

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Computer Networking |
| **Semester**: | 2nd semester of the academic year **2023-2024** |
| **Major**: | Software Engineering |
| **Class**: | 2022 |
| **Student Name**: | 吴孜远 |
| **Student ID:** | 222022321062009 |
| **Teacher:** | Chunming Wu (吴春明) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | 路由实验 | | | |
| Date | | 2024.5.15 | Type | | □Confirmatory  √ Design  √ Comprehensive |
| 1. **Objective & Requirements**   掌握直连路由与静态路由、路由协议RIP和OSPF、以太网交换、VLAN配置和通信。 | | | | | |
| 1. **Experimental environment (**platform and software**)**   Windows11  eNSP  wireshark | | | | | |
| 1. **Experimental content and design** (Main Content, Procedure, Codes and Results) 2. 网络设备基本操作   拓扑1：  只有同一网段的两台pc和一个交换机：    两台pc配置如下：    为什么这里不配置网关呢？  我觉得不用配置网关啊，两台pc都是一个网段下的，链路层交换机可以直接转发frame  然后启动三台设备，在pc1中ping一下pc2看看能不能ping通：    显然可以，那么我们再调试一下交换机看看：    首先看到我执行了两次disp mac-address命令，第一次什么都没显示，因为这只是一张缓存表，一段时间不ping，缓存会被清除，自然命令没用了，所以我相互ping了一下两台pc，才输出mac表。  可以看到交换机中记录了两台机器的mac address和对应的port  交换机是如何得知两个主机的MAC地址的？  交换机通过记录端口接收frame中的源MAC address和port的对应关系来进行MAC地址表学习，有了这两个值的映射，交换机就可以基于目的MAC address进行转发，效率高。  接下来我们进入GE0/0/1接口视图    输入disp int gi0/0/1命令，这个命令完整写法是disp interface GigabitEthernet0/0/1，这里我们可以看到这个接口的流量统计等。  我这里发送的单播包有10个（因为我第一次ping的时候没ping通，如果没有这一次，应该是5个）而广播包有11个，为什么呢？  拓扑2：  有两个pc，但在不同网段，通过一个路由器通信。    两台pc的配置如下：    观察与上一实验配置的区别，思考为什么？  首先这一次我们将两台pc放在两个网段中，这样才需要路由设备转发，我们的实验就是针对路由设备的嘛  其次，我们配置了网关，给我们的pc一个连接外网的出口，这样才能互相通信。  然后我们配置路由器：  interface e0/0/0 (int e0/0/0)：进入接口Ethernet0/0/0视图  ip address 2.0.0.2 255.0.0.0 (ip ad 2.0.0.2 8)：设置接口ip    然后我们看一下端口信息：  可以看到路由器接口ip都配置好了。    接下来我们测试一下：  打印RTA的routing-table，发现已经配置好了：如2.0.0.0/8网段连接到0.0.0端口    在主机2.0.0.2中Ping3.0.0.2主机，发现可以ping通：说明我们的RTA已经能够识别3.0.0.0网段。    我们再抓一下RTA的0/0/0接口的包看看：    拓扑3：  建立如下拓扑：    Pc的ip和网关配置如上图标记所示，这里就不再单独展示  配置路由器两个接口的ip，过程与上个实验一致，这里我就只展示最后的routing-table：    然后我们用pc 10.0.0.1来分别ping 10.0.0.2和20.0.0.1：    10.0.0.1和10.0.0.2毕竟在一个网段下，通过交换机就可以转发packet，当然可以ping通。    20.0.0.1也可以ping通，说明我们的路由配置成功了，路由器成功转发了来自10  .0.0.0网段的包。  抓包：    也是抓到了两次ping发送和接收的所有包   1. 静态路由与子网划分   实验1：直连路由  跟上个实验的组网一样，我们就用这个拓扑：    Pc配置好了ip和gw，路由器接口ip也配置好了  路由表如下：    这里两个网段分别连到了路由的两个接口。这里显示proto为direct，也就是说是直连路由，意思是这两个网段直接通过接口连接到路由器。  为什么会通？  当路由器收到了发往网段2.0.0.0的请求（可能来自网段3.0.0.0），就能在路由表中匹配到，知道下一步转发到哪个端口ip。  实验2：IP子网划分  判断两个IP是否在同一个网段，最关键的还是要看mask。如果是mask为28，那么32位mask中前28位都是1，也就是255.255.255.240，一个网段中只有16个ip。  路由器的两个接口ip**不能**在同一网段中（这样做也没啥意义），所以如果e0/0/0为192.168.1.11/28，那么他所在的网段是192.168.1.1-192.168.1.14，e0/0/1不能为192.168.1.13。  根据这个拓扑回答问题：    PCA与AR1通否？ 不通，因为不在同一个网段  修改AR1的E0/0/0口IP为192.168.1.14/28，通否？ 通了  AR1的E0/0/0口IP在哪个范围内可通？ 192.168.1.1-192.168.1.14（全0全1不要）  实验3：静态路由  配置好了以下拓扑：    当PC1、PC2均不配置网关时：  PC1>ping 192.168.0.1（通，为什么？ 在同一网段）  PC1>ping 2.0.0.1（不通，为什么？ 没有配置网关，pc不知道将这个网段的包发给谁）  为PC1配置网关192.168.0.1 ：  PC1>ping 2.0.0.1（通，为什么？ 配置了网关，pc1知道先将包发给192.168.0.1，然后由路由器转发到2.0.0.1；路由表中也有发往192.168.0.0/24网段的包的下一跳，也能发回去，所以通了）  PC1>ping 2.0.0.2（不通，为什么？ 包可以到达2.0.0.2，因为R1路由表上规定了，但是R2路由器接收到来自2.0.0.2的包后不知道将这个响应的192.168.0.0/24网段的包转发到哪个接口ip去）  在R2上配置静态路由[R2]ip route-static 192.168.0.0/24 2.0.0.1    PC1>ping 2.0.0.2（通，为什么？ 路由器R2的路由表中静态配置了发往192.168.0.0/24网段的包的下一跳，也就是2.0.0.1，2.0.0.1发往接口2.0.0.2）  PC1>ping 3.0.0.1（不通，为什么？ 路由器R1不知道发往3.0.0.0/8网段的包应该转发到哪一个接口ip）  在R1上配置静态路由[R1]ip route-static 3.0.0.0 8 2.0.0.2    PC1>ping 3.0.0.1（通，为什么？ 因为R1路由表中静态配置了发往3.0.0.0/8网段的包的下一跳，也就是2.0.0.2，从2.0.0.1接口发出）  PC1>ping 3.0.0.2（不通，为什么？ 因为3.0.0.2没有配置网关，无法响应，不知道应该将192.168.0.0/24网段的包如何发出）  为PC2配置网关3.0.0.1 ：  PC1>ping 3.0.0.2 （通，为什么？ 3.0.0.2配置了网关，将不同网段的包都发到路由器R2）  2.0.0.0/30网段为何配置30位长掩码？  为了避免路由环路  如果我们的掩码小于30，假设目地IP为2.0.0.3，那么我们可以看上面的路由表，R1会将这个包转发到2.0.0.1接口，R2收到这个包会将这个包转发到2.0.0.2接口发给R1，就这么来回转发。这是由于PPP协议的特殊性：没有ARP查找机制，无论任何包都要发向PPP对方节点（有路由的前提下）。  所以我们直接用只有2位可用ip的30位掩码来解决这个问题，不会出现第三个此网段的ip。  实验4：路由环路  在实验3的基础上，配置两个默认路由  RT1：    RT2：    接下来我们在PC1中tracert 一个不存在的网段的ip：4.0.0.1。    这个包被两个路由器来回转发，形成回路，就是因为它们都不知道这个网段应该如何转发，所以都按照默认路由。  如何避免路由环路在实验3中有说明。  作业：  配置如下拓扑：    如何配置默认路由来使得pc3能ping通pc4？  首先第一想法：只要使得每个路由器能够识别到所有网段就好了嘛，我们来试一试：  对于两个与pc连接的路由器，配置很简单，直接将收到的不认识的包（对R3来说是30 40 50，对R6来说是10 20 30）默认发到它认识的路由器就好了，剩下的交给中间两个路由器。  对于R4，它已经有了20和30的直连路由，我们要考虑将发往10.0.0.0网段的包直接传给R3，至于其它的（40和50），我们就全都发给R5。  对于R5，它已经有了30和40的直连路由，我们考虑将50可以直接传给R6，剩下的10 和20就全部发给R4。    Ping通还是意料之中的，  但是这样有个问题，如果说我们现在pc1发送了一个完全未知的网段（70）的包，则会造成路由环路：    原因我们可以看到在30网段，R4和R5互相转发这个未知的包，都是按照默认路由的规则。按照我们的配置，对于R4，它会让10 20 30 以外的网段都使用默认路由通过30传给R5；对于R5，它会让30 40 50 以外的网段都使用默认路由通过30传给R4。可以看到，这两个路由器包含了我们之前全部的网段。但是现在我们传70网段的包，这个包在两个路由器都使用默认路由，就会来回传输，形成闭环。  如何解决呢？  其实我们只要让闭环中的某一个路由器不配置默认路由或者不让两个路由器的默认路由的下一跳指向对方就可以。所以我们让R5做那个无所不知的路由器，如果包发到R5仍然无法在路由表上匹配到下一跳，就直接丢掉。具体操作就是把R5原来不知道的10和20也配置为静态路由： ip route-static 20.0.0.0 8 30.0.0.1 and ip route-static 10.0.0.0 8 30.0.0.1  将原来的默认路由操作undo掉  最后的R5routing-table：     1. 动态路由RIP   **RIP1实验：**  配置如下拓扑：    分别给两个路由器配置rip协议：    RTA：    这里RTA就学习到了RTB路由表中的192.168.2.0，下一跳为RTB路由器的入接口。  RTB：    这里RTB就学习到了RTA路由表中的192.168.0.0，下一跳为RTA路由器的入接口。  不出所料，也是成功ping通了：    **RIP2实验：**  配置如下拓扑，并按照rip1配置路由器  RTA：    RTB：    注意看我们的RTA，  这里学到的为什么是10.0.0.0/8呢？我设置的掩码明明是24。  RIP-1是有类别路由协议（Classful Routing Protocol），其协议报文中没有携带掩码信息，它只能识别A、B、C类这样的自然网段的路由。换句话说，当输入network 命令后，华三设备会将network后面的ip地址变为其对应的主类地址，实际上宣告的是输入ip的主类地址（拿一个ip对应的自然掩码和该ip相与），也就是自然网段。而我们这里10.0.0.0/24显然是非自然网段路由，所以RIP1只能学到10.0.0.0/8这个非自然网段。  接下来我们配置RIP2解决这个问题：  分别输入：    得到RTA：    RTB没有变化就不展示了。  可以看到RIP2学到的就是10.0.0.0/24，认可了我们的非自然网段。  RIP2 验证实验：  我们在RTA和RTB中分别输入这两个配置  [RTA-Serial0/0/0]rip authentication-mode md5 usual aaaaa  [RTB-Serial0/0/0]rip authentication-mode md5 usual 12345  发现RIP路由消失了，为什么呢？  rip authentication-mode md5这条命令用于启用RIP协议的MD5认证，usual aaaaa 和 usual 12345这些命令设置了RIP协议使用的认证密钥。  当两台路由器使用RIP协议进行通信时，它们会检查传来的路由更新是否具有正确的MD5认证信息。如果认证失败（例如，密钥不匹配），路由器将不接受该路由更新，从而保护网络不受错误的路由信息影响。所以**为了使两台路由器能够成功地进行RIP通信，它们之间的接口必须配置相同的认证密钥。**  可以用  rip authentication-mode simple abcdef  修改验证方式。   1. 动态路由OSPF   实验1：单区域OSPF  配置如下拓扑：    然后在RTA和RTB中配置ospf：  [RTA]router id 1.1.1.1  [RTA]ospf  （ospf命令用来创建并运行OSPF进程）  [RTA-ospf-1]area 0  （area命令用来创建OSPF区域,并进入OSPF区域视图）  [RTA-ospf-1-area-0.0.0.0]network 1.1.1.1 0.0.0.0  （network命令用来指定运行OSPF协议的接口和接口所属的区域）  [RTA-ospf-1-area-0.0.0.0]network 10.0.0.0 0.0.0.255  [RTA-ospf-1-area-0.0.0.0]network 20.0.0.0 0.0.0.255  [RTB]router id 2.2.2.2  [RTB]ospf  [RTB-ospf-1]area 0  [RTB-ospf-1-area-0.0.0.0]network 2.2.2.2 0.0.0.0  [RTB-ospf-1-area-0.0.0.0]network 10.1.0.0 0.0.0.255  [RTB-ospf-1-area-0.0.0.0]network 20.0.0.0 0.0.0.255  这里将上面掩码改为255.255.255.0，可行否？  不行，因为network命令的格式是：network （network-address wildcard-mask），wildcard-mask是掩码的反码。写成255.255.255.0那代表掩码是0.0.0.255，不是一个有效的掩码。  我们可以得到如下路由表：  RTA：    RTB：    可以看到，ospf已经学到了非直连路由。  检验得，这个网络已经连通。  RTA：    **实验2：路由选择**  配置如图拓扑：    首先我们还是配置好路由接口ip：  对于RTA：   [RTA]int s 0/0/0   [RTA-Serial0/0/0]ip add 10.0.0.1 24   [RTA-Serial0/0/0]int s 0/0/1   [RTA-Serial0/0/1]ip add 20.0.0.1 24   [RTA-Serial0/0/1]int LoopBack 0  （进入虚拟接口LoopBack 0的配置模式，通常用于配置管理IP地址）   [RTA-LoopBack0]ip add 1.1.1.1 32  （为LoopBack 0接口配置IP地址1.1.1.1，子网掩码为255.255.255.255，表示这是一个单一的IP地址）   [RTA-LoopBack0]quit   [RTA]ospf   [RTA-ospf-1]area 0   [RTA-ospf-1-area-0.0.0.0]network 1.1.1.1 0.0.0.0 （将LoopBack 0接口的IP地址1.1.1.1加入到OSPF区域0中）   [RTA-ospf-1-area-0.0.0.0]network 10.0.0.0 0.0.0.255   [RTA-ospf-1-area-0.0.0.0]network 20.0.0.0 0.0.0.255  为什么配置loopback？  在OSPF包头中的路由器ID域，是由路由器上的最大IP地址确定的，在建立和维护毗邻关系时，这个ID值起确定唯一路由器的作用。这个IP可以是位于物理接口上，甚至不用管这个接口是否运行OSPF进程。因此，从这点上看，配置一个loopback接口并不是必须的。但如果与该地址相对应的接口down了，那么路由器就不得继续使用这个IP地址作用其路由器ID，而要重新选择一个值最大的IP地址作为路由器ID。路由器的ID发生改变后，必须在其所有链路上对其邻居重新介绍自己。  RTB配置几乎一样，只是Loopback的id改为2.2.2.2。  在RTA中ping2.2.2.2，也是ping通了：    然后我们使用disp ospf brief命令看一下：    可以看到，routerID为10.0.0.1，为什么是这个值呢？  我们先看看Router-ID选举规则：  如果通过Router-ID命令配置了Router-ID，则按照配置结果设置。在没有配置Router-ID的情况下，如果存在配置了IP地址的Loopback接口，则选择Loopback接口地址中最大的地址作为Router-ID；如果没有已配置IP地址的Loopback接口，则从其他接口的IP地址中选择最大的地址作为Router-ID。**（注意：已经选取了一个非Loopback接口地址后又配置了一个Loopback接口地址）**  接口配置顺序会影响Router-ID的选举，因为我们这里第一次配置的物理接口的地址为10.0.0.1，该动作会触发Router-ID的选举，而此刻，设备上也有且仅有该物理地址，所以该地址便会Router-ID所使用，后续即使再配置了环回接口地址也不会使用。同理，如果第一次配置的是其他接口物理接口的地址，或者是环回接口的地址，都会被Router-ID所使用。  使用disp ospf lsdb命令：    显示的是OSPF链路状态数据库（LSDB）的内容。链路状态数据库是每个运行OSPF的路由器上存储的关于网络拓扑结构的数据库。  LinkState ID是链路状态的ID，对于Router类型，这通常是路由器的Router ID。  可以推断出Router ID为10.0.0.1的路由器（RTA）已经学习到了Router ID为10.0.0.2（RTB）的存在，并且两者都在OSPF区域0中。至于RTB的router ID为什么选10.0.0.2，也是因为这个接口ip被率先配置。  使用disp ospf routing命令：    有两个目的网络为2.2.2.2/32的路由，成本为1562，这表明存在到同一个目的地的两条等成本路由（ECMP - Equal-Cost Multi-Path），分别通过10.0.0.2和20.0.0.2两个下一跳路由器。  还有两个目的网络分别为10.0.0.0/24和20.0.0.0/24的路由，这两个路由分别对应于Serial0/0/0和Serial0/0/1接口所在的网络，成本为1562。比如10.0.0.0网段的包就发到接口10.0.0.1。  Ospf路由表和IP路由表有何区别？  OSPF路由表：专门用于存储OSPF协议学习到的路由信息。它包含了目的网络、成本、类型、**下一跳路由器**、区域等详细信息。  IP路由表：是路由器用来决定如何转发IP数据包的通用路由表。它包含了目的网络、子网掩码、**下一跳地址**、接口等信息。  **注意OSPF路由表中多写了advrouter，也就是下一跳路由器的routerID。**  我们可以看到如下路由表：  RTA：    RTB：    为什么会出现两条目的地为对方路由器loopbackID的路由？代表什么含意？  因为这两个路由器可以由不同的两个网络连接，10 和20，由serial0/0/0和serial0/0/1连接。OSPF支持等价成本多路径，这意味着如果有多条到达同一目的地的成本相同的路径，路由器会将流量分散到这些路径上，以实现负载均衡。这种情况下，路由表中会出现多条到达同一目的地的路由，但它们的下一跳地址可能不同（出接口ip不同）。  **更改接口COST：**  [RTA]int s0/0/0  [RTA-Serial0/0/0]ospf cost 150  [RTA-Serial0/0/0]disp ospf routing    为何比上次少了一条路由？少了哪一条？  因为我们修改了serial0/0/0接口的cost，减少为150，所以从这个接口（网络）到RTB的cost比接口serial0/0/1小，直接将cost大的路由（serial0/0/1那条，连接20.0.0.0网络，下一跳是20.0.0.2）删除掉就好了。  我们还可以看到，上面那个10.0.0.0网络的路由cost也成了150.  我们再看看routing-table：    IP路由表中到2.2.2.2为何只有一条路由了？  Nexthop为20.0.0.2的那一条被删掉了，原因刚才也说了。之前有两条是因为cost相同，正好留着平衡负载。现在serial0/0/0这一条cost改低了，那自然就只保留这一条了。  观察RTB的路由表，到RTA有几条路由？为什么？    到RTA有两条路由，毕竟RTB没有修改cost。  **实验3：多区域**  配置此拓扑：    RTA两个接口都属于Area\*\* ，RTB两个接口分属于Area\*\*和Area\*\*， Area是基于(路由器/接口)？  先配置好路由接口ip，这都老朋友了，这里就不展示。  接下来我们配置ospf：  RTA：    RTB：两个接口还不属于一个area，所以要分别配置。    RTC：    此时两台pc是否互通？    可以ping通。  使用disp ospf peer来检验一下：        RTA和RTC为什么一个是Master，一个是Slave？  在OSPF中，选举Master和Slave接口的目的是为了在多接口连接到同一网络的情况下，避免不必要的路由信息交换和可能的路由环路。Master接口负责与邻居路由器建立邻接关系和交换路由信息，而Slave接口则处于被动监听状态，不会主动发送路由信息，但仍然可以接收来自邻居的路由更新。  选举的原则是什么呢？  路由器上的每个接口都可以配置一个优先级。在选举过程中，具有最高优先级设置的接口会成为Master。如果没有明确设置优先级，或者优先级相同，那么具有最高IP地址的接口会成为Master。如果接口优先级和IP地址相同，路由器ID的数值大小也会影响选举，具有较高Router ID的路由器上的接口可能会成为Master。  **我们这里RTA成为master的原因是它先被配置了接口ip和router-id。和上一个实验选取routerid的原理类似。**  检验：  使用disp ospf routing 查看路由器RTA的ospf路由：    可以看到这里显示学到了两个跨区域（inter-area）路由，也就是10.1.0.0网络和30.0.0.0网络的路由，下一跳都是RTB的入接口。Cost是20.0.0.0网络路由的两倍，因为需要让两个路由器转发才能到达这些网络。  我们再看一看RTA的路由表：    RTB路由表：    前面配置中  [RTA-ospf-1-area-0.0.0.0]network 1.1.1.1 0.0.0.0  [RTB-ospf-1-area-0.0.0.0]network 2.2.2.2 0.0.0.0  [RTC-ospf-1-area-0.0.0.1]network 3.3.3.3 0.0.0.0  这三句是否必须要？这里为何要加？  network IP地址 0.0.0.0，该方式精确指定只有该IP地址的接口才能发送OSPF报文，其它没有宣告到的接口将不能发送OSPF报文。  我觉得并不是必须要的，因为我们宣告的ip是Router ID，而不是任何物理接口的IP地址。但是为什么这里加上了呢，可能是通过宣告这个IP地址，可以确保网络中的其他路由器知道这个Router ID是可达的。   1. 以太网交换基础   实验拓扑1：    使用ipconfig 命令查看mac address（physical address）：      交换机LSW1的mac-address table：    此table并不是一开始就有的，而是主机ping后，LSW1 从packets中学到的。  PC1对应哪个端口，TYPE是什么？属于哪个VLAN？  从mac-address table 中可以看到，PC1连接port GE0/0/1，type为dynamic，属于VLAN 1  如果PC1要给PC2发送一个数据包，其转发流程是什么？  先用target IP与自己的子网掩码相与，判断是否属于同一个子网，如果是，那么使用arp广播，询问网络上的所有设备，寻找1.1.1.2的位置。网络上的设备收到ARP请求后，会检查请求中的目的IP地址是否与自己的IP地址匹配。如果匹配，该设备将发送一个ARP响应，提供其MAC地址。一旦PC1收到ARP响应，它将使用获得的MAC地址将数据包封装成frame，并发送到交换机的GE0/0/1。交换机接收到frame后，会查看自己的mac-address table（这里它已经通过arp包学习到了双方的mac address）将frame转发到下一跳GE0/0/2。发到PC2。  查看MAC地址老化时间，是多少？利用mac-address timer 500，将老化时间改为500S  使用命令**disp mac-address aging-time**查看：300seconds    修改老化时间：这里用的命令是**mac-address aging-time**    过几分钟后，在交换机上再执行disp mac-address，还有无结果？要想再看到MAC表，该如何操作？  超过设定的老化时间就没有mac记录了，我们可以再ping一次，就能看到源和目的mac记录了。  **配置static MAC address：**    此时PC1的MAC TYPE为static，说明配置成功  然后删除PC1与交换机的连线，重新建立PC1与交换机GE0/0/3接口的连线：    在PC1上ping PC2，通了  看交换机的mac-address table，可以看到由GE0/0/3的接口。  因为我们的mac是动态学习来的，况且现在我们的pc1是用GE0/0/3端口，所以不受静态mac的影响    实验2：  配置如图拓扑：    在标出的位置进行抓包  我们看到PC1发出了ARP broadcast：询问1.0.0.4在哪里，    它经过了每一个设备，所以每一个接口都捕获到了这个广播，然后LSW3的0/0/3和LSW1的0/0/1，和LSW2的GE0/0/1收到了PC4返回的包。  在交换机的mac失效前再ping一次，我们发现只抓到了来回的包，并没有ARP广播，因为LSW2中已经记录了PC4的mac addresss。  这时候我查看每一个交换机的mac addresss table，发现都记录了同样的两个mac address，就是PC1和PC4的mac。  总结一下ARP的工作原理：  PC1先用target IP与自己的子网掩码相与，判断是否属于同一个子网，如果是，那么使用ARP broadcast，询问网络上的所有设备，交换机会将这个ARP广播frame复制到除接收接口之外的所有其他接口。网络上的设备收到ARP请求后，会检查请求中的目的IP地址是否与自己的IP地址匹配。如果匹配，该设备将发送一个ARP响应，提供其MAC地址。PC1接收到ARP响应后，会将响应中提供的MAC地址与目标IP地址关联起来，并存储在自己的ARP缓存表中。有了这个MAC地址，frames就可以通过交换机的mac address table做出匹配，交换机就知道自己该将它发送到哪个接口。  ARP广播只在本地子网内有效，如果目标IP地址不在本地子网，PC1需要将数据发送到默认网关（路由器），由路由器来处理跨子网的通信  实验3：  配置如下拓扑：记得配网关    我们抓如上所示6个点的包  我们先用PC1 ping PC4，看看每个接口抓到了啥，我们从左到右看：  **LSW2 GE0/0/1：**    首先，它发出的ARP广播在询问1.0.0.3的mac address，这是因为我们PC1的网关配置的是1.0.0.3。我们ping的是2.0.0.3，不在同一个网段，所以PC1试图将packet发给网关，而它又不知道网关的mac地址。  接下来就是找到了网关之后收发ping的packets。  **LSW2 GE0/0/2：**    只有一个收到的ARP广播，之后的传输都没经过这个接口。  **R1 e0/0/0：**    ARP广播也到了这里，这个接口ip就是1.0.0.3，可谓是撞到家门口了。之后就是PC1知道PC4的mac，开始通过交换机转发packet到网关1.0.0.3，最后到PC4，这里就捕获到了这些ping的packets。  **R1 e0/0/1：**    这里我们可以看到，已经有一个request packet 先通过了这里，之后PC4已经收到了PC1发来的request packet，现在PC4要将包发给PC1。发现自己不和PC1一个网段，所以转发到网关，但是自己没有网关的mac address缓存，所以发送ARP广播，搜寻网关2.0.0.1的mac。找到了2.0.0.1的mac后就PC4就可以发送reply packet。  **LSW3 GE0/0/2：**    这是通往PC3的接口，也是收到了发往2.0.0.3的request（为什么？如果目标MAC地址是已知的，并且与某个接口相关联，交换机会将帧转发到该接口。如果目标MAC地址未知，交换机会将帧广播到所有接口，除了接收帧的接口。现在就是未知的情况），之后PC4开始ARP广播后当然也是广播到这里来了，可惜2.0.0.1不在这里。之后的packet也不在这里通过。  **LSW3 GE0/0/3：**    收到了发往2.0.0.3的request，同时这里也是之后PC4广播的第一个ip。  之后马上再ping一次PC4，发现这次六个接口都没有收到ARP广播了，因为PC1和PC4中已经分别记录了各自网关的mac，直接转发到交换机即可。  我们这里看到了不同网段下两个PC是如何利用ARP的，之前是广播搜寻同网段的PC的mac，现在是广播搜寻网关的mac。值得注意的是接收方在相应的时候也要进行ARP广播来寻找网关的mac。但是一个ARP广播的范围不会超过路由器，也就是不会超出本网段。  **同一交换机不同网段：**    两个pc在不同网段，互为网关。  可以ping通  原因如下：  交换机是两层交换，用MAC通信，不识别三层的IP。若不在同一网段，则看有没有配置网关  无，则直接丢弃该PING请求，提示目标主机不可达（网络层功能）；  有，就交给链路层处理，链路层先看自己ARP缓存中有无网关的MAC（不涉及IP）  有，就直接封装成帧，单播发送给网关(目的MAC为网关的MAC) ；  无，则发送ARP广播找网关(目的MAC为FF:FF:FF:FF:FF:FF)，交换机先收到该广播，先看自己MAC表中有无网关的MAC，有的话直接告诉主机，没有就转发广播，直到找到网关的mac，然后主机将frame单播给网关。  **这里我们并非通过网络层将数据包发给网关**   1. VLAN基础   实验拓扑1：  配置如下拓扑：    使得1和3可以互通，2和4可以互通，但两组不互通。我们可以使用access类型接口将这个网络划分为VLAN1和VLAN2。  通过disp vlan命令可以查看vlan配置：    目前24个接口都属于VLAN1，我们要添加一个VLAN2.  但是我们ensp中的交换机设备是华为的，默认接口类型都是hybrid：    我们可以先更改接口类型为access，再进入VLAN2 加入我们的接口1和3：    这样我们就配置好了vlan2：    此时PC1和PC3能ping通，PC2和PC4能ping通，但是这两组相互之间不能ping通  实验拓扑2：    此时交换机的接口可以是什么类型？  连到PC的接口可以是access类型，而由于VLAN1和VLAN2的包都要在交换机之间传输，所以交换机之间的两个接口可以设置为trunk类型。  接下来我们配置交换机，将PC1和PC3所在端口设置为VLAN 2：      测试PCA与PCC通否？（不通，两个交换机的GE0/0/3都属于VLAN 1，不允许VLAN 2的frame从这里通过）  测试PCA与PCB通否？（不通，不在同一VLAN）  测试PCB与PCD通否？ （通，两个交换机的GE0/0/3都属于VLAN 1，只允许VLAN 1的frame从这里通过）  我们可以将两个交换机的GE0/0/3都配置成trunk类型，设置允许vlan 2的frame通过：      我们查看一下端口配置：      再看看vlan配置:    UT (Untagged)：表示端口连接的设备不需要VLAN标记。  TG (Tagged)：表示端口携带VLAN标记信息。  SWA收到PC1的frame后，从trunk口GE0/0/3转发，发现其没有tag，加上此端口的pvid 2作为vlan tag后转发到SWB GE0/0/3，这是一个trunk口，发现此frame的tag在trunk口允许通过的范围内，于是去掉tag，接收。   1. VLAN通信（使用三层交换机）   有三种方式实现VLAN之间的通信：利用多物理端口，利用三层交换机，利用单臂路由。我们这里只实验利用三层交换机的方法：  三层交换机实际上就是实现了物理层、链路层和网络层的交换机，拥有简单的路由器的功能。  实验拓扑如下：    我们先添加VLAN2和VLAN3，然后为三个VLAN虚拟路由接口分别配置ip地址，操作与路由器一样：    然后我们将三个物理接口的类型改为access，并将其分别加入vlan123：    查看一下此时的vlan配置：      实验的时候不知道为什么，PC1断开连接了，所以这里看到GE0/0/1断开，也没有10.1.1.0网段的路由信息，不过无伤大雅。  用PC2 ping PC3，通了。    三层交换机在每个VLAN中维护一个独立的ARP表和MAC地址表，三层交换机根据数据包的目的IP地址，使用路由表来决定下一跳，这里就是利用的网络层协议来转发包。而我们这里的交换机接口都是access，不涉及vlan tag；如果packet有tag的话，三层交换机会根据目的VLAN的配置，决定是否需要为数据修改VLAN标签。总之根据路由表就可以强制转发。 | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   所有实验期间遇到的问题都在上面提出并且给出了答案。  本实验涵盖了路由器中静态路由动态路由，交换机中的配置和vlan。很多问题不能只学理论知识，也要自己动手配置一下，才能发现一些细小的知识点。  可谓是一个庞大的实验报告，同时也用作以后复习使用 | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |