或非信任接口。在交换机上,将连接合法 DHCP 服务器的接口配置为 trust。只有 trust 接口上收到的来自 DHCPserver 的报文 (如 DHCP OFFER, DHCP ACK, DHCP NAK, 或者 DHCP REQUEST) 才会被放行,相反,在 untrust 接口上收到的来自 DHCPserver 的报文将被过滤掉,这样一来就可以防止非法的 DHCPserver 接入。同时在部署了 DHCP Snooping 的交换机本地,还能维护一张 **DHCP snooping 的绑定数据库** (binding database),用于保存侦听到的 DHCP 交互的表项,信息包括 (针对 untrust 接口的): **MAC 地址、IP 地址 (DHCP 分配的)、租期、绑定类型、VLAN 号、接口编号 (DHCP 客户端也就是连接客户端 PC 的 untrust 接口)**。这个 DHCP snooping binding database 除了可以做一些基本的安全接入控制,还能够用于 DAI 等防 ARP 欺骗的解决方案。

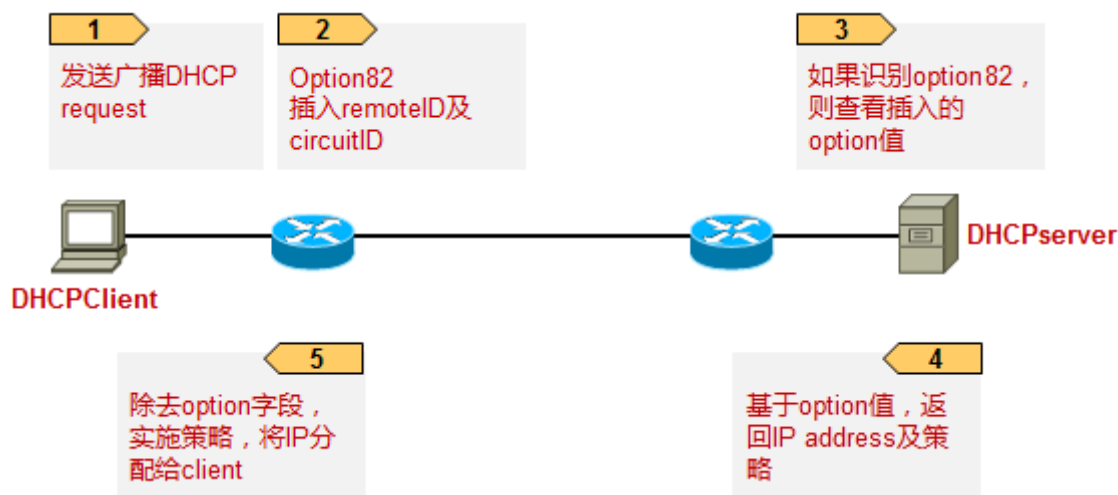
一台支持 DHCP snooping 的交换机 , 如果在其 **untrust** 接口上 , 收到来自下游交换机发送的、且带有 option82 的 DHCP 报文 , 则默认的动作是丢弃这些报文。如果该交换机开启了 DHCP snooping 并且带有 option82 的 DHCP 报文是在 **trusted** 接口上收到的 , 则交换机接收这些报文 , 但是不会根据报文中包含的相关信息建立 DHCP binding database 表项。

如果交换机确实通过一个 untrust 接口连接了下游交换机 , 并且希望放行该接口收到的、下游交换机发送出来的带有 option82 的 DHCP 报文 , 则可使用全局命令 : `ip dhcp snooping information option allow-untrusted` , 同时由于是通过 untrust 接口收到的 DHCP 报文 , 因此会根据侦听的结果创建 DHCP snooping binding database 表项。

6.3.2 Option82

DHCP option 82 又称为 DHCP 中继代理信息选项 (Relay Agent Information Option) , 是 DHCP 报文中的一个选项 , 其编号为 82

CISCO 注释 : The DHCP option-82 feature is supported only when DHCP snooping is globally enabled and on the VLANs to which subscriber devices using this feature are assigned.



基本的交互过程如下 (下面描述的过程与上面的配图无关):

1. DHCP client 广播 DHCP request
2. 交换机收到 DHCP request , 它在报文中插入 option82 信息 (当然 , 要你先做了相应的配置后) , 在插入的信息中 , remote-ID suboption 是该交换机的 MAC ; circuit-ID suboption 是收到 DHCP request 报文的接口 ID。
3. 如果交换机上还部署了 DHCP relay , 那么交换机将中继地址也添加进 DHCP 报文中
4. 交换机将携带了 option82 信息的 DHCP 报文转发给 DHCP 服务器

5. DHCP 服务器接收到了 DHCP 报文，如果这台服务器是能够识别 option82 的，那么它将根据 option82 中携带的信息，来为客户端分配 IP 地址，以及选择相应的策略。然后服务器回应一个 DHCP 包，其中也包含 option82 信息。
6. 交换机收到该 DHCP 报文后，发现报文的 option82 信息正是从本地始发，交换机将 option82 信息移除后，将 DHCP 报文转发给 DHCP client。

在上述事件的发生过程中，下面字段的内容不会发生改变：

- **Circuit-ID suboption fields**
 - Suboption type
 - Length of the suboption type
 - Circuit-ID type
 - Length of the circuit-ID type
- **Remote-ID suboption fields**
 - Suboption type
 - Length of the suboption type
 - Remote-ID type
 - Length of the remote-ID type

In the port field of the circuit-ID suboption, the port numbers start at 3.也就是说这个字段中，端口号是从 3 开似乎的，第一个端口号就是 3，依次类推。例如，一个带了 SFP 插槽的 24 口交换机，port3 是 fast x/0/1，port4 是 fast x/0/2，以此类推，port 27 就是 sfq 模块的 x/0/1 口。其中 x 为堆叠成员编号。

Option 报文结构：

DHCP option 82 又称为 DHCP 中继代理信息选项 (Relay Agent Information Option)，是 DHCP 报文中的一个选项，其编号为 82。rfc3046 定义了 option 82，选项位置在 option 255 之前而在其它 option 之后。

Code	Len	Agent Information Field					
82	N	i1	i2	i3	i4	...	iN

- CODE：82
- LEN：Agent Information field 的字节数，不包含 code 及 len 字段的长度

Option82 可包含多个 suboption，rfc3046 定义了如下两个：

SubOpt	Len	Sub-option Value						...	
1	N	s1	s2	s3	s4	...	sN		

SubOpt	Len	Sub-option Value						...	
2	N	i1	i2	i3	i4	...	iN		

- Subopt : 子选项编号, 如果是 circuit ID 则值为 1, remote ID 则值为 2
- Len : Sub-option Value 的字节个数, 不包括 SubOpt 和 Len 字段的两个字节

option82 子选项 1 : 即 Circuit ID, 它表示接收到的 DHCP 请求报文来自的电路标识, 这个标识只在中继代理节点内部有意义, 在服务器端不可以解析其含义, 只作为一个不具含义的标识使用。一般情况下, 默认是接收到 DHCP 请求报文的接入交换机 VlanID 加接入二层端口名称, 如 Vlan2+Ethernet0/0/10, 也可以由用户指定自己的代理电路 ID。通常子选项 1 与子选项 2 要共同使用来标识 DHCP 客户端的信息。

option82 子选项 2 : 即 Remote ID, 一般情况下为插入该 option82 信息的接入层交换机的 MAC。

CISCO 注释 : If the DHCP relay agent is enabled but DHCP snooping is disabled, the DHCP option-82 data insertion feature is not supported.

6.3.3 DHCP Snooping Binding Database

在交换机上激活 DHCP snooping 后, 交换机会使用在 untrust 接口上 “侦听到的” DHCP 交互内容形成一个表: dhcp snooping binding database。如下:

SW1#show ip dhcp snooping binding

MacAddress	ipAddress	Lease(sec)	Type	VLAN	Interface
00:B0:64:04:09:99	192.168.10.1	86025	dhcp-snooping	10	FastEthernet0/1

Total number of bindings: 1

默认情况下, 这个数据库是动态的, 也就是说当交换机重启, 数据库内的表项全部丢失。这个事情的结果可能会造成网络的中断。因此我们可以将这个数据库以文件的形式存储起来, 这个就是 DHCP snooping database agent。配置命令非常简单, 全局命令:

SW2(config)#ip dhcp snooping database ?

flash: Database agent URL


```
ftp:      Database agent URL
http:     Database agent URL
https:    Database agent URL
rcp:      Database agent URL
scp:      Database agent URL
tftp:     Database agent URL
timeout   Configure abort timeout interval
write-delay  Configure delay timer for writes to URL
```

由于 flash 空间非常有限，因此建议将文件存储在 TFTP 服务器上。当部署在 TFTP 服务器上时，注意先创建一个空的文件以对应配置命令中的 URL，这个文件用于 dhcp snooping binding database 的写入（视具体设备而定）。

形成的文件的格式如下：

```
<initial-checksum>
TYPE DHCP-SNOOPING
VERSION 1
BEGIN
<entry-1>    <checksum-1>
<entry-2>    <checksum-1-2>
...
...
<entry-n>    <checksum-1-2-..-n>
END
```

下面是一个例子：

```
2bb4c2a1
TYPE DHCP-SNOOPING
VERSION 1
BEGIN
192.1.168.1 3 0003.47d8.c91f 2BB6488E interface-id 21ae5fbb
192.1.168.3 3 0003.44d6.c52f 2BB648EB interface-id 1bdb223f
192.1.168.2 3 0003.47d9.c8f1 2BB648AB interface-id 584a38f0
END
```

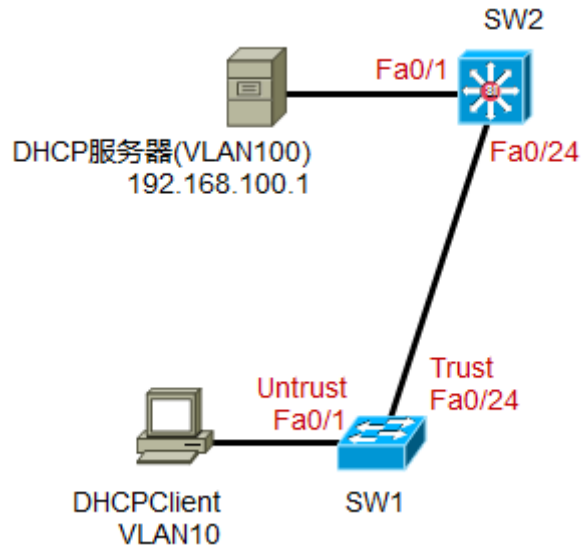
6.3.4 配置示例

1. 实验 1：基础实验

PC 属于 VLAN10，网关在 SW2 上。DHCPserver 属于 VLAN100，网关在 SW2 上。

SW1 为接入层交换机，部署 DHCP snooping，将 Fa0/24 定义为 trust 接口。

SW2 为核心层交换机，部署 DHCP relay



DHCPserver 的配置如下：

```

no ip routing
ip default-gateway 192.168.100.254
Interface fast0/0
    ip address 192.168.100.1 255.255.255.0
    no shutdown
    exit
Service dhcp
Ip dhcp pool vlan10
    network 192.168.10.0 /24
    default-router 192.168.10.254

```

SW1 的配置如下：

```

vlan 10
!
ip dhcp snooping                !! 全局开启 DHCP snooping
ip dhcp snooping vlan 10        !! vlan10 激活 DHCP snooping
no ip dhcp snooping information option  !! 不写入 option82
!
Interface fast0/1

```

```
switchport access vlan 10
interface fast0/24
switchport trunk encapsulation dot1q
switchport mode trunk
ip dhcp snooping trust
```

SW2 的配置如下：

```
vlan 10
vlan 100
name Server
!
Interface fast0/1
switchport access vlan 100
interface fast0/24
switchport trunk encapsulation dot1q
switchport mode trunk
Interface vlan 10
ip address 192.168.10.254 255.255.255.0
ip helper-address 192.168.100.1
Interface vlan 100
ip address 192.168.100.254 255.255.255.0
```

实验结果：

PC 获取到了地址，在 SW1 上查看 dhcp snooping binding database：

SW1#show ip dhcp snooping binding

MacAddress	ipAddress	Lease(sec)	Type	VLAN	Interface
00:B0:64:04:09:99	192.168.10.1	86025	dhcp-snooping	10	FastEthernet0/1

Total number of bindings: 1

2. 实验 2：接入交换机插入 option82，核心交换机做中继

在 SW1 上开启 DHCP snooping，同时不去 no ip dhcp snooping information option，也就是插入 option82。其他设备配置与上一个实验相同，发现 PC 机无法获取地址，通过在网关设备上，也即是 SW2 上 debug 会发现报错 relay information option exists, but giaddr is zero。giaddr 字段意思为该报文做经过的第一个 DHCP 中继代理的 IP 地址。

在 SW2 上增加配置 : (全局配置模式) ip dhcp relay information trust-all

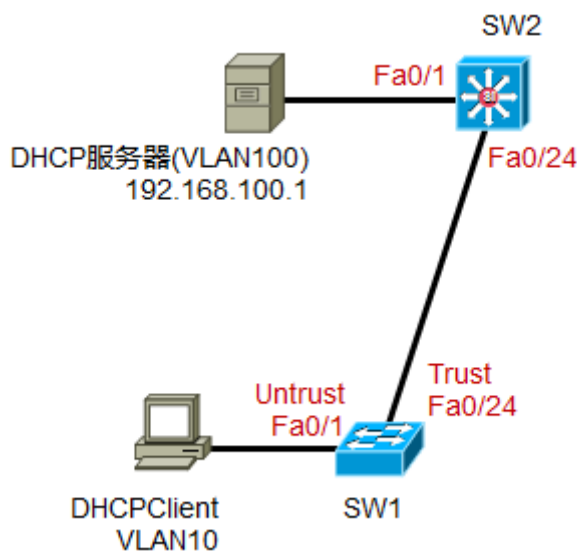
或 :

Interface vlan 10

ip dhcp relay information trusted

上述命令, 将使交换机接受携带 option82 信息, 同时 giaddr 为 0 的 dhcp 报文, 然后做中继。

如此一来即可成功获取地址。



SW1 的配置如下 :

```

vlan 10
!
ip dhcp snooping
ip dhcp snooping vlan 10
Interface fast0/1
    switchport access vlan 10
interface fast0/24
    switchport trunk encapsulation dot1q
    switchport mode trunk
    ip dhcp snooping trust
  
```

SW2 的配置如下 :

```

vlan 10
vlan 100
    name Server
!
  
```

ip dhcp relay information trust-all

Interface fast0/1

switchport access vlan 100

interface fast0/24

switchport trunk encapsulation dot1q

switchport mode trunk

Interface vlan 10

ip address 192.168.10.254 255.255.255.0

ip helper-address 192.168.100.1

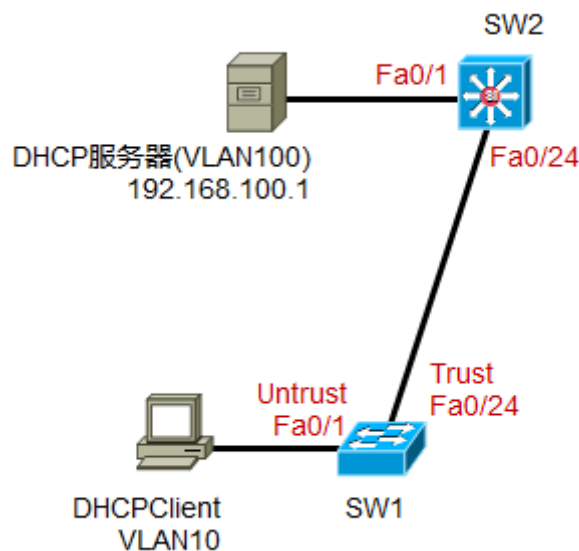
Interface vlan 100

ip address 192.168.100.254 255.255.255.0

3. 实验3：接入层交换机插入 option82，上游核心交换机也开启 dhcp snooping

接入层交换机 SW1 开启 dhcp snooping，插入 option82；

核心层交换机 SW2 也开启 dhcp snooping，默认所有端口都是 untrust 的，这时如果 SW2 的 fast0/24 口收到携带了 option82 的 dhcp 报文，则丢弃。解决办法之一，是将 SW2 的 fast0/24 口配置为 trust 接口，但是这么一来通过该 trust 接口收到的 dhcp 报文，是不会创建 dhcp snooping binding 表项的。另一个方法是，使用命令：ip dhcp snooping information option allow-untrusted。这条命令放行携带了 option82 的 DHCP 报文，同时会在本地创建 dhcp snooping binding 表项。



SW1 的配置如下：

vlan 10

!

ip dhcp snooping

```
ip dhcp snooping vlan 10
Interface fast0/1
    switchport access vlan 10
interface fast0/24
    switchport trunk encapsulation dot1q
    switchport mode trunk
    ip dhcp snooping trust
```

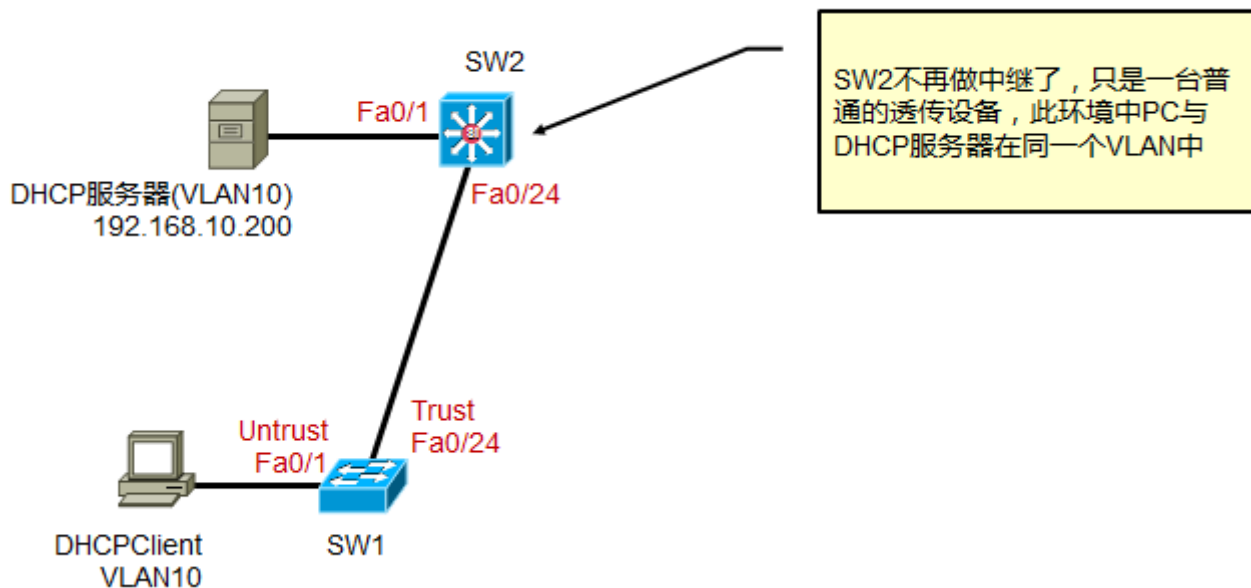
SW2 的配置如下：

```
vlan 10
vlan 100
    name Server
!
ip dhcp snooping information option allow-untrusted
ip dhcp relay information trust-all

!!注意区分上述两条命令

Interface fast0/1
    switchport access vlan 100
interface fast0/24
    switchport trunk encapsulation dot1q
    switchport mode trunk
Interface vlan 10
    ip address 192.168.10.254 255.255.255.0
    ip helper-address 192.168.100.1
Interface vlan 100
    ip address 192.168.100.254 255.255.255.0
```

4. 无中继环境中，option82 及相关问题



这个环境比较简单，PC 与 DHCP 服务器同处一个 VLAN，我主要想证实一下普通交换机对携带了 option82 的 DHCP 包的处理情况。

SW1 仍然开启 dhcp snooping，同时插入 option82

SW2 将服务器划入与 PC 相同的 VLAN，相当于纯做透传。

其中 SW1 的配置如下：

```
vlan 10
!
ip dhcp snooping
ip dhcp snooping vlan 10
Interface fast0/1
    switchport access vlan 10
interface fast0/24
    switchport trunk encapsulation dot1q
    switchport mode trunk
    ip dhcp snooping trust
```

SW2 的配置如下：

```
vlan 10
!
Interface fast0/1
    switchport access vlan 10
```

```
interface fast0/24
    switchport trunk encapsulation dot1q
    switchport mode trunk
Interface vlan 10
    ip address 192.168.10.254 255.255.255.0
```

DHCP server 的配置就是常规配置，这里不再赘述。

实验的结果是，PC 无法获取地址。

我们分析一下，SW1 将 PC 发出来的 DHCP 请求消息插入 option82，随后发给 SW2。通过 SW2 上的 debug 信息我们发现：

```
*Mar 16 17:09:21.858: DHCPD: inconsistent relay information.
```

```
*Mar 16 17:09:21.858: DHCPD: relay information option exists, but giaddr is zero.
```

SW2 意识到，DHCP 报文中有中继信息选项，也就是 option82，但是 giaddr 是全 0。SW2 虽然意识到了这个问题，但还是乖乖将这个数据交给了 DHCPserver，通过在 DHCPserver 上的 debug 信息：

```
*Mar  4 07:02:23.473: DHCPD: inconsistent relay information.
```

```
*Mar  4 07:02:23.473: DHCPD: relay information option exists, but giaddr is zero.
```

显然，DHCP 服务器也意识到了上面的问题，它采取的动作是，忽略这个 DHCP 请求。因此 PC 无法获取地址。

解决的办法之一，是 SW1 不插入 option82，另一个方法，是在 DHCPserver 上，使用配置：

（全局配置模式）ip dhcp relay information trust-all

或：

```
Interface fast0/0
    ip dhcp relay information trusted
```

思路和实验 2 一样。这样一来 SW2 会接受这些 DHCP 报文并进行处理，PC 就能够获取到地址了。

5. 其他配置

```
ip dhcp snooping limit rate rate
```

注意，在交换机开启了 DHCP snooping 后，由于对 dhcp 报文都要进行窥探，因此当网络中存在 DHCP 攻击行为，将会严重消耗交换机的性能，因此，可以通过该命令限制接口上允许收到的 DHCP 报文数量。Configure the number of DHCP packets per second that an interface can receive. The range is 1 to 2048. By default, no rate limit is configured.

```
ip dhcp snooping verify mac-address
```


Configure the switch to verify that the source MAC address in a DHCP packet received on untrusted ports matches the client hardware address in the packet. The default is to verify that the source MAC address matches the client hardware address in the packet.

```
ip dhcp snooping binding mac-address vlan vlan-id ip-address interface interface-id expiry seconds
```

静态配置一条 dhcp snooping binding database 表项，该命令是在特权模式下配置。

6.4 DAI - Dynamic ARP Inspection

6.4.1 ARP 协议原理

1. 协议概述

- Address Resolution Protocol
- 在以太网环境中，节点之间互相通信，需知晓对方的 MAC 地址
- 在现实环境中，一般采用 IP 地址标示通信的对象，而 ARP 的功能就是将 IP“解析”到对应的 MAC 地址。

2. 协议漏洞

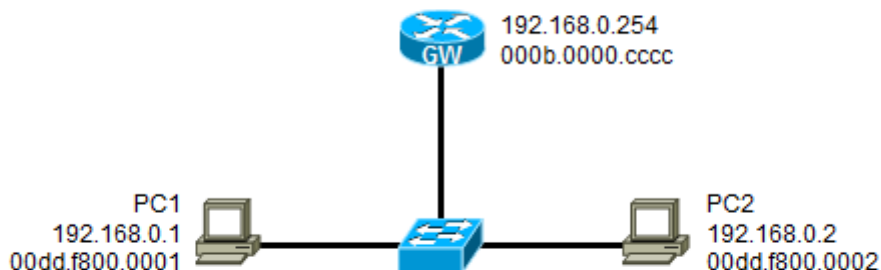
- 基于广播，不可靠
- ARP 响应报文无需请求即可直接发送，这给攻击者留下巨大漏洞
- 没有确认机制，任何人都可以发起 arp 请求或 response
- IPv6 中通过特定的机制规避掉 ARP 以及其漏洞

6.4.2 ARP 欺骗原理

1. 正常情况下各设备的 ARP 表项

show arp

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	192.168.0.254	-	000b.0000.cccc	ARPA	Fa0/0
Internet	192.168.0.1	-	00dd.f800.0001	ARPA	Fa0/0
Internet	192.168.0.2	-	00dd.f800.0002	ARPA	Fa0/0



arp -a

Internet Address	Physical Address	Type
192.168.0.2	00-dd-f8-00-00-02	dynamic
192.168.0.254	00-0b-00-00-cc-cc	dynamic

arp -a

Internet Address	Physical Address	Type
192.168.0.1	00-dd-f8-00-00-01	dynamic
192.168.0.254	00-0b-00-00-cc-cc	dynamic

2. 欺骗过程 ARP Request

show arp

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	192.168.0.254	-	000b.0000.cccc	ARPA	Fa0/0
Internet	ethernetII Header , src 00dd.f800.0001 dst FFFF-FFFF-FFFF				
Internet	Arp Request				
	SenderMac	00dd.f800.0001			
	SenderIP	192.168.0.254			
	TargetMac	00-00-00-00-00-00			
	TargetIP	192.168.0.2			



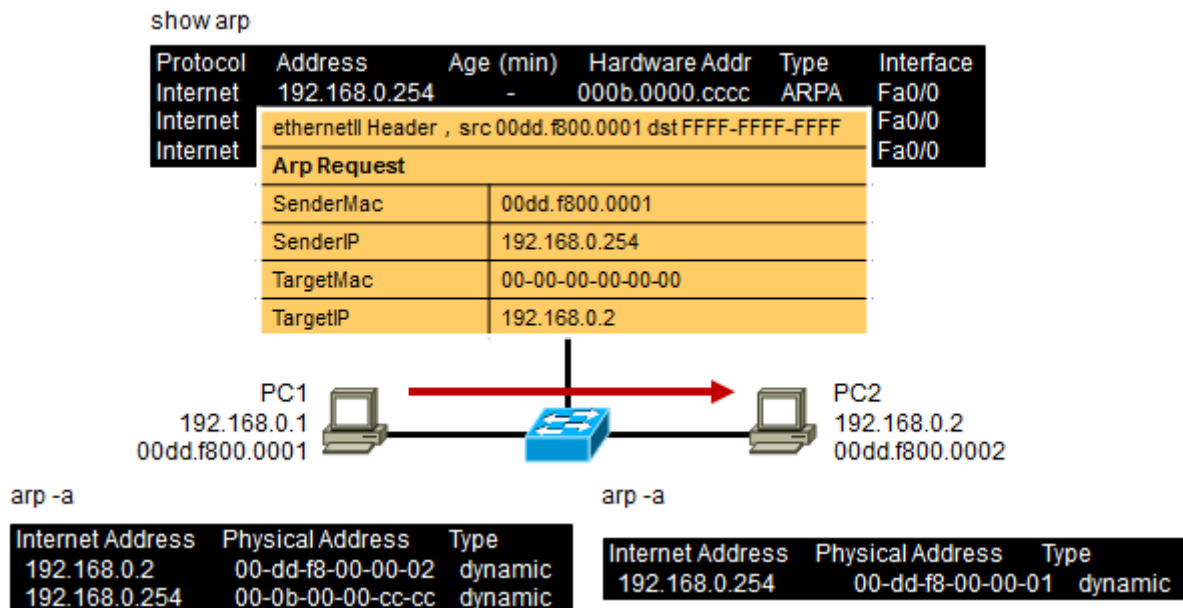
arp -a

Internet Address	Physical Address	Type
192.168.0.2	00-dd-f8-00-00-02	dynamic
192.168.0.254	00-0b-00-00-cc-cc	dynamic

arp -a

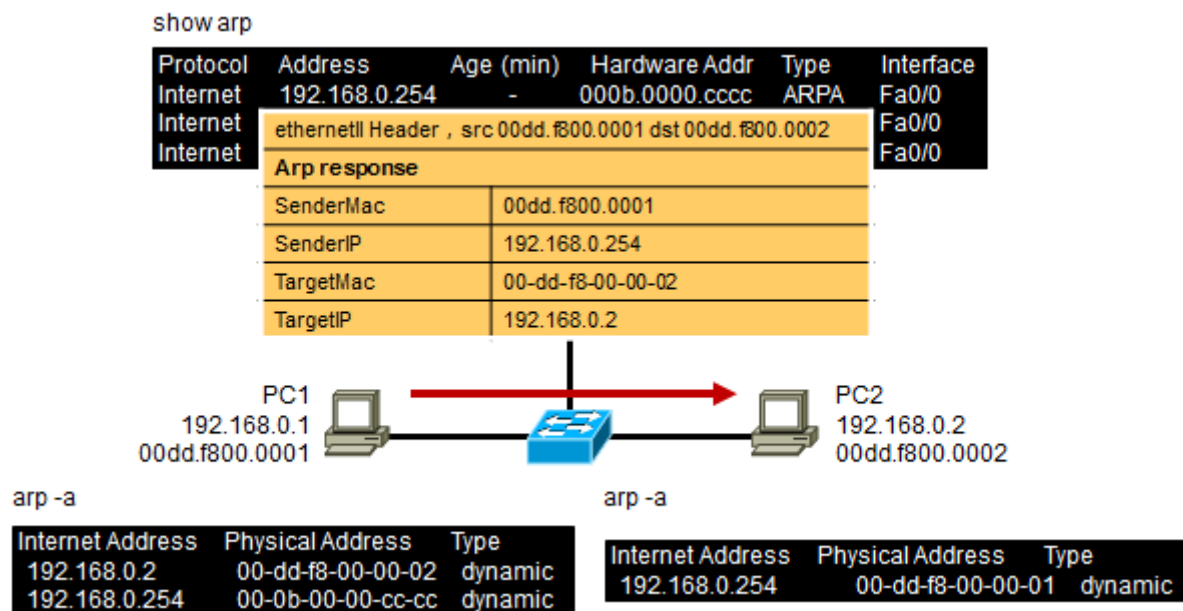
Internet Address	Physical Address	Type
192.168.0.1	00-dd-f8-00-00-01	dynamic
192.168.0.254	00-0b-00-00-cc-cc	dynamic

PC1 要欺骗 PC2，通过造包工具，如 anysends 等，发出一个 ARP request 广播，这个 ARP 数据帧的以太网头部中，源 MAC 为 PC1 的 MAC，目的 MAC 为全 F，关键内容在于 ARP 协议报文（ARP body），也就是这个数据帧的以太网帧头后的内容，这才是 ARP 的真正内容，里头包含的主要元素有：发送者 mac、发送者 IP、接收者 MAC 及接收者 IP。PC1 发出的这个 ARP 数据帧中，sender MAC 为自己的 MAC 地址，但是 SenderIP 为网关 IP 即 192.168.0.254，此刻，PC2 收到这个 arp 消息后，将更新自己的 arp 表项，如下：



如此一来，PC2 发送数据到自己的网关 0.254 时，由于网关的 MAC 已经被欺骗成了 PC1 的 MAC，因此数据都被转发给了 PC1，那么 PC2 就断网了。如果 PC1 再机灵点，将 PC2 发过来的数据再转给真实的网关，同时在本机运行一个报文分析工具窥探 PC2 发过来的数据，那么就可以在 PC2 不断网的情况下，神不知鬼不觉的窥探 PC2 的上网流量，这就是 the man in the middle，中间人攻击。

3. 欺骗过程 ARP Response



由于 ARP 的 response 并不需要 arp request 为先决条件即可直接发送，因此攻击者可以构造 arp reply 消息，并发送给被攻击对象，从而刷新被攻击者的 arp 表，同样能达到 arp 欺骗的目的。

4. 理解 invalid arp 表项

什么是 invalid arp 表项：

ARP 表中的 MAC 地址为零 (Windows 主机) 或 “No completed” (网络设备)

```
Ruijie#show arp
Total Numbers of Arp: 5
Protocol Address Age(min) Hardware Type Interface
Internet 192.168.1.1 <---> <No completed> arpa VLAN 1
Internet 192.168.1.254 -- 001a.a908.9f0b arpa VLAN 1
Internet 192.168.0.2 9 00d0.f800.0002 arpa VLAN 26
Internet 192.168.0.1 2 00d0.f800.0001 arpa VLAN 26
Internet 192.168.0.254 -- 001a.a908.9f0b arpa VLAN 26
```

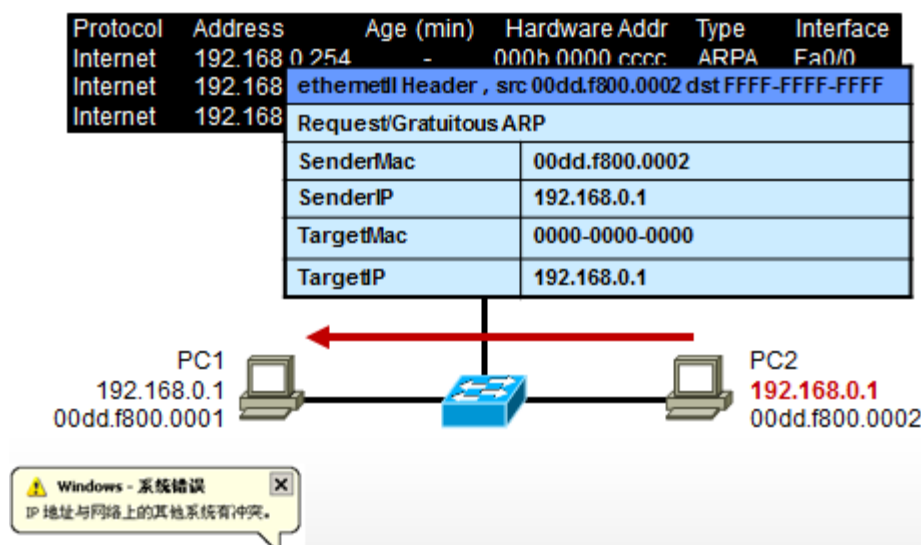
产生的原因：发送 ARP Request 后，为接收 ARP Reply 做准备

大量存在的原因：同网段扫描（主机）、跨网段扫描（网络设备）

5. Gratuitous ARP

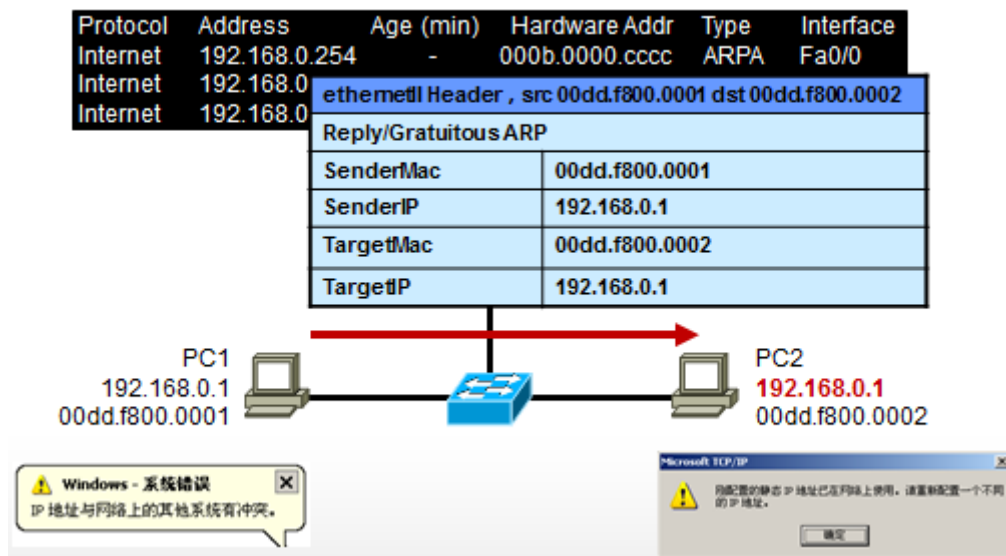
Gratuitous ARP，免费 ARP 是一种特殊的 ARP Request/response 报文，即 Sender IP 与 Target IP 一致（一般用于 IP 冲突检测）

我们看下面的例子：



PC2是新接入的主机，配置了 192.168.0.1 的地址，完成地址配置后，PC2向网络中发送一个 Gratuitous ARP，以防内网中有人使用相同的 IP 地址，这个 Gratuitous ARP 的 senderIP 和 target IP 都是自身，senderMAC 是自己的 MAC。

那么当 PC1 收到这个 Gratuitous ARP 后，由于 senderIP 和自己冲突了，于是在本地弹出一个气泡（如果是 windows 系统），同时回复一个 Gratuitous ARP response 以告知对端出现了 IP 地址重复。



对端收到这个 Gratuitous ARP 消息后，知道内网中已经存在这个 IP 地址的使用者了，于是告警。

Gratuitous ARP 的另外一个作用是，某些厂家用于防止 ARP 欺骗。例如在某些路由器上，配置了基于 Gratuitous ARP 的防 arp 欺骗解决方案，则该路由器（的接口）将以一定的时间间隔发送 Gratuitous ARP，目的就是刷新底下 PC 的 arp 表，以保持网关 IP 对应的 MAC 始终是正确的，当然，这种防 arp 的解决方案挺土鳖的。

6. 如何判断内网中是否存在 arp 欺骗

现在市面上其实已经有许多 arp 防火墙可以识别到，当然有一些简单的方法，例如 arp -a 看 MAC 是否正确。或者抓包等。

7. Arp 欺骗的解决方案

解决方案 1：手工绑定 IP+MAC

例如在 PC 上，用 arp -s 静态绑定网关 IP 及网关的 MAC，以防止 arp 欺骗，或者在关键节点的网络设备上，使用命令 arp 进行绑定。

显然这种方式扩展性太差。

解决方案 2：免费 arp

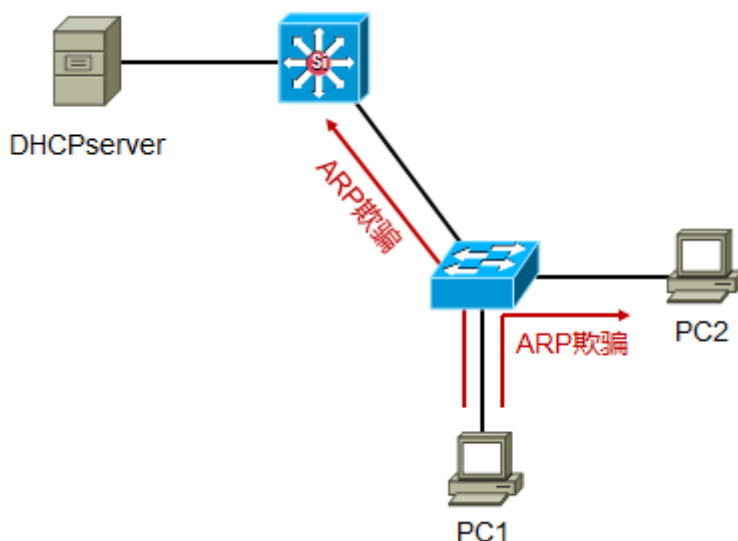
这个前面已经说过了

解决方案 3：DAI

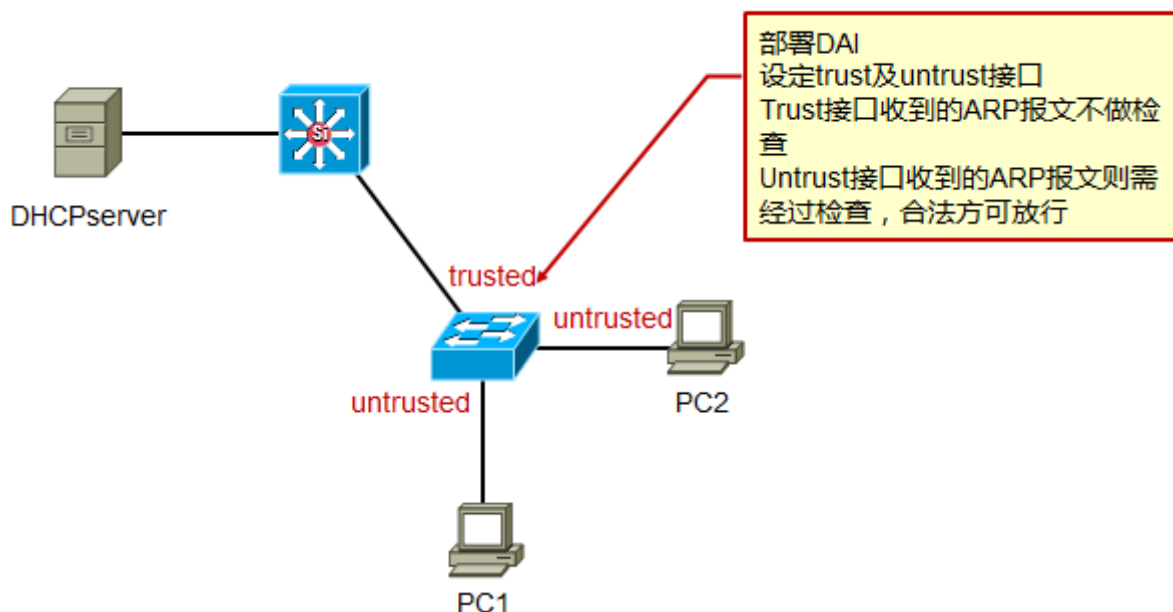
其他，其实防 ARP 欺骗的解决方案业内还是有许多的，每个厂家都有自己的方法。

6.4.3 DAI 工作机制

1. 机制概述

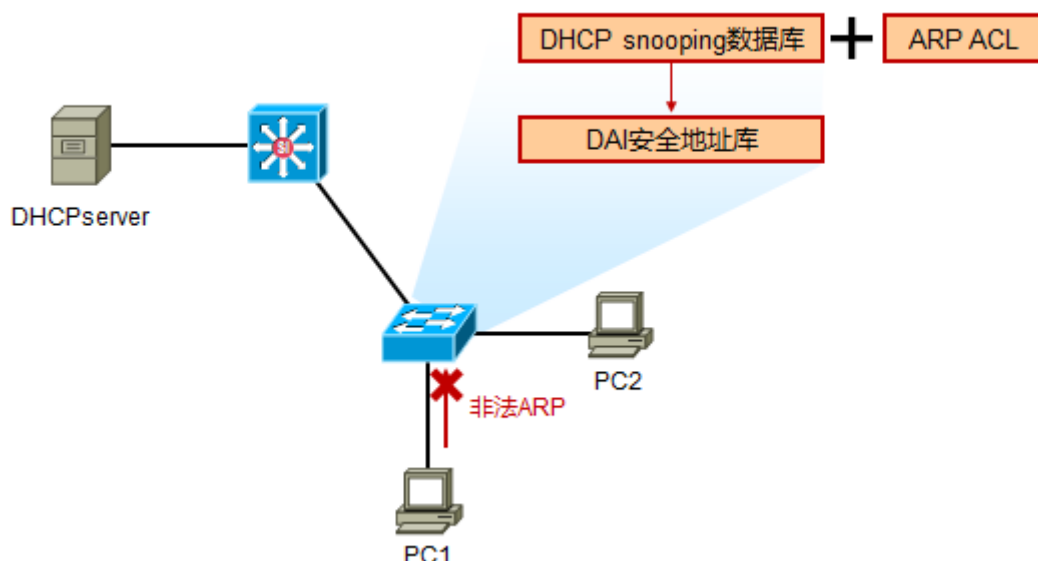


部署 DAI 前，内网如果出现 ARP 欺骗行为，例如 PC 发送非法的 ARP 报文，对于交换机而言，是无法检测并作出防御动作的。



部署 DAI 后，我们可以定义交换机接口的信任状态：trusted，或 untrusted，对于信任端口，将不对收到的

ARP 报文做检测，直接放行。而对于 untrusted 接口，ARP 报文在收到后回进行检查，只有合法的 ARP 报文才会被放行，如果出现非法的 ARP 报文，则会被 log，同时丢弃。



DAI 依赖 DHCP snooping 技术，在一个用户动态获取 IP 地址的网络环境中，我们通过部署 DHCP snooping 一来可以起到防御 DHCP 欺骗的效果，另一方面，会在交换机上得到 DHCP snooping binding database 这个数据库是 DHCP Snooping 侦听 DHCP 交互过程后的信息记录，里头包含客户端的 IP、MAC、所属 VLAN 等等相关信息，而这些信息，恰恰可以作为 ARP 合法性校验的依据。

DAI 部署于交换机上，用于确保合法的 ARP request 或 response 被放行而非法的 ARP 消息被丢弃。

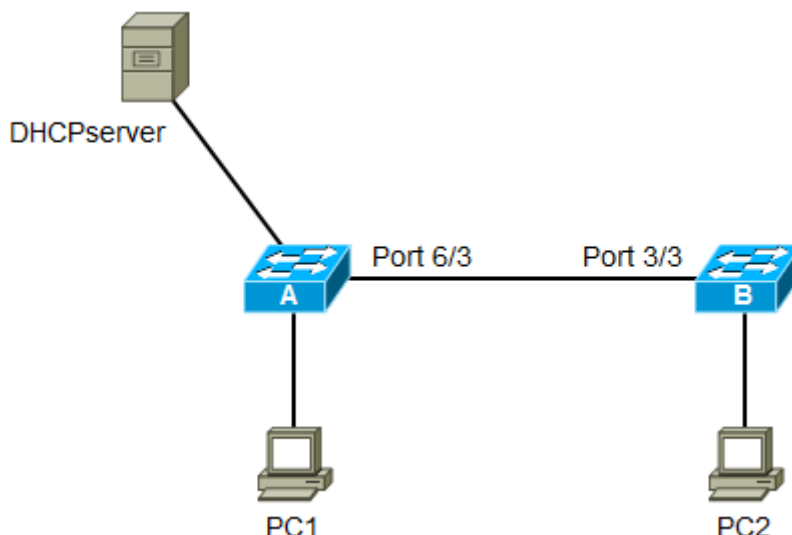
交换机部署 DAI 后的主要动作如下：

- 在 DAI untrust 接口上（注意与 DHCP snooping 的 untrust 接口区分开）阻拦一切 ARP requests 或 response 消息并作校验
- 在更新自己 ARP 表或将收到的 ARP 消息转发出去之前先进行合法性校验，主要看 ARP 报文中的 IP 及 MAC 对应关系是否合法
- 丢弃非法的 ARP 报文，交换机会在丢弃非法的 ARP 后进行 log

前面说了 DAI 借助于 DHCP snooping 的绑定数据库进行 ARP 合法性校验。另一个合法性校验的依据是手工配置的 ARP ACL。

你也可以配置 DAI 来丢弃那些以太网帧头源 MAC 与 ARP body 里的 MAC 不一致的非法 ARP。这个后面有做进一步的介绍。

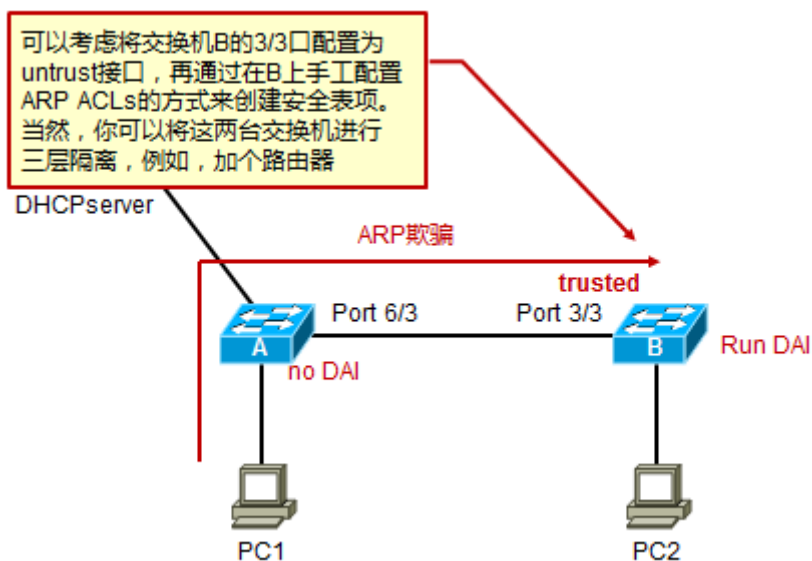
2. 接口信任状态及网络的安全问题



一般来说，我们认为攻击行为多来自于终端的 PC，可能是中毒或者黑客行为等。所以我们常会将连接终端 PC 的接口配置为 DAI untrust 接口，而交换机之间的接口配置为 trust 接口。

看上图，我们假设交换机 A 和 B 都运行 DAI，而 PC1 及 PC2 都通过 DHCP 服务器获取地址，那么这时候对于 PC1 而言，只有在交换机 A 上的 DHCP snooping database 中才有其 IP+MAC 的绑定，交换机 B 则没有。因此如果 A 和 B 之间的接口为 DAI 的 untrust，则来自 PC1 的 ARP 报文，将被交换机 B 丢弃，因为它认为这些报文是非法的，如此一来，PC1 和 2 的通信就出问题了。所以建议 A 和 B 之间的接口设置为 DAI 的 trust。

但是，并非所有的交换机互联接口设置为 DAI 的 trust 都会没有问题，在特定的环境下，这会留下一个安全隐患。例如，仍然是上面的图，假设 A 交换机不支持 DAI，如果 B 的 3/3 口配置为 trust，那么 PC1 就可以大摇大摆的去对 B 和 PC2 进行 ARP 欺骗了，即使 B 运行了 DAI。所以 DAI 仅仅是保证运行 DAI 的交换机本身所连接的终端 PC 不能进行非法的 ARP 动作。像刚才所述的情况，可以考虑将交换机 B 的 3/3 口配置为 untrust 接口，再通过 B 上手工配置 ARP ACLs 的方式来创建安全表项。当然，你可以将这两台交换机进行三层隔离，例如，加个路由器神马的。



3. ARP 报文的 rate limit

对于 DAI 而言,是需要损耗交换机 CPU 资源的,如果开启了 DAI,就有可能会成为拒绝服务攻击的对象。所以我们对于运行了 DAI 的交换机接口,有 ARP 报文的限制。默认 DAI untrust 接口的 rate limit 是 15 个 P/S 也就是 15pps, trust 接口则完全没有限制。可以通过 `ip arp inspection limit` 这条接口级的命令来修改。

当接口收到的 ARP 报文超出这个阈值,接口将进入 `err-disable`。可以使用 `shutdown no shutdown` 的方式手工重新恢复这个接口。或者,使用全局命令 `the errdisable recovery` 来让接口在一定的时间间隔后自动恢复。

4. ARP ACL 及 DHCP snooping database 条目的相对优先级

ARP ACL 的优先级高于 DHCP snooping database 条目。在你使用全局命令 `ip arp inspection filter` 指定了 ARP ACL 后,如果 ARP 报文被 ARP ACL deny 掉了,那么这些报文就被直接丢弃,即使在 DHCP snooping database 中有合法的表项匹配这些 ARP 报文。

6.4.4 DAI 的配置方针

1. DAI 是 ingress 安全特性,不会做 egress 的安全校验
2. DAI 的工作需依赖 DHCP snooping
3. 如果 DHCP snooping 被关闭或者,这是一个无 DHCP 的网络环境,例如纯静态 IP 地址的环境,使用 ARP ACLs 来放行或丢弃 ARP 报文

4. DAI 在 access 接口, trunk 接口, EtherChannel 接口, 及 private VLAN 接口上都支持

6.4.5 DAI 配置命令

1. 基本配置

```
ip arp inspection vlan {vlan_ID | vlan_range}
```

激活特定 VLAN 的 DAI

```
ip arp inspection trust
```

接口模式，将接口配置为 DAI trusted，默认为 untrusted

2. 应用 ARP ACLs 到 DAI

```
ip arp inspection filter arp_acl_name vlan {vlan_ID | vlan_range} [static]
```

将配置好的 ARP ACL 应用到 DAI

如果不使用 static 关键字，则 ARP 报文会先被 ARP ACL 匹配，如果没有任何匹配项，则再被 DHCP Snooping database 的安全表项再做一次校验，如果有合法表项，则放行，如果没有，则丢弃。

如果使用 static 关键字，则意味着 ARP ACLs 启用隐含 deny any 机制。也就是说如果 ARP 报文与 ARP ACLs 匹配，而结果是没有任何匹配项，则直接丢弃，不管 DHCP Snooping database 里是否有合法的条目。

3. 配置 ARP 报文 rate limit

```
ip arp inspection limit {rate pps [burst interval seconds] | none}
```

接口级命令。通过限制接口上收到的 ARP 报文的数量，可以有效的防止开启 DAI 的交换机被 DoS 攻击。

注意事项：

- 默认在 DAI untrusted 接口上 rate 是 15pps，而 trusted 接口则无限制
- Ip arp inspection limit rate none，是无限限制
- Burst interval，默认是 1s，可选配置。这是一个连续的检测时间段，用来检测在这个时间段内的 ARP 报文数量，默认是 1S，所以就是说 1S 内，如果超出了 15 个（默认）ARP，则违例。这个时间可选区间为 1-15S
- 当出现违例，接口进入 err-disable 状态

errdisable recovery cause arp-inspection

激活由于 arp-inspection 违例导致的 err-disable 接口的自我恢复。

4. 配置附加的校验动作

```
ip arp inspection validate { [dst-mac] [ip] [src-mac] }
```

我们可以通过配置一些额外的选项，让 DAI 做进一步的校验。

有三个可选关键字，dst-mac、ip、src-mac

- dst-mac 检测收到的 ARP 报文中，以太网帧头的目的 MAC 与 ARP body 里的 target MAC 是否一致，这个 check 针对 ARP response。当开启这个选项后，如果两个 MAC 不一致，则 ARP 报文被丢弃
- ip 检测 ARP body 里的 IP 地址是否是无效或非预期的，如 0.0.0.0、255.255.255.255，以及组播 IP 地址，这些都被认为是无效的 IP。SenderIP 是无论报文为 ARP request 或 response 都会进行校验。而 targetIP 只有在报文是 ARP response 时才会进行校验
- src-mac 检测收到的 ARP 报文中，以太网帧头的源 MAC 与 ARP body 里的 sender MAC 是否一致。这个校验动作在 ARP request 及 response 报文中都会进行。

5. 配置 DAI logging

当 DAI 丢弃一个 ARP 报文，交换机会在 log buffer 里存储一条信息，随后产生一条 system message 系统信息，在这条系统信息产生后，交换机从 log buffer 里清除之前存储的条目。每一个条目都包含例如 VLAN、端口号、源目的 IP 地址、源目的 MAC 等信息。

一个存储在 log buffer 中的条目可以代表多个报文，例如交换机从同一个接口同一个 VLAN 收到具有相同 ARP 元素的 ARP 包，那么这些报文对应一个条目，以节省 buffer 的占用。同时只会产生一条系统信息。

```
ip arp inspection log-buffer entries number
```

配置 DAI 占用 log buffer 的大小（最大条目数量），区间是 0-1024

默认是 32 条

```
ip arp inspection log-buffer logs number_of_messages interval length_in_seconds
```

配置 DAI login system message，系统消息将以每 **length_in_seconds** 发送 **number_of_messages** 的速率来约束系统消息的发送。**number_of_messages** 默认为 5，范围是 0-1024，如果配置为 0 则表示条目只出现在 log buffer 中而不产生系统消息；**length_in_seconds** 默认是 1S，范围是 0-86400（1 天），如果配置为 0 则表示系统消息将会立即产生，而 log buffer 中不再存储条目，如果将 **length_in_seconds** 配置为 0，则 **number_of_messages** 也会变为 0。

```
ip arp inspection vlan vlan_range logging {acl-match {matchlog | none} | dhcp-bindings {all | none | permit}}
```

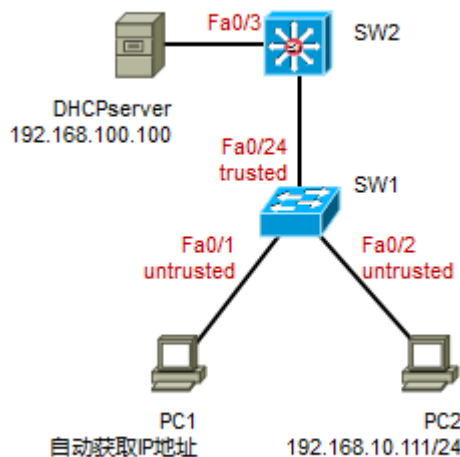
默认情况下，只要 DAI 丢弃报文，则会被 logged

- **acl-match matchlog**—Logs packets based on the DAI ACL configuration. If you specify the
- **matchlog** keyword in this command and the **log** keyword in the **permit** or **deny** ARP access-list configuration command, ARP packets permitted or denied by the ACL are logged.
- **acl-match none**—Does not log packets that match ACLs.
- **dhcp-bindings all**—Logs all packets that match DHCP bindings.
- **dhcp-bindings none**—Does not log packets that match DHCP bindings.
- **dhcp-bindings permit**—Logs DHCP-binding permitted packets.

```
ip arp inspection vlan 100 logging acl-match none
```

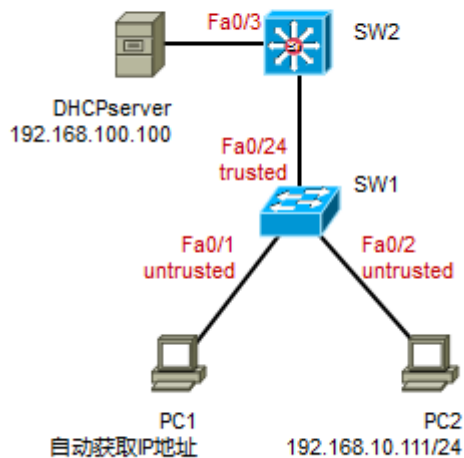
意思是针对 VLAN100 中的流量，不去 log 被 ARP ACL 匹配丢弃的报文

6.4.6 DAI 配置范例



- 完成基本配置
- 在SW1上开启DHCP snooping，PC1能够获取到地址，查看DHCP snooping binding database
- PC1能够ping通PC2
- 在SW1上开启DAI，配置接口信任状态
- PC1无法ping通PC2
- 在SW1上创建静态的DHCO snooping表项 / 再测试手工ARP ACL 两种方法均要使得PC1及2之间能够通信
- 修改PC1的IP地址，再去访问PC2
- 限制fast0/1口的arp消息门限，以防DoS攻击

基本配置如下：

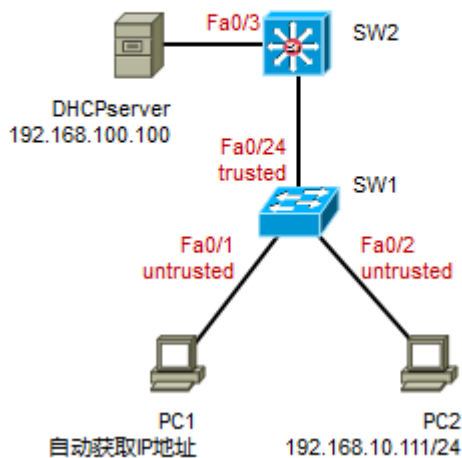


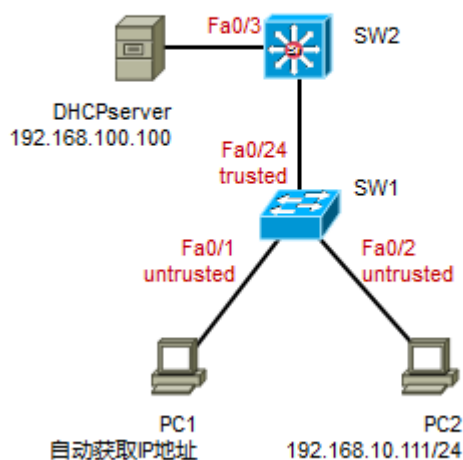
DHCPserver的配置如下：

```
no ip routing
!
ip default-gateway 192.168.100.254
!
Interface fast1/0
ip address 192.168.100.100 255.255.255.0
no shutdown
exit
!
Service dhcp
!
Ip dhcp pool vlan10
network 192.168.10.0 /24
default-router 192.168.10.254
```

SW2的配置如下：

```
vlan 10
vlan 100
!
Interface fast0/3
switchport mode access
switchport access vlan 100
interface fast0/24
switchport trunk encapsulation dot1q
switchport mode trunk
Interface vlan 10
ip address 192.168.10.254 255.255.255.0
ip helper-address 192.168.100.100
Interface vlan 100
ip address 192.168.100.254 255.255.255.0
```





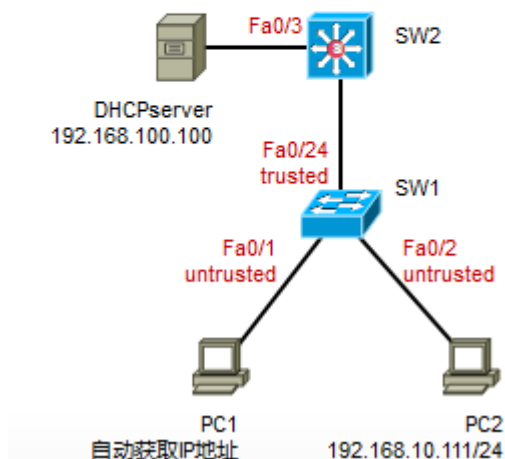
SW1的配置如下：

```
vlan 10
!
Interface range fast 0/1 -2
 switchport mode access
 switchport access vlan 10
Interface fast 0/24
 switchport trunk encapsulation dot1q
 switchport mode trunk
!
ip dhcp snooping
ip dhcp snooping vlan 10
no ip dhcp snooping information option
!
interface fast0/24
 ip dhcp snooping trust
```

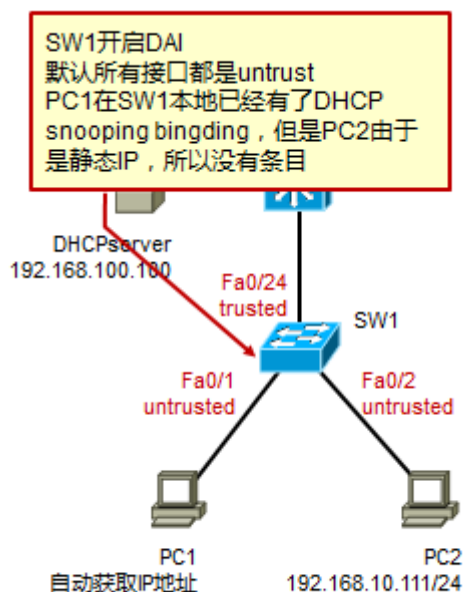
基本配置完成后，来看一下：

SW1#sh ip dhcp snooping binding

MacAddress	IpAddress	Lease(sec)	Type	VLAN	Interface
00:B0:64:04:09:A1	192.168.10.2	86366	dhcp-snooping	10	FastEthernet0/1
Total number of bindings: 1					



接下去在 SW1 上开启 DAI：



SW1的增加配置如下：

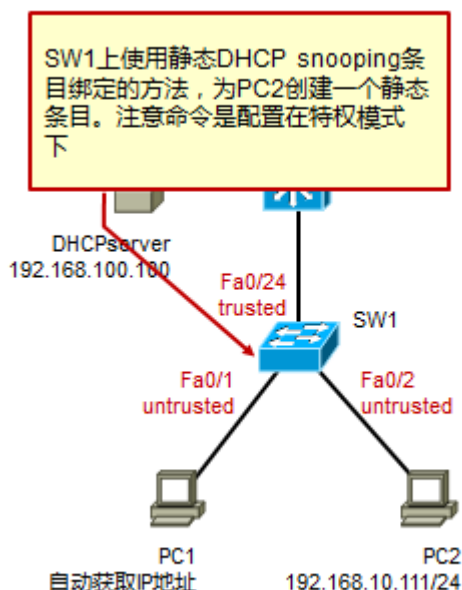
```
ip arp inspection vlan 10
!
Interface fast 0/24
ip arp inspection trust
```

SW1上立即出现如下系统消息：

```
*Mar 1 00:25:18.827: %SW_DAI-4-
DHCP_SNOOPING_DENY: 1 Invalid ARPs
(Res) on Fa0/2, vlan
10.([000b.5f35.70a1/192.168.10.111/00b0.64
04.09a1/192.168.10.2/00:25:18 UTC Mon
Mar 1 1993])
```

PC1与PC2无法通信

为了使得 PC2 发出的 ARP 消息能够被放行，我们首先使用手工配置 DHCP snooping binding 表项的方法：



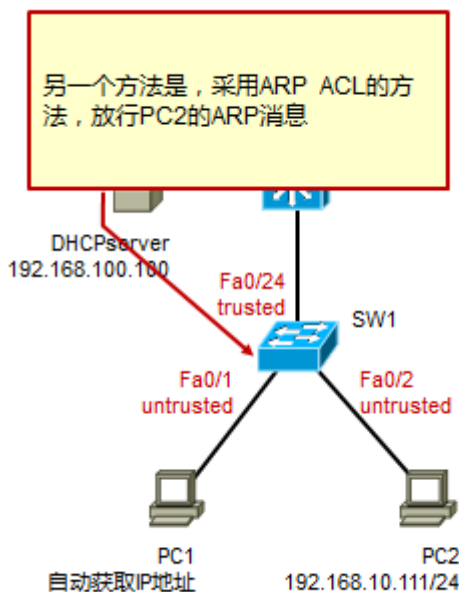
SW1的增加配置如下：

```
ip dhcp snooping binding 000b.5f35.70a1
vlan 10 192.168.10.111 interface fast 0/2
expiry 1000
```

SW1上使用show ip dhcp snooping binding
可以看到手工创建的绑定

如此一来PC1即可获取到PC2的MAC，从而ping通PC2

另一个方法，是用 ARP ACL：



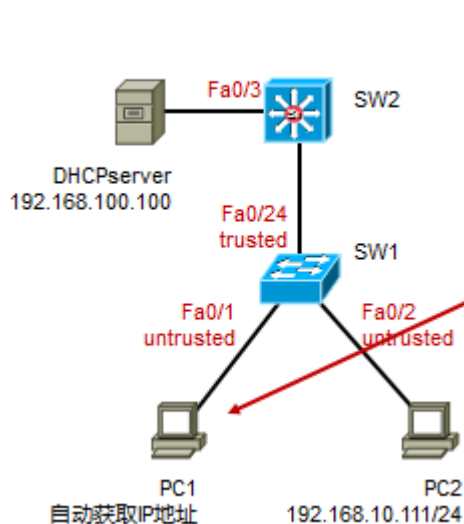
SW1的增加配置如下：

```
arp access-list ccie
permit ip host 192.168.10.111 mac host 000b.5f35.70a1
!
ip arp inspection filter ccie vlan 10
```

有了ARP ACL，同时应用到DAI上，PC2的ARP消息在进入untrusted接口fa0/2后，会被ARP ACL ccie所匹配并放行。

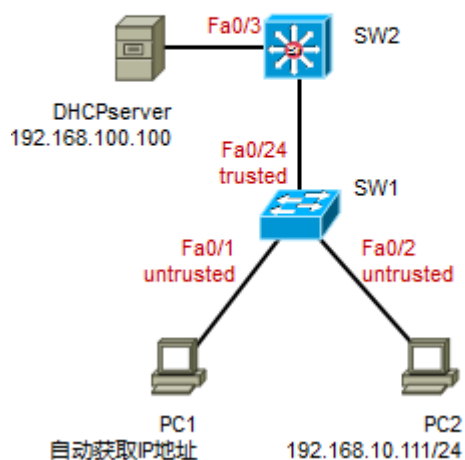
PC1及PC2即可通信

再来做一个测试，就是PC1私自修改IP地址：



修改PC1的IP地址，PC1无法正常上网，因此DAI可以防止内网私设IP。

由于 DAI 开启后，下联的 PC 如果段时间发送大量的无效 ARP，可能会导致交换机性能大量损耗，也就是 DoS 攻击。那么建议在接口上限定 ARP 消息门限。



SW1的增加配置如下：

```
errdisable recovery cause arp-inspection
errdisable recovery interval 30
Interface fast 0/1
ip arp inspection limit rate 10
```

为了防止开启了DAI的交换机受到DoS攻击，建议在接口上限制ARP消息的数量，当超出这个阈值，接口回被err-disable，搭配errdisable recovery 命令，可以使得交换机接口能够在一定时间后自我恢复（前提是违例行为停止）。